# HermitCore – A Unikernel for Extreme Scale Computing

Stefan Lankes[1], Simon Pickartz[1], Jens Breitbart[2]

[1]RWTH Aachen University, Germany

[2]Technische Universität München, Germany

E.ON Energy Research Center | RWTH AACHEN UNIVERSITY

# Agenda

- Motivation
- OS Architectures
- HermitCore Design
- Performance Evaluation
- Conclusion and Outlook

# Motivation

- Yet Another Multi-Kernel Approach
- Nearly the same motivation like Balazs Gerofi et al.[1]
- Complexity of high-end HPC systems keeps growing
  - Extreme degree of parallelism
  - Heterogeneous core architectures
  - Deep memory hierarchy
  - Power constrains
    - ⇒ Need for scalable, reliable performance and capability to rapidly adapt to new HW
- Applications have also become complex
  - In-situ analysis, workflows
  - Sophisticated monitoring and tools support, etc. . .
  - Isolated, consistent simulation performance
    - ⇒ Dependence on POSIX and the rich Linux APIs
- Seemingly contradictory requirements. . .

[1] B. Gerofi et al. "Exploring the Design Space of Combining Linux with Lightweight Kernels for Extreme Scale Computing". In: 5th Int. Workshop on Runtime and Operating Systems for Supercomputers. 2015.

# Motivation

- Yet Another Multi-Kernel Approach
- Nearly the same motivation like Balazs Gerofi et al.[1]
- Complexity of high-end HPC systems keeps growing
  - Extreme degree of parallelism
  - Heterogeneous core architectures
  - Deep memory hierarchy
  - Power constrains
    - ⇒ Need for scalable, reliable performance and capability to rapidly adapt to new HW
- Applications have also become complex
  - In-situ analysis, workflows
  - Sophisticated monitoring and tools support, etc. . .
  - Isolated, consistent simulation performance
    - ⇒ Dependence on POSIX ~~and the rich Linux APIs~~, MPI and OpenMP
- Seemingly contradictory requirements. . .

[1] B. Gerofi et al. "Exploring the Design Space of Combining Linux with Lightweight Kernels for Extreme Scale Computing". In: 5th Int. Workshop on Runtime and Operating Systems for Supercomputers. 2015.

# OS Architectures

## Light-weight and / or Multi-Kernels for HPC

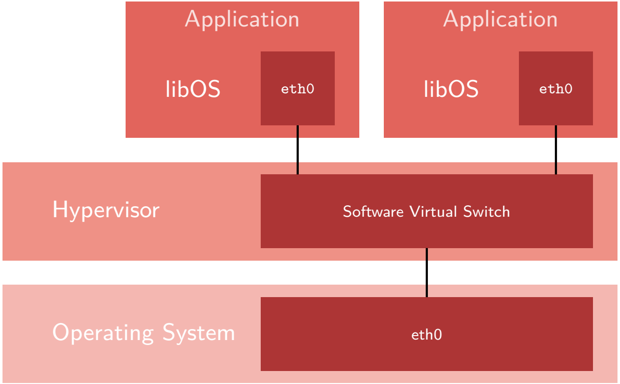
- mOS, McKernel, Catamount, ZeptoOS, FusedOS, L4, FFMK, Hobbes, Kitten, CNK. . .
- Detailed analyzes in the next talk[2]

## Unikernels / LibraryOS

- Basic ideas already developed in the *Exokernel Era*
  - Each process has it own hardware abstraction layer
- Regained relevance in the area of cloud computing (e. g., IncludeOS, MirageOS)
  - With Qemu / KVM the abstraction layer is already defined
- HermitCore is a combination of a multi-kernel and a unikernel

[2] B. Gerofi et al. "A Multi-Kernel Survey for High-Performance Computing". In: 6th Int. Workshop on Runtime and Operating Systems for Supercomputers. 2016.

