



"Architectures, tools and methodologies for developing efficient large scale scientific computing applications"

Ce.U.B. - Bertinoro (FC) 19 - 25 October 2014



Introduction & Concepts of
Performance and Efficiency

Peter Elmer - Princeton University

ESC School Goals

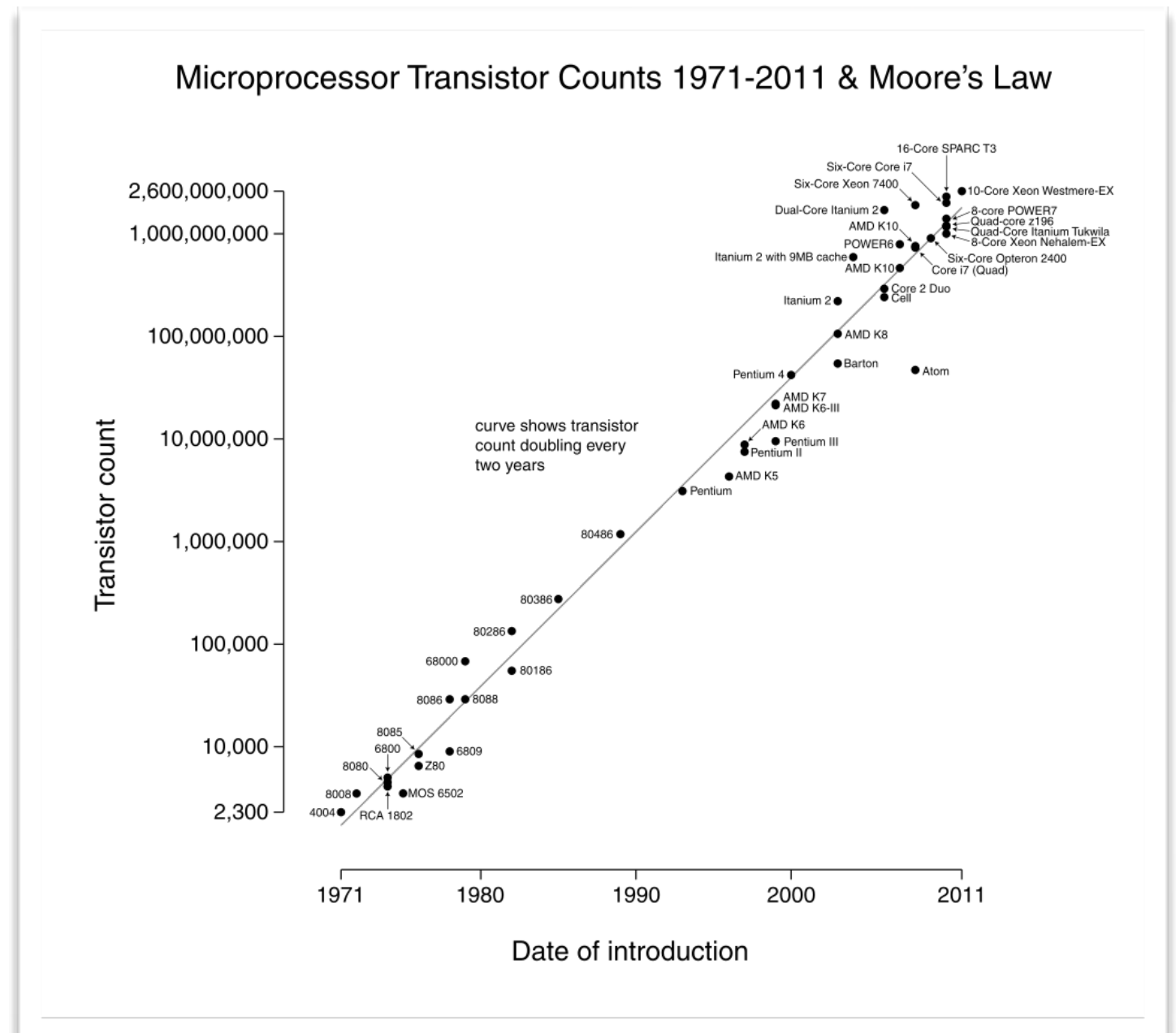
WELCOME TO ESC14

The school offers a limited number of young students and researchers an invaluable opportunity to learn from very qualified and experienced scientists how to exploit at best the evolution of modern computing systems used in doing science.



Moore's Law

- Cost of transistors drops exponentially over time, permitting chips with ever greater numbers of transistors.
- What matters more is how these transistors are deployed to achieve ever greater (exponential) growth in application performance.
- Through about 2005 this was done in such a way as to increase the performance of single, sequential applications: you could run the same software (and usually the same binary) on a newer processor and it would run faster.

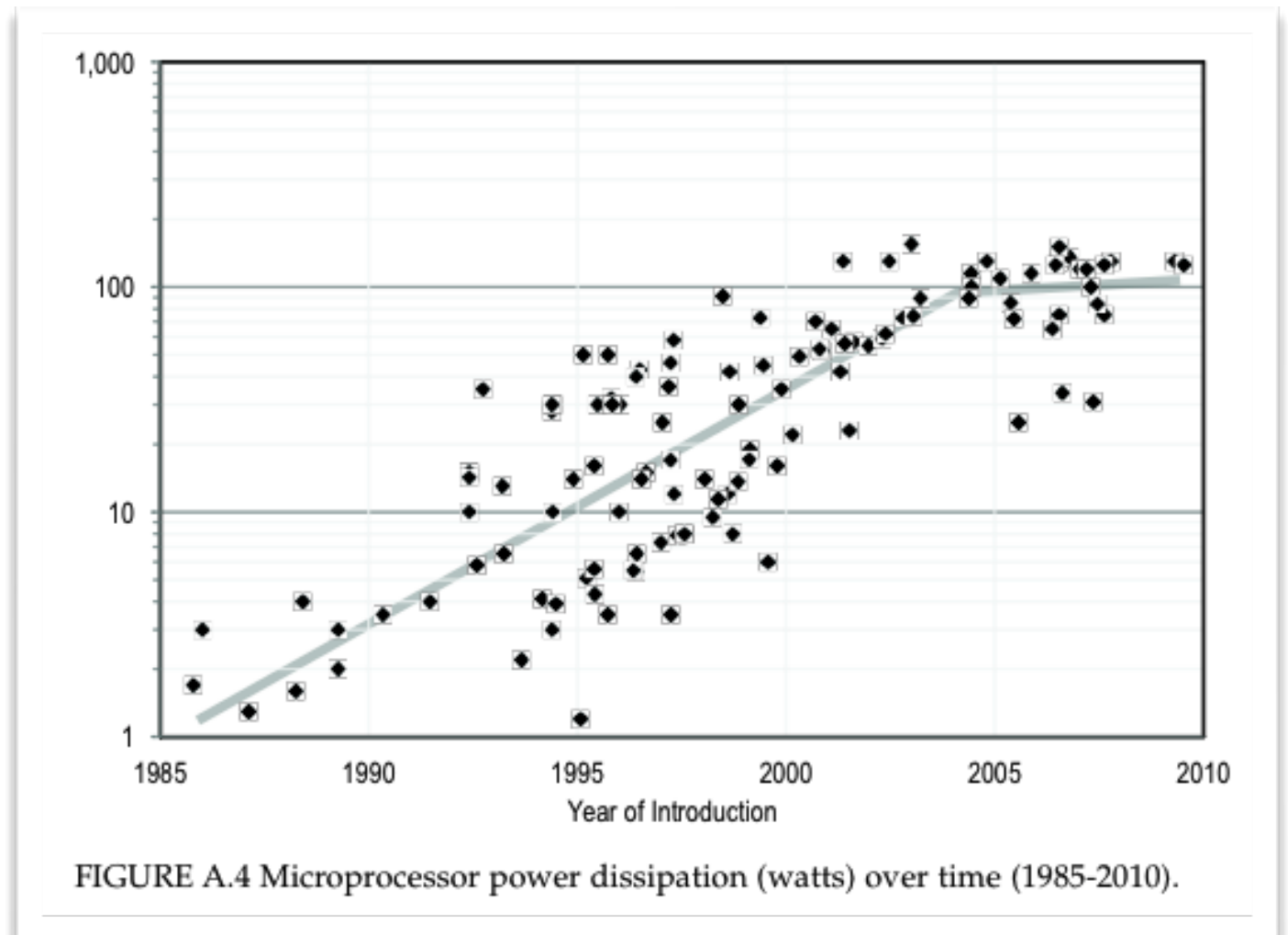


Moore's Law

- Moore's Law had a profound effect on many things
- One could plan for things that were impossible today (by very large factors) and make cost estimates 10 years out
- It also had an important effect on software development: if you know you will have (exponential) performance gains from new hardware, more emphasis can be placed on other important aspects of software engineering: maintainability, extensibility, portability, etc.

