GeantV – I/O, MCTruth and Validation Repository

Witek Pokorski (CERN) for the GeantV development team

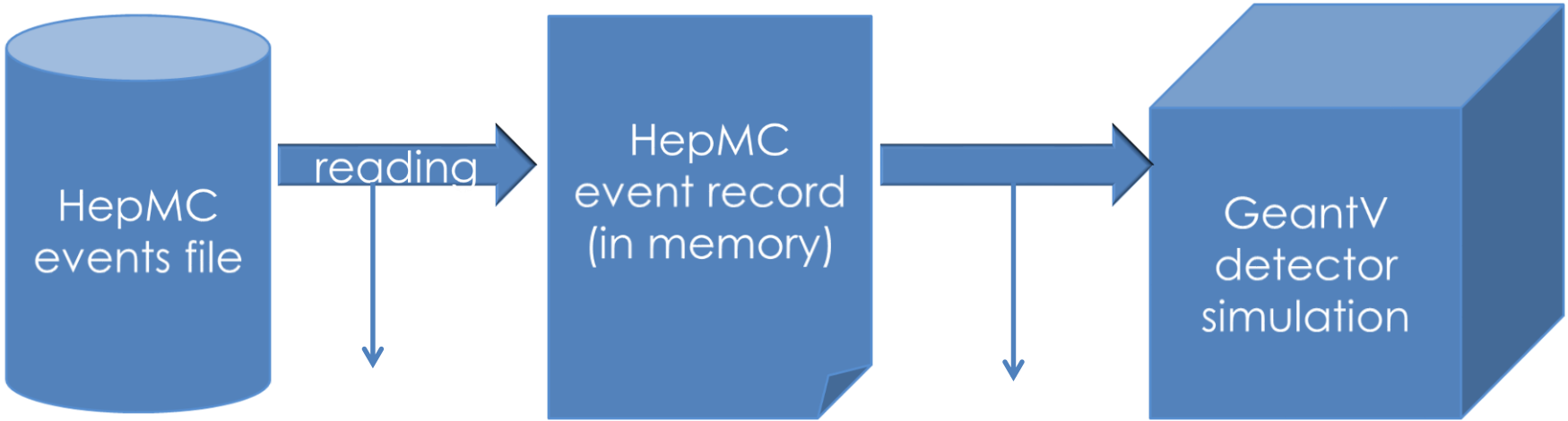
Outline

- GeantV input
- GeantV output
  - hits
  - kinematics
- Physics validation repository
- Conclusion

GeantV Input

- Simulation input
  - particles to be transported through the detector
    - Realistic collision events produced by Monte Carlo event generators (Pythia8, Herwig++, etc)
    - Single particles (like the test beam) to study particular response
- Use of interface (event record) make the generation and the simulation steps independent
  - GeantV does not ‘care’ where the particles are coming from
- Simulation threads concurrently process particles from the input

HepMC as GeantV input format



typical time for 13 TeV pp event ~o(1ms)(negligible compared to simulation step)

Particles fed into concurrent simulation threads

GeantV input implementation

- interface implemented in HepMCGenerator class
  - depends (of course) on HepMC
- can read HepMC ascii and root files
  - automatically recognizes them by extension
- selects stable (outgoing) particles from the event
- applies (if any) Eta, Phi, and momentum cuts

Status of GeantV Input

- implementation complete and fully functional
  - based on HepMC3 event record
- nothing pending for short term development

GeantV Output

- hits
- kinematics (MCTruth)

GeantV hits

- Physics simulation produces ‘hits’ i.e. energy depositions in the sensitive parts of the detector
- Those hits are produced concurrently by all the simulation (TransportTracks) threads
- Thread-safe queues have been implemented to handle asynchronous generation of hits by several threads