# OS Designs for Cloud Computing – LibraryOS




- Now, every system call is a function call $\Rightarrow$ Low overhead

# HermitCore – Basic ideas

- Combination of the Unikernel and Multi-Kernel to reduce the overhead
  - Support of bare-metal execution
  - Unikernel ⇒ system calls are realized as function call
- Single-address space operating system ⇒ No TLB Shootdown
- System software should be designed for the hardware
  - Hierarchical approach (like the hardware)
- One kernel per NUMA node
  - Only local memory accesses (UMA)
  - Message passing between NUMA nodes
- Support of dominant programming models (MPI, OpenMP)
- One FWK (Linux) in the system to get access to a broader driver support
  - Only a backup for pre- / post-processing
  - Critical path should be handled by HermitCore
    - Most *system calls* handled by HermitCore
    - E. g., memory allocation, access to the network interface

# Booting HermitCore

- By detection of a HermitCore app, a proxy will be started.

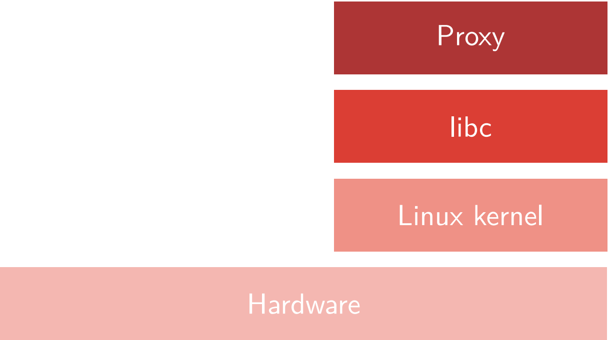
Proxy

libc

Linux kernel

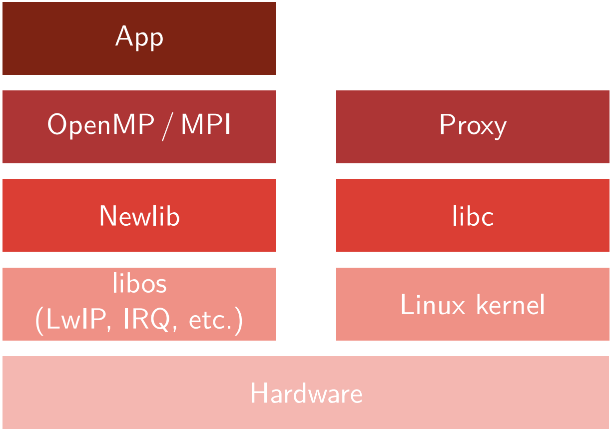
Hardware

# Booting HermitCore



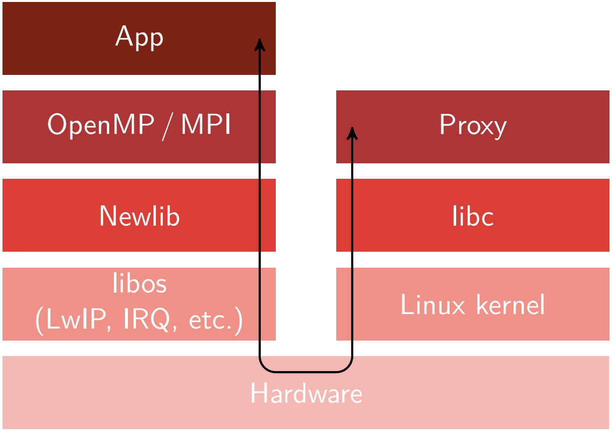

- By detection of a HermitCore app, a proxy will be started.
- The proxy unplugs a set of cores.

# Booting HermitCore



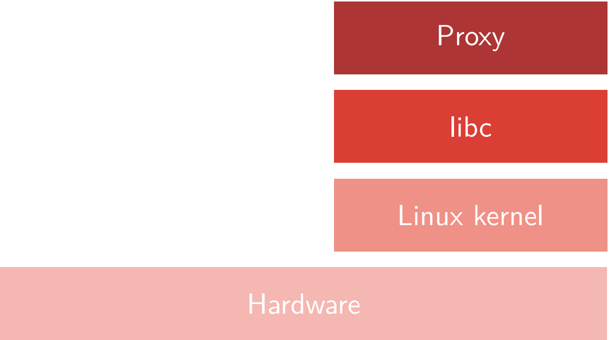

- By detection of a HermitCore app, a proxy will be started.
- The proxy unplugs a set of cores.
- Triggers Linux to boot HermitCore on the unused cores.

