Geant4 Particle Transportation on GPU

Philippe Canal, Daniel Elvira, Soon Yung Jun
James Kowalkowski, Marc Paterno, Panagiotis Spentzouris
Fermilab

Dongwook Jang
Carnegie Mellon University

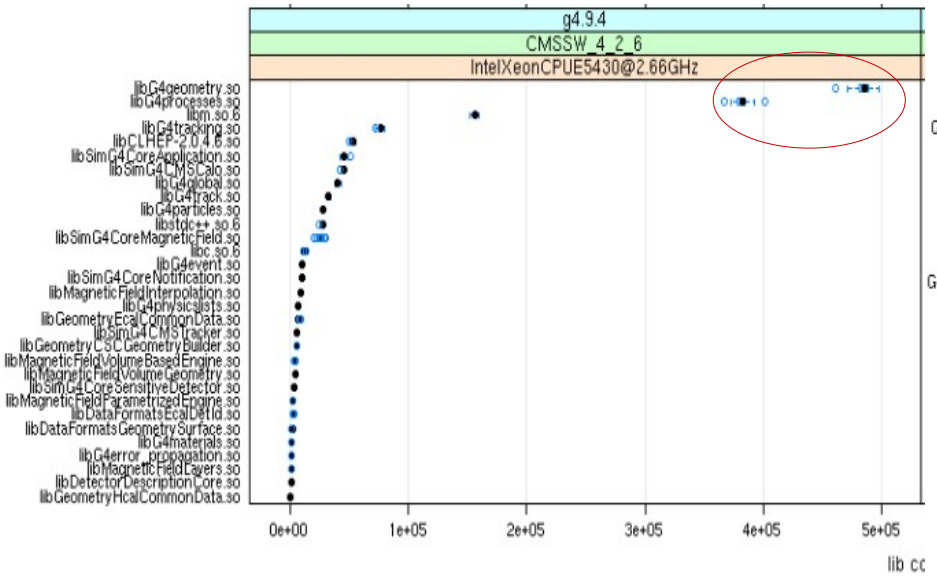
Concurrent Programming Models and Frameworks
May 9, 2012

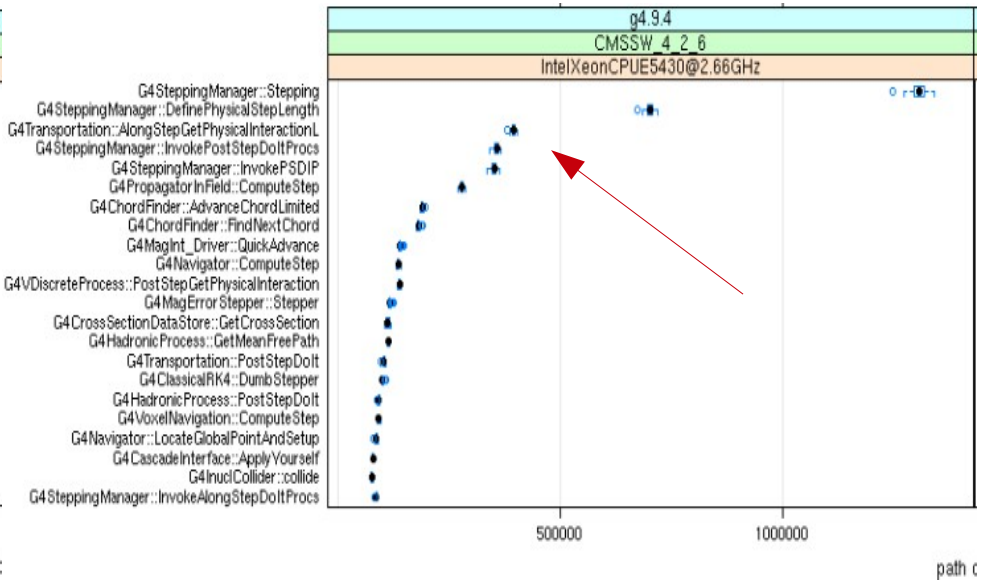
Outline

- Introduction: particle transportation
- Hardware: host and device
- Software: device codes and interfaces
- Performance: CPU/GPU processing time
- Conclusions: lessons and outlooks

Introduction

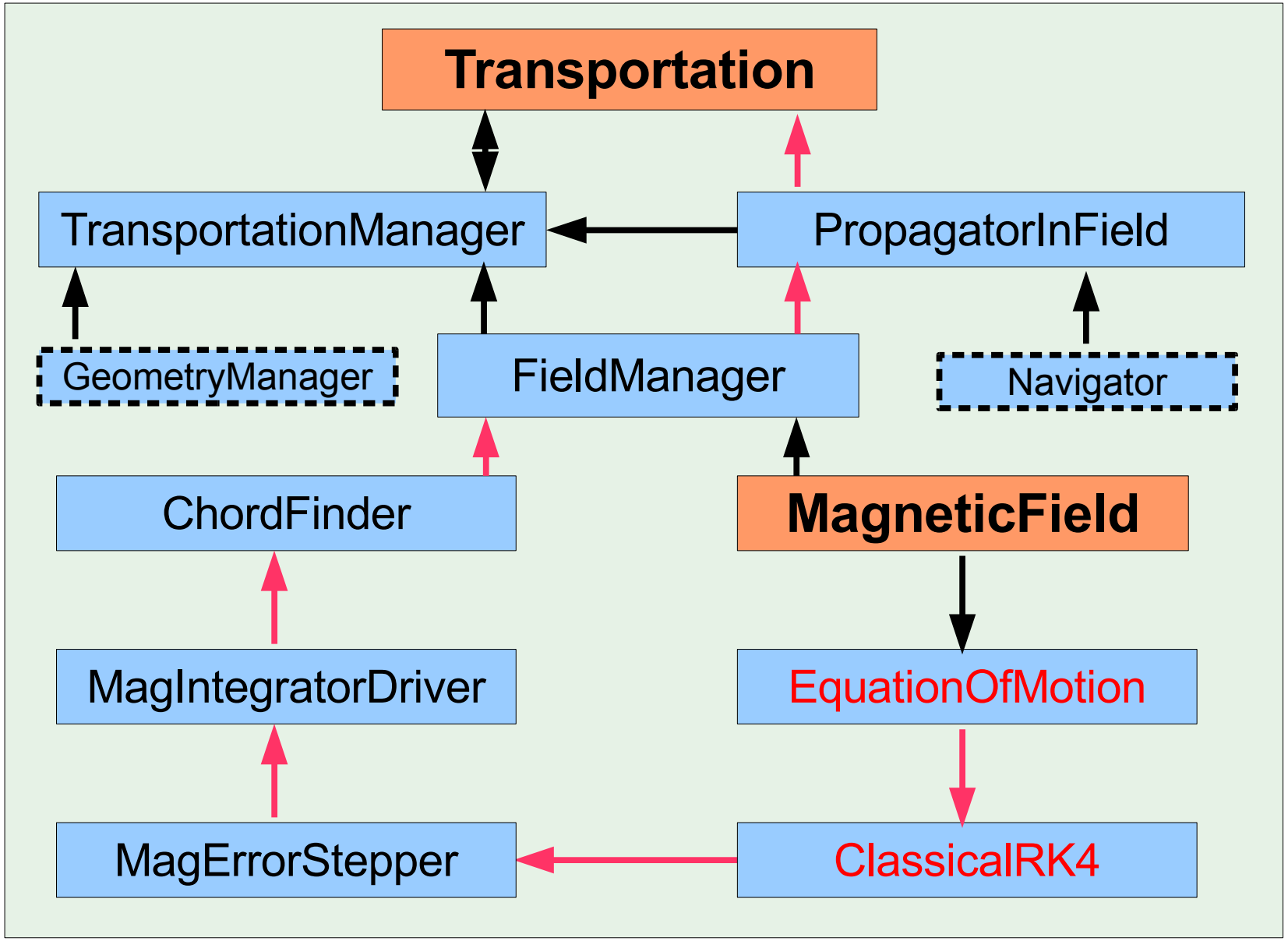
- How can we use many-core for Geant4?
- Geant4 performance studies with the CMS detector





  - geometry and processes are top libraries in lib count
  - one of hot spots of processes is transportation
- Concurrent particle transportation engine
  - study particle transportation on GPGPU

