IHK/McKernel Overview and Status Report

Balazs Gerofi
RIKEN Advanced Institute for Computational Science (AICS)
JAPAN

2015/11/17 OS/R BoF @ SC'15

RIKEN

Motivation

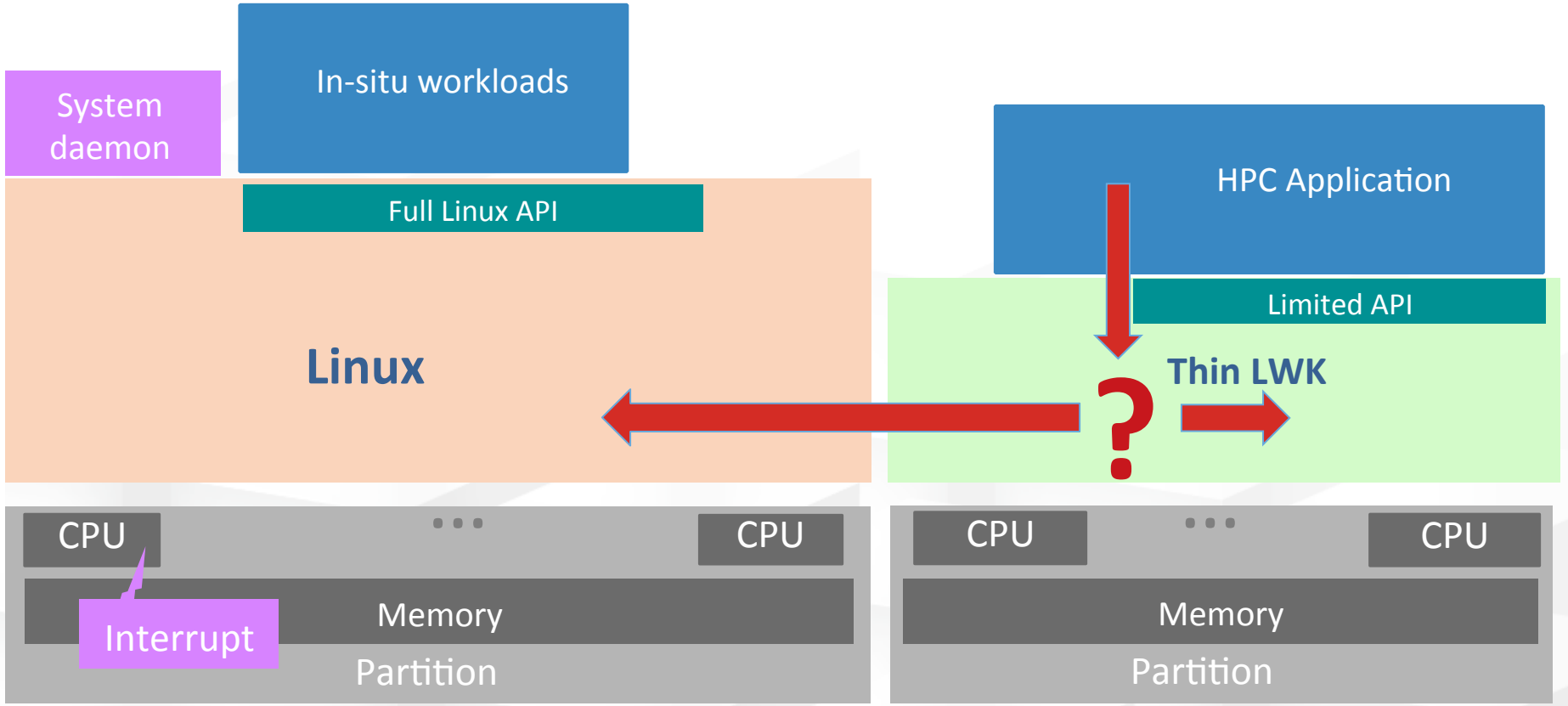
- **Complexity of high-end HPC systems keeps growing**
  - Extreme degree of parallelism
  - Heterogeneous core architectures
  - Deep memory hierarchy
  - Power constrains
- **Applications have also become complex**
  - In-situ data analytics, computational steering, workflows, etc..
  - Sophisticated monitoring and tools support
- **Need for scalable, consistent performance and capability to rapidly adapt to new HW features and programming paradigms**
  - Historically provided by LWKs, but usually provide only limited APIs
- **Growing importance of POSIX and the Linux APIs**
  - /proc, /sys, etc..

