The Benefits of System Simulation for Debugging Multicore Software

Dr Mikael Bergqvist
Virtutech, Stockholm

Topic

- Debugging parallel software
  - For multiprocessors and multicore hardware
- Using virtual hardware
  - As a complement to physical hardware
- Outline
  - Introduction to virtual hardware for software development
  - Multicore computing and parallel software problems
  - Debugging parallel software using virtual hardware
  - War stories
  - Validity discussion

What Virtutech Does

- Provider of Simics: a high-performance, high fidelity, full system simulator
  - **High Performance** – fast enough to run ***real*** software loads (typically 100’s of MIPS, up to multiple GIPS)
  - **High Fidelity** – run full production software, including firmware, device drivers, hypervisor, RTOS/OS, application software
  - **Full System** – simulate entire systems, not just processor cores, or SoCs, or boards
    - Complete machines, backplanes, networks of networks
- The true value of Simics is through enablement of process change: Virtualized Software Development
  - Especially interesting for multicore and concurrent machines

virtutech

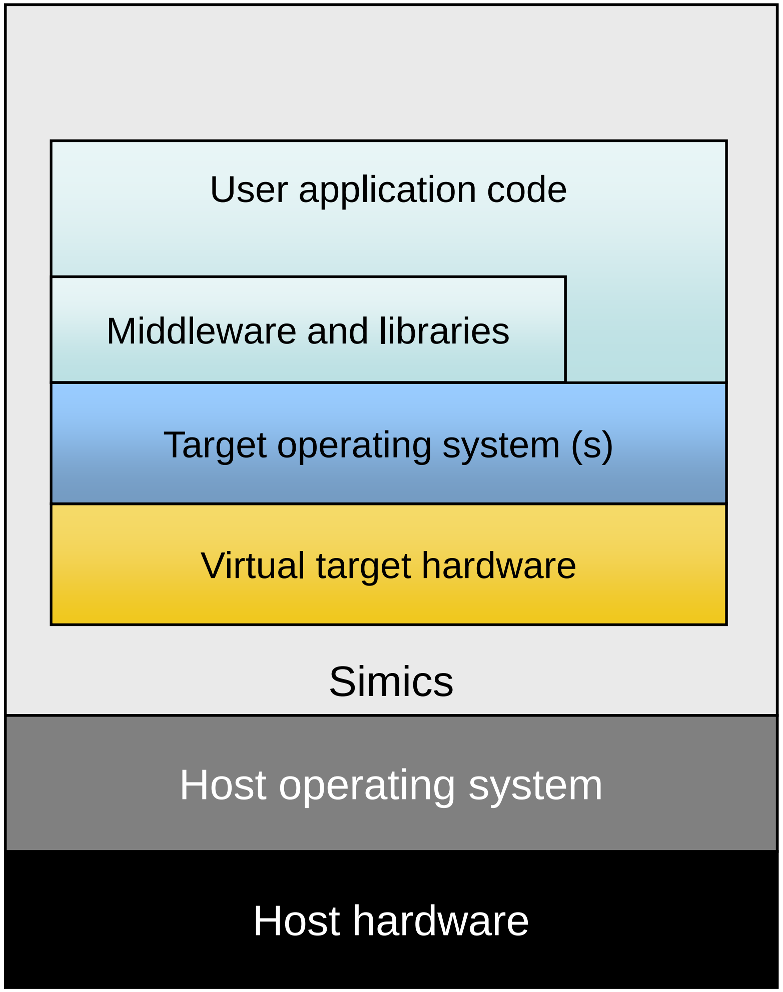
Virtualization for Software Developers

What is Virtual Hardware?

- A piece of software
- Running on a regular PC, server, or workstation
- Functionally identical to a particular hardware
- Runs the same software as the physical hardware system

Virtutech Core Technology

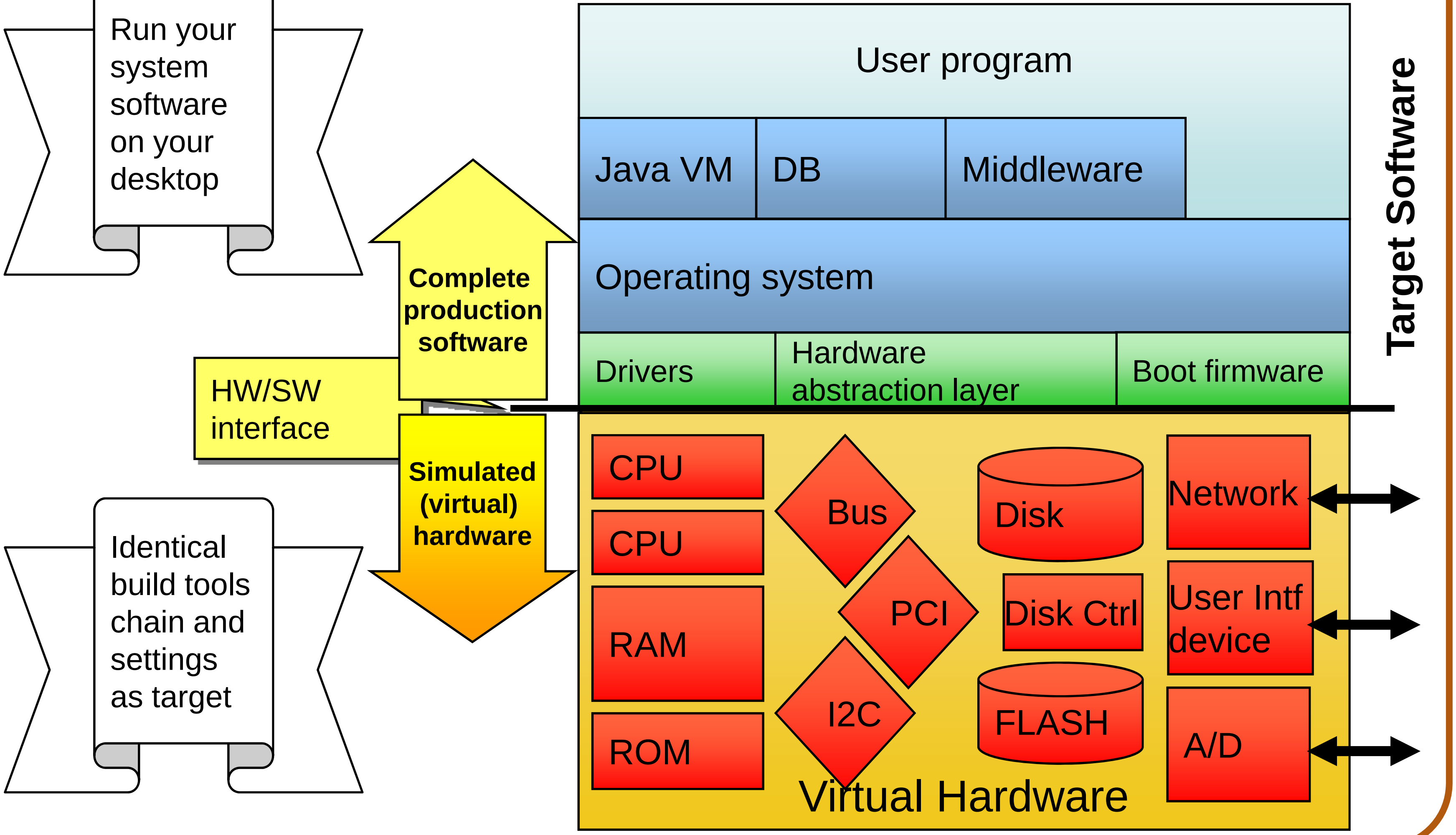
- Model any electronic system on a PC or workstation
  - Simics is a software program, no hardware required
- Run the exact same software as the physical target (complete binary)
- Run it fast (100s of MIPS)
- Model any target system
  - Networks, SoCs, boards, ASICs, ... no limits
- For the benefit of software developers and hardware providers
- Enables process change in software development