G4Transporation::AlongStepGPIL for Charged Particles in B-Field



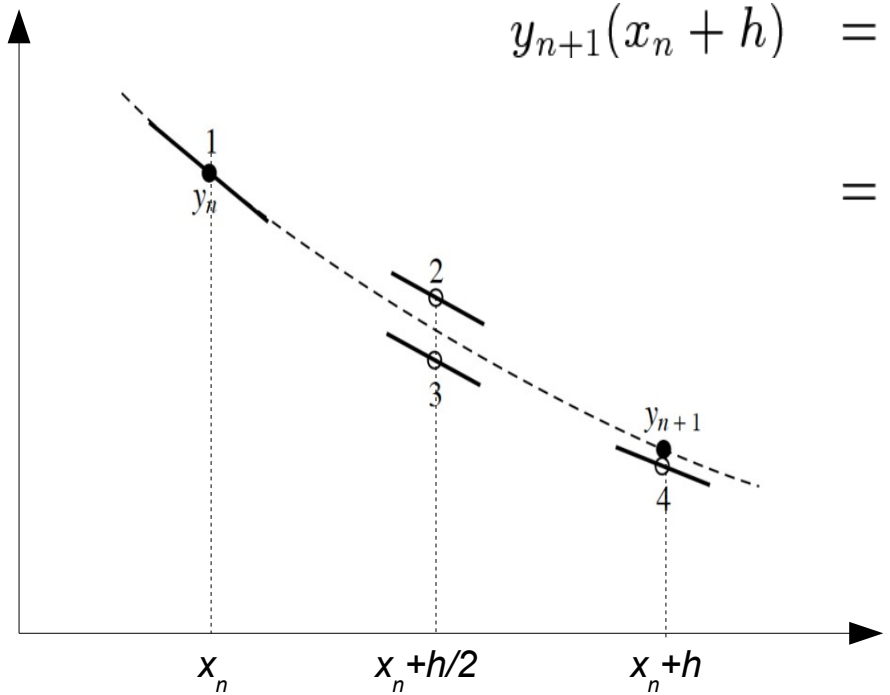
Equation of Motion and Runge-Kutta Method

- Trajectories: equation of motion in a magnetic field

$$\frac{d^2\vec{x}}{ds^2} = \frac{q}{p}\frac{d\vec{x}}{ds} \times \vec{B}(\vec{x}) \quad \rightarrow \quad \frac{dy}{ds} = f(x,y), \qquad y(x_0) = y_0$$

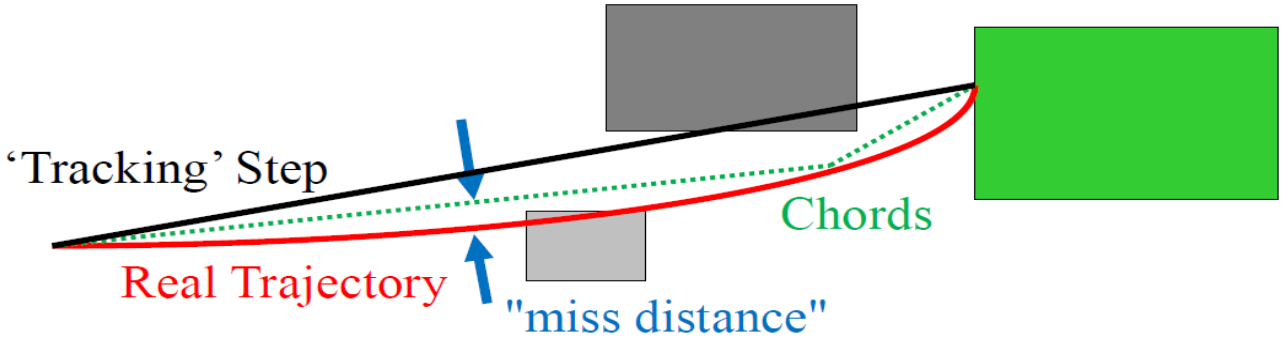
- 4-th order Runge-Kutta (RK4): 4 evaluations of f(x,y)

$$\begin{aligned} y_{n+1}(x_n + h) &= y_n + hf(x_n, y_n) \\ &= y_n + \frac{h}{6}\sum_{i=1}^{4}[f(x_n + \alpha_i, y_n + \beta_i)] + \mathcal{O}(h^5) \end{aligned}$$

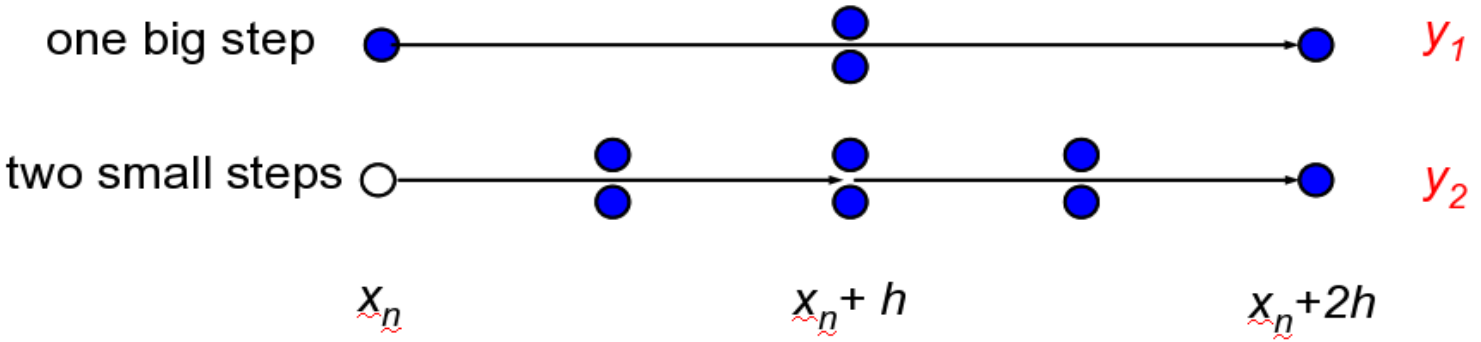


Runga-Kutta Driver with Adaptive Step Size

- Quick advance: miss distance < dmax



- Accurate advance: truncation errors of step doubling in RK4: difference between one big step and two small steps - 11 evaluations of rhs of the equation of motion



