GeantV

John Apostolakis (CERN) for the GeantV development team

G. Amadio (UNESP), A. Ananya (CERN), J. Apostolakis (CERN) , A. Arora (CERN), M. Bandieramonte (CERN), A. Bhattacharyya (BARC), C. Bianchini (UNESP), R. Brun (CERN), P. Canal (FNAL), F. Carminati (CERN), L. Duhem (intel), D. Elvira (FNAL), A. Gheata (CERN), M. Gheata (CERN), I. Goulas (CERN), F. Hariri (CERN), R. Iope (UNESP), S. Y. Jun (FNAL), G. Lima (FNAL), A. Mohanty (BARC), T.Nikitina (CERN), M.Novak (CERN), W. Pokorski (CERN), A. Ribon (CERN), R. Sehgal (BARC), O. Shadura (CERN), S. Vallecorsa (CERN), S. Wenzel (CERN), Y. Zhang (CERN)

HEP Software Foundation Workshop

LAL, May 2 – 4, 2016

Outline

- **GeantV**ectorized
  - Challenges, ideas, goals
- Main components
  - Geometry, physics, propagation in field, scheduler
- Development- different components in different stages
- Performance and benchmarks
  - Vectorization: overheads vs. gains
  - Geometry and physics performance: CPU and acceleratorsbenchmarks
- Results on parts, simple setups and full detector

The challenge

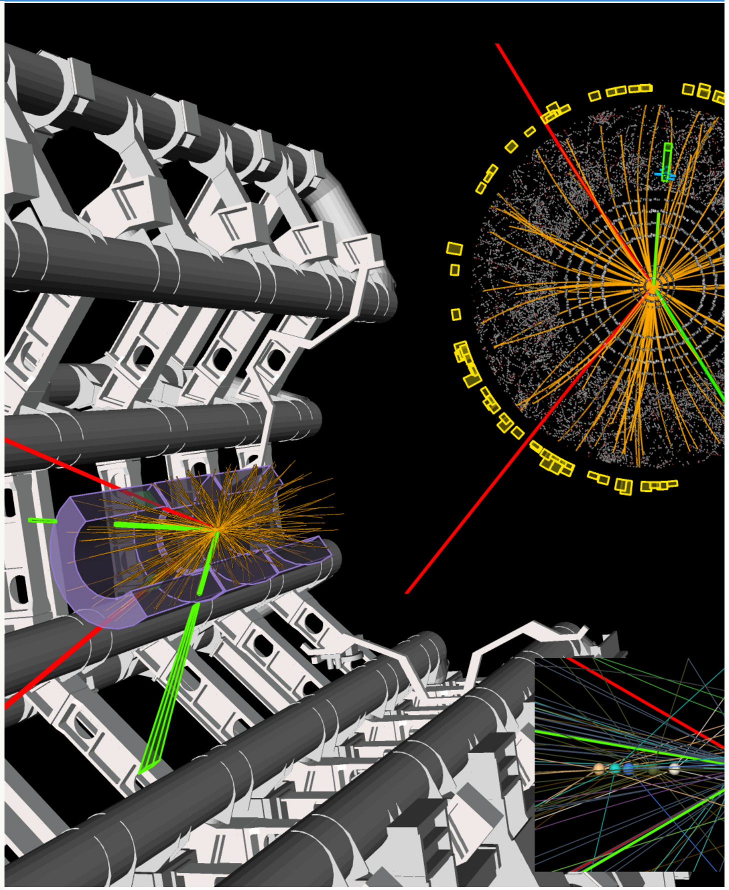
Goal of improving performance by a factor between 2.5 and 5 for large HEP detectors

Parts of transport of particles

- Navigating in complex geometries - millions of volumes
- Interactions using physics models,
- Propagation in EM field

Strongly CPU-bound, but must also generate user-determined output (hits & 'truth' information)

The LHC uses about 50% of its distributed GRID power for detector simulation



http://atlas.ch

The ideas

- Transport particles in groups (vectors) rather than one by one
  - Group particles by geometry volume or same physics
  - No free lunch: data gathering overheads < vector gains
- Dispatch SoA to functions with vector signatures
  - Use backends to abstract interface: vector, scalar
  - Use backends to insulate technology/library: Vc, Cilk+, VecMic, …
- Redesign the library and workflow to target fine grain parallelism
  - CPU, GPU, Phi, Atom, …
  - Aim for a 2.5x-5x faster code, understand hard limits for more

```
distance( double &);
```

