The growth of commodity computing and HEP software – do they mix?

ATLAS Software & Computing Week

June 11th 2012

Andrzej Nowak, CERN openlab

based on the work of Sverre Jarp, Alfio Lazzaro, Julien Leduc, Andrzej Nowak

CERN openlab

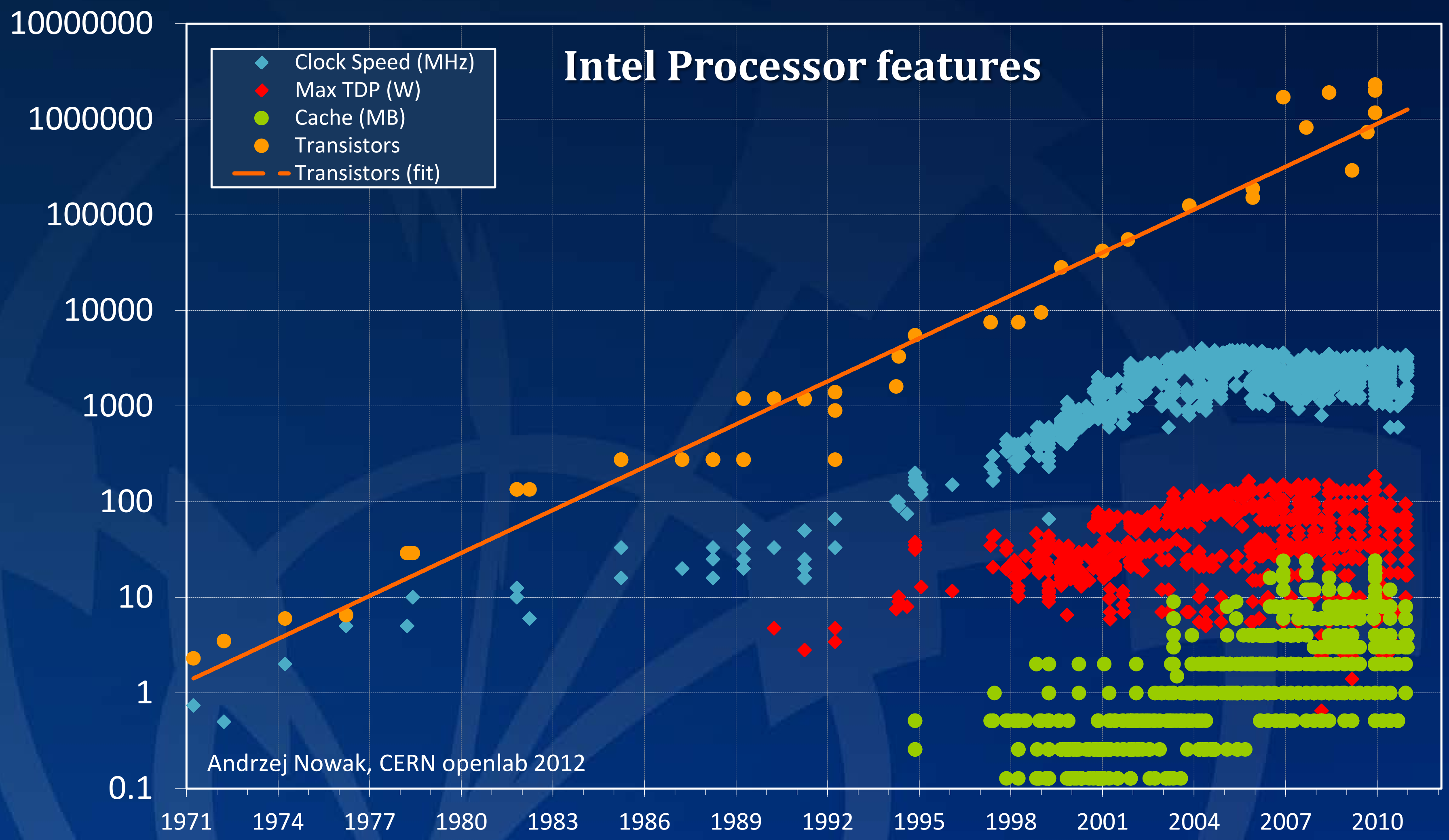
Agenda

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**

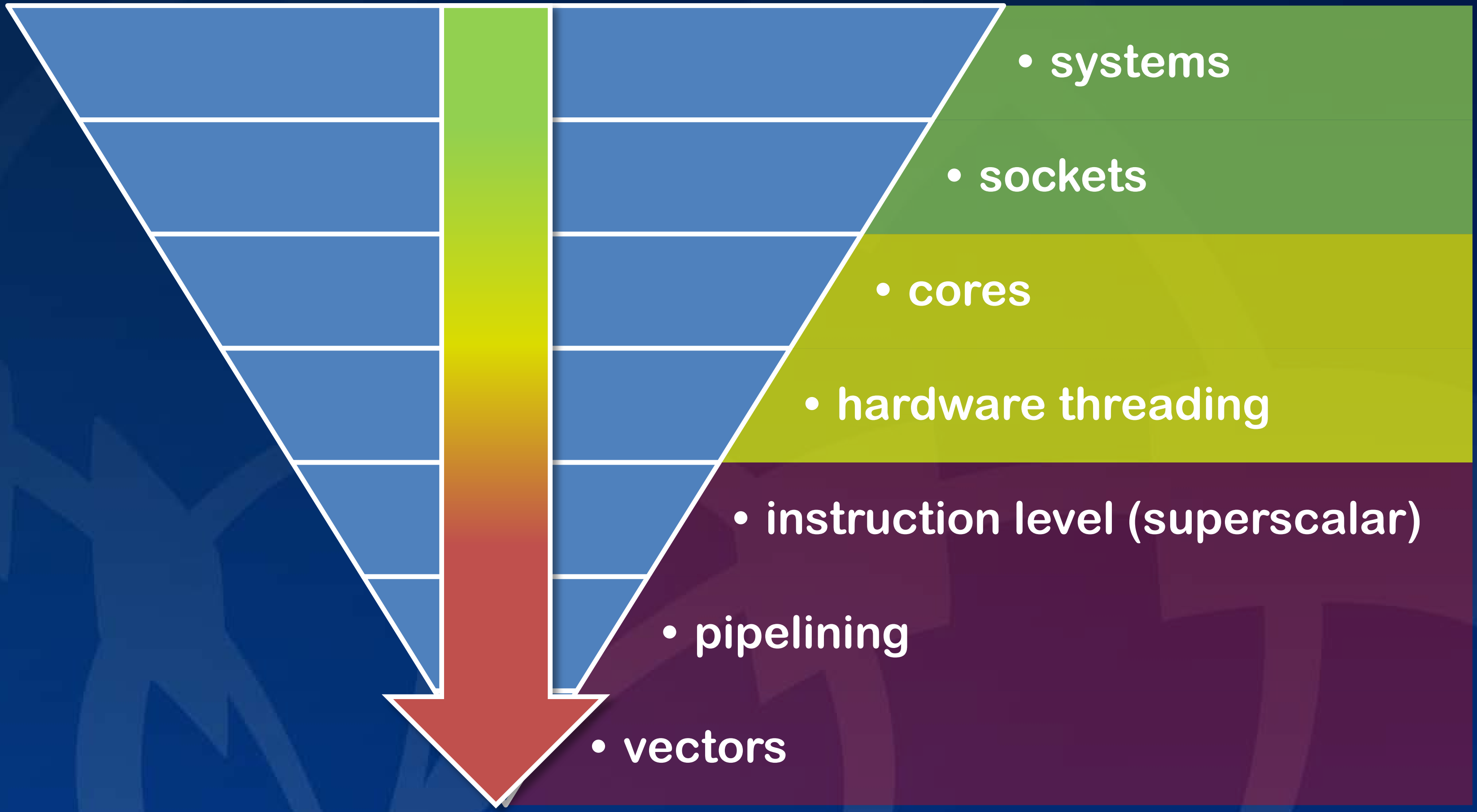
Hardware landscape



Hardware landscape – the 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

Omnipresent multiplicative parallelism



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

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

Medium-term hardware trends

- Pricing follows market pressure, not technology
- IO, disk and memory do not progress at the same rate as compute power
  - bytes/FLOP decreasing
  - pJ/FLOP decreasing
- Bulk of improvements in x86 comes from Moore's Law still being in effect
- 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

Vectors

- **Comeback with a vengeance but lessons learned 20 years ago: growing substantially**
  - 128-bit SSE → 256-bit AVX (designed for more)
  - AVX: new execution units
  - LRBni (Intel MIC): 512 bits, new vector instructions, FMA, 3-4op
- **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 “magic bulet” solution for HEP
  - good vectorization requires a data centric design (sacrifices have to be made)
  - bad past experience in HEP

Intel64 & ILP considerations

- **x86 microarchitecture**
  - steady, but limited improvements (<10% per "tock")
  - increasingly advanced features – can HEP benefit?
- **Frequency – very modest changes, if any**
  - Rise of the Turbo boost
- **CPI for HEP code is often too high, literally wasting CPU power**
  - CPI figures for the major experiments hover around 0.9-1.5
- **Significant side-effects of C++ abuse**
  - Very frequent jumps and calls + more
  - Dynamic code has penalties – x86 is already quite good at executing it but there are limits
- **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%
  - Recommendation: use whenever possible, good for simulation
- Does not help in all cases
- Not extensively employed, but now seriously considered

