A concurrent vector-based steering framework for particle transport

ACAT 2013

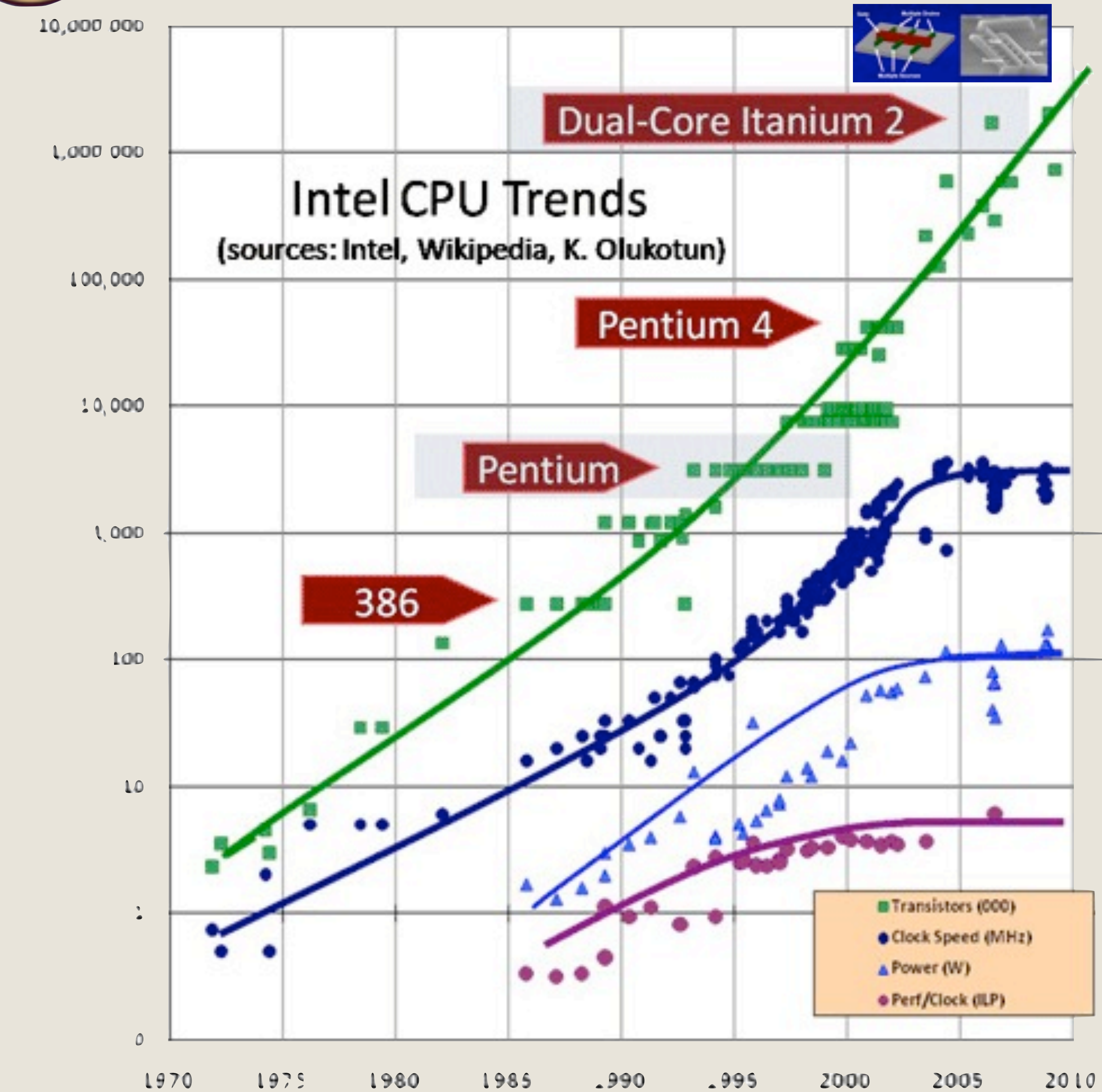
15 May 2013

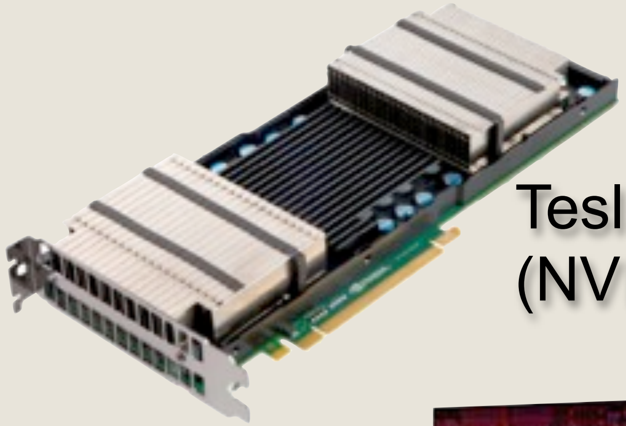
René Brun, Federico Carminati,
Andrei Gheata

The global perspective

- Massive-parallelism is here – Moore’s law has to be reinterpreted
  - You do not get more speed for free, you get more optimisation opportunities to exploit
- We “got away” many times, but now we probably can’t
  - Difficult to ask Funding Agencies for (much) more computing and, at the same time, confess that we use only part of the “bare iron”
- “Embarrassing parallelism” and “throughput computing” reduced the push to shorten “time to solution”

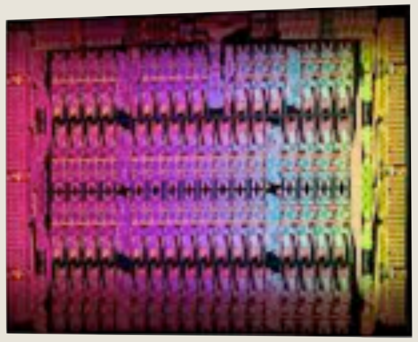
Trends…

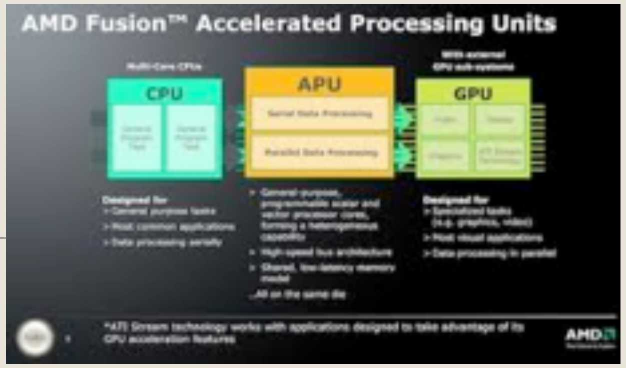




Tesla k10 GPU (NVIDIA)

Intel Many Integrated Core Architecture





AMD Fusion™ Accelerated Processing Units

Designed for
- General purpose tasks
- Most common applications
- Data processing serially

- General-purpose, programmable scalar and vector processor cores, forming a heterogeneous capability
- High-speed bus architecture
- Shared, low-latency memory model

…All on the same die

Designed for
- Specialized tasks (e.g. graphics, video)
- Most visual applications
- Data processing in parallel

*ATI Stream technology works with applications designed to take advantage of its GPU acceleration features

AMD "on board" GPU for fine grain, low latency GPU applications

Texas Instruments DSPs

ARM CPUs

ATOM CPUs

While transistor density has been following Moore's law, frequency and power consumption did not…

The Eight dimensions

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

The Eight dimensions

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

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

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

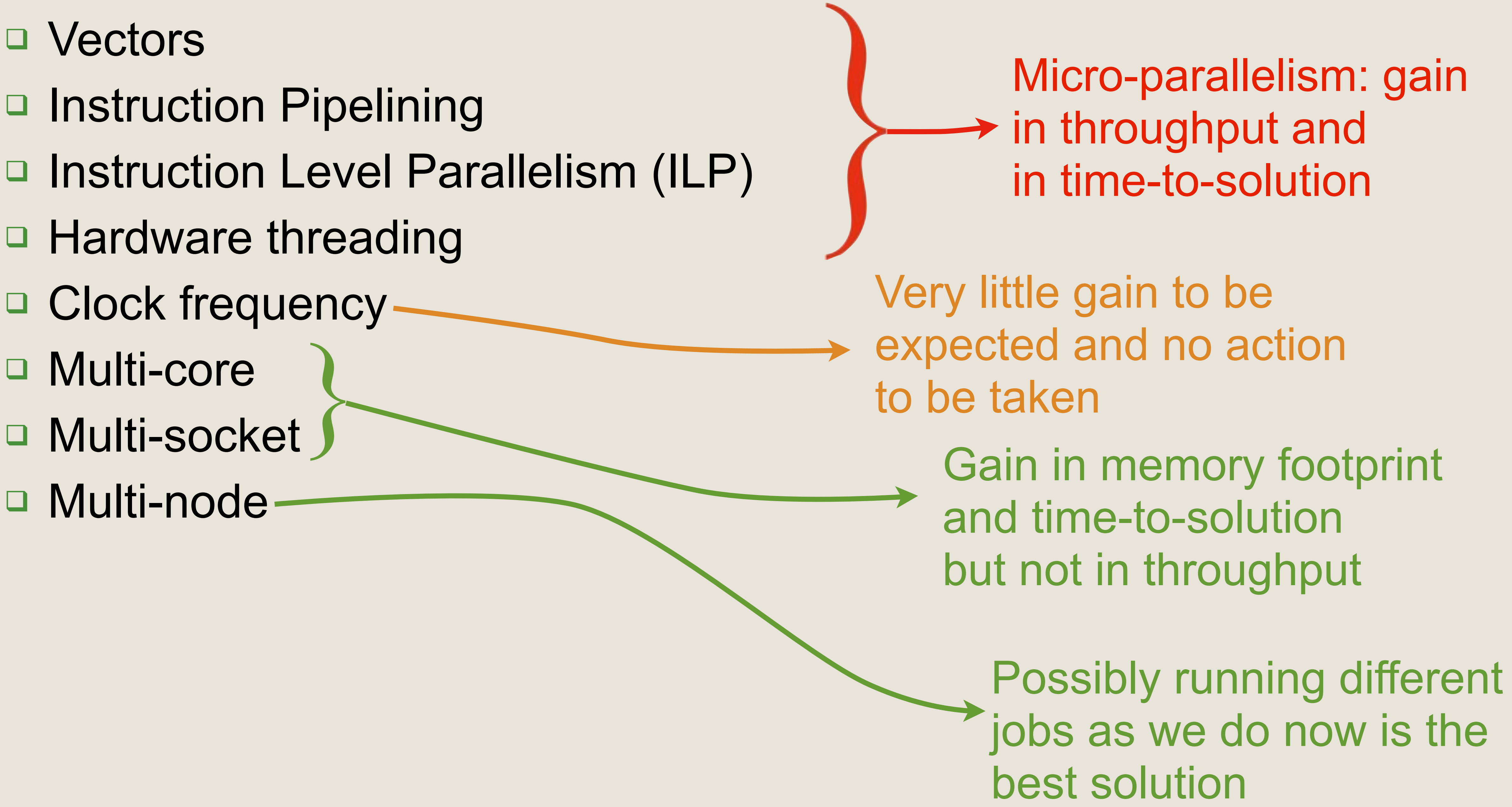
Very little gain to be expected and no action to be taken

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

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
MAX	8	4	1.35
INDUSTRY	6	1.57	1.25
HEP	1	0.8	1.25

Expected limits on performance scaling (multiplied)

	SIMD	ILP	HW THREADS
MAX	8	32	43.2
INDUSTRY	6	9.43	11.79
HEP	1	0.8	1

OpenLab@CHEP12

The Eight dimensions

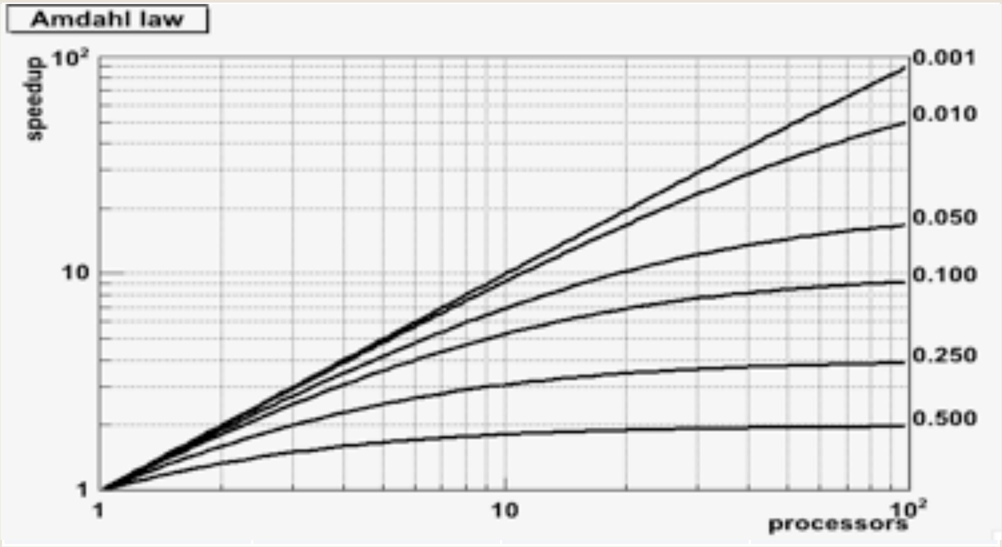
- The “dimensions of performance”
  - Vectors
  - Instruction Pipelining
  - Instruction Level Parallelism (ILP)
  - Hardware threading
  - Clock frequency
  - Multi-core
  - Multi-socket
  - Multi-node

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



OpenLab@CHEP12

Why it is so hard

- Wide span of “programming proficiency” among developers
- O(10) MLOC
- Up to three releases per week
  - On the Grid svp
- The performance scaling with SpecInt (!)
- A lot of parallelism
  - But we are doing throughput computing, what matters is not time-to-solution, but events per second
- Particle transport and tracking are complex and heterogeneous problems
  - Lot of “disuniformity”
  - Tens of particle types to propagate in thousands of different geometry volumes

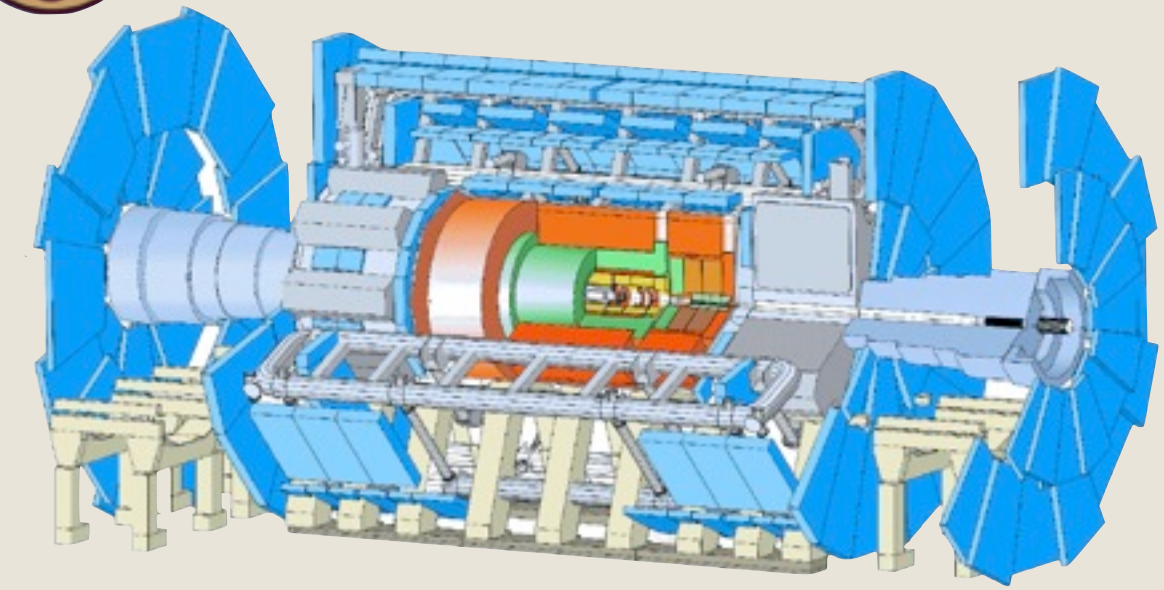
Initiatives taken so far

- A **Concurrency Forum** was established last year
  - Share knowledge amongst the whole community
  - Form a consensus on the best concurrent programming models and on technology choices
  - Develop and adopt common solutions
- Bi-weekly meeting with an active and growing participation of different laboratories and experiments
- An R&D programme of work on a number of **demonstrators** for exercising different capabilities, with clear deliverables and goals
  - 16 projects are in progress started by different groups in all corners of the community

