SCALABILITY AND HETEROGENEITY

Nils Asmussen

Dresden, 12/05/2017
Layers

Application

Runtime, Services, ...

OS Kernel

Core Core

Mem Mem

Coherency

Interconnect

Core Mem
Commodity System with GPU
Current Trend

Scalability and Heterogeneity Slide 4 of 39
• 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
<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>
NUMA Mechanisms

- Moe
- DS2
- DS1

NUMA Task
- Fac
- Thrd
- Sched
- Rm
- DS1
- DS2

NUMA Manager
- Fac
- Thrd
- Sched
- Rm
- DS1
- DS2

Capability
- Thrd = Thread
- Fac = Factory
- Sched = Scheduler
- Rm = Region Map
- DS = Dataspase
- DSP = Dataspase Proxy


Scalability and Heterogeneity
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
Scalability and Heterogeneity
Research Prototypes
Barrelfish


Scalability and Heterogeneity

Slide 14 of 39
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

- 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

App

Messages

App

Mon

Messages

Mon

K

Core

Interconnect

Core

Acc

Mem

Mem

Mem
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
Factored Operating System

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

- 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

Application

Runtime

K Messages K

Core

Mem

Interconnect

Core

Acc

Mem

Mem
- 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

Local channel

Remote channel stub
Our Own Work
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
- Microkernel-based system for het. manycores
- Implemented from scratch
- Mechanisms for PEs, memory and communication
- Drivers, filesystems, … are implemented on top
- Kernel manages permissions, using capabilities
- DTU enforces permissions (communication, memory access)
- Kernel is independent of other cores in the system
• Creating VPE yields a VPE cap. and memory cap.
• Library provides primitives like `fork` and `exec`
• Creating VPE yields a VPE cap. and memory cap.
• Library provides primitives like `fork` and `exec`.

### Execute function on different PE

```cpp
VPE vpe("test");
vpe.run_async([]() {
    Serial::get() << "Hello World!\n";
    return 0;
});
int exitcode = vpe.wait();
```
- FS service is implemented outside of kernel
- m3fs is (currently) an in-memory filesystem
• FS service is implemented outside of kernel
• m3fs is (currently) an in-memory filesystem
• FS service is implemented outside of kernel
• m3fs is (currently) an in-memory filesystem
• FS service is implemented outside of kernel
• m3fs is (currently) an in-memory filesystem
• FS service is implemented outside of kernel
• m3fs is (currently) an in-memory filesystem
- Cores attached to NoC with DTU
- No privileged mode
- No MMU, no caches, but SPM
- Only simple DTU + SW emulation
• $M^3$ 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
• 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