Portable HPC for High-Performance Simulation

Federico Carminati

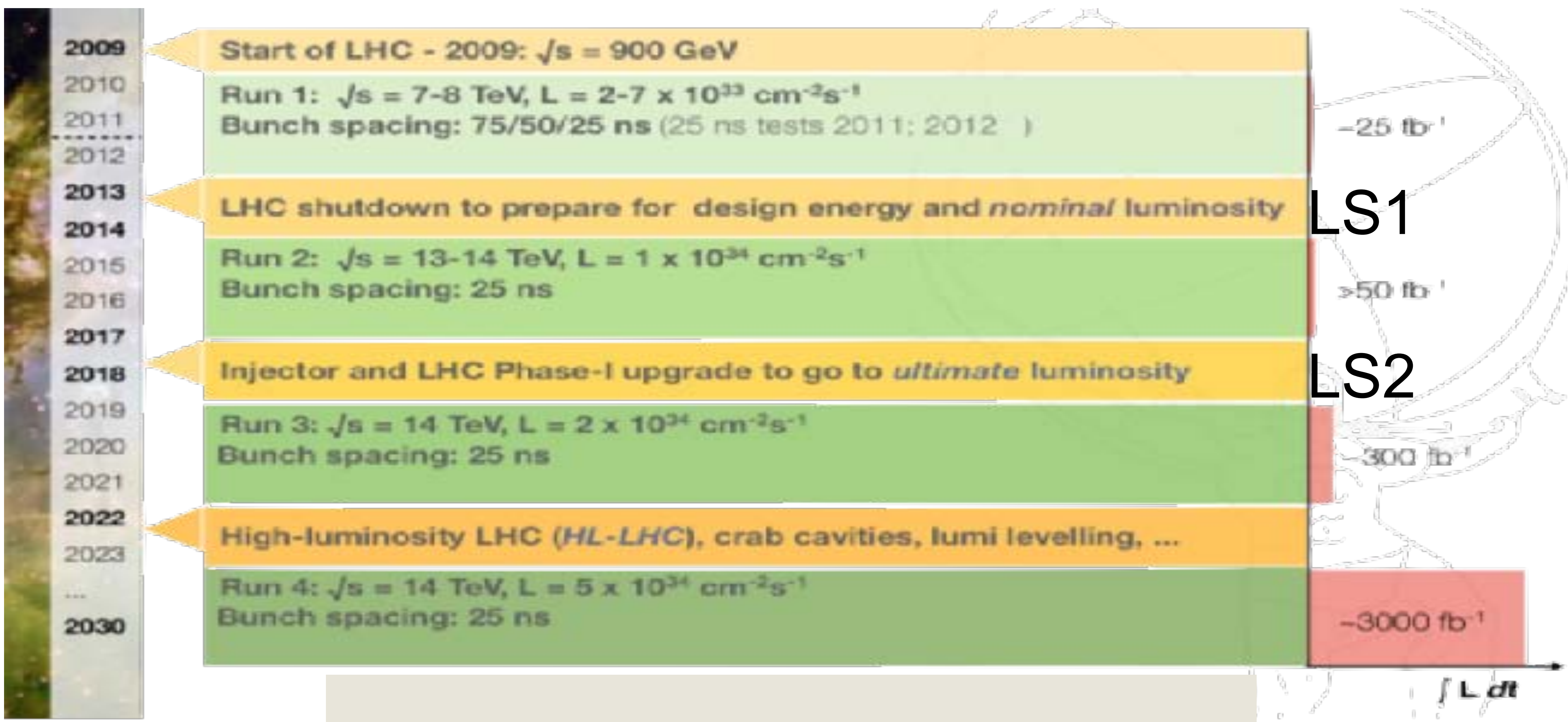
Future of Many-Core Computing in Science

29 May 2014 Wigner Datacenter

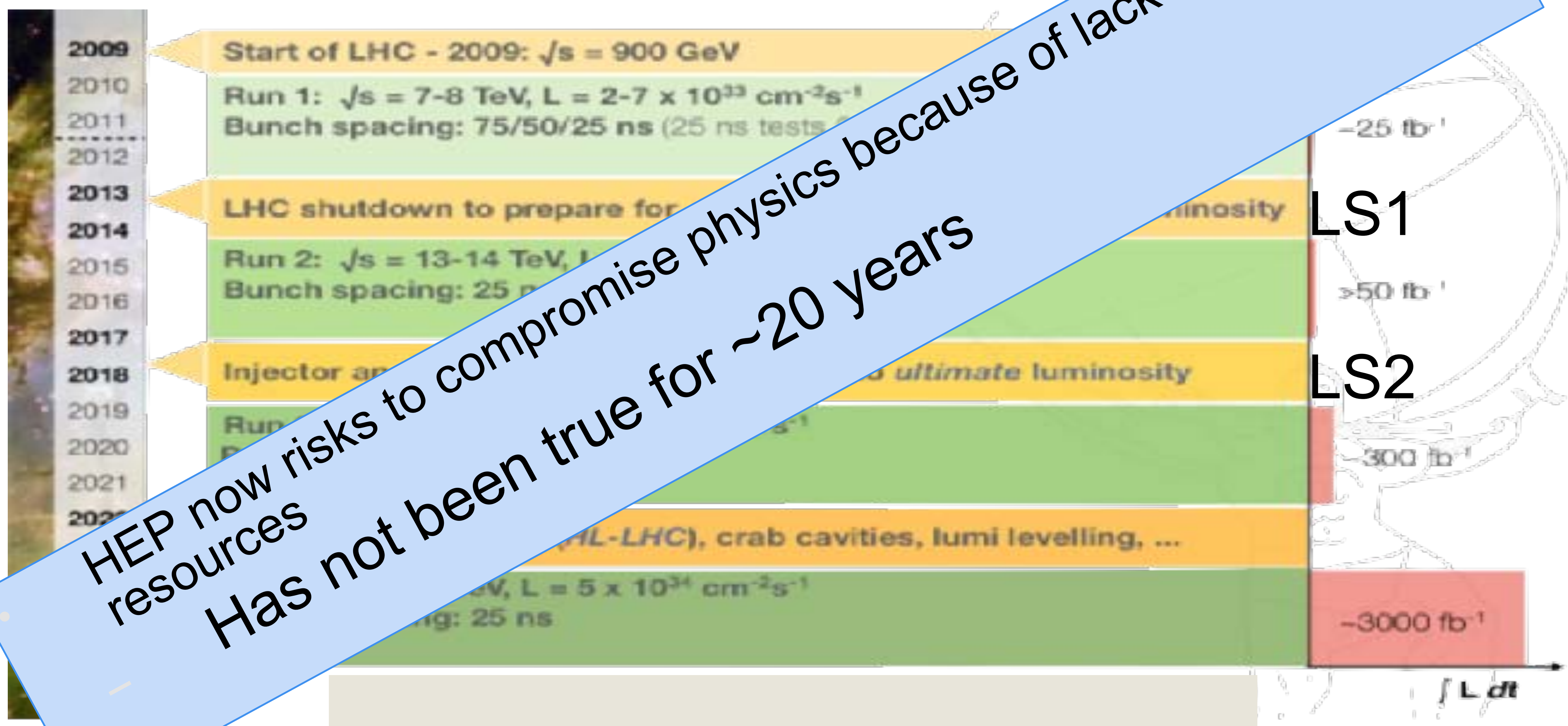
Thanks to

- John Apostolakis (CERN)
- Marilena Bandieramonte (Catania U.)
- Georgios Bitzes (CERN openlab)
- Rene Brun (CERN)
- Philippe Canal (FNAL)
- Federico Carminati (CERN)
- Johannes Christof De Fine Licht (CERN)
- Laurent Duhem (Intel)
- Daniel Elvira (FNAL)
- Andrei Gheata (CERN)
- Soon Yung Jun (FNAL)
- Guilherme Lima (FNAL)
- Mihaly Novak (CERN)
- Raman Sehgal (BARC)
- Oksana Shadura (CERN)
- Sandro Christian Wenzel (CERN)

A luminous future for HEP...



A luminous future for HEP...



HEP now risks to compromise physics because of lack of computing resources

Has not been true for ~20 years

CERN openlab BoS – May 8, 2014

The Eight dimensions

- The "dimensions of performance"
  - Vectors
  - Instruction Pipelining
  - Instruction Level Parallelism (ILP)
  - Hardware threading
  - Clock frequency
  - Multi-core
  - Multi-socket
  - Multi-node

Micro-parallelism: gain in throughput and in time-to-solution

Very little gain to be expected and no action to be taken

Gain in memory footprint and time-to-solution but not in throughput

Possibly running different jobs as we do now is the best solution

The Eight dimensions

- The "dimensions of performance"
  - Vectors
  - Instruction Pipelining
  - Instruction Level Parallelism (ILP)
  - Hardware threading
  - Clock frequency
  - Multi-core
  - Multi-socket
  - Multi-node

Micro-parallelism: gain in throughput and in time-to-solution

Very little gain to be expected and no action to be taken

Gain in memory footprint and time-to-solution but not in throughput

Possibly running different jobs as we do now is the best solution

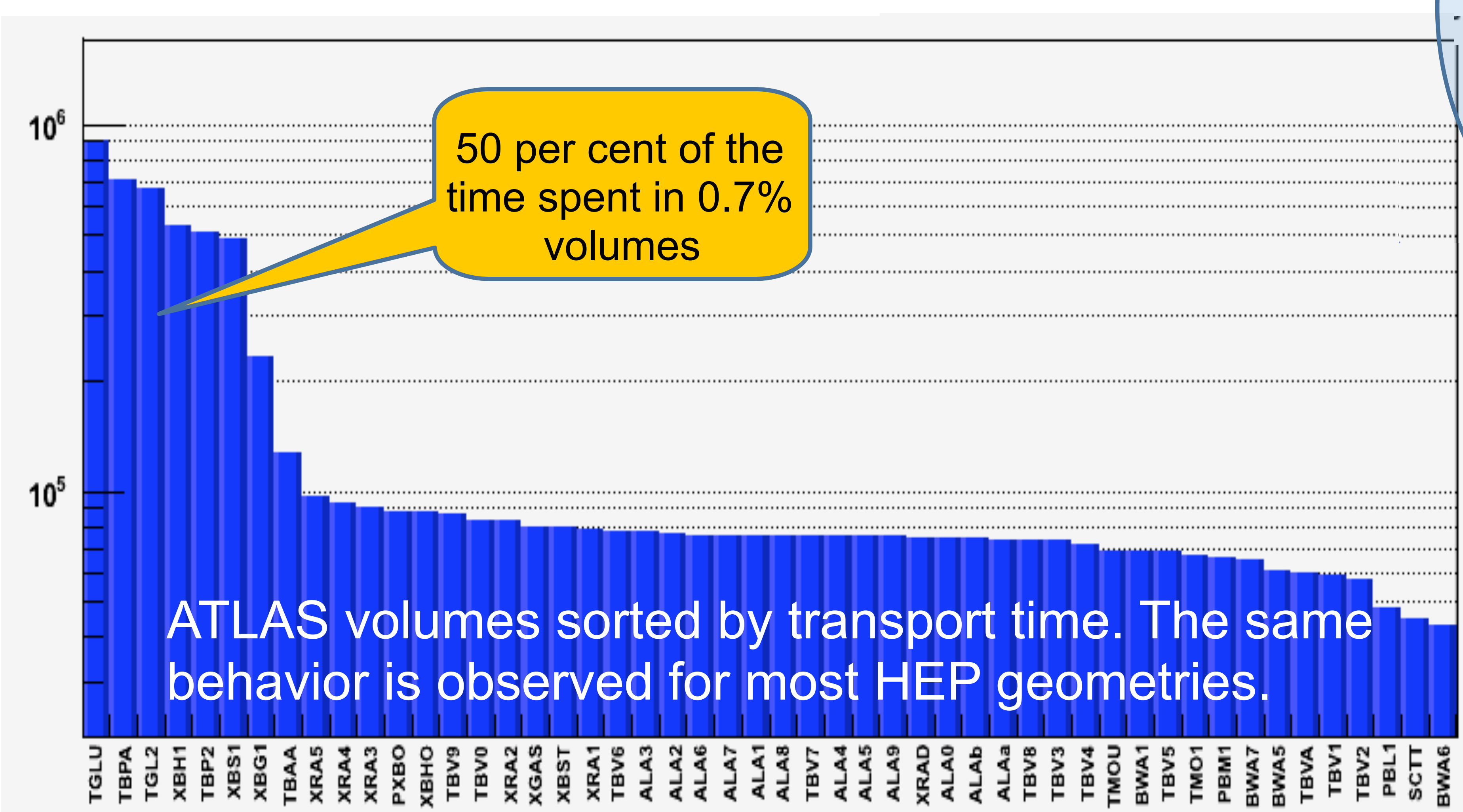
Expected limits on performance scaling			
	SIMD	ILP	HW
THEORY	8	4	1.35
OPTIMISED	6	1.57	1.25
HEP	1	0.8	1.25
Expected limits on performance scaling (multiplied)			
	SIMD	ILP	HW
THEORY	8	32	43.2
OPTIMISED	6	9.43	11.79
HEP	1	0.8	1

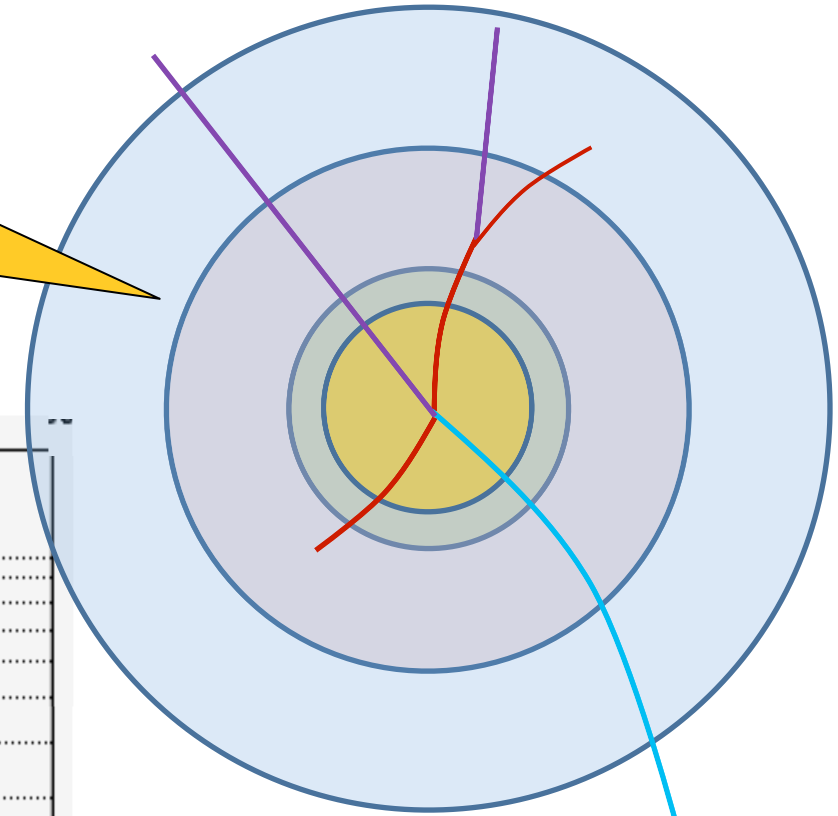
OpenLab@CHEP12

Classical HEP transport is mostly local

- Event- or event track-level parallelism will better use resources but won't improve these points

