Microkernels Meet Recursive Virtual Machines

Bryan Ford, Mike Hibler, et. al

Presented by Neal H. Walfield

Jan. 17, 2007

Research Goals

- Identify fundamental properties for efficient recursive virtualization

Goals

- Modularity
- Flexibility
- Extensibility
- Efficiency

Microkernels and VMs

- Horizontal vs vertical decomposition

VMs

- Virtualize existing hardware interface
- Minimize divergence from underlying architecture
- Support naïve OSes
- Bad at stacking

RVMs

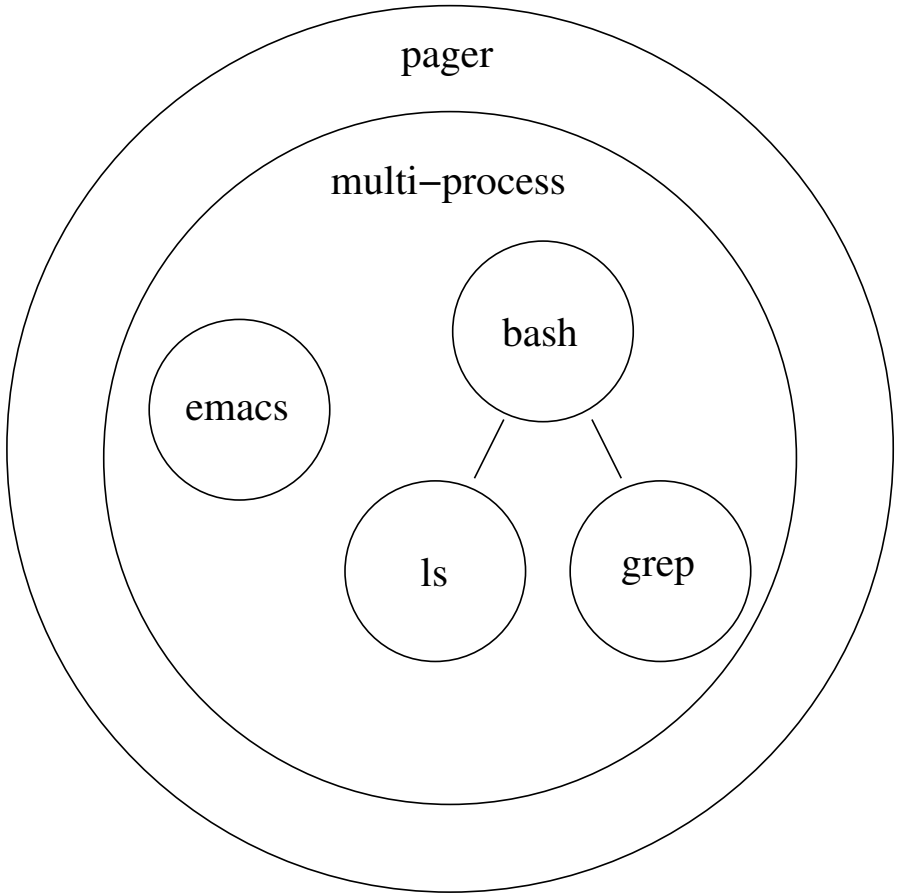
- Not faithful to an existing hardware interface
- New architecture
  - Stacking - nested process model
  - Capabilities
  - Selective interposition
  - Better abstractions (file handles vs I/O registers)
  - Services don't need to be adapted to multidomain case
- Advantages
  - Short circuit hierarchy traversal via capabilities
  - Only interfere where required
  - Eliminate non-relevant code (e.g., paging code)

RVMs

- Not faithful to an existing hardware interface
- New architecture
  - Stacking - nested process model
  - Capabilities
  - Selective interposition
  - Better abstractions (file handles vs I/O registers)
  - Services don't need to be adapted to multidomain case
- Advantages
  - Short circuit hierarchy traversal via capabilities
  - Only interfere where required
  - Eliminate non-relevant code (e.g., paging code)

Services

- Paging
- Checkpointing
- Multi-process environment
- Reference monitors
- Debugging and tracing



Necessary Properties

- State Encapsulation
  - Entire state of child available to parent
- Border Control
  - All external communication passes via border
  - Reference monitors

State Encapsulation

- Hierarchical resource management - easier to cleanup
- State is often hidden in kernel
  - Long running syscalls
  - Update visible registers
  - `Call` appears as a `Receive` after send phase
- Reference relativity
  - Avoid absolute addresses – especially for internal objects
  - Use a naming context
  - Important for migration and check pointing

Fluke

- Basic instruction set
  - Single implementation
  - Microkernel
- Common API
  - Independently implementable
  - Interposable

Evaluation

- System can run gcc, etc.
- Each layer represents a 0% to 30% slowdown

Questions

- Where are the file systems? Like a multi-server system?
- If servers get memory from clients, what is the strategy to protect from abuse.
- How is memory dynamically reallocated (what does revocation look like?)