Current and future developments in commodity computing and many-core

PH-SFT Forum on Concurrent Programming Models and Frameworks
April 25th 2012

Andrzej Nowak, CERN openlab

CERN openlab

“I think we should first agree what it is we will need to agree on”

(Quote from the first meeting)

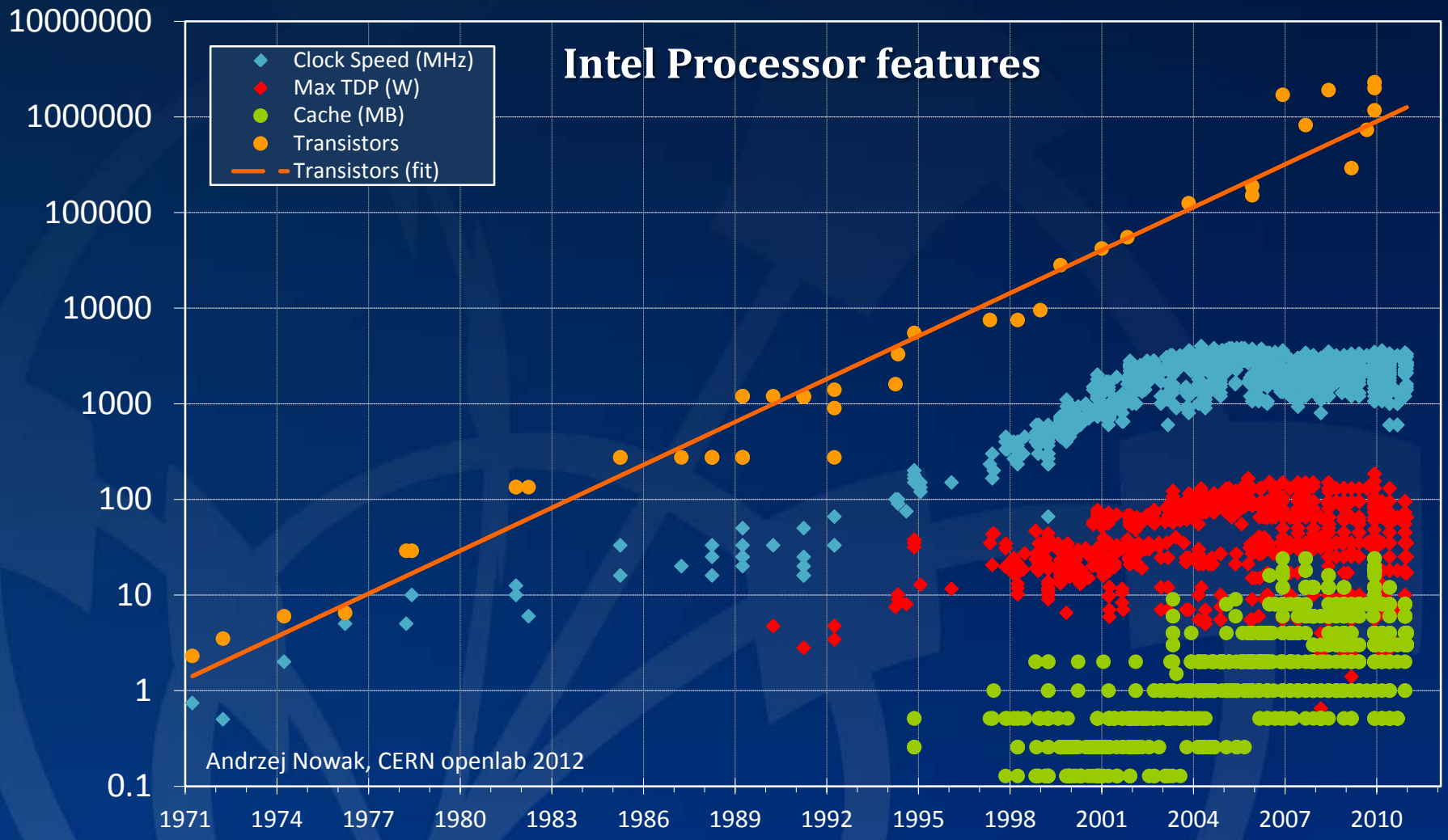
Agenda

1. Why co-design is important
2. Hardware landscape today (x86)
3. Major trends and shifts
4. Tradeoffs and how to program for a moving target?
5. Summary, 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 landscape



Hardware landscape - benefits

- **The days of worrying over IBM, Cray, Apollo, VAX are gone**
  - single system
  - single hardware model, moving towards even further homogenity
- **Continued growth, but not in a software-transparent way**
- **Well-defined, expandable, multiplicative performance dimensions**

