Multicore Pain (and Gain) From a Virtual Platform perspective

Dr. Jakob Engblom, Technical Marketing Manager

jakob@virtutech.com
What has Multicore Meant for Us?

- As a software development organization, multicore has impacted Virtutech in several ways
  
  - How we build our product
  
  - How our customers use our product
  
  - The nature of the products our customers build
  
  - How our customers build their own products (using our product)
WHERE DO WE COME FROM?
Virtual platform == software

Running on a regular PC, server, or workstation

Functionally identical to the target hardware

Runs the same software as the physical hardware system

Purpose: to help customers developed systems better
  - In particular software at all levels
The *Simics* product

**Complete target simulation**
- Run OS, drivers, all other software

**Very fast simulation**

**Models any computer system**
- Processor, SoCs, FPGA, ASICs
- Networks, multiple boards/machines
- Multicore processors

**Targets:**
- Single-core aerospace systems
- ... To multicore network processors
- ... To massive multiprocessor servers
DOES ANYONE CARE ABOUT MULTICORE, ANYWAY?
Stackoverflow.com: Multicore tag

Tagged Questions

1. Can I force cache coherency on a multicore x86 CPU?
   - The other week, I wrote a little thread class and a one-way message pipe to allow communication between threads (two pipes per thread, obviously, for bidirectional communication). ...
   - Modified yesterday
   - By Casey
   - 1,011 views
   - 10 votes
   - 9 answers
   - C++ multithreading multicore x86

2. Code for detecting APIC id returns same ids for different logical processors
   - I run my NT service on an Intel Core2 based Win2k3 machine where I need to iterate through all logical CPUs (all bits in process affinity). To do so I call GetProcessAffinityMask() ...
   - Modified 2 days ago
   - By sharptooth
   - 18.5k views
   - 2 votes
   - 2 answers
   - multicore windows affinity asm C++

   - I have a Visual Studio 2008 C++ project that has support for using multiple CPUs/cores when compiling. In the VCPRJ file I see this: <Tool Name="VCCLCompilerTool" Add ...
   - Modified Sep 14 at 17:19
   - By Kirill V. Lyadvinsky
   - 6,817 views
   - 1 vote
   - 2 answers
   - visual-studio multicore compile C++

4. How do I run processes piped with bash on multiple cores?
   - I have a simple bash script that pipes output of one process to another. Namely: dostuff | filterstuff. It happens that on my Linux system (openSUSE if it matters) these both p ...
   - Modified Sep 10 at 2:49
   - By Dennis Williamson
   - 1,677 views
   - 3 votes
   - 3 answers
   - bash linux multicore processes scheduling

5. How are you taking advantage of Multicore?
   - As someone in the world of HPC who came from the world of enterprise web development, I'm always curious to see how developers back in the "real world" are taking advantage of para ...
   - Modified Sep 9 at 5:51
   - By Shaan Daniels
   - 467 views
   - 22 votes
   - 19 answers
   - parallel-processing multicore scalability programming concurrency

116 questions tagged multicore
Stackoverflow.com: Some other Tags

- Ent duplicates)
  - Richard Corden
  - 4,607 votes
  - 2,161,462

13,991 questions tagged

- Boost when compiling
  - teeks99
  - 482 votes

530 questions tagged

Unanswered

- Is available?
  - Sample, the docs for
  - 5 minutes ago

2,568 questions tagged

<table>
<thead>
<tr>
<th>Tag</th>
<th>Posts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicore</td>
<td>116</td>
</tr>
<tr>
<td>Multithreading</td>
<td>2568</td>
</tr>
<tr>
<td>Boost (C++ library)</td>
<td>530</td>
</tr>
<tr>
<td>C++</td>
<td>13991</td>
</tr>
<tr>
<td>C#</td>
<td>39808</td>
</tr>
</tbody>
</table>
Øredev 2009: Meanwhile, Multicore

- Øredev 2009: the Premier programming conference in Scandinavia
- Multicore/parallel programming is in the program – but not under any obvious heading

Here is a sample of what’s on offer:

- Avoiding pitfalls in parallel programming, Berth Andersson
- Designing for the device neutral mobile web with JavaScript, Brian L. Roux
- Playing on the edge, Craig Taverner
- Our obsession with efficiency, Dan North
- Keeping your options open, even if the cloud is not, Doug Tidwell
- JavaScript: The good parts, Douglas Crockford
- Understanding the origins of destructive leadership, Leo Kant
Embedded Software

Resources
- Free Webinar - Watch Now: Multicore Made Easy: Solving the programming and manageability challenges of multicore systems. Click to Watch!
- White Paper: Enea Multicore high performance server processors enabled with a hybrid SMP|MPR| OS technology
- Brochure: Need a Linux Platform? The Enea Embedded Linux project framework delivers an optimized development foundation that reduces costs and accelerates development. Read more about the Enea Linux Project Framework.

