Multi-Kernel OSes for Extreme-Scale HPC

Rolf Riesen, Balazs Gerofi

17 November 2016

Welcome

The goals for this Birds of a Feather session:

- Give a brief overview of the state of the art
- Interact with the community to learn about the needs and wishes of HPC developers and designers
  - This is your chance to influence and contribute to these projects!

Please use the “ask a question” button on the SC’16 page for this BoF.

You can find it by clicking on the BoF title on the SC Program page.

You can also vote questions up or down.

Please participate!

Agenda

mOS

- **12:15 - 12:17** Welcome (Rolf Riesen)
- **12:17 - 12:23** Intro to multi-kernels (Robert W. Wisniewski)
- **12:23 - 12:29** McKernel (Balazs Gerofi)
- **12:29 - 12:35** FFMK (Carsten Weinhold)
- **12:35 - 12:41** Kitten/Hobbes (Kevin Pedretti)
- **12:41 - 12:47** mOS (Rolf Riesen)
- **12:47 - 13:15** Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

Introduction

- 12:15 - 12:17 Welcome (Rolf Riesen)
- **12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)**
- 12:23 - 12:29 McKernel (Balazs Gerofi)
- 12:29 - 12:35 FFMK (Carsten Weinhold)
- 12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)
- 12:41 - 12:47 mOS (Rolf Riesen)
- 12:47 - 13:15 Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

Introduction to multi-kernels

Dr. Robert W. Wisniewski
Chief Software Architect Extreme Scale Computing
Senior Principal Engineer, Intel

McKernel

- 12:15 - 12:17 Welcome (Rolf Riesen)
- 12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)
- **12:23 - 12:29 McKernel (Balazs Gerofi)**
- 12:29 - 12:35 FFMK (Carsten Weinhold)
- 12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)
- 12:41 - 12:47 mOS (Rolf Riesen)
- 12:47 - 13:15 Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

IHK/McKernel

Balazs Gerofi
RIKEN Advanced Institute for Computational Science, JAPAN

2016/Dec/17 Multi-Kernel OSes for Extreme-Scale HPC BoF @ SC'16

RIKEN

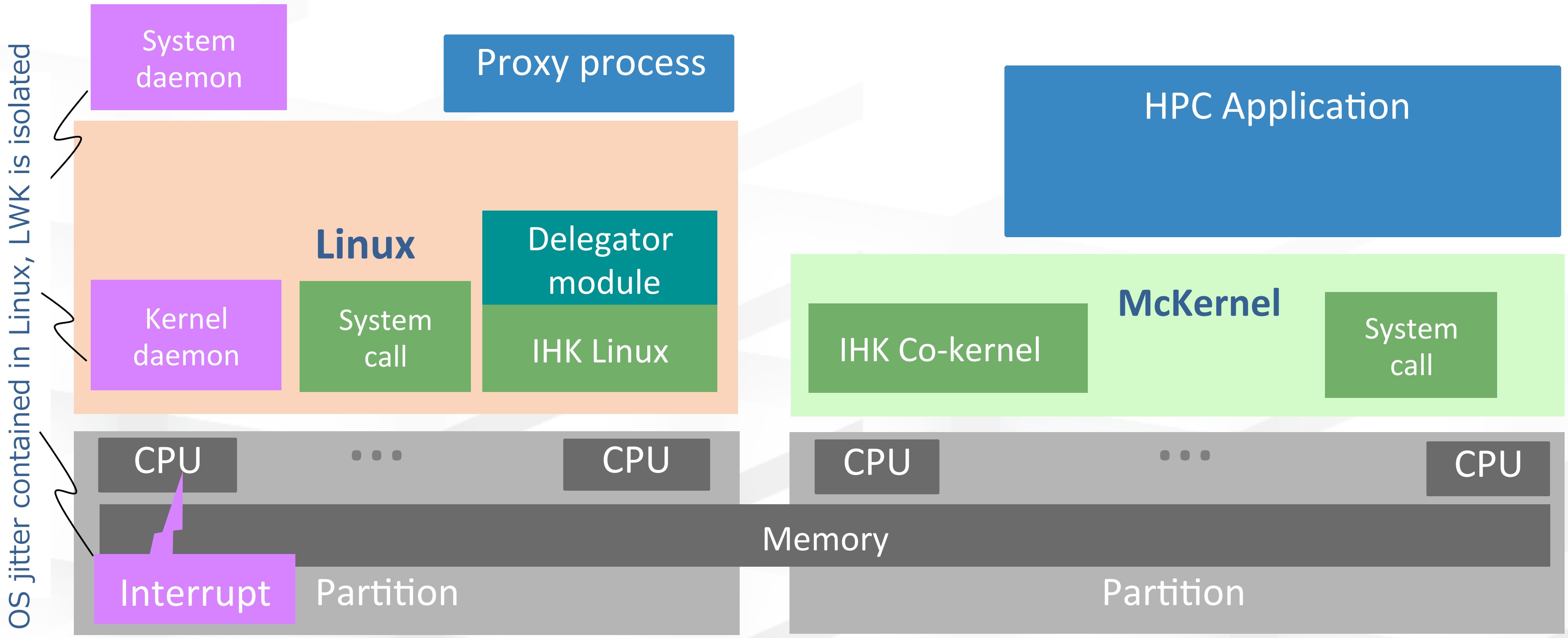
Motivation: what do we need?

- Lightweight kernel performance/scalability for large scale parallel apps
- Support for Linux APIs
- Full control over HW resources
- Ability to adapt to HW changes
- Performance isolation
- Dynamic reconfiguration
- Transparent access to Linux device drivers
- Avoid Linux modifications

RIKEN

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 (roughly process and memory management) and the rest are offloaded to Linux



McKernel and System Calls

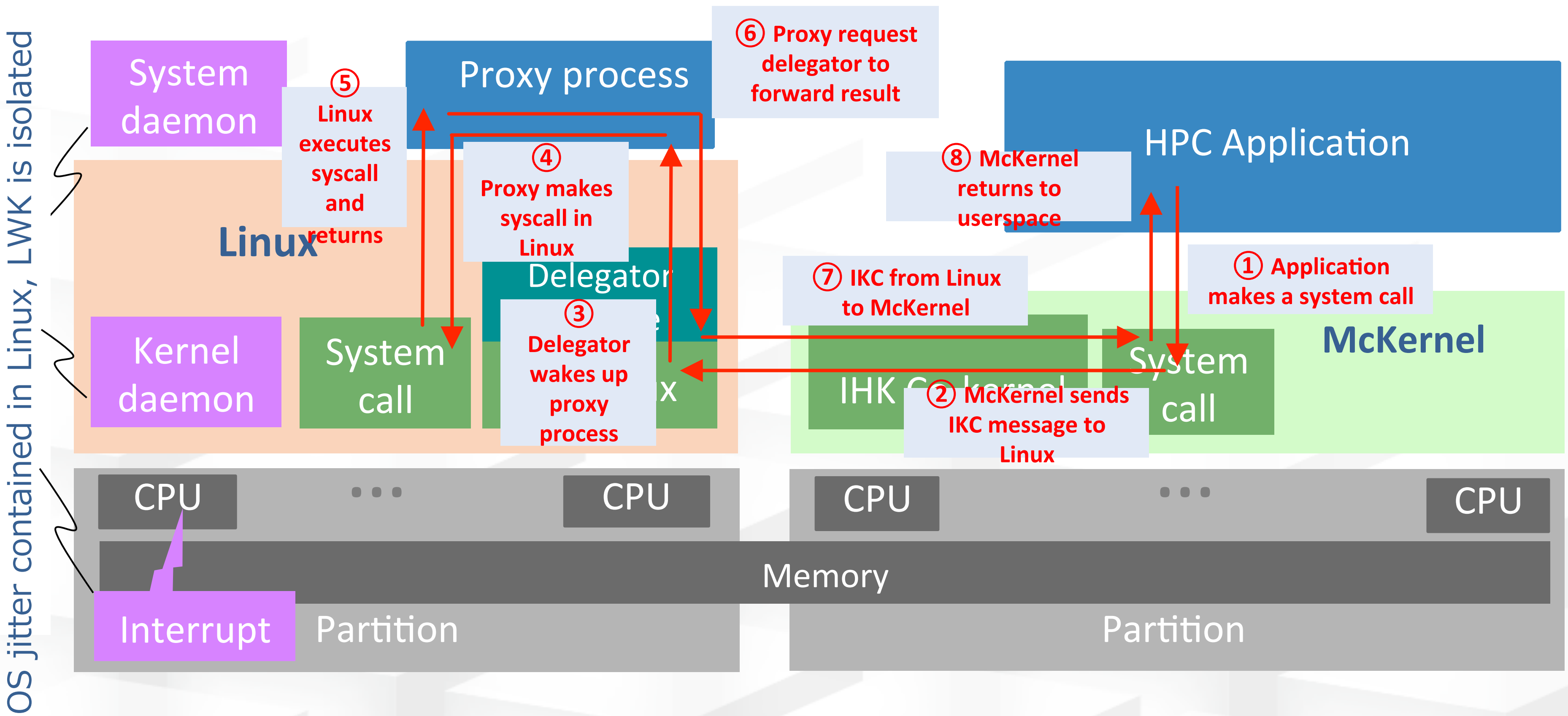
- McKernel is a lightweight (co-)kernel designed for HPC
- Linux ABI compatible
- McKernel only boots from IHK (no intention to boot it stand-alone)
- Noiseless, simple, with a minimal set of features implemented and the rest offloaded to Linux

	Implemented	Planned
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, ptrace	get_thread_area, getrlimit, rt_sigtimedwait, set_thread_area, setrlimit
Memory management	brk, gettid, madvise, mlock, mmap, mprotect, mremap, munlock, munmap, remap_file_pages, shmat, shmctl, shmdt, shmget, mbind, set_mempolicy, get_mempolicy	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 (in progress)

