Heterogeneity, memory hierarchies, availability – challenges of manycore systems

Frank Feinbube, M.Sc.
Operating Systems and Middleware
Hasso-Plattner-Institut at the University of Potsdam
Frank.Feinbube@hpi.uni-potsdam.de
Agenda

- Paradigm shift – Introduction, Motivation
- Research at OSM group
  - Hybrid Computing
  - Intel SCC
  - Predictable VM Migration
- Service-Oriented Systems Engineering
  - FutureSOC Lab @ HPI
- Conclusions
Service-oriented Systems Engineering represents a symbiosis of best practices in:
- Component-based development,
- Distributed computing, and
- Business process management

Interconnects the HPI research groups
- 28 PhD students, 2 PostDoc
- 6 PhD students at Cape Town Univ.
- 10 PhD students at Technion, Haifa
- Growing success (publications, workshops/conferences/symposia)

http://kolleg.hpi.uni-potsdam.de
Operating systems and Middleware – the group

Prof. Dr. rer. nat. habil. Andreas Polze
Dr. Martin von Löwis
Dr. Peter Tröger
Dipl.-Inf. Bernhard Rabe
Dipl.-Inf. Alexander Schmidt
M.Sc. Frank Feinbube
M.Sc. Uwe Hentschel
M.Sc. Jan-Arne Sobania
M.Sc. Robert Wierschke
Dipl.-Inf. Alexander Schacht
Dipl.-Inf. Theodor Heinze
Dipl.-Inf. Christian Neuhaus
M.Sc. Fahad Khalid
Sabine Wagner
Extending Middleware for Predictable Computing
- Paradigms, patterns, implementation strategies
- Real-Time, Security, Fault-Tolerance

Wide-Area Distributed Systems
- Predictable Behavior in heterogeneous distributed systems
- Service-Oriented Grid-Computing Environments
- Cloud Computing

Embedded Systems
- Composite Objects - interconnecting CORBA middleware and embedded systems
- Analytic Redundancy and online replacement
- Dynamic (Re-) Configuration of component-based systems

Operating System Support for Resource Management
- Windows Research Kernel (WRK), NTrace
- SkyLab / Instant Lab – OS research in the Cloud
- Fault-tolerant Hypervisor
Changes in Computer Architecture have implications on (software) systems

Multi-Core, Many-Core Computing
  • Programming Models
  • Energy consumption

Predictability, Availability
  • Fault-tolerance, Fault-injection
  • Virtualization, VM migration

In-Memory Computing
  • Data organization, redundancy
  • Memory hierarchy
Hybrid Computing

Flavors of massively multi-core systems:

- Radically new architectures under evaluation (Intel SCC)
- Accelerators accompany general purpose CPUs (Hybrid Systems)

Hybrid Systems

- **GPU Compute Devices**
  High Performance Computing
  (3 out of top 5 supercomputers are GPU-based!)
  - Business Servers,
  - Home/Desktop Computers,
  - Mobile and Embedded Systems

- **Special-Purpose Accelerators**
  - (de)compression,
  - XML parsing,
  - (en|de)cryption, regular expression matching
GPU Computing – performance gains at a price

Difficult to program:
- Asynchronous operation
- Little memory
- Conflicting accesses
- Branch prediction
- Vector instructions

**AMD** - R700, R800, R900

**NVIDIA** - G80, G92, GT200, GF100, GF110
Geforce, Quadro, Tesla, ION

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
GPU Computing today: Open Compute Language (OpenCL)

Programmer must hand in specific, non-portable C source code...

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
Our Approach:
Parallel.ForGPU - .NET language integration

How to integrate GPU computing with C#/.NET?
- Parallel computing library for .NET
- Executed on OpenCL-enabled GPUs
- Built upon existing .NET 4.0 concepts (Parallel.For)

```csharp
int[] data = ... // Get vector
Action<int> action = delegate(int i)

    data[i] = data[i]+1;

}

// Task Parallel Library
System.Threading.Tasks.Parallel.For(0, 100, action);

// Our Library
Hybrid.Parallel.For(0, 100, action);
Hybrid.Parallel.For(0, 100, 0, 100, action2D);
```

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
Proof-of-Concept: Parallel.ForGPU and UIUC micro-benchmarks

