

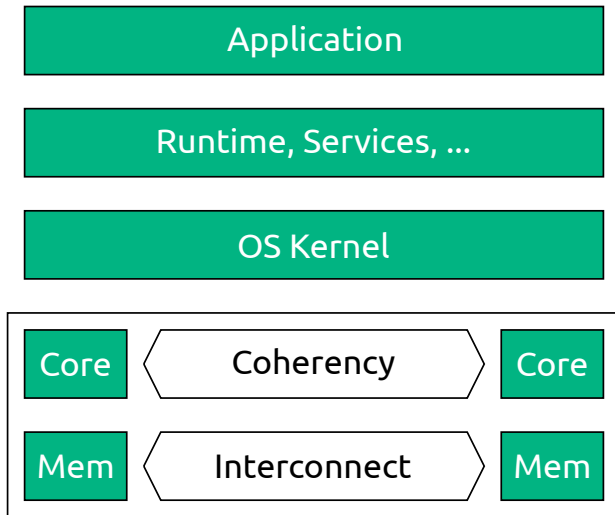


**TECHNISCHE
UNIVERSITÄT
DRESDEN**

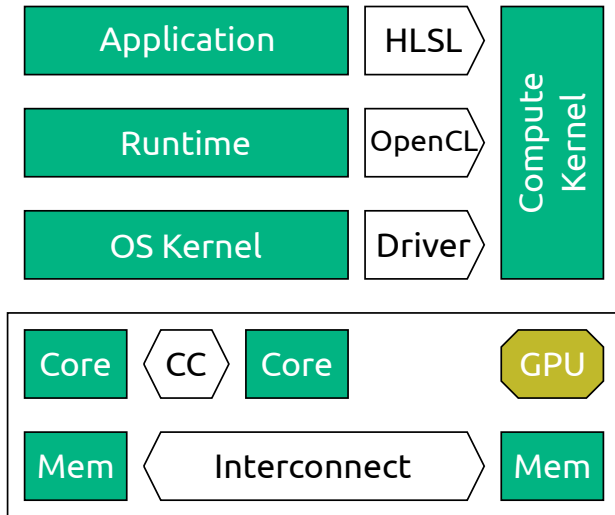
Faculty of Computer Science, Institute for System Architecture, Operating Systems Group

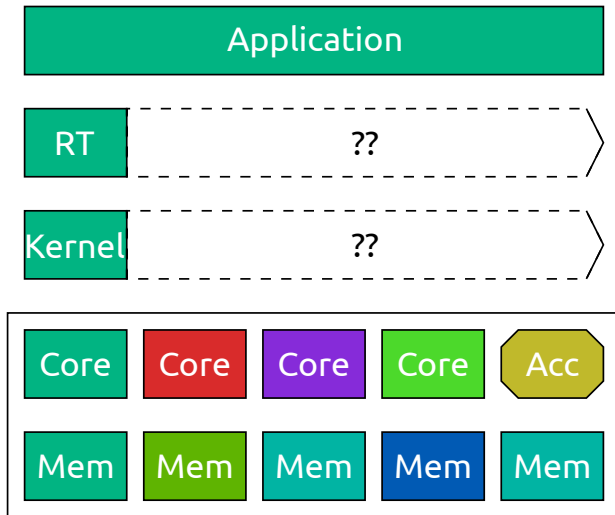
SCALABILITY AND HETEROGENEITY

Nils Asmussen



Commodity System with GPU





- 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

Measuring NUMA effects on:

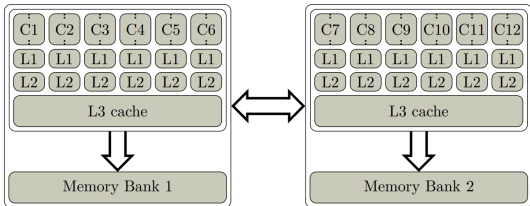
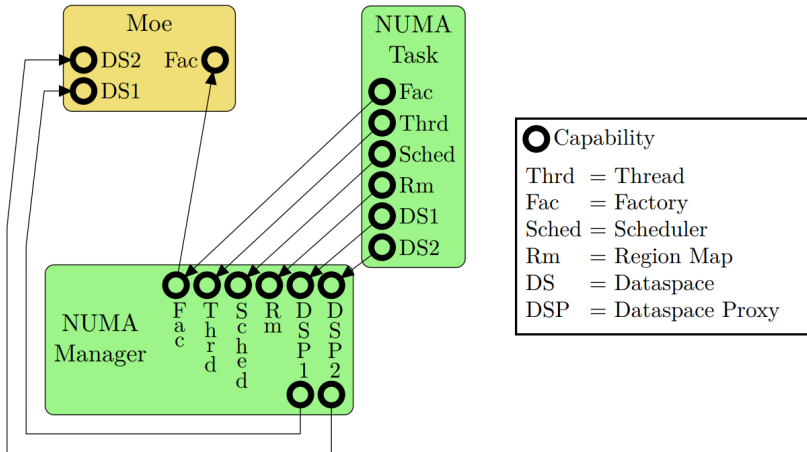
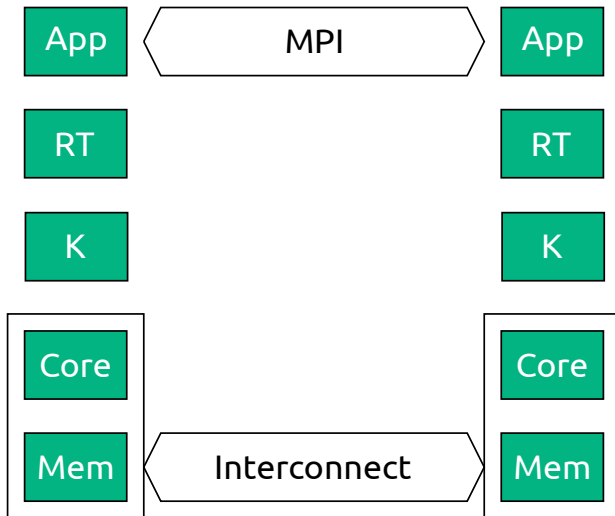


Figure 3.1: Dell Precision T7500 System Overview

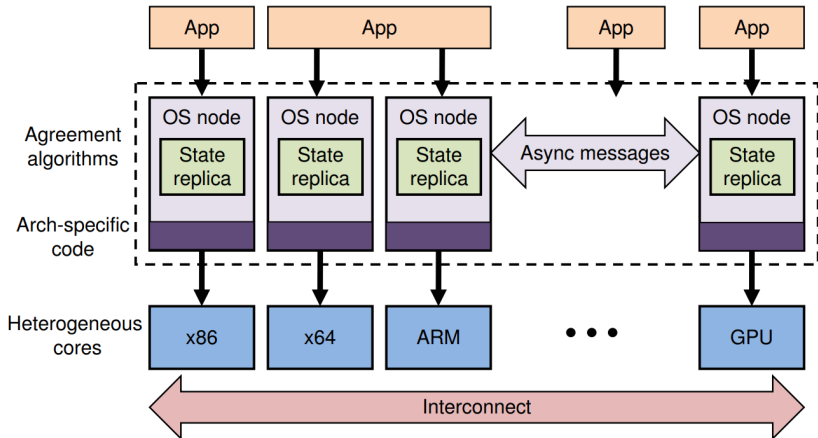
Operation	Access	Time	NUMA Factor
read	local	37.420s	1.000
read	remote	53.223s	1.422
write	local	23.555s	1.000
write	remote	23.976s	1.018