•Geometry navigation (local)
- Material – X-section tables
- Particle type - physics processes

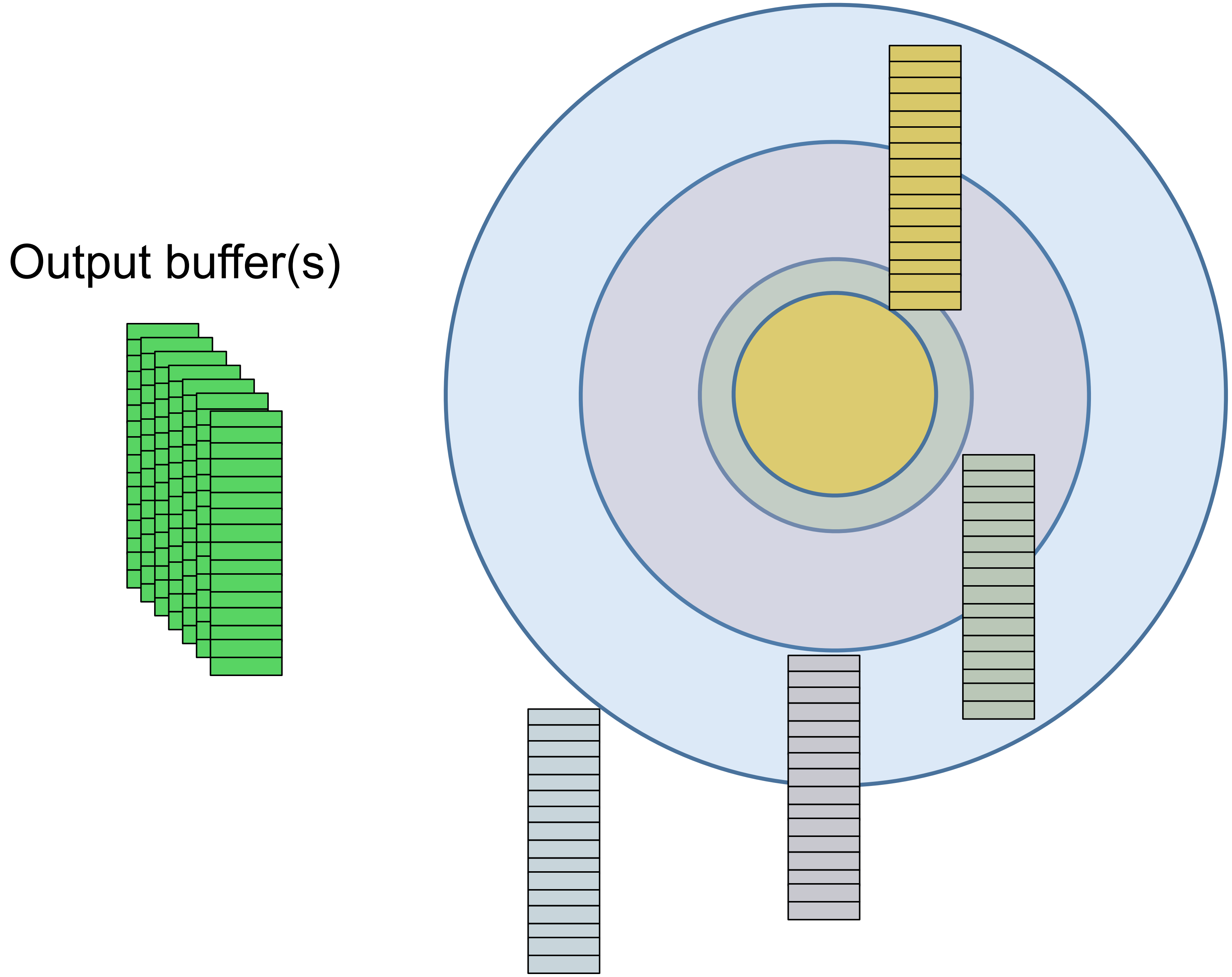




- Navigating very large data structures
- No locality
- OO abused: very deep instruction stack
- Cache misses

Introduced basketized transport

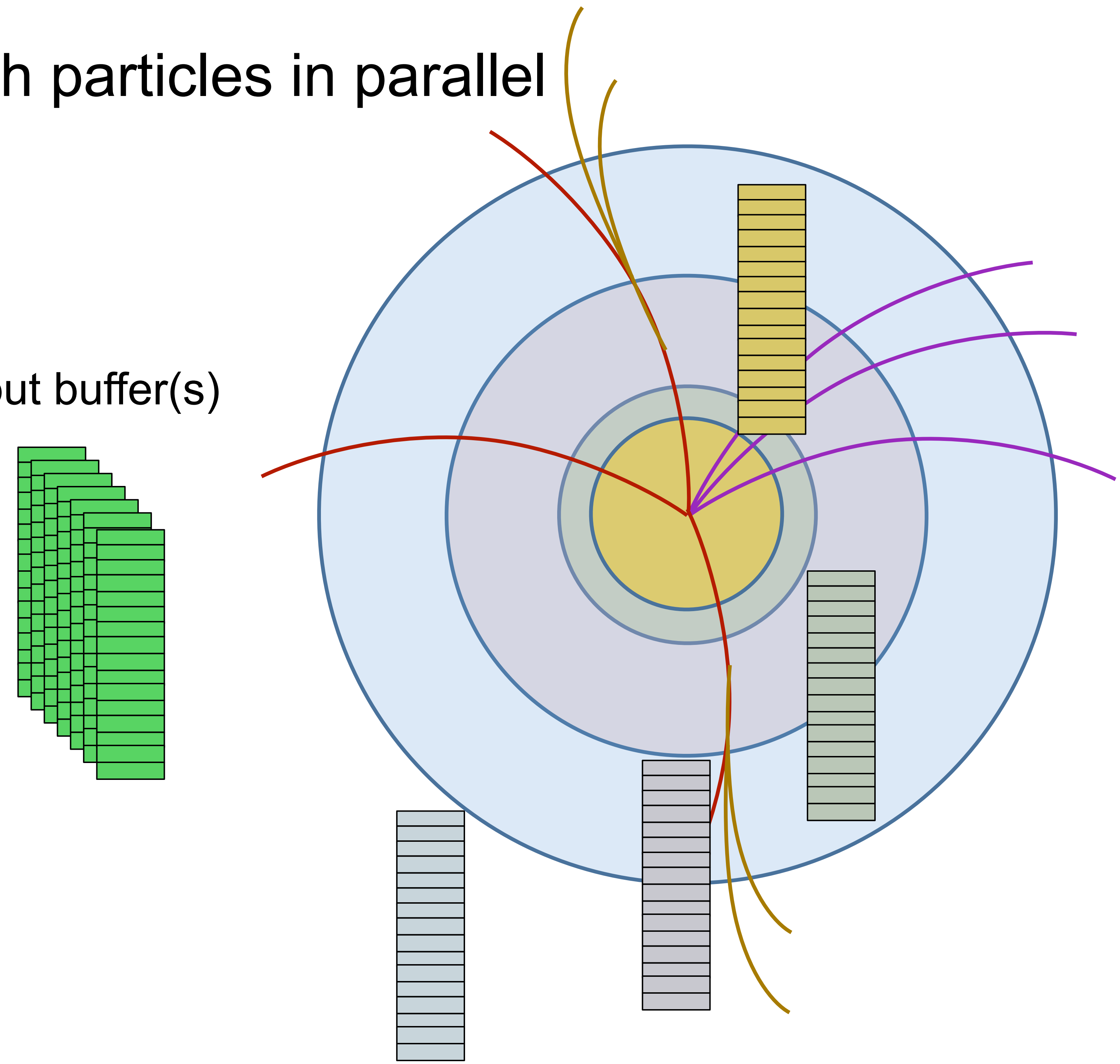
Deal with particles in parallel



Introduced basketized transport

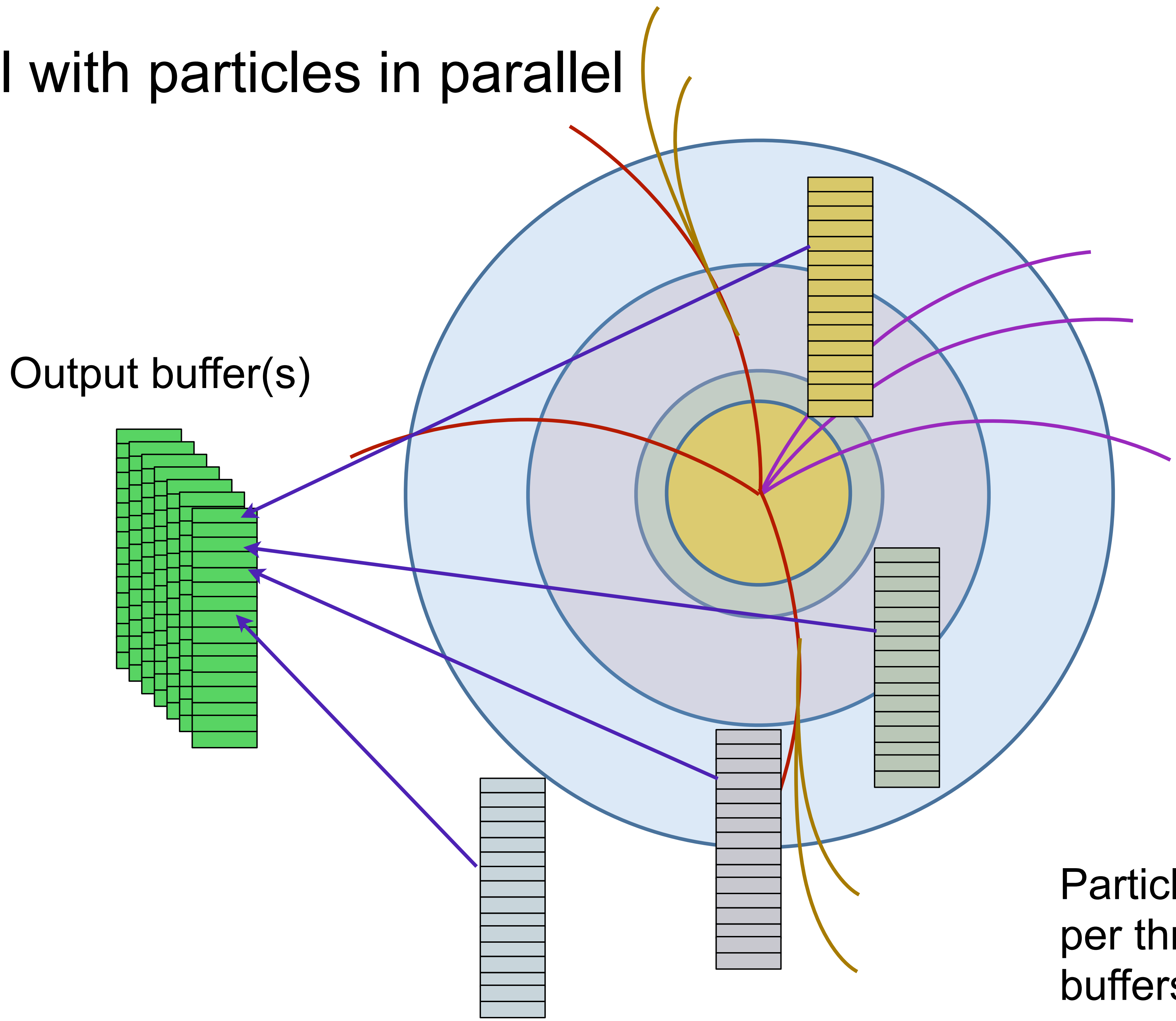
Deal with particles in parallel

Output buffer(s)



