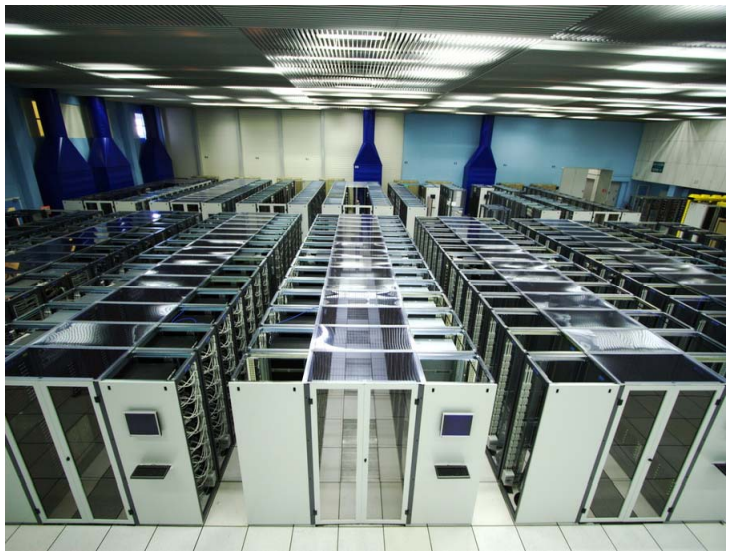
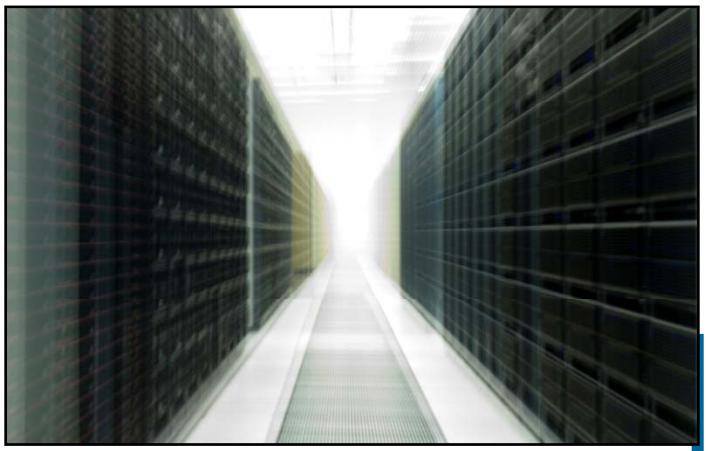
Journées informatiques 2008

Faire face aux nouvelles architectures de processeurs : la physique des particules est-elle prête?





Sverre Jarp

CERN openlab

Obernai – 30.09.2008

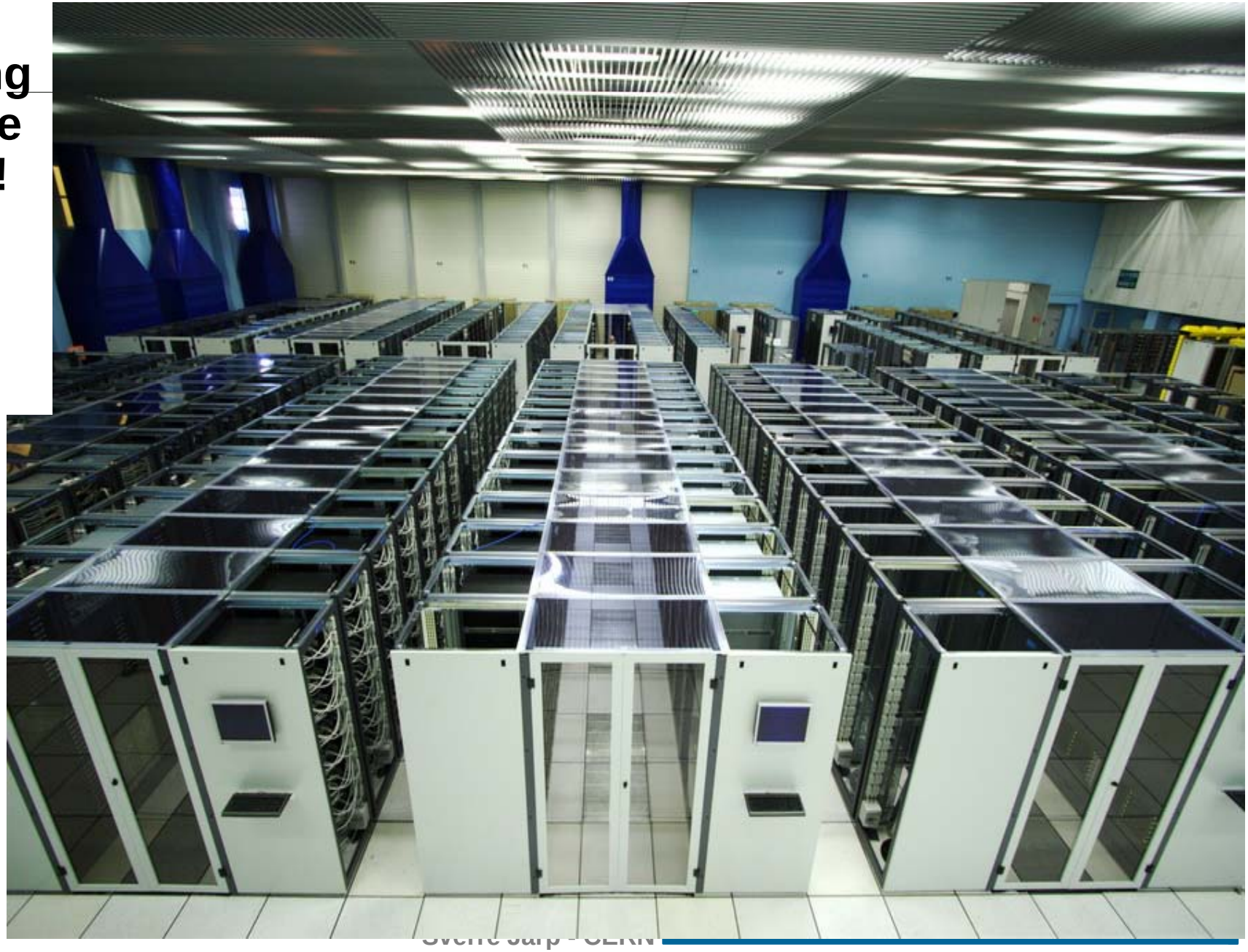
Contents

- **The good news**
  - CERN's computing infra-structure for LHC is ready
  - New silicon processes are on their way
- **The lurking issues**
  - and, there are several!
- **Possible remedies**
  - But are they painless?
- **Conclusions**