Runs the Same Software

The software can't tell the difference

Runs binaries from real target



Run your system software on your desktop

Identical build tools chain and settings as target

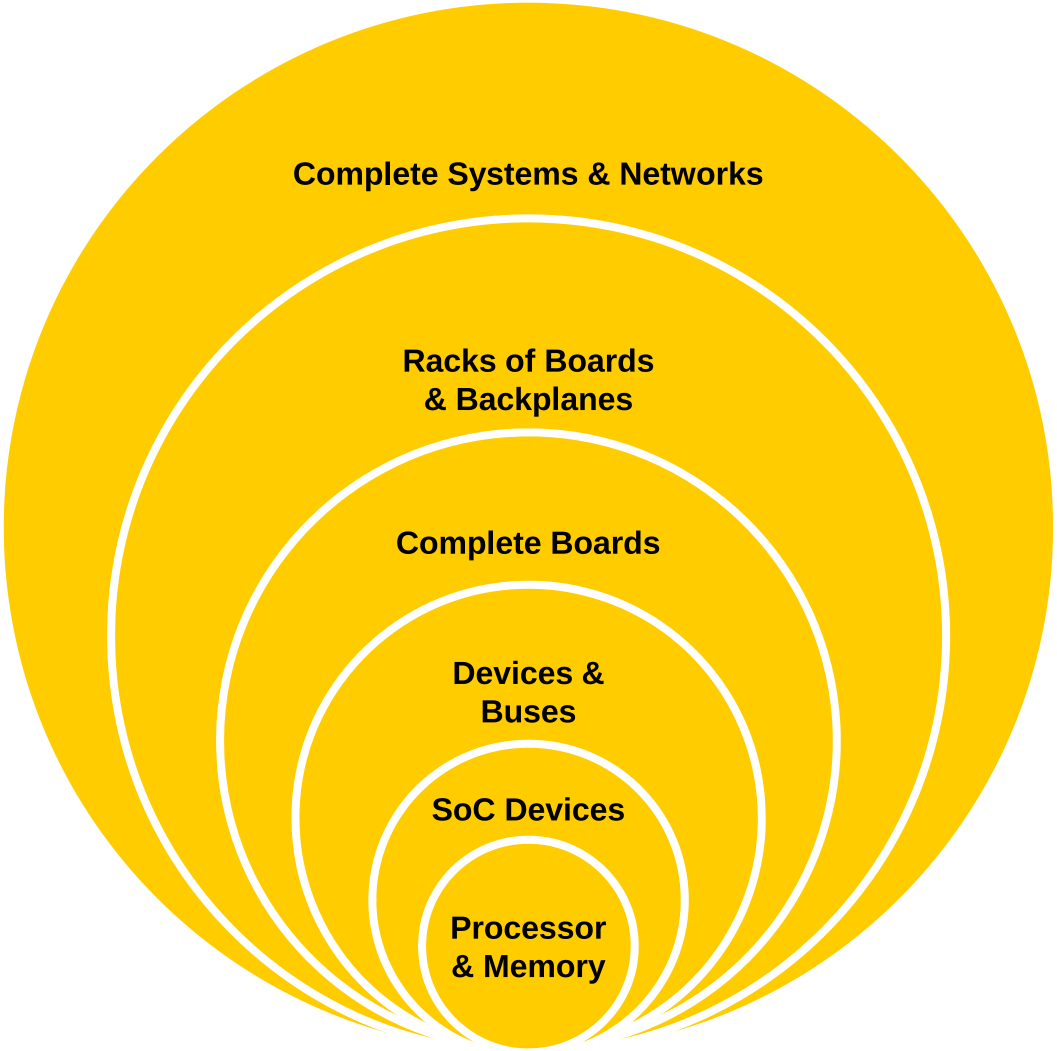
HW/SW interface

Complete production software

Simulated (virtual) hardware

virtutech

What Types of Systems Can be Virtualized?



Examples

- Freescale PPC cores such as e300, e500-mc, e600
- Freescale QorIQ P4080, MPC8572E, MPC8548E
- PCI, PCI-X, RapidIO, Custom ASICS
- MPC8548CDS, MPC8572DS
- Telecom rack, avionics bay, blade server
- Satellite, telecom network, backbone net

Why do we use Virtual Hardware?

