Run/Event/Detector Updates

A.Dotti, M.Asai (SLAC)

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Run Category Update

SLAC

As of release 10.1:

- memory consumption has been reduced in MT for more than a factor of 2 as planned
- new UI command: /run/useMaximumLogicalCores
- use of workspaces when available (improve code of thread initialization)
- option to seed workers only once per run (needed for performances for large number of small events)

For 10.2:

- some MT-related improvements:
  1. removed explicit initialization of Bertini in run-manager
  2. added possibility to create a non-worker thread (e.g. visualization thread)
- added integration with new fully-MT compliant visualization
- allow setting affinity of worker-threads to cpu cores to increase CPU performances (only available for linux)
- increase granularity of virtual methods in G4[Worker|MT]RunManager for event loop: ease integration w/ TBB framework

Event Category Update

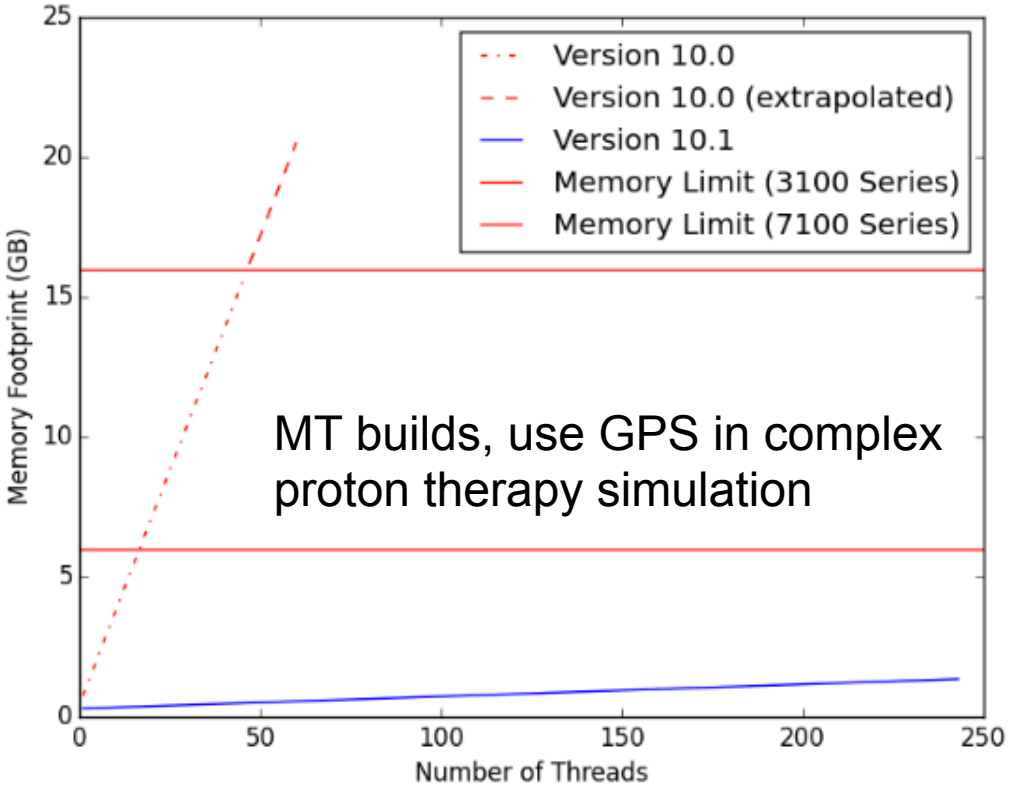
SLAC

General Particle Source migration to MT has been concluded

- very large memory reduction achieved sharing data structures among threads

Reviewed UI commands: some non-logical assumption found and fixed

- propose to remove deprecated commands in 10.2 (not done up to now due to MT migration)



Detector Category Update

SLAC

- New requirement from ATLAS: attach more than one SensitiveDetector to a single LogicalVolume
  - needed in special "calibration hits" runs
  - requires to have zero overhead in "normal" runs
- Introduced a new class G4MultiSensitiveDetector: a collection of G4SensitiveDetectors, calls to API are forwarded to all associated SDs
- Updated G4VUserDetectorConstruction::
  SetSensitiveDetector(G4LogicalVolume* , G4SensitiveDetector*)
  - when setting first SD, nothing changed
  - if a second SD is attached to a LV, trigger mechanism of MultiSD:
    1. Create a G4MultiSD
    2. Move existing SD as the first registrant of G4MultiSD
    3. Add new SD to G4MultiSD
    4. Attach G4MultiSD to LV

Moved to 2016

- Use std::thread concurrency from C++11 instead of pthread
  - motivation: Windows support in next version of VS
- C++11 support for Intel Compiler has turned out to be more problematic than expected (limited support for Xeon Phi)
- Decided to reschedule task to 2016