CERN openlab III

Short GPU/multi-core "slide-show" from CHEP-2010

Sverre Jarp, CERN IT
2 November 2010

Presentations/BOFs

- Plenary:
  - The experiment offline systems after one year (R.Jones)
  - How to harness the performance potential of current Multi-Core CPUs and GPUs (S.Jarp)
  - Computing Paths to the Future (R.Goff/DELL)
- Parallel:
  - Multicore-aware Applications in CMS
  - Parallelizing Atlas Reconstruction and Simulation: Issues and Optimization Solutions for Scaling on Multi- and Many-CPU Platforms
  - Multi-threaded Event Reconstruction with JANA
  - Track Finding in a High-Rate Time Projection Chamber Using GPUs
  - Fast Parallel Tracking Algorithm for the Muon System and Transition Radiation Detector of the CBM Experiment at FAIR
  - Real Time Pixel Data Reduction with GPUs And Other HEP GPU Applications
  - Algorithm Acceleration from GPGPUs for the ATLAS Upgrade
  - Maximum Likelihood Fits on Graphics Processing Units
  - Partial Wave Analysis on Graphics Processing Units
  - Many-Core Scalability of the Online Event Reconstruction in the CBM Experiment
- BOF 3:
  - GPUs: High Performance Co-Processors

Plenary on Monday by Roger Jones(ATLAS)

- "The experiment offline systems after one year"

LANCASTER UNIVERSITY 10/18/10 RWL Jones CHEP2010 26

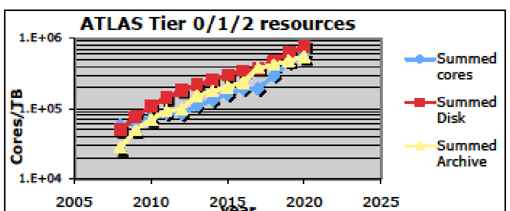
Related developments

- IO challenges being (partly) addressed by fast merging
- Re-write of Gaudi with stronger memory model planned
- Down the line, we may need to parallize the code
  - This could be either for many-core processors or for Graphical Processing Units – but the development might address both
    - GPUs having big success & cost savings in other fields
    - Harder for us to use, but funders will continue to ask
    - We need the R&D to know which path to take
  - Developments require O(3 years) to implement
  - This includes Geant4 – architectural review this year

LANCASTER UNIVERSITY 10/18/10 RWL Jones CHEP2010

Future Challenges

- We assume we can use growth in CPU
  - But this implies changing architectures
  - And handle the data throughput



- Experiments already working to deal with multi cores
  - Many cores and GPGPUs are down the line
- We need to use them or be very clear why we cannot

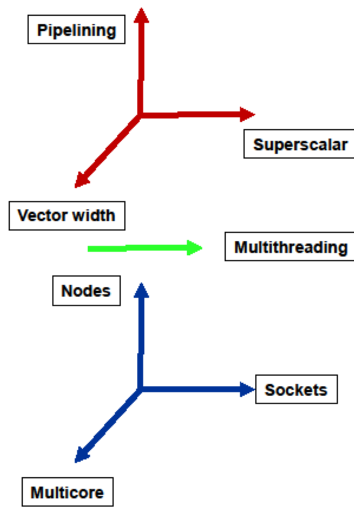
Plenary on Monday by S.Jarp

- "How to harness the performance potential of current Multi-Core CPUs and GPUs"

CHEP 2010, Taipei

Today: Seven dimensions of multiplicative performance

- **First three dimensions:**
  - Pipelined execution units
  - Large superscalar design
  - Wide vector width (SIMD)
- **Next dimension is a "pseudo" dimension:**
  - Hardware multithreading
- **Last three dimensions:**
  - Multiple cores
  - Multiple sockets
  - Multiple compute nodes



SIMD = Single Instruction Multiple Data

Sverre Jarp - CERN

CHEP 2010, Taipei

What are the multi-core options?

- **There is a discussion in the community about the best way(s) forward:**

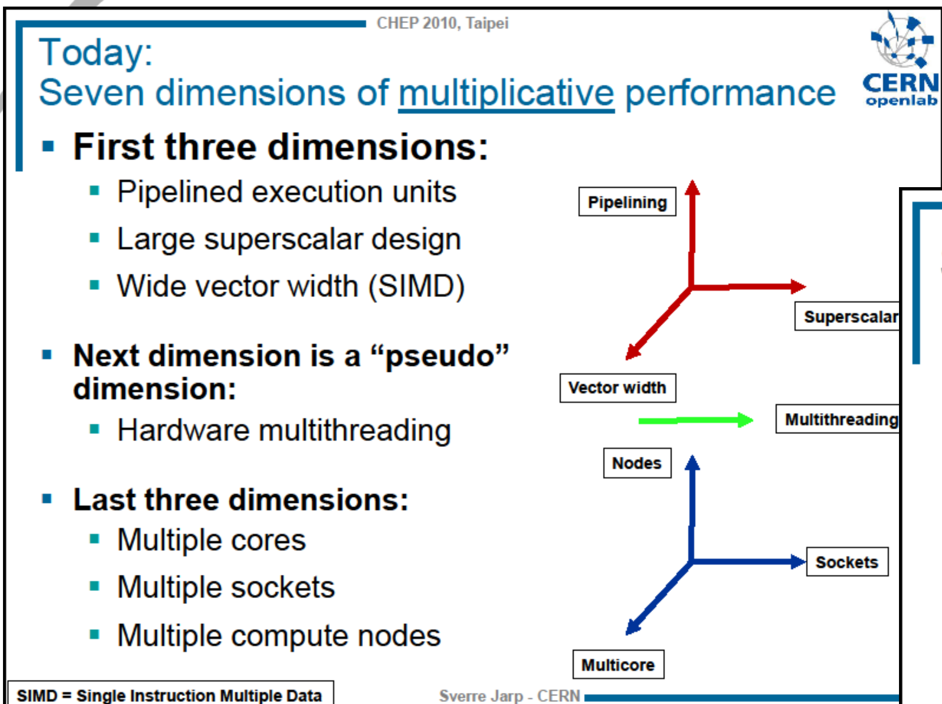
1) Stay with event-level parallelism (and entirely independent processes)
   - Assume that the necessary memory remains affordable
   - Or rely on tools, such as KSM, to help share pages
2) Rely on forking:
   - Start the first process; Run through the first "event"
   - Fork N other processes
   - Rely on the OS to do "copy on write", in case pages are modified
3) Move to a fully multi-threaded paradigm
   - Still using coarse-grained (event-level) parallelism
     - But, watch out for increased complexity