- **How to address all these at the same time?**

RIKEN

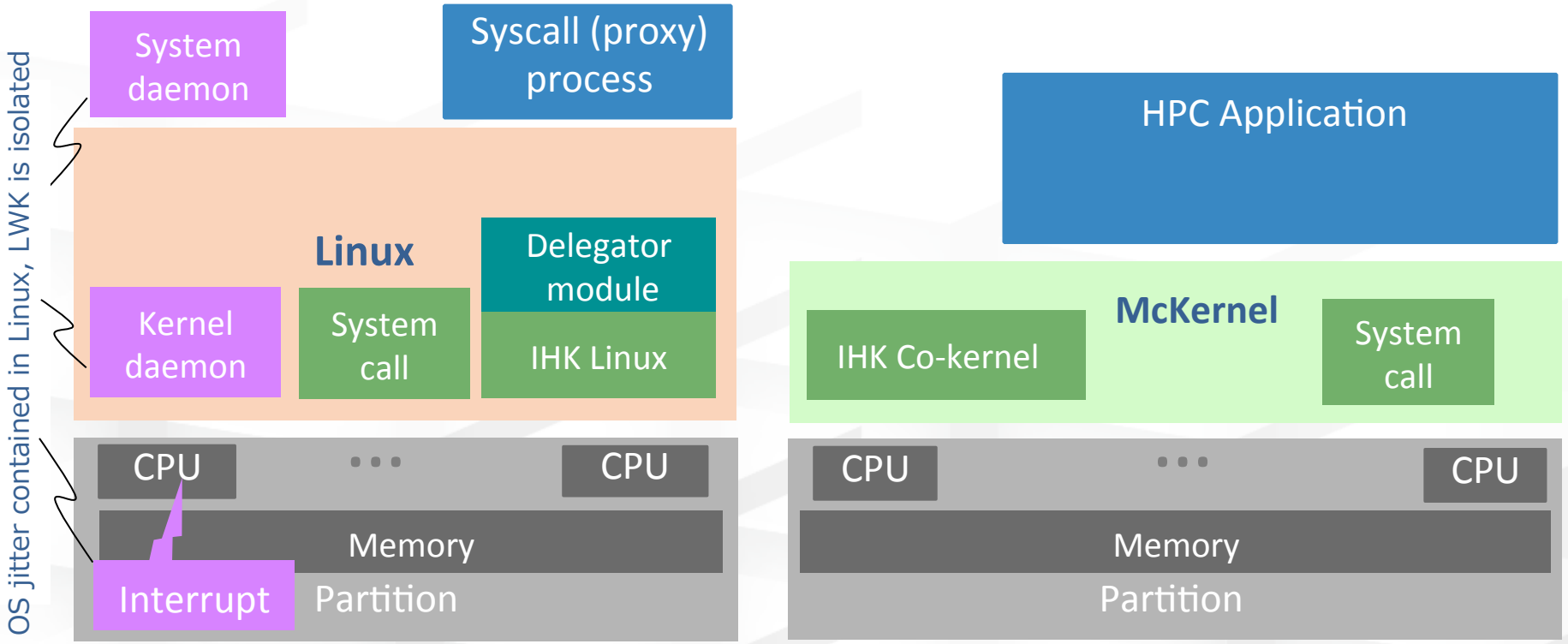
High Level Approach: Hybrid Linux + LWK

- **With the abundance of CPU cores comes the new hybrid approach: run Linux and LWK side-by-side in compute nodes!**
- **Partition resources (CPU core, memory) explicitly**
- **Run HPC apps on LWK**
- **Selectively serve OS features with the help of Linux by offloading requests**



