CERN openlab: optimization, parallelization, evaluation

Alfio Lazzaro
Collaborators: Sverre Jarp, A.L., Julien Leduc, Andrzej Nowak

Ireland's High-Performance Computing Centre
Dublin
April 5th, 2011

About the speaker

- PhD at University of Milan in 2007
- Working on the BaBar experiment (at SLAC, Menlo Park, CA) with the prof. F. Palombo
  - BaBar was the first experiment in the High Energy Physics (HEP) community to use entirely C++
    - Since 1997, before the C++ standard!
- After the PhD also involved in the Atlas experiment at CERN
- My experience is mainly in physics data analysis
- Since 2010: CERN fellow at the IT/Openlab (www.cern.ch/openlab-about)
  - The only large-scale structure at CERN for developing industrial R&D partnerships (main partner HP, Intel, Oracle, Siemens)

CERN openlab

CERN openlab

www.cern.ch/openlab

PARTNERS

- Phase 3 (2009-2011) on-going
  - I'm involved in the activity with Intel
- Divided in competence centers
  - HP: wireless networking
  - Intel: advanced hardware and software evaluations and integrations
  - Oracle: database and storage
  - Siemens: automating control systems
- Phase 4 (2012-2015) in preparation!

Overview of the presentation

- CERN and the Large Hadron Collider (LHC)
- Computing at CERN
- Openlab and Intel collaboration
- Software evaluations and developments (mostly related to activity with students)
  - Use of different parallel techniques
  - Evaluation of accelerators, i.e. GPUs
- Future activities

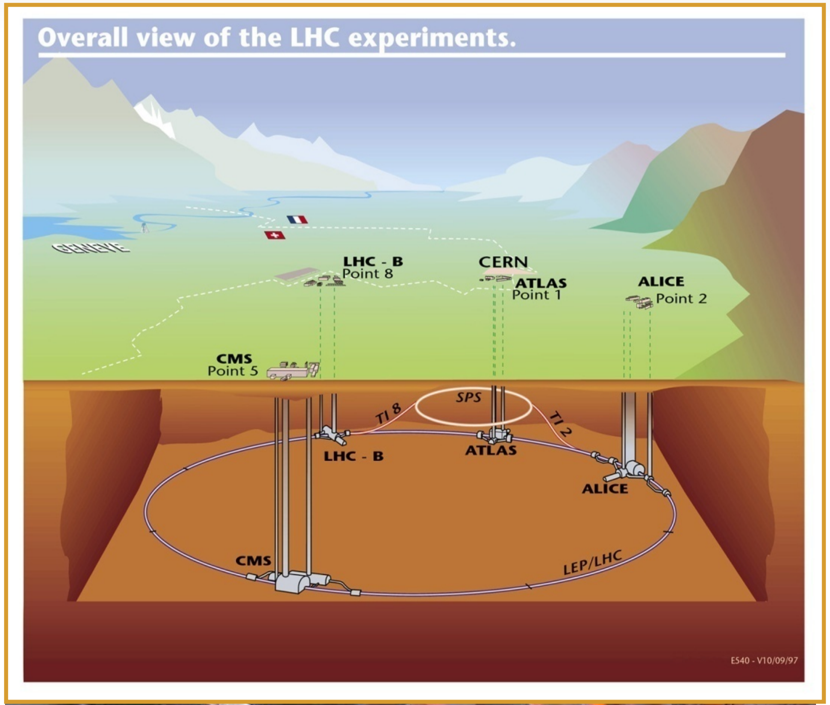
CERN and LHC

About CERN

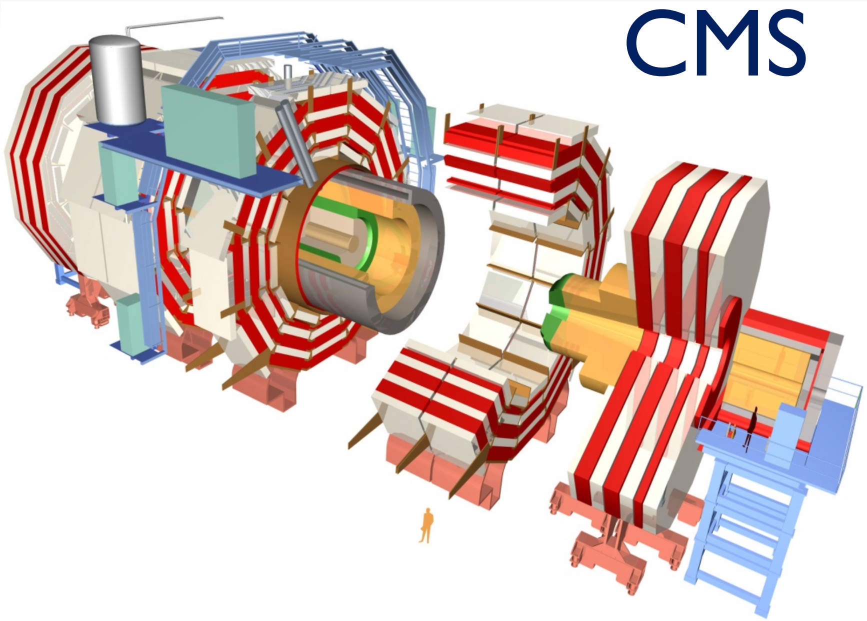
- CERN is the European Organization for Nuclear Research in Geneva
  - 20 Member States
  - 1 Candidate for Accession to Membership of CERN: Romania
  - 8 Observers to Council: India, Israel, Japan, the Russian Federation, the United States of America, Turkey, the European Commission
  - ~2300 staff
  - ~790 students and fellows
  - > 10,000 users (about 5000 on-site)
  - Budget (2010) 1,100 MCHF

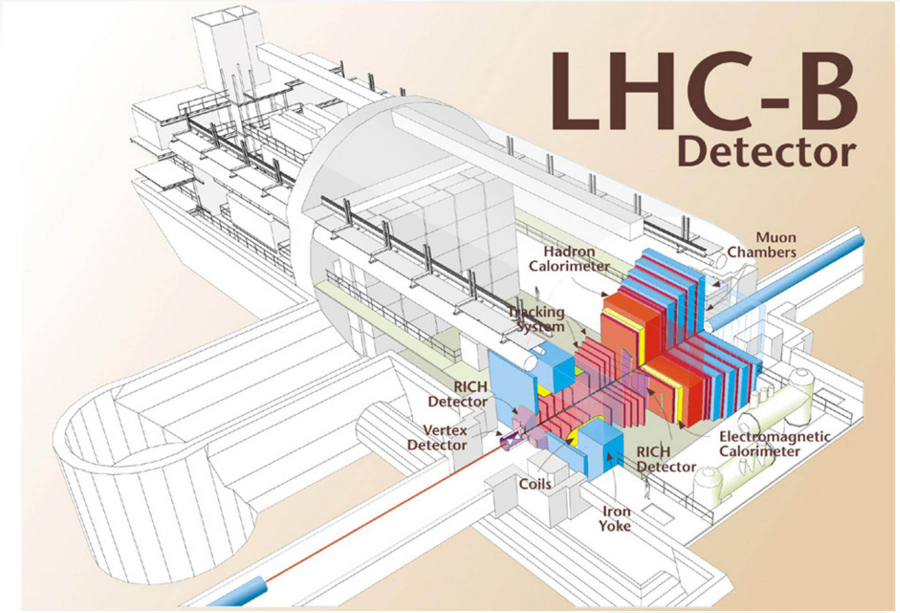
The LHC

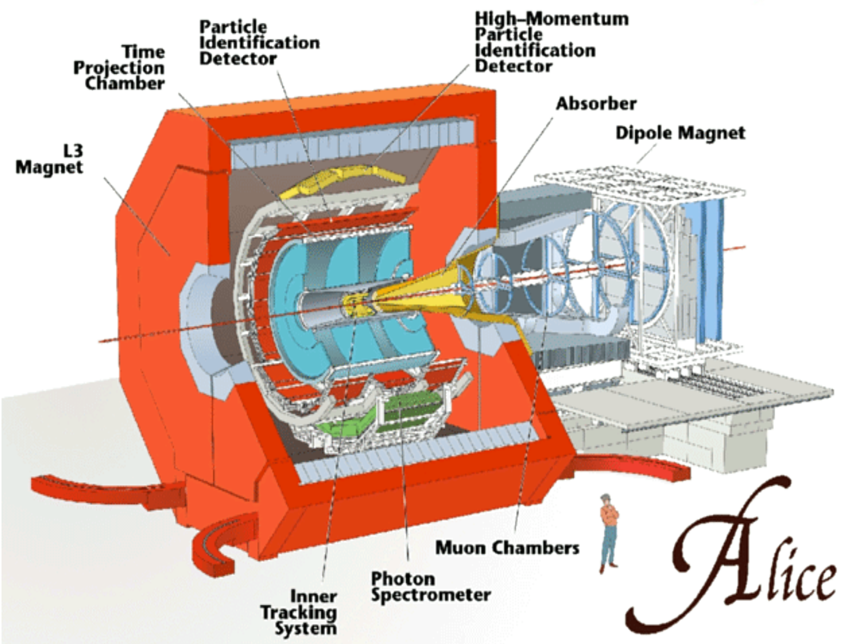
- The biggest machine ever build by man
  - 27 km, 100 meters below ground
- Accelerating protons at 3.5 TeV
  - Already billions of events produced
  - Plan approved for running until 2020
    - A stop foreseen in 2013 for upgrade to 7 TeV

