SAND2010-5039P

Programming and Run-Time Models for Heavily Threaded Systems

Run-Time Systems Panel

Ron Brightwell
Sandia National Laboratories
Scalable System Software
rbbrigh@sandia.gov

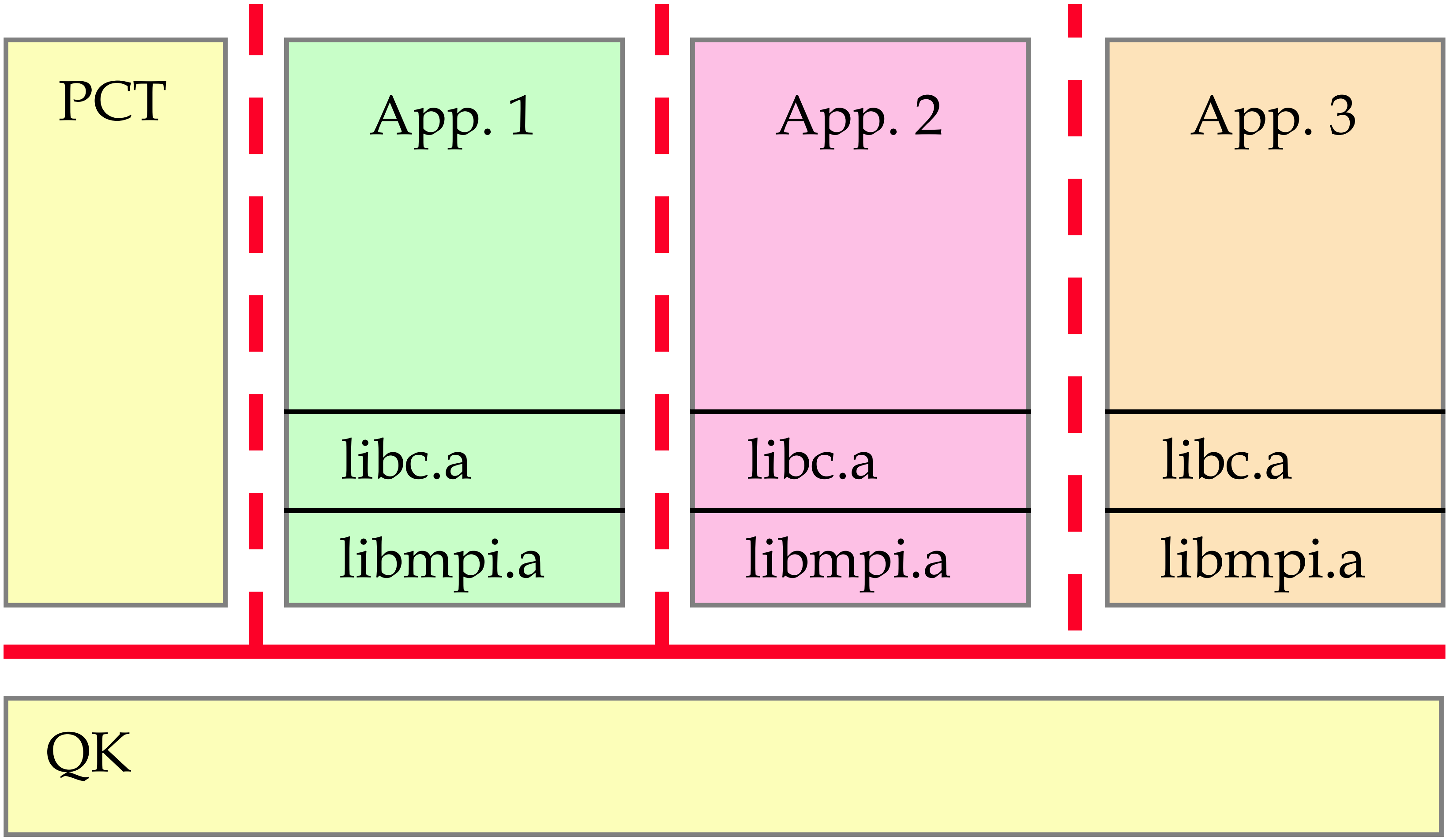
July 27, 2010

Sandia is a Multiprogram Laboratory Operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy Under Contract DE-ACO4-94AL85000.

