More or Less? A Discussion of the Abstraction Level of Future Operating Systems

1st International Workshop on Next-Generation Operating Systems for Cyber-Physical Systems (NGOSCPS)
Montreal, Canada
April 15th, 2019

Hendrik Borghorst
https://ess.cs.tu-dortmund.de/~hb
hendrik.borghorst@udo.edu

Michael Müller
https://ess.cs.uos.de/~mml
michael.mueller@uos.de

Olaf Spinczyk
https://ess.cs.uos.de/~os
olaf.spinczyk@uos.de
Motivation

Nvidia Xavier (8C ARMv8, Volta GPU),
https://en.wikichip.org/wiki/nvidia/tegra/xavier

Xilinx Zynq Ultrascale+ (4C ARMv8, 2C Cortex-R5, FPGA),

AMD Ryzen APU (4C/8T Zen, Vega GPU),
Future hardware

- **Heterogeneous** hardware
  - GPUs
  - FPGAs
  - AI accelerators
- Diverse input sensor arrays
- **Energy efficiency** for deep embedded systems
- More *distributed computing*
  - Complex many-core CPUs
  - Networked systems
Challenges for the OS

- Complex memory hierarchies
  - Most applications request memory on demand
- Unpredictable hardware (execution times)
  - Hardware threads interacting with each other (shared resources)
- Integration of heterogeneous hardware
  - Diversity of APIs (OpenCL, proprietary FPGAs, ...)
- High core count
  - Synchronization bottleneck, data movement, ...
- Reliability requirements
- Distribution
  - No central OS state
Challenges for the OS

- Complex memory hierarchies
  - Most applications request memory on demand
- Unpredictable hardware (execution times)
  - Hardware threads interacting with each other (shared resources)
- Integration of heterogeneous hardware
  - Diversity of APIs (OpenCL, proprietary FPGAs, ...)
- High core count
  - Synchronization bottleneck, data movement, ...
- Reliability requirements
- Distribution
  - No central OS state

How to tackle these?
Complex memory hierarchies

- Challenging to exploit
- Data placement with enormous impact on performance
- Inter-core interference
- Coherency bus traffic
  - e.g. false sharing
- Access latency (NUMA hops)

https://www.nextplatform.com/2017/06/20/competition-returns-x86-servers-epyc-fashion/
CyPhOS

- Component-based
  - Self-contained (data, bss, code)
  - Events & Triggers to connect
- Only **asynchronous** event handling
- Implicit synchronization (per component, monitor synchronization)

CyPhOS component-based structure [1]

Source code available at:
https://github.com/ESS-Group/CyPhOS
Hardware-assisted cache management

ARM *lockdown by way*:  
- Lockdown per cache way  
- Allows cache partitioning per core  
  - With planned preloading / writeback phases

Intel CAT®:  
- Cache divided in parts  
  - Cache ways?  
- Cache partitioning per thread (core)  
  - Parallel prefetching / writeback possible
Cache management

- CyPhOS integrates **central** cache management
- **No** unpredictable cache evictions & memory accesses
- OS taking control over cache content
- Reduces inter-core interference
- Performance & predictability improved
- Available for ARMv7 and AMD64

Execution times of synthetic memory load (on all cores) on Intel Xeon E5-1620 v4 [1]
Synchronization bottleneck

- Traditional locks perform bad on many-core CPUs
- Memory and bus contention
- Best solution is to avoid locks at all

→ How?

4000x3000 Mandelbrot calculation with mutexes (maximum achieved speedup: ~25%) [2]
Task-based OS

**MxKernel [2]**

- Short tasks with **run-to-completion** semantic
- Kernel aware of data objects
- Resource-aware scheduling
  - Reduces NUMA traffic
  - Reduces Cache coherence traffic

4000x3000 Mandelbrot calculation with tasks
(maximum achieved speedup: ~45%) [2]
Heterogeneous hardware

**GPUs:**
- OpenCL industry standard
- User-level libraries
- Not managed by OS
- No “first-class” citizen
- Lack of openness of GPUs (with exceptions)

**FPGAs:**
- No industry standard?
- Case-to-case integration of specific acceleration appl.
- No widespread use
  - Only in certain enterprise areas
Heterogeneous Hardware (cont.)

GPUs:
- PTasks [4]:
  - Deeper integration in OS
  - Data-flow orientated
  - Co-scheduling with CPU
- Gdev [5]:
  - Integration of GPUs in OS
  - Allows OS and appl. to use it

FPGAs:
- BORPH [6] and ReconOS[7]:
  - Integrates HW processes
  - Reconfigures FPGA parts on HW-process creation
Reliability

- SW-based replication for cheap CPS
  - Virtualization technology
  - e.g. HA via XEN (Remus)
- Transparent to applications
- Periodical checkpointing

Response times of a ping server, No checkpointing a), periodical checkpointing (b), c)) [3]
Self-determined Replication

- Replication point defined by applications
- Application’s knowledge used to reduce effort
- More stable & faster response time

Response times of a ping server, periodical checkpointing b), self-determined checkpointing [3]
Ideas for the future

• **Standardized** hardware modeling
  – OS aware of hardware it is running on

• New OS ↔ Application interfaces?:
  – Metadata interface
    • Memory requirements (Allows better NUMA/NUCA placement)
    • Compute requirements (CPU, GPU, FPGA, AI accelerator)
    • Timing requirements (Better scheduling, perhaps even static)
  – Enables OS to really **plan** resource usage **ahead**
    • Eliminates the use of heuristics
Ideas for the future

- **Standardized** hardware modeling
  - OS aware of hardware it is running on

- New OS ↔ Application interfaces?

  Applications should provide more knowledge down to the OS

  - Compute requirements (CPU, GPU, FPGA, AI accelerator)
    - Timing requirements (Better scheduling, perhaps even static)
  - Enables OS to really **plan** resource usage **ahead**
    - Eliminates the use of heuristics
Appeal

• Many good research ideas & projects land in oblivion
• OS community needs more focus on integration
  – e.g. DEADLINE scheduling in Linux
  – XEN project
• More open source research projects?!
  – Increased collaboration on these?!
References