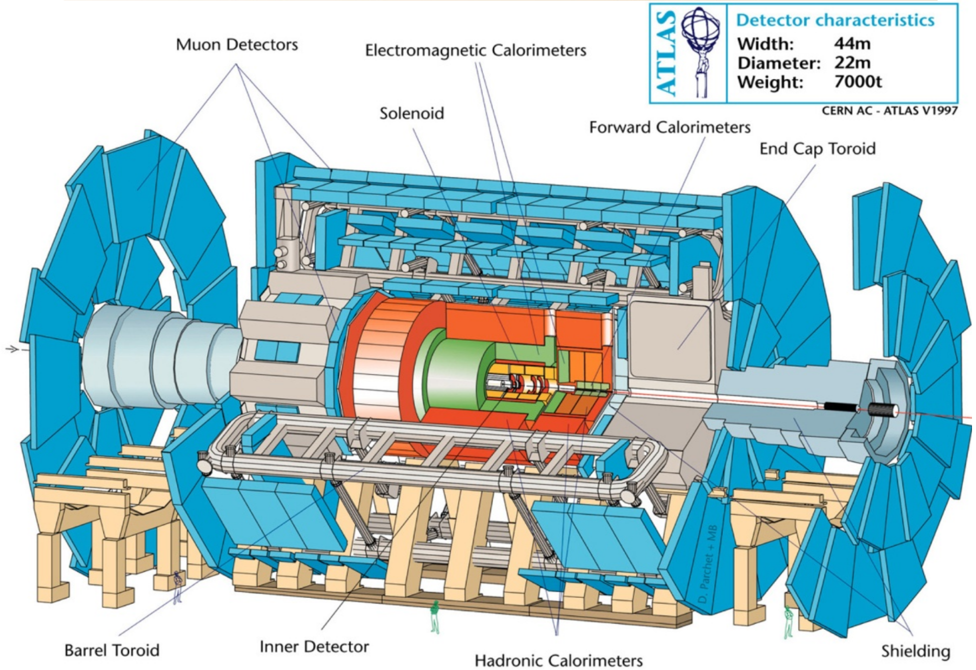


The LHC Experiments

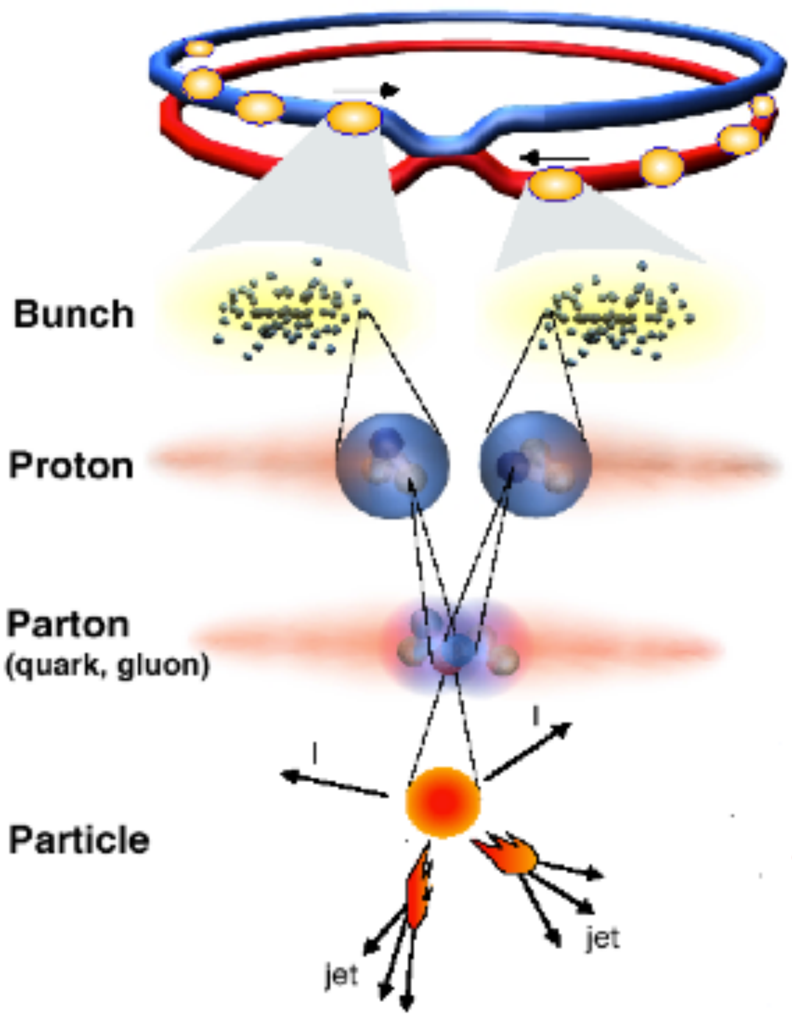








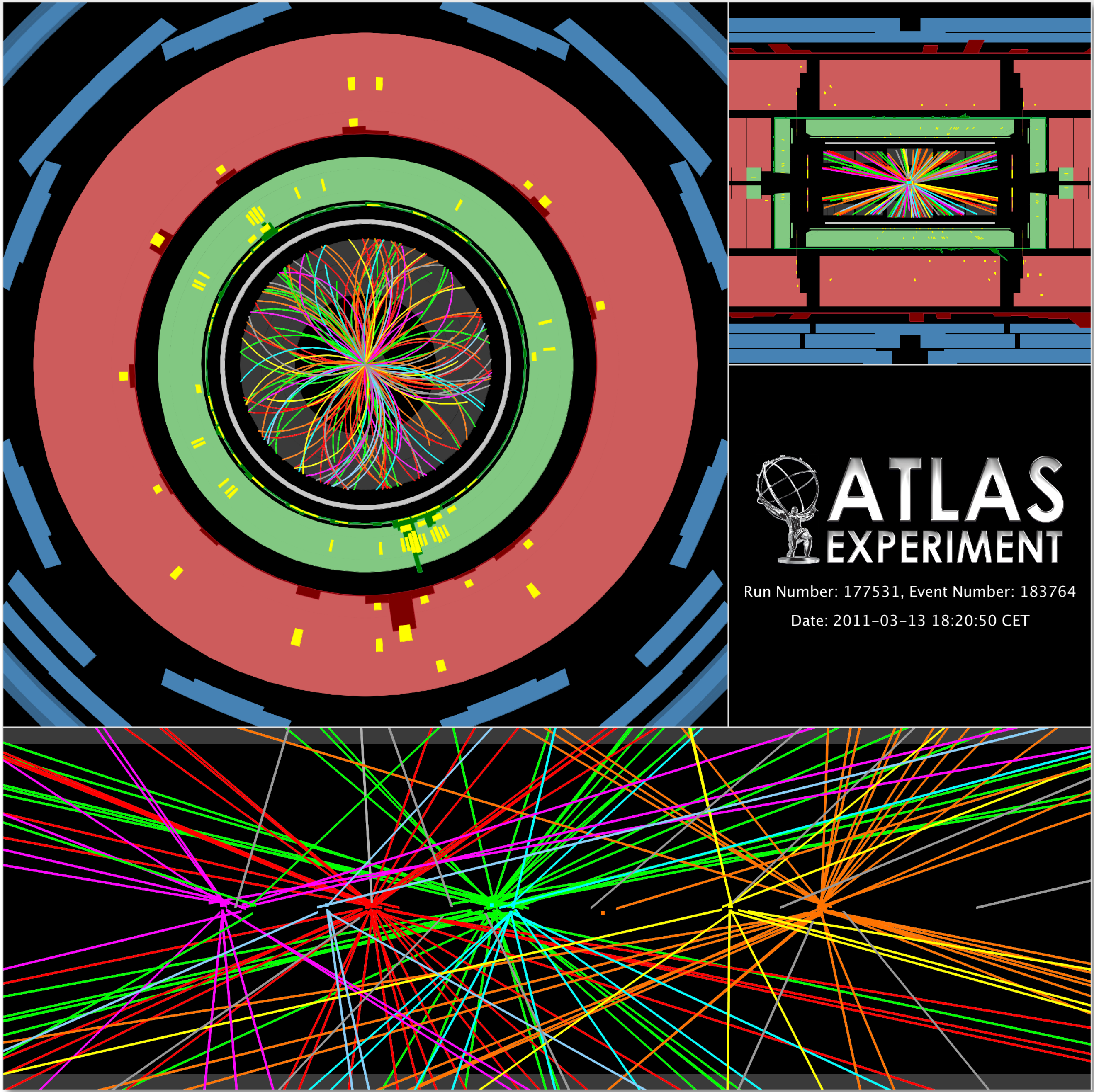
Data acquisition



- Collisions at LHC
  - Proton-Proton or Pb-Pb
  - 40 MHz crossing rate
  - Collisions >10^7 Hz (up to ~50 collisions per bunch crossing)
- Total initial rate: ~1 PB/s
- Several levels of selection of the events (online)
  - Hardware (Level 1), software (Level 2, 3)
  - Final rate for storing: 200 Hz (300 MB/s, ~3 PB/year)

Events are independent: trivial parallelism over the events!