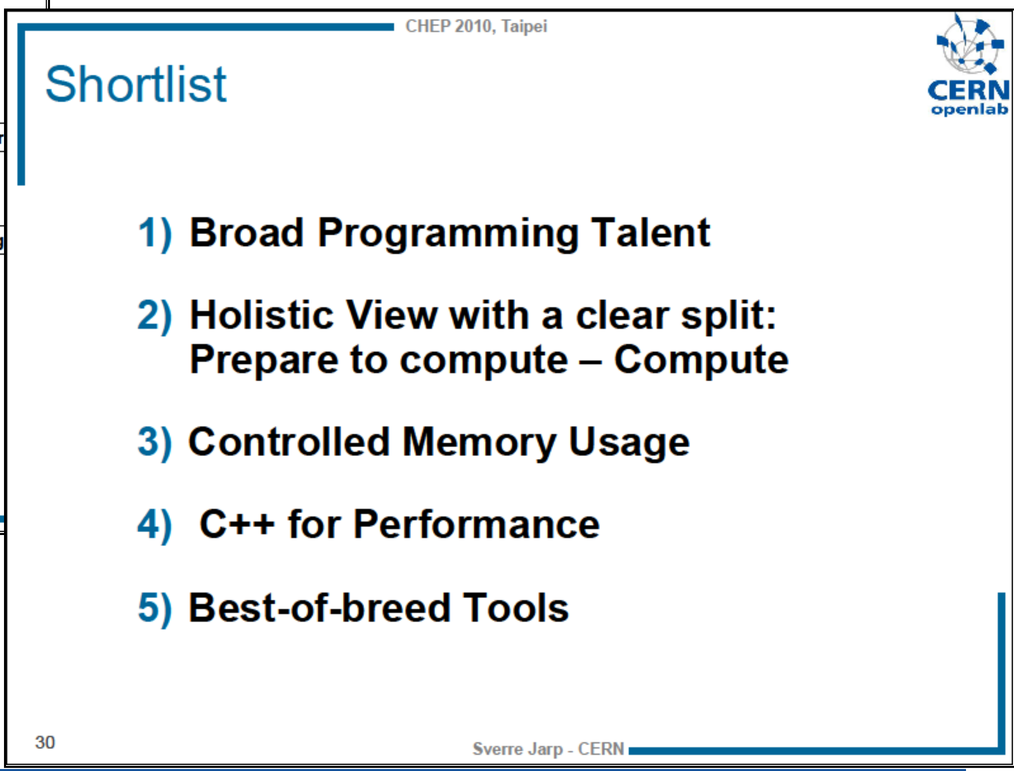
17

Sverre Jarp - CERN

Plenary on Monday by S.Jarp

- "How to harness the performance potential of current Multi-Core CPUs and GPUs"





Plenary on Tuesday by Roger Goff/DELL

- Plenary “vendor” session:

Computing Paths to the Future

Roger Goff
Dell Global CERN/LHC Technologist
+1 970 672 1252 | Roger_Goff@dell.com

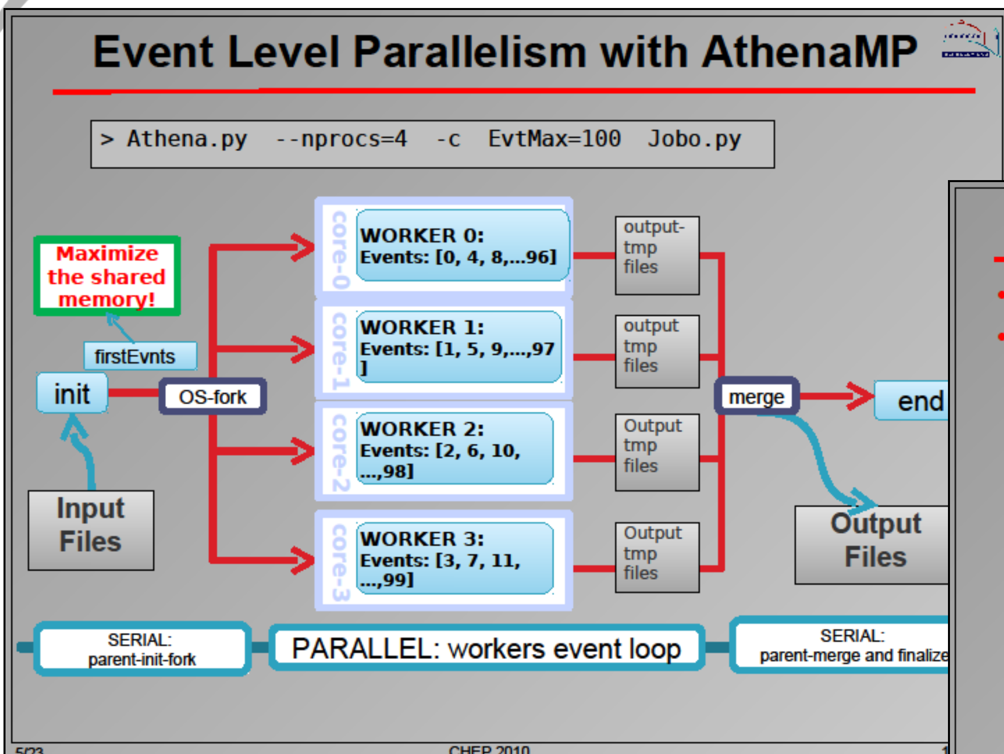
Final Takeaways

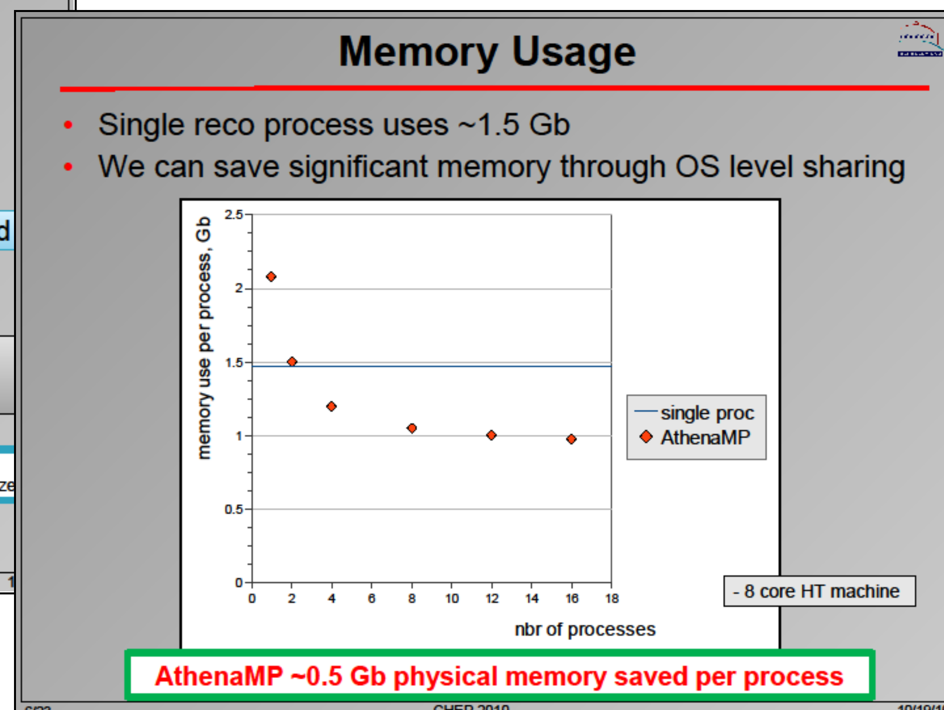
1. CPU cores are not getting faster.
2. Co-processors are here to stay.
3. Heterogeneous processors are inevitable.
4. Preparing applications for extreme parallelism will enable users to get the most out of future systems.