Introduced basketized transport

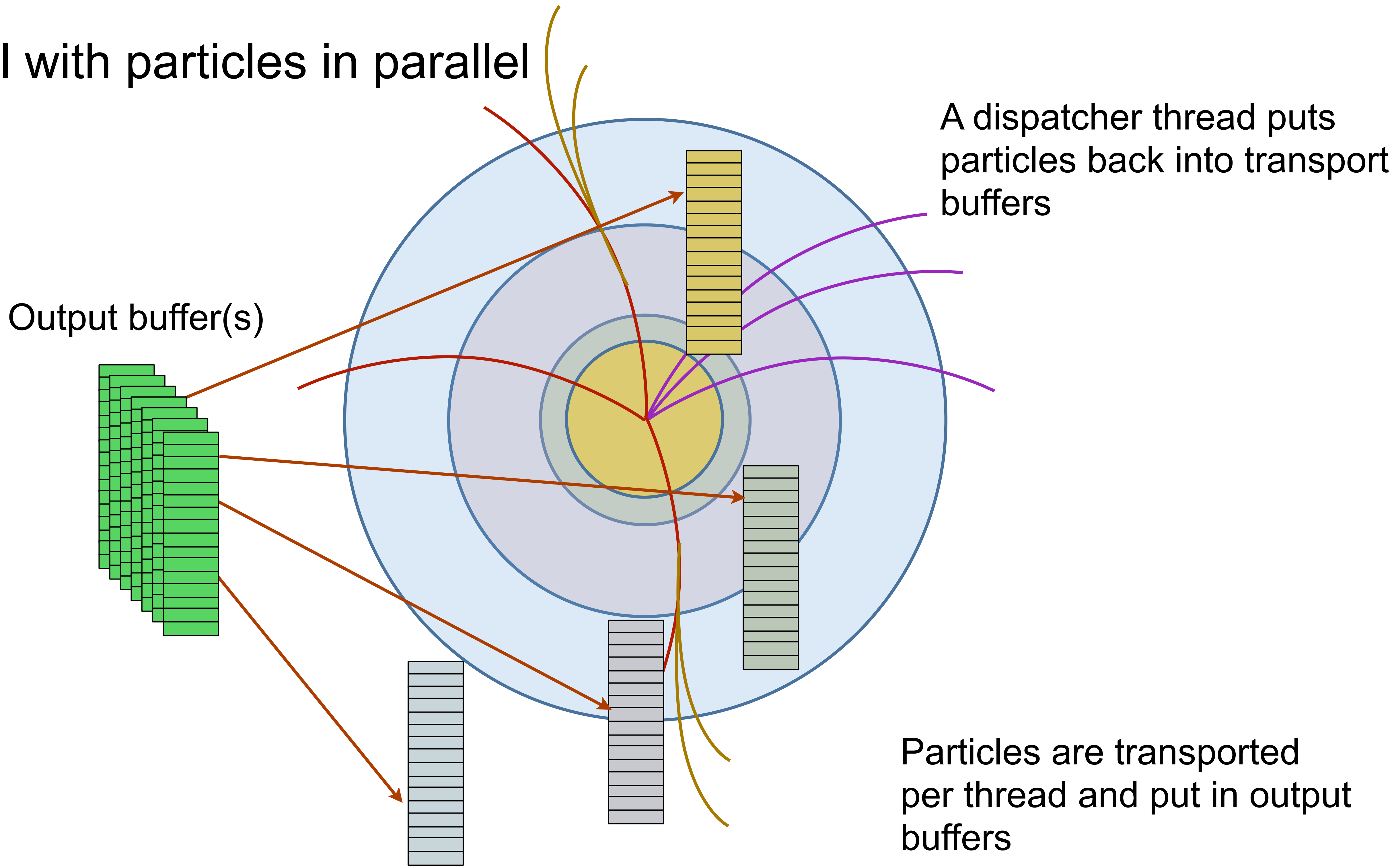
Deal with particles in parallel



Particles are transported per thread and put in output buffers

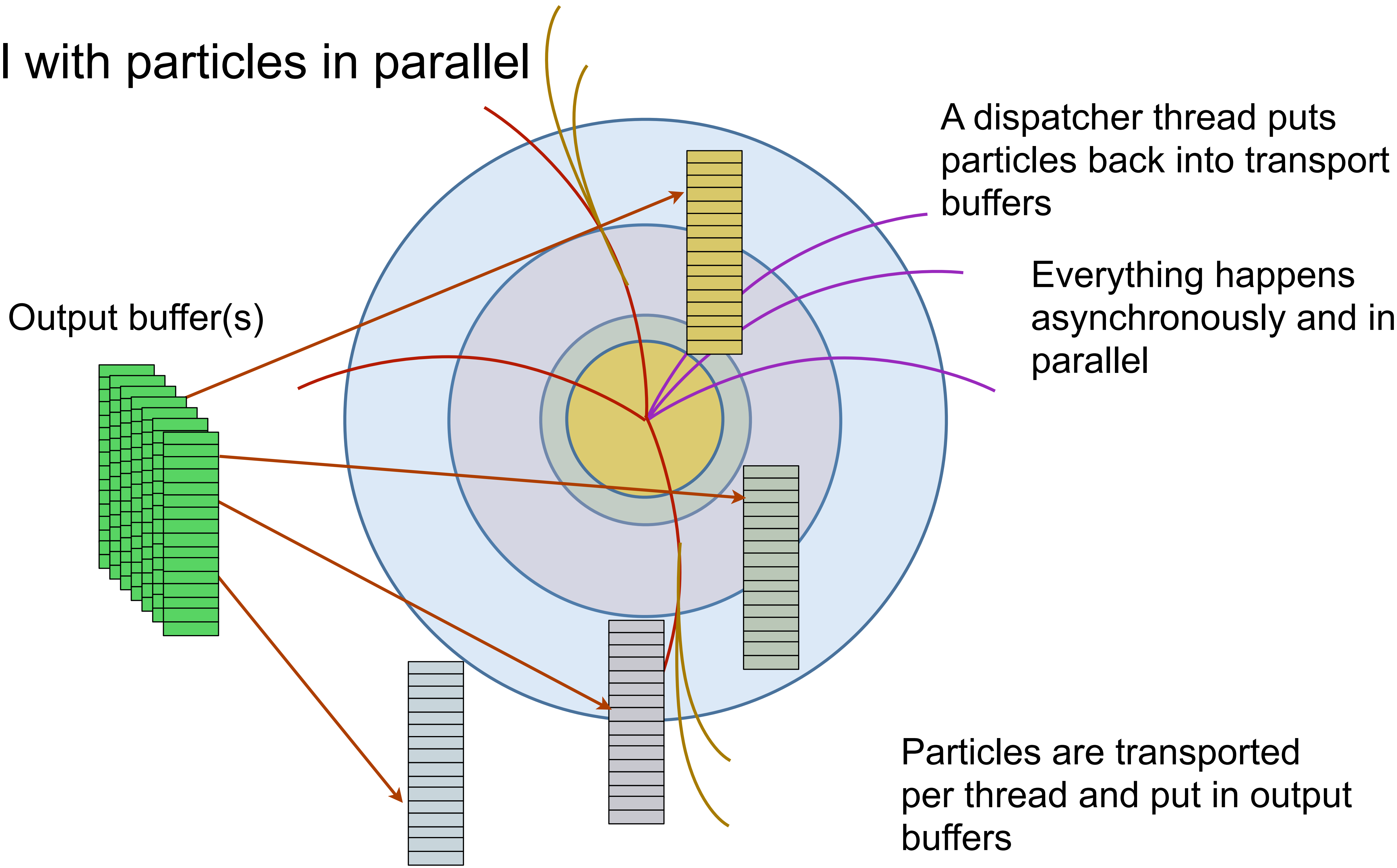
Introduced basketized transport

Deal with particles in parallel



Introduced basketized transport

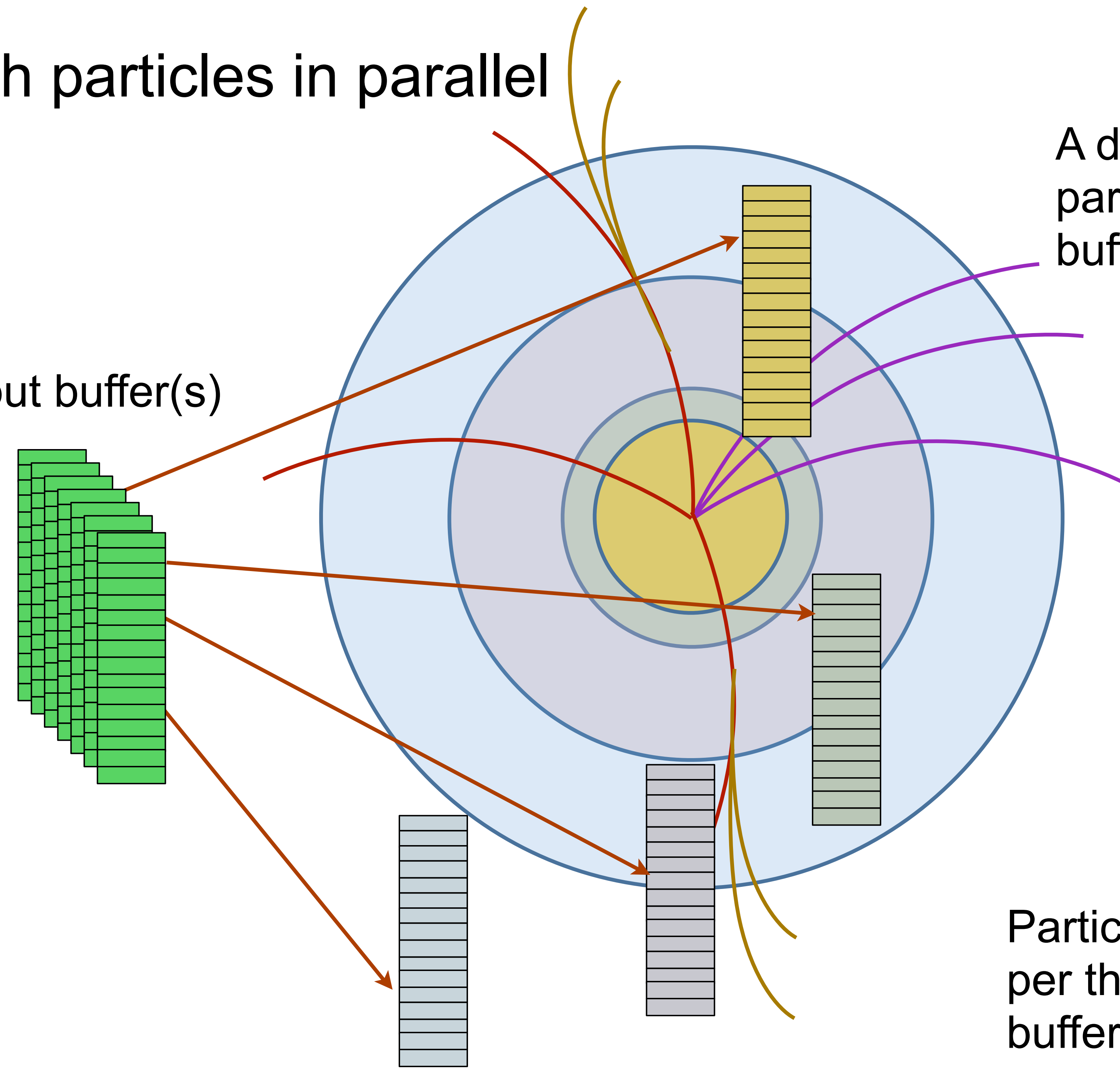
Deal with particles in parallel



Introduced basketized transport

Deal with particles in parallel

Output buffer(s)



A dispatcher thread puts particles back into transport buffers

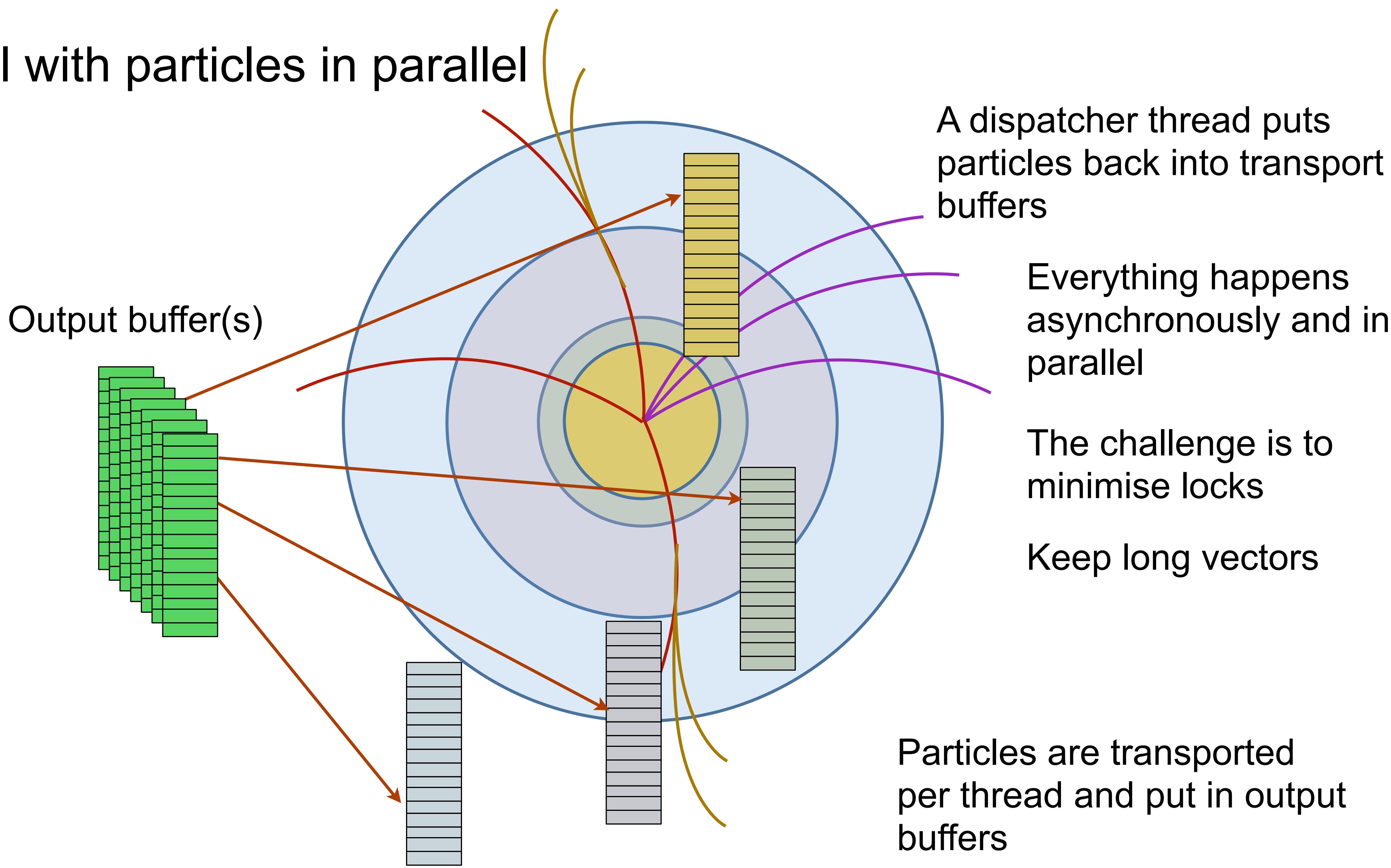
Everything happens asynchronously and in parallel

The challenge is to minimise locks

Particles are transported per thread and put in output buffers

Introduced basketized transport

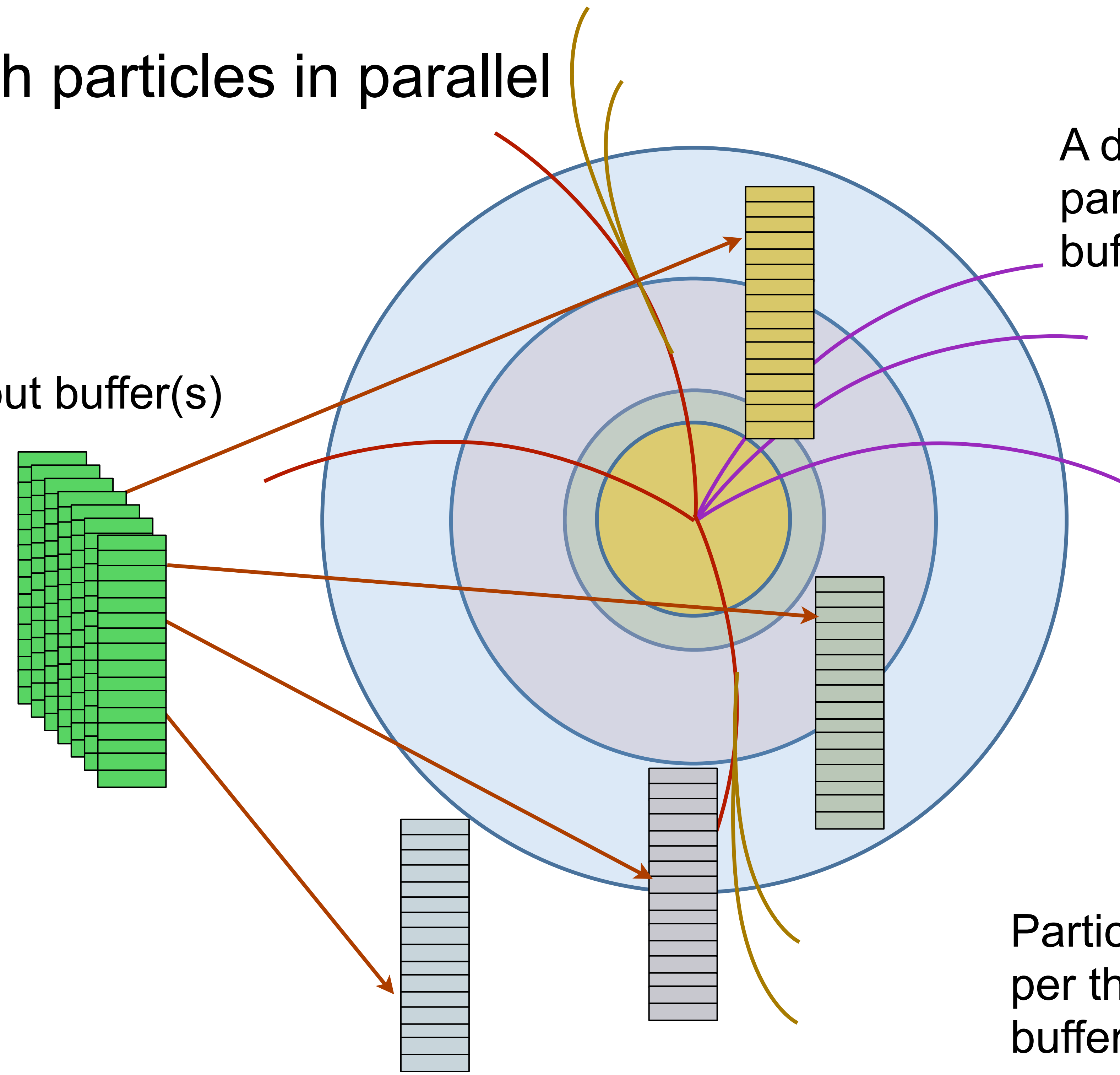
Deal with particles in parallel



Introduced basketized transport

Deal with particles in parallel

Output buffer(s)



