Hobbes: Composition and Virtualization as the Foundations of an Extreme-Scale OS/R

Ron Brightwell, Ron Oldfield
Sandia National Laboratories

Arthur B. Maccabe, David E. Bernholdt
Oak Ridge National Laboratory

Workshop on Runtime and Operating Systems for Supercomputers
June 10, 2013
Eugene, OR
Summarize the OS/R-specific challenges

Describe a model to integrate DOE-sponsored research with vendor products and support

Assess the requirements of and impact on facilities, production support, tools, programming models, and hardware architecture

Identify promising methods and novel approaches

Write a report that can be referenced by FOA
Council Members

- Pete Beckman, ANL (co-chair)
- Ron Brightwell, SNL (co-chair)
- Bronis de Supinski, LLNL
- Maya Ghokale, LLNL
- Steven Hofmeyr, LBNL
- Sriram Krishnamoorthy, PNNL
- Mike Lang, LANL
- Barney Maccabe, ORNL
- John Shalf, LBNL
- Marc Snir, ANL
Council Meetings

- March 21-22, 2012 – Washington, DC
- April 19, 2012 – Portland, OR (@ Exascale Planning Workshop)
- June 11-12, 2012 – Washington, DC
- July 20-21, 2012 – Washington, DC (Vendor meeting)
- August 21, 2012 – VTC
- September 12-13, 2012 – Washington, DC & VTC
- October 3-4, 2012 – Washington, DC Workshop
- November 14, 2012 – Salt Lake City, Supercomputing 2012
Key Observations for ExaOSR

- Massive Parallelism (exponential growth)
  - Dynamic parallelism and decomposition
  - Advanced run-time systems to manage tasks, dependencies, and messaging linked with scheduler
  - (with dynamic RTS, power and fault mgmt: “OS Noise” not an issue)
- Power as a managed system resource
  - Adjusting arithmetic precision, fault probability, directing power within global view at several levels
- Fault tolerance actively managed in software at many levels
  - Fault management with nodes and at global view
- Architecture organization (significant OS/R changes):
  - Heterogeneous cores, variable precision, specialized functional units
  - Deep memory hierarchies: 3D RAM, NVRAM on node
    - New models for deep memory hierarchy
    - Multi-level Parallelism within the node to hide latency
    - Memory logic
Other Challenges: Business/Social/Total Cost

- Preserving code base
- Vendor business models
- Sustainability/portability
- “Scale Down” important: from the extreme scale to the broader HPC marketplace
- Must address broad range of scientific domains
- DOE does not want an unsupported OS/R
Application OS/R Requirements: Feedback

- Support for:
  - I/O
  - Resilience and system health
  - Dynamic libraries
  - Debugging at scale and ease of use
  - In situ analytics and real-time visualization
  - Threads: creation, management, synchronization

- Desire to automate or be agnostic of power/energy and resilience

- Support new features (eg., non-blocking collectives, neighborhood collectives, ..)
Tool OS/R Requirements Overlap Those of Applications

- Bulk launch for scalability; mapping & affinity matter
- Low overhead way to cross protection domains
- Quality of service concerns for shared resources
- Can have extensive I/O requirements
  - Support for in-situ analysis is critical
- Need OS/R support to handle heterogeneity & scale
  - Synchronization for monitoring
- Need well defined APIs for information about key exascale challenges
  - Power and resilience
  - Asynchrony (API needs may be distinct)
Tool OS/R Requirements Extend Those of Applications

- Must launch with access to application processes
- Low overhead timers, counters & notifications
- Monitoring, access to protected resources
- Attribution mechanisms
  - Aggregation and differentiation
  - Process, resource and source code (including call stack) correspondence
  - Need HW support for shared activities?
- Measurement conversions?
- Multicast/reduction network (shared with OS/R)
- Less clear where tool ends and OS/R begins
System View

External Monitoring & Control
- Operator console
- Event logging Database
- Workflow manager
- Batch scheduler

 admonition {hit}

 Application Enclave

• Initial resource allocation
• Dynamic configuration change
• Monitoring & event logging

