GeantV – I/O, MCTruth and Validation Database

Witek Pokorski (CERN) for the GeantV development team

G.Amadio (UNESP), Ananya (CERN), J.Apostolakis (CERN) , A.Arora (CERN), M.Bandieramonte (CERN), A.Bhattacharyya (BARC), C.Bianchini (UNESP), R.Brun (CERN), Ph.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), H.Kumawat (BARC), 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)

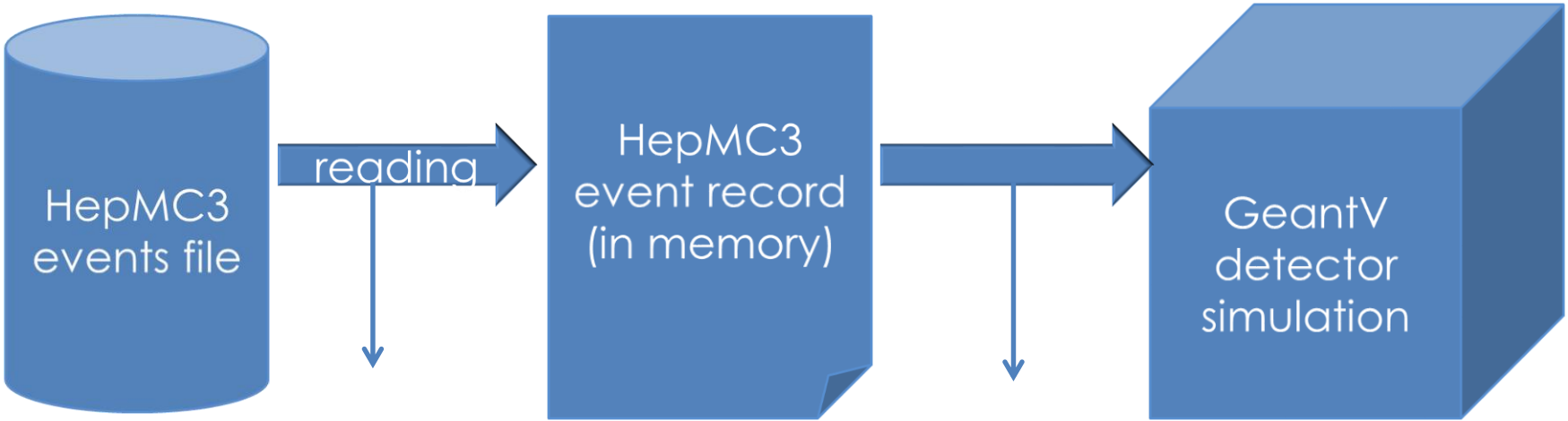
Outline

- GeantV input
- GeantV output
  - hits
  - kinematics
- Physics validation DB
- 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

HepMC3 as GeantV input format



0.6ms / 14TeV pp signal event (negligible compared to simulation step)

Particles fed into concurrent simulation threads

GeantV input implementation

- interface implemented in HepMCGenerator class
  - depends (of course) on HepMC3
- can read HepMC3 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
- nothing pending for short term development

