Experience with multi- and many-core at openlab

Andrzej Nowak, CERN openlab

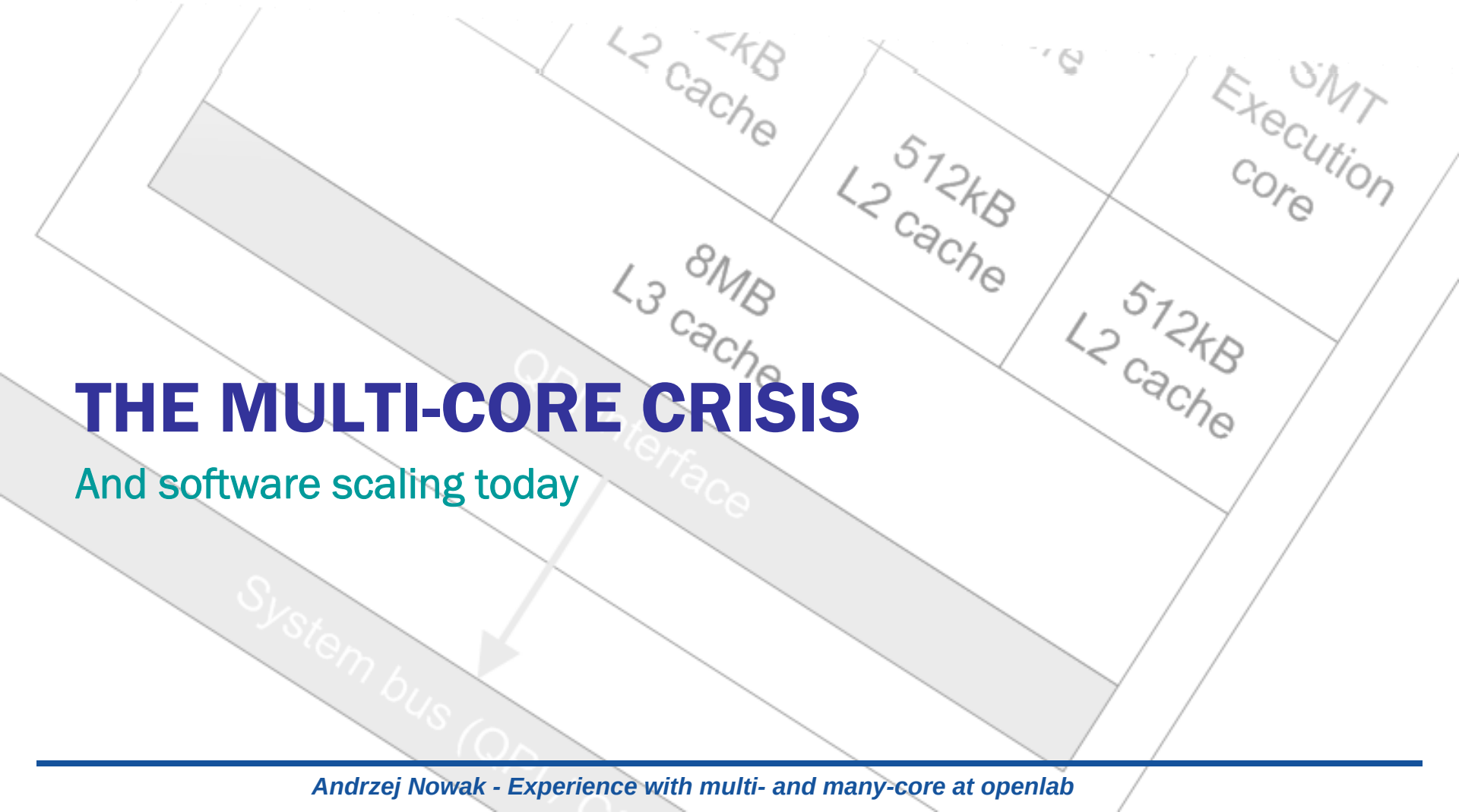
June 25th 2009

Workshop on adapting applications and computing services to multi-core and virtualization

About openlab

- openlab – hi-tech computing research collaboration
- Working closely with industrial partners

- One of the activities: evaluating cutting-edge hardware and software on a regular basis
- Presented views stem from our experience