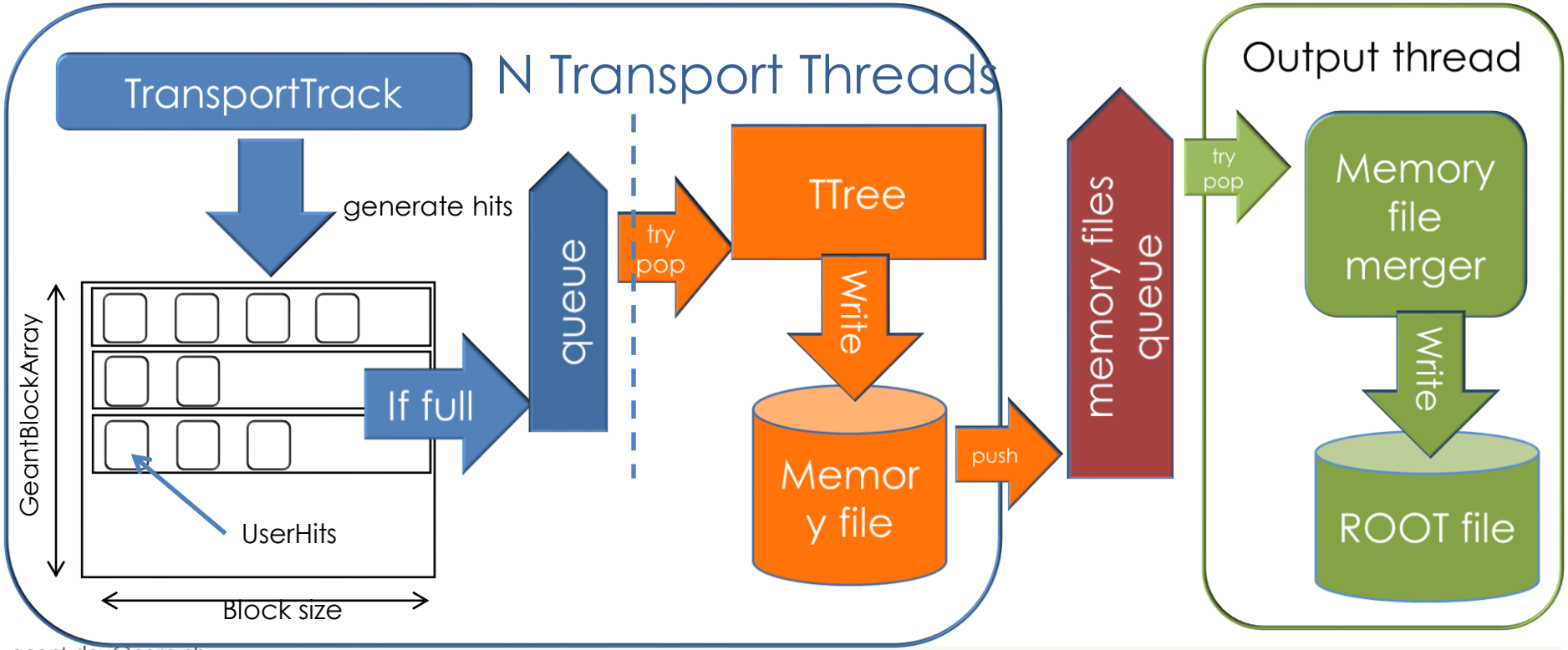
GeantV output threads

- (several) TransportTracks threads generate hits
  - GeantFactory machinery takes care of grouping the hits in HitBlocks and putting them in a queue(s)
- two possible approaches
  - serialization done within (one) output thread
    - discovered a bottleneck
  - performing serialization within each transport thread solved the problem
- ‘writing to disc’ implemented within (single) OutputThread

