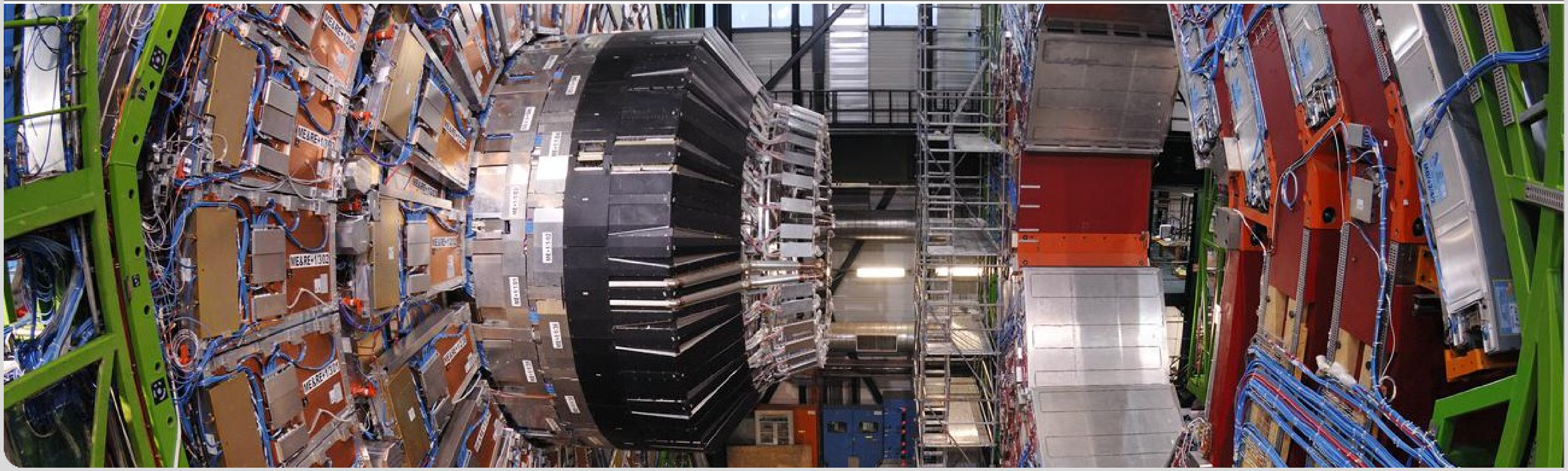
Heterogeneous Computations in the Context of the CMS Reconstruction

Annual Concurrency Forum Meeting - Fermilab, 5.2.2013

Thomas Hauth, Vincenzo Innocente, Danilo Piparo



CERN KIT Karlsruhe Institute of Technology

Hardware Acceleration: Available Options

- A diverse set of technologies exist to incorporate GPU and accelerator resources into the data processing work-flow of an application
- These vary in the level of abstraction they provide and the range of systems they support.

Classical GPU Programming

NVIDIA CUDA

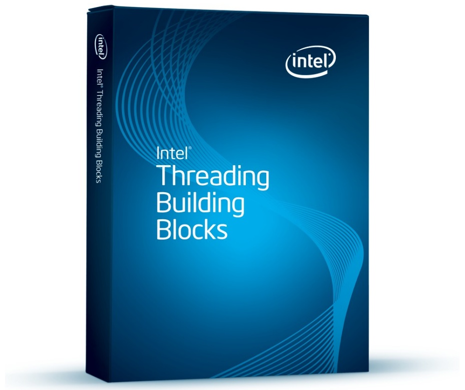
OpenCL

Annotated C/C++ code

OpenACC

NVIDIA PGI CRAY CAPS OAK RIDGE Georgia Tech CSCS

High-level C++ library



... list not complete

Hardware Acceleration in Offline Workflows

- The following talk will mainly focus on the requirements and challenges for offline event reconstruction
  - Some aspects are similar or the same of online / trigger
  - But certain requirements are very different between these areas of applications

- Modern frameworks allow for a great deal of customization in the workflow
- Offline data reconstruction can be very complex in terms of:
  - Data flow
  - Control flow

Some examples

- The CMS default reconstruction workflow consists of
  - ~ 800 distinct algorithms operating on the input data
  - ~ 2500 sets of configuration parameters
  - ~ 240 individual execution sequences

- Quiet obviously, our frameworks are to complex and flexible to be fully ported to GPU/Accelerator resource: We need to think about other solutions.

