Prototyping Geant4 on GPU

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

Dongwook Jang
Carnegie Mellon University

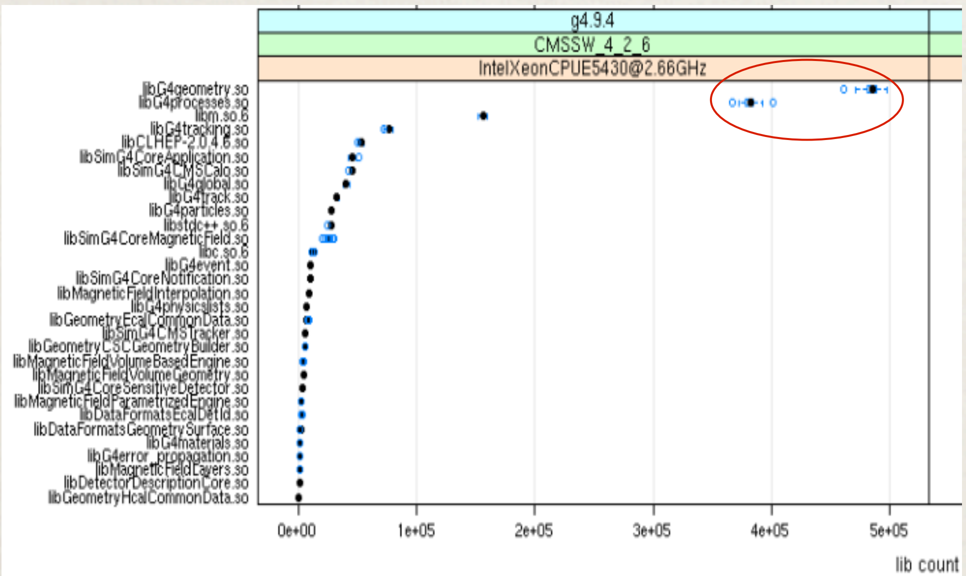
July 11, 2012

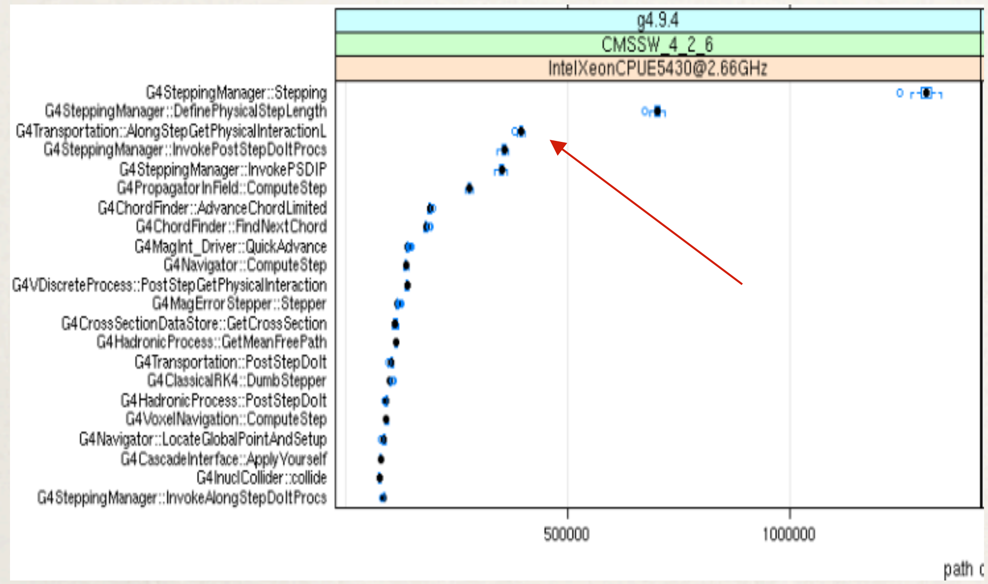
Outline

* **Introduction**: Geant4 particle transportation
* **Hardware**: host and device
* **Software**: device codes and interfaces
* **Performance**: CPU/GPU time measurement
* **Conclusions**: lessons and outlooks

Introduction

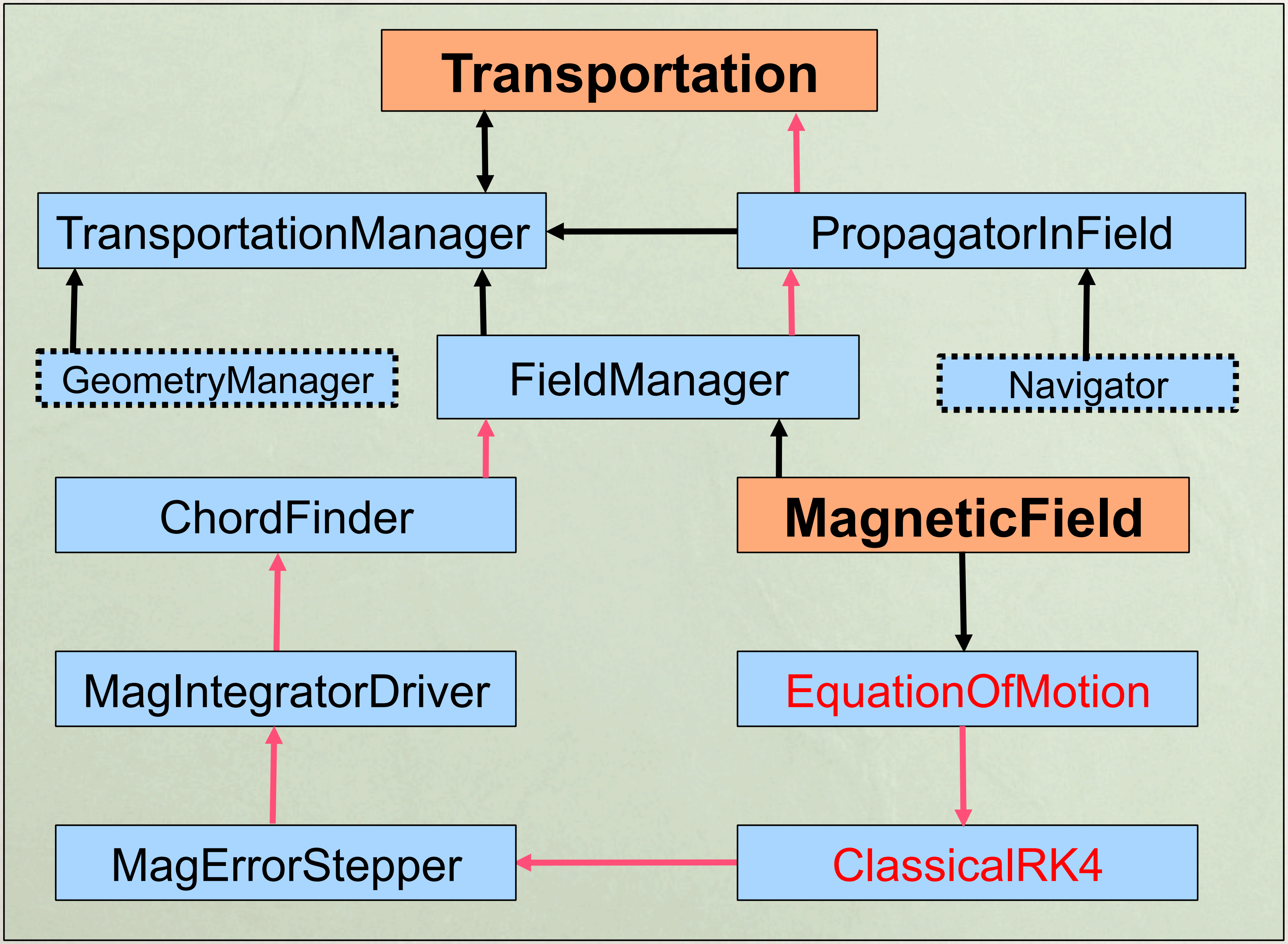
- How can we use many-core for Geant4?
- Geant4 performance studies with CMS detector