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

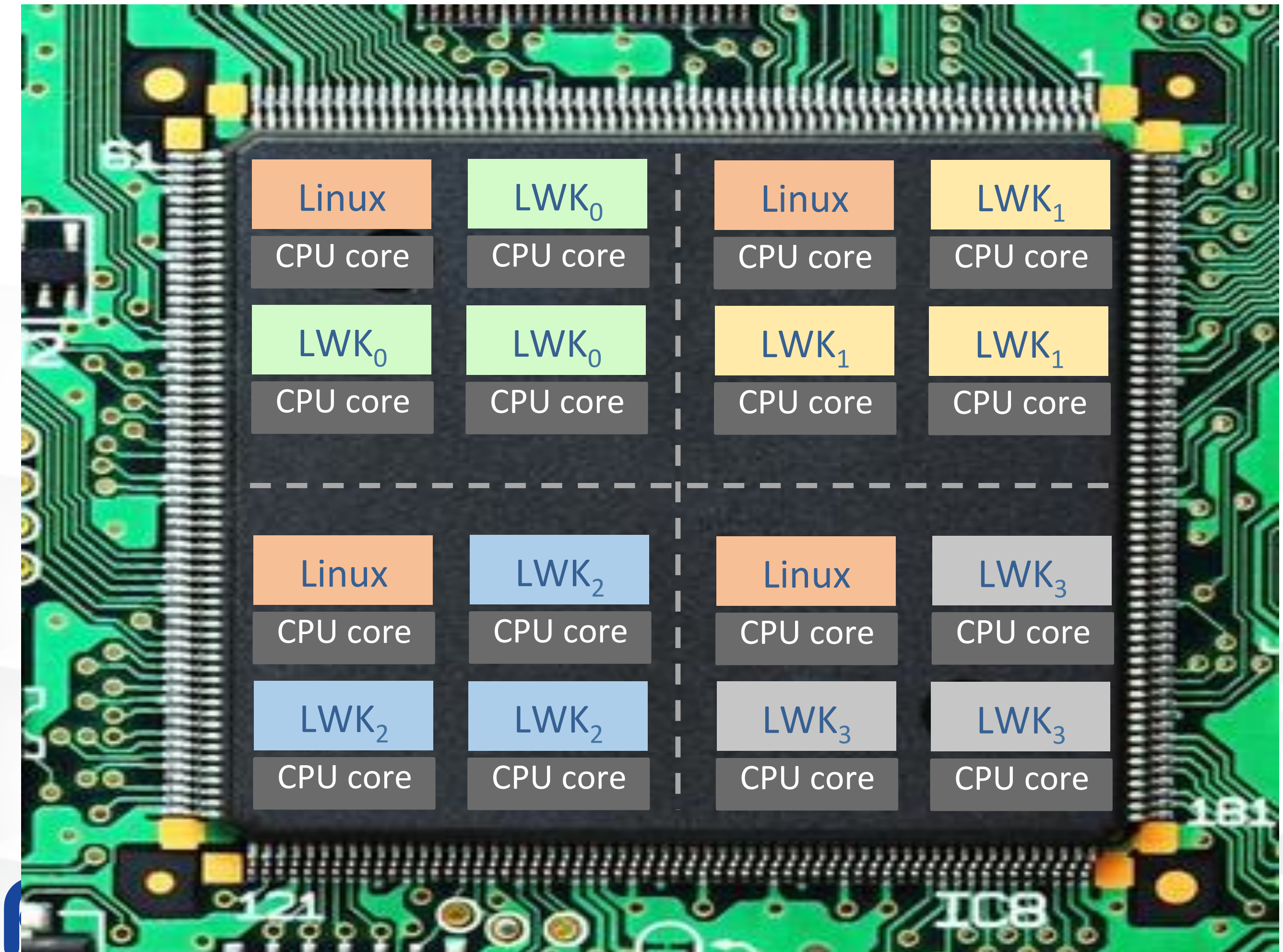
Proxy Process and System Call Offloading in IHK/McKernel

- **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)



Outlook to 1000s of CPU cores?

- **How will cache-coherence for synchronization perform on 1000s of CPU cores?**
- **Importance of topology awareness and exploitation of data locality for efficient synchronization and communication (K42 EuroSys'06, Multikernel SOSP'09, Ramos *et al.* HPDC'15, Kaestle *et al.* OSDI'16, etc.)**



- **Single monolithic kernel? OR**
- **Multiple, workload specialized, independent co-kernels?**
  - Laid out to suit HW topology, no implicit sharing of kernel data structures
- **Shared state is replicated and synchronized with explicit message passing?**
- **Dynamic repartitioning in response to workload requirements?**

FFMK

- 12:15 - 12:17 Welcome (Rolf Riesen)
- 12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)
- 12:23 - 12:29 McKernel (Balazs Gerofi)
- **12:29 - 12:35 FFMK (Carsten Weinhold)**
- 12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)
- 12:41 - 12:47 mOS (Rolf Riesen)
- 12:47 - 13:15 Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

