The path toward HEP High Performance Computing

(with a closer look at simulation)

CHEP 2013

October 17, 2013

J.Apostolakis, R.Brun, F.Carminati, A.Gheata, S.Wenzel

Acknowledgements

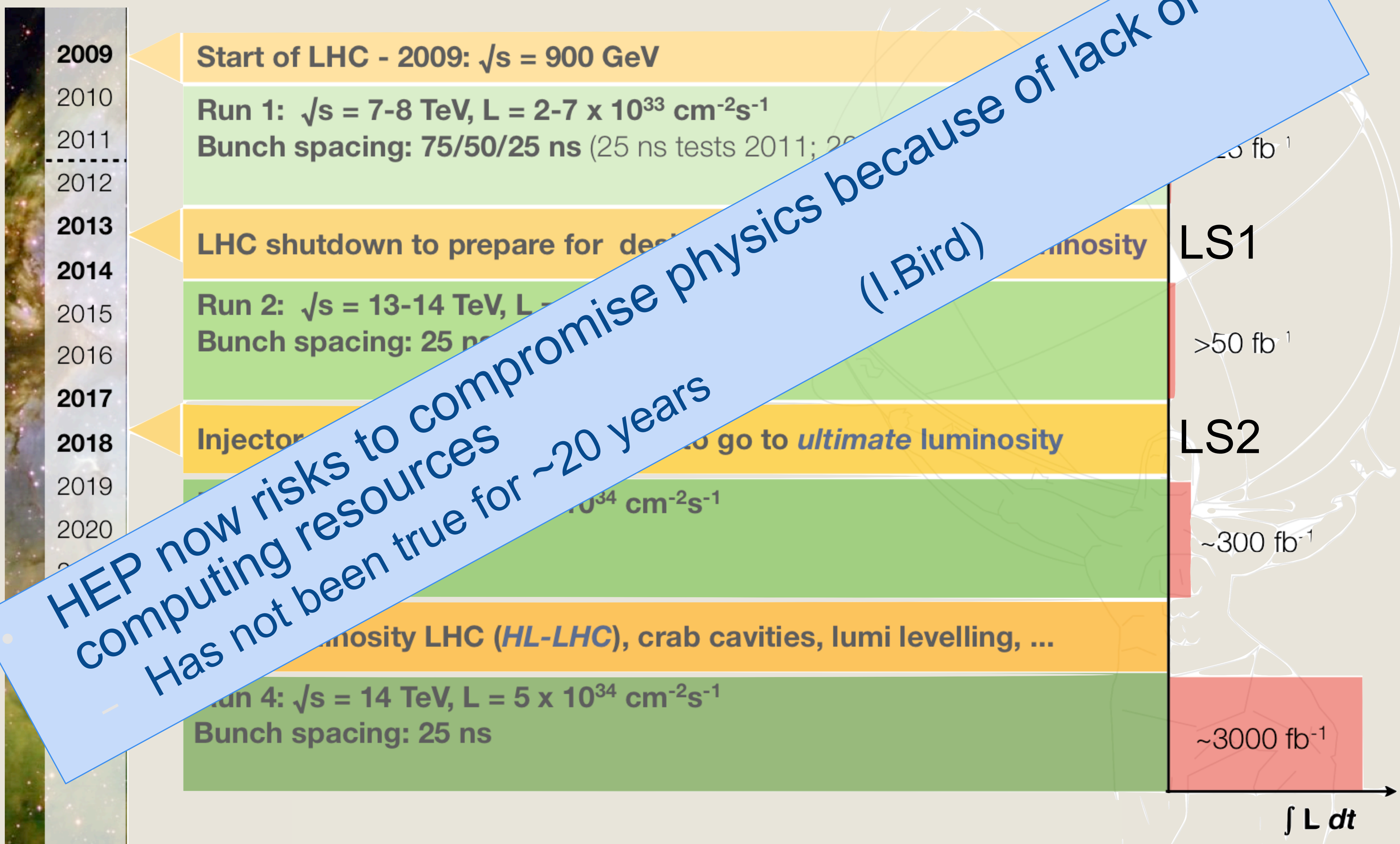
- M.Bandieramonte (Catania Univ.)
- L.Durhem (Intel)
- A.Nowak (CERN OpenLab)
- R.Seghal (BARC)

A luminous future for HEP...

		$\int L\,dt$
2009	**Start of LHC - 2009: √s = 900 GeV**	
2010 2011 2012	**Run 1: √s = 7-8 TeV, L = 2-7 x 10^{33} $cm^{-2}s^{-1}$** **Bunch spacing: 75/50/25 ns** (25 ns tests 2011; 2012)	~25 fb^{-1}
2013 **2014**	**LHC shutdown to prepare for design energy and *nominal* luminosity**	LS1
2015 2016 **2017**	**Run 2: √s = 13-14 TeV, L = 1 x 10^{34} $cm^{-2}s^{-1}$** **Bunch spacing: 25 ns**	>50 fb^{-1}
2018	**Injector and LHC Phase-I upgrade to go to *ultimate* luminosity**	LS2
2019 2020 2021	**Run 3: √s = 14 TeV, L = 2 x 10^{34} $cm^{-2}s^{-1}$** **Bunch spacing: 25 ns**	~300 fb^{-1}
2022	**High-luminosity LHC (*HL-LHC*), crab cavities, lumi levelling, ...**	
2023 ... **2030**	**Run 4: √s = 14 TeV, L = 5 x 10^{34} $cm^{-2}s^{-1}$** **Bunch spacing: 25 ns**	~3000 fb^{-1}

∫ L *dt*

A luminous future for HEP...



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

Has not been true for ~20 years

(I.Bird)

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 THREADS
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 THREADS
THEORY	8	32	43.2
OPTIMISED	6	9.43	11.79
HEP	1	0.8	1

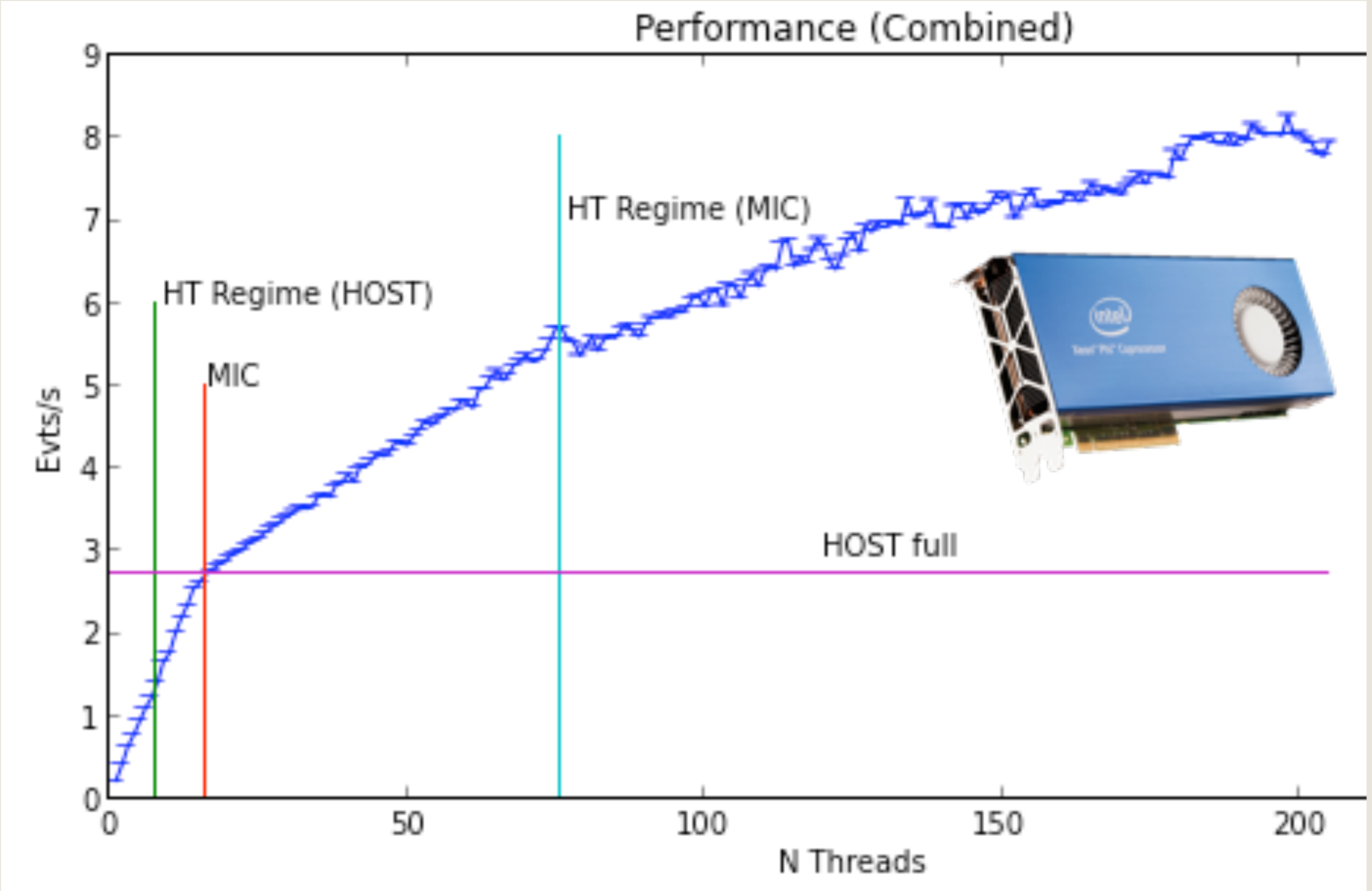
OpenLab@CHEP12

Initiatives so far

- A Concurrency Forum has been established in 2011 to
  - Share knowledge amongst the whole community, create consensus and develop and adopt common solutions
  - Bi-weekly meetings and an R&D programme of work on a number of demonstrators (16+) to explore technology
  - http://concurrency.web.cern.ch
- A TechLab with diverse and advanced hardware & software to test and connection to the companies' engineers
  - Building on the model pioneered by CERN OpenLab
  - Open to our community and complementary to similar facilities elsewhere
  - Technology driven, to generate and motivate demand from the users
  - See https://twiki.cern.ch/twiki/bin/viewauth/IT/TechLab

Geant4 Multi-threading

- Parallelism at level of event - for simple migration of experiments' "user" code
  - Part of next Geant4 10.0 production release (Dec 2013)



Preliminary, Courtesy of A.Dotti, SLAC

- Demonstrates
  - Linear scaling of throughput with number of threads
  - Large savings in memory: 40MB extra memory per thread
- Extension of parallelism to the track level
  - But deeper changes in "user" code

FNAL Geant GPU Prototype

- CERN-FNAL collaboration to
  - Develop and study the performance of various strategies and algorithms that will enable Geant4 to use multiple computational threads
- See P.Canal's presentation (ID: 3)
- Kernel scheduling and CPU/GPU communication
  - The GPU Prototype as part of a full vectorized prototype for end-to-end test
  - A broker than can schedule the processing of tracks on the GPU with maximum flexibility
- Focus has been on NVidia hardware
- We have step up our collaboration with them with the idea to converge to a single code base