- Business Reasons
- Develop software before hardware becomes available
  - Shorten time-to-market
- Decouple hardware and software development
- Reduce software risk
- Increase quality
- Availability & Flexibility
  - Engineering workstation can be "any" system
  - Easy to change the system
  - Easy to distribute and supply to engineers
  - Infinite supply of test hardware

- Engineering Reasons
- Deterministic
- Virtual time
  - Precisely synchronized
  - Stopped at any point
- Checkpoint & restore
- Reverse execution
- Configurable
- Control
  - Any variable or property can be changed
  - Controlled experiments, no real-world randomness
- Inspection power
  - Any state or variable
- No debug bandwidth limit

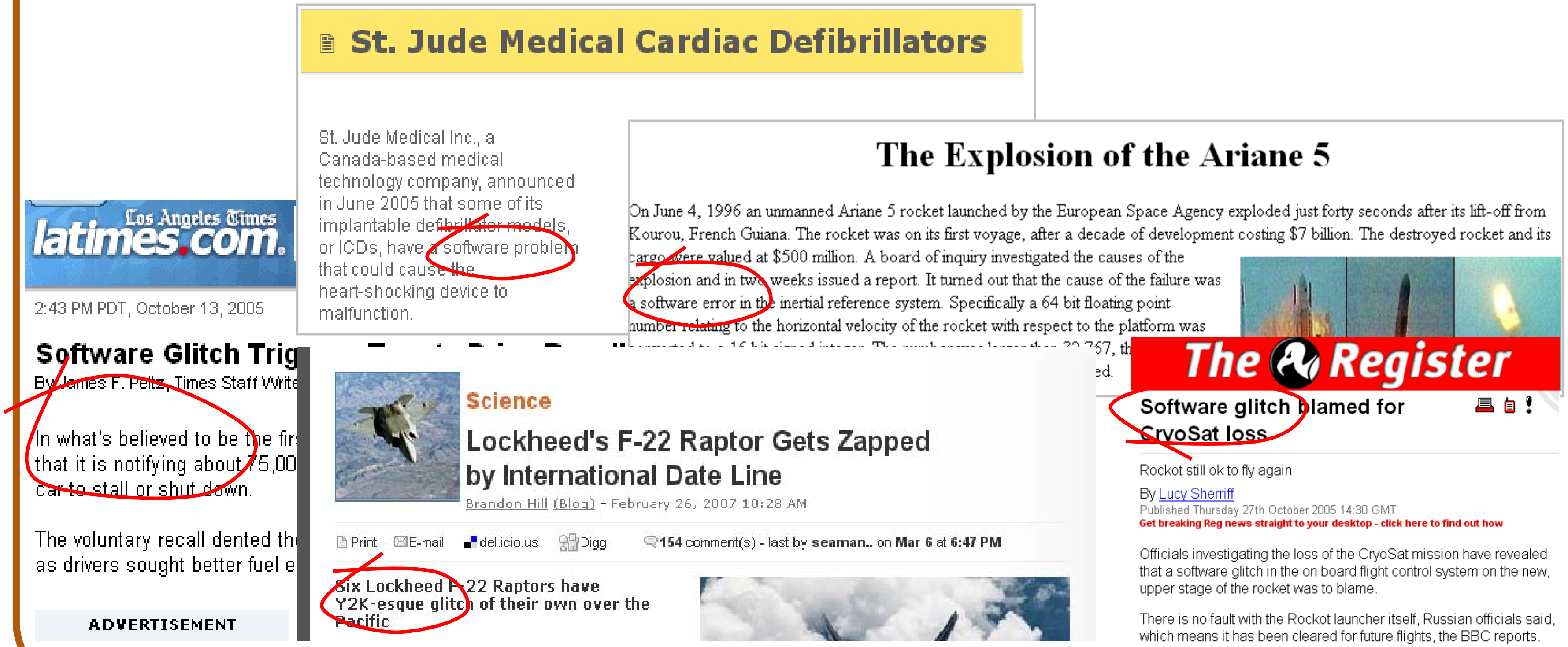
virtutech

Multicore Computing and Parallel Software

virtutech

Electronics Is Software

Electronics is software. Shipping a system is largely about identifying and removing defects from the software and keeping them from creeping back in as the product evolves.



St. Jude Medical Cardiac Defibrillators

St. Jude Medical Inc., a Canada-based medical technology company, announced in June 2005 that some of its implantable defibrillator models, or ICDs, have a software problem that could cause the heart-shocking device to malfunction.

The Explosion of the Ariane 5

On June 4, 1996 an unmanned Ariane 5 rocket launched by the European Space Agency exploded just forty seconds after its lift-off from Kourou, French Guiana. The rocket was on its first voyage, after a decade of development costing $7 billion. The destroyed rocket and its cargo were valued at $500 million. A board of inquiry investigated the causes of the explosion and in two weeks issued a report. It turned out that the cause of the failure was a software error in the inertial reference system. Specifically a 64 bit floating point number relating to the horizontal velocity of the rocket with respect to the platform was

latimes.com Los Angeles Times

2:43 PM PDT, October 13, 2005

Software Glitch Trig

By James F. Peltz, Times Staff Write

In what's believed to be the fir that it is notifying about 75,00 car to stall or shut down.