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

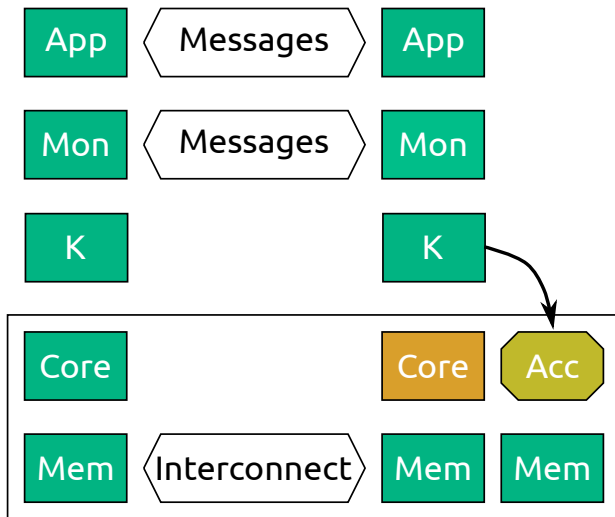


Research Prototypes

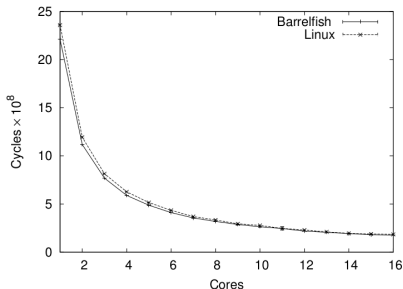


- Concept: multikernel,
implementation: barrelfish
- Treat the machine as cores with a network
- “CPU driver” plus exokernel-ish structure
- No inter-core sharing at the lower levels
- Monitors coordinate system-wide state via replication and synchronization

- 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

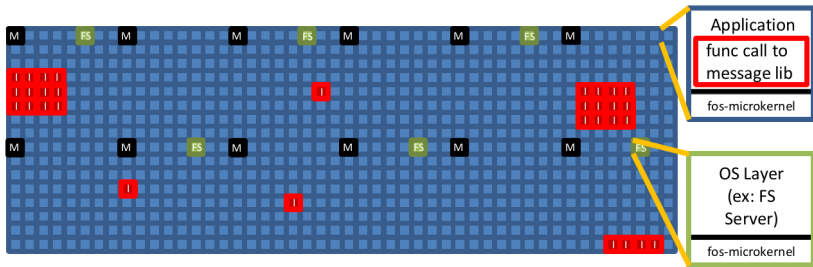


- Driven by scalability issues of shared kernel designs and cache coherence
- This might not be a pressing issue today



(d) SPLASH-2 Barnes-Hut

Factored Operating System



I - Idle Processor Core

■ - Application

FS FS ... FS - Fleet of File System Servers

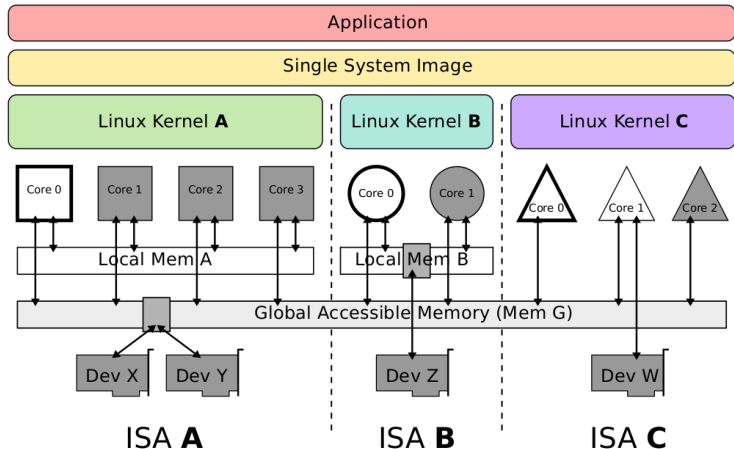
M M ... M - Fleet of Physical Memory Allocation Servers