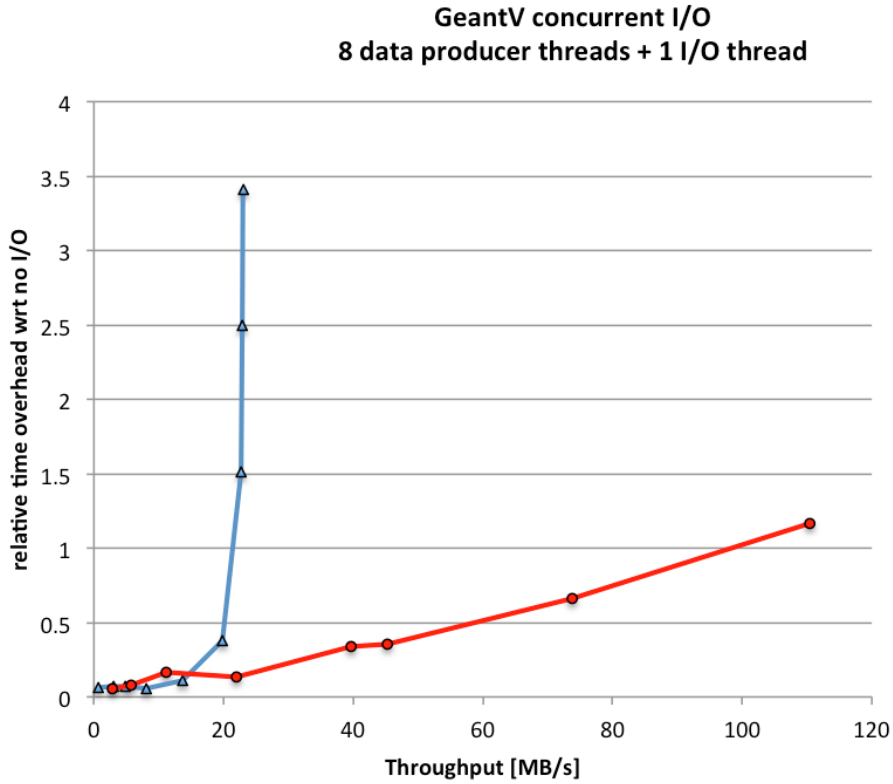
GeantV output implementation

- idea:
  - several ROOT TTrees should be filled in parallel by each of the TransportTracks threads
  - Output thread should be ‘only’ in charge of merging TTrees and writing them to I/O
- implementation:
  - derived from ROOT TParallelMergingFile and parallelMergerServer which are socket-based
  - TThreadMergingFile and TThreadMergingServer use a queue (dcqueue<TBufferFile*>*) as the ‘communication channel’

GeantV I/O data flow



GeantV Output performance



Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz (Haswell)

2x8 cores, HT=2 (i.e. 16 native threads, 32 in hyperthreading mode)

Disk: SSD ~430 MB/s non-cached write speed (measured with: dd if=/dev/zero of=/tmp/testfile bs=1G count=1 oflag=direct)

single thread serialization I/O

GeantV I/O

Hits Output status

- (optional) hits persistency available in GeantV
  - further improvements (reordering of hits in file, etc) possible at later stage
- tested in multithreaded environment with high number of hits generated
  - test hit rate much above typical simulation applications

GeantV kinematics output (MC truth)

- handling of MC truth is problematic per se
  - which particles to store, how to keep connections, where to connect hits
- multithreading adds the complexity
  - order of processing of particles is ‘random’
    - processing of ‘daughter’ particle may be completed before ‘mother’ particle ‘end of life’
  - events need to be ‘put together’ after parallel processing

MC truth

- we can’t (and we don’t need) to store all particles
  - typically no delta-e, no low-E gamma showers, etc needed
- we need to store particles necessary to understand the given event (process)
- we need to store particles to associate hits
- in all cases, we need to (re)connect particles to have consistent event trees

MC truth handling requirements

- no MC truth-handling strategy is perfect, nor complete, but:
  - we need to give user a way to decide
- transport need to provide/allow
  - links between mother and daughter particles
  - the possibility to flag particles as ‘to be stored’
  - possibility to introduce ‘rules’ what to store
  - a way to ‘reconnect’ tracks and hits if some are skipped
    - if we don’t store a particle, we need to update the daughter particles to point back to the last stored one in the chain
- for the final output we need to have some event record
  - for our proof of principle, we can start with HepMC

