Experiences with a Lightweight Multi-Kernel Operating System for Extreme Scale Computing

Balazs Gerofi <bgerofi@riken.jp>

System Software R&D Team

RIKEN Center for Computational Science

Intel eXtreme Performance Users Group (IXPUG), held together with HPCAsia'20 2020/Jan/17, Fukuoka, Japan

McKernel

Outline

- Motivation
- Lightweight Multi-kernels
- McKernel Design and Implementation
- Oakforest–PACS Evaluation
- Preliminary Results on ARM ThunderX2
- Future Perspectives
- Summary

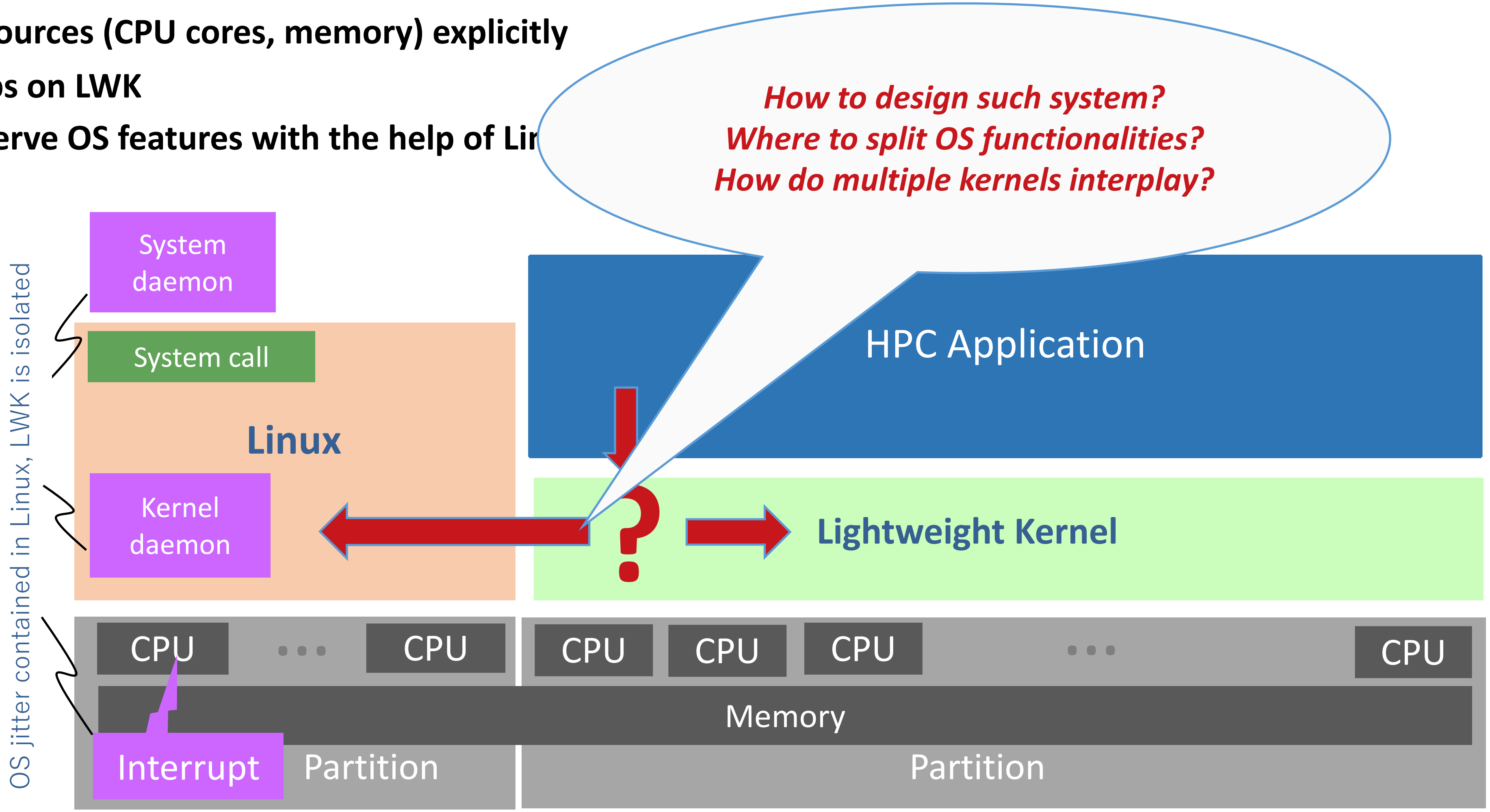
Motivation

- **Node architecture: increasing complexity**
  - Large number of (heterogeneous) processing elements (e.g., CPU cores), deep memory hierarchy, complex cache/NUMA topology
- **Applications: ever expanding diversity**
  - Traditional/regular HPC simulations +
  - in-situ data analytics +
  - Big Data processing +
  - Machine Learning +
  - Workflows, etc.
- **What do we need from the system software/OS?**
  - Performance and scalability for large scale parallel apps
  - Support for Linux APIs – tools, productivity, monitoring, etc.
  - Full control over HW resources
  - Ability to adapt to HW changes!
    - Emerging memory technologies, parallelism, power constrains
  - Performance isolation and dynamic reconfiguration
    - According to workload characteristics, support for co-location, multi-tenancy

Design and Implementation

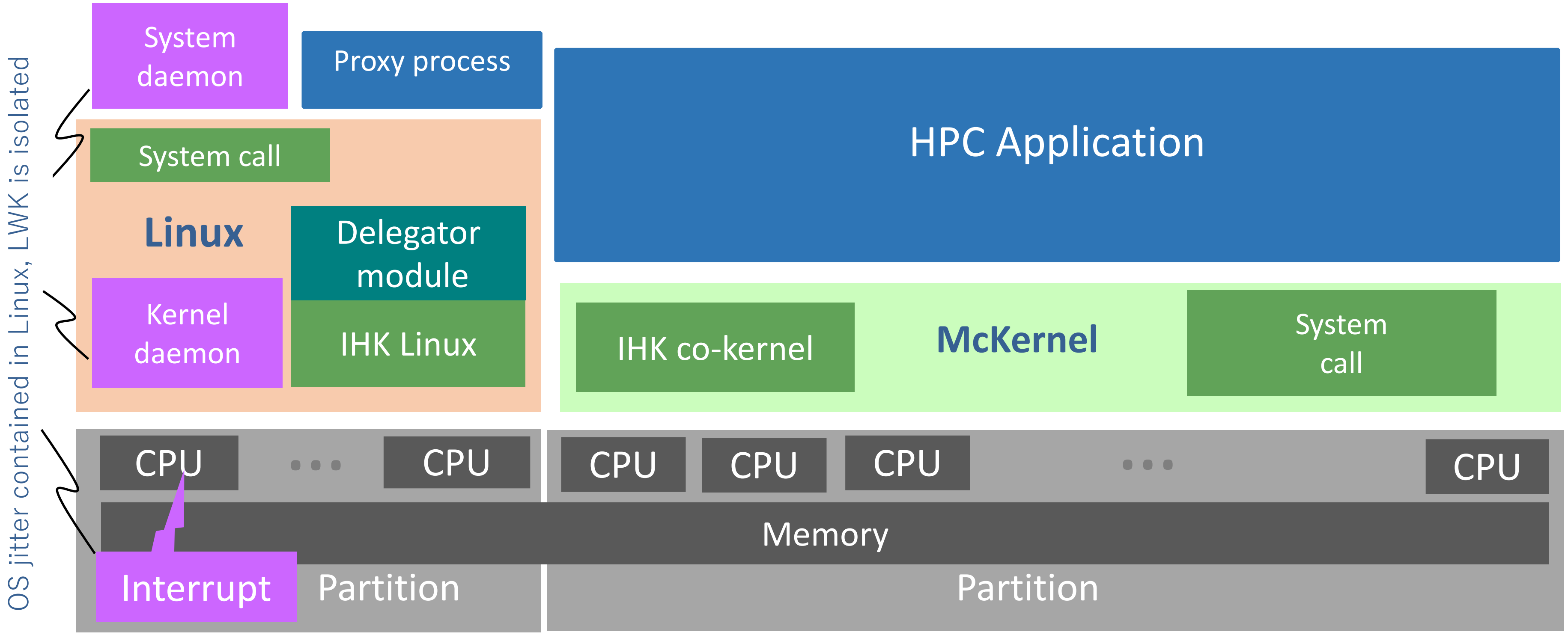
Approach: Lightweight Multi-kernel

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



IHK/McKernel: Architectural Overview

- **Interface for Heterogeneous Kernels (IHK):**
  - Allows ***dynamic partitioning*** of node resources (i.e., CPU cores, physical memory, etc.)
  - Enables management of multi-kernels (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
    - Mostly process and memory management, the rest are offloaded to Linux