  - Geometry and processes are the libraries taking up most of the time
  - One of hot spots of processes is transportation
- Investigating concurrent particle transportation engine
  - Study particle transportation on GPGPU
  - Track level parallelism (dispatcher)

G4Transporation AlongStepGPIL For Charged Particles



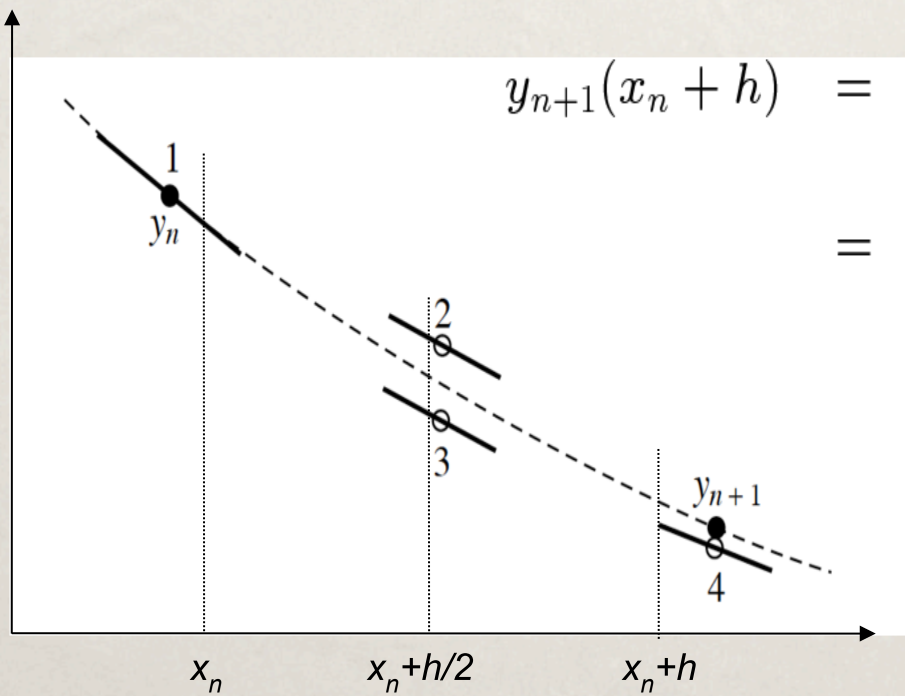
Equation of Motion and Runge-Kutta Method

* **Equation of motion in a magnetic field**

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

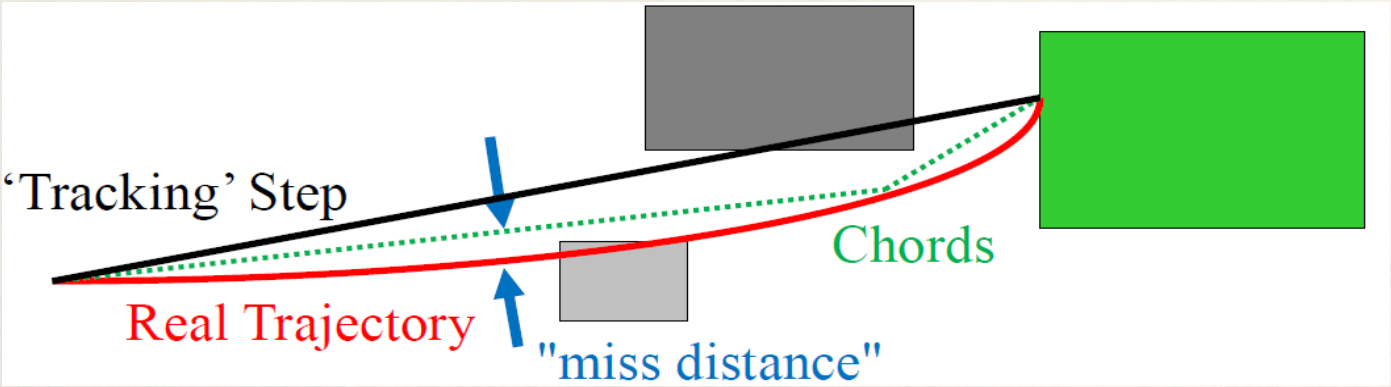
* **4-th order Runge-Kutta (RK4): 4 evaluations of f(x,y)**

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

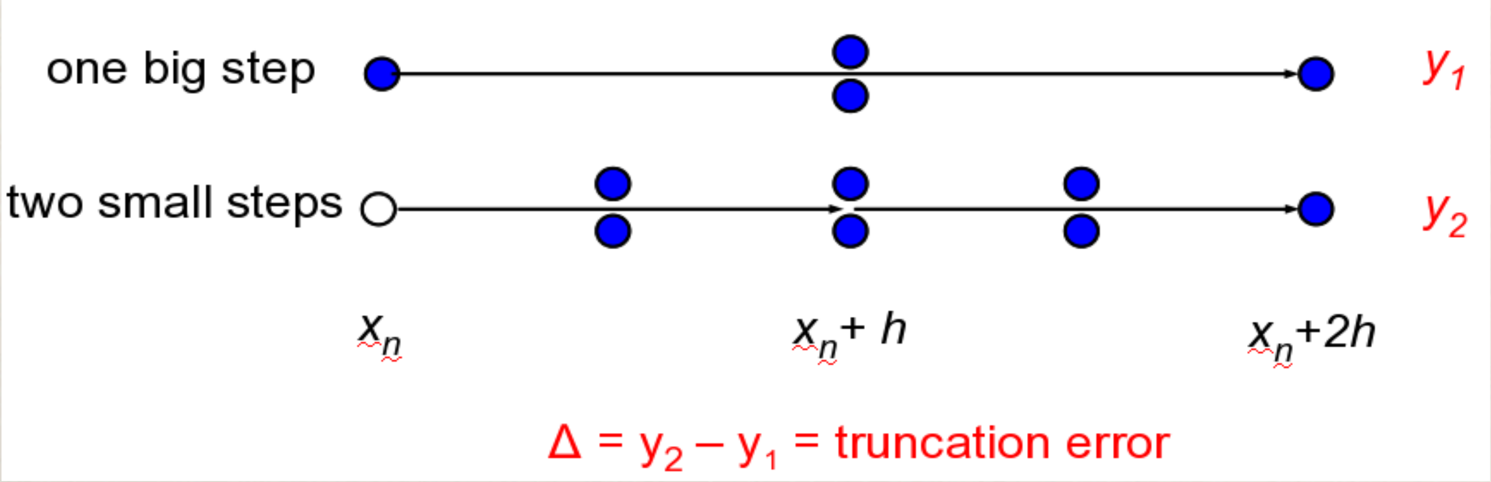


Adaptive Step Size Control

* **Quick advance**: miss distance < dmax



* **Accurate advance**: truncation errors of step doubling in RK4:11 evaluations of rhs of the equation of motion - difference in (x,p)