Scalar interface

```
distance( vector_type  &);
```

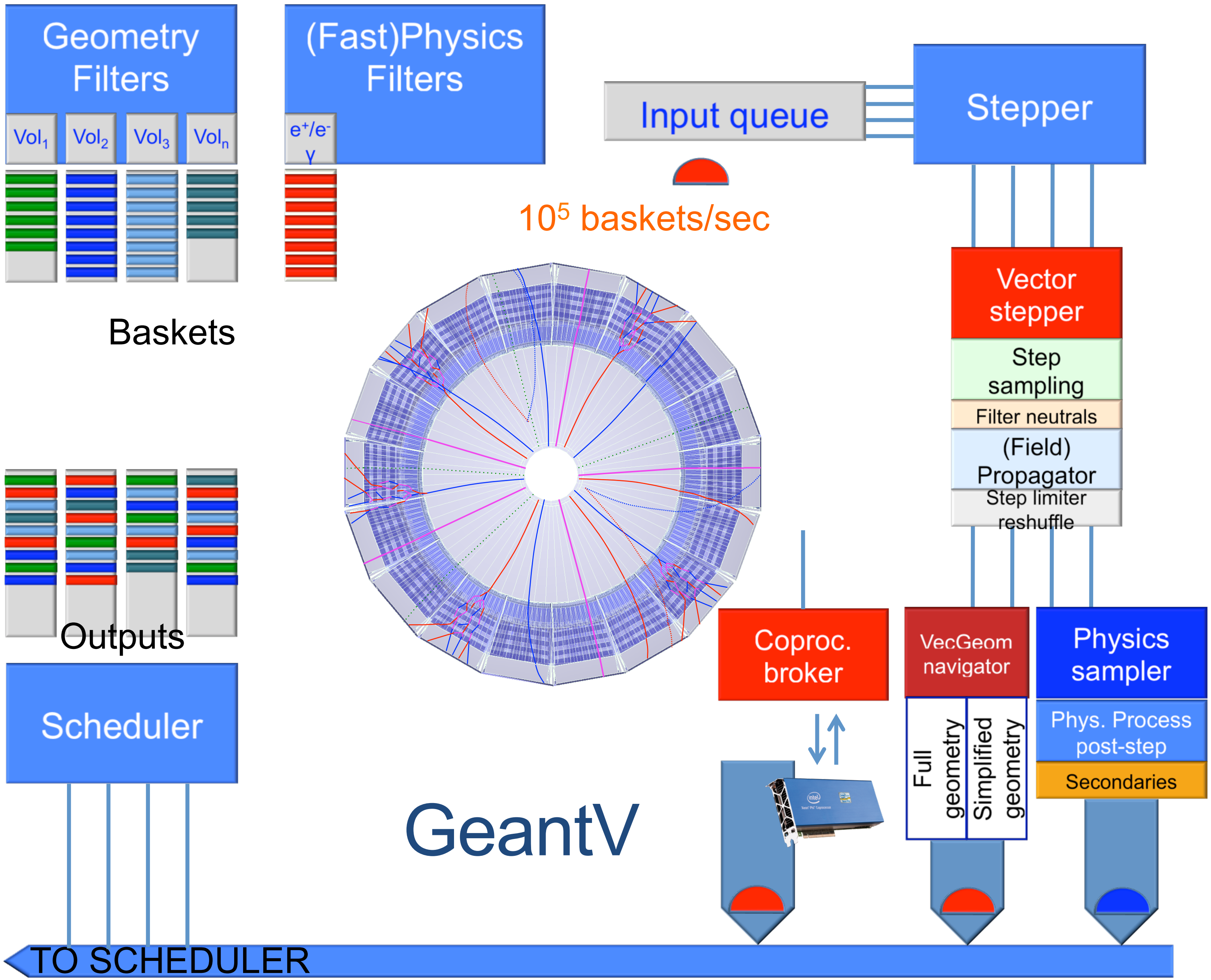
Vector interface

```
template<class Backend>
Backend::double_t
common_distance_function( Backend::double_t input )
{
   // Single kernel algorithm using Backend types
}
```

```
struct ScalarBackend
{
   typedef double double_t;
   typedef bool   bool_t;
   static const bool IsScalar=true;
   static const bool IsSIMD=false;
};
// Functions operating with backend types
```

```
struct VectorVcBackend
{
   typedef Vc::double_v double_t;
   typedef Vc::double_m bool_t;
   static const boolIsScalar=false;
   static const bool IsSIMD=true;
};
// Functions operating with backend types
```

code.compeng.uni-frankfurt.de/projects/vc


Geometry Filters
Vol_1 Vol_2 Vol_3 Vol_n
(Fast)Physics Filters
e^+/e^- γ
Baskets
Input queue
Stepper
10^5 baskets/sec
Vector stepper
Step sampling
Filter neutrals
(Field) Propagator
Step limiter reshuffle
Outputs
Scheduler
GeantV
Coproc. broker
VecGeom navigator
Full geometry
Simplified geometry
Physics sampler
Phys. Process post-step
Secondaries
TO SCHEDULER

Performance & Benchmarks

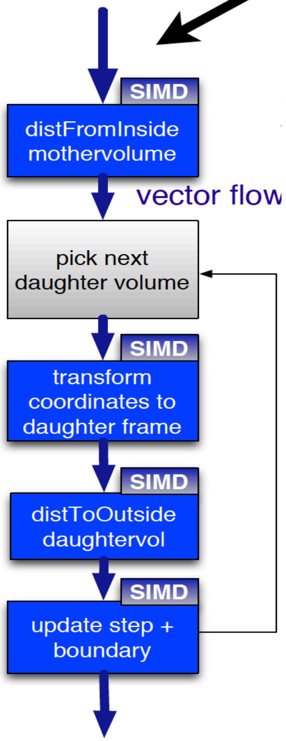
- Computing performance is a raison d’etre for GeantV
- Almost every component/class has a test and a benchmark for scalar and vector performance
  - Automated test, which must be run for each solid
  - Individual physics model sampling methods
- Benchmarks for
  - Navigation in simple and full detector geometries
  - ‘Full’ simulation using all integrated components