McKernel and system calls

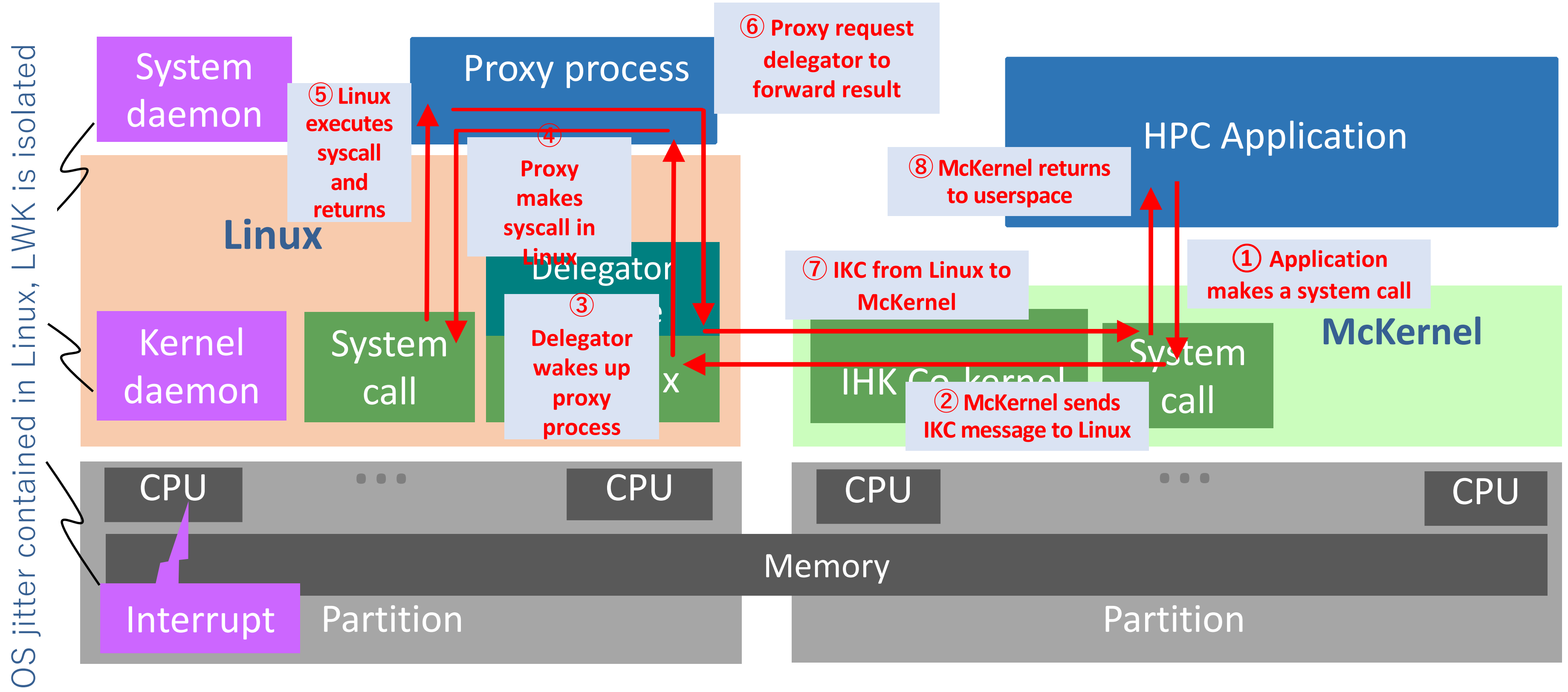
- **McKernel is a lightweight (co-)kernel designed for HPC**
- **Linux ABI compatible**
- **Boots from IHK (no intention to boot it stand-alone)**

	Implemented	Planned/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,	ftrace?
Memory management	brk, sbrk, madvise, mlock, mmap, mprotect, mremap, munlock, munmap, remap_file_pages, shmat, shmctl, shmdt, shmget, mbind, set_mempolicy, get_mempolicy, mbind, move_pages	
Scheduling	sched_getaffinity, sched_setaffinity, getitimer, gettimeofday, nanosleep, sched_yield, settimeofday	
Performance counters	direct PMC interface: pmc_init, pmc_start, pmc_stop, pmc_reset, perf_event_open, PAPI Interface	perf_event_open improvements

- **System calls not listed above are offloaded to Linux**
- **POSIX compliance: almost the entire LTP test suite passes (on x86)! (2013 version: 100%, 2015: 99%)**

Proxy Process and System Call Offloading

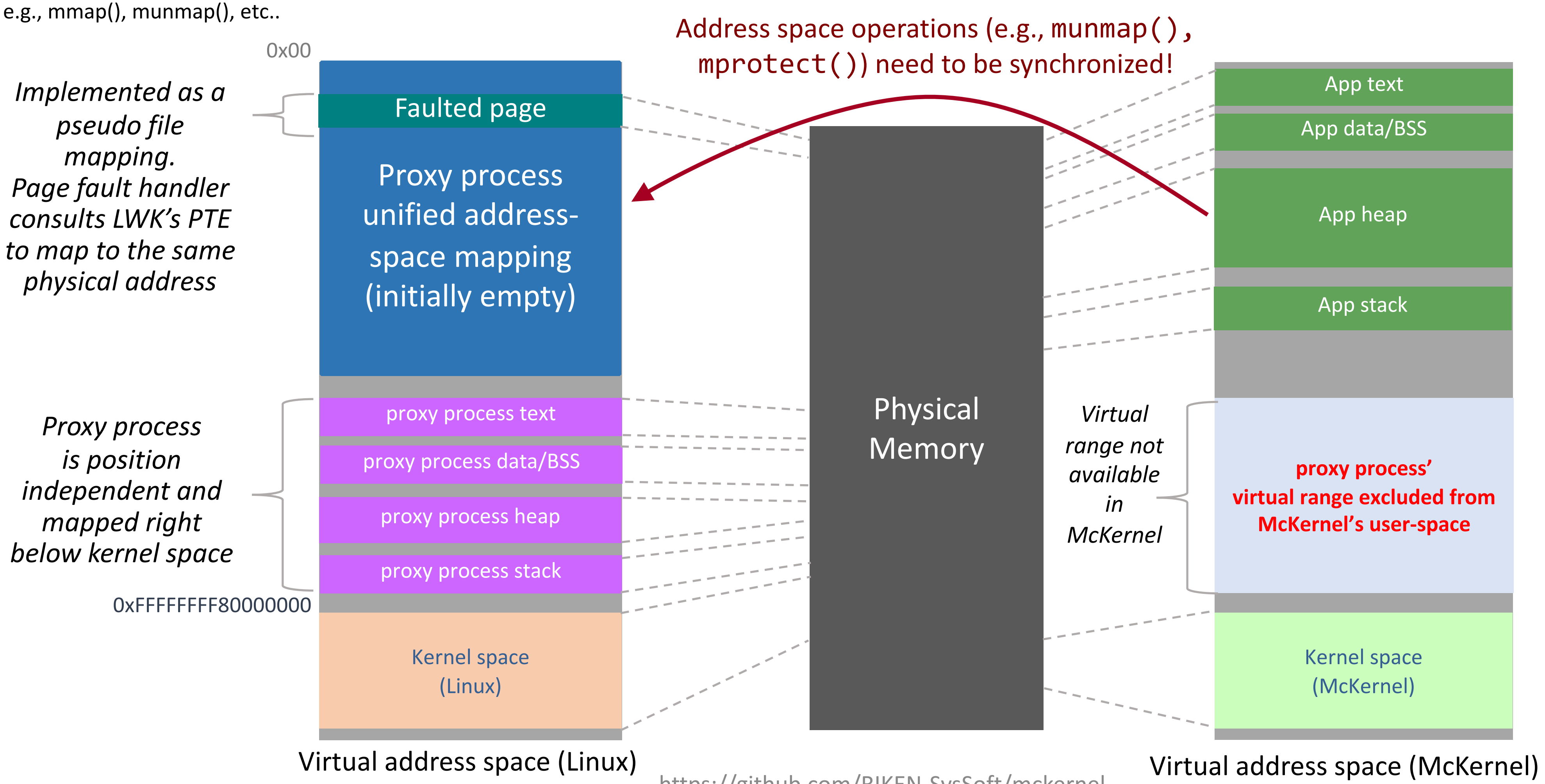
- For each application process a “proxy-process” resides on Linux
- Proxy process:
- Provides execution context on behalf of the application so that offloaded calls can be directly invoked in Linux
- Enables Linux to maintain certain state information that would have to be otherwise kept track of in the LWK
- (e.g., file descriptor table is maintained by Linux)



