# GeantV Prototype
# an update

Philippe Canal
for the GeantV project

F.Carminati (CERN), A.Gheata (CERN), S.Wenzel (CERN), M.Bandieramonte (Catania), J.de Fine Licht (CERN), R.Brun (CERN), A.Mohanty (BARC), A.Bhattacharyya (BARC), R.Seghal (BARC), Ph.Canal (FNAL), D.Elvira (FNAL), M.Novak (CERN), J.Apostolakis (CERN), G.Lima (FNAL), S.Jun (FNAL), O.Shadura (CERN),W.Pokorski (CERN), T.Nikitina (CERN), G.Cosmo (CERN), A.Ribon (CERN), G.Folger (CERN), G.Bitzes (openlab), G.Biro (Budapest U), L.Durhem (intel), H.Kim (Korea U)

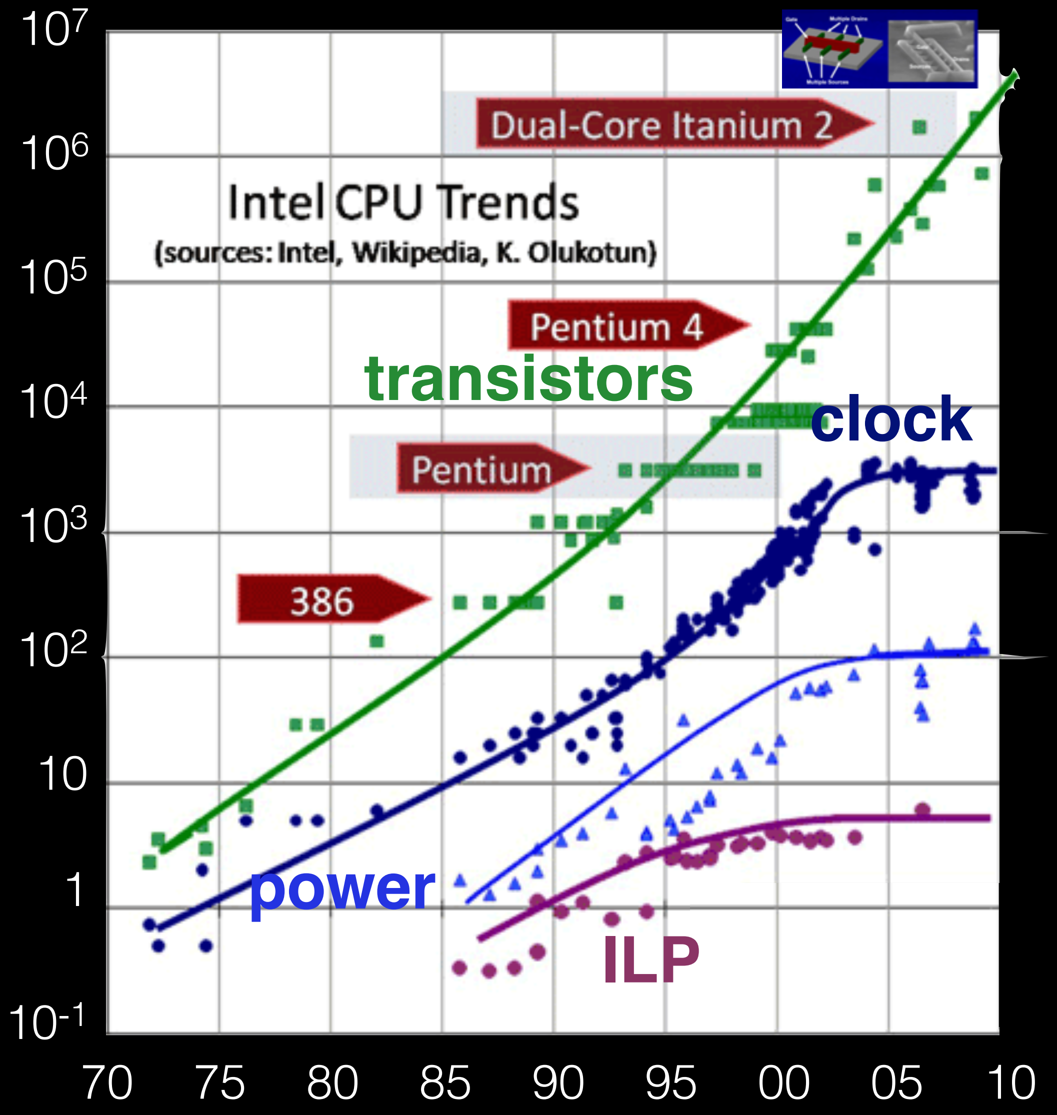
# Disclaimer

- Most of the material in these slides is from my colleagues
- I have added a few tweaks and all the mistakes

# GeantV: motivations

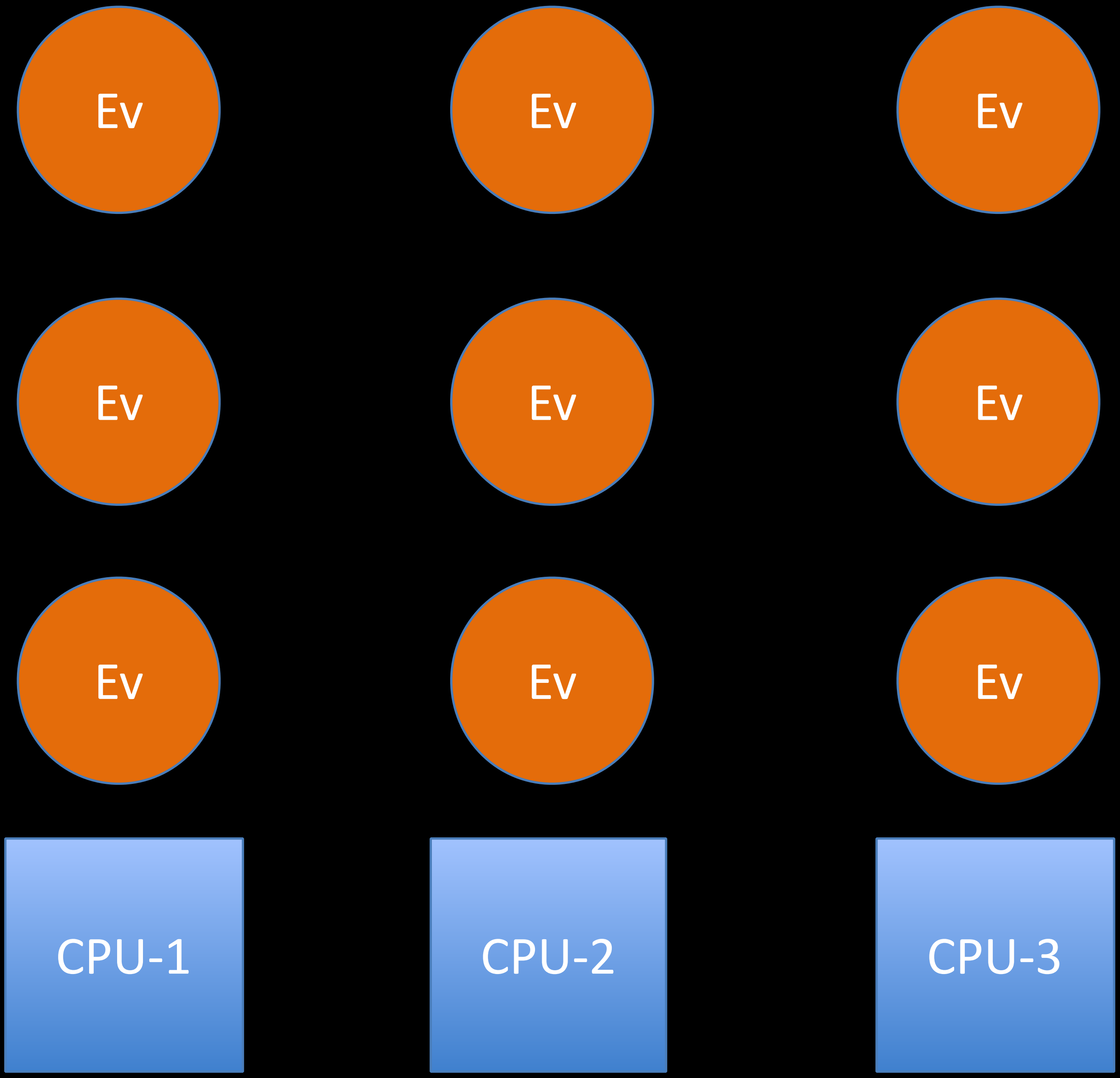
## (even if you are *familiar* with them)

- Performance of our code scales with clock cycle (hence is stagnant!)
- Needs will increase more than tenfold and the budget will be constant at best
- HEP code needs to exploit new architectures and to team with other disciplines to share the optimization effort
  - Data & instruction locality and vectorisation
- Portability, better physics and optimization will be the targets
- Simulation can lead the way to show how to exploit today's CPU's resources more effectively in complex applications



- Seeking ways to write code portable between CPU with vector units or not and accelerators (GPU, Xeon Phi)