Atlas event


ATLAS
EXPERIMENT
Run Number: 177531, Event Number: 183764
Date: 2011-03-13 18:20:50 CET

Computing at CERN

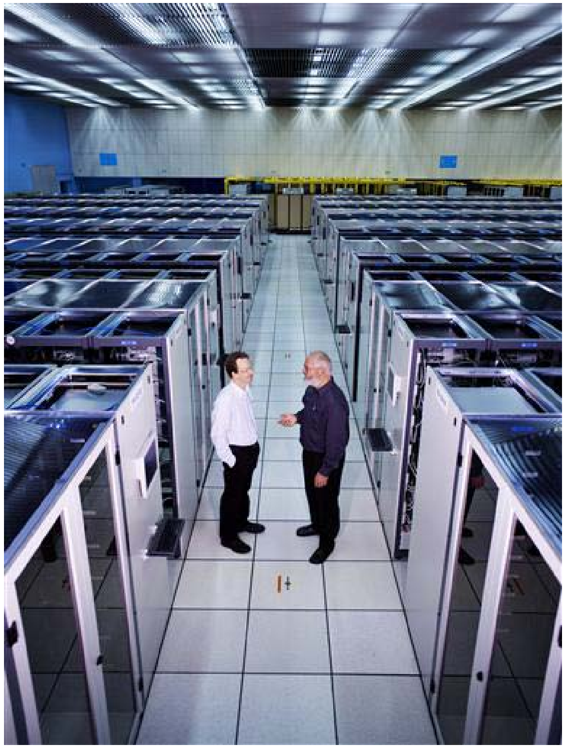
Computing Tasks

- Online (from the detector)
  - Fast event selection (trigger)
  - Initial reconstruction at CERN
  - Storage of the data (on tapes)
- Offline
  - Monte Carlo simulation: generation of the events and simulation of the detector response
    - CPU intensive, particles passing through matter – major phase
    - Generation, Digitization, Reconstruction (more I/O intensive and shorter)
  - Events (from read detector or simulation) are then fully reconstructed, skimmed, distributed
    - Automatic checking of the data
  - Final step is the data analysis: CPU-intesive

Frameworks

- Code is centralized in framework
  - Bulk of the data is read-only: versions rather than updates
  - Very large aggregate requirements: computation, data, input/output
  - More than 1M lines of code (C++), thousands libraries
  - More that 10 years of development
  - About 1000 developers, but only few software experts
  - Not clear hotspot in the computation: thousands routines with small contribution
- Centralization guarantees optimization of resources, in particular storage
  - First reconstruction in the Tier0 at CERN, then data are distributed using GRID
- Final data analysis can be run standalone by each user:
  - Chaotic workload: Unpredictable, Unlimited demand
- Key foundation: Linux together with GNU C++ compiler

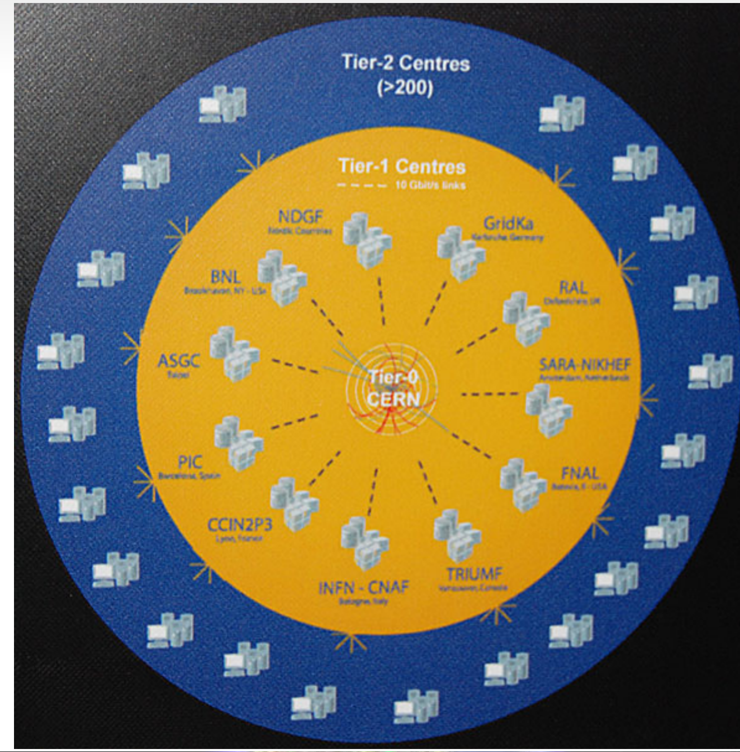
Tier0 @ CERN

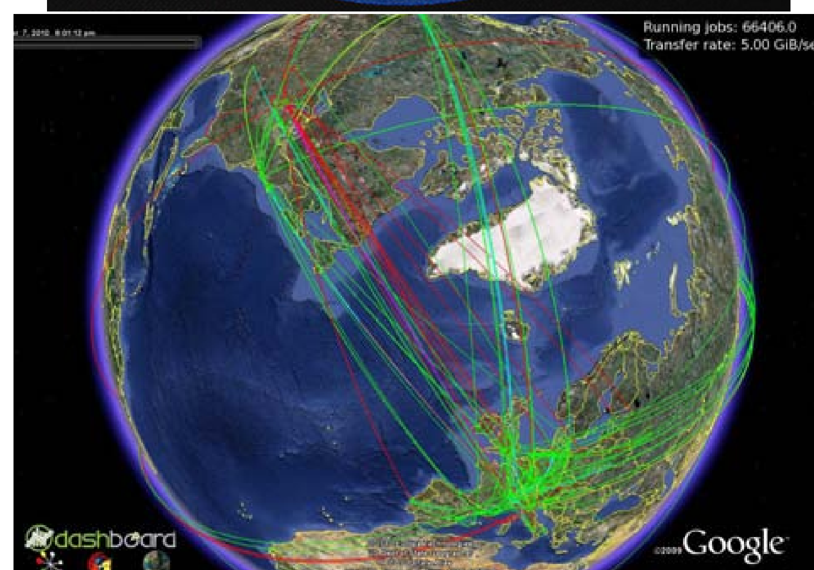


- Three parts (CPU, disk, tape) interconnected by Ethernet and the joined by the AFS file system
- 2.9MW machine room
  - ~7’000 commodity servers with ~40’000 cores – nearly all Linux (RHEL based, DP)
  - 14 PB of disk, 34 PB of tape on 45’000 tapes
- Continues to be upgraded
  - Last purchases: Intel Westmere-EP, AMD Magny-Court (4-socket)
    - low frequency, throughput oriented

LHC Computing GRID

- Replicates data
  - Tier0 to Tier1 per each country
  - Tier1 to Tier2 in some institutions
  - Tier3 for local analysis in each institution
- Fully operational
  - 260’000 cores (all IA and all Linux)
  - Able to handle the full physics load
- Users can easily submit their jobs
  - About 20 million jobs per month
- Dozens of PB of storage
  - Preserve data
- Continues to be upgraded
  - Desire to move to interoperability with Clouds, use of virtualisation





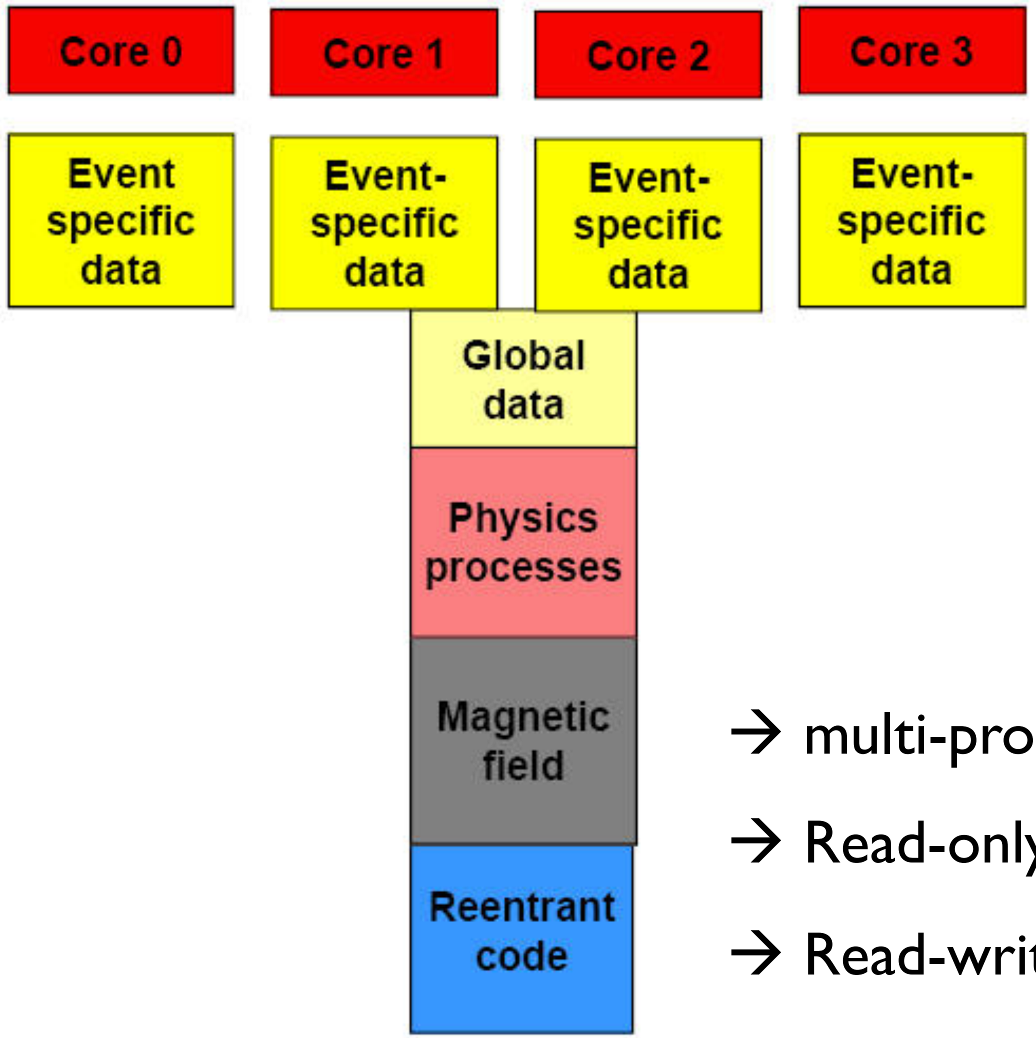
Open issues (short term)

- Framework (offline) are “monolithic”
  - No parallelization, use the trivial concept of parallelism
    - parallelism over the events (inside an event is still not an issue), limited by I/O
    - Each core runs an instance of the whole application, i.e. not shared memory, running part of the total events
    - Merging of the results between jobs at the end of the running
  - Memory footprint is the main limitation at the moment
    - GRID requirement is 2GB per core, expected to increase in the near future (due to physics demand)
    - 96GB on AMD Magny-Court accounts for about half of the power consumption!
  - Projects in the collaborations to share most of the
    - COW, fork, threading… everything to reduce memory footprint
    - No easy task, especially for the GRID prospect

Event parallelism

Opportunity: Reconstruction Memory-Footprint shows large condition data

How to share common data between different process?



CMS experiment:
1GB total Memory Footprint
Event Size 1 MB
Sharable data 250MB
Shared code 130MB
Private Data 400MB !!