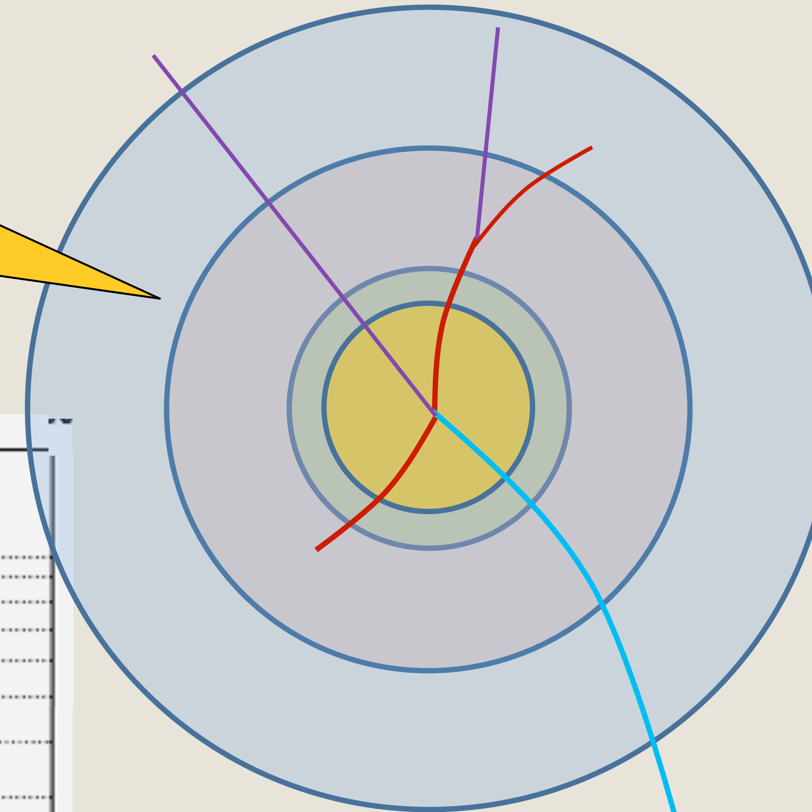
A fresh look at the Simulation

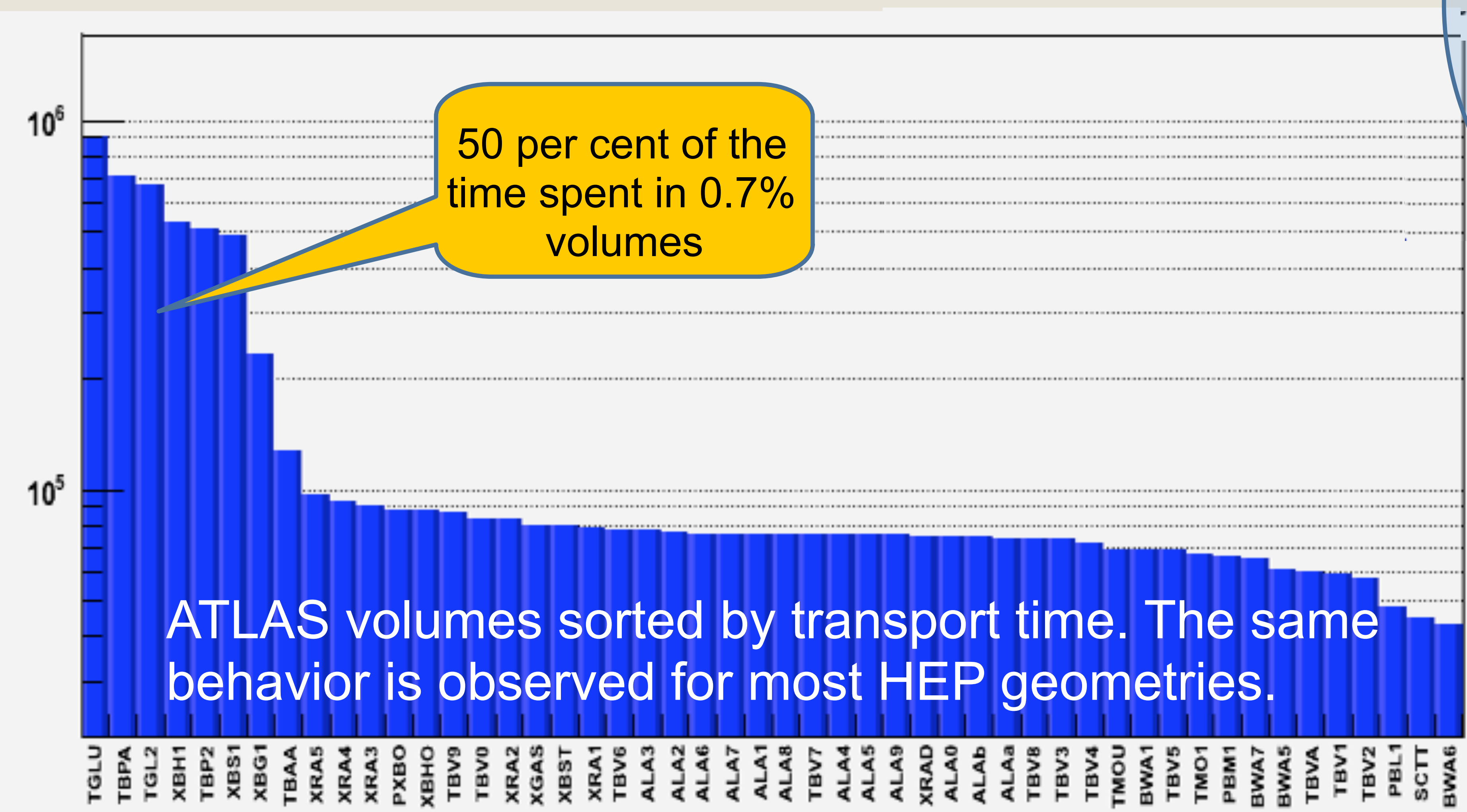
- More than a factor 10 increase expected in the simulation needs in the next few years!
- The most CPU-bound and time-consuming application in HEP with large room for speed-up
  - Largely experiment independent
  - Precision depends on (the inverse of the sqrt of) the number of events
- Grand strategy
  - Explore opportunities with no constraints from existing code
  - Expose the parallelism at all levels, from coarse granularity to micro-parallelism
  - Integrate slow and fast simulation to optimise both in the same framework
- Improvements (in geometry for instance) and techniques are expected to feed back into other HEP applications

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

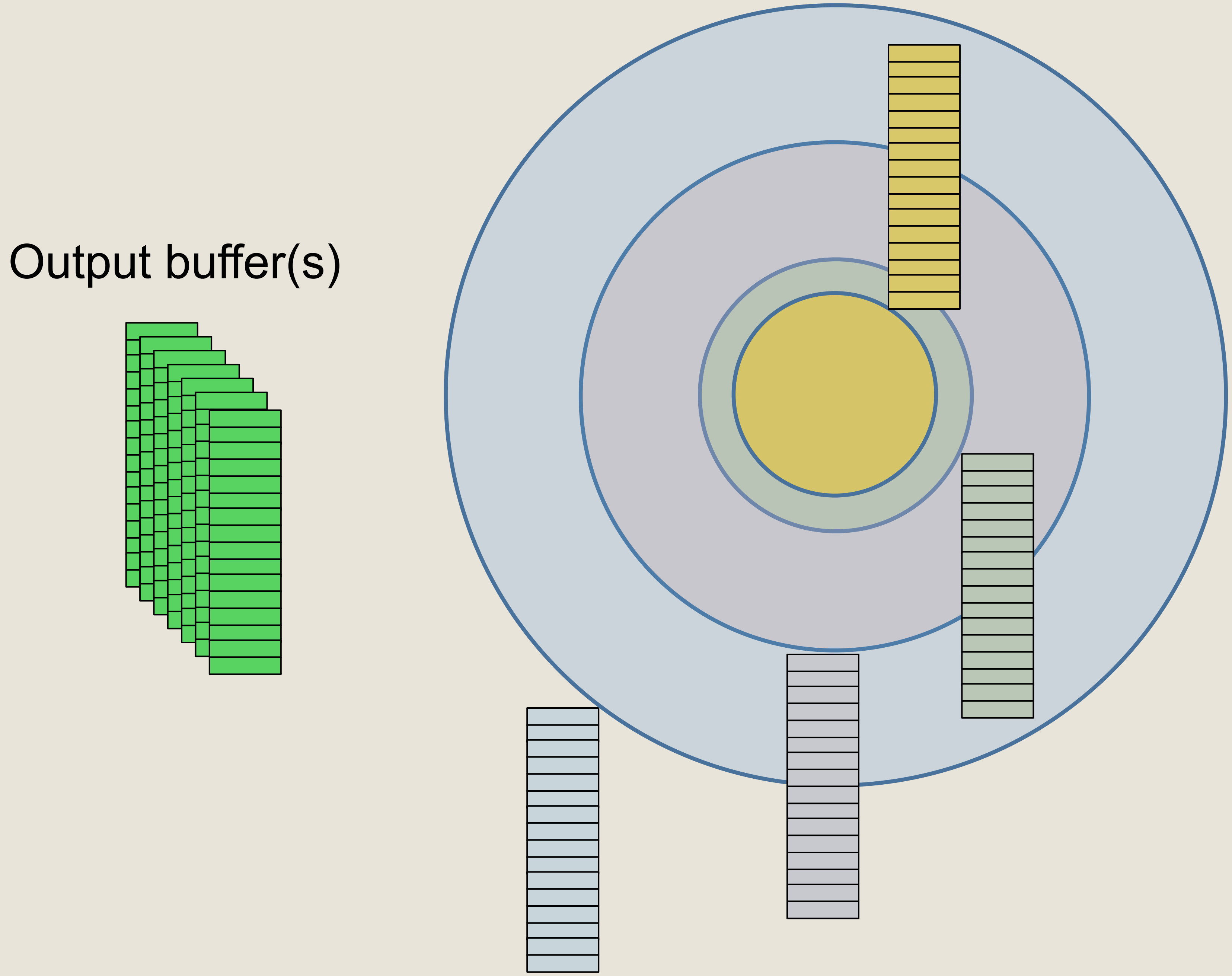




- Navigating very large data structures
- No locality
- OO abused: very deep instruction stack
- Cache misses