Introduction

The Computer Centre and the Grid

- **In the Computing Centre, we are ready!**



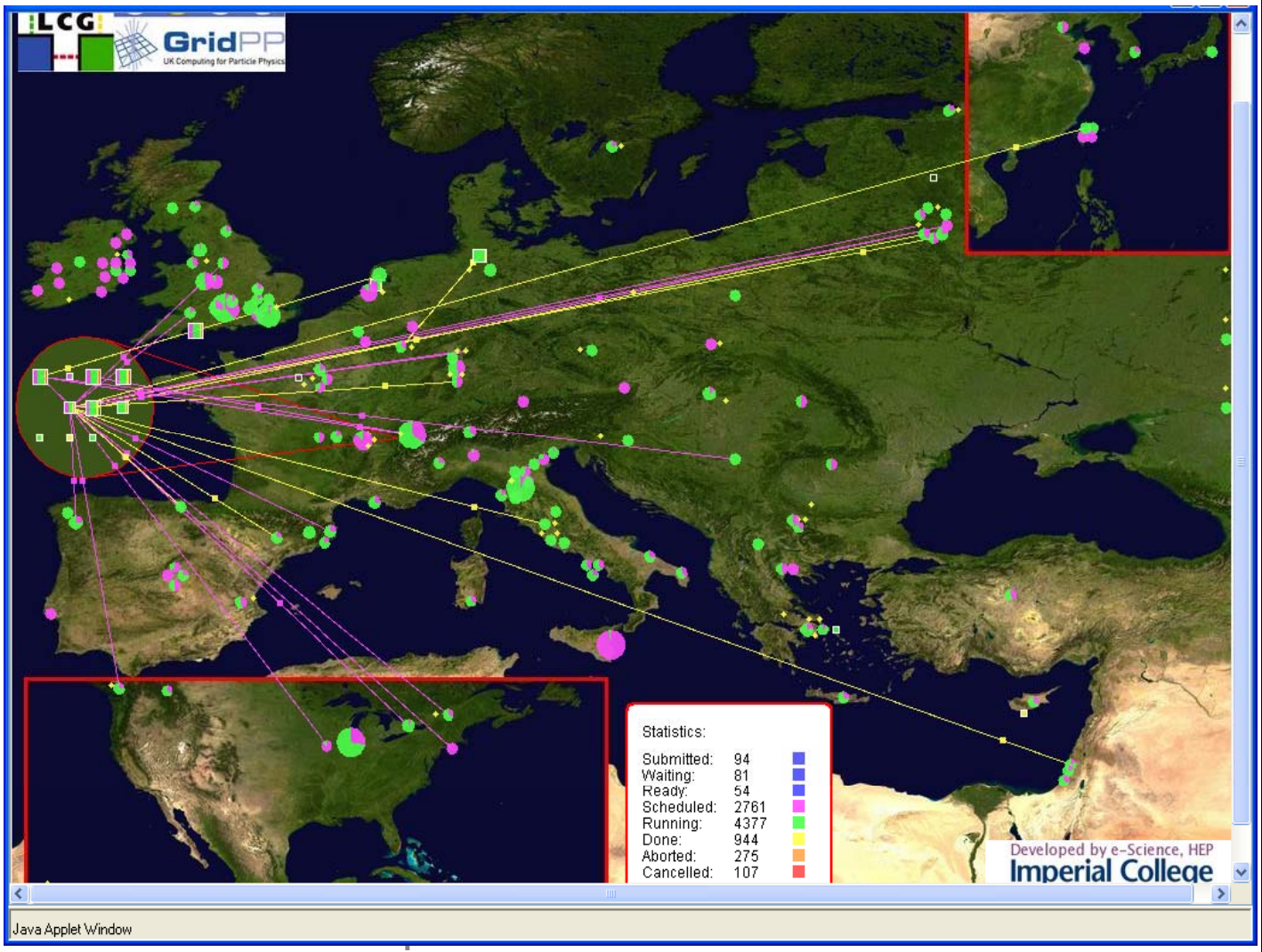
LHC Computing Grid

- **Largest Grid service in the world !**

• Around 200 sites in 40 countries

• Tens of thousands of Linux servers

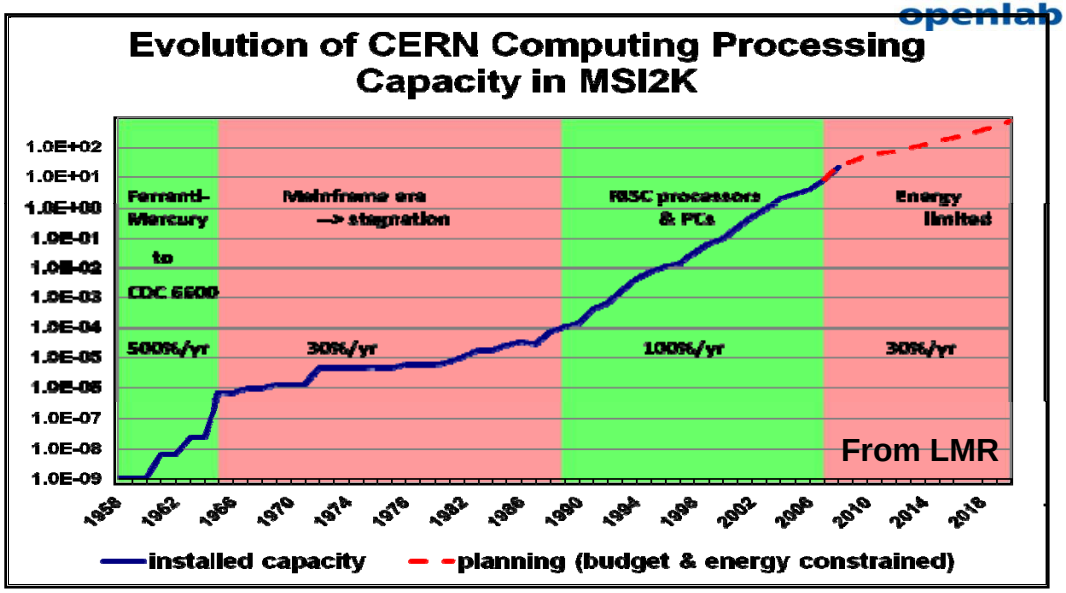
• Tens of petabytes of storage

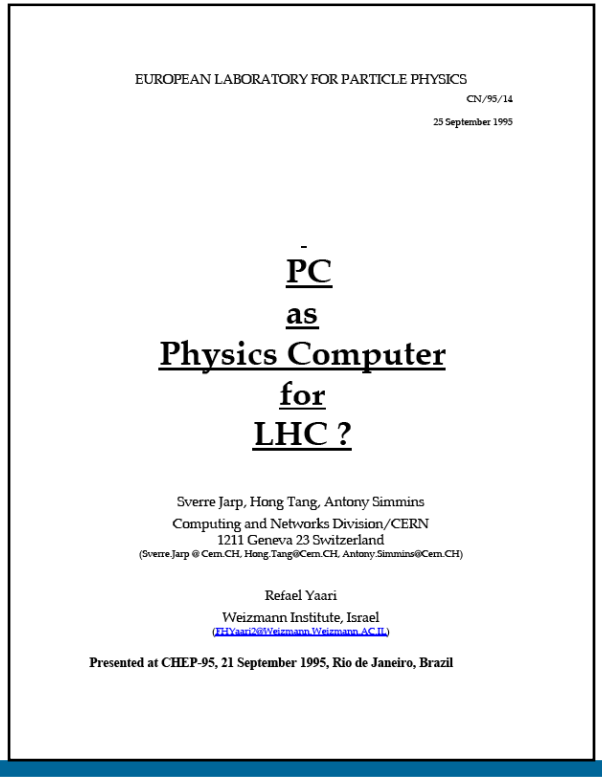


The issues

Evolution of CERN's computing capacity

- **During LEP era (1989 – 2000):**
  - Doubling of compute power every year
  - Initiated with the move from mainframes to RISC systems
- **At the CHEP-95 conference in Rio:**
  - I made the first recommendation to move to PCs





EUROPEAN LABORATORY FOR PARTICLE PHYSICS

CN/95/14

25 September 1995

PC
as
Physics Computer
for
LHC ?