# With some simplification…



Ev Ev Ev

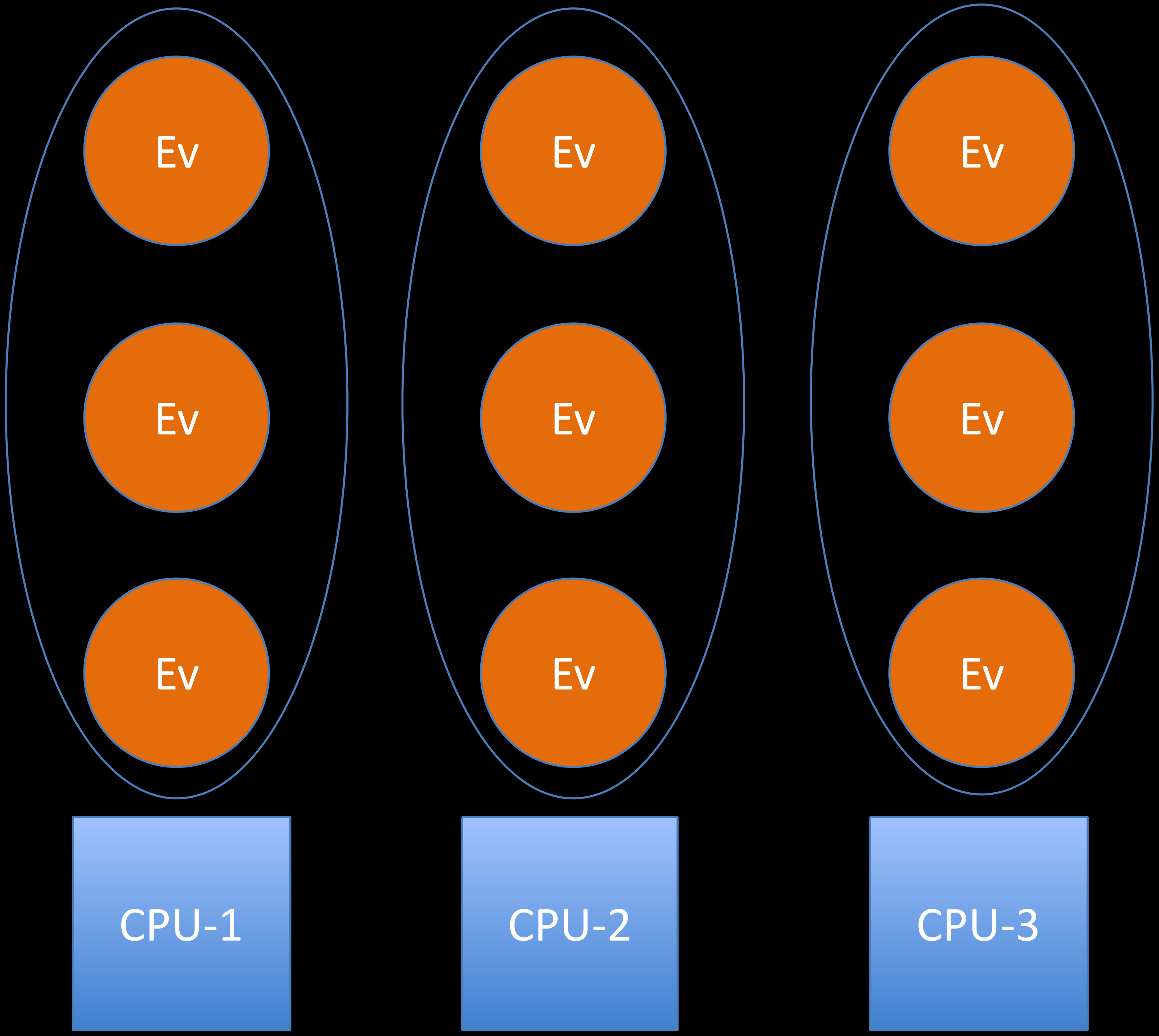
Ev Ev Ev

Ev Ev Ev

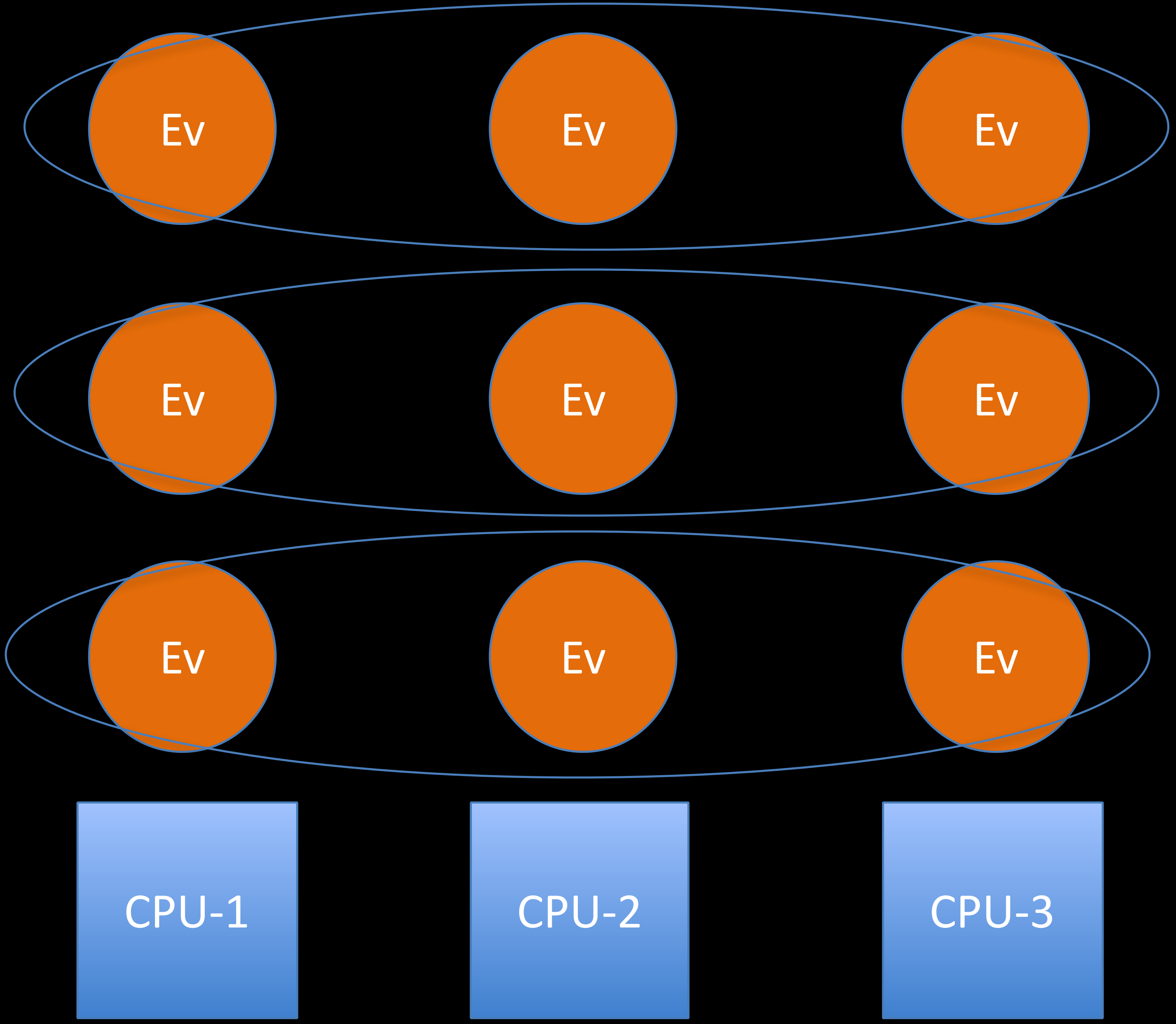
CPU-1 CPU-2 CPU-3

# With some simplification…

Good Old Way
ETA: 3*Time/EV

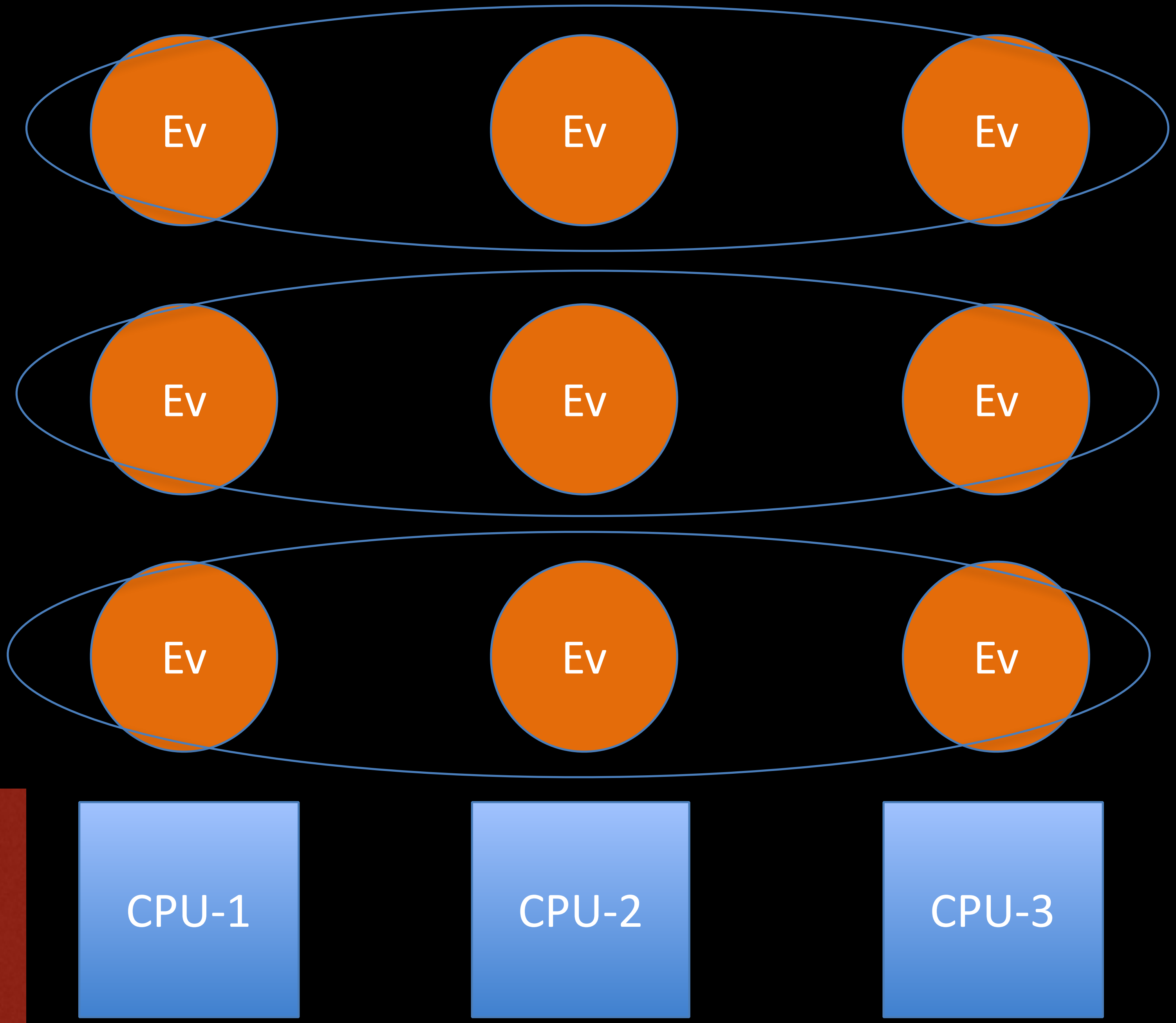


# With some simplification…



Brand New World
ETA: 3*Time/EV

# With some simplification…



Brand New World
ETA: 3*Time/EV

Of course one should not forget cache effects, memory footprint and so on… still these are second order effects

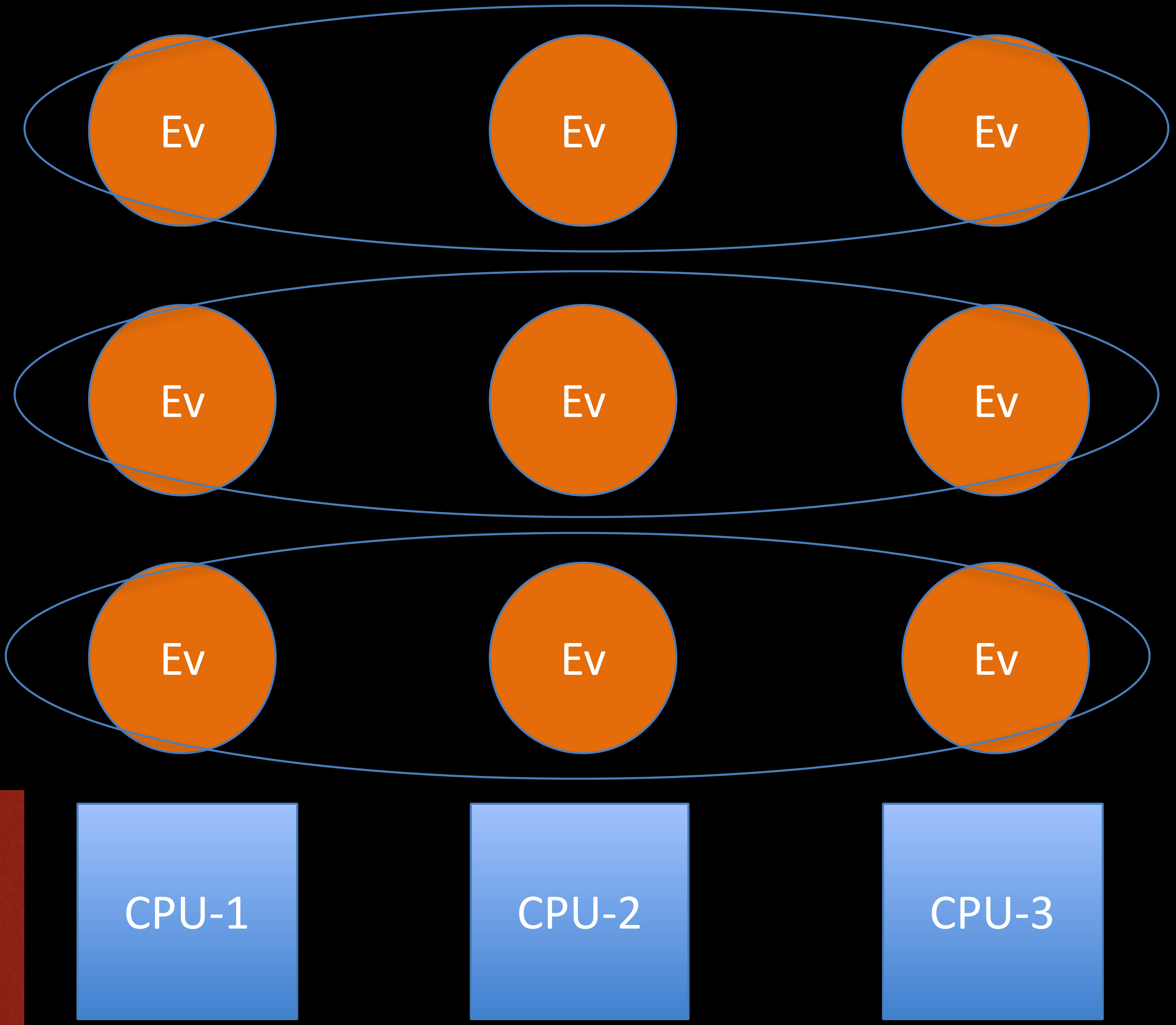
# With some simplification…

G4MT: CPU time for 100EV on 16 cores is the same as sequential.

RAM (reduced footprint) is reduced.

To reduce CPU time the flow of work (instructions, data) must change!

Brand New World
ETA: 3*Time/EV



Of course one should not forget cache effects, memory footprint and so on… still these are second order effects

# With some simplification…

G4MT: CPU time for 100EV on 16 cores is the same as sequential.

RAM (reduced footprint) is reduced.

To reduce CPU time the flow of work (instructions, data) must change!

Brand New World
ETA: 3*Time/EV

Challenges

- Use CPU's resources to the max: instruction, L1 & L2 caches, vector instructions, ILP
- Reuse instructions and data => deal with multiple tracks / events at a time
  - Current HEP code scores 0.8 IPC!
  - *Fat CPUs* can deliver > 4 results per clock (float & integer, .. )
- Create portable code !

Of course one should not forget cache effects, memory footprint and so on… still these are second order effects

CPU-1

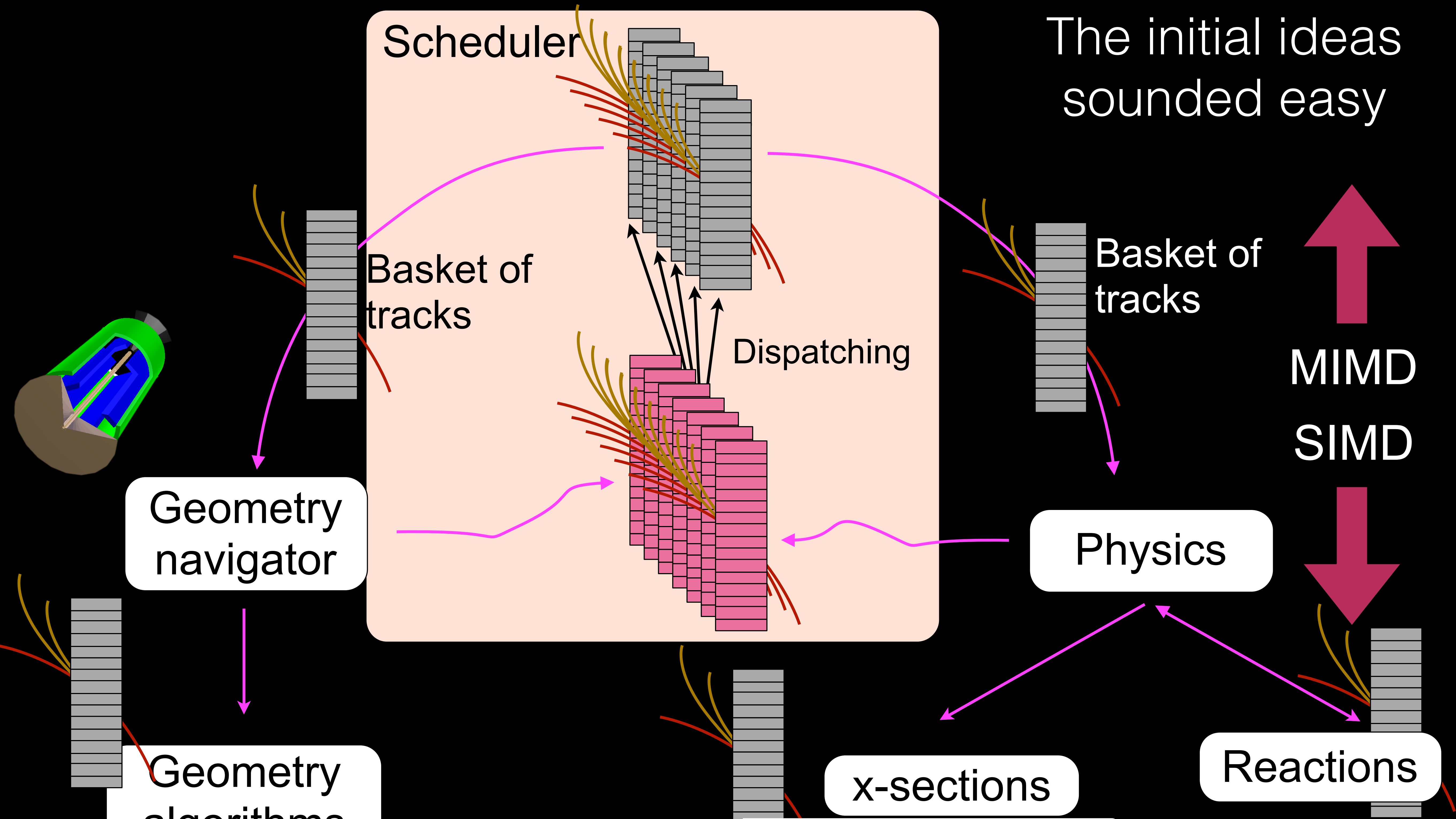
CPU-2

CPU-3

# What do we want to do?

- Develop an all-particle transport simulation software with
  - Improved state-of-the-art physics
  - A performance between 2 and 5 times greater than Geant4
  - Full simulation and various options for fast simulation
  - Portable on different architectures, including accelerators (GPUs and Xeon Phi’s)
- Understand the limiting factors for a one-order-of-magnitude (10x) improvement