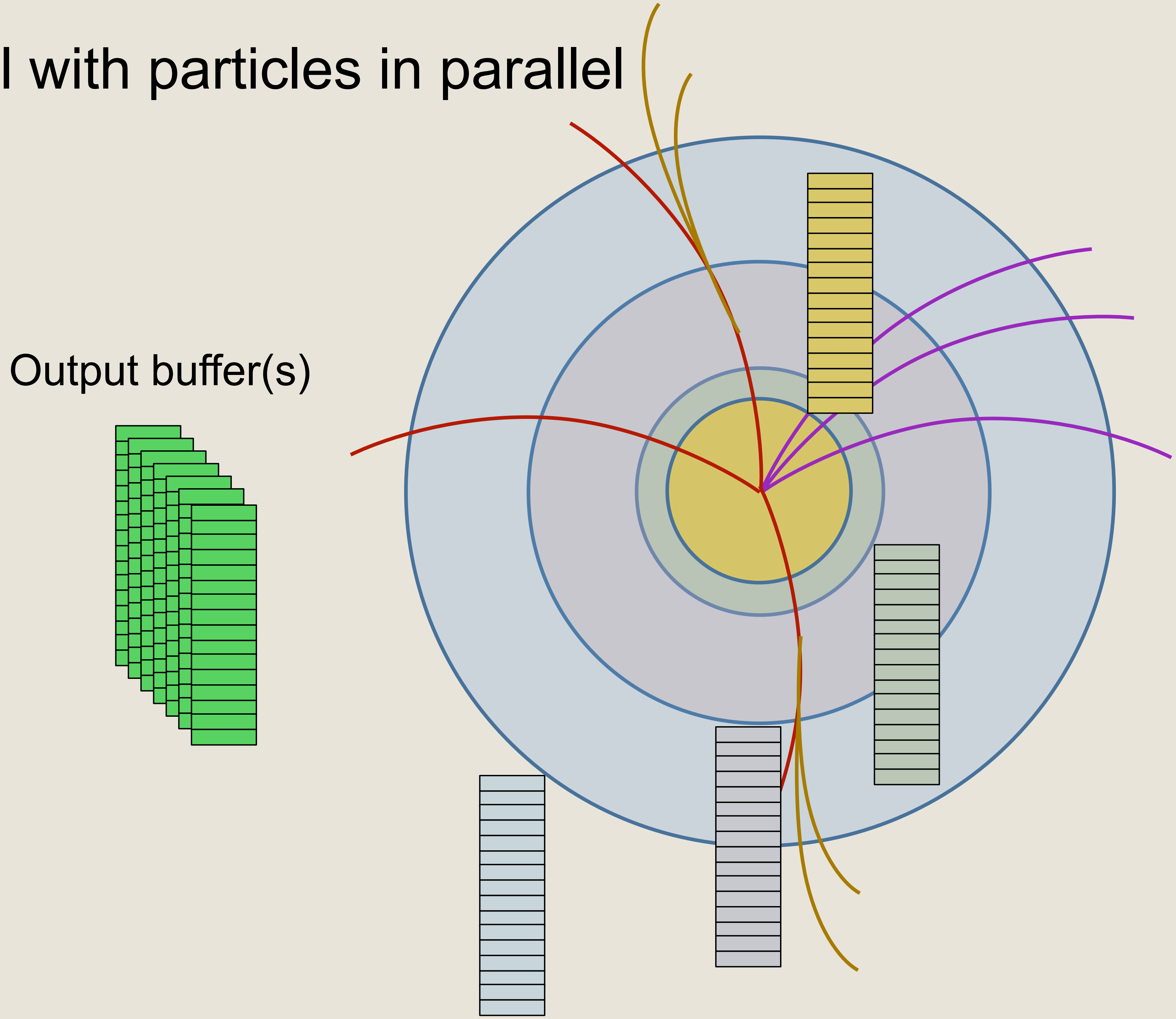
Introduce “basketised” transport

Deal with particles in parallel



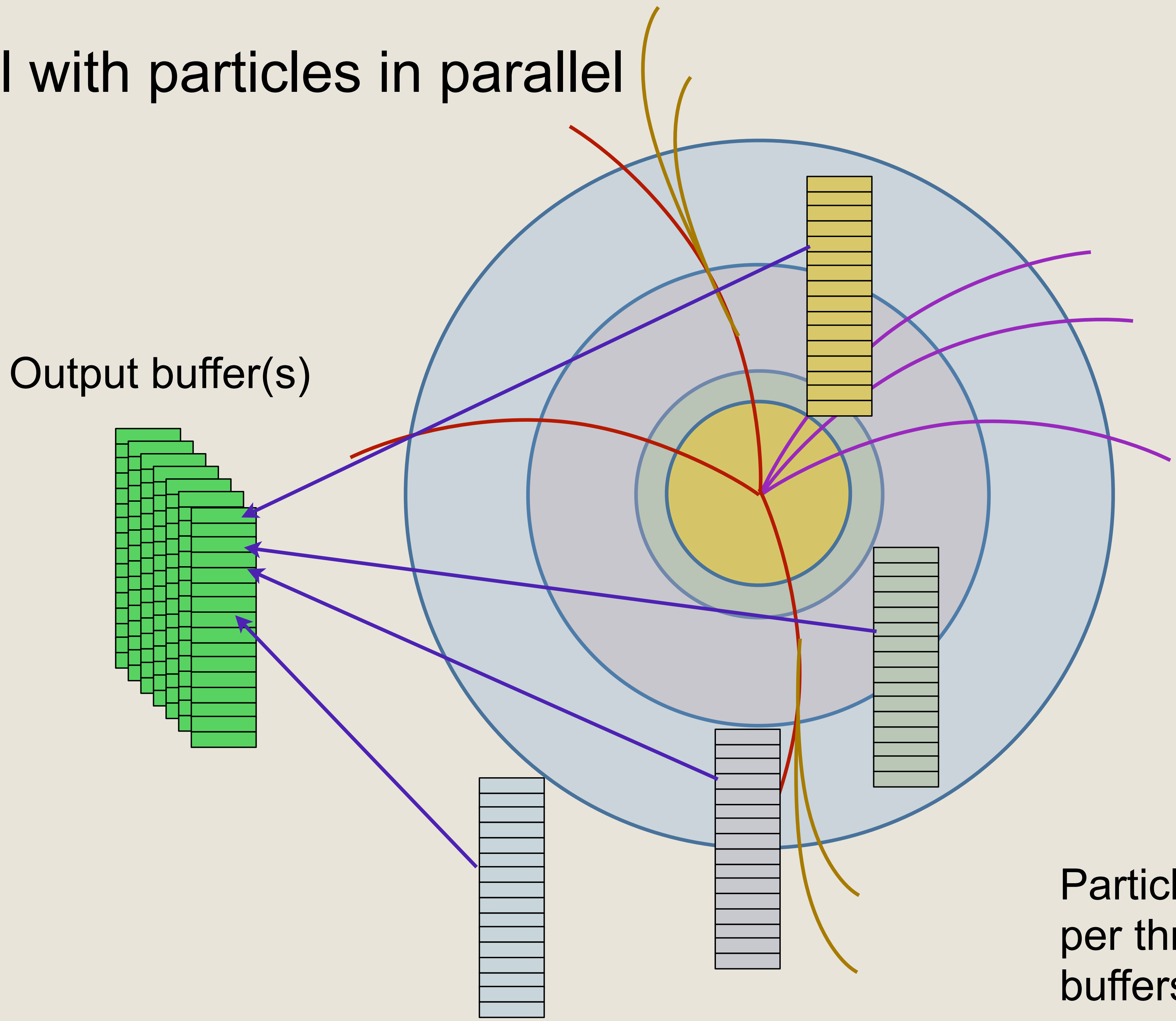
Introduce "basketised" transport

Deal with particles in parallel



Introduce "basketised" transport

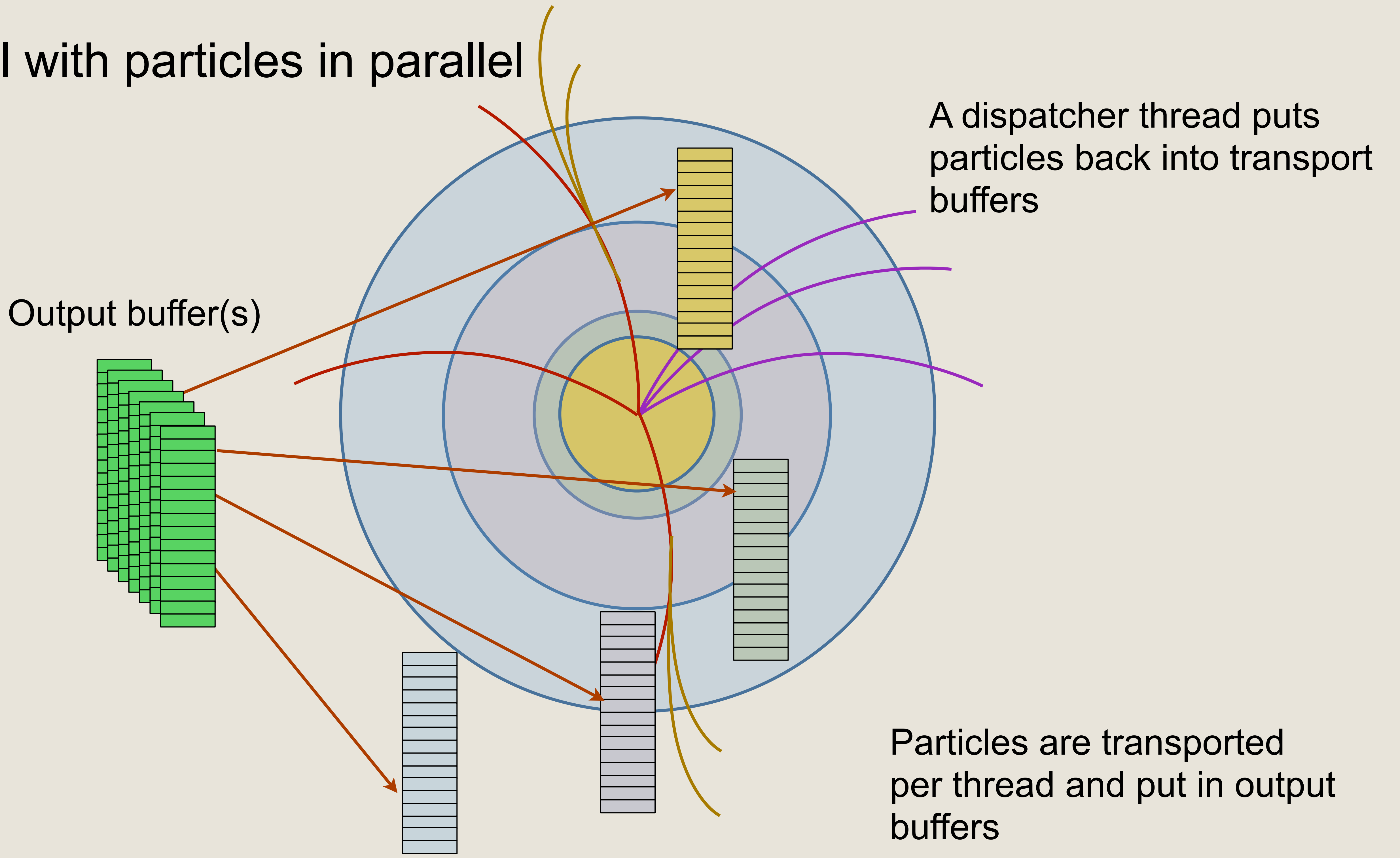
Deal with particles in parallel



Particles are transported per thread and put in output buffers

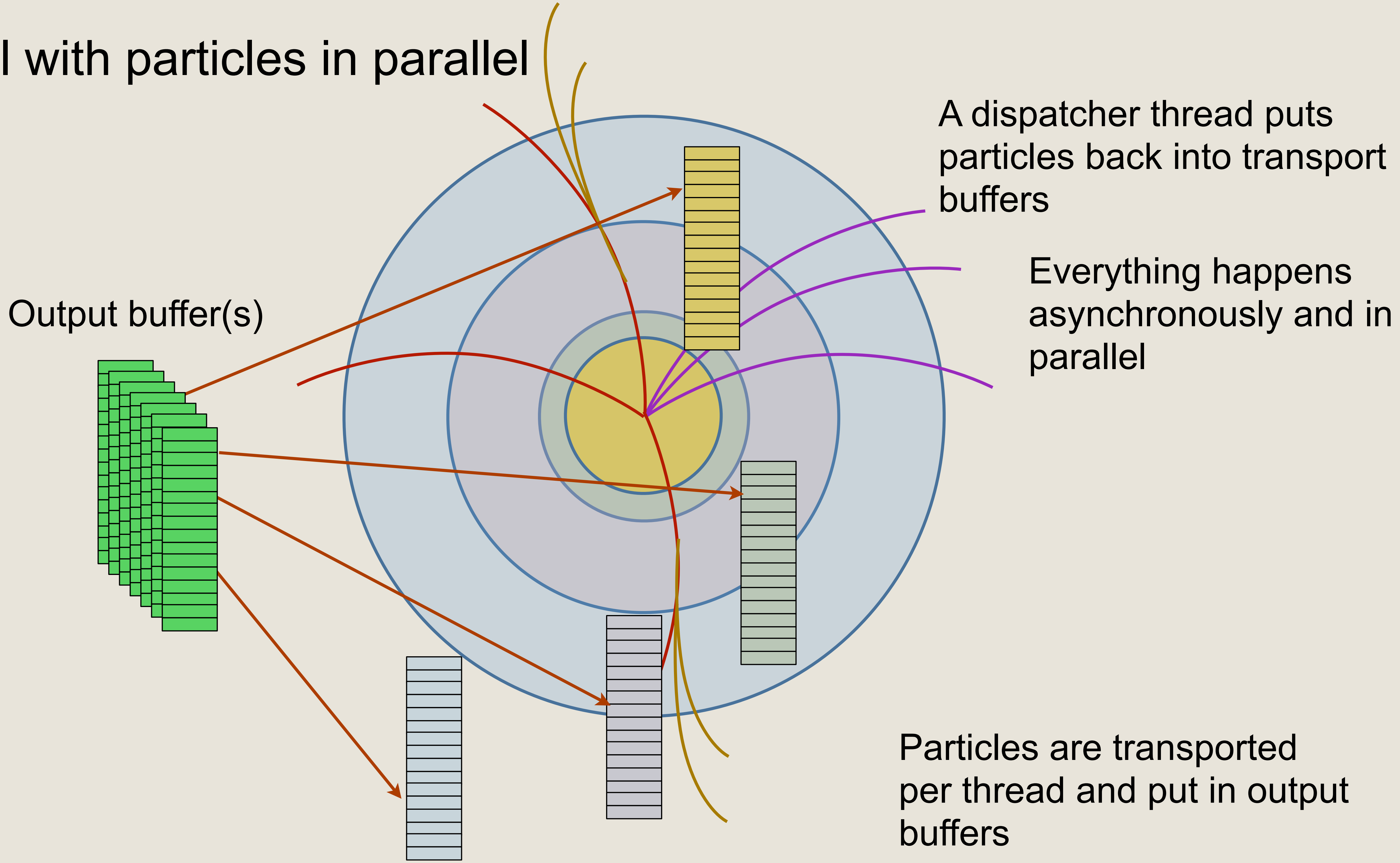
Introduce "basketised" transport

