

Exceptional service in the national interest



OS/Runtime and Execution Time Productivity

Ron Brightwell, Technical Manager
Scalable System Software Department



Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Application Developers Are Evil

- Ron: I have a lightweight operating system that will significantly improve performance and scalability
- App developers:
 - Changing our makefiles is really hard
 - We really don't want to use a cross compiler
 - We don't want to make a platform- or hardware-specific optimizations
 - Any change has to improve performance on all machines
 - Performance portability is critical
 - It's too much work to re-qualify the code

Five Years Later...

- App developers:
 - There's a "new thing" we really want to use
 - It's a custom piece of hardware
 - That requires a cross compiler
 - And its own programming language
 - That we had to change our makefiles to use
 - That isn't portable
 - And we had to restructure our application for it
 - But it really increases performance
- Ron: What?!
 - (OK, maybe the potential performance increase is not identical)

Hardware Designers are Evil

- HW designers:
 - Here's a great new piece of hardware
 - Here's the list of 22 errata for it
 - You system software folks will have to deal with some of those
 - And we had to take out that <memory bus locking> feature the previous version of your OS relied upon
- Ron:
 - Sigh.
 - Great.
 - Thanks.

Some Years Later...

- HW designers:
 - We have more transistors than we know what do with
 - Co-design sounds like a great idea
 - We're going to ask the application developers what hardware features they think they might need
- Ron: What?!

What OS/Runtime Design Choices Impact Productivity?



Tension in Dealing with Transparency

- Both definitions of transparency
 - Magic happens
 - Visibility into lower layers
- Abstract the hardware
 - Focus on interface rather than mechanism
 - But provide direct access when desired
- Provide a portable programming interface for system services
 - But maintain performance and scalability
- Expose locality without exposing locality
 - Caches were supposed to be invisible

Lightweight OS Approach

- Provide deterministic performance
- OS resource usage is fixed
- Resource management is explicit
 - How much memory is available?
 - Application has more control over its resources
- Eliminate unused functionality
 - Demand paging
 - Page pinning
- Implement desired OS policy rather than enable fixing wrong OS policy
 - Page pinning
 - First touch
- Smaller code base increases reliability
- Compatibility with glibc and Linux toolchain is important