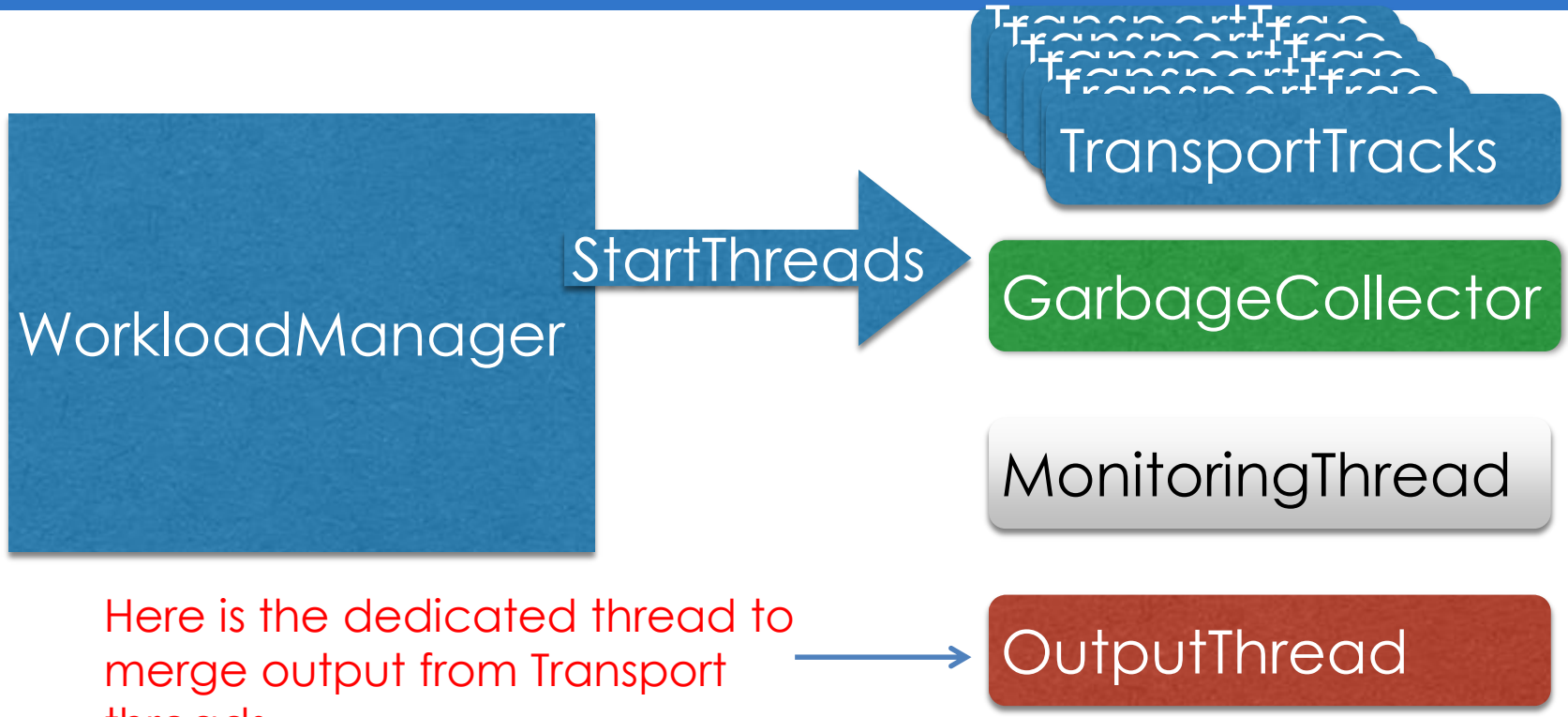
GeantV Output

- 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
- hits from concurrent transport threads merged by the output thread