Deal with particles in parallel



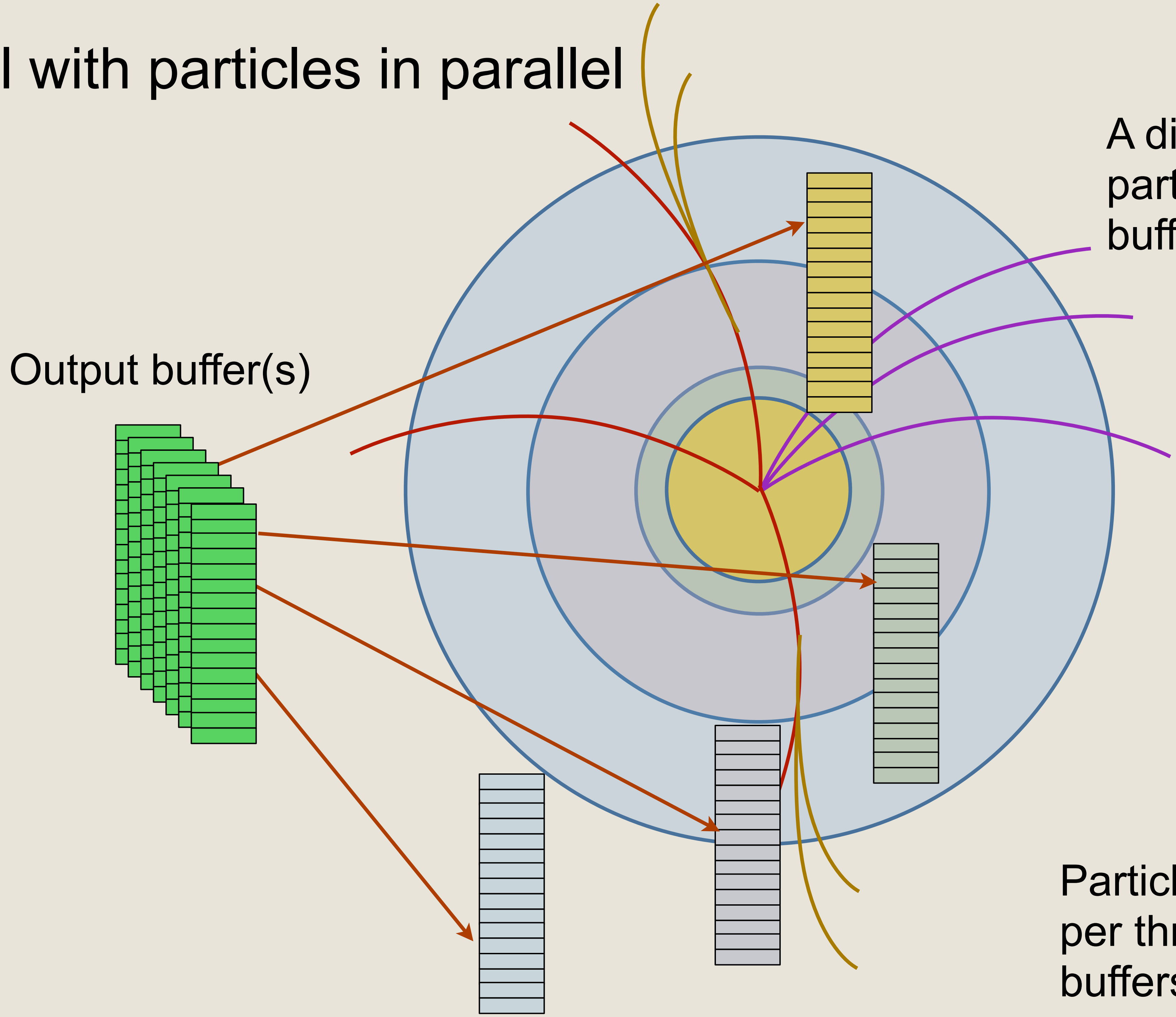
Introduce "basketised" transport

Deal with particles in parallel



Introduce "basketised" transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

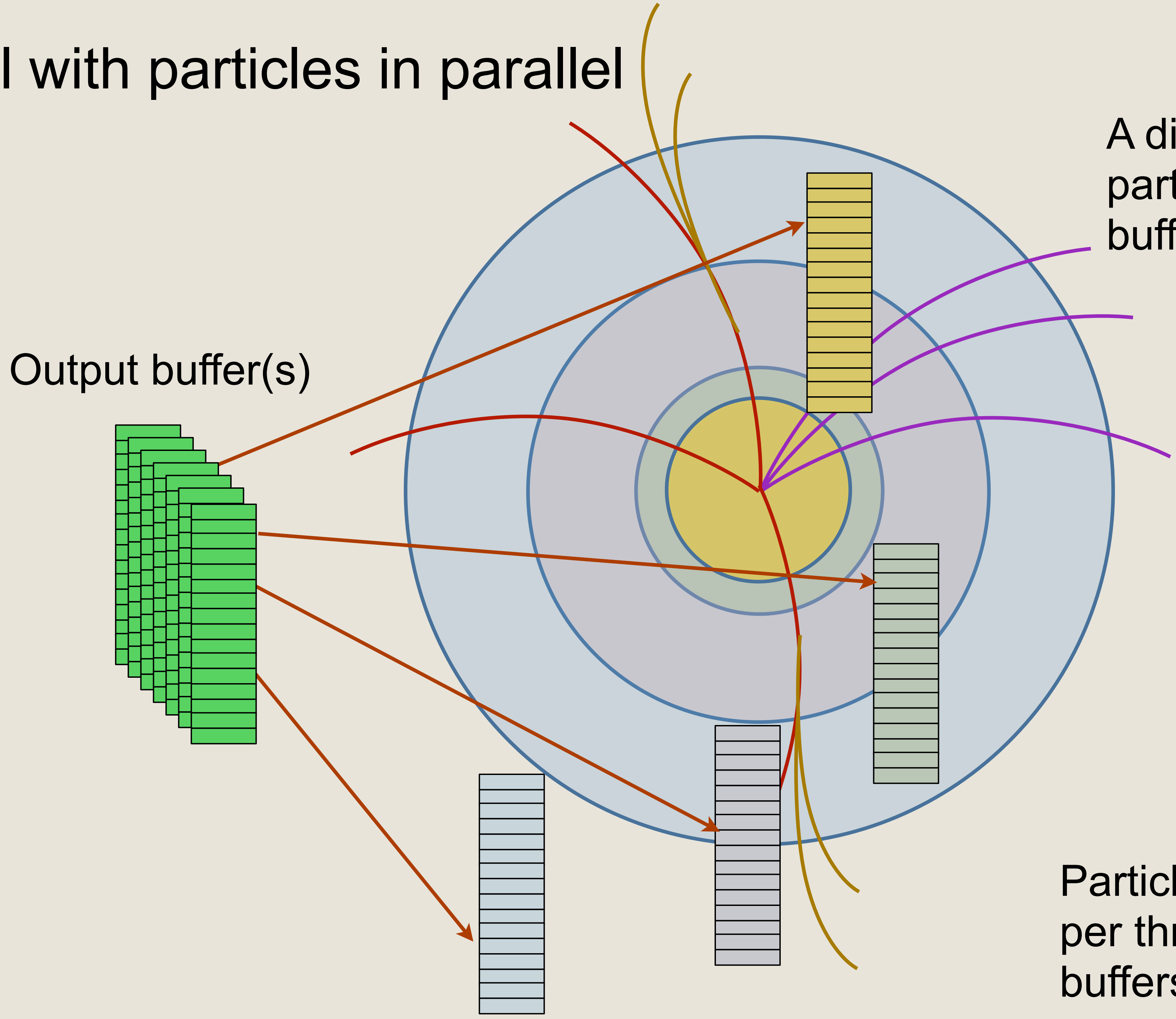
Everything happens asynchronously and in parallel

The challenge is to minimise locks

Particles are transported per thread and put in output buffers

Introduce "basketised" transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