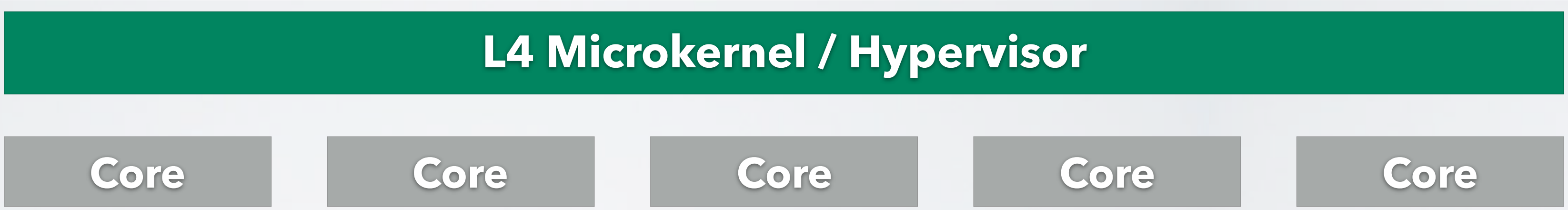
TECHNISCHE UNIVERSITÄT DRESDEN

FFMK: L4 MICROKERNEL + LINUX AS AN HPC OPERATING SYSTEM

ADAM LACKORZYNSKI, <u>CARSTEN WEINHOLD,</u> HERMANN HÄRTIG
TU DRESDEN, GERMANY

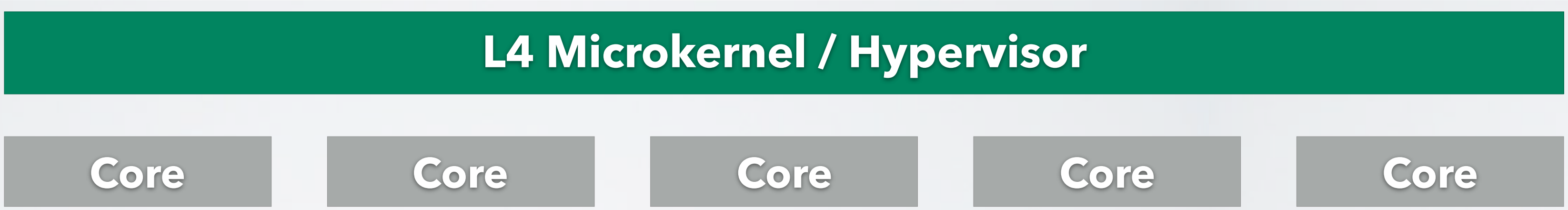
L4 + L⁴LINUX

- **L4 microkernel** controls the node



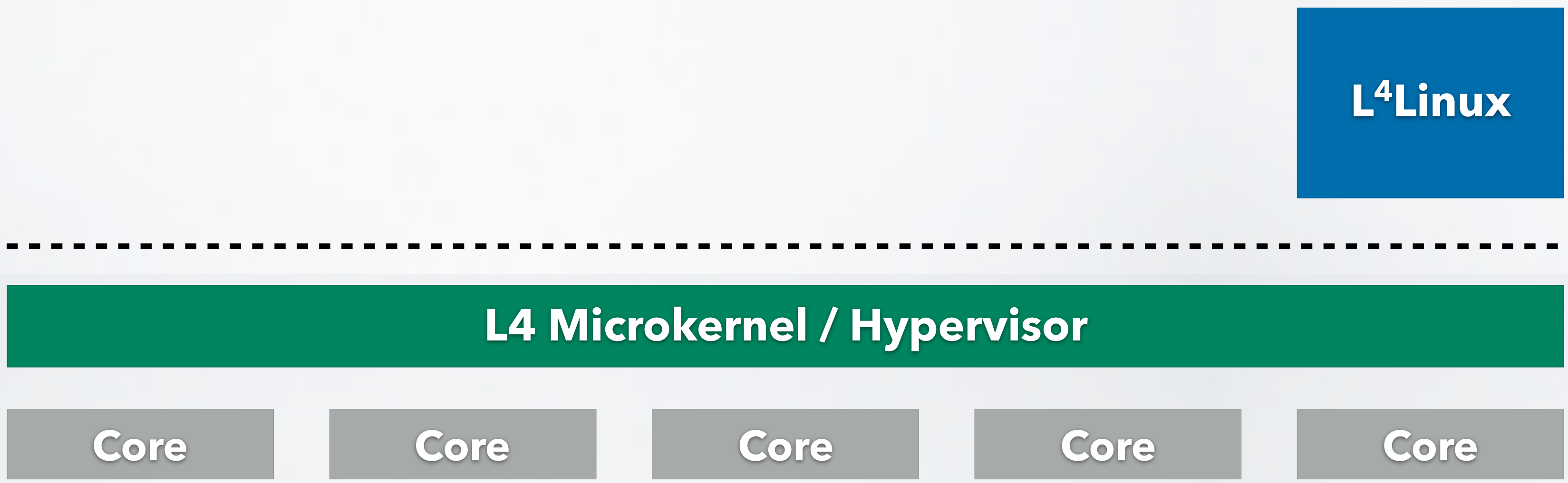
L4 + L^{4}LINUX

- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**



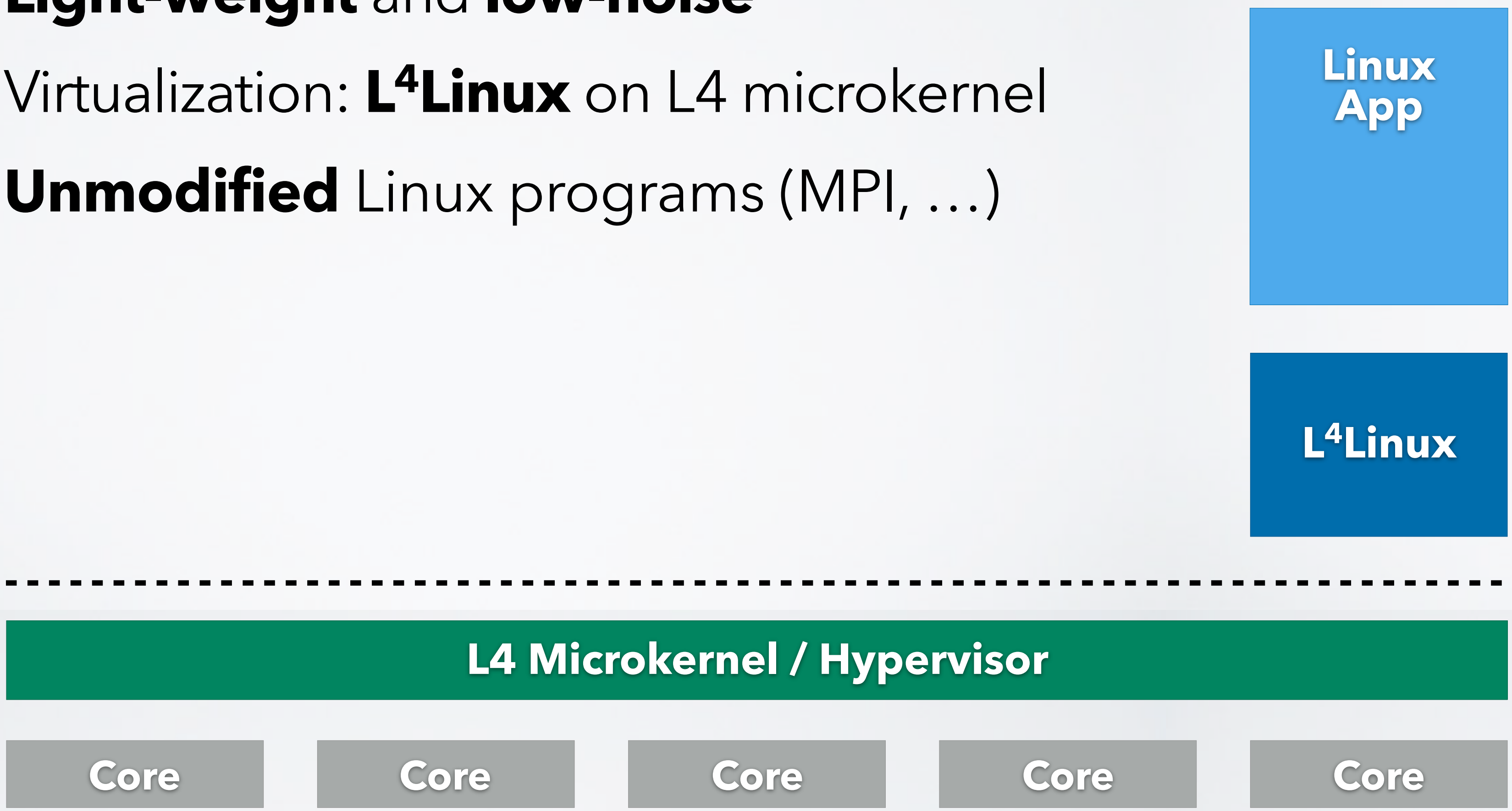
L4 + L⁴LINUX

- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**
- Virtualization: **L^{4}Linux** on L4 microkernel



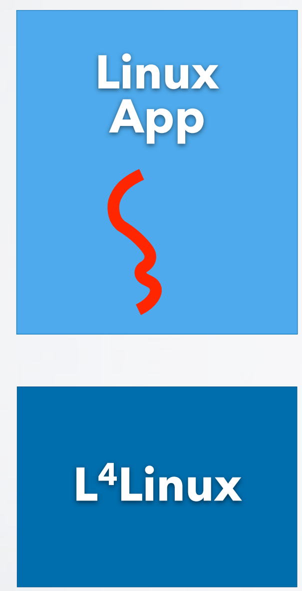
L4 + L⁴LINUX

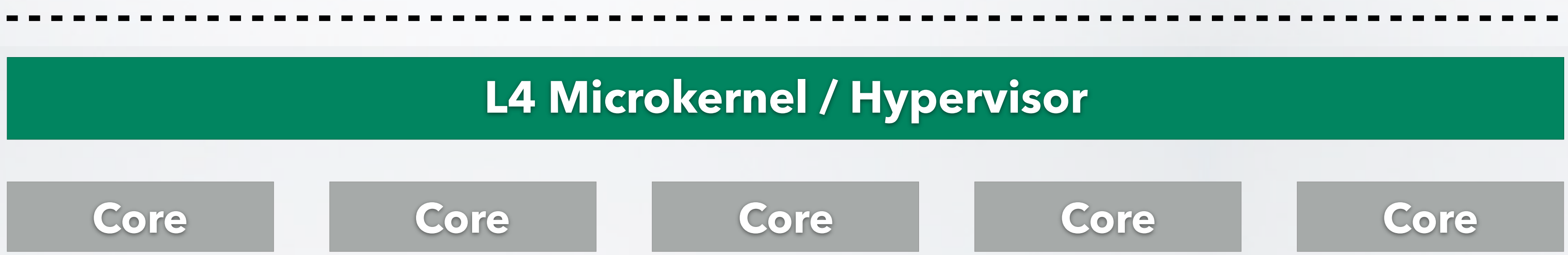
- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**
- Virtualization: **L⁴Linux** on L4 microkernel
- **Unmodified** Linux programs (MPI, …)



L4 + L^{4}LINUX

- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**
- Virtualization: **L^{4}Linux** on L4 microkernel
- **Unmodified** Linux programs (MPI, …)
- **Linux process** = **L4 task** + **L4 threads**

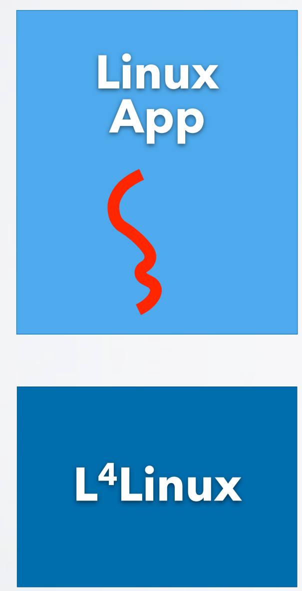


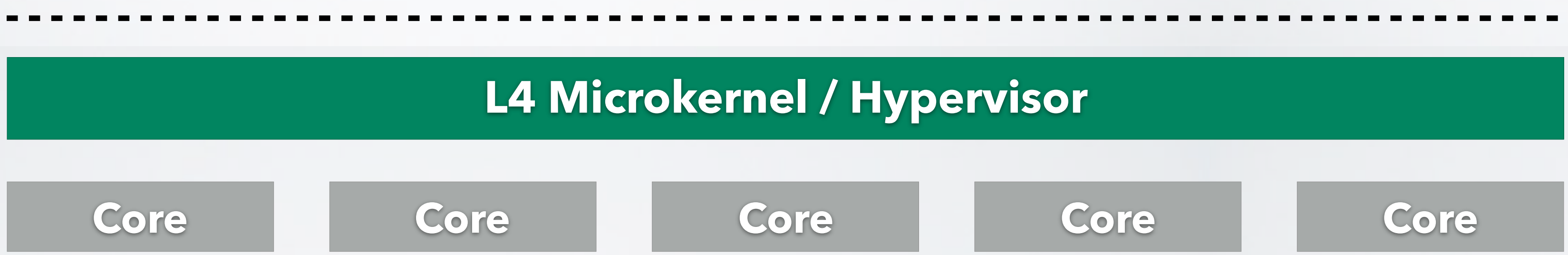


TECHNISCHE UNIVERSITÄT DRESDEN

L4 + L^{4}LINUX

- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**
- Virtualization: **L^{4}Linux** on L4 microkernel
- **Unmodified** Linux programs (MPI, …)
- **Linux process** = **L4 task** + **L4 threads**
- Linux syscalls / exceptions:
  generic forwarding to L^{4}Linux kernel