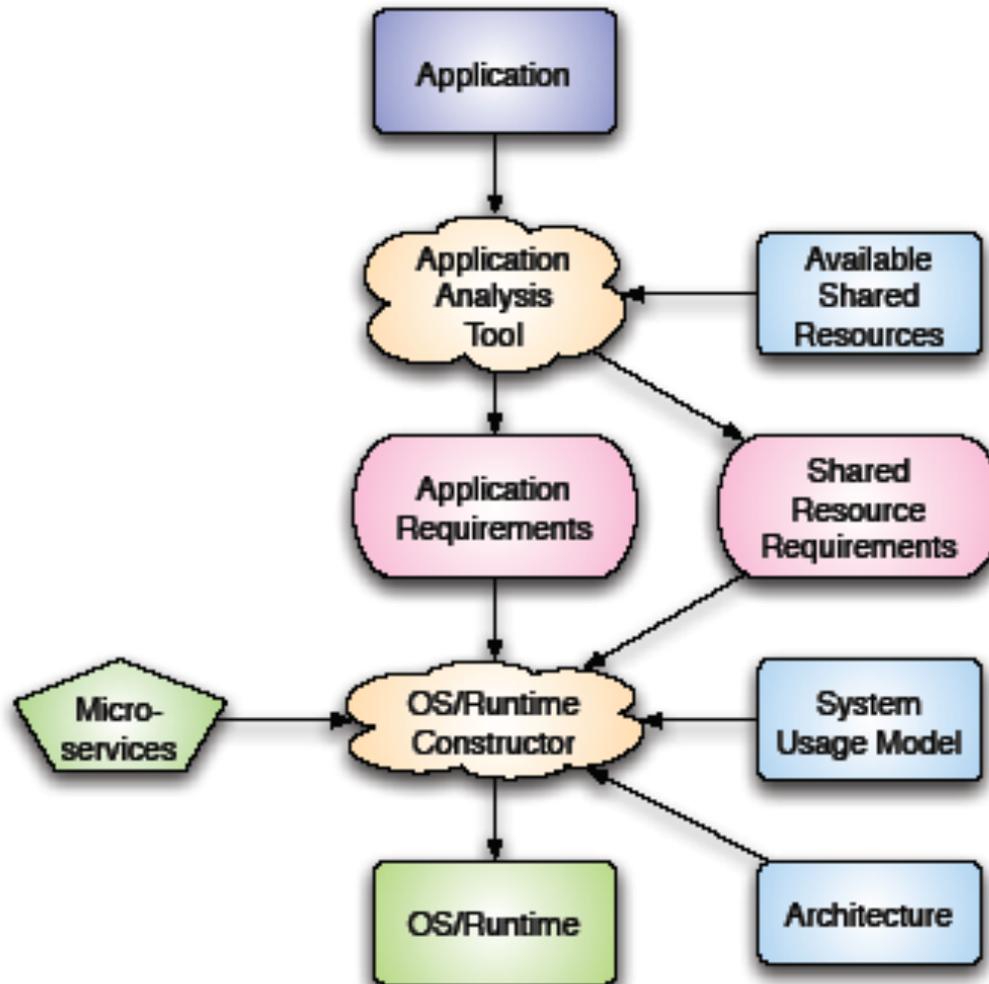
Scaling Issues

- Sandia philosophy has been to “fail fast”
 - Identify scalability issues early by initially setting resources low
 - Avoid providing band aids
 - Counter to what most vendors want to do
- Enable scaling from laptop to extreme-scale
 - Initial MPP users focused on large systems
 - Ideally provide the same software environment
 - But focus should be on scaling down rather than scaling up
 - Consider scaling aspect of system calls and services
 - Exploring virtualization as a potential mechanism
- Performance portability
 - Across all systems or across extreme-scale systems?
 - Should be a secondary goal in the face of fundamental challenges

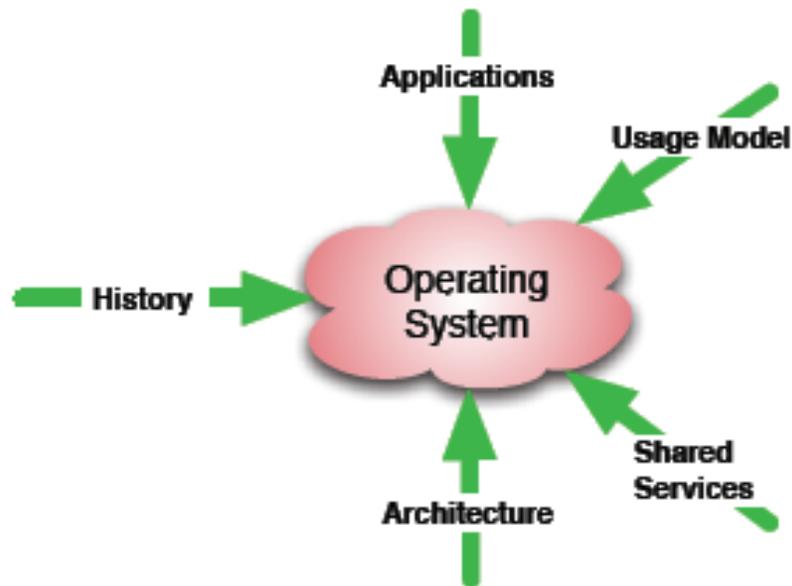
Runtime Systems

- Approach has largely been static
 - Give resources to application and get out of the way
 - Taking on the complexity of dynamic resource management
 - Introspection and adaptivity
 - Significantly more complexity to manage
 - Trying to do what humans have only been mildly successful in doing
 - Need ways for applications and compilers to constrain and guide adaptivity options and mechanisms
 - Power/energy management is a new challenge
 - Application no longer has complete control over resources
 - Runtimes are expected to be performance portable too
-
- 