Everything happens asynchronously and in parallel

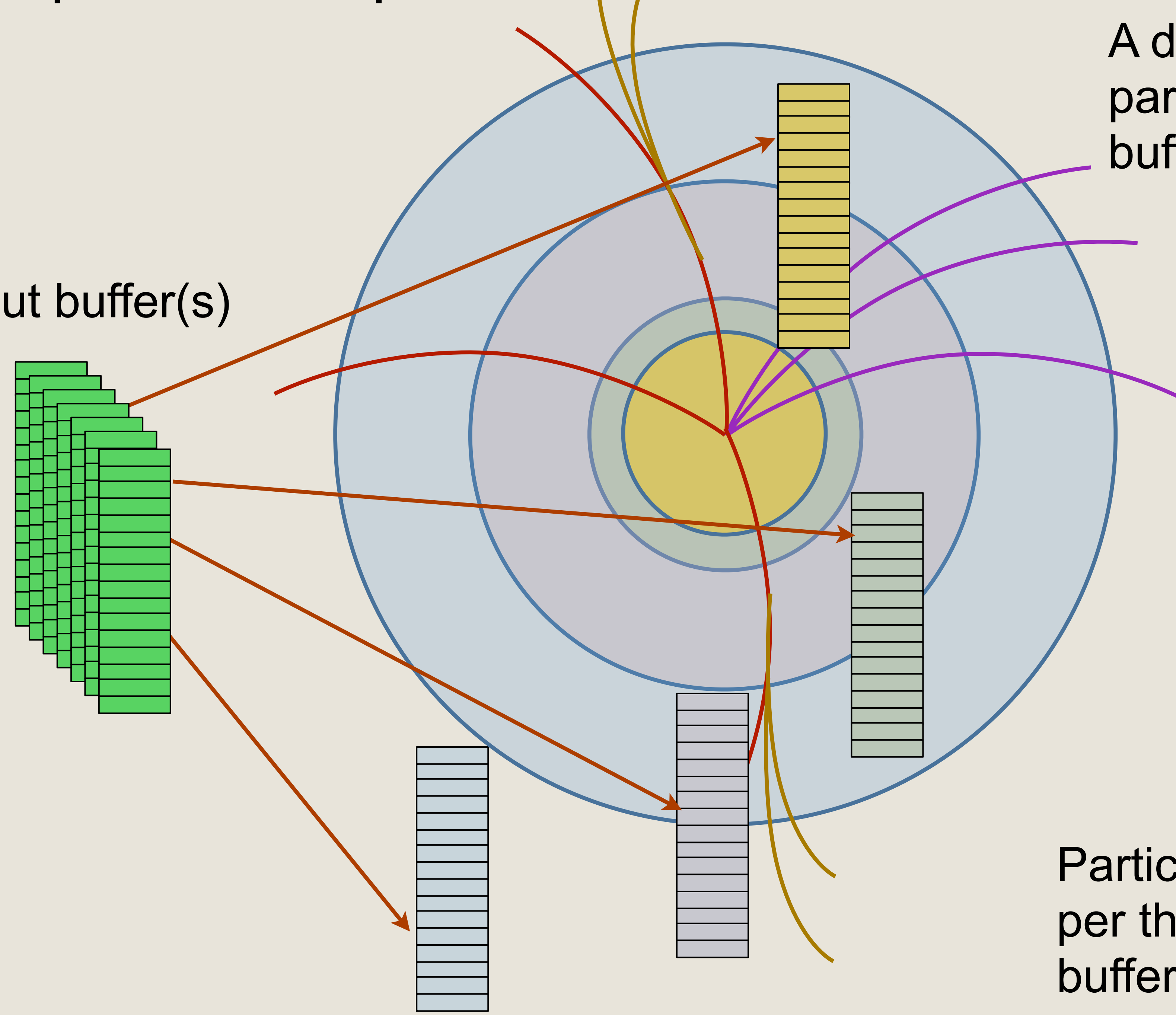
The challenge is to minimise locks

Keep long vectors

Particles are transported per thread and put in output buffers

Introduce "basketised" transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

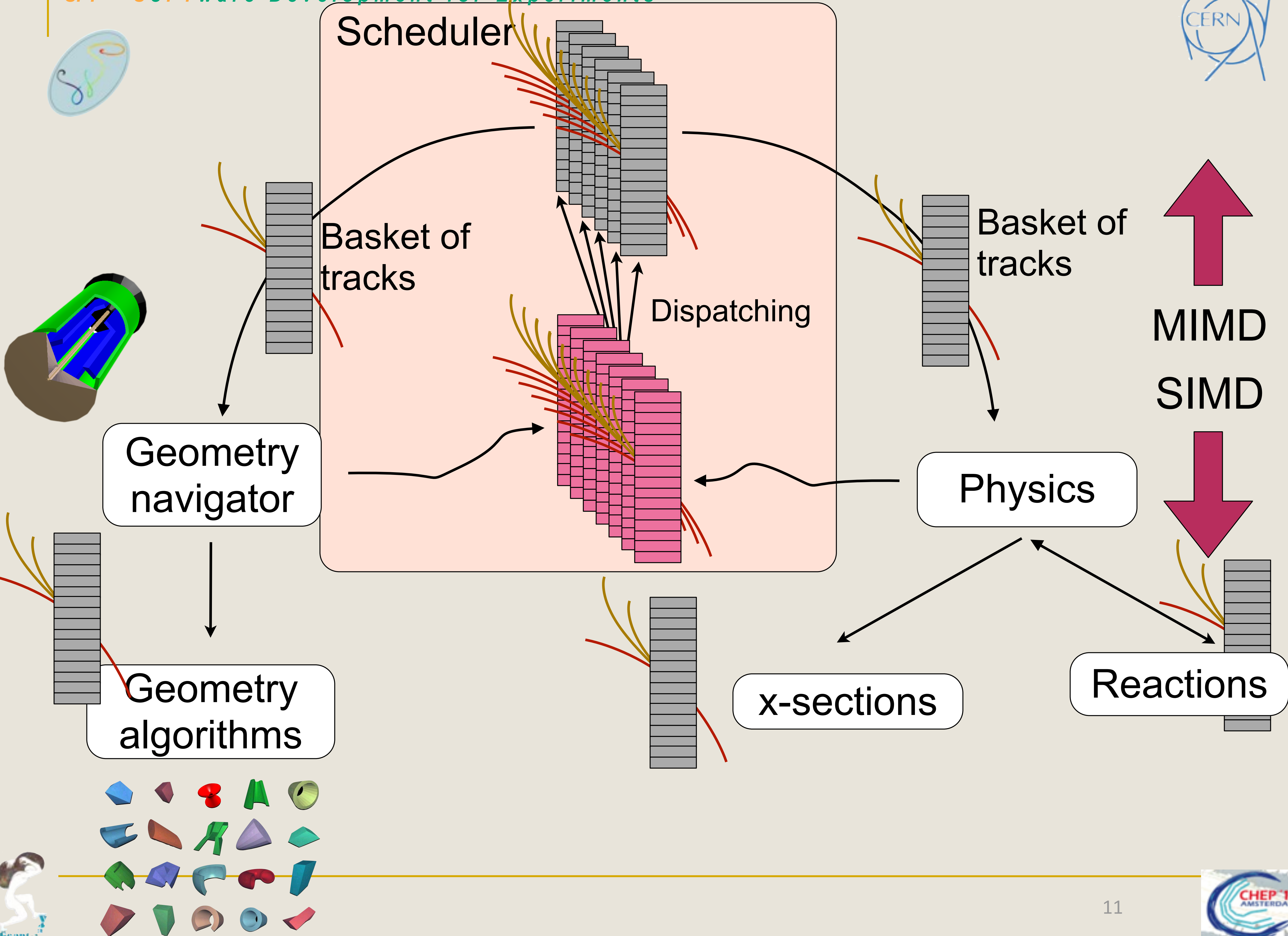
Everything happens asynchronously and in parallel

