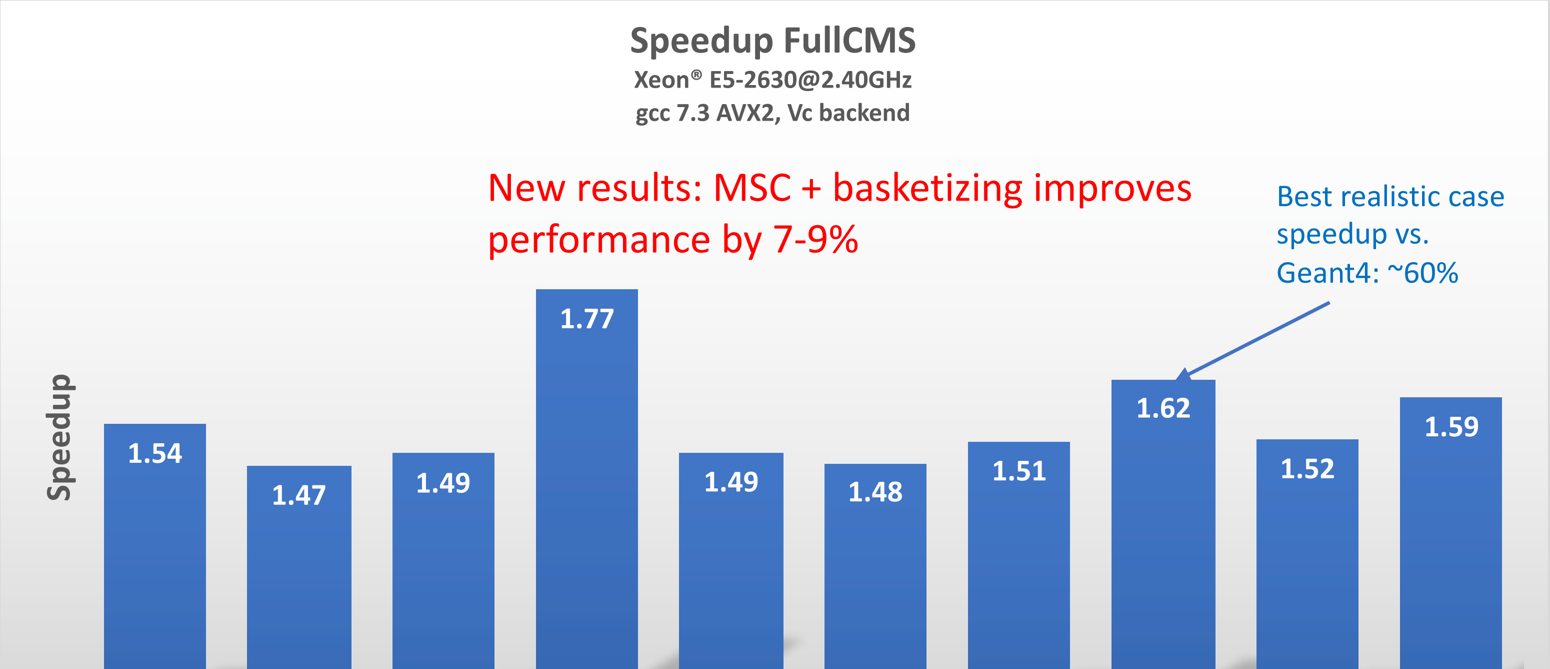
GeantV open issues & work items

A. Gheata

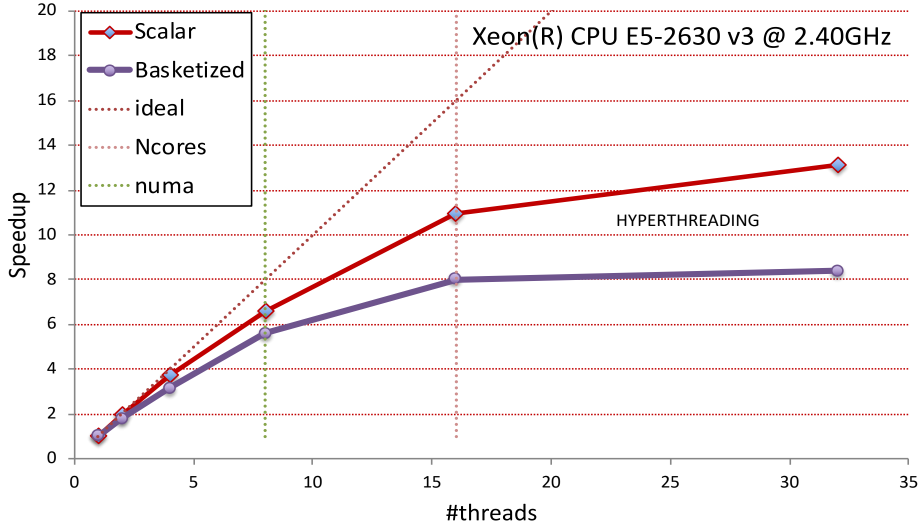
Performance problems

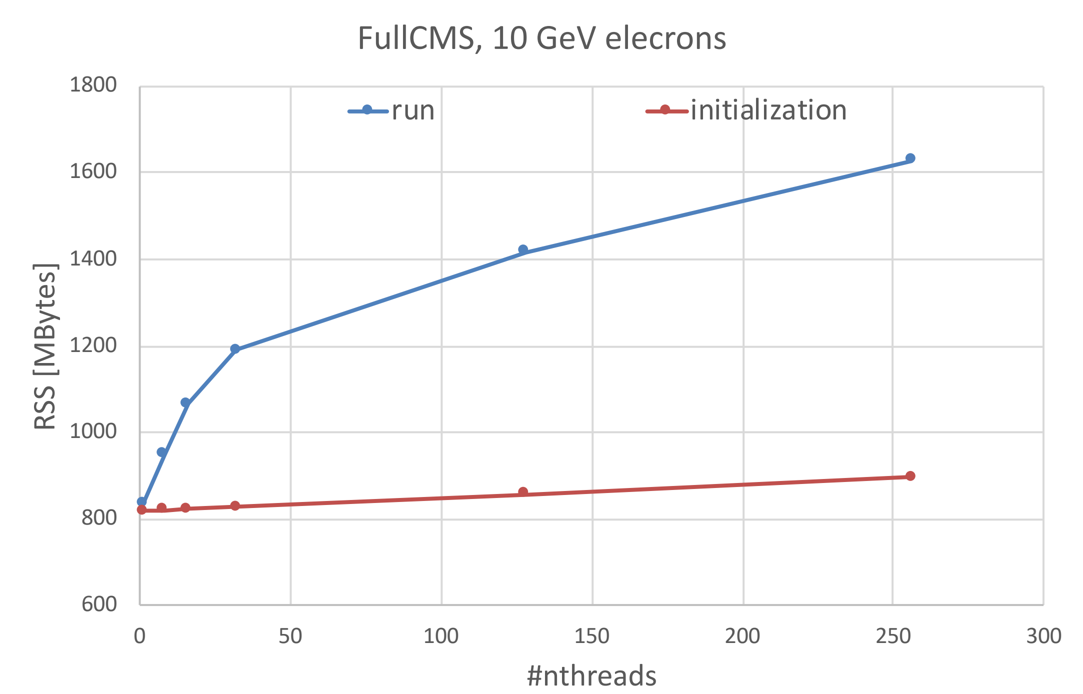
- Basketizing overhead
  - Basketizing is the service that concurrently groups particles undergoing the same processing phase, controlled by handlers
  - Overheads
    - single thread ~1.5% for magnetic field, ~4.5% for physics post step actions,~4% in geometry
    - MT: loss in scalability (Amdahl) & loss in basket efficiency due to need to flush more oftenly
- Vectorization gains: ~9% mag. Field, ~5-9% MSC, -4% geometry, -4% physics for CMS setup
  - Geometry is a real issue, has large scalar bottlenecks for relocation and computation of global transformations
- Scalability in MT: max speed-up of ~13 on a 16 core machine (even over-subscribing)
  - Price to pay for fine-grain track level parallelism
- Memory footprint
  - 3x the memory required by Geant4, but behaving good with #workers

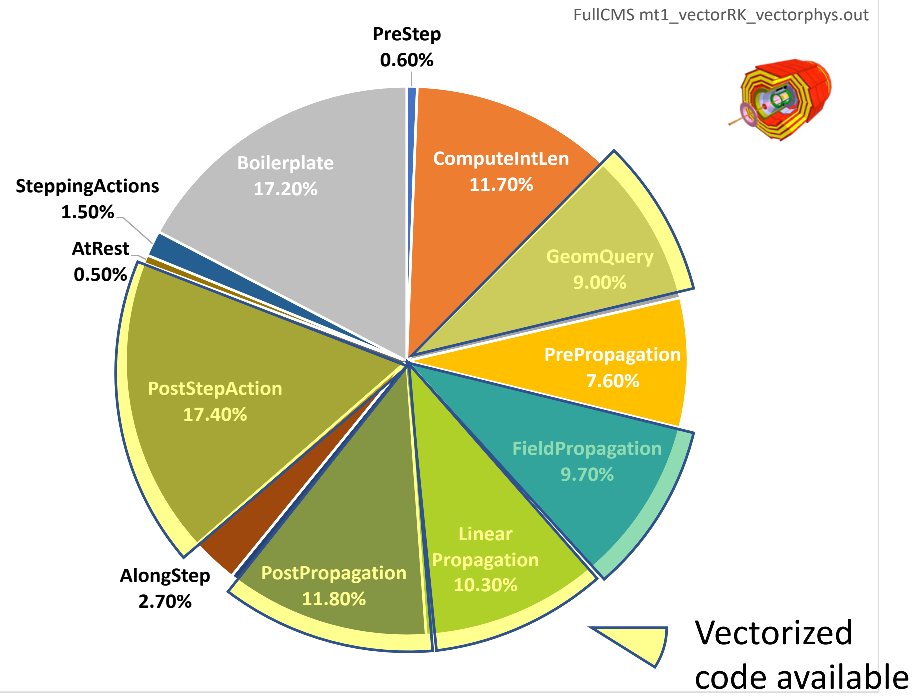


New results: MSC + basketizing improves performance by 7-9%

Best realistic case speedup vs. Geant4: ~60%







Vectorized code available

Work items

- Performance monitoring: activate nightly benchmarking suite
  - Scripts ~ready, need to work on the setup, find an appropriate machine
- Lightweight basketizing + scheduling
  - Hot basketizers can work per thread (enough tracks)
  - Investigating a mode where all basketizers are per thread
    - No track exchange between workers (lower grain parallelism)
- Geometry improvements
  - Activating topology-aware navigators
  - Better matrix caching
- Reproducibility
  - Add RNG state pointer in track and management of RNG pool (1 pointer/track + N states alive – as many as tracks)

Work items (cont)

- Vectorization of interaction length sampling stage
  - A lot of indexed accesses, ~12% slice to optimize
- Magnetic field
  - Activate/test RK vectorized + field map
- Error recovery
  - Bookkeeping state of simulation (events completed, which worker processing what)
  - Graceful shutdown: monitoring/controlling process/thread detecting error conditions and controlling shutdown scenarios
    - Sending shutdown signals to workers, propagated to user code via dedicated API