IHK/McKernel Architectural Overview

- **Interface for Heterogeneous Kernels (IHK):**
  - Allows dynamically partitioning of node resources (i.e., CPU cores, physical memory, etc.)
    - SMP chip and accelerator support (e.g., Xeon Phi)
  - Enables management of LWKs (assign resources, load, boot, destroy, etc..)
  - Provides inter-kernel communication (IKC), messaging and notification
- **McKernel:**
  - A lightweight kernel developed from scratch, boots from IHK
  - Designed for HPC, noiseless, simple, implements only performance sensitive system calls (roughly process and memory management) and the rest are offloaded to Linux



McKernel Implementation Status (system calls, Linux APIs, GDB support)

- System calls not listed in the table are offloaded to Linux
- POSIX compliance: 100% of the LTP test suite (2013 version) passes!

	Implemented	In progress
Process Thread	arch_prctl, clone, execve, exit, exit_group, fork, futex, getpid, getrlimit, kill, pause, ptrace, rt_sigaction, rt_sigpending, rt_sigprocmask, rt_sigqueueinfo, rt_sigreturn, rt_sigsuspend, set_tid_address, setpgid, sigaltstack, tgkill, vfork, wait4, signalfd, signalfd4	get_thread_area, getrlimit, ptrace, rt_sigtimedwait, set_thread_area, setrlimit
Memory management	brk, gettid, madvise, mlock, mmap, mprotect, mremap, munlock, munmap, remap_file_pages, shmat, shmctl, shmdt, shmget	get_robust_list, mincore, mlockall, modify_ldt, munlockall, set_robust_list
Scheduling	sched_getaffinity, sched_setaffinity, getitimer, gettimeofday, nanosleep, sched_yield, settimeofday	setitimer, time, times
Performance Counter	Direct PMC interface: pmc_init, pmc_start, pmc_stop, pmc_reset	PAPI Interface

- **`/proc` and `/sys`:**
  - In progress (built on top of a modified version of `overlayfs`)
- **GDB support:**
  - In progress (basically `ptrace()` support)
- **Exposing NUMA information, memory types + standard NUMA calls (`numa_*()`, `mbind()`, `move_pages()`, etc..)**
  - In progress (IHK + McKernel physical memory management changes)

