TECHNISCHE UNIVERSITÄT DRESDEN

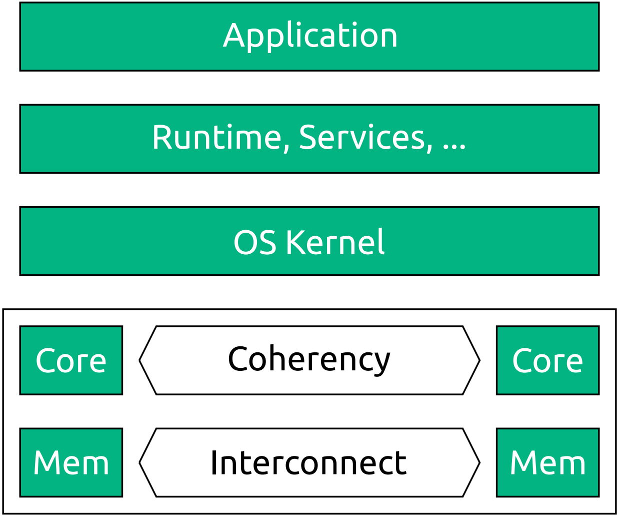
Faculty of Computer Science, Institute for System Architecture, Operating Systems Group

SCALABILITY AND HETEROGENEITY

Nils Asmussen

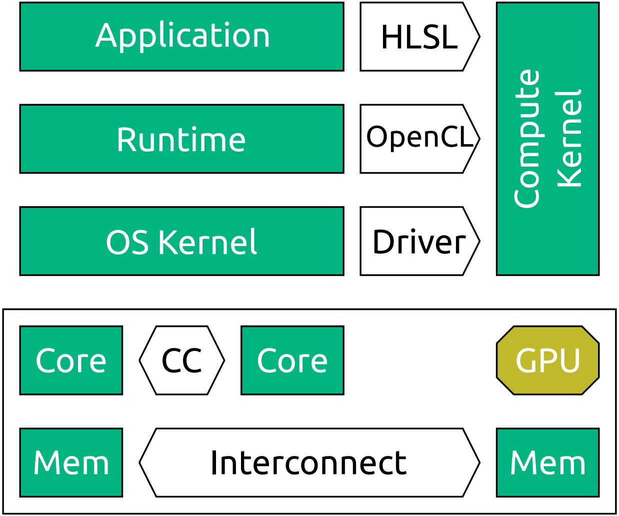
Dresden, 11/27/2018

Layers

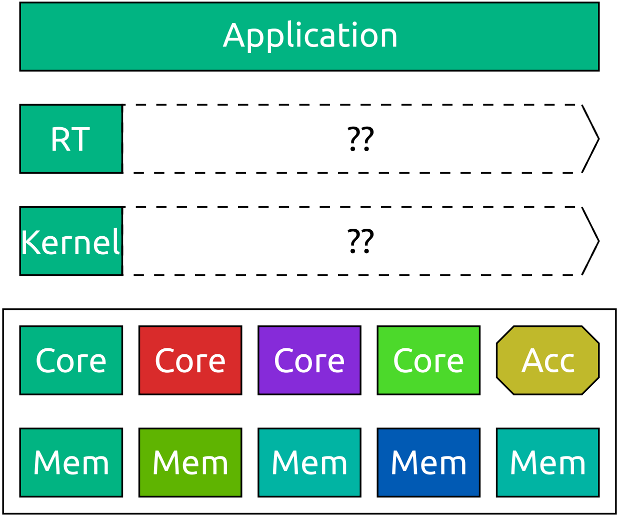

Application
Runtime, Services, ...
OS Kernel
Core
Coherency
Core
Mem
Interconnect
Mem

TECHNISCHE UNIVERSITÄT DRESDEN

Commodity System with GPU



Current Trend



Why?