# Booting HermitCore



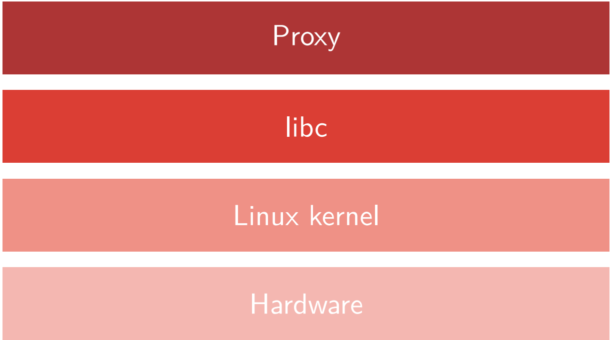

- By detection of a HermitCore app, a proxy will be started.
- The proxy unplugs a set of cores.
- Triggers Linux to boot HermitCore on the unused cores.
- A reliable connection will be established.

# Booting HermitCore




- By detection of a HermitCore app, a proxy will be started.
- The proxy unplugs a set of cores.
- Triggers Linux to boot HermitCore on the unused cores.
- A reliable connection will be established.
- By termination, the cores are set to the HALT state.

# Booting HermitCore




- By detection of a HermitCore app, a proxy will be started.
- The proxy unplugs a set of cores.
- Triggers Linux to boot HermitCore on the unused cores.
- A reliable connection will be established.
- By termination, the cores are set to the HALT state.
- Finally, reregistering of the cores to Linux.

# HermitCore's Toolchain (I)

## Memory Layout

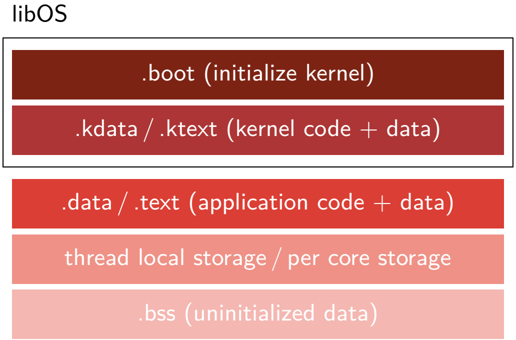
libOS

| |
|---|
| .boot (initialize kernel) |
| .kdata / .ktext (kernel code + data) |
| .data / .text (application code + data) |
| thread local storage / per core storage |
| .bss (uninitialized data) |

- Basic OS services (e. g., interrupt handling) are separated in a library
- Linked to a *normal* application like the C library
- A fix address for the init code is required
  - Defined in the linker script
  - Part of HermitCore's cross toolchain
    - GCC 5.3.0 & Binutils
    - Support of C / C++ & Fortran
  - No changes to the common build process

# HermitCore's Toolchain (II)

## Memory Layout



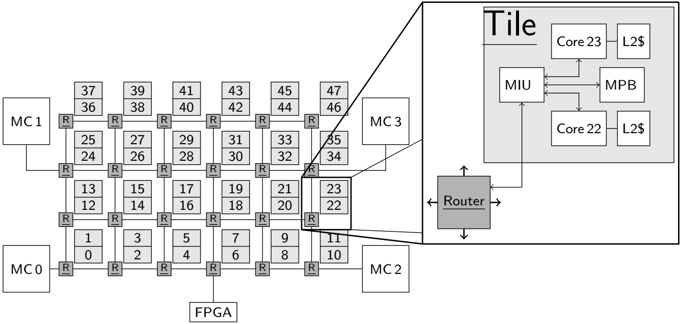

- Transparant loading of HermitCore apps
- Definition of a new ELF ABI
  - Only the magic number for the OS has been changed in the ELF format
  - Minor modifications to GCC & binutils
- By Linux support of miscellaneous binary formats (*binfmt*), the loader checks the magic number for the OS
  1. Detection of the magic number
  2. Starting the proxy
  3. Proxy initiates via *sysfs* the boot process of HermitCore apps
- No changes to the common build process



E.ON Energy Research Center RWTHAACHEN UNIVERSITY

# Runtime Support

- SSE, AVX, FMA,...
- Full C-library support (newlib)
- IP interface & BSD sockets (LwIP)
  - IP packets are forwarded to Linux
  - Shared memory interface
- Pthreads
  - Thread binding at start time
  - No load balancing $\Rightarrow$ less housekeeping
- OpenMP
- iRCCE- & MPI (via SCC-MPICH)