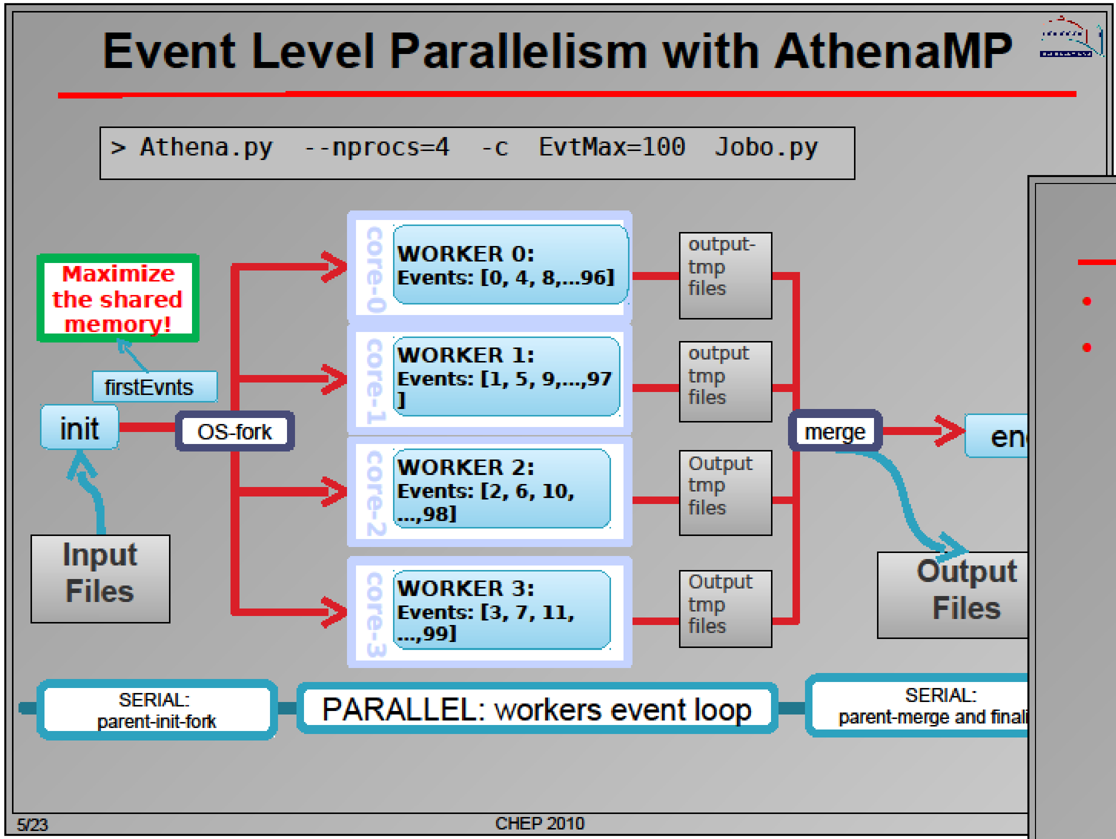
→ multi-process vs multi-threaded

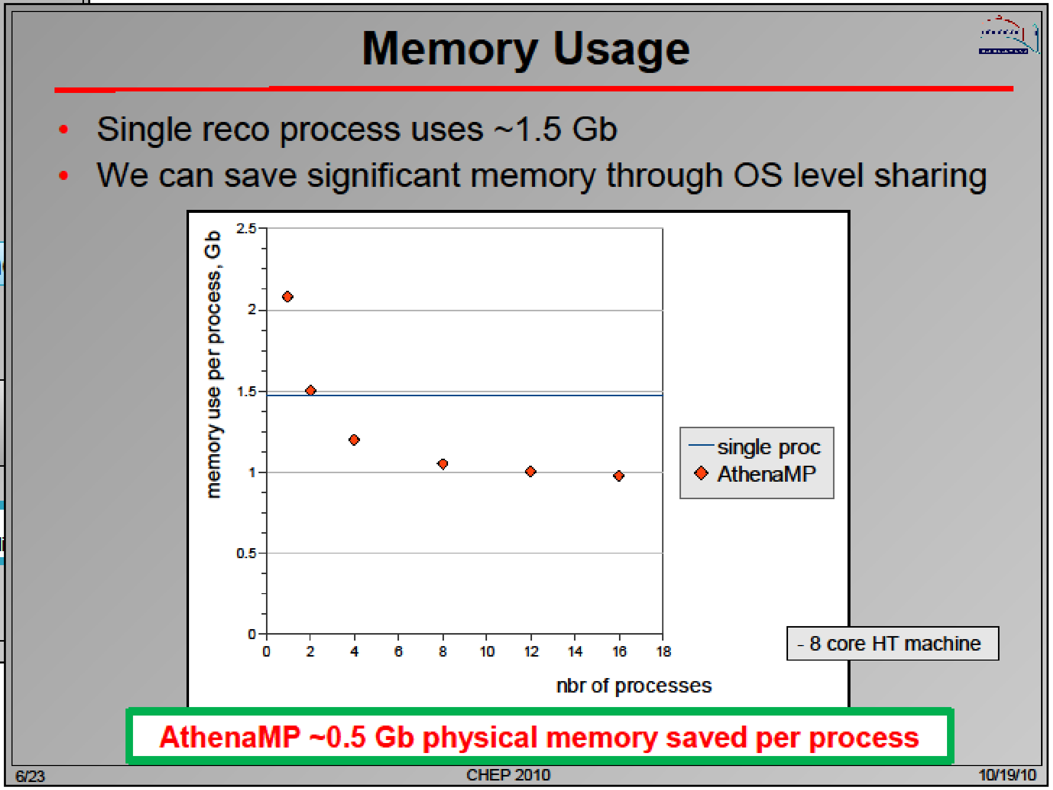
→ Read-only: Copy-on-write, Shared Libraries

→ Read-write: Shared Memory, sockets, files

CHEP10 Parallel by C. Leggett (ATLAS)

"Parallelizing Atlas reconstruction and simulation on multi-core platforms"





CHEP10 Parallel by C. Jones (CMS)

"Multi-core aware Applications in CMS"

Why Bother?

HEP processing is naturally parallelizable

- We have billions of events
- Each event can be processed independently

Memory is becoming a limitation

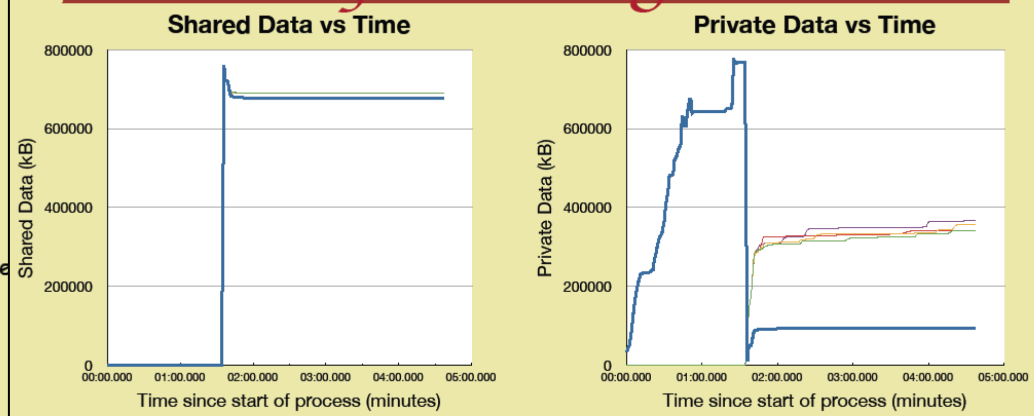
- Historically GB/US$ increases at the same rate as number of transistors in a CPU
  - http://www.jcmit.com/memoryprice.htm
- Funding levels are not guaranteed to stay this high
  - We can afford 2GB/core now but may not in the future
- Opportunistic use of grid sites improves if we lower our memory requirements
  - Not all grid sites have 2GB/core
- Technical limitations on connecting many cores to shared system memory
  - http://www.intel.com/technology/itj/2007/v11i3/3-bandwidth/7-conclusion.htm

Multi-core aware applications can improve memory sharing

- Threading
  - All threads share the same address space but have to worry about concurrent usage
- Forking
  - Each child process gets its own address space
  - Untouched memory setup by the parent is shared between the child processes

Multi-core Aware Applications in CMS 3 CHE

Memory Sharing



Measurements done using reconstruction with 64bit software on 4 CPU, 8 core/CPU 2GHz AMD Opteron(tm) Processor 6128

- Shared memory per child: ~700MB
- Private memory per child: ~375MB
- Total memory used by 32 children: 13GB
- Total memory used by 32 separate jobs: 34 GB

Saved 62% of memory

Multi-core Aware Applications in CMS 11 CHEP 2010

Open issues (long term)

- Plan to increase the rate of collisions of a factor 100x (Super-LHC, 2016?)
  - It will require to move part of the offline reconstruction to be run in the online, so that the trigger can be more efficient
  - Several projects to use GPUs (or FPGAs) for fast computation
    - Also an attempt to use CELL processors
  - Other future experiments have the same problem and there is a common agreement in using accelerators for fast and real-time computation
- Data analysis will play a crucial role when more data will be available (2012?)
  - Mandatory to speed-up the execution using HPC concepts: vectorization, shared/distributed memory, accelerators
  - Very chaotic situation: several projects ongoing from different groups
    - Here the target is to use commodity systems (e.g. laptops or desktops)

CERN openlab and Intel

CERN openlab and Intel

- The Intel collaboration is driven by the Platform Competence Center
- Example of activities
  - CPUs and platforms evaluation: Xeon EP and EX, Atom, Many Integrated Core (MIC), Itanium, other hardware (like networking)
  - Intel Software tools: Compiler studies, Performance monitoring and optimization, Multi-threading and many core studies
  - Teaching and dissemination: workshops, summer students, technical students (master thesis)
  - Thermal optimization
- Other activities, not directly related to Intel:
  - Development and testing of software, in collaboration with the physics community
    - Bouquet" of applications to check different HEP workloads on different hardware: online, simulation, data analysis
  - Evaluation of GPUs (AMD, NVIDIA) for data analysis