- More cores can (for some usecases) deliver more performance
- Specialization is the next step
- Cache coherency gets more expensive (performance, complexity and energy) with more (and heterogeneous) cores

Commodity Hardware

Non-Uniform Memory Access

- Core-to-RAM distance differs
- Various interconnect topologies:
  bus, star, ring, mesh, . . .
- The good: all memory can be directly addressed
- The bad: different access latencies
- Consider placement of data

NUMA Machine

Measuring NUMA effects on:

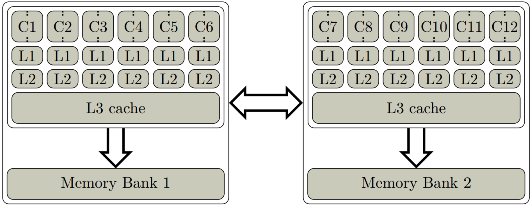


Figure 3.1: Dell Precision T7500 System Overview

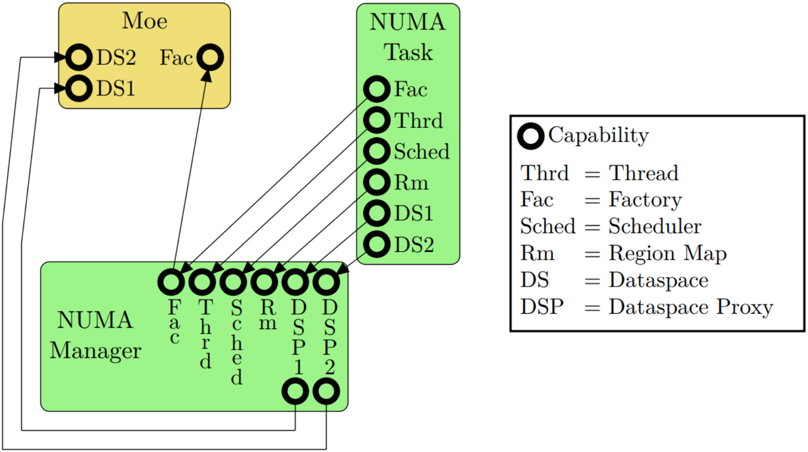
Daniel Müller: Memory and Thread Management on NUMA Systems, Diploma Thesis, 2013

NUMA Effects

Operation	Access	Time	NUMA Factor
read	local	37.420s	1.000
read	remote	53.223s	1.422
write	local	23.555s	1.000
write	remote	23.976s	1.018

Daniel Müller: Memory and Thread Management on NUMA Systems, Diploma Thesis, 2013

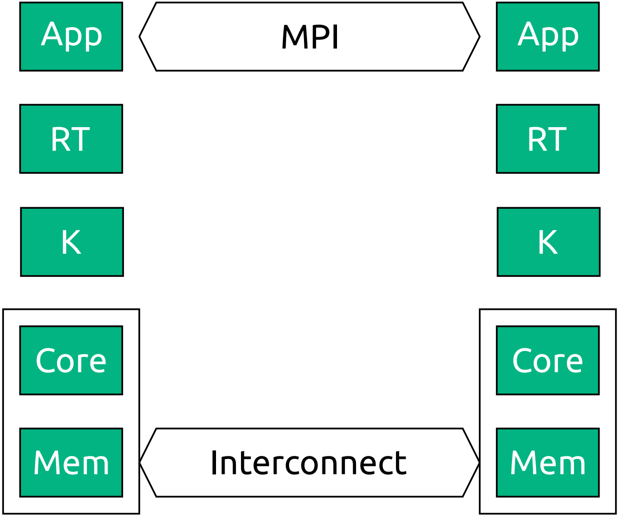
NUMA Mechanisms



Daniel Müller: Memory and Thread Management on NUMA Systems, Diploma Thesis, 2013

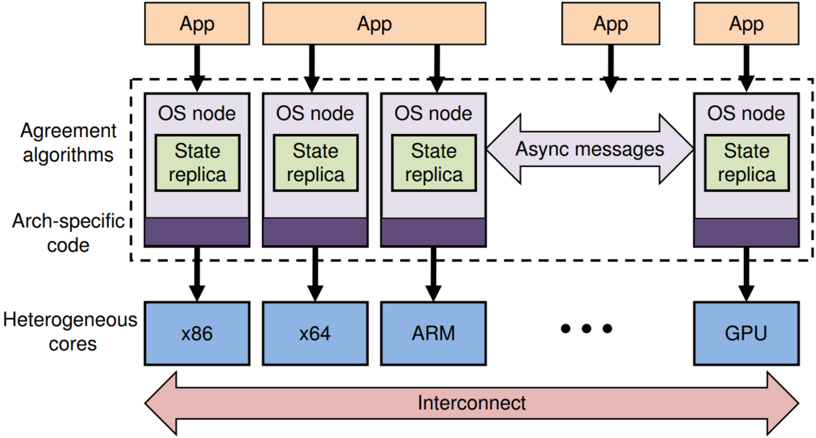
NUMA Policies

- fundamental options:
  migrate thread vs. migrate data
- use performance counters to decide
- dynamic management shows $> 10\%$ performance benefit compared to best static placement