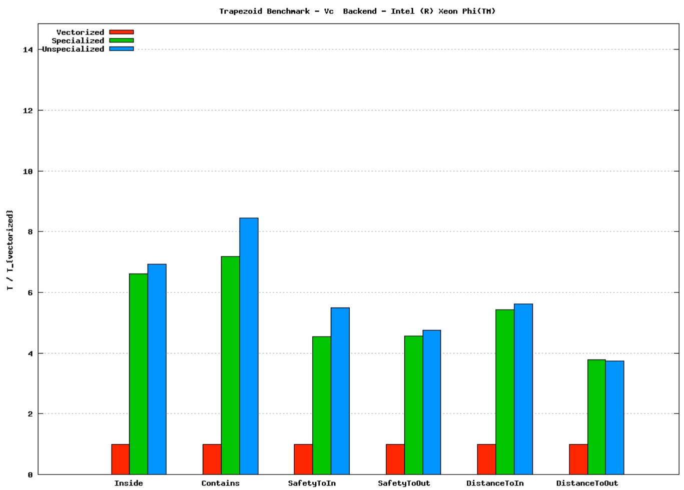
Geometry solid performance

- Geometry is 30-40% of the total CPU time in Geant4
- Vectorized geometry algorithms to take maximum advantage of SIMD
- Benchmark each method against implementations in Root, Geant4 and USolids library
- Performance gains also in scalar mode for some cases
- Testing the same on GPU

Speedup vs Number of particles	16 particles	1024 particles	SIMD max
Intel Ivy-Bridge (AVX)	2.8	4	4
Intel Haswell (AVX2)	3	5	4
Intel Xeon Phi (AVX-512)	4.1	4.8	8

Overall performance for a simplified detector vs. scalar ROOT/5.34.17

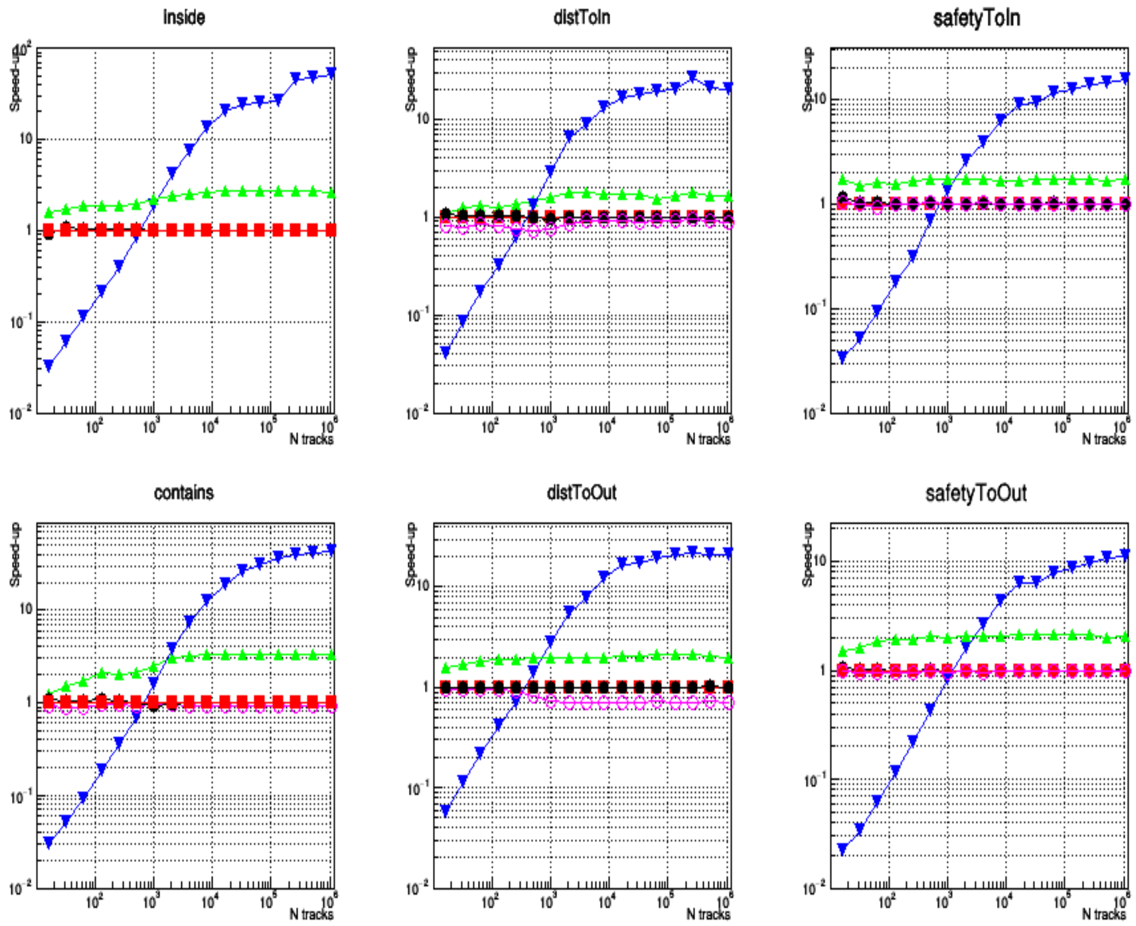




Vectorization performance for trapezoid shape navigation (Xeon®Phi® C0PRQ-7120 P)