Building Custom Operating/Runtime Systems



Lightweight Kernel Influences

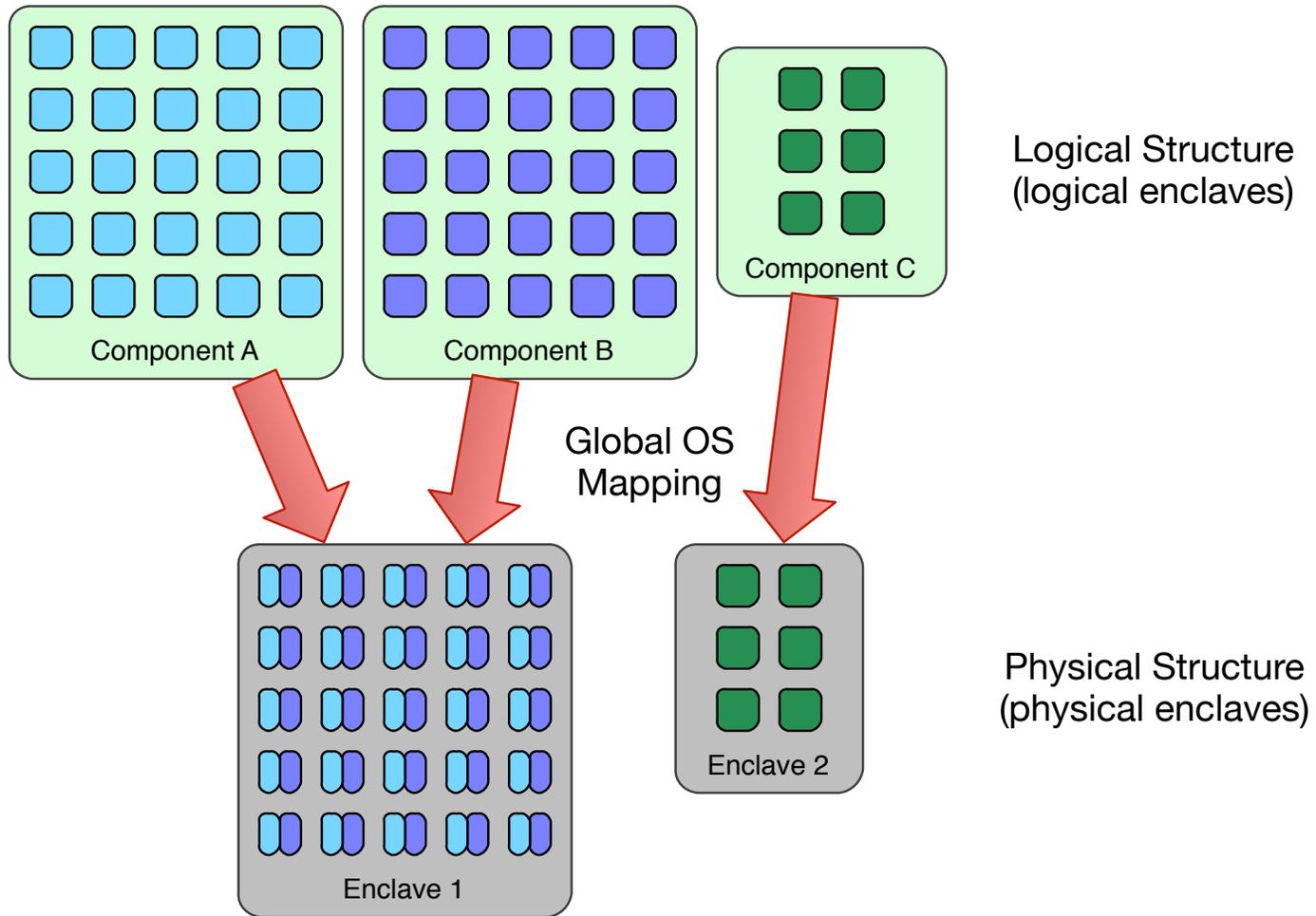


- Lightweight OS
 - Small collection of apps
 - Single programming model
 - Single architecture
 - Single usage model
 - Small set of shared services
 - No history
- Puma/Cougar
 - MPI
 - Distributed memory
 - Space-shared
 - Parallel file system
 - Batch scheduler