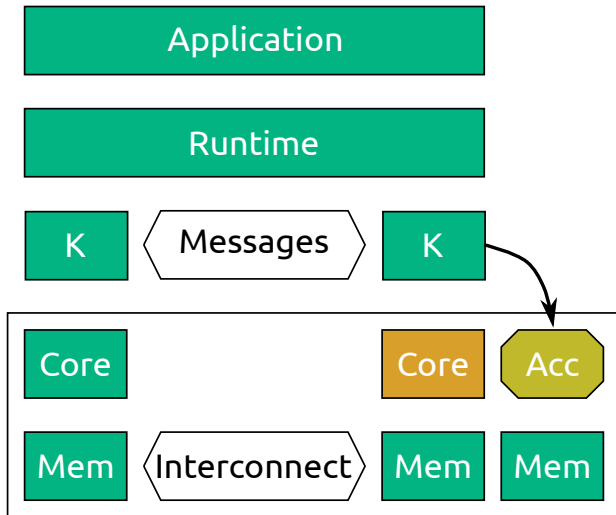
David Wentzlaff, Anant Agarwal: Factored Operating Systems (fos): The Case for a Scalable Operating System for Multicores, SIGOPS OSR 2009

Scalability and Heterogeneity

Slide 19 of 42

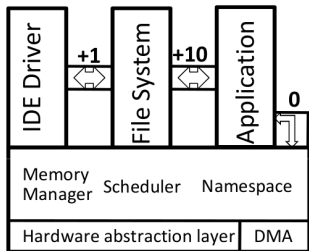
- Idea: multiple Linux's on one system
- Provide the illusion of an POSIX SMP system
- Kernels communicate to sync/exchange state
- Does not rely on global shared memory
- Distributed shared memory, if necessary
- Processes can migrate between kernels





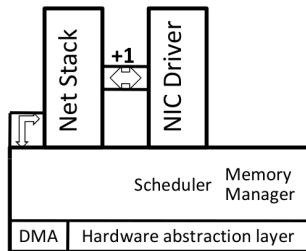
- Idea: heterogeneous ISA systems need some kind of compiler support
- ISA-specific kernels: “satellite kernels”
- Provide uniform OS abstractions
- Memory management, scheduling
- Bootstrap: first kernel becomes coordinator, boots other cores