Parallelism everywhere

- systems
- sockets
- cores
- hardware threading
- instruction level (superscalar)
- pipelining
- vectors

Vectors

- **Comeback with a vengeance but lessons learned 20 years ago**
- **Good news:**
  - can now hold 4 doubles
  - only one architecture to worry about
  - plenty of technologies to choose from
  - compilers getting increasingly better at autovectorization (can get 2x)
- **Bad news:**
  - increasingly a key element in the performance equation
  - not everything will vectorize
  - iterative and auto-vectorization are promoted, but are not the solution for HEP
  - good vectorization requires a data centric design (sacrifices have to be made)

ILP considerations

- CPI for HEP code is often too high, literally wasting >50% of CPU power
- CPI figures for the major experiments hover around 1.5-2.5
- Significant side-effects of C++ abuse
  - Very frequent jumps and calls + more
- Pipelining not discussed explicitly as it folds into ILP

Hardware threading

- A part of the solution to bad ILP
- Free money
  - Usually between +20% and +30%
- But does not help in all cases
- Not extensively employed, but now seriously considered

Many-core, multi-socket, etc.

- **A typical workhorse machine has 12-16 cores**
  - Some have 48 (AMD)
- **HEP appears to be doing well using multiple machines and sockets**
- **Many-core is not multi-core**
  - Memory hierarchy issues pop up
    - Cache coherency
    - NUMA
    - Memory bandwidth or IO paths may be constraining
  - Strong scalability tanks (need weakly scalable workloads)
- **Multiprocess is a convenient model that does the job, but it is not sustainable nor scalable**

Where are we now?

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	4	1.35	12	4
TYPICAL	2.5	1.4	1.25	8	2
HEP	1	0.55	1	6	2

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	4	16	21.6	259.2	1036.8
TYPICAL	2.5	3.5	4.375	35	70
HEP	1	0.55	0.55	3.3	6.6

Trends

- **IO, disk and memory do not progress at the same rate as compute power**
  - bytes/FLOP decreasing
  - pJ/FLOP decreasing
- **# 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**
  - The trend is important, not the end amount
- **Two factors to consider:**
  - Enterprise and HPC-targeted developments "trickle down" to support datacenter developments (where cost effective)

Where will we be tomorrow?

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	8	4	1.35	16	4
TYPICAL	6	1.4	1.25	10	2
HEP	1	0.55	1	8	2

	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	8	32	43.2	691.2	2764.8
TYPICAL	6	8.4	10.5	105	210
HEP	1	0.55	0.55	4.4	8.8

Upcoming hardware – Intel SNB/IVB

- **Sandy Bridge-EP is entering the datacenter now**
  - 256 bit AVX
  - 4 socket EP part (but no 4- or 8- socket EX)
  - 8 cores per chip vs. 6 with Westmere
  - As usual, evaluation published by openlab: http://cern.ch/openlab
- **Ivy Bridge-EP is a shrink to 22nm (minor improvements)**

Upcoming hardware – Intel HSW

- **Particular Haswell features:**
  - AVX2 (256 bit integer, FMA3)
  - Transactional memory
    - Transactional regions
    - Hardware Lock Elision (HLE) – EAFP concept
    - Restricted Transactional Memory (RTM) – transaction and fallback setup, lower level than HLE
  - Core count increase
- **The following shrink will be "Broadwell" on 14nm**

Upcoming hardware – Intel Skylake

- 2015 / 2016 target
- Core count increase (>100 in a single machine)
- DDR4
- PCI4?
- Successor on 10nm is Skymont

Upcoming hardware – 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 toolchain
- Projected end of 2012 release, unknown cost