App
MPI
App
RT
RT
K
K
Core
Core
Mem
Interconnect
Mem

Research Prototypes

Barrelfish



Andrew Baumann et al.: The Multikernel: A new OS architecture for scalable multicore systems, SOSP 2009

Barrelfish

- Concept: multikernel,
  implementation: barrelfish
- Treat the machine as cores with a network
- “CPU driver” plus exokernel-ish structure
- No inter-core sharing at the lower levels
- Monitors coordinate system-wide state via replication and synchronization

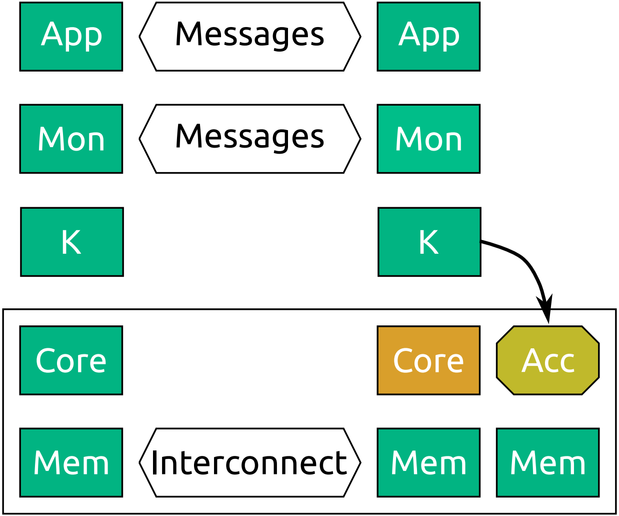
Andrew Baumann et al.: The Multikernel: A new OS architecture for scalable multicore systems, SOSP 2009

Cosh

- Based on Barrelfish
- Introduces abstractions for non-CC systems
- Takes advantage of CC, if possible
- Otherwise, data transfers via, e.g., DMA units
- Used to implement OS services (net, fs, . . .)
- Evaluated for Intel i7 CPU + Intel Knights Ferry

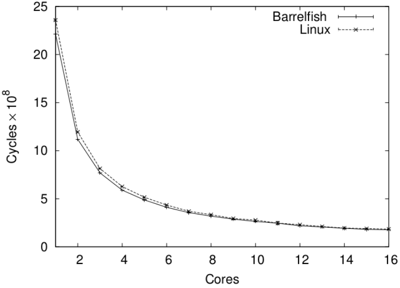
Andrew Baumann et al.: Cosh: clear OS data sharing in an incoherent world, TRIOS 2014

Barrelfish + Cosh



Barrelfish Scalability

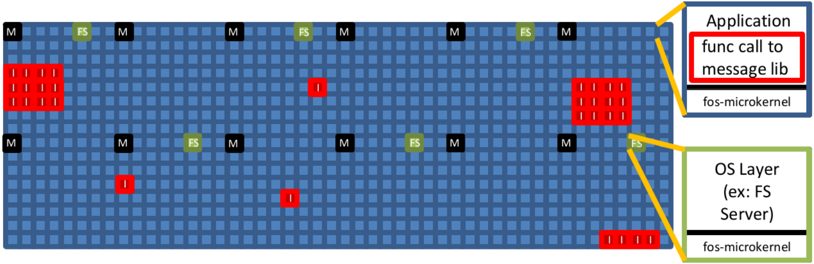
- Driven by scalability issues of shared kernel designs and cache coherence
- This might not be a pressing issue today



(d) SPLASH-2 Barnes-Hut

Andrew Baumann et al.: The Multikernel: A new OS architecture for scalable multicore systems, SOSP 2009

Factored Operating System



- Idle Processor Core
- Application
- Fleet of File System Servers
- Fleet of Physical Memory Allocation Servers