Application Composition Will Be Increasingly Important at Extreme-Scale

- More complex workflows are driving need for advanced OS services and capability
 - Exascale applications will continue to evolve beyond a space-shared batch scheduled approach
- HPC application developers are employing ad-hoc solutions
 - Interfaces and tools like mmap, ptrace, python for coupling codes and sharing data
- Tools stress OS functionality because of these legacy APIs and services
- More attention needed on how multiple applications are composed
- Several use cases
 - Ensemble calculations for uncertainty quantification
 - Multi-{material, physics, scale} simulations
 - In-situ analysis
 - Graph analytics
 - Performance and correctness tools
- Requirements are driven by applications
 - Not necessarily by parallel programming model
 - Somewhat insulated from hardware advancements

Composition in Hobbes

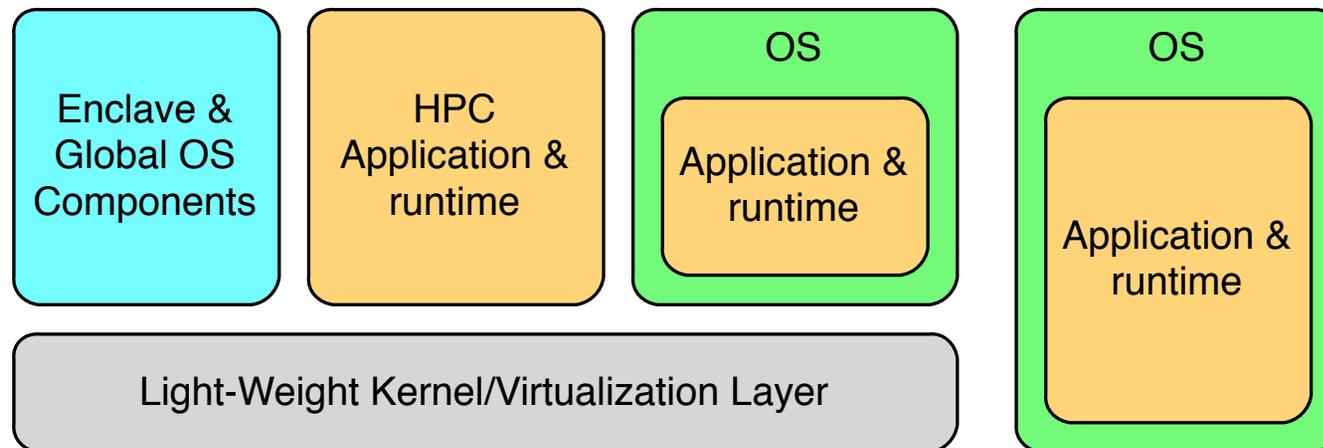


Lightweight Virtualization is a Key Differentiator

- Kitten lightweight kernel
 - Small, highly reliable code base
 - Focused on scalable HPC applications
 - Low noise
 - Small memory footprint
 - Application-based resource management
 - Being enhanced to support dynamic adaptive runtime systems
 - Support for Runtime Interface to OS (RIOS) as part XPRESS project
- Palacios lightweight hypervisor
 - OS-independent virtual machine monitor
 - Can be combined with Kitten or Linux
 - Full system virtualization
 - Guest OS does not need to be modified
 - Supports running multiple guests concurrently
 - Passthrough resource partitioning
 - Extensive configurability
 - Low noise

Hobbes Node Virtualization Layer

- Virtualization layer provides light-weight OS functionality
- Trust is limited to the light-weight OS/virtualization layer
- Enclave and Global OS policies implemented directly on the virtualization layer



Power/Energy Thoughts

- Need three orders of magnitude reduction
 - Largest impact must come from hardware
- Double digit percentage improvement from software may be best case
 - Will be harder to maintain given hardware improvements
- Most explicit data movement has been addressed
 - Not much OS/R can do about implicit data movement like register spill
- Applications becoming more dynamic
 - But bulk synchronous simplifies resource management
- Other power-aware software environments aren't sophisticated
 - Turn devices off when not being used