An Example of a Lightweight Kernel

Ron Brightwell
Sandia National Labs
Scalable Computing Systems Department
rbbrigh@sandia.gov
Goals of Puma

• Targets high performance scientific and engineering applications on tightly coupled distributed memory architectures
• Scalable to tens of thousands of processors
• Fast message passing and execution
• Small memory footprint
• Persistent (fault tolerant)
Approach

• Separate policy decision from policy enforcement
• Move resource management as close to application as possible
• Protect applications from each other
• Get out of the way
General Structure

- **PCT**
  - libc.a
  - libmpi.a

- **App. 1**
  - libc.a

- **App. 2**
  - libc.a
  - libnx.a

- **App. 3**
  - libc.a
  - libvertex.a

Q-Kernel: message passing, memory protection
The Quintessential Kernel (Qk)

- Policy enforcer
- Initializes hardware
- Handles interrupts and exceptions
- Maintain hardware virtual addressing
- No virtual memory paging
- Static size
- Small size
- Non-blocking
- Few, well defined entry points
The Process Control Thread

- Runs in user space
- More privileges than user applications
- Policy maker
  - Process loading
  - Process scheduling
  - Virtual address space management
  - Name server
  - Fault handling
PCT (cont’d)

• Customizable
  – Singletasking or multitasking
  – Round robin or priority scheduling
  – High performance, or debugging and profiling version

• Changes behavior of OS without changing the kernel
Levels of Trust

- App.
- PCT
- QK
- Node Hardware
- Network
CPU Modes

• Chosen at job load time
• Heater mode
  – LWK and app on system processor
• Message co-processor mode
  – LWK on system processor
  – App on second processor
• Compute co-processor mode
  – LWK and app on system processor
  – App co-routines on on second processor
• Virtual node mode
  – LWK and app on system processor
  – Second app process on second processor
Portals Message Passing

- Basic building blocks for any high-level message passing system
- All structures are in user space
- A portal consists of one or more of the following:
  - A memory descriptor
  - A matching list
- Avoids costly memory copies
- Avoids costly context switches to user mode (up call)
Key Ideas

• Protection
• Kernel is small
  – Very reliable
• Kernel has static size
  – No structures depend on how many processes are running
  – All message passing structures are in user space
• Resource management pushed out of the kernel to the process and the runtime system
• Services pushed out of the kernel to the PCT and the runtime system