Vectorization/GPU Summary: Vector & GPU Prototypes

V. Daniel Elvira

Fermi National Accelerator Laboratory

The New Computing Paradigm

New computing paradigm → R&D program for future HEP software (in particular detector simulation)

- Hardware landscape rapidly changing for power efficiency
- Parallelism is no longer optional, it must be explored thoroughly and presents many challenges
- Maximize instruction throughput and data locality

A vision for HEP/HPC simulation

- Massively parallelized particle (track level) transportation engine
- Comply with different architectures such as GPU, MIC, etc

A Vector Prototype
(F. Carminati – CERN)

Starting all over:

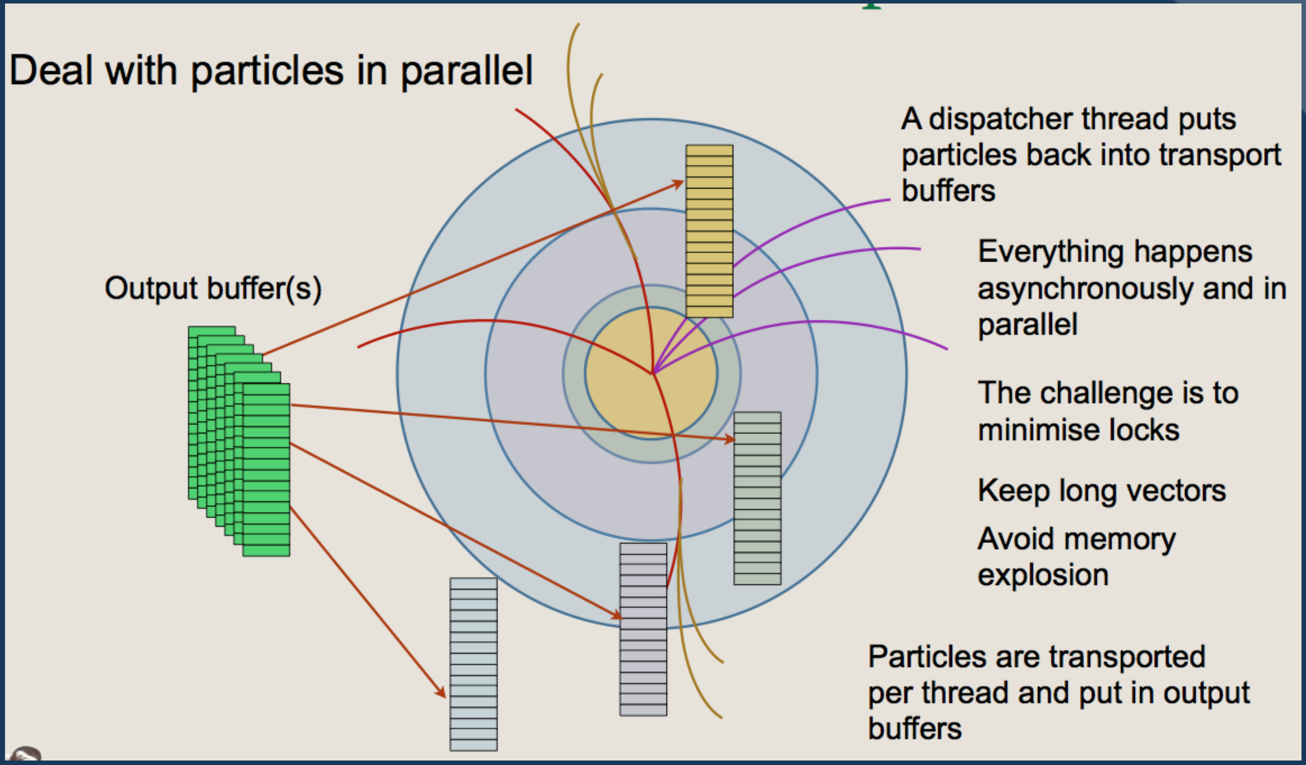
- Grand strategy
  - Explore from a performance perspective, no constraints from existing code
  - Expose the parallelism at all levels, from coarse granularity to micro-parallelism at the algorithm level
  - Integrate from the beginning slow and fast simulation in order to optimise both in the same framework
  - Explore if-and-how existing physics code (GEANT4) can be optimised in this framework Improvements (in geometry for instance) and techniques are expected to feed back into reconstruction

HEP transport is mostly local !

