OS for HPC in Exascale Era

Ruibo Wang
National University of Defense Technology
China
Content

• OS for HPC
• Lightweight kernel
• Full-weight kernel
• Multi-kernel
• Our attempt
Content

• OS for HPC
• Lightweight kernel
• Full-weight kernel
• Multi-kernel
• Our attempt
OS for HPC

• Two goals:
  • Performance
  • Application compatibility

• Performance
  • Deliver the maximum capability of the hardware
  • Requires thin OS or lightweight OS

• Application compatibility
  • To provide Linux environment most application assume
  • Requires full-weight OS
OS for HPC

• Trends:
  • LWK -> FWK, add Linux environment or API to LWK
  • FWK -> LWK, strip Linux to be lightweight
  • LWK & FWK, multi-kernel on the same node, aiming to achieve the two contradictory goals
Content

• OS for HPC
• **Lightweight kernel**
• Full-weight kernel
• Multi-kernel
• Our attempt
LWK’s origin

• Early days
  • Vector machine, MPP
    • Highly specialized chips and architecture, rare commodity hardware
  • Apps are highly coupled with hardware system
    • Highly involved in hardware management
  • Narrow range of app
    • Scientific computing
  • Small memory on compute node
LWK’s philosophy

• Highly customized OS
  • Minimal features
    • Low OS noise, highly scalable
    • Emphasize efficiency over functionality
  • Thin hardware management and abstraction layer
    • User-managed

• Small memory footprint

• High message-passing performance
SUNMOS (1991)

- For Intel Paragon (1993), much like an app launcher
- Single tasking
  - Application manages all the resources
- Small memory footprint
  - Only 16MB on compute node, SUNMOS occupies 250k
Catamount (2004)

- Move as much functionalities out to user-space (PCT) as possible
  - Policy part in PCT and mechanism part in QK
    - Memory and process management
    - Job queueing
    - May have several different PCTs
- Compute nodes only focus on high performance computing
  - Relies on service nodes (running Linux) to provide wide functionalities
Kitten (1999)

- LWK + VMM hypervisor
- Linux env is provided by guest OS over hypervisor
- Kernel/init are like QT/PCT in Catamount
- Based on Linux code
  - Performance critical part rewritten memory management, task management, virtual memory management
- Linux ABI and syscall compatible
- Like Linux striping way

- For Blue Gene/L, Blue Gene/P, Blue Gene/Q
- Provide Linux-like environment while keep LWK advantages
  - Libc and syscall level
- IO/service proxy
  - Delegated to IO/service nodes
- Performance critical
  - Non-preemptive scheduler
  - Static TLB mapping
  - Big memory allocation
LWK Focuses on performance

• Design space:
  • HPC is more for space-sharing rather than time-sharing

• Process schedule
  • Non-preemptive
    • Pros: low noise, high scalability – good for HPC
    • Cons: limit different combination of threads, overcommit of threads – do not care

• Memory management、Simple memory mapping
  • Large page
    • Pros: less TLB/cache miss – good for HPC
    • Cons: more memory waste – do not care
Content

• OS for HPC
• Lightweight kernel
• **Full-weight kernel**
• Multi-kernel
• Our attempt
Full-weight Kernel

- Commodity clusters over MPP
- Linux dominated
  - Commodity clusters and hardware
  - Applications need Linux environment
- Various Commodity hardware is driven by Linux
  - A burden work for HPC world to adapt their OS to
- Application developers assume Linux environment
  - Various code base and support
  - Out-of-the-box running
- Tuning Linux to achieve high performance and scalability
Full-weight Kernel

• The design choices of OSs were most often trade-offs
• Linux is designed for server market, need to be tuned to fit HPC
• Server market vs HPC

<table>
<thead>
<tr>
<th>Design concerns</th>
<th>Server market</th>
<th>HPC compute node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Scheduling</td>
<td>Max overall throughput</td>
<td>Max single app performance</td>
</tr>
<tr>
<td>Memory management/virtual space</td>
<td>Max overall usage, lazy allocation</td>
<td>Max single app performance</td>
</tr>
<tr>
<td>Space management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IO</td>
<td>Frequent Small files access</td>
<td>Large file access</td>
</tr>
<tr>
<td>Message passing/network</td>
<td>Multi-layers, multi-protocol supported</td>
<td>Max message passing performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Less copy</td>
</tr>
<tr>
<td>Application compatibility</td>
<td>Linux environment</td>
<td>Linux environment</td>
</tr>
</tbody>
</table>
Compute Node Linux (CNL) (2005-2007)

- Catamount for Cray’s XT3/4 -> CNL for Cray’s XT5
- Simplified scheduling, memory management, network, file system
- Large page
- Confine background kernel services and interrupt handling on some cores