https://github.com/RIKEN-SysSoft/mckernel

Unified Address Space between Linux and LWK

- **Issue: how to handle memory addresses in system call arguments?**
  - Consider the target buffer of a read() system call
- **There is a need for the proxy process to access the application's memory (running on McKernel)**
- **Unified address space ensures proxy process can transparently see applications memory contents and reflect virtual memory operations**
  - e.g., mmap(), munmap(), etc..



https://github.com/RIKEN-SysSoft/mckernel

Evaluation

Oakforest PACS Overview

- **8k Intel Xeon Phi (Knights Landing) compute nodes**
  - Intel OmniPath v1 interconnect
  - Peak performance: ~25 PF

- **Intel Xeon Phi CPU 7250 model:**
  - 68 CPU cores @ 1.40GHz
  - 4 HW thread / core
    - 272 logical OS CPUs altogether
  - 64 CPU cores used for McKernel, 4 for Linux
  - 16 GB MCDRAM high-bandwidth memory
    - Hot-pluggable in BIOS
  - 96 GB DRAM
  - **Quadrant flat mode**

Software Environment

- **Linux:**
  - CentOS, Linux kernel 3.10.0-693.11.6
  - IFS-10.7-0
  - nohz_full on 256 cores
  - MCDRAM as movable_node (in flat Quadrant mode)

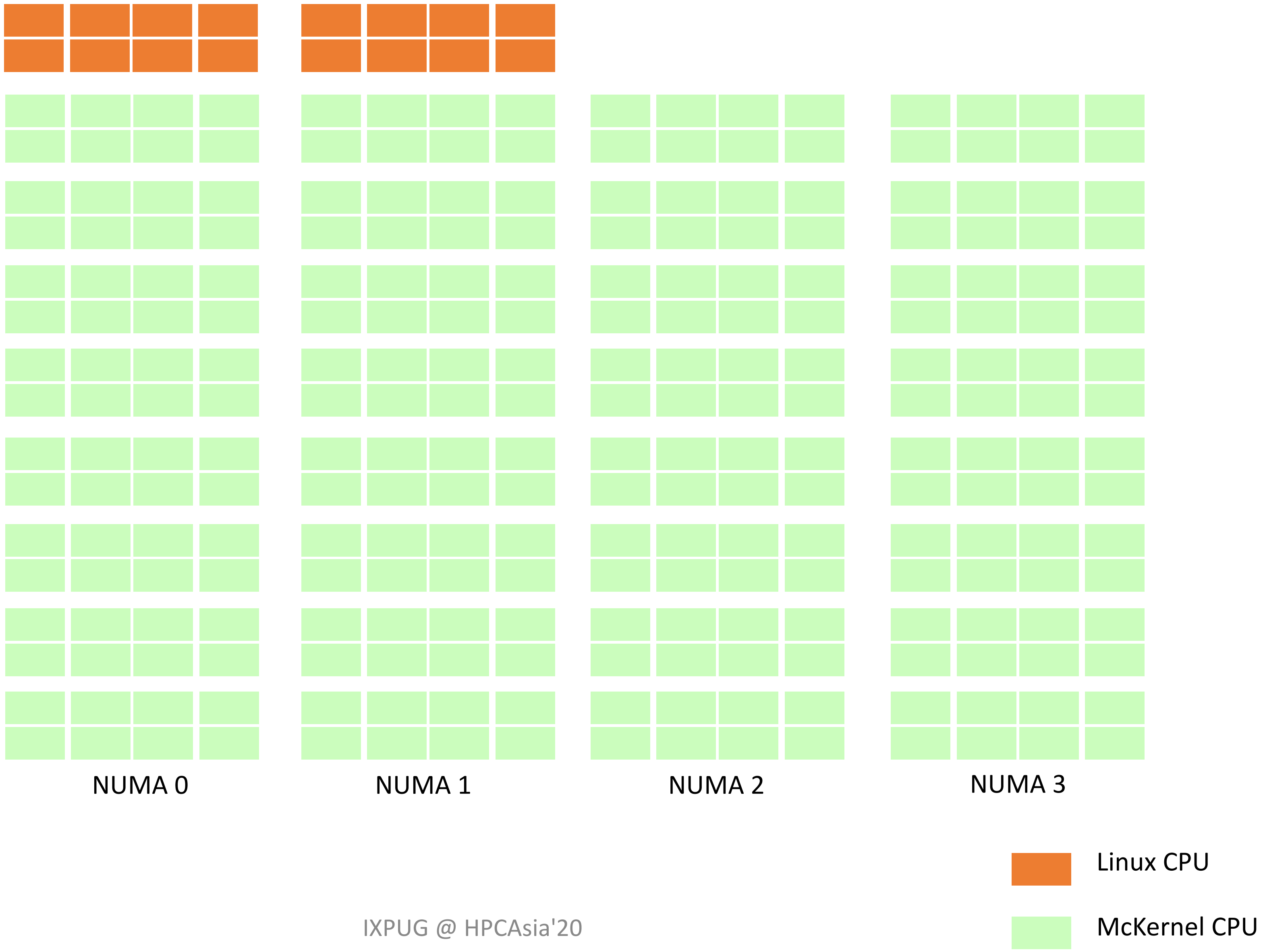
- **Linux+corespec:**
  - Linux + I_MPI_PIN_PROCESSOR_EXCLUDE_LIST=0-3,68-71,136-139,204-207
  - i.e., excludes OS CPU cores (same cores used as in McKernel, also set to nohz_full)

- **IHK/McKernel:**
  - IHK: 50a13c89
  - McKernel: 9f82c54b (HFI1 PicoDriver integrated)
  - + ANON mmap rewrite
  - + reboot script modifications (to boot from /tmp)

- **Fujitsu job submission system**

Oakforest PACS: Linux vs. McKernel CPUs

- **LWK runs on the majority of the chip**
- **A few CPU cores are reserved for Linux**
- **Mechanism to map inter-core communication to MPI process layout**