Locality not exploited by classical transport

A Vector Prototype

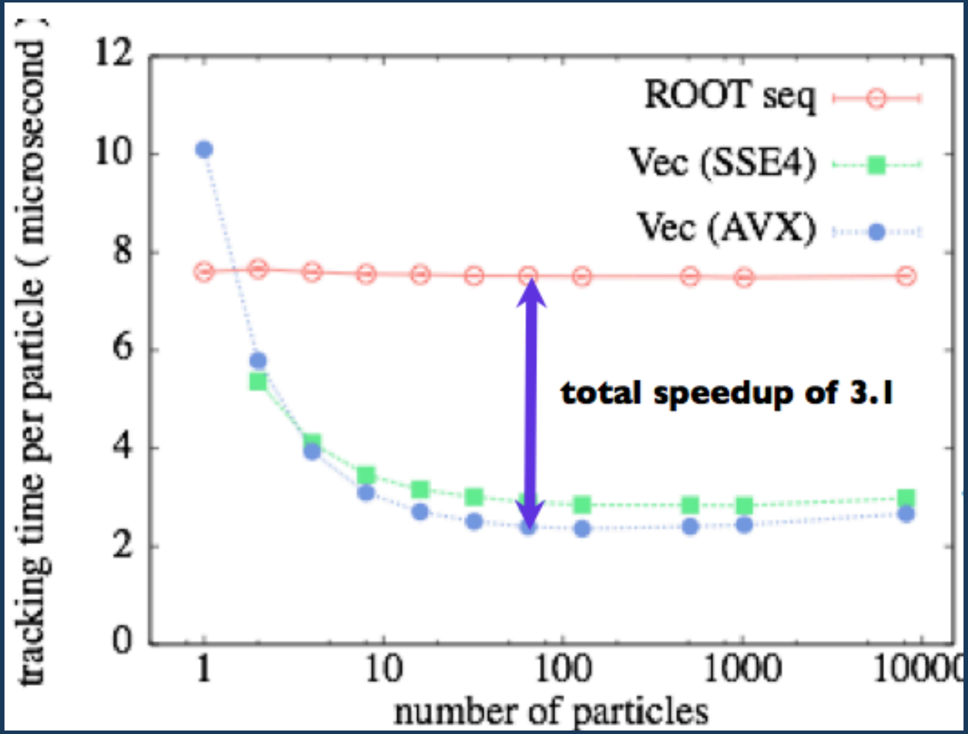
"Basketised" transport, tabulated G4 x-sections



The real gain in speed comes at the end by exploiting the (G)CPU hardware (vectors, instruction pipelining, instruction level parallelism)