$\Delta = y_2 - y_1$ = truncation error

Problem Definition

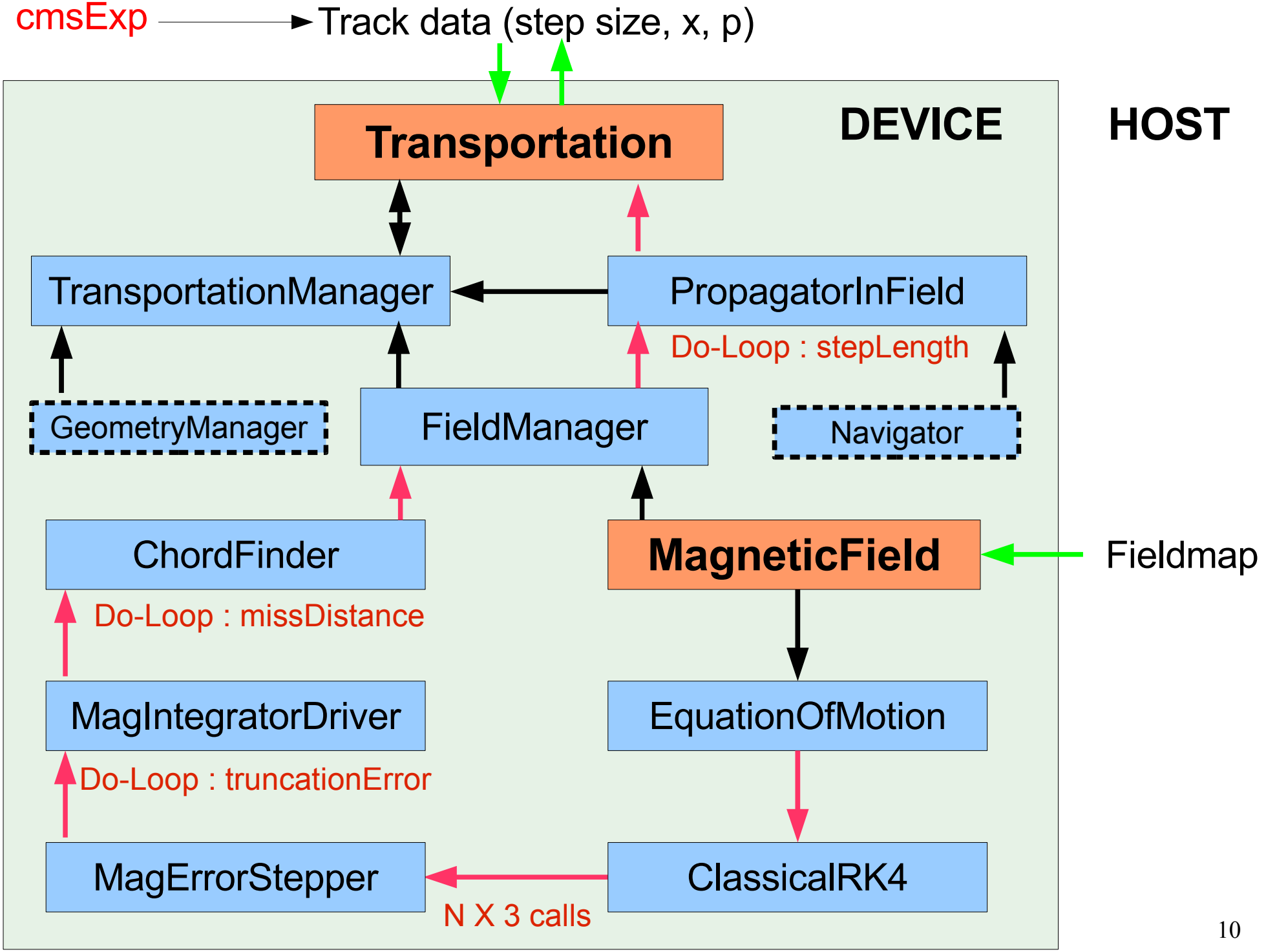
- Isolate key components of Geant4 particle transportation
  - evaluation of magnetic field (B) values
  - rhs of the equation of motion in a given B
  - evaluation of the 4^{th} order Runge-Kutta (RK4)
- Measure performance with the Runga-Kutta driver for adaptive step size control
- Test Geant4 transportation with realistic data
  - prepare bundles of tracks from simulated events
  - measure processing times for AlongStepGPIL on CPU and GPU

Hardware: Host and Device

- Host: AMD Opteron Process 6136
  - CPU: 2.4 GHz, Processors: 32 cores
  - L1/L2/L3 Cache Size: 128/512/12288 (KB)
  - L3 Cache speed: 2400 MHz
- Device: NVIDIA Tesla M2070
  - GPU clock speed: 1.15 GHz
  - 14 Multiprocessors x 32 CUDA Cores: 448 CUDA cores
  - Memory: global 5.4 GB, constant 65 KB, shared 50KB
  - L2 Cache size: 786 KB
  - Maximum thread per block: 1024
  - Warp size: 32
  - CUDA Capability Major/Minor: 2.0

Software: Interface and Device Codes

- An experimental software environment: cmsExp
  - CMS geometry (GDML) and magnetic field map (2-dim grid of volume based field extracted from CMSSW)
  - Geant4 application with an interface to device codes or a standalone framework
- Device codes I
  - literal translation of Geant4 C++ classes to C structures
  - use same implementation for both __host__ and __device__
  - input data to device memory: magnetic feld map and bundles of secondary tracks produced by cmsExp
- Device codes II optimized for GPU
  - 4^{th} order Runge-Kutta, field map with texture memory and etc.

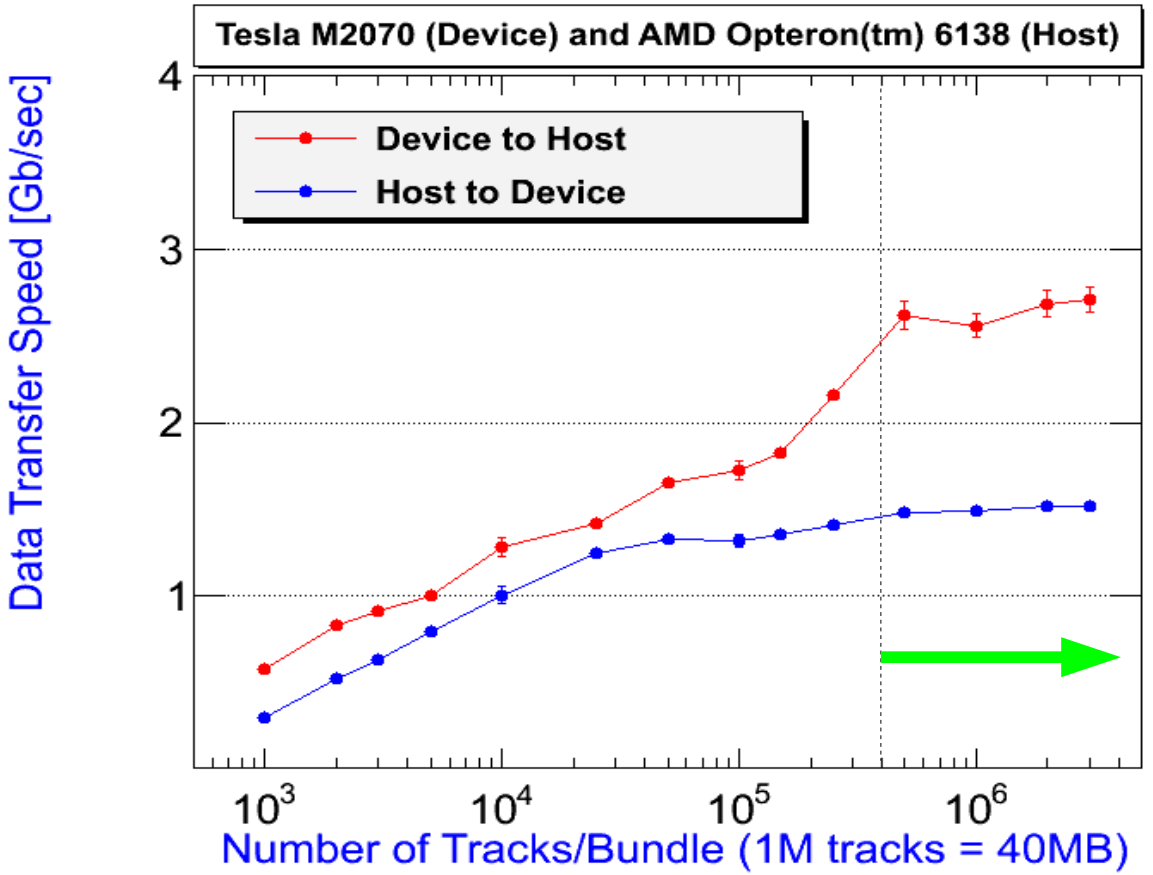

cmsExp
Track data (step size, x, p)
DEVICE
HOST
Transportation
TransportationManager
PropagatorInField
Do-Loop : stepLength
GeometryManager
FieldManager
Navigator
ChordFinder
MagneticField
Fieldmap
Do-Loop : missDistance
MagIntegratorDriver
EquationOfMotion
Do-Loop : truncationError
MagErrorStepper
ClassicalRK4
N X 3 calls

Performance Measure

- Performance measurements in execution time
  - 1 CPU vs. 448 GPU cores
  - cuda event timer (cudaEventElapsedTime)
  - GPU time = kernel execution + data transfer
  - default kernel: blocks=32, threads=128
  - default step size = 1cm
  - default size of tracks/bundle = 100K tracks
  - errors: RMS of measurements with 100 events