8 Dell CERN/LHC Program

Charles Leggett/ATLAS

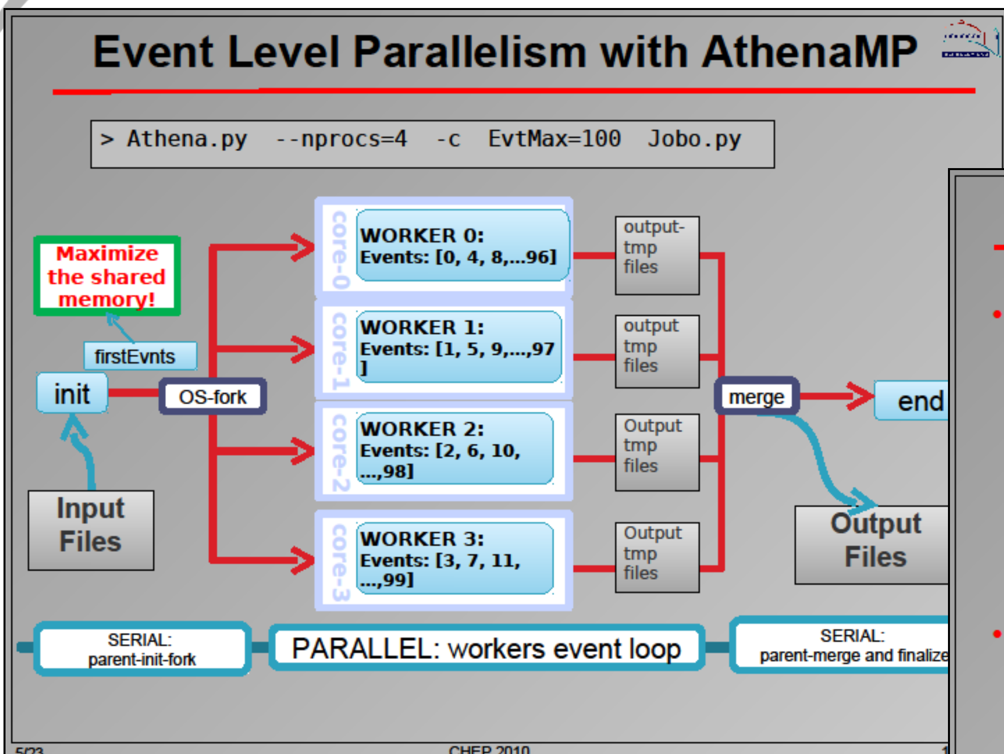
- “Parallelizing Atlas reconstruction and simulation on multi-core platforms”

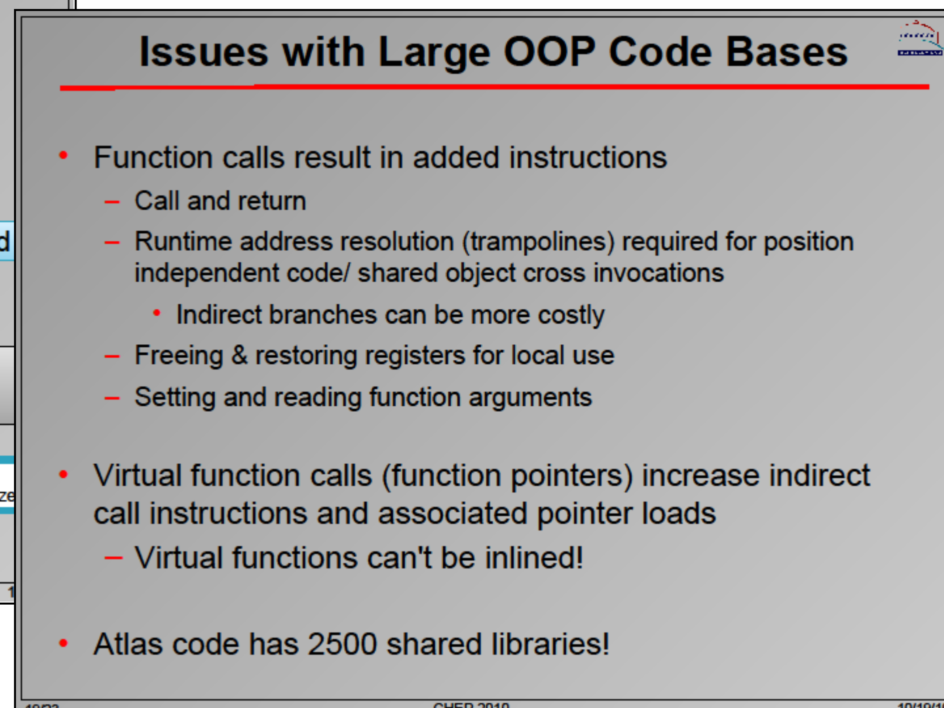




Charles Leggett/ATLAS

- “Parallelizing Atlas reconstruction and simulation on multi-core platforms”





Issues with Large OOP Code Bases

- Function calls result in added instructions
  - Call and return
  - Runtime address resolution (trampolines) required for position independent code/ shared object cross invocations
    - Indirect branches can be more costly
  - Freeing & restoring registers for local use
  - Setting and reading function arguments
- Virtual function calls (function pointers) increase indirect call instructions and associated pointer loads
  - Virtual functions can't be inlined!
- Atlas code has 2500 shared libraries!

19/23 CHEP 2010 10/19/10

- “Multi-core aware Applications in CMS”

Why Bother?

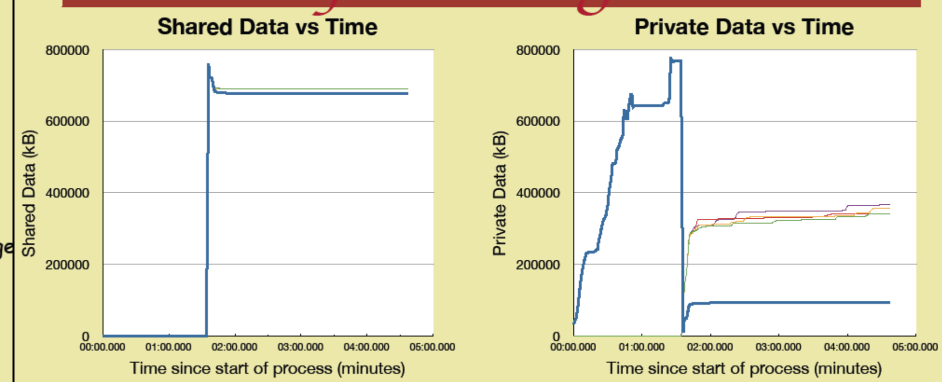
HEP processing is naturally parallelizable
- We have billions of events
- Each event can be processed independently

