HermitCore

A Library Operating System for Cloud and High-Performance Computing

Stefan Lankes

RWTH Aachen University, Germany

E.ON Energy Research Center | RWTH AACHEN UNIVERSITY

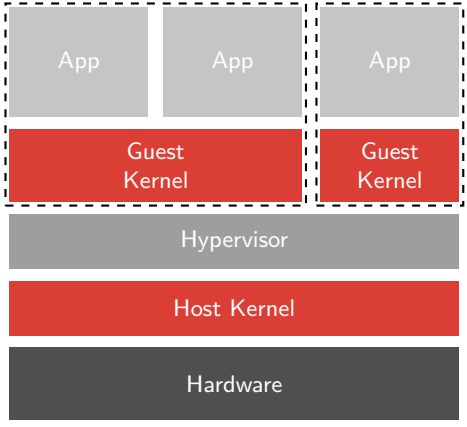
Pros and Cons of Virtualization Technologies

Advantages

- Flexibility (e. g., OS customization)
- Performance isolation
- Reliability (e. g., checkpointing)
- Load balancing via migration

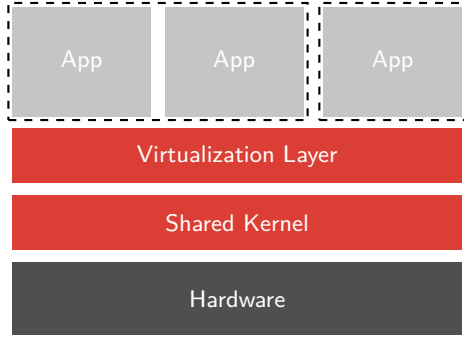
Disadvantages

- Complexity and overhead (e. g., nested page tables)
- Double management of ressources
  - Two schedulers
  - Two software stacks for I / O handling



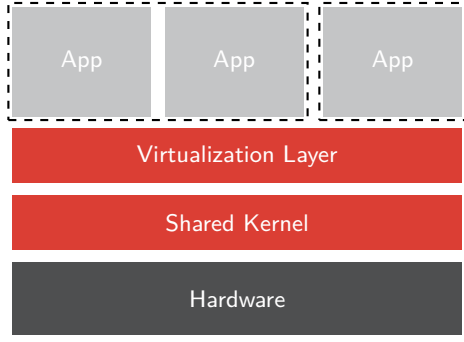
Light-weight Virtualization via Containers

- Building virtual borders
  - namespaces
  - cgroups
- One shared kernel
  - Host is vulnurable to attacs from wihtin containers



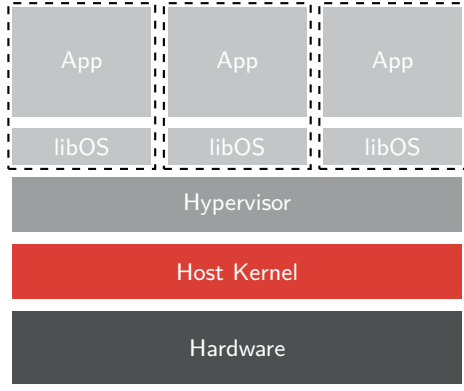
Light-weight Virtualization via Containers

- Building virtual borders
  - namespaces
  - cgroups
- One shared kernel
  - Host is vulnurable to attacs from wihtin containers
- Why do we prefer a multi-user multi-tasking environment?
- Why doesn't a user get direct hardware access?
  - But we don't have any problem to download and to install untrusted code?



Unikernels / Library Operating Systems

- Basic ideas come from the *Exokernel Era*
  - Each process has it own hardware abstraction layer
- Regained relevance
  - With Qemu / KVM the abstraction layer is already defined
- System calls are a common function call
- Single-address space $\Rightarrow$ single processing
  - No TLB shoot-down
- Minimal overhead



Comparison to Related Unikernels

- Rump kernels[1]
  - Part of NetBSD ⇒ (e. g., NetBSD's TCP / IP stack is available as library)
  - Not directly bootable on a standard hypervisor (e. g., KVM)
- IncludeOS[2]
  - Runs natively on the hardware ⇒ minimal Overhead
  - Neither 64 bit, nor SMP support (as far as I know)
- MirageOS[3]
  - Designed for the high-level language OCaml ⇒ uncommon in HPC
- OSv
  - see previous talk

[1] A. Kantee and J. Cormack. "Rump Kernels – No OS? No Problem!" In: ; login: 2014.