Featured Case Study
ST-Ericsson - In 2008 Enea supplied software to about 450 million new mobiles, primarily 3G phones. A Swedish Bagship! Ericsson has deployed Enea’s real-time operating system, Enea OSE, in its mobile platforms since 1990.

Events
ESC Boston 2009
Enea will be exhibiting at the Industry’s Leading Embedded Systems Event - Register today!
Freescale Market Series: Design with Freescale | US | San Jose
Freescale Offers Solutions for Automotive, Consumer, Enabling, Industrial, and Networking Electronics. Enea will be sponsoring and speaking at the event!
ITU Telecom World 2009
Geneva, Switzerland, Oct 5-9
Open Source Project - Free Seminar (Sweden only)
Embedded Hardware

OCTEON Multi-Core Processor Family

OVERVIEW
The Cavium Networks OCTEON family of Multi-Core MIPS64 processors is the
Multicore is having a Huge Impact in Embedded

- (Control-Plane) Chips going multicore at a rapid pace
  - Across all architectures and markets, Power Architecture, MIPS, ARM, x86, …
  - Not just in data plane, much more important and painful in the control plane

- Software
  - All players moving to support multicore

- Embedded systems very exposed to the nature of the hardware
  - Design particular hardware, integrated software/hardware solutions

- Typical IT programming less exposed to hardware
  - On top of a LAMP stack, SQL database, .net engine, JVM, do not see multicore: it is hidden inside the middleware layers
  - Most applications either trivially parallel (servers) or not performance sensitive
## Our Customers do Care about Multicore

### Servers
- IBM, (Sun), ...
- Multiprocessor, multicore, multithreading standard since 1990s
- Always doing SMP systems and software
- Multicore just piles on the cores faster

### Infrastructure
- Ericsson, Cisco, ...
- Massive distributed, heterogeneous, networked, multi-board, multi-processor systems
- Adopting multicore everywhere in their system
- Parallelizing software, maintaining legacy, getting suppliers up to speed with multicore tools and software kits

### Semiconductors
- Freescale, IBM, ...
- Have to build multicore chips to stay competitive
- Get multicore-aware software stacks in place to make multicore chips sellable
- Develop ecosystem of programming tools, operating systems, applications, frameworks, APIs

### Military & Aerospace
- Boeing, BAE, NASA, Wind River, Lockheed-Martin, Honeywell, ...
- Still on single-processor single-core boards
- Scared of multicore, just like they were scared of caches
- Dislike all things being hidden inside an SoC
- Know they cannot avoid multicore in order to get next-generation performance
MULTICORE IMPACT ON VIRTUTECH CUSTOMERS AND SIMICS USERS
Multicore Changed Things

► Customer demand and requirements
  – The types of hardware and software systems Simics is used for
  – The required performance of the simulator
  – The required features of the simulator

► Our technology
  – Simulating multicore
  – Features for multicore
  – Performance tuning of Simics
  – Multithreading the simulator
Simics-customer Systems in the Multicore Era

- **Baseline processor is now a multicore SoC with shared memory**
  - Used to be a single processor until 2005

- **Growth in the use and availability of hardware accelerators**
  - Crypto, security, pattern matching, network protocols, signal processing, …
  - Accelerators make up 50% of the chip area in recent Cavium, Freescale chips

- **Massive DSP and NP farms**
  - Hundreds of DSPs and NPs are common in network processing

- **Software is the main value carrier**
  - 80% or more of system value added by software

- **Mostly standard parts**
  - Off-the-shelf multicore SoCs, with some application-specific FPGAs or ASICs

- **Software defines the system and glues it together**
  - Coordination between cores, chips, boards, racks, network nodes, …
Software Setup Example from Freescale

- **Hypervisor runs the system**
  - Hardware devices allocated to partitions by hypervisor
  - Some virtual hardware devices introduced by hypervisor