A dispatcher thread puts particles back into transport buffers

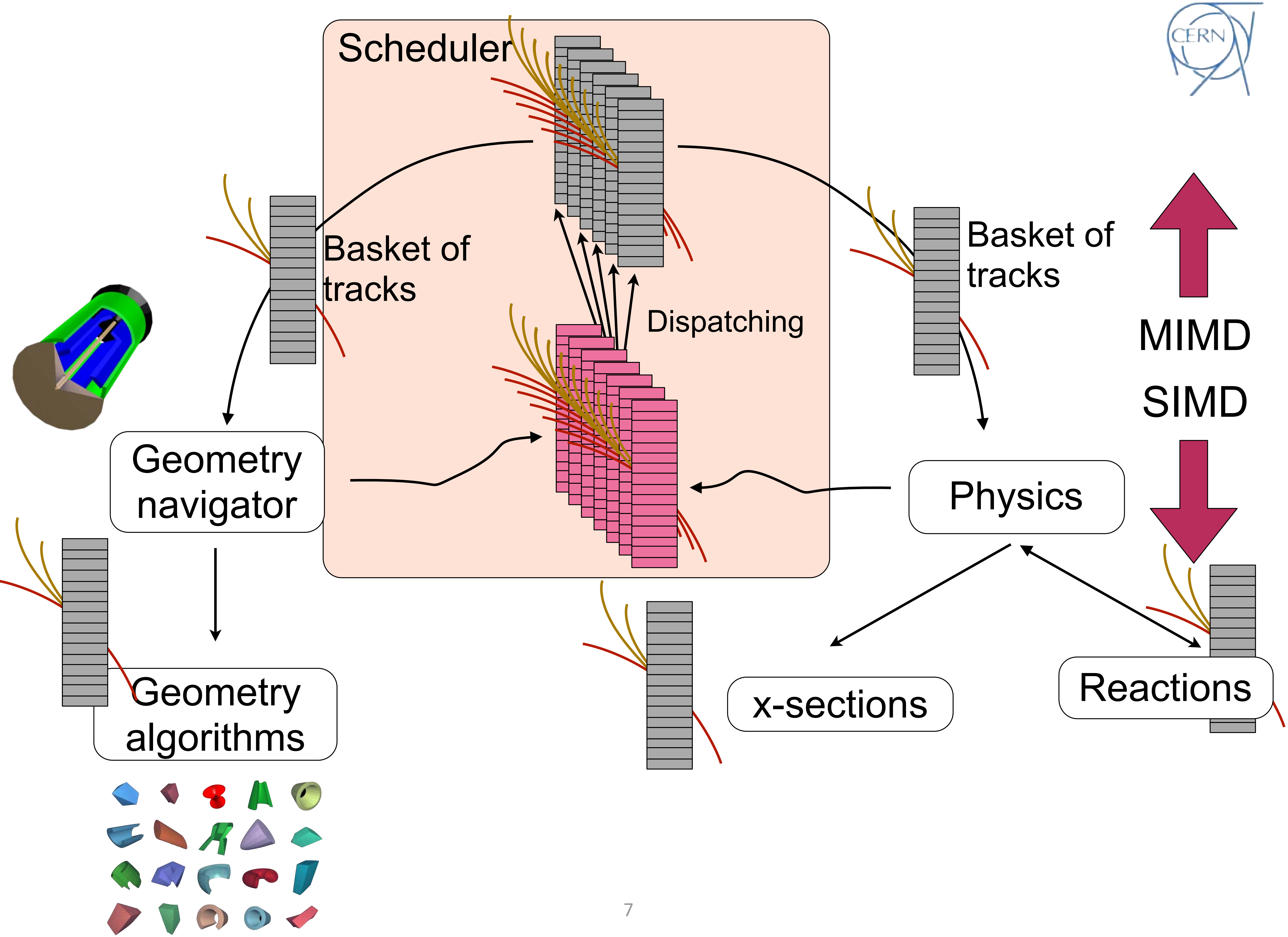
Everything happens asynchronously and in parallel

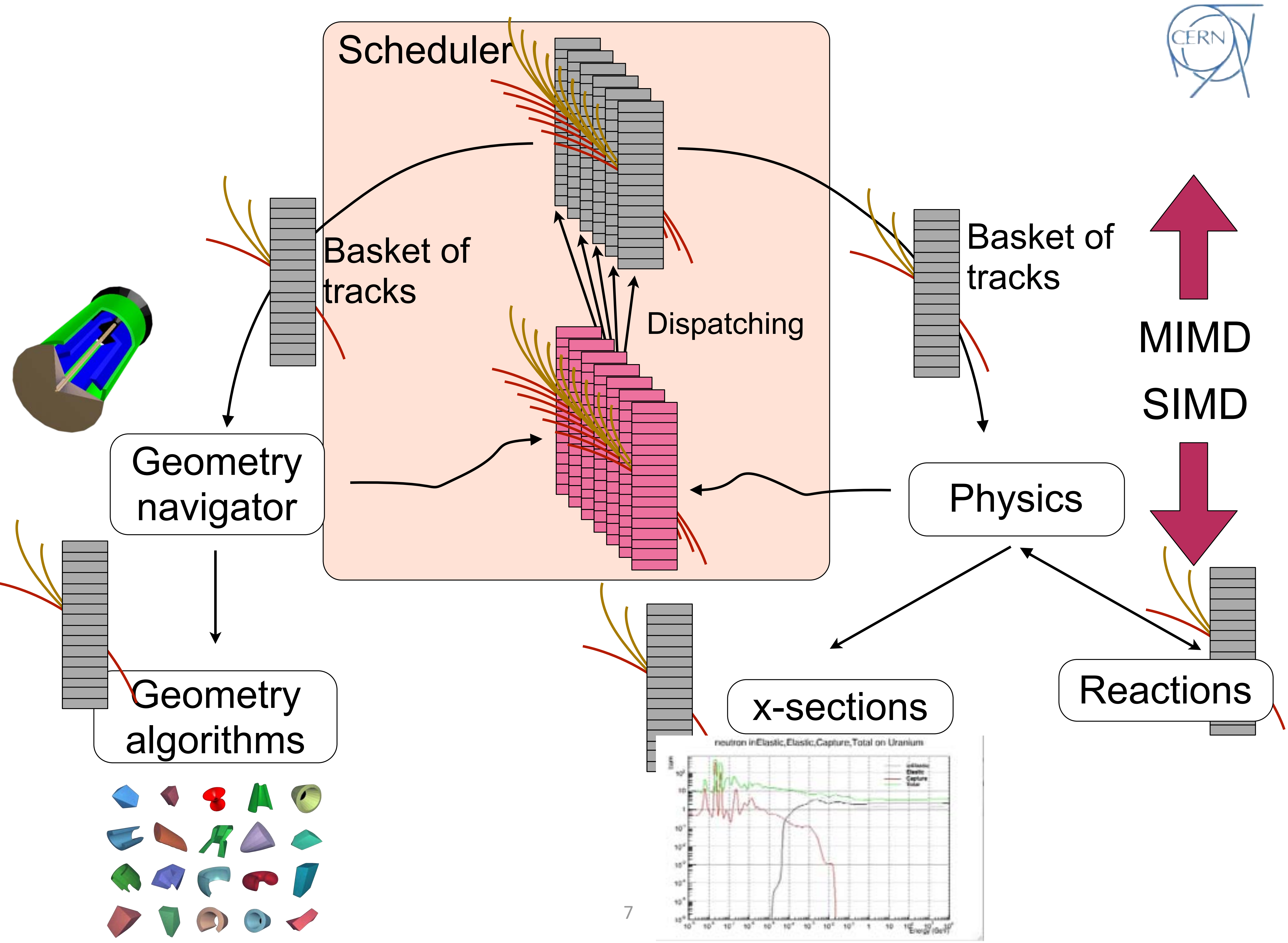
The challenge is to minimise locks

Keep long vectors

Avoid memory explosion

Particles are transported per thread and put in output buffers

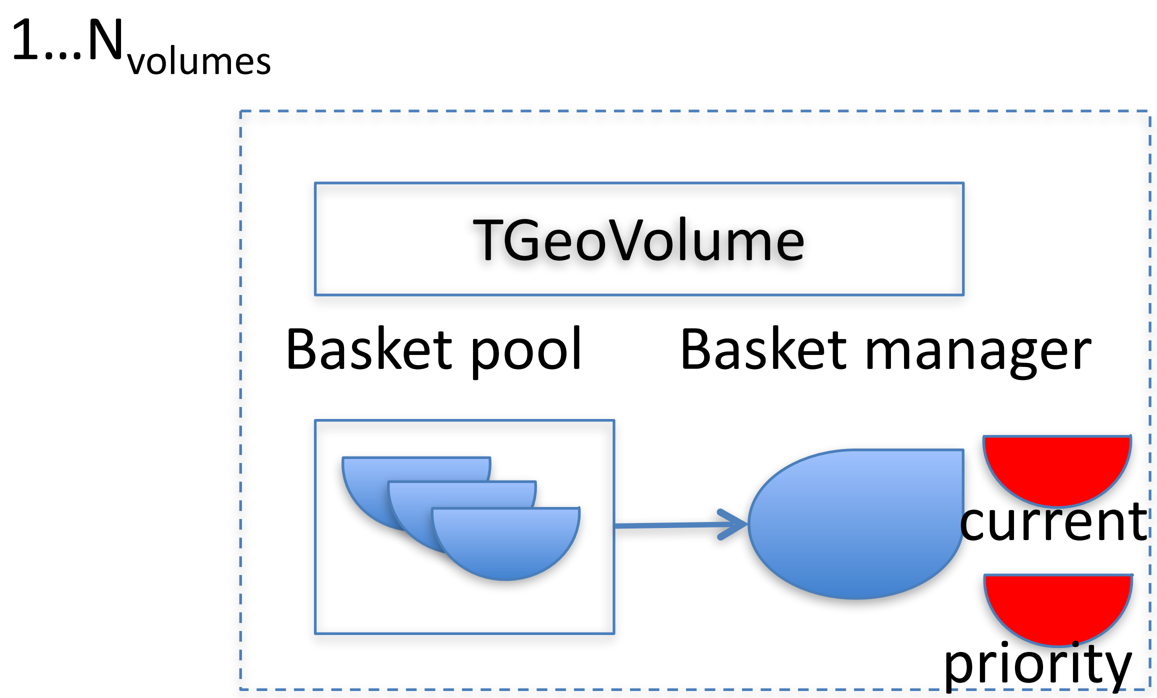

Scheduler
Basket of tracks
Dispatching
Basket of tracks
MIMD
SIMD
Geometry navigator
Physics
Geometry algorithms
x-sections
Reactions



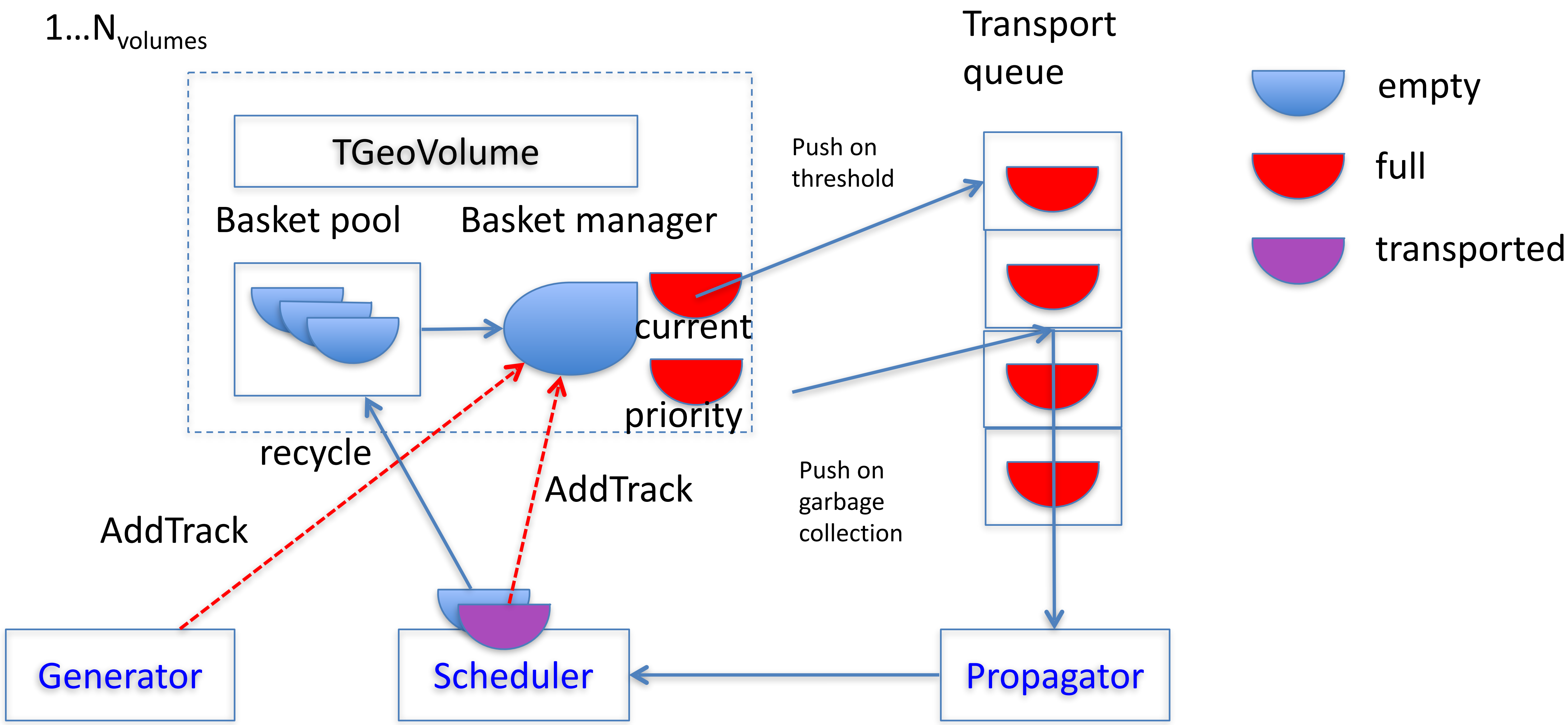
Scheduler
Basket of tracks
Dispatching
Basket of tracks
MIMD
SIMD
Geometry navigator
Physics
Geometry algorithms
x-sections
Reactions
neutron inElastic,Elastic,Capture,Total on Uranium
CERN

Basket managers

- One basket manager per volume
  - Receiving tracks entering the volume from generator or scheduler
  - Accessed by scheduler only
- Pool of empty baskets, one current basket + one basket for prioritized tracks
- Lock-free access for unique scheduler (only one thread can add tracks)
- Transportability threshold per manager
  - If threshold reached when adding tracks, the current basket is pushed in the work queue and replaced from the pool. Tracks added with the priority flag go to the priority basket which gets pushed to the priority side of the queue
  - Threshold(vol) = Ntracks_in_flight(vol)/2N_threads rounded to %4 (min 4, max 256)



