Reproducibility Tests

Alberto Ribon
CERN EP/SFT

Geant4 Collaboration meeting, 21 September 2020

Reproducibility in Geant4

- Work started in 2012, and in G4 **9.6** we achieved for the first time the sequential reproducibility for FTFP_BERT
- Leveraging on the sequential reproducibility, the next major achievement was the MT *vs.* SEQ reproducibility in G4 **10.0**
  - For NeutronHP, it MT *vs.* SEQ reproducibility was fixed in G4 **10.1**
  - For De-excitation and Radioactive Decay intermittent violations of reproducibility, finally fixed in G4 **10.4**
  - For INCLXX, intermittent violations of reproducibility, fixed in G4 **10.5**
- Reproducibility is tested at each reference tag and public release
  - In the recent past, once or twice per year we need to investigate and debug reproducibility violations

“Weak” and “Strong” Reproducibility

- Weak reproducibility :
  - Executing twice the same Geant4 application, starting with the same random engine status, we get the same random number sequence
    - Using: ***/random/resetEngineFrom start.rndm*** # In all cases
- Strong reproducibility :
  - We execute a Geant4 application with **N** events – long run – saving the random engine status at the beginning of each event; then we execute the same application for **1** event – short run – setting the random engine status of the ***k**-th* short run as the saved one at the beginning of the ***k**-th* event of the long run;
    we check that the random sequence is the same, for ***k = 1 ,…, N***
    - Using: ***/random/setSavingFlag 1*** # In all cases
      /random/saveEachEventFlag 1 # For MT

SimplifiedCalo

- For the reproducibility tests, we use the SimplifiedCalo application
  - Hadronic showers in simplified calorimeters
    - 1000 events for 5 configurations: 1) 20 GeV K0L on Fe-Sci
      2) 20 GeV pi- on Cu-LAr ; 3) 20 GeV K- on PbWO4
      4) 20 GeV p on W-LAr ; 5) 20 GeV n on Pb-LAr
- At the end of the event (*i.e.* of an hadronic shower), we print out a flat random number
  - Reproducibility means that this end-of-the-event random number is the same in the two cases that we compare
    - E.g. the *k-th* event of a long-run *vs.*
      the single event of a short-run with random engine set to be the same as the one at the beginning of the *k-th* event of the long run

Example (1/2)

/random/resetEngineFrom start.rndm
/random/setSavingFlag 1
/random/saveEachEventFlag 1
…
/gun/particle pi-
/gun/energy 20 GeV
/mydet/absorberMaterial Copper
/mydet/activeMaterial LiquidArgon
…
/run/beamOn 1000

Long run in MT mode

=> Producing in output the files:

G4Worker3_run0evt0.rndm , G4Worker3_run0evt1.rndm , …
G4Worker5_run0evt23.rndm , … , G4Worker7_run0evt999.rndm

Example (2/2)

/random/resetEngineFrom G4Worker5_run0evt23.rndm

…

/gun/particle pi-

/gun/energy 20 GeV

/mydet/absorberMaterial Copper

/mydet/activeMaterial LiquidArgon

…

/run/beamOn 1

Short run in SEQ mode

=> From the log file of this short run:

--- EndOfEventAction --- event= 0 ***random=0.7805***

To be compared with this line of the output of the long run:

G4WT5 > --- EndOfEventAction --- event= 23 ***random=0.7805***

Planned Tests

- Long-run with one of the following:
  - export G4FORCE_RUN_MANAGER_TYPE=**MT**
  - export G4FORCE_RUN_MANAGER_TYPE=**Tasking**
  - export G4FORCE_RUN_MANAGER_TYPE=**TBB**
- And then short-runs with:
  - export G4FORCE_RUN_MANAGER_TYPE=**Serial**