OpenCL as one possible candidate

The OpenCL standard has – in our opinion – some distinct features which make it very favorable to use in the Offline domain

- Wide hardware support by virtually all GPUs / Accelerators
- OpenCL programs can also run unchanged on the CPU, if no suitable GPU is available – *write once, run everywhere*

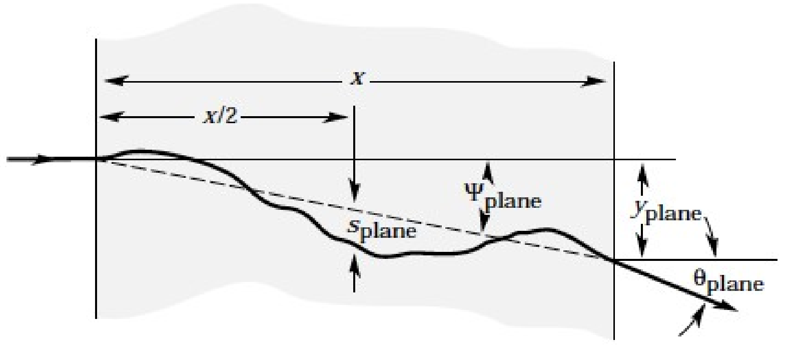
One important aspect in recent hardware development:

- CPU vendors integrate GPU-based accelerators on the die
- Not only for gamers, but also for server hardware
- AMD Heterogeneous System Architecture ("HSA")[1], formerly AMD Fusion
    - Expected to integrate the “Southern Islands” GPU family
- Intel Ivy Bridge [2][3]
    - Released early 2013, some server chips include the HD 4000 GPU
    - The Haswell architecture (probably mid 2013) is expected to have a even more powerful GPU (not confirmed)
- All of these GPUs have full OpenCL support
- If no OpenCL programs are available, they stay idle during regular x86 execution

[1] http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-system-architecture-hsa/