New Architectures

- Over the past ten years processors have hit power limitations which place significant constraints on "Moore's Law" scaling.
- The first casualty was scaling for single sequential applications, giving birth to multi-core processors.



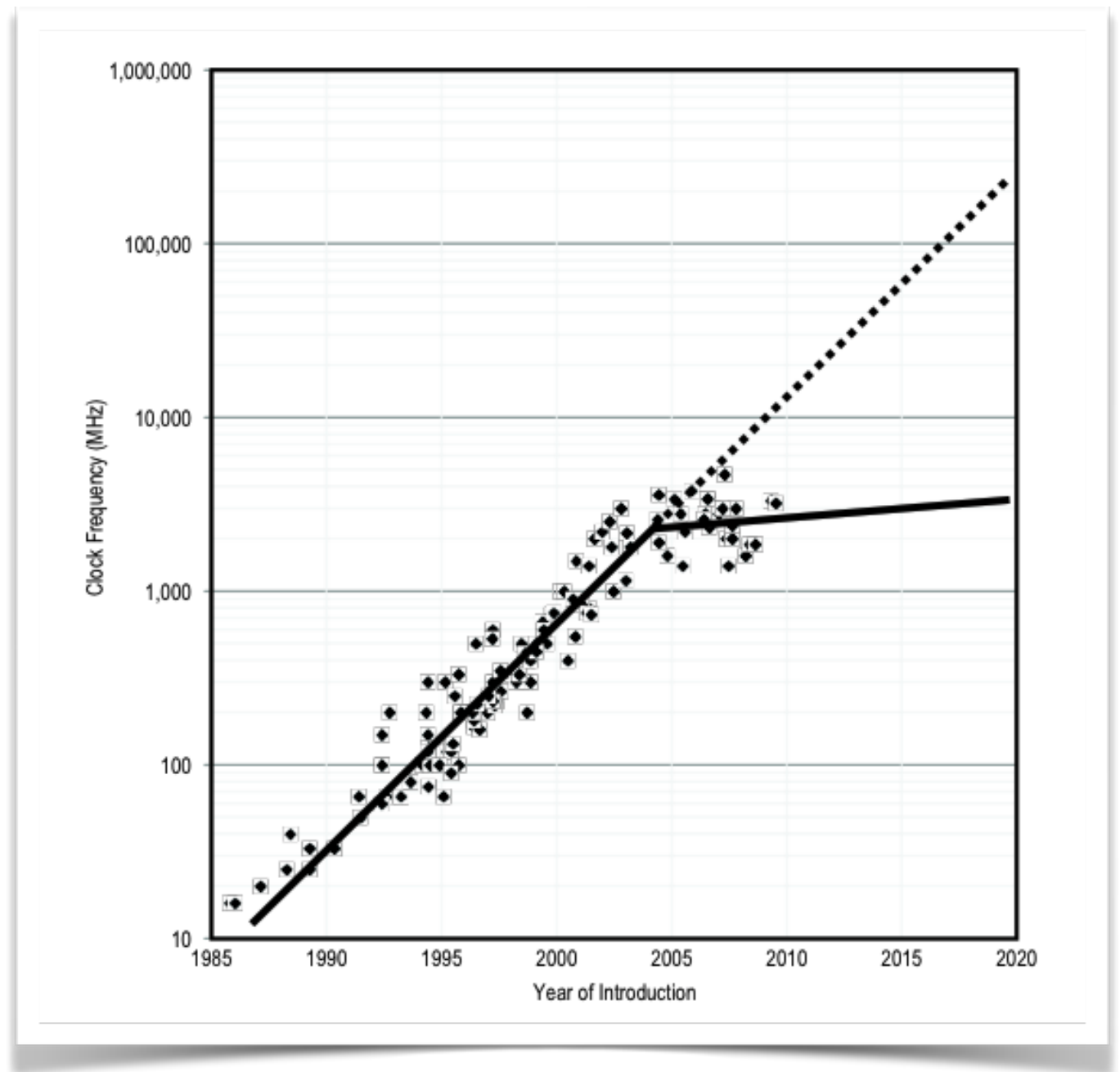
From: "The Future of Computing Performance: Game Over or Next Level?"

Multicore

- First response to power limits was the deployment of multicore CPU's, with more than one general-purpose functional processor on a chip
- While there were no performance gains for single, sequential applications, simple threaded applications and applications which could profitably run more than one instance (true for all of high throughput and high performance scientific computing) could benefit.
- The main downside is that some costs (memory, #database or file connections) now grow with each generation of processor

New Architectures

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.



From: "The Future of Computing Performance: Game Over or Next Level?"

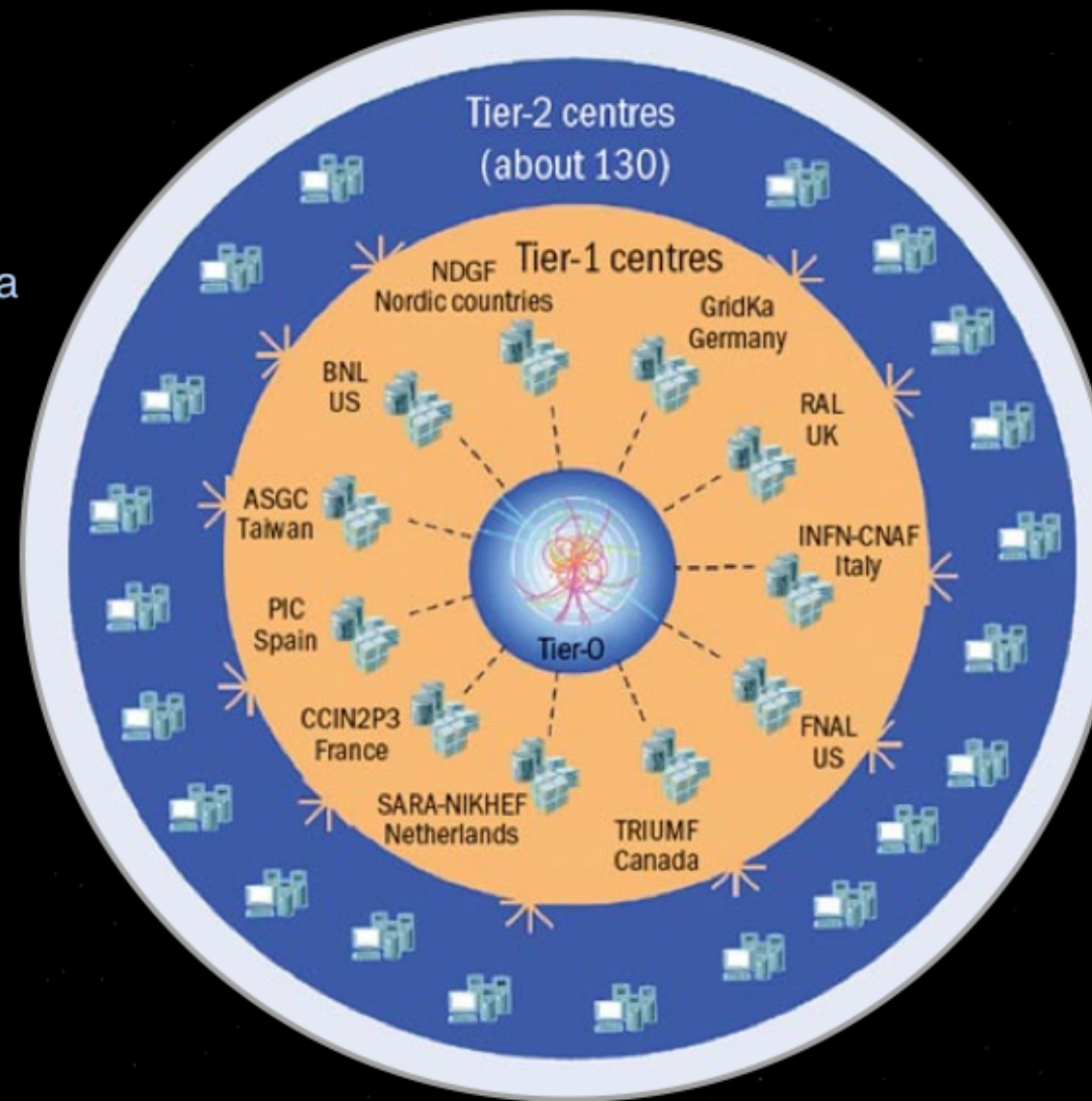
WLCG as Distributed Supercomputer

The Worldwide LHC Computing Grid

Tier-0 (CERN): data recording, reconstruction and distribution

Tier-1: permanent storage, re-processing, analysis

Tier-2: Simulation, end-user analysis



nearly 160 sites,
35 countries

~350'000 cores

200 PB of storage

> 2 million jobs/day

10 Gb links

WLCG:

