

The future of commodity computing and many-core versus the interests of HEP software

CHEP 2012
May 24th 2012

Andrzej Nowak, Sverre Jarp, Alfio Lazzaro
CERN openlab

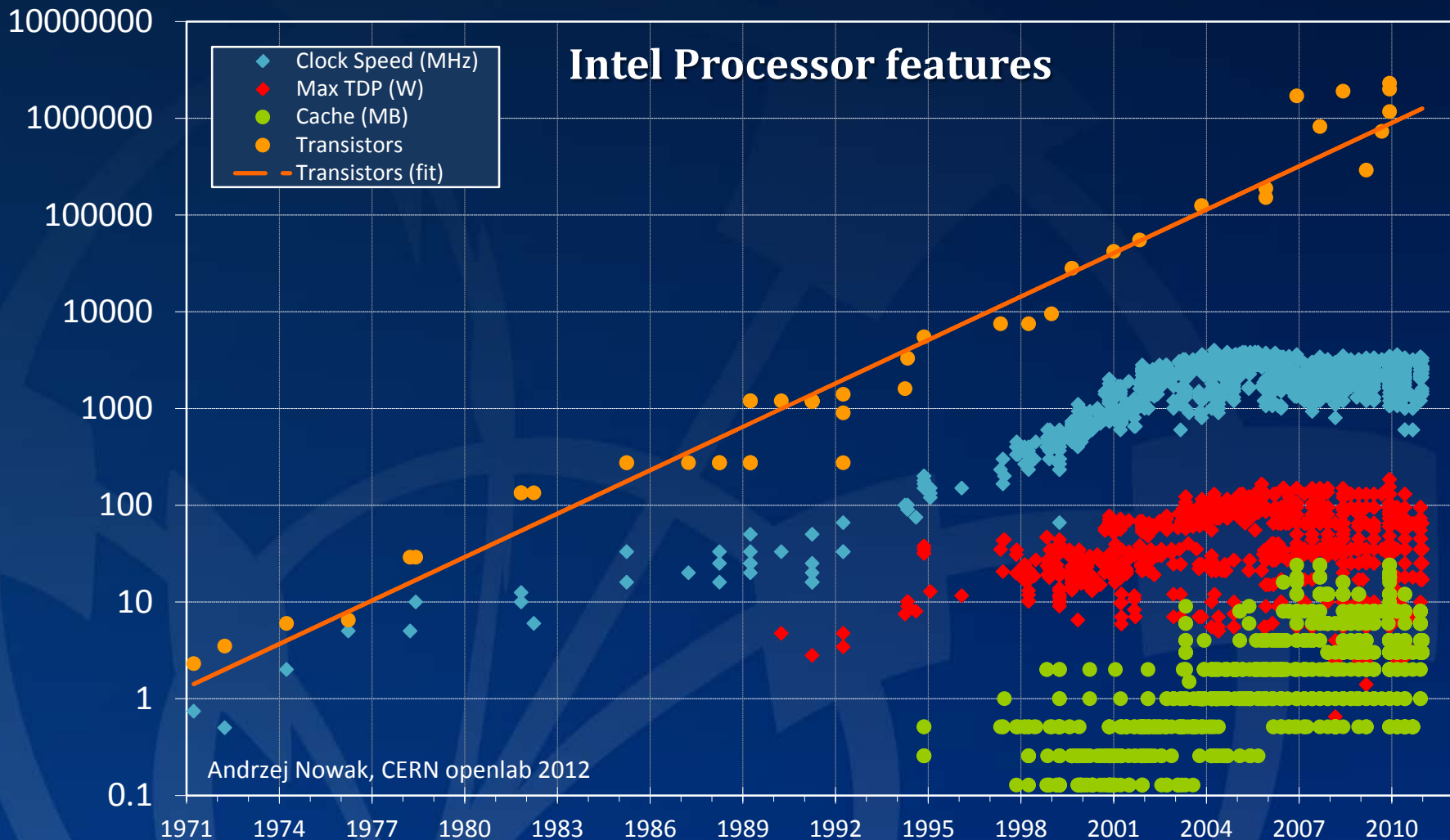
Agenda

1. Why co-design is important
2. Hardware landscape today (x86)
3. Major trends and “napkin calculations”
4. Possible directions and related tradeoffs
5. Recommendations and outlook