Problem Definition

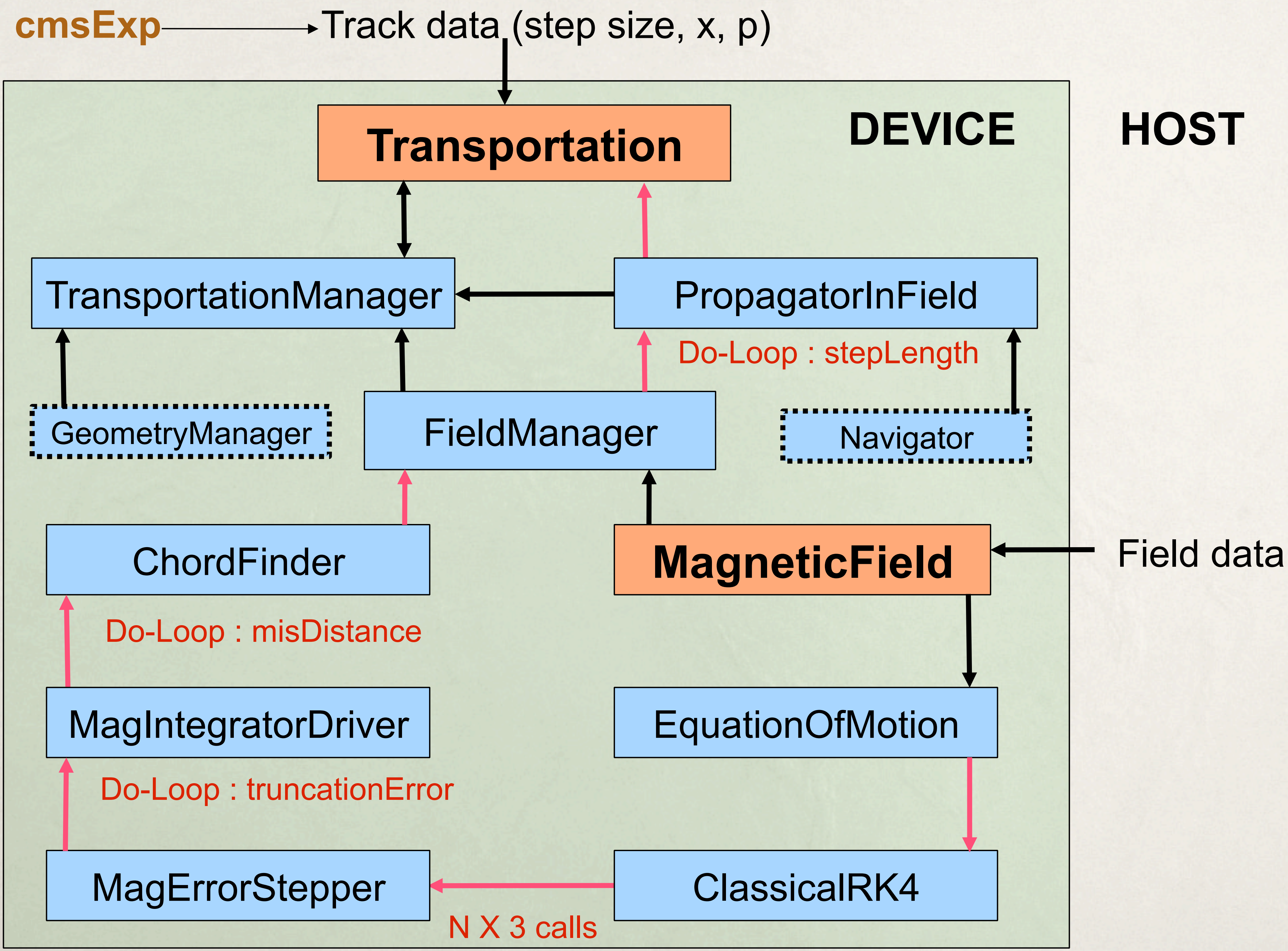
* **Isolate key components of Geant4 particle transportation**
  * evaluation of magnetic field (B) values
  * rhs of the equation of motion in a given B
  * evaluation of the 4th order Runge-Kutta (RK4)
* **Measure performance with the Runga-Kutta driver for adaptive step size control**
* **Test Geant4 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 Processor 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

- **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 code version 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 a bundle of secondary tracks produced by cmsExp
- **Device code version II optimized for GPU**
  - 4th order Runge-Kutta, field map with texture memory and etc.

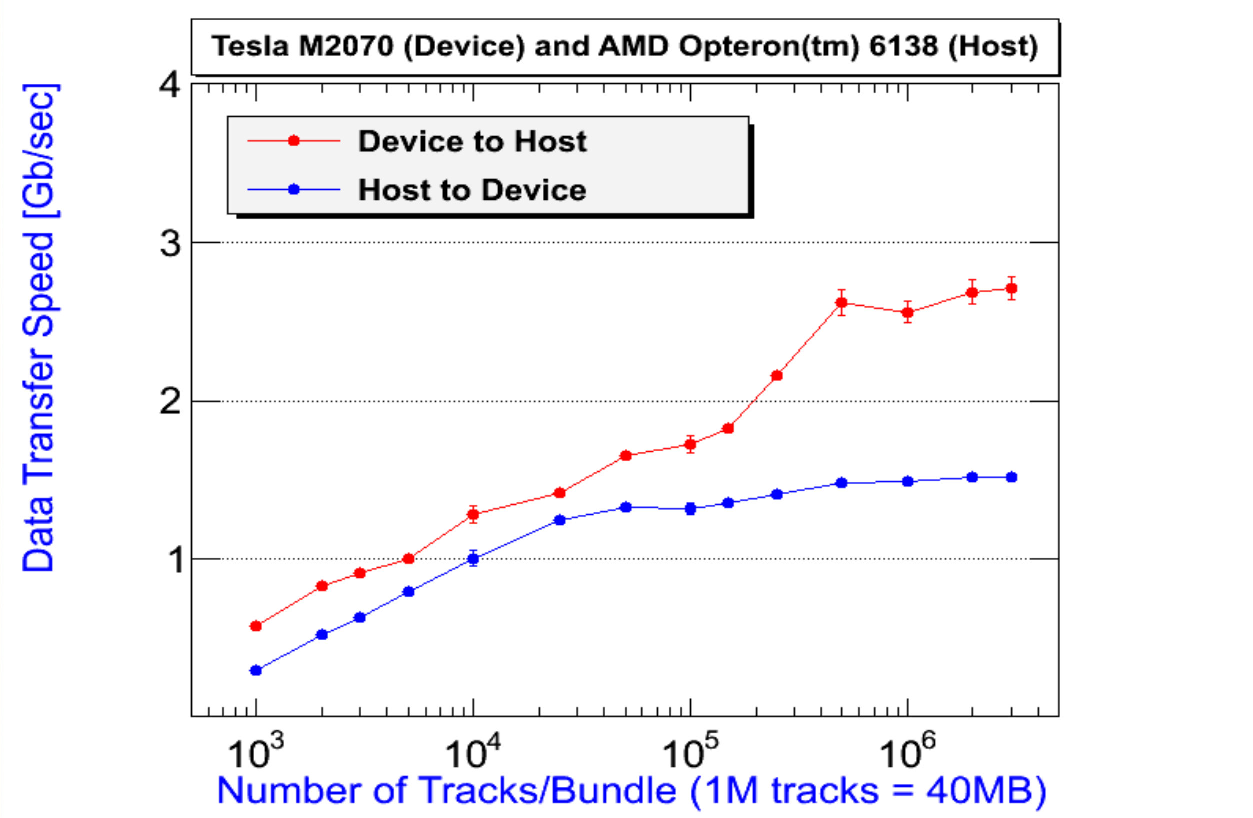

cmsExp
Track data (step size, x, p)
DEVICE
HOST
Transportation
TransportationManager
PropagatorInField
Do-Loop : stepLength
GeometryManager
FieldManager
Navigator
ChordFinder
MagneticField
Field data
Do-Loop : misDistance
MagIntegratorDriver
EquationOfMotion
Do-Loop : truncationError
MagErrorStepper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