Performance: Data Transfer Rate

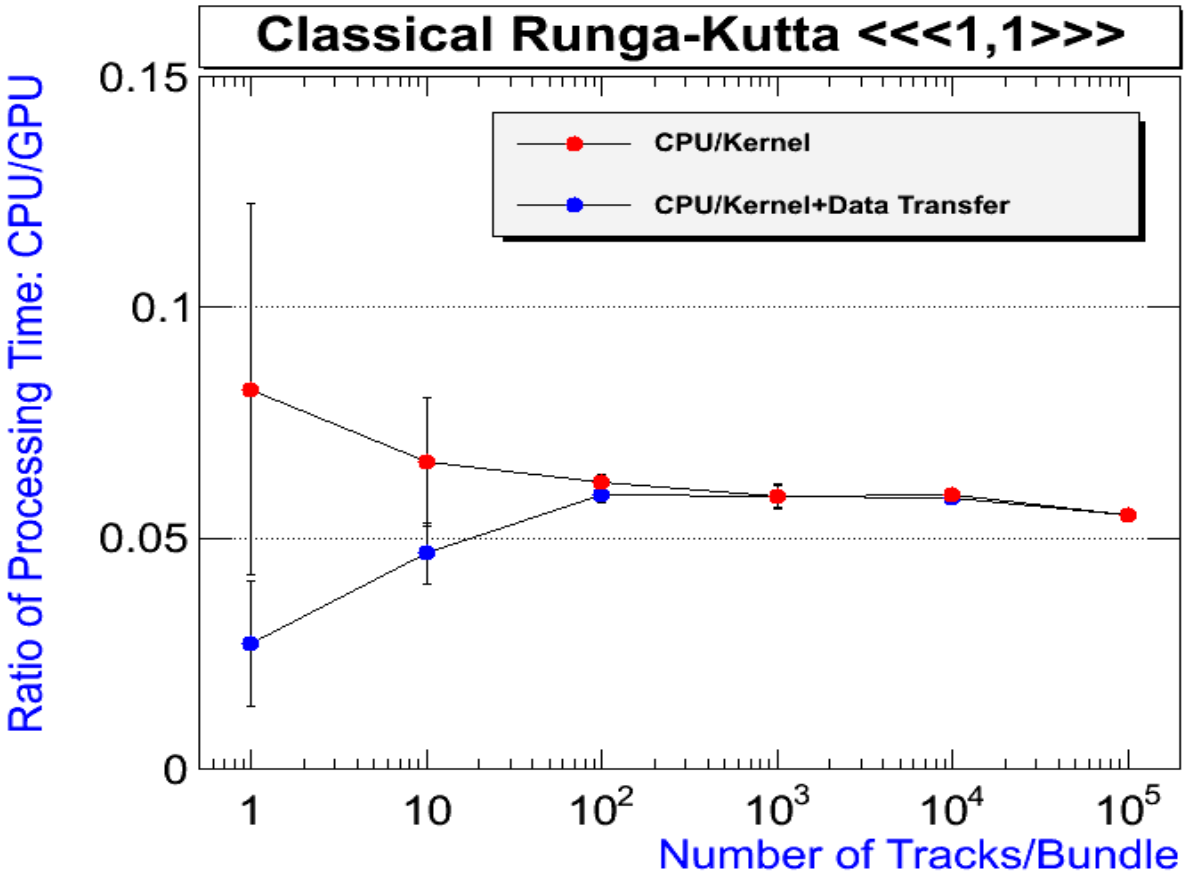
- Data transfer speed for track bundles between host and device



- Minimize data transfer between host and device
  - bandwidth device-device is $O(10^2)$ (GB/sec)
  - one large transfer is better than many small transfers

Baseline Performance: Single Thread GPU

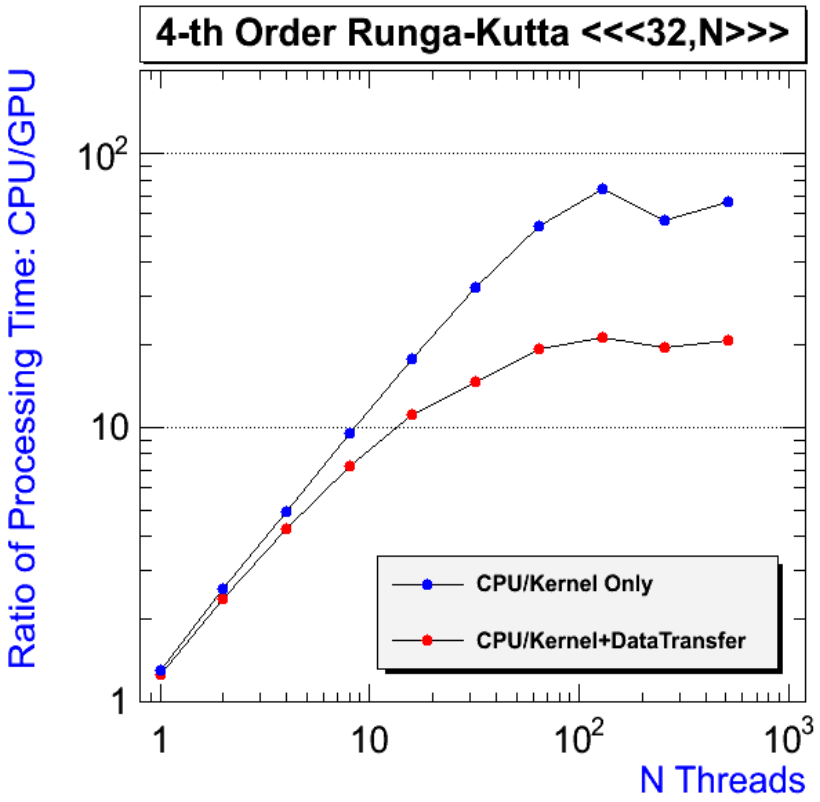
- <<<BLOCK=1,THREAD=1>>> for 4th order Runge-Kutta

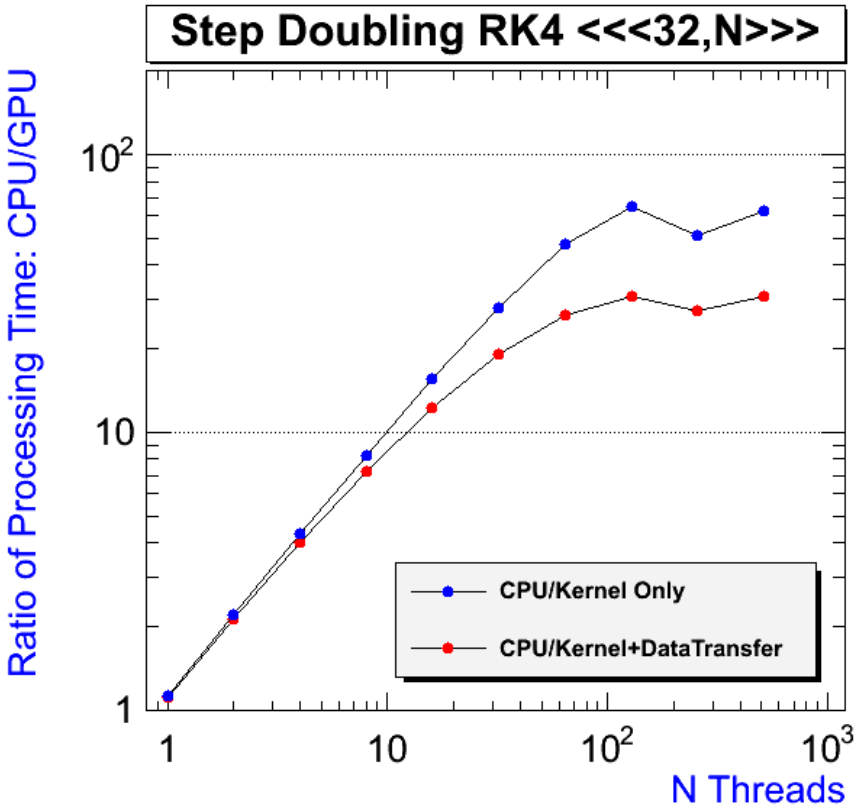


- Bottom line: 1 GPU core ≈ (1/18) CPU
  - clock speed (½), floating point calculation (¼), and etc.