The voluntary recall dented th as drivers sought better fuel e

ADVERTISEMENT

Science

Lockheed's F-22 Raptor Gets Zapped by International Date Line

Brandon Hill (Blog) - February 26, 2007 10:28 AM

Print E-mail del.icio.us Digg 154 comment(s) - last by seaman.. on Mar 6 at 6:47 PM

Six Lockheed F-22 Raptors have Y2K-esque glitch of their own over the Pacific

The Register

Software glitch blamed for CryoSat loss

Rockot still ok to fly again

By Lucy Sherriff

Published Thursday 27th October 2005 14:30 GMT

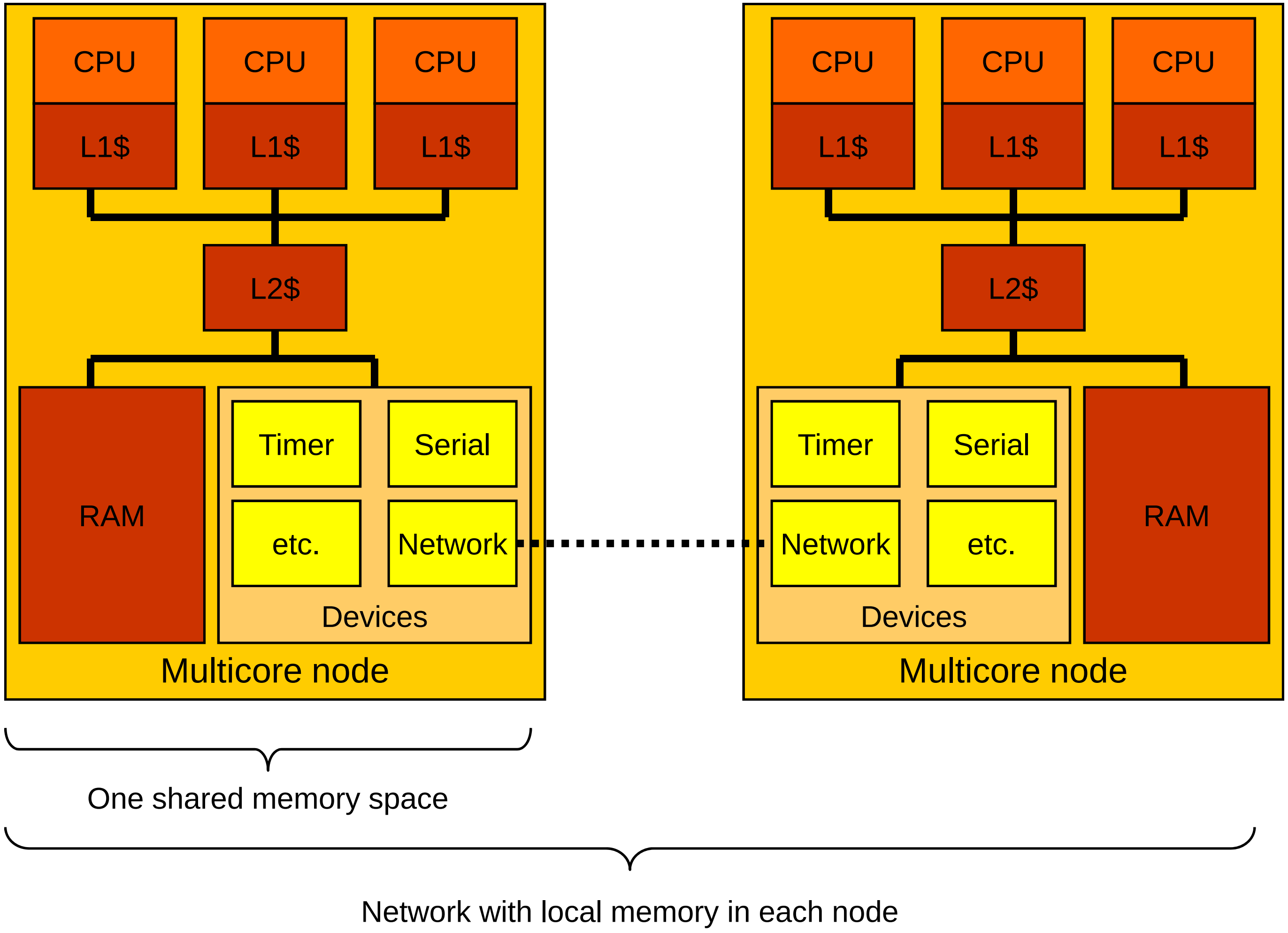
Get breaking Reg news straight to your desktop - click here to find out how

Officials investigating the loss of the CryoSat mission have revealed that a software glitch in the on board flight control system on the new, upper stage of the rocket was to blame.

There is no fault with the Rockot launcher itself, Russian officials said, which means it has been cleared for future flights, the BBC reports.

virtutech

Future Embedded Systems Template



Software Development for Multicore

- Parallelism required to gain performance
  - Parallel hardware is “easy” to design
  - Parallel software is (very) hard to write
- Fundamentally hard to grasp true concurrency
  - Especially in complex software environments
- Existing software assumes single-processor
  - Might break in new and interesting ways
  - Multi*tasking* no guarantee to run on multi*processor*
- ***These are difficult issues for software developers***
  - Requires additional tool support
  - Physical hardware is often not the best development platform