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



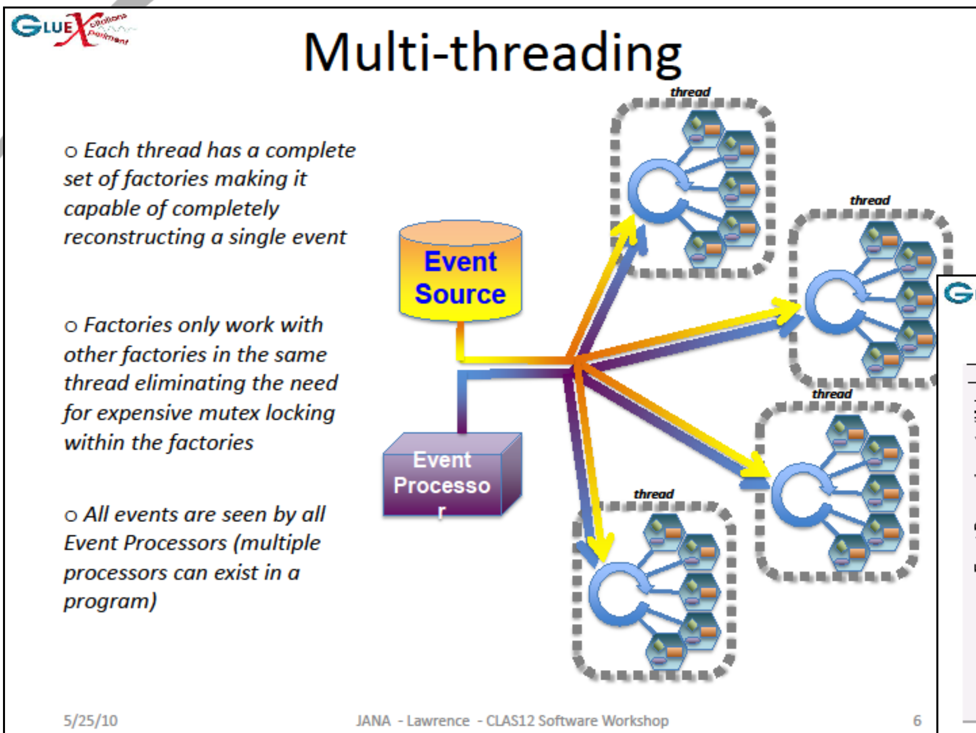
Measurements done using reconstruction with 64bit software on 4 CPU, 8 core/CPU 2GHz AMD Opteron(tm) Processor 6128
- Shared memory per child: ~700MB
- Private memory per child: ~375MB
- Total memory used by 32 children: 13GB
- Total memory used by 32 separate jobs: 34 GB

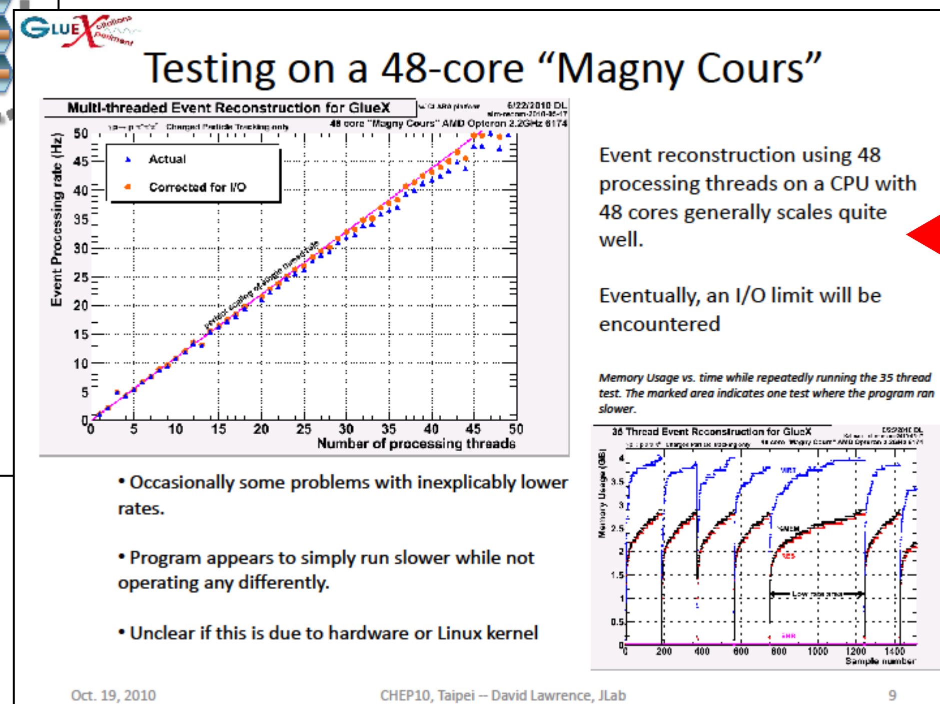
Saved 62% of memory

Multi-core Aware Applications in CMS 11 CHEP 2010

David Lawrence/JLab

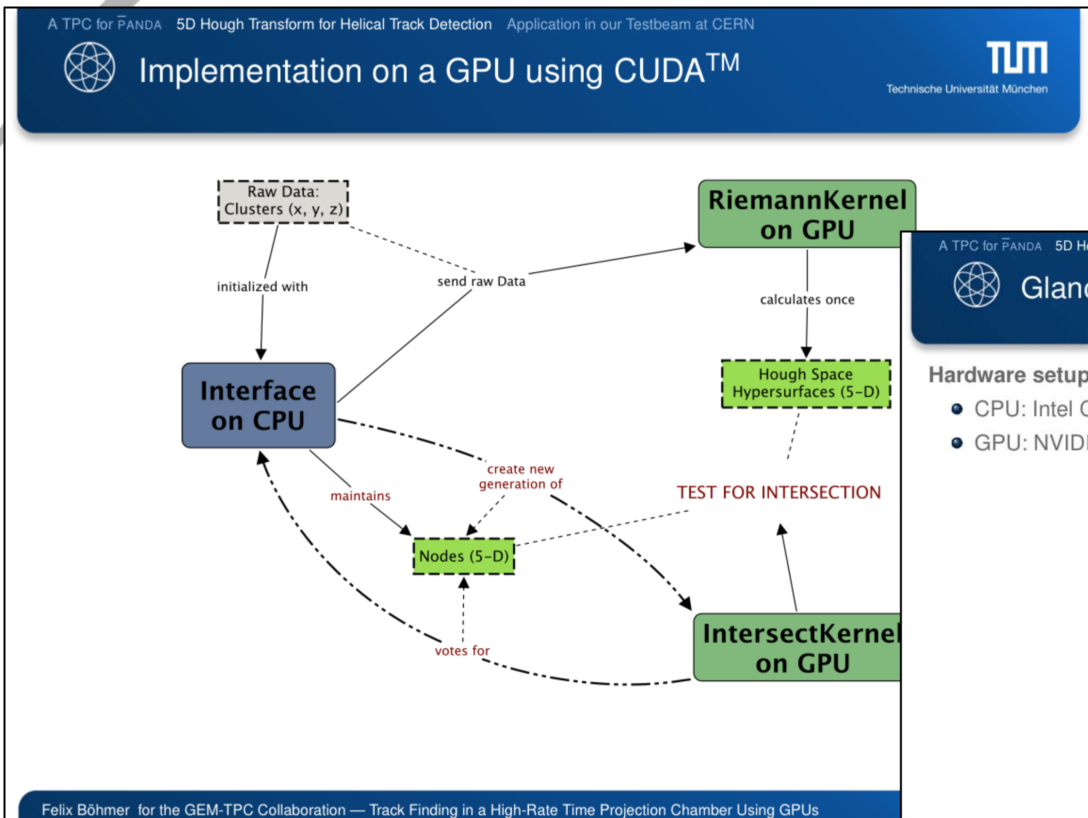
- “Multithreaded event reconstruction with JANA”

