Collective Computations in G4beamline

Tom Roberts, *Muons, Inc.*

- Collective beam computations, such as space charge, require the locations and momenta of all particles at each step (in time)
- This is parallel processing in a different context from threads
- I now have it working for a limited set of physics processes; single thread, no mutex-s
- In collective mode there has been a reduction in management classes:

  RunManager
  EventManager
  TrackManager
  StackManager

  

  BLRunManager
  (derives from G4RunManager)

- In collective mode, the BLRunManager loops over a std::vector<BLTrackData>, managing each track's status, setting the Navigator and ProcessManager for each track, and directly calling trackData->steppingManager->Stepping()

Every class that keeps an internal state used for tracking must have a per-track instance

- The ones I have identified (contents of BLTrackData) are:
  - G4Event
  - G4Track
  - G4SteppingManger
  - G4Navigator
  - G4ProcessManager
  - G4PropagatorInField
  - Every physics process used by the particle
- My approach is to:
  - Add a clone() function to those that need it
  - Let the normal startup initialize one set of them
  - Then clone() or copy them for each track

A Simpler Way

- After implementing this as parallel tracking, I realized there is a simpler way for this particular type of computation:
  - Add a process to suspend a track when it reaches the time of the next step (part of both implementations)
  - Start tracking as usual
  - Stack up suspended tracks in a special stack
  - When all tracks are suspended, do the collective computation
  - Then un-suspend all tracks in the stack, increment to the next time step, and track them all again
  - Stop when all tracks have been killed
- As this is rather easy to implement, and avoids changes to any Geant4 code (just needs a new RunManager), I intend to try it and see how important the extra overheads are
- That will provide a good check of the validity of both methods