# The initial ideas sounded easy



Scheduler

Basket of tracks

Dispatching

Basket of tracks

MIMD

SIMD

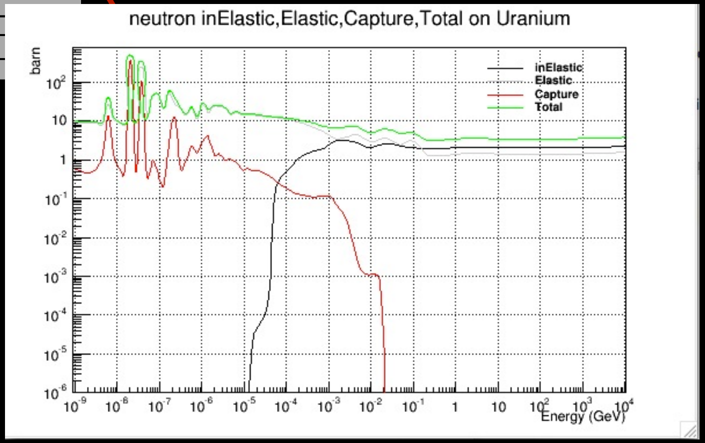
Geometry navigator

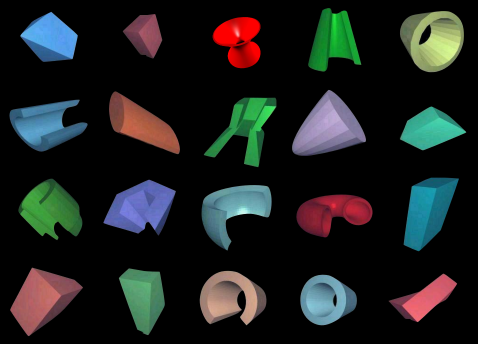
Physics

Geometry algorithms

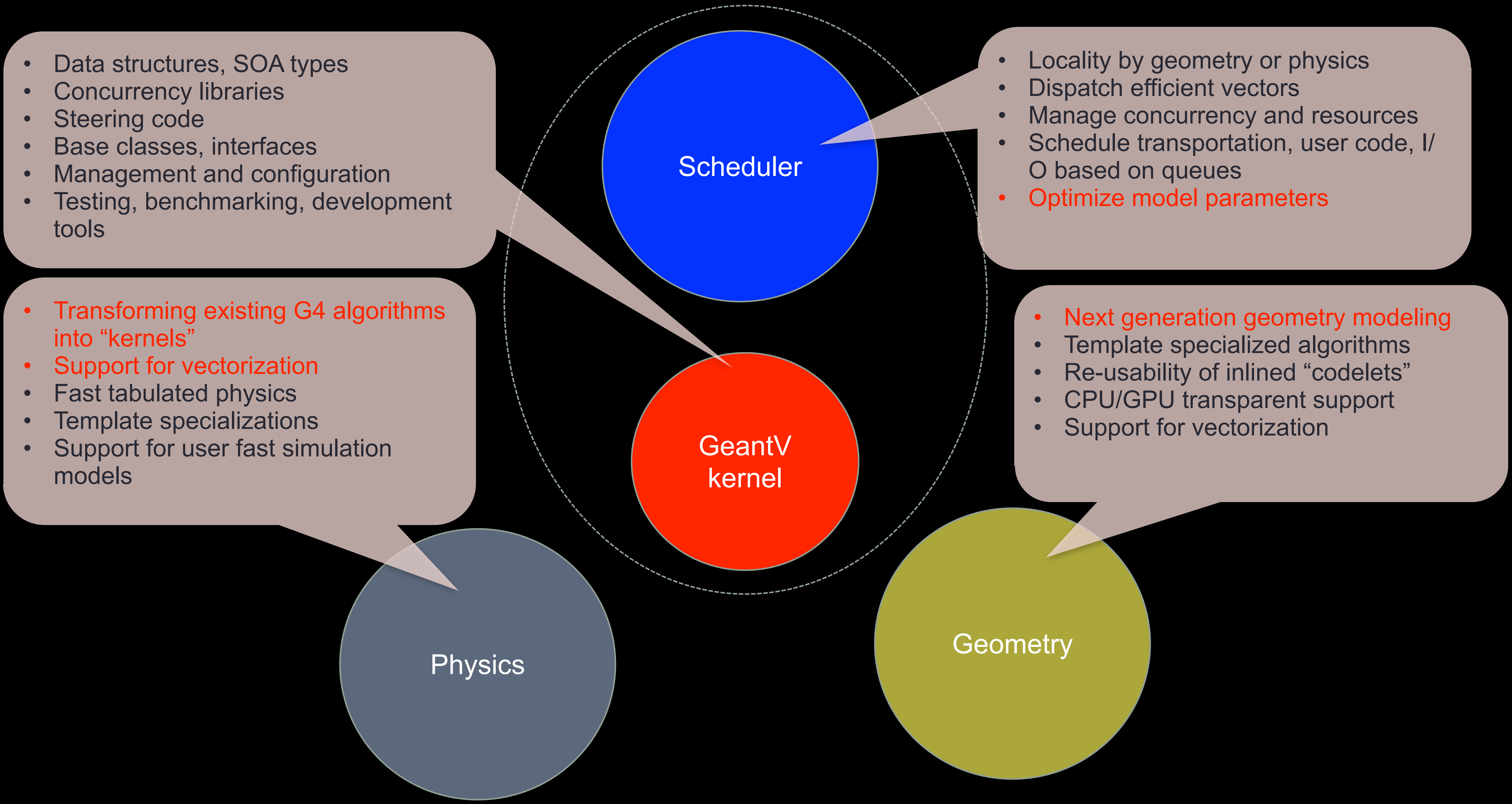
x-sections

Reactions





# R&D directions



- Data structures, SOA types
- Concurrency libraries
- Steering code
- Base classes, interfaces
- Management and configuration
- Testing, benchmarking, development tools

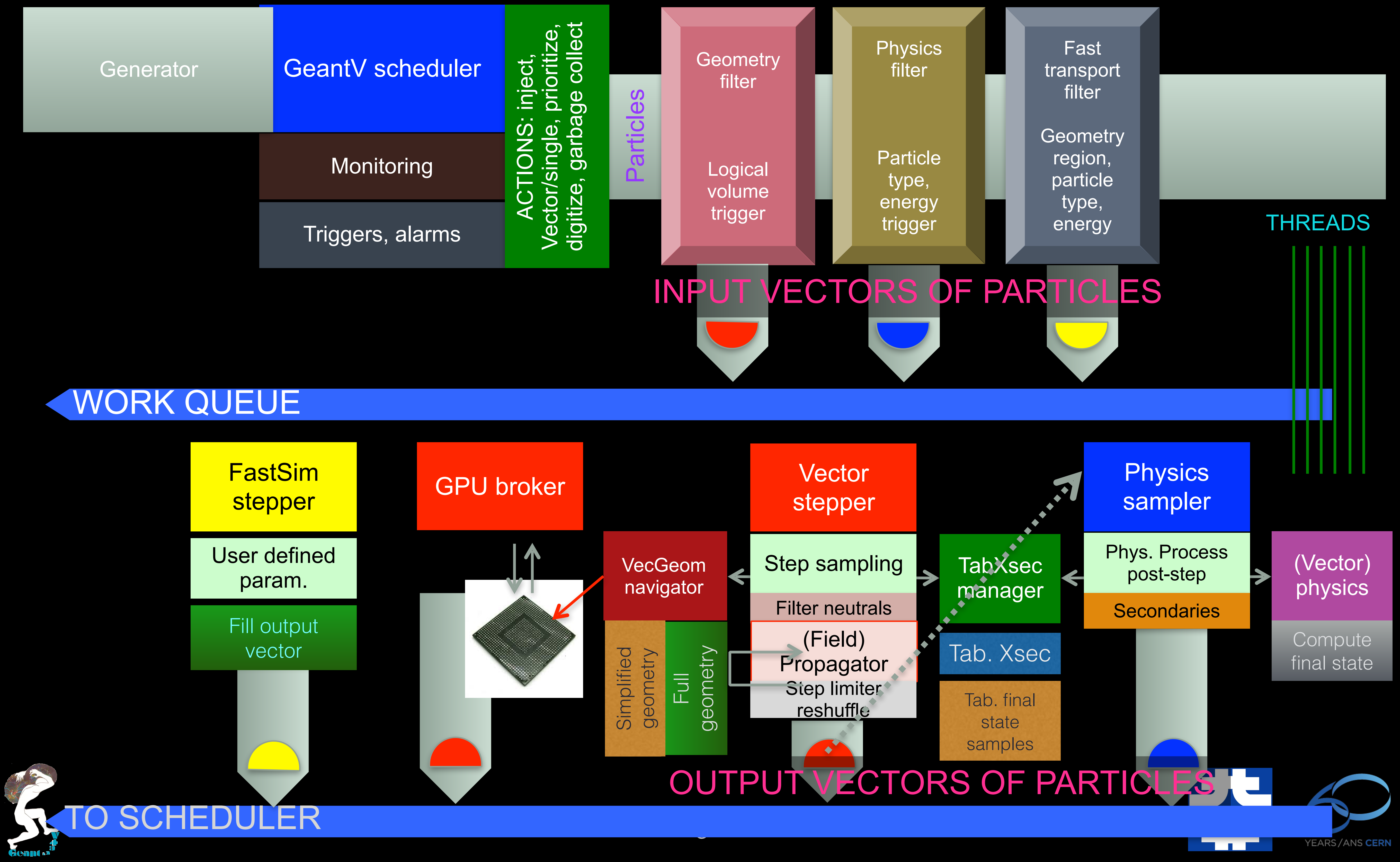
- Locality by geometry or physics
- Dispatch efficient vectors
- Manage concurrency and resources
- Schedule transportation, user code, I/O based on queues
- Optimize model parameters

- Transforming existing G4 algorithms into “kernels”
- Support for vectorization
- Fast tabulated physics
- Template specializations
- Support for user fast simulation models

- Next generation geometry modeling
- Template specialized algorithms
- Re-usability of inlined “codelets”
- CPU/GPU transparent support
- Support for vectorization

geant.web.cern.ch

# The Scheduler



Generator

GeantV scheduler

Monitoring

Triggers, alarms

ACTIONS: inject, Vector/single, prioritize, digitize, garbage collect

Particles

Geometry filter

Logical volume trigger

Physics filter

Particle type, energy trigger

Fast transport filter

Geometry region, particle type, energy

THREADS

INPUT VECTORS OF PARTICLES

WORK QUEUE

FastSim stepper

User defined param.