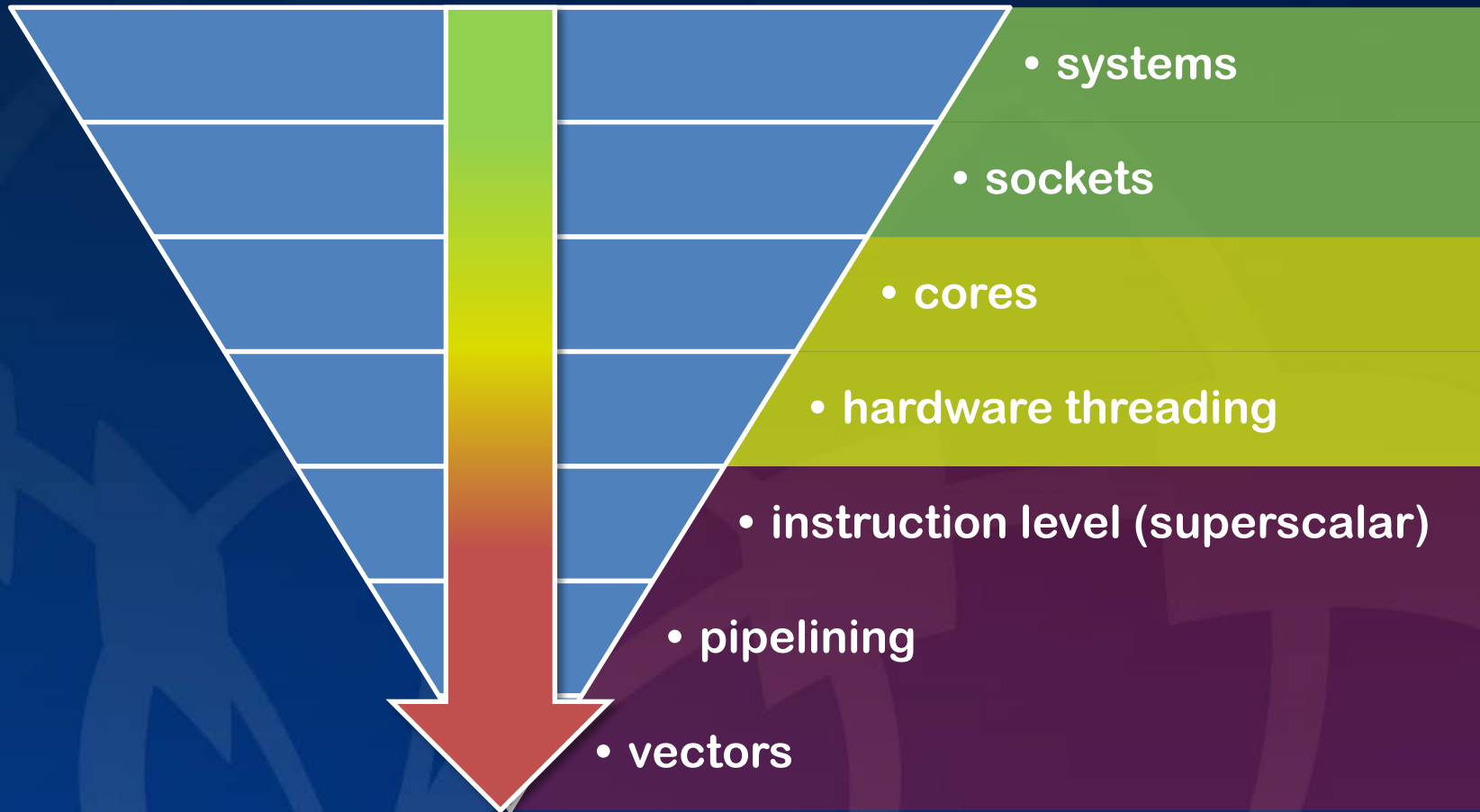
Co-design

- **Two philosophies**
 1. Design software independently of the hardware
 2. Optimize software for the hardware
- **The two aren't mutually exclusive**
- **Today, functionality and programmability considerations rule. The result is inoptimal scalability which will get relatively worse.**
- **A consistent enforcement of option 1 has brought us here today so that synergies with option 2 can be examined**
 - What can be gained?
 - What are, realistically, the available options?
- **Hardware is still non-negotiable**
- **Procurement plays a role**

Hardware landscape



Omnipresent multiplicative parallelism



Where are we now? (software)

- **Large C++ frameworks with millions of lines of code**
 - Thousands of shared libraries in a distribution, gigabytes of binaries
 - Low number of key players but high number of brief contributors
- **Large regions of memory read only or accessed infrequently**
- **Characteristics:**
 - Significant portion of double precision floating point (10%+)
 - Loads/stores up to 60% of instructions
 - Unfavorable for the x86 microarchitecture (even worse for others)
 - Low number of instructions between jumps (<10)
 - Low number of instructions between calls (several dozen)
- **For the most part, code not fit for accelerators in its current shape**

Where are we now? (hardware)

- **Very limited or no vectorization**
 - Online has somewhat better conditions to vectorize
- **Sub-optimal instruction level parallelism (CPI at >1)**
- **Hardware threading unused, but often beneficial**
- **Cores used well through multiprocessing – bar the stiff memory requirements**
 - However, systems put in production with delays
- **Sockets used well**
- **Multiple systems used very well**
- **Relying on in-core improvements and # cores for scaling**

Where are we now?