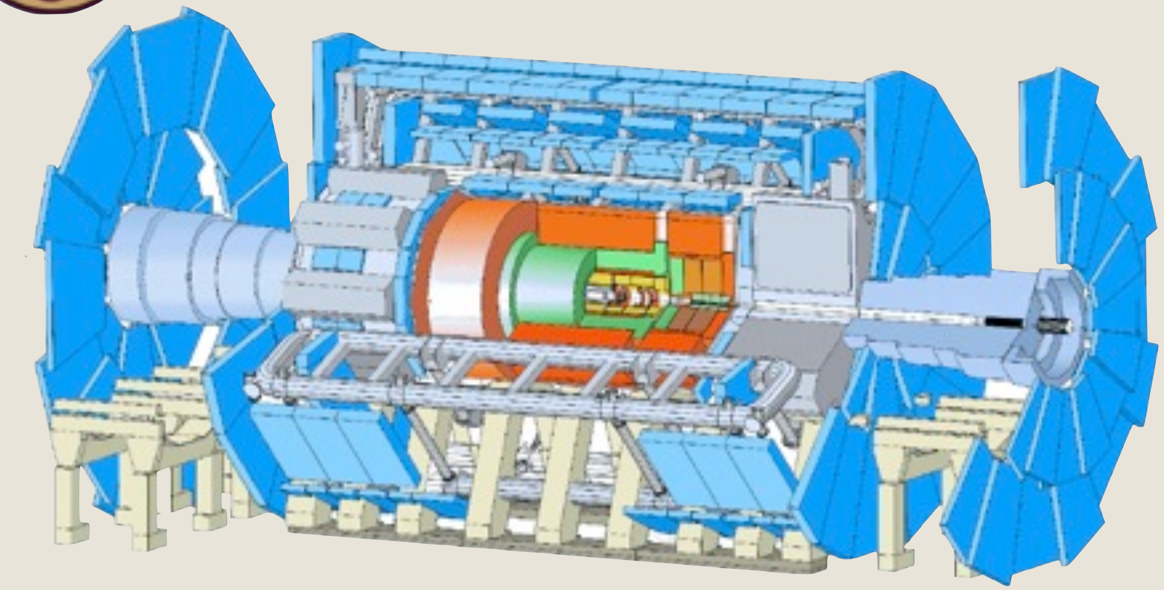
Starting all over – simulation

- 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
- Improvements (in geometry for instance) and techniques are expected to feed back into reconstruction
- 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

The Atlas detector simulation



The Atlas detector simulation

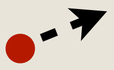


- O(10^7) particles to be transported per simulation
- ~1.5GB of RAM required
- Future detectors will be even more challenging

- ~29 million volumes
- ~7,000 volume prototypes
- Picometer level precision in particle transport
- O(10^{11}) events to be simulated
- O(3m) to simulate an event

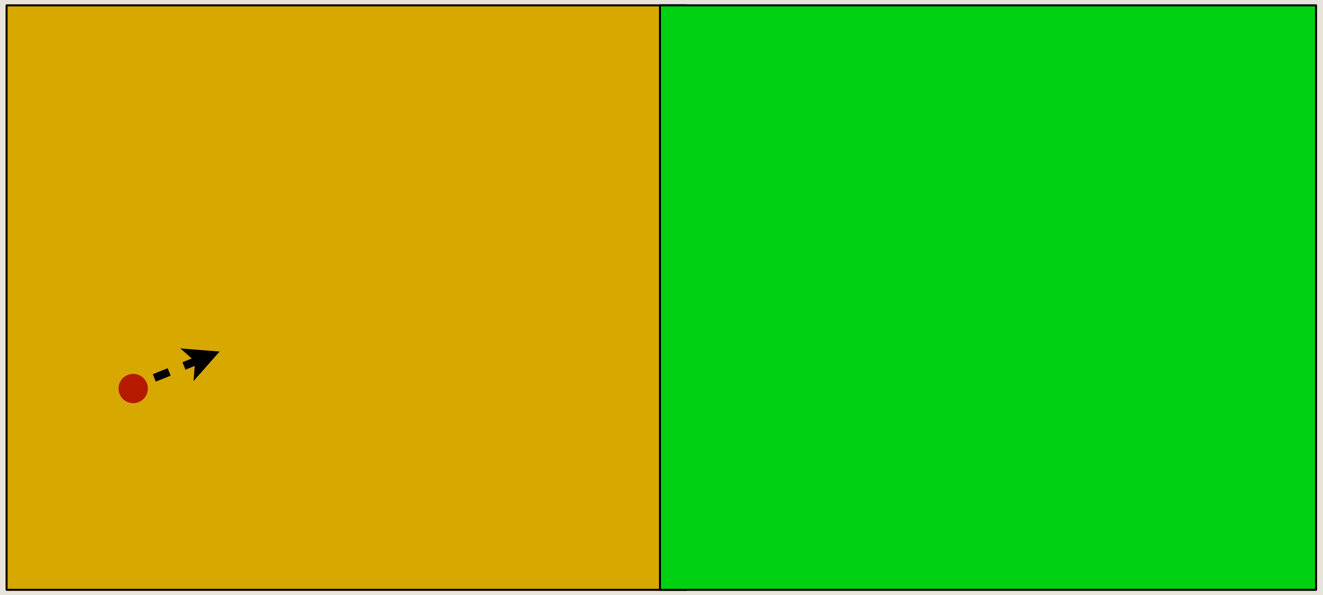
Transport in a nutshell

Transport in a nutshell



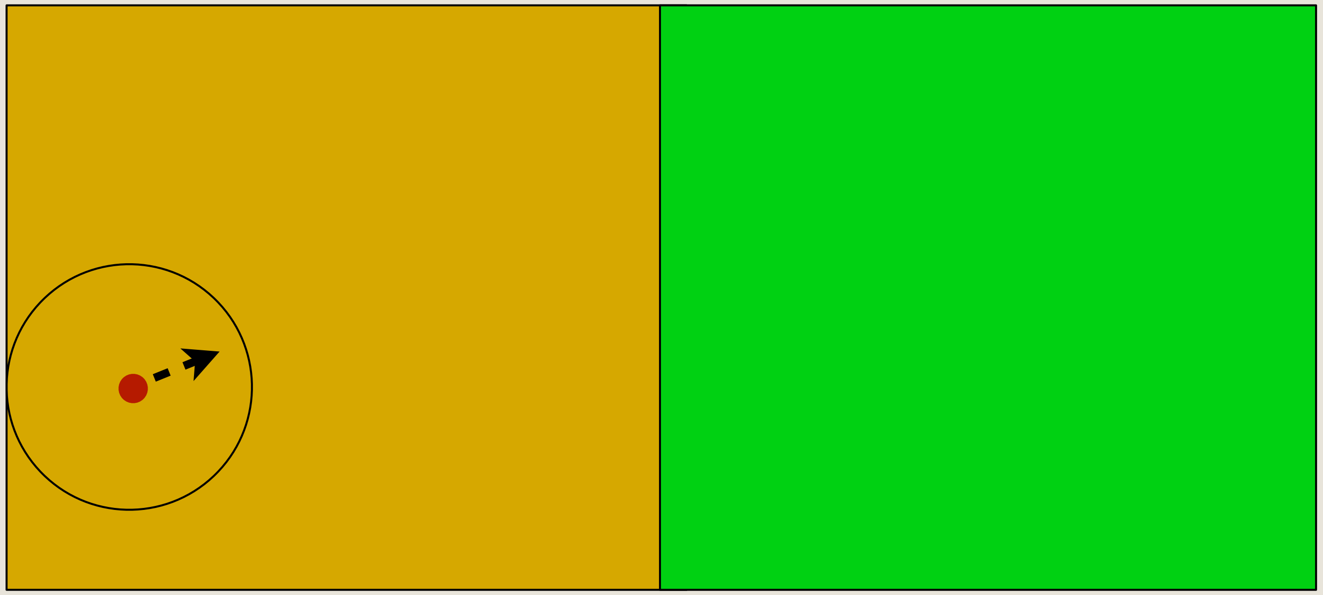
- Particle in flight

Transport in a nutshell



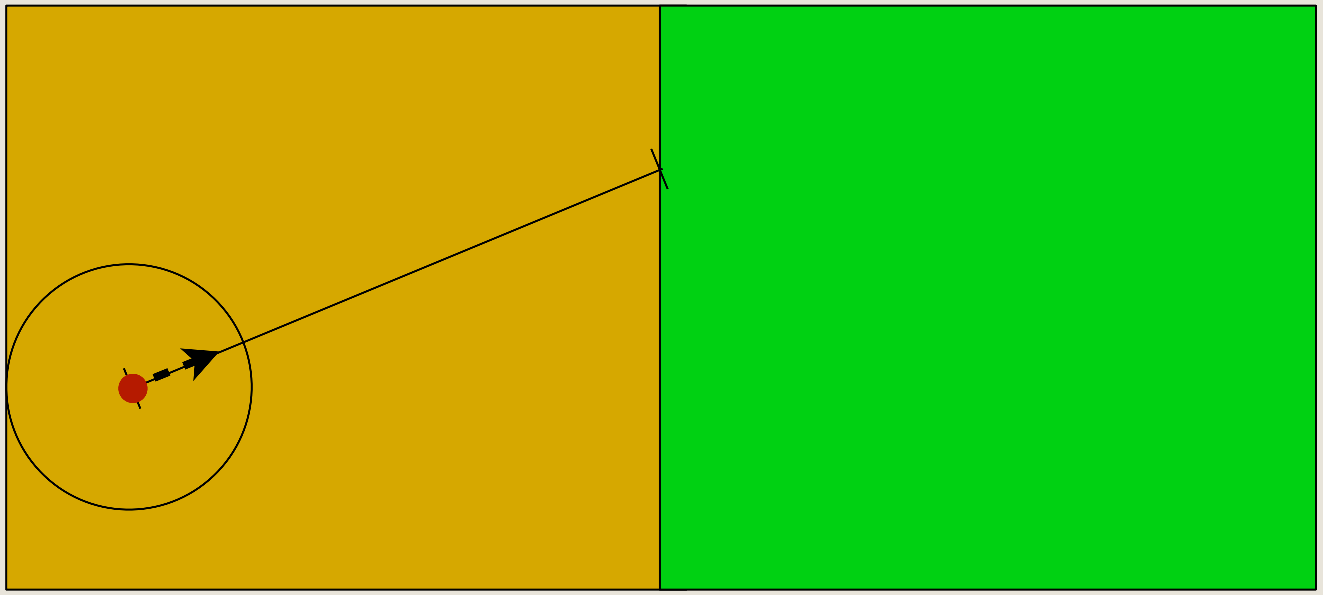
- Particle in flight
- Where am I?

Transport in a nutshell



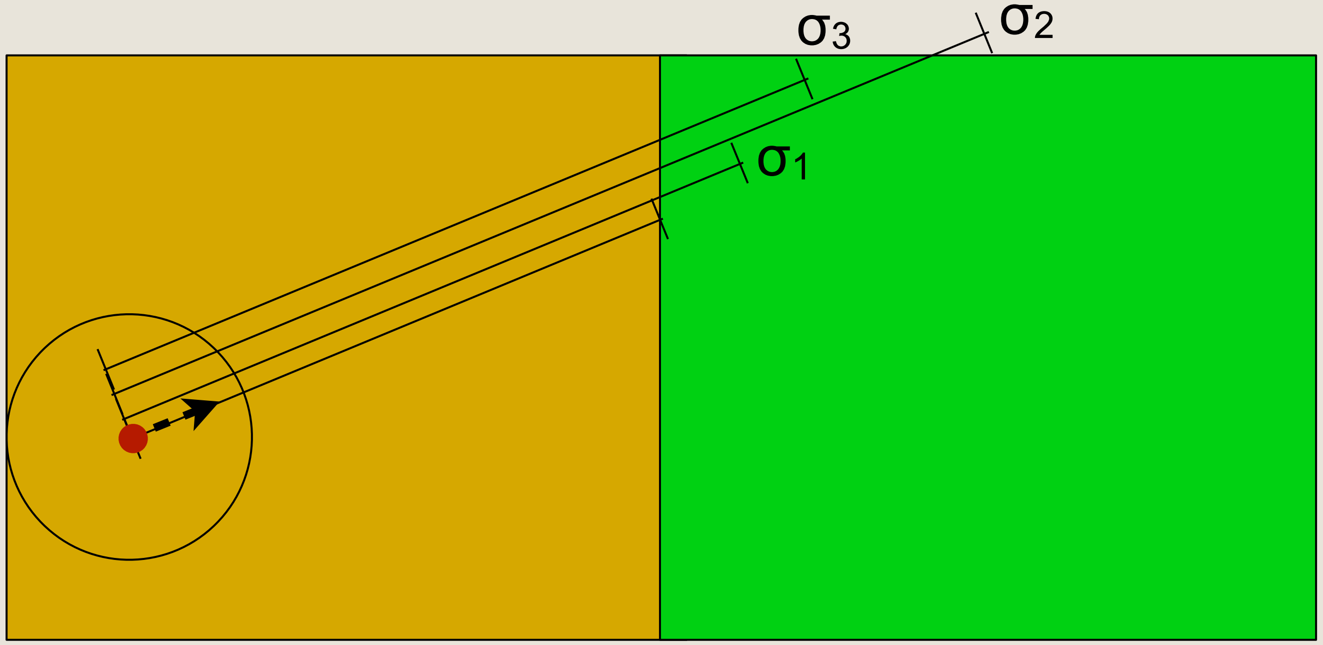
- Particle in flight
- Where am I?
- What is my "safety radius"?

Transport in a nutshell



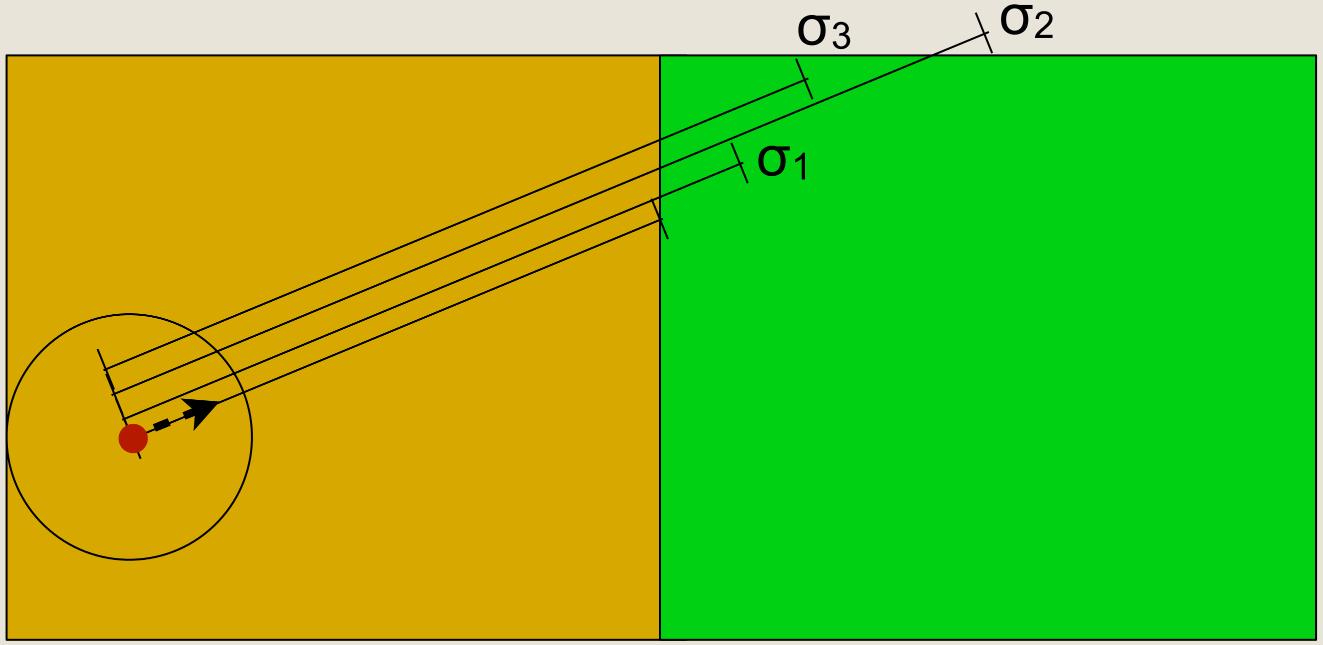
- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?

Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

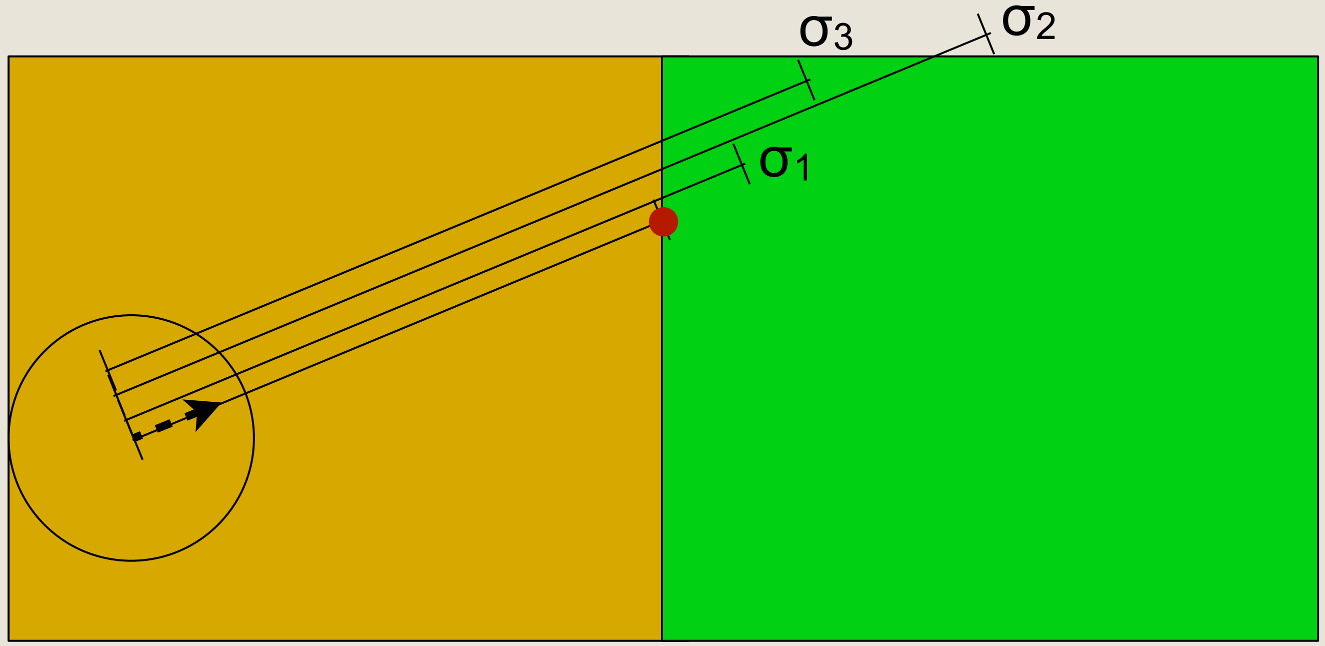
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins

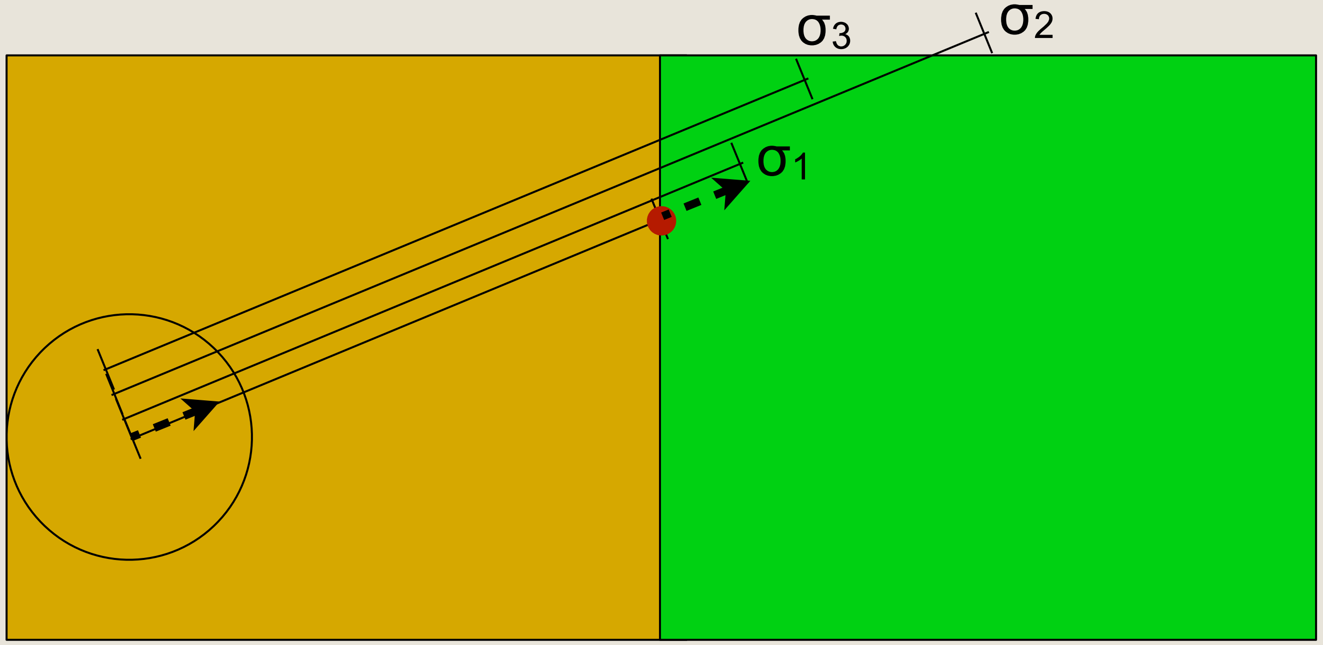
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins
- Transport to boundary

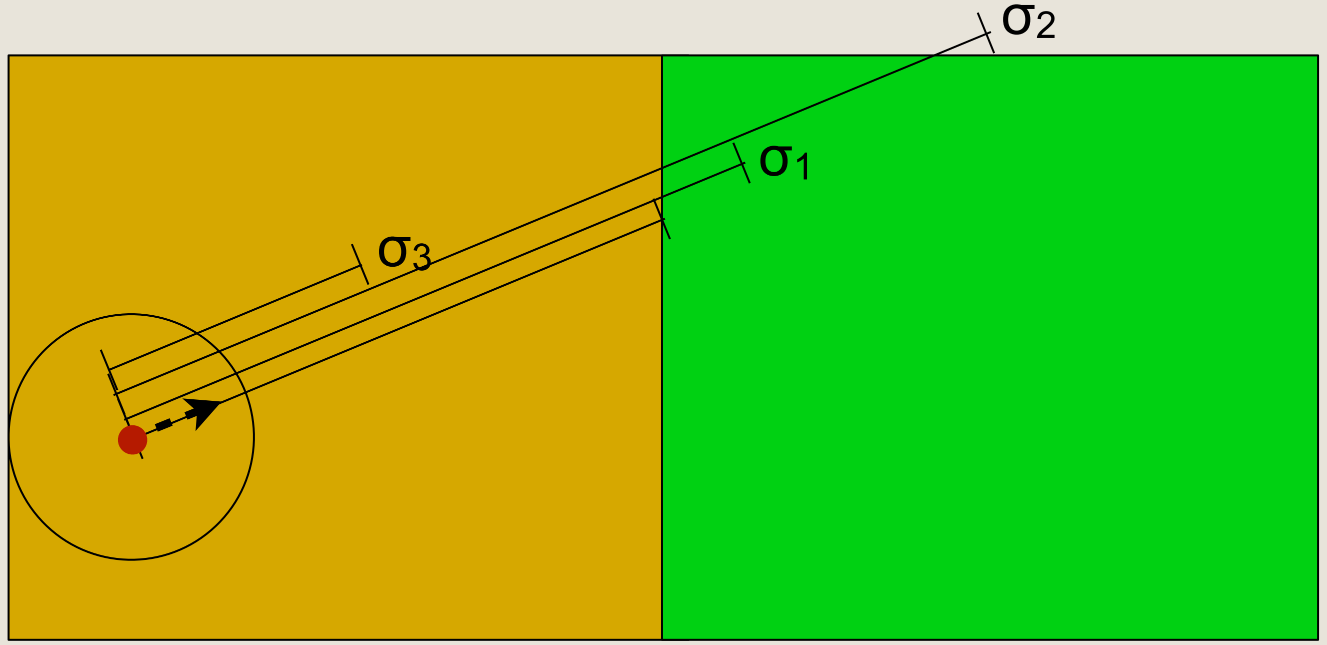
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins
- Transport to boundary
- Continue transport

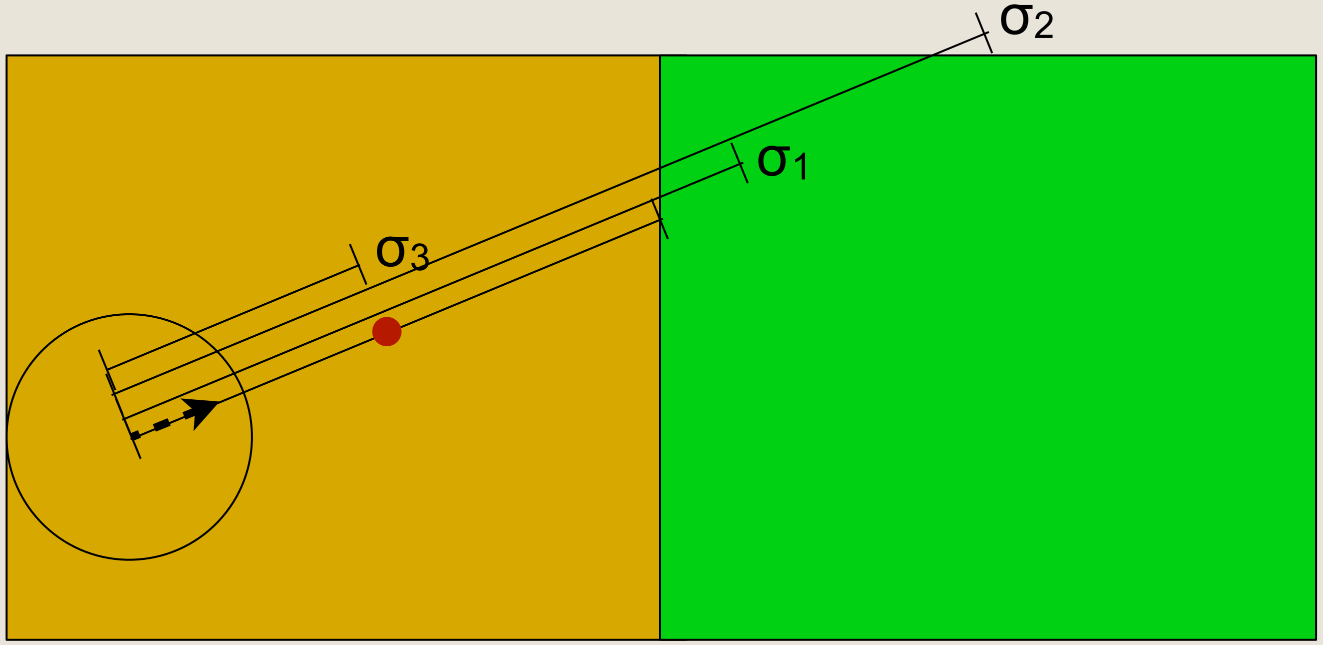
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins
- Transport to boundary
- Continue transport
- Physics wins

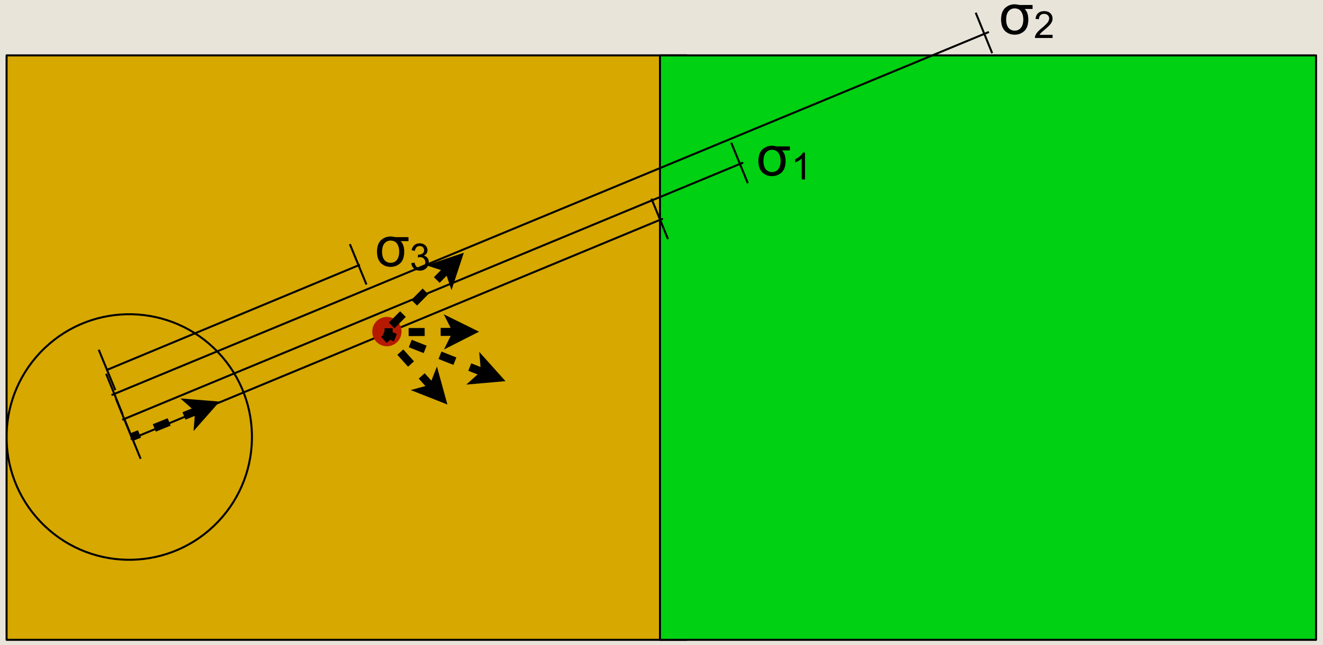
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins
- Transport to boundary
- Continue transport
- Physics wins
- Transport to interaction point

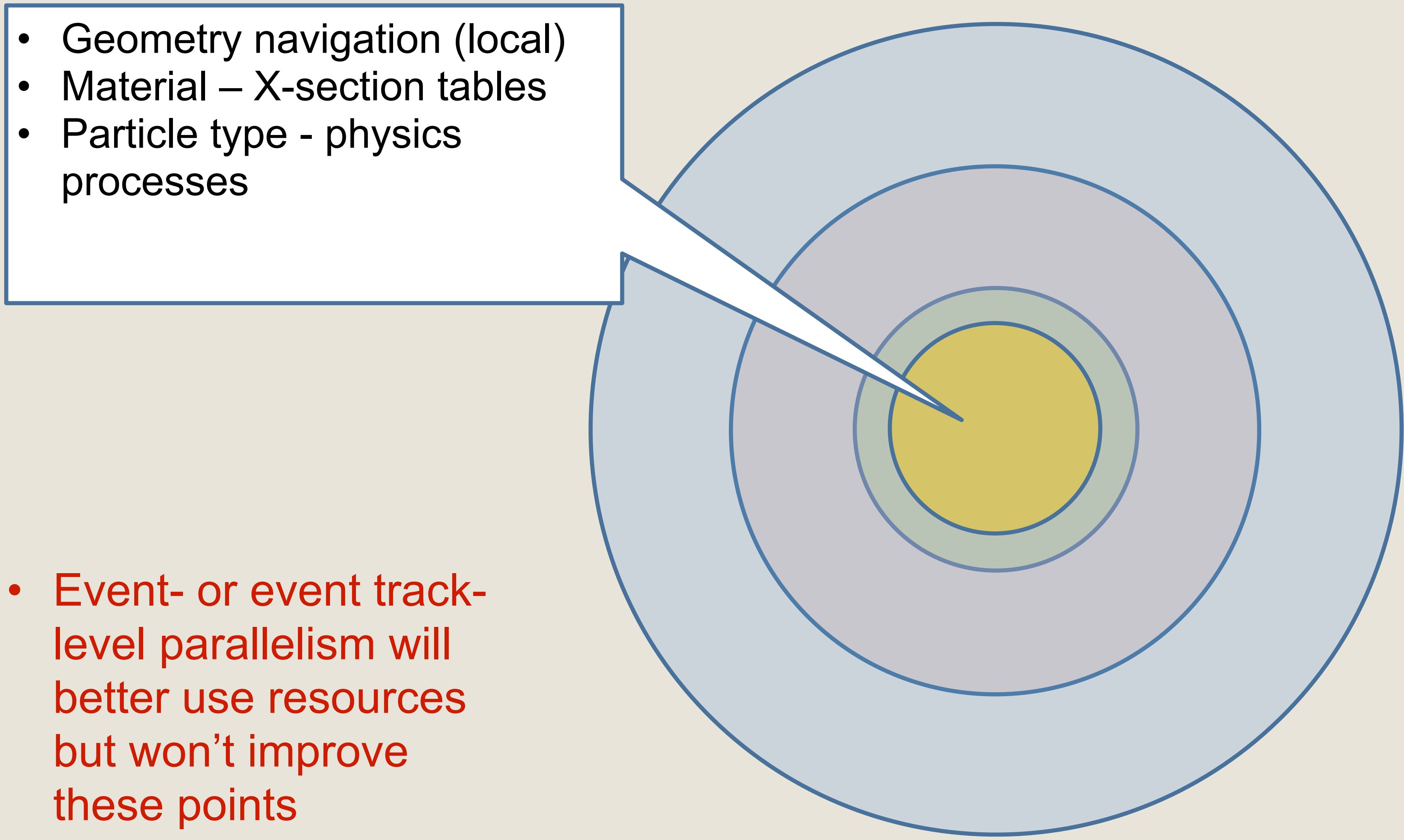
Transport in a nutshell



- Particle in flight
- Where am I?
- What is my “safety radius”?
- What is the distance to the boundary?
- What is the sampled distance to physics interaction?