Fill output vector

GPU broker

VecGeom navigator

Simplified geometry

Full geometry

Vector stepper

Step sampling

Filter neutrals

(Field) Propagator

Step limiter reshuffle

TabXsec manager

Tab. Xsec

Tab. final state samples

Physics sampler

Phys. Process post-step

Secondaries

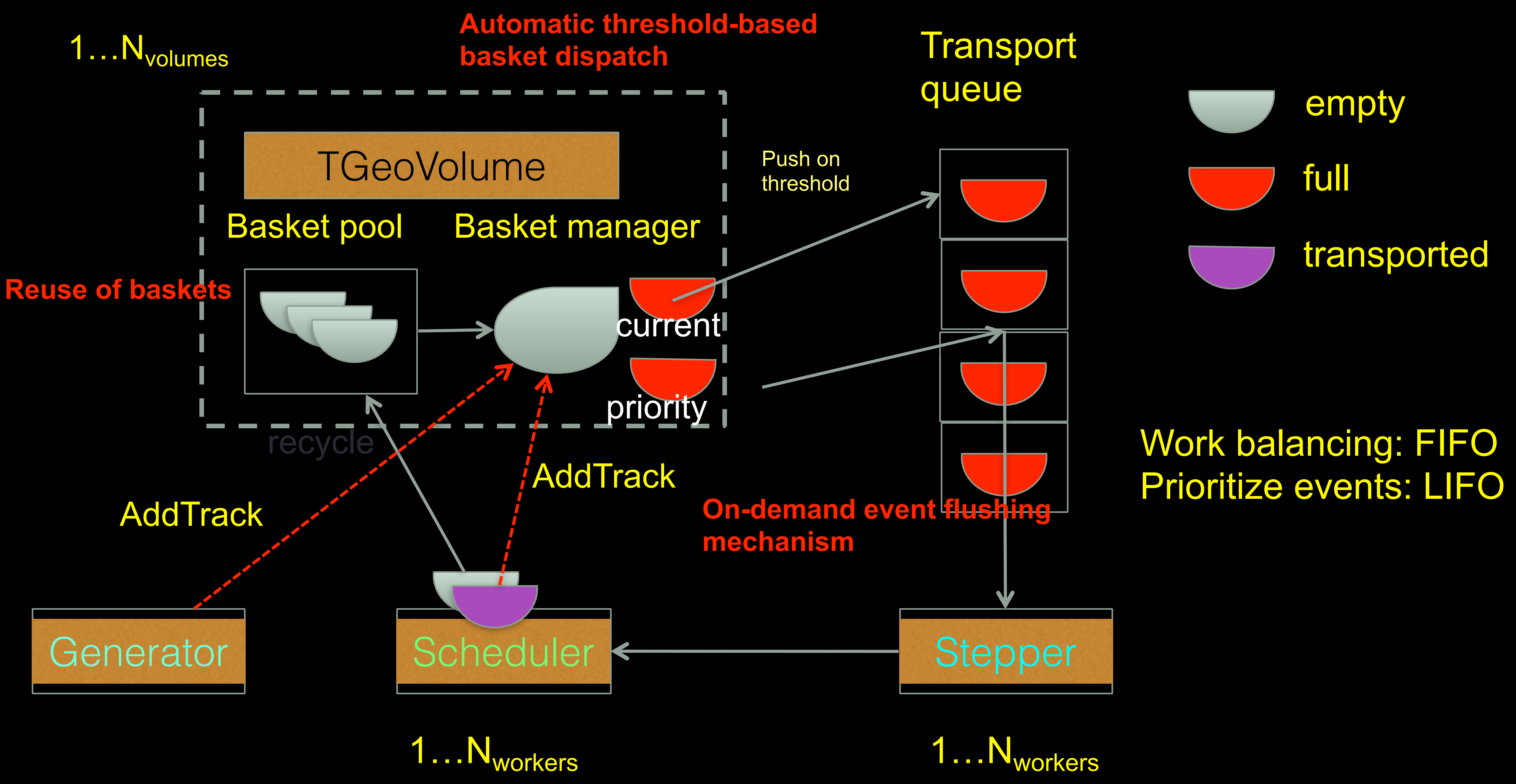
(Vector) physics

Compute final state

OUTPUT VECTORS OF PARTICLES

TO SCHEDULER

# Scheduling features



Automatic threshold-based basket dispatch

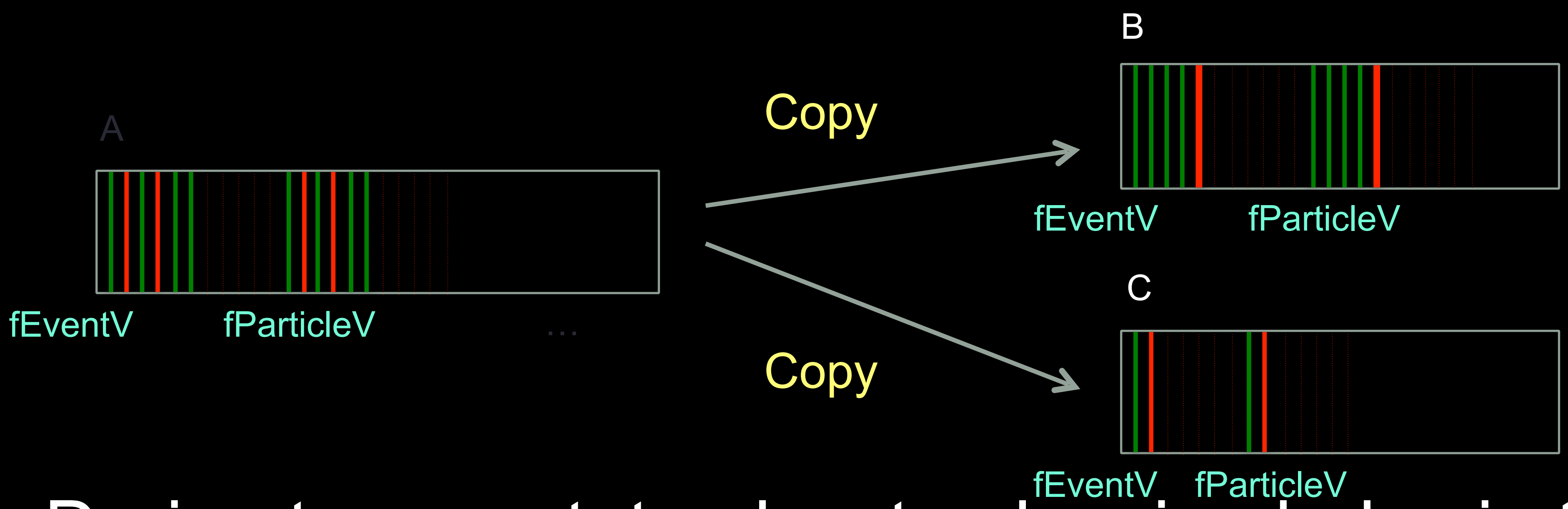
Reuse of baskets

On-demand event flushing mechanism

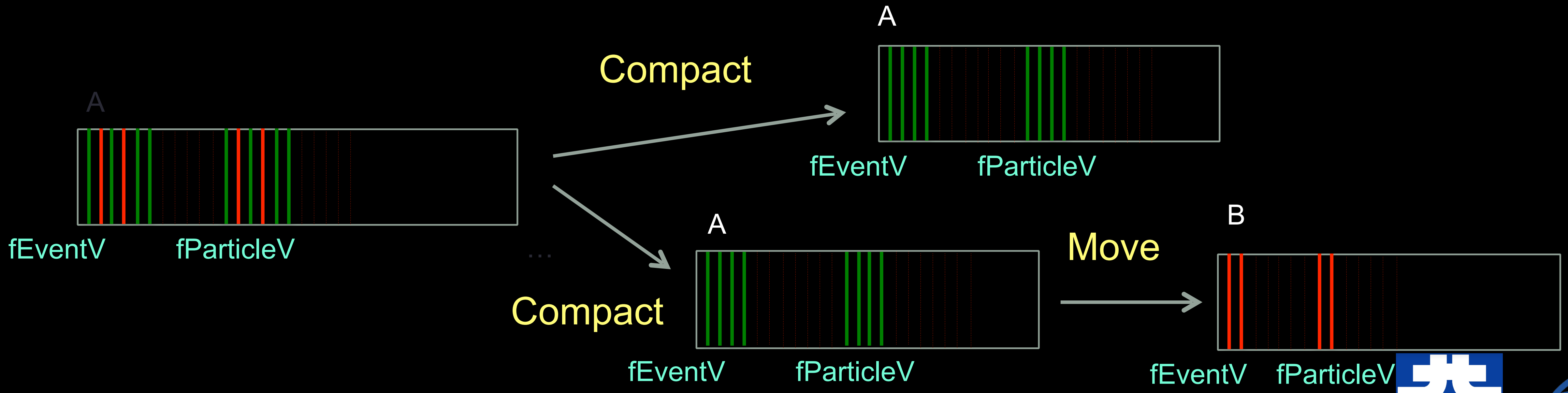
Work balancing: FIFO
Prioritize events: LIFO

# Challenges for vectorisation

- Tracks have to be copied to a receiver during rescheduling

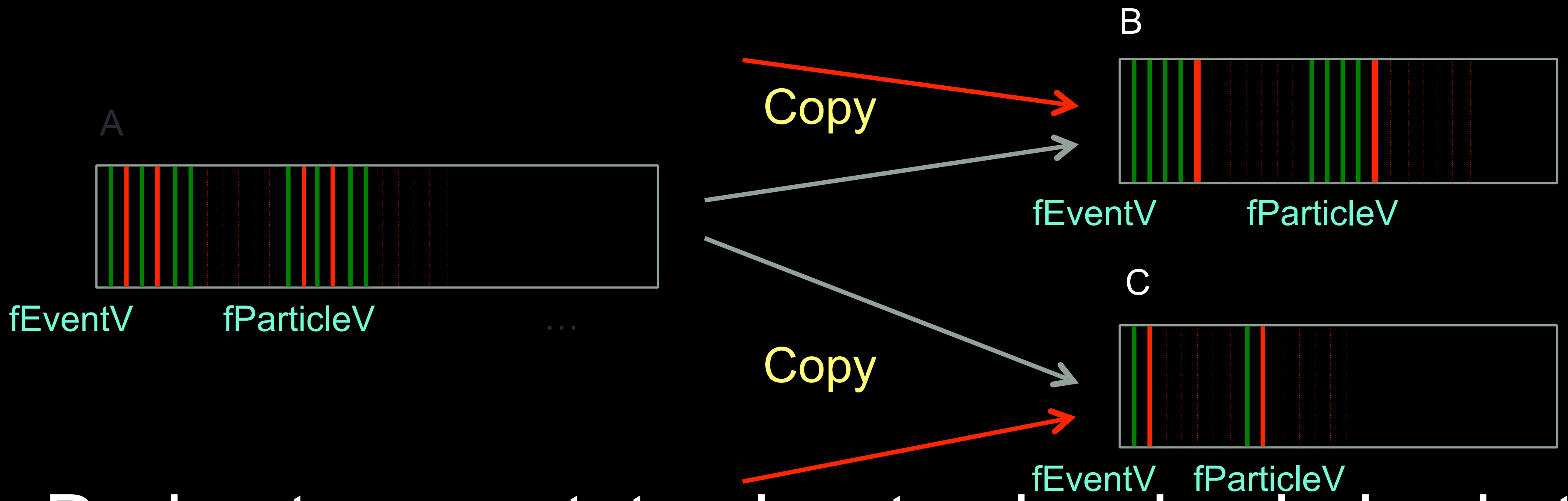


- During transport, tracks stop leaving holes in the container



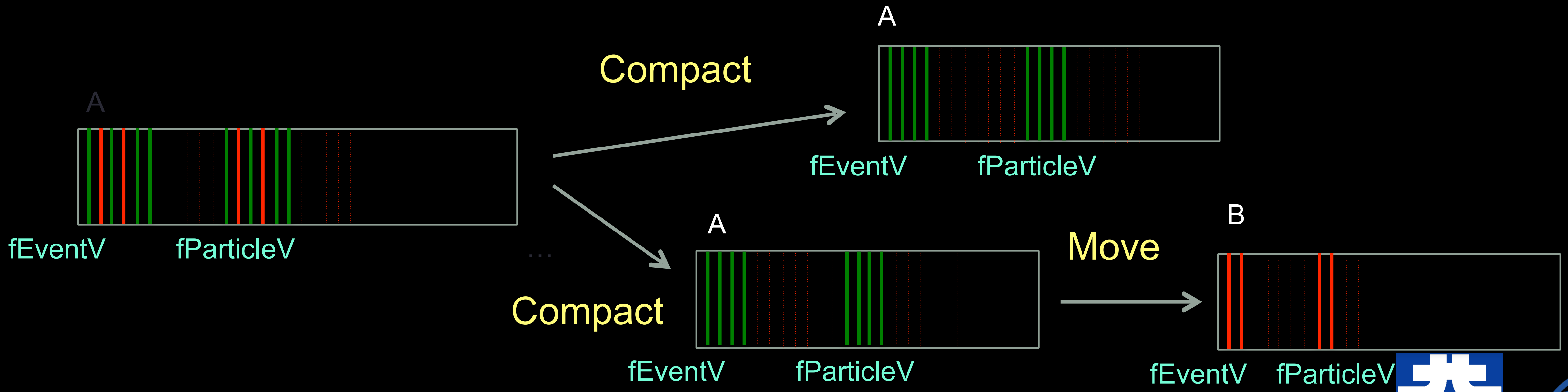
# Challenges for vectorisation

- Tracks have to be copied to a receiver during rescheduling



multithreaded

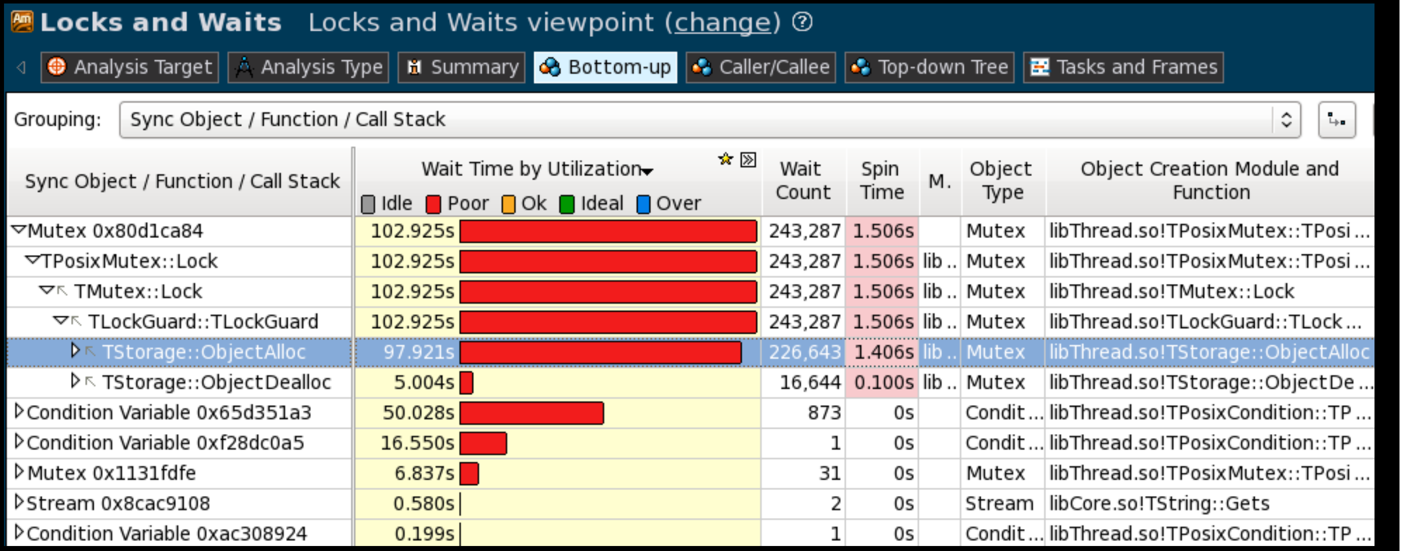
- During transport, tracks stop leaving holes in the container