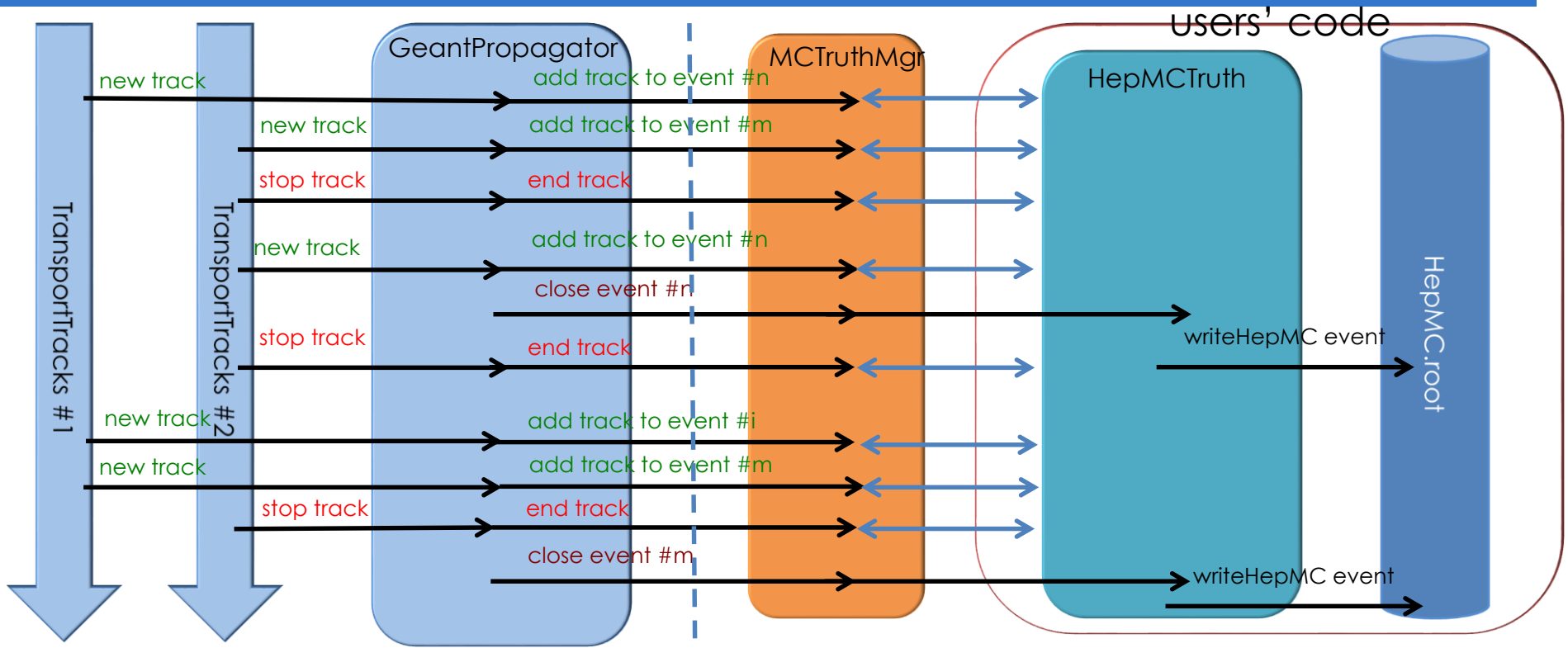
MC truth handling architecture

- light coupling to transport
  - minimal ‘disturbance’ to transport threads
  - maximal flexibility of implementing custom particle history handlers
- interface provided by MCTruthMgr
  - receives (concurrent) notifications from transport threads about
    - adding (primary or secondary) new particles
    - ending particles
    - finishing events
  - delegates processing of particles history to concrete MC truth implementation