[2] http://en.wikipedia.org/wiki/Intel_HD_Graphics [3] http://ark.intel.com/products/65726

Use Case: Using OpenCL for CMS Algorithms

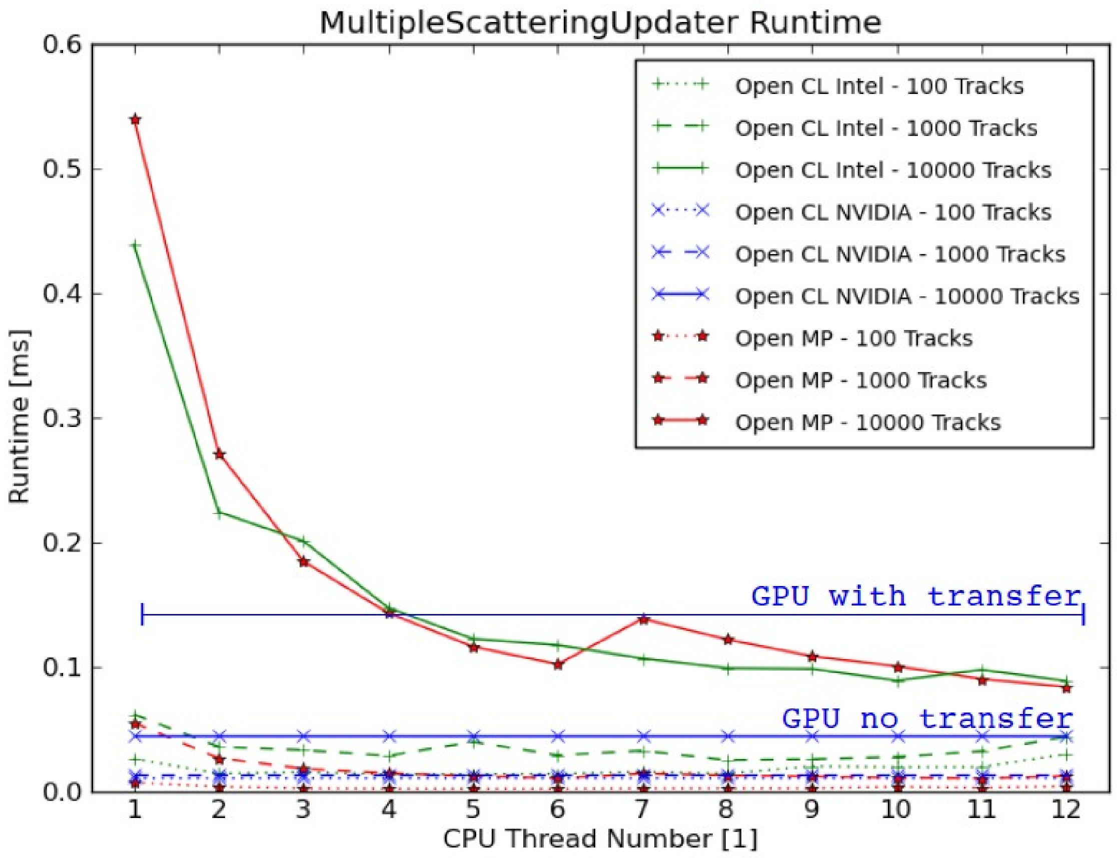
- The multiple scattering algorithm from the CMS track reconstruction was picked from the CMSSW source code[1] and run in an stand-alone application using OpenCL:
  - Calculate the maximum scattering angle of a particle passing through a material layer
  - Implementation of the Highland formula for multiple coulomb scattering
- OpenCL verision of this algorithm is not intended for insertion in CMS production code but served as a “sandbox” to test OpenCL with CMS input data in an isolated fashion
- Called several times during the reconstruction of a single track
- Useful figure: 500 to 1000 Tracks during the 2012 run period
- In terms of mathematical operations:
  - Multiplications, divisions, sums and a logarithm.
  - I/O: 4 double precision floating points in, 3 of them out.
  - About 40 lines of code, 1 branch
  - Same source is run on the CPU and the GPU
- Reference implementations were created using OpenMP and TBB to be able to compare the OpenCL results