# Scalability for MT is challenging

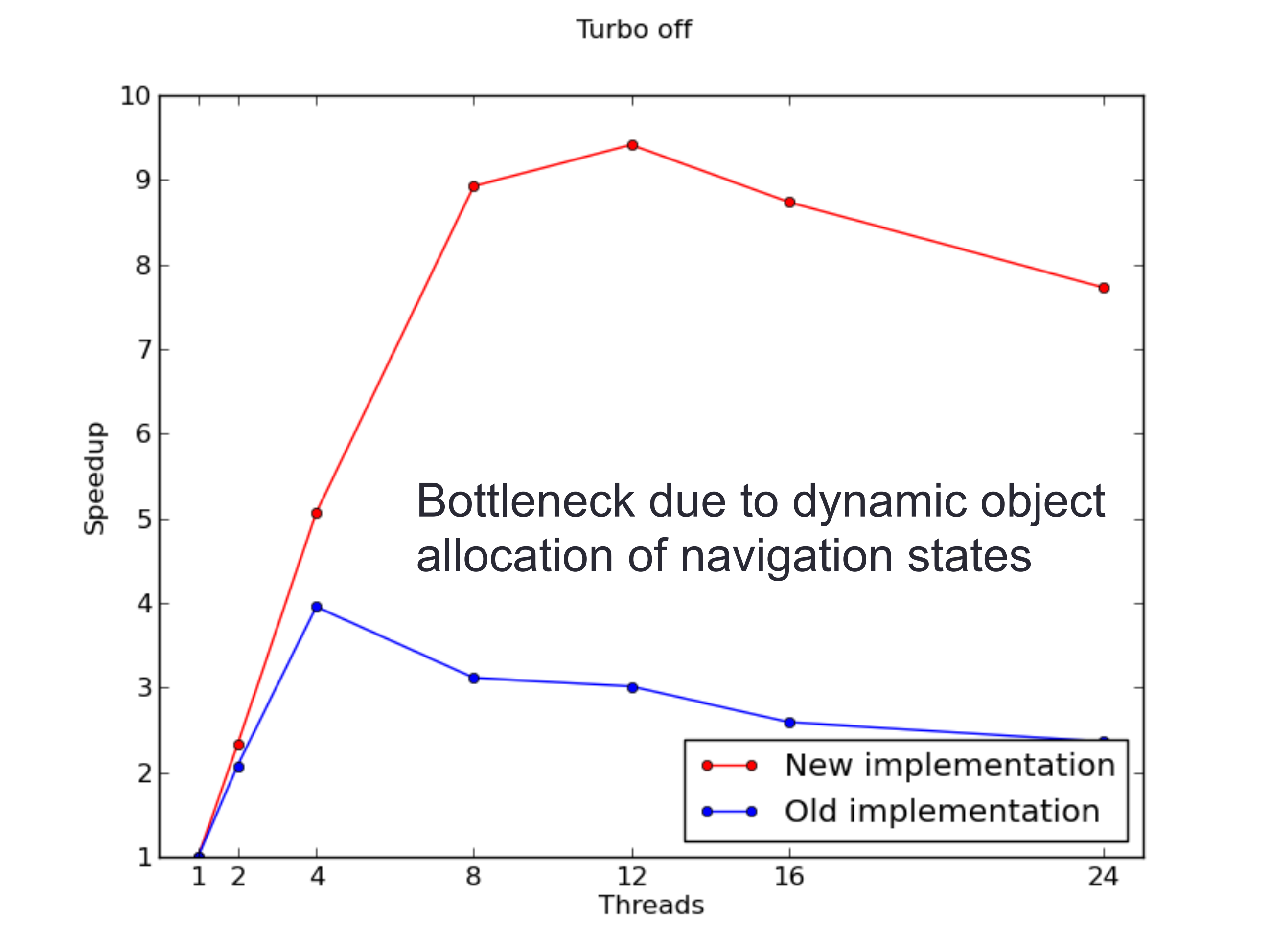
- Performance is constantly monitored
  - Jenkins module run daily
- Allows detecting and fixing bottlenecks
- Amdahl still high due to criticality of basket-to-queue dispatching operations

*1000 events with 100 tracks each, measured on a 24-core dual socket E5-2695 v2 @ 2.40GHz (IVB).*



Grouping: Sync Object / Function / Call Stack

| Sync Object / Function / Call Stack | Wait Time by Utilization<br>Idle Poor Ok Ideal Over | Wait Count | Spin Time | M. | Object Type | Object Creation Module and Function |
|---|---|---|---|---|---|---|
| ▽Mutex 0x80d1ca84 | 102.925s | 243,287 | 1.506s | | Mutex | libThread.so!TPosixMutex::TPosi ... |
| ▽TPosixMutex::Lock | 102.925s | 243,287 | 1.506s | lib .. | Mutex | libThread.so!TPosixMutex::TPosi ... |
| ▽↖ TMutex::Lock | 102.925s | 243,287 | 1.506s | lib .. | Mutex | libThread.so!TMutex::Lock |
| ▽↖ TLockGuard::TLockGuard | 102.925s | 243,287 | 1.506s | lib .. | Mutex | libThread.so!TLockGuard::TLock ... |
| ▷↖ TStorage::ObjectAlloc | 97.921s | 226,643 | 1.406s | lib .. | Mutex | libThread.so!TStorage::ObjectAlloc |
| ▷↖ TStorage::ObjectDealloc | 5.004s | 16,644 | 0.100s | lib .. | Mutex | libThread.so!TStorage::ObjectDe ... |
| ▷Condition Variable 0x65d351a3 | 50.028s | 873 | 0s | | Condit ... | libThread.so!TPosixCondition::TP ... |
| ▷Condition Variable 0xf28dc0a5 | 16.550s | 1 | 0s | | Condit ... | libThread.so!TPosixCondition::TP ... |
| ▷Mutex 0x1131fdfe | 6.837s | 31 | 0s | | Mutex | libThread.so!TPosixMutex::TPosi ... |
| ▷Stream 0x8cac9108 | 0.580s | 2 | 0s | | Stream | libCore.so!TString::Gets |
| ▷Condition Variable 0xac308924 | 0.199s | 1 | 0s | | Condit ... | libThread.so!TPosixCondition::TP ... |

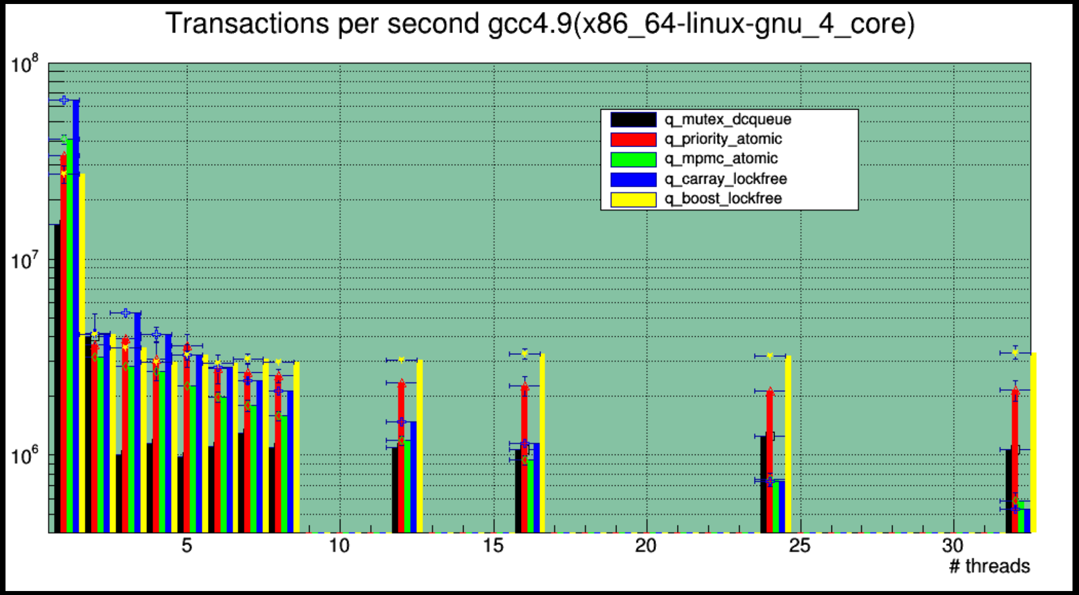


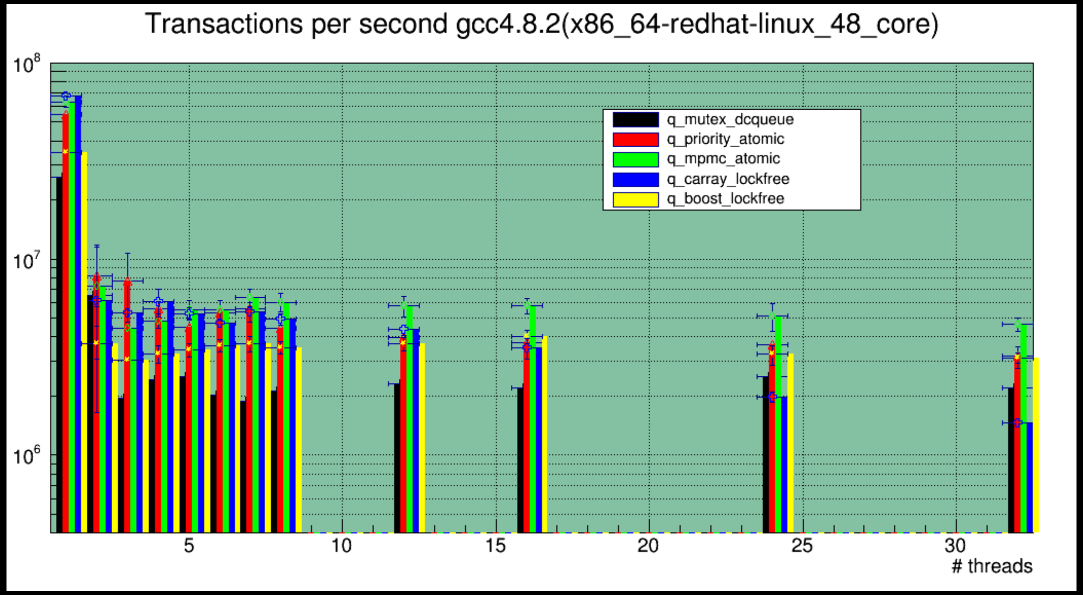
Bottleneck due to dynamic object allocation of navigation states

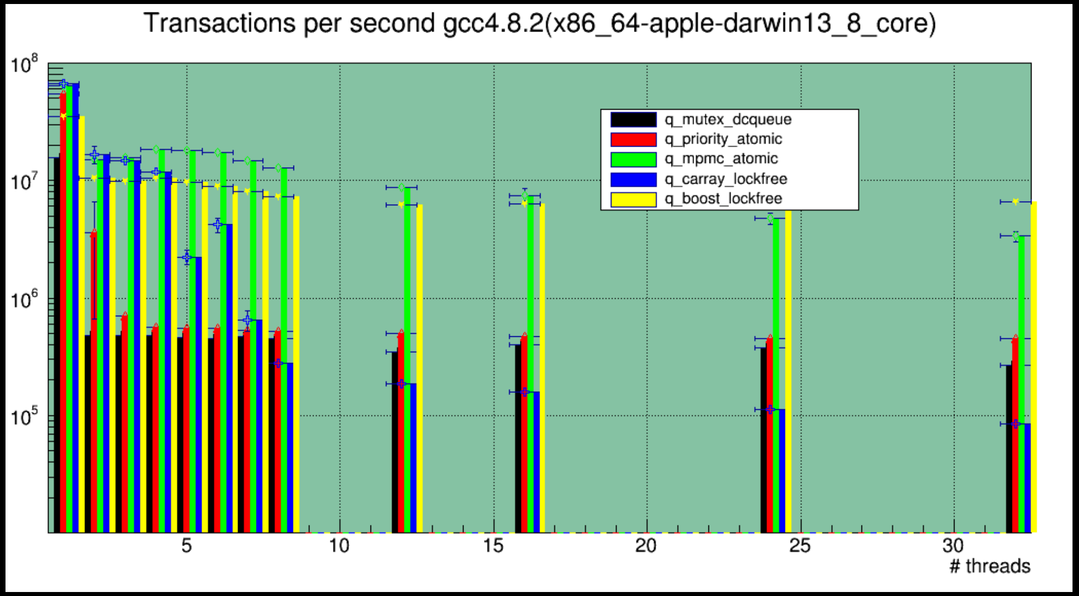
# Queues in GeantV

- Mutex based dcqueue
  - In production as work queue, provides priority
- Mutex/atomic hybrid priority_atomic
  - Mutexed only in high concurrency regime, provides priority
- Atomic CAS (compare and swap) mpmc_atomic
  - In production for basketiser queues, replacing dcqueue
  - Circular buffer, no priority
- Array lockfree carray_lockfree (ported by Omar)
  - Another implementation of circular buffer queue
- Boost lock free queue boost_lockfree (ported by Omar)
  - Boost implementation of lock free queue

# Performance


Transactions per second gcc4.9(x86_64-linux-gnu_4_core)