An International collaboration to distribute and analyse LHC data

Integrates computer centres worldwide that provide computing and storage resource into a single infrastructure accessible by all LHC physicists

WLCG as Distributed Supercomputer - Power

- Not only would the the WLCG be one of the top supercomputers in terms of performance if it were considered as such, but it also shares another characteristic which is less obvious.
- Using the mix of hardware available at FNAL (and known power use), we estimate the aggregate power cost to be of order 10MW

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National University of Defense Technology China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer , SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Texas Advanced Computing Center/Univ. of Texas United States	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
7	Forschungszentrum Juelich (FZJ) Germany	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
8	DOE/NNSA/LLNL United States	Vulcan - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
9	Leibniz Rechenzentrum Germany	SuperMUC - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423
10	National Supercomputing Center in Tianjin China	Tianhe-1A - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT	186368	2566.0	4701.0	4040

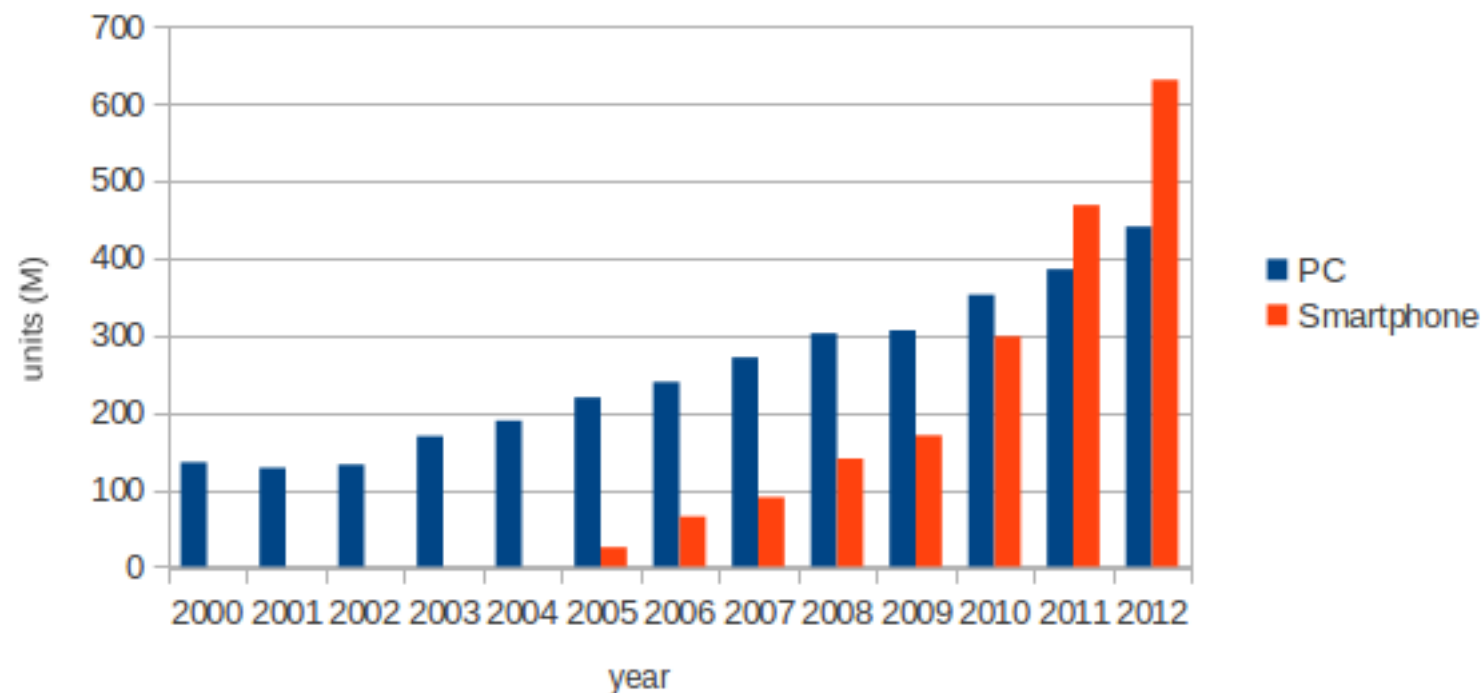
Indicative technologies today

- ARM processors - low power, higher performance/watt, simpler processor cores (mobile market more explicitly dominated by power)
- Graphics processing units (GPU) - e.g. Tesla from NVidia, coprocessors
- Xeon Phi (Intel MIC architecture) - coprocessor with ~60 lightweight in-order cores, each with big vector units
- Not unlikely that the next step will be heterogenous mixes

Market Evolution - ARM processors

- RISC processor with a long history going back to the BBC Micro. Of interest today as the core processor used in the vast majority of mobile devices.
- Current generation ARMv7/32bit, ARMv8/64bit products expected in 2014

PC and Smartphone sales per year



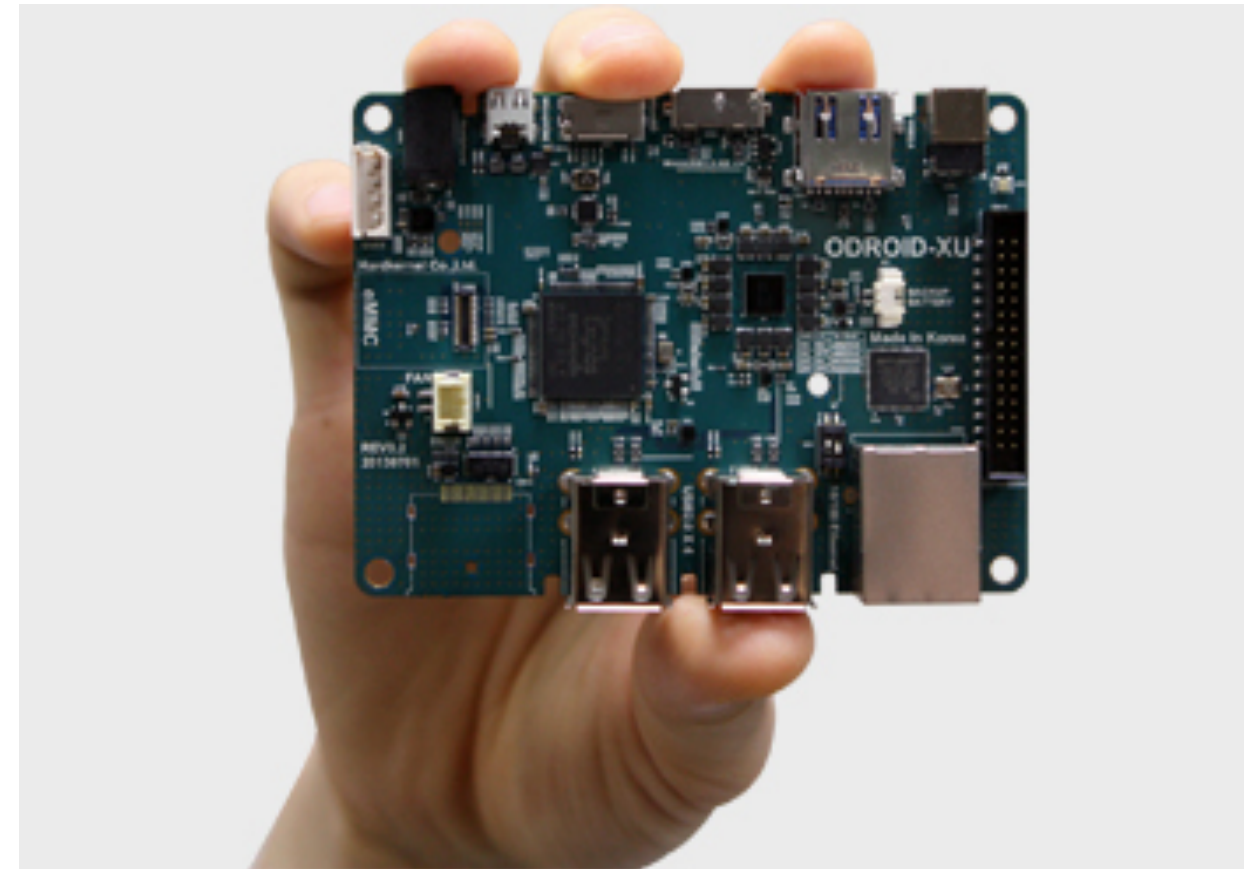
- Unit sales increasing dramatically in recent years (typically cost and profit/unit, however)