RIKEN

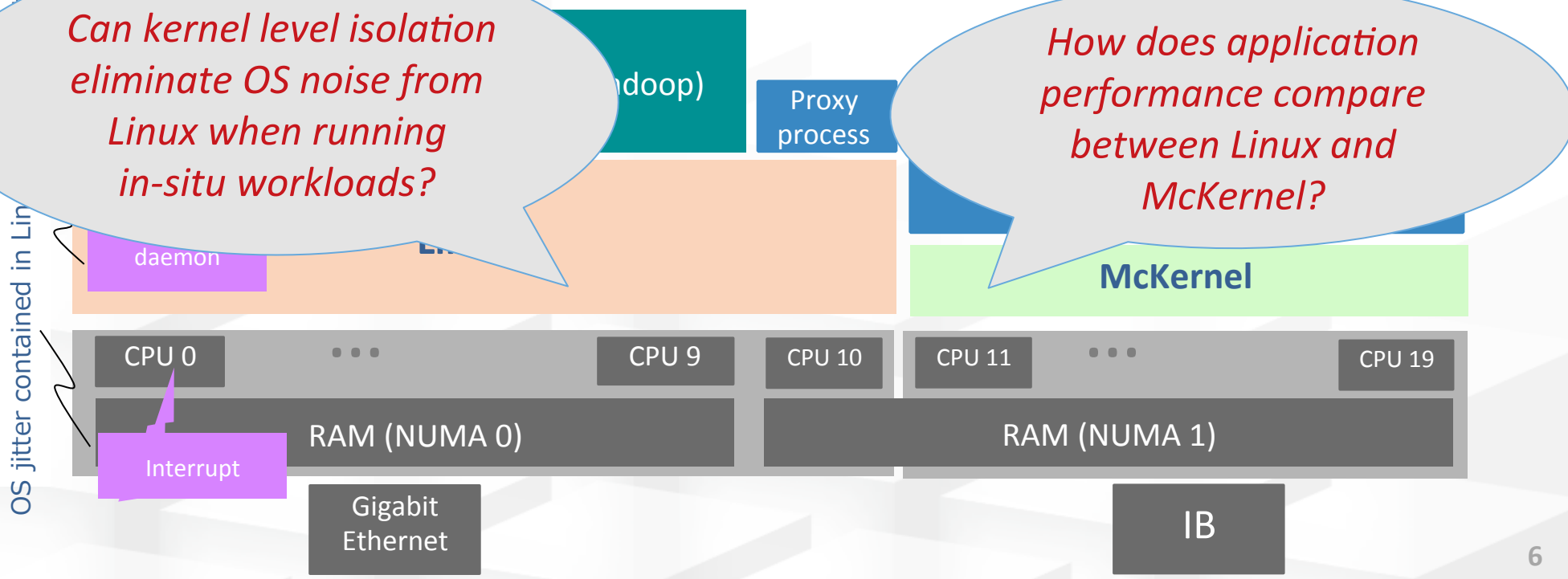
IHK/McKernel Scalability and Performance Isolation Measurements with and without in-situ Hadoop Workloads

- **Test platform (64 nodes cluster):**
  - Intel Xeon E5-2680 (Ivy Bridge) - 2 sockets, 10 core
  - 64GB RAM (2 NUMA domains)
  - Gigabit Ethernet and Mellanox Infiniband MT
- **Software environment:**
  - Red Hat Enterprise Linux ComputeNode Rel
    - Linux kernel version 2.6.32
  - MVAPICH 2.2a
  - Intel Compiler version 16.0.0
  - Hadoop 2.7.1 + HiBench

McKernel runs the exact same binary as Linux!
No MPI or Linux device driver changes!

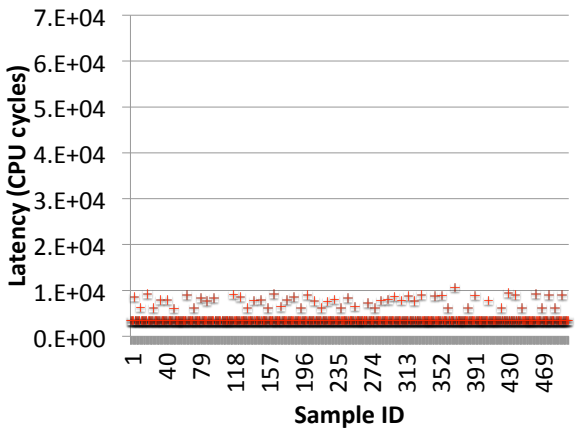
Can kernel level isolation eliminate OS noise from Linux when running in-situ workloads?

How does application performance compare between Linux and McKernel?