 Global Information Bus

 Service Enclave

• Discovery, Configuration
• Monitoring events

 System-Global OS

• Monitoring and control
• Resource management

 Configuration, power, resilience

 Discovery, Configuration

 Hardware Abstraction Layer

• Bring-up
• Monitoring
• Diagnosis

 Hardware & Firmware

• External Services
  - WAN Network
  - Tape Storage

 2013 Workshop on Extreme-Scale Parallel Architectures and Systems
Parallel components
- time or space partitioning
Programming model
- Specific runtime system
Power
- Resilience
- Performance Data

External Interfaces

Tools

Application Component

Enclave Component Runtime

Library

Library Runtime

Enclave Common Runtime

Enclave OS
NODE-LOCAL VIEW

- **Application / Library Code**
- **Library / Language / Model Specific Services**
  - **Common Runtime Services**
    - Thread/task and messaging services
    - Memory, power, and fault services
    - Performance data collection
    - Local instance of Enclave RT
  - **Kernel**
    - Core Kernel Services
    - Local instance of Enclave OS
    - Proxy for SGOS
Exascale OS/R Report

Exascale Operating and Runtime Systems Program

- $6M of funding for OS/R research at DOE labs
- Focus areas
  - Power management
    - Adaptive power management to meet 20 MW goal
  - Support for dynamic programming environments
    - Manage billions of threads
  - Programmability and tuning support
    - Dynamic adaptation and debugging
  - Resilience
    - Predict, detect, contain, and recover from faults
  - Heterogeneity
    - Hierarchical process and memory systems
  - Memory management
    - Use of new memory technologies
  - Global optimization
    - Manage resources with a system-wide view
Exascale OS/R Focus is on Hardware

- Reliability/Resilience
- Power/Energy
- Heterogeneity
- Memory hierarchy
- Cores, cores, and more cores

- Risk
  - Hardware advancements and investments can provide orders of magnitude improvement
  - OS/R advancements can provide double-digit percentage improvement
OS 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
What About Applications?

- Focus is on parallel (multi-core) programming model
  - Advanced runtime systems
  - Node-level resource allocation and management
  - Managing locality
  - Extracting parallelism
  - Introspective, adaptive capabilities
    - This is really hard (Sanjay’s keynote 😊)

- Risk
  - Incremental approach (OpenMP) wins
    - Advanced runtime capabilities are overkill
  - No clear on-node parallel programming model winner
    - Difficult to optimize OS/R
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
Hobbes* Project

- Hardware challenges (power, resilience) are systemic
  - OS alone cannot solve these challenges
  - OS needs to provide infrastructure for exploring solutions
- Significant existing investment in runtime system research
- Lightweight virtualization is a key technology
  - Efficient sharing and isolation of hardware resources
  - Manage expectations of overhead versus flexibility
  - Leverage Kitten/Palacios lightweight virtualization environment
- Create APIs and mechanisms for application composition
- Multi-institutional team
  - Sandia, Lawrence Berkeley, Los Alamos, and Oak Ridge national labs

*The cat, not the philosopher
Policies to manage the VMs on a single node.

Additional mechanisms needed to manage multiple VMs. Run in kernel mode to take advantage of VM support in modern processors (Palacios).

VMs can share the resources via time sharing or space sharing. This is managed by the SGOS.

Basic mechanisms needed to virtualize hardware resources like address spaces (Kitten).
Policies to manage the VMs on a single node.

Sharing among applications is managed by the NOS and Unified RT. The SGOS has a minimal role.

Additional mechanisms needed to manage multiple VMs. Run in kernel mode to take advantage of VM support in modern processors (Palacios).

Basic mechanisms needed to virtualize hardware resources like address spaces (Kitten).
Hobbes Node Architecture
HPC Application and Runtime

- Policies to manage the VMs on a single node.
- On Node Management:
  - EOS
  - SGOS
- VM Management Module
- HAL (Hardware Virtualization)
- Additional mechanisms needed to manage multiple VMs. Run in kernel mode to take advantage of VM support in modern processors (Palacios).
- Basic mechanisms needed to virtualize hardware resources like address spaces (Kitten).
- No sharing of node resources. The on-node GOS is minimal and the VM module might be gone.

2013 Workshop on Runtime and Operating Systems for Supercomputers
More questions?