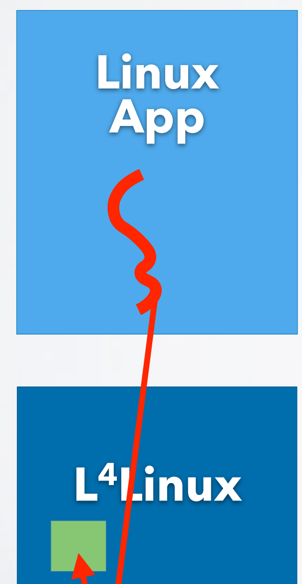


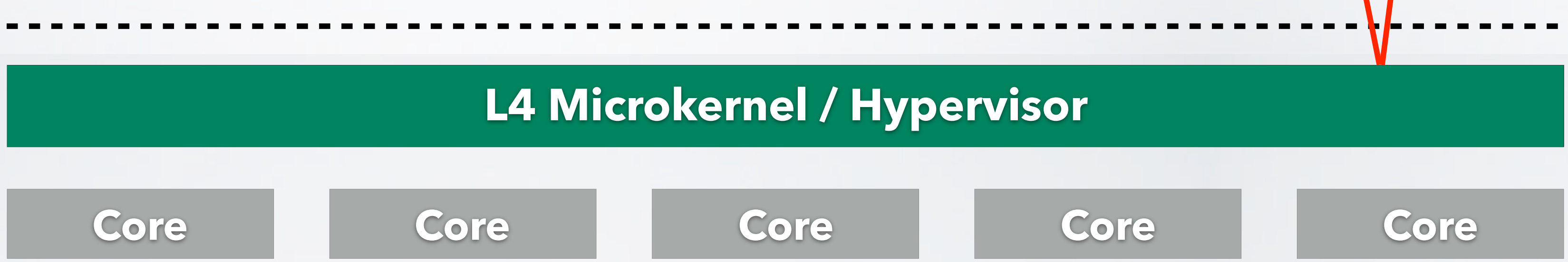


TECHNISCHE UNIVERSITÄT DRESDEN

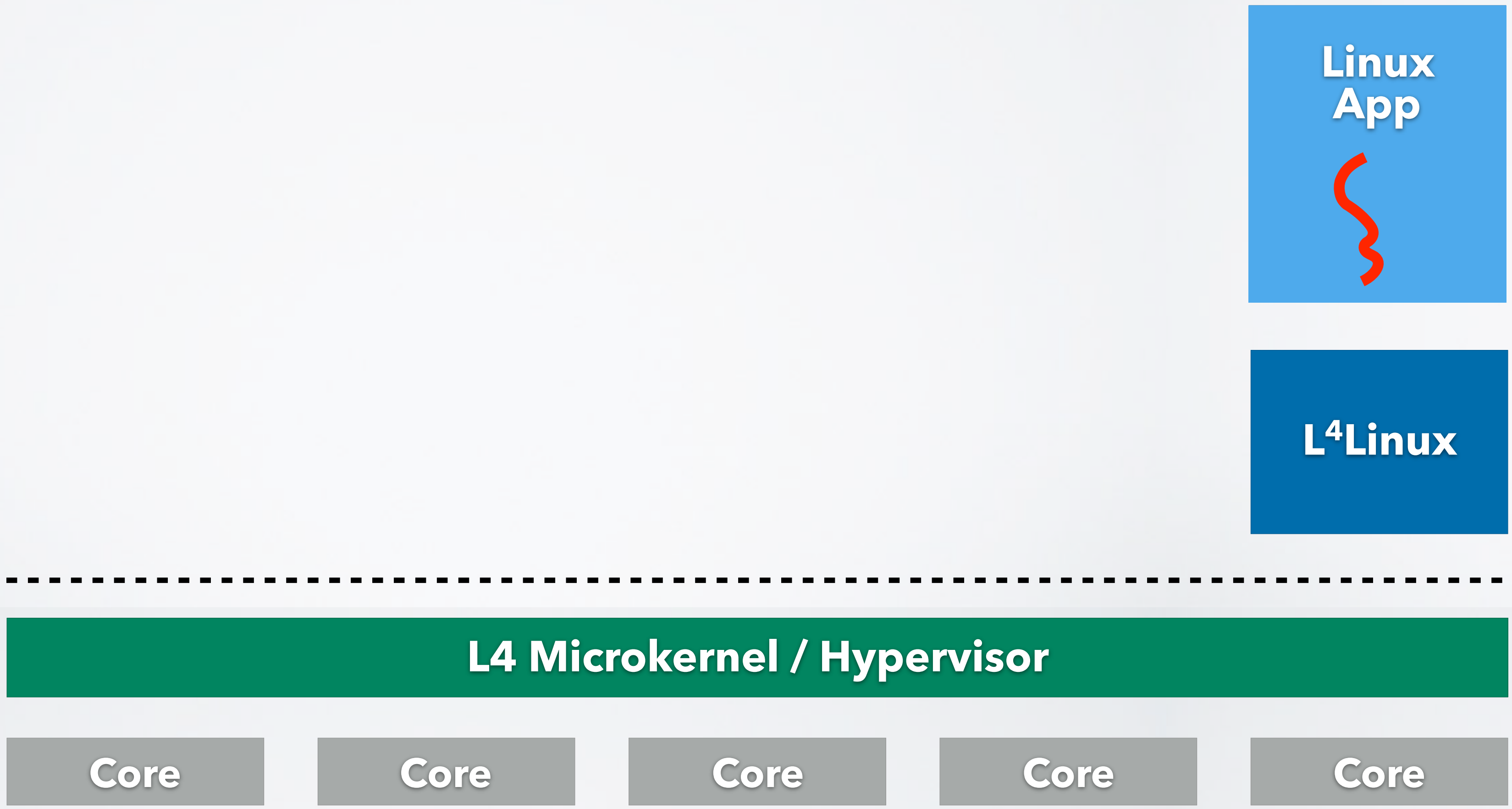
L4 + L⁴LINUX

- **L4 microkernel** controls the node
- **Light-weight** and **low-noise**
- Virtualization: **L⁴Linux** on L4 microkernel
- **Unmodified** Linux programs (MPI, ...)
- **Linux process** = **L4 task** + **L4 threads**
- Linux syscalls / exceptions: **generic forwarding** to L⁴Linux kernel



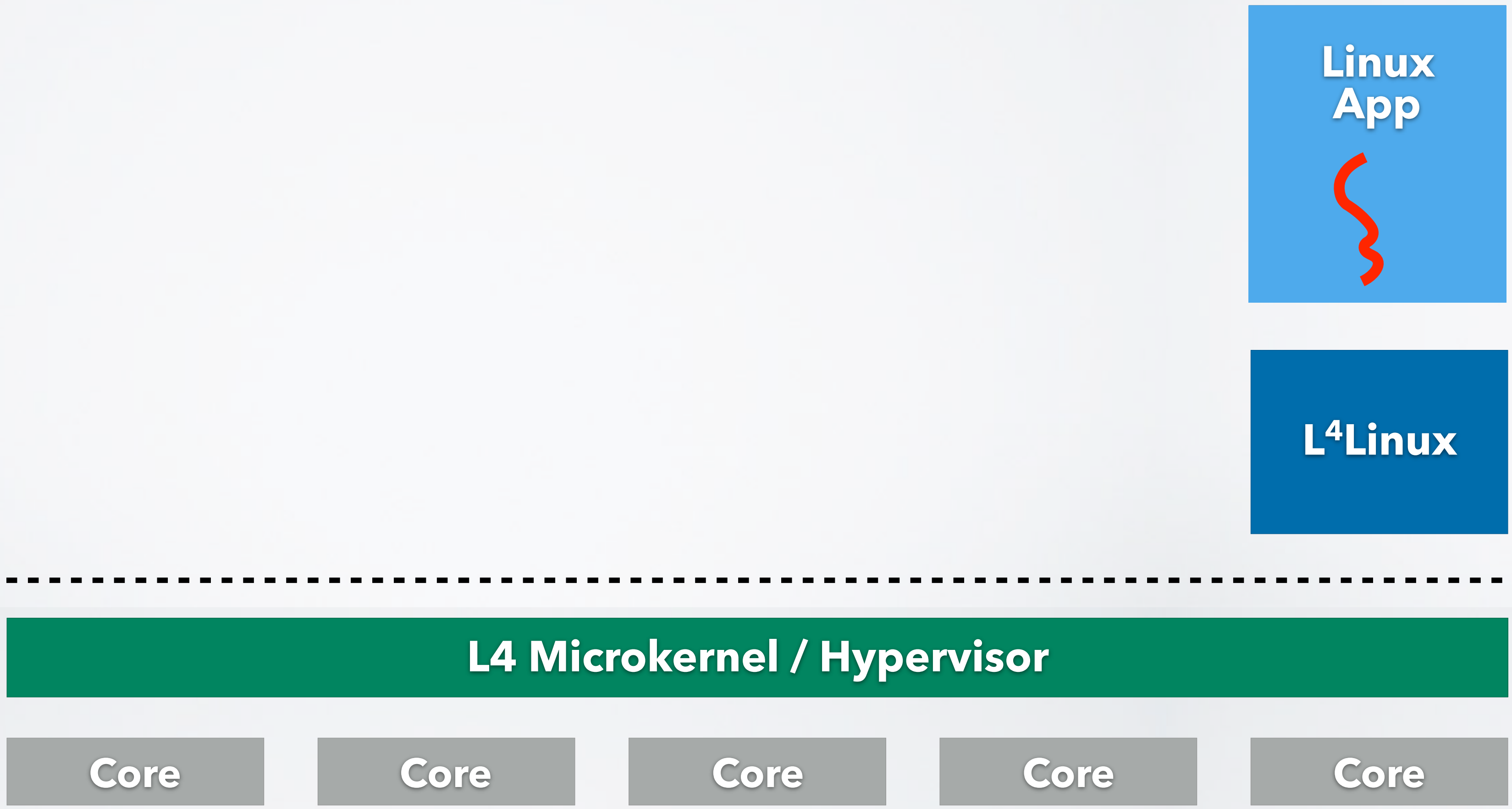


DECOUPLED EXECUTION



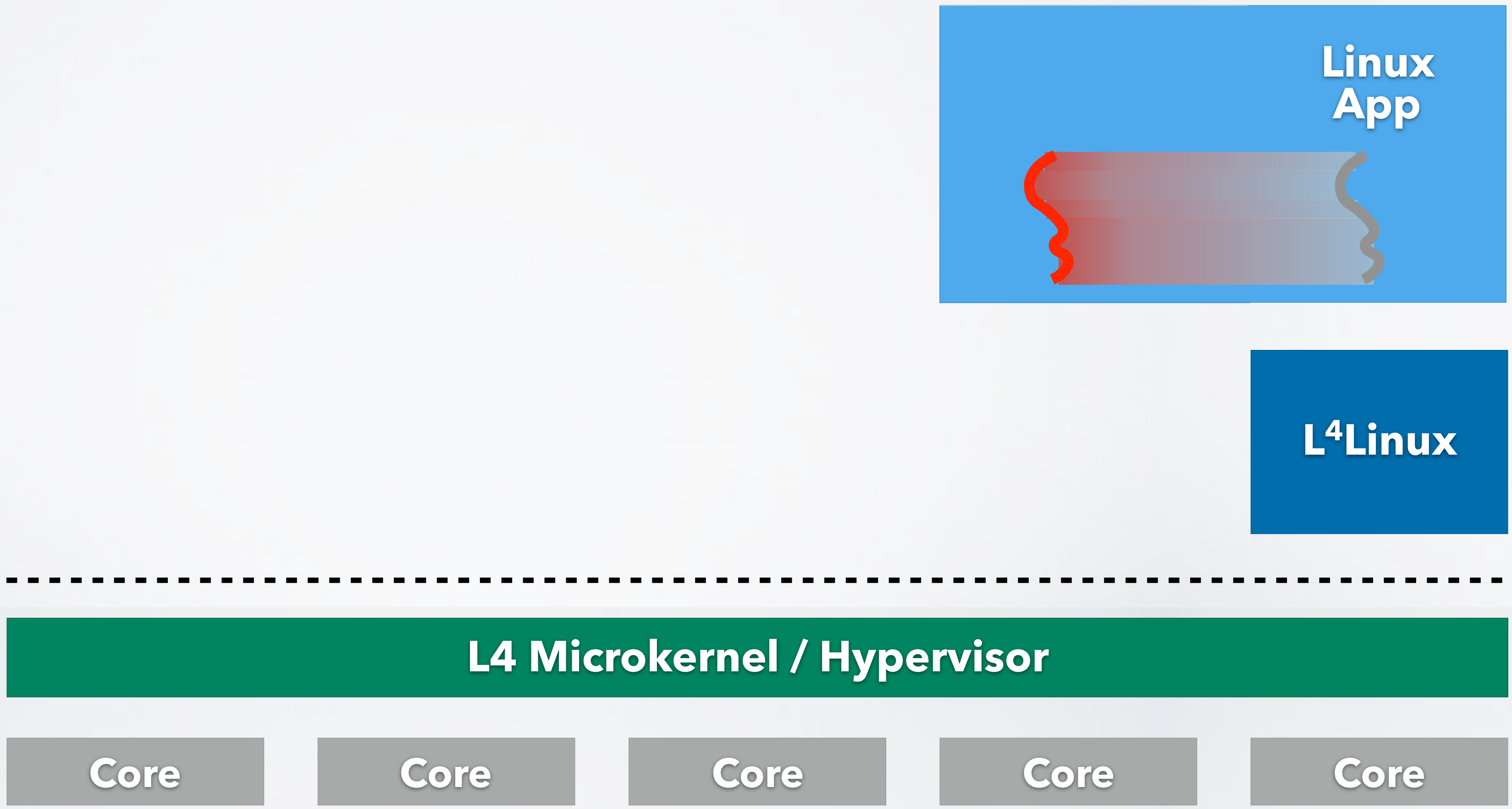
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core