Performance: Kernel with 32 Blocks

- RK4: Time(Kernel only) vs. Time(Kernel+data transfer)

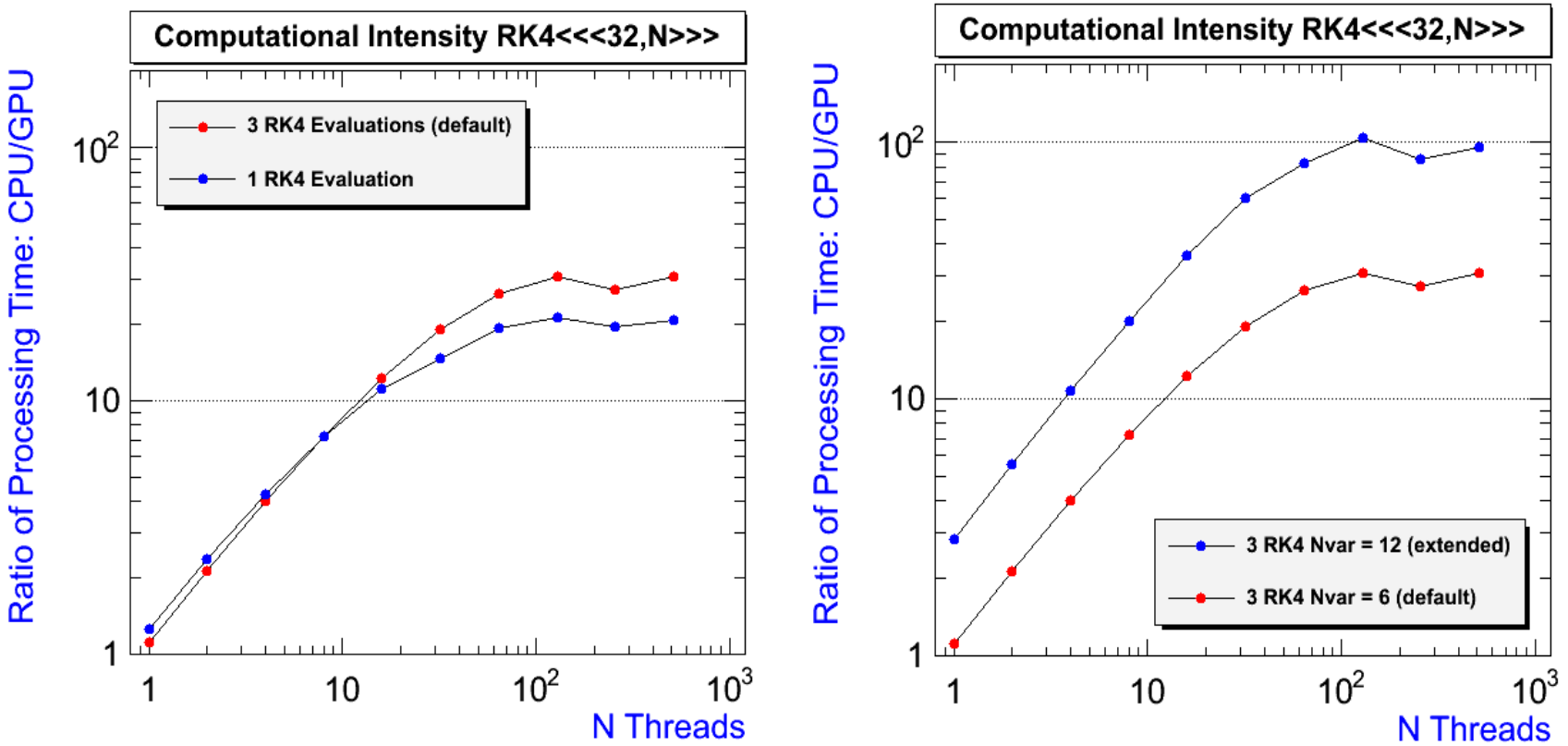




- Optimize kernel execution
  - overall (kernel+data)/kernel ~3 for RK4 and ~2 Adaptive RK4
  - minimize data transfer between host and device

Performance: Computational Intensity

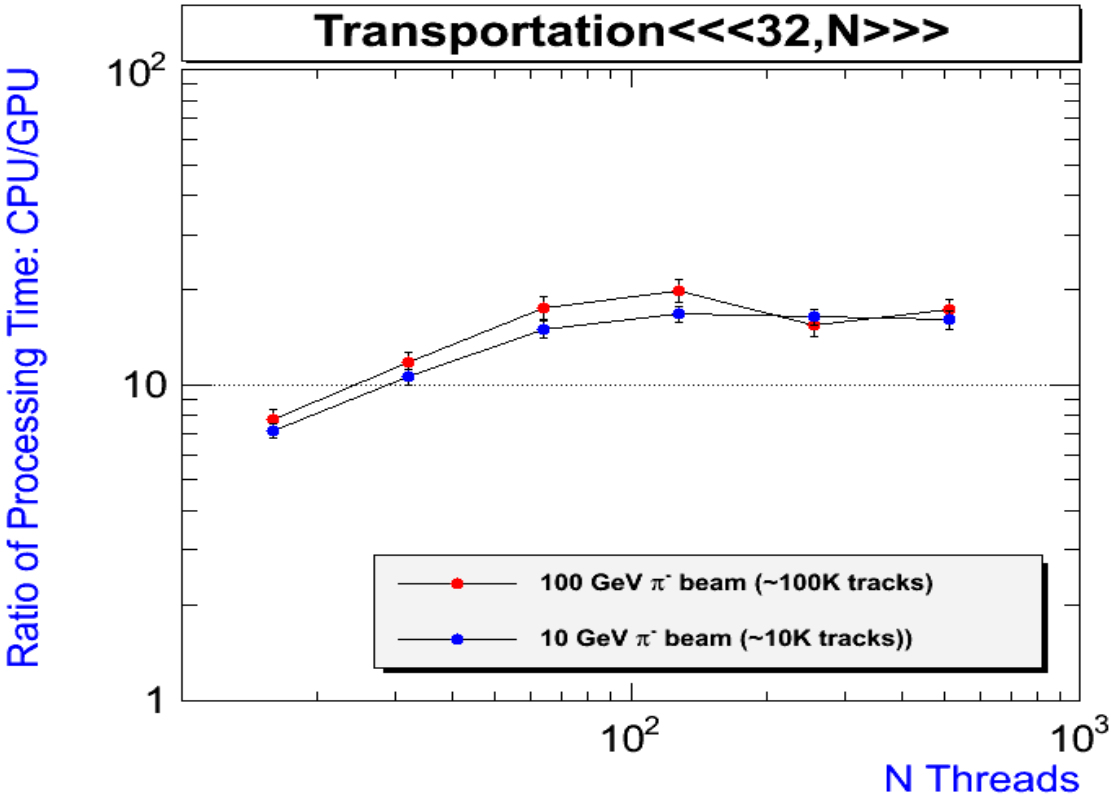
- Number of RK4 evaluations and number of variables



- Optimize arithmetic with computational intensity
  - do more arithmetic calculations on GPU
  - maximize independent parallelism (more for-loops)

Performance: Realistic Data from cmsExp

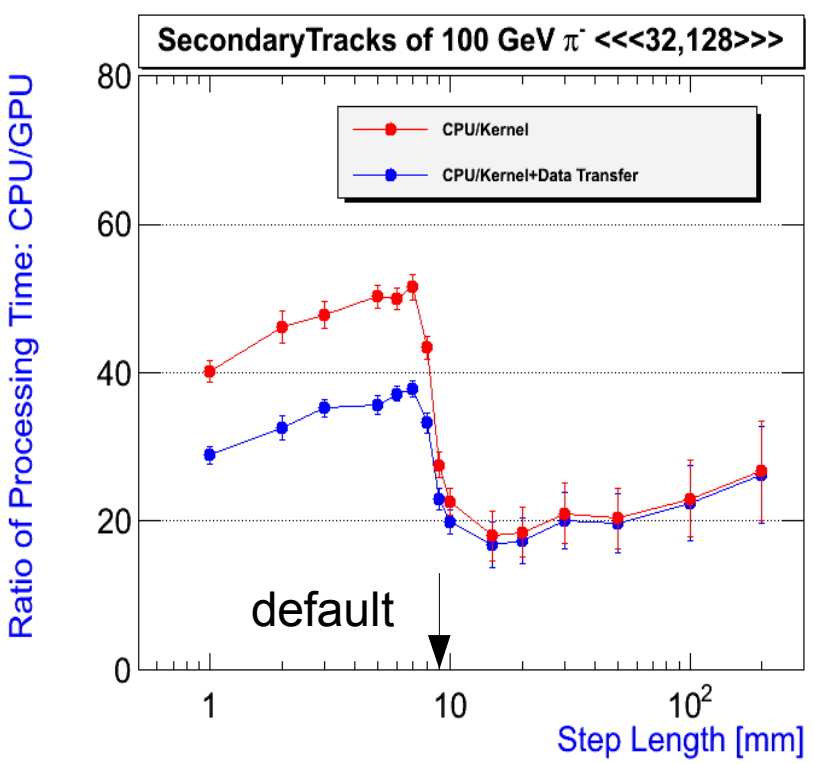
- CPU/GPU for the first step transportation for secondary particles produced by 10 GeV pions and 100 GeV pions
- Full chain of transportation with a step length = 1cm