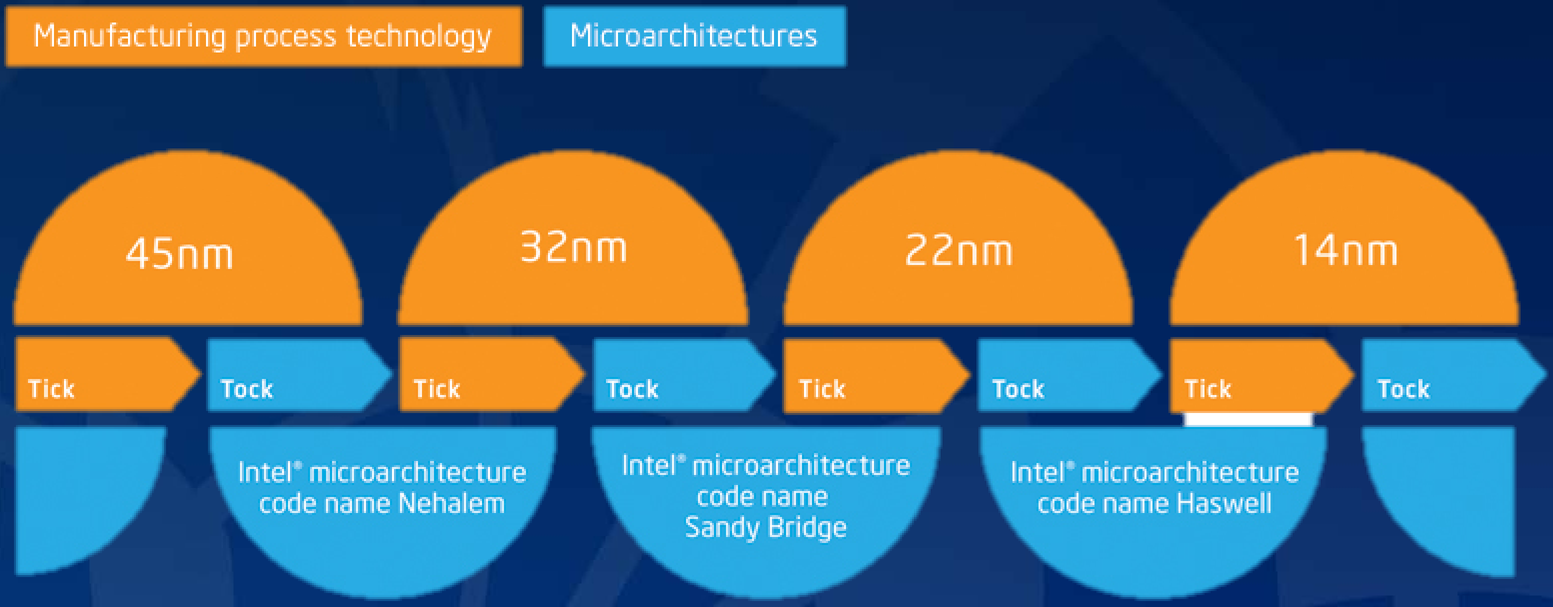
Multi-core vs. many-core

- **A typical modern workhorse machine has 12-16 cores**
  - Some have 48 (AMD)
  - CERN (and many Grid sites) are often 1 or 2 generations behind
- **# 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 (per platform)**
- **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
  - Is the trend sustainable? What about all these transistors?

Multiple sockets and systems

- **Sockets – slight growth with a limit, ultimately impacts core count per platform**
- **HEP appears to be doing well using multiple machines and sockets, but is not buying latest hardware**
- **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**

The tick-tock model


Manufacturing process technology
Microarchitectures
45nm
32nm
22nm
14nm
Tick
Tock
Tick
Tock
Tick
Tock
Tick
Tock
Intel® microarchitecture code name Nehalem
Intel® microarchitecture code name Sandy Bridge
Intel® microarchitecture code name Haswell

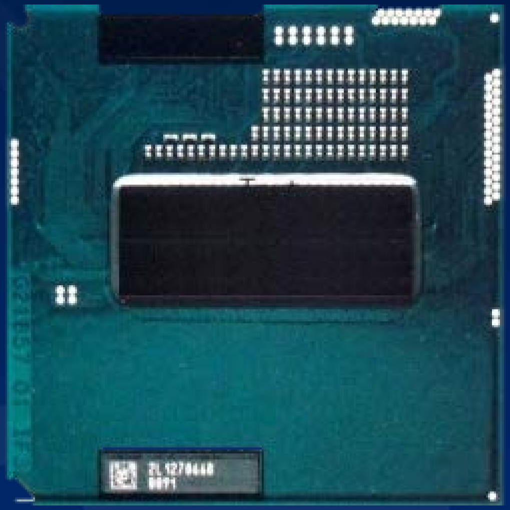
Source: Intel

Upcoming hardware – Intel SNB/IVB

- Sandy Bridge-EP entered the datacenter last quarter
  - 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 (also a paper at CHEP 2012)
- Ivy Bridge-EP is a shrink to 22nm (minor improvements, same platform)