(Embedded) Software Reality Today

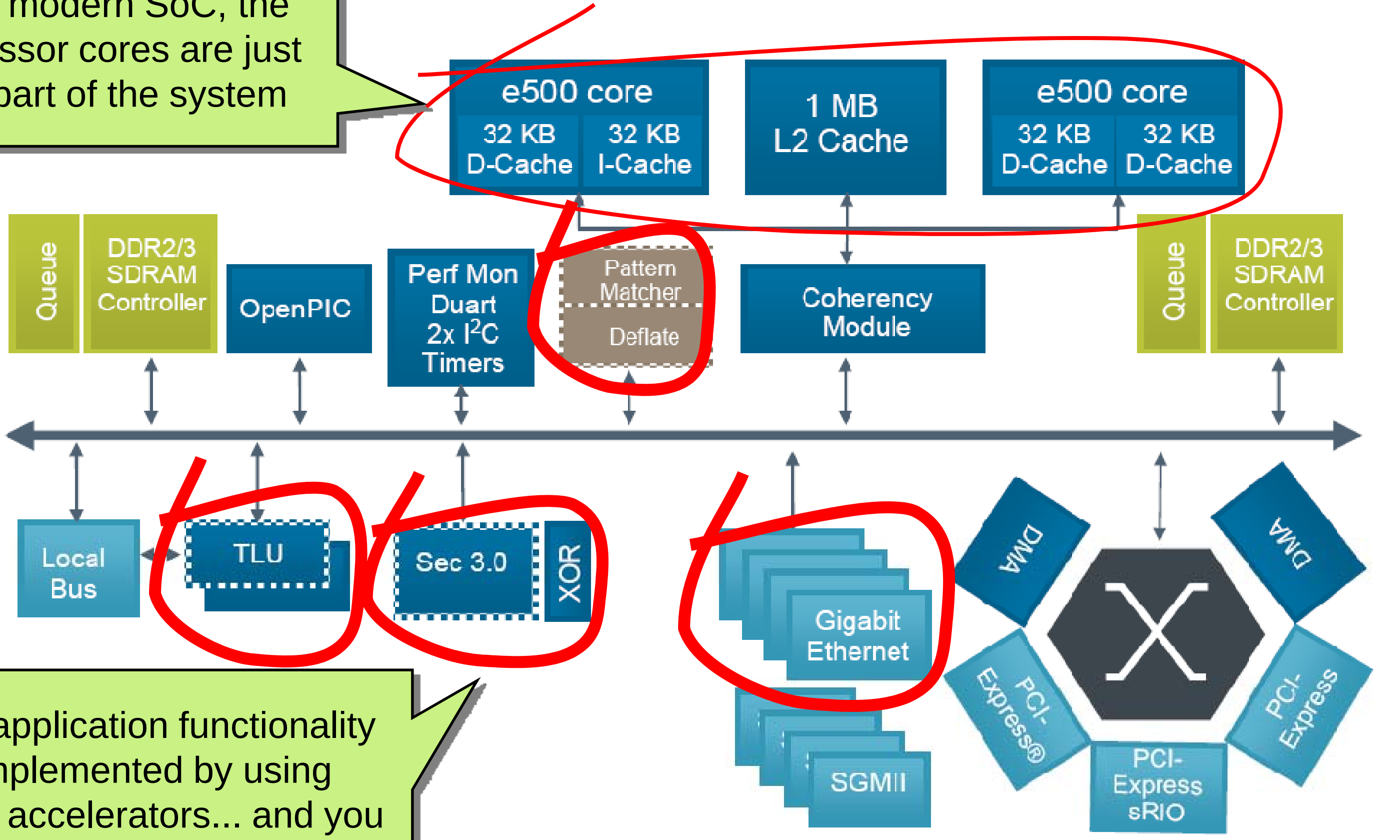
- Programmers used to single-threaded programs
- Legacy code in C, C++, Java, Ada, assembler, Plex
  - Essentially sequential languages
  - Very little in concurrent languages like Erlang
- Fine-grained parallelism added to sequential code
  - OpenMP, pthreads, OS threads, MPI, special C variants, Java threads, Ada concurrency, ...
- Debuggers designed for single processors
  - Or multiple instances of single processors
- If we programmed using better languages, libraries, and tools, many problems would go away.

Multiprocessors & Debug

- Limited visibility into hardware
  - Single debug port, multiple processors
  - High speed, concurrent execution
- Timing-sensitive chaotic behavior
  - Small changes in timing alters system behavior radically
  - Hardware variations impact software behavior
- Lack of determinism
  - Rerunning a program gives different results
  - Hard to reproduce bugs
- Heisenbugs
  - Inserting probes to trace behavior alters behavior
  - Bugs hide when they are being debugged
- System keeps running even if one core stopped

Code is not just about CPUs

On a modern SoC, the processor cores are just one part of the system



Much application functionality is implemented by using special accelerators... and you need to debug their interaction with the processors & software

virtutech

Virtual Hardware to the Rescue!

Three Steps of Debugging

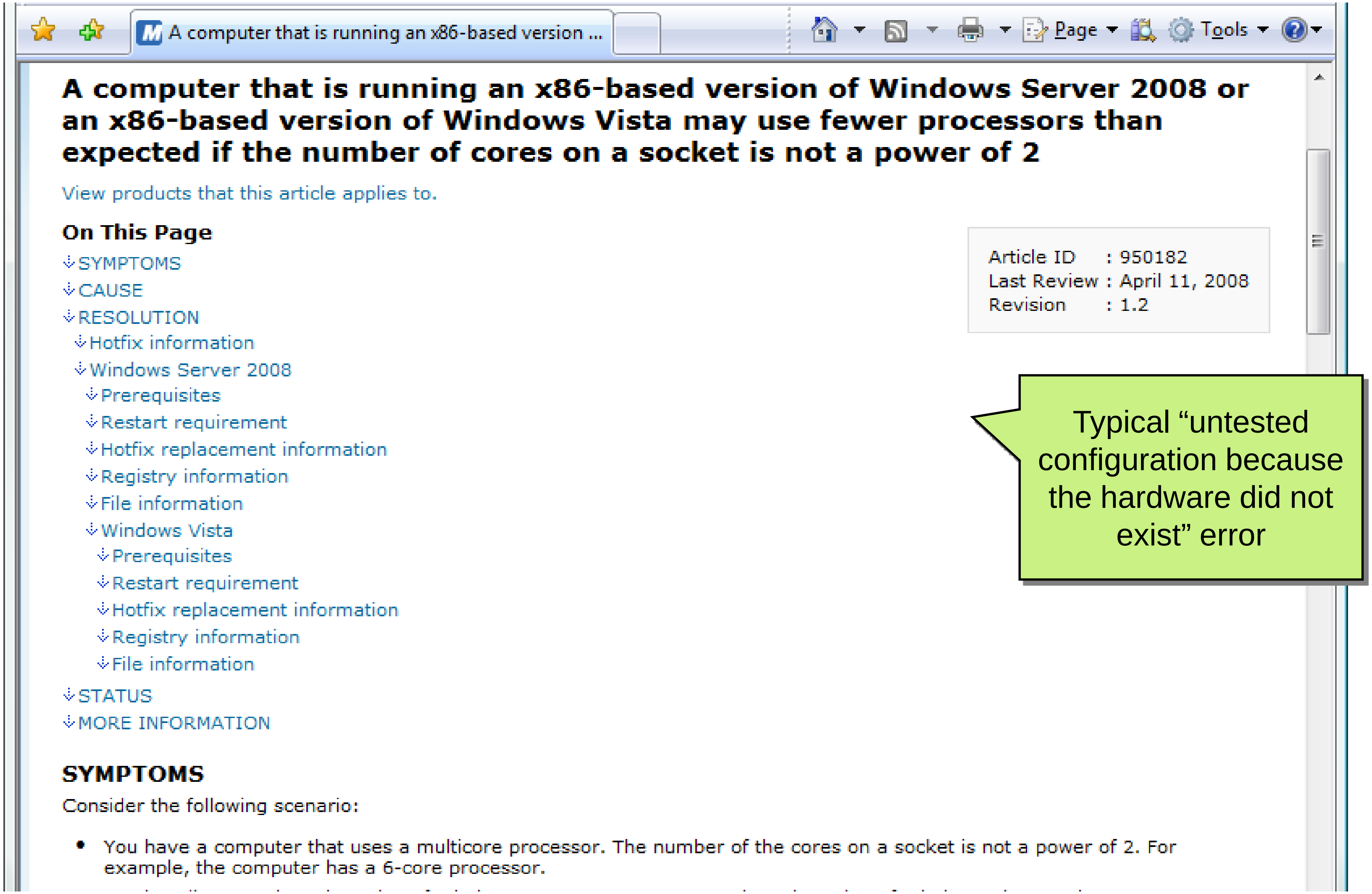
1. Provoking errors
   - Forcing the system to a state where things break
2. Reproducing errors
   - Recreating a provoked error reliably
3. Locating and fixing errors
   - Investigating the program flow and data
   - Depends on success in reproduction