- **Several static partitions, corresponding to what used to be separate chips or boards**

Source: Freescale Multicore Introduction, see [http://jakob.engbloms.se/archives/877](http://jakob.engbloms.se/archives/877)
Example Multicore System Setups

1. Classic single-processor system
2. Single OS across all cores
3. Control-plane on SMP OS
4. Line-card processing on local-memory Oses (can be of different types)
5. The next generation: target OS running under the control of a hypervisor
6. Potentially with more OS instances than cores
Software Porting Pain

► Operating systems has to go multicore
  – Shared-memory symmetric MP (SMP) necessary to handle control-plane
  – Local-memory asymmetric MP (AMP) typically also supported
  – Hypervisors coming in to manage the complexity
  – Enea OSE, Wind River VxWorks the two big transitions we have seen

► Target software has to go parallel
  – Old code tends not to work out of the box
  – Multithreaded programs ≠ multicore programs
  – Classic embedded programming uses locks and priorities, does not port well
  – Find strategies to reuse existing code in compartments

► Performance an ongoing exercise
  – 1 core to 2 cores – just go parallel at all
  – 2 cores to 3 cores – breaks things that assumed ”me or the other core”
  – 8 cores – need to rethink parallelism and work division
  – 20 cores – another rethink, remove any vestiges of global synchronization
  – 100 cores – yet another rethink and redesign
  – ... Just keep going ...

► Yes... I know that using Erlang, MDA, Matlab, MPI, DSL, all help
Parallel software systems are *non-deterministic* and *chaotic*
- Very small timing disturbances can lead to totally different system execution
- Less stability
- Heisenbugs are common

Traditional rerun-to-repeat debugging is typically impossible

Increased stress on old software
- Running software truly concurrently exposes latent bugs
- Bugs revealed by changes to OS scheduler or compiler libraries

Less insight into the system
- Single debug port on a chip hides many cores, buses, caches
- (Getting better now, finally)

Hard to look at part of a running system without killing it
- Stopping one core, leave others running, for example
VIRTUTECH SIMICS AND MULTICORE
What do Developers do with Simics

► **Software development**
  - Operating systems
  - Middleware
  - Applications
  - Distributed applications

► **Testing**
  - Software integration
  - Fault tolerance and reconfiguration
  - Hardware in the loop

► **System architecture**
  - Function partitioning, processor placement, selection, memory sizes

► **Computer architecture**
  - Especially in academic settings

► **In almost all cases, speed of simulation is crucial to Simics value**
  - Strong culture of making Simics go fast, very fast
  - We are talking 1000 of MHz or MIPS, slowdown on real code around 5 observed
**Simulation Helps with Multicore Systems Development**

- **Multicore a shift that helps introduce new technology**
  - Gain for Simics, as we solve customer pain in a unique way

- **Simics Value to OEMs**
  - No more man-year bugs
  - Hardware independence

- **Offer an execution environment**
  - Repeatable
  - Reversible
  - Encapsulate systems, with global stop
  - Debug and trace anything
  - Configurable and variable
  - Arbitrarily scalable

---

On hardware, only some runs reproduce an error

On virtual hardware, debugging is much easier
Multicore Simulation in Simics

- Multicore identical to multiprocessors & multiboard
  - Several active processors in the system
  - Packaging not relevant at the level that Simics operates
  - Simics multi-processor from the start, thanks to Sun in 1998

- Multiple tightly-coupled processors (typically, sharing memory) simulated using round robin scheduling
  - Simulator sets the semantics of the simulation, independent of host

- Temporal decoupling for performance
  - Run each CPU for a time slice before switching to next

- Idle-time and idle-loop hypersimulation
  - Only simulate active units in the system

- Parallel simulation of loosely-coupled simulation units
  - Typically, between separate networked boards or machines
  - Still, with controlled semantics
Performance: Good Old Days

- Intel/AMD performance competition drove host speed ahead of target speed
- Target processors single-core
- A few single-processor boards in a network
Pain: Multicore Targets Increase Performance Demands

- Single-core host performance mostly stagnant
- Target processors going multicore
- Target systems adding more boards
- = more target cores per host core
Pain: Parallelizing Simics