Market Evolution

- ARM processors may or may not make it into the (micro-)server market. Intel of course is preparing a new generation of System-On-Chip (SoC) low power processors, too, for the mobile, embedded and micro-server environments: see Atom/Silvermont.
- The important thing to recall is that designing processors and building chip fabs is a very expensive game, especially with the latest fabrication processes (needed to push onwards as per Moore's Law)
- Big markets are thus needed for economies of scale and suitable profits

Example ARM System-on-Chip (SoC)

- Here in ODROID XU+E dev. board, processor from Samsung Galaxy S4
- Exynos5 octa (5410) CPU
- 1.6GHz Cortex-A15 quad core + 1.2GHz Cortex-A7 quad-core (big.LITTLE heterogeneous mix), PowerVR SGX544MP3 GPU
- 2GB L-DDR memory (total)



- Example of what they are trying to do in power-limited mobile market

General Purpose Graphics Programming Unit (GPGPU's)

- Originally used as specialized graphics processors, since around 2005 they have been positioned as providing also general purpose computing capabilities for certain types of applications.
- In particular highly parallelizable compute-intensive applications may see significant performance improvements on GPU's relative to general purpose processors.



Intel Xeon Phi (7110P)

- 61 in-order lightweight cores with big vector units
- Coprocessor packaging on PCIe bus, 16GB GDDR5 memory
- HPC-like competitor for GPGPU's



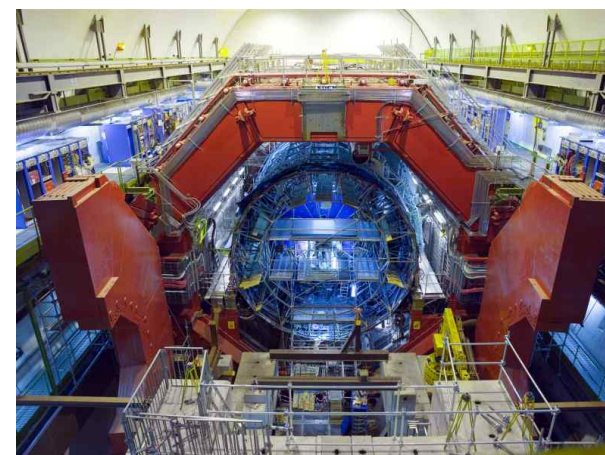
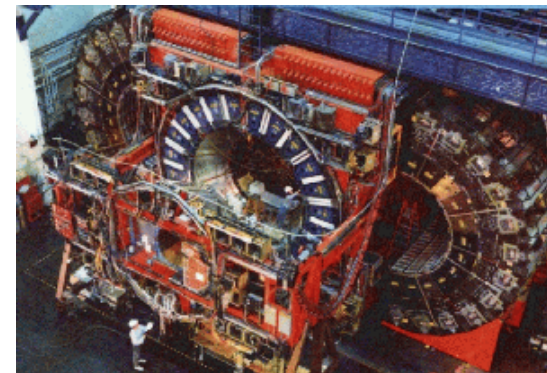
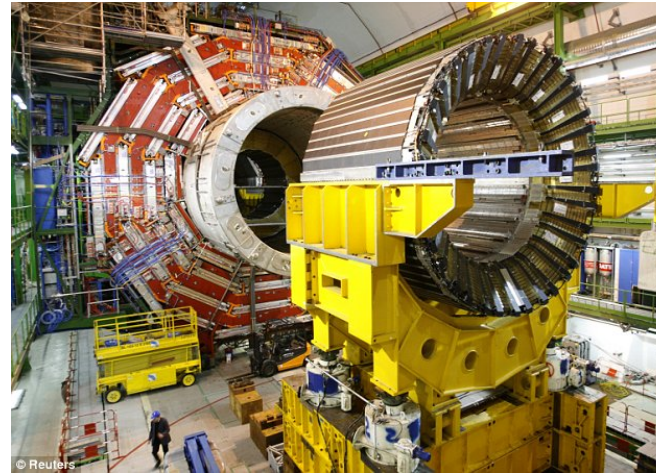
General Features

- Parallelism at multiple levels
- Heterogeneous resources on chips
- Performance/watt likely to be as important a metric as performance/unit-cost

High Energy Physics

HEP computing is *embarrassingly (data) parallel*: N independent instances of an application can be started as simple unix processes, each one processing an independent sets of events. No real communication is needed between the separate processes.

Significant parallelism also exists in many other aspects of our problems: hits, tracks, vertices, jets, etc.



The Art of Application Performance

- What kinds of things are relevant to improve the performance of a single application?
- A number of ingredients affect the realizable performance:

Hardware - CPU, Memory subsystem, I/O

Software - Application code, libraries, compiler
and operating system

Algorithms - CS/Knuth, Scientific, Parallelization

High level algorithm choices

- Often the things which most directly determine the performance are simple choices made as to what the program is actually doing, i.e. the high level algorithms.
- For example, if you are running a simulation: are you simulating only the relevant things? Is the level of detail greater than what is needed or needed for all parts of the simulation?
- Such high level considerations can often result in large factors in the time to completion (or resources needed) for any given task.
- It is important to ask such questions near the beginning, and confirm via profiling that the main performance drivers have been identified, before rolling up one's sleeves and diving into the more technical performance tuning.

Processor hardware/memory/etc.

- We of course compute on actual physical “computers” and thus their evolving capabilities are the most basic component of the achievable performance of some application
- Moore's Law – number of transistors available per unit cost doubles every 1.5 years
- A number of factors conspired to make it possible for many years (1990's through ~2005) to take applications (often without recompiling!) and run them on the next generation of hardware and see a performance gain out-of-the-box.
- This easy ride is over, however. Without changes many applications will not run faster on newer hardware (and many at times actually run slower).
- In addition to “multicore”, exploiting new potentially heterogeneous architectures is a challenge.
- Understanding the basics of how to best exploit the hardware going forward will be the topic of several lectures this week.

Operating Systems

- For the most part Linux is the primary operating system considered in these presentations
- The capabilities of the operating system and its runtime environment have can have an important impact on performance, for example:
 - Virtual Memory subsystem – using or abusing this can affect performance
 - Shared libraries and/or other details of “code packaging” can have an impact on performance
 - Math libraries – by default you may be taking the math library (libm) from the system, unless you've made a conscious decision to do otherwise

Compilers

- The compiler is clearly one of the most important tools for achieving optimum code performance
- Unless we want to hand-code everything in assembly, we rely on it to take our code, written in a high-level language like C++, and produce the fastest code possible.
- Usually we also want it to accomplish that in the shortest time possible, to use as little memory as possible doing it, to produce the smallest code possible, etc.
- Note however that compilers cannot always find and optimize things that a human might immediately recognize. In particular compilers are (usually) conservative and will choose code that is guaranteed to be correct over code that might be wrong in some cases.