Basket lifecycle

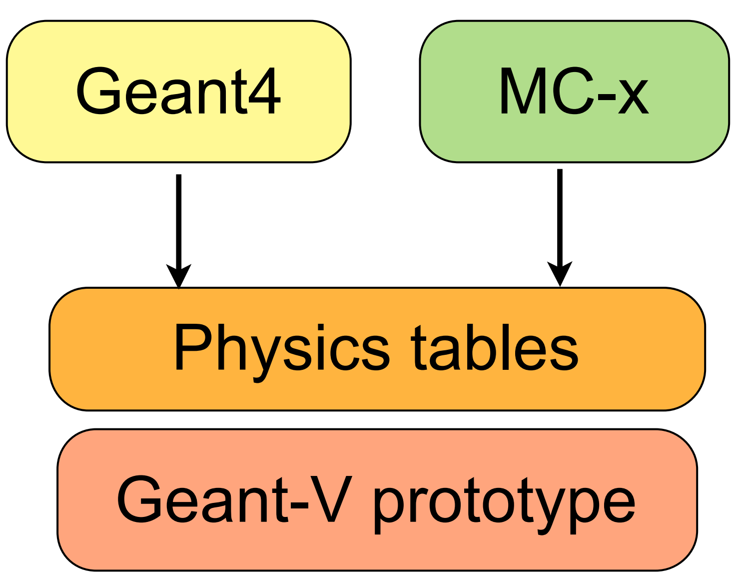


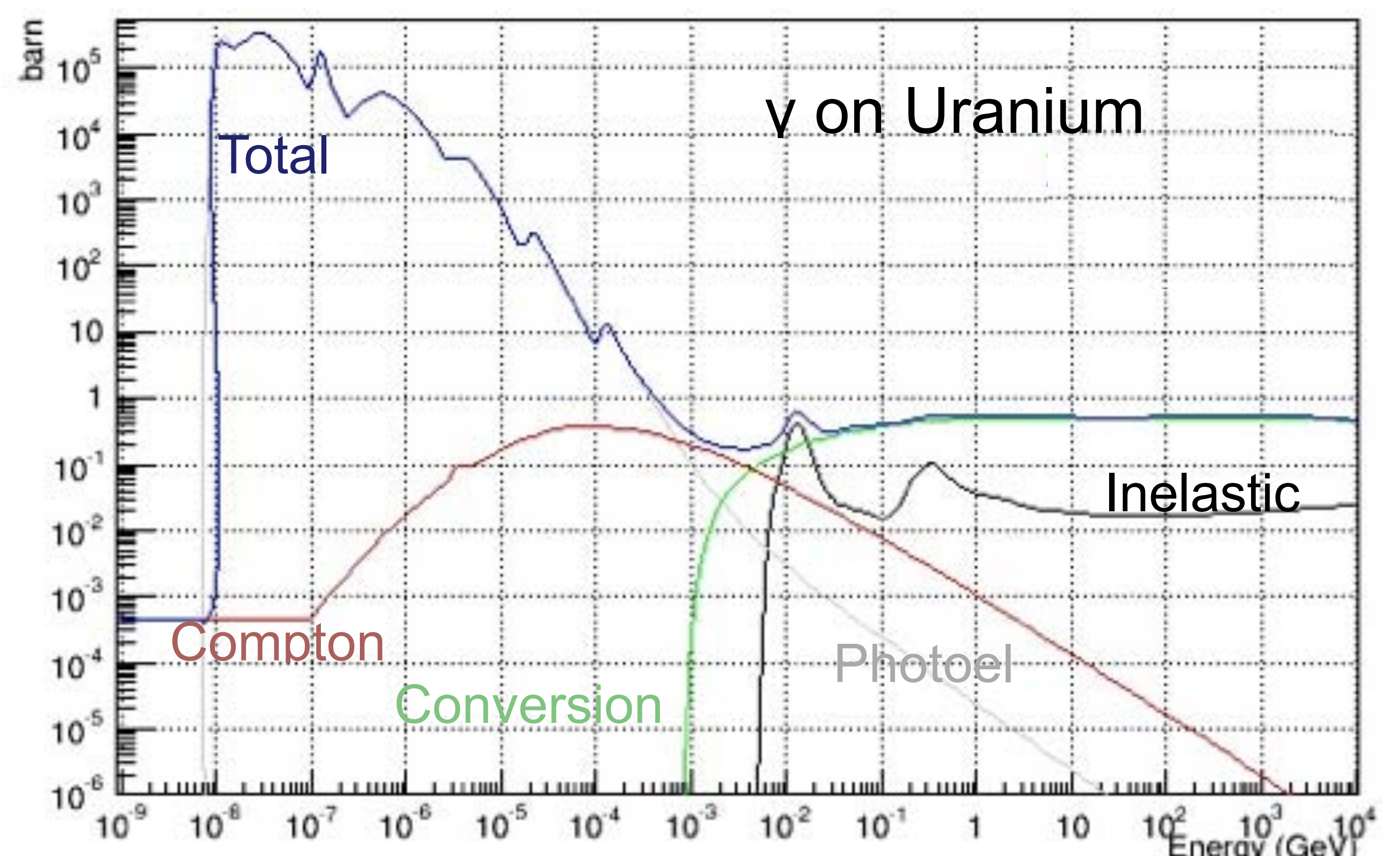
Scheduling policies

- Workload balancing
  - Divide the work evenly to scale with number of workers
  - Queue control: garbage collection on work queue depletion
  - Improvement: schedule physics as separate task (process selection and discrete processes post-step)
- Memory management
  - Not active currently, the idea it to trigger hit/digits collection and memory cleanup on thresholds
- Keep large vectors
  - Raise transportability thresholds per volume
  - Postpone sparse tracks when not in garbage collection mode
- Trigger single track mode when vectorization gives just overhead

Physics

- A lightweight physics for realistic shower development
- Select the major mechanisms
  - Bremsstrahlung, e+ annihilation, Compton, Decay, Delta ray, Elastic hadron, Inelastic hadron, Pair production, Photoelectric, Capture + dE/dx & MS
- Tabulate all x-secs (100 bins -> 90MB)
- Generate (10-50) final states (300kB per final state & element)
- Not good as Geant4, but it could be the seed of a fast simulation option
- Independent from the MonteCarlo that actually generates the tables



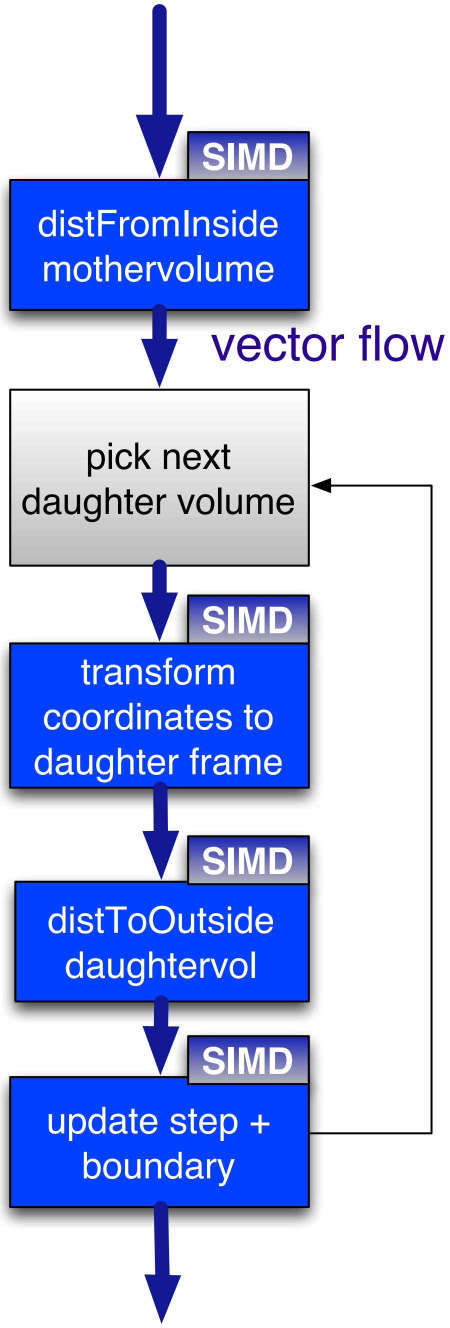


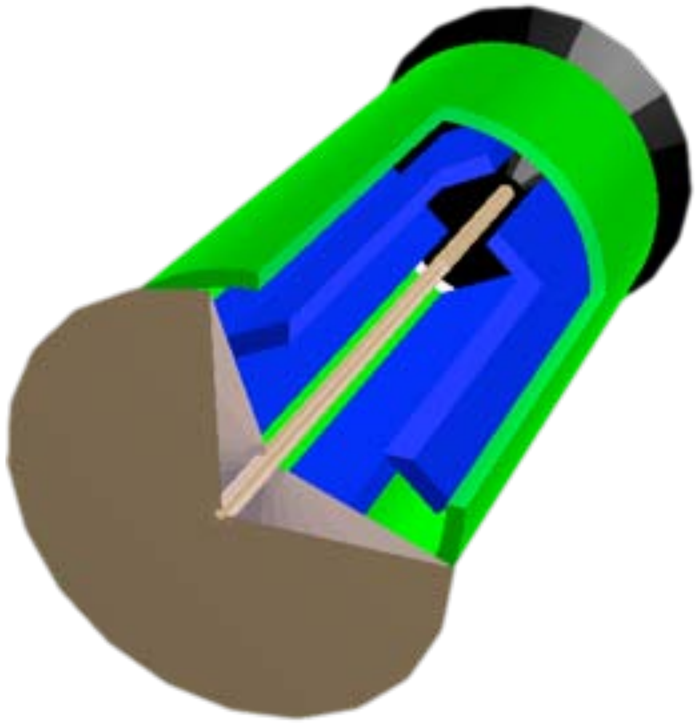
11

Recap of performance status

- provided new optimized vector interfaces for some elementary solids and geometric base classes (implemented important functions for particle navigation)

- overall performance gain in a standard navigation benchmark (in toy detector with 4 boxes, 3 tubes, 2 cones) - comparison to ROOT/ 5.34.17





	16 particles	1024 particles
Intel IvyBridge (AVX)	~2.8x	~4.0x
Intel Haswell (AVX2)	~3.0x	~5.0x
Intel Xeon-Phi (AVX512)	~4.1x	~4.8x

Xeon-Phi and Haswell benchmarks by CERN Openlab (Georgios Bitzes)

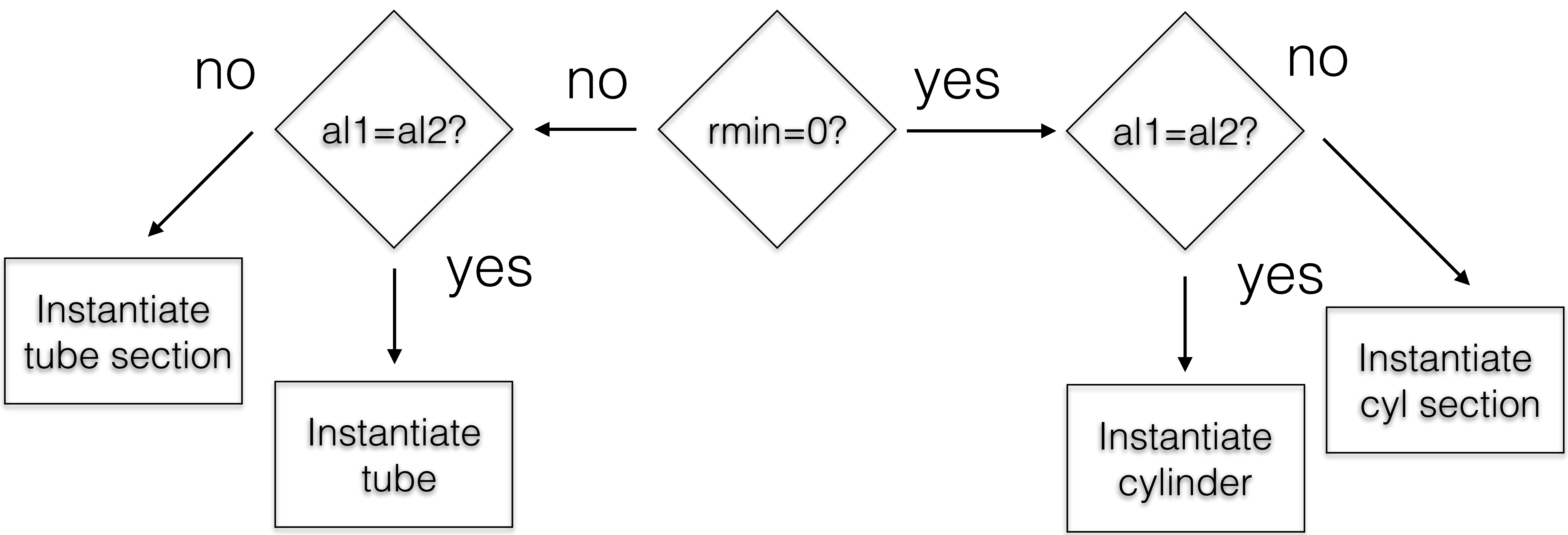
CHEP13 paper: http://arxiv.org/pdf/1312.0816.pdf

Portable HPC?