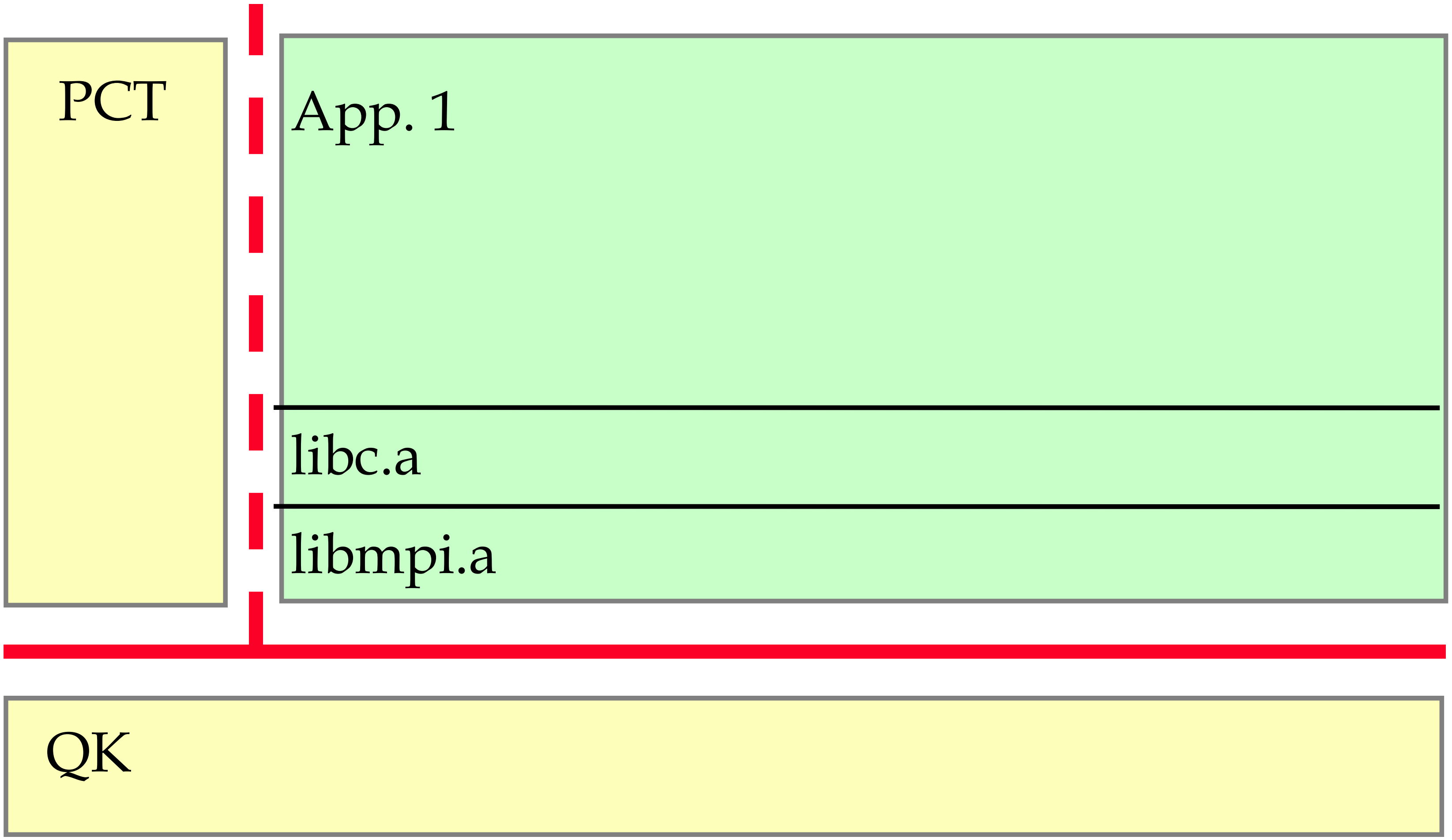
Sandia Lightweight Kernel (LWK) Approach

- Separate policy decision from policy enforcement
- Move resource management as close to application as possible
- Protect applications from each other
- Let user processes (libraries) manage resources
- Get out of the way

LWK - General Structure



LWK - Typical Usage



Quintessential Kernel (QK)

- Policy enforcer
- Initializes hardware
- Handles interrupts and exceptions
- Maintains hardware virtual addressing
- No virtual memory support
- Static size
- Non-blocking
- Small number of well-defined entry points

Process Control Thread (PCT)

- Runs in user space
- More privileged than user applications
- Policy maker
  - Process loading
  - Process scheduling
  - Virtual address space management
  - Fault handling
  - Signals

Pros and Cons of LWK Approach (From a Run-Time Perspective)

- Cons
  - Node-level resource allocation and management is static
    - Memory allocation happens at application load time
    - Bad for shared memory on NUMA systems
  - Run-time components only communicate on set-up and tear-down
- Pros
  - Supports an application-specific run-time
    - Never happened in practice
    - OSFA worked for MPI applications
  - User-level networking
    - Run-time system can use same network interface as applications
    - No need for communication stack inside the OS
  - Memory management and scheduling are greatly simplified

2. What support does the run-time need from hardware to do a better job at exascale?
How is that different from current situation?

National Nuclear Security Administration

Sandia National Laboratories

Hardware Support for Run-Time Systems

- Network hardware support for thread activation
  - Run-time system components must communicate across nodes
  - Message reception in current networks occurs by recognizing change in memory
    - Leads to polling
  - Need hardware mechanism to block/unblock threads on network events
  - Active message model only makes sense with hardware support
    - Waiting until there's nothing to do to notice incoming messages is bad
- More advanced network functions (eureka, dynamic hierarchy)
- More sophisticated mode switch / protection hardware
- Hardware performance information
  - Dynamic resource management decisions will need performance info
  - Current performance counters only capture a subset of what is needed
- Thread scheduling
  - Hardware support for efficient scheduling
  - Must be flexible (programmable?)
  - Should allow for operating on groups of threads

3. What protection rings should be available to an exascale run-time? Should those be organized differently than in current systems? What is the role that virtualization will play at exascale? What layers should be virtualized? What is a Parallel Virtual Machine in the exascale era?

Protection Rings

- Current scalable HPC applications don’t make system calls
  - Allows the ratio of full-featured service nodes to lightweight nodes to be small
  - All “real” system calls on Sandia LWK were serialized through one process
- Current run-time systems don’t make system calls either
  - Only at set-up and tear-down
- Probably only need a small subset of cores with ring 0 capability
  - System calls will turn into run-time thread activation response
- May need to have more sophisticated network protection mechanism
  - Would like to have run-time system threads invoked on message arrival

6. Current HPC systems are built atop a large number of individual OS images, with tight coupling at the application level but very limited coupling at the OS level. Will such an approach scale to exascale?

Yes

- This is part of what defines the OS and differentiates run-time system
  - The lowest level of local hardware management
- Need hierarchical structure to allow for scalability
- Exascale will require tighter coupling between some components
  - Run-time system components
  - RAS system and run-time system
  - Application and run-time system
- Need to provide information while minimizing dependencies
  - Use all information but limit required information
  - OS shouldn't require non-local information