[1] `TrackingTools/MaterialEffects/interface/MultipleScatteringUpdator.h`

OpenCL Performance Results

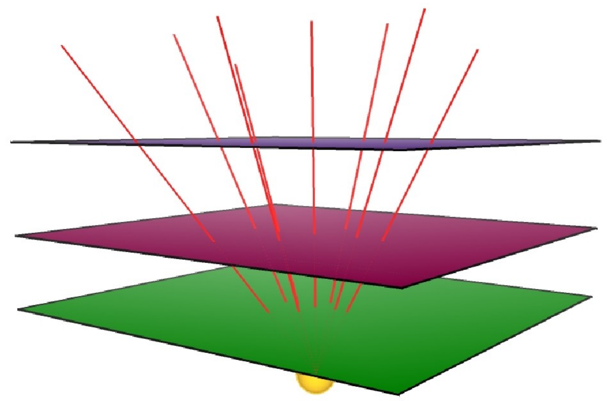
- The algorithm has been computed for various amount of input tracks (100, 1.000, 10.000)
- A varying amount of threads have been used on the CPU (Intel Core i7-3930K – 6 cores)
- The measurements on the GPU always use the full device (NVIDIA GeForce GTX 560)

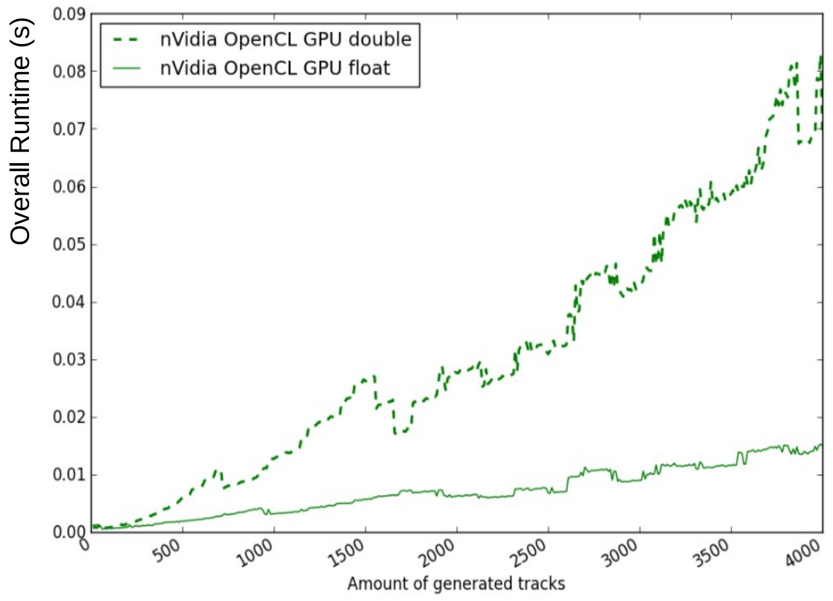


- Transferring the data from and to the graphic cards adds ~0.1ms to the runtime on the GPU
- Reuse of data transferred to the GPU is desirable
- The Intel OpenCL implementation performs as well as the classic OpenMP does
- The same OpenCL kernel code can be also run on the GPU, if available
- For less then 10.000 track input size, the OpenCL scheduling overhead must be considered
- TBB performs equally well as OpenMP (see backup)

Use Case: OpenCL for CMS Tracking

- A summer student project (finished in August 2012) had the goal to start this effort and demonstrate the feasibility of OpenCL for track seeding in CMS
- A simplified representation for the geometry of the CMS tracker in OpenCL memory was created
- Various types of track seeding algorithms were investigated on GPU and CPU
- OpenCL kernels were found to be portable between the different devices without changes
- All OpenCL platforms were able to handle the increasing amount of tracks without a significant drop in the time per track
- Building on the initial work, we are building a more complete OpenCL tracking technology demonstrator


Simplified visualization of particle tracks crossing the first three detector layer. These three hits are used for the initial seeding.



Double precision floating point on GPU: huge price to pay

Our conclusions on the OpenCL technology

Infrastructure & Library

- The promises of OpenCL are maintained: the same kernels run smoothly and without modifications on CPUs and GPUs. No-vendor lock-in!
- The specification is mature enough to base long-term software projects on it
- With the very generic abstraction approach of OpenCL some optimization possibilities might be lost
  - A classical trade-off between flexibility and performance
  - Vendor-specific, new features need some time to show up in the OpenCL standard (if at all), typical example: NVIDIA CUDA is more feature-rich

Software Development

- Some difficulties can be mitigated by providing high-level wrapper libraries for OpenCL
- Still, some concurrent expert knowledge is required to implement OpenCL algorithms
  - This is also true for any other concurrent programming system: CUDA, TBB