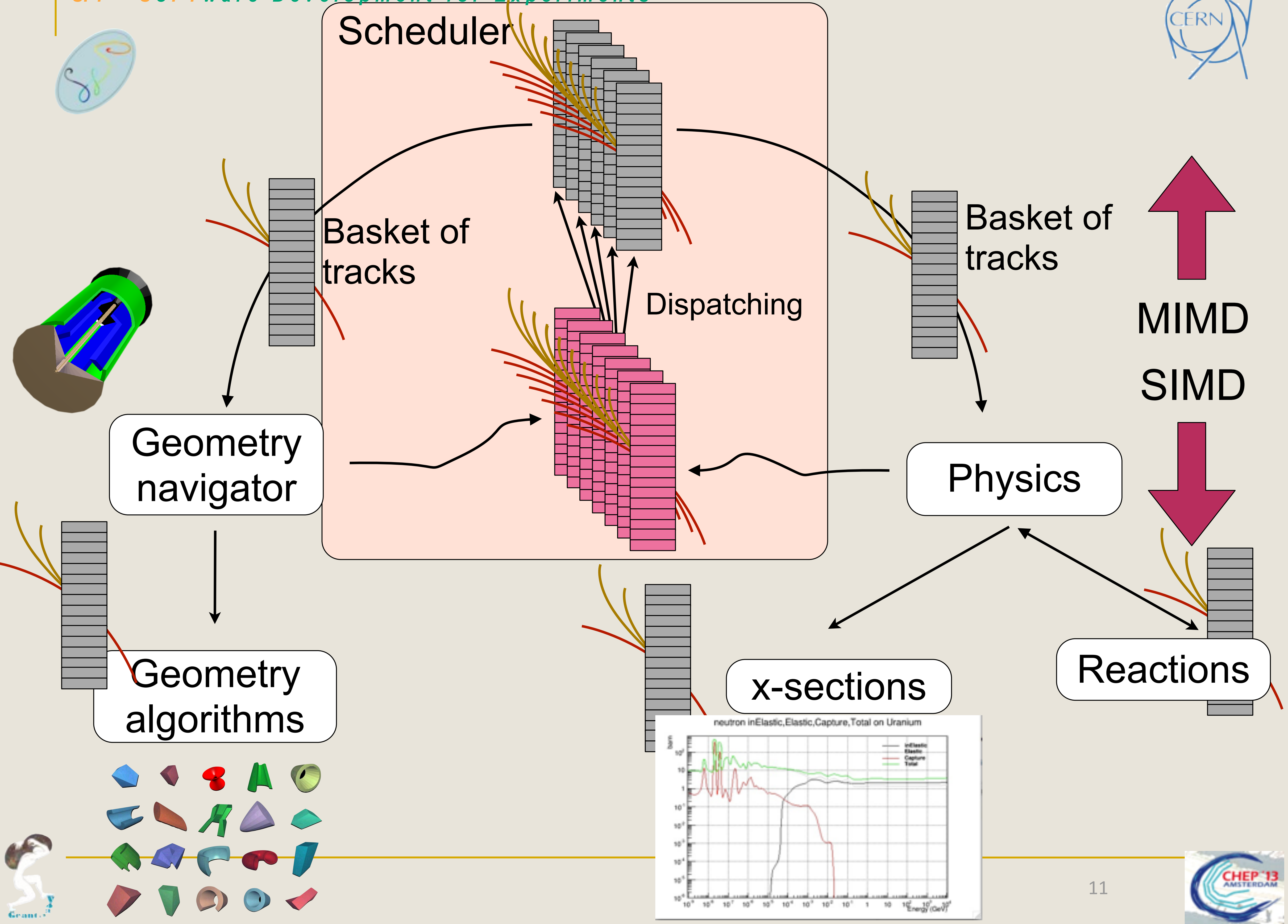
The challenge is to minimise locks

Keep long vectors

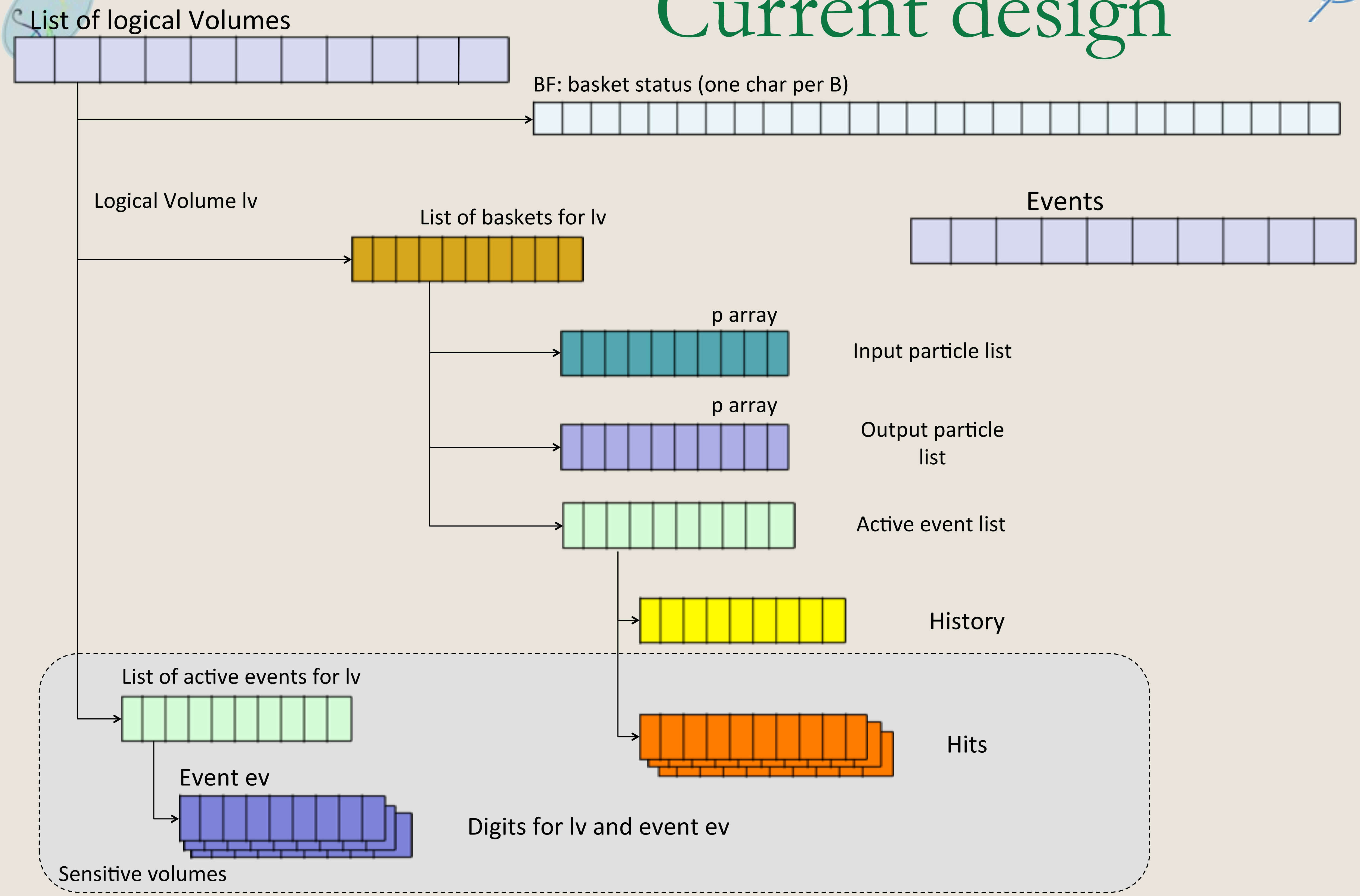
Avoid memory explosion

Particles are transported per thread and put in output buffers

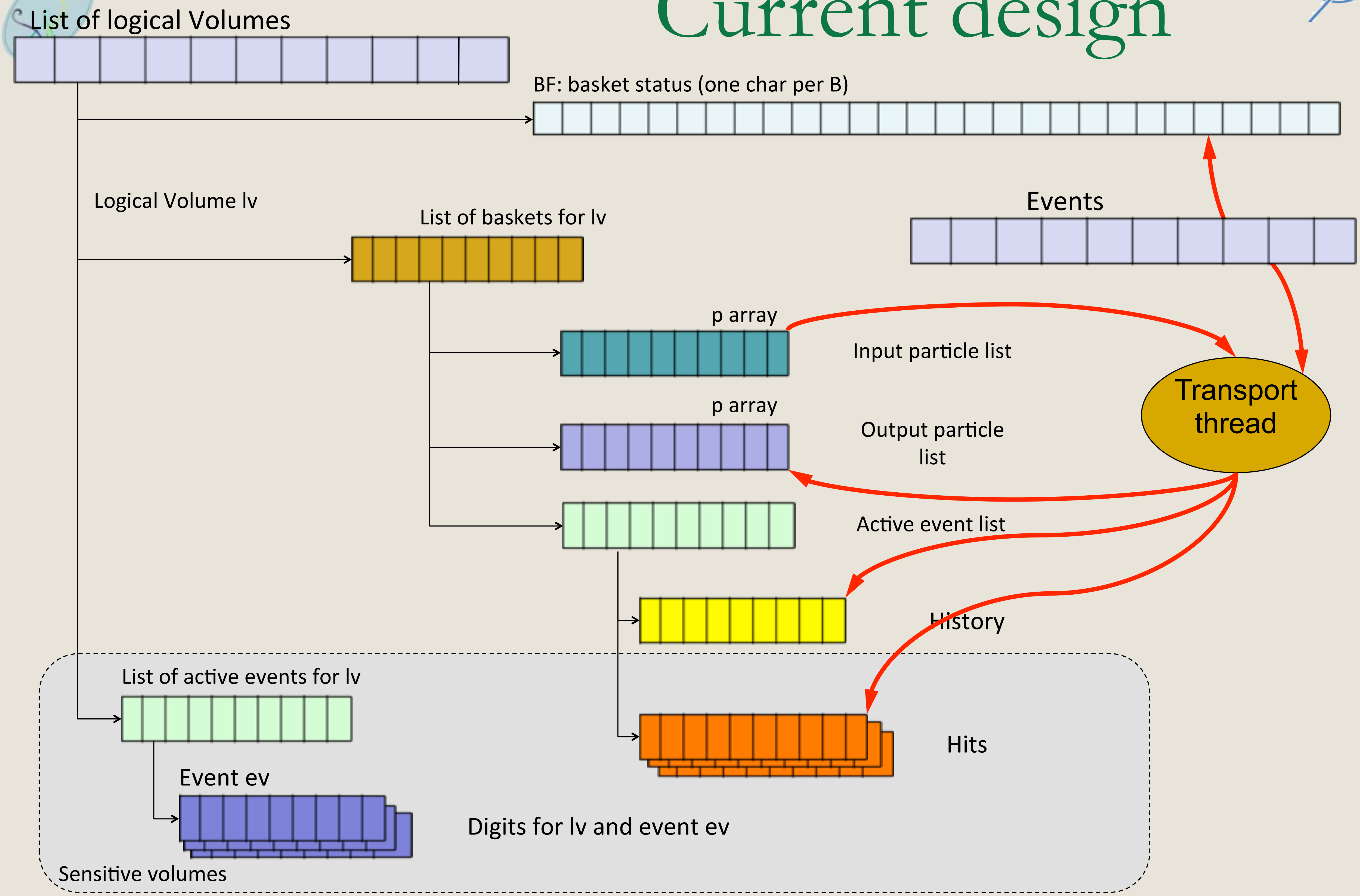

Scheduler
Basket of tracks
Dispatching
Basket of tracks
MIMD
SIMD
Geometry navigator
Physics
Geometry algorithms
x-sections
Reactions