GNU Compiler Collection (GCC)

- The workhorse open source compiler, used by most of us, most of the time, these days...
- Front ends for C, C++, Fortran (Ada, Objective-C(++), Java and others)
- Back ends for x86, x86_64 (Alpha, ARM, ia-64, PowerPC, Sparc and many others)
- Most software today is easily configured to build with gcc
- Although most of work on linux/x86(_64) today, or at most MacOSX/x86_64, at least in non-DAQ environments, the wide availability of gcc for different OS/CPU combinations once eased porting C/C++ from one to another.

Intel Compiler (icc)

- Intel's showcase Fortran/C/C++ compiler(s)
- Arguably focused on demonstrating the best possible performance to be obtained from their processors
- Independent compiler (language syntax, code quality)
- Generates code for all of the Intel processors, plus in principle other x86/x86_64 compatible processors, e.g. AMD
- Available for Linux/MacOSX/Windows, proprietary license
- Only realistic compiler today for use with Intel Xeon Phi
- The default behavior for floating point may or may not be what is desired (see presentations about floating point this week)

Clang/LLVM

- "Recent" open source compiler project, aiming to build a set of modular compiler components
- A separate project (Clang) is a front-end for C/C++/Objective-C and is (by now) fairly mature and other front ends have been developed.
- Targets both static compilation as well as just-in-time (JIT) compilation
- Sponsorship (in particular) by Apple

Parallel Programming

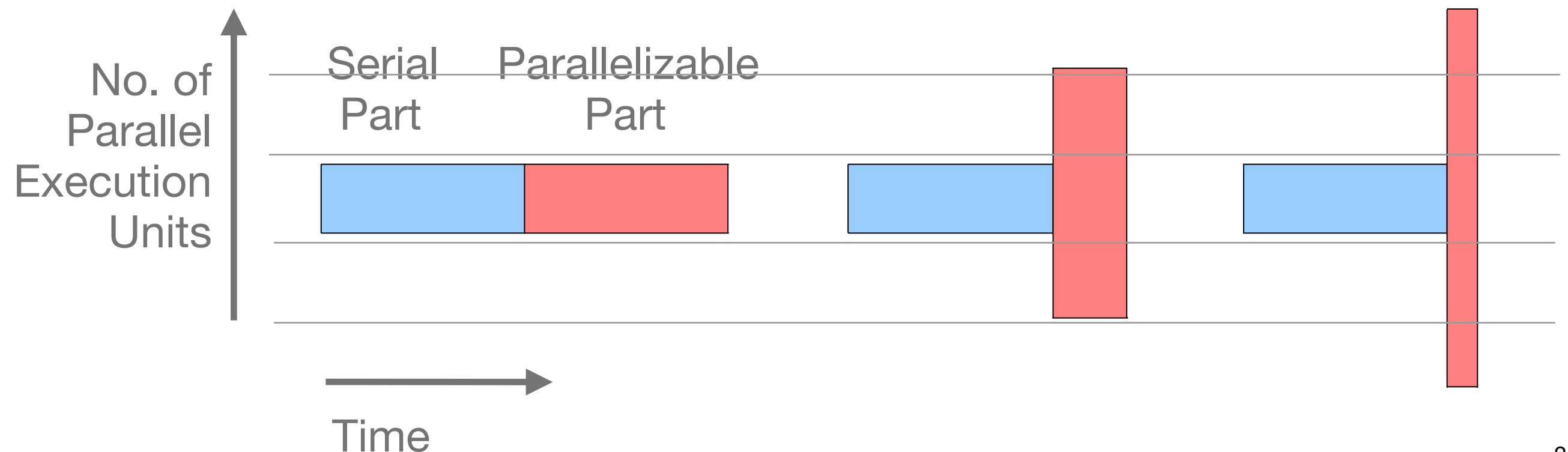
- The increased emphasis on parallelization and parallel programming, plus the heterogeneous hardware environment, implies new software components are needed to support parallel programming
- Later in the week you will hear about some of these (and the pros and cons): OpenMP, MPI, CUDA, OpenACC, OpenCL, etc.

Profiling Tools

- You probably want to make sure that the time you dedicate to working on software performance and efficiency will help
- To do this you should be making decisions based on performance profiles for your application(s)
- In this school you will use several example profiling tools:
 - IgProf – simply statistical profiler and memory profiler
 - Valgrind – general memory debugger/profile
 - A variety of Linux system tools
 - PMU-based CPU performance counters
- In your experiment, institute or project you may use others
- The important thing is to use profilers as a guide to where the problems/opportunities are, don't guess!

Amdahl's Law

- The improvement in the total time due to improvements to one part is limited by the amount that part is used
- A similar restatement is: when parallelizing one part of an application, you can never do better than the remaining serial part.



Lecturers

Sverre Jarpe (CERN Openlab)

Peter Elmer (Princeton) – LHC/CMS

Vincenzo Innocente (CERN) – LHC/CMS

Francesco Giacomini (CNAF)

Tim Mattson (Intel)

Monday

Monday, 20 October 2014

08:30 - 18:00	Session 1
08:30	Welcome and opening remarks 30' Speaker: Mauro Morandin (INFN - Padova)
09:00	Introduction and Concepts of Performance and Efficiency 45' Speaker: Dr. Peter Elmer (Princeton University)
09:45	Computer Architecture and Performance Tuning 45' Speaker: Mr. Sverre Jarp (CERN)
10:30	Coffee break 30'
11:00	Computer Architecture and Performance Tuning 45' Speaker: Mr. Sverre Jarp (CERN)
11:45	Hands-on environment checkout 45' Speaker: Dr. Peter Elmer (Princeton University)
12:45	Lunch break 1h30'
14:15	Floating point computing efficiency 45' Speaker: Vincenzo Innocente (CERN) Material: Slides 
15:00	Floating point computing efficiency 45' Speaker: Vincenzo Innocente (CERN)
15:45	Coffee break 30'
16:15	Lecture 45'
17:00	Student lightning presentations 45'
20:30 - 20:30	Dinner

Tuesday

Tuesday, 21 October 2014

08:30 - 18:00	Session 2
08:30	Efficient C++ coding 45' Speaker: Dr. Francesco Giacomini (CNAF)
09:15	Efficient C++ coding 45' Speaker: Dr. Francesco Giacomini (CNAF)
10:00	Coffee break 30'
10:30	Efficient C++ coding 45' Speaker: Dr. Francesco Giacomini (CNAF)
11:15	Efficient C++ coding 45' Speaker: Dr. Francesco Giacomini (CNAF)
12:00	Self-managed time 45'
12:45	Lunch break 1h30'
14:15	Efficient Memory management 45' Speaker: Dr. Peter Elmer (Princeton University)
15:00	Introduction to parallel computing 45' Speaker: Dr. Tim Mattson (Intel)
15:45	Coffee break 30'
16:15	Introduction to parallel computing 45' Speaker: Dr. Tim Mattson (Intel)
17:00	Students lightning presentations 45' Speaker: Dr. Peter Elmer (Princeton University)
20:30 - 20:30	Dinner

Wednesday

Wednesday, 22 October 2014

08:30 - 14:15	Session 3
08:30	Introduction to parallel computing 45' Speaker: Dr. Tim Mattson (Intel)
09:15	A "Hands-on" introduction to OpenMP 45' Speaker: Dr. Tim Mattson (Intel)
10:00	Coffee break 30'
10:30	A "Hands-on" introduction to OpenMP 45' Speaker: Dr. Tim Mattson (Intel)
11:15	A "Hands-on" introduction to OpenMP 45'
12:00	Self-managed time 45'
12:45	Lunch break 1h30'
14:15 - 16:15	Session 4
14:15	Vectorization 45' Speaker: Vincenzo Innocente (CERN)
15:00	Vectorization 45' Speaker: Vincenzo Innocente (CERN)
15:45	Coffee break 30'
16:30 - 19:30	Social Tour
19:30 - 22:00	Social dinner <i>Casa Artusi Restaurant</i>