- Geometry wins
- Transport to boundary
- Continue transport
- Physics wins
- Transport to interaction point
- Generate process

Classical particle transport



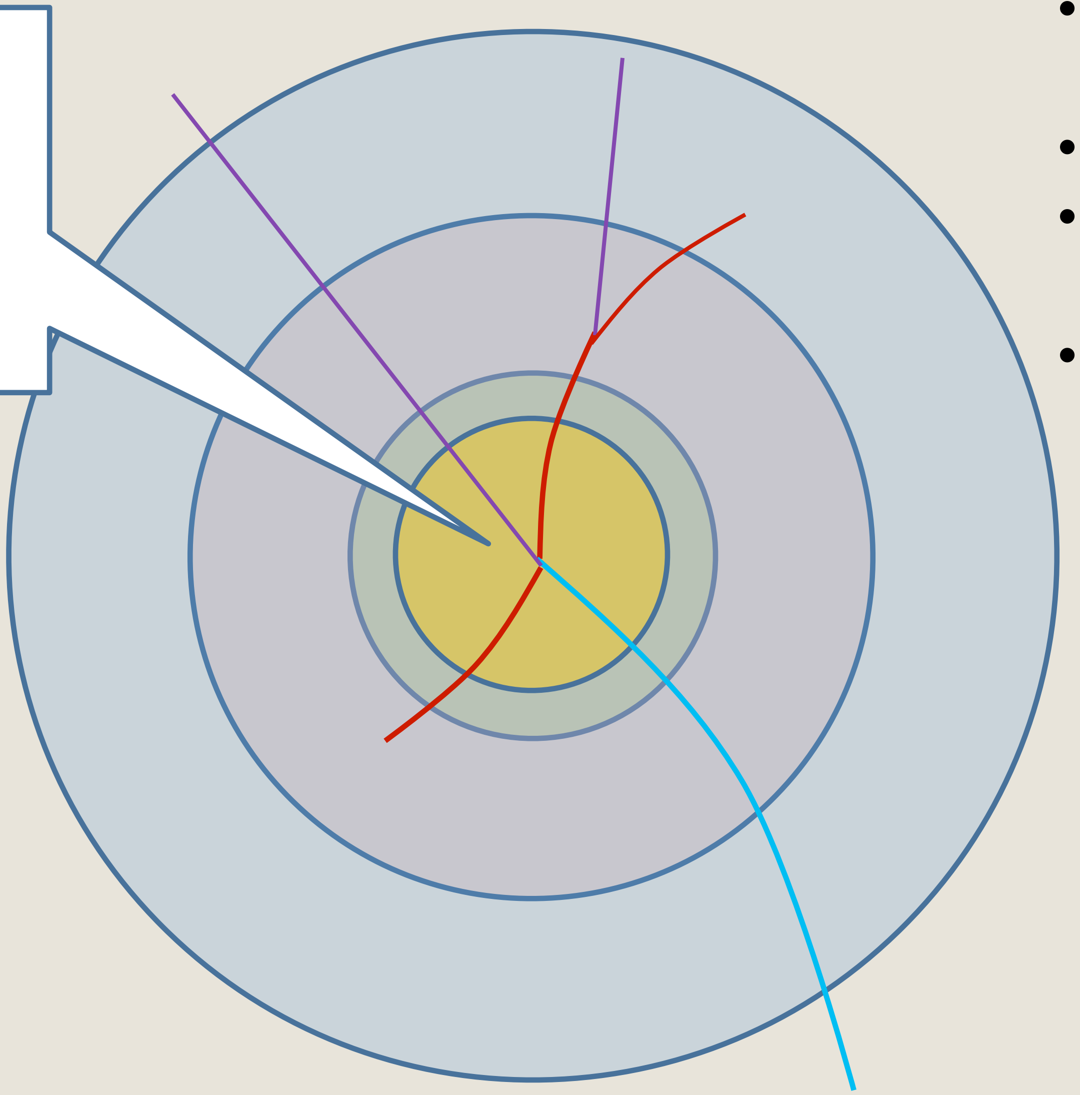
- Geometry navigation (local)
- Material – X-section tables
- Particle type - physics processes

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

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

Classical particle transport

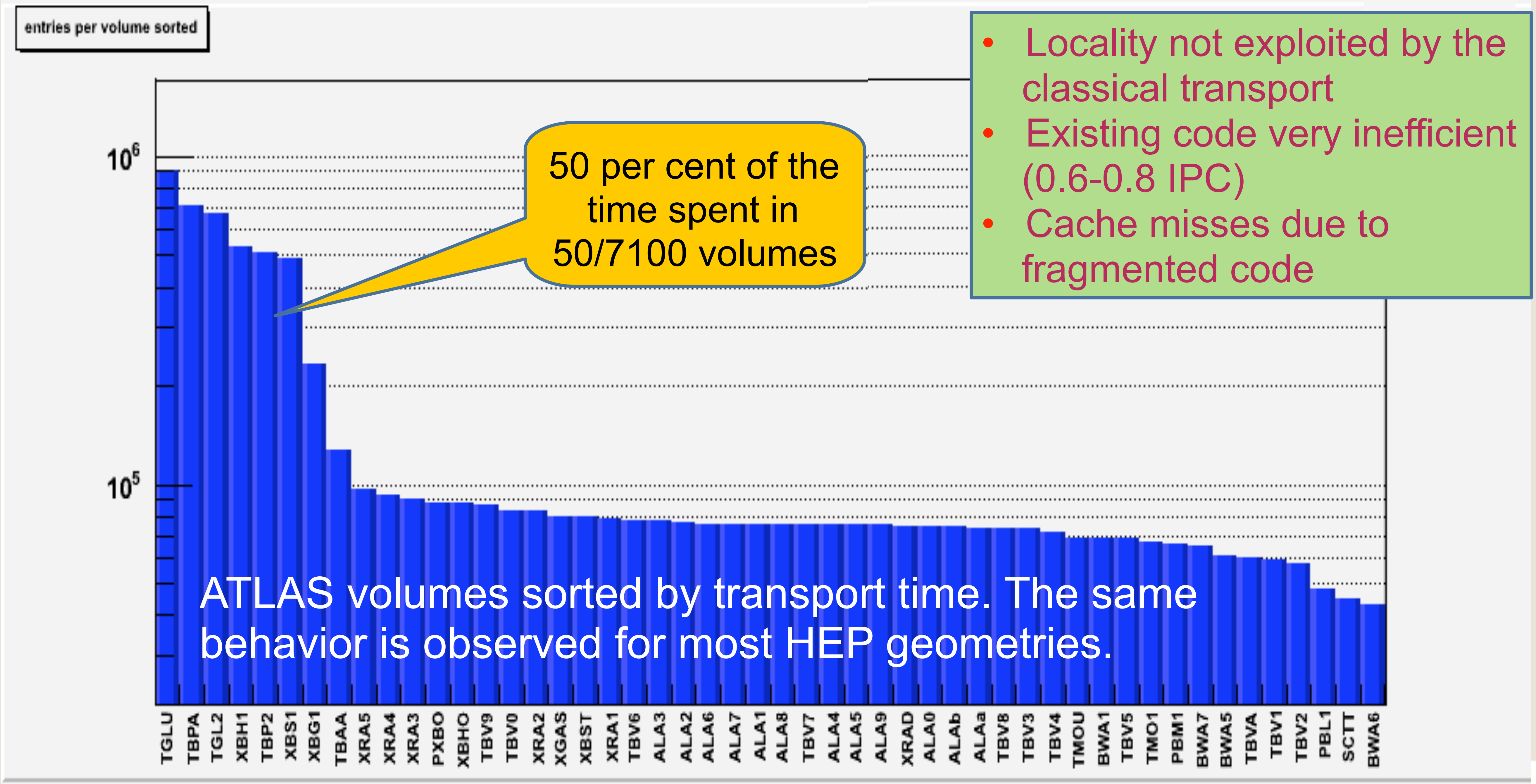
- Geometry navigation (local)
- Material – X-section tables
- Particle type - physics processes



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

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

HEP transport is mostly local !



50 per cent of the time spent in 50/7100 volumes

- Locality not exploited by the classical transport
- Existing code very inefficient (0.6-0.8 IPC)
- Cache misses due to fragmented code

ATLAS volumes sorted by transport time. The same behavior is observed for most HEP geometries.

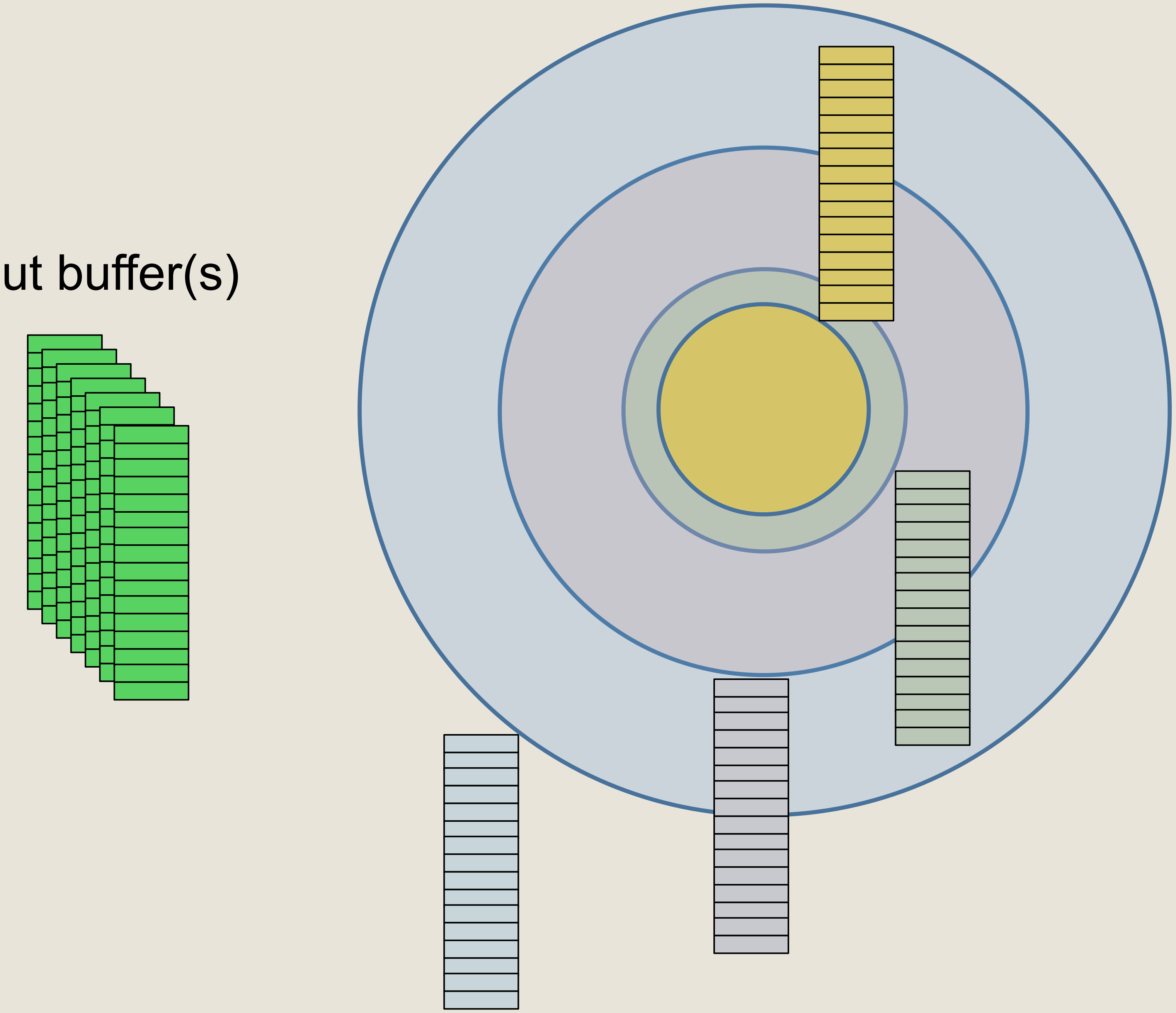
A playground for new ideas

- Explore parallelism and efficiency issues ⇒ can we
  - Implement a parallel transport exploiting data locality and vectorisation?
  - Achieve a continuous data flow from generation to digitization and I/O
- Events and tracks are independent
  - Transport together a vector of tracks from many events
  - Study how does scatter/gather of vectors (tracks, hits…) impact performance
- Particles traversing a volume are transported together until it is empty
  - Same volume ➔ local (vs. global) navigation, same material and same x-sections
  - Load balance: distribute particles in a logical-volume into work units (baskets) to be transported by one thread
  - Work with vectors to allow for micro-parallelism

- Particles exiting a volume are distributed to baskets of neighbor volumes until they exit the setup or disappear
  - Like a champagne cascade, but lower glasses can also fill higher one
  - Wait for the glass to be full before drinking the champagne…

"Basketised" transport

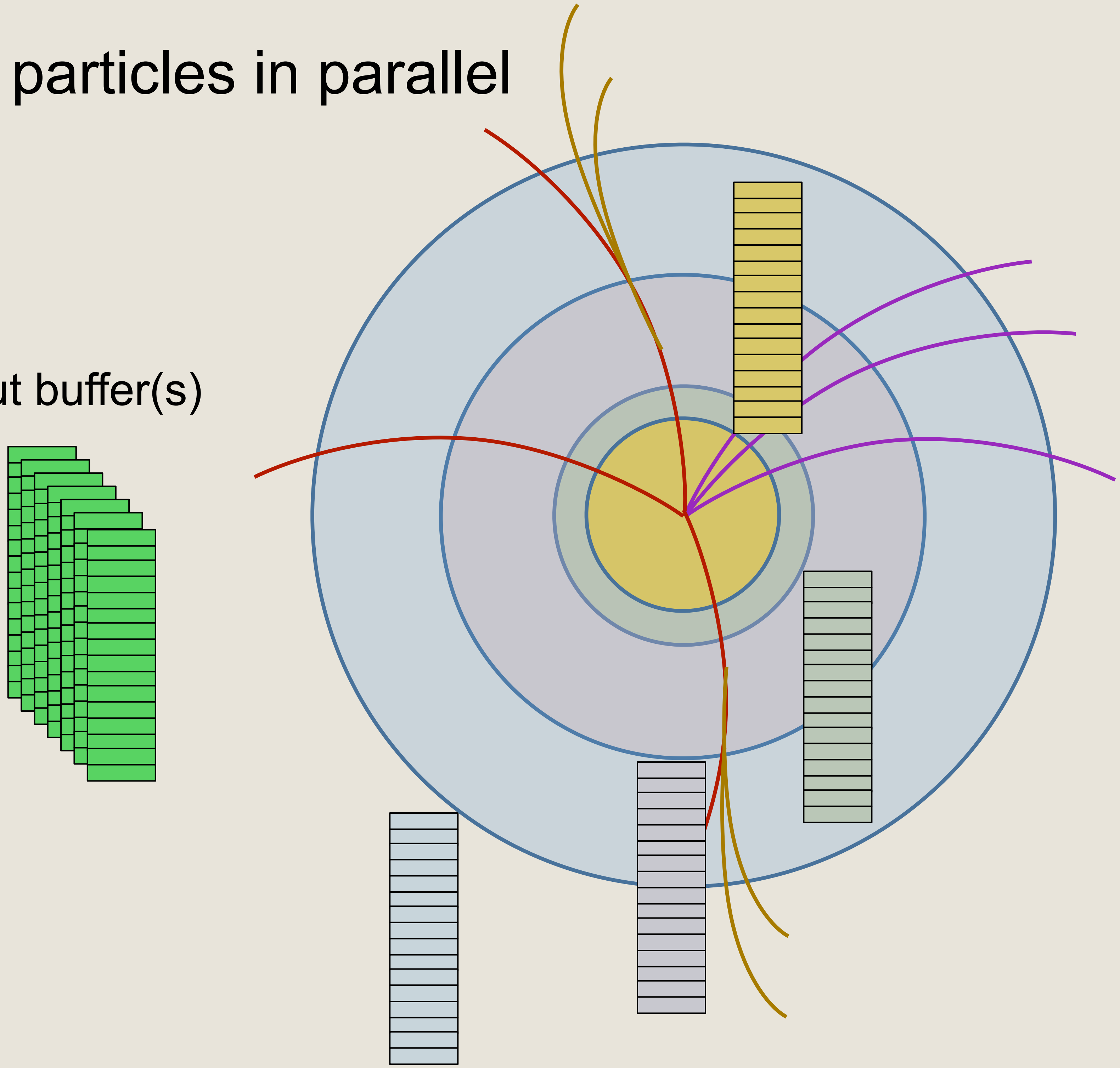
Deal with particles in parallel



“Basketised” transport

Deal with particles in parallel

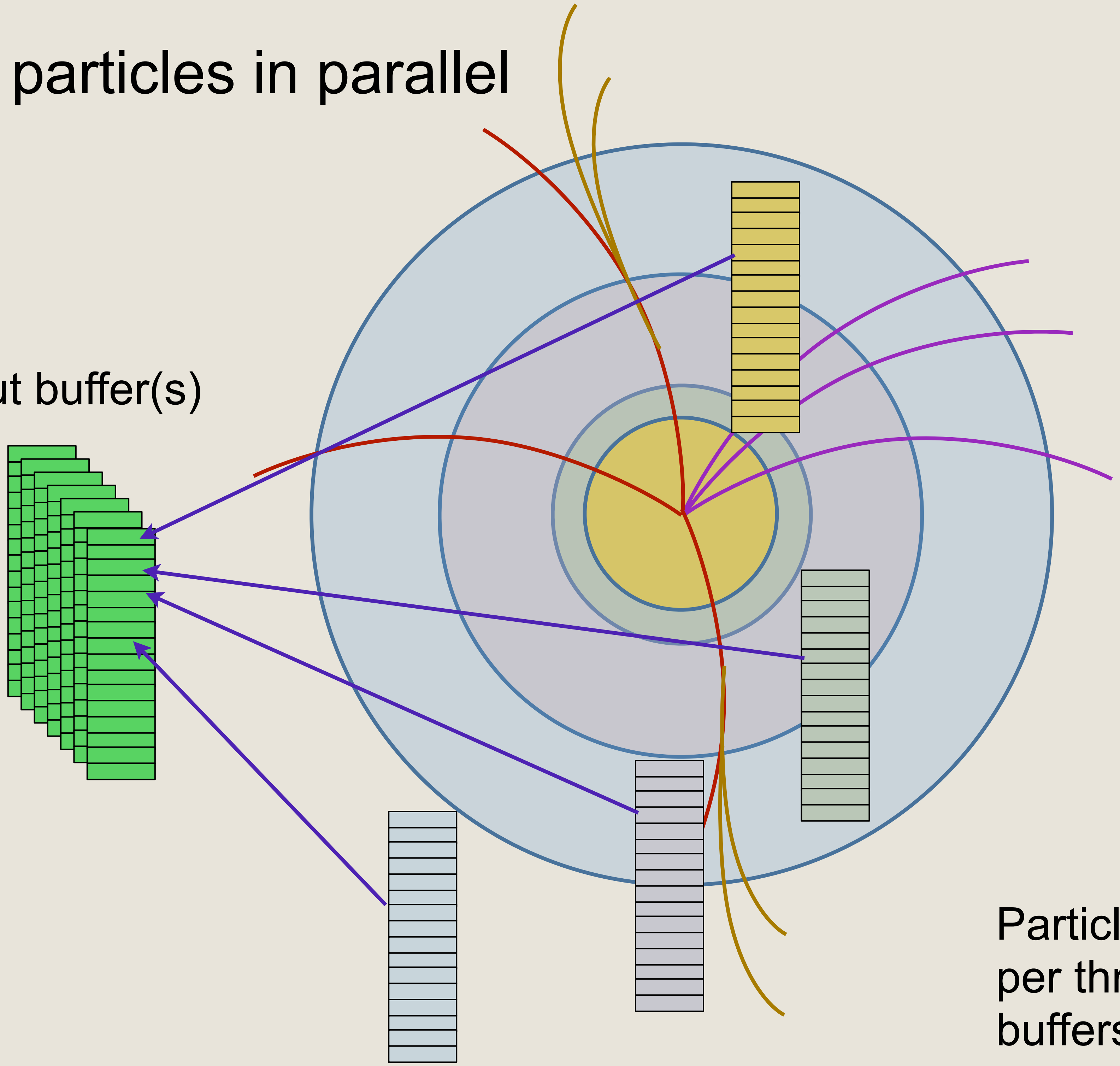
Output buffer(s)