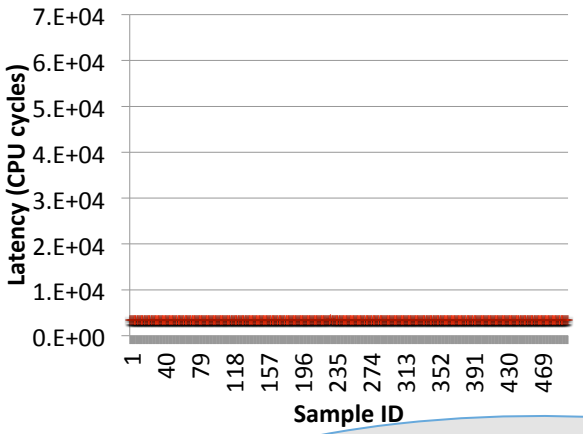


FWQ noise measurements

Without in-situ workload

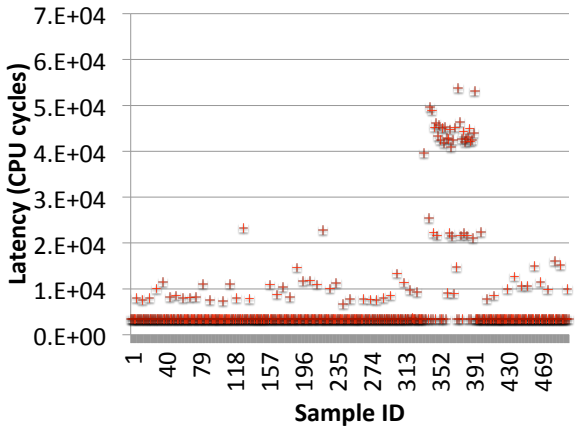


Linux + cgroup

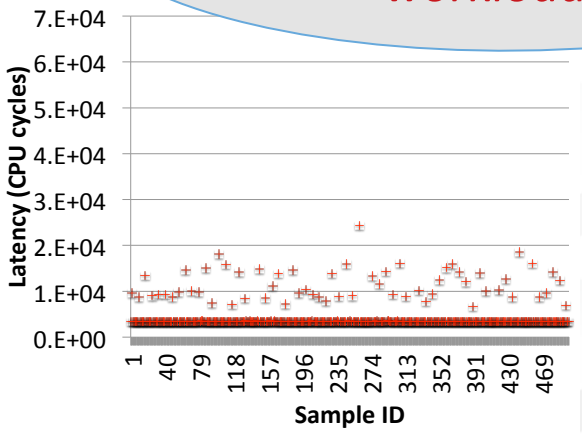


- **Five configurations:**
- **Without Hadoop:**
  - Linux + cgroup
  - McKernel
- **With Hadoop:**
  - Linux + cgroup
  - Linux + cgroup + isolcpus
  - McKernel