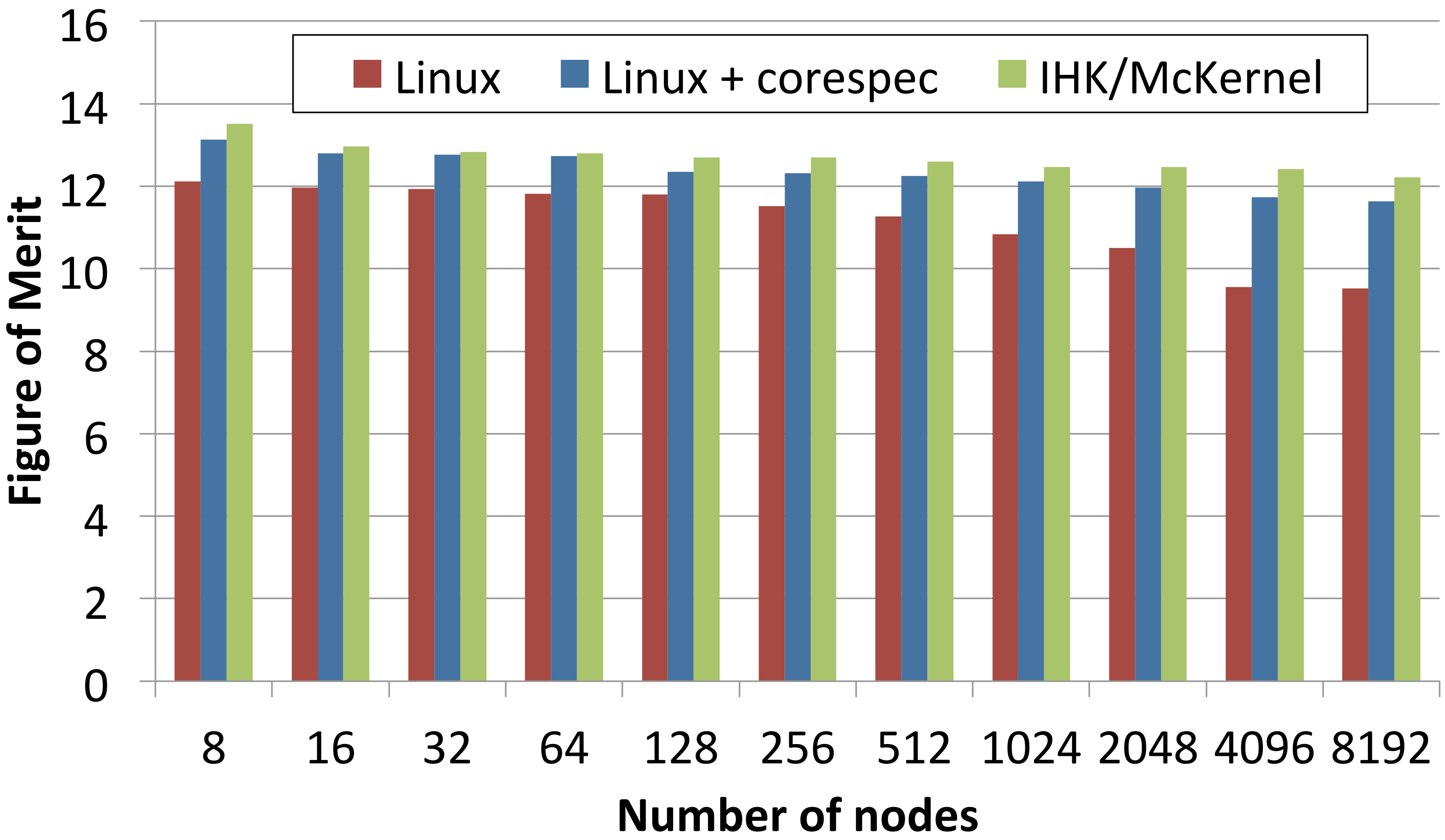
Mini-apps

- GeoFEM (Univ. of Tokyo)
- AMG2013 (CORAL)
- miniFE (CORAL)
- MILC (CORAL)
- Lulesh (CORAL)
- LAMMPS (CORAL)
- Nekbone (CORAL)
- HPCG (CORAL)
- GAMERA (Univ. of Tokyo)

Mini-apps: MPI ranks and OpenMP threads

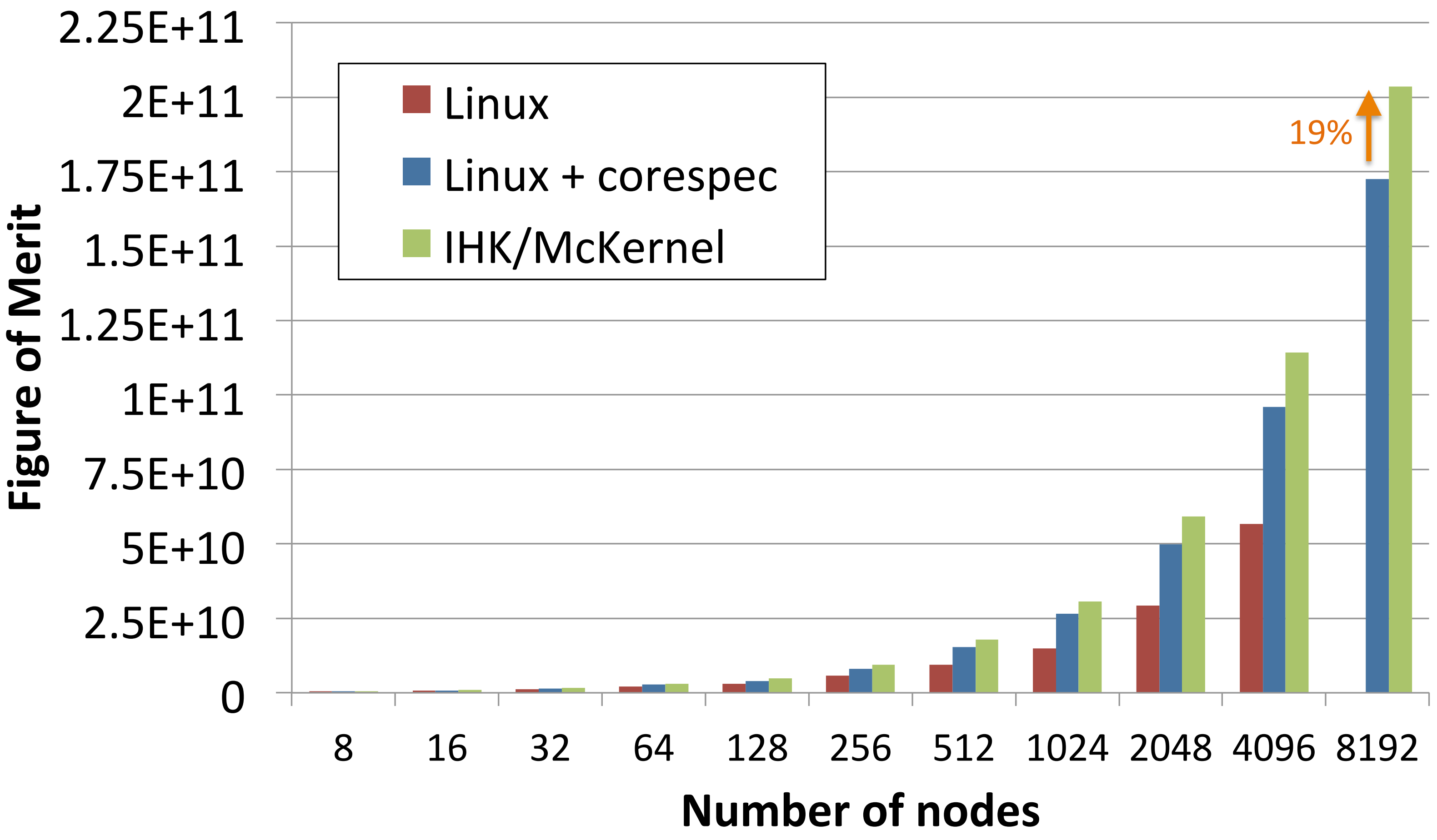
Property/ Mini-App	Ranks/ node	Threads/ rank	I_MPI_PIN_ORDER	KMP_AFFINITY	KMP_HW_SUBSET
GeoFEM	16	8	compact	compact	2t
AMG2013	16	16	compact	compact	N/A
MiniFE	16	16	compact	compact	N/A
MILC	32	4	compact	compact	2t
Lulesh	8	16	compact	compact	2t
LAMMPS	32	4	compact	compact	2t
Nekbone	32	4	compact	compact	2t
HPCG	32	4	compact	compact	2t
GAMERA	8	8	compact	compact	8c,1t