- Various light-weight problems from the parallel computing domain
- C implementation
  - OpenMP examples from the Universal Parallel Computing Research Center Summer School at UIUC
  - .NET implementation with Parallel.For
  - unoptimized execution via Parallel.ForGPU
- CPU: Intel Xeon E8500: 2 cores, 3.17 Ghz
- GPU: Geforce GTX 275: 30 cores, 1.4 Ghz

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
Hybrid Computing – Lessons learned

- GPU Computing will become tomorrow's SIMD supercomputer
- Huge product diversity
  - Notion of "a standard GPU architecture" not yet established
  - Most GPU-code today is written from scratch
- Need to refactor existing code
  - Source code transformation
  - Binary transformation
- Changing memory hierarchies remain major obstacle
  - No hardware support for consistency protocols
  - Limited amounts of memory

Goal: Ability to run same code on GPU and CPU
Intel Single-Chip Cloud Computer (SCC): a radically new architecture

- 24 tiles with two IA32 cores per tile
- A 24-router mesh network with 256GB bisection bandwidth
- 4 integrated DDR3 memory controllers
- Hardware support for message passing
- Power management:
  24 frequency islands, 6 voltage islands
- No hardware cache-coherency
SCC Tile Architecture

SCC has 24 dual-core tiles interconnected by mesh network.
Dynamically reconfigurable memory layout, no memory coherency protocols

- Original Pentium: Physical Address would be seen on address bus
- On SCC: Physical addresses mapped to **System addresses** via LookUp Table (LUT)

- No tile-local memory
  - Memory accesses via interconnection network
- 8-bit LUT index translates to:
  - 1-bit bypass
  - 8-bit route (msg tile)
  - 3-bit destination (mem ctrl)
  - 10-bit address extension

- Up to 34 bit (16GB) accessible per memory controller
SCC Projects @ OSM

- **Bootstrap**
  - Boot (more or less) stock kernel on SCC, by emulating BIOS calls needed for bootstrap

- **Debugging support**
  - SCC TTY redirected to MCPC, for “local” login or kernel debug
  - Virtual UART, emulated by MCPC over PCIe connection

- **PhD topic: Fault tolerance**
  - Understanding SCC as failover cluster
  - Research on FT techniques for SCC

- **PhD topic: Single System Image (SSI)**
  - Span “virtual” OS instance over networked nodes
    - All nodes have the same “view” of OS resources
  - System-wide `top`, file system, console, ...
Intel SCC – lessons learned

• Radically different memory model exposed to OS (and application)

• Not all cores are equal
  • Memory nodes (connected to memory controller)
  • I/O nodes (connected to ASIC and PCIe host link)
  • Management nodes

• Some tasks are simple in SW and hard in HW – e.g.; graphics
  • Distributed frame buffer in HW (...weeks...)
  • X Window redirection in SW (...hours...)

Programmers must understand computer architecture implications

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
CPU level:
Online Hardware Failure Prediction

Using X86 hardware performance events

- Instruction retirement, cache miss, branch miss-prediction, ...
  - Limited number of hardware counter units -> exploit event correlations
  - Threshold-triggered, time-triggered

- Applicable to major cellular multiprocessing platforms (Intel, AMD, SPARC, IBM Power)
## Memory level: observations from our FutureSOC Lab

<table>
<thead>
<tr>
<th>Date</th>
<th>Severity</th>
<th>Event</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-Jun-2010</td>
<td>Info</td>
<td>No</td>
<td>BIOS</td>
<td>System boot (POST complete)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - 'MEM4_DIMM-2D' memory training failed</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - 'MEM4_DIMM-1D' memory training failed</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - 'MEM4_DIMM-2B' memory training failed</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Major</td>
<td>No</td>
<td>[0x00:00]</td>
<td>POST - 'MEM4_DIMM-1B' memory training failed</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>'MEM4_DIMM-1D' Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>'MEM4_DIMM-1C' Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>'MEM4_DIMM-1B' Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>SMI</td>
<td>'MEM4_DIMM-1A' Memory: Uncorrectable error (ECC)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>'MEM4_DIMM-2D': Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>'MEM4_DIMM-1D': Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>'MEM4_DIMM-2B': Memory module failed (disabled)</td>
</tr>
<tr>
<td>14-Jun-2010</td>
<td>Critical</td>
<td>Yes</td>
<td>iRMC S2</td>
<td>'MEM4_DIMM-1B': Memory module failed (disabled)</td>
</tr>
<tr>
<td>15-Jun-2010</td>
<td>Info</td>
<td>No</td>
<td>BIOS</td>
<td>System boot (POST complete)</td>
</tr>
</tbody>
</table>
OS level:
our NTrace for Windows