THE MULTI-CORE CRISIS

And software scaling today

The multi-core crisis

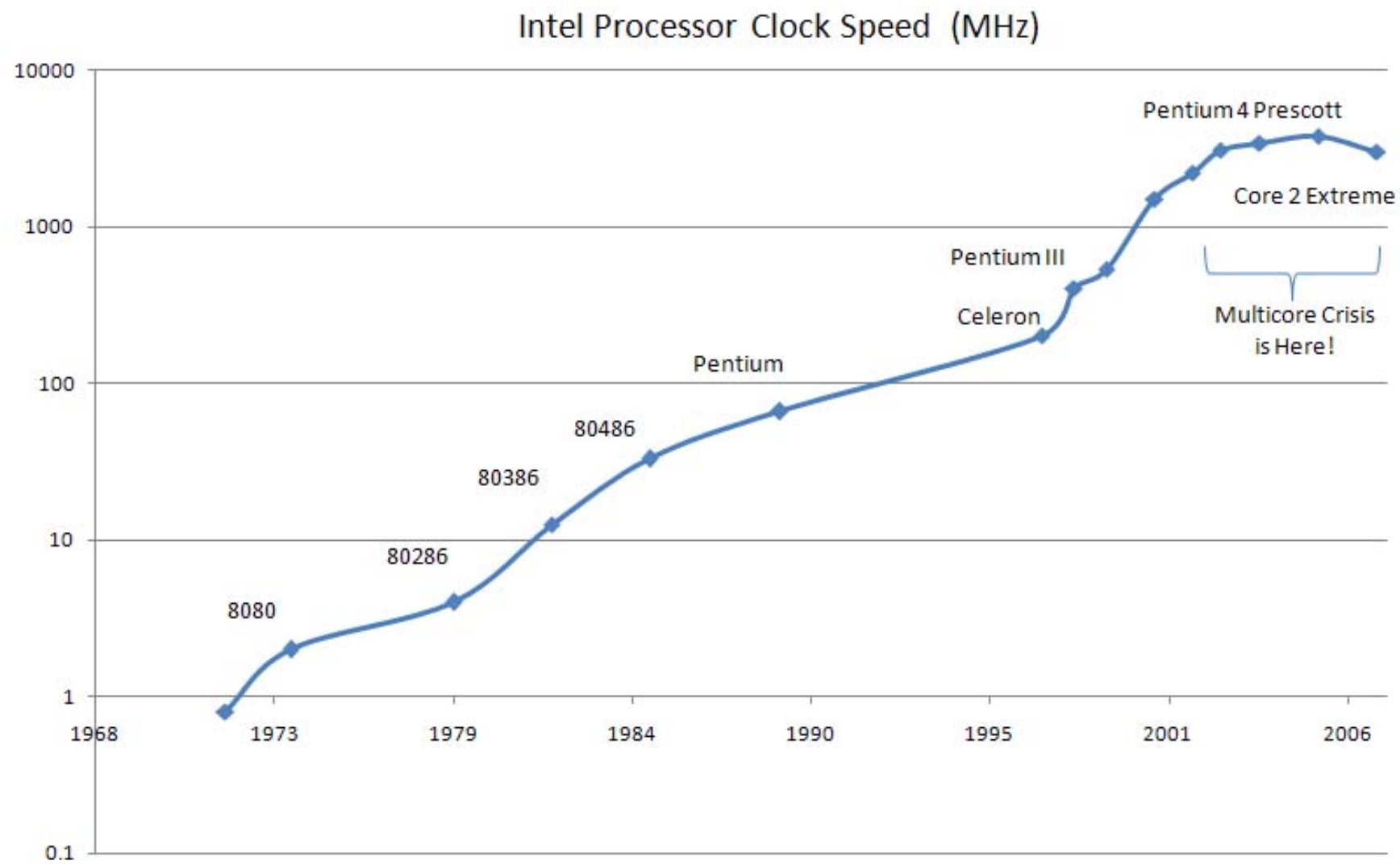
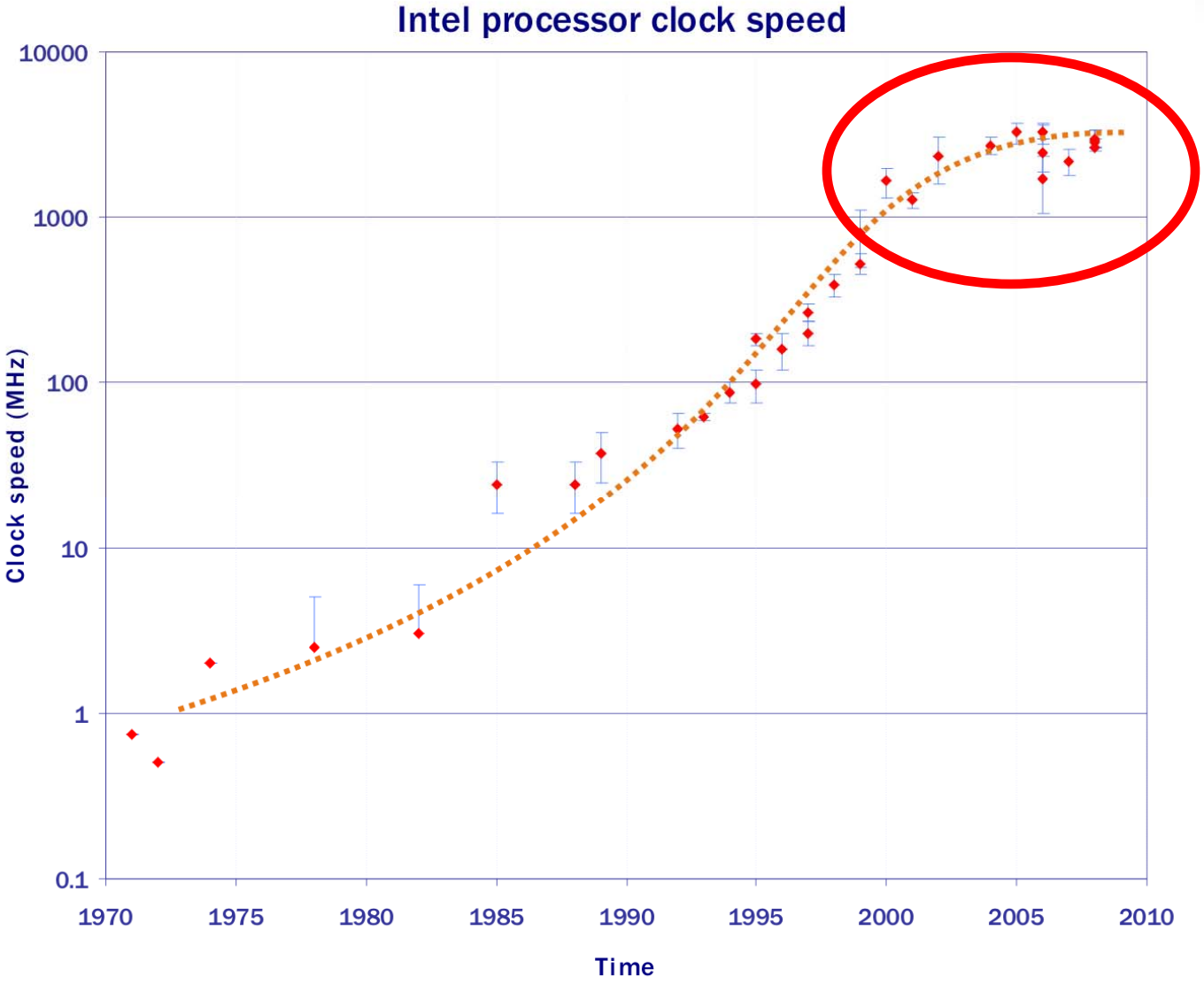