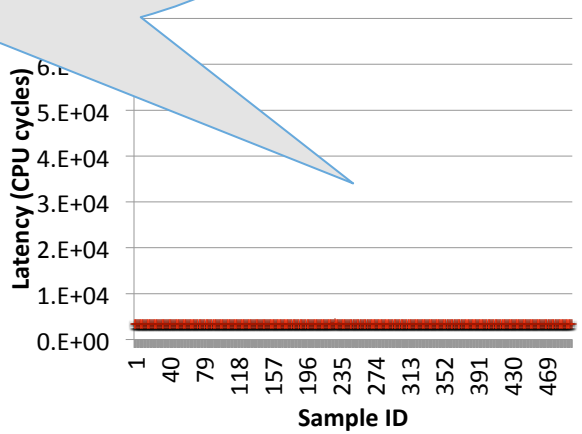
With in-situ workload



Linux + cgroup



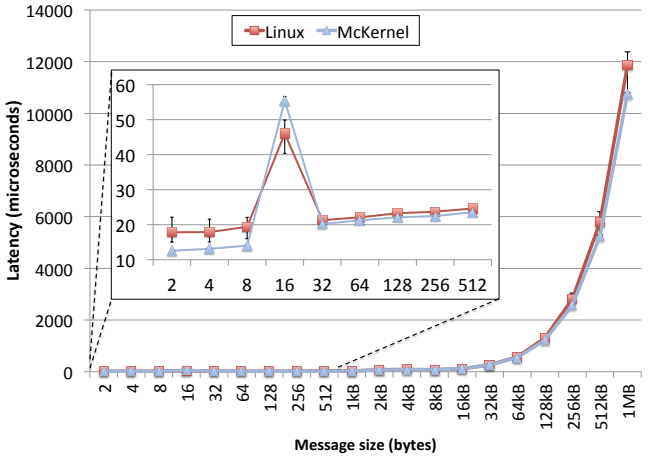
Linux + cgroup + isolcpus



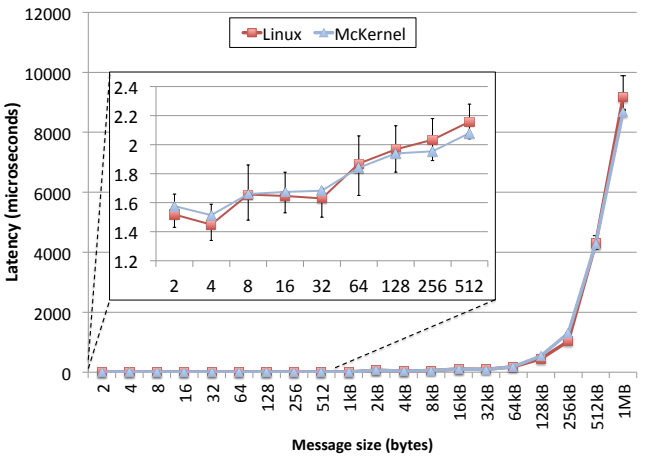
McKernel

Completely noiseless even in face of in-situ workloads!

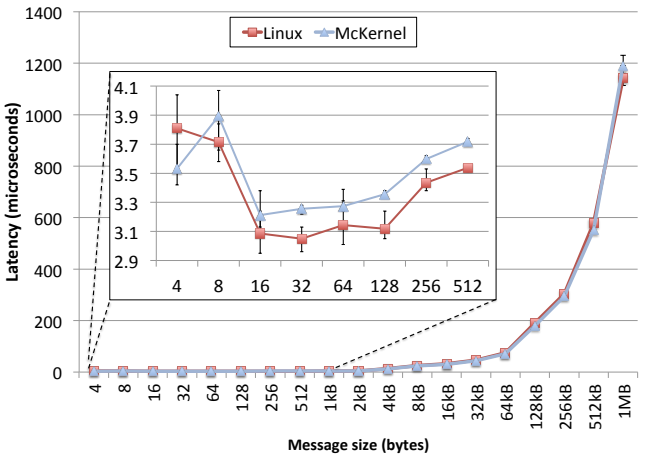
OSU MPI Benchmarks – Collective Routines (64 nodes)



