Highly Configurable Operating Systems for Ultrascale Architectures

Arthur B. (Barney) Maccabe
Computer Science & HPC@UNM
University of New Mexico
What’s wrong with current operating systems?
Clusters and NICs

- DMA between NIC and Host Memory
  - Physical addresses on NIC
  - Can be user or system space
  - Memory descriptors on NIC

- Benefits associated with offloading
  - Reduced overhead
  - Increased bandwidth
  - Reduced latency
OS Bypass ➔ Splintering

Cluster Architecture

Host Processor
Host Memory
Memory Bus
Bridge
I/O Bus
Network Interface

OS as a Bottleneck

data → NIC → Operating System → data → Application

OS Bypass

packets → NIC → allocate → app desc → data & control → Application

Offload

data → NIC → TCP/IP → data → Operating System → data → Application

Splintering

packets → NIC → L2 → L1 → Operating System → data → Application

Distributing little bits of the OS

COSET-I
June 26, 2004
Comparing Cougar and Linux

Millions of operations per second

NAS B: mg

Number of nodes

Cougar
Linux

Comparing Cougar and Linux

Millions of operations per second

NAS B: cg

Number of nodes

Cougar
Linux
Other Issues

- Full featured operating systems are subject to “rogue” OS effects (Fabrizio ‘03, Jones ‘03)
  - “noise” caused by OS services running at the wrong time
  - can lead to significant expansion in execution time for applications

- Lightweight operating systems do not provide features needed by many applications

- OS is an impediment to new architectures and programming models
Factors Impacting OS Design
Factors

- Lightweight OS
  - small collection of apps
    - single programming model
  - single architecture
  - single usage model
  - small set of shared services
  - no history :)

- Puma/Cougar
  - MPI
  - Distributed memory
  - space shared
  - parallel file system
Programming Models

- MPI
- Open MP
- Parcels

Runtime Models
- Process
- Thread
- Active Message

Architecture Models
- Distributed Memory
- Shared Memory
- PIM
Usage Models

Number of apps running

Capability to Capacity

Principal Apps

time

Number
of apps
running
Our Approach: micro-services that can be distributed (splintering) and composed

Build a custom OS for each application, architecture and usage model

More composition than adaptation
Building Custom OSes

- Application
  - Application Analysis Tool
    - Application Requirements
      - Available Shared Resources
      - Shared Resource Requirements
      - System Usage Model
      - Architecture
    - OS/Runtime Constructor
      - OS/Runtime
      - Micro-services
Example: Signal Delivery

Application

Interrupt Listener

Coalescing Listener

Extensible Listener

Message Listener

Observer

Observer

Observer

Observer

CPU scheduler

CPU scheduler

CPU scheduler
Related Work & Conclusions

- Related work -- OS/runtime Specialization
  - Microkernels
    - K42, L4, Pebble, Mach, ....
    - Exo-kernel
  - Extensible OSes
    - Spin, Vino, sandboxing, ..... 
    - Modules, everyone has modules.... :) 
  - Configurable OS/Runtime
    - Scout, Think, Flux OSKit, eCos, TinyOS
    - SREAMS, x-kernel, CORDS
- Framework for micro-services
  - distribute
  - compose and analyze
  - usage models
Thanks

- UNM
  - Patrick Bridges
  - about 15 students
- Sandia National Labs
  - Ron Brightwell
  - Rolf Riesen
- OS Research
  - Trammel Hudson
- CalTech
  - Thomas Sterling