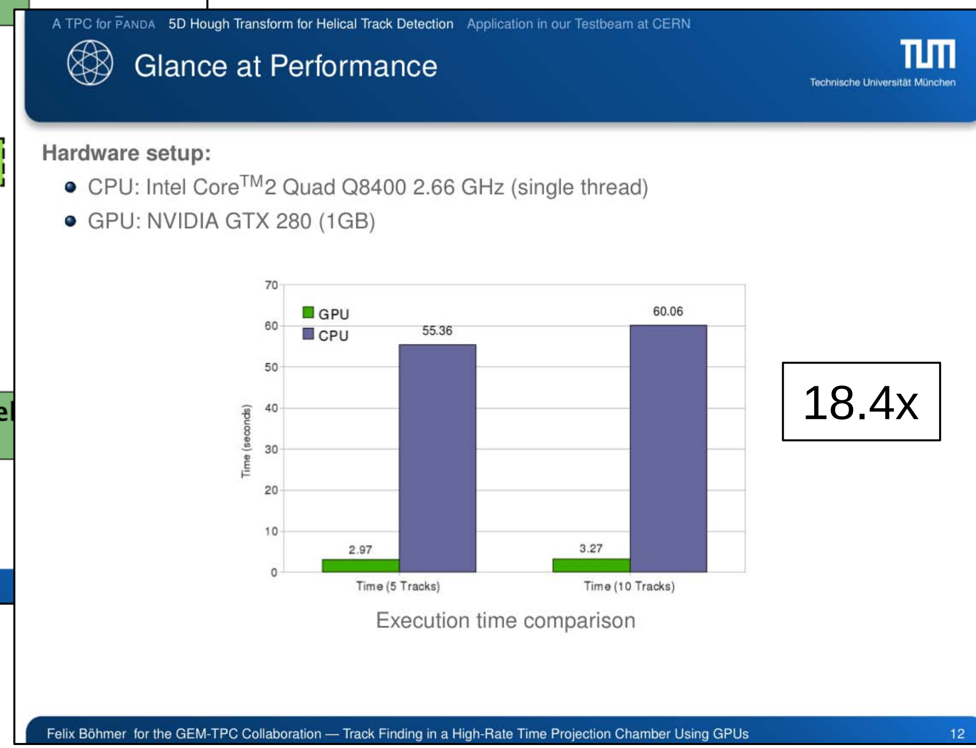




Felix Böhmer/PANDA

- "Track finding in a high-rate time projection Chamber using GPUs"





18.4x

- "Partial wave analysis at BES III – Harnessing the power of GPUs"

Parallel PWA on GPU

Events are independent - calculate terms in the sum in parallel
- Use many PCs
- Use parallel hardware and make use of Single Instruction - Multiple Data (SIMD) capabilities
- Very strong here: Graphics processors (GPUs): Cheap and powerful hardware

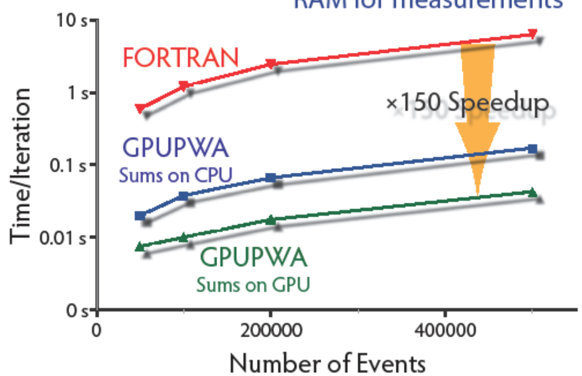
PWA is embarassingly parallel:
- Exactly the same (relatively simple) calculation for each event
- Every event has its own data, on fit parameters are shared
- Ideal for GPU implementation
- True for many HEP applications

13 … e Analysis on GPUs — Niklaus Berger CHEP 2010, T…

Performance

We use a toy model $J/\psi \rightarrow \gamma K^+K^-$ analysis for all performance studies

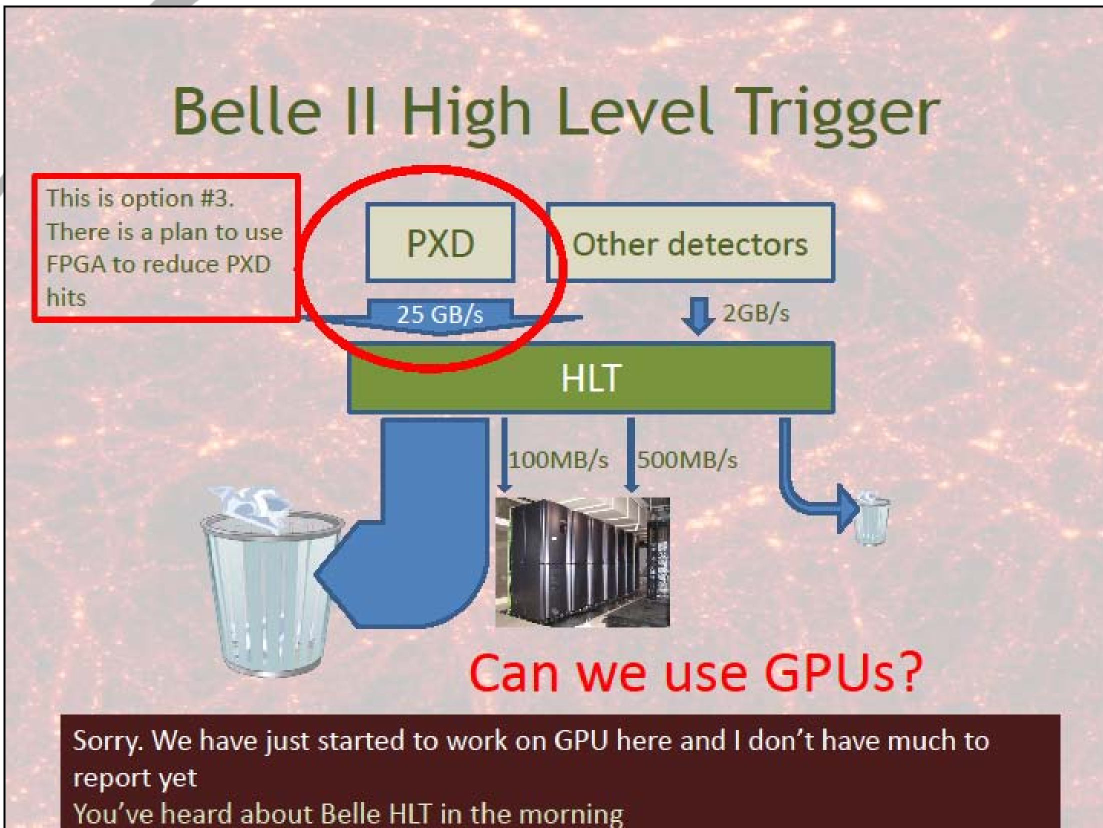
Using an Intel Core 2 Quad 2.4 GHz workstation with 2 GB of RAM and an ATI Radeon 4870 GPU with 512 MB of RAM for measurements