Virtualized hardware helps with all three steps

Provoking Errors

- Virtualization provides **complete control** over system configuration and execution
- Replicate failing tests from production units
- Vary system hardware configuration
  - Like testing on a variety of real-world machines
- Vary system software configuration
  - Easy to test different software loads on different machines
- Systematically provoke corner cases
  - Use oddball processor counts
  - Make a processor slower or stop it entirely
  - Increase communication latencies
  - Slow down individual processors to increase perceived load

Funny Recent Issue



A computer that is running an x86-based version of Windows Server 2008 or an x86-based version of Windows Vista may use fewer processors than expected if the number of cores on a socket is not a power of 2

View products that this article applies to.

Article ID	: 950182
Last Review	: April 11, 2008
Revision	: 1.2

On This Page

- SYMPTOMS
- CAUSE
- RESOLUTION
  - Hotfix information
  - Windows Server 2008
    - Prerequisites
    - Restart requirement
    - Hotfix replacement information
    - Registry information
    - File information
    - Windows Vista
      - Prerequisites
      - Restart requirement
      - Hotfix replacement information
      - Registry information
      - File information
- STATUS
- MORE INFORMATION

SYMPTOMS

Consider the following scenario:

- You have a computer that uses a multicore processor. The number of the cores on a socket is not a power of 2. For example, the computer has a 6-core processor.

Typical “untested configuration because the hardware did not exist” error

Reproducing Errors

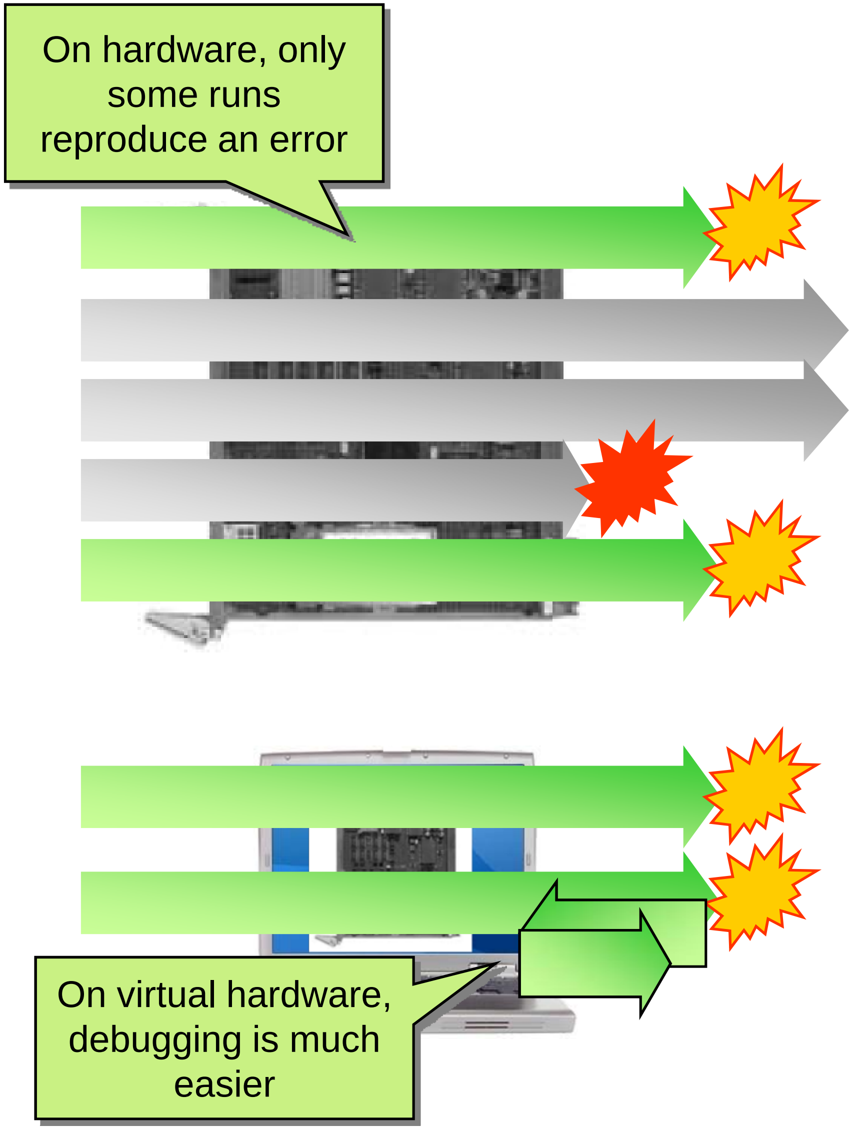
- Virtual hardware state can be **checkpointed**
- Virtual hardware execution is **deterministic**
  - Simulation engine imposes a well-defined sequential semantic to the parallel execution of the target machine
  - All machine-internal events have deterministic time
  - Input from real world recorded & replayed
  - Successive
- An error is provoked in simulation can be reproduced
  - Reset back to initial state (or restore a checkpoint)
  - Rerun the test case that ended in error
  - Same error state results
  - ...any number of times
  - ...on any machine running the simulator
  - ...from a checkpoint distributed to multiple developers