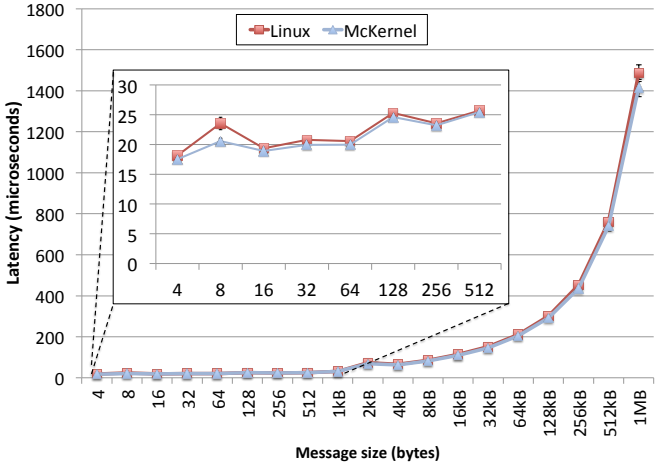
MPI_Scatter



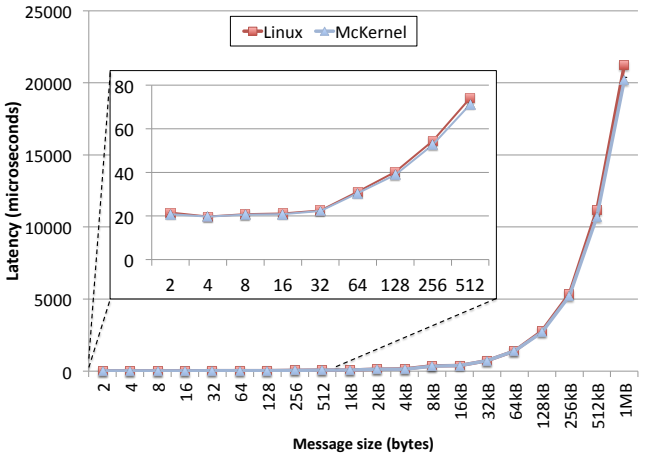
MPI_Gather



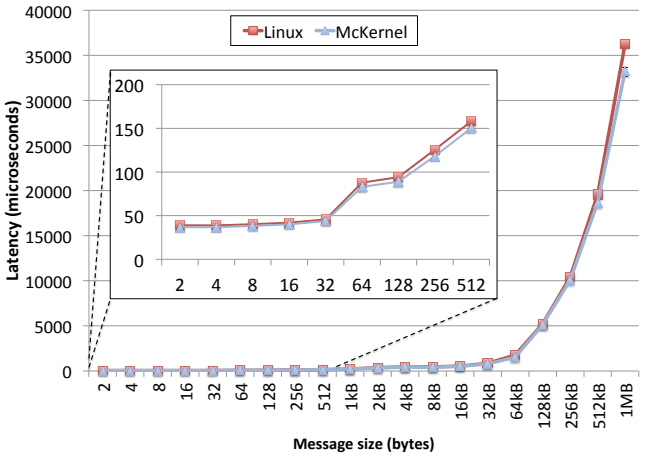
MPI_Reduce



MPI_Allreduce



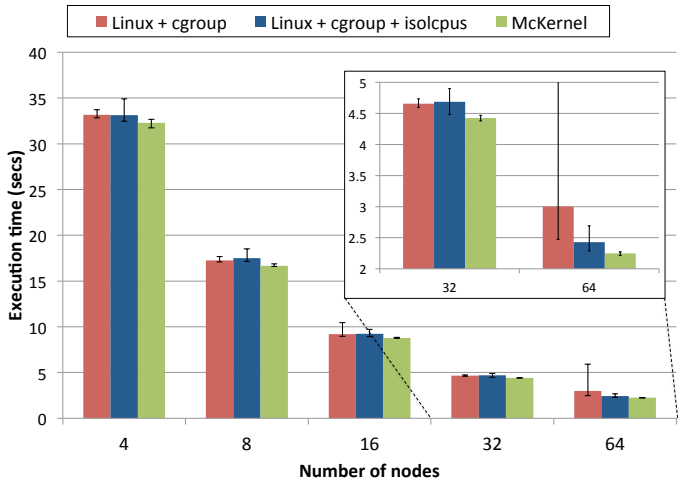
MPI_Allgather



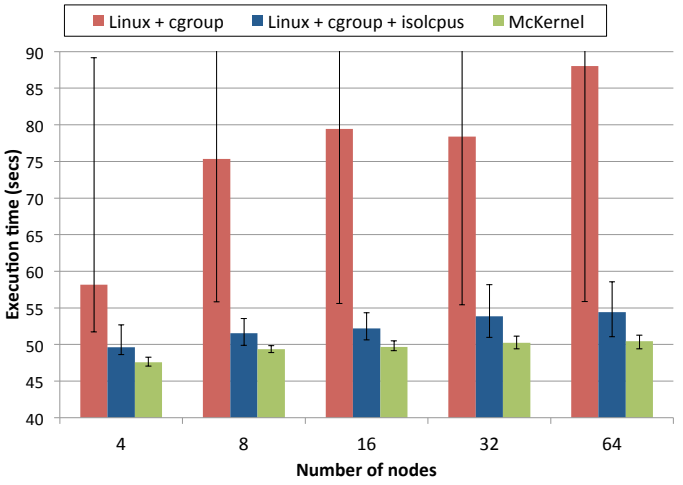
MPI_Alltoall

- **McKernel performs on par with Linux**
- **But most notably: performance variation is smaller on McKernel!**