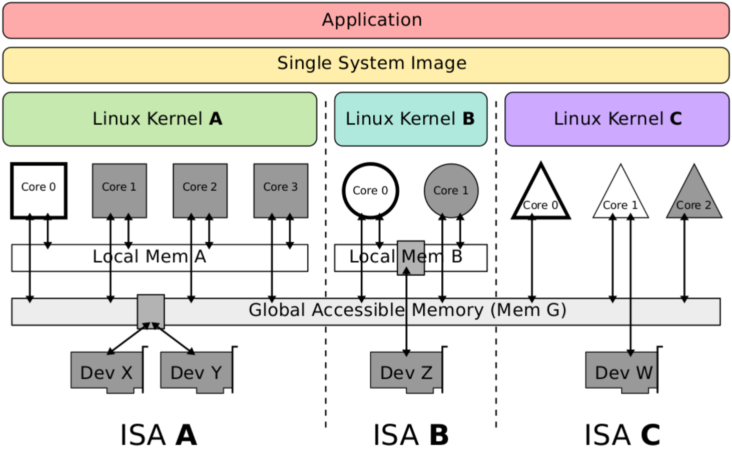
David Wentzlaff, Anant Agarwal: Factored Operating Systems (fos): The Case for a Scalable Operating System for Multicores, SIGOPS OSR 2009

Popcorn Linux

- Idea: multiple Linux's on one system
- Provide the illusion of an POSIX SMP system
- Kernels communicate to sync/exchange state
- Does not rely on global shared memory
- Distributed shared memory, if necessary
- Processes can migrate between kernels

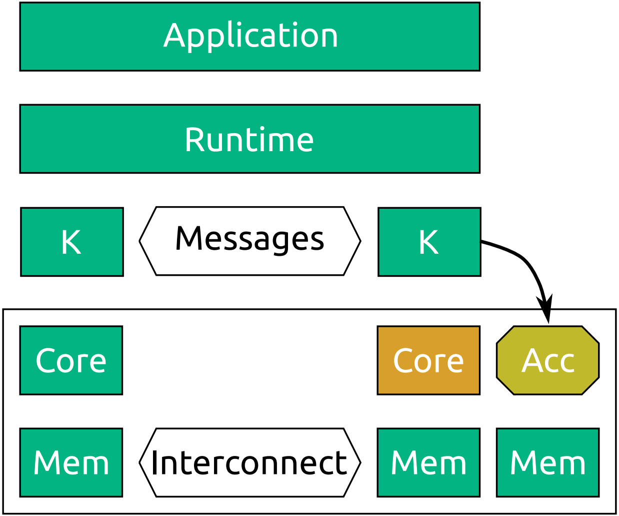
Barbalace et al.: Popcorn: Bridging the Programmability Gap in Heterogeneous-ISA Platforms, EuroSys 2015

Popcorn Linux



Barbalace et al.: Popcorn: Bridging the Programmability Gap in Heterogeneous-ISA Platforms, EuroSys 2015

Popcorn Linux



Helios

- Idea: heterogeneous ISA systems need some kind of compiler support
- ISA-specific kernels: “satellite kernels”
- Provide uniform OS abstractions
- Memory management, scheduling
- Bootstrap: first kernel becomes coordinator, boots other cores

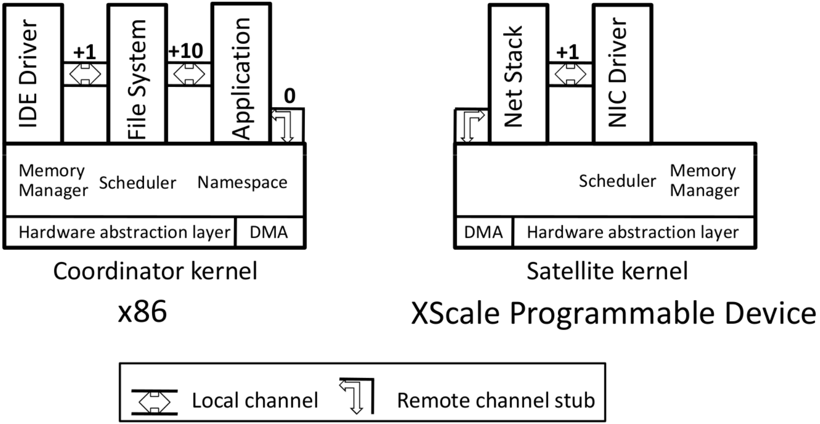
Nightingale et al.: Helios: Heterogeneous Multiprocessing with Satellite Kernels, SOSP 2009