# OpenMP Runtime

- GCC includes a OpenMP Runtime (libgomp)
  - Reuse synchronization primitives of the Pthread library
  - Other OpenMP runtimes scales better
  - In addition, our Pthread library was originally not designed for HPC
- Integration of Intel's OpenMP Runtime
  - Include its own synchronization primitives
  - Binary compatible to GCC's OpenMP Runtime
  - Changes for the HermitCore support are small
    - Mostly deactivation of function to define the thread affinity
  - Transparent usage
    - For the end-user, no changes in the build process

# Support of compilers beside GCC

- Just avoid the standard environment (–ffreestanding)
- Set include path to HermitCore's toolchain
- Be sure that the ELF file use HermitCore's ABI
  - Patching object files via elfedit
- Use the GCC to link the binary

```
LD = x86_64-hermit-gcc
#CC = x86_64-hermit-gcc
#CFLAGS = -O3 -mtune=native -march=native -fopenmp -mno-red-zone
CC = icc -D__hermit__
CFLAGS = -O3 -xHost -mno-red-zone -ffreestanding -I$(HERMIT_DIR) -openmp
ELFEDIT = x86_64-hermit-elfedit

stream.o: stream.c
        $(CC) $(CFLAGS) -c -o $@ $<
        $(ELFEDIT) --output-osabi HermitCore $@

stream: stream.o
        $(LD) -o $@ $< $(LDFLAGS) $(CFLAGS)
```

# Operating System Micro-Benchmarks

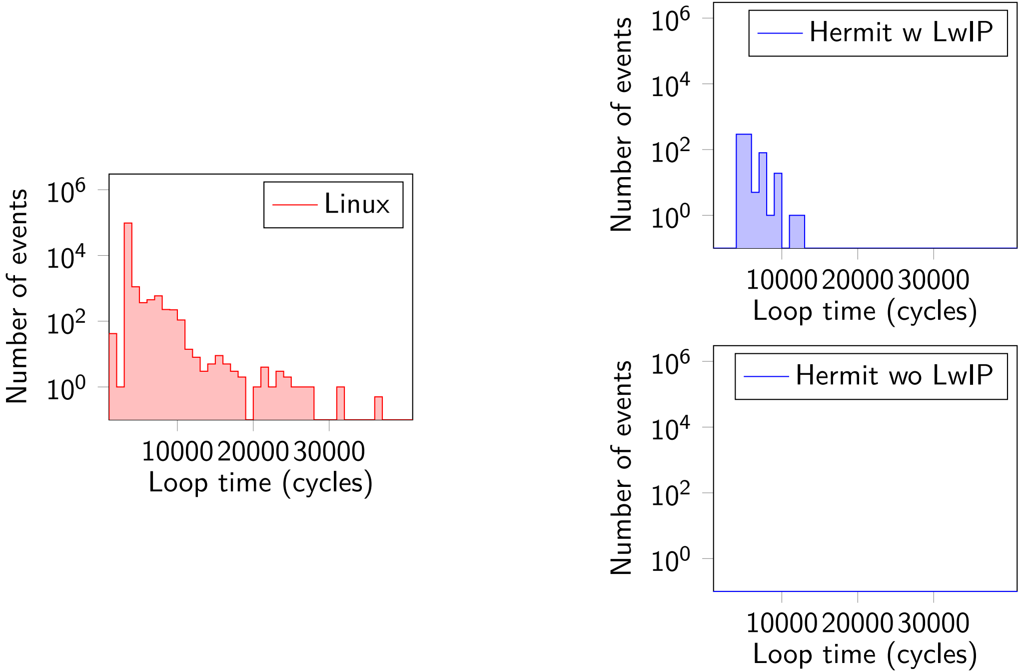
- Test system
  - Intel Haswell CPUs (E5-2650 v3) clocked at 2.3 GHz
  - 64 GiB DDR4 RAM and 25 MB L3 cache
  - SpeedStep Technology and TurboMode are deactivated
  - 4.2.5 Linux kernel on Fedora 23 (Workstation Edition)
  - gcc 5.3.x, AVX- & FMA-Support enabled (−mtune=native)
- Results in CPU cycles

| System activity | HermitCore | Linux |
|---|---|---|
| `getpid()` | 14 | 143 |
| `sched_yield()` | 97 | 370 |
| `write()` | 3520 | 1079 |
| `malloc()` | 3772 | 6575 |
| first write access to a page | 2014 | 4007 |