In the rest of the talk I will focus on data analysis related activities

Maximum Likelihood Fits

- We have a sample composed by *N* events, belonging to *s* different specie (signals, backgrounds), and we want to extract the number of events for each species and other parameters
- We use the Maximum Likelihood fit technique to estimate the values of the free parameters, minimizing the Negative Log-Likelihood (*NLL*) function

$$NLL = \sum_{j=1}^{s} n_j - \sum_{i=1}^{N} \left(\ln \sum_{j=1}^{s} n_j \mathcal{P}_j(x_i;\theta_j) \right)$$

j species (signals, backgrounds)
n_j number of events
$\mathcal{P}_j$ probability density function (PDF)
θ_j Free parameters in the PDFs

MINUIT

- Numerical minimization of the *NLL* using MINUIT (F. James, Minuit, Function Minimization and Error Analysis, CERN long write-up D506, 1970)
- MINUIT uses the gradient of the function to find local minimum (MIGRAD), requiring
  - The calculation of the gradient of the function for each free parameter, naively

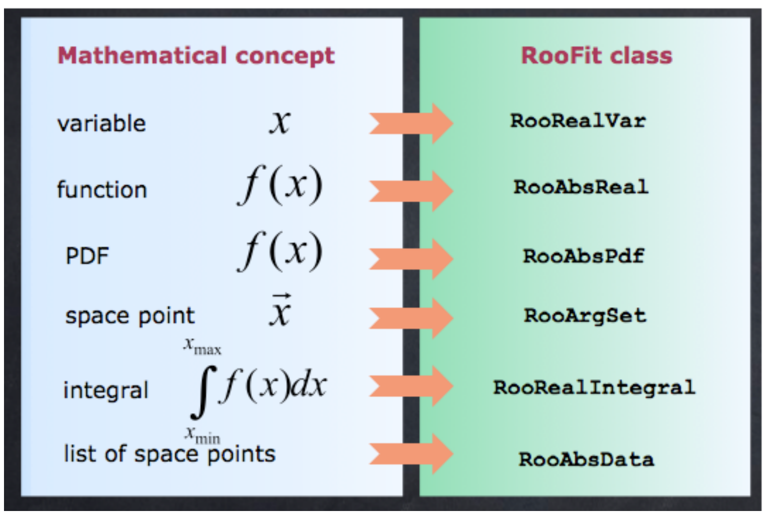
$$\frac{\partial NLL}{\partial \hat{\theta}}\bigg|_{\hat{\theta}_0} \approx \frac{NLL(\hat{\theta}_0 + \hat{\mathrm{d}}) - NLL(\hat{\theta}_0 - \hat{\mathrm{d}})}{2\hat{\mathrm{d}}}$$

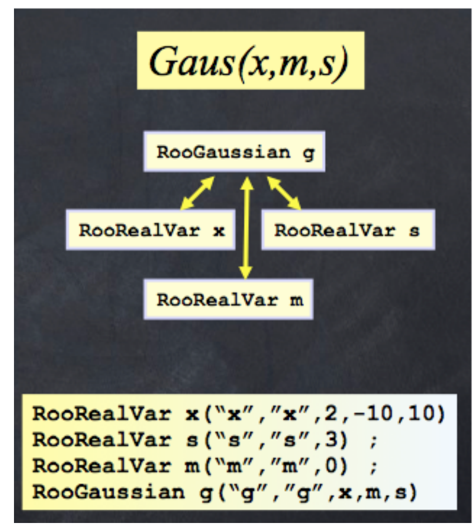
2 function calls per each parameter

  - The calculation of the covariance matrix of the free parameters (which means the second order derivatives)
- The minimization is done in several steps moving in the Newton direction: each step requires the calculation of the gradient
  - ➪ Several calls to the *NLL*

Building models: RooFit

- RooFit is a Maximum Likelihood fitting package (W. Verkerke and D. Kirkby) for the *NLL* calculation
  - Details at http://root.cern.ch/drupal/content/roofit)
  - Allows to build complex models and declare the likelihood function
  - Mathematical concepts are represented as C++ objects



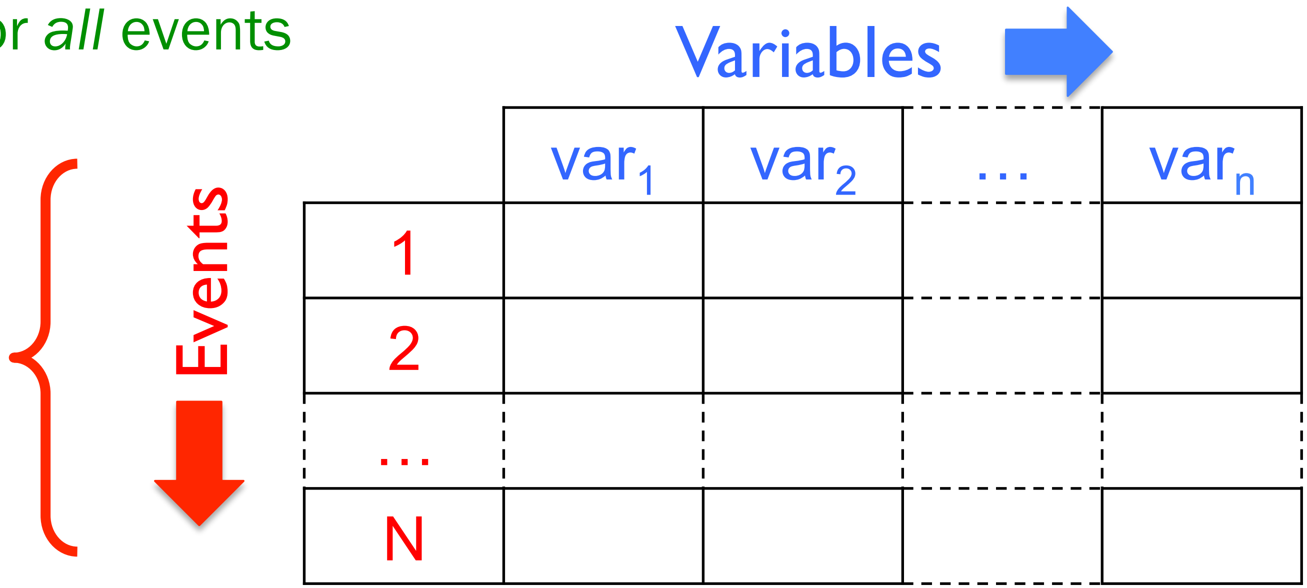


- On top of RooFit developed another package for advanced data analysis techniques, RooStats
  - Limits and intervals on Higgs mass and New Physics effects

Likelihood Function calculation in RooFit

1. Read the values of the variables for each event
2. Make the calculation of PDFs for each event
   - Each PDF has a common interface declared inside the class RooAbsPdf with a virtual method which defines the function
   - Automatic calculation of the normalization integrals for each PDF
   - Calculation of composite PDFs: sums, products, extendend PDFs
3. Loop on all events and make the calculation of the *NLL*
   - A *single* loop for *all* events

Parallel execution over the events, with final reduction of the contribution



New Algorithm

New approach to the *NLL* calculation:

1. Read all events and store in arrays in memory
2. For each PDF make the calculation on all events
   - Corresponding array of results is produced for each PDF
   - Evaluation of the function inside the local PDF, i.e. not need a virtual function (drawback: require more memory to store temporary results: 1 double prevision value per each event and PDF)
   - Apply normalization
3. Combine the arrays of results (composite PDFs)
4. Calculation of the *NLL*

Parallelization splitting calculation of each PDF over the events

- Particularly suitable for thread parallelism on GPU, requiring one thread for each PDF/event
- Easy parallelization of the loop using OpenMP (now we do parallelization for each local loop of each PDF)
- Vectorization (auto-vectorization)

Complex Model Test

$$n_a[f_{1,a}G_{1,a}(x) + (1 - f_{1,a})G_{2,a}(x)]AG_{1,a}(y)AG_{2,a}(z) +$$
$$n_bG_{1,b}(x)BW_{1,b}(y)G_{2,b}(z) +$$
$$n_cAR_{1,c}(x)P_{1,c}(y)P_{2,c}(z) +$$
$$n_dP_{1,d}(x)G_{1,d}(y)AG_{1,d}(z)$$

17 PDFs in total, 3 variables, 4 components, 35 parameters

- G: Gaussian
- AG: Asymmetric Gaussian
- BW: Breit-Wigner
- AR: Argus function
- P: Polynomial

40% of the execution time is spent in exp's calculation

Note: all PDFs have analytical normalization integral, i.e. >98% of the sequential portion can be paralellized

Test on CPU in sequential

- Dual socket Intel Westmere-based system: CPU (L5640) @ 2.27GHz (12 physical cores, 24 hardware threads in total), 10x4096MB DDR3 memory @ 1333MHz
- Intel C++ compiler version 11.1 (20100414)