Upcoming hardware – Intel HSW

- **Particular Haswell features:**
  - AVX 2 (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**



Leaked HSW spyshot
Source: bit-tech

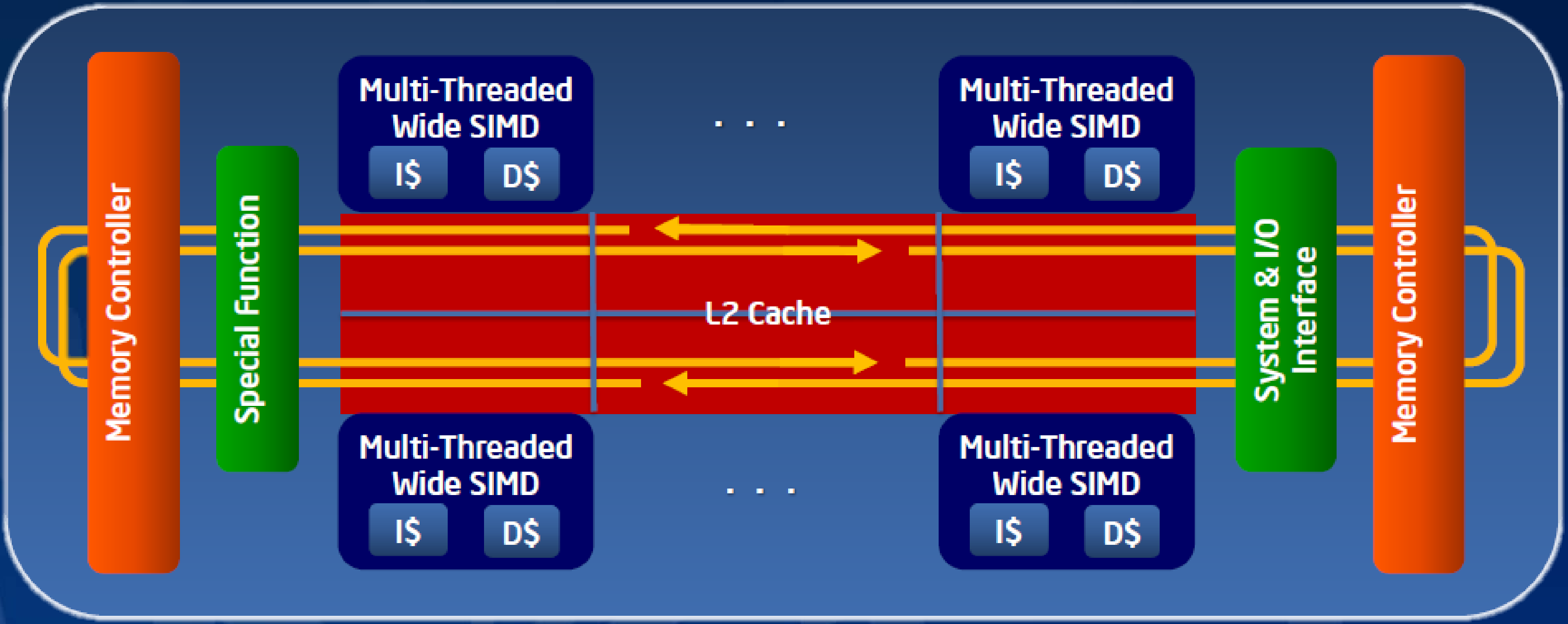
Upcoming hardware – Intel Skylake

- 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
- Promising, but is it indicative of future architectures?

KNF/KNC architecture



Source: Intel

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 involved for many years)
- 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

Disruptive technologies on the horizon

- Phase-change memory
- Memristors
- Silicon photonics
- Nanocircuits
- Stacking
- New materials

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?