Thursday

Thursday, 23 October 2014

08:30 - 12:45	Session 5
08:30	GPUs and the Heterogeneous programming problem 45' Speaker: Dr. Tim Mattson (Intel)
09:15	GPUs and the Heterogeneous programming problem 45' Speaker: Dr. Tim Mattson (Intel)
10:00	Coffee break 30'
10:30	OpenCL and CUDA: A hands-on Introduction 45' Speaker: Dr. Tim Mattson (Intel)
11:15	OpenCL and CUDA: A hands-on Introduction 45' Speaker: Dr. Tim Mattson (Intel)
12:00	Self-managed time 45' Speaker: Dr. Tim Mattson (Intel)
12:45 - 14:15	Lunch
14:15 - 19:00	Session 6
14:15	OpenCL and CUDA: A hands-on Introduction 45' Speaker: Dr. Tim Mattson (Intel)
15:00	Lecture 45'
15:45	Coffee break 30'
16:15	Lecture 45'
18:00	Evening lecture: "Big Data" 1h0' Speaker: Dr. Tim Mattson (Intel)
20:00 - 20:00	Dinner

Friday

Friday, 24 October 2014

08:30 - 12:45	Session 7
08:30	Efficient Memory management 45' Speaker: Dr. Peter Elmer (Princeton University)
09:15	Efficient Memory management 45' Speaker: Dr. Peter Elmer (Princeton University)
10:00	Coffee break 30'
10:30	Efficient Memory management 45' Speaker: Dr. Tim Mattson (Intel)
11:15	Efficient Memory management 45' Speaker: Dr. Peter Elmer (Princeton University)
12:00	Self-managed time 45'
12:45 - 14:15	Lunch
14:15 - 18:00	Session 8
14:15	A hands-on introduction to Custer Computing 45' Speaker: Dr. Tim Mattson (Intel)
15:00	A hands-on introduction to Cluster Computing 45' Speaker: Dr. Tim Mattson (Intel)
15:45	Coffee break 30'
16:15	A hands-on introduction to Cluster Computing 45' Speaker: Dr. Tim Mattson (Intel)
20:30 - 20:30	Dinner

By the end of the week...

- ... you should have a good working knowledge of performance issues related to:
 - The evolution of CPU architectures
 - The memory subsystem
 - C++ programming
 - Vectorization and floating point
 - (And especially) Parallel Computing
- And you will have seen various related tools and done exercises for all of these topics.
- It is a very large number of topics for a few days, but you should be well positioned after this week to understand and improve the performance of your own applications.

Saturday

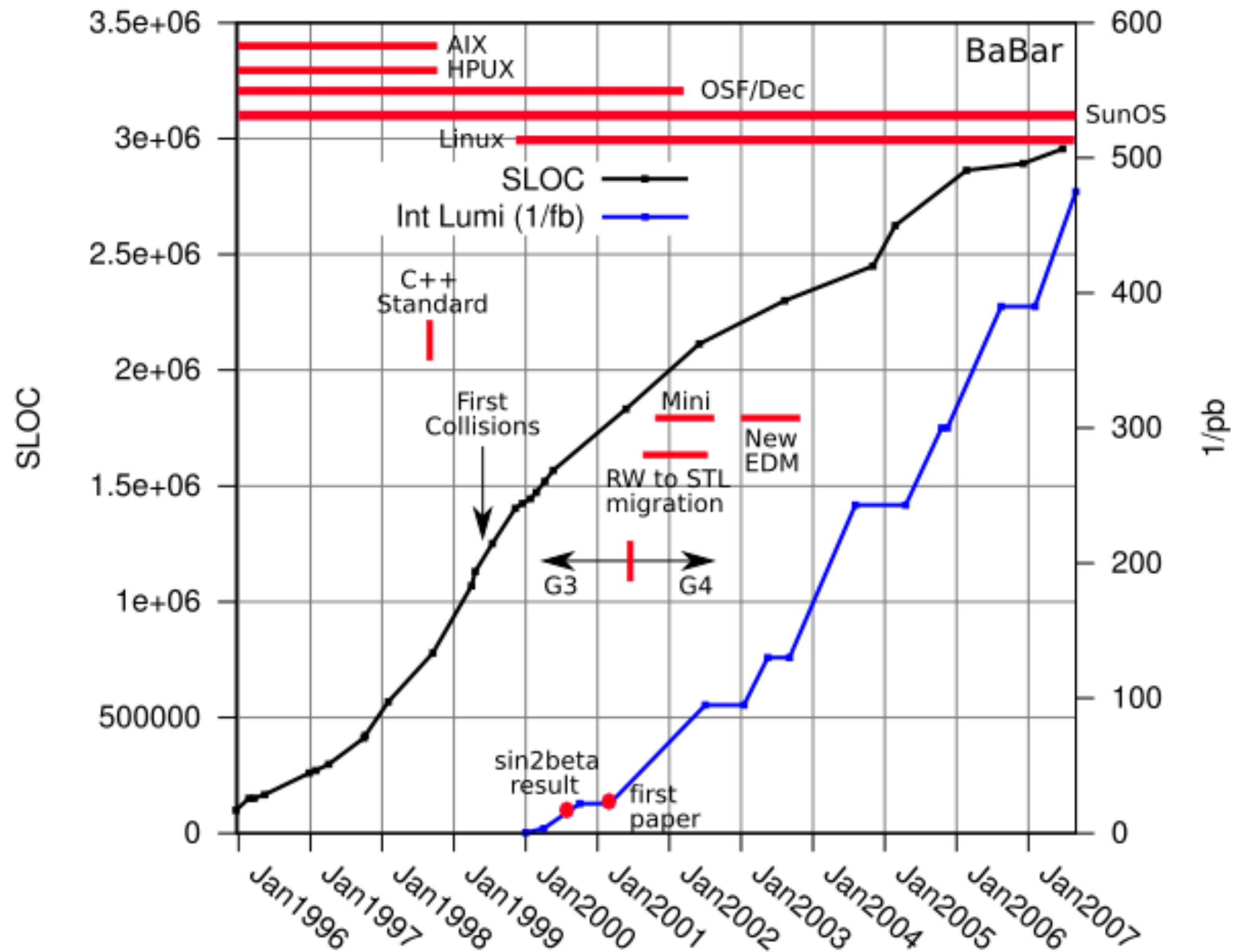
Saturday, 25 October 2014

08:30 - 14:50	Session 9
08:30	Students feedback 30'
09:00	Final examination 2h0'
11:00	Coffee break 30'
11:30	Delivery of certificates of attendance 30'
12:00	Lunch 1h15'
14:00	Shuttle departure (to Forli' railway station) 20'