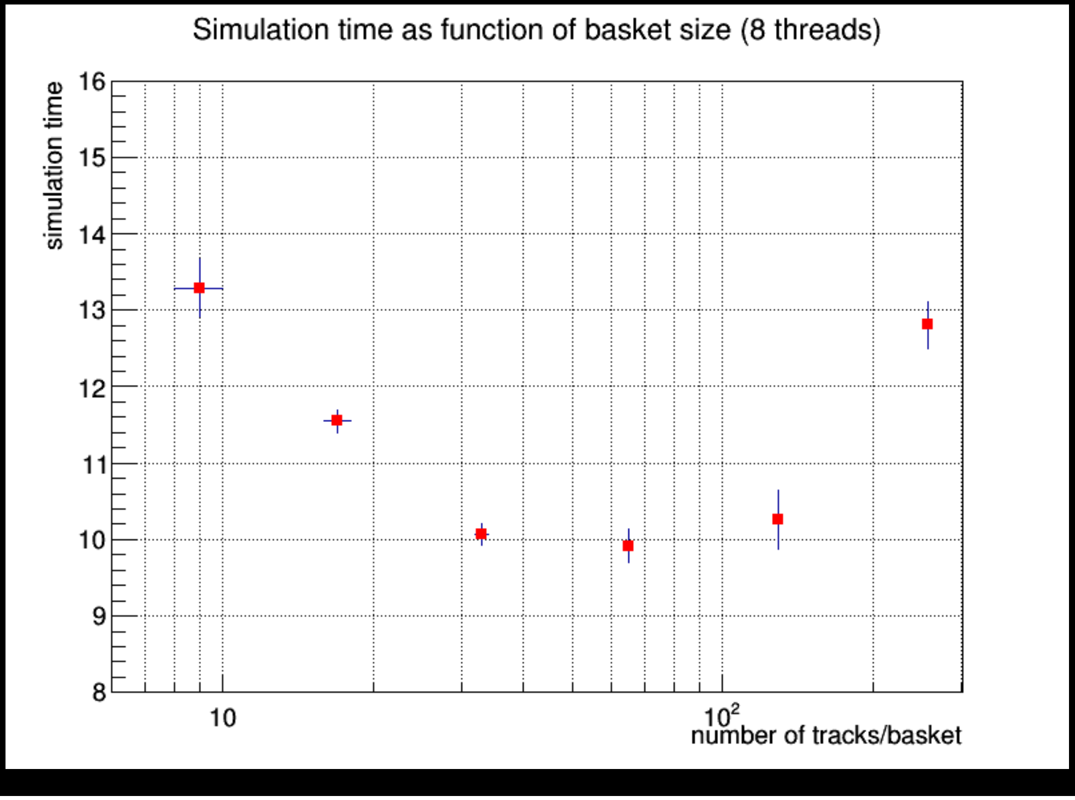

Transactions per second gcc4.8.2(x86_64-redhat-linux_48_core)

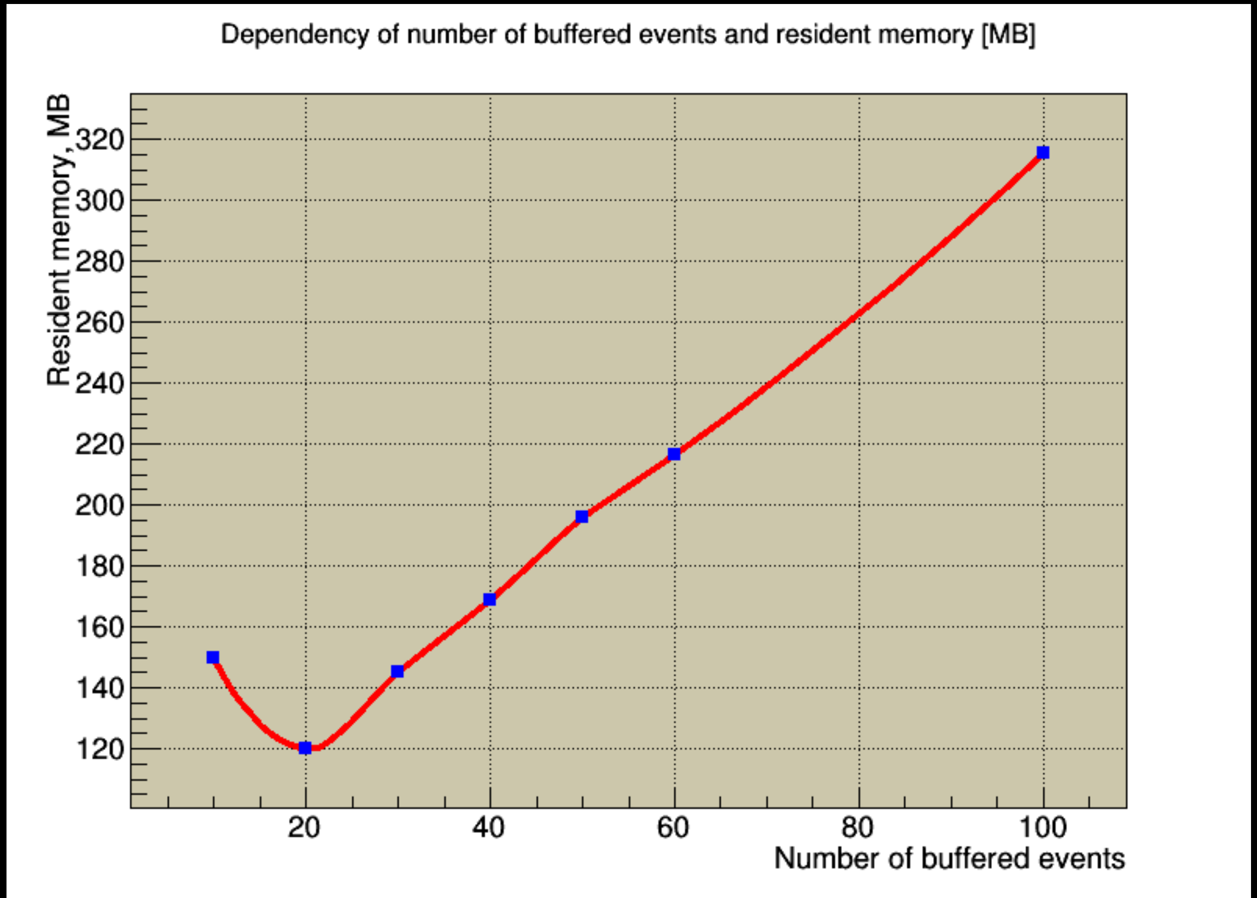

Transactions per second gcc4.8.2(x86_64-apple-darwin13_8_core)

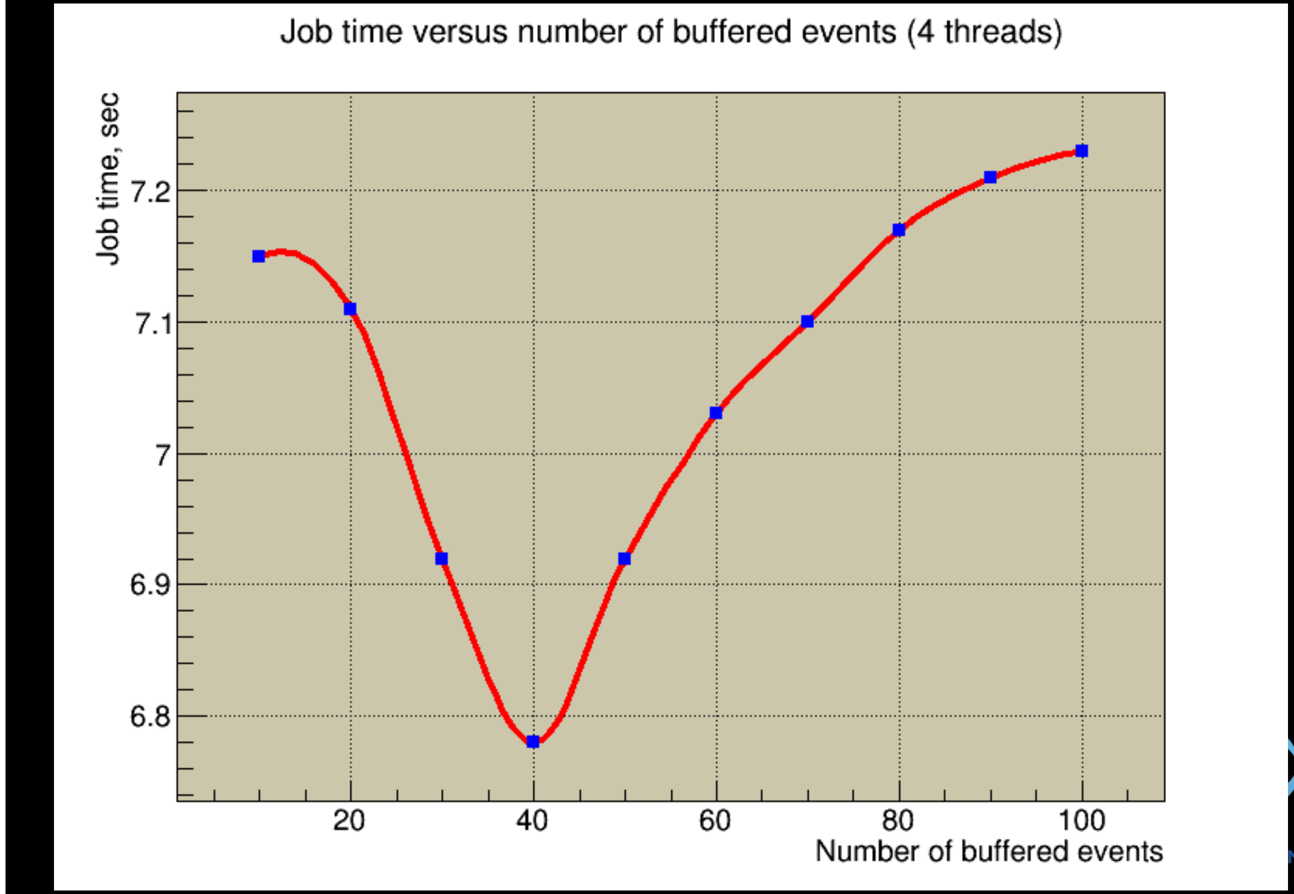
- Our current dcqueue is outperformed by all the others on all platforms
- We currently work at ~$10^5$ transactions/sec
- Lockfree queues are doing great on Mac compared to mutex-based ones (50x factor!)
- priority_atomic is the only current replacement for dcqueue (must provide priority)
- We can expect a factor of 2 queueing improvement on x86_64 linux
- Reducing Amdahl requires revisiting the basketizing model

# A lot of parameters!

- Keep $N_{buff}$ in memory (from $N_{total}$ to be simulated)
  - As an event gets flushed, inject a new one
- The vector size is a major parameter of the model
  - Small vectors = inefficient vectorization, dispatching becomes an overhead
  - Large vectors = larger overheads for scatter/gather, more garbage collections
- Automatic adjustment of vector size per volume