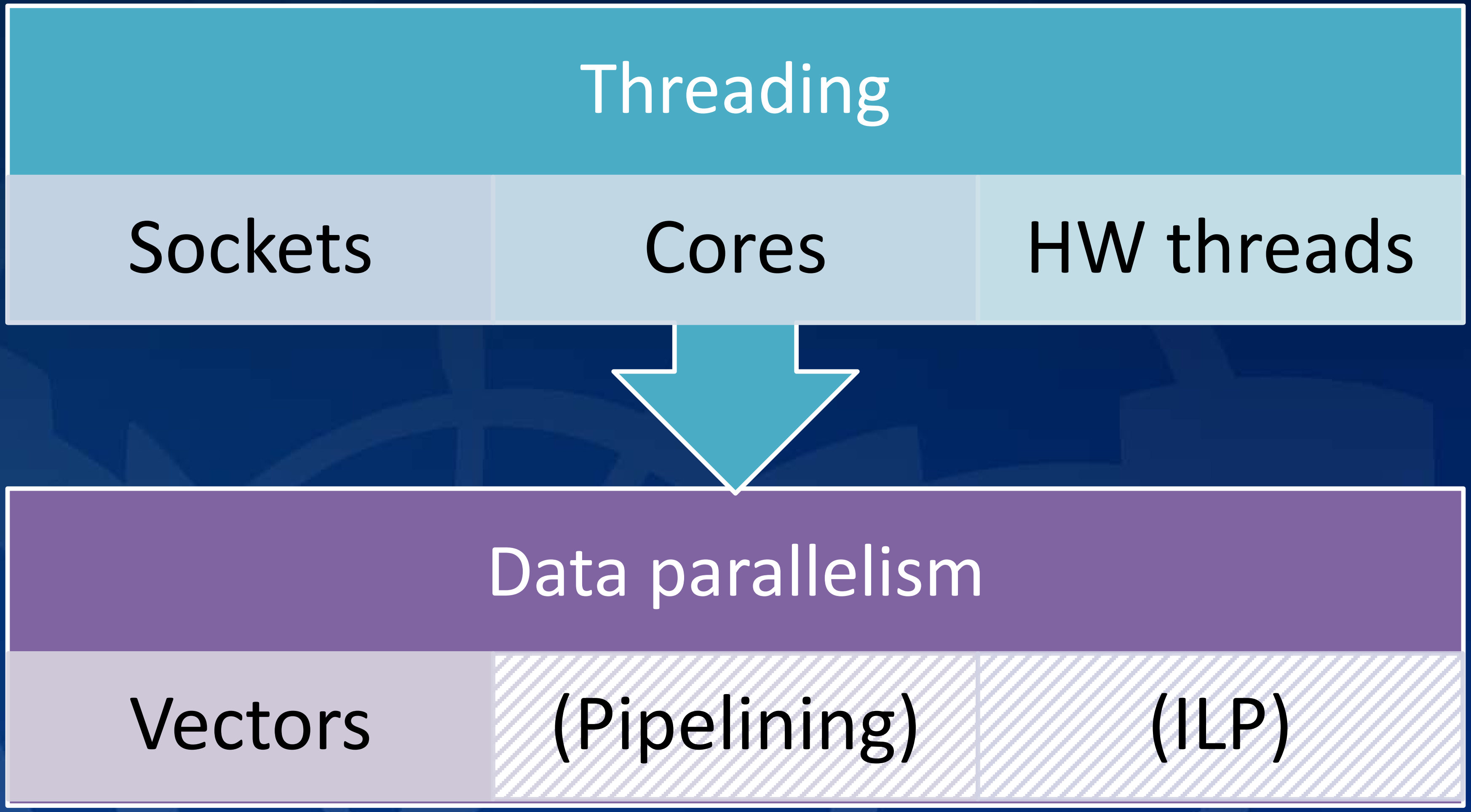
	SIMD	ILP	HW THREADS	CORES	SOCKETS
MAX	8	4	1.35	12	4
TYPICAL	6	1.57	1.25	10	2
HEP	1	0.80	1.25	8	2