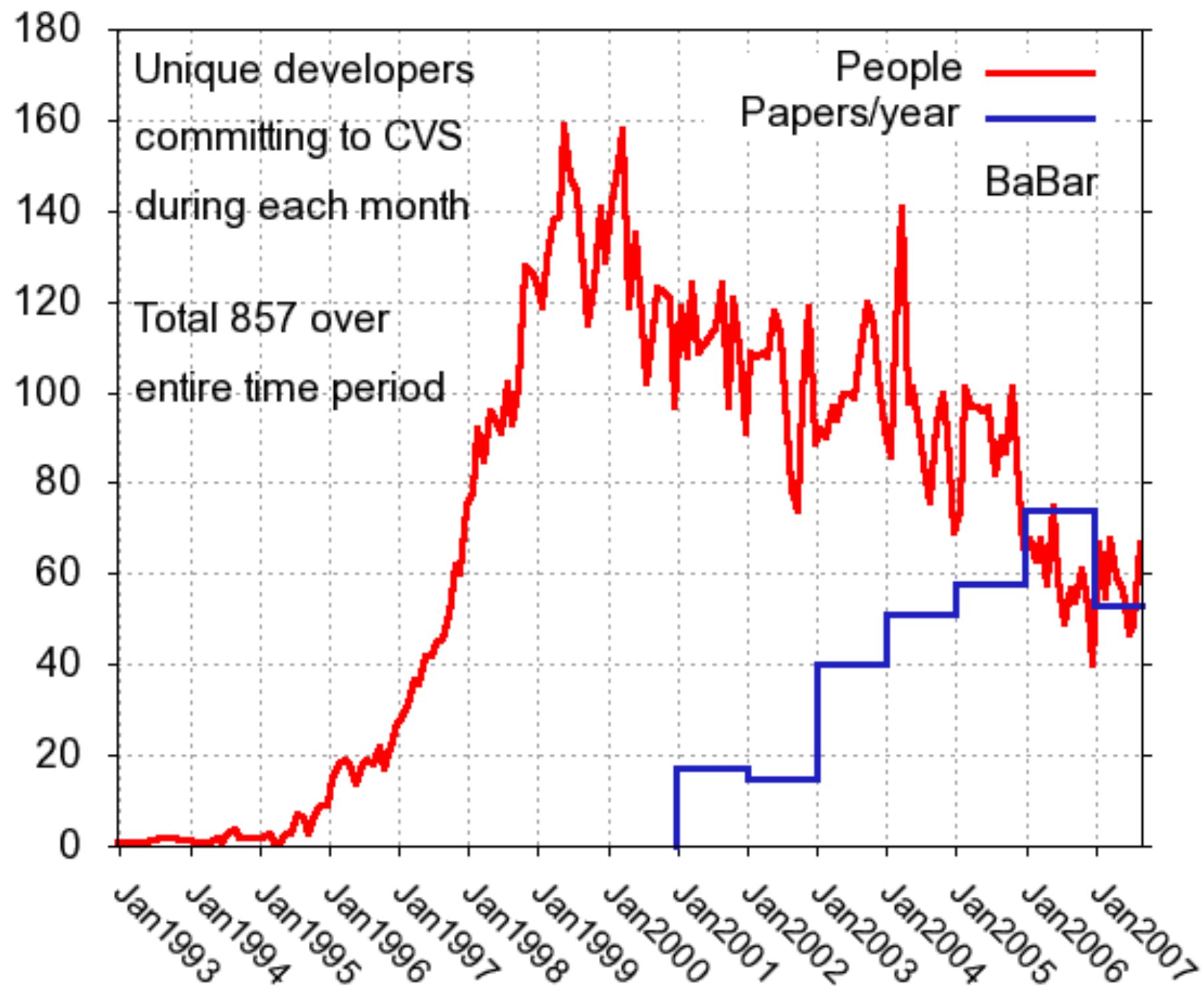
Code Lifetimes

- Large scientific projects by definition will extend over many years and sometimes decades
- Technologies change over time and in any regime where underlying laws are exponential (i.e. Moore's Law), the one thing you can guarantee is that new challenges will arise...

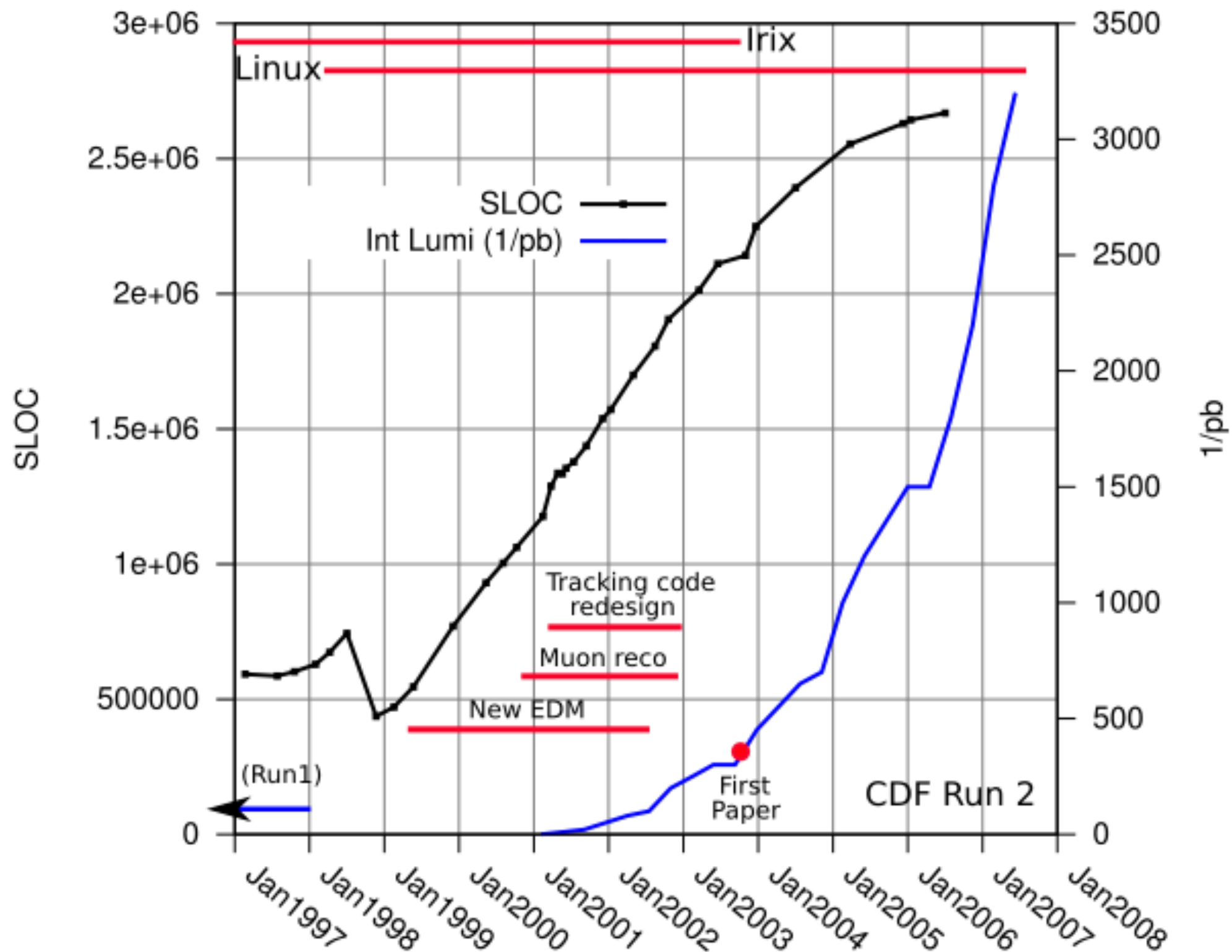
Code Evolution - BaBar at PEP-II (SLAC)



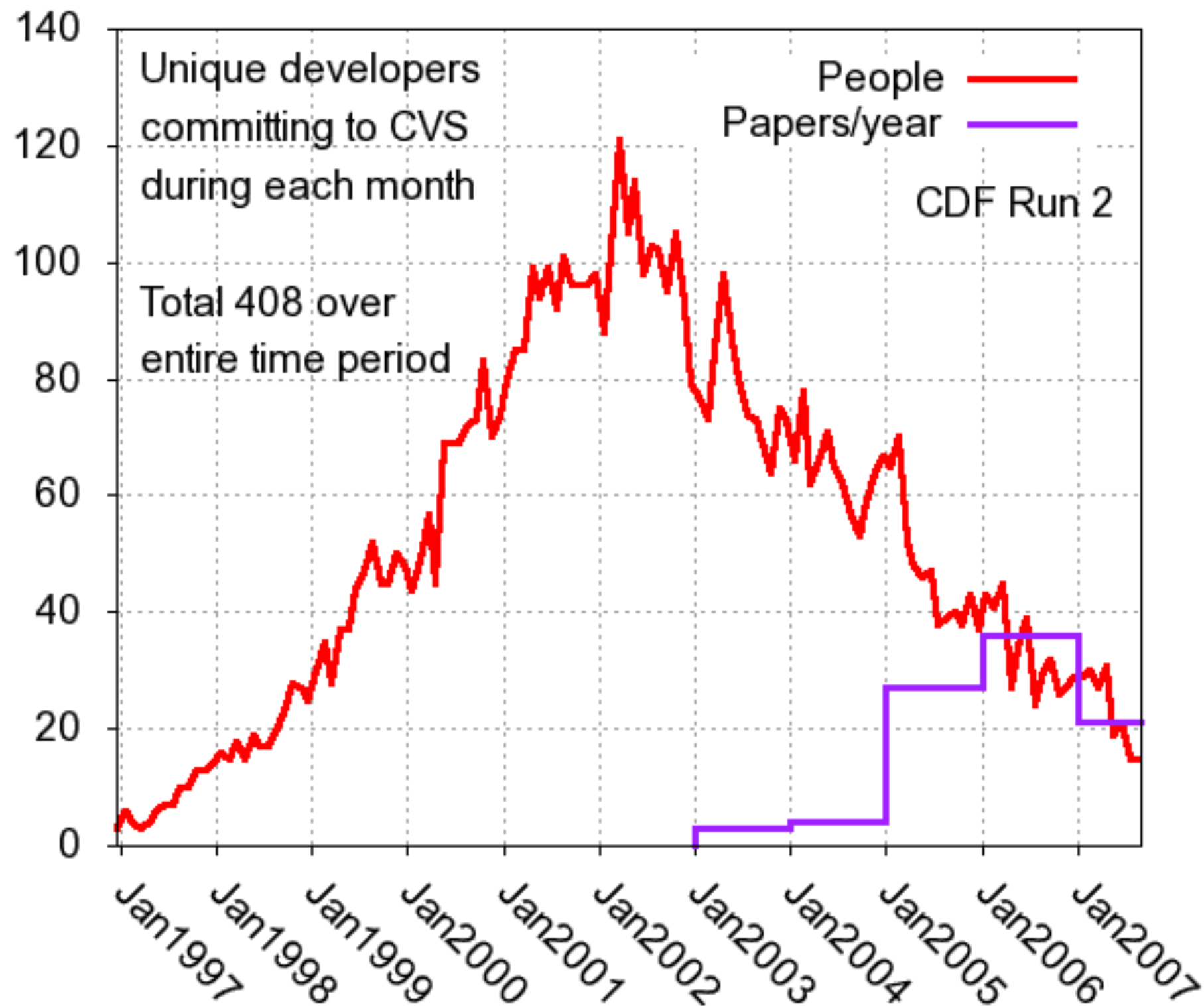
Code Evolution - BaBar at PEP-II (SLAC)