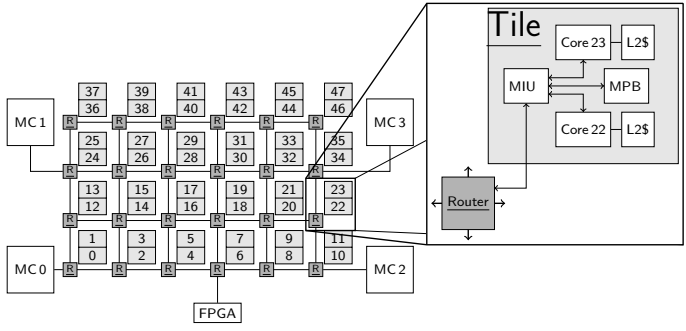
[2] A. Bratterud et al. "IncludeOS: A Resource Efficient Unikernel for Cloud Services". In: 7th Int. Conference on Cloud Computing Technology and Science. 2015.

[3] 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.

Runtime Support

- GNU Cross-Compilers for C / C++, Fortran & Go
- 64bit, AVX(2), AVX512, SMP...
- Full C-library support (newlib)
- IP interface & BSD sockets (LwIP)

- Pthreads
  - Thread binding at start time
  - No load balancing ⇒ 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

First steps...

Binary package

- The whole toolchain is available as Debian packages

```
echo "deb [trusted=yes] https://dl.bintray.com/rwth-os/hermitcore
vivid main" | sudo tee -a /etc/apt/sources.list
sudo apt-get -qq update
sudo apt-get install binutils-hermit newlib-hermit \
                     pthread-embedded-hermit gcc-hermit \
                     libhermit
```

- Afterwards the whole toolchain is located in `/opt/hermit/bin`
- Register HermitCore's proxy

```
sudo echo ":hermit:M:7:\\x42::/opt/hermit/bin/proxy:" \
          > /proc/sys/fs/binfmt_misc/register
```

Why is a Proxy Required?

- HermitCore defines its own object format
- By starting HermitCore application, Linux asks the proxy to handle this request
- Proxy is able to load and to start the kernel side-by-side to Linux
  - Bare-metal execution[4]
  - Not part of this talk
- Proxy is also able to boot the application within a VM
  - No changes in the binary required
  - `HERMIT_ISLE` defines the NUMA node (bare-metal execution) or the kind of the VM

```
time HERMIT_ISLE=qemu ./hello
```

[4] S. Lankes, S. Pickartz, and J. Breitbart. "HermitCore – A Unikernel for Extreme Scale Computing". In: Proc. of the International Workshop on Runtime and Operating Systems for Supercomputers. 2016.

Why is the Start Time so High?

- View kernel messages to see the boot time of the kernel

```
time HERMIT_ISLE=qemu HERMIT_VERBOSE=1 ./hello
```

Why is the Start Time so High?

- View kernel messages to see the boot time of the kernel

```
time HERMIT_ISLE=qemu HERMIT_VERBOSE=1 ./hello
```

- Qemu needs too much time to initialize
  - a whole (virtual) PC,
  - KVM support,
  - an internal system monitor,
  - options to debug the system
  - ...

Why is the Start Time so High?

- View kernel messages to see the boot time of the kernel

```
time HERMIT_ISLE=qemu HERMIT_VERBOSE=1 ./hello
```

- Qemu needs too much time to initialize
  - a whole (virtual) PC,
  - KVM support,
  - an internal system monitor,
  - options to debug the system
  - ...
- Direct integration of the hypervisor into the proxy

```
time HERMIT_ISLE=uhyve HERMIT_VERBOSE=1 ./hello
```

Why is the Start Time so High?