	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

SW performance dimensions



Threading

Sockets

Cores

HW threads

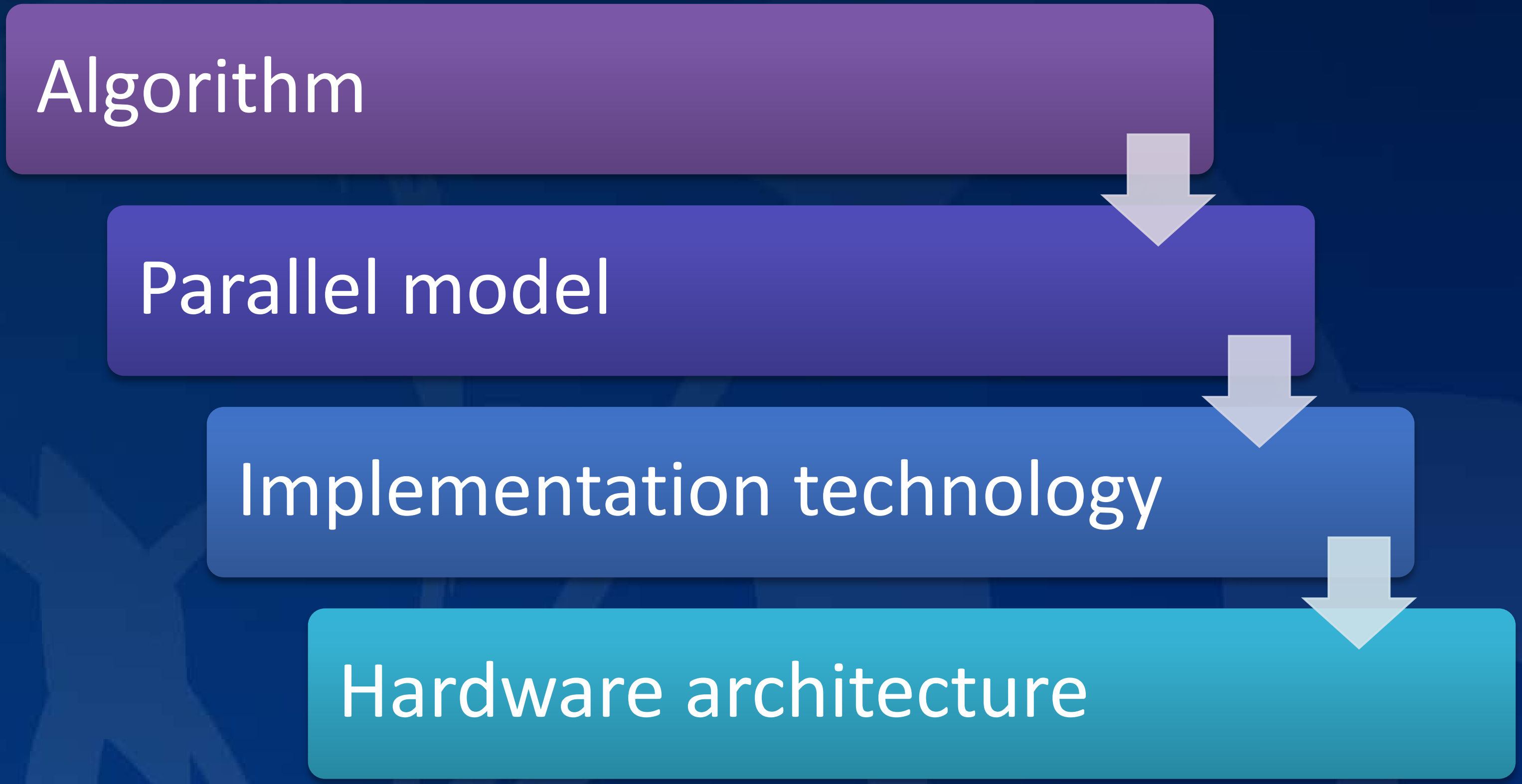
Data parallelism

Vectors

(Pipelining)

(ILP)