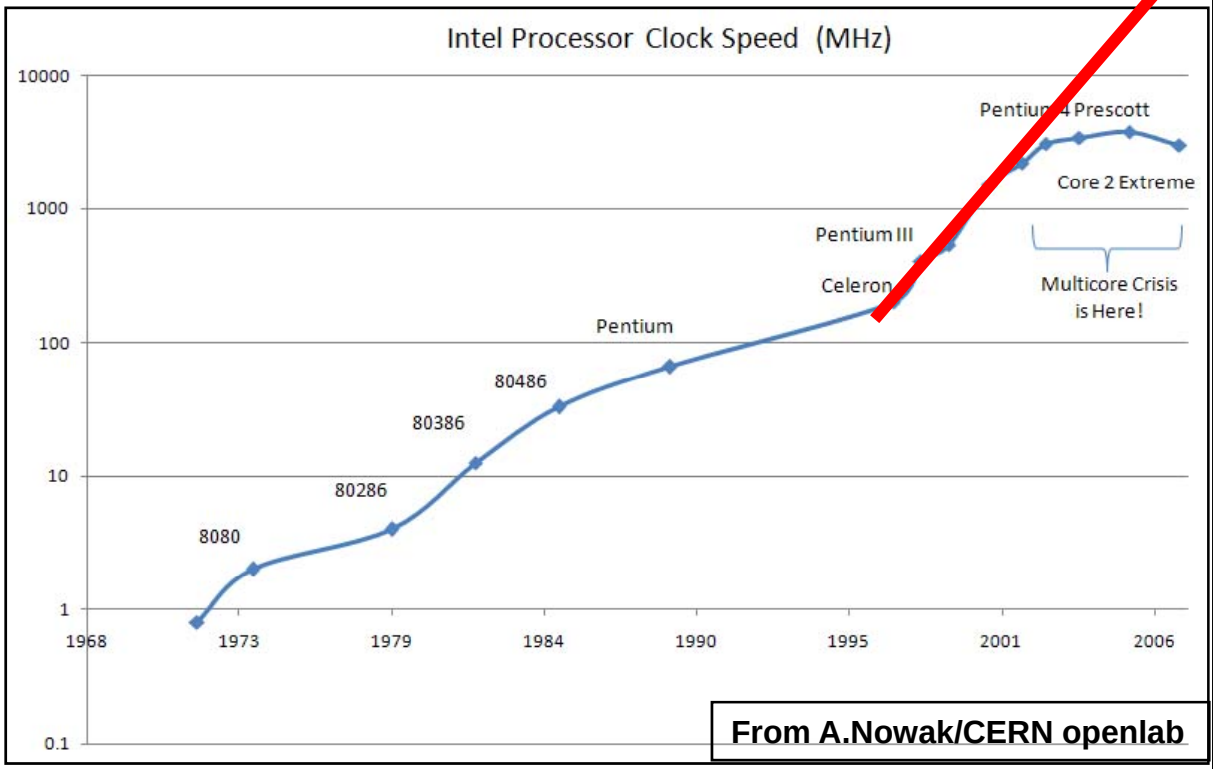
Sverre Jarp, Hong Tang, Antony Simmins
Computing and Networks Division/CERN
1211 Geneva 23 Switzerland
(Sverre.Jarp @ Cern.CH, Hong.Tang@Cern.CH, Antony.Simmins@Cern.CH)

Refael Yaari
Weizmann Institute, Israel
(FHYaari2@Weizmann.Weizmann.AC.IL)

Presented at CHEP-95, 21 September 1995, Rio de Janeiro, Brazil

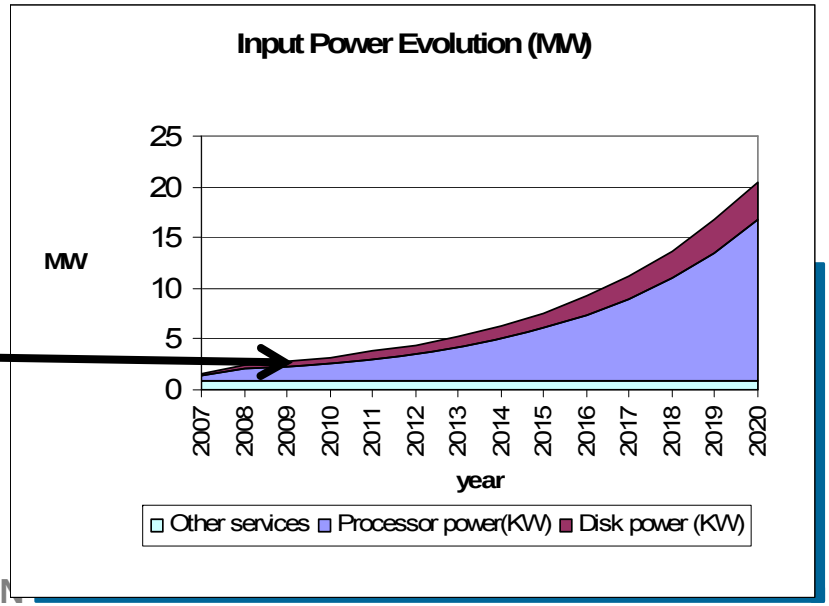
1) Frequency scaling is over!

- **The 7 “fat” years of frequency scaling in HEP**
  - Pentium Pro (1996): 150 MHz
  - Pentium 4 (2003): 3.8 GHz (~25x)
- **Since then**
  - Core 2 systems:
    - Peak: ~3 GHz
    - Quad-core
- **Frequency now frozen!**