Threads started by WorkloadManager



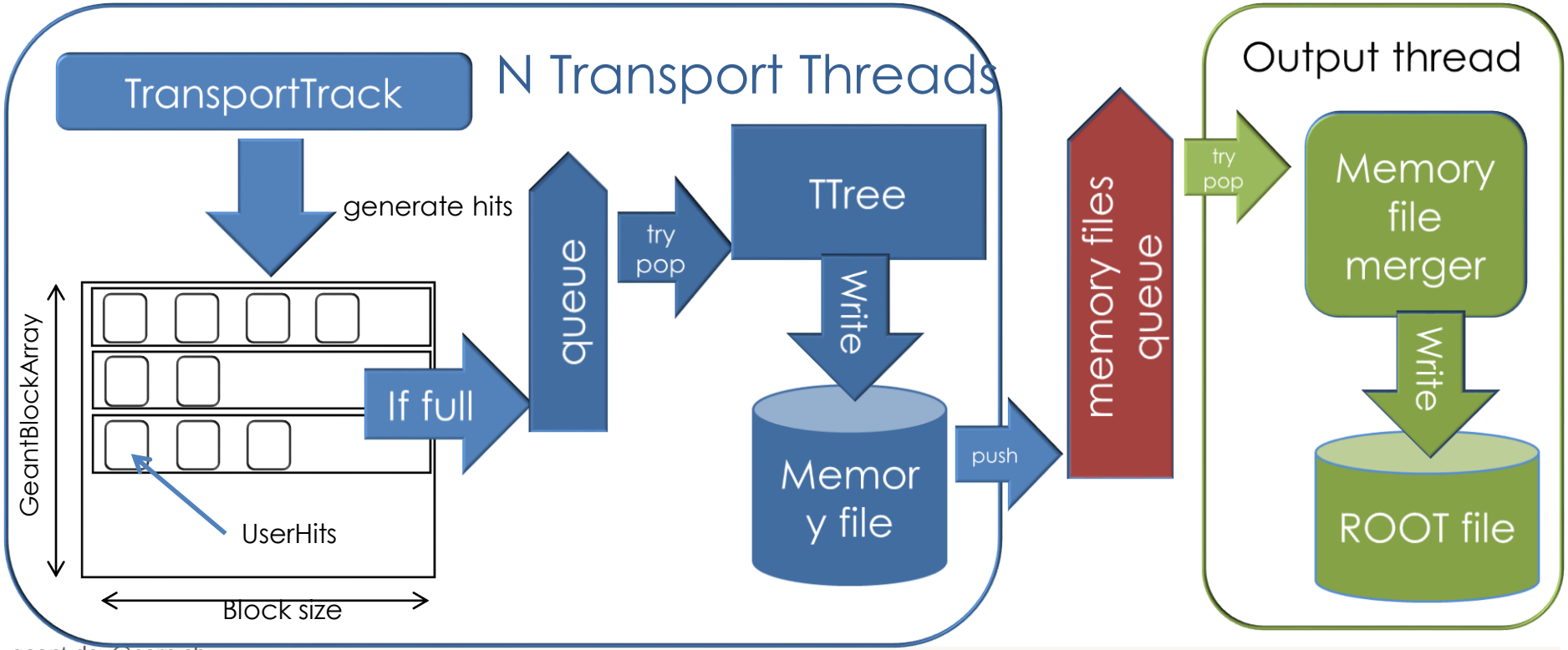
GeantV output thread

- (several) TransportTracks threads generate hits
  - GeantFactory machinery takes care of grouping the hits in HitBlocks and putting them in a queue
- Output related functionality implemented within OutputThread
  - first approach where serialization done within (one) output thread was creating a bottleneck
  - performing serialization within each transport thread solves the problem