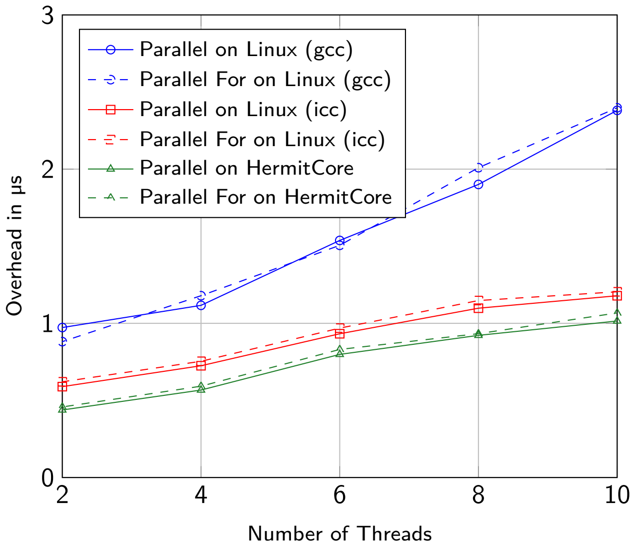
# Hourglass Benchmark

- Benchmarks reads permanently the time step counter
- (Larger) Gaps $\Rightarrow$ OS takes computation time (e. g., for housekeeping, devices drivers)
- Results in CPU cycles

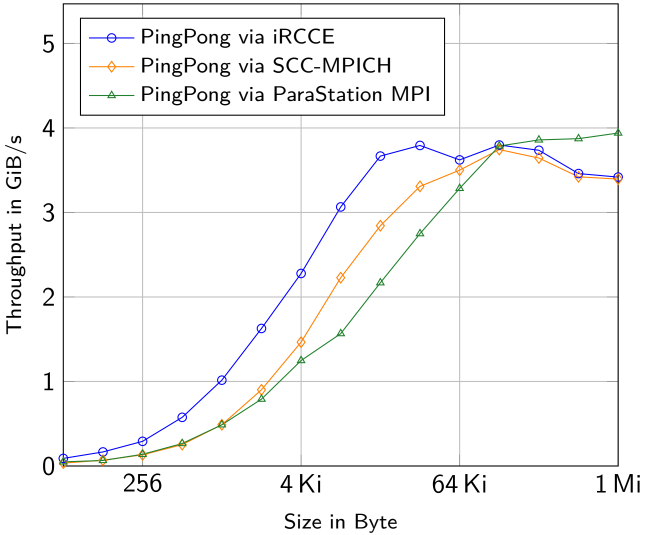
| **OS** | **Gaps** | |
|---|---|---|
| | Avg | Max |
| Linux | 69 | 31068 |
| HermitCore (w/ LwIP) | 68 | 12688 |
| HermitCore (w/o LwIP) | 68 | 376 |


Linux
Number of events
$10^6$
$10^4$
$10^2$
$10^0$
10000 20000 30000
Loop time (cycles)
Hermit w LwIP
Number of events
$10^6$
$10^4$
$10^2$
$10^0$
10000 20000 30000
Loop time (cycles)
Hermit wo LwIP
Number of events
$10^6$
$10^4$
$10^2$
$10^0$
10000 20000 30000
Loop time (cycles)