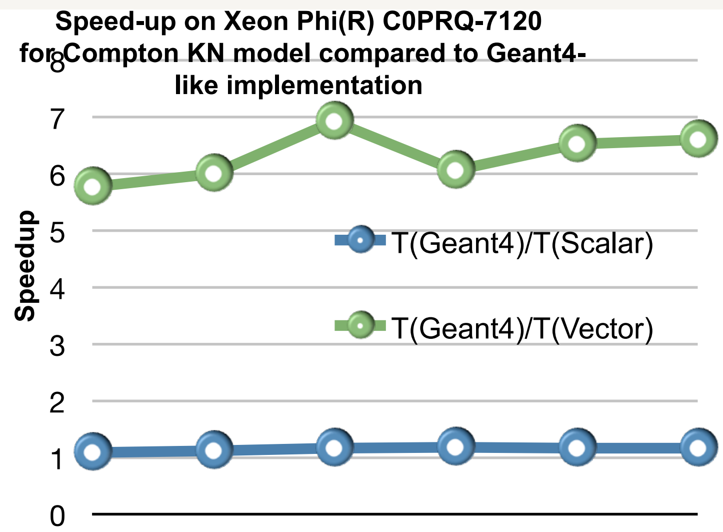
Geometry performance on K20 GPU

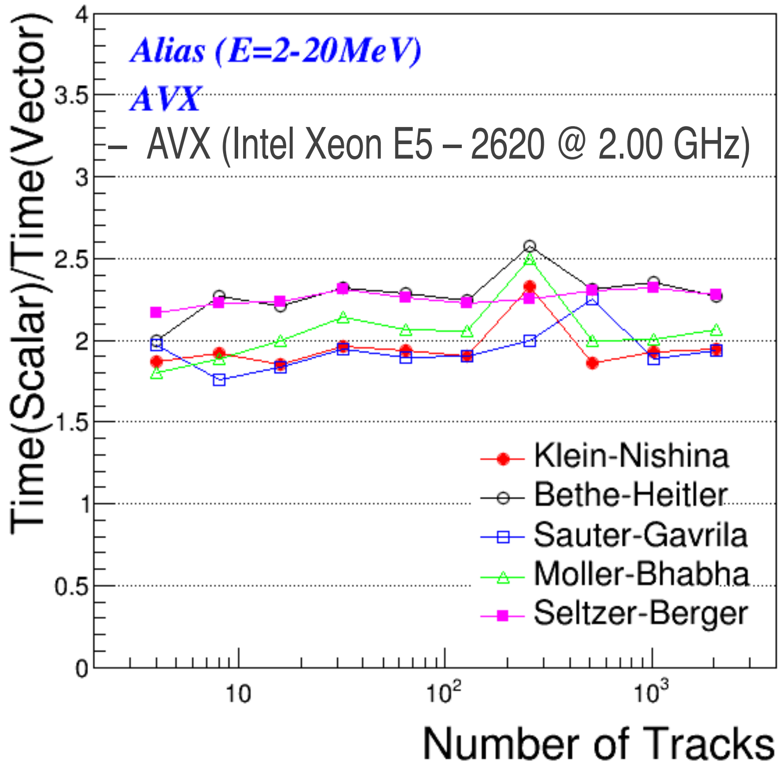
- Speedup for different navigation methods of the box shape, normalized to scalar CPU
  - Scalar (specialized/unspecialized)
  - Vector
  - GPU (Kepler K20)
  - ROOT
- Data transfer in/out is asynchronous
  - Measured only the kernel performance – relying on constant throughput to hide transfer latency
- Will explore how to saturate the die: either with large track containers, running a single kernel, or with smaller containers dynamically scheduled.
- Demonstrates that we can run the same code on CPU/accelerators; further optimization anticipated



Physics

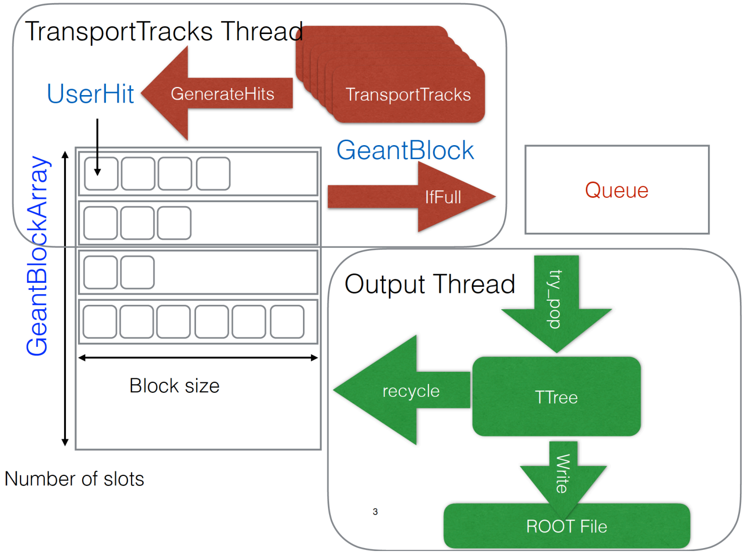
- Objective: a vector/accelerator friendly re-write of physics code
  - Alias sampling (first choice)
  - Accept/reject (when needed)
- Identify when
- The initial vectorized gamma models show performance gain already for small vector size
  - 1.2x – 2.3x on SSE
  - 1.8x – 2.3x on AVX
  - 6x on Xeon Phi
- Use profilers to identify hotspots (vtune, Mac Instruments, igprof)