MC truth infrastructure and users code

- MCTruthMgr provides interface and underlying infrastructure for particles history
  - light-weight transient, intermediate event record
- users code:
  - decision making (filtering) algorithm
  - conversion to users' event format
- concrete example implementation provided based on HepMC3

MC truth call sequence



MC truth output status

- GeantV MC truth manager provides handles to deal with particles history
  - allows ‘physics’ studies
  - first implementation, further iterations possible to look in detail at performance
- example implementation based on HepMC3 provided
- further performance testing/improvements in highly concurrent environment to be studied

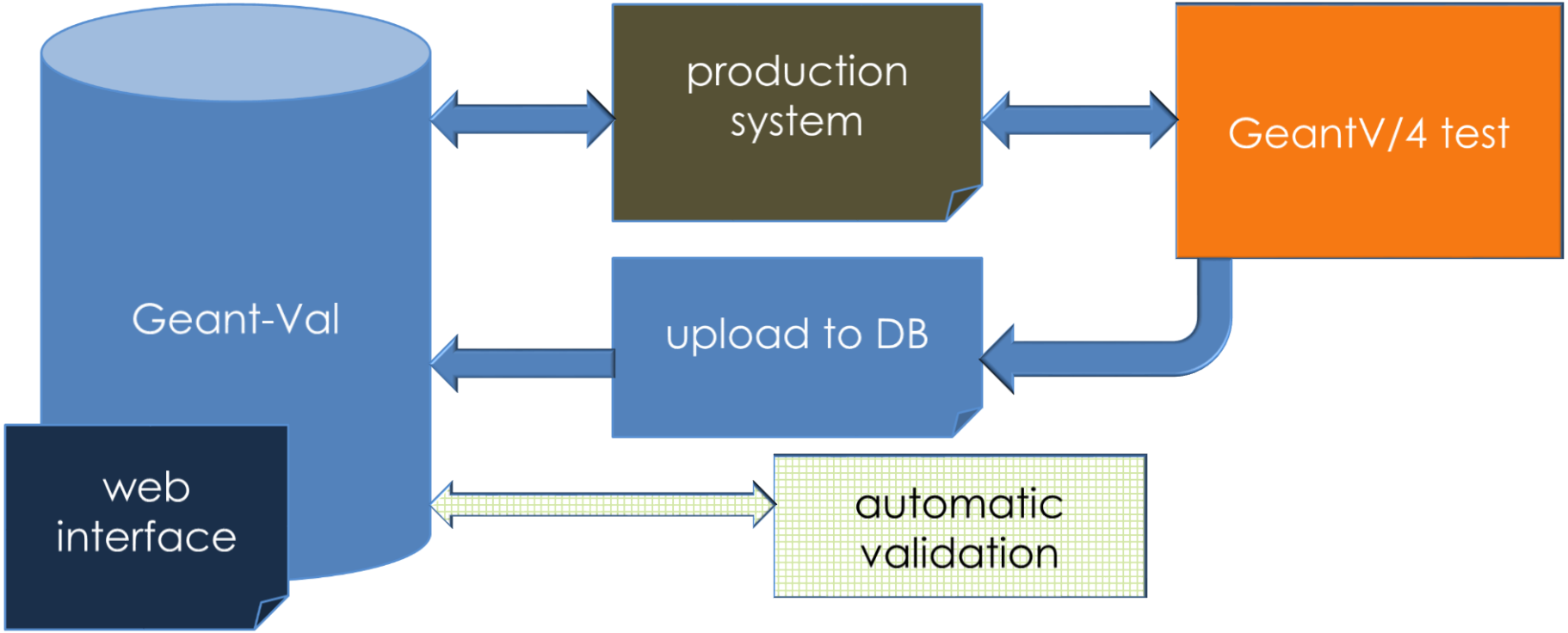
Physics Validation Repository

- provide repository for tests results allowing regression testing
  - physics tests for any new version of GeantV compared to the previous version
- provide repository for experimental data allowing physics validation
  - ‘physics’ tests should have corresponding experimental data in a central repository