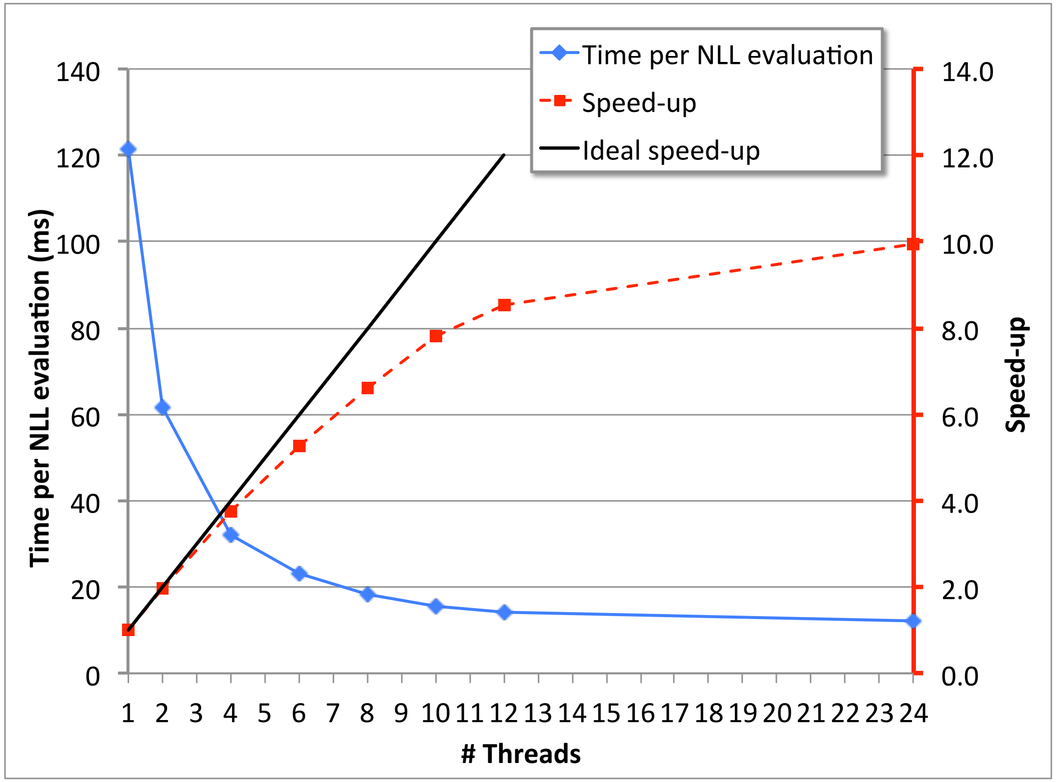
# Events	10,000	25,000	50,000	100,000
RooFit				
# *NLL* evaluations	15810	14540	19041	12834
Time (s)	826.0	1889.0	5192.9	6778.9
Time per *NLL* evaluation (ms)	52.25	129.92	272.72	528.19
OpenMP (w/o vectorization)				
# *NLL* evaluations	15237	17671	15761	11396
Time (s)	315.1	916.0	1642.6	2397.3
Time per *NLL* evaluation (ms)	20.68	51.84	104.22	210.36
w.r.t. RooFit	2.5x	2.5x	2.6x	2.5x
OpenMP (w/ vectorization)				
# *NLL* evaluations	15304	17163	15331	12665
Time (s)	178.8	492.1	924.2	1536.9
Time per *NLL* evaluation (ms)	11.68	28.67	60.28	121.35
w.r.t. RooFit	4.5x	4.5x	4.4x	4.4x

4.5x faster!

Vectorization gives a 1.8x speed-up (SSE)

Test on CPU in parallel

- Same system as before, with 100,000 events

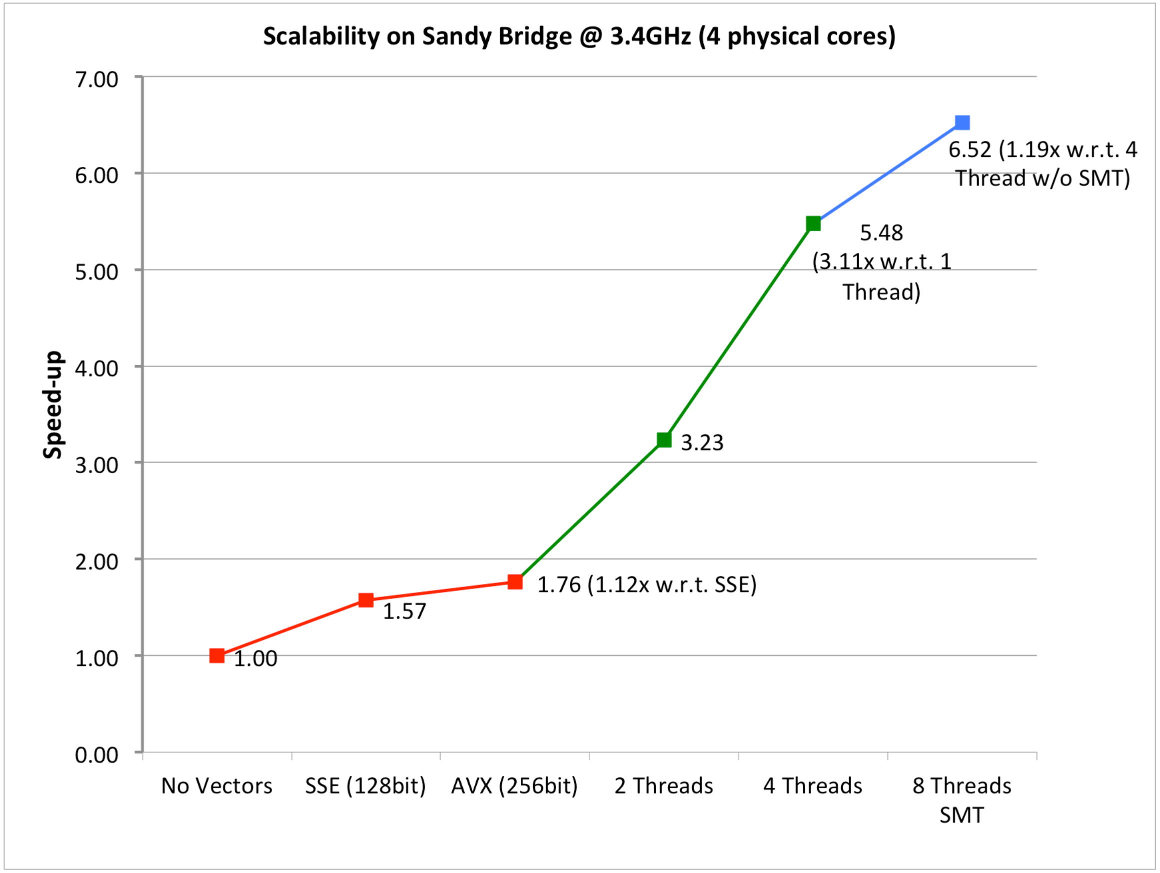


A paper will be published on Journal of Physics: Conf. Series (Proceeding of the CHEP presentation)

- Data is shared, i.e. no significant increase in the memory footprint
  - Possibility to use Hyper-threading (about 20% improvement)
- Limited by the sequential part, OpenMP overhead, and memory access to data

Using Intel Sandy Bridge (AVX)

- Take benefit from AVX new instructions (256bit) on Sandy Bridge (desktop version in this test)



  - AVX gives +12% more than SSE

GPU Test environment

- PCs
  - CPU: Intel Nehalem @ 3.2GHz: 4 cores – 8 hw-threads
  - OS: SLC5 64bit

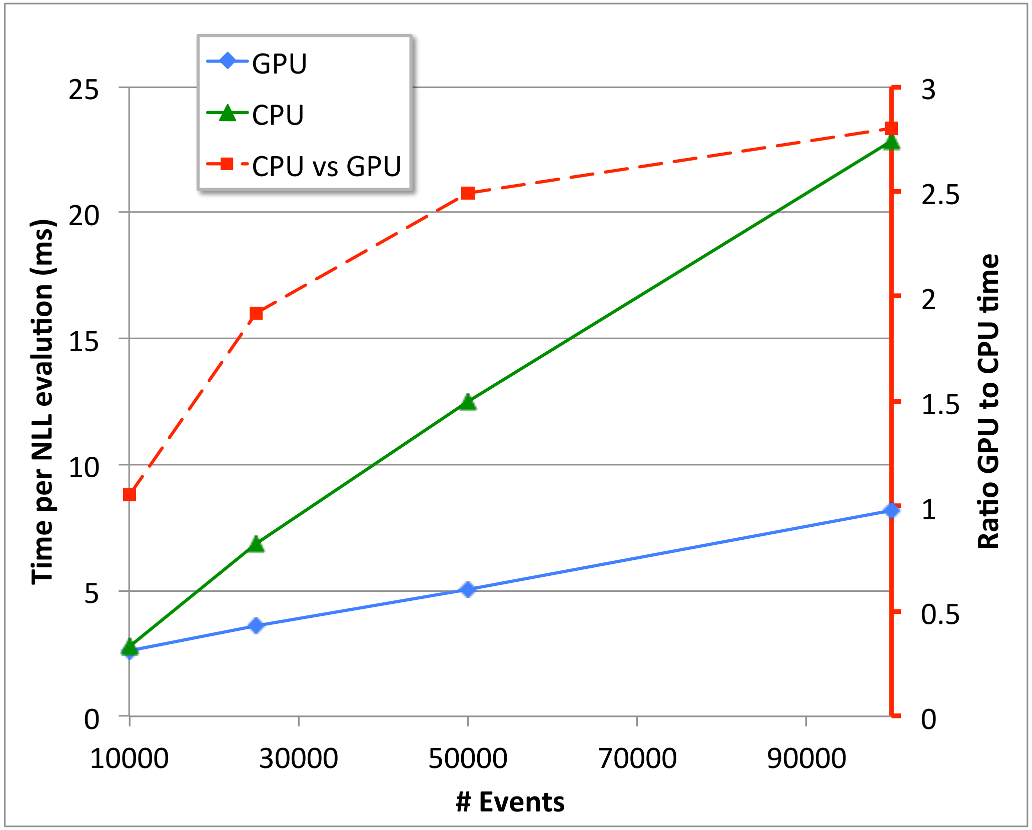
- GPU: ASUS nVidia GTX470 PCI-e 2.0
  - Commodity card (for gamers)
  - Architecture: GF100 (Fermi)
  - Memory: 1280MB DDR5
  - Core/Memory Clock: 607MHz/837MHz
  - Maximum # of Threads per Block: 1024
  - Number of SMs: 14
  - CUDA Toolkit 3.2
  - Power Consumption 200W
  - Price ~$340