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

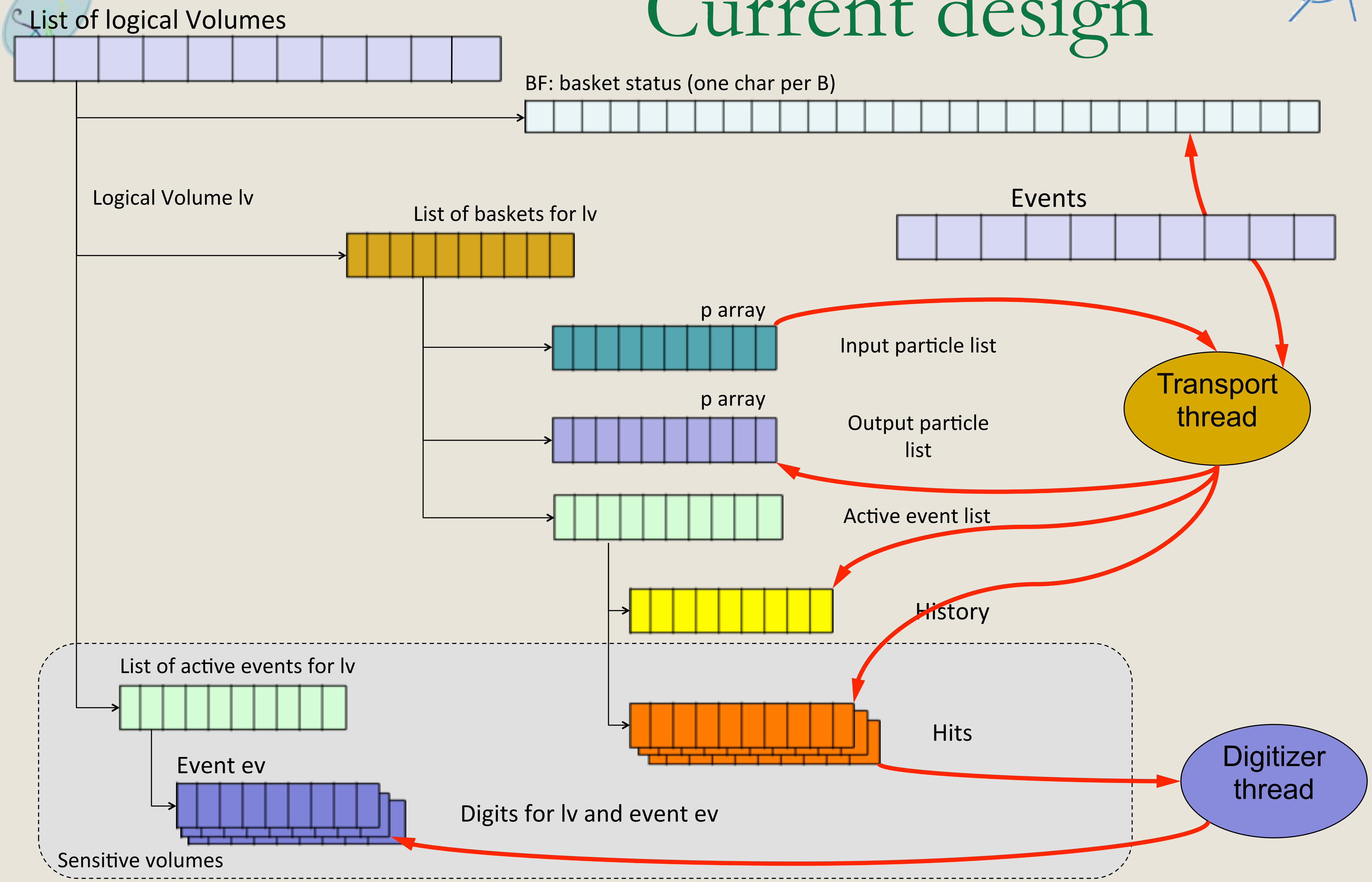
Current design



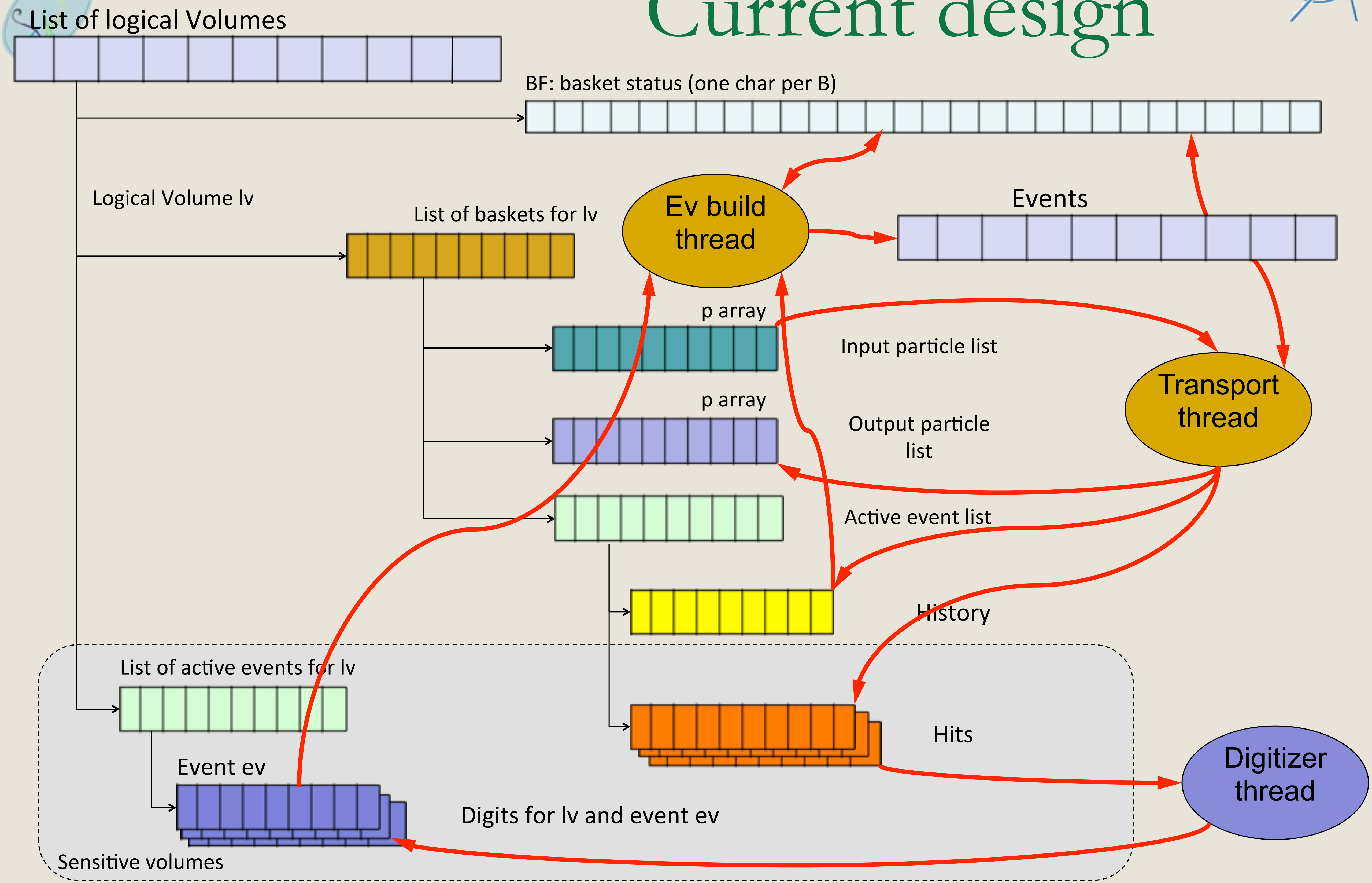
Current design



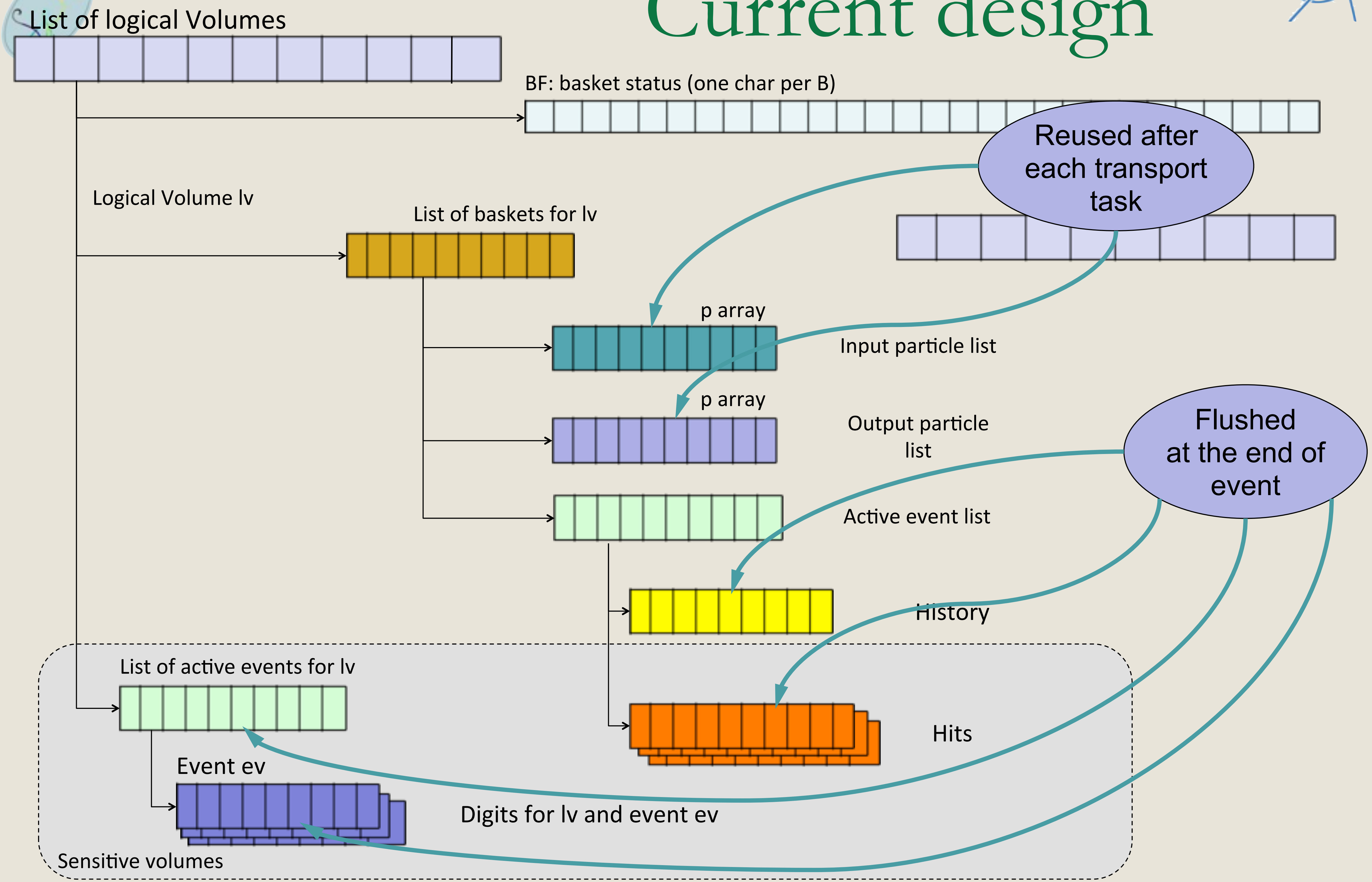
Current design



Current design

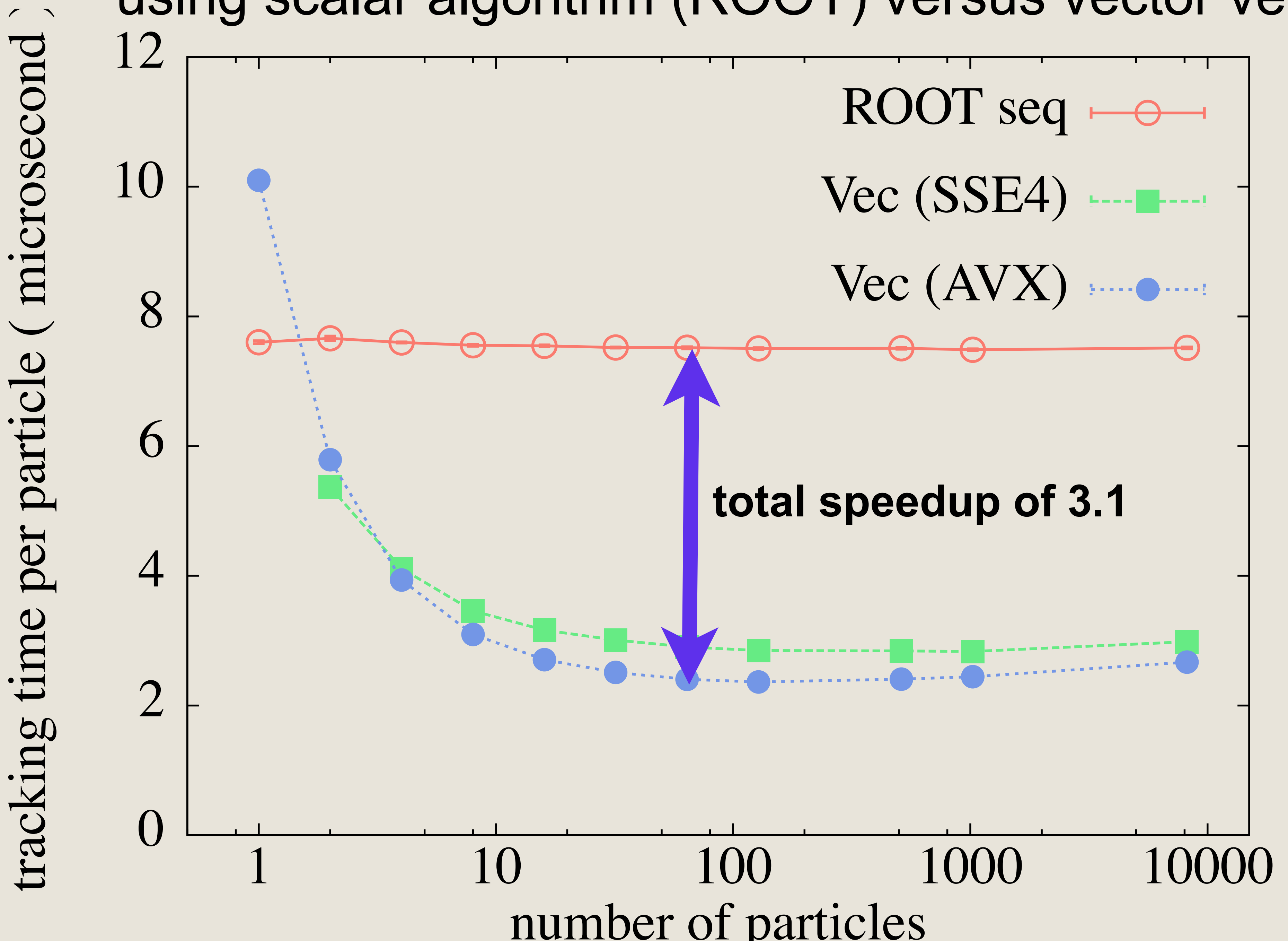


Current design



Gains from microparallelism & SIMD

- Time of processing/navigating N particles (P repetitions) using scalar algorithm (ROOT) versus vector version