Performance Measure

* **Performance Gain 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

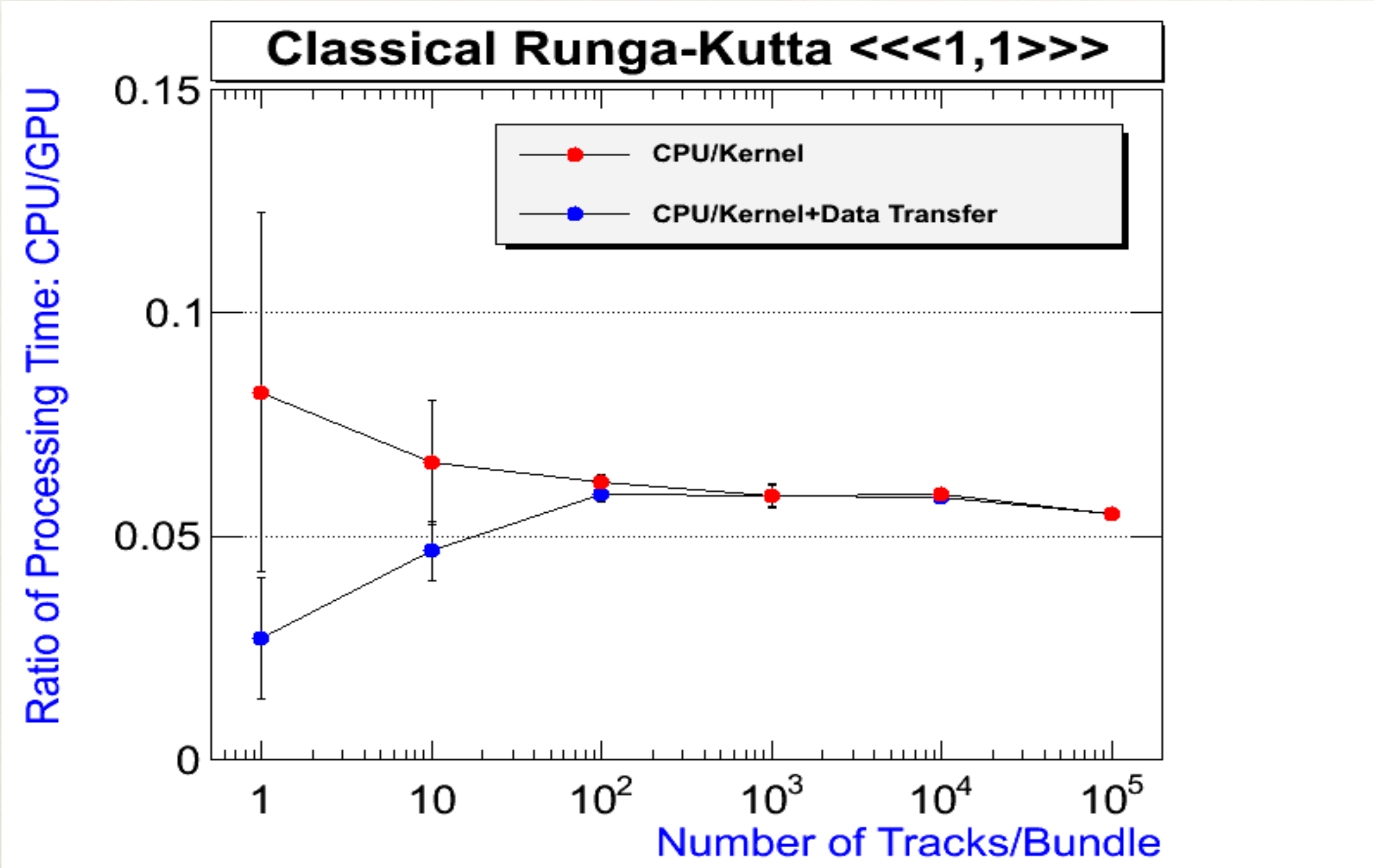
* **Data transfer speed** for track bundles between host and device



* **Minimize data transfer between host and device**
  * Bandwidth device-device is O(100) (GB/sec)
  * One large transfer is better than many small transfers

Performance: Single Thread GPU

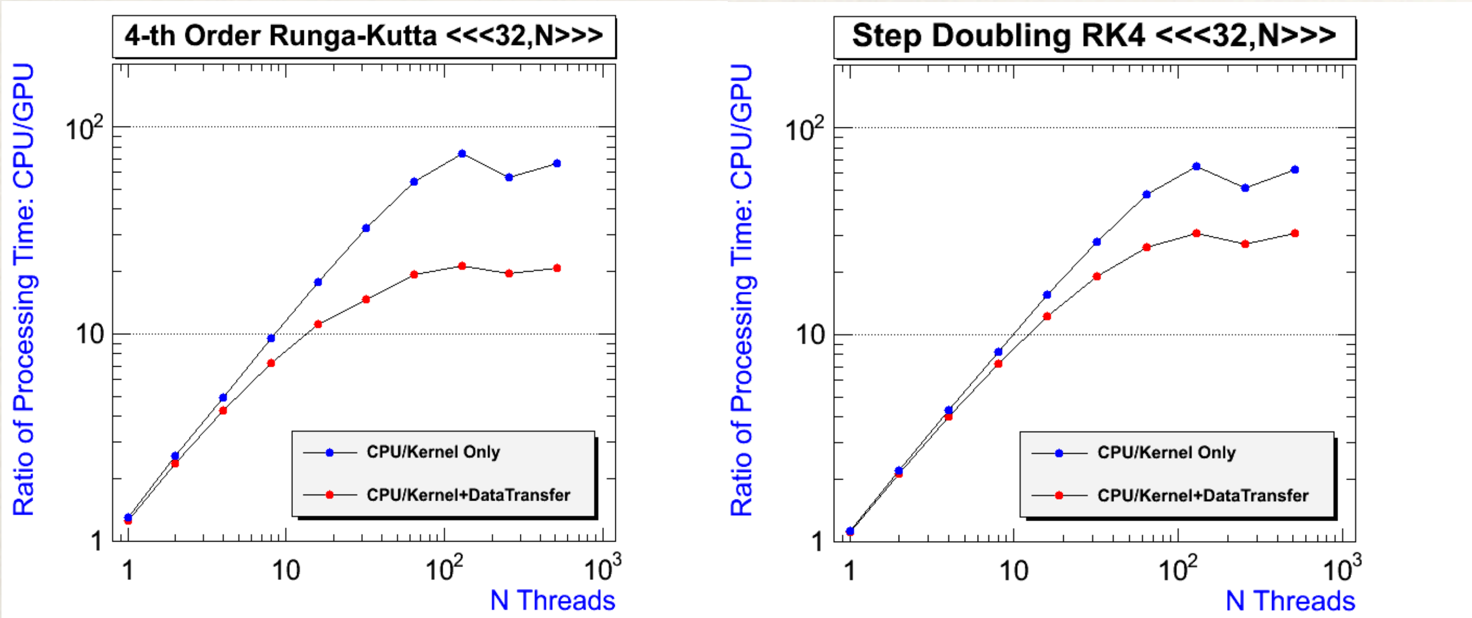
- **<<<BLOCK=1,THREAD=1>>> for 4th order Runge-Kutta**



- **1 GPU cores ≈ 1/18 CPU cores**
  - Clock speed (½), floating point calculation (¼), and etc.