GeoFEM – 16 ranks/node, 8 OMP threads/rank



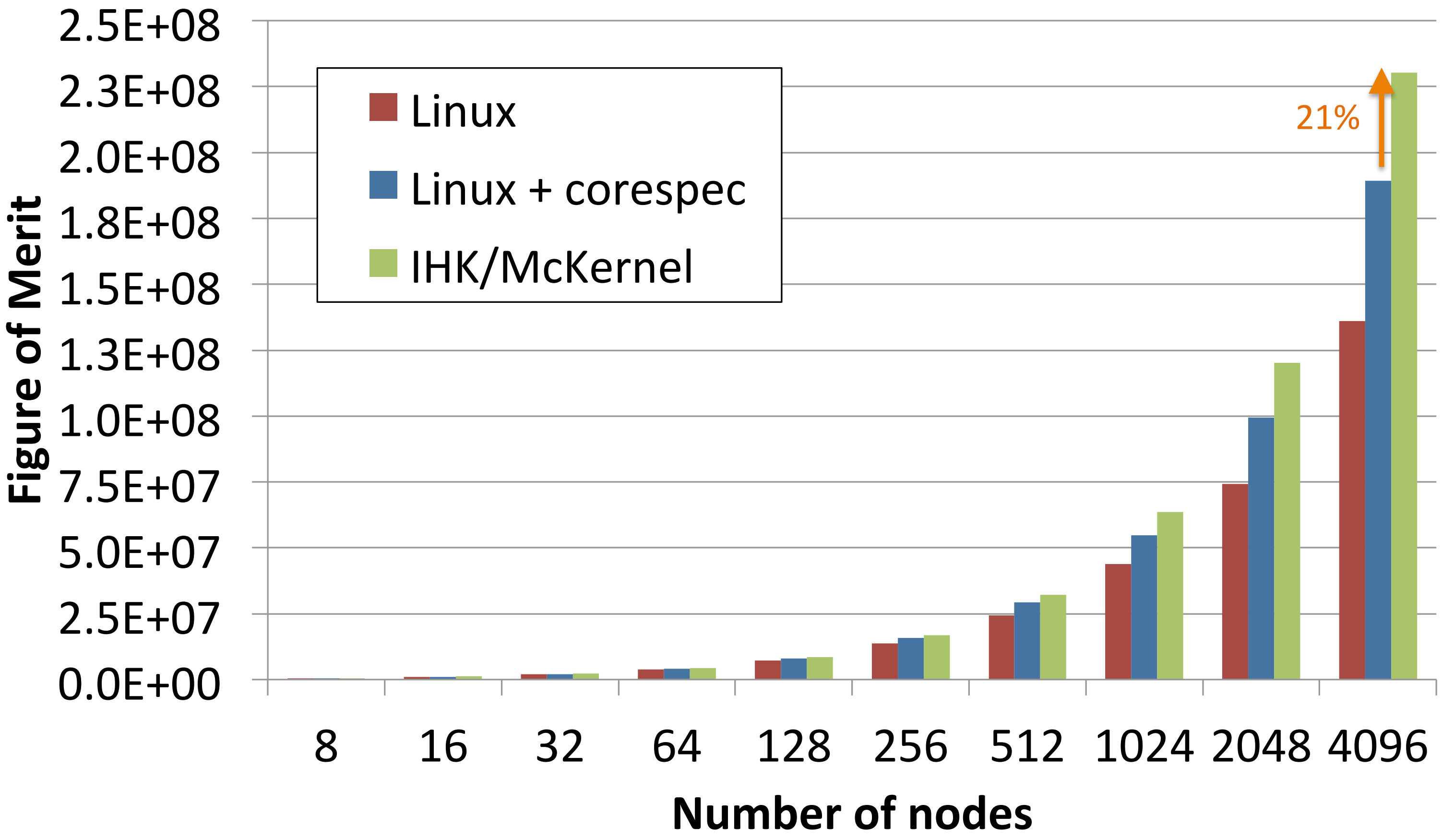
- **Weak scaled, up to 6% improvement**
- **Linux core specialization makes a big difference!**

AMG2013 – 16 ranks/node, 16 OMP threads/rank



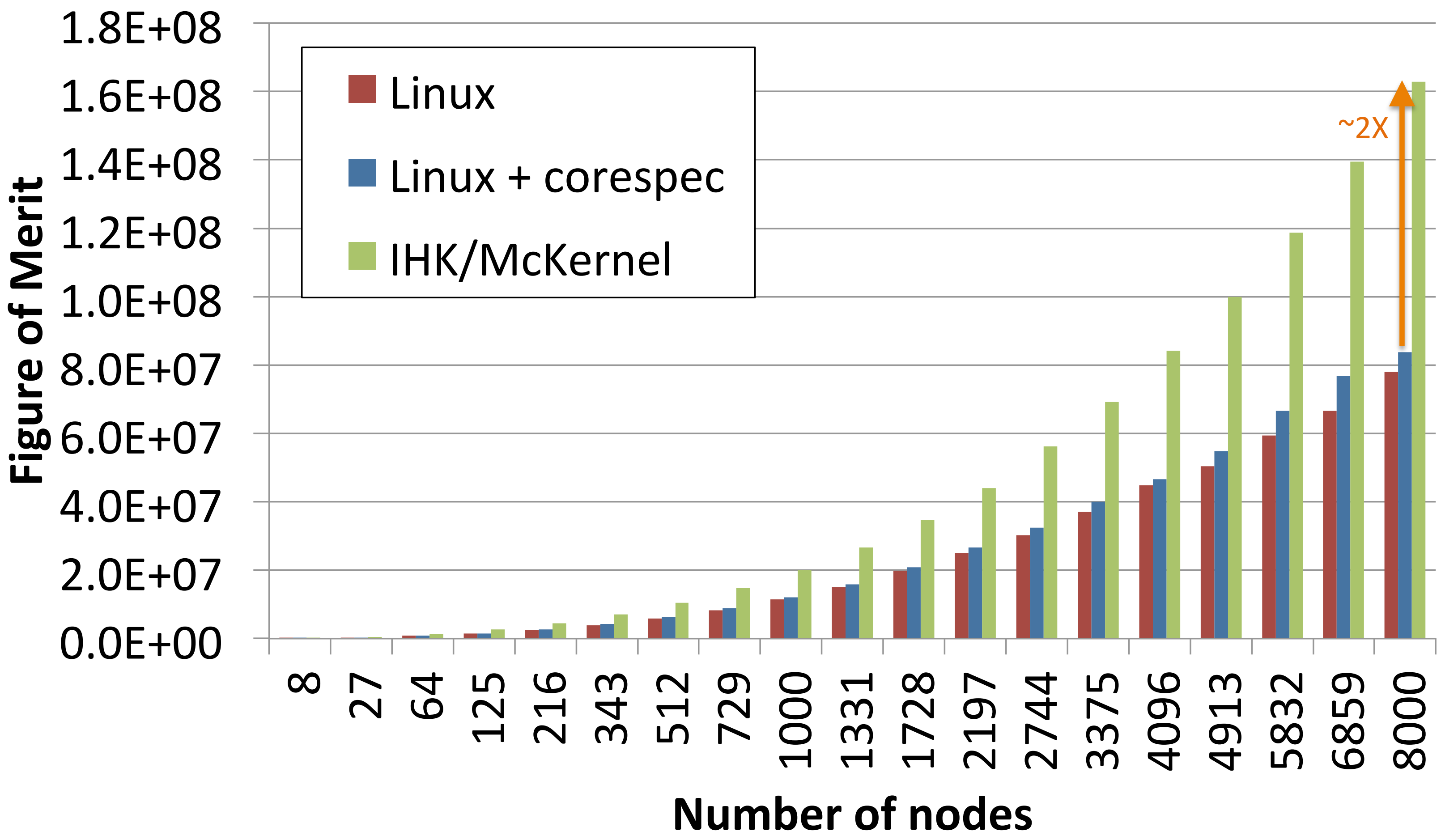
- **Weak scaled**
- **Linux (without core-spec) on 8192 nodes failed**