- Straight “vectorisation” of existing code is difficult to impossible
- Resulting code is hard to read and maintain
    - And it is largely compiler-dependency
- Porting to different high end devices is very difficult
- Explored solution is to use template specification for solid placement, specialisation and code generation
- Highly optimised modular “codelets” à la STL are used to construct algorithms

Solid specialisation

CreateTube(rmin, rmax, al1, al2)



Illustrating scalar/SIMD abstraction and kernels

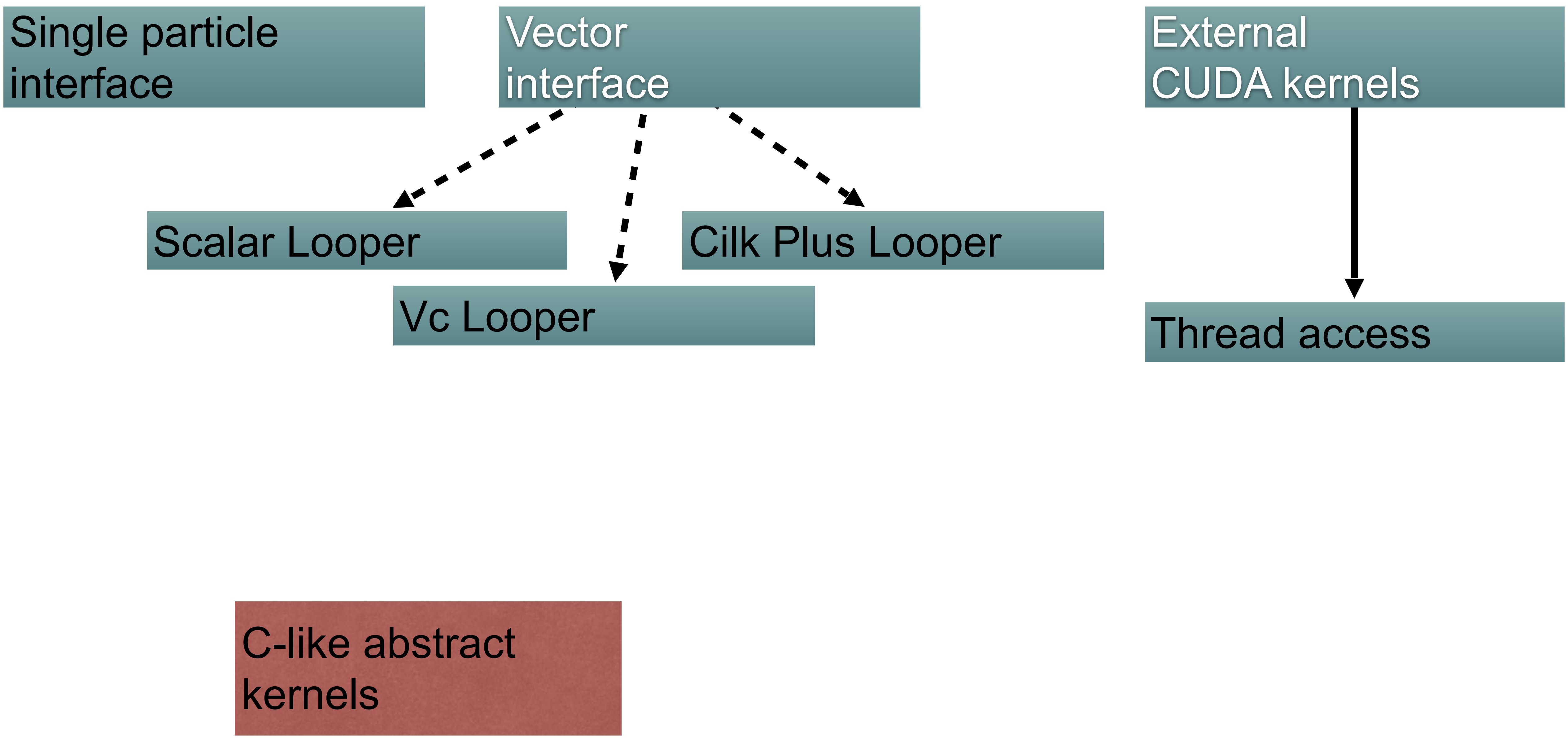
Single particle interface

Vector interface

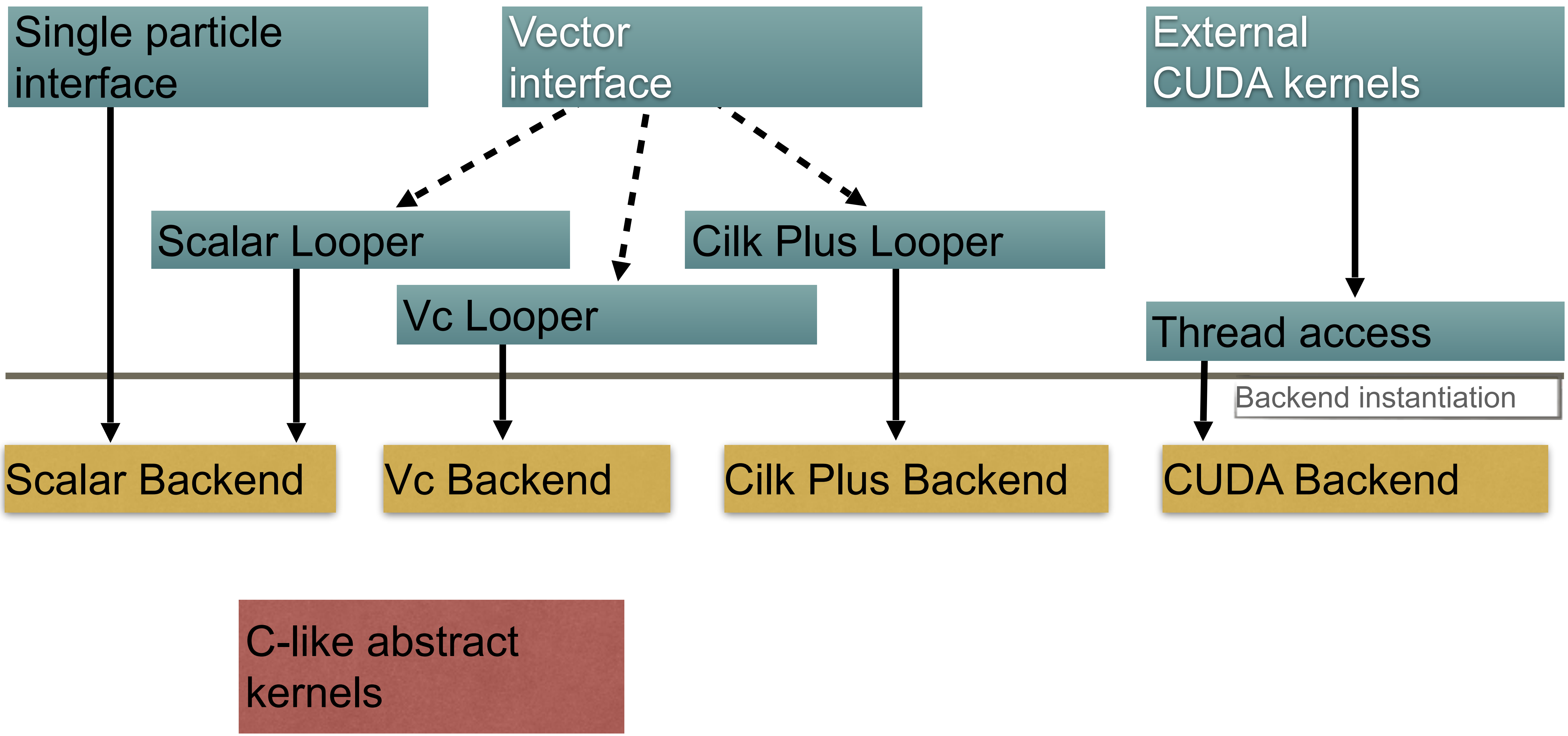
External CUDA kernels

C-like abstract kernels

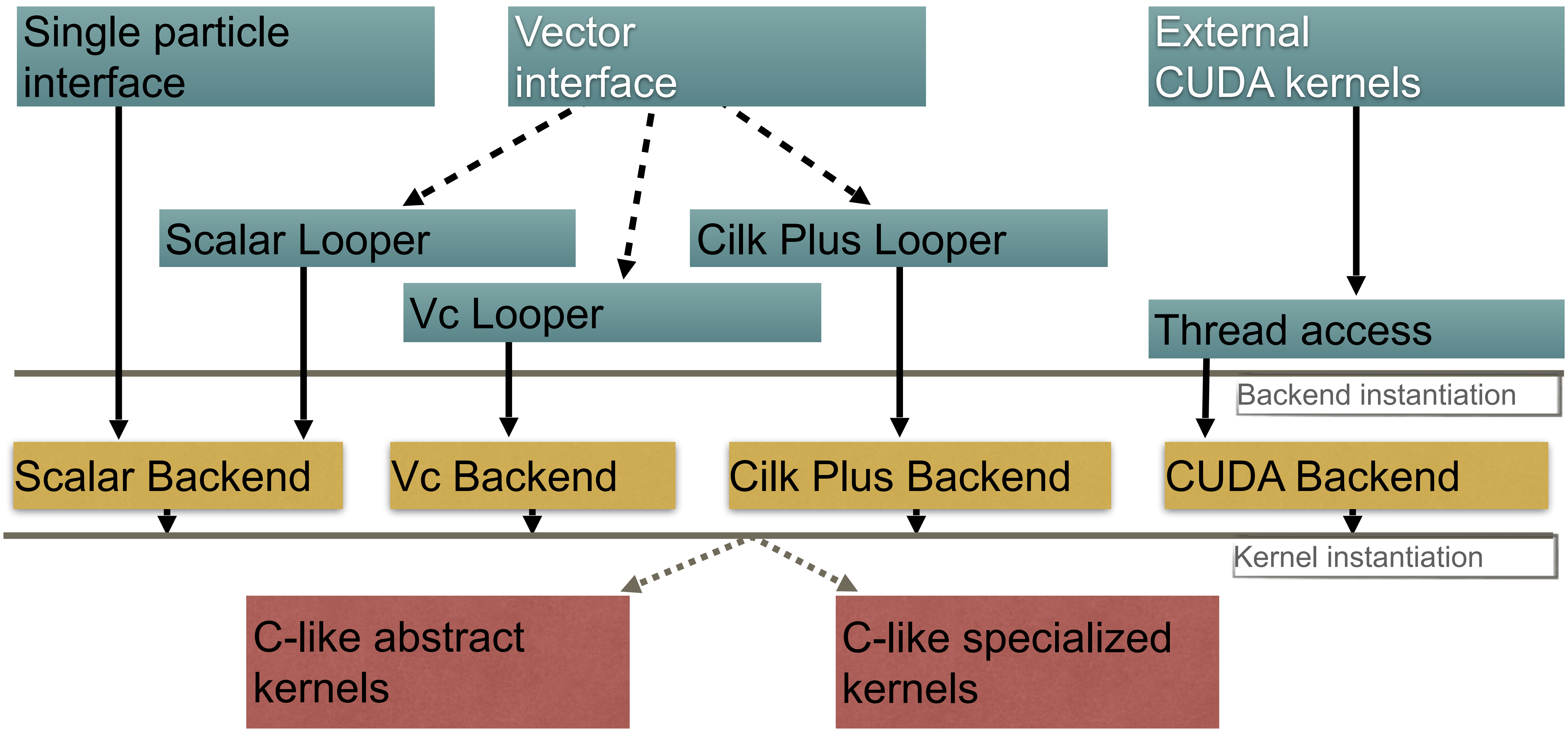
Illustrating scalar/SIMD abstraction and kernels



Illustrating scalar/SIMD abstraction and kernels

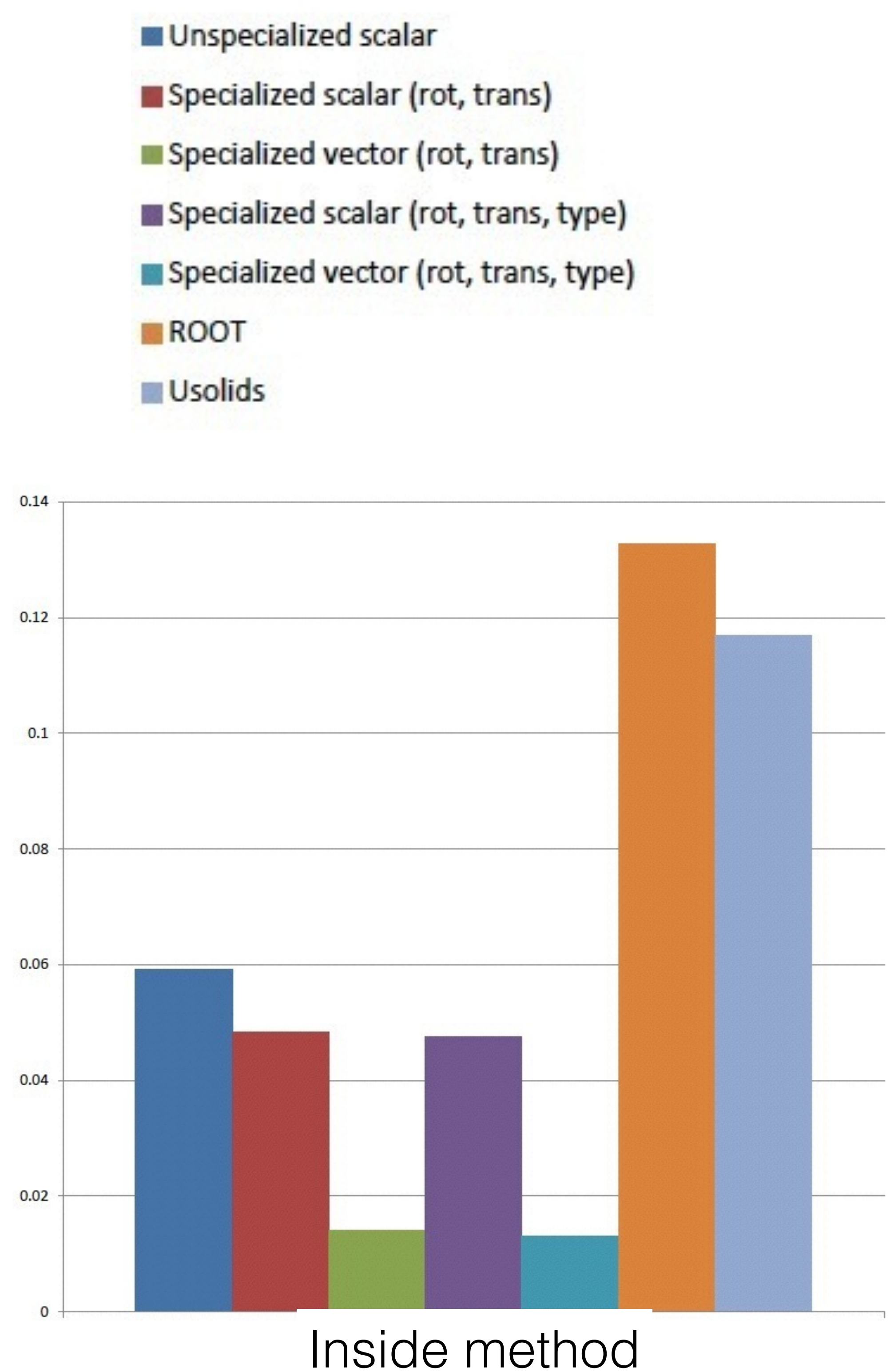


Illustrating scalar/SIMD abstraction and kernels



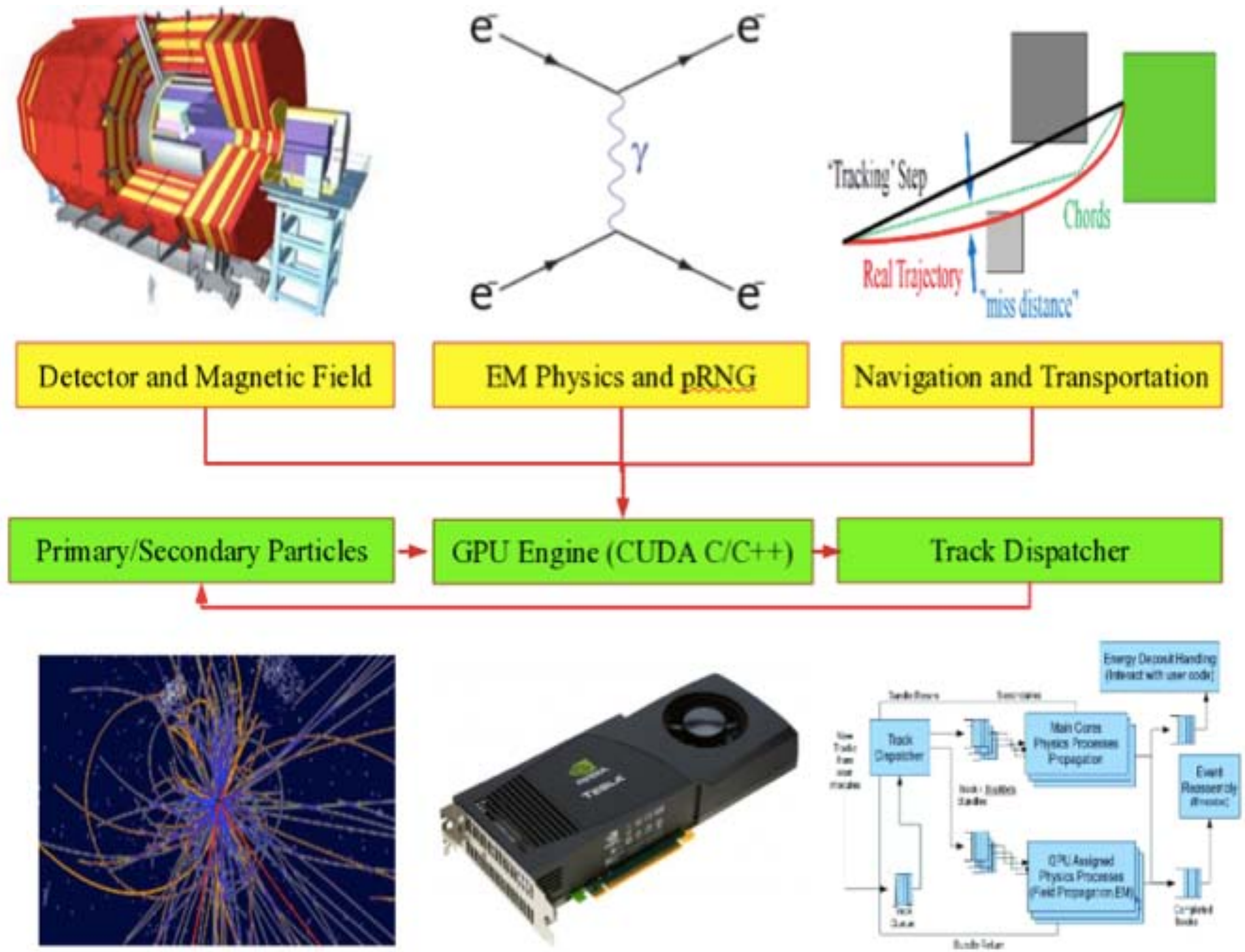
Performance examples

- Performance of the inside method for a tube on an Intel CPU
- As you can see the performance is improved even for the scalar version