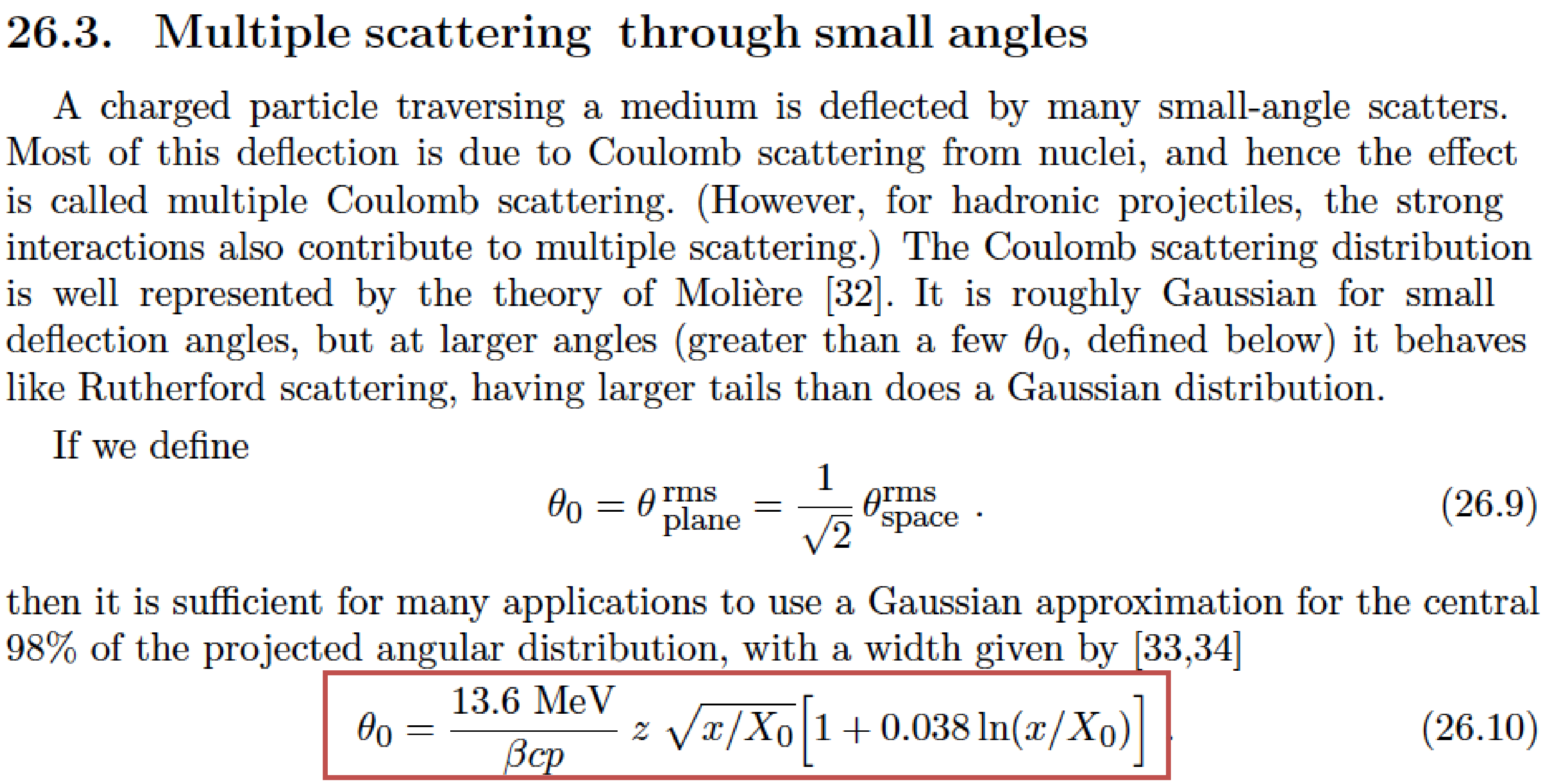
Considerations on the Framework Side

- An ideal exploitation of all available resources must be the goal
- Present HEP Frameworks incorporate many individual data processing steps
- It is not feasible to implement all of them using GPUs/Accelerators:
  - Some processing steps don't fit well with the data-parallel processing of GPUs
  - External libraries are used (Geant4, FastJet)

A hybrid approach can offer a solution
  - The data and control flow is controlled by the framework
  - Certain, selected algorithms can be implemented using OpenCL
  - They receive their input data from framework facilities and hand them back to the framework on completion
  - The framework guarantees to run the correct OpenCL programs at the right time and on the right input data
  - To improve the efficiency of GPU-offloads, it is beneficial to process the data of more than one event on the GPU (transfer of data buffers and program execution)
    - next-generation multi-threaded frameworks already support multiple events “in-flight” at the same time

BACKUP

Passage of particles through matter



26.3. Multiple scattering through small angles

A charged particle traversing a medium is deflected by many small-angle scatters. Most of this deflection is due to Coulomb scattering from nuclei, and hence the effect is called multiple Coulomb scattering. (However, for hadronic projectiles, the strong interactions also contribute to multiple scattering.) The Coulomb scattering distribution is well represented by the theory of Molière [32]. It is roughly Gaussian for small deflection angles, but at larger angles (greater than a few θ_0, defined below) it behaves like Rutherford scattering, having larger tails than does a Gaussian distribution.