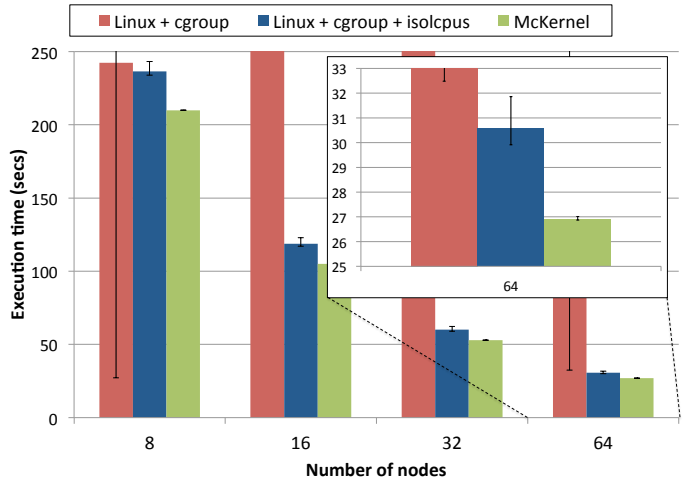
MiniApp Performance in face of in-situ Hadoop



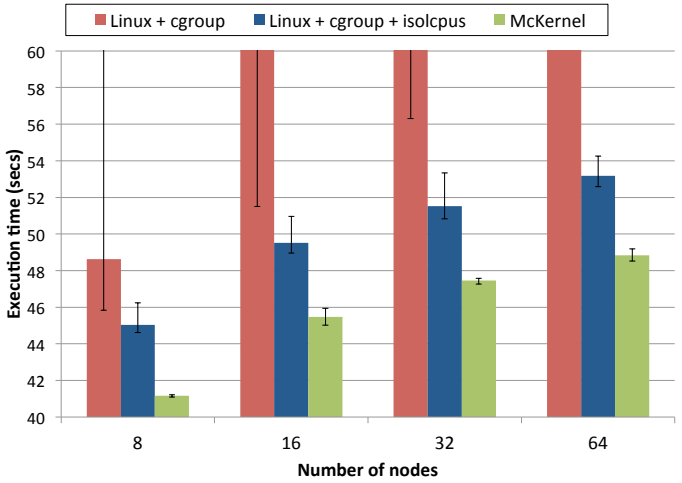
miniFE (CORAL suite)



HPC-CG (Mantevo suite)



Modylas (Fyber suite)



FFVC (Fyber suite)

- **McKernel has substantially lower performance variation, isolates simulation and in-situ workload more efficiently**

Conclusions (Distinguishing Features)

- **Scalable and consistent performance for bulk synchronous codes**
- **Full Linux ABI compatibility**
  - System call I/F and VDSO
  - Runs the exact same binary as Linux
- **Full Linux API support on LWK**
  - Including `/proc` and `/sys` filesystems (in progress)
- **Kernel level workload isolation**
- **Transparent Linux device driver resuse**
  - No need for porting drivers to the LWK
  - No need for Infiniband or MPI modifications
- **No Linux kernel modifications**
  - Implemented as a collection of kernel modules
- **Independent LWK code base**
  - Enables rapid OS prototyping

RIKEN

Thank you for your attention!
Questions?

- **Interested in more details?**
- **Come and join one of the McKernel tutorials @ PC Cluster Consortium booth!**
  - Tue 13:30~
  - Wed 10:00~
  - Thu 13:30~

RIKEN