- 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



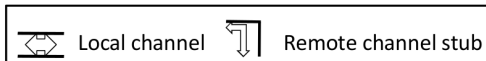
Coordinator kernel

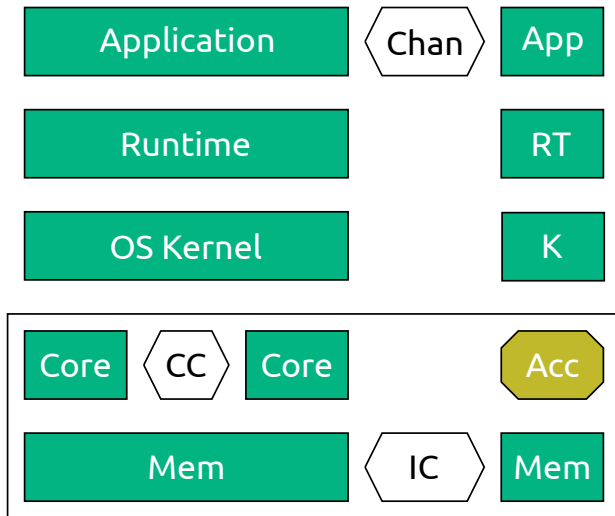
x86



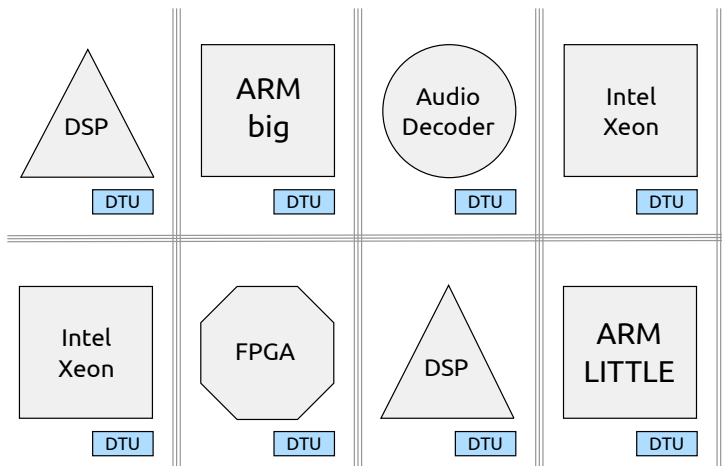
Satellite kernel

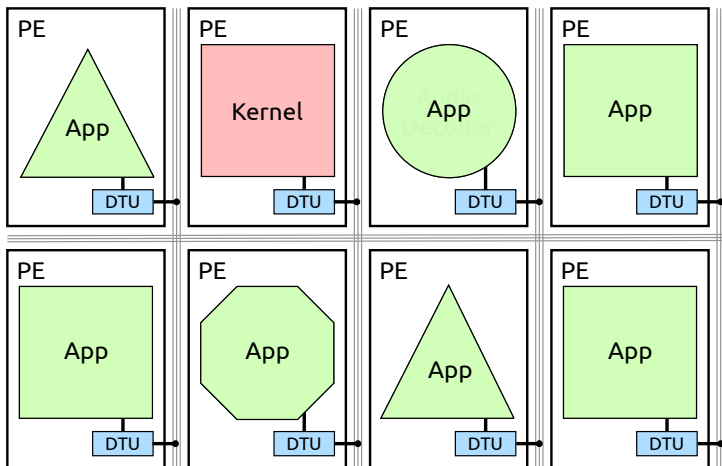
XScale Programmable Device





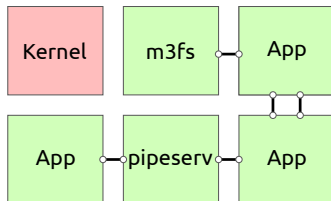
Our Own Work

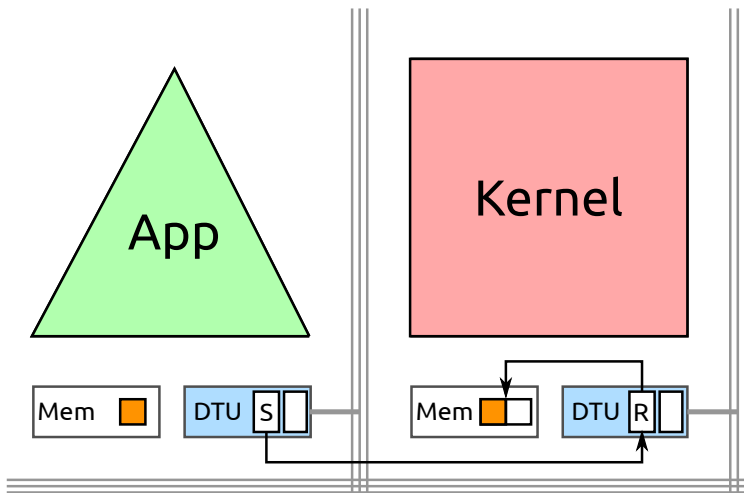




- Supports memory access and message passing
- Provides a number of endpoints
- Each endpoint can be configured for:
 - ➊ Accessing memory (contiguous range, byte granular)
 - ➋ Receiving messages into a ringbuffer
 - ➌ Sending messages to a receiving endpoint
- Configuration only by kernel, usage by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

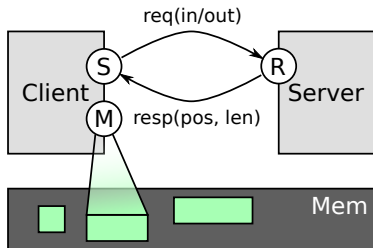
- M^3 : **Microkernel-based system** for het. **manycores** (or $L4 \pm 1$)
- Implemented from scratch
- Drivers, filesystems, ... are implemented on top
- Kernel manages permissions, using capabilities
- DTU enforces permissions (communication, memory access)
- Kernel is independent of other CUs in the system





- M³ 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
 - ...