- We had to build a parallel program ourselves (which was painful)
  - Correctness and performance
- Maintaining determinism, checkpointing, reversibility
- User adaptation and education
  - Updating model semantics with maximal backwards-compatibility
  - We had to introduce the concept of parallel "cells" in a simulation
  - Stricter rules on how models and machines communicate. User modules are local-data share-nothing.
  - Essentially, most users write models that fit in well-defined "plugs" in our framework, and which are simple sequential event-driven modules
- Performance tuning
  - Parallelizing Simics exposed many previously unimportant bottlenecks in the framework
  - More tuning parameters were added to the simulation, in particular data propagation latencies
- Parallelizing tightly-coupled processor cores proved futile
  - Too much synchronization killed the benefits
Parallel Simics: Hierarchical Synchronization

- **Deterministic semantics**
  - Regardless of host # cores

- **Periodic synchronization between different cells and target machines**
  - Puts a minimum latency on communication propagation
  - *Synch interval determines simulation results*, not number of execution threads in Simics

- **Latency within a cell**:
  - 1000-10000 cycles
  - Works well for SMP OS

- **Latency between cells**:
  - 10 to 1000 ms
  - Works well for latency-tolerant networks

- **Builds on current Simics experience in temporally decoupled simulation**
  - Tried-and-tested, only executing faster on a multicore host
Gain: Parallelizing Simics

Gain: Scalability on single host

- Multicore hosts increase Simics scalability almost linearly
- ”Claw back” some of the performance pain caused by more complex target systems
- With 8-core or 16-core hosts, excellent scalability
- Much easier to control a single process than a distributed simulation
  - Which we have done since 1998
Performance Profile of Simics, Example

**Computation-Intense Benchmark on 8-Core P4080**

- Single P4080
- With second P4080 idling
- With second P4080 multithreaded
- Overhead of second P4080 idling

Multithreading effectively removes the overhead of the second machine.

As does hypersimulation, if the temporal decoupling is high enough.
Simics is a branch-intense, data-sparse, irregular, integer program

- General CPU
  - Branch prediction, large caches, high clock frequency
  - More real cores, SMT/"hyperthreading" not very helpful at maximum load
  - Simics does not get much from multiple-issue machines (sometimes 2 IPC in JIT)
  - Virtualization extensions on x86 useful for speeding things up

- GPU
  - Depends on massive data-level parallelism
  - Assumes little synchronization between threads
  - Assumes very many threads ready to run, not the handful that Simics tends to use
  - Very poor match for Simics

- FPGA
  - If we could run Simics on an FPGA, we would be rich
  - Shortcut to perfect CPU implementations…
Pain: Multicore Features

- **Multicore-aware debug**
  - Never assume a single core, ever
  - No default processor, always point out the machine and processor you are working on

- **Parallel scripting in Simics**
  - Parallel machines, operating systems
  - Parallel scripts controlling parallel targets, it can get messy 😊
  - Our Simics CLI scripting system now has threads, barriers, and fifos

- **Integration with other tools**

- **Modeling infrastructure**
  - Make it easy to configure
  - Provide understanding
SUMMARY AND FUTURE PERSPECTIVES
What did Multicore Mean for Virtutech?

**Gain**

- Greater customer value from the simulator
  - More complex systems, more value
- Market appreciation for our unique features
- Increased simulation scalability on a single host machine

**Pain**

- Performance pressure
  - Multicore chips in the targets
  - More things in the targets in general
- Target system complexity
- Multicore-proofing features and tools in Simics
- Parallelizing Simics itself
Future Perspective: Making Sense of Trace Data

- Modern simulator and hardware trace units produce ridiculous amounts of trace data
  - Just tracing does not find bugs, it just produces raw data

- Need tools to convert data into information:
  - Visualization
  - Scripting
  - Automated understanding
  - Detection of suspicious activity
  - Tie to program semantics and code

- Finding bugs and suspicious behavior in a huge pile of data
- Could be a rich research area!
Hypervisors are everywhere

Embedded cores are adding support for hypervision

- Freescale e500mc
- Cavium cnMIPS v2
- IBM Power Architecture
- Intel x86 VT
- And it is used on other cores as well

Would deserve some research…
QUESTIONS OR COMMENTS?

Also see http://jakob.engbloms.se for my blog, http://www.engbloms.se/jakob.html for previous talks and papers, and http://www.virtutech.com/ (dive into whitepapers) for more on Virtutech and our products.