Helios

- Share-nothing, even on ccNUMA
- Processes cannot span across kernels
- Implementation based on Singularity
- Applications compiled into intermediate code
- 2nd stage compilation to native code of all available ISAs at install time
- Placement based on affinity hints

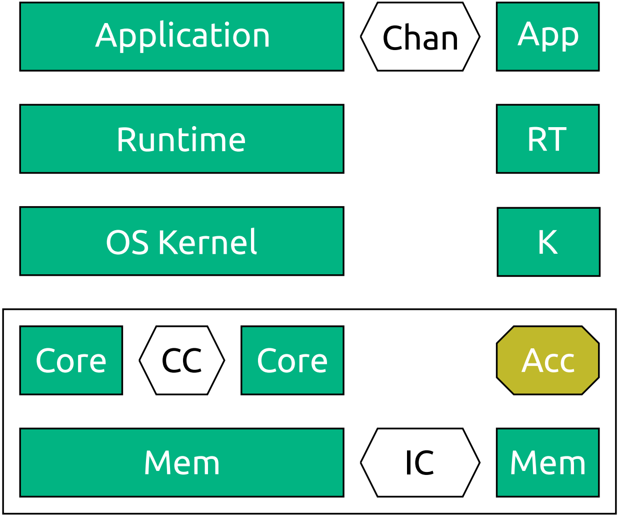
Nightingale et al.: Helios: Heterogeneous Multiprocessing with Satellite Kernels, SOSP 2009

Helios



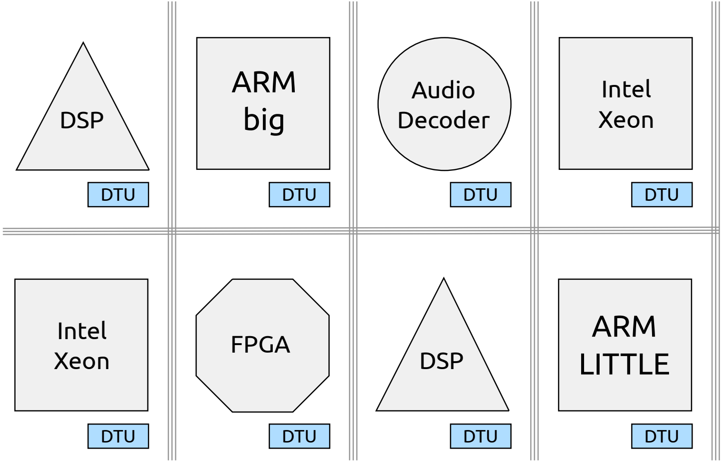
Nightingale et al.: Helios: Heterogeneous Multiprocessing with Satellite Kernels, SOSP 2009

Helios

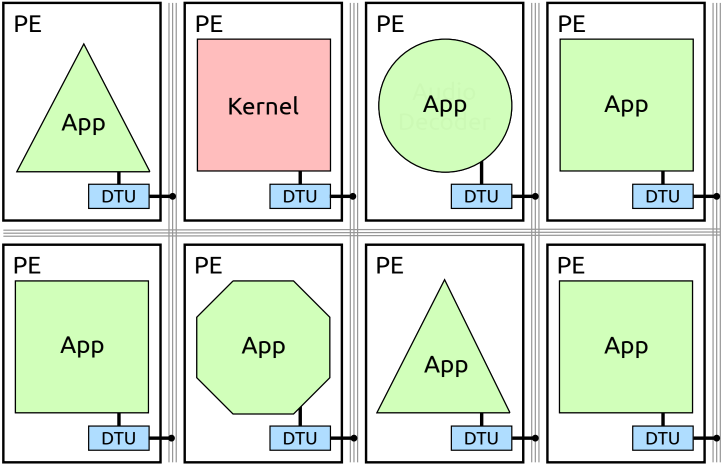

Application
Chan
App
Runtime
RT
OS Kernel
K
Core
CC
Core
Acc
Mem
IC
Mem

Our Own Work

M³ Approach – Hardware



M³ Approach – Software

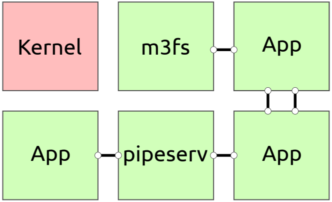


Data Transfer Unit

- Supports memory access and message passing
- Provides a number of endpoints
- Each endpoint can be configured for:
  1. Accessing memory (contiguous range, byte granular)
  2. Receiving messages into a ringbuffer
  3. Sending messages to a receiving endpoint
- Configuration only by kernel, usage by application
- Direct reply on received messages