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	4	1.35	8	4
TYPICAL	2.5	1.43	1.25	8	2
HEP	1	0.80	1	6	2

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	16	21.6	172.8	691.2
TYPICAL	2.5	3.57	4.46	35.71	71.43
HEP	1	0.80	0.80	4.80	9.60

Using a low single digit percentage of raw machine power available today

_%

Write your
percentage here



Hardware trends (1)

- **Pricing follows market pressure, not technology**
- **Vectors – growing substantially**
 - AVX: 256 bits, designed for more
 - AVX: new execution units
 - LRBni (Intel MIC): 512 bits, new vector instructions, FMA, 3-4op
- **x86 microarchitecture**
 - steady, but limited improvements (<10% per “tock”)
 - increasingly advanced features – can HEP benefit?
- **Frequency – very modest changes, if any**
- **IO, disk and memory do not progress at the same rate as compute power**
 - bytes/FLOP decreasing
 - pJ/FLOP decreasing

Hardware trends (2)

- **# of cores “at home” grows arithmetically**
 - various reasons, most linked to the way people use their computers
- **# of cores in the enterprise space still grows geometrically**
- **The number of cores in the datacenter grows between the two, will slow down in the long run**
 - The trend is important, not the end amount
- **Sockets – slight growth with a limit, ultimately impacts core count per platform**
- **Two factors to consider:**
 - Enterprise and HPC-targeted developments “trickle down” to support datacenter developments (where cost effective)
 - Heterogeneous architectures – cross platform, cross socket, hybrid CPUs, accelerators, split into throughput and classic computing

Intel KNC

- The Knights Corner accelerator card builds on the findings of the “Knights Ferry” and “Larrabee” projects
- PCI express format
- >50 x86 cores, **8 GB memory**
- 512-bit wide vectors
- Standard Intel software toolchain
- Projected end of 2012 release, unknown cost
- Promising, but is it indicative of future architectures?

Corollary

**Raw platform performance is expanding
in multiple dimensions simultaneously**

Projections for HEP software

- **Assuming current course without a major change:**
 - No vectorization
 - No change in ILP
 - Hardware threading (SMT/HT) turned on
 - No change in procurement strategy
 - Upper dimensions used at the same level of efficiency as today

Where will we be tomorrow?