“Basketised” transport

Deal with particles in parallel

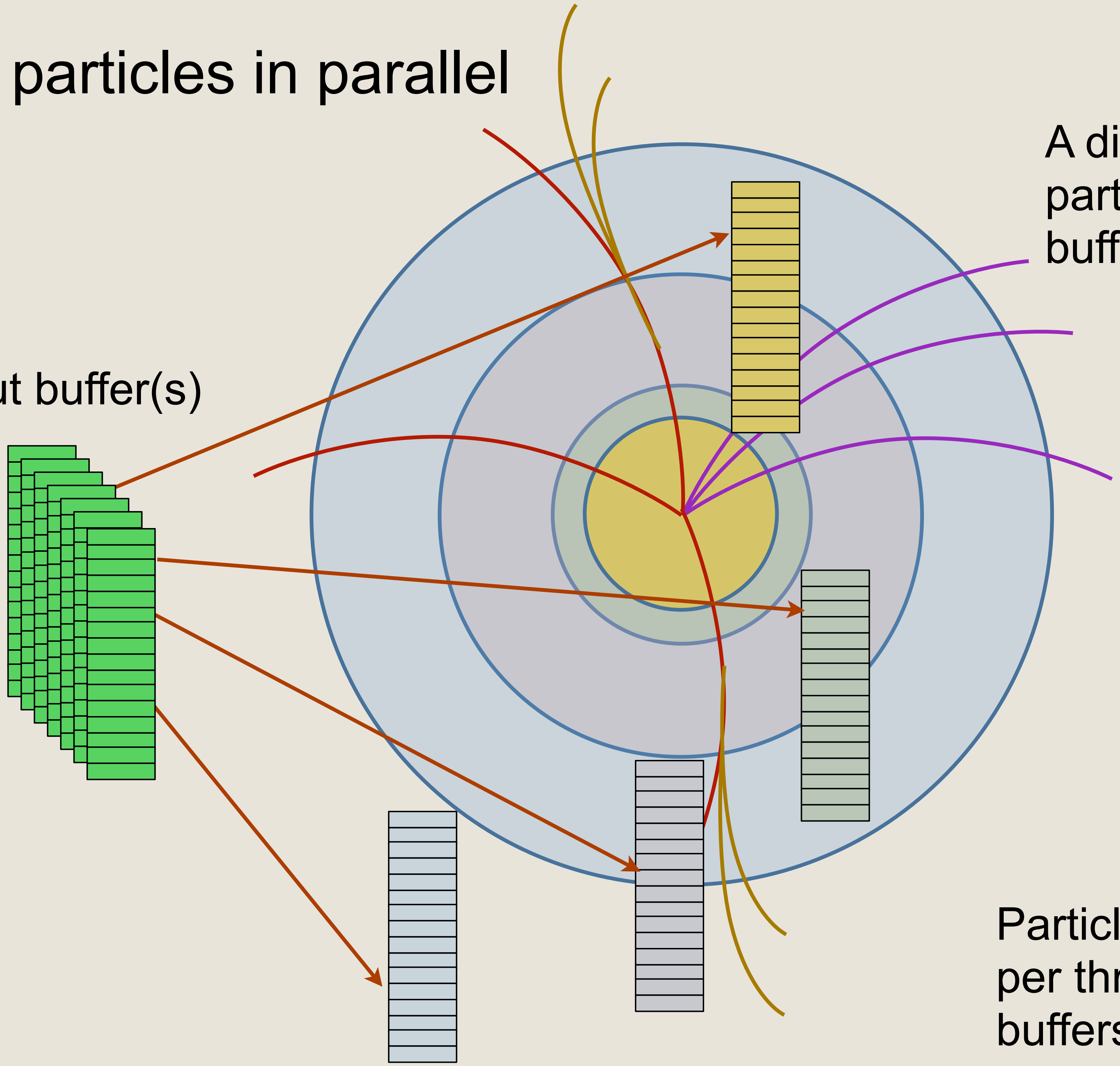
Output buffer(s)



Particles are transported per thread and put in output buffers

"Basketised" transport

Deal with particles in parallel



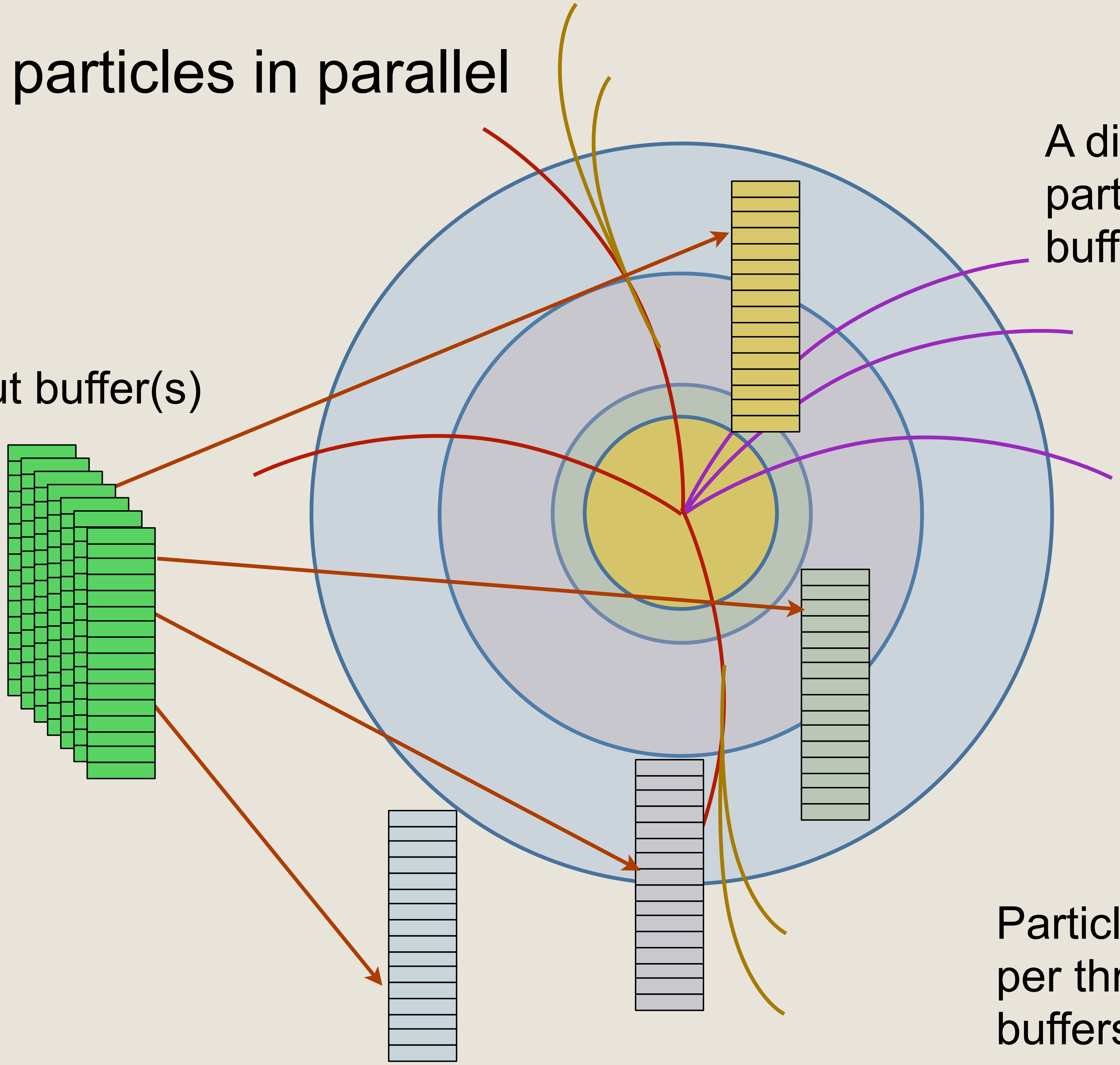
Output buffer(s)

A dispatcher thread puts particles back into transport buffers

Particles are transported per thread and put in output buffers

“Basketised” transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

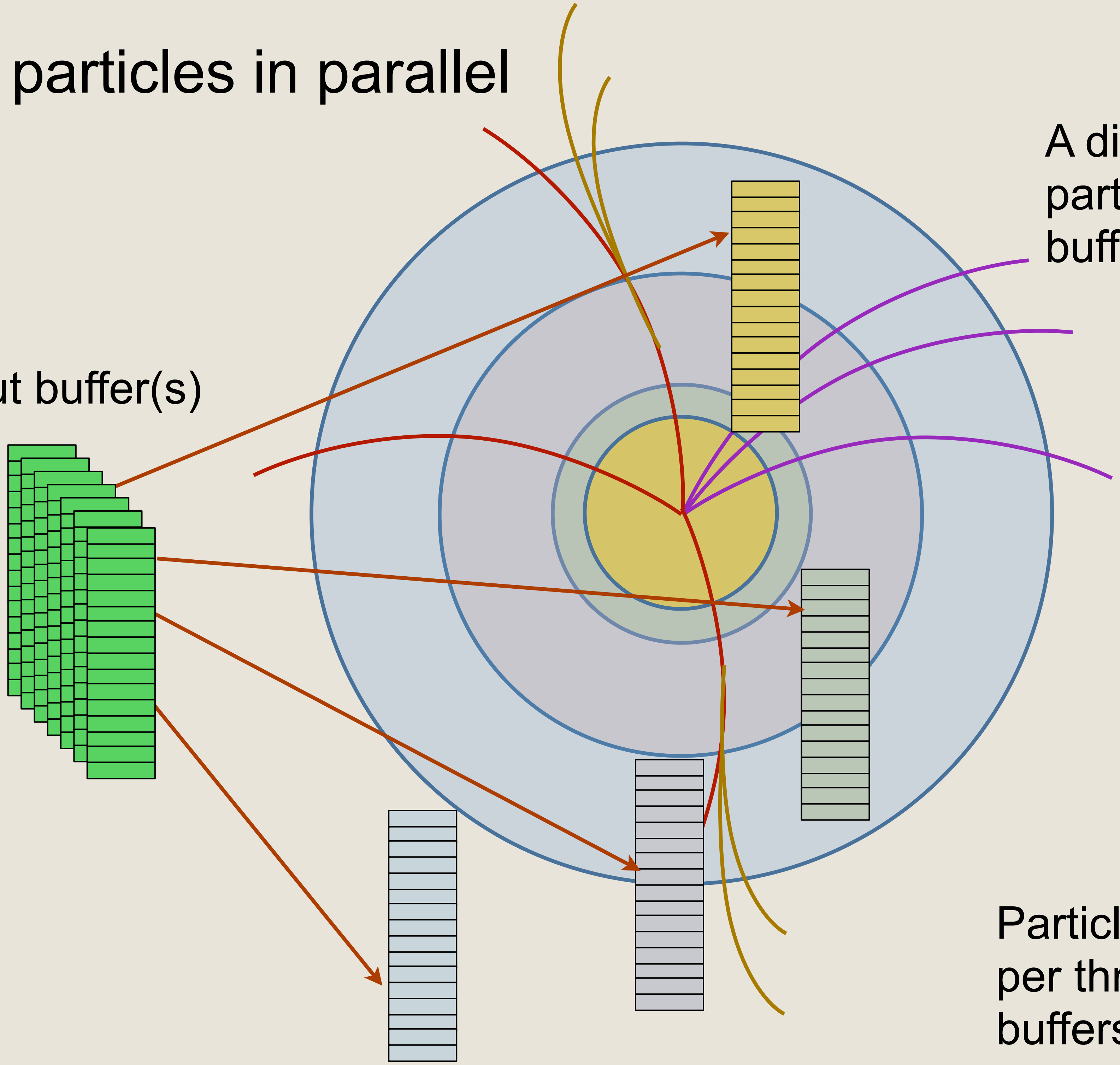
Everything happens asynchronously and in parallel

Output buffer(s)

Particles are transported per thread and put in output buffers

"Basketised" transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

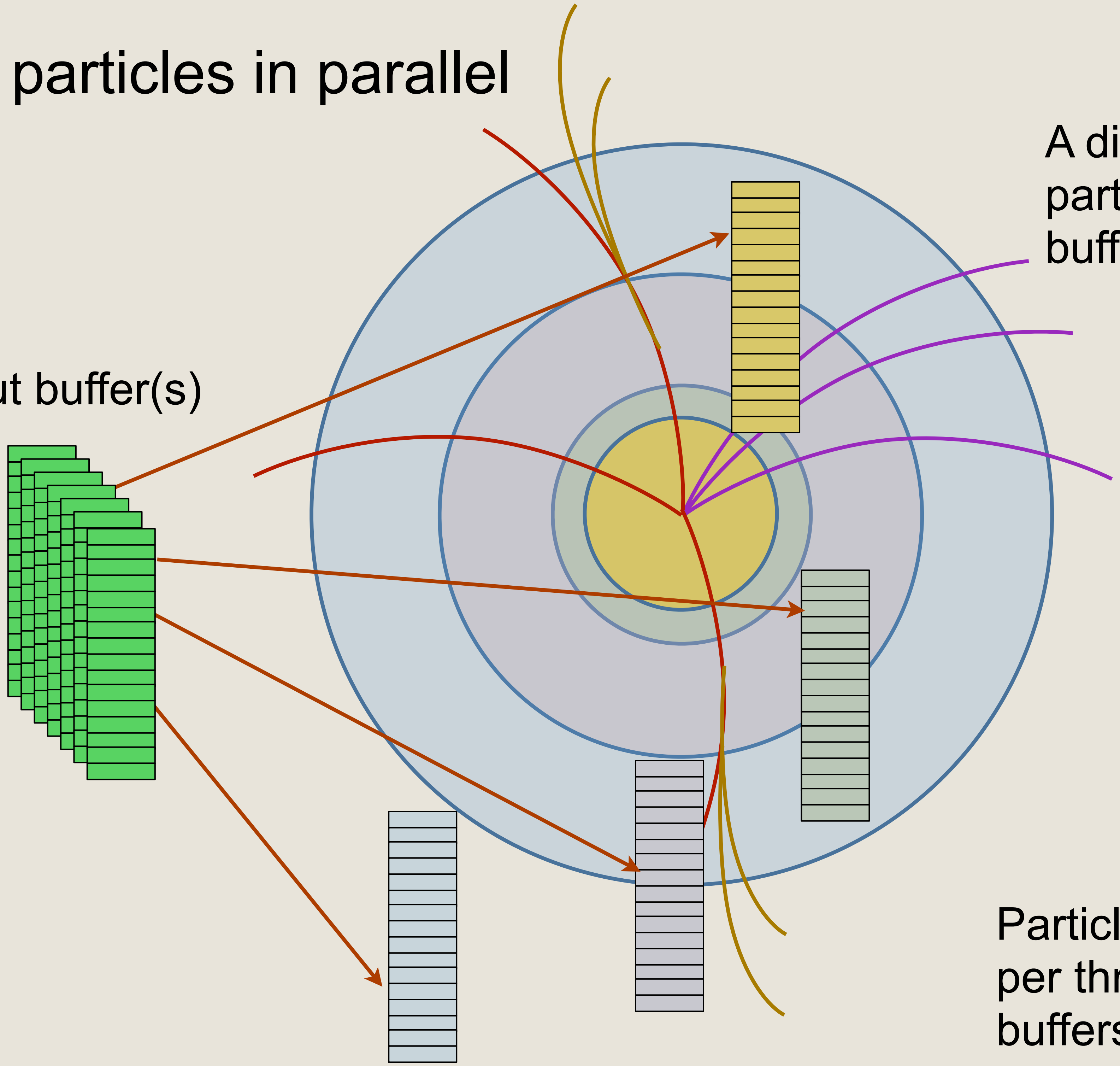
Everything happens asynchronously and in parallel