Particle Transport on the GPU

GPU Prototype: Three Core Components

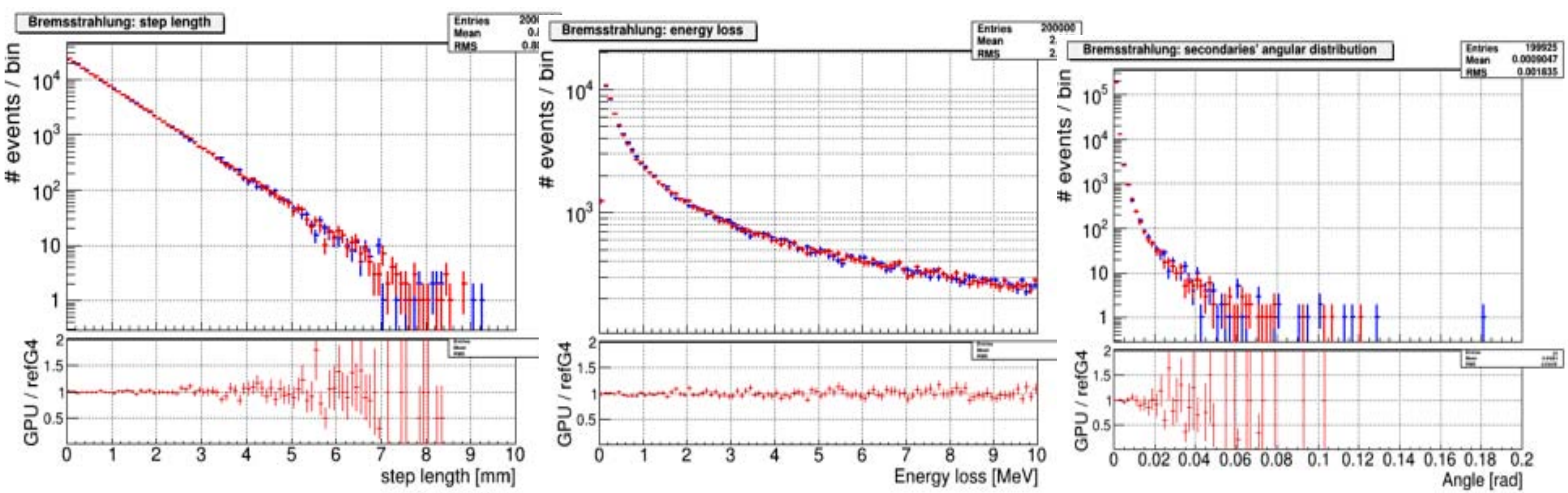


Fermilab

Physics Validation of GPU Physics

- Compare simulated physics outputs

RED BLUE



Performance Evaluation

- Hardware (host + device)

	Host (CPU)	Device (GPU)
M2090	AMD Opteron™ 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

Performance: Realistic Simulation

- A simple calorimeter (CMS Ecal) with the CMS b-field map
- Tracking for one step: split kernels (GPIL+sorting+DoIt)

	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)*

- Performance by each kernel (% of the total application time)

Fermilab

Performance Issues and Considerations

- Observed issues (by the Nvidia profiler)

- Considerations

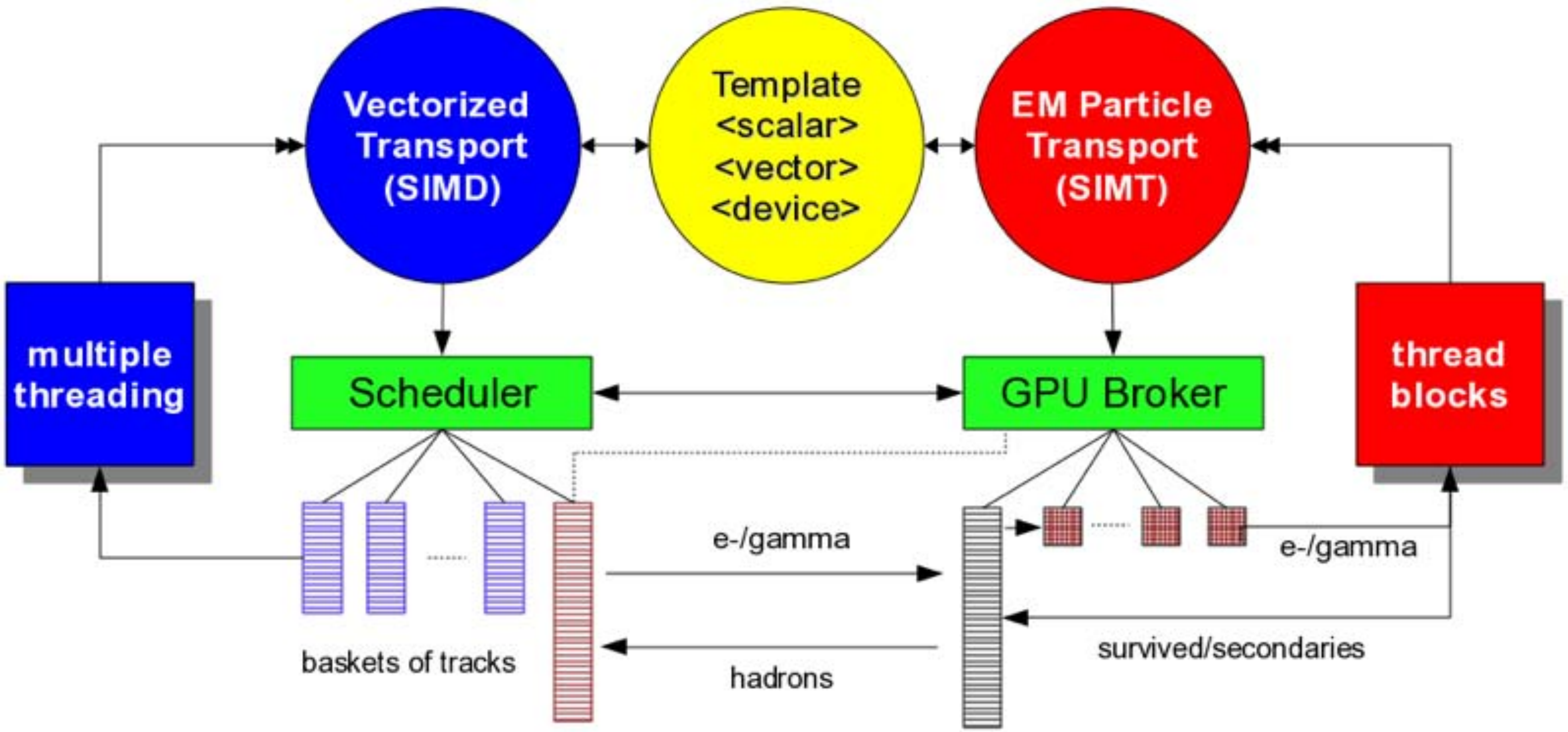
Challenges and New Strategies

- HEP detector simulation (ex. Geant4) is a giant
- Coprocessor architectures
- Top-down approach

Incorporating into the Vector prototype

- The vector prototype started at CERN (talk by F. Carminati)

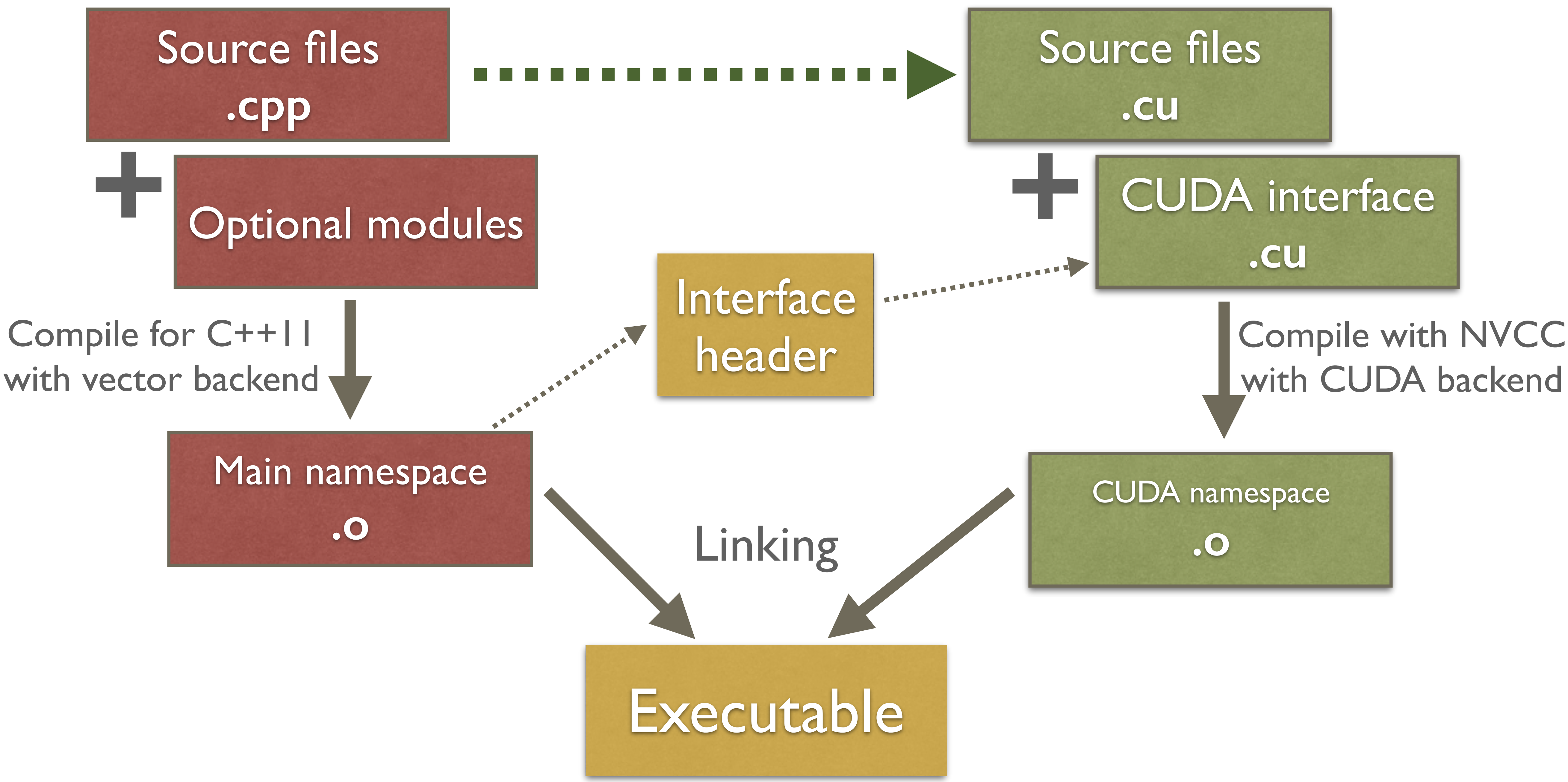
- Integration to the vector prototype (GPU broker)



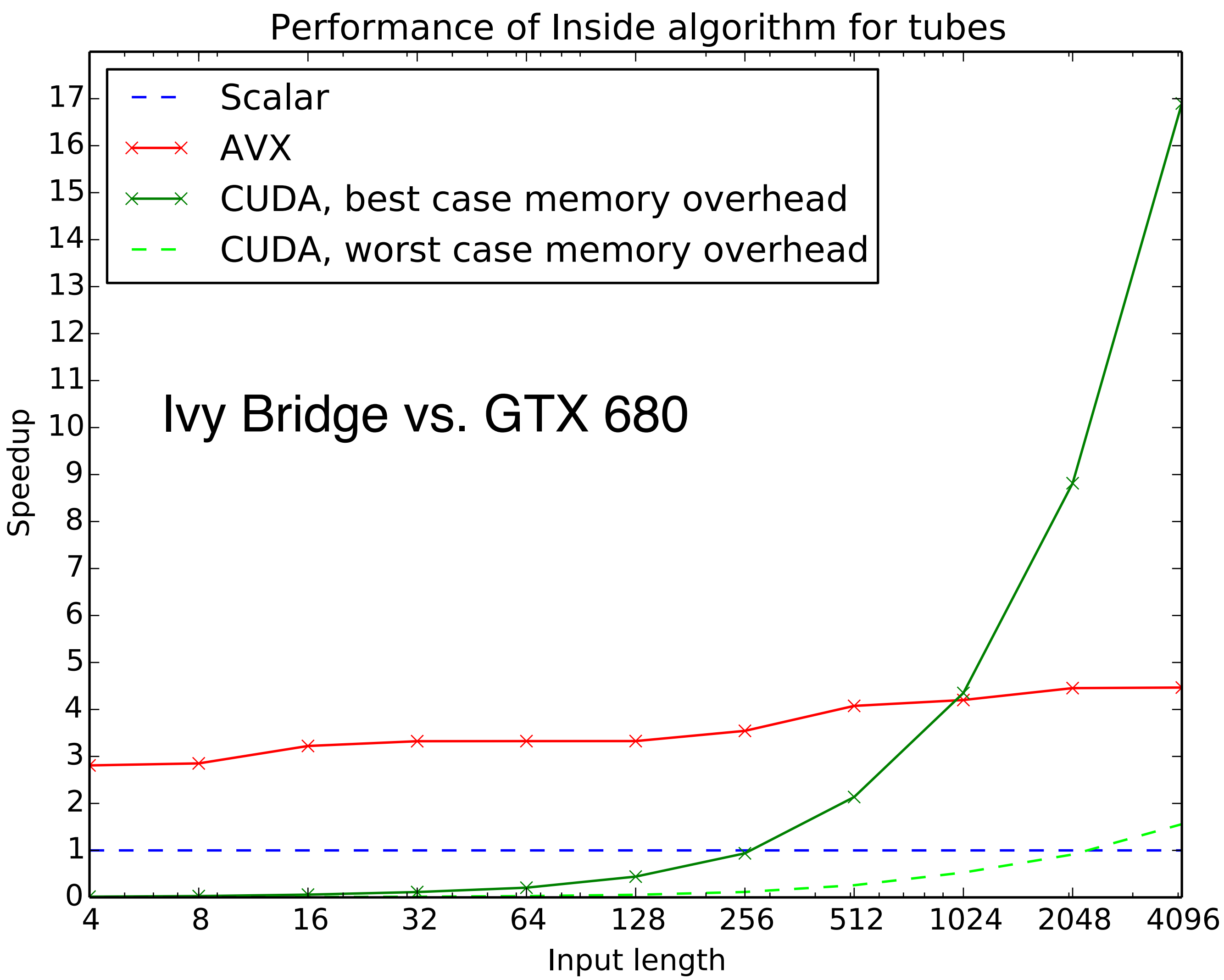
CUDA in GeantV geometry

- Enable dispatching to CUDA as a co-processor
- Run separate to scalar and vectorized code
- Use same codebase
- Achieved by abstracted, templated algorithms