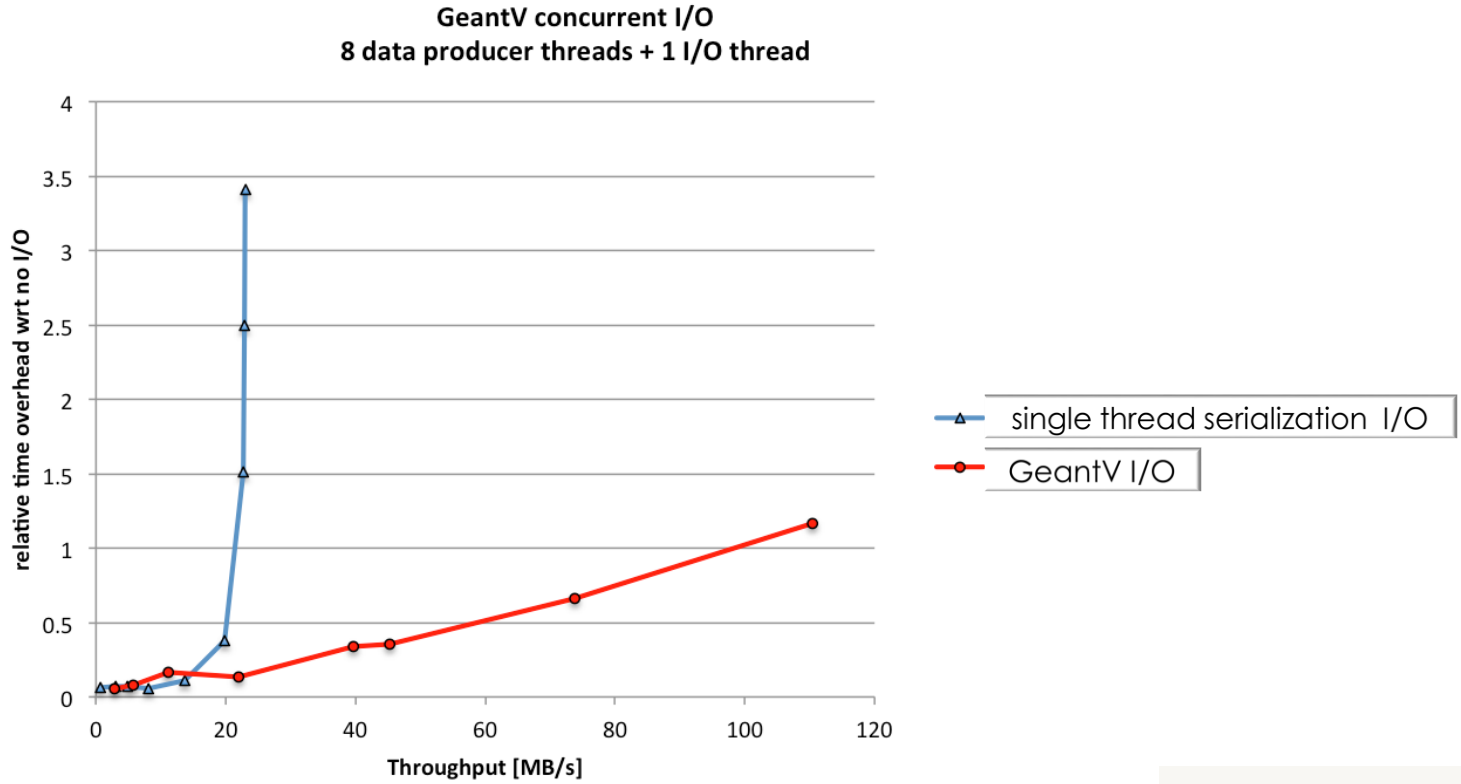
GeantV output implementation

- 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



Hits Output status

- (optional) hits persistency available in GeantV
  - further improvements (reordering of hits in file, etc) possible at later stage
- tested in multithread environment with several concurrent transport threads

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 want) to store all particles
  - no delta-e, no gamma showers, etc
- we need to store particles necessary to understand the given event (process)
- we need to store particles to associate hits
- 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 have
  - 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
  - we can start with HepMC

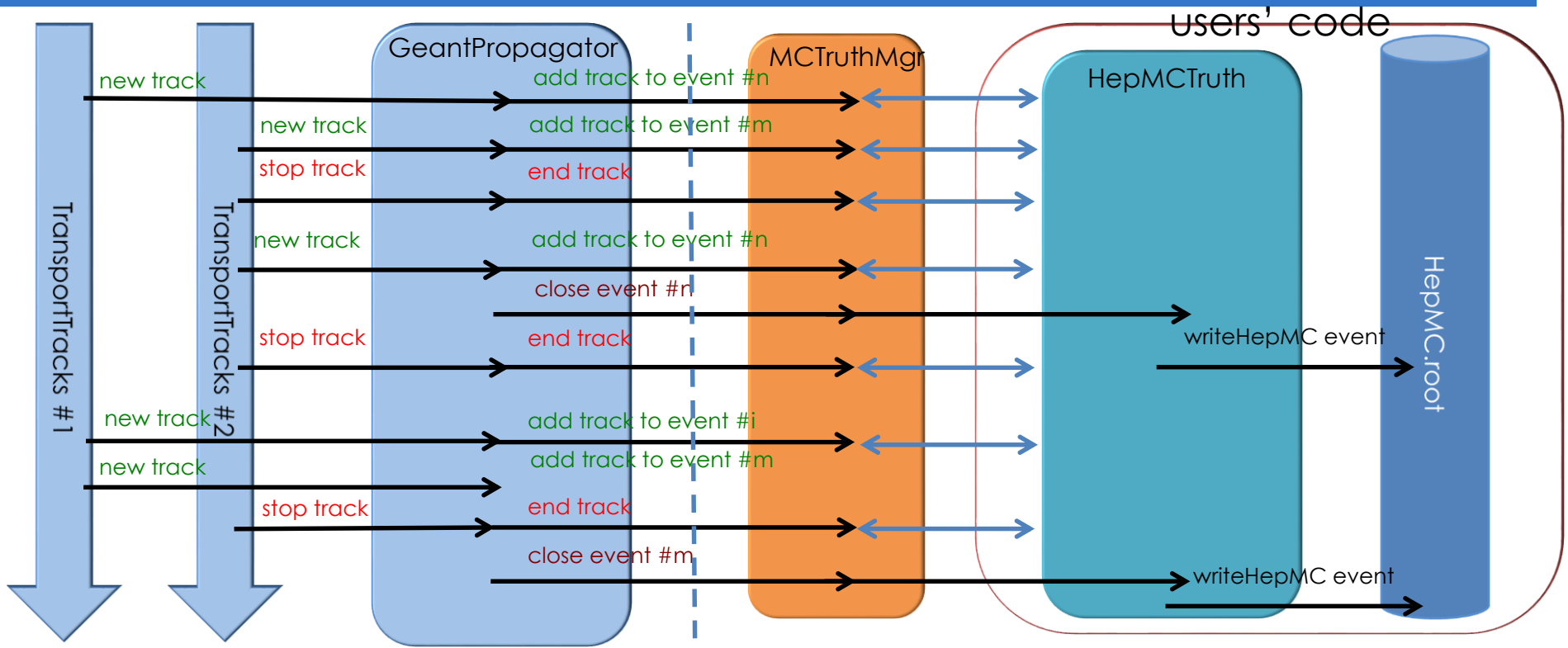
MC truth handling architecture

- light coupling to transport
  - minimal ‘disturbance’ to transport threads
  - maximal flexibility of implementing custom particle history handlers
- lightweight interface provided by MCTruthMgr
  - receives (concurrent) calls from transport threads to
    - add particles
    - end particles
    - finish 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