MILC – 32 ranks/node, 4 OMP threads/rank



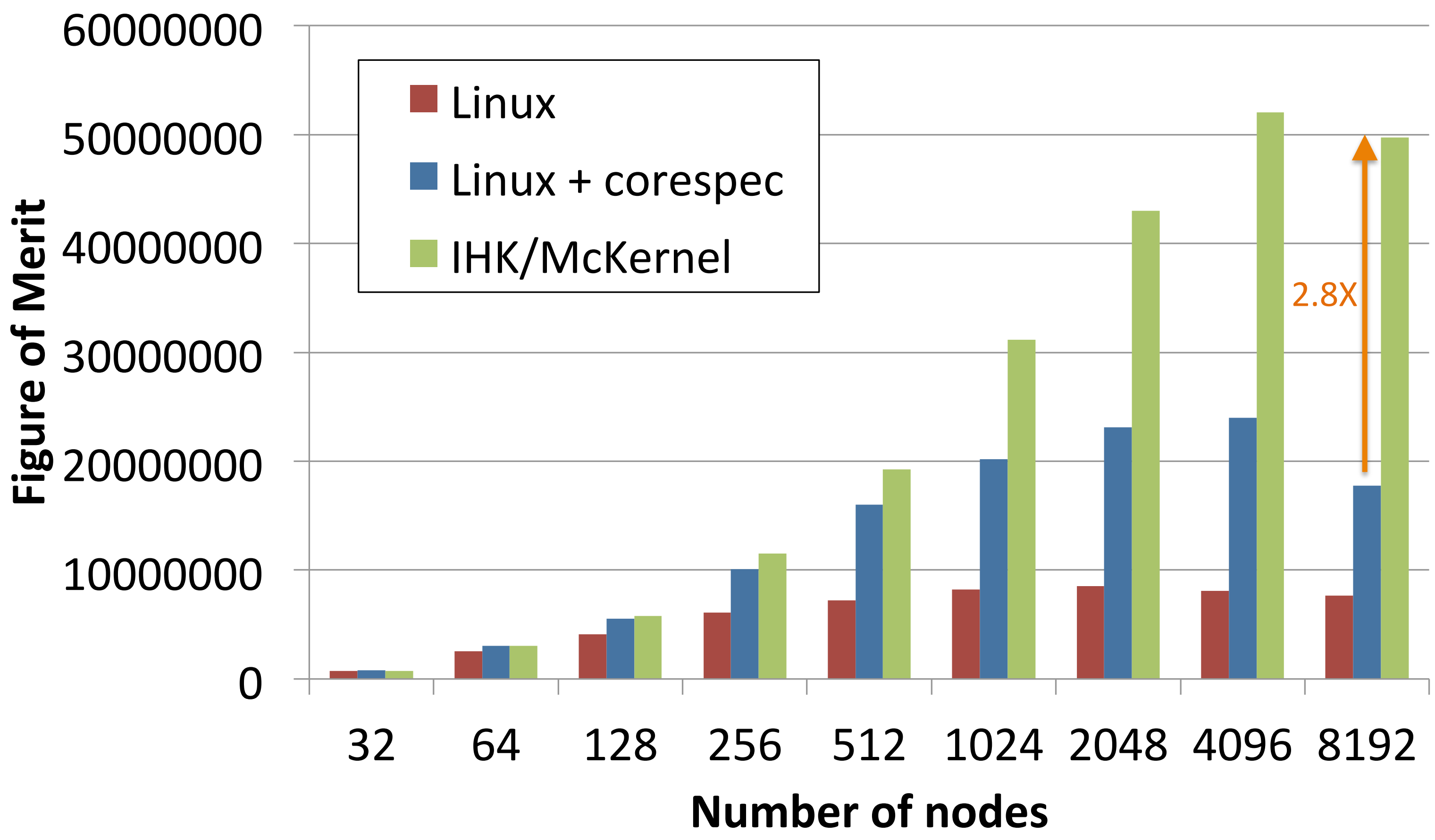
- **Weak scaled**

Lulesh – 8 ranks/node, 16 OMP threads/rank



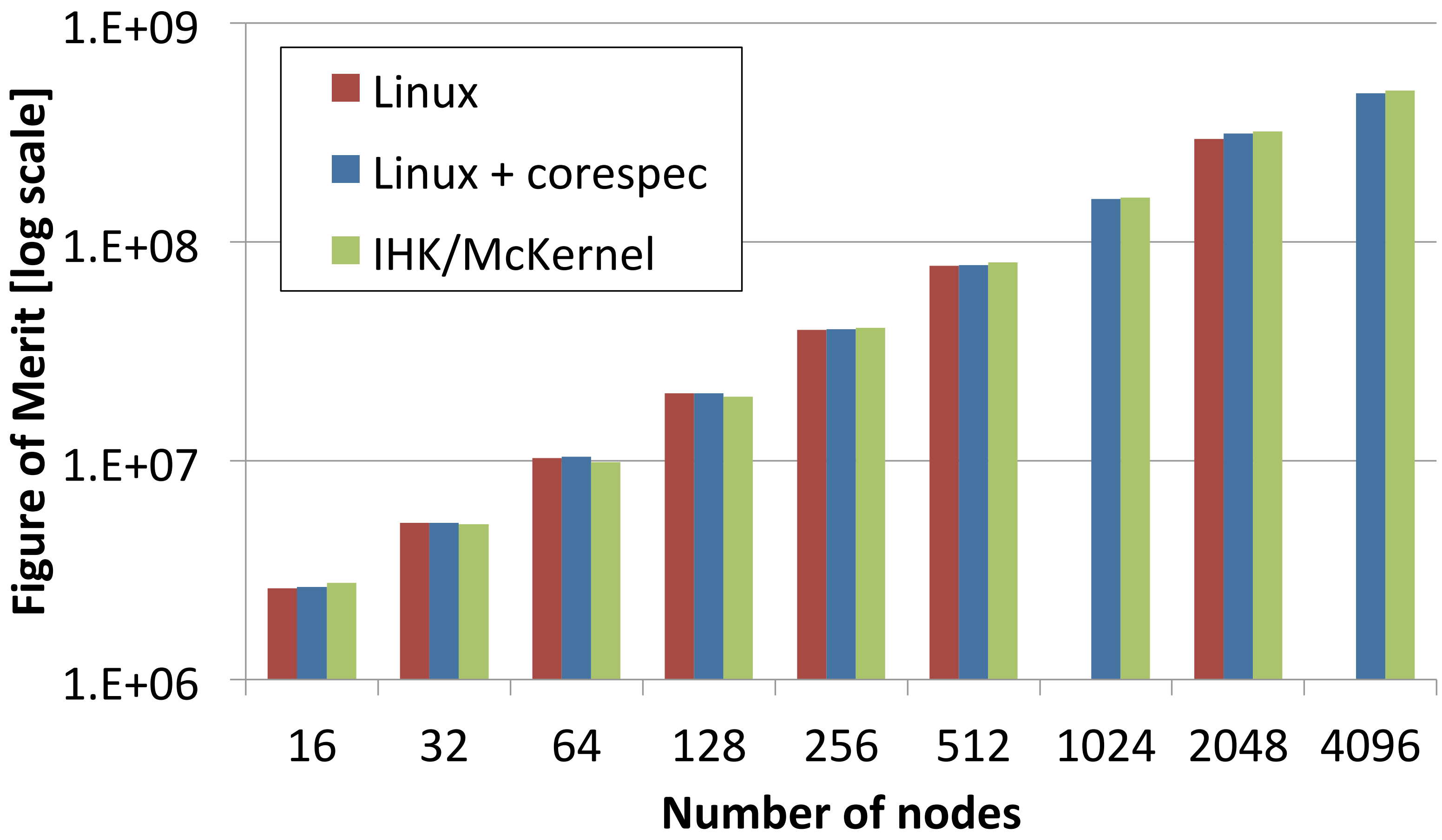
- **Weak scaled**
- **Requires n^3 number of ranks**