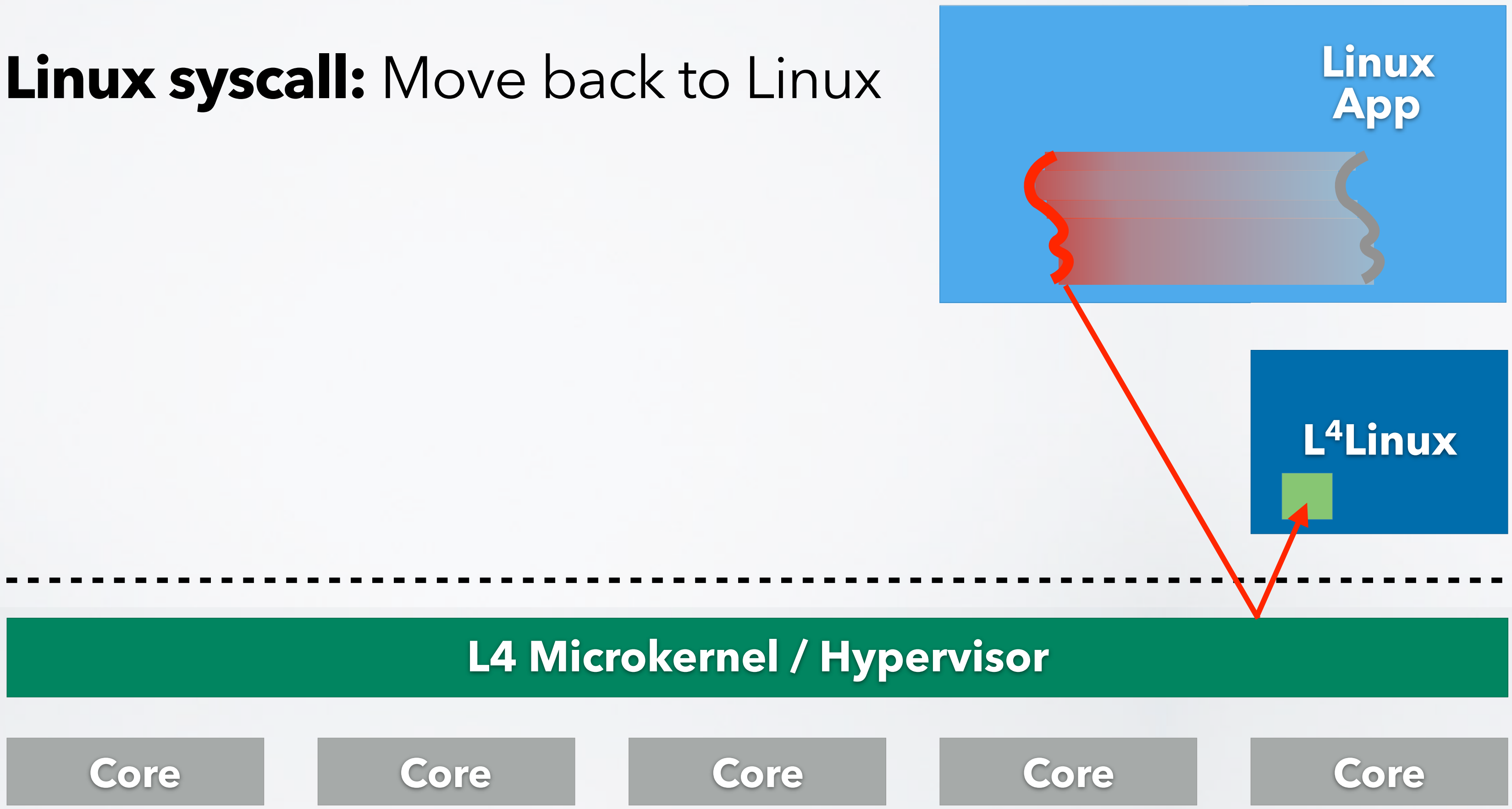
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core



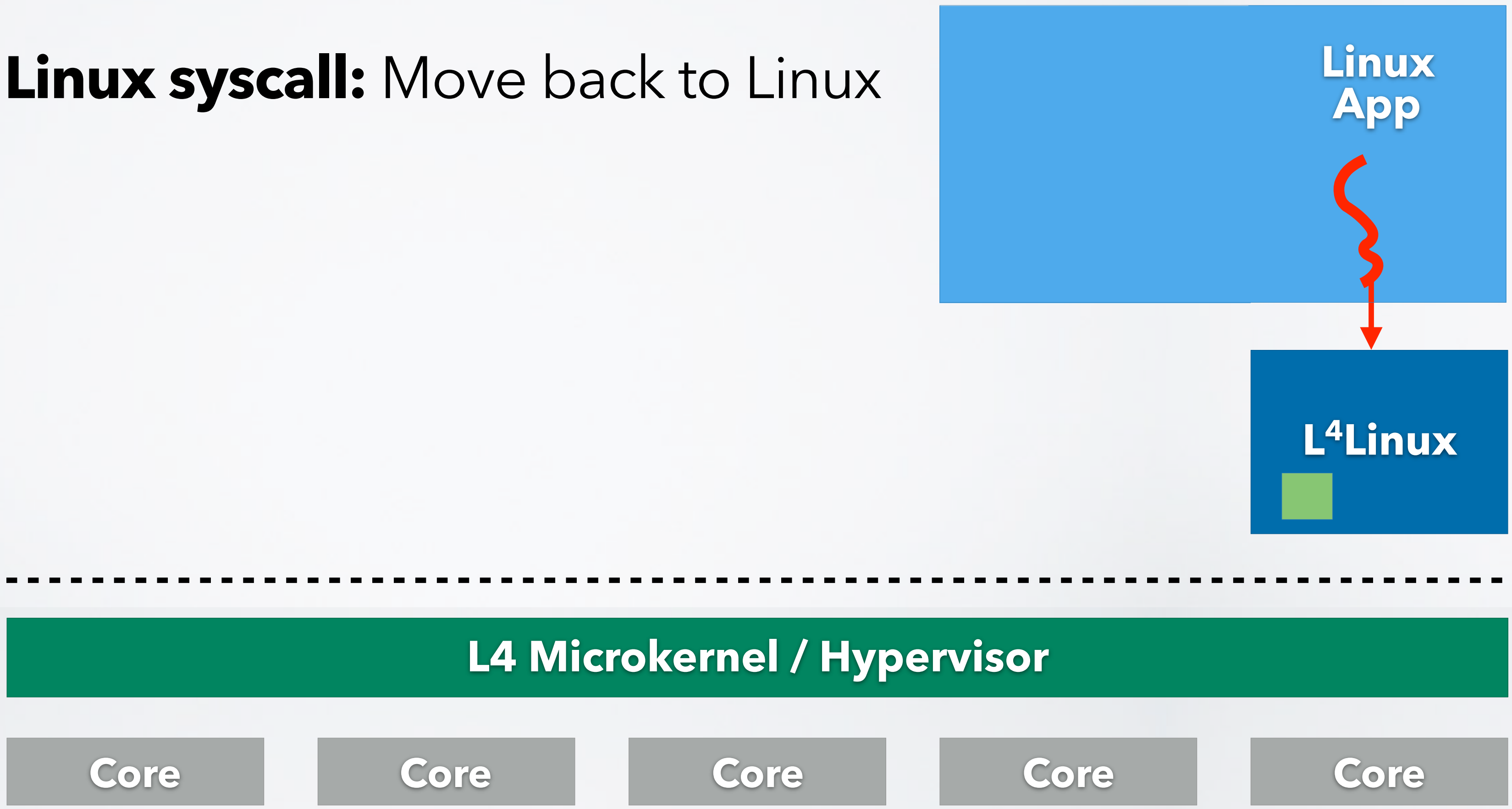
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux



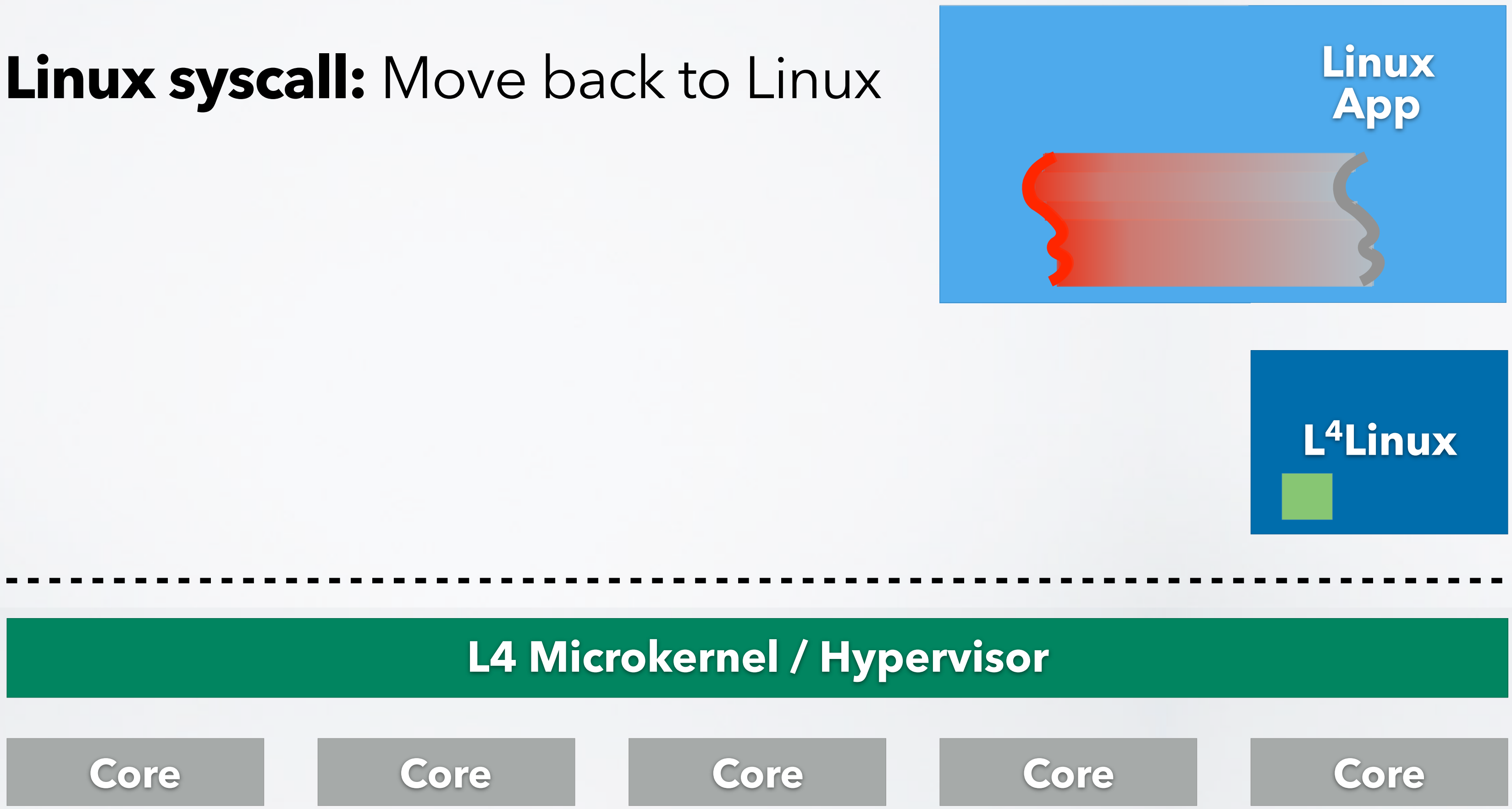
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux



