ROS

Barret Rhoden, Kevin Klues, David Zhu, Paul Pearce, Andrew Waterman, Eric Brewer

Par Lab, CS Division, University of California at Berkeley

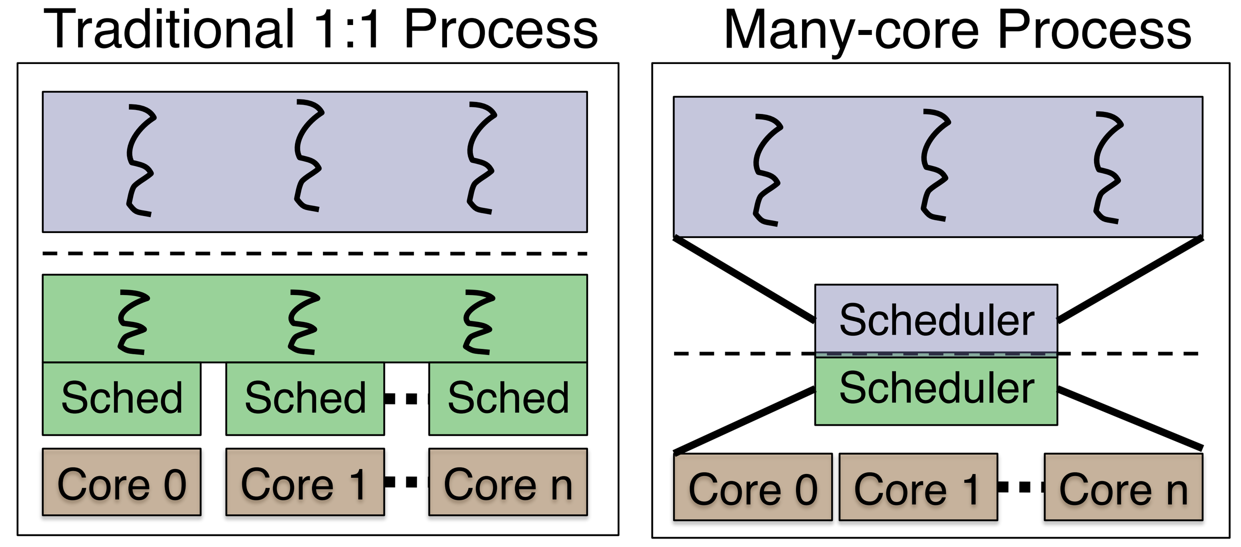
Kevin Klues Paul Pearce Barret Rhoden David Zhu A. Waterman