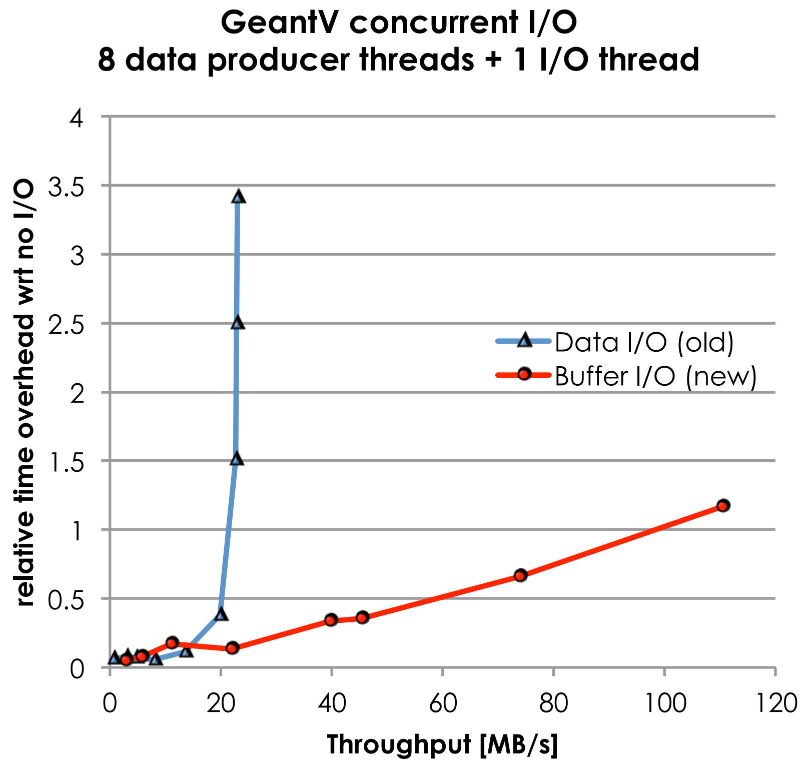
Hits/digits I/O

- "Data" mode
  - Send concurrently data to one thread dealing with full I/O



- "Buffer" mode
  - Send concurrently local trees connected to memory files produced by workers to one thread dealing with merging/write to disk

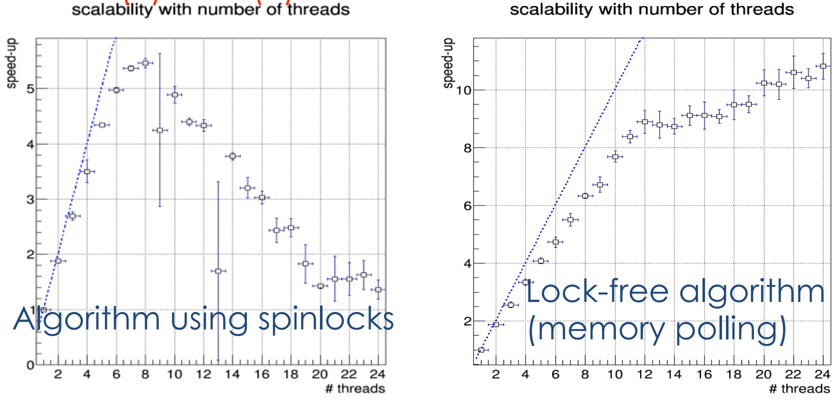
- Integrating user code with a highly concurrent framework should not spoil performance

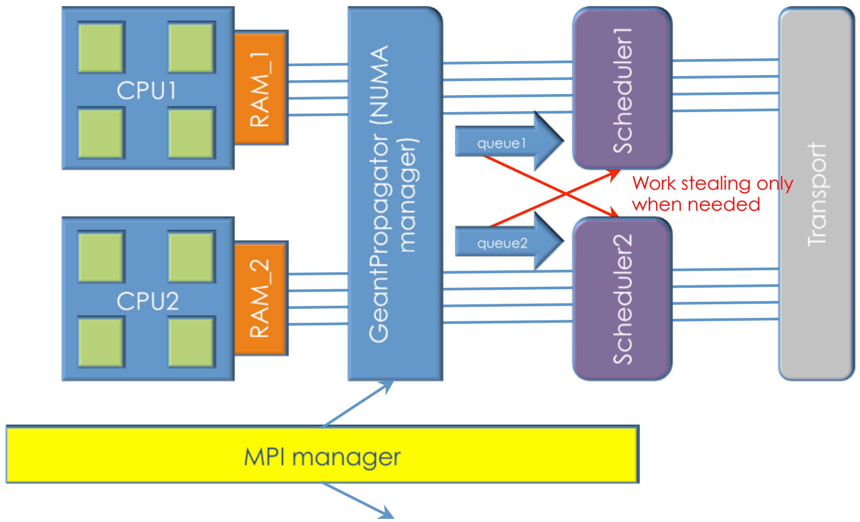


Basketizer performance

- Investigated different ways of scheduling & sharing work - lock free queues, ..
  - Changes in scheduler require non-trivial effort (rewrite)
- Seqerial still large, due to high re-basketizing load (concurrent copying)
  - $O(10^5)$ baskets/second on Intel Core i7™
  - Algorithm already lock free
  - Rate will go down with addition of physics processes
- Ongoing work to improve scalability
  - Re-use baskets for several steps
  - Introduce NUMA awareness
  - Clone scheduler in NUMA-aware groups for use in many cores (e.g. KNL)

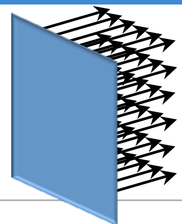
Rebasketizing
2x Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz

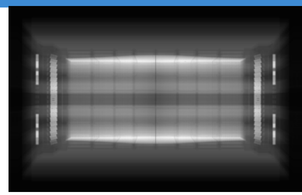


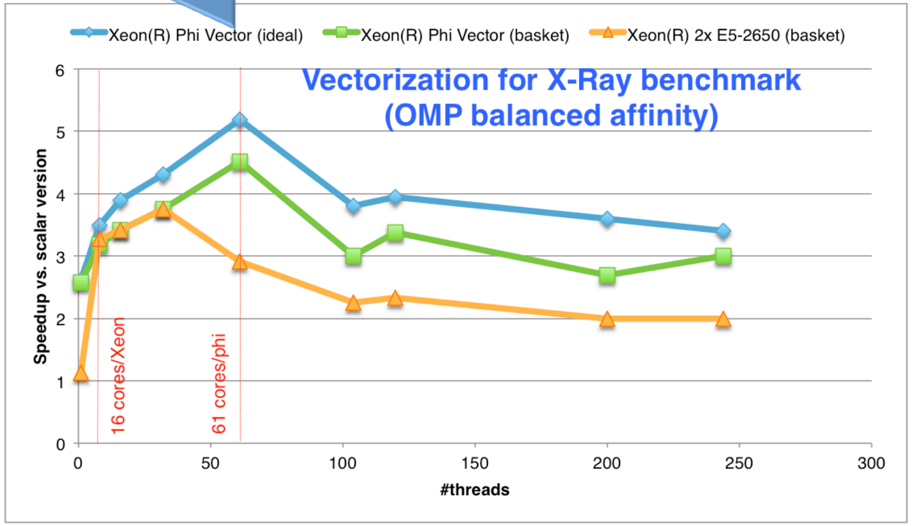


X-ray benchmark: Vector performance

- The X-Ray benchmark tests geometry navigation in a detector geometry
- In simple geometry example (**concentric tubes**) emulates a tracker detector - used for Xeon©Phi benchmark
  - OMP parallelism + "basket" model
- Gained up to 4.5 from vectorization in basketized mode
  - Approaching the ideal case.



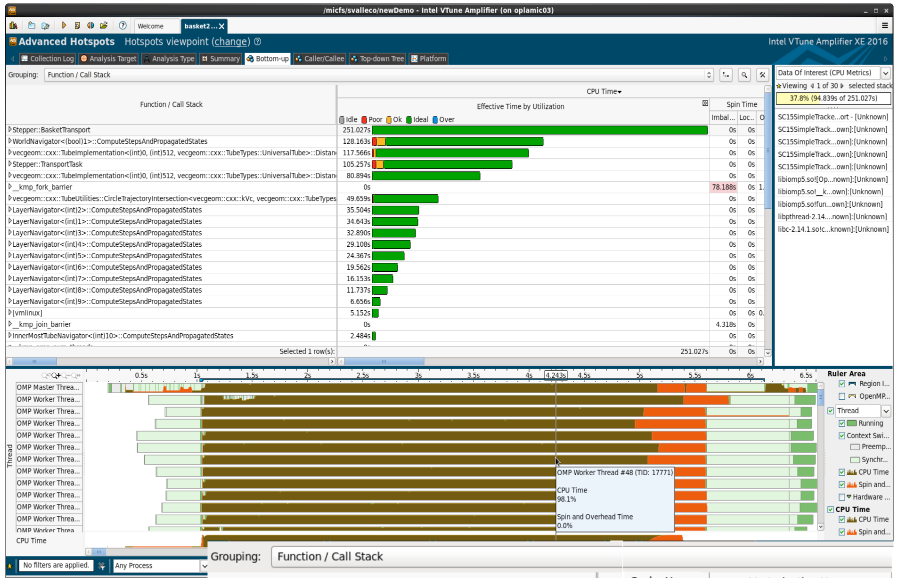


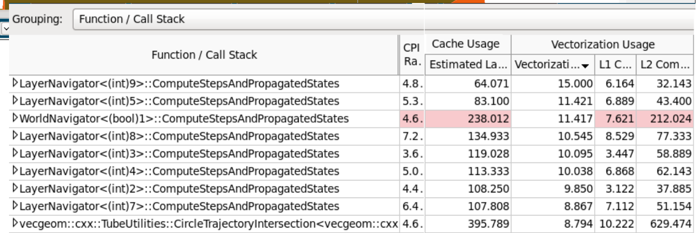


- Scalar case: Simple loop over pixels
- Ideal vectorization case: Fill vectors with N times the same X-ray
- Realistic (basket) case: Group baskets per geometry volume

Profiling for the X-Ray benchmark

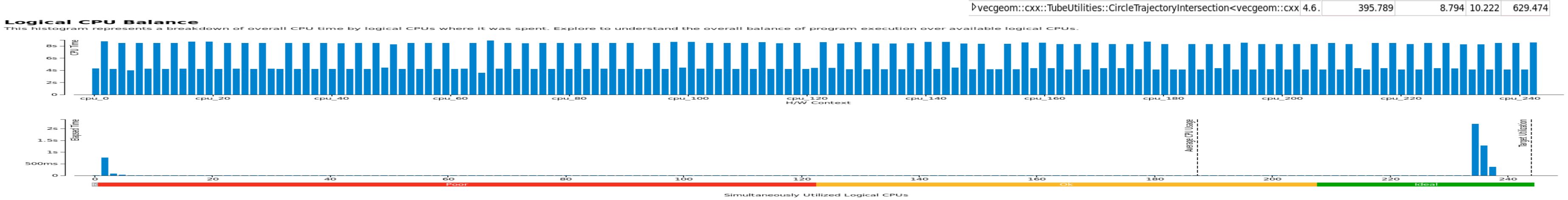
- Good vectorization intensity, thread activity and core usage for the X-Ray basketized benchmark on a Xeon Phi (61 core C0PRQ-7120 P)
- The performance tools gave us good insight on the current performance of GeantV