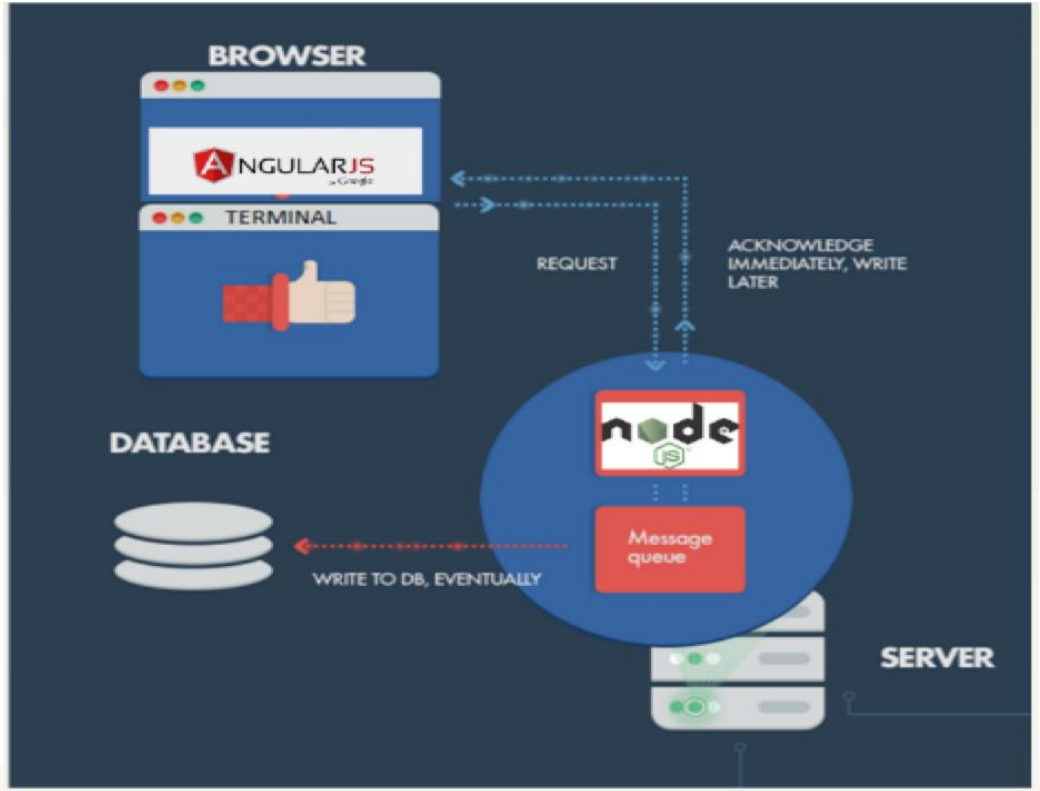
Physics Validation Database

- 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
  - no ‘physics’ tests should be run without corresponding experimental data in the validation DB