Compiler/linker switch
- /hotpatch, /functionpadmin
- Microsoft C compiler shipped with Windows Server 2003 SP1 and later

Hotpatchable:
- Windows Research Kernel

Foo-5:

Foo:

CallProxy:

EntryThunk:

... retn 10
nop
nop
cmp
NtfsPinMappedData:

... mov edi, edi
push ebp
mov ebp, esp
mov ecx, [ebp+18h]
mov edx, [ebp+0Ch]
...
Our idea: Global System Health Indicator

Multi-Level Failure Prediction

- Application & Middleware level
- Operating System Level
- VMM Level
- Hardware level

- Application-specific counters, JSR-77, AppServer Monitoring
- Dtrace, Windows Monitoring Kernel
- VMware vProbe
- Machine Check Architecture, CPU Hardware Profiling

Virtualization Cluster Management

System Health Indicator
VM Migration – how long does it take?
VMWare ESX 4

**blackout time vs. cpu load**

**blackout time vs. physical RAM usage**

**migration_time of VMware_ESX_vSphere_4.0 : vmsize = 4096**

**migration_time of VMware_ESX_vSphere_4.0 : vmsize = 8192**

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
FutureSOC-Lab – lessons learned

- Servers have evolved
  - Ever growing number of CPU cores
  - Tremendous amounts of memory
- Reliability will become the most sought-after feature of future server systems
  - Higher density, integration levels in future CPUs will lead to multi-bit faults
  - Failure prediction and VM migration as promising concept
- Must have fault isolation boundaries (LPARs, blades)

Server systems call for new programming and management models
Future SOC Lab @ HPI

- Vision was to establish an open research platform for tomorrow’s IT landscape, start: June 2010

- Industry partners
  - Fujitsu
  - Hewlett-Packard
  - SAP
  - EMC
  - VMware
  - NetApp

Testbed:
MultiCore MultiThreading Hardware, huge memories, NehalemEX-based, GPU computing

HP ProLiant DL980 G6:
64 Cores, 1-2TB
Fujitsu Primergy RX600S:
32 Cores, 1TB

Application areas:
- Large Databases
- Consolidation, Virtualization
- High-Performance Computing

Steering committee from industry and academia

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
FutureSOC Lab – Call for Projects (Deadline: 25\textsuperscript{th} of October, 2011)

www.hpi.uni-potsdam.de/forschung/future_soc_lab.html?L=1

- Multicore architectures
- In-memory business applications
- Real-time Business Process Management
- Service-Oriented Computing and Service-Oriented Architectures
- Cloud Computing
- Software-as-a-Service (SaaS)

Intel MARC Symposium, December 8/9 2011, Potsdam, Germany

http://www.dcl.hpi.uni-potsdam.de/marc2011
Cooperations

Deutsche Post IT-Solutions
- AOP tools
- Embedded Systems

Microsoft / Microsoft Research
- Micro.NET, Phoenix
- Windows Research Kernel
- Curriculum Research Kit

Hewlett-Packard
- Server Computing Summit
- OpenVMS beta program
- FutureSOC Lab

IBM Labor Böblingen
- RASMUS / fault analysis
- z/PDT, fault injector for z/Linux

Intel Lab Braunschweig
- Single Chip Cloud Computer

EU-Projects
- Adaptive Services Grid
- Leonardo Da Vinci

Fontane-Projekt
- BMBF
- ZAB / ILB

Bachelorprojekte
- Daimler Research
- Siemens AG, Beckhoff
- Software AG
- Biotronik, Getemed

Frank Feinbube, Operating Systems and Middleware Group, HPI Potsdam
Servers have evolved...
- New form factors
- Higher density
- Standard architectures
- Multicore/multithreaded

Advances in operating systems
- Virtualization
- Thrustworthiness/security
- Clustering
- Need for new programming models, SW Architectures, Services

Virtualization problems
- Security: extended attack surface
- Virtualization-based malware
- Must trust hypervisor

Cloud Computing – the three layers

Challenges:
- Has to abstract underlying hardware
- Be elastic in scaling to demand
- Pay per use basis

Hybrid Computing
OpenCL: New Programming Models

One Host + one or more Compute Devices
- Each Compute Device is composed of one or more Compute Units
- Each Compute Unit is further divided into one or more Processing Elements