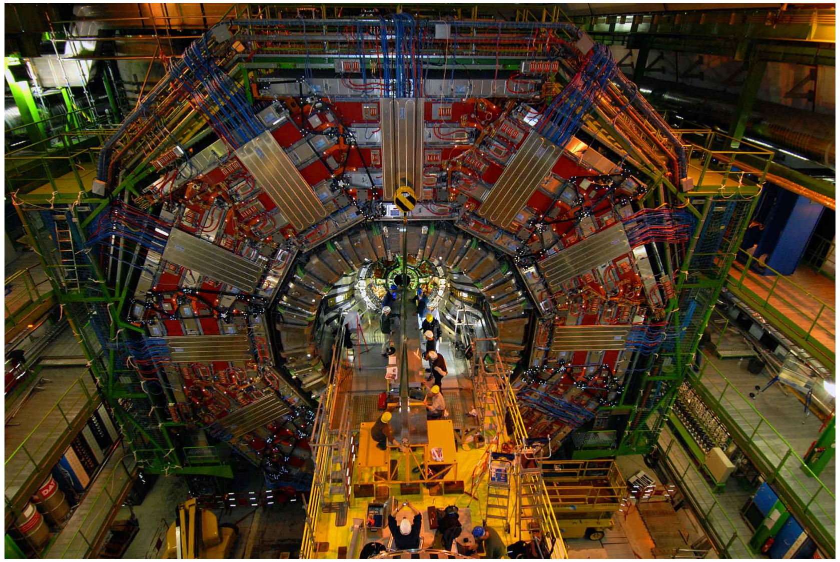


Grouping: Function / Call Stack

Function / Call Stack	CPI Ra.	Cache Usage: Estimated La...	Vectorization Usage: Vectorizati...	L1 C...	L2 Com...
LayerNavigator<(int)9>::ComputeStepsAndPropagatedStates	4.8.	64.071	15.000	6.164	32.143
LayerNavigator<(int)5>::ComputeStepsAndPropagatedStates	5.3.	83.100	11.421	6.889	43.400
WorldNavigator<(bool)1>::ComputeStepsAndPropagatedStates	4.6.	238.012	11.417	7.621	212.024
LayerNavigator<(int)8>::ComputeStepsAndPropagatedStates	7.2.	134.933	10.545	8.529	77.333
LayerNavigator<(int)3>::ComputeStepsAndPropagatedStates	3.6.	119.028	10.095	3.447	58.889
LayerNavigator<(int)4>::ComputeStepsAndPropagatedStates	5.0.	113.333	10.038	6.868	62.143
LayerNavigator<(int)2>::ComputeStepsAndPropagatedStates	4.4.	108.250	9.850	3.122	37.885
LayerNavigator<(int)7>::ComputeStepsAndPropagatedStates	6.4.	107.808	8.867	7.112	51.154
vecgeom::cxx::TubeUtilities::CircleTrajectoryIntersection<vecgeom::cxx	4.6.	395.789	8.794	10.222	629.474



Putting It All Together - CMS Yardstick



- Some of the improvements can be back ported to G4
- Overhead of basket handling is under control
- Ready to take advantage of vectorization throughout.

Improvement Factors (total) with respect to G4

Legacy (TGeo) Geometry library:

- **1.5** → Algorithmic improvements in infrastructure.

2015 VecGeom (estimate)

- **2.4** → Algorithmic improvements in Geometry

Upcoming VecGeom (early result)

- **3.3** → Further Geometric algorithmic improvements and some vectorization

Summary

- GeantV is in development mode – and in many cases researching different approaches or techniques
- A significant performance gain is a raison d'etre of GeantV
- Benchmarking of individual classes is constantly done
- Benchmarks of simple setups are created to monitor, understand and improve the performance of new components
- Profilers are key tools in performance improvement

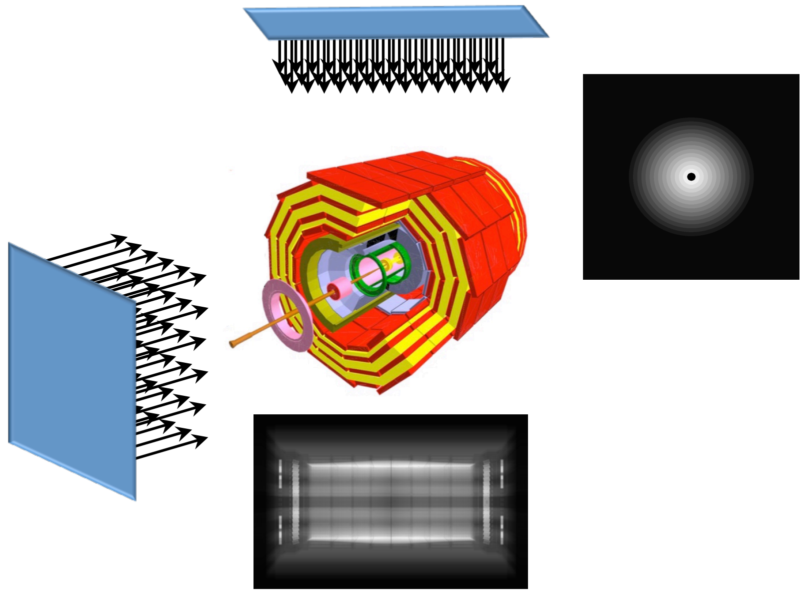
The authors acknowledge the contribution and support from

- Intel (funding, technical and expert advisory, SW/HW)
- CERN openlab

Thank you!