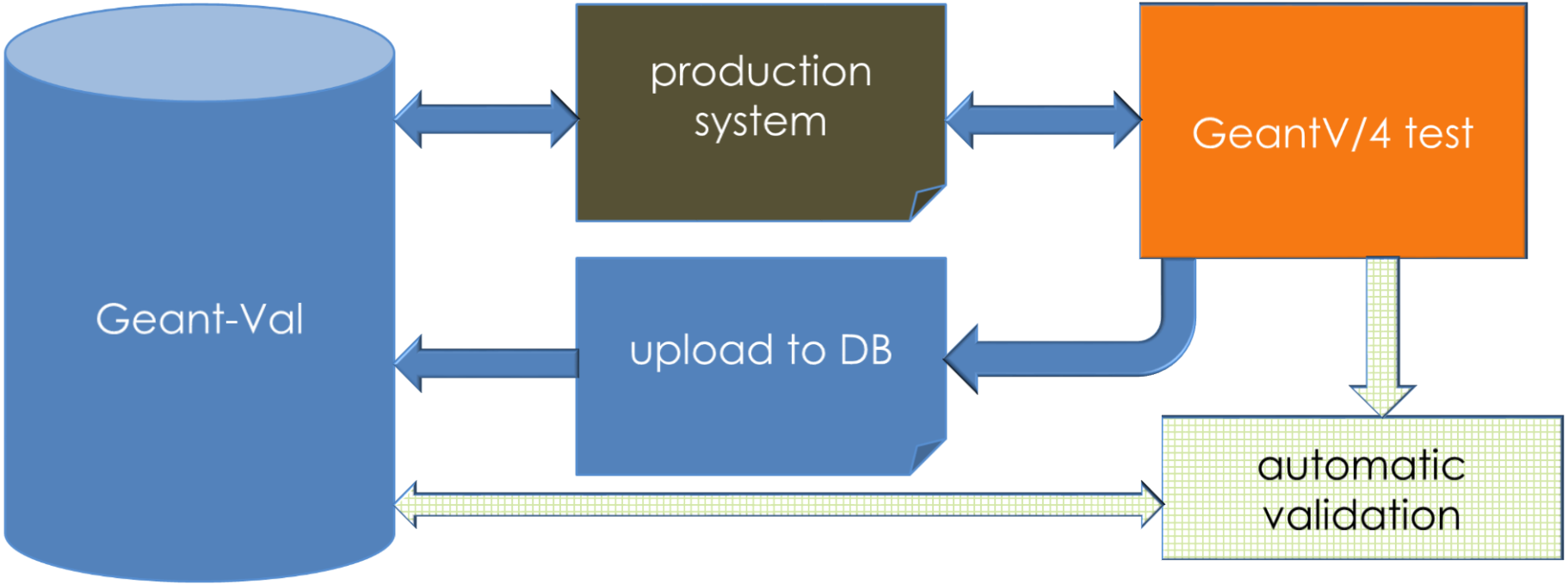
Physics Validation DB approach

- next generation of g4-val service (implemented in 2014, based on django framework)
  - efficient web 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
- modernizing architecture and implementation

Physics Validation DB architecture



Validation Data flow



Geant Validation Portal

Visualisation for test results

test X

Version

Select From Existing Ones

GEANT4: 10.3.beta01 ✖ GEANTV: 17/10/2016 ✖

Beam

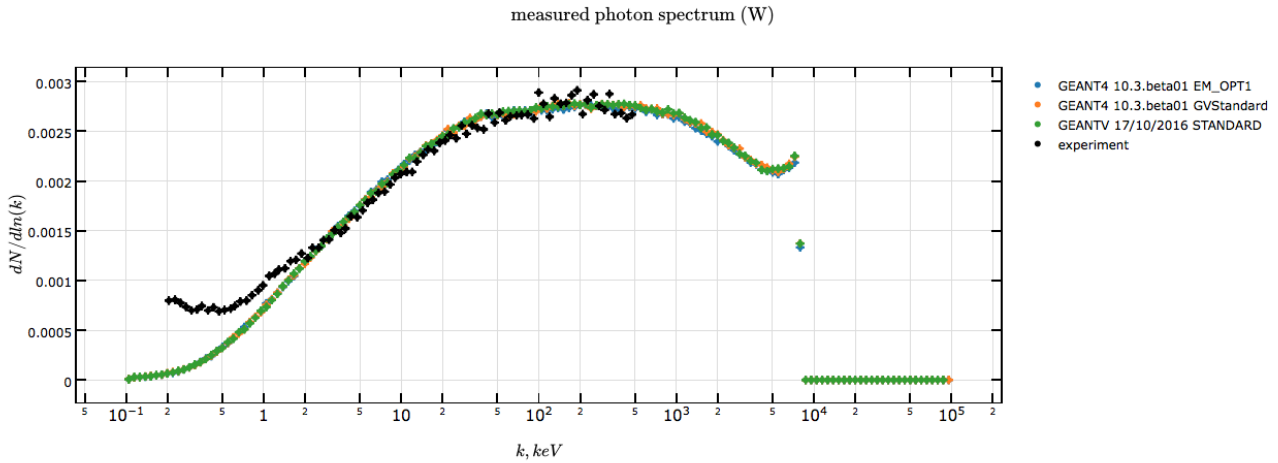
Select From Existing Ones

e- ✖

Physics List

Select From Existing Ones

X axis: log



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 repository provides the functionality needed for regression testing and physics validation