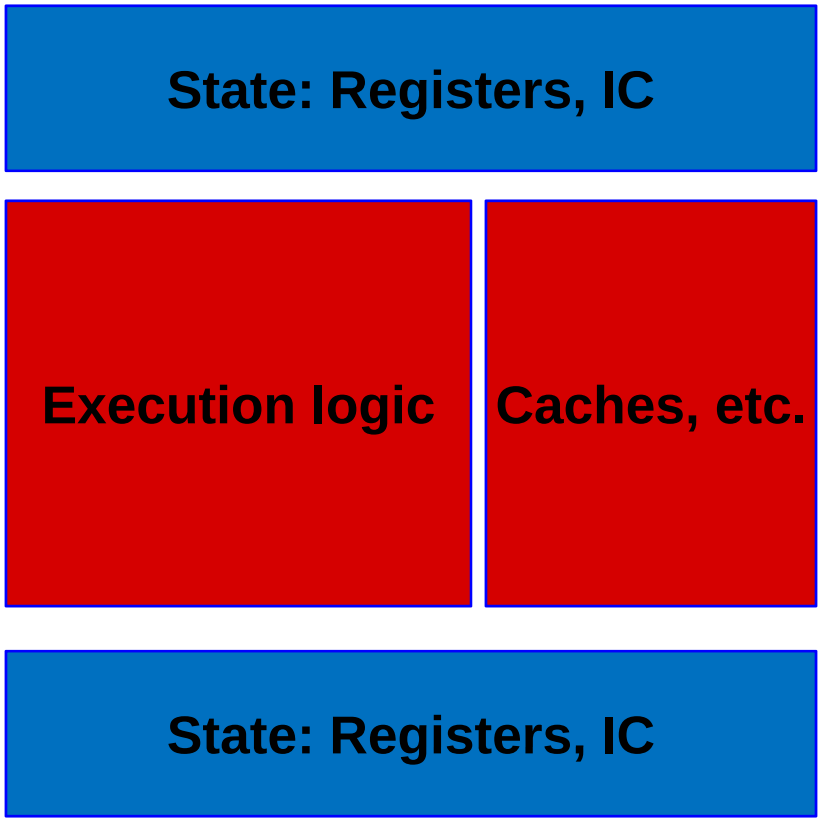
2) The Power Wall

- **The CERN Computer Centre can "only" supply 2.5MW of electric power**
  - Plus 2MW to remove the corresponding heat!
- **Spread over a complex infrastructure:**
  - CPU servers; Disk servers
  - Tape servers + robotic equipment
  - Database servers
  - Other infrastructure servers
    - AFS, LSF, Windows, Build, etc.
  - Network switches and routers
- **This limit will be reached in 2009!**



Definition of a hardware core/thread

- **Core**
  - A complete ensemble of execution logic and cache storage (etc.) as well as register files plus instruction counter (IC) for executing a software process or thread
- **Hardware thread**
  - Addition of a set of register files plus IC
- **Both appear as CPUs**
  - cat /proc/cpuinfo



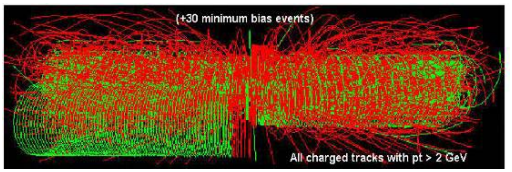
The sharing of the execution logic can be coarse-grained or fine-grained.

The move to many-core systems

- **Let's define "dispatch slots": sockets * cores * hw-threads**
  - The total of what you see in /proc/cpuinfo!
  - Today:
    - Dual-socket Intel quad-core (Harpertown): 2 * 4 * 1 = 8
    - Dual-socket Sun Niagara (T2) processors w/8 cores and 8 threads:
      - 2 * 8 * 8 = 128
  - Tomorrow:
    - Dual-socket Intel Nehalem "octo-core" with dual hw-threading
      - 2 * 8 * 2 = 32
    - Quad-socket Sun Niagara (T2+) processors:
      - 4 * 8 * 8 = 256
- **In the near future: Hundreds of "dispatch slots"!**
- **And, by the time, new software is ready: Thousands !!**

3) Our programming paradigm

- **Event-level parallelism has been used for decades**
  - Process event-by-event in a single process

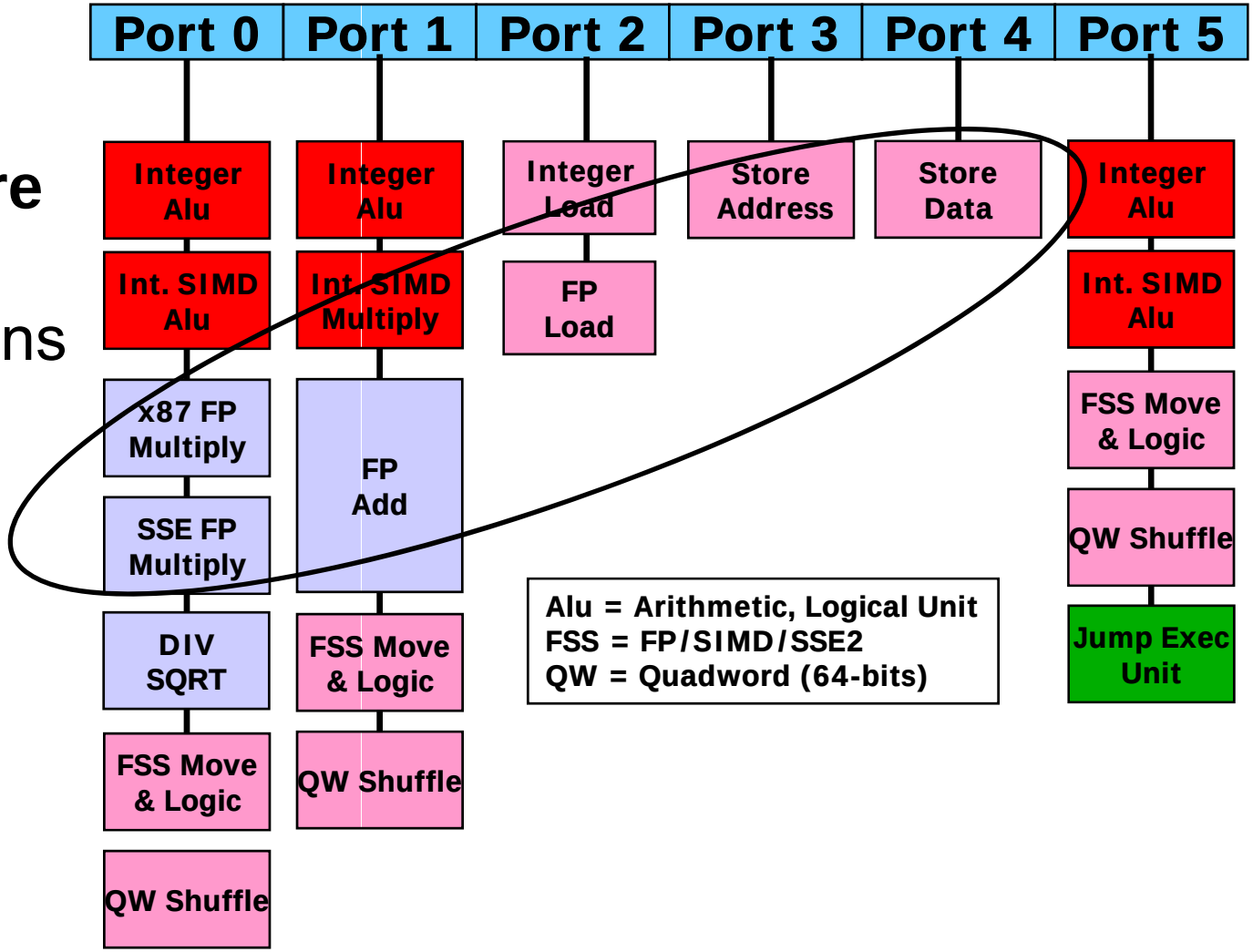


- **Advantage**
  - Large jobs can be split into N efficient processes, each responsible for processing M events
    - Built-in scalability

- **Disadvantage**
  - Memory must be made available to each process
    - With 2 – 4 GB per process
    - A dual-socket server with Quad-core processors
      - Needs 16 – 32 GB (or more) – we currently buy only 16!