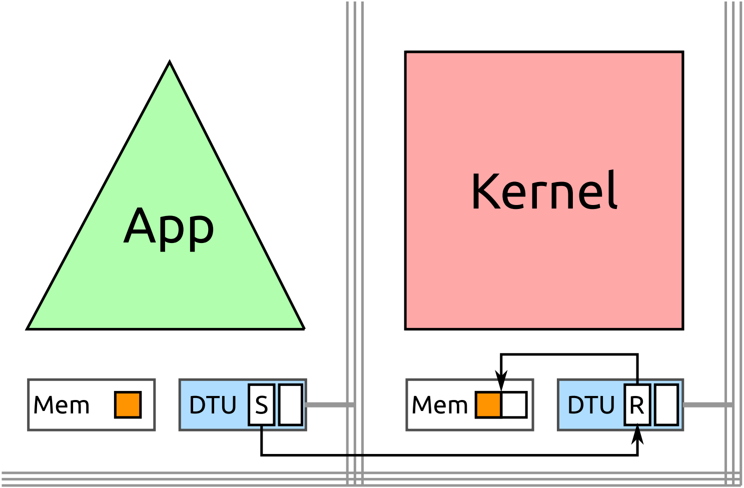
M^3 = L4 ± 1

- M^3: **M**icrokernel-based syste**m** for het. **m**anycores (or L4 ± 1)
- Implemented from scratch
- Drivers, filesystems, ... are implemented on top



- Kernel manages permissions, using capabilities
- DTU enforces permissions (communication, memory access)
- Kernel is independent of other CUs in the system

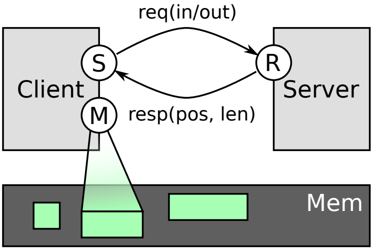
M^3 System Call



Virtual PEs

- M^3 kernel manages user PEs in terms of VPEs
- VPE is combination of a process and a thread
- VPE creation yields a VPE cap. and memory cap.
- Library provides primitives like `fork` and `exec`
- VPEs are used for *all* PEs:
  - Accelerators are not handled differently by the kernel
  - All VPEs can perform system calls
  - All VPEs can have time slices and priorities
  - ...

File Protocol



- Used for all file-like objects
- Simple for accelerators, yet flexible for software
- Software uses POSIX-like API on top of the protocol
- Server configures client's memory endpoint
- Client accesses data via DTU, without involving others
- `req(in)` requests next input piece
- `req(out)` requests next output piece
- Receiving `resp(n, 0)` indicates EOF

Example Implementations

Filesystem

- *m3fs* is an in-memory file system
- m3fs organizes the file's data in extents
- Extent is contiguous region defined by start and length
- `req(in/out)` configures memory endpoint to next extent

TECHNISCHE UNIVERSITÄT DRESDEN

Example Implementations

Filesystem

- *m3fs* is an in-memory file system
- m3fs organizes the file's data in extents
- Extent is contiguous region defined by start and length
- `req(in/out)` configures memory endpoint to next extent

Pipe

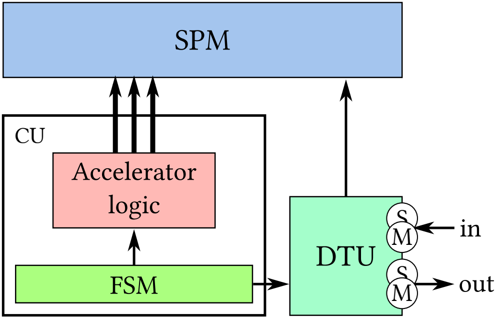
- M^3 provides a server that offers UNIX-like pipes
- Data is exchanged via shared memory area
- Client's memory EP is configured once for SHM
- Pipe server tells clients read/write positions within SHM area

Accel. Example: Stream Processing

- Accelerator works on scratchpad memory
- Input data needs to be loaded into scratchpad
- Result needs to be stored elsewhere