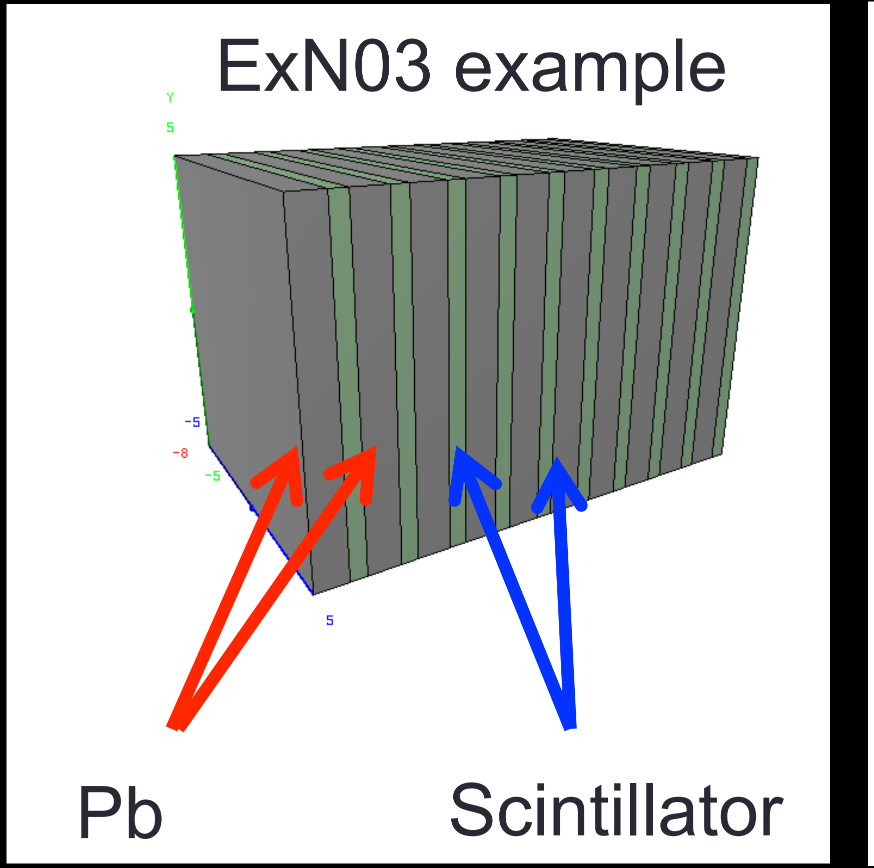


# GeantV & optimisation

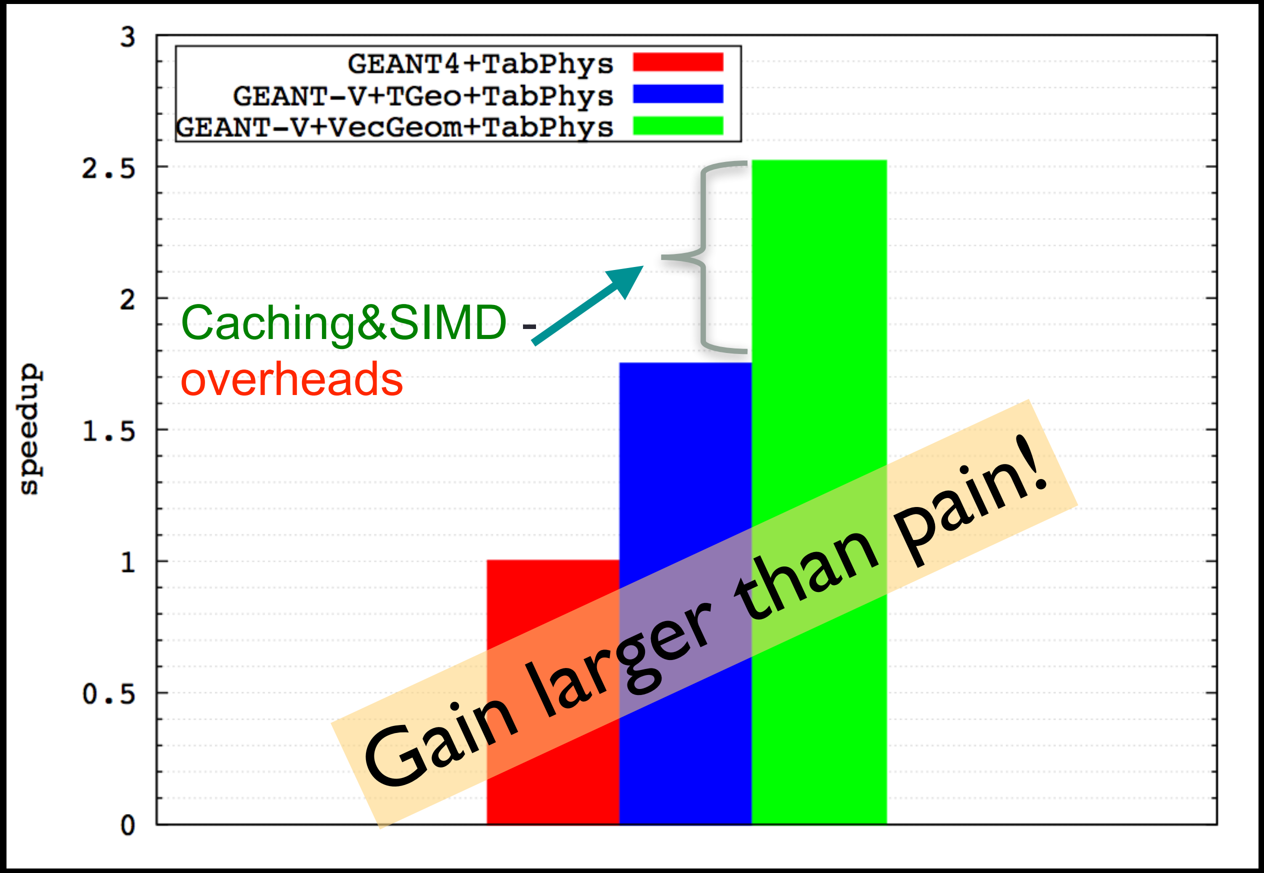
- Optimize GeantV scheduler model
    - Use genetic algorithms to find the optimum in the parameter space for different setups
- Model chromosomes:
    - Thresholds for prioritizing events, basket size, number of threads
    - Threshold for switching to single track mode, size of event buffer
- Fitness function: minimize simulation time while keeping in predefined memory limits
- TMVA analysis will come next

# “VecGeom” in action

- Geant-Vector prototype can run complete first particle-detector simulations using VecGeom ( or with ROOT/TGeo )
- measured a **total simulation runtime improvement of 40%** going from ROOT/TGeo to VecGeom for small example



ExN03 example



Caching&SIMD - overheads

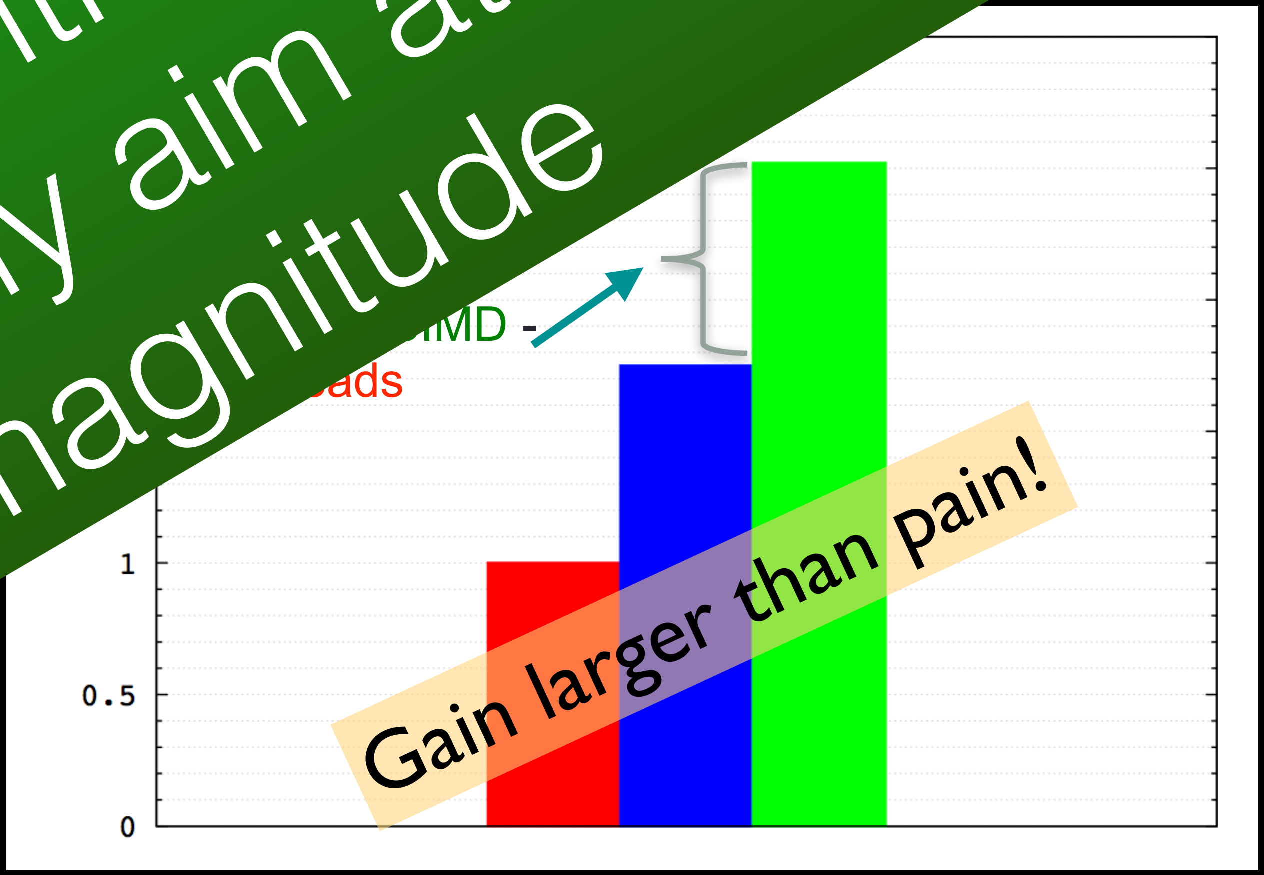
Gain larger than pain!

# “VecGeom” in action

- Geant-Vector prototype can run c detector simulations using Ve
- measured a **total simul** going from ROOT/T

A speedup between 3 & 5 seems today within reach

We should really aim at one order of magnitude



Gain larger than pain!

# Using GPU

- Can we use GPU just as we use worker threads on CPU now?
  - Initialise geometry, physics tables; same as on CPU - done
  - Pick-up CPU baskets, re-basketize GPU friendly – done
  - Asynchronous data transfer kernels and propagation kernels – partially done
  - Deliver back transported baskets – to do
- Prerequisites
  - Propagation code in GeantV to compile as kernel with NVCC — to do
  - Contiguous GeantTrack_v container to avoid gather/scatter towards GPU, refactoring non-POD navigation history
- CPU-GPU data exchange — starting now
  - Expecting issues in load balancing, latency, propagating action requests (e.g. garbage collection)

# Other accelerators

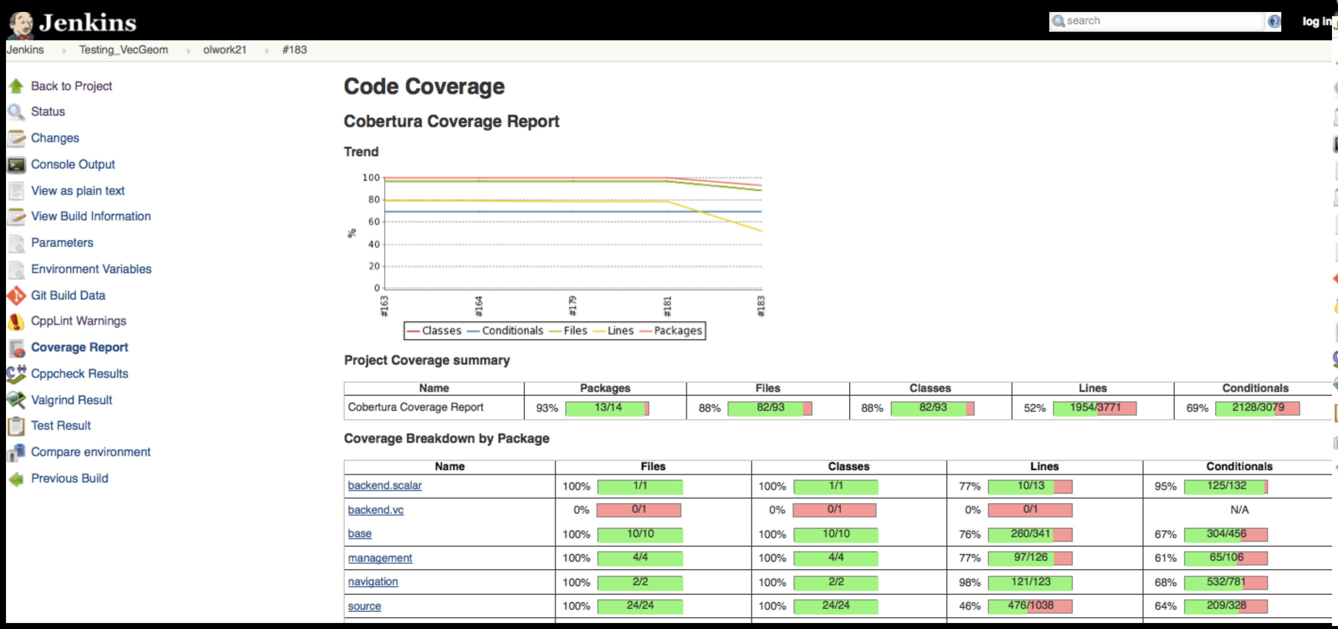
- Xeon Phi
  - Keen interest from Intel
  - Some of the code already ported by intel onto Phi
  - IPCC submitted, hope to get 2 FTE x 2 y
- AMD
  - Offer to pay a doctoral student
  - Identified a thesis director (Prof. D. Hill, Clermont-Ferrant university)
  - Looking for students

# FastSim

- FCC studies are now being made with GEANT4 fast simulation option and ATLFAST parametrisations (Anna & Themis)
- As soon as this works, we will do the same with GeantV
- We may have our first customer in production!

# GeantV/VecGeom Jenkins

- https://geantbuild.cern.ch available to geant-dev egroup
- SL5/SL6 (i686/x86_64), OS X (10.8/10.9), gcc4.8.1/4.9.1, clang and CUDA builds
- Nightly builds of GeantV and VecGeom with email plugin for notification users in e-group
- Code coverage
- Coding conventions (currently only Google, looking at clang-tidy)
- CTests with future CDash integration
- Code checks: Coverity, Cppcheck..
- Benchmarks/prototype checks (future possibility to store benchmarks in DB and build live plots)



## Code Coverage

### Cobertura Coverage Report

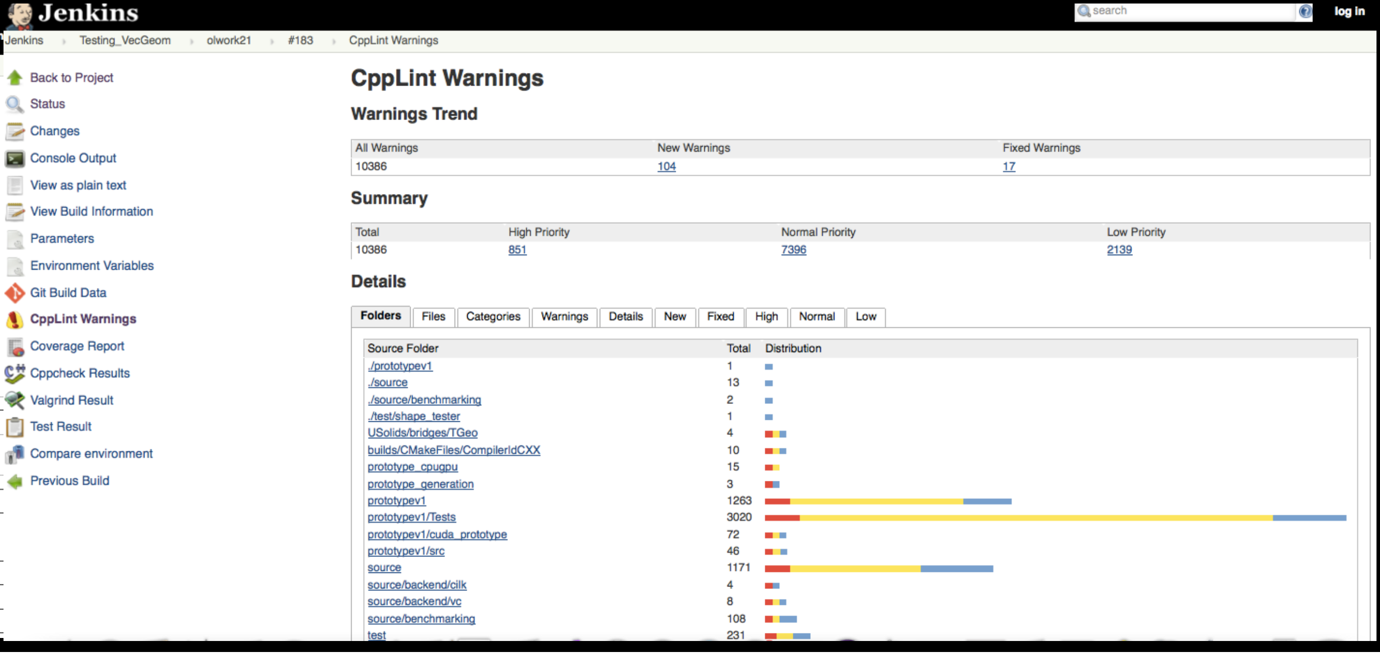
**Trend**

**Project Coverage summary**

| Name | Packages | | Files | | Classes | | Lines | | Conditionals | |
|---|---|---|---|---|---|---|---|---|---|---|
| Cobertura Coverage Report | 93% | 13/14 | 88% | 82/93 | 88% | 82/93 | 52% | 1954/3771 | 69% | 2128/3079 |