Core 2 execution ports

- **Intel's Core microarchitecture can handle:**
  - Four instructions in parallel:
  - Every cycle
  - Data width of 128 bits



13

Issue ports in the Core 2 micro-architecture (from Intel Manual No. 248966-016)

4) HEP code density

- **Averages about 1 instruction per cycle.**
  - This “extreme” example shows even less:

High level C++ code →

```
if (abs(point[0] - origin[0]) > xhalfsz) return FALSE;
```

Assembler instructions →

```
movsd 16(%rsi), %xmm0
subsd 48(%rdi), %xmm0   // load & subtract
andpd _2il0floatpacket.1(%rip), %xmm0 // and with a mask
comisd 24(%rdi), %xmm0 // load and compare
jbe ..B5.3      # Prob 43%  // jump if FALSE
```

Cycle	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
1			load point[0]			
2			load origin[0]			
3						
4						
5						
6		subsd	load float-packet			
7						
8			load xhalfsz			
9						
10	andpd					
11						
12	comisd					
13						jbe

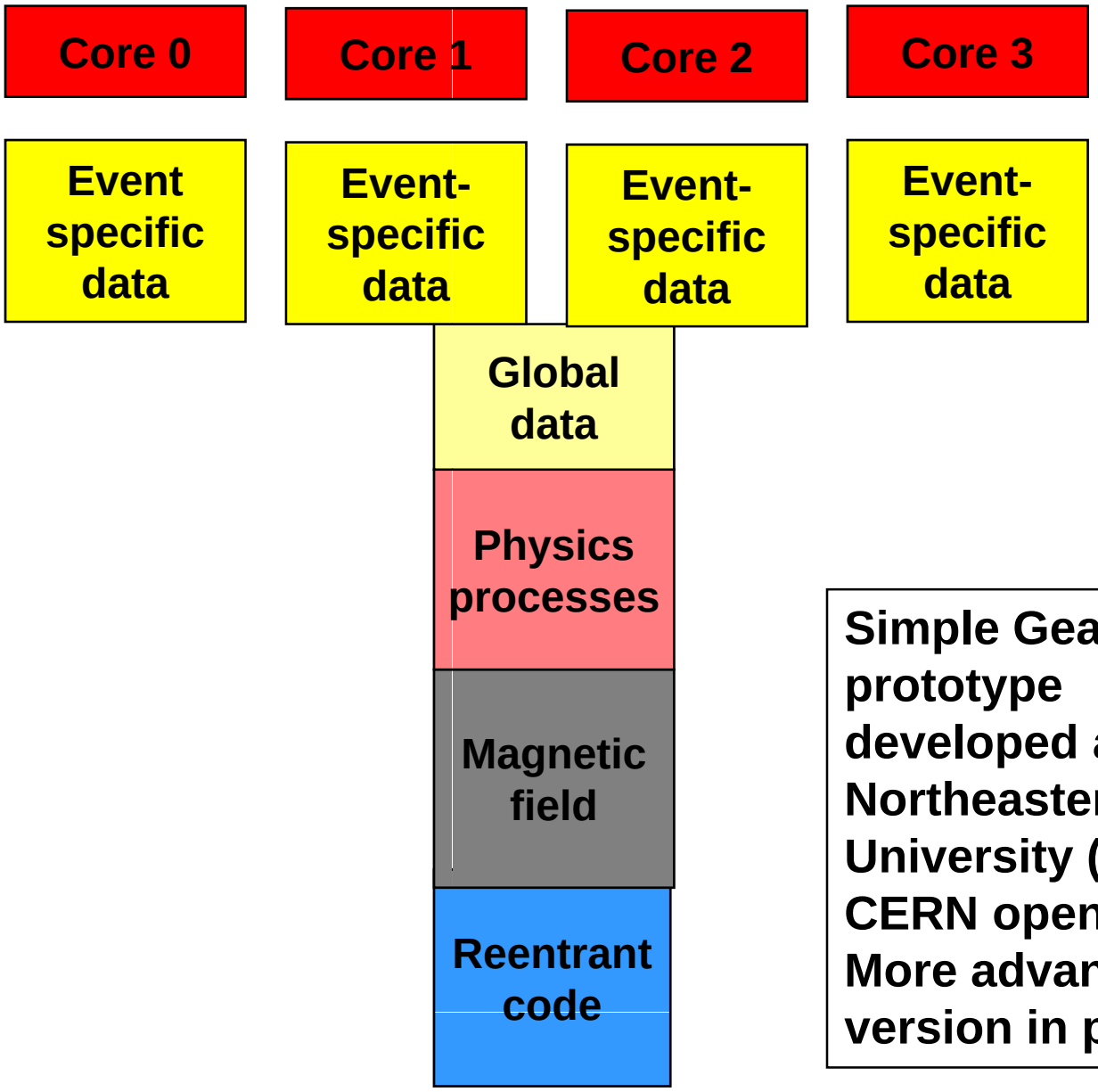
Same instructions laid out according to latencies on the Core 2 processor →

NB: Out-of-order scheduling not taken into account.

Possible remedies

1) More efficient memory footprint

- As follows:



Simple Geant4 prototype developed at Northeastern University (and CERN openlab) – More advanced version in plan.

2) Embrace parallelism

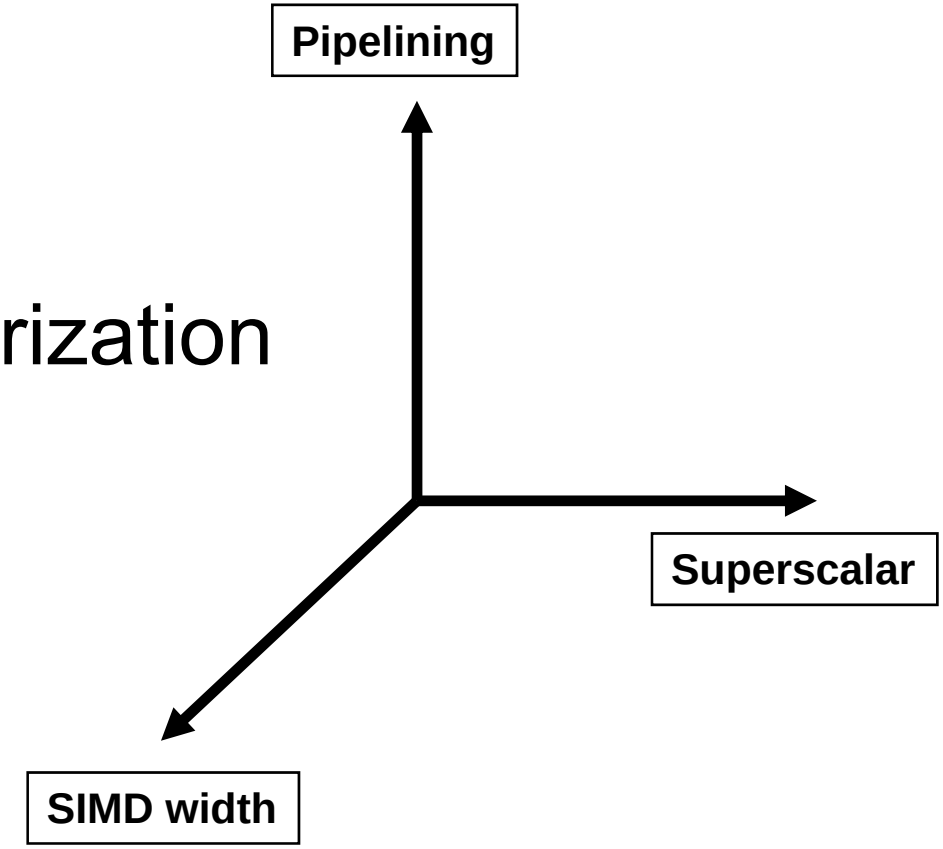
- **Our contribution:**
- **Two openlab workshops arranged together w/Intel in 2007**