# EPCC OpenMP Micro-Benchmarks

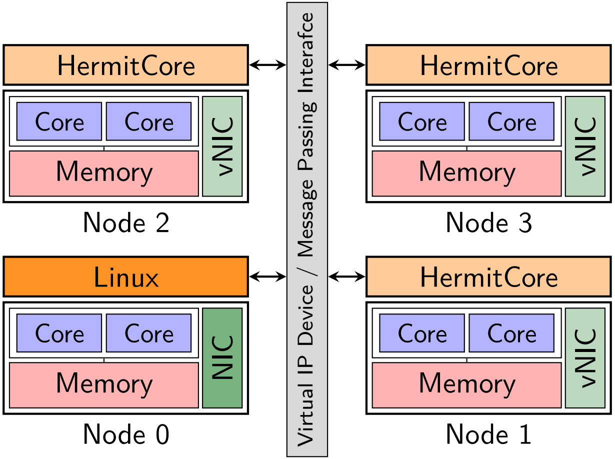


# Throughput Results of the Inter-kernel Communication Layer



# Outlook

- A fast direct access to the interconnect is required
- SR-IOV simplifies the coordination between Linux & HermitCore



# Conclusions

- Prototype works
- Nearly no OS noise
- First performance results are promising
- Suitable for Real-Time Computing?
- Try it out!

`http://www.hermitcore.org`

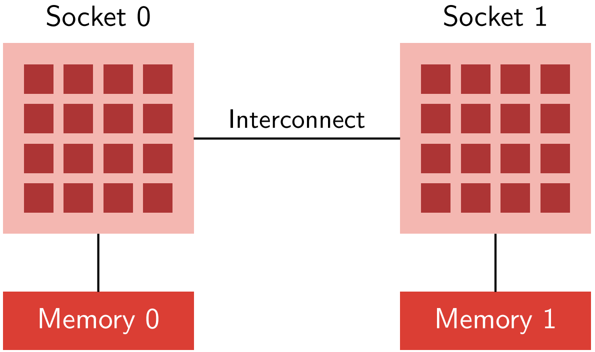
Thank you for your kind attention!

# Backup slides

# Lack of programmability

## Non-Uniform Memory Access

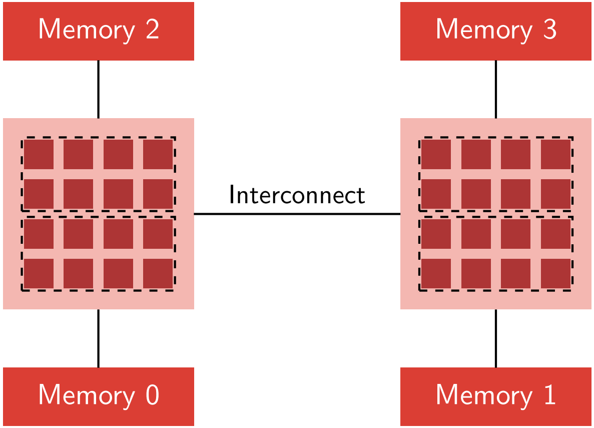
- Costs for memory access may vary
- Run processes where memory is allocated
- Allocate memory where the process resides
- Implications for the performance
  - Where should the applications store the data?
  - Who should decide the location?
    - The operating system?
    - The application developers?



# Lack of programmability

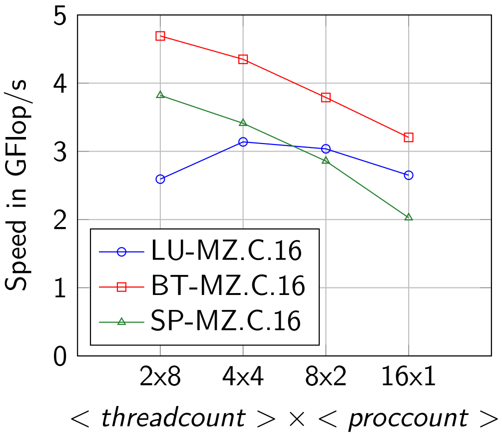
## Non-Uniform Memory Access

- Costs for memory access may vary
- Run processes where memory is allocated
- Allocate memory where the process resides
- Implications for the performance
  - Where should the applications store the data?
  - Who should decide the location?
    - The operating system?
    - The application developers?