A Scalable Operating System For Parallel Applications On Many-core Architectures

Design

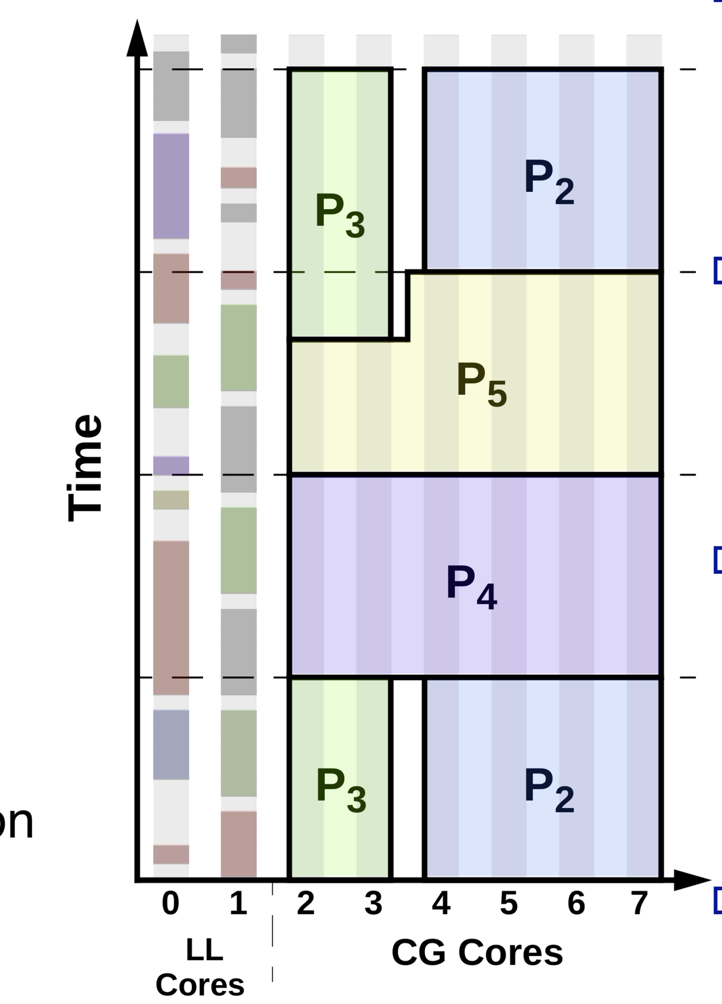
- GOAL: Explicitly support parallel applications while improving kernel scalability
- Many-core Process (MCP)
  - No longer a single thread in a virtual processor
  - Multiple cores 'owned' by a single process
  - All cores gang scheduled
  - Information exposed up, requests sent down
- Asymmetric Use of Cores
  - Low-Latency vs. Coarse-Grained Cores
  - Asynchronous Remote Calls (ARCs)
  - Kernel control path on a limited number of cores
- Resource Provisioning
  - Provisions setup before allocation takes place
  - Increases isolation between processes
  - Enables predictable application performance
  - Allows the system to utilize unused resources

Many-Core Process



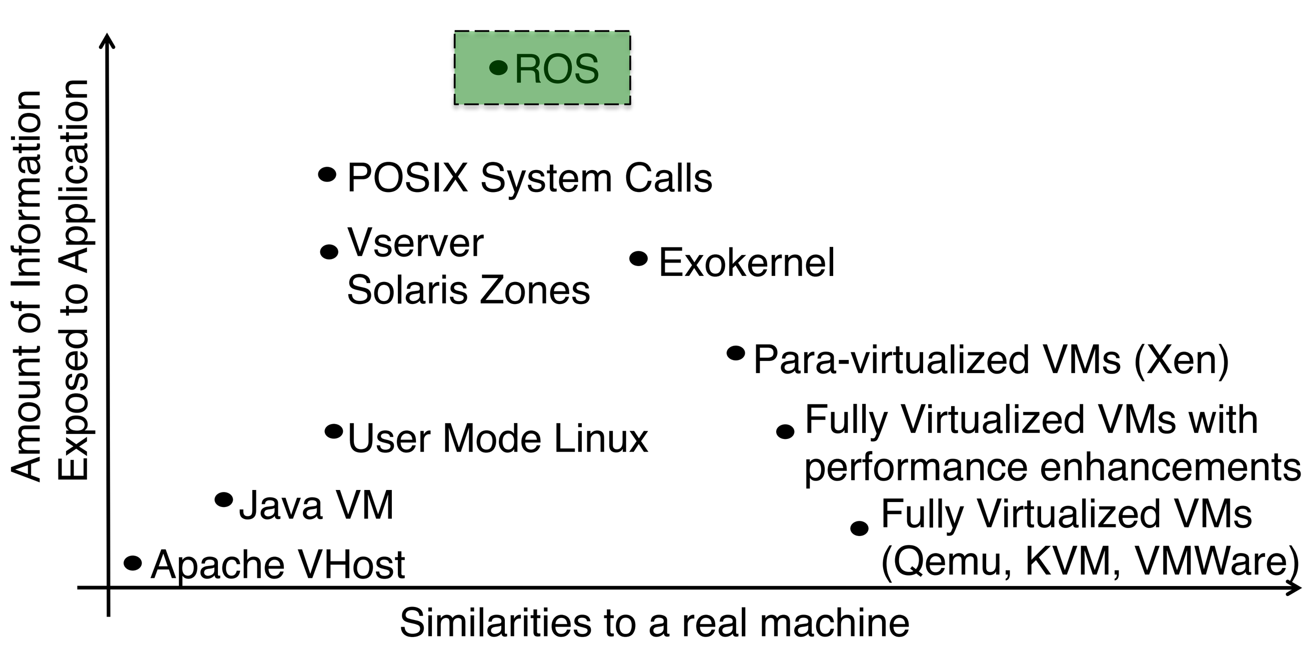
- More scalable than traditional process models
  - No mapping of user-level threads to kernel threads (the kernel is completely event-based)
  - No per-core run queues
- Provides richer set of resource guarantees to processes
  - Expose more information about system resource utilization
  - MCPs make explicit requests for those resources
- All cores granted to an MCP are gang scheduled
- No unexpected interrupts or blocking system calls (ARCs)

