SCALABILITY AND HETEROGENEITY

Nils Asmussen

Dresden, 11/27/2018
Layers

Application

Runtime, Services, ...

OS Kernel

Core

Coherency

Interconnect

Core

Mem

Mem
Commodity System with GPU

Application

Runtime

OS Kernel

Core Core

Mem

Interconnect

Application

HLSL

Compute Kernel

Runtime

OpenCL

Driver

OS Kernel

Core CC Core

GPU

Core

CC

Core

Mem

Interconnect

Mem

Interconnect

Mem
Current Trend

Application

RT ??

Kernel ??

Core Core Core Core Acc
Mem Mem Mem Mem Mem
• More cores can (for some use cases) 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:

![Diagram of NUMA machine with labels C1 to C12, L1 to L2, L3 cache, Memory Bank 1, Memory Bank 2.]

Figure 3.1: Dell Precision T7500 System Overview
<table>
<thead>
<tr>
<th>Operation</th>
<th>Access</th>
<th>Time</th>
<th>NUMA Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>read</td>
<td>local</td>
<td>37.420s</td>
<td>1.000</td>
</tr>
<tr>
<td>read</td>
<td>remote</td>
<td>53.223s</td>
<td>1.422</td>
</tr>
<tr>
<td>write</td>
<td>local</td>
<td>23.555s</td>
<td>1.000</td>
</tr>
<tr>
<td>write</td>
<td>remote</td>
<td>23.976s</td>
<td>1.018</td>
</tr>
</tbody>
</table>
• 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 Barrelish
• 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
Factored Operating System

Application func call to message lib
fos-microkernel

OS Layer (ex: FS Server)
fos-microkernel

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


Scalability and Heterogeneity
• 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
Popcorn Linux


Scalability and Heterogeneity
Popcorn Linux

Application

Runtime

Messages

Core

Acc

Mem

Interconnect
• 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
Application

Runtime

OS Kernel

Core

CC

Core

Mem

IC

Mem

Chan

App

RT

K

Acc

Scalability and Heterogeneity

Slide 26 of 42
Our Own Work
M³ Approach – Hardware
M³ Approach – Software

Scalability and Heterogeneity  Slide 28 of 42
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
• **M³**: Microkernel-based system for het. manycores (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
M³ System Call

Scalability and Heterogeneity
• $M^3$ 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
  – …
- 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
- \texttt{req(in)} requests next input piece
- \texttt{req(out)} requests next output piece
- Receiving \texttt{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
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**
- *M³* 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
• Accelerator works on scratchpad memory
• Input data needs to be loaded into scratchpad
• Result needs to be stored elsewhere
• Accelerator works on scratchpad memory
• Input data needs to be loaded into scratchpad
• Result needs to be stored elsewhere
- $\text{M}^3$ allows to use accelerators from the shell:
  \[
  \text{preproc} \ | \ \text{accel1} \ | \ \text{accel2} > \text{output.dat}
  \]
- 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
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

Scalability and Heterogeneity
Summary and Outlook

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