![Graph showing performance metrics]
ZeptoOS (2004)

- Ported to IBM Blue Gene’s compute nodes, Linux based
- Memory management: Big Memory
  - Large v-p mapping
- Linux can be performance competitive
K OS (2011)

• For K
• Optimize scheduling of system Daemon
K OS (2011)

- For K
- RDMA to send data noiselessly
K OS (2011)

• For K
• Large page
  • Reserve some at boot time
Full-weight Kernel

- The design choices of OSs were most often trade-offs
- Linux is designed for server market, need to be tuned to fit HPC
- Server market vs HPC

<table>
<thead>
<tr>
<th>Design concerns</th>
<th>Server market</th>
<th>HPC compute node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Scheduling</td>
<td>Max overall throughput</td>
<td>Max single app performance</td>
</tr>
<tr>
<td>Memory management/virtual space管理</td>
<td>Max overall usage, lazy allocation</td>
<td>Max single app performance</td>
</tr>
<tr>
<td>IO</td>
<td>Frequent Small files access</td>
<td>Large file access</td>
</tr>
<tr>
<td>Message passing/network</td>
<td>Multi-layers, multi-protocol supported</td>
<td>Max message passing performance</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Less copy</td>
</tr>
<tr>
<td>Application compatibility</td>
<td>Linux environment</td>
<td>Linux environment</td>
</tr>
</tbody>
</table>
Content

• OS for HPC
• Lightweight kernel
• Full-weight kernel
• Multi-kernel
• Our attempt
Multi-kernel

- Achieve the two contradictory goals with two kernels running on the same node
  - Performance – LWK
  - App compatibility – Linux
- Many-core and heterogeneous architecture
  - Linux on large cores for general service
  - LWK on small cores for lightweight computing
FusedOS (2011)

• Successor of IBM CNK
• CNK library (CL) as proxy process
  • Syscall offloaded to CL

![Diagram showing the architecture of FusedOS](image-url)
Hobbes (2013)

• Different jobs in an app calls for different environments

• Key components
  • Pisces resource management – hardware resource partitioning
  • Kitten LWK
  • Palacios VMM

• Kitten LWK + Linux over VMM
  • LWK + FWK
McKernel (2012)

- Manages compute node of Fugaku
- McKernel LWK implements
  - Performance critical syscall
  - Others offload to Linux
  - CPU and memory management
  - Independent of Linux, standalone code
- Interface for Heterogeneous Kernels (IHK)
  - Communication between FWK and LWK
  - Partition of resource
mOS (2014)

- Code integrated into Linux
  - Leverage Linux process struct
  - Leverage most Linux
- LWK implement performance critical part
  - Scheduling
  - Memory management
- Syscall delegation
  - by migrating process to FWK
FFMK (2009)

- L4 Microkernel + $l^4$ Linux
- LWK first manages hardware
- $l^4$ Linux (paravirtualized) as FWK
HermitCore (2016)

• Unikernel
  • Can be run directly on bare hardware
  • Can act as LWK along with Linux
Multi-kernel

- Promising to achieve high performance and app compatibility
  - By partitioning resource and designs to cater to different needs
  - What about FWK + several LWK
    - many kernels
    - To cater to different needs
Content

• OS for HPC
• Lightweight kernel
• Full-weight kernel
• Multi-kernel
• Our attempt
Our attempt

• We are facing unprecedented diversity in new era
• Different application needs
• Different hardware
Our attempt

• Three parts
  • Device management and partition
  • Service zone
  • Specialized zone
Our attempt

• Three parts
  • Device management and partition
  • Service zone – a FWK, offers Linux support
  • Specialized zone – many LWKs, specialized execution for different apps
Our attempt

• To upper applications
  • Linux compatibility – provide by service proxy and Linux
  • Application running environment – provide by running env and service proxy

- High throughput
- High performance
- Low energy
- Delegate
- Communication
- Linux
- Service
- Device management
- Hardware
- Running env
- Service proxy
- Hardware Management
Our attempt

• To underlying hardware
  • Hardware management mechanism – provided by Device management
  • Hardware management policy – provided by different LWK
    • To different app needs
Our attempt

• Like many kernels
• Large containers
  • Contains app env as well as kernel functionalities
  • Specialized for application, in terms of both running environment and hardware management
  • Isolation
Proposal for exascale

• Very large scale, 100k+ nodes
• heterogenous, accelerator is inevitable
• CPU centered -> NIC centered
  • The control node controls multiple compute node with different computing resources
  • Different accelerators or CPUs
  • General services delegated to control node
  • Network very fast inside a supernode
    • On the same board or near
Thank you

ruibo@nudt.edu.cn