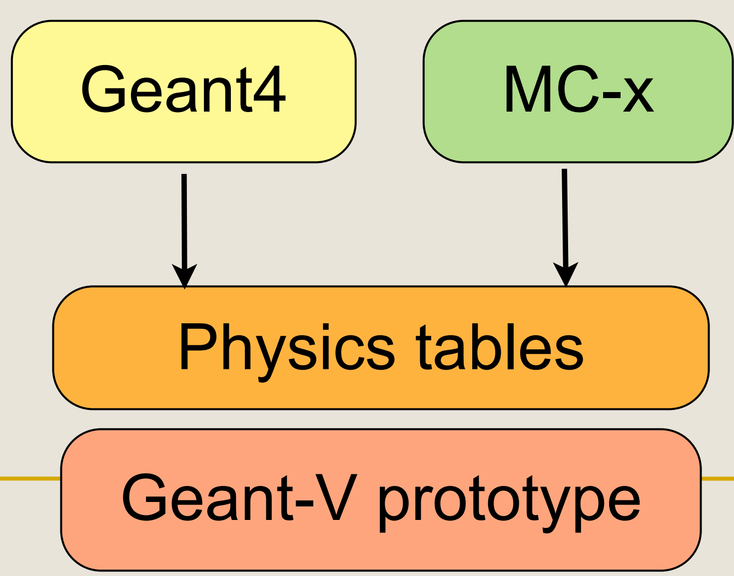


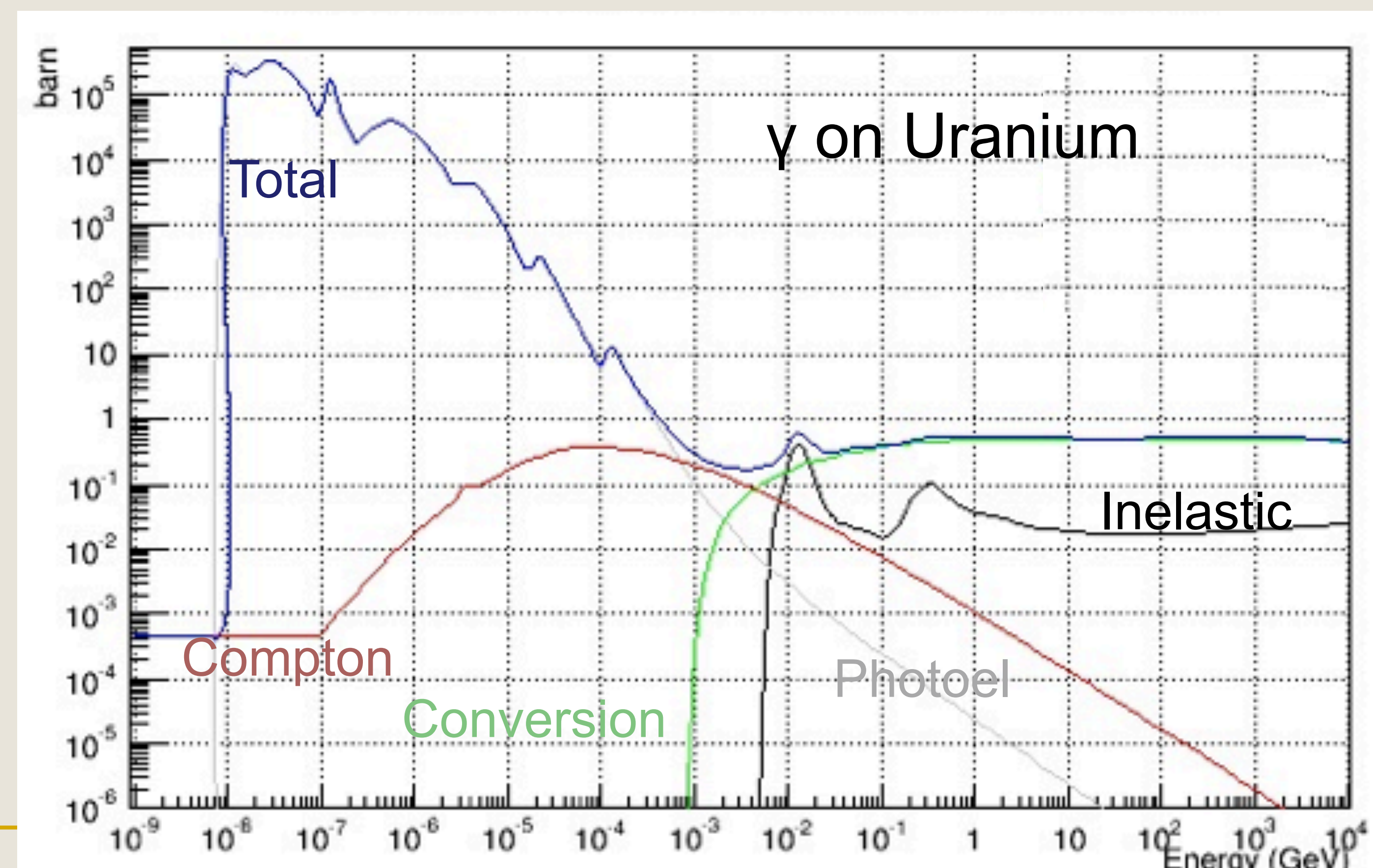
- excellent speedup for SSE4 version
- some further gain with AVX
- already gain considerably for small N
- there is an optimal point of operation (performance degradation for large N)

For more fun see A.Gheata's talk!
https://indico.cern.ch/contributionDisplay.py?contribId=453&confId=214784

Physics

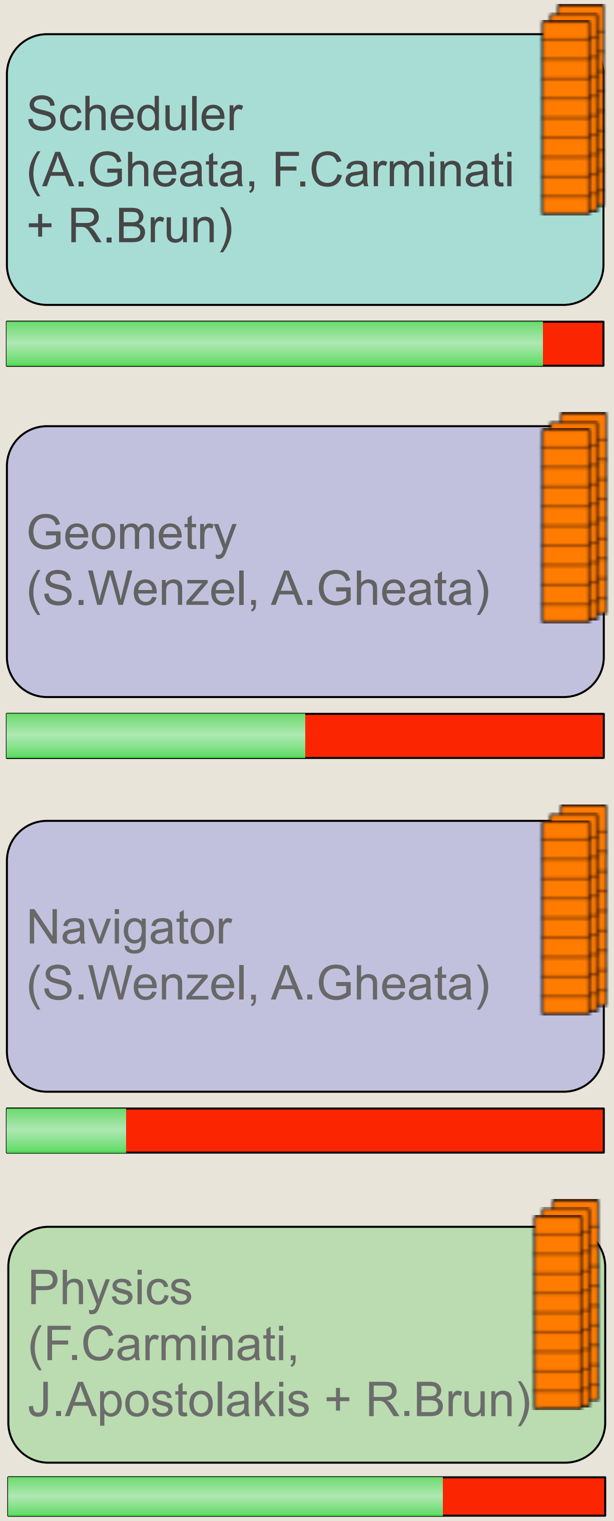
- A lightweight physics for realistic shower development
  - Select the major mechanisms
    - Bremsstrahlung, e+ annihilation, Compton, Decay, Delta ray, Elastic hadron, Inelastic hadron, Pair production, Photoelectric, Capture + dE/dx & MS
  - Tabulate all x-secs (100 bins -> 90MB)
  - Generate (10-50)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





Where are we now?

- Scheduler
  - The new version, hopefully improved of the scheduler has been committed and we are testing it
- Geometry
  - The proof or principle that we can achieve large speedups (3-5+) is there (see A.Gheata's talk), 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 (S.Wenzel), 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



Targets

- By the end of the year we will "glue" the different pieces together
  - And hopefully demonstrate the speedup potential of MT, locality and SIMD
- Measure the evolution of the memory footprint and the performance of the code at least in terms of hardware counters
- Absolute performance measurements will be harder
  - Difficult compare apples to apples
  - Probably we need to develop dedicated benchmarks
- Compare physics performance with full MC's
- For the moment we use Xeon architecture for the SIMD, but we intend to extend to GPU and to Xeon PHI
- We are working closely with Geant4 for the physics tables
- Once the prototyping phase over, we will have to sit down with the stakeholders and decide how to proceed from there

Summary

- HEP needs all the cycles it can obtain, nowadays this means using parallelism and SIMD
- Simulation is the ideal primary target for investigation for its relative experiment independence and its importance in the use of computing resources
- The Geant Vector project aims at demonstrating substantial speedup (3-5+) on modern architectures
- The work is done in close collaboration with the stakeholders and with Geant4