Asymmetric Use of Cores

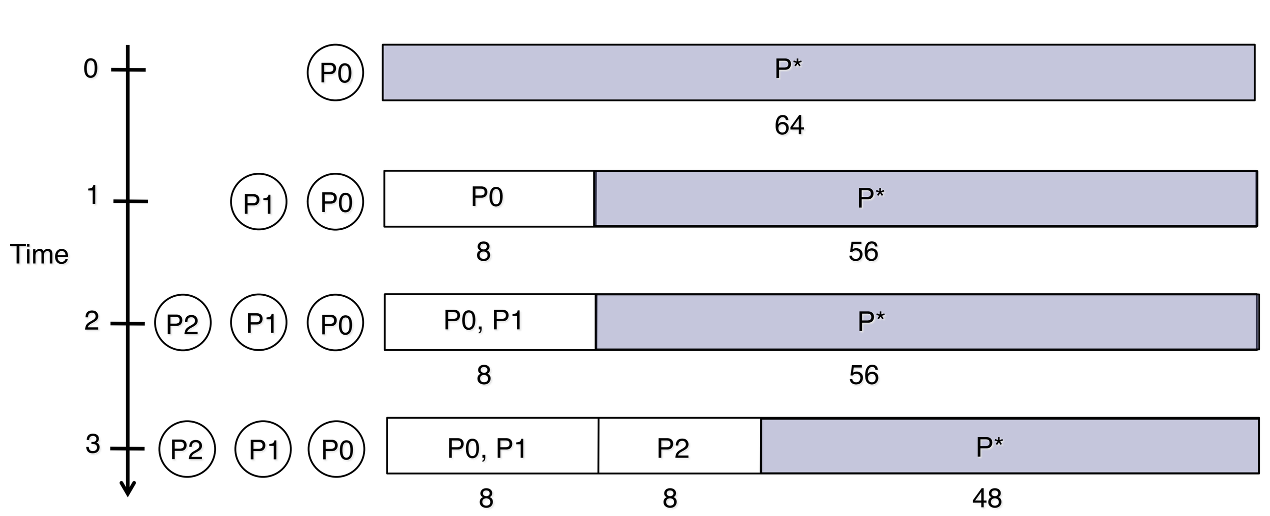


- Coarse-Grained Cores
  - Used for parallel computations requiring predictable performance
  - Time-sliced at coarse-granularity
  - Granted to apps running as MCPs
- Low-Latency Cores
  - Handle time-critical events out of band
  - Always runnable, not gang-scheduled
  - Time-sliced at fine-granularity
  - Examples: UI events, TCP ACKs, etc.
- Asynchronous Remote Calls (ARCs)
  - System calls serviced asynchronously on Low Latency Cores
  - Increase per core cache locality
  - Decrease cross core lock contention
  - Limit kernel interference with apps
- Small set of cores control the system
  - Manages what processes run where
  - No need for per core run queues