The challenge is to minimise locks

Particles are transported per thread and put in output buffers

“Basketised” transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

Everything happens asynchronously and in parallel

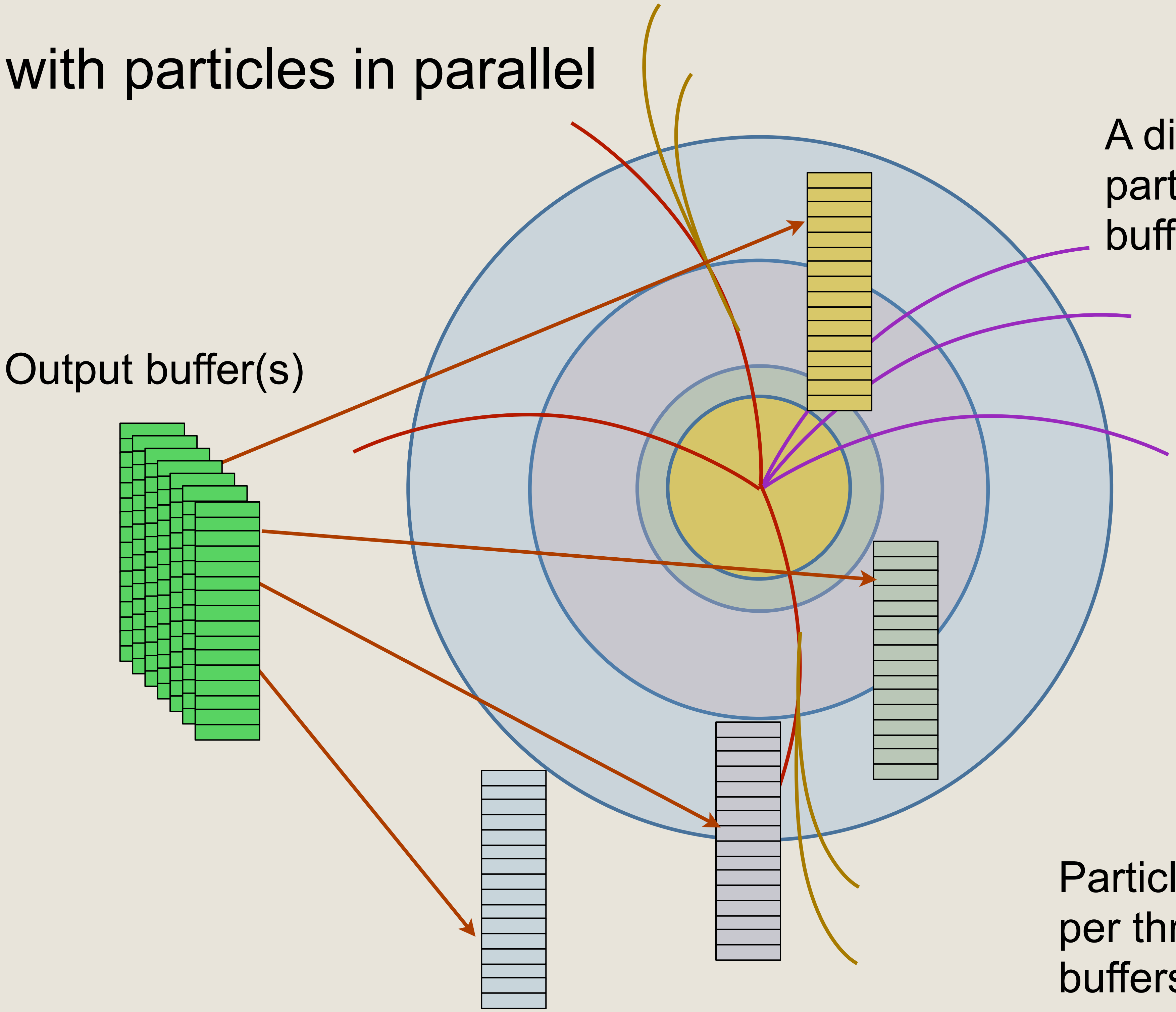
The challenge is to minimise locks

Keep long vectors

Particles are transported per thread and put in output buffers

“Basketised” transport

Deal with particles in parallel



A dispatcher thread puts particles back into transport buffers

Everything happens asynchronously and in parallel

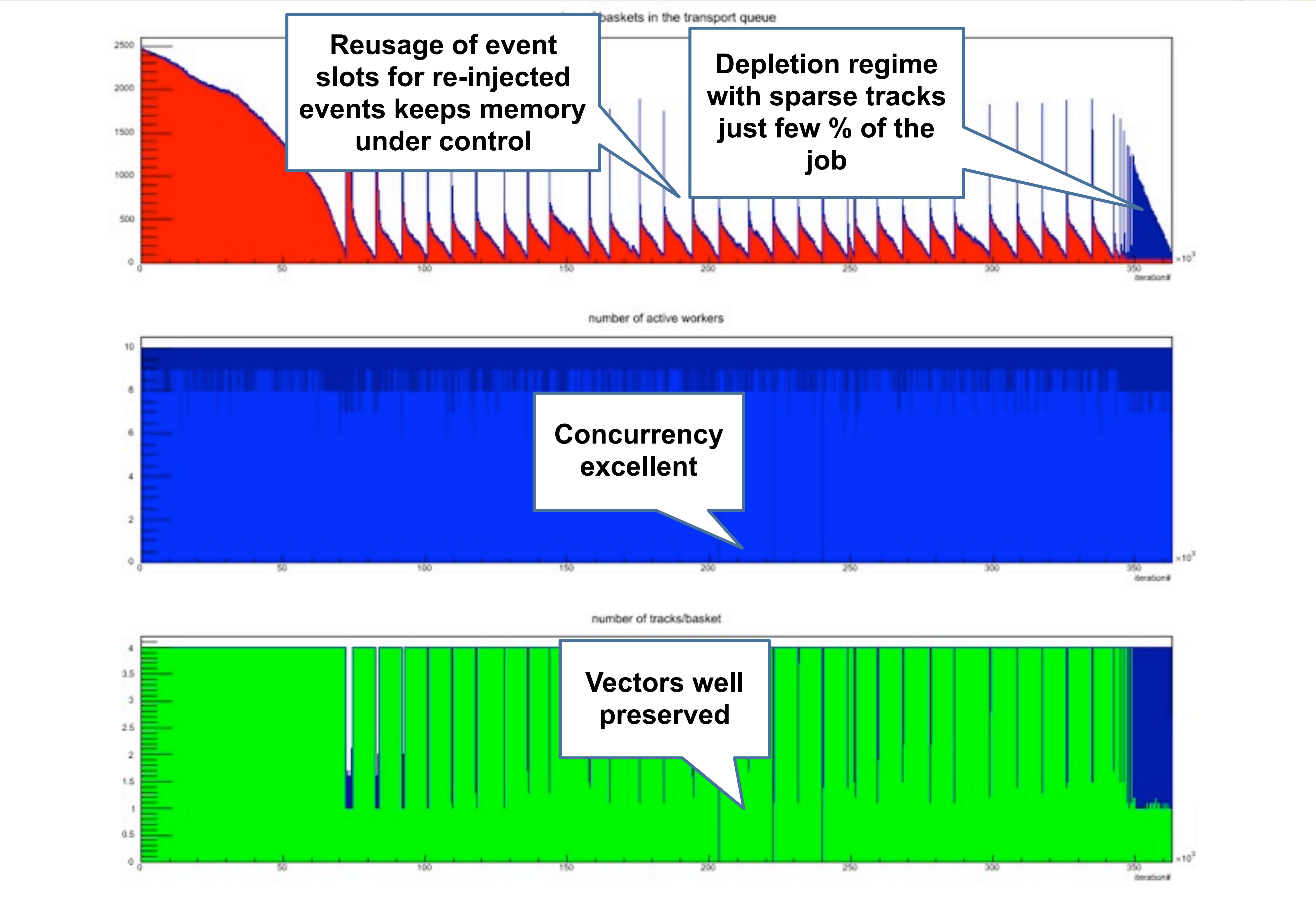
The challenge is to minimise locks

Keep long vectors

Avoid memory explosion

Particles are transported per thread and put in output buffers

Buffered events & re-injection



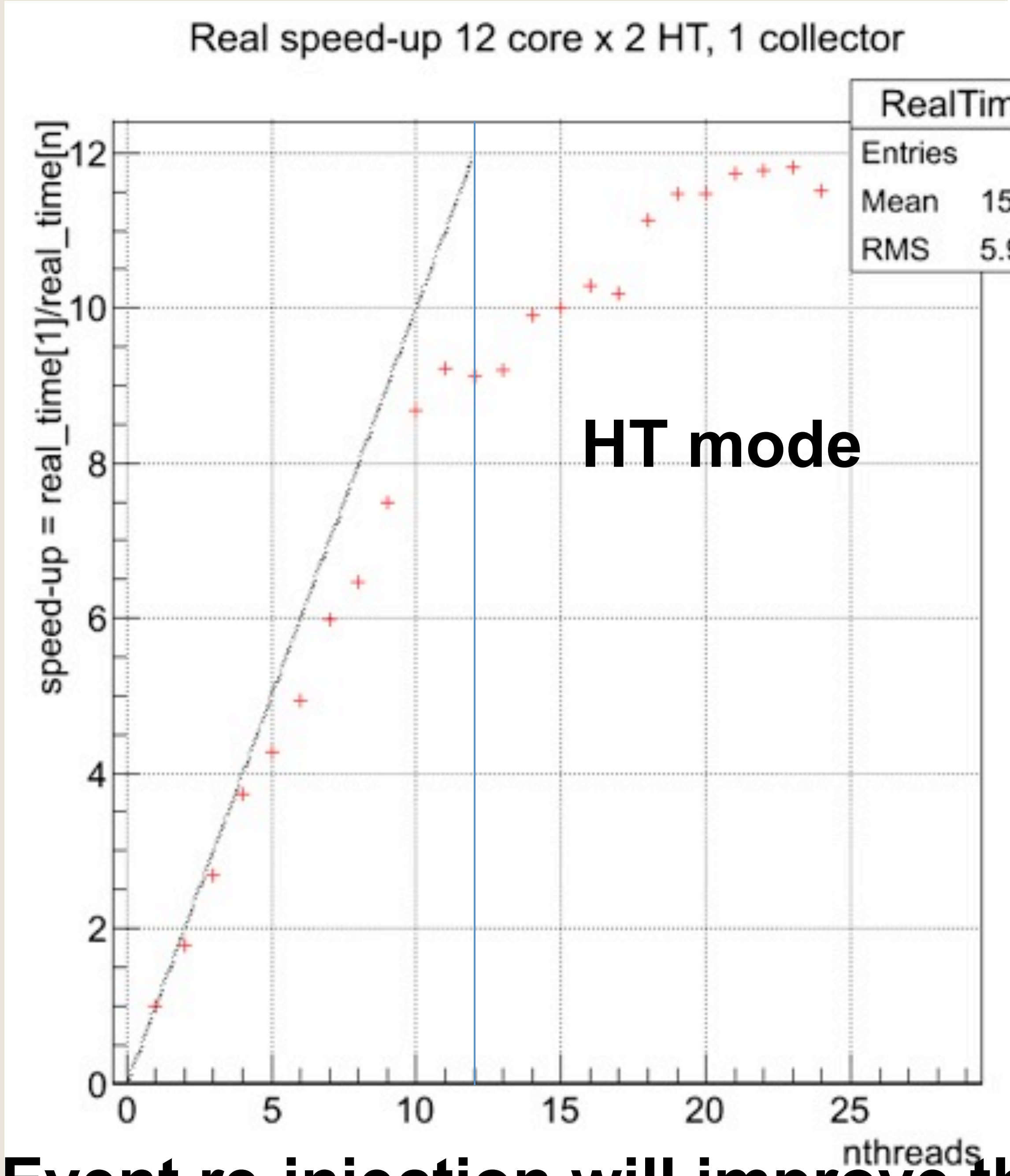
Reusage of event slots for re-injected events keeps memory under control

Depletion regime with sparse tracks just few % of the job

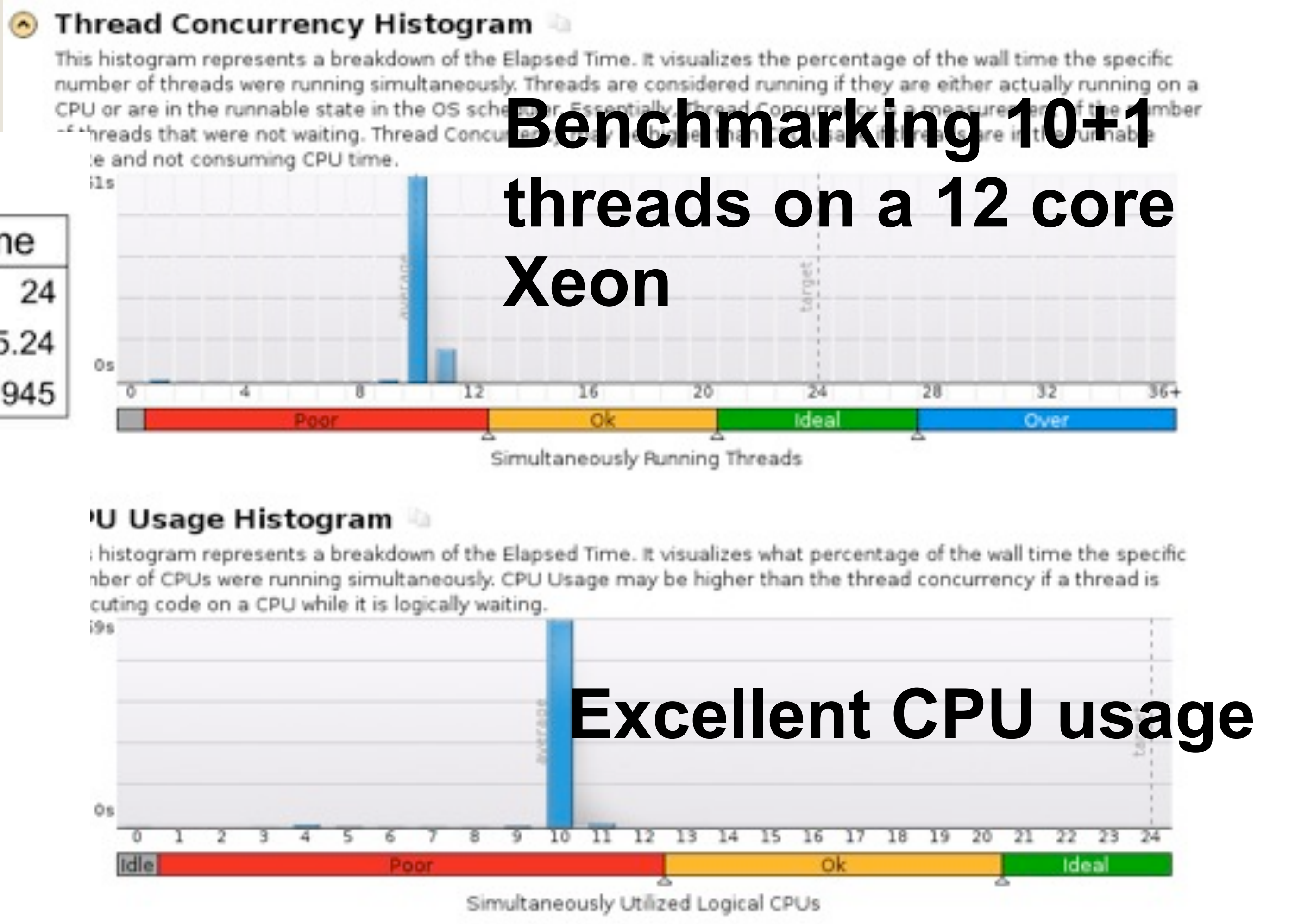
Concurrency excellent

Vectors well preserved

Preliminary benchmarks



HT mode



Thread Concurrency Histogram

This histogram represents a breakdown of the Elapsed Time. It visualizes the percentage of the wall time the specific number of threads were running simultaneously. Threads are considered running if they are either actually running on a CPU or are in the runnable state in the OS sche... of threads that were not waiting. Thread Concu... e and not consuming CPU time.