Intel KNC - Particular opportunities for HEP

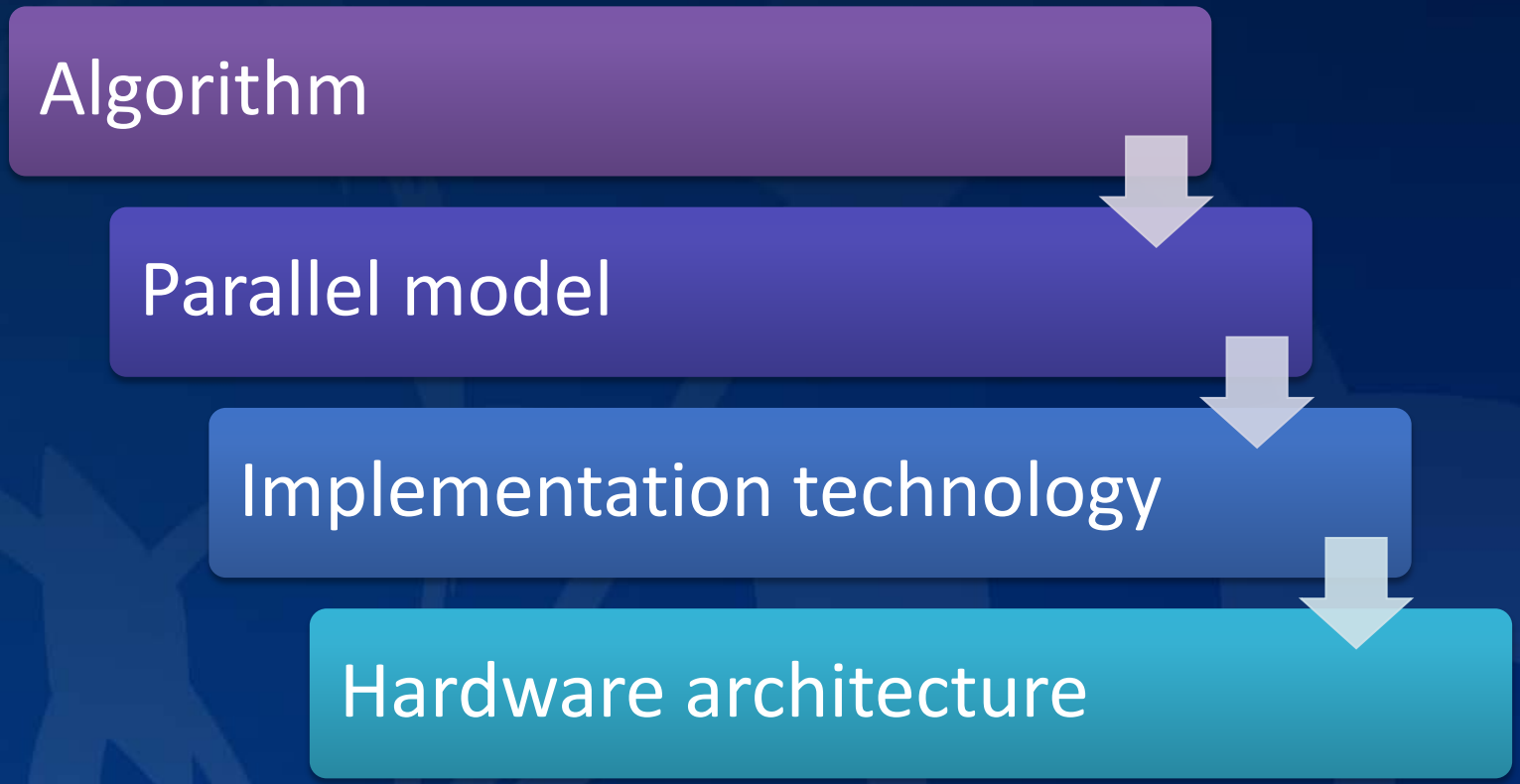
- **Potential use:**
  - as a general purpose performance expansion in workstations
  - as an accelerator in datacenters
- **Highly programmable vector machine**
- **Direct access and flexibility**
- **At CERN: in-house expertise (openlab)**
- **BUT: your machine now becomes a fat/small core mix**

Knights vs. GPU

- See paper from Victor Lee: "Debunking the 100X GPU vs. CPU myth"

	GPU	MIC / KNC
Architecture	Proprietary	x86
Language	Limited to vendor API Derived	Standard C, C++, Fortran
Programmability	Limited to vendor API	Mainstream techs
Xeon optimization payoff	No	Yes
Direct access	No	Yes
IEEE 754 compliance	Evolving	Standard
SSE compatibility	No	Yes
Binary FP compat. w/ Xeon	No	?
Performance	Known	To be established
Price	Acceptable	To be established

The technology stack



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 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

- Preceding scientific evaluation
- Is the choice realistic?
- Is it necessary?
- Is it achievable?
- Maturity
- Longevity
- Openness
- External support

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
- Long time to produce and stabilize
  - Consequently: faraway targets should be considered, not current

Code scalability and middle-men

- Ideally, code should flexibly scale in various performance dimensions independently
- The parallel runtime, if any, acts as a middle-man
- There are numerous proofs that parallel runtimes can obstruct execution and reduce performance
  - 2x drop is routine
  - A tradeoff
  - Still some way to go

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?**

Recommendations

- Set clear, realistic, contextualized goals before embarking on a long journey
- Devise metrics for performance and tax for violations
- Implement a scalability process
- Promote joint work across stakeholders to share the load
- Careful embrace of emerging technology
- Performance gains can translate to various end results: better performance, accuracy, reduced cost etc.

Summary

Program for tomorrow's hardware today

THANK YOU

Q & A

CERN openlab

Backup: About openlab

- **CERN openlab is a framework for evaluating and integrating cutting-edge IT technologies or services in partnership with 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