The X-Ray benchmark

- The X-Ray benchmark tests geometry navigation in a real detector geometry
- *X-Ray* scans a module with virtual rays in a grid corresponding to pixels on the final image
  - Each ray is propagated from boundary to boundary
  - Pixel gray level determined by number of crossings
- A simple geometry example (concentric tubes) emulating a tracker detector used for Xeon©Phi benchmark
  - To probe the vectorized geometry elements + global navigation as task
  - OMP parallelism + “basket” model

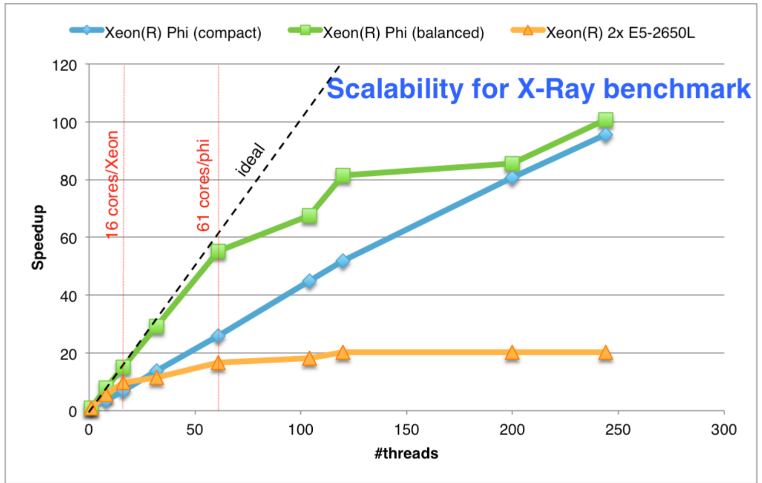


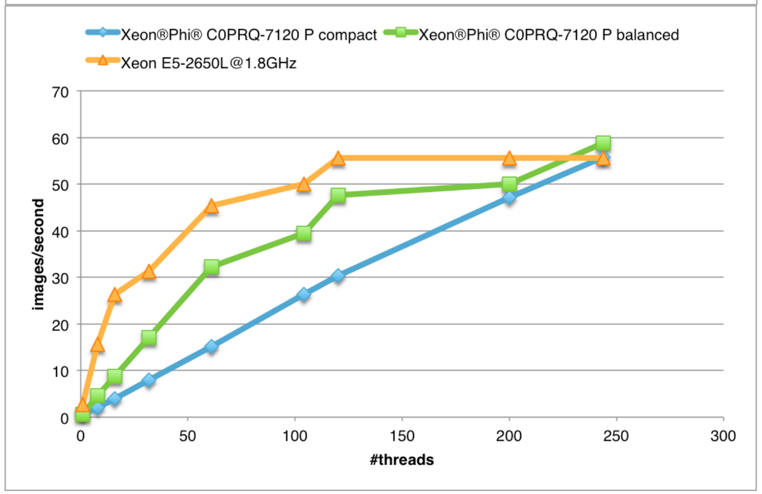
Scalability and throughput

- Better behavior using OMP balanced
  - Approaching well the ideal curve up to native cores count
  - Balanced threading converges towards the compact model as all the thread slots are filled

- It's worth to run Xeon Phi saturated for our application

- The throughput performance for a saturated KNC is equivalent (for this setup) to the dual Xeon E5-2650L@1.8GHz server which hosts the card.



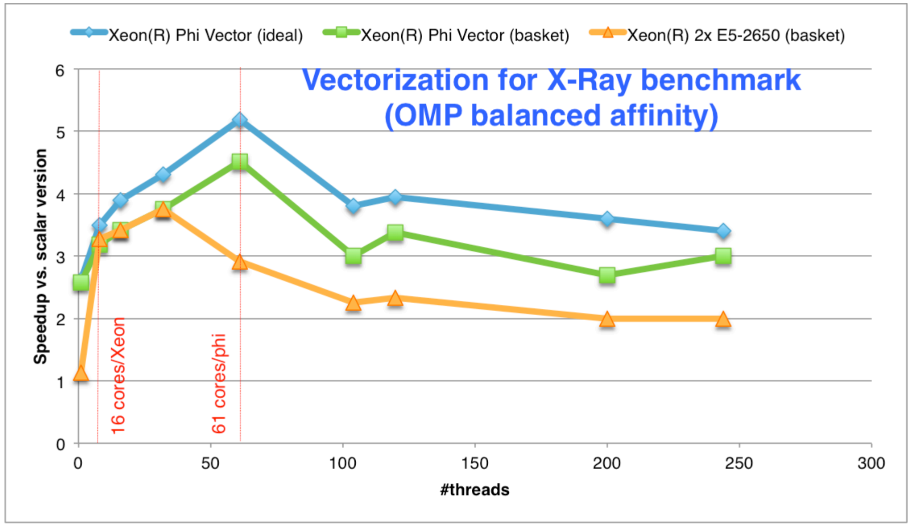


Next steps

- Repeat the test with the introduction of
  - Vectorised EM physics
  - Vectorised transport in Mag Field
- Develop simple classes for materials and particles to be able to run on coprocessors to enable physics on the GPU and Xeon Phi full CMS yardstick
- … implementing a "preliminary performance yard-stick" combining all prototype features
  - SIMD gains in the full CMS experiment setup
  - Coprocessor broker in action: part of the full transport kernel running on Xeon®Phi® and GPGPU
  - Scalability and NUMA awareness for rebasketizing procedure
  - … achieving these just moves the target a bit further
- … testing scaling up to large node count through MPI, e.g. on CORI
  - Input distribution and Output gathering.

Vector performance

- Gaining up to 4.5 from vectorization in basketized mode
  - Approaching the ideal vectorization case (when no regrouping of vectors is done) .
- Vector starvation starts when filling more thread slots than the core count
  - Performance loss is not dramatic
  - Better vectorization compared to the Sandy-Bridge host (expected)



- Scalar case: Simple loop over pixels
- Ideal vectorization case: Fill vectors with N times the same X-ray
- Realistic (basket) case: Group baskets per geometry volume