Image source unknown

The multi-core crisis



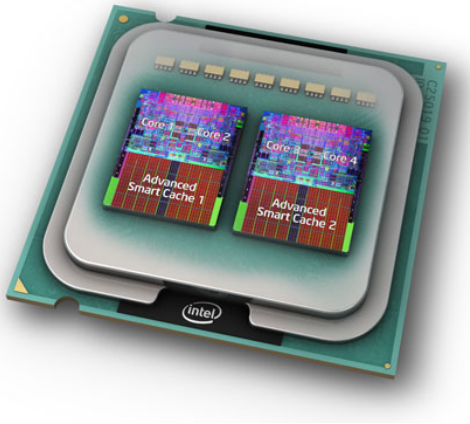
The quest for more computing power (1)

- **Moore's Law continues to drive silicon**
- **Ongoing miniaturization leads to die space savings**
  - 90nm, 65nm, 45nm today (+32nm)
  - (32nm), 22nm, 10nm planned
- **But… the free ride is over**
  - Pentium 4 was supposed to scale up to 10GHz
  - Frequency scaling options understood and exhausted – too much power lost
  - CPUs available today demand threading, more memory and more bandwidth

The quest for more computing power (2)

- **More die space = more transistors**
  - More cache
  - New logic
    - Dedicated units?
    - Cloned units?
    - Either way - PARALLELISM