- **Each event:**
  - 1 day lectures, 1 day exercises
  - Multiple lecturers (Intel + CERN); 45 participants; typically oversubscribed
  - Survey: 100% said expectations met
  - Next workshop: November 2008

- **Licenses for the Intel Threading Tools (and other SW products) available**
  - to all CERN users

Opportunities for parallelism exist already inside a core

- **First three dimensions:**
- **Data parallelism via**
  - Loop/straight-line vectorization
  - Superscalar: Fill the ports
  - Pipelined: Fill the stages
  - SIMD: Fill the register width



SIMD = Single Instruction Multiple Data
SSE = Streaming SIMD Extensions

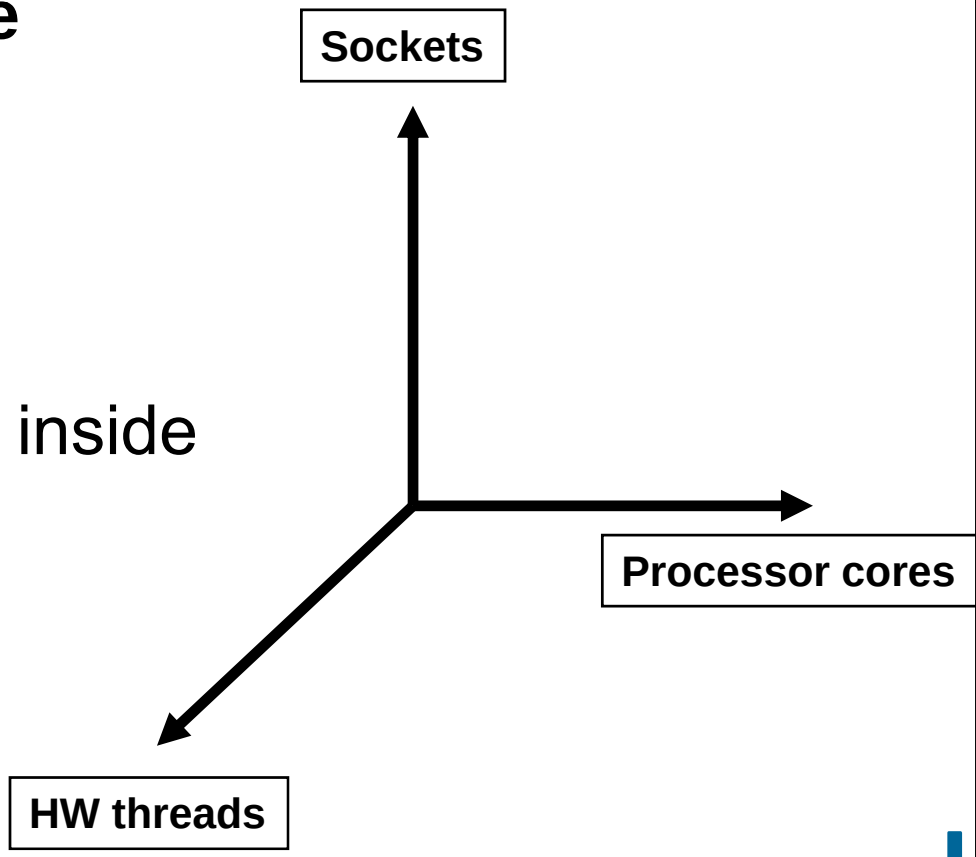
3) Bet on Symmetric Multithreading

- **Provided the memory issue is solved !**
  - We could easily tolerate 4x SMT

Cycle	Port 0	Port 1	Port 2	Port 3	Port 4	Port 5
1			load point[0]			
2			load origin[0]			
3						
4						
5						
6		subsd	load float-packet			
7						
8			load xhalfsz			
9						
10	andpd					
11						
12	comisd					
13						jbe

Parallel execution across multithreaded cores

- **After having tried to maximize capacity within a core**
  - First three dimensions
- **We move to the next level:**
  - Three additional dimensions inside a node:
    - HW threads
    - Processor cores
    - Sockets



- **Ideal for thread-level parallelism**

Seventh dimension represented by multiple nodes.

What are the options?

- **There is currently a discussion in the community about the best way forwards (in a many-core world):**

1) Stay with event-level parallelism (and independent processes)
   - Assume that the necessary memory remains affordable
2) Move to a fully multi-threaded paradigm
   - Using gross-grained (event-level?) parallelism
3) Rely on forking:
   - Start the first process
   - Fork N others
   - Rely on the OS to do “copy on write”, in case pages are written to

Rethink concurrency in HEP

- **We are “blessed” with lots of it:**
  - Events
  - Particles, tracks and vertices
  - Physics processes
  - I/O streams (Trees, branches)
  - Buffer manipulations (also compaction, etc.)
  - Fitting variables
  - Partial sums, partial histograms
  - and many others …..

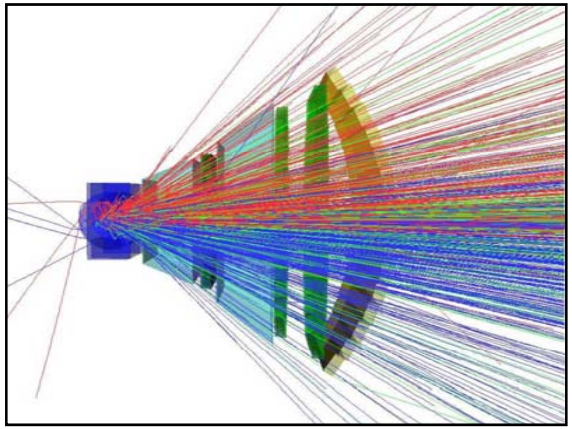
C++ multithreading support

- **Beyond auto-vectorization/auto-parallelization,**
- **Large selection of low-level tools:**
  - OpenMP
  - MPI
  - pthreads/Windows threads
  - Threading Building Blocks (TBB)
  - TOP-C (from NE University)
  - RapidMind
  - Ct (in preparation from Intel)
  - etc.