Performance: Kernel

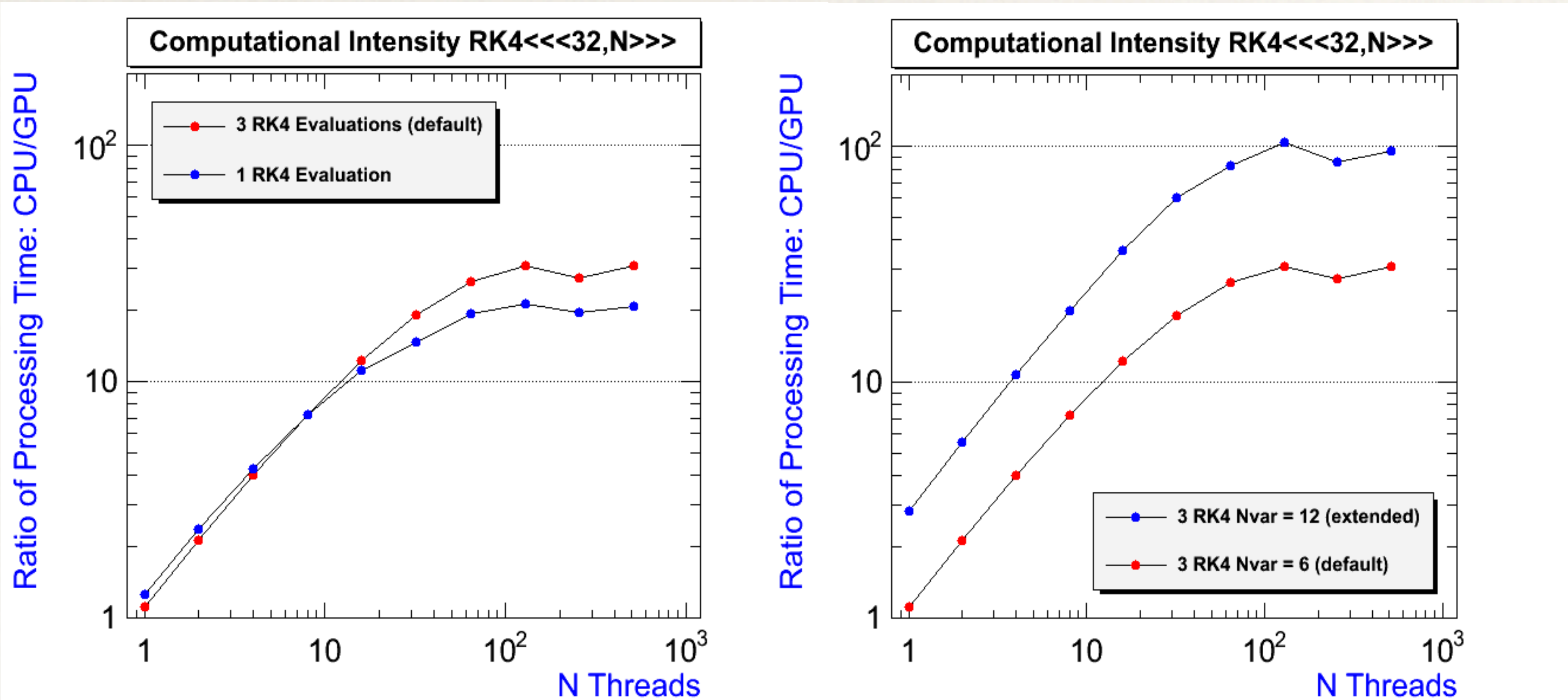
- **RK4: Time(Kernel only) vs. Time(Kernel+data transfer)**



- **Optimize kernel execution**
  - Overall (kernel+data)/kernel ~3 for RK4 and ~2 Adaptive RK4
  - Need to minimize data transfer between host and device
- **Much better gain (x90)**
  - Better GPGPU memory access pattern

Performance: Computational Intensity

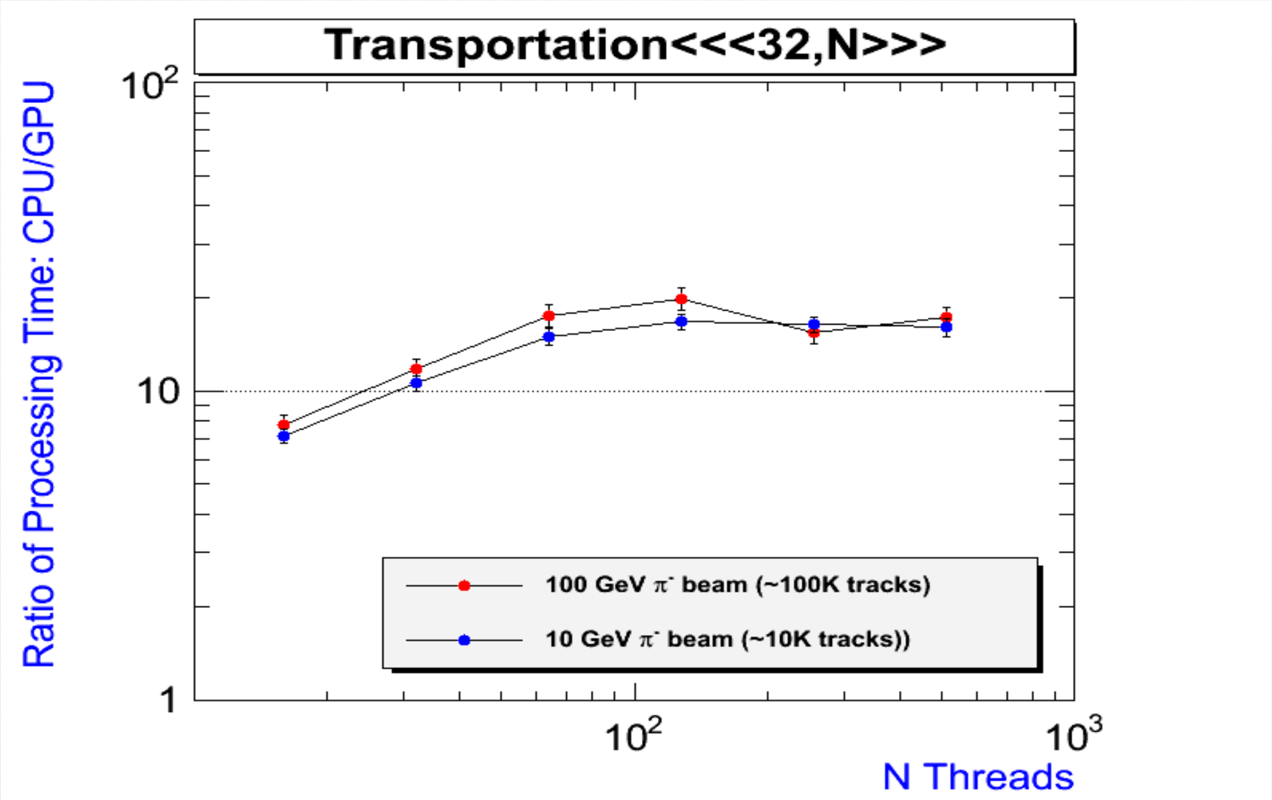
* **Number of RK4 evaluations and number of variables**



* **Optimize arithmetic for computational intensity**
  * Do more arithmetic calculations on GPU
  * Maximize independent parallelism (more for-loops)