Accel. Example: Stream Processing

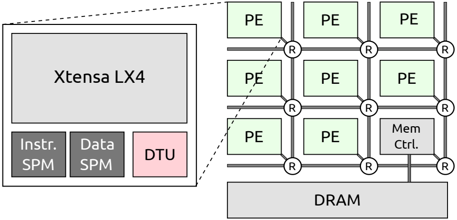
- Accelerator works on scratchpad memory
- Input data needs to be loaded into scratchpad
- Result needs to be stored elsewhere

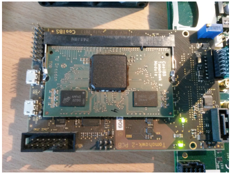


Shell Integration

- M^3 allows to use accelerators from the shell:
  `preproc | accel1 | accel2 > output.dat`
- Shell connects the EPs according to stdin/stdout
- Accelerators work autonomously afterwards
- Requires about 30 additional lines in the shell

Prototype Platforms – Tomahawk





- Cores attached to NoC with DTU
- No privileged mode
- No MMU, no caches, but SPM
- Only simple DTU + SW emulation

TECHNISCHE UNIVERSITÄT DRESDEN

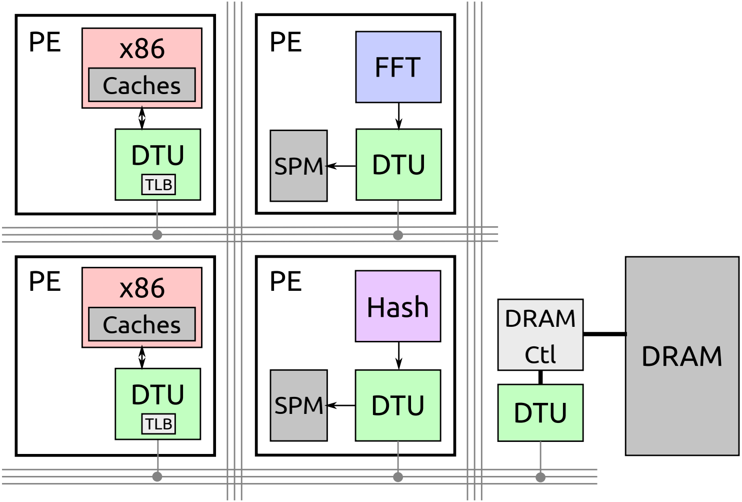
Prototype Platforms – Linux

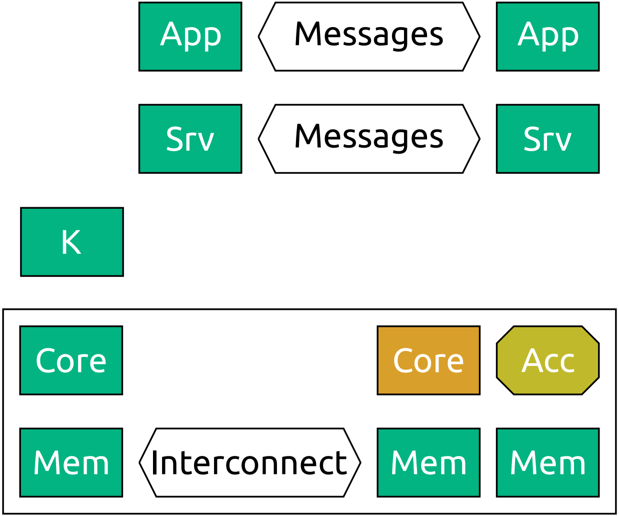
- M^3 runs on Linux using it as a virtual machine
- A process simulates a PE, having two threads (CPU + DTU)
- DTUs communicate over UNIX domain sockets
- No accuracy because
  - Programs are directly executed on host
  - Data transfers have huge overhead compared to HW
- Very useful for debugging and early prototyping

Prototype Platforms – gem5

- Modular platform for computer-system architecture research
- Supports various ISAs (x86, ARM, Alpha, . . .)
- Cycle-accurate simulation
- Has an out-of-order CPU model
- We built a DTU for gem5
- Support for caches and virtual memory

gem5 – Example Configuration




App
Messages
App
Srv
Messages
Srv
K
Core
Core
Acc
Mem
Interconnect
Mem
Mem

Summary and Outlook

- Various different approaches
- Not clear yet how to handle heterogeneity
- Memory will get heterogeneous as well (NVM)
- Reconfigurable hardware will emerge