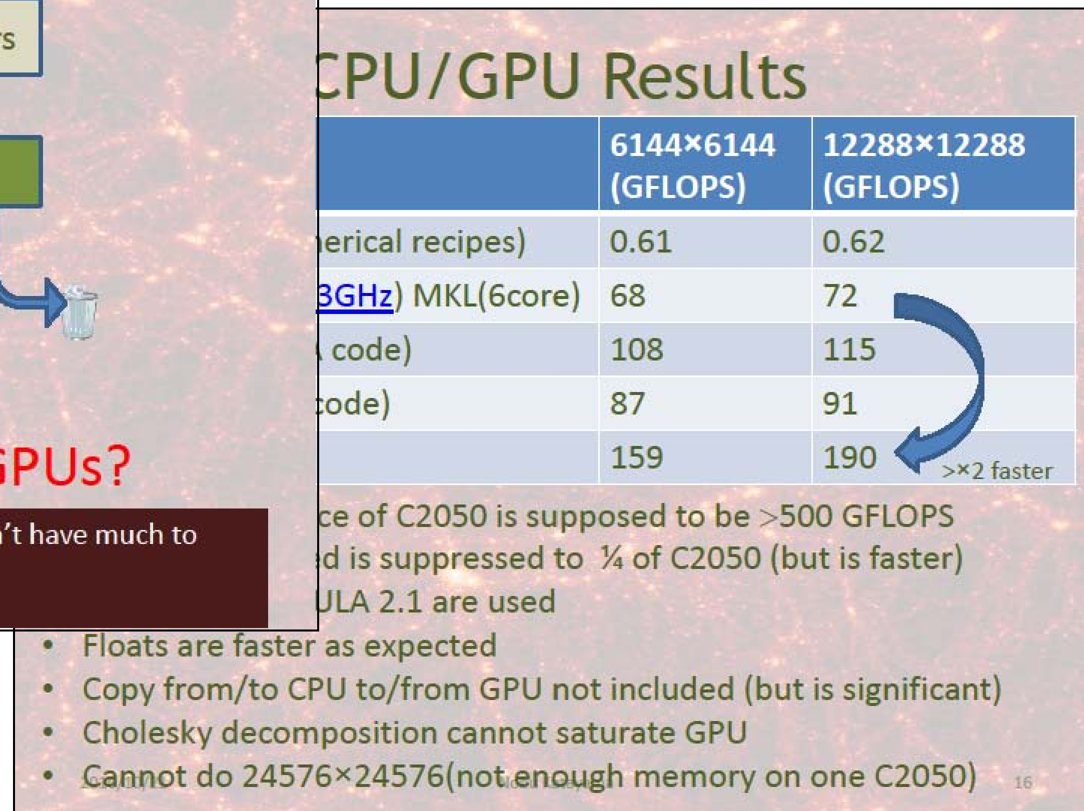


18 Partial Wave Analysis on GPUs — Niklaus Berger CHEP 2010, Taipeh

Nobuhiko Katayama/KEK

- "HLT (and other things) with GPUs"





Nobuhiko Katayama/KEK

- "HLT (and other things) with GPUs"

Inflation

- Inflation started 10^{-36} sec. after the birth
universe and lasted for 10^{-34} sec.
- During that period, the universe expande
order of e^{60}, from Plank scale to a meter
(our observable universe)
- Inflation was caused by a particle (field)
energy scale of 10^{16} GeV
- Cosmic Microwave Background Radiation
is the probe to measure its energy scale
  - Let me use the parameter "r" to represent

CPU/GPU Results

	6144×6144 (GFLOPS)	12288×12288 (GFLOPS)
CPU (i-7 920) (numerical recipes)	0.61	0.62
CPU(i-7 X980@3.33GHz) MKL(6core)	68	72
GTX480 (our CUDA code)	108	115
C2050 (our CUDA code)	87	91
C2050 (CULA)	159	190

>×2 faster

- Peak performance of C2050 is supposed to be >500 GFLOPS
- GTX480 DP speed is suppressed to ¼ of C2050 (but is faster)
- CUDA 3.1 and CULA 2.1 are used
- Floats are faster as expected
- Copy from/to CPU to/from GPU not included (but is significant)
- Cholesky decomposition cannot saturate GPU
- Cannot do 24576×24576(not enough memory on one C2050)

Andrew Washbrook/ATLAS

- "Algorithm acceleration from GPGPUs for the ATLAS upgrade"

GPU Computing | GPGPUs
Z Finder | GPU Projects at Edinburgh
Kalman Filter | Project Resources
Summary | ATLAS Trigger

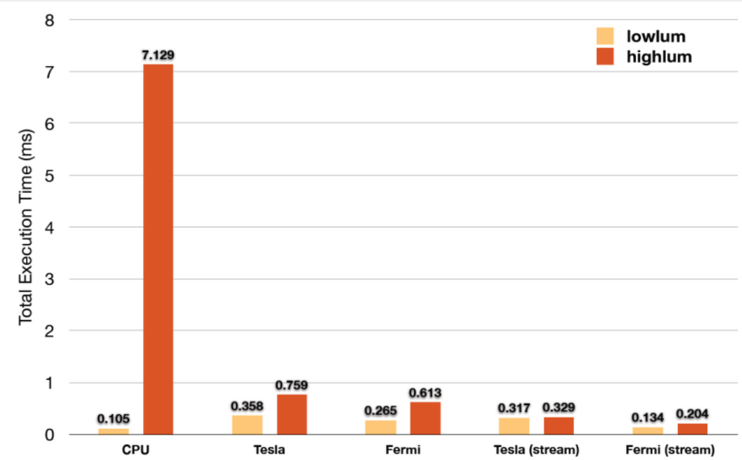
GPU Projects at Edinburgh

- Number of GPU related projects at Edinburgh over the summer:
- **Chris Jones** - *"Porting the Z finder algorithm to GPU"* (MSc in High Performance Computing)
- **Maria Rovatsou** - *"SIMT design of the High Level Trigger Kalman Fitter"* (MSc School of Informatics)
- **James Henderson** - *"An Investigation Into Particles Tracking and Simulation Algorithms using GPUs"*
- Project reports and source code available at: ATLAS Edinburgh GPU Computing

3 / 22

GPU Computing | GPU Motivation
Z Finder | Algorithm and Test case
Kalman Filter | Z Finder Kernel
Summary | Timing Results

ming Results



- Results for spacepoint pairs show up to *35x* speed-up (Fermi).
- Initial results for spacepoint *triplets* also show speed-up.

12 / 22

Alfio Lazzaro/CERN openlab

- “Maximum likelihood fits using GPUs”

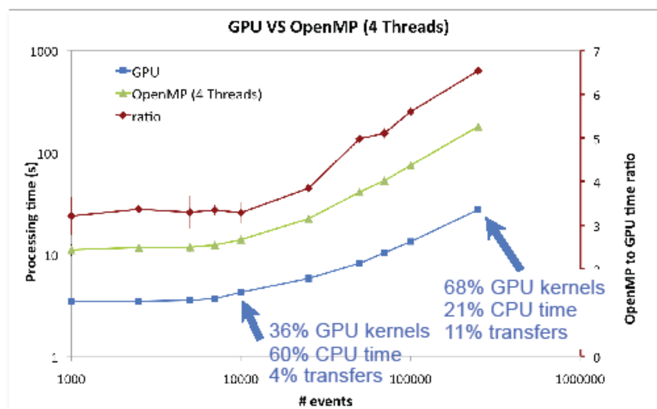
Test environment

- PCs
  - CPU: Nehalem @ 3.2GHz: 4 cores – 8 hw-threads
  - OS: SLC5 64bit - GCC 4.3.4
  - ROOT trunk (October 11th, 2010)
- 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 3.1 06/2010
  - Developer Driver 256.40
  - Power Consumption 200W
  - Price ~$340

Alfio Lazzaro (alfio.lazzaro@cern.ch)

PDF-event-base: GPU VS OpenMP

- Fair comparison
  - Same algorithm
  - Algorithm on CPU optimized and parallelized (4 threads)
  - CPU does the final sum of the *NLL* and normalization integral calculations
- Check that the results are compatible: asymmetry less than 10^{-12}



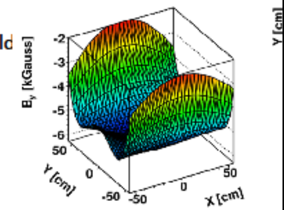
- Speed-up increases with the dimension of the sample, taking benefit from the data streaming on GPU and the integral calculation only on the CPU
  - ~3x for small samples, up to ~7x for large samples