Performance: Realistic Data From cmsExp

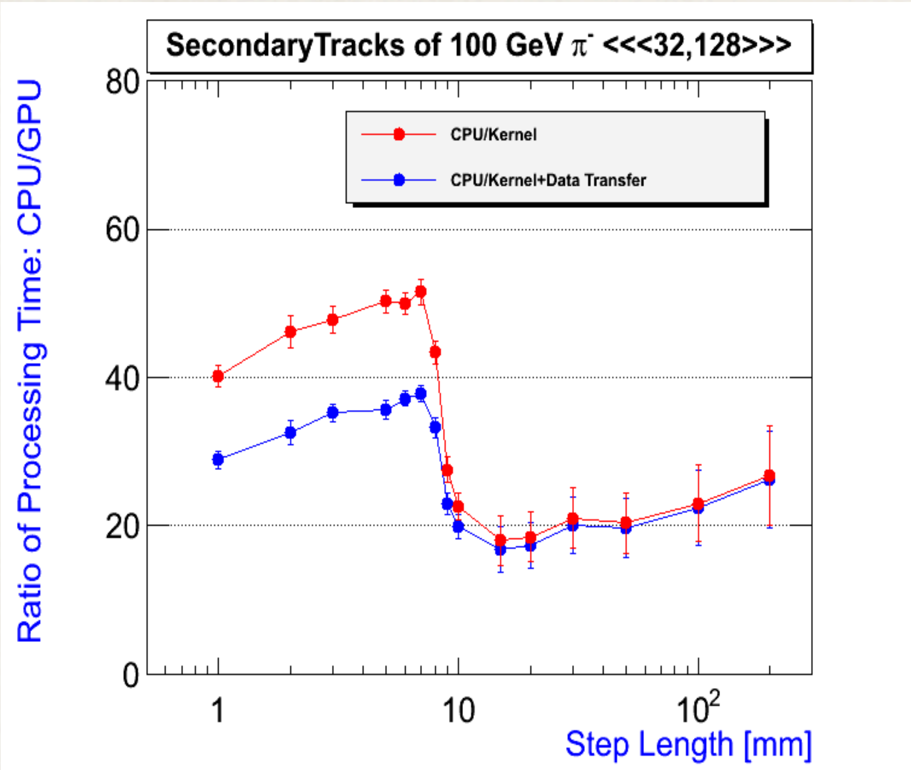
- CPU/GPU for the first step transportation for secondary particles produced by 10 GeV pions and 100 GeV pions
- Full chain of transportation with a step length = 1cm

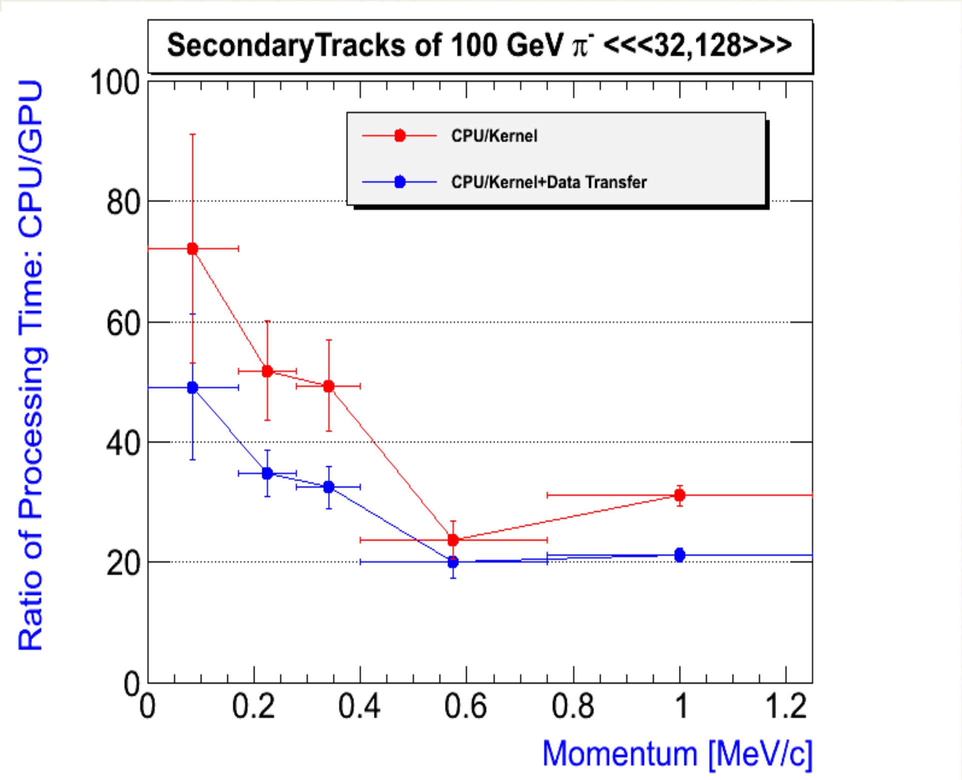


- **Need additional arithmetic logistics to improve the gain**

Dependencies: Momentum and Step Length

* **CPU/GPU for the first step of secondary tracks**





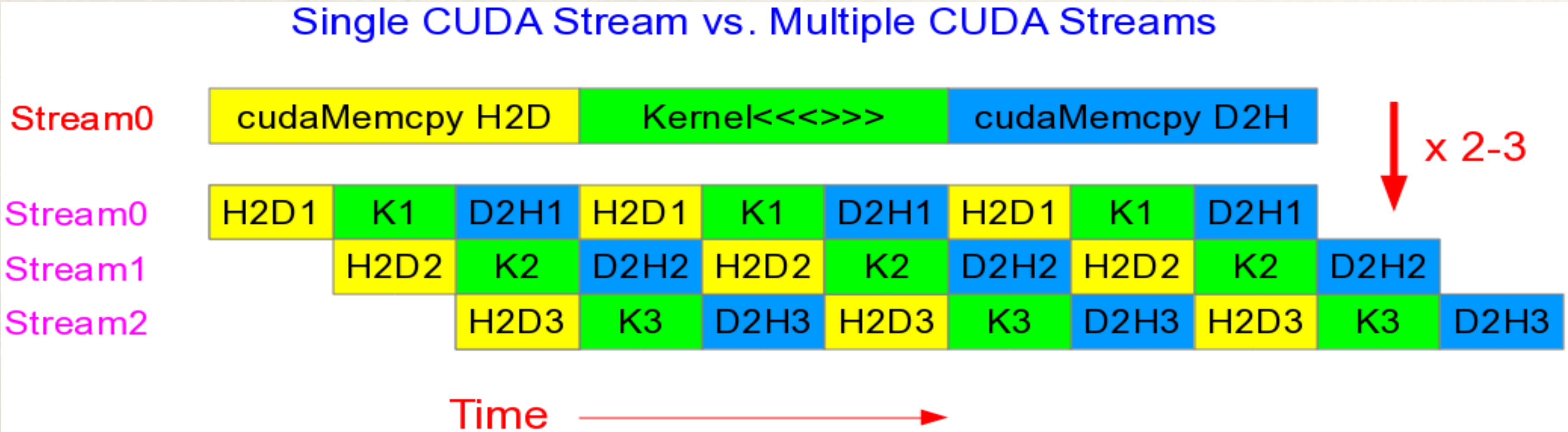
* **Optimize calculation uniformity**
  * Keep GPU multiprocessors equally busy
  * Group tracks with same number of RK4 evaluations as possible

Performance: 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 which is suitable for using the texture memory
- Texture interpolation is **twice as fast** as the explicit interpolation (for random access)
- Gain seems to be due to memory latency: no difference if input data are ordered
- **No noticeable difference in realistic data** for three evaluations of RK4 with/without using the texture

Concurrent Kernel/Stream

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



- To get the theoretical gain, need twice as much arithmetic work as the single RK steps
- On our examples, the gain seems to be the same range as the accuracy of the timings with the GPU optimized codes
- Estimating from profiling using callgrind and IgProf, it looks like there is only 40% more work to be gained without any geometry.
- **Need to add more calculations on GPU**

Going Beyond Transportation

- Support a minimal set of geometry classes to assist transportation and EM physics processes with a simple geometry (something similar to the CMS Ecal Barrel).