Locating and Fixing Errors

- Determinism and control key features
- No probe effect from instrumentation
  - Tracing and observation from the outside, not by code mod
- No timing disturbance from debugging
  - Breakpoints behave like hardware breakpoints
- Global system stop
  - For practical reasons, this has a small skid of 1-100kcycle
- Heisenbugs cannot occur
  - No intrusion in timing behavior, no varying behavior

Repeatability and Reverse Debugging

- Repeat any run trivially
  - No need to rerun and hope for bug to reoccur
- Stop & go back in time
  - Instead of rerunning program from start
  - Breakpoints & watchpoints backwards in time
  - Investigate exactly what happened this time
- This control and reliable repeatability is very powerful for parallel code!



On hardware, only some runs reproduce an error

On virtual hardware, debugging is much easier

virtutech

Some Bug Stories

Divide-by-zero in OS Kernel

- Operating-system kernel crash in virtual model
  - Divide-by-zero right in the kernel
  - Algorithm to determine and compensate for clock skew
  - Division by difference in time between two processors
- Virtual model had zero clock skew = provoked error
  - Could have happened on a real system
  - Just not very likely
  - Typical rare problem in the field
  - Essentially testing a rare corner case in system state

Race Condition in Serial Driver

- The problem:
  - Dual-core MPC8641D machine
  - Changed clock frequency from 800 to 833 Mhz
  - OS froze on startup – quite unexpectedly
- Investigation:
  - Only happened at 832.9 to 833.3 MHz
  - Determinism: 100% reproduction of error trivial
  - Time control: single-step code feasible
  - Insight: look at complete system state, log interrupts, check the call stack at the point of the freeze, check lock state
- What we found:
  - An interrupt service routine attempted to take a lock, before re-enabling interrupts. In the case that froze, the lock was already taken when the service routine was entered, and with no interrupts enabled there was no way for it to be released.

The Disk Corruption Example Bug

- Distributed fault-tolerant file system got corrupted
  - Rack-based system with many (single-processor) boards
  - Intermittent error
  - Error seen as a composite state across multiple disks: they got inconsistent, for some reason
  - Months spent chasing it on physical hardware
- Simics solution:
  - Reproduce corruption in Simics model of target
  - Pin-point time when it happens, by interval halving
  - Around the critical time, take periodic snapshots of disks
  - Check consistency of disk states in offline scripts
- Result:
  - Found the *precise instruction* causing the problem
  - Capture the network traffic pattern causing the issue
  - Communicated the complete setup and reproduction instructions to development, greatly facilitating fixing the bug

virtutech

Validity of Virtual Debug

“But it is just a simulation”

- Sometimes people do not believe in debugging using a simulator... it is just a simulation after all, not the real thing

- *All* experience contradicts this
  - Any bug found in a virtual environment is a real bug!
  - Our customers are very happy with virtual debugging

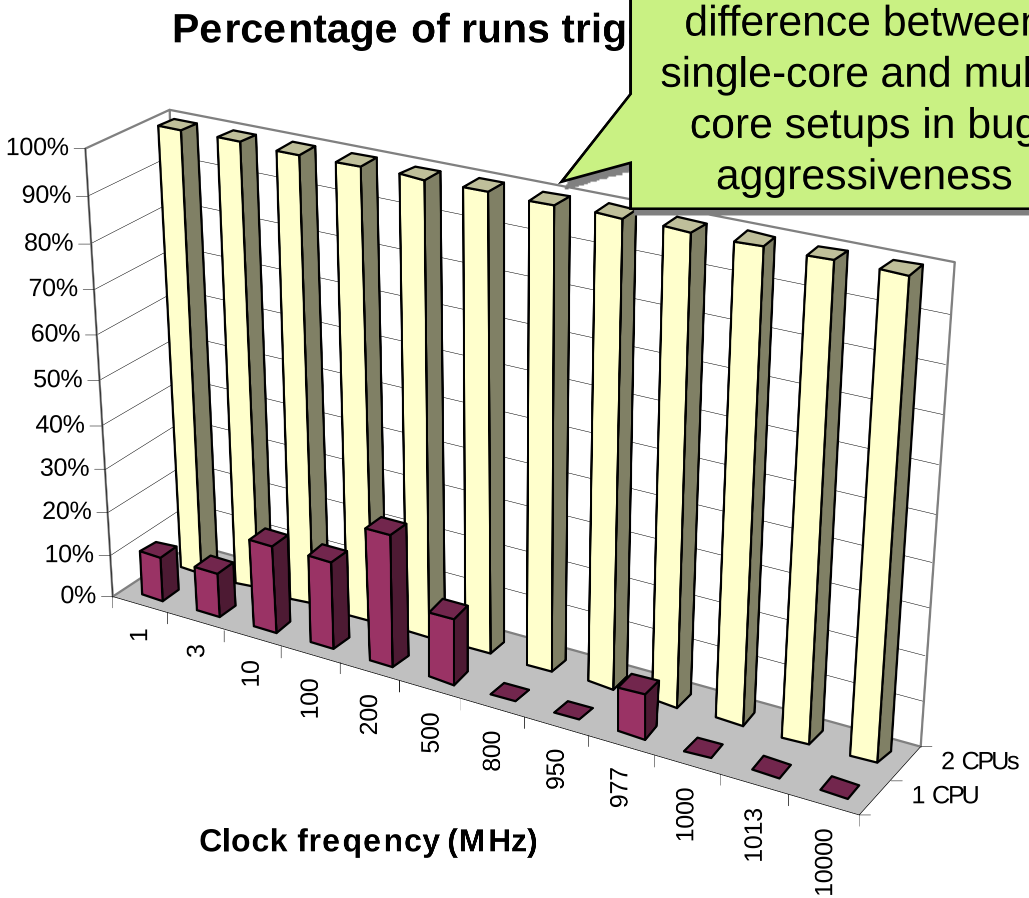
- Experimental evidence back up usefulness
  - Coming up in next few slides!