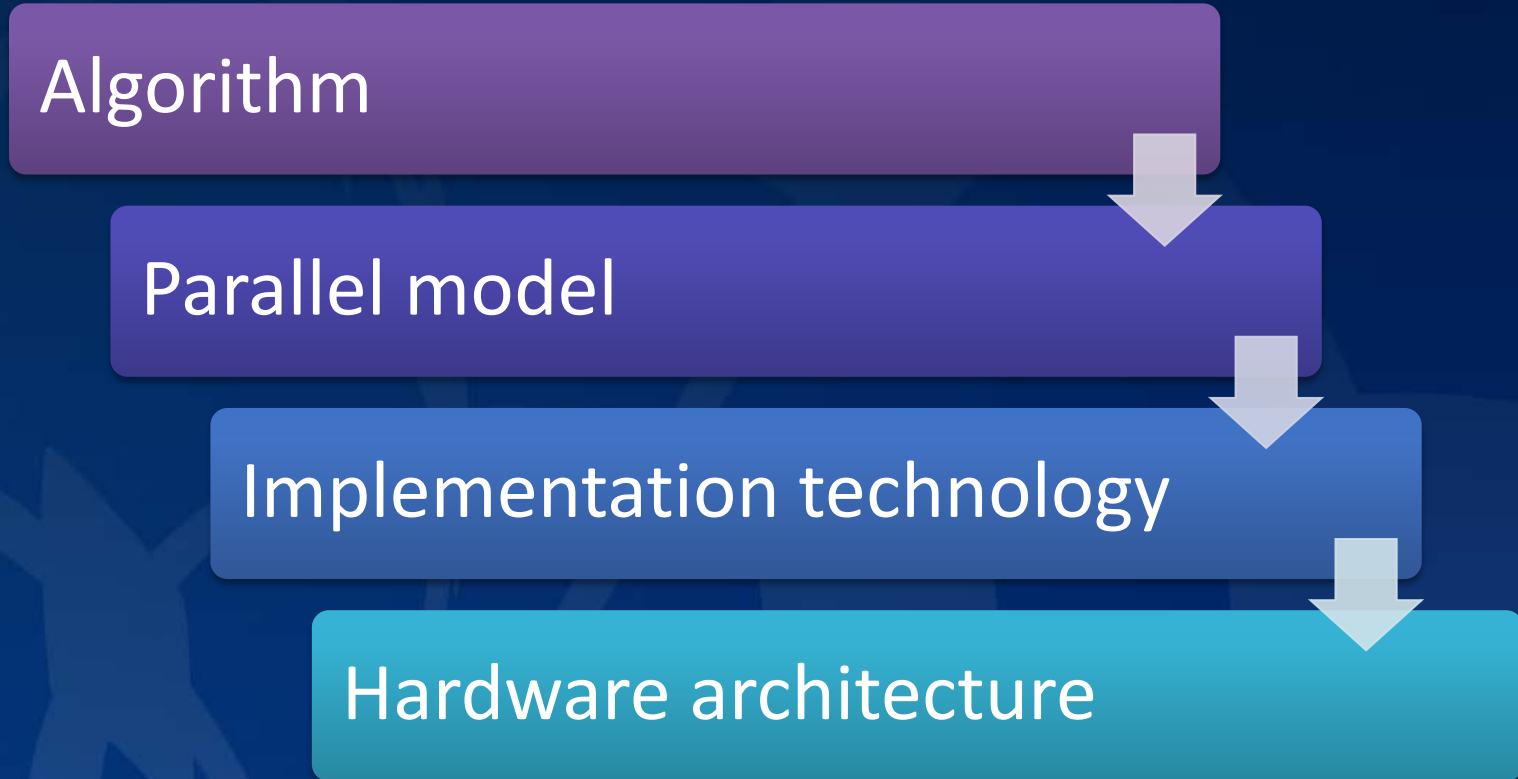
	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	8	4	1.35	12	4
TYPICAL	6	1.57	1.25	10	2
HEP	1	0.80	1.25	8	2

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	8	32	43.2	518.4	2073.6
TYPICAL	6	9.43	11.79	117.86	235.71
HEP	1	0.8	1	8	16

Corollary

**Need to program for tomorrow's
hardware today**

The parallel technology stack



The Hype Cycle

Peak of Inflated Expectations

Plateau of Productivity

Slope of Enlightenment

Trough of Disillusionment

Technology Trigger

Modeled after Gartner Inc.

Tradeoffs in software development (with focus on hardware)

- **Flexibility and programmability vs. performance**
 - Impacts the choice of the programming language, technologies etc
- **Revamp vs. iterative improvement**
- **Homogeneous vs. heterogeneous processing model**
- **Single/multi process vs. multi-threaded**
- **Data-centric software design or not?**
- **Kernels vs. heavy code**
- **Program for specific architectures or not?**

Key aspects in any choice



The complexity of a large software project

- Strategy and hardware-related requirements are a must when the hardware is a variable
- Hard to plan for unknowns, but easier to plan for changes
- Software middle-men threaten scalability
- Long time to produce and stabilize
 - Consequently: faraway targets should be considered, not current

How to program for a moving target?

- The question is not “whether” to take advantage of parallelism, but “how?” and “who will take care of it”?
 - Should the Physicist be oblivious to the hardware (in particular, parallelism), or not?
 - Are the hardware vendors lagging behind in support for developers?
- Is many-core and the return of vectors THE revolution or does something else await?
- Memory issues – present and future
- In any case, for various reasons HEP is several years late to the game

Recommendations

- **introduce a systematized R&D program focused on parallelism with deliverables**
- **restate the real needs of the HEP community starting with a tabula rasa**
- **setting clear, realistic, contextualized goals for development in line with the real needs of the HEP community**
- **devising better metrics for performance and taxing for violations**
- **implementing a scalability process focused on rapid response**
- **promoting joint work across stakeholders to share the load**
- **a careful embrace of emerging technology**
- **a conscious consideration of where any performance gains should be reinvested (e.g. reduced cost, improved raw performance, accuracy etc)**

THANK YOU

Q & A



CERN
openlab

Backup: About openlab

- CERN openlab is a framework for evaluating and integrating cutting-edge IT technologies in partnership with the industry:
<http://cern.ch/openlab>
- The Platform Competence Center (PCC) of the CERN openlab has worked closely with Intel for the past decade and focuses on:
 - many-core scalability
 - performance tuning and optimization
 - benchmarking and thermal optimization
 - teaching