- View kernel messages to see the boot time of the kernel

```
time HERMIT_ISLE=qemu HERMIT_VERBOSE=1 ./hello
```

- Qemu needs too much time to initialize
  - a whole (virtual) PC,
  - KVM support,
  - an internal system monitor,
  - options to debug the system
  - ...
- Direct integration of the hypervisor into the proxy

```
time HERMIT_ISLE=uhyve HERMIT_VERBOSE=1 ./hello
```

- Currently, a proof of concept

Echo Server Written in Go

```
func main() {
        http.HandleFunc("/", handler)
        log.Fatal(http.ListenAndServe(":8000", nil))
}

func handler(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintf(w, "%s %s %s\n", r.Method, r.URL, r.Proto)
        for k, v := range r.Header {
                fmt.Fprintf(w, "Header[%q] = %q\n", k, v)
        }
        fmt.Fprintf(w, "Host = %q\n", r.Host)
        fmt.Fprintf(w, "RemoteAddr = %q\n", r.RemoteAddr)
        if err := r.ParseForm(); err != nil {
                log.Print(err)
        }
        for k, v := range r.Form {
                fmt.Fprintf(w, "Form[%q] = %q\n", k, v)
        }
}
```

Support of compilers beside GCC

- Just avoid the standard environment (−ffreestanding)
- Set include path to HermitCore's toolchain
- Ensure 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
CC = icc -D__hermit__
CFLAGS = -O3 -xHost -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

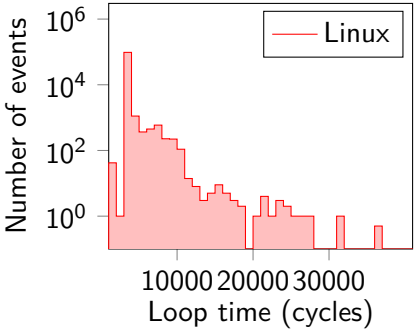
- Test system
  - Intel Haswell CPUs (E5-2650 v3) clocked at 2.3 GHz
  - 64 GiB DDR4 RAM and 25 MB L3 cache
  - SpeedStep Technology and TurboMode are deactivated
  - 4.2.5 Linux kernel on Fedora 23 (Workstation Edition)
  - gcc 5.3.x, AVX- & FMA-Support enabled (−mtune=native)
- Results in CPU cycles

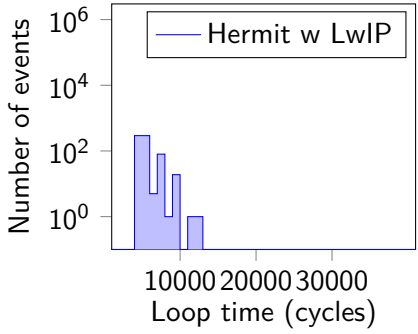
System activity	HermitCore	Linux
`getpid()`	14	143
`sched_yield()`	97	370
`write()`	3520	1079
`malloc()`	3772	6575
first write access to a page	2014	4007

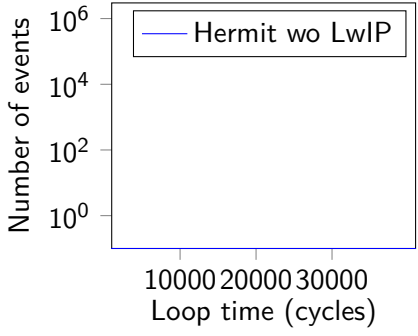
Hourglass Benchmark

- Benchmarks reads permanently the time step counter
- (Larger) Gaps ⇒ 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)

Conclusions

- Prototype works[5]
- 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!

[5] S. Lankes, S. Pickartz, and J. Breitbart. "HermitCore – A Unikernel for Extreme Scale Computing". In: Proc. of the International Workshop on Runtime and Operating Systems for Supercomputers. 2016.

Conclusion and Outlook

Thank you for your kind attention!

Stefan Lankes – 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