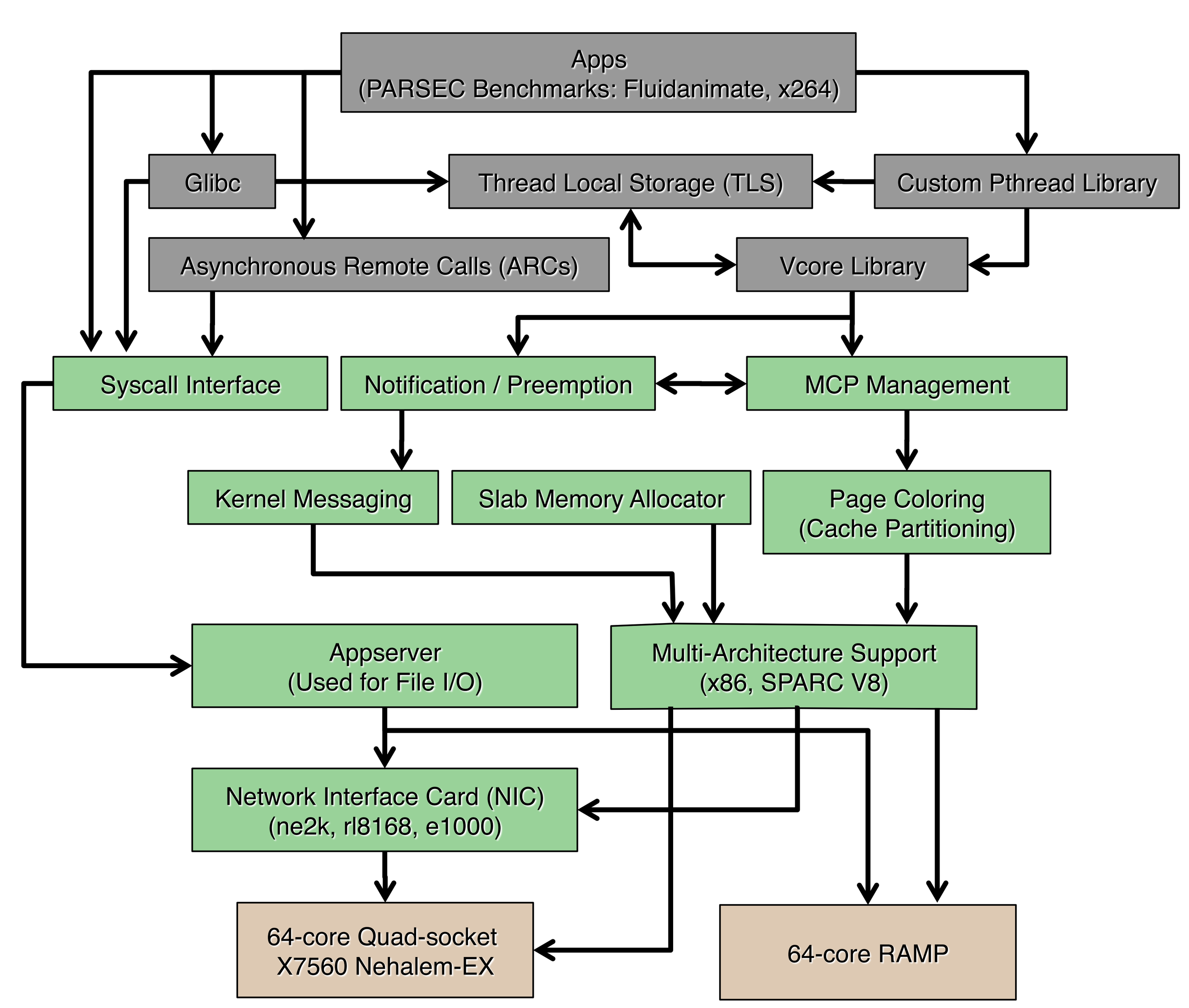
Resource Provisioning



- Resources **provisioned** to MCPs based on future needs
- Resources **allocated** to MCPs based on immediate needs
- Processes scheduled based on meeting resource guarantees (QoS)
- Resource guarantees enforced either in hardware or in software



Current Implementation



Preliminary Results