Alfio Lazzaro (alfio.lazzaro@cern.ch) 15

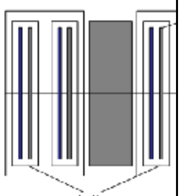
- “Fast track reconstruction of the muon system and transition radiation detector”

Optimization of the algorithm

- Minimize access to global memory
  - Approximation of the 70 MB large magnetic field map
    - 7 degree polynomial in the detector planes was proven the best



- Simplification of the detector geometry
  - Problem
    - Monte-Carlo geometry consists of 800000 nodes
    - Geometry navigation based on ROOT TGeo
    - Take into account absorbers and staggered Z positions of the stations
  - Solution
    - Create simplified geometry by converting Monte-Carlo geometry
    - Implement fast geometry navigation for the simplified geometry



- Computational optimization of the Kalman Filter
  - From **double** to **float**
  - Implicit calculation on non-trivial matrix elements
  - Loop unrolling
  - Branches (if then else ..) have been eliminated

All these steps are necessary to implement SIMD tracking

A. Lebedev, “Track reconstruction in the muon system and transition radiation detector of the CBM experiment”

Performance of the track fit **in MUCH**

Track fit quality

Residuals					Pulls				
X [cm]	Y [cm]	Tx $*10^{-3}$	Ty $*10^{-3}$	q/p $*10^{-3}$ [GeV^{-1}]	X	Y	Tx	Ty	q/p
0.38	**0.39**	**9.1**	**8.7**	**3.4**	**1.02**	**0.99**	**1.08**	**1.08**	**0.92**

Speedup of the track fitter

	Time [µs/track]	Speedup
Initial	**1200**	**-**
Optimization	**13**	**92**
SIMDization	**4.4**	**3**
Multithreading	**0.5**	**8.8**
Final	**0.5**	**2400**

Throughput: $2*10^6$ tracks/s

Computer with 2xCPUs Intel Core i7 (8 cores in total) at 2.67 GHz

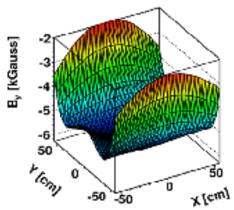
A. Lebedev, “Track reconstruction in the muon system and transition radiation detector of the CBM experiment” 11

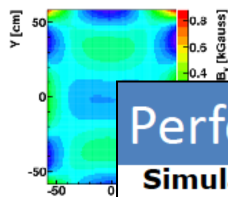
Andrey Lebedev/CBM

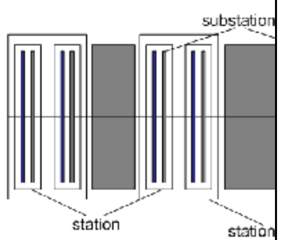
- “Fast track reconstruction of the muon system and transition radiation detector”

Optimization of the algorithm

- Minimize access to global memory
  - Approximation of the 70 MB large magnetic field map
    - 7 degree polynomial in the detector planes was proven the best
- Simplification of the detector geometry
  - Problem
    - Monte-Carlo geometry consists of 800000 nodes
    - Geometry navigation based on ROOT TGeo
    - Take into account absorbers and staggered Z positions of the stations
  - Solution
    - Create simplified geometry by converting Monte-Carlo geometry
    - Implement fast geometry navigation for the simplified geometry
- Computational optimization of the Kalman Filter
  - From **double** to **float**
  - Implicit calculation on non-trivial matrix elements
  - Loop unrolling
  - Branches (if then else ..) have been eliminated







All these steps are necessary to implement SIMD tracking

A. Lebedev, “Track reconstruction in the muon system and transition radiation detector of the CBM experiment”

Performance of the tracking in MUCH

Simulation:
- 1000 UrQMD events at 25 AGeV Au-Au collisions + 5 μ+ and 5 μ- embedded in each event

	Initial version	Parallel version
Efficiency [%]	94.7	94.0

Speedup of the track finder

	Time [ms/event]	Speedup
Initial	730	-
Optimization	7.2	101
SIMDization	4.8	1.5
Multithreading	1.5	3.3
Final	1.5	487

Computer with 2xCPUs Intel Core i7 (8 cores in total) at 2.67 GHz

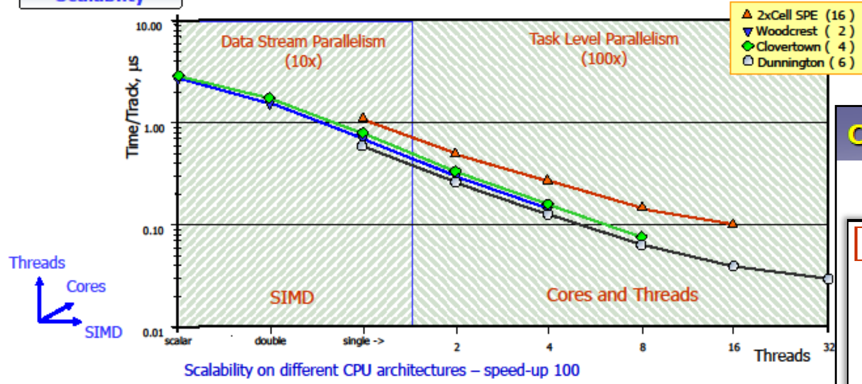
A. Lebedev, “Track reconstruction in the muon system and transition radiation detector of the CBM experiment” 12

Ivan Kisel/CBM

- "Many-core scalability of the online reconstruction in CBM"

Performance of the KF Track Fit on CPU/GPU Systems

Scalabilty



Scalability on different CPU architectures – speed-up 100

CPU

Type	Cores	Clock, GHz	Time/track, ns
Core 2	2	2.66	260
Core i7	8	2.67	52

Real-time performance on different Intel CPU platforms

GPU

NVIDIA Unit	Clock, GHz	Throughput
8800 GTS 512	1.6	13.
GTX 280	1.3	21.

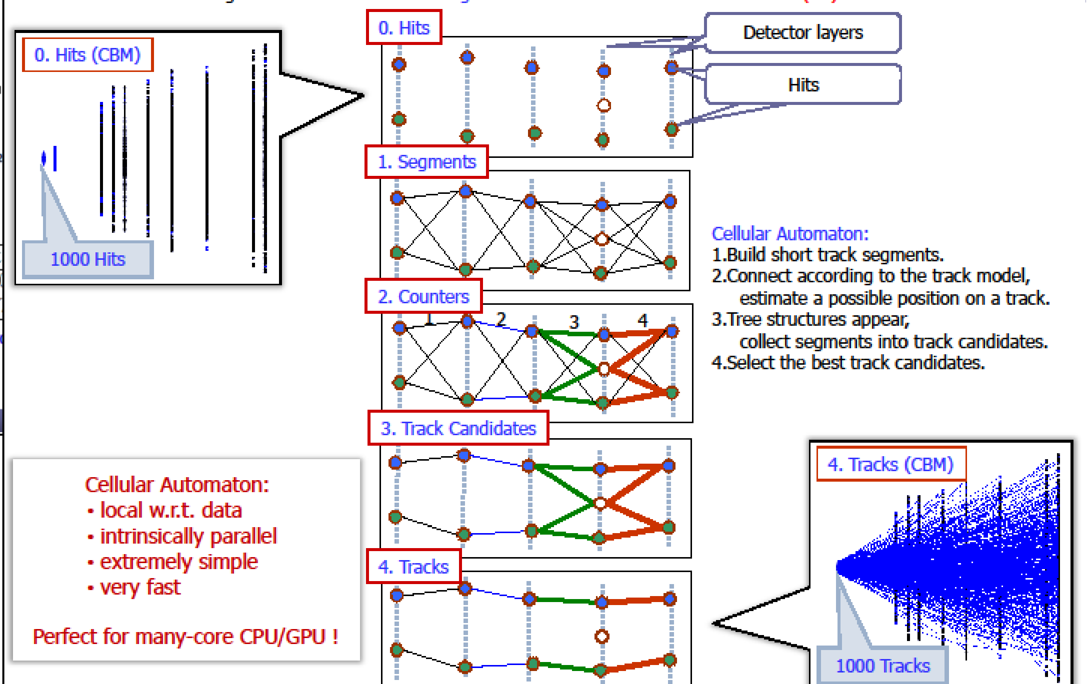
Real-time performance on NVIDIA GPU graphic car