The parallel technology stack



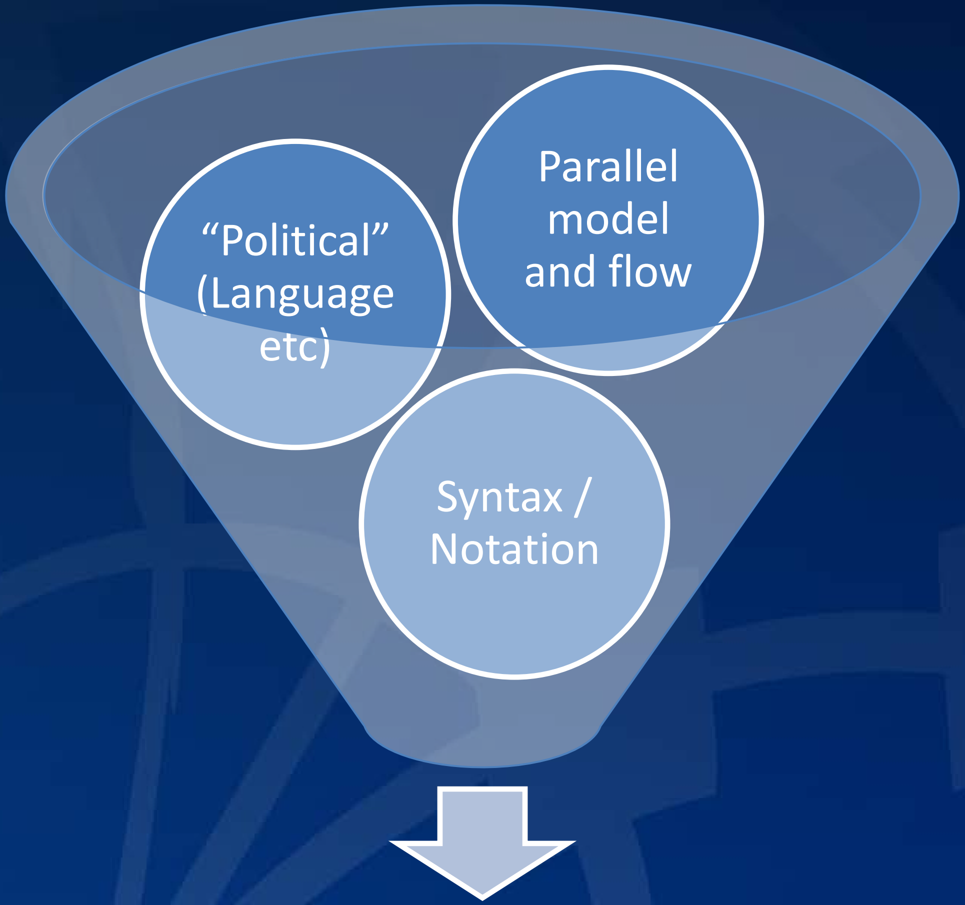
Algorithm

Parallel model

Implementation technology

Hardware architecture

Interactions



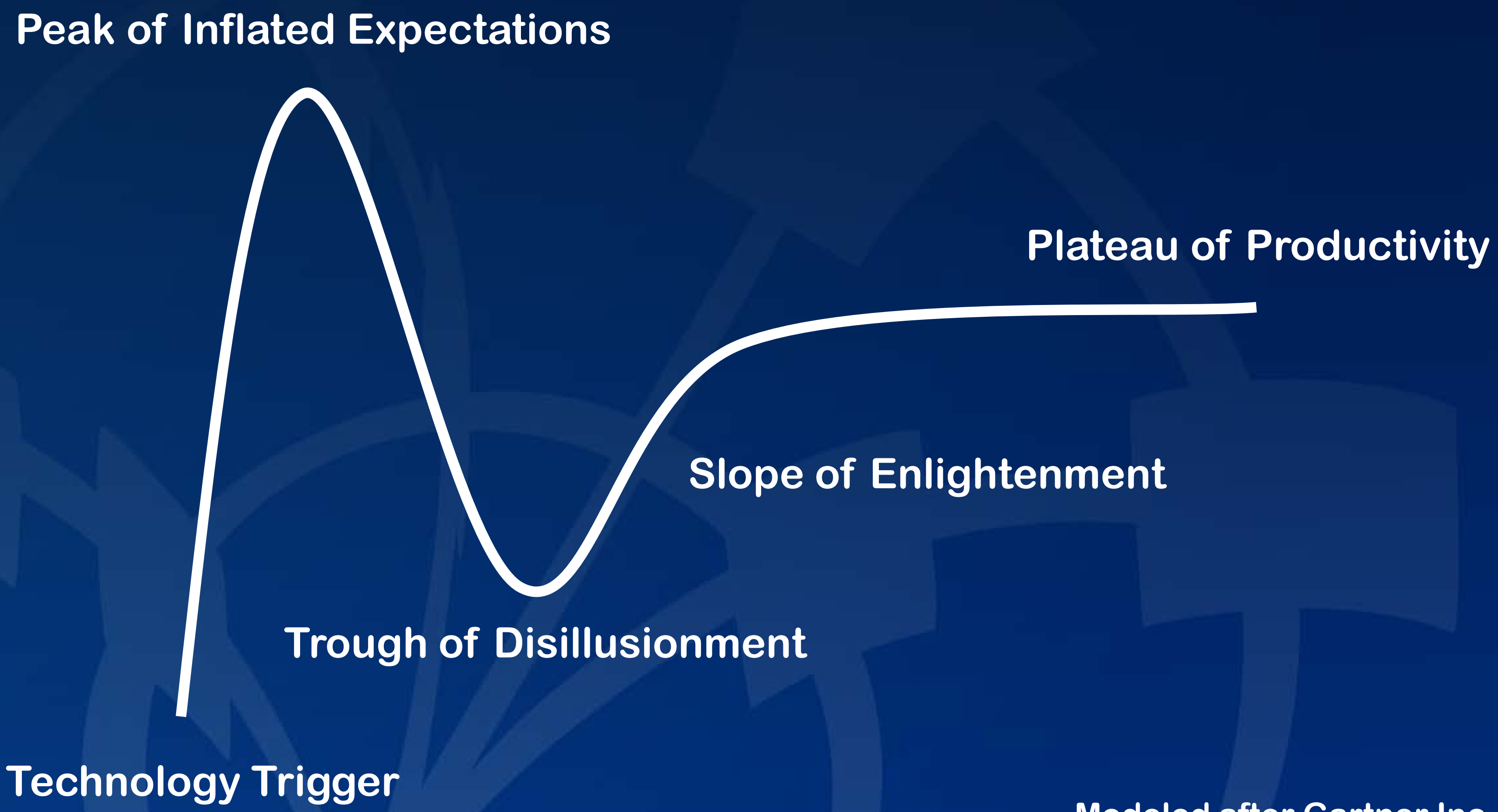
Implementation technology

A plethora of options (subset)

- OpenMP
- CUDA
- pthreads
- MPI
- Cilk
- Ct/RapidMind/ArBB
- TBB
- OpenCL
- Boost threads
- Concurrent Collections

- Less mainstream:
  - Axum
  - Co-array Fortran
  - UPC
  - Go
  - Chapel
  - Fortress
  - X10
  - Erlang
  - Linda
  - Haskell