Separate compilation of backends



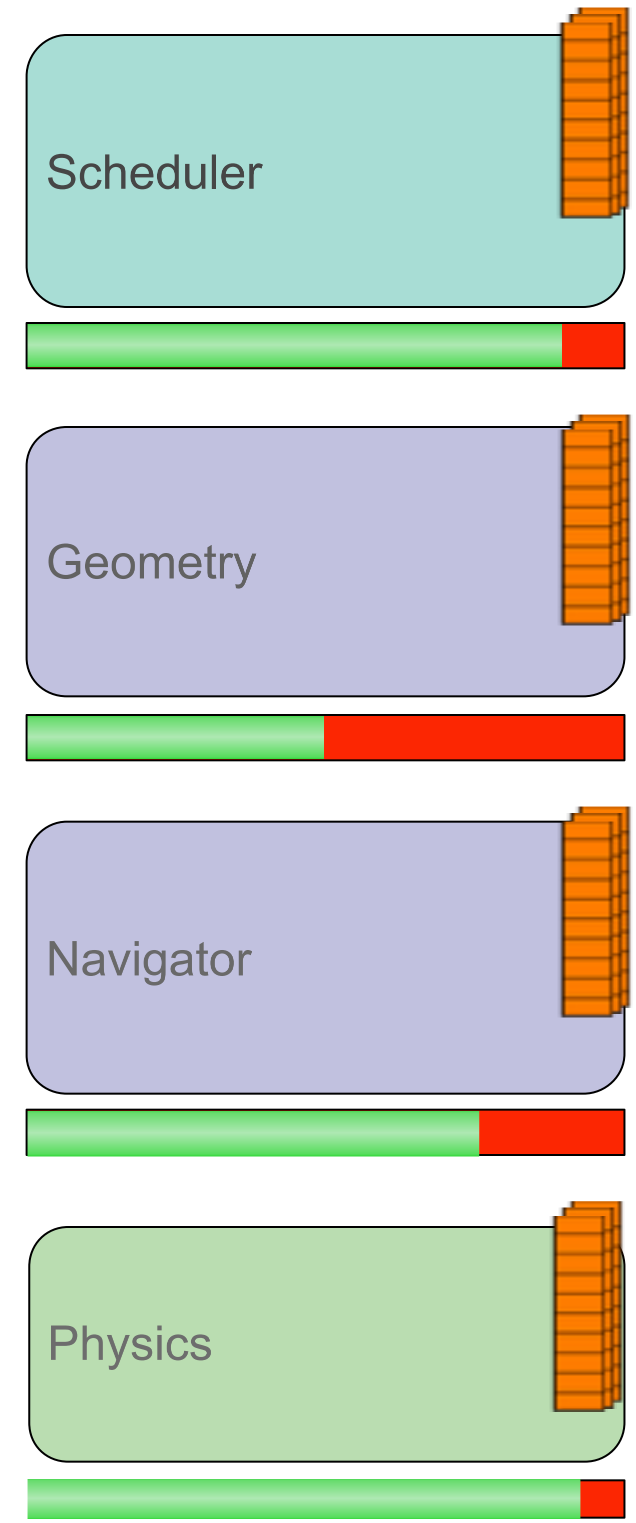
Shape measurements



- Scalar, vector and CUDA code templated from <u>same</u> abstracted algorithm
- Dispatch on the fly to optimal processor
- Typical GPU scaling; high minimum input threshold

Where are we now?

- Scheduler
  - The new version, hopefully improved of the scheduler has been committed and we are testing it
- Solids
  - The proof or principle that we can achieve large speedups (3-5+) is there, however a lot of work lays ahead
- Navigator
  - “Percolating” vectors through the navigator is a difficult business. We have a simplified navigator that achieves that, but more work is needed here
- Physics
  - Can generate x-secs and final states and sample them, but there are still many points to be clarified with Geant4 experts



Testing - benchmarking

Geant4

G4 nav

G4 phys

G4 geom

Prototype

Prot nav

Tab phys

UGeom*

With small, large & venti geometries

*UGeom == USolid + navigation

Testing - benchmarking

Geant4

G4 nav	G4 nav
G4 phys	Tab phys
G4 geom	G4 geom

Prototype

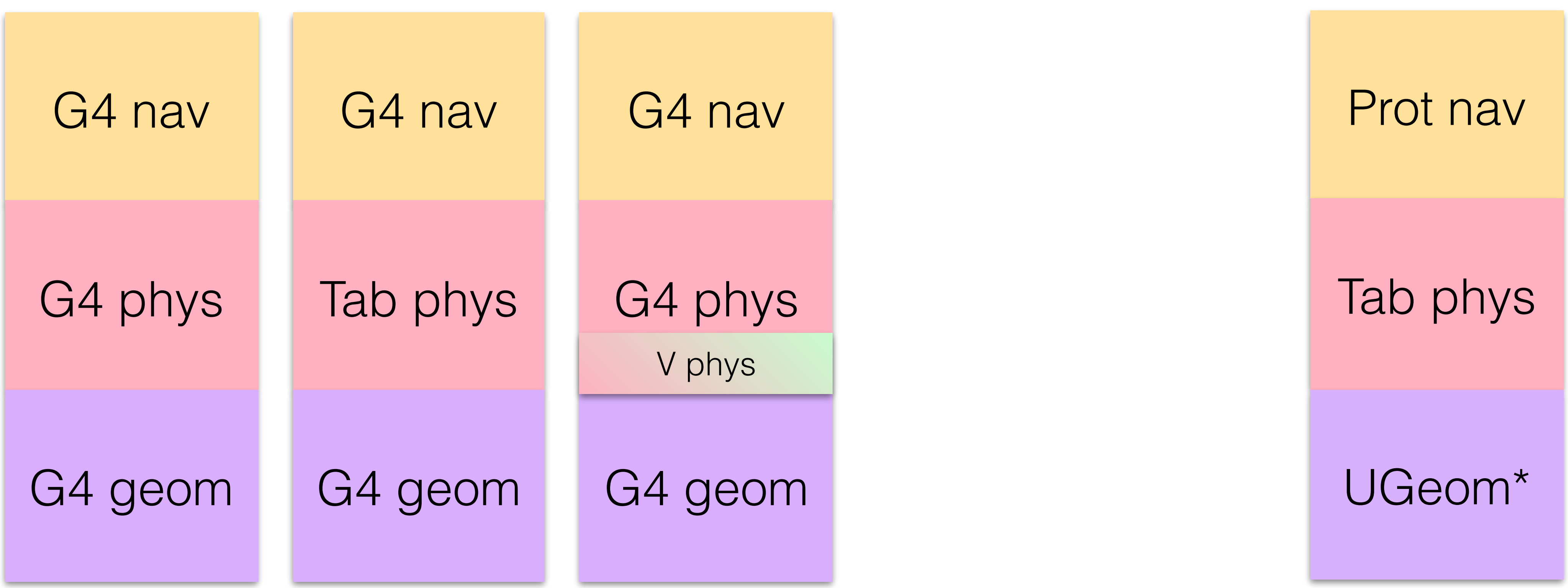
Prot nav
Tab phys
UGeom*

With small, large & venti geometries

*UGeom == USolid + navigation

Testing - benchmarking

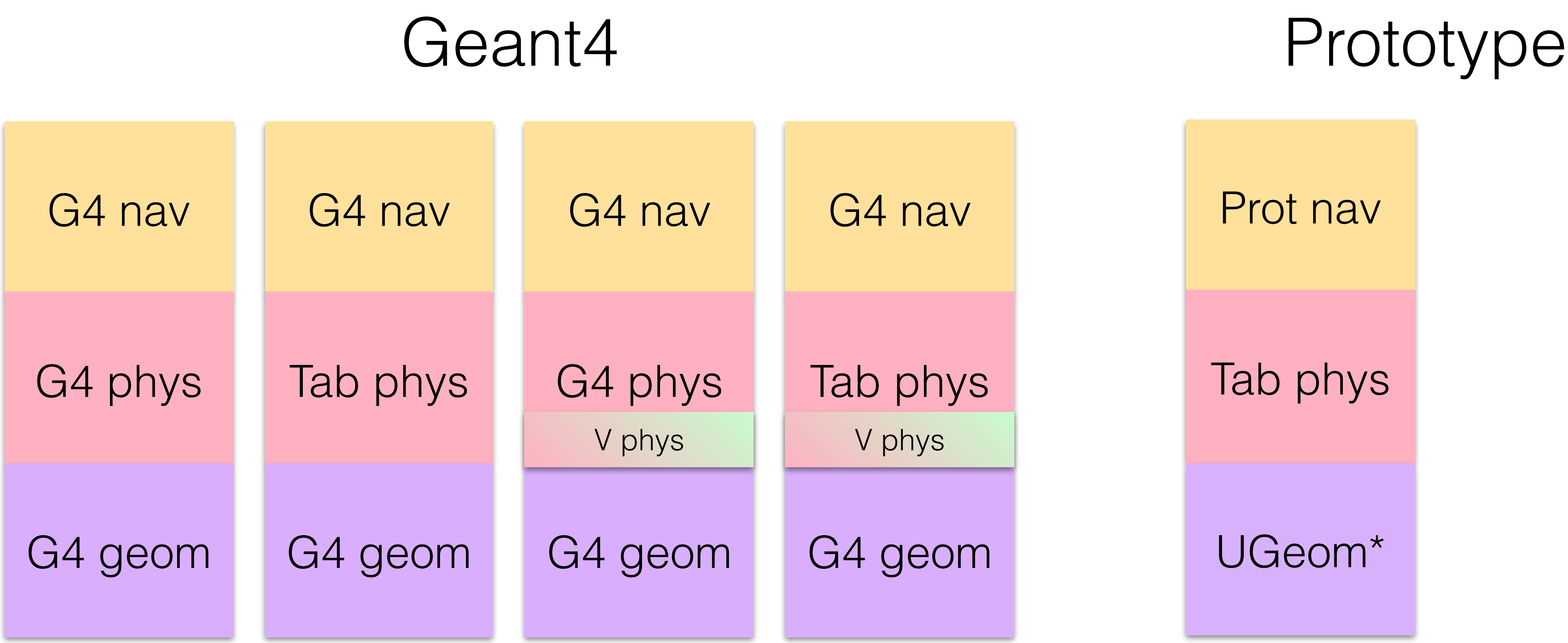
Geant4 Prototype



With small, large & venti geometries

*UGeom == USolid + navigation

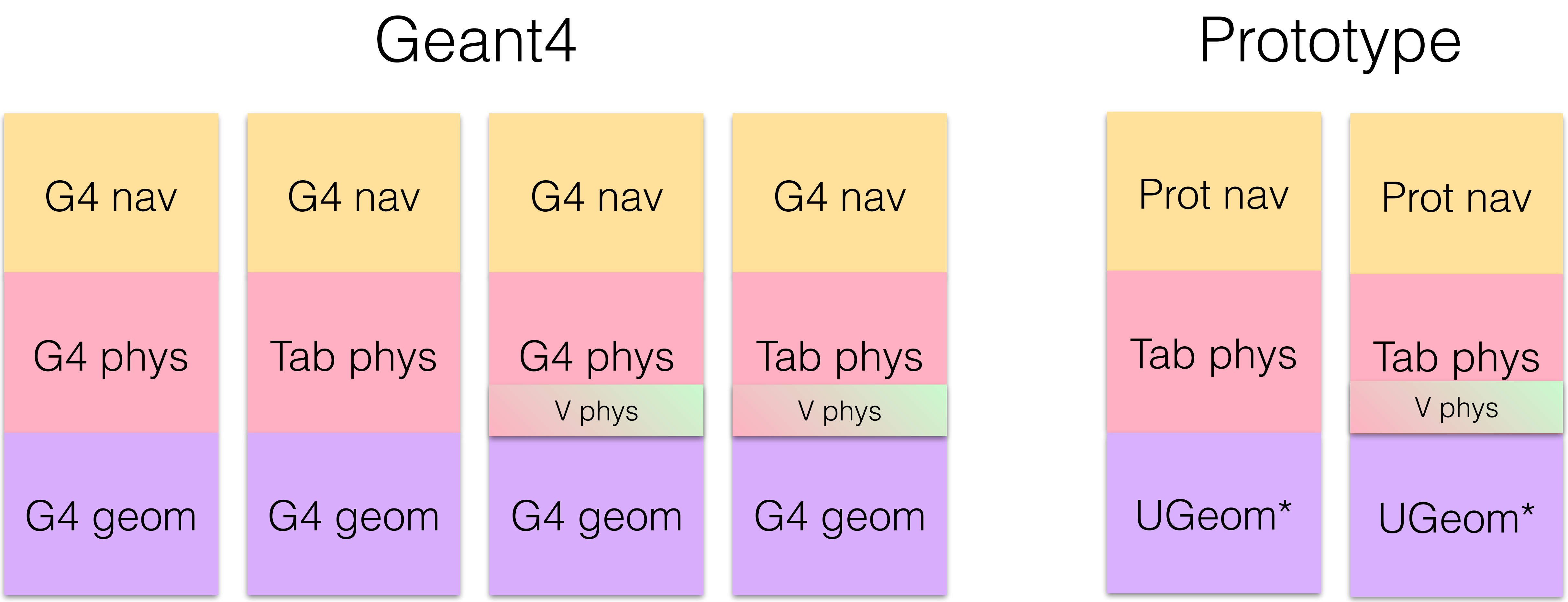
Testing - benchmarking



With small, large & venti geometries

*UGeom == USolid + navigation

Testing - benchmarking



With small, large & venti geometries

*UGeom == USolid + navigation

Conclusion

- Work on a full prototype is progressing
- The first performance figures are very encouraging
- The template specialisation technique seems to provide a reasonable model for portable HPC
- We hope to have a demonstrator by the end of the year