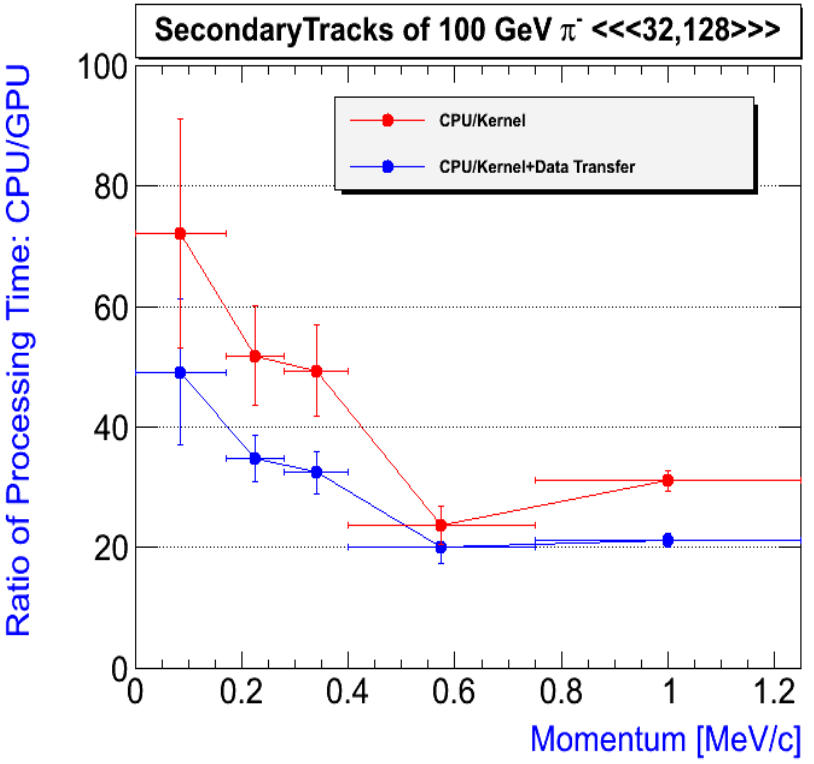


- Need additional arithmetic logistics to improve the gain

Dependence: Momentum and Step Length

- CPU/GPU for the first step of secondary tracks

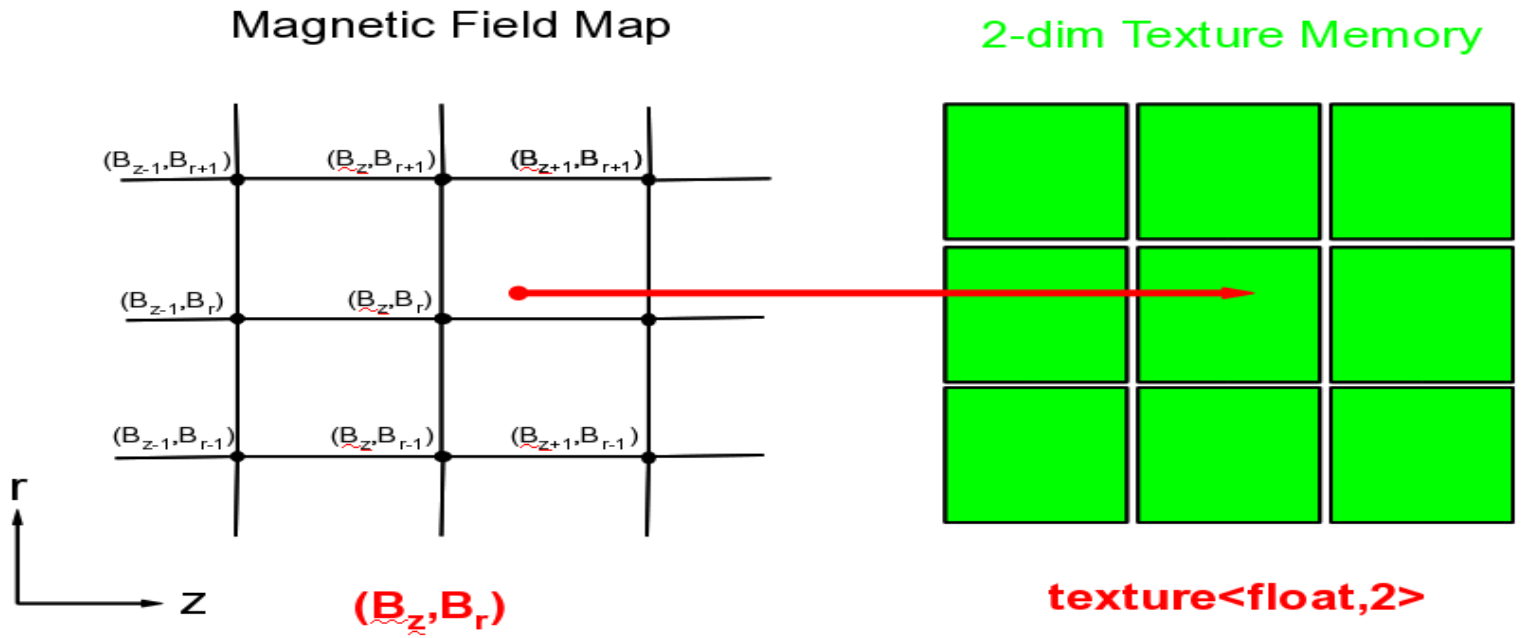




- Optimize calculation uniformity
  - Keep GPU multiprocessors equally busy
  - group tracks with same number of RK4 evaluations as possible

Magnetic Field Access : Texture memory

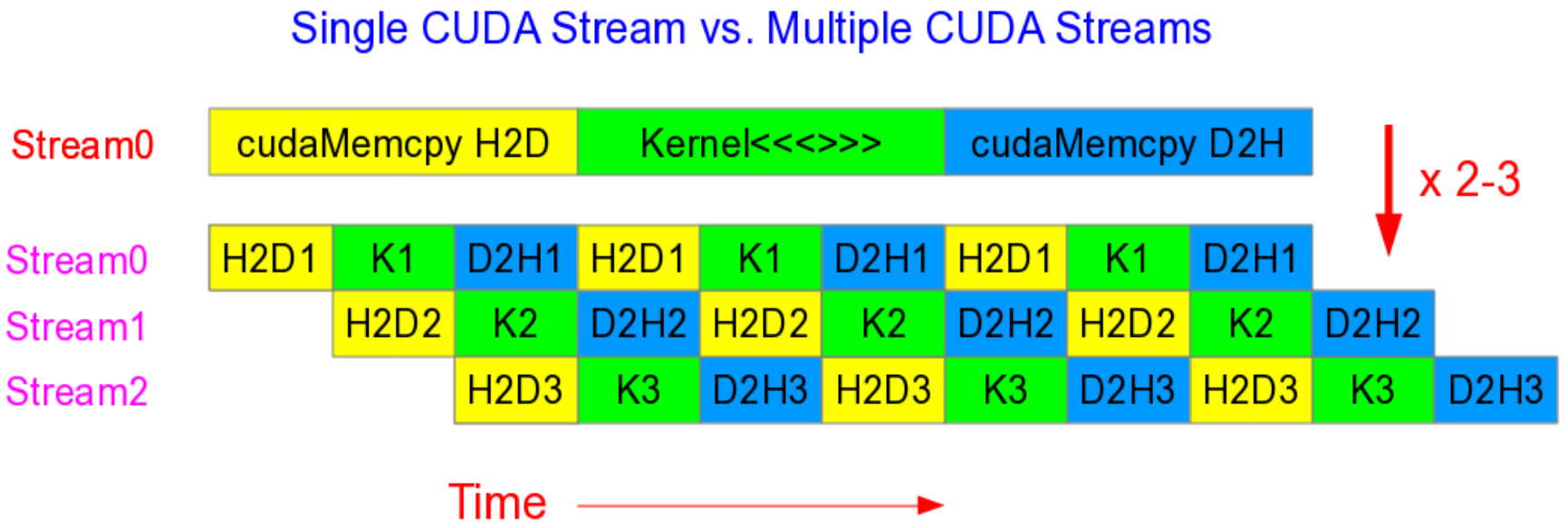
- Texture memory is cached on chip and designed for memory access with spatial locality
- Magnetic field map is a typically 2(3)-dimensional grid