GPU Implementation

- Everything in double precision
- Data is copied on the GPU once
- Results for each PDF are resident only on the GPU
    - Arrays of results are allocated on the global memory once and they are deallocated at the end of the fitting procedure
        - Minimize CPU ⇔ GPU communication
    - Only the final results are copied on the CPU for the final sum to compute *NLL*
- Device algorithm performance with a linear polynomial PDF and 1,000,000 events
    - 112 GFLOPS (not including communications), about 82% of the peak performance

GPU performance

- OpenMP runs on the 4 cores for the CPU reference



Work done with the contribution of a summer student (2010), Felice Pantaleo

- A paper accepted to be presented at 12th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing, May 16-20, 2011, Anchorage (USA)

Conclusion

- Optimization of the existing RooFit algorithm gave a 4.5x speed-up when running it in sequential
- Very easy to implement it with OpenMP
  - Good scalability (3.8x with 4 threads/core)
- Implementation of the algorithm in CUDA required not so drastic changes in the existing RooFit code
  - Up to a factor 2.8x with respect to OpenMP with 4 threads
- Note that our target is running fits at the user-level on the GPU of small systems (laptops), i.e. with small number of CPU cores and commodity GPU cards
  - Main limitation is the double precision
  - No limitation due to CPU⇔GPU communication
- Soon the code will be released to the HEP community in the standard RooFit

Future work (1)

- A technical student, Yngve Sneen Lindal, is working to have an OpenCL implementation
  - Possibility to have hardware-independent code, i.e. GPUs and CPUs
  - Preliminary tests show that we have same performance of the CUDA implementation, with a minimal effort to implement the new code
  - However OpenCL implementation doesn't scale well when running on the CPU, so a *common implementation seems not the best solution* in terms of performance
    - Hybrid: OpenMP on CPU and CUDA/OpenCL on the GPU
  - In contact with a guru, Tim Mattson (Intel)

Future work (2)

- OpenMP is the current reference for the CPU
  - Limitation to the scalability from OpenMP overhead: try to improve the implementation (there is also a NUMA effect to take in account in the algorithm)
- A preliminary implementation of the code based on TBB show that we don't gain in performance
  - TBB has a better programming style suitable for C++ code
- Working to use other technologies for parallelization, such as Intel Concurrent Collections C++ (CnC)
  - Collaboration with Intel experts

Future work (3)

- This summer will start an implementation based on MPI
  - Our algorithm doesn't require a lot of communications
    - Suitable for systems with commodity network links
    - Tests on Intel Micro-server
  - Possibility to reduce the OpenMP overhead, running MPI on the same node
    - Increase of the memory footprint to take in account
  - Hybrid parallelization with OpenMP and CUDA/OpenCL

- We are working on the evaluation of the Knights Ferry (32 cores) and soon of the Single-Chip Cloud Computer (48 cores, no cache coherency), as part of the collaboration with Intel
  - Very promising architectures for massive parallelization with intensive calculations

Backup slides

Benchmarks

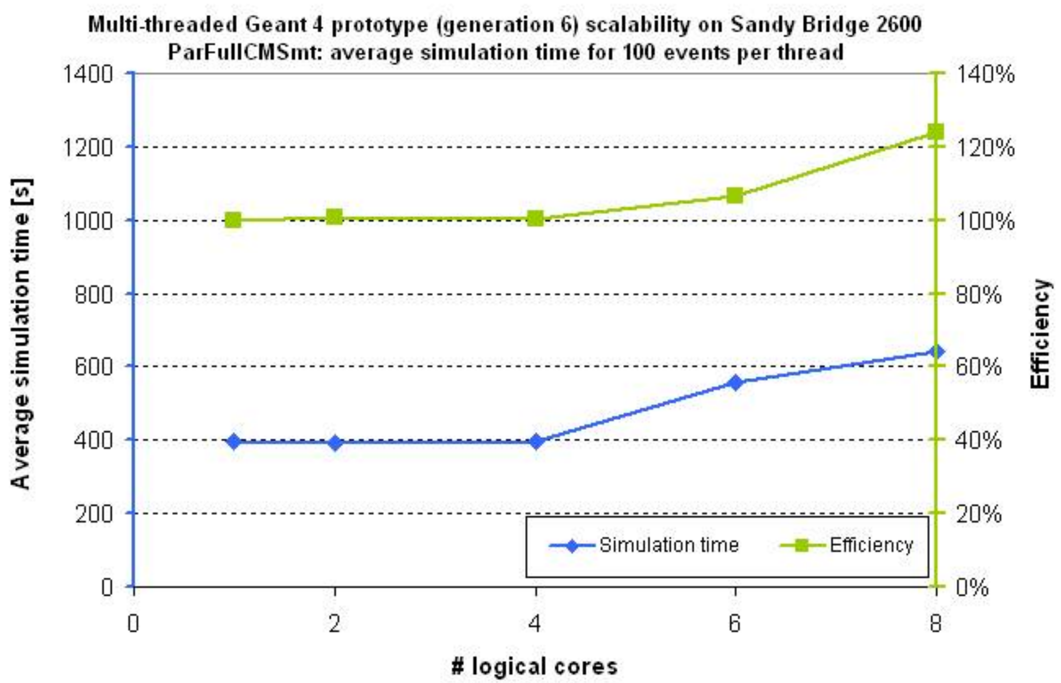
- "Bouquet" of applications to check different HEP workloads on different hardware
  1. HEPSPEC06 performance
     - "Brute" performance with the standard HEP benchmark
  2. Multi-threaded Geant4 prototype (Offline Simulation)
     - Throughput performance scalability (pthreaded workload)
  3. Parallel Maximum Likelihood fit with ROOT/RooFit (Data analysis)
     - Strong scaling (latency)
     - Prototype developed by us (Vectorized, OpenMP, MPI)
  4. ALICE Trackfitter/Trackfinder (Online)
     - Throughput performance scalability (Vectorized, pthread, OpenMP, ArBB)
- Other benchmarks
  - Power consumption vs performance: HEPSPEC06 per Swiss Franc per Watt
  - Non Uniform Memory Access aspects
  - Solid State Disk performance

Evaluations ongoing

- Intel Sandy Bridge ("tock" at 32nm)
  - New design respect to previous CPU, i.e. Westmere
    - Many new features, such as introduction of AVX instructions for vector operation at 256bit
  - Desktop version (single socket)
  - Core i7-2600 CPU @ 3.4GHz, 4 cores, 4GB memory
- AMD Magny-Cours (as reference to the Intel systems)
  - Opteron Processor 6164 HE @ 1.7GHz, 12 cores per processor, 48 cores in total, 96GB memory
  - Comparison with respect to Westmere-EP, Nehalem-EX systems
- Intel Micro-Server Proof-of-Concept
- Intel MIC "Knights Ferry Software Development Platform"

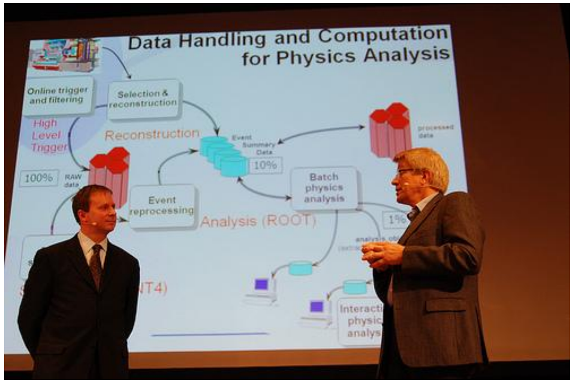
Multi-threaded Geant4 on Sandy Bridge

- Good scalability up to 4 cores
- Hardware multi-threading benefit is 25% (4 to 8 cores)
- Comparing to Westmere-EP, Core i7-2600 (Sandy Bridge based desktop) has ~10% better performance (frequency scaled)
  - Mainly due to the new chip design



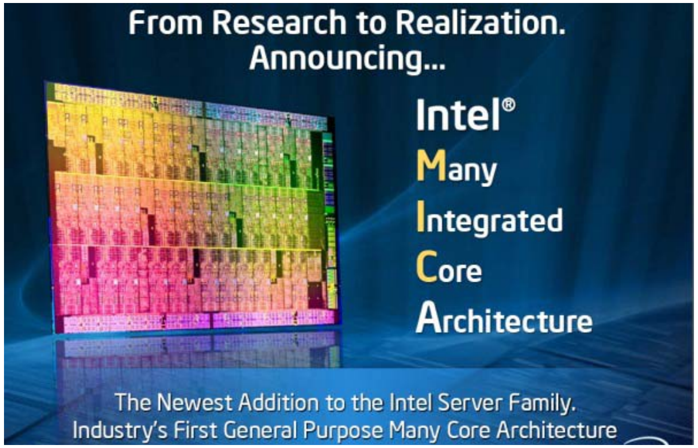
- More details will be available in a report which will be published in the upcoming months

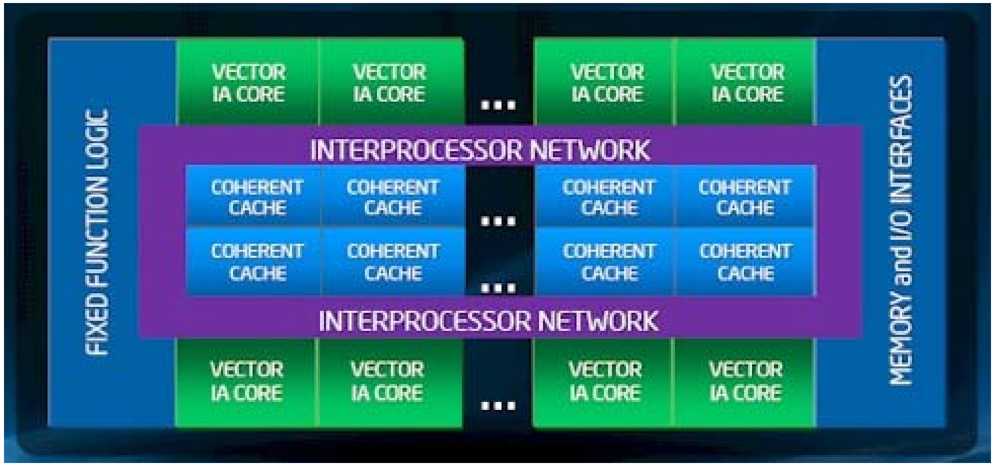
Intel "Many Integrated Cores" architecture