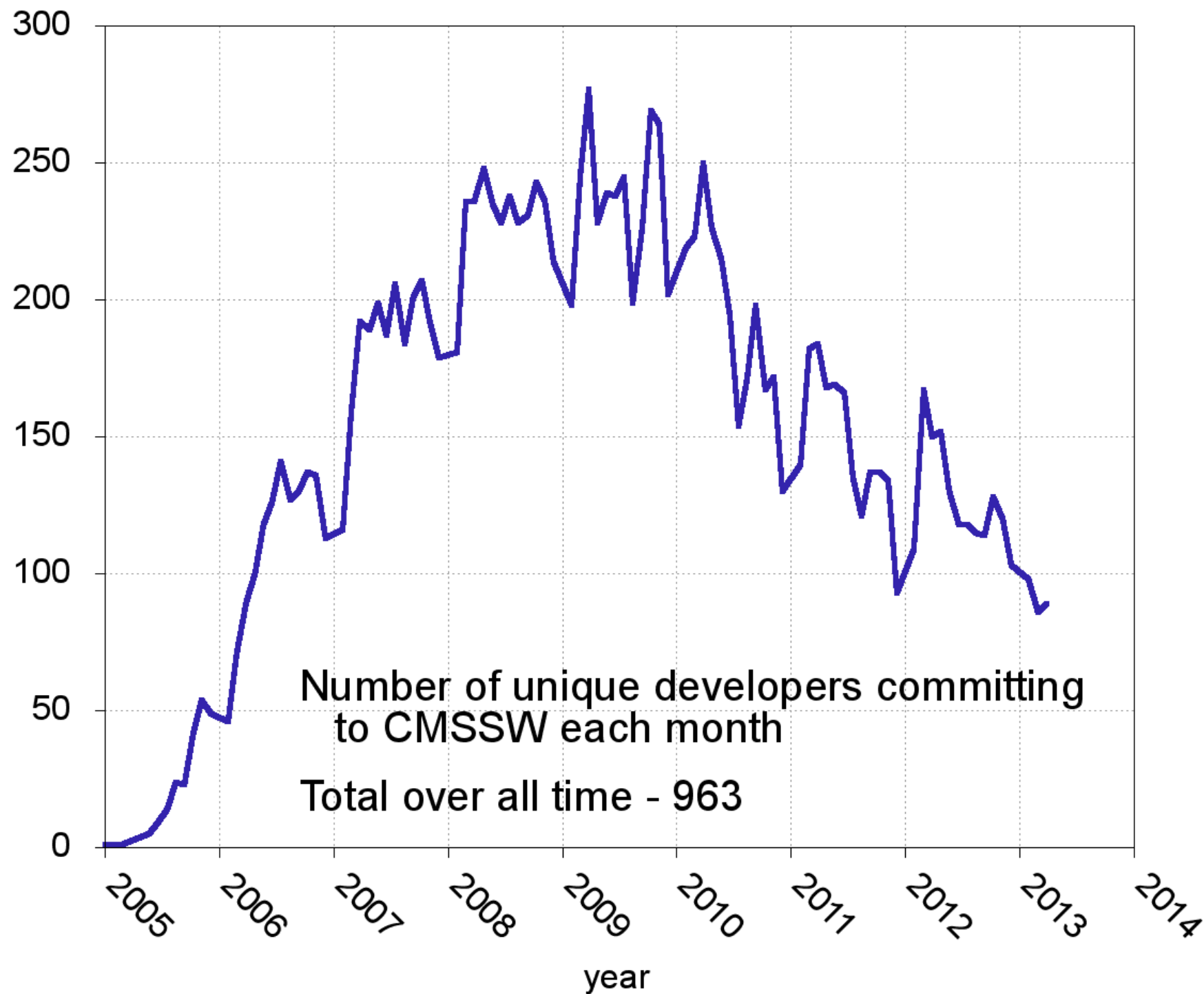
Code Evolution - CDF Run II at the Tevatron (FNAL)



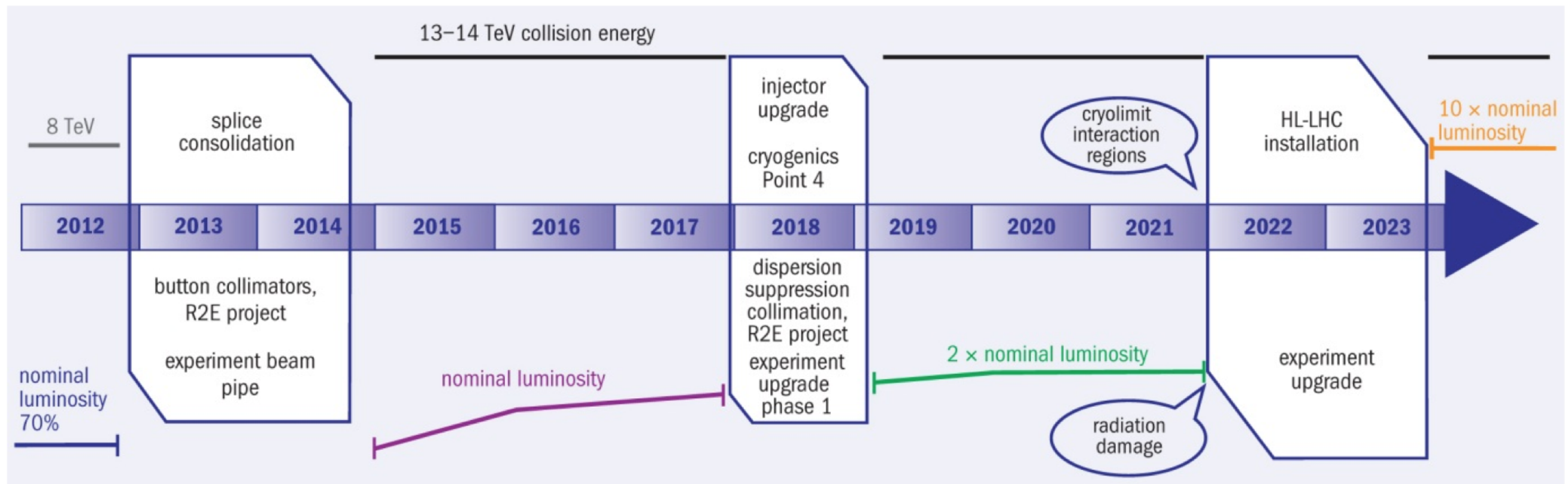
Code Evolution - CDF Run II at the Tevatron (FNAL)



Code Evolution - CMS at the LHC (CERN)



CERN LHC Plans



The long time scale for the LHC is one extreme example, however most projects in the coming years are likely to face challenges (and opportunities) from the technology evolution and the need for maximizing software performance.

Conclusions

Have a productive week!