- 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



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

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

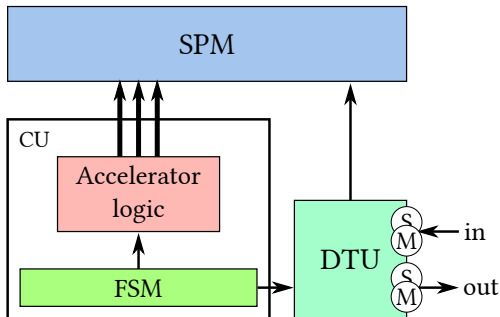
- M^3 provides a server that offers UNIX-like pipes
- Data is exchanged via shared memory area
- Client's memory EP is configured once for SHM
- Pipe server tells clients read/write positions within SHM area

Accel. Example: Stream Processing

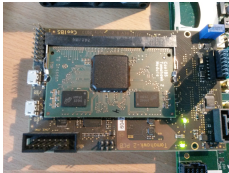
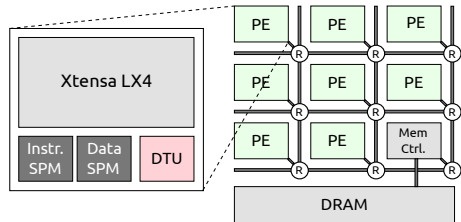
- Accelerator works on scratchpad memory
- Input data needs to be loaded into scratchpad
- Result needs to be stored elsewhere

Accel. Example: Stream Processing

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



- 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

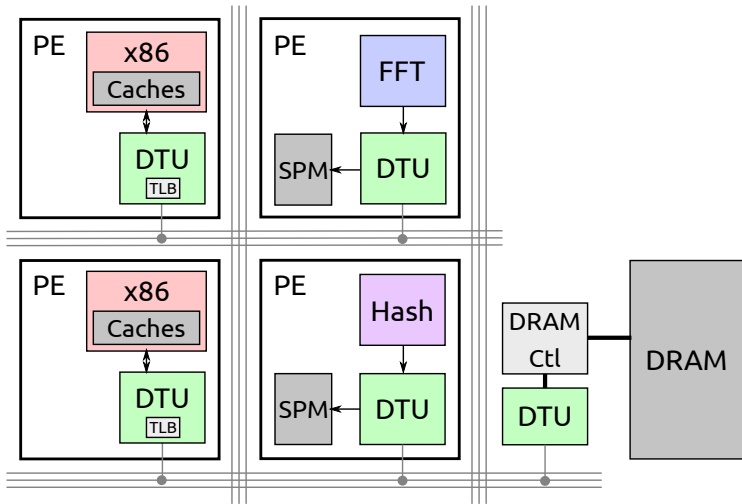


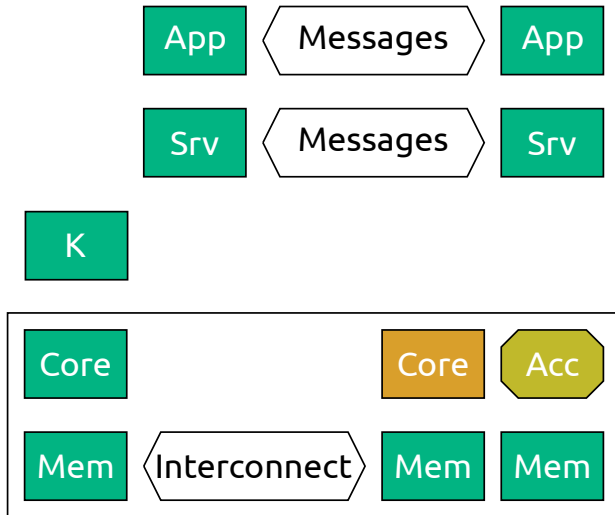
- Cores attached to NoC with DTU
- No privileged mode
- No MMU, no caches, but SPM
- Only simple DTU + SW emulation

- M³ 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

- 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





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