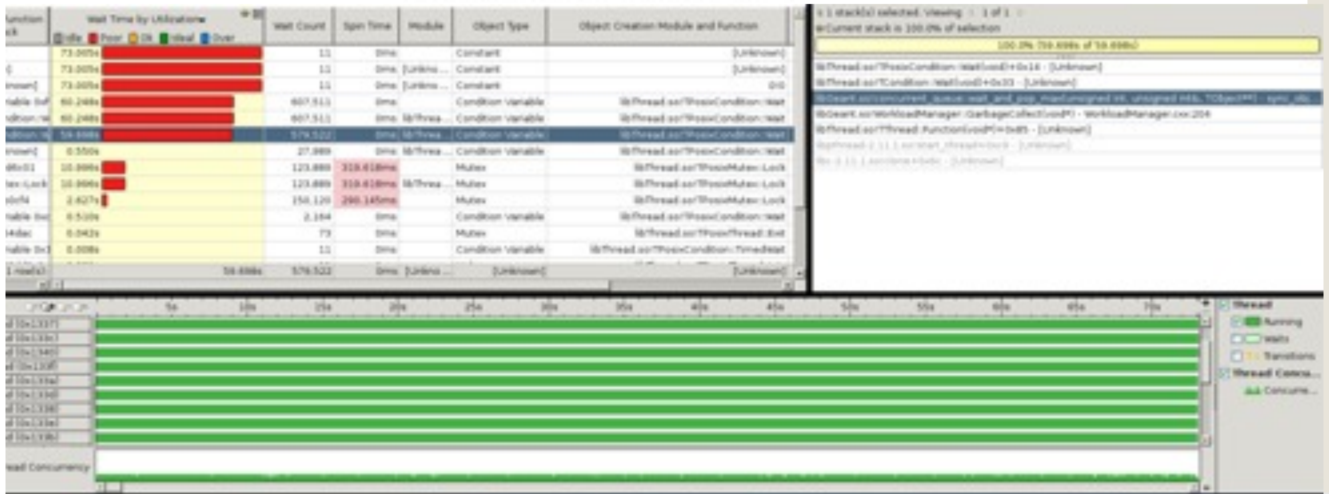
'U Usage Histogram

histogram represents a breakdown of the Elapsed Time. It visualizes what percentage of the wall time the specific ...ber of CPUs were running simultaneously. CPU Usage may be higher than the thread concurrency if a thread is ...cuting code on a CPU while it is logically waiting.

Benchmarking 10+1 threads on a 12 core Xeon

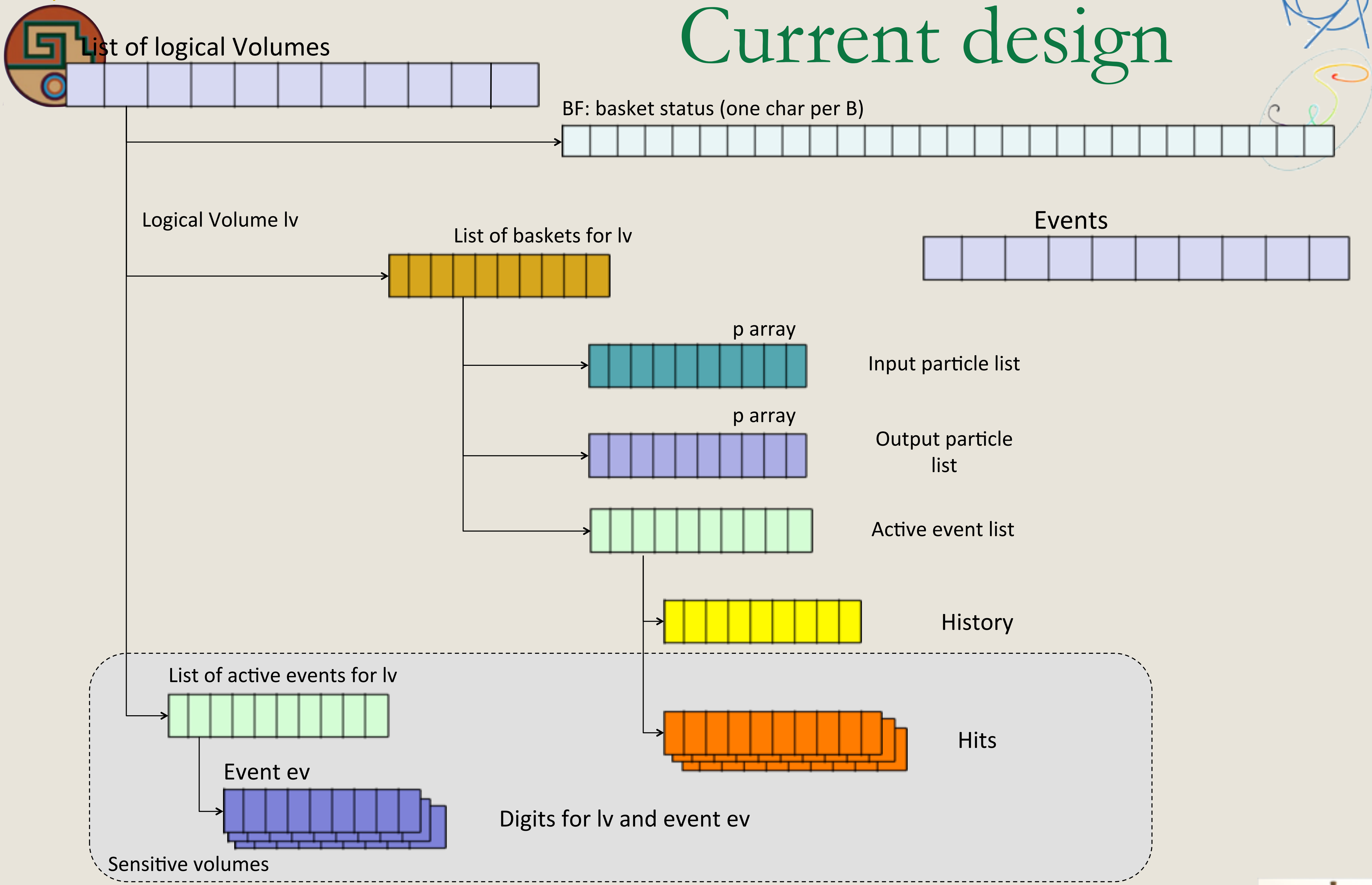
Excellent CPU usage

Event re-injection will improve the speed-up



Locks and waits: some overhead due to transitions coming from exchanging baskets via concurrent queues

Current design



List of logical Volumes

BF: basket status (one char per B)