- Texture interpolation twice as fast as the explicit interpolation for random access
- No noticeable difference in real data for 3 evaluations of RK4 w/wo using the texture - input data are ordered

Concurrent Kernel/Stream

- Multiple CUDA streams provide the task parallelism (kernel execution and memory copies simultaneously)



- Using multiple CUDA streams for Runga-Kutta diriver
  - no significant gain observed: balance work load evenly
  - Callgrind and IgProf profiling shows that there is only 40% more work to be gained without any geometry
  - add more calculations on device

Conclusion I

- A core part of Geant4 particle transportation has been tested on GPU
  - ratio of processing time for CPU/GPU ~ 20 with realistic data using 448 cuda cores
  - Identified key factors to maximize the GPU's ALU capacities
- Lessens learned
  - increase computational intensity on GPU
  - look for other transportation algorithms suitable for uniformity of calculations
  - organize input data for optimal efficiencies of kernel executions and data transfers

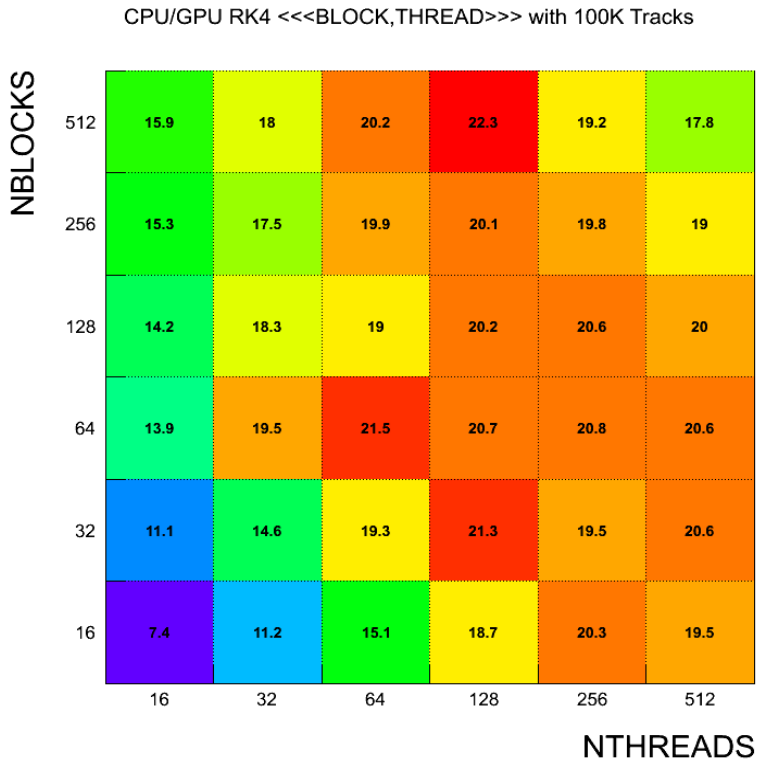
Conclusion II: Outlooks

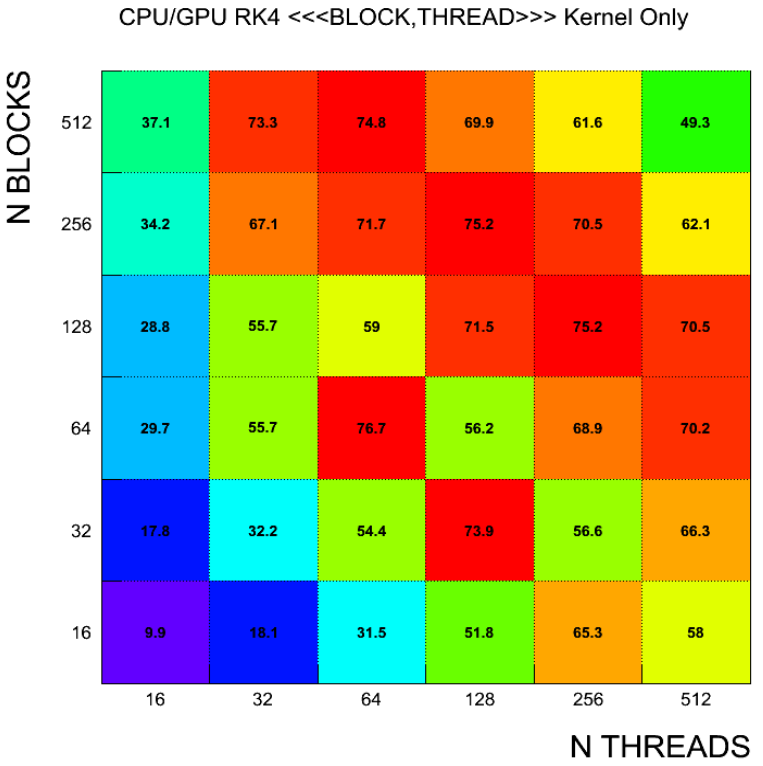
- Adding geometry on device
  - a simple detector (something like CMS crystals)
  - generalize transportation including photons and intersection with geometry
- Develop device codes for EM physics
  - multiple stepping on device to increase computations
  - generalize transportation including post step actions and pipelines for handling hits and secondaries
- Optimize GPU resources
  - more tests for multiple CUDA streams (concurrent kernel execution and copying data up/down to GPU)