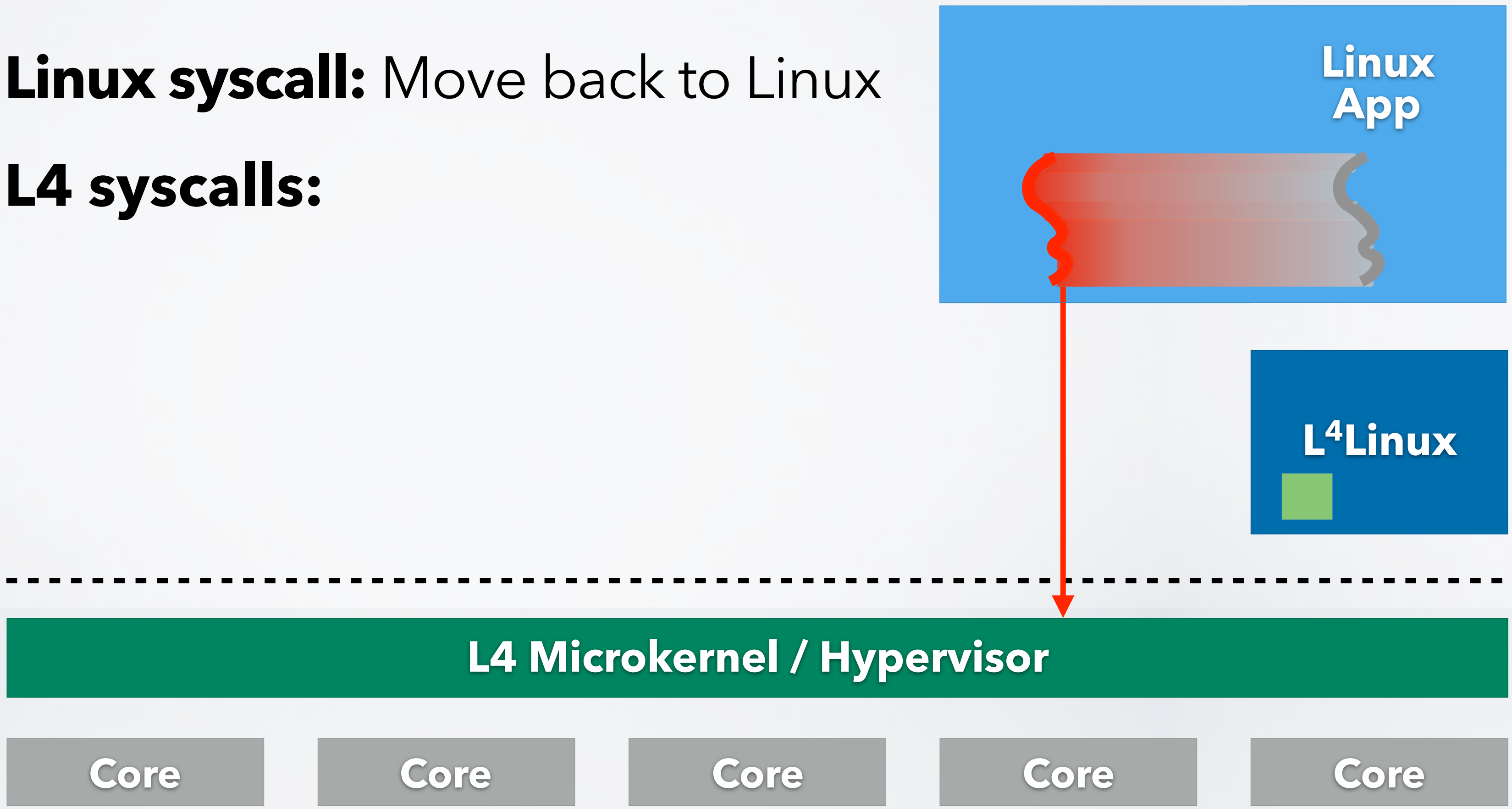
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux



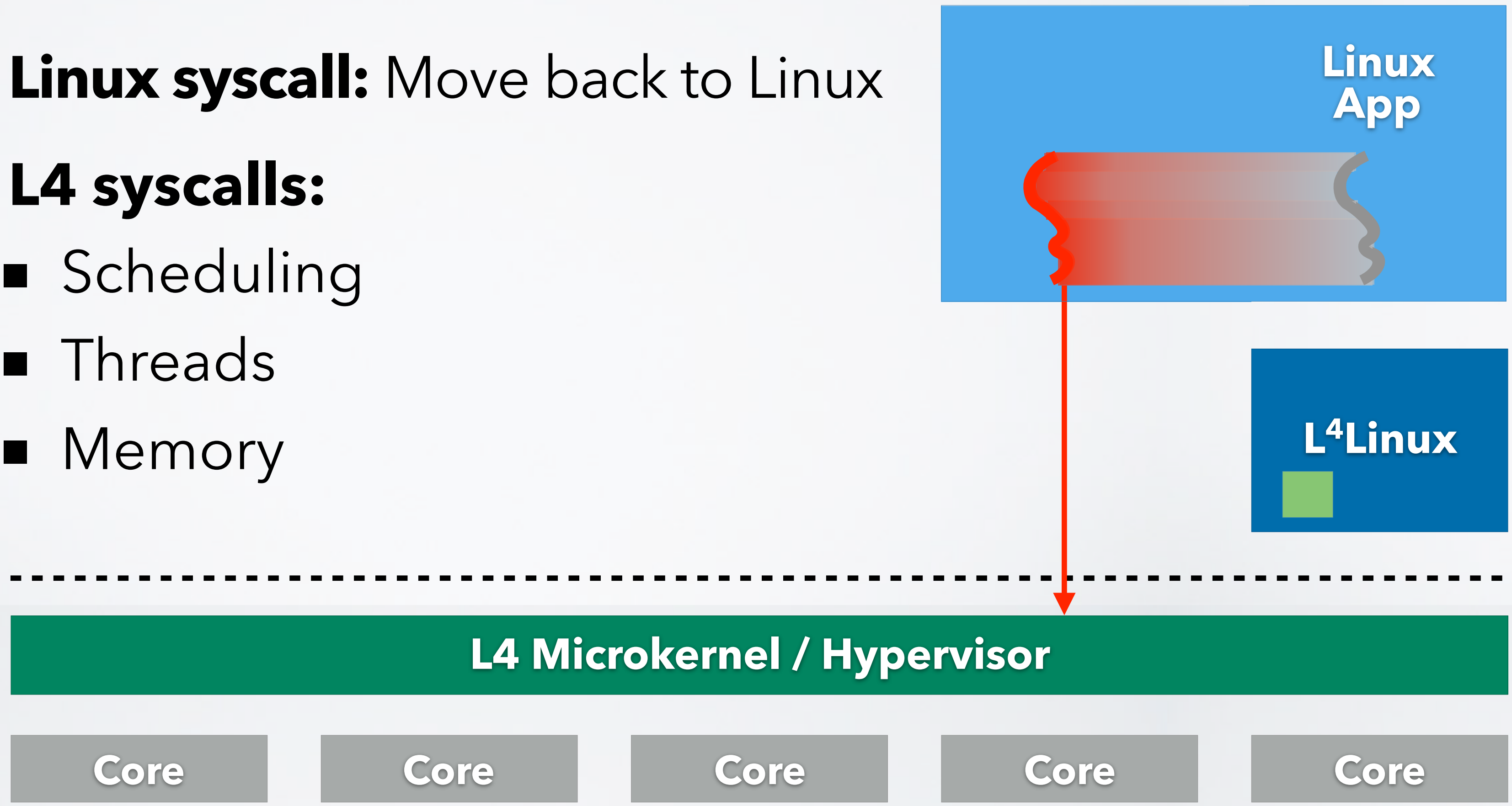
DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux
- **L4 syscalls:**



DECOUPLED EXECUTION

- **Decoupling:** move Linux thread to new L4 thread on its own core
- **Linux syscall:** Move back to Linux
- **L4 syscalls:**
  - Scheduling
  - Threads
  - Memory



FFMK/L4 PLATFORM

FFMK/L4 PLATFORM

- Adapted existing building blocks:

- Adapted existing building blocks:
  - Made public in 1997, kept **up-to-date** since

FFMK/L4 PLATFORM

- Adapted existing building blocks:
  - Made public in 1997, kept **up-to-date** since
  - Runs on **x86**, **ARM**, **MIPS** (all 32/64 bit)

FFMK/L4 PLATFORM

- Adapted existing building blocks:
  - Made public in 1997, kept **up-to-date** since
  - Runs on **x86**, **ARM**, **MIPS** (all 32/64 bit)
- Linux compatible + flexible L4 interfaces

FFMK/L4 PLATFORM

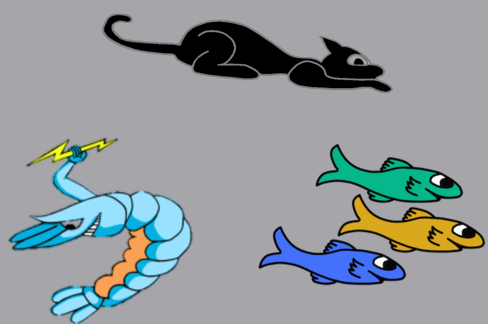
- Adapted existing building blocks:
  - Made public in 1997, kept **up-to-date** since
  - Runs on **x86**, **ARM**, **MIPS** (all 32/64 bit)
- Linux compatible + flexible L4 interfaces
- Better support for your runtime system?

FFMK/L4 PLATFORM

- Adapted existing building blocks:
  - Made public in 1997, kept **up-to-date** since
  - Runs on **x86**, **ARM**, **MIPS** (all 32/64 bit)
- Linux compatible + flexible L4 interfaces
- Better support for your runtime system?
- Code + docs: ffmk.tudos.org and l4re.org

Hobbes

- 12:15 - 12:17 Welcome (Rolf Riesen)
- 12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)
- 12:23 - 12:29 McKernel (Balazs Gerofi)
- 12:29 - 12:35 FFMK (Carsten Weinhold)
- **12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)**
- 12:41 - 12:47 mOS (Rolf Riesen)
- 12:47 - 13:15 Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions



SC'16 Panel: Multi-Kernel OSes for Extreme-Scale HPC

November 17, 2016

Kevin Pedretti
Center for Computing Research
Sandia National Laboratories

Center for Computing Research

Sandia National Laboratories

Exceptional service in the national interest

U.S. DEPARTMENT OF ENERGY

National Nuclear Security Administration

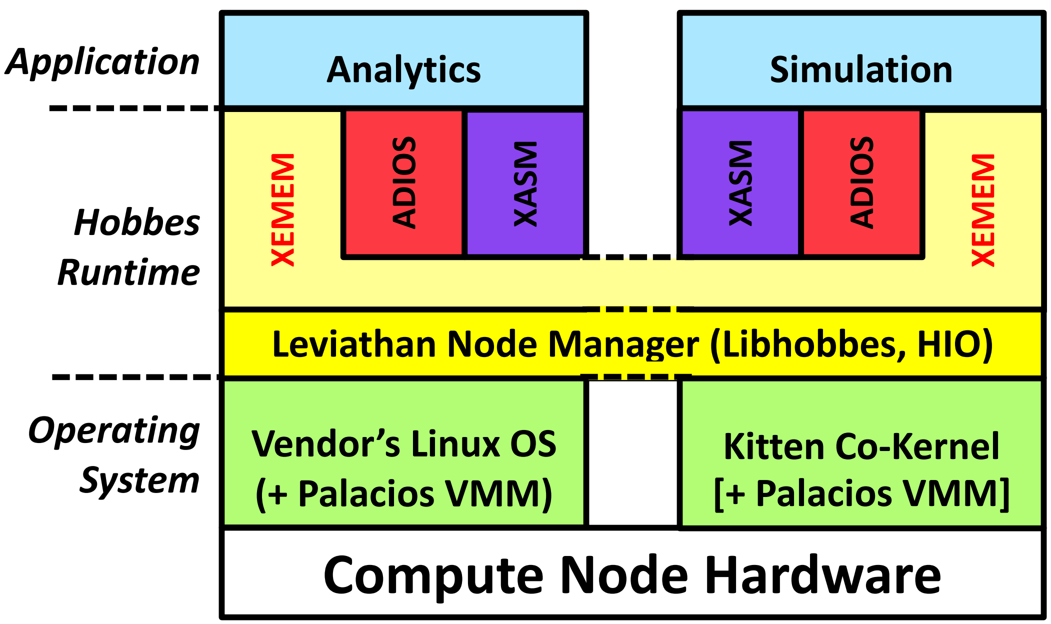
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Application Workflows are Evolving

