mOS

An Architecture for Extreme-Scale Operating Systems

Robert W. Wisniewski, Todd Inglett, Pardo Keppel, Ravi Murty, Rolf Riesen

ROSS 2014 (Best Paper Award)

Motivation

- Exascale HPC systems
- Established applications and libraries

Motivation

- Exascale HPC systems
- Established applications and libraries

Evolution: Full-Weight kernel (FWK)

- Linux or Linux API

Revolution: Light-Weight kernel (LWK)

- performance
- scalability
- reliability
- flexibility

Motivation

- Exascale HPC systems
- Established applications and libraries

Evolution: Full-Weight kernel (FWK)

- Linux or Linux API

$\Rightarrow$ remove features

Revolution: Light-Weight kernel (LWK)

- performance
- scalability
- reliability
- flexibility

$\Rightarrow$ add functionality

multi Operating System (mOS) = LWK + FWK

Goals

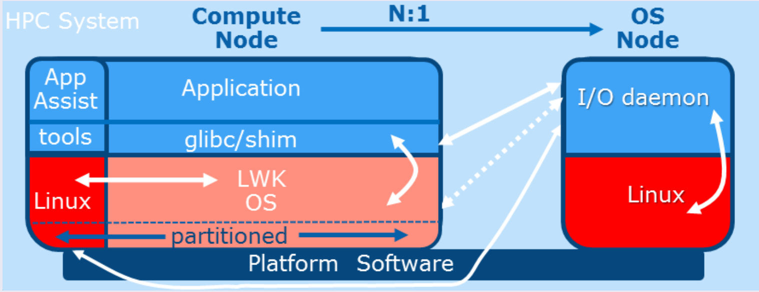
- Symbiosis of LWK and FWK: “best of both worlds”
- Minimal modifications to Linux
- Testbed for new technologies
- Hierarchical syscall mechanism (LWK > FWK > OS node)

multi Operating System (mOS) = LWK + FWK

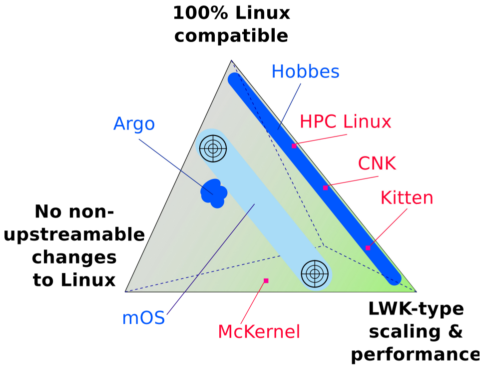
Goals

- Symbiosis of LWK and FWK: “best of both worlds”
- Minimal modifications to Linux
- Testbed for new technologies
- Hierarchical syscall mechanism (LWK > FWK > OS node)

System Architecture



Design Space



Argo	ServiceOS for setup + specialized ComputeOS instances
McKernel	Linux + LWK, originally for CPU + Xeon Phi, no OS nodes
Hobbes	Virtualization for collocation of dependent applications
Kitten	LWK with embedded VMM *Palacios*
CNK	*Compute Node Kernel*, minimalistic LWK used on *Blue Gene*

Architecture

- Linux + LWK on each compute node
- Leverage Linux to simplify LWK
- Configurable resource partitions
- *Cooperative Agent Kernel Extensions* (CAKE), cp. FUSE

Linux

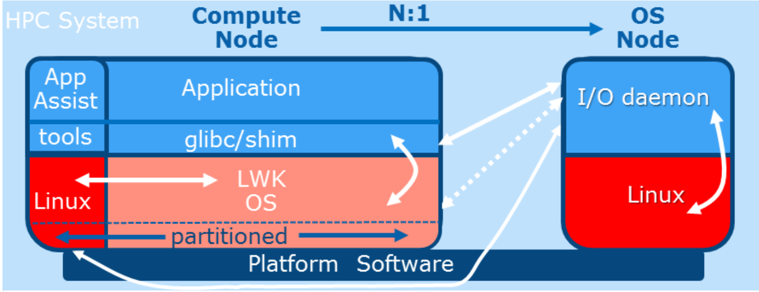
- Booting + hardware setup
- Linux functionality (e.g. TCP/IP sockets, signaling mechanisms, . . .)
- Infrastructure services (e.g. job launch and monitoring)

LWK

- Memory management: physically contiguous regions
- Scheduling: cooperative multitasking
- System call forwarding to local FWK or OSN

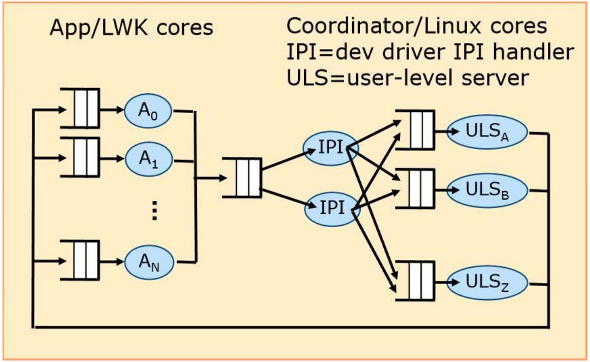
System Calls

- Handled by different parts of the system — LWK, FWK, OSN
- Hierarchical triage
  - User-level interception (and aggregation) in *glibc* via `LD_PRELOAD`
  - Offload to OS node or forward to LWK
  - LWK: forward syscall to FWK if not performance critical



LWK-FWK Communication

- Explicit communication between LWK and FWK (*function shipping*)
- Messages via channels built with shared memory and IPIs
- Driven by FWK cores to reduce effect on application
- LWK channels may disable IPIs



LWK-FWK Communication II

Message Send Example

```
read(fd, buf, len):
  payload = { READ, fd, buf, len }
  msg = { route=pid, context=lwk_pid, payload }
  was = linux_channel.q.insert(msg)
  if (!was):
    linux_channel.receiver.send_ipi()
  lwk_channel.wait_for_ack()
```

IPI Receive/Dispatch Example

```
ipi handler():
  msg_list = linux_channel.q.get_list()
  while ((msg = msg_list.pop()) != NULL):
    pid = msg$\rightarrow$route
    channel[pid].q.insert(msg)
    sched_run(pid)
```

OS Nodes and Partitioning

OS Nodes

- Perform parallel filesystem I/O on behalf compute nodes
- Reduce number of PFS clients
- Reduce jitter, cache pollution and memory usage on compute nodes

Resource Partitioning

- Static
- Cores, memory: restrict Linux' resource usage via kernel command line (`mem=m`, `maxcpus=n`) and adapt ACPI tables
- Devices
  - Linux configures HPC network, then hands it to the application on LWK
  - Other devices handled by Linux

Evaluation

No implementation/prototype available

Conclusion

Summary

- mOS: LWK + FWK (+ OS nodes)
- Hardware partitioning
- Hierarchical syscall triage
- Explicit communication

Conclusion

Summary

- mOS: LWK + FWK (+ OS nodes)
- Hardware partitioning
- Hierarchical syscall triage
- Explicit communication

Discussion Points

- Design space
- Early development stage
- mOS vs. FFMK