- **More transistors =**



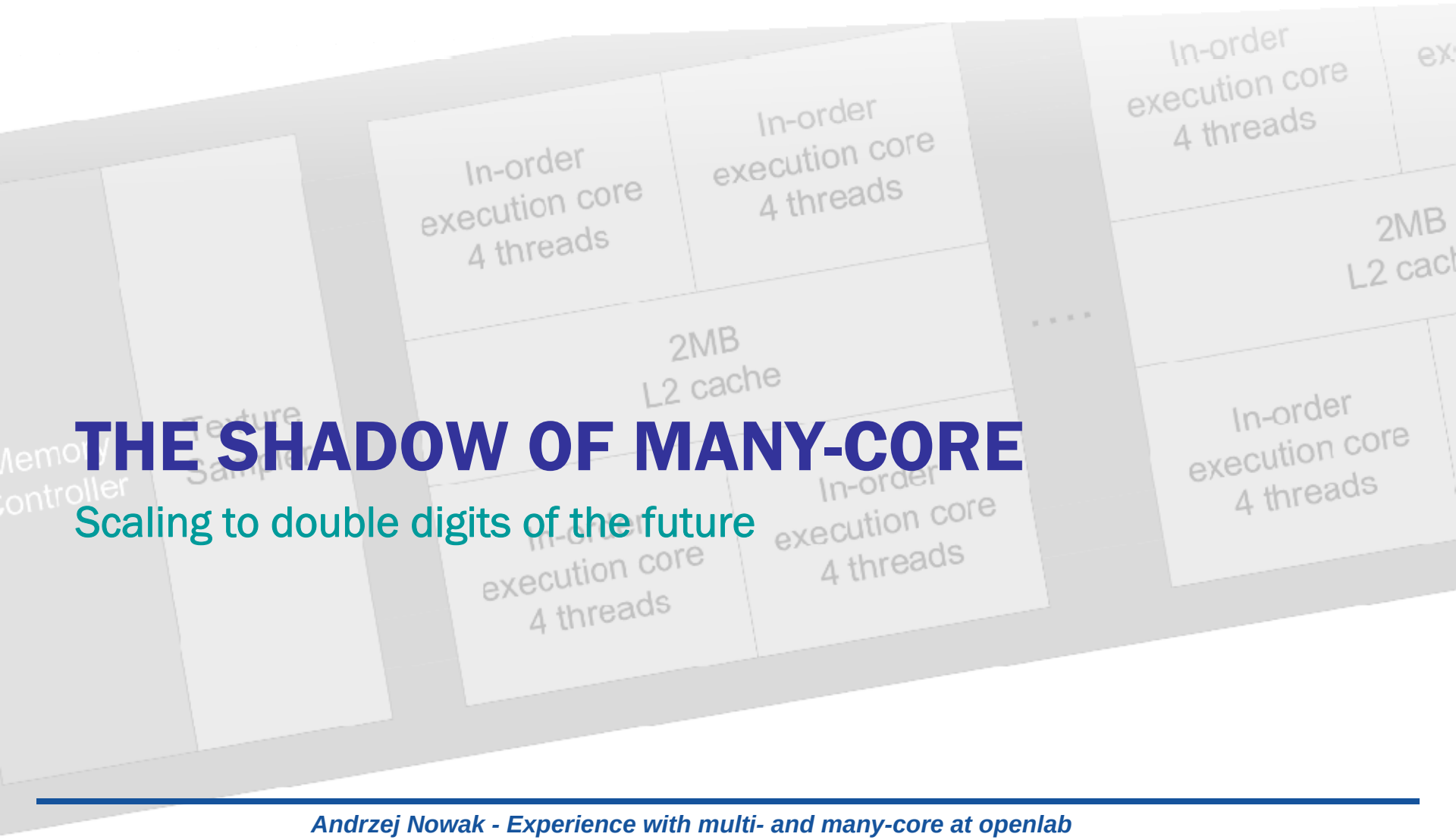
multi-core

Currently available hardware

- **What is multi-core?**
  - Less than 10 cores? Less than 50 cores?
- **2-4 cores on your desktop**
- **2 cores in your iPod – multi-core!**
- **8 cores in your commodity server**
  - Prevalent configuration in CERN computing servers
  - Typical Intel Xeon configuration is 2x4 cores, similar for AMD
- **16 cores in one box are quite common**
- **Safe to say we're in the multi-core era today**

Are we ready for multi-core?

- **The multi-process model still works well in this case**
- **Threading**
  - Most tested software scales up to 4-8 threads
  - Numerous efforts ongoing (as described by Vincenzo Innocente)
- **General experience with threading/scaling:**
  - Going from 1 to 2: quite hard
  - Going from 2 to 4: not difficult
  - Going from 4 to 8: slightly more difficult
  - Going from 8 to 12: much more difficult
  - Going from 8 to 16/24: very hard, long process
- **We're still talking about TODAY**



THE SHADOW OF MANY-CORE

Scaling to double digits of the future

Approaching – the many-core mega-crisis?

- **What is many-core?**
  - More than 10 cores? More than 50 cores?
- **Clearly many-core hardware on the close (2009) horizon**
  - 128 threads in one chip/box – “Beckton” / “LRB”
  - In terms of software development time, “end of 2009” is like “today”
- **Multi-core vs. Many-core**
  - Going from multi- to many- might prove to be as difficult as going from one to multi-
  - On the other hand, lessons learned will facilitate a smoother transition
- **Safe to say we’re on the verge of the many-core era**

Upcoming hardware, consumer & DP segment

> Westmere architecture (Intel)

- Standard CPU in the Xeon family will have 6 cores (“Gulftown”), 12 threads
- 24 threads in one dual-socket box from early 2010

> Sandy Bridge architecture (Intel)

- Standard CPU: 6-8 cores, 12-16 threads
- 24-32 threads in one dual-socket box from late 2010

> Intel “Larrabee” GPU

- Up to 32 64-bit x86 cores in one chip
- 4 threads per core
- Up to 128 threads in one chip from late 2009

Upcoming hardware, MP segment

- **Nehalem architecture (Intel)**
  - 8 core, 16 thread CPU
  - Up to 128 threads in one box
  - Coming late this year! (2009)
- **Westmere architecture (Intel)**
  - Most likely similar parameters, ~2011

- **Bottom line: up to 128 x86-64 threads in one box by the end of this year**

Upcoming hardware, vector width

- **Vector width is increasing**
- **Sandy Bridge (Intel)**
  - 256-bit vectors
  - Late 2010
- **Intel "Larrabee" GPU**
  - 512-bit vectors
  - Hundreds of floats per cycle
  - Late 2009
- **Outlook:**
  - 1024-bit not unreasonable?
  - Processing 2 doubles per op per chip today
  - Processing 256 doubles per op per chip in 2-3 years?

Are we ready for many-core? (1)

- **Standard CERN memory requirement – 2 GB per process**
  - Is it still feasible to run one process per core?
  - Will we be able to easily provide dozens of GB of memory per box? Will the hardware even support it?
  - Will hardware other than the CPU support generated I/O?
- **Scalability tests are being run at openlab in conjunction with Intel**
  - Scaling to 24 and more threads
  - Testing with a multi-threaded Geant4 prototype and other benchmarks

Are we ready for many-core? (2)

- **We don't have threading solutions for multi-core yet**
- **We're not doing a lot about many-core yet**
- **Scaling up to double-digit numbers within one application is very hard**
  - Few and short locks can already cause significant scaling problems
  - I/O problems – disk, memory
  - Many algorithms need to be redesigned, not just "parallelized" or "reimplemented"?
- **Processing model**
  - Communication difficult
  - Autonomy is favored
  - Lessons to be learned from the MPI world? Similar problem, different scale – have we already been there?