- More compositional approach, where overall application is a composition of **coupled simulation, analysis, and tool components**
- Each component may have different OS and Runtime (OS/R) requirements, in general there is **no "one-size-fits-all" solution**
- Co-locating application components can be used to reduce data movement, but may **introduce cross component performance interference**
  - Need system software **infrastructure for application composition**
  - Need to maintain **performance isolation**
  - Need to provide **cross-component data sharing capabilities**
  - Need to fit into vendor's **production system software stack**

Hobbes: Multi-Stack Approach for Application Composition

Node Virtualization Layer (NVL)



HPDC'15

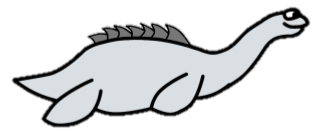
Team:

- Kevin Pedretti, Jay Lofstead,
  Brian Gaines, Shyamali Mukherjee, Noah Evans (SNL)
- Jack Lange, Brian Kocoloski, Jiannan Ouyang (Pitt)
- Patrick Bridges, Oscar Mondragon (UNM)
- Peter Dinda, Kyle Hale (Northwestern)
- Mike Lang (LANL)
- David Bernholdt (Enclave lead), Hasan Abbasi (ORNL)
- Jai Dayal (GaTech)

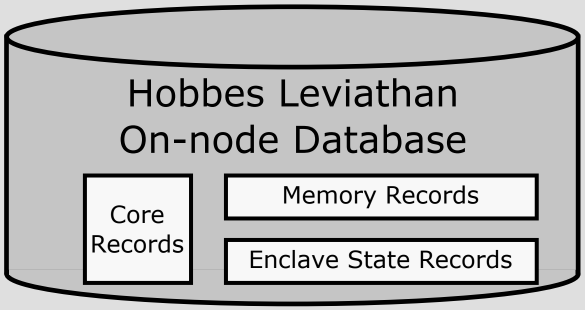
- Key Ideas
  - No one-size-fits-all OS/R
  - Partition node-level resources into "enclaves"
  - Run (potentially) different OS/R stack in each enclave
- Challenges
  - Performance isolation
  - Composition mechanisms
- Approach
  - Build a real, working system
  - Leverage Kitten LWK OS and Palacios Hypervisor
  - Use standard Linux host for bootstrap and enclave control
  - Develop libhobbes for use by Apps/Tools/Services

http://github.com/hobbesosr/nvl

Leviathan On-Node Manager Ties Things Together



Node Information Service



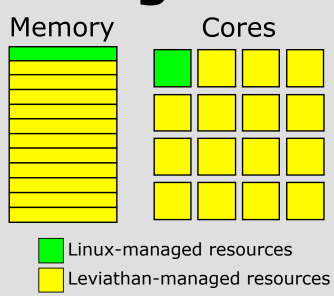
State of all resources tracked in in-memory NoSQL database

Enclave Lifecycle Management

```
Launch/Destroy Enclaves
Launch/Destroy Virtual Machines
Launch/Destroy Applications
```

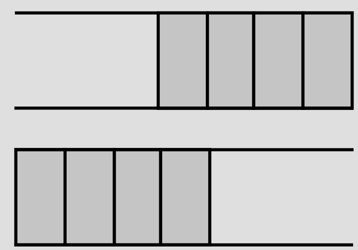
The Leviathan shell provides commands to form enclaves and launch applications

User-Level Resource Management



User-level has explicit control of physical resources managed by Leviathan

Inter-Enclave Communication



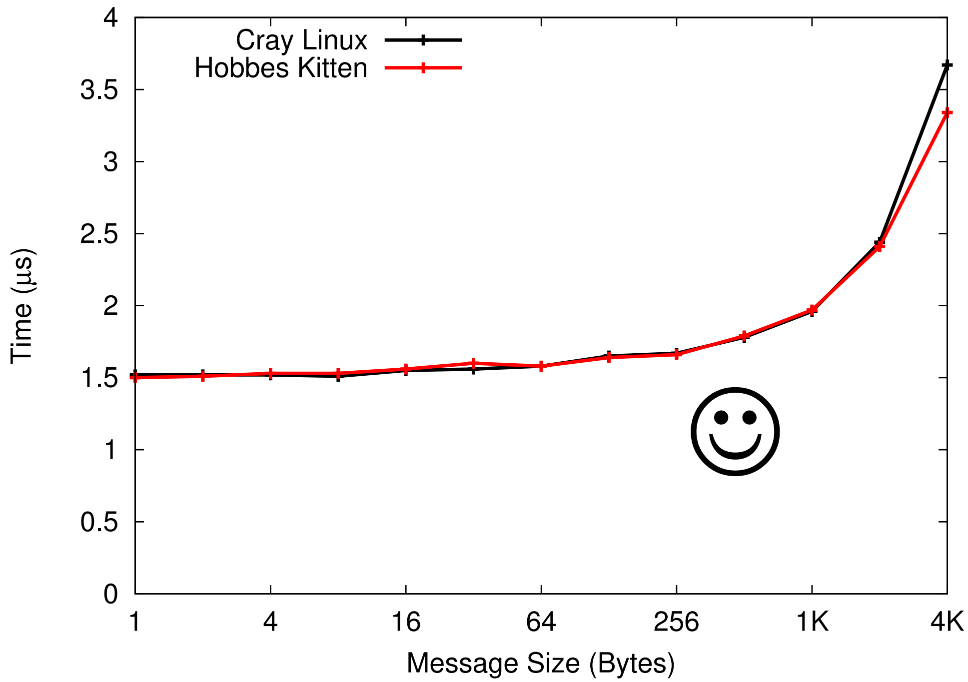
Built-in services for command queues, discovery, global IDs, and generic RPC

Hobbes Node Virtualization Layer Status and Plans: Networking is Working

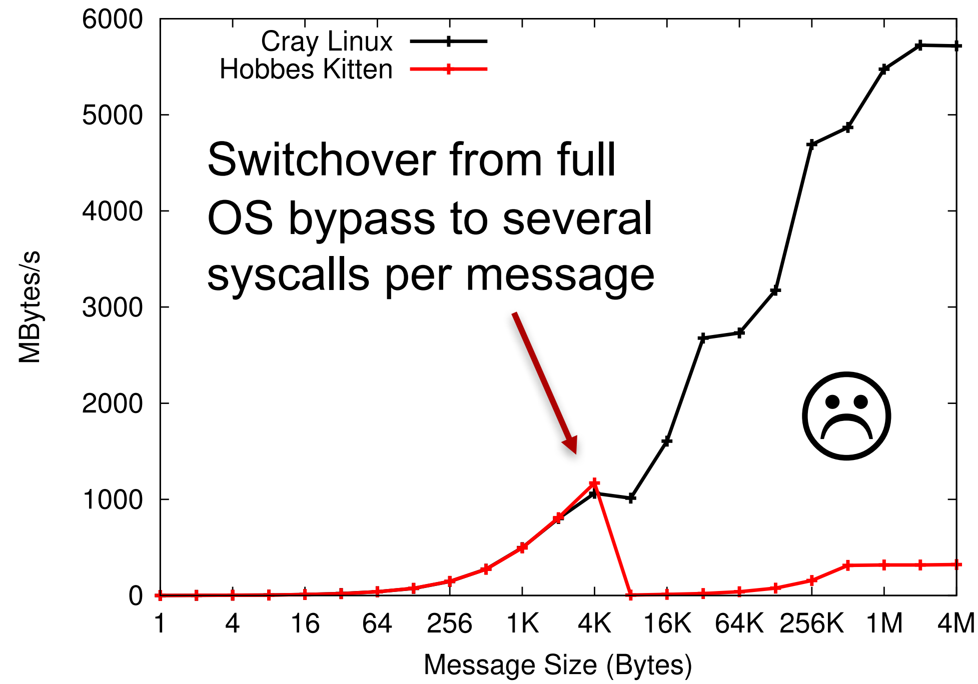
- Host-IO (HIO) system call forwarding layer complete
  - Adopts unified address space approach pioneered by McKernel
  - Applications built with Cray's default toolchain run on Kitten

```
aprun -N 1 -n 2 -L 6,7 ./hobbes launch_app
kitten-enclave-0 -with-hio=stub IMB-MPI1.openmpi
```

MPI Small Message Latency on Gemini

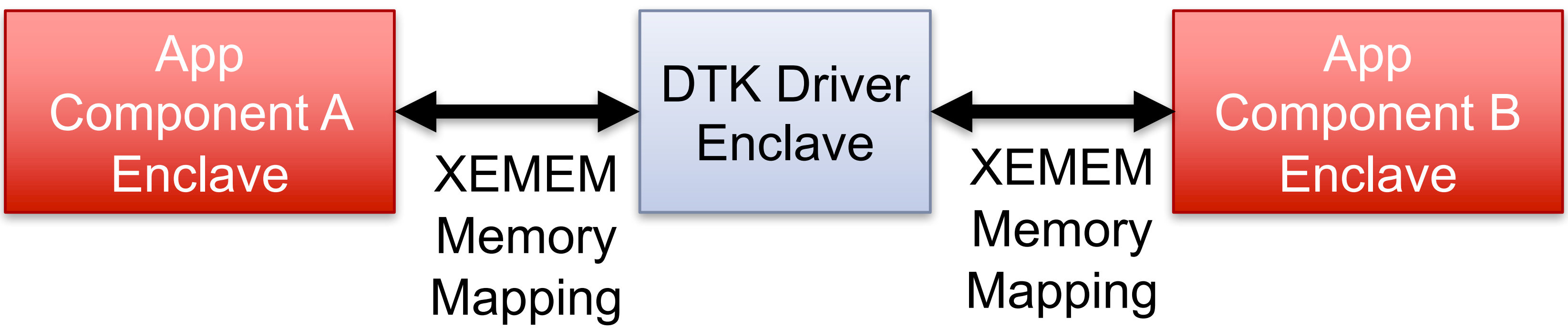


Bandwidth



Hobbes Node Virtualization Layer Status and Plans: Evaluation

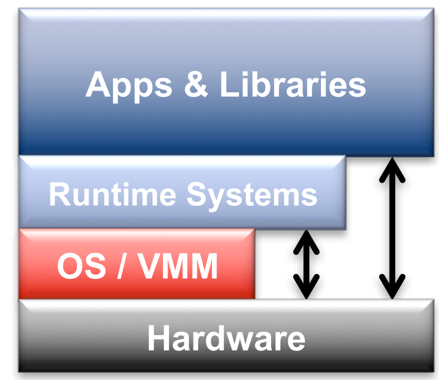
- Application composition examples
  - HPC + HPDA examples
  - DTK/STKmesh multi-enclave example developed by Hobbes enclave team (Vallee, Naughton, Slattery, Bernholdt)



- Empirical performance experiments
  - Lots of multi-kernels, no large-scale results -> need to do (!)
  - Evaluate benefit of LWK resource management policies
  - Understand importance of OS noise on modern platforms and apps

Why Virtualization in Large-Scale HPC?