If we define

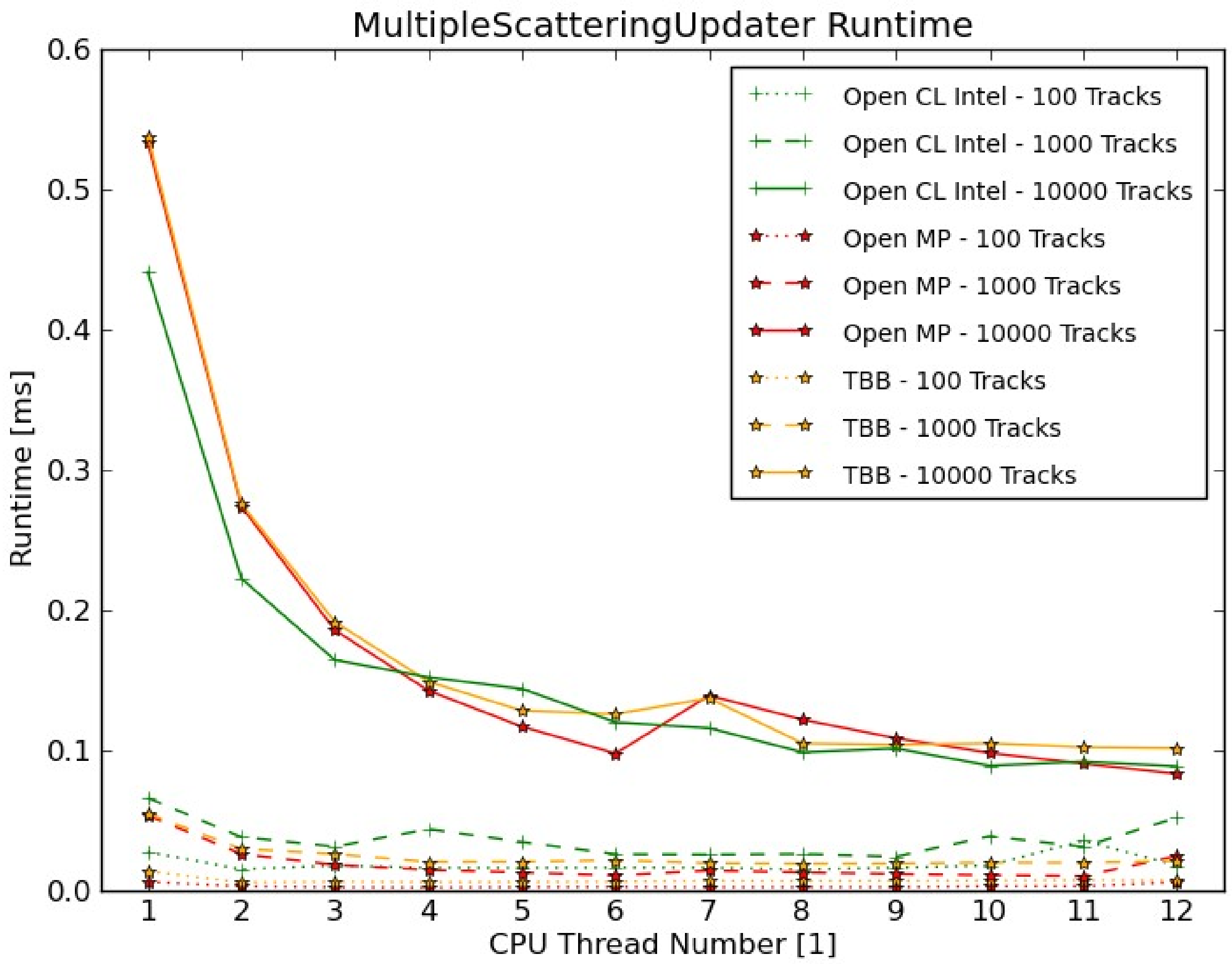
$$\theta_0 = \theta_{\text{plane}}^{\text{rms}} = \frac{1}{\sqrt{2}}\,\theta_{\text{space}}^{\text{rms}} \; . \tag{26.9}$$

then it is sufficient for many applications to use a Gaussian approximation for the central 98% of the projected angular distribution, with a width given by [33,34]

$$\boxed{\theta_0 = \frac{13.6\ \text{MeV}}{\beta c p}\, z\, \sqrt{x/X_0}\Big[1 + 0.038\ln(x/X_0)\Big]} \, . \tag{26.10}$$

Source: PDG – Review of Particle Physics, 2010

OpenCL on CPU vs. TBB vs. OpenMP


MultipleScatteringUpdater Runtime
Runtime [ms]
CPU Thread Number [1]
Open CL Intel - 100 Tracks
Open CL Intel - 1000 Tracks
Open CL Intel - 10000 Tracks
Open MP - 100 Tracks
Open MP - 1000 Tracks
Open MP - 10000 Tracks
TBB - 100 Tracks
TBB - 1000 Tracks
TBB - 10000 Tracks