The Hype Cycle

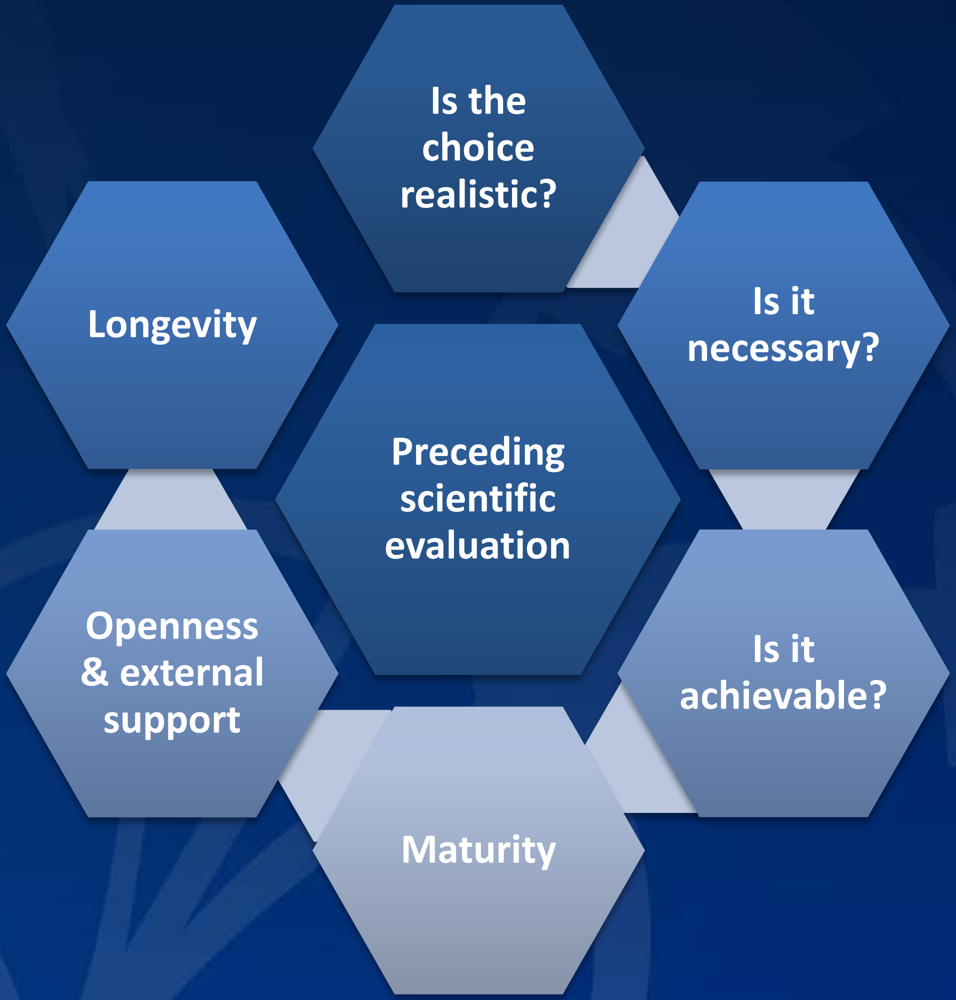


Modeled after Gartner Inc.

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



Is the choice realistic?

Is it necessary?

Longevity

Preceding scientific evaluation

Openness & external support

Is it achievable?

Maturity

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
- Requirements management
- Software middle-men threaten scalability
- 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?
- Memory issues – present and future
- In any case, for various reasons HEP is several years late to the game

Recommendations

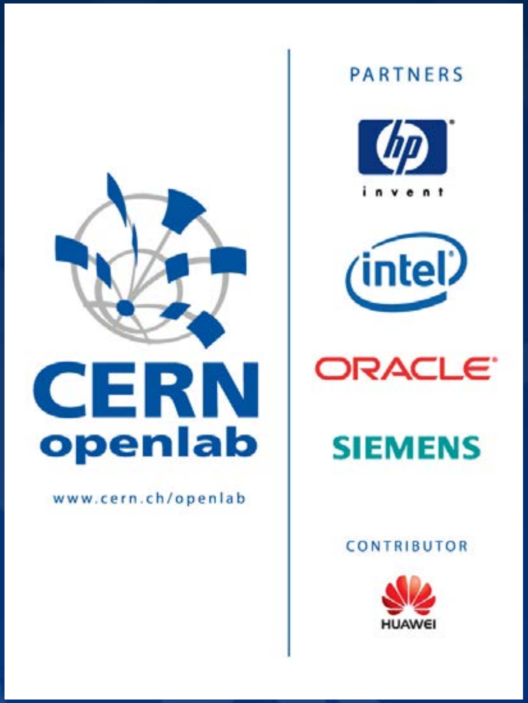
- introduce a systematized R&D program focused on parallelism with deliverables
- restate the real needs of the HEP community starting with a tabula rasa
- setting clear, realistic, contextualized goals for development in line with the real needs of the HEP community
- devising better metrics for performance 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