- Adopted geometry classes by Otto Seiskari: use geometry interface classes (Geometry, BasicGeometry) to relocate geometry pointers on GPU.

- Convert a set of geometry classes of geant4.9.5 to c-structs which work for host and device simultaneously.

- An initial implementation includes a navigator (normal navigation) and a couple of solids (box, trd, tubs) along with logical/physics volume classes. The current set is good only for ComputeStep for neutral particles (gamma) or charged without a magnetic field. To fully work with the transportation codes, an extension is necessary to include classes for voxel navigation, intersection locator and touchable history.

Going Beyond Transportation

* **Performance of the linear navigator (for gamma) for one step.**
* Ratio of processing time for [1 CPU]/[448 Cuda Cores] is around 25 to compute a step with a very simple geometry (4-phi x 3-z segments).
* Performance gain highly depends on the number of geometry volumes. The more complicated geometry yields the higher relative CPU/GPU performance gain due to the fact that initializing the navigation history takes more time for the first step (so far, we only do one step).
* The current implementation (with only normal navigation) is far from the geometry look-up logistics of the standard Geant4 which utilizes the touchable history, and voxel or other navigation.
* Adding navigator (i.e., geometry) for gamma affects the performance of transportation for charged particles even though the navigator is not called at all - around 20% degradation in the GPU time. It seems that loading geometry changes memory caching mechanism of GPU.

Going Beyond Transportation

* **Current status**
* Understanding performance with the current set of implementation with various geometry configuration.
* Evaluating the performance quantitatively for the normal navigation.
* Identified necessary extension and plan to implement additional structs (voxel navigation, multilevel locator, touchable history and etc.) to fully support the transportation process.

Conclusions

* **A core part of Geant4 Particle transportation has been tested on GPU**
  * Time CPU/GPU ~ 20 with realistic data using 448 Cuda cores
  * Identified key factors to maximize the GPU's ALU capacities
* **Lessons learned**
  * Increase computational intensity on GPU
  * Look for other transportation algorithm suitable for uniformity of calculations
  * Organize input data for optimal efficiency of kernel executions and data transfers

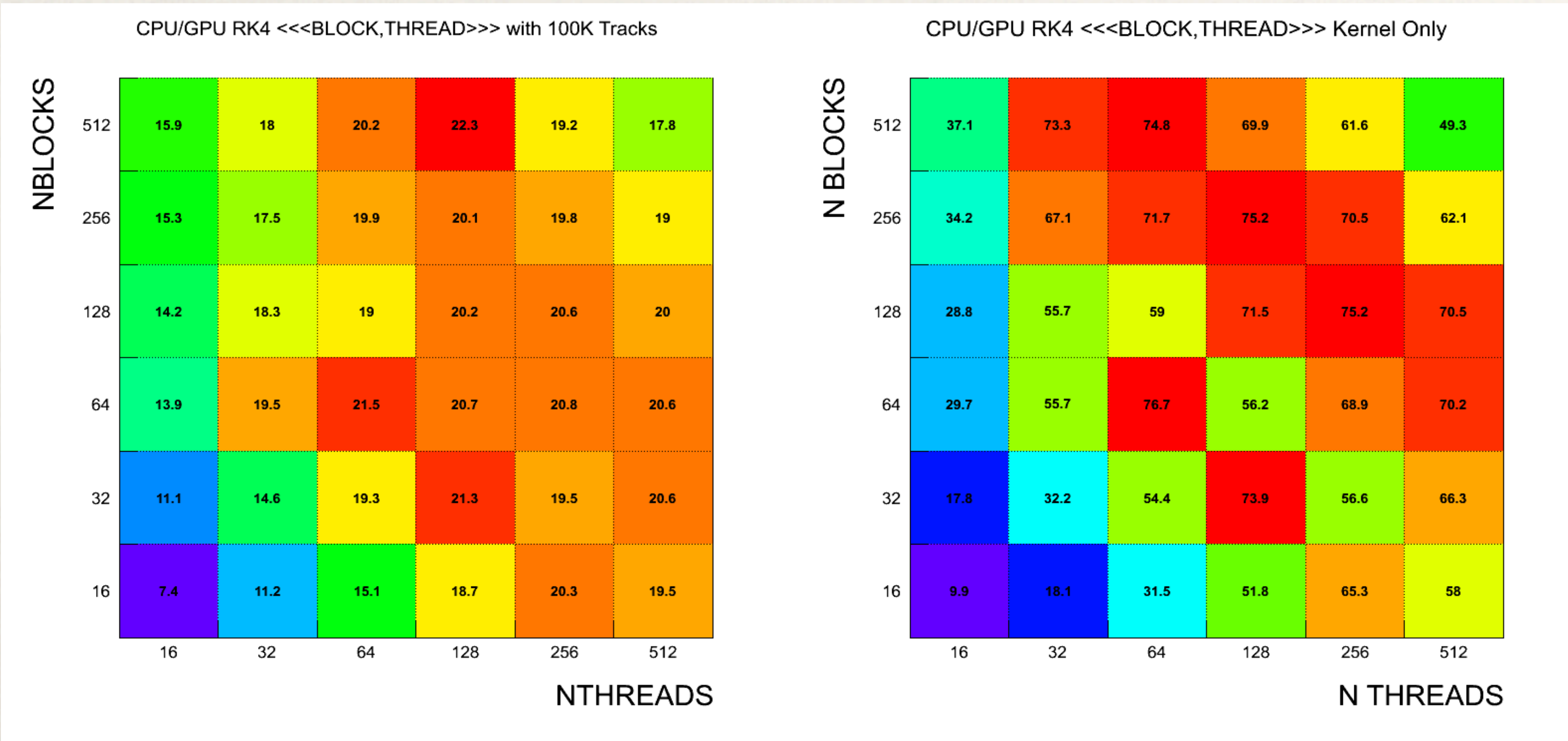
Outlooks

* **Adding geometry on device**
  * Simple EM detector (something like CMS crystals)
  * Generalize transportation including neutrals and intersection with geometry
* **Develop device codes for EM physics**
  * Multiple stepping on device to increase computations
  * Generalize transportation including post step actions and secondary particles handling
* **Optimize GPU resources**
  * Test multiple streams (concurrent calculation and copying data up/down to GPU)