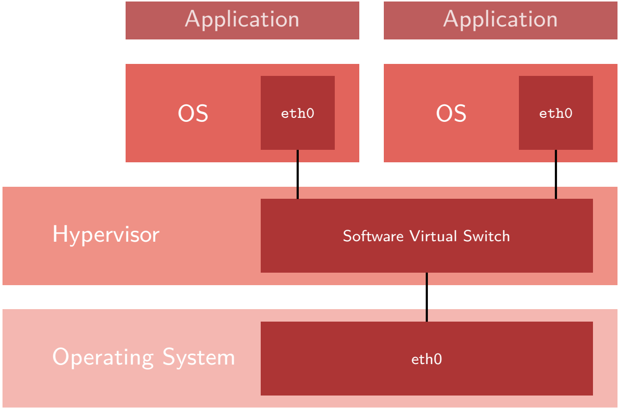


# Tuning Tricks

- Parallelization via Shared Memory (OpenMP)
  - Many side-effects and error-prone
  - Incremental parallelization
- Parallelization via Message Passing (MPI)
  - Restructuring of the sequential code
  - Less side-effects
- Performance Tuning
  - Bind MPI applications on one NUMA node
  - ⇒ No remote memory access

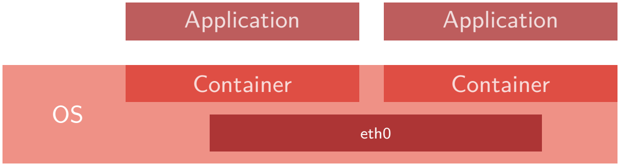


# OS Designs for Cloud Computing – Usage of Common OS




- Two operating systems to maintain one computer?
- Double Management!

# OS Designs for Cloud Computing – Container




- Building of virtual borders (namespaces)
- Containers and their processes doesn't see each other
- Fast access to OS services
- Less secure because an exploit for the container attacks also the host OS

# Comparison to related Unikernels

- Rump kernels[3]
  - Part of NetBSD ⇒ e. g., NetBSD's TCP / IP stack is available as library
  - Strong dependencies to the hypervisor
  - Not directly bootable on a standard hypervisor (e. g., KVM)
- IncludeOS[4]
  - Runs natively on the hardware ⇒ minimal Overhead
  - Neither 64 bit, nor SMP support
- MirageOS[5]
  - Designed for the high-level language OCaml ⇒ uncommon in HPC

[3] A. Kantee and J. Cormack. "Rump Kernels – No OS? No Problem!" In: ; login: 2014.
[4] A. Bratterud et al. "IncludeOS: A Resource Efficient Unikernel for Cloud Services". In: 7th Int. Conference on Cloud Computing Technology and Science. 2015.
[5] A. Madhavapeddy et al. "Unikernels: Library Operating Systems for the Cloud". In: 8th Int. Conference on Architectural Support for Programming Languages and Operating Systems. 2013.

# Is the software stack difficult to maintain?

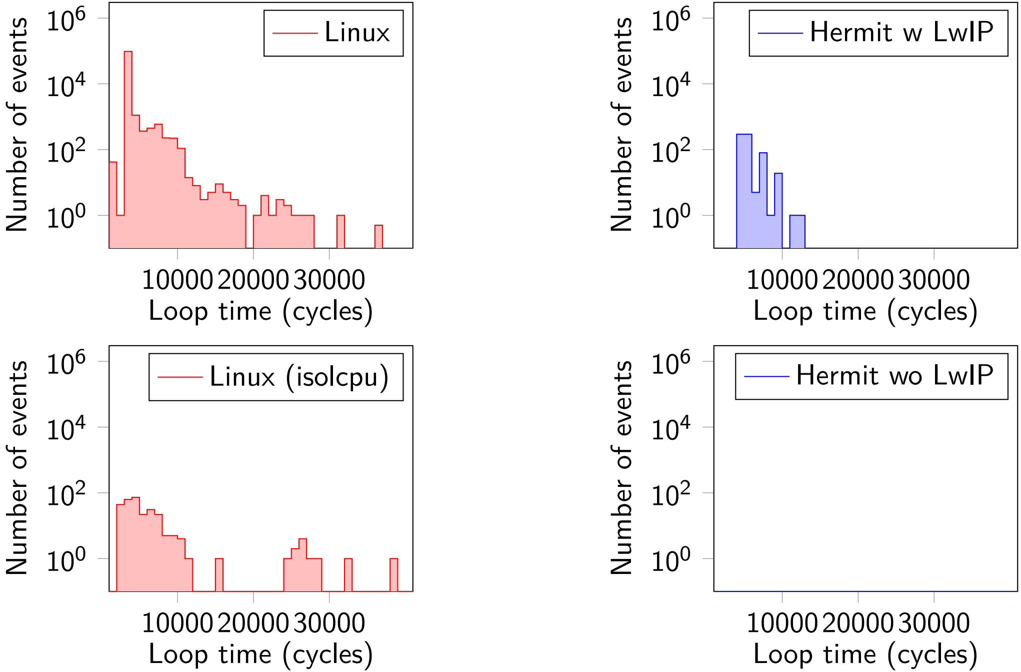
- Changes to the common software stack determined with `cloc`