A Vector Prototype

"Basketised" transport, tabulated G4 x-sections



Geometry navigation gain from vector processing of particles

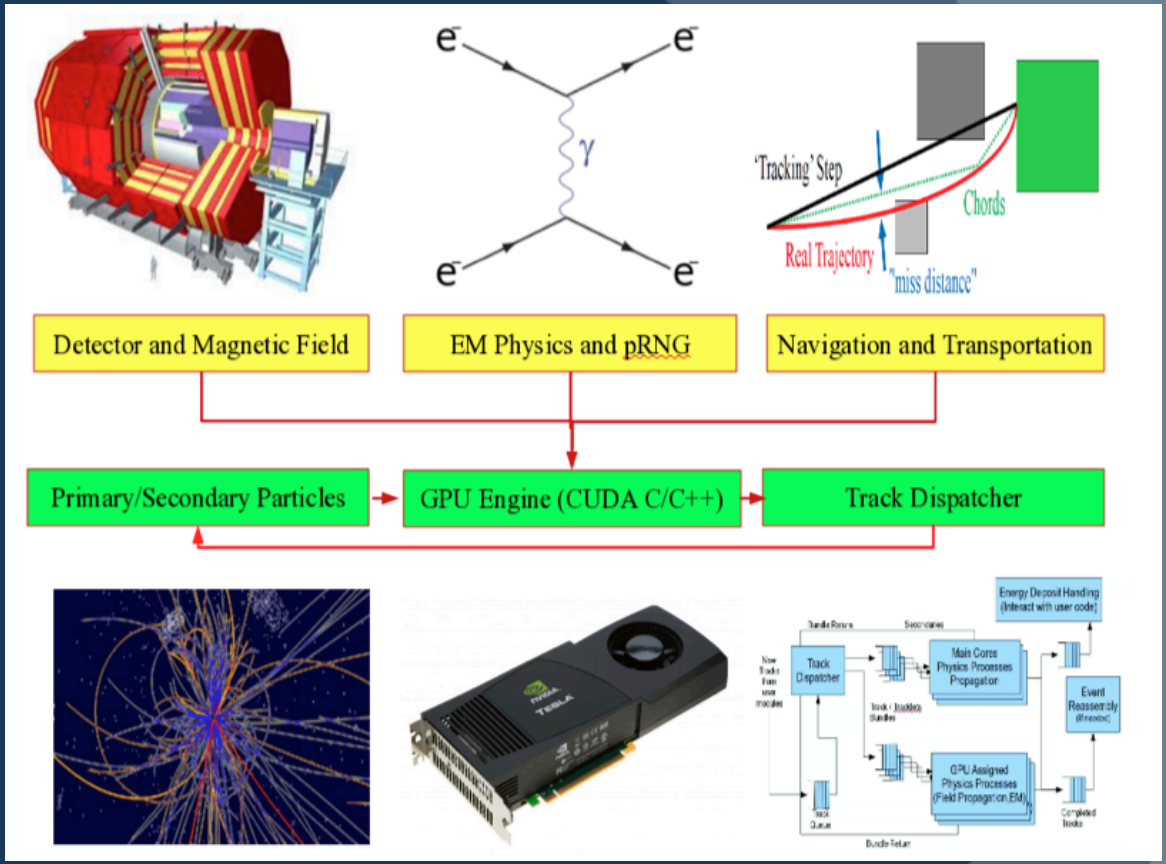
- Benefits from SIMD instruction sets and cache reuse

A GPU Prototype

(Soon Jun – FNAL)

Key components (CUDA code):

- Transportation in realistic EM field
- Geometry (simple “a la” CMS EM calorimeter)
- EM Physics (brem, ioniz., mult. Scatt. for e; Compton, γ-conv., P.E. for γ)
- Concurrent CUDA kernels



Considerations: Reduce branches, reuse data, …

A GPU Prototype

- Hardware (host + device)

	Host (CPU)	Device (GPU)
M2090	AMD Opertron™ 6134 32 cores @ 2.4 GHz	Nvidia M2090 (Fermi) 512 cores @ 1.3 GHz
K20	Intel® Xeon® E5-2620 24 cores @ 2.0 GHz	Nvidia K20 (Kepler) 2496 cores @ 0.7GHz

- Performance measurement
  - Gain = Time(1 CPU core)/Time(total GPU cores)
    Time=(data transfer + kernel execution)

	CPU [ms]	GPU [ms]	CPU/GPU
AMD+M2090	748	37.8 (62.6)*	19.8 (11.9)*
Intel®+K20M	571	30.4 (81.9)*	18.7 (7.0)*

GPU time gain of ~20, power consumption and relative hardware cost not included in the equation ➔ need to further improve algorithms

Integration and future

Vector prototype can serve as track bucket provider to the GPU prototype

- Developing "GPU connector" interface between GPU and vector prototypes
- Agree on common geometry implementation, data layout, physics, transport

Plan includes to share components, minimize branches, maximize locality, improve algorithms

Goal is to demonstrate substantial speed up in modern architectures