Back-up Slides

Performance: CPU/GPU

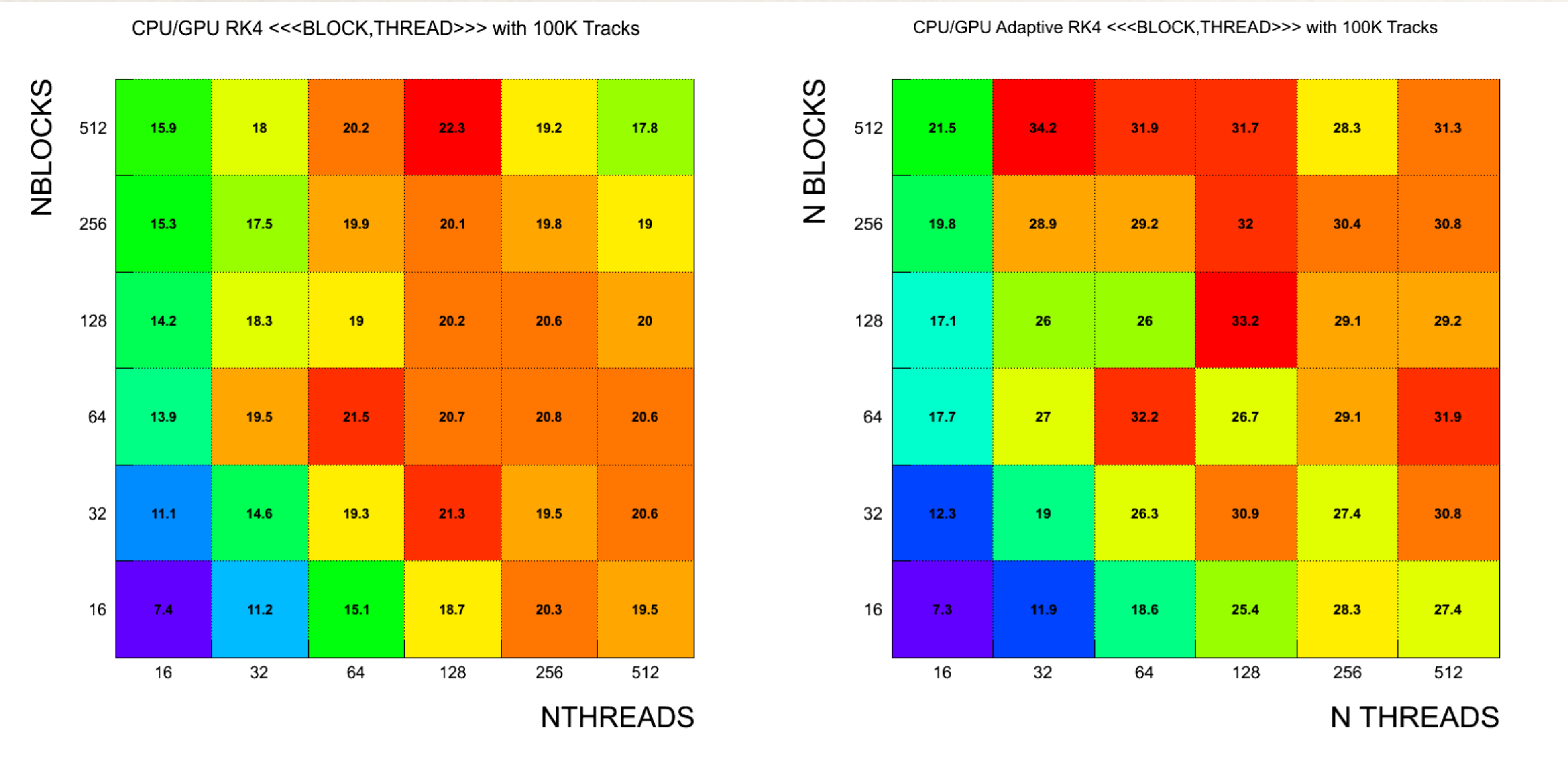
- RK4: Kernel+Data Transfer vs. Kernel Only



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

Performance: Computational Density

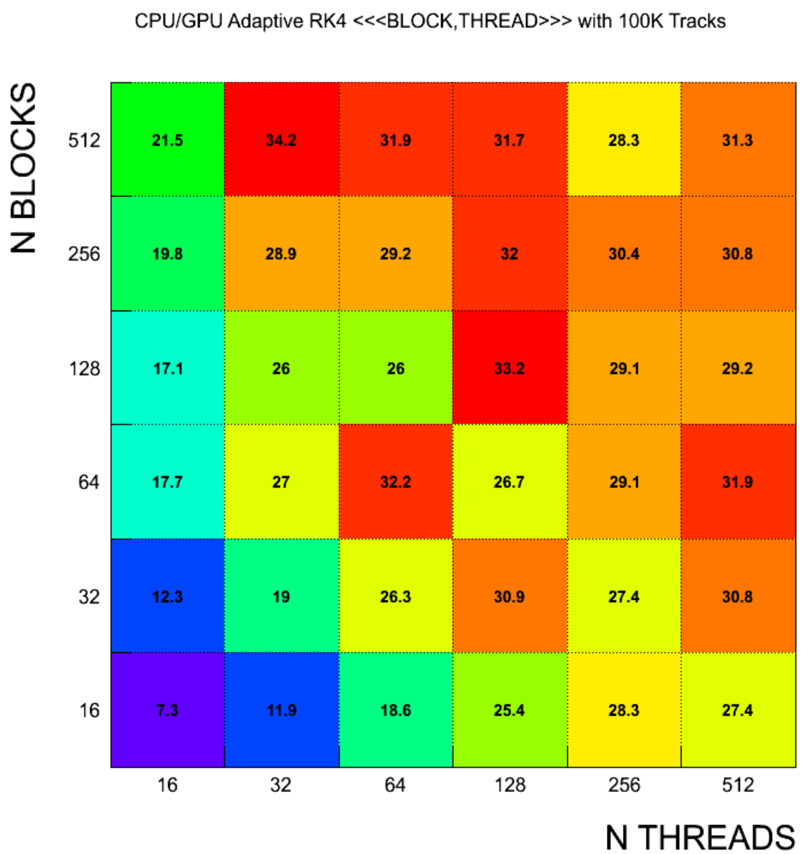
- Number of RK evaluations: 1-RK4 vs. 3-RK4

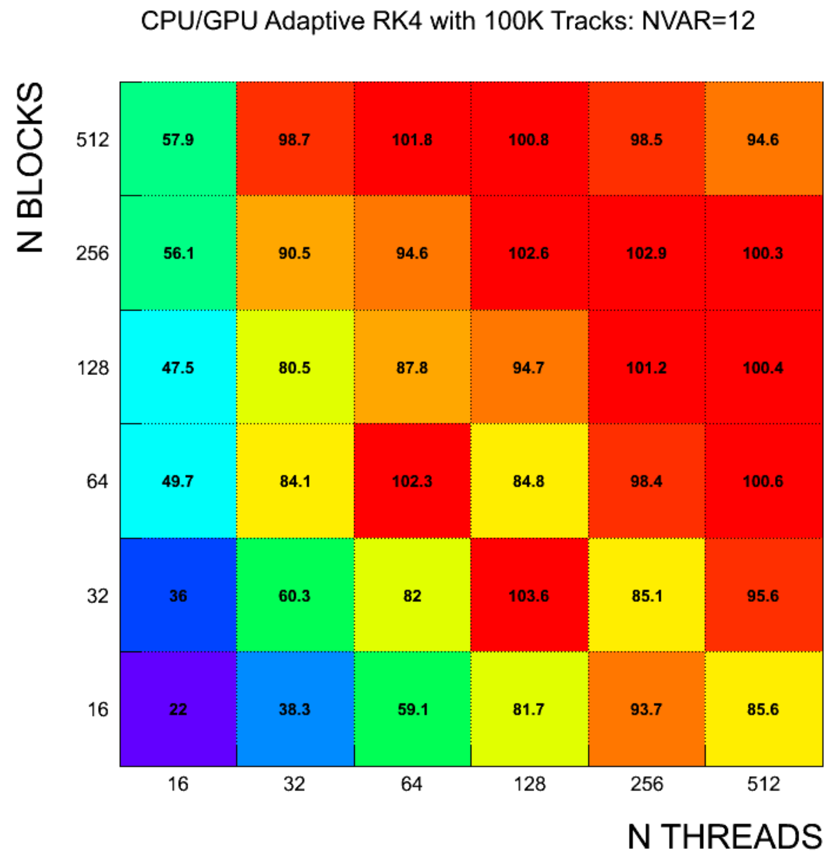


- (Left) one RK4 evaluation (Right) three-RK4 evaluations

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