The Kalman Filter algorithm performs at ns level

21 October 2010, CHEP-2010 Ivan Kisel, GSI

Cellular Automaton (CA) as Track Finder

Track finding: Wich hits in detector belong to the same track? – Cellular Automaton (CA)



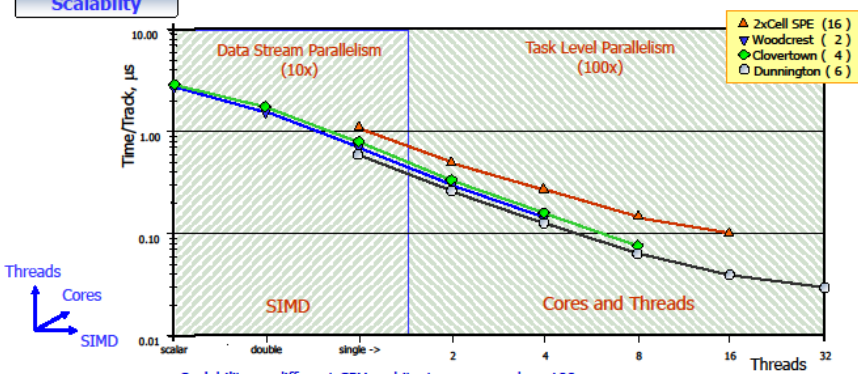
21 October 2010, CHEP-2010 Ivan Kisel, GSI 7/11

Ivan Kisel/CBM

- “Many-core scalability of the online reconstruction in CBM”

Performance of the KF Track Fit on CPU/GPU Systems

Scalabilty



Scalability on different CPU architectures – speed-up 100

CPU

Type	Cores	Clock, GHz	Time/track, ns
Core 2	2	2.66	260
Core i7	8	2.67	52

Real-time performance on different Intel CPU platforms

GPU

NVIDIA Unit	Clock, GHz	Throughput, 10
8800 GTS 512	1.6	13.0
GTX 280	1.3	21.7

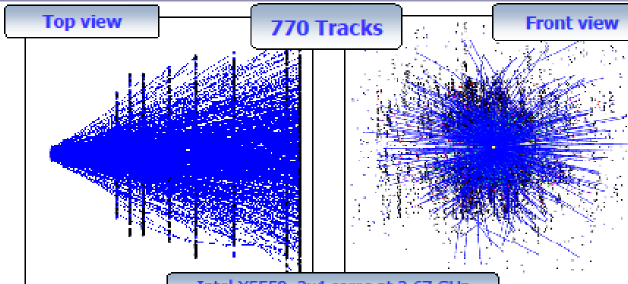
Real-time performance on NVIDIA GPU graphic cards

The Kalman Filter algorithm performs at ns level

21 October 2010, CHEP-2010 Ivan Kisel, GSI

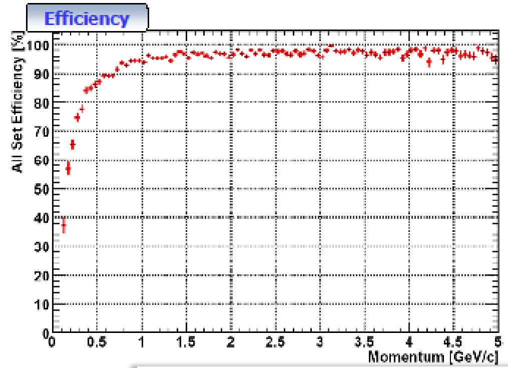
CBM Cellular Automaton Track Finder

Top view | 770 Tracks | Front view

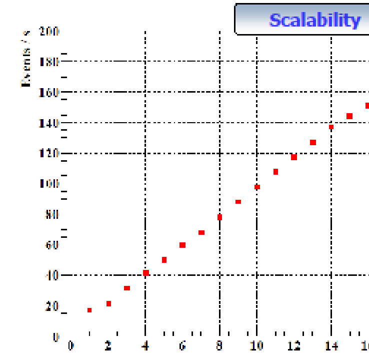


Intel X5550, 2x4 cores at 2.67 GHz

Efficiency



Scalability



Highly efficient reconstruction of 150 central collisions per second

21 October 2010, CHEP-2010 Ivan Kisel, GSI 8/11

BOF by Mohammad Al Turany and Alfio

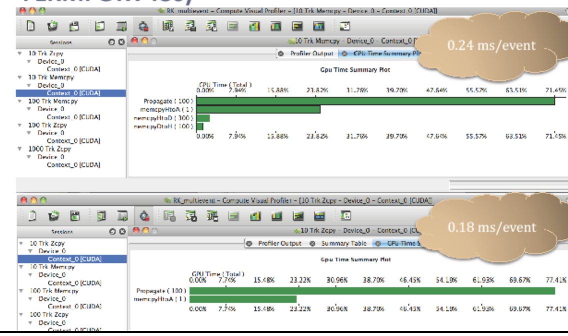
- GPUs

Cuda Toolkit

- NVCC C compiler
- CUDA FFT and BLAS libraries for the GPU
- CUDA-gdb hardware debugger
- CUDA Visual Profiler
- CUDA runtime driver (also available in the standard NVIDIA GPU driver)
- CUDA programming manual

10/21/10 GPUs: High Performance Co-Processors 14

NVIDIA Compute Visual Profiler (OpenCl/CUDA) (Runge-Kutta propagation Zcpy vs. Memcpy using FERMI GTX 480)



10/21/10 GPUs: High Performance Co-Processors

Summary

- Cuda is an easy to learn and to use tool that allows heterogeneous programming.
- Depending on the use case one can win factors in performance compared to CPU
- Texture memory can be used to solve problems that require lookup tables effectively
- Pinned Memory simplify some problems, gives also better performance.
- With Fermi we are getting towards the end of the distinction between CPUs and GPUs
  - The GPU increasingly taking on the form of a massively parallel co-processor

Tools
Jobs

10/21/10 GPUs: High Performance Co-Processors 27

Conclusions

- More and more people are working on multi-core, many-core, and accelerators
  - Especially, in the on-line domain
  - And, in new or upgraded experiments
  - Positive outcome: Code is revisited and made more “C-like”
- But, many people forget to do a “fair” comparison:
  - GPU code should expose “rich” loops for threading
    - Transfer times must be included
  - CPU code should exploit vectors + threads
- As usual it is important to perform a comprehensive calculation of
  - Throughput/W/CHF
- Expect more activity in this domain in the coming months/years
- In openlab we will continue to participate actively in realistic and relevant evaluations