Simplified Timing and Multicore

- Simplified simulation is *necessary* to run workloads
  - Simplify target timing to get performance [spare slide]
  - Billions and billions of instructions to execute [spare slide]
- You still expose common concurrency bugs and coarse-grain performance effects
  - Race conditions
  - Missing locks
  - Lock contention
  - Locking overhead
- The way to find bugs is not precise simulation of the target but deliberate variation
  - Do not rely on physical randomness
  - Introduce controlled patterns, known variations
  - Rely on repeatability of the simulator to find root errors
- Low-level code sometimes break because of timing-related details such as cache coherency issues
  - Use **hybrid simulation** to investigate failure scenarios with full timing and pipeline and memory system details

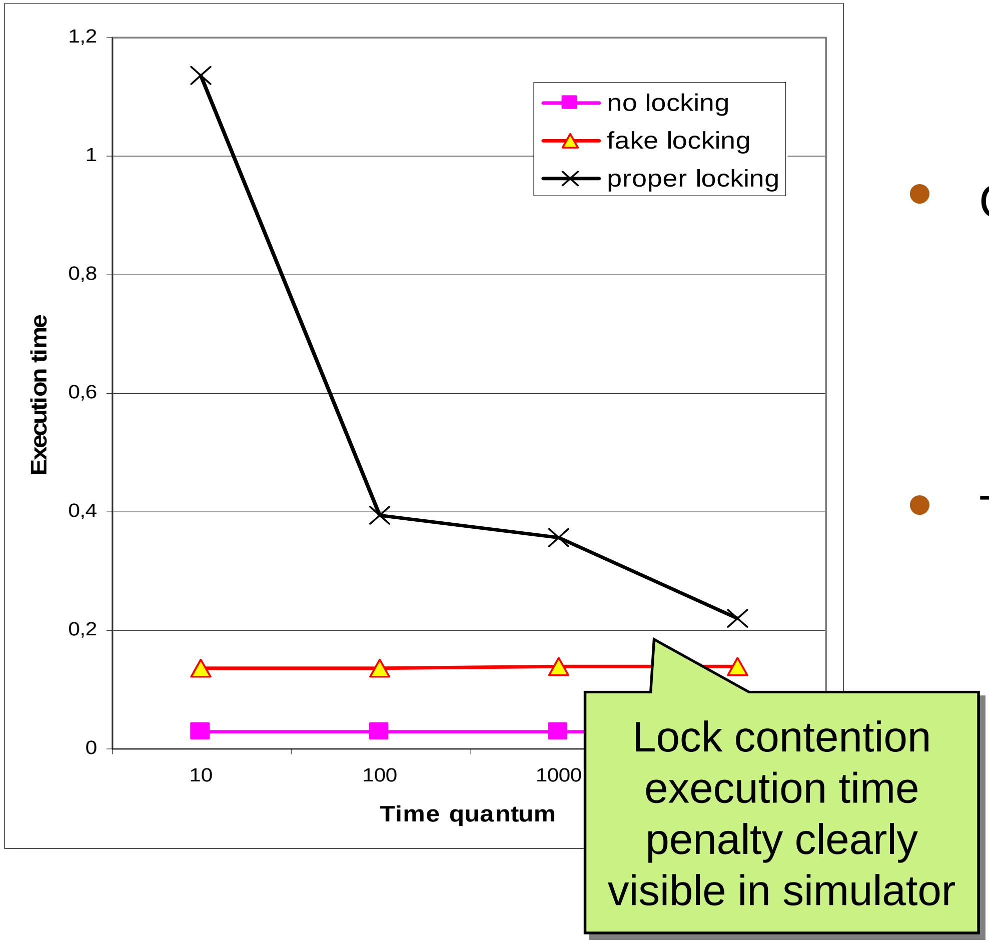
Locking Test Program: Find Race

- Test on single core and dualcore setups
  - Range of frequencies
  - Test program run 20 times on each setup
  - Count percentage of runs triggering race
- Results:
  - Race always triggers in dual-CPU mode
  - Triggers around 10% in single-CPU mode
  - Higher clock = less chance to trigger
- Simulator: simple timing, quantum 1000
- **Simplified timing does not hide the race!**



Simulator shows the difference between single-core and multi-core setups in bug aggressiveness

Locking Test Program: Contention



Lock contention execution time penalty clearly visible in simulator

- Observations:
  - Locking overhead visible
  - Lock contention visible
  - Only proper locking varies in execution time
- On real hardware:
  - no << fake << proper
  - Same relation seen in simulation, even if magnitude varies
- Test program details
  - 2 threads
  - 1000000 iterations
  - MPC8641D virtual target
  - All locking disciplines
  - Time quantum 10-10000

Summary

- Virtual hardware provides an additional tool for embedded software development
- Frees software from hardware dependence
- Especially useful for the tough show-stopper bugs
  - Parallel software
  - Hardware-software interaction
  - ”Heisenbugs”

Our Customers are Convinced

“ Simics is really the only way to develop multi-core software”
-- Tomas Evensen, Chief Technical Officer, Wind River.

“IBM has historically been at the forefront of developing best practices for hardware development, which is especially important as the company continues to create new, complex technologies”
-- Erich Baier, vice president of hardware development, IBM.

“Simics allows us to test our software and validate it while the underlying hardware design is being developed"
-- Gerry Vossler, vice president, Advanced Marketing & Technology, GE.

“The processing potential of multi-core devices remains untapped because multicore systems are only as effective as software’s ability to handle parallelism”
-- Chekib Akrout, former vice president and general manager of Freescale’s Networking System Division.

virtutech

Questions