| Software Stack | LoC | Changes |
|---|---:|---:|
| binutils | 5 121 217 | 226 |
| gcc | 6 850 382 | 4 821 |
| Linux | 15 276 013 | 1 296 |
| Newlib | 1 040 826 | 5 472 |
| LwIP | 38 883 | 832 |
| Pthread | 13 768 | 466 |
| OpenMP RT | 61 594 | 324 |
| HermitCore | – | 10 597 |

# Hourglass Benchmark

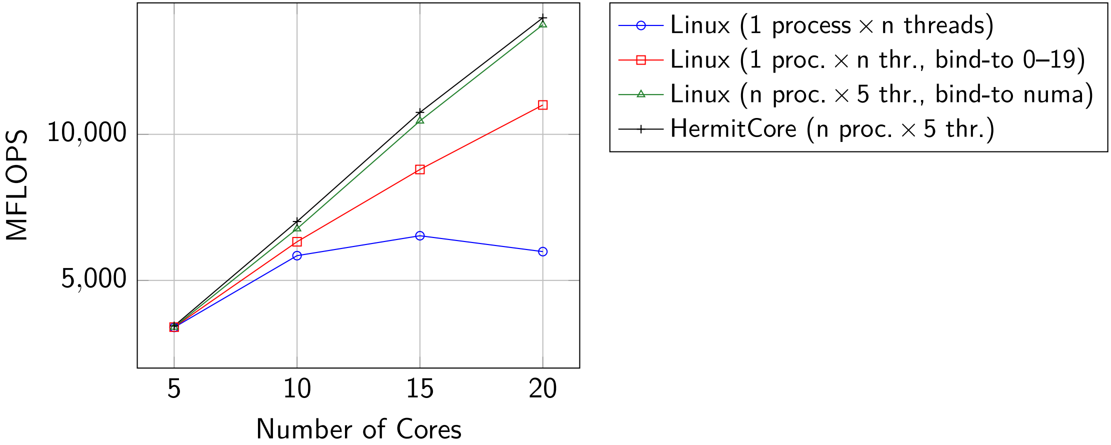
- Benchmarks reads permanently the time step counter
- (Larger) Gaps $\Rightarrow$ OS takes computation time (e. g., for housekeeping, devices drivers)
- Results in CPU cycles

| OS | Gaps | |
|---|---|---|
| | Avg | Max |
| Linux | 69 | 31068 |
| Linux (isolcpu) | 69 | 51840 |
| HermitCore (w/ LwIP) | 68 | 12688 |
| HermitCore (w/o LwIP) | 68 | 376 |



Linux
Hermit w LwIP
Linux (isolcpu)
Hermit wo LwIP
Number of events
$10^0$
$10^2$
$10^4$
$10^6$
10000
20000
30000
Loop time (cycles)

# Hydro (preliminary results)



# Which kind of security do we need?

- Unikernels $\Rightarrow$ no system calls $\Rightarrow$ unsecure?
- In HPC, security could be realized by a cluster management tool
- Could Intel's MPX (Memory Protection Extensions) protect the kernel for uncontrolled access?
  - Part of the Skylake architecture
  - MPX introduces new bounds registers to protect the system against buffer overflows
  - Kernel could be the lower bound of a buffer...
- A (bare-metal) hypervisor solves the problem completely

Thank you for your kind attention!

**Stefan Lankes et al.** – slankes@eonerc.rwth-aachen.de

Institute for Automation of Complex Power Systems
E.ON Energy Research Center, RWTH Aachen University
Mathieustraße 10
52074 Aachen

www.acs.eonerc.rwth-aachen.de