Examples of parallelism: CBM track fitting

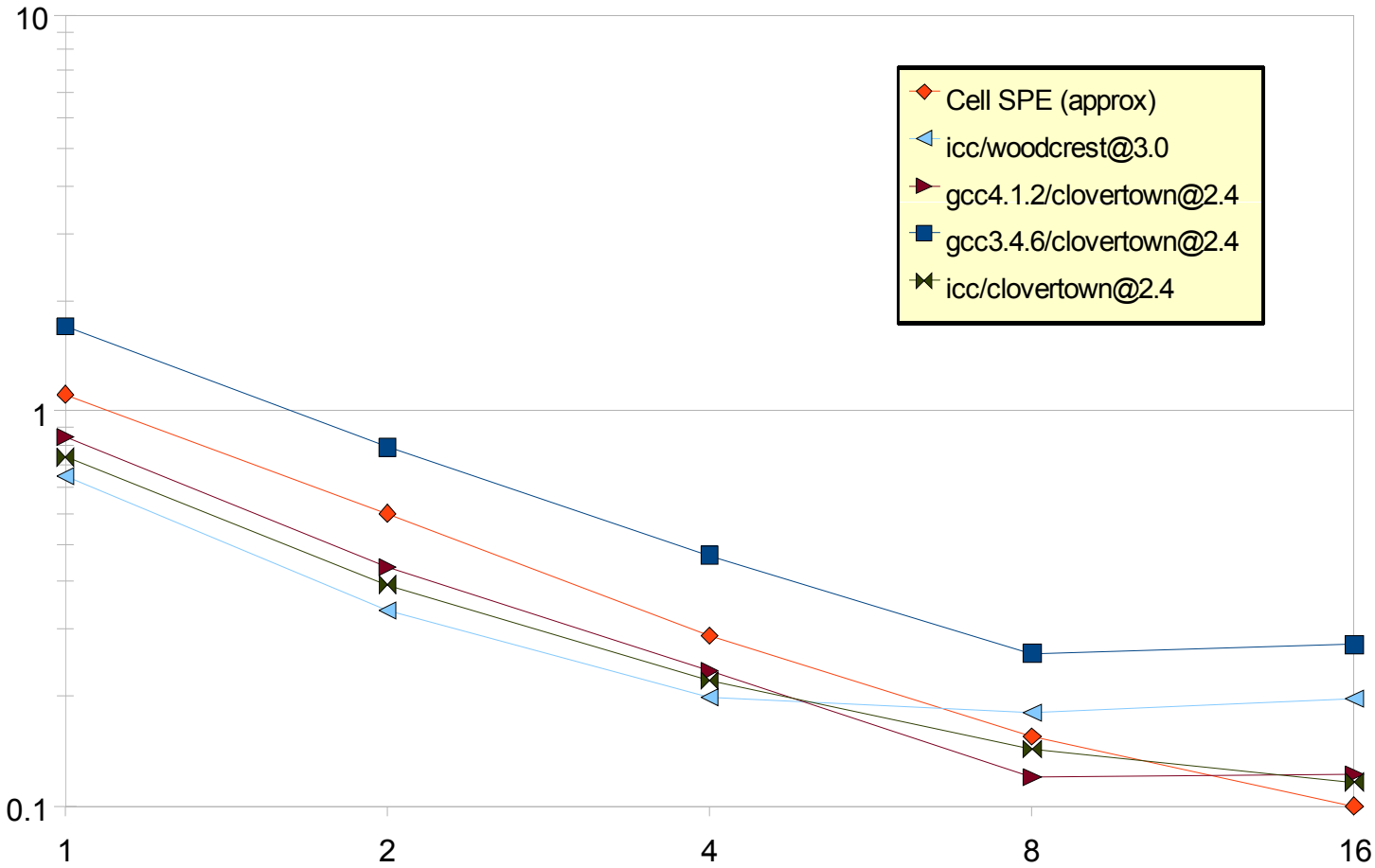
I.Kisel/GSI: "Fast SIMDized Kalman filter based track fit"
http://www-linux.gsi.de/~ikisel/reco/CBM/
DOC-2007-Mar-127-1.pdf

- **Extracted from CBM's High Level Trigger Code**
  - Originally ported to IBM's Cell processor
- **Tracing particles in a magnetic field**
  - Embarrassingly parallel code
- **Re-optimization on Intel Core systems**
  - Step 1: use SSE vectors instead of scalars
    - Operator overloading allows seamless change of data types, even between primitives (e.g. float) and classes
    - Two classes
      - P4_F32vec4 – packed single; operator + = _mm_add_ps
      - P4_F64vec2 – packed double; operator + = _mm_add_pd
  - Step 2: add multithreading (via TBB)
    - Enable scaling with core count



CBM HLT benchmark runs

- **Real fit time/track (μs) as a function of the core count:**



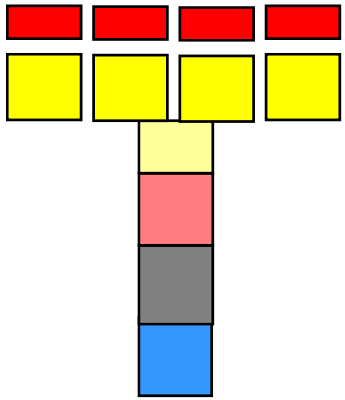
Logarithmic scale!

From H.Bjerke/CERN openlab

Examples of parallelism: GEANT4

- **ParGeant4**
  - implements event-level parallelism to simulate separate events across remote nodes.
  - Typical simulations demonstrate a nearly linear speedup in running time as the processor count increases.
    - See examples/extended/parallel in Geant4 for details

- **New development re-implements thread-safe event-level parallelism inside a multi-core node**
  - Done by NEU student: ExampleN02
  - Required change of lots of existing classes:
    - *static*, *global*, "*extrn*", and *const* declarations
    - Also: CLHEP is not thread-safe
      - affecting the random-number generation

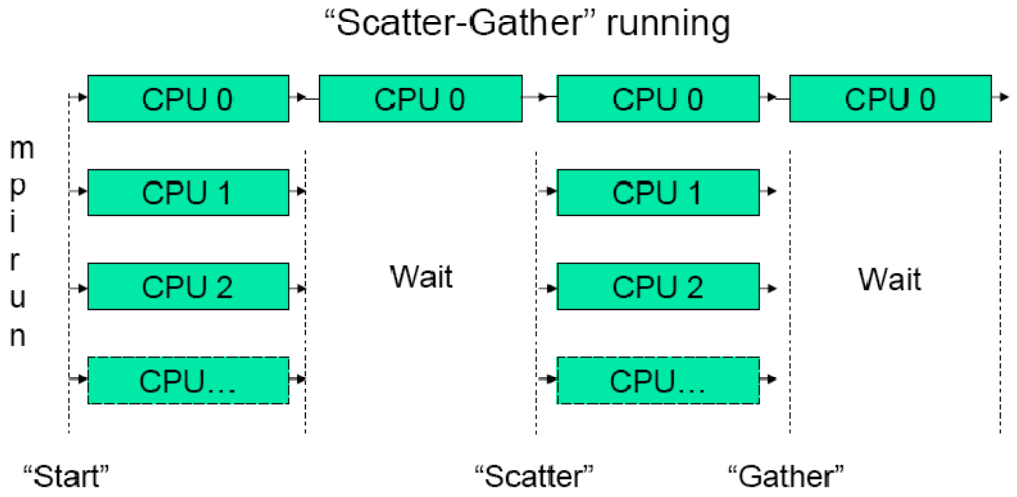


More work is needed, but interesting first step!