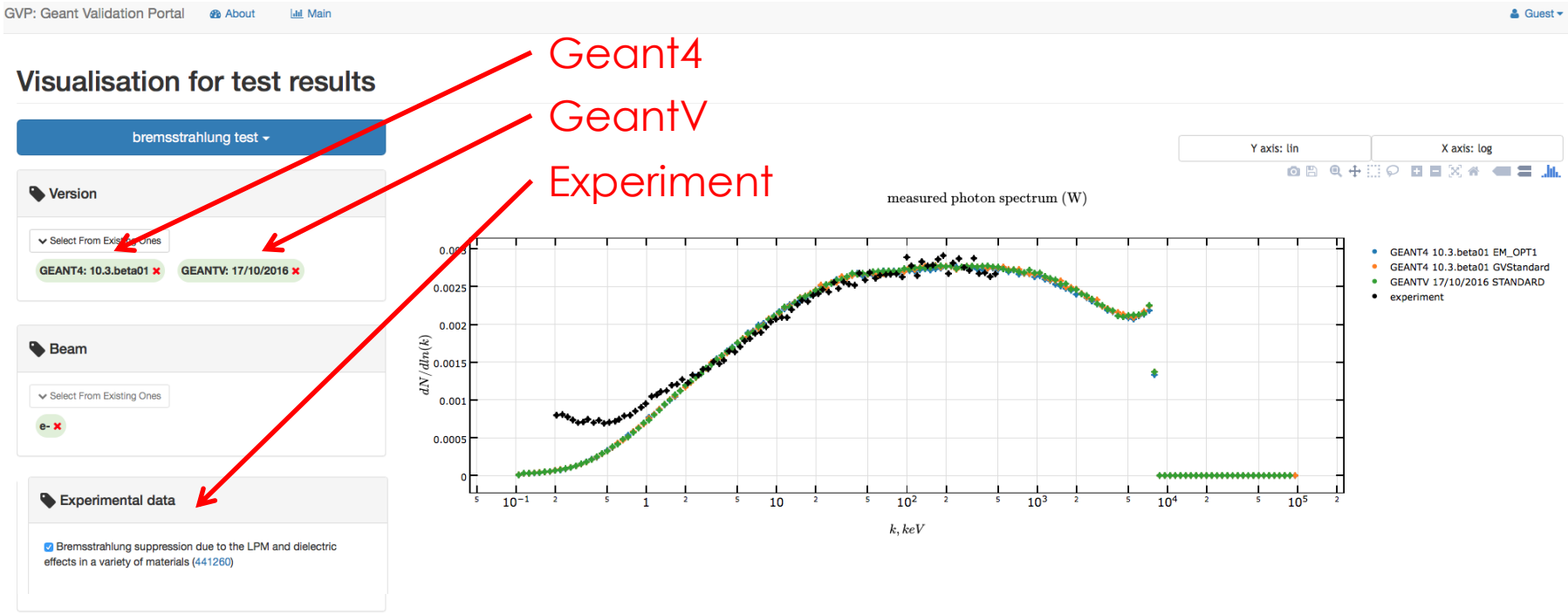
Geant Physics Validation approach

- collaboration with FNAL
- access via web services
  - efficient interface allowing interactive comparison of plots
  - interface to production system allowing automatic production and uploading of results
- extending the functionality to GeantV (in addition to Geant4)
- adding more tests

Validation Data “flow”



Geant Validation Portal



Conclusion

- interface to HepMC3 for particles input available
- users hits persistency possible with memory file merging
- lightweight MC truth interface available allowing to study particle history according to users' specific selections
- Geant Validation service provides the functionality needed for regression testing and physics validation

Backup slides

Physics Validation service architecture