**Coverage Breakdown by Package**

| Name | Files | | Classes | | Lines | | Conditionals | |
|---|---|---|---|---|---|---|---|---|
| backend.scalar | 100% | 1/1 | 100% | 1/1 | 77% | 10/13 | 95% | 125/132 |
| backend.vc | 0% | 0/1 | 0% | 0/1 | 0% | 0/1 | N/A | |
| base | 100% | 10/10 | 100% | 10/10 | 76% | 260/341 | 67% | 304/456 |
| management | 100% | 4/4 | 100% | 4/4 | 77% | 97/126 | 61% | 65/106 |
| navigation | 100% | 2/2 | 100% | 2/2 | 98% | 121/123 | 68% | 532/781 |
| source | 100% | 24/24 | 100% | 24/24 | 46% | 476/1038 | 64% | 209/328 |



## CppLint Warnings

### Warnings Trend

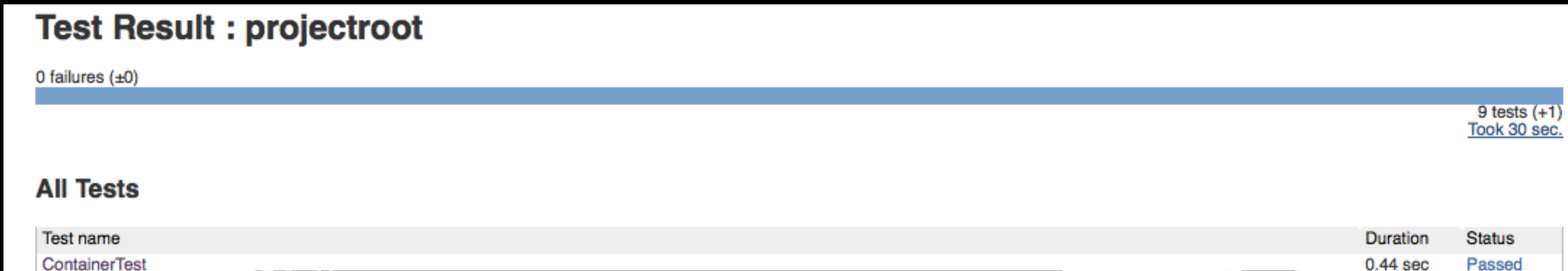
| All Warnings | New Warnings | Fixed Warnings |
|---|---|---|
| 10386 | 104 | 17 |

### Summary

| Total | High Priority | Normal Priority | Low Priority |
|---|---|---|---|
| 10386 | 851 | 7396 | 2139 |

### Details

| Source Folder | Total |
|---|---|
| ./prototypev1 | 1 |
| ./source | 13 |
| ./source/benchmarking | 2 |
| ./test/shape_tester | 1 |
| USolids/bridges/TGeo | 4 |
| builds/CMakeFiles/CompilerIdCXX | 10 |
| prototype_cpugpu | 15 |
| prototype_generation | 3 |
| prototypev1 | 1263 |
| prototypev1/Tests | 3020 |
| prototypev1/cuda_prototype | 72 |
| prototypev1/src | 46 |
| source | 1171 |
| source/backend/cilk | 4 |
| source/backend/vc | 8 |
| source/benchmarking | 108 |
| test | 231 |



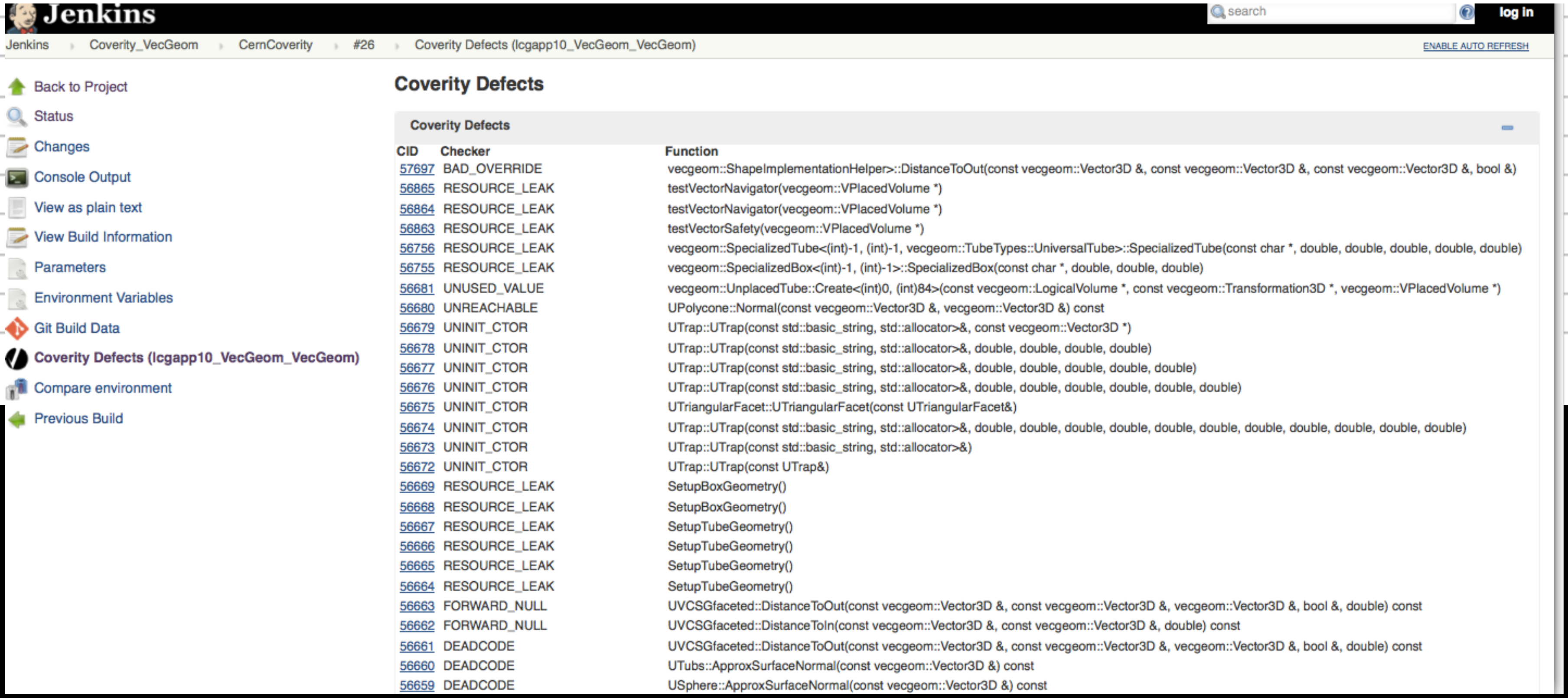
# Test Result : projectroot

0 failures (±0)

9 tests (+1)
Took 30 sec.

## All Tests

| Test name | Duration | Status |
|---|---|---|
| ContainerTest | 0.44 sec | Passed |
| ImportFromRootFileTest | 0.36 sec | Passed |
| TestExportToROOT | 0.71 sec | Passed |
| Transformation3DTest | 0.34 sec | Passed |
| complex_test1 | 5.9 sec | Passed |
| create_geometry | 0.4 sec | Passed |
| root_geometry | 0.35 sec | Passed |
| testVectorSafety | 0.58 sec | Passed |
| trd_validation | 21 sec | Passed |



## Coverity Defects

**Coverity Defects**

| CID | Checker | Function |
|---|---|---|
| 57697 | BAD_OVERRIDE | vecgeom::ShapeImplementationHelper>::DistanceToOut(const vecgeom::Vector3D &, const vecgeom::Vector3D &, const vecgeom::Vector3D &, bool &) |
| 56865 | RESOURCE_LEAK | testVectorNavigator(vecgeom::VPlacedVolume *) |
| 56864 | RESOURCE_LEAK | testVectorNavigator(vecgeom::VPlacedVolume *) |
| 56863 | RESOURCE_LEAK | testVectorSafety(vecgeom::VPlacedVolume *) |
| 56756 | RESOURCE_LEAK | vecgeom::SpecializedTube<(int)-1, (int)-1, vecgeom::TubeTypes::UniversalTube>::SpecializedTube(const char *, double, double, double, double, double) |
| 56755 | RESOURCE_LEAK | vecgeom::SpecializedBox<(int)-1, (int)-1>::SpecializedBox(const char *, double, double, double) |
| 56681 | UNUSED_VALUE | vecgeom::UnplacedTube::Create<(int)0, (int)84>(const vecgeom::LogicalVolume *, const vecgeom::Transformation3D *, vecgeom::VPlacedVolume *) |
| 56680 | UNREACHABLE | UPolycone::Normal(const vecgeom::Vector3D &, vecgeom::Vector3D &) const |
| 56679 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&, const vecgeom::Vector3D *) |
| 56678 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&, double, double, double, double) |
| 56677 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&, double, double, double, double, double) |
| 56676 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&, double, double, double, double, double, double) |
| 56675 | UNINIT_CTOR | UTriangularFacet::UTriangularFacet(const UTriangularFacet&) |
| 56674 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&, double, double, double, double, double, double, double, double, double, double, double) |
| 56673 | UNINIT_CTOR | UTrap::UTrap(const std::basic_string, std::allocator>&) |
| 56672 | UNINIT_CTOR | UTrap::UTrap(const UTrap&) |
| 56669 | RESOURCE_LEAK | SetupBoxGeometry() |
| 56668 | RESOURCE_LEAK | SetupBoxGeometry() |
| 56667 | RESOURCE_LEAK | SetupTubeGeometry() |
| 56666 | RESOURCE_LEAK | SetupTubeGeometry() |
| 56665 | RESOURCE_LEAK | SetupTubeGeometry() |
| 56664 | RESOURCE_LEAK | SetupTubeGeometry() |
| 56663 | FORWARD_NULL | UVCSGfaceted::DistanceToOut(const vecgeom::Vector3D &, const vecgeom::Vector3D &, vecgeom::Vector3D &, bool &, double) const |
| 56662 | FORWARD_NULL | UVCSGfaceted::DistanceToIn(const vecgeom::Vector3D &, const vecgeom::Vector3D &, double) const |
| 56661 | DEADCODE | UVCSGfaceted::DistanceToOut(const vecgeom::Vector3D &, const vecgeom::Vector3D &, vecgeom::Vector3D &, bool &, double) const |
| 56660 | DEADCODE | UTubs::ApproxSurfaceNormal(const vecgeom::Vector3D &) const |
| 56659 | DEADCODE | USphere::ApproxSurfaceNormal(const vecgeom::Vector3D &) const |

# Short term plans

- Run the prototype with the full (simplified) CMS geometry
- Develop simple *event record* for kinematics
- Target are x86 and NVIDIA
- Compare *results* with Geant4 FTP_BERT & Tabulated physics lists
- In the new year optimise the code and present the results at CHEP

# 2015 plans

- Study vectorisation of time consuming EM effects (Multiple scattering and ionisation)
- Port on Xeon Phi, ARM and AMD
    - Optimisation effort will depend on early results and vendors' help
- Introduce I/O for hits and study I/O parallelism
- Continue optimisation of Geometry, in particular with the introduction of fast voxelisation
- Develop *fast simulation* hooks / framework
- Perform optimisation of programme parameters

# 2015 plans

- Design & Install final development system
  - Continuous integration, build and test system (jenkins, cdash, coverity…)
  - Coding conventions (need to find the tool)
  - Gitlab development model
  - Validation & non-regression infrastructure

# 2015 plans

- Design of major subsystems
  - Electromagnetic
  - Hadronic
  - Scoring & Hits
  - I/O
  - Generator interface
  - Event model

# Without forgetting…

- Documentation
- Coding conventions
- Some type and function naming which are confusing
- Support for AMD (OpenCL, sycl)
- Testing, testing, testing ( standalone unit tests, shape stress tests, continuous integration )
- Extend benchmarks
- Continuous performance monitoring
- Issue tracking (bugs should be reported ... )

# Longer term plans

- Development of improved, high-performance electromagnetic and hadronic packages
- Integrated fast / full simulation framework
- Low energy neutron integration
- Biased sampling
- …sky is the limit ;)

# Synergies, synergies, synergies

- We are currently developing a *vector signature* math library
  - It would make sense to have it as a part of TMath
- Our Jenkins / git infrastructure could be merged with the one being developed for the group
  - Work ongoing Oksana + Patricia
- VecGeom will be usable by GEANT4 soon behind the Usolid interface
  - I think we should move one step beyond toward ugeom
- We have defined coding conventions, but we do not have a tool

# Outlook

- Encouraging status
    - Expose parallelism, minimize contentions, real size application, stay architecture agnostic and portable
- Very large program of work
- Involvement of other communities would be very important
    - On the model of the collaboration FNAL - CERN
- Wish us good luck