CERN openlab

LOOKING FOR SCALABILITY

Software that will help with the hardware

Tools will help

- **Many factors can inhibit scalability**
  - Algorithm design
  - Algorithm implementation
  - System issues
  - Hardware issues
- **The right tools and hardware are needed to investigate scaling**
  - Examples stemming from our experience presented
  - Initial black-box analysis moving to white-box
  - None of the tools presented require recompilation (although it might help in some cases)

Useful tools – perfmon2

- **Purpose: investigating PMU-based events in realtime + basic analysis**
- **Features (perfmon2 / pfmon)**
  - Simple to use
  - Event counting
  - Event sampling (with profiling)
  - Process context awareness (automatic context switching)
  - Wide range of hardware supported (Intel, AMD, IBM, many more)
  - Virtually no performance overhead
- **Source: http://sf.net/projects/perfmon2**

Useful tools – Intel VTune

- **Purpose: investigating PMU-based events + analysis**
- **Features:**
  - Call graphs
  - Hotspot analysis and troubleshooting
  - Visual event and event set selection aid
  - Linux and Windows supported
- **Source: http://software.intel.com**

Useful tools – Intel PTU

- **Purpose: investigating PMU-based events + analysis**
- **Features:**
  - Event sampling (with IP tracing)
  - Statistical call graph
  - Basic block analysis
  - Loop analysis
  - Data access profiling
  - Heap profiling
  - Instrumentation based call graph/count
- **Source: http://software.intel.com**

Useful tools – Intel Thread Checker

- **Purpose: detecting unnoticeable and hard to find threading errors**
- **Features:**
  - Detects hidden potential errors (maps to source code)
  - Supports POSIX threads, OpenMP, Intel Threading Building Blocks, ICC, GCC
  - Linux and Windows supported
- **Source: http://software.intel.com**

Useful tools – Intel Thread Profiler

- **Purpose: tuning multi-threaded applications for performance**
- **Features:**
  - Visualization of application behavior even to the microsecond level
  - Visualization of synchronization events (locks, barriers, etc) – concurrency, timeline
  - Visualization of event distribution
  - Maps events to source code
  - Linux and Windows supported
- **Source: http://software.intel.com**

Intel Tools available site-wide

- **Intel Tools are made available by openlab**
  - Compilers, debugger, threading tools, etc – for both Windows and Linux
  - Pre-installed tools for Linux available from AFS, ready to be used instantly
  - Installers and installation keys available
- **Look in /afs/cern.ch/sw/IntelSoftware/**
- **Documentation: https://twiki.cern.ch/twiki/bin/view/Openlab/IntelTools**

Useful tools – strace

- **Purpose: to investigate system call behavior**
- **Features:**
  - System call histogram – number of calls, time
  - Detailed system call traces, including arguments, return values, time taken
  - Filtering (by call type of by overhead)
  - IP tracing
- **Even more useful to analyze traces using home made tools**
- **Source: any reasonable Linux distribution**

Useful tools – SystemTap and utrace

- **Purpose: to investigate function calls within the kernel (SystemTap) and userspace (utrace)**
- **Features:**
  - Kernel- and user-space function probing (calls, returns)
  - Backtraces, symbol resolution
  - Scripting language for probes
- **Extremely powerful, similar to DTrace**
- **Moving forward very fast**
- **Source:**
  - http://sourceware.org/systemtap/
  - http://people.redhat.com/roland/utrace/ + Roland's GIT tree
  - #systemtap on Freenode

Summary

- **Multi-core is here, many-core is at the door**
- **Re-writing most of HEP code is out of the question, porting is painful**
  - A strategy is needed to deal with the transition
  - Proper tools will help a lot
- **Which model should be adopted for multi- and many-core?**
  - Multi-processing
  - Multi-threading
  - Multi-processing + multi-threading?
  - New model?

Q & A

CERN openlab