Logical Volume lv

List of baskets for lv

Events

p array

Input particle list

p array

Output particle list

Active event list

History

List of active events for lv

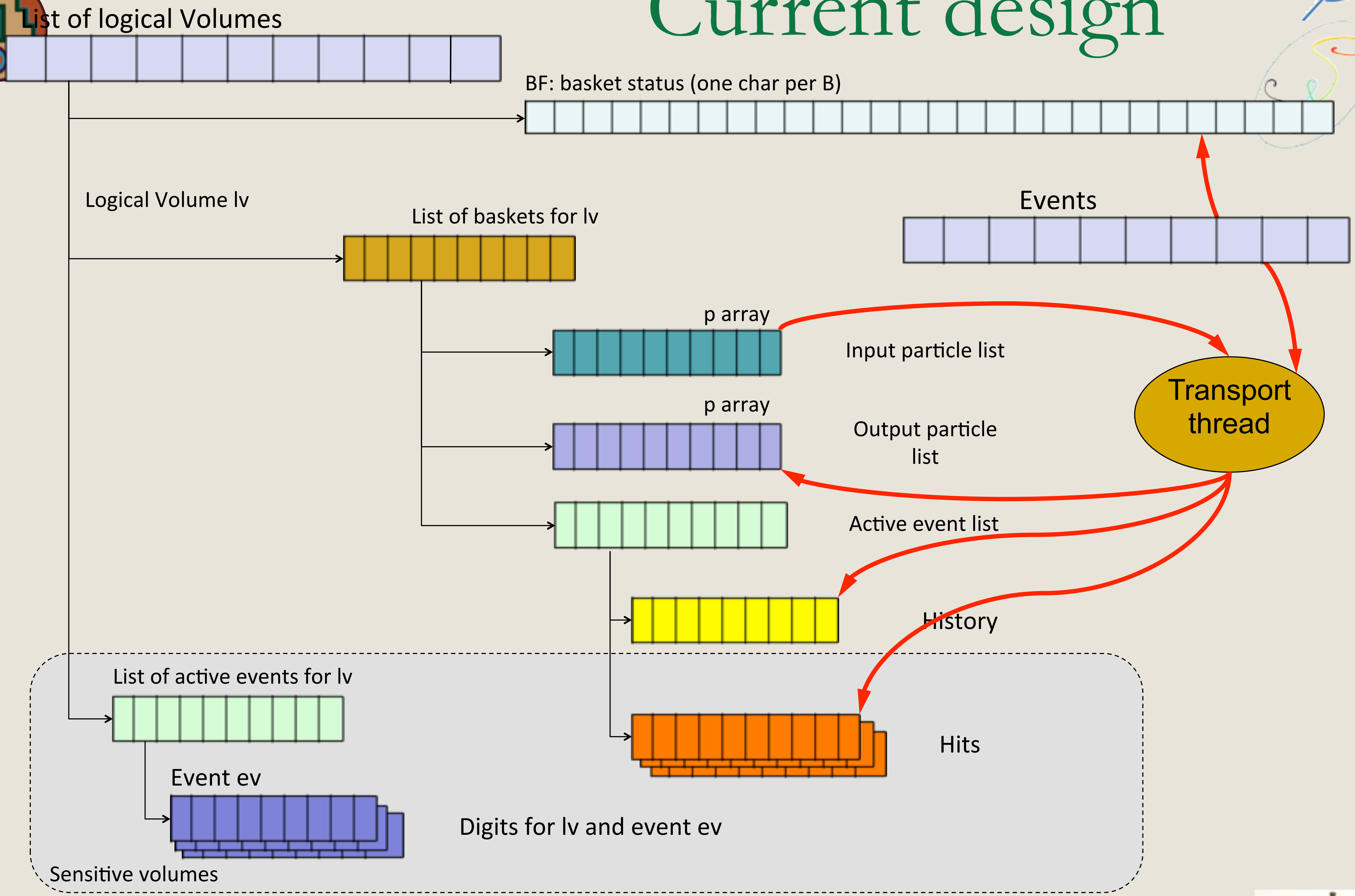
Hits

Event ev

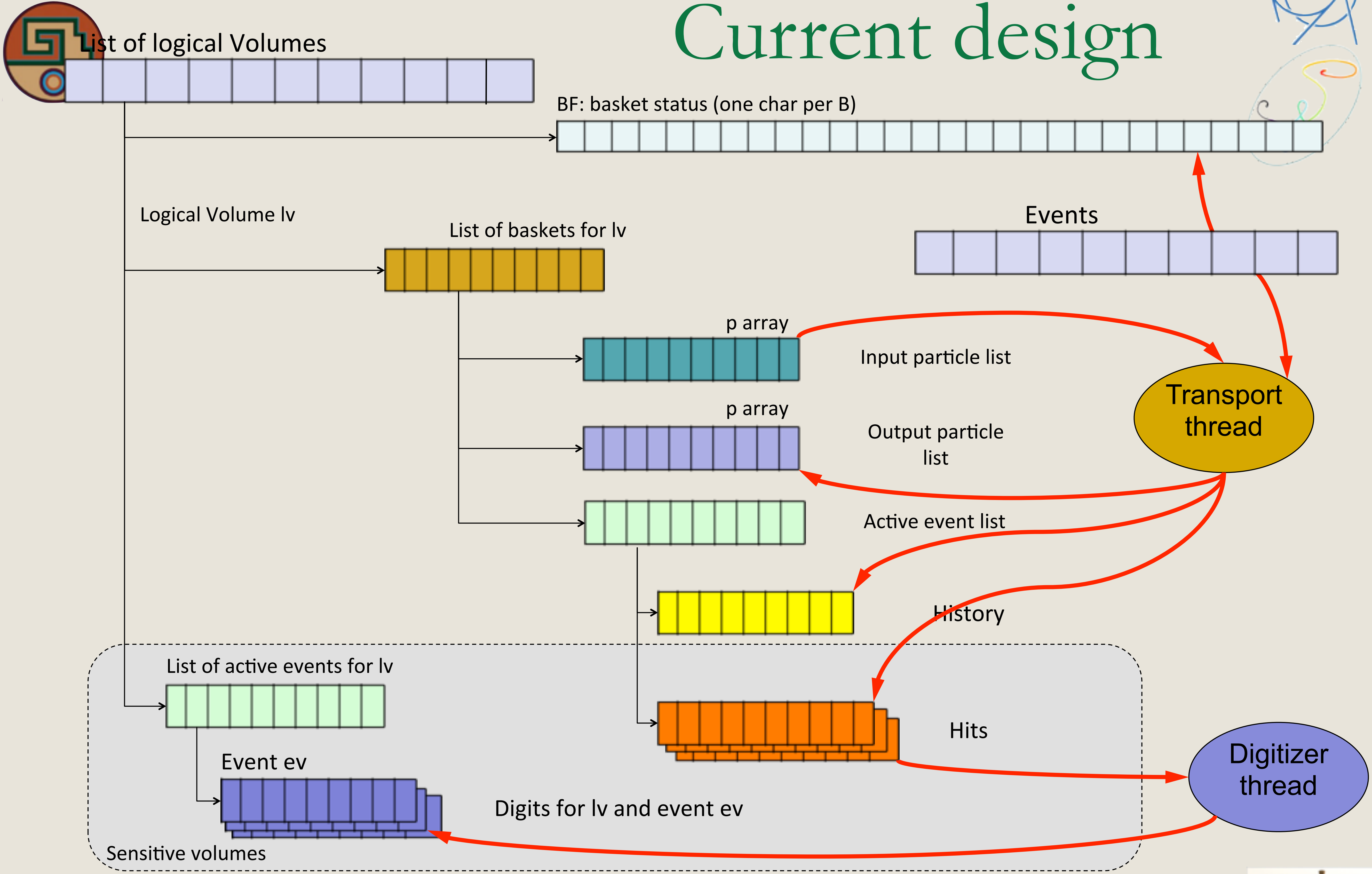
Digits for lv and event ev

Sensitive volumes

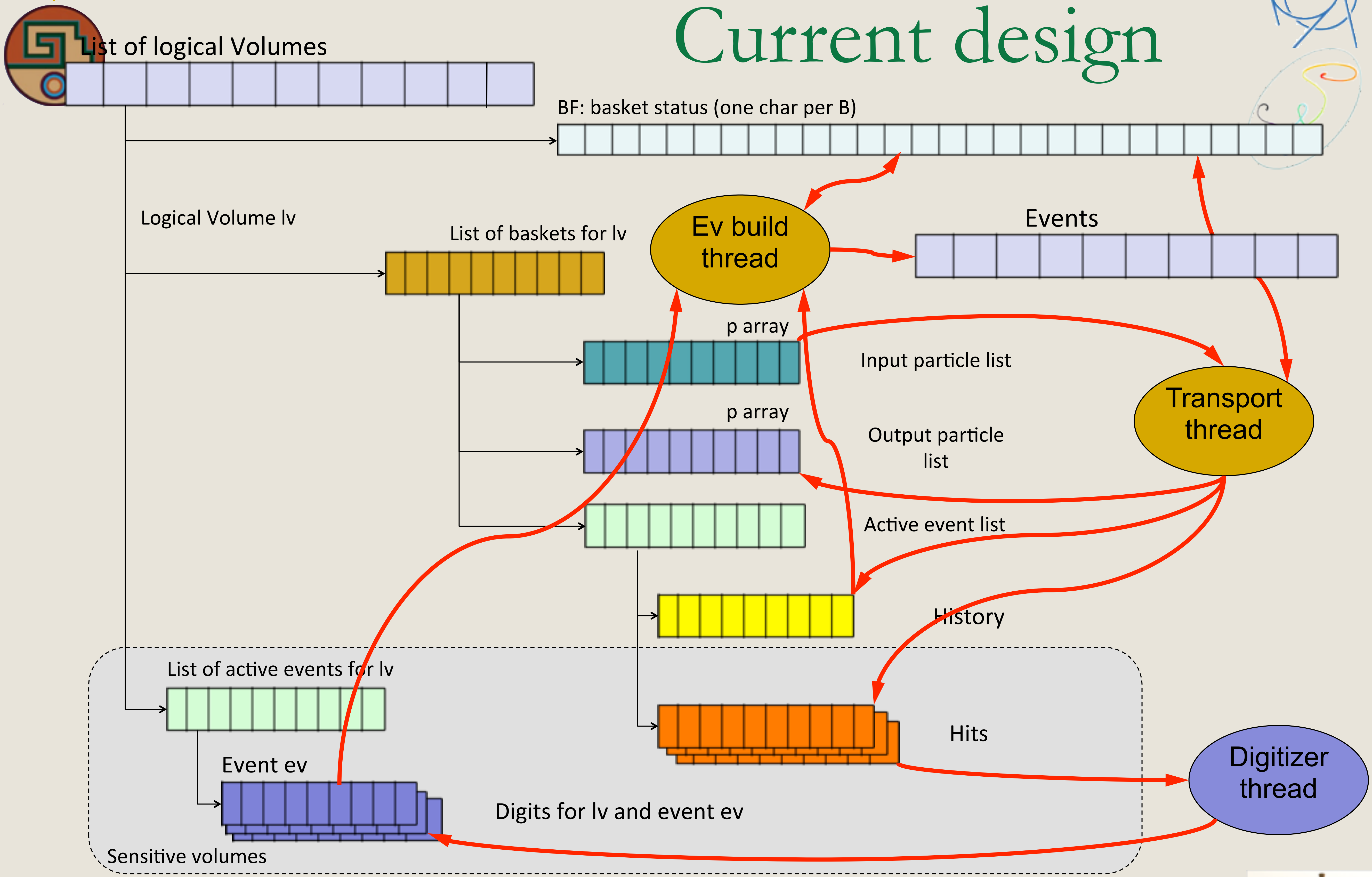
Current design



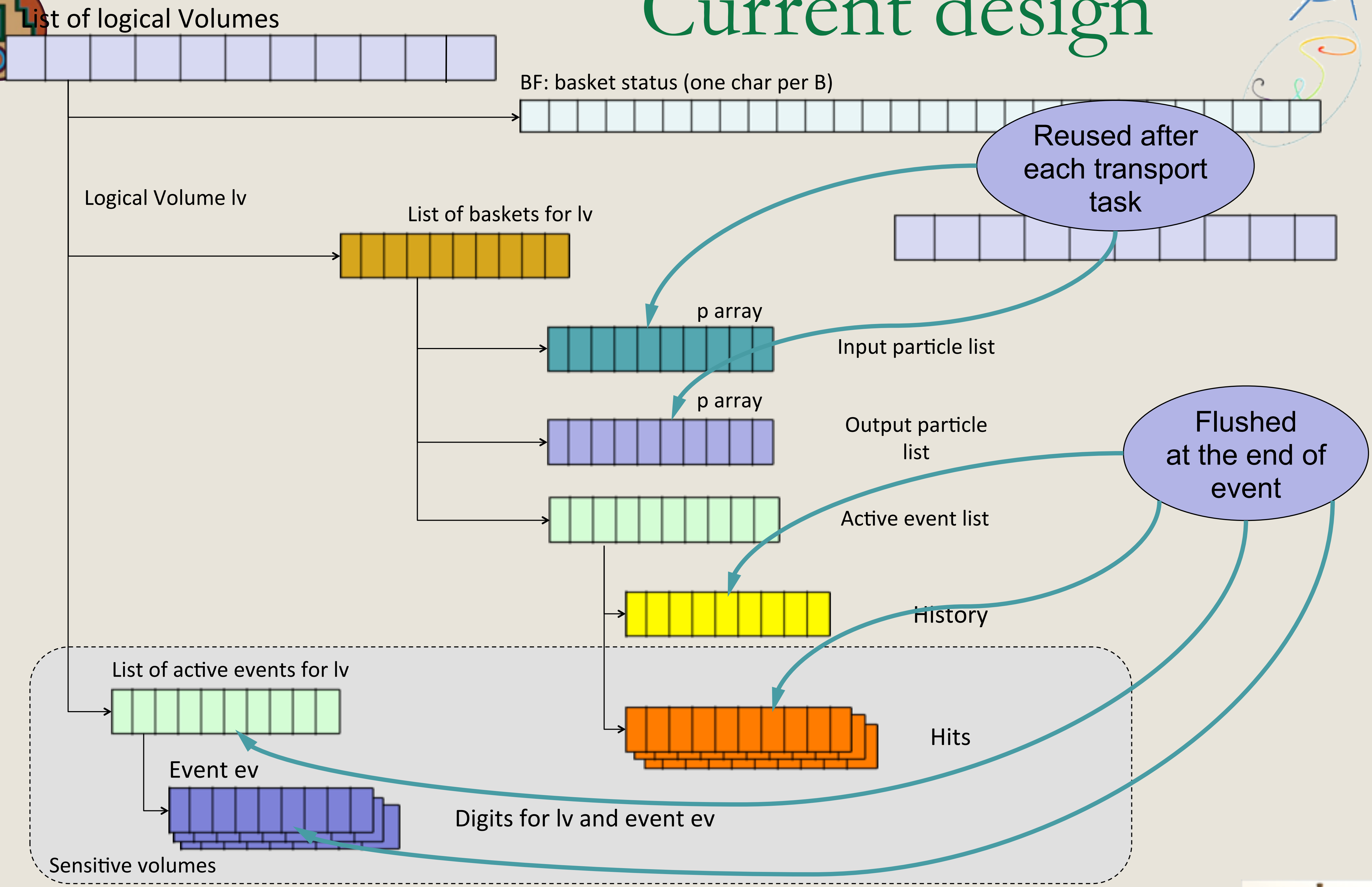
Current design



Current design



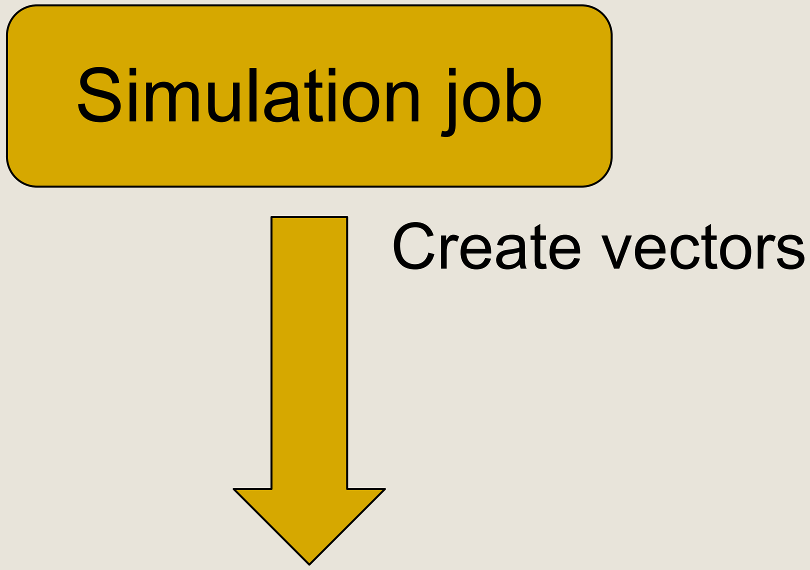
Current design



Questions

- How many dispatcher, digitizer and event-builder threads?
  - Difficult to say, we need some more quantitative design work, more realistic physics and actual prototyping
  - Measurements with G4 simulations could help
- Transport thread numbers will have to adapt to the size of simulation and of the detector
  - In ATLAS for instance 50% of the time is spent in 0.75% of the volumes
  - Threads could be distributed proportionally to the time spent in the different LVs

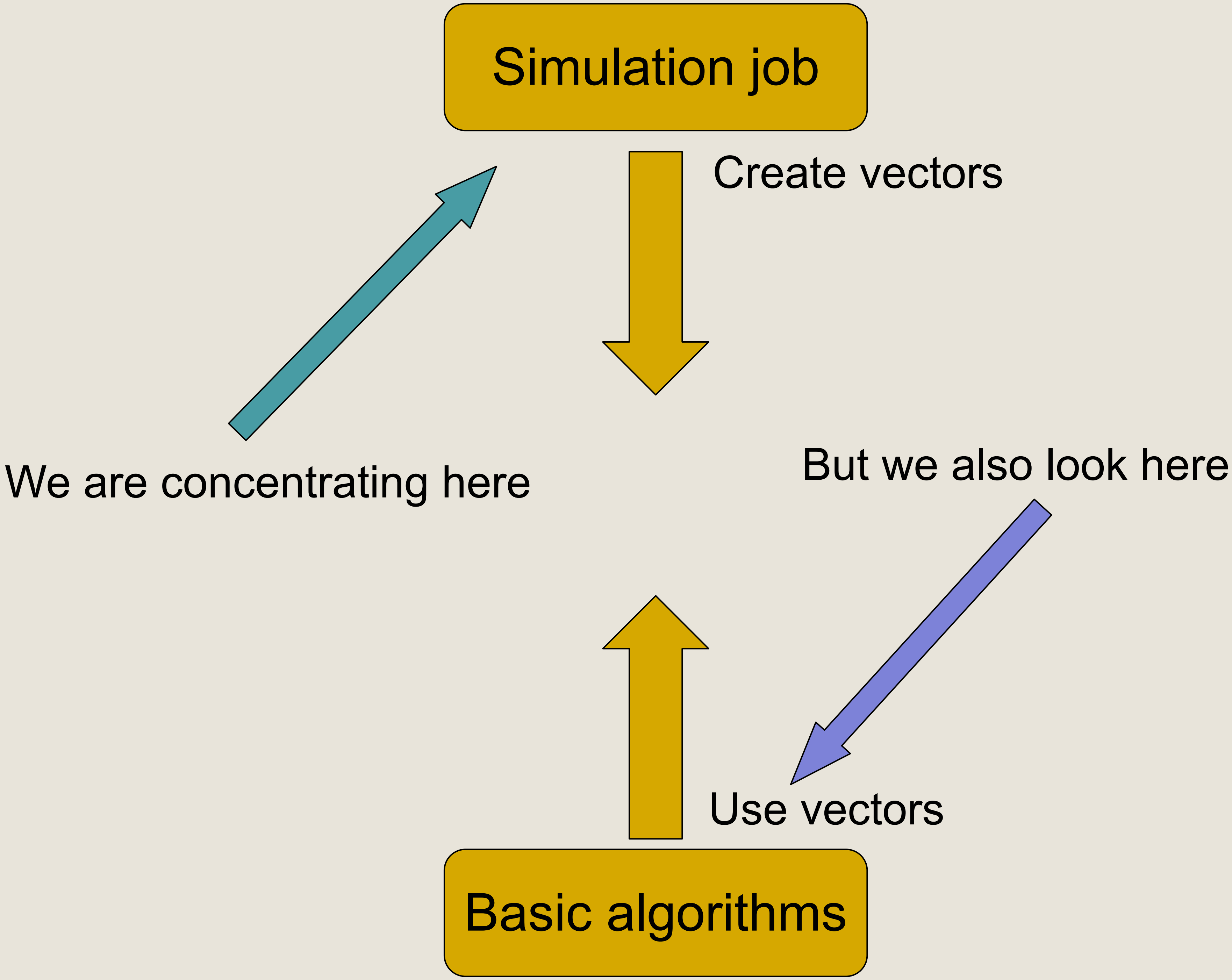
Grand strategy



Simulation job

Create vectors

Grand strategy



Simulation job

Create vectors

We are concentrating here

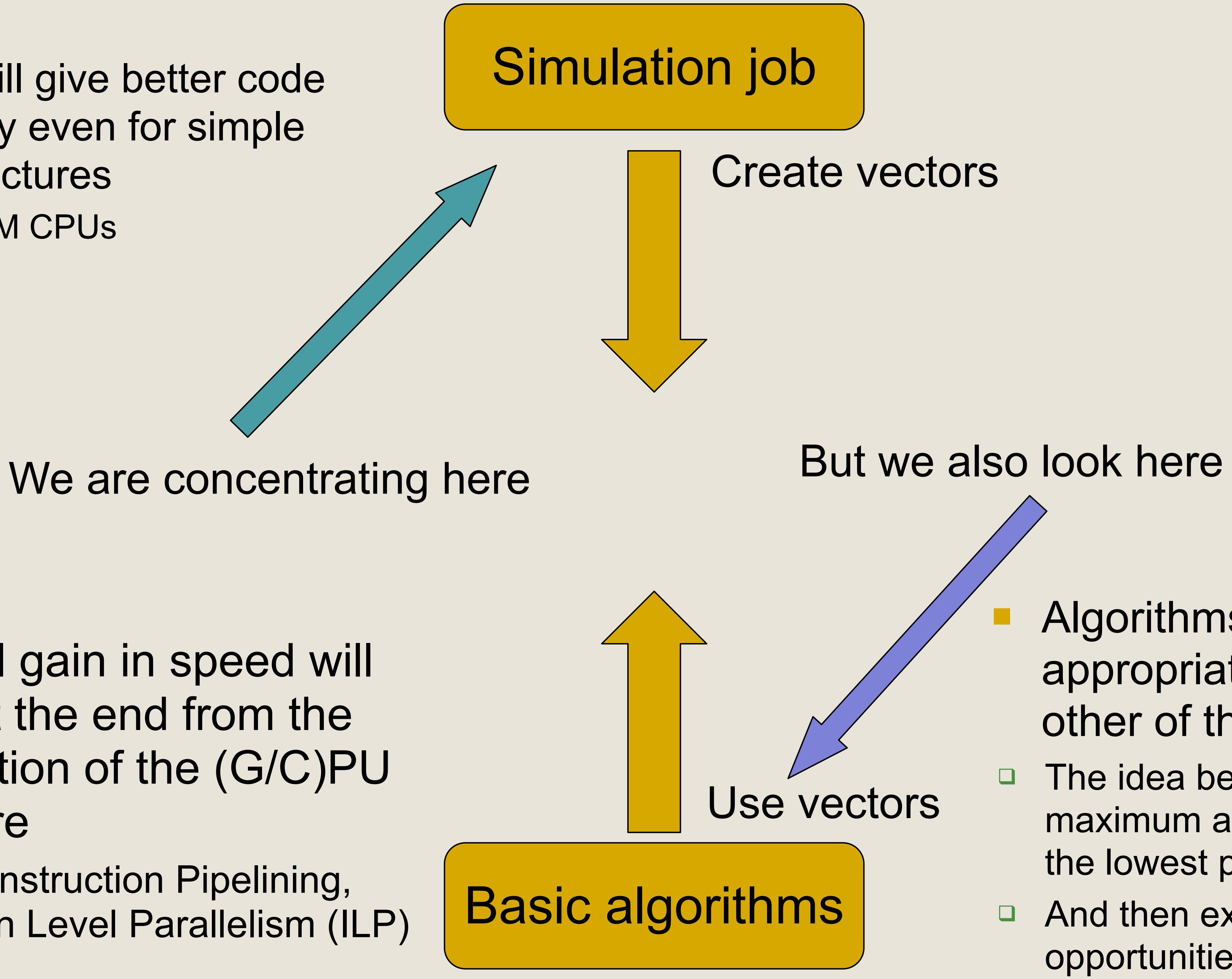
But we also look here

Use vectors

Basic algorithms

Grand strategy

- This will give better code anyway even for simple architectures
  - e.g. ARM CPUs



- The real gain in speed will come at the end from the exploitation of the (G/C)PU hardware
  - Vectors, Instruction Pipelining, Instruction Level Parallelism (ILP)

- Algorithms will be more appropriate for one or the other of these techniques
  - The idea being to expose the maximum amount of parallelism at the lowest possible granularity level
  - And then explore the optimisation opportunities

Physics

- We will select a number of relevant mechanisms
  - Bremsstrahlung, e+ annihilation, Compton, Decay, Delta ray, Elastic hadron, Inelastic hadron, Pair production, Photoelectric, Capture...
  - We have already energy loss
- For each one of those and for Z=1-100 we will tabulate G4 cross sections, say with E=100keV – 1TeV
- For each reaction and each energy bin we generate 50 final states with G4
- When a reaction is selected
  - Select the set of final states closer in energy
  - Randomly pick a final state
  - Scale its CMS energy to the CMS energy of the actual reaction
  - Random rotate it around φ and rotoboost according to the incoming particle

Advantages

- This model with tables should be quite appropriate for vectorization.
- Data locality is optimised (cross-sections/per material/ logical volume)
- This is also a very good model for a fast MC
  - A probably a good alternative to calling G4-like routines for cross-sections and interactions if we increase the number of pre-computed interactions per bin (say from 50 to 200)
  - Of course to be tested

And further in the future…

- More refined physics models
- Variance reducing & biased transport
- Low energy neutrons saga
- Radio-protection applications
- Optical photon propagation
- Collaboration with other fields (health physics, medicine…)
- Coupling with CAD, Finite Elements, structural modelling, heath transport…
- Beam transport dynamics
- Material damage, heating, mechanical impact

Opportunity

- From 1992~2004 there was an intense R&D about future HEP software
  - These were the C++ years
- From 2004 people just sat down to work and put in practice what they learnt
  - Learning much more… and building the current production systems
- I think a new cycle is beginning
  - LHC upgrade, ILC/CLIC studies, FAIR preparation
- Starting the prototype now is the right moment to make an impact
- More generally I believe we should promote a new round of brainstorming between IT, PH, OpenLab and experiments within and without CERN

Possible timeline

- By summer 2013
  - Implement a prototype according to the present design
  - Get essential data from G4
    - x-sections and shower library
  - Vectorize, GPU-ize, Phi-ize at least three geometry classes (simple, intermediate, hard)
  - Vectorize, GPU-ize, Phi-ize at least a couple of EM simplified methods (from G4?)
- Fall 2013
  - Interface the methods above to the prototype to realise a first prototype of vectorized transport
- Manpower still very scarce, need help from the community

Possible timeline

- By summer
  - Implement a gn
  - Get essentia
    - x-sections a
  - Vectorize, G lasses (simple, intermediate
  - Vectorize, G implified methods (fro
- Fall 2013
  - Interface the se a first prototype of
- Manpower s he community

Conclusions

- Improving throughput for simulation requires rethinking the transport
  - Better use of locality and improvement in the low level optimizations (caching, pipelining, vectorization)
- The blackboard exercise is moving into a fully functional prototype
  - Most aspects of the new model understood, still many ideas to test and benchmark
- The project gains momentum in ideas and contributors
- More and more people convinced on "the way to go" and discussing the implementation
- Community participation would be highly appreciated

Thank you!