- Announced at ISC10 (June 2010)
  - S. Jarp participated at the presentation
- Current version (codename "Knights Ferry SDP")
  - Enhanced x86 instruction set + vector extensions
  - 32 cores + 4-way hardware multithreaded + 512-bit vector/SIMD units
- Successful (easy) porting of our benchmark applications
  - ALICE Trackfitter/Trackfinder
  - Multithreaded Geant4 prototype
  - Maximum Likelihood data analysis prototype

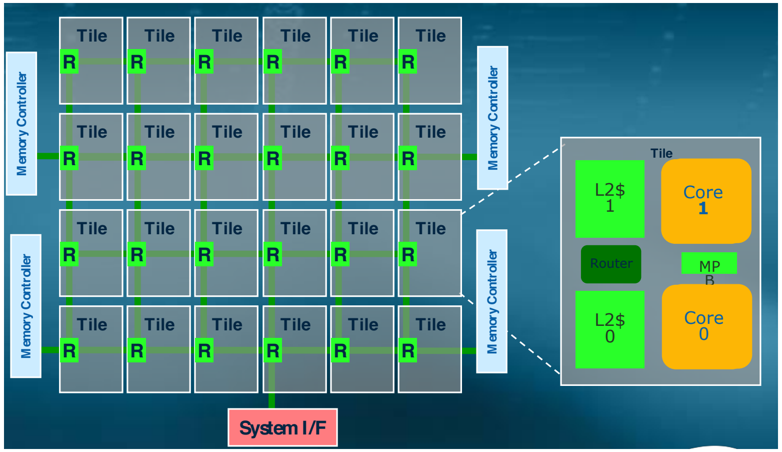
Graphics: INTEL





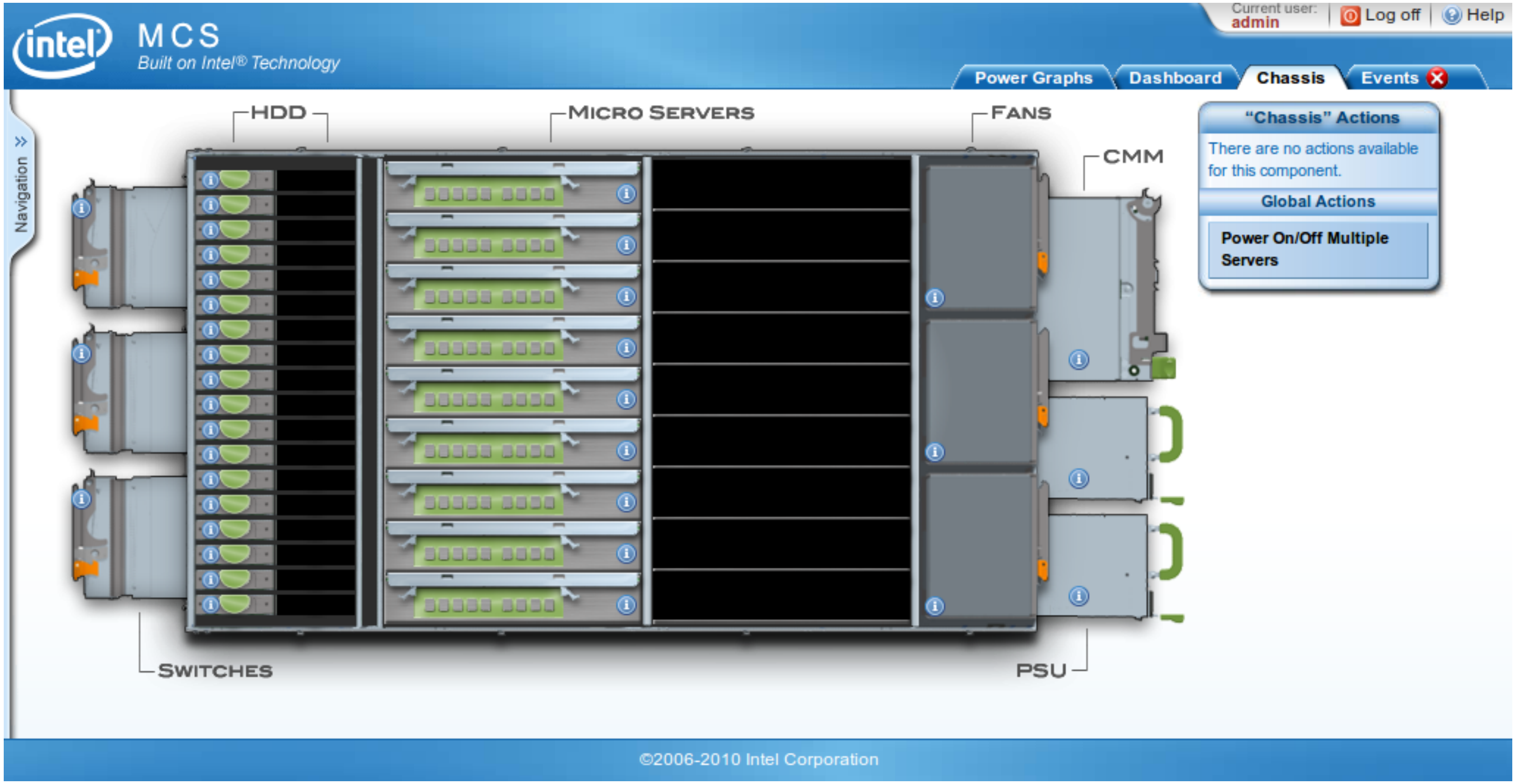
Single-chip Cloud Computer

- 48 Core Research Microprocessor
  - Experimental Research Processor – Not A Product
  - "Cluster-on-die" architecture (new concept): 48 independent Pentium cores
    - Parallel programmability using MPI
  - Interesting possibilities: a lot of parameters can be configured via software, such as operational voltage and frequency
- Our research proposal was accepted and we are waiting for the system to be delivered
  - Participated in MARC forum in Braunschweig (9 November 2010)
  - Close relation with Tim Mattson, who described to us the chip during his visit at openlab in September



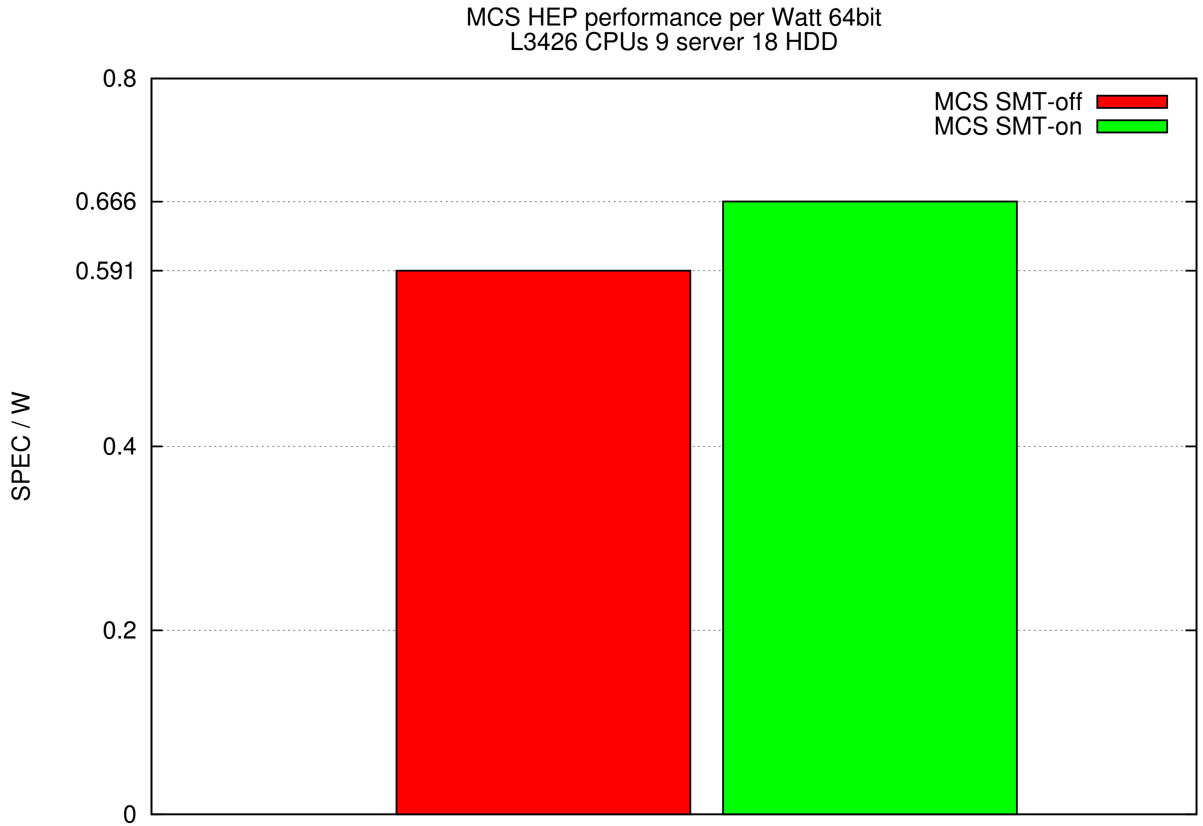
Micro-Server Proof-of-Concept

- Small system that can be densely packed in a larger chassis
- Openlab system embeds
  - 9 microserver boards (18 fit in the chassis)
  - each microserver board counting 1x Intel Xeon Processor L3426 (Nehalem, 8M Cache, 1.86 Ghz, 4 core) + 4x2GB of memory and 2x120GB 2.5" SATA HDD



Micro-Server Proof-of-Concept

- Results for HEPSPEC06 per Swiss Franc per Watt



For reference Westmere-EP has:

- SMT-off: 0.506 (–16%)
- SMT-on: 0.611 (–9%)