miniFE – 16 ranks/node, 16 OMP threads/rank



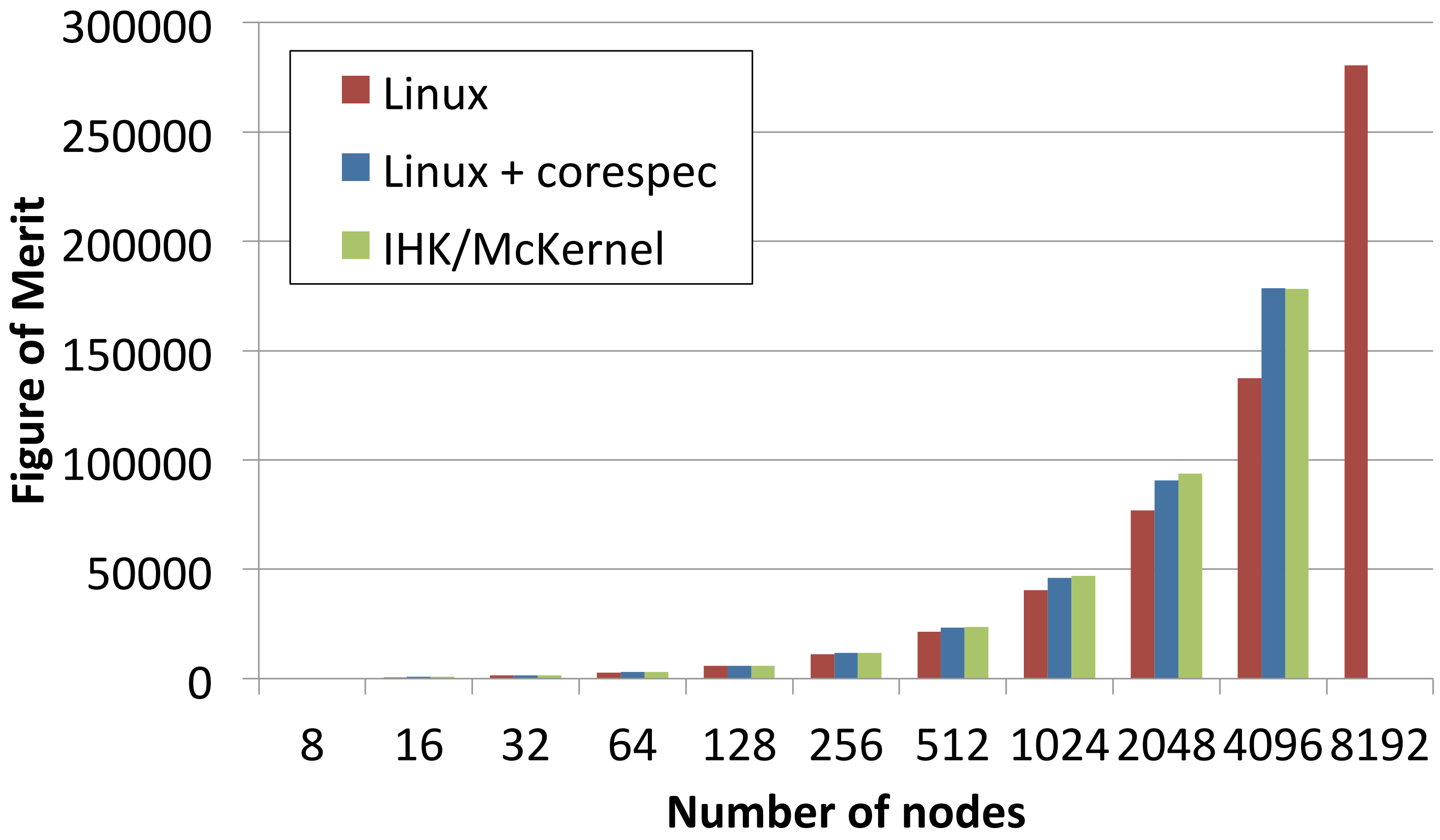
- **Large data set 1200x1200x1200**
- **Strong scaled**

Nekbone – 32 ranks/node, 4 OMP threads/rank



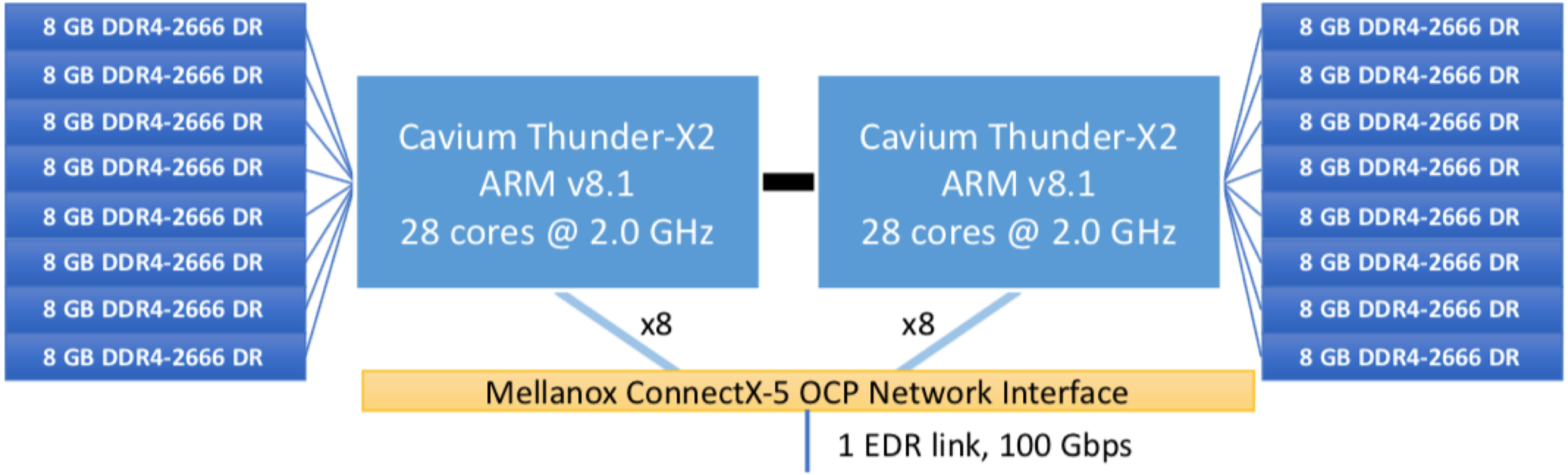
- **Weak scaled**
- **Linux failed on 1k and 4k nodes…**

HPCG – 32 ranks/node, 4 OMP threads/rank



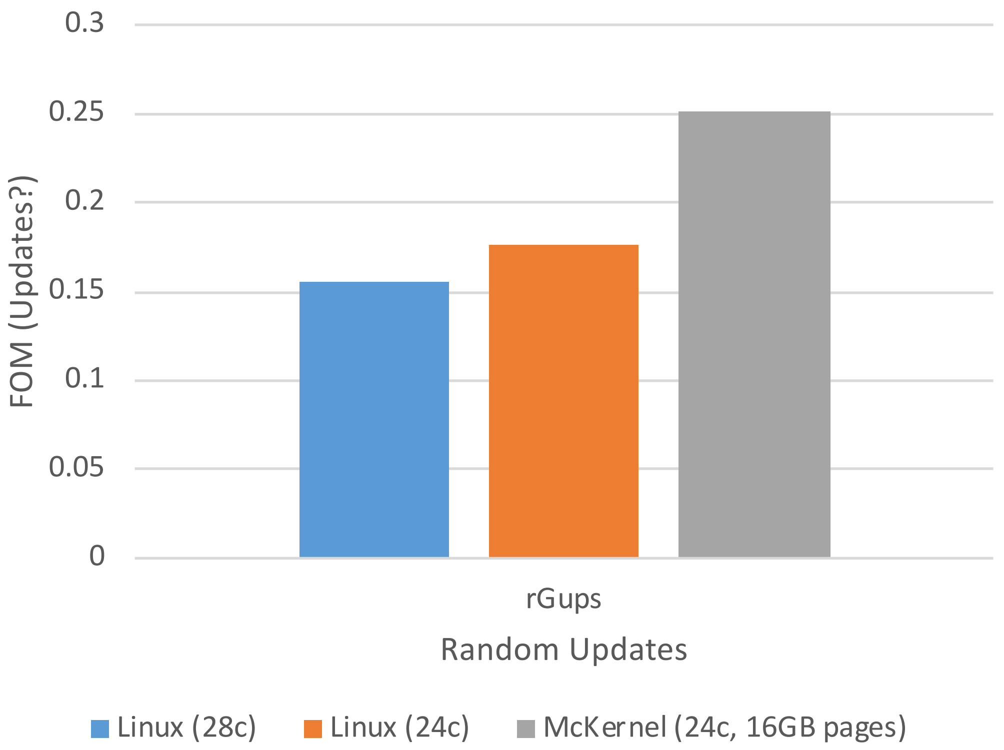
- **Weak scaled**
- **MPI_Init() timed out on 8k nodes run**