Examples of parallelism: RooFit (1)

- **Example of Data Analysis (Fitting) in BaBar (SLAC)**
  - Uses MPI to run scatter/gather
    - Based on the Negative-Log Likelihood function which requires the calculation of separate values for each free parameter in each minimization step

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



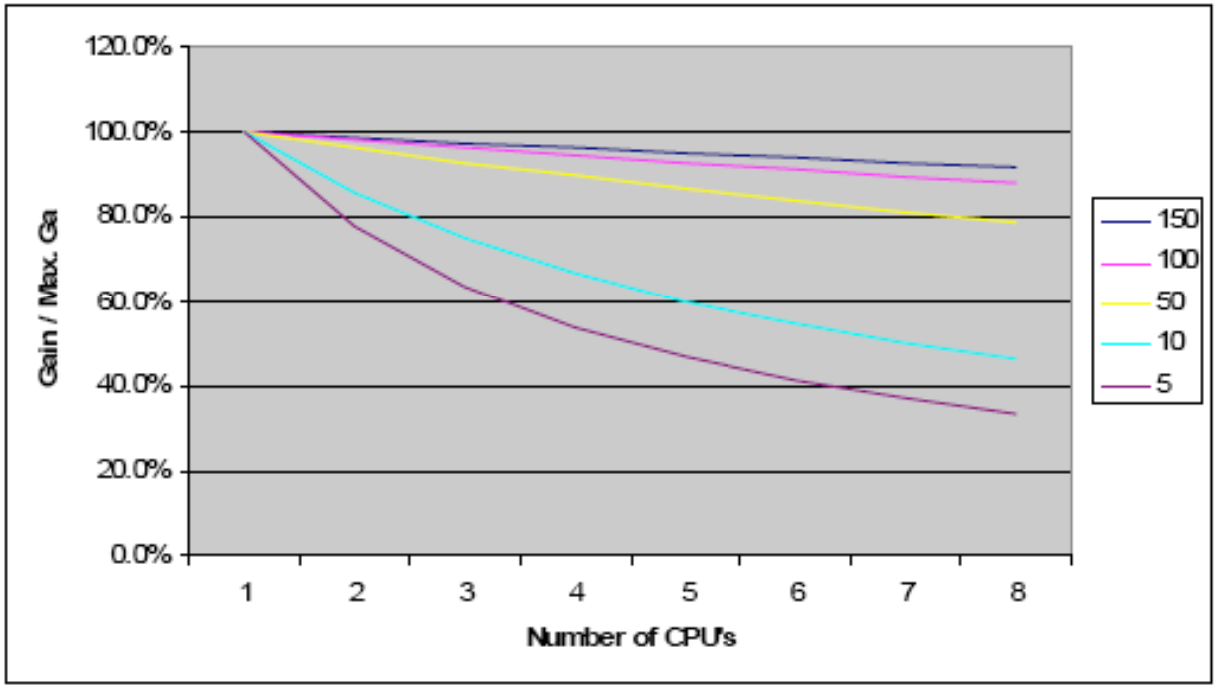
From B.Meadows's talk at RooFit Mini Workshop @ SLAC (December 2007): http://www.slac.stanford.edu/BFROOT/www/doc/Workshops/2007/BaBar_RooFit/Agenda.html

RooFit (2)

- **It works well in case of large number of parameters**

Gain ~ NCPU*(NPAR + 2) / (NPAR + 2*NCPU)

Max. Gain = NCPU



Programming strategies/priorities

- **As I see them:**
  - Get memory usage (per process) under control
    - To allow higher multiprogramming level per server
  - Draw maximum benefit from hardware threading
  - Introduce coarse-grained software multithreading
    - To allow further scaling with large core counts
  - Revisit data parallel constructs at the very base
    - Gain performance inside each core
- **In all cases, use appropriate tools (pfmon/Thread Profiler, etc.)**
  - To monitor detailed program behaviour

Conclusions

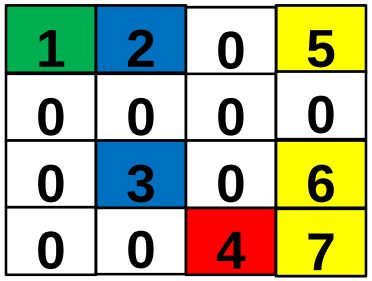
- **In spite of the fact that we have thousands of servers locally (and tens of thousands in the Grid), we are confronted by several computing issues!**
- **Some solutions are easier than others:**
  - Switch on SMT in the BIOS
  - Reduce memory foot-print
  - Gradually introduce parallelism (across threads/cores/sockets)
  - Build an additional (5 MW?) computer centre
  - Revisit vectors (data parallelism)
- **Will we be ready for 1000 cores/threads??**
- **Whilst, maintaining software scalability and portability**
- **The bottom line: master the 7 hardware dimensions!**

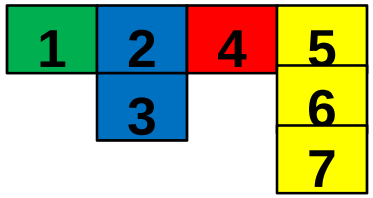
Q & A

Ct Language

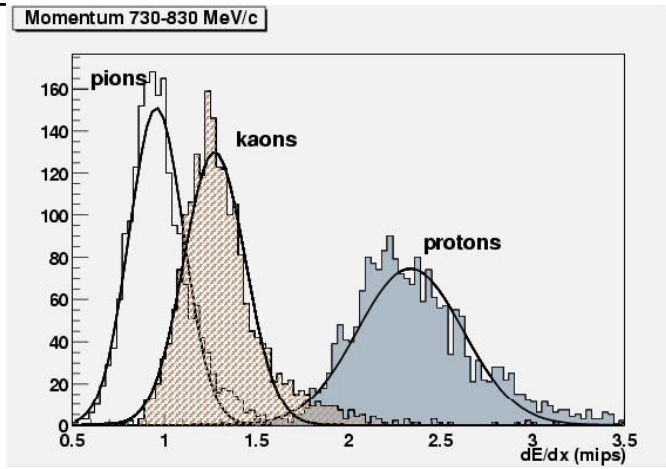
A.Ghuloum/Intel:
Programming Challenges for Manycore Computing

- **New effort by Intel to extend C++ for Throughput Computing**
- **Features:**
  - Addition of new data types (parallel vectors) & operators
    - NeSL/SASAL-inspired: irregularly nested and sparse/indexed vectors
  - Abstracting away architectural details
    - Vector width/Core count/Memory Model: Virtual Intel Platform
      - Forward-scaling (Future-proof!)
    - Nested data parallelism and deterministic task parallelism
- **Incremental adoption path:**
  - Dedicated Ct-enabled libraries
  - Rewritten “kernels” in Ct
  - Pervasive use of Ct





Examples of parallelism: ROOT



- **ROOT:**
  - Originally developed without a strong focus on concurrency
    - Still, set of thread classes provided:
      - TThread, TMutex, TCondition, TRWLock, etc.
  - Work continues in order to improve the situation:
    - Collection Classes:
      - The ROOT Collection Classes have been made thread-safe with one big lock; Need to cover STL collection classes as well
    - Graphics:
      - WIN32 back-end is thread-safe and threaded; Not yet X-Windows
    - ROOT Global State needs protection
  - Wider use of TBB is under investigation