Back-up Slides

Performance: CPU/GPU

- RK4: Kernel+Data Transfer vs. Kernel Only

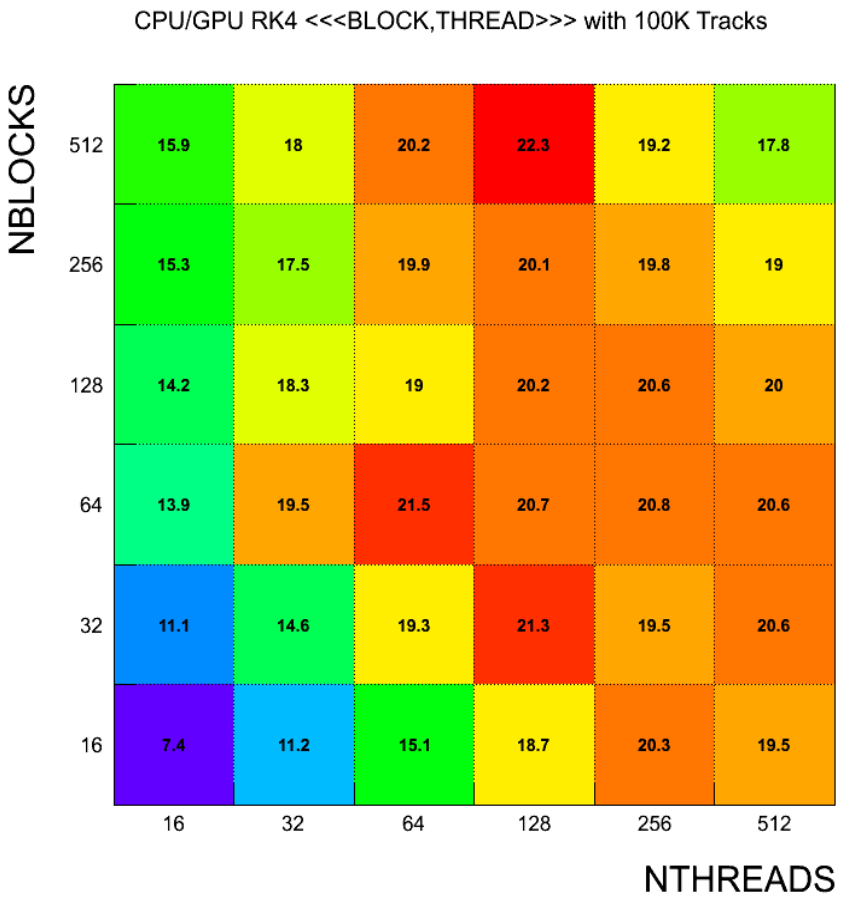


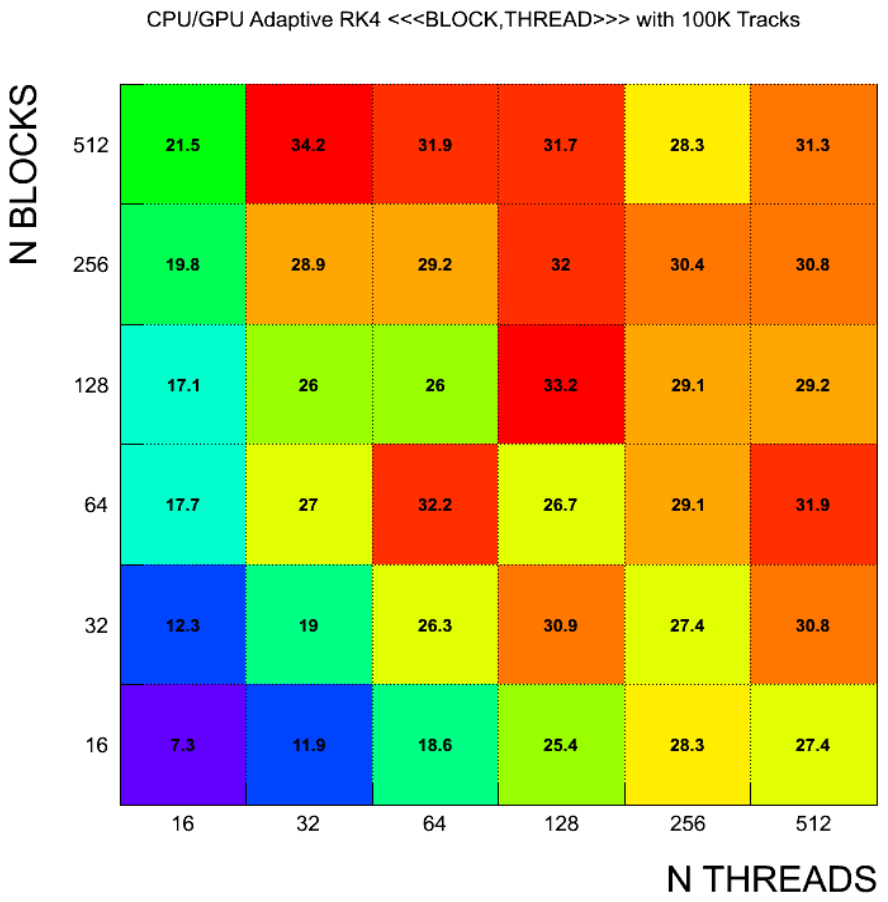


- Optimize Kernel Execution
  - Overall (Kernel+data)/Kernel = 3 for RK4
  - Minimize data transfer between host and device

Performance: Computational Density

- Number of RK evaluations: 1-RK4 vs. 3-RK4

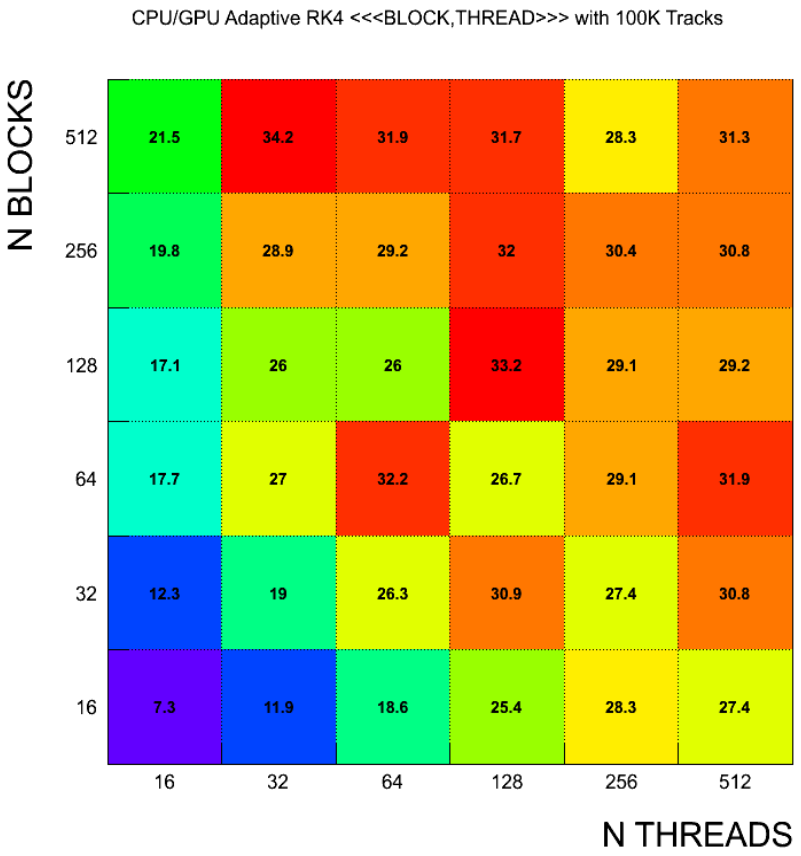


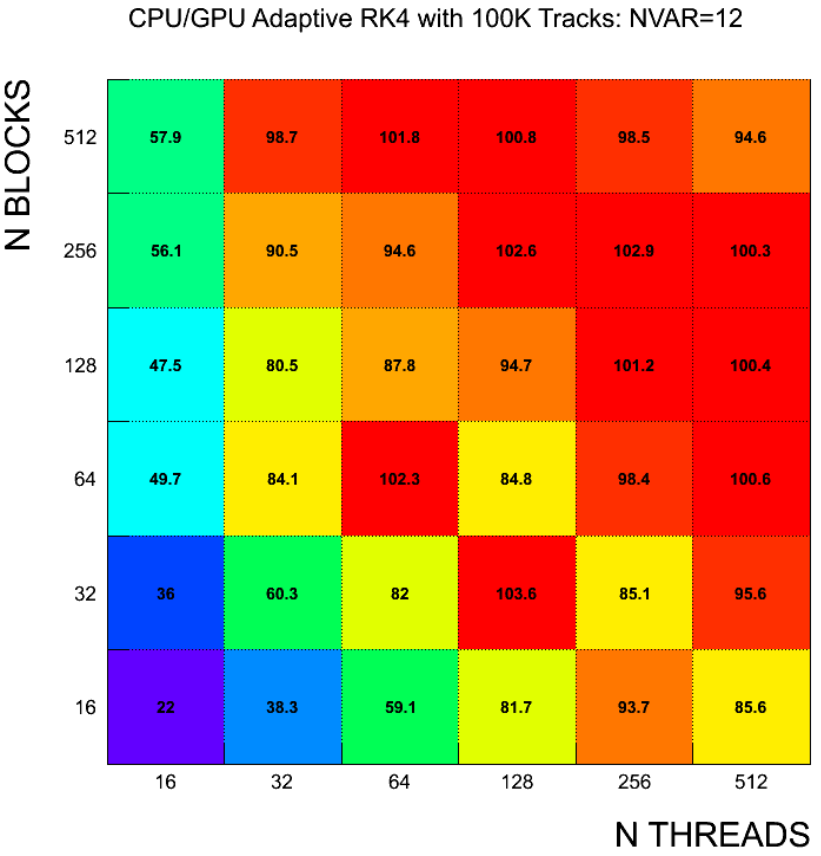


(Left) one RK4 evaluation (Right) three-RK4 evaluations

Performance: Computational

- Number of variables in equation of motion: 6 variables (default) vs. 12 (extended) for adaptive RK4.





- (Left) 6 variables (x,p) (Right) 12 variables (x,p,t,s)

Performance: Simulation Data with cmsExp

- CPU/GPU for the first step transportation for secondary particles produced by 10 GeV pions and 100 GeV pions
- Full chain of transportation with a step length = 1cm