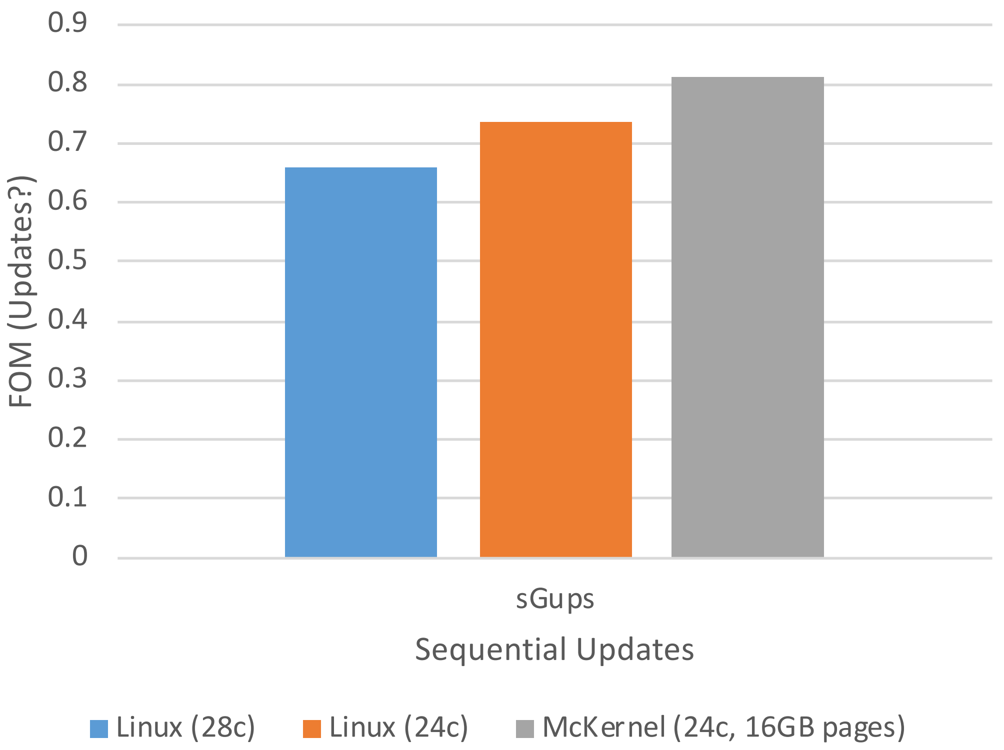
ThunderX2 Overview



- **8 DDR channels**
  - Up to ~250 GB/s memory bandwidth
- **Two sockets, 28 CPU cores / socket**
  - Can do 1-, 2- and 4-way SMT
- **Multi-rail IB**
  - Up to 13GB/s unidirectional bandwidth
- **McKernel now runs on ARM64**

Single node: GUPS (random access benchmark) ~47% improvement





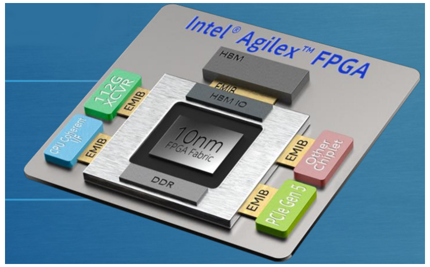
- **Linux:**
  - $ OMP_NUM_THREADS=28 OMP_PROC_BIND=close OMP_PLACES=cores numactl -C 0-27 -m 0 ./gups-atse-gcc 32 32768 8192
  - $ OMP_NUM_THREADS=24 OMP_PROC_BIND=close OMP_PLACES=cores numactl -C 4-27 -m 0 ./gups-atse-gcc 32 32768 8192

  McKernel (using 16GB pages on heap):
  - $ OMP_NUM_THREADS=24 mcexec --extend-heap-by=16G numactl -C 0-24 -m 0 ./gups-atse-gcc 32 32768 8192

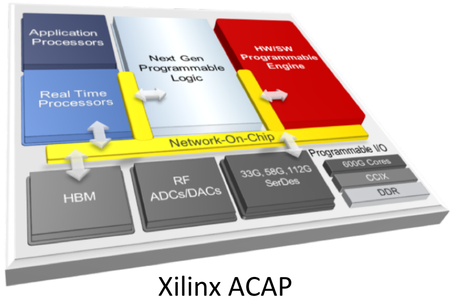
Future Directions

System Software in the Era of Heterogeneous Processing

- **Computer manufacturing technologies are approaching their physical limits (~5nm transistors)**
  - Moore's Law driven predictable performance increase is coming close to its end
- **More efficient architectures and specialization in the form of heterogeneous processing elements (PE)**
  - GPUs, FPGAs, a lot of upcoming AI specific chips
- **Operating systems (OS) traditionally provide:**
  - Mechanisms to manage (e.g., to allocate or multiplex) hardware resources
  - Isolation and secure access to devices
- **Up till recently heterogeneous PEs are/were:**
  - Special purpose devices are mainly single-user access
  - Slow data connection to host (e.g., PCIe) with dedicated physical memory
    - Separate address spaces, non cache-coherent access between host and accelerator
- **However, trends are changing:**
  - Incorporation of these devices into the memory system of the CPU
    - Cache-coherent interconnect standards for accelerators
      - E.g.: CCIX, GenZ, OpenCAPI, Intel CXL
    - Tighter integration of various processing elements in recent hardware platforms
      - E.g.: Intel's AgileX FPGA, Xilinx' Adaptive Compute Acceleration Platform (ACAP), Intel's Xe(?)
  - Eventually opens up the way for PEs to be treated as first-class citizens in the OS



Intel AgileX



Xilinx ACAP

System Software in the Era of Heterogeneous Processing

- **It will become difficult to manage these devices efficiently**
  - Especially by explicit user level management
- **New OS approaches for scheduling computing elements and managing multiple memory resources will be needed**
- **Challenges and opportunities for the system software?**
  - Co-design SW interfaces with the hardware to establish standard boundaries
  - Portable communication interfaces
    - Message passing, notifications, interrupts
  - Task dispatching
    - What are the right execution model abstractions?
      - E.g., non-interruptible, run-to-completion tasks
  - Memory management
    - Multi-level, heterogeneous devices, unified address spaces
- **With new, low-latency interconnects (e.g., optical):**
  - Disaggregation of compute/memory resources becomes available
  - Dynamic reconfiguration based on application demands
  - How will these concepts work in HPC?

Summary

- **Lightweight kernels benefit HPC workloads**
- **Multi-kernel approach adds Linux compatibility to LWK scalability**
  - Runs the same Linux binary

- **Building a full OS is not easy**
- **Lots of corner cases, especially with POSIX compatibility**

- **With regards to Fugaku**
  - If we inspired Fujitsu for some of its Linux design decisions (e.g., memory management) that's already a win!

- **Looking for collaborators to extend these concepts over heterogeneous PEs**

Thank you for your attention!
Questions?