- Support multiple system software stacks in same platform
  - Vendor's stack good for physics simulations, bad for data analytics
  - Virtualization adds flexibility, deploy custom images on demand
  - Not just user-space containers, need ability to run different OS kernels
    - Special-purpose Lightweight Kernels: mOS, McKernel, FFMK, Kitten 
    - Large-scale emulation experiments, networks + systems
    - Other custom OSes, unikernels, …
  - Leverage industry momentum, student mindshare
- Virtualization overhead can be very low
  - Don't oversubscribe, space share nodes, pin everything, use large pages, physically contiguous virtual memory
  - Demonstrated < 5% overhead in practice on 4K nodes (VEE'11)
- Challenges
  - Deployment: getting into vendor's software stack
  - Networking: need full OS bypass and hardware with virtualization support
  - Complex nodes: heterogeneous memory, many-core, SMT, NUMA, …



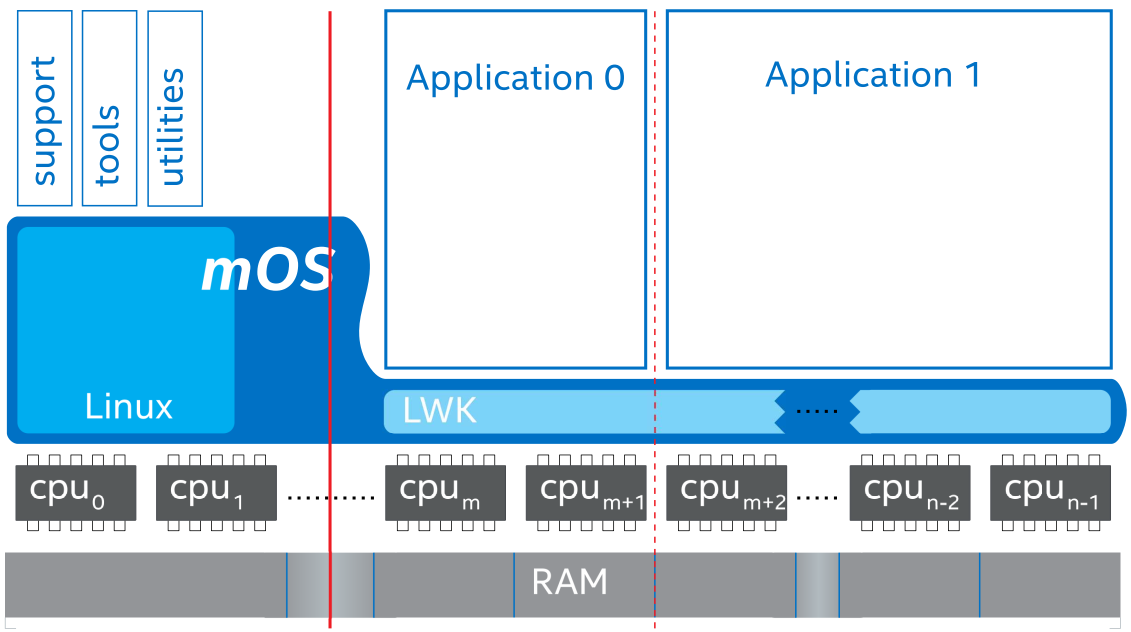
Compute Node System Software Stack, OS Bypass

mOS

- 12:15 - 12:17 Welcome (Rolf Riesen)
- 12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)
- 12:23 - 12:29 McKernel (Balazs Gerofi)
- 12:29 - 12:35 FFMK (Carsten Weinhold)
- 12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)
- **12:41 - 12:47 mOS (Rolf Riesen)**
- 12:47 - 13:15 Discussion with audience
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

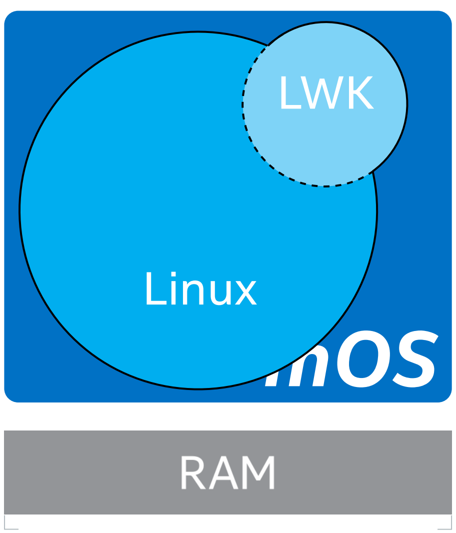
High-level architecture

- Dedicate a few cores in a many-core system to Linux
- The remaining cores run compute intensive processes on LWK
- Strong partitioning: Service versus compute side
- The LWK manages memory, but Linux can access it

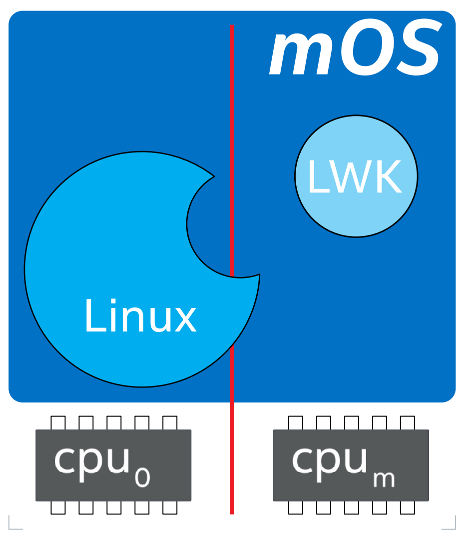


An embedded LWK

- We're neither trimming Linux to an LWK
- Nor are we adding Linux functionality to an LWK
- We are compiling our LWK into the Linux kernel
- Then, for each logical CPU, decide which kernel has control



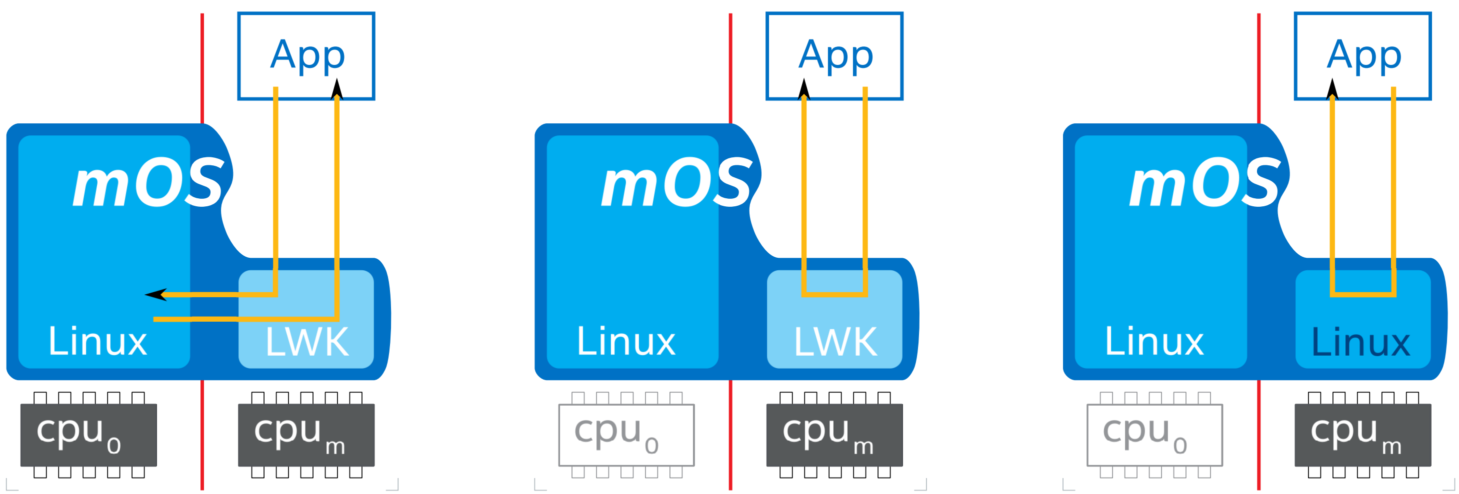
Code in memory



Code running

System call locality

- System calls can execute locally or “remote”
- Can use Linux or LWK code



Call Linux remotely | Call LWK directly | Call Linux directly

Status

- In the process of making *mOS* open source
- Things like CORAL benchmarks run
- In most cases beat Linux performance and run-to-run variability
  - Working on the ones where we don't yet
  - Expect bigger performance gap at higher node counts
- Starting to work with runtime system designers
  - Can we do something in *mOS* that is difficult in Linux and helps performance and scalability?
- Starting to work with hardware designers
  - *mOS* makes it easier to adapt to hardware features/quirks

Community interaction

- 12:15 - 12:17 Welcome (Rolf Riesen)
- 12:17 - 12:23 Intro to multi-kernels (Robert W. Wisniewski)
- 12:23 - 12:29 McKernel (Balazs Gerofi)
- 12:29 - 12:35 FFMK (Carsten Weinhold)
- 12:35 - 12:41 Kitten/Hobbes (Kevin Pedretti)
- 12:41 - 12:47 mOS (Rolf Riesen)
- **12:47 - 13:15 Discussion with audience**
  - Influence the work these teams are doing
  - Submit requests and give feedback
  - Ask questions

Discussion

- What feature / system call / tuning knob that Linux does not provide would make your life easier?
- Which applications should we support / optimize for?
- What information should the OS make available to you?
    - And how?

Discussion (cont.)

- Would you be willing to try these kernels on your system with your application?
- How much performance gain does a multi-OS need to deliver before you would consider switching?
  - 1990s LWKs were shunned due to lack of Linux compatibility
  - That's why we need multi-OSes!
  - Given the higher level of Linux compatibility, is 10% performance gain enough to convince you to switch?

Discussion (cont.)

- Multi-OSes have other advantages too (not just performance and scalability)
- Which ones are of importance to you?
  - Better defaults for HPC
  - Better control of hardware resources
  - Better handling of deep memory hierarchies
  - ?

Discussion (cont.)

- Online and audience questions and comments

Thank you!