Refactoring Geant4 stepping: a discussion

Andrei Gheata

Geant4 Task Force for R&D meeting

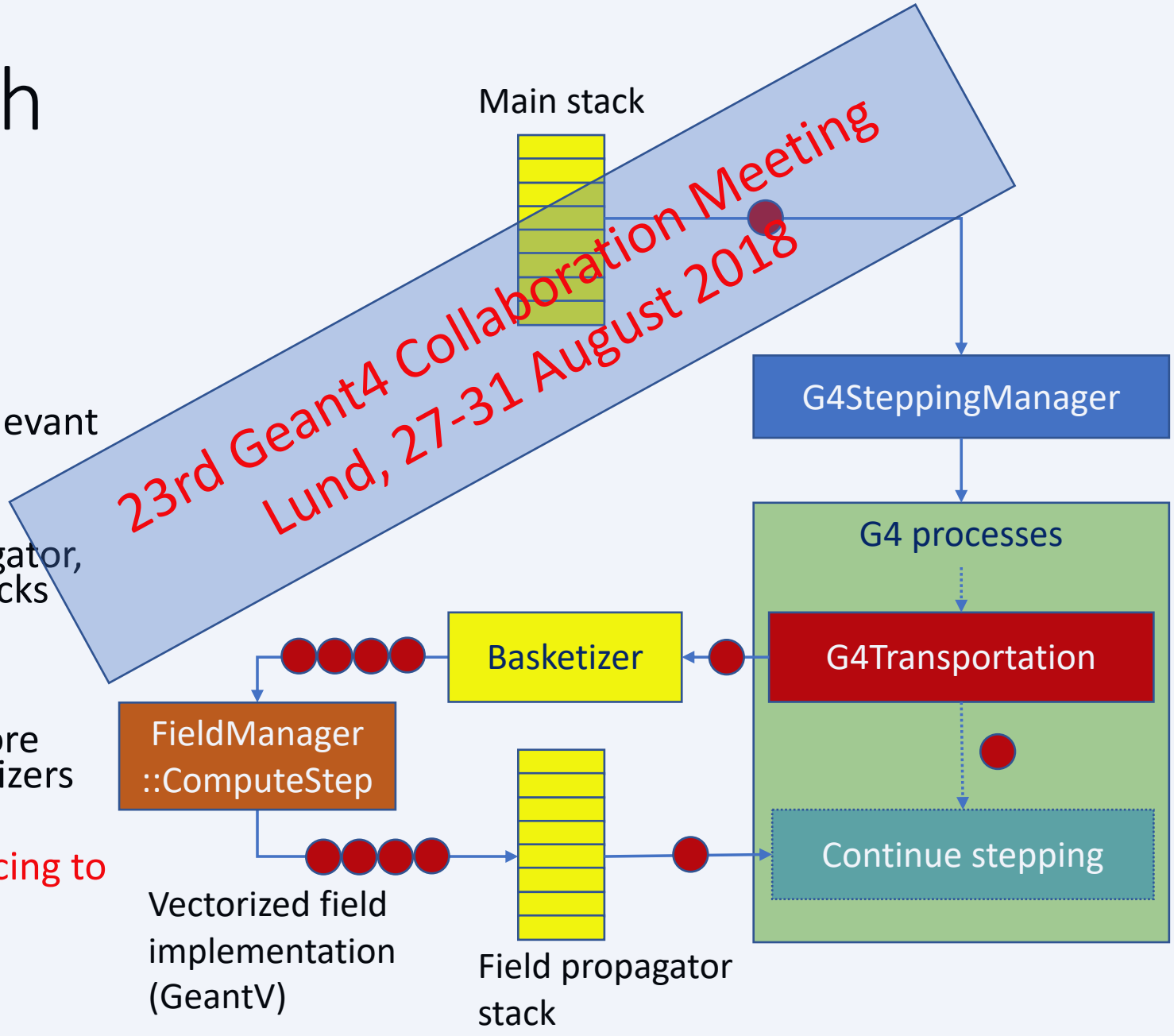
April 2, 2019

Rationale

- Instruction-level locality benefits proven by GeantV R&D for at least some important stepping hotspots
  - Magnetic field propagation, MSC, some “hot” physics models
- Usage of accelerators (like GPGPU) in simulation needs massive data and instruction level parallelism
  - Tracks doing the same thing…
  - Hard to achieve due to current G4 stepping sequencing
- The above require track-level parallelism ability
  - Exposing step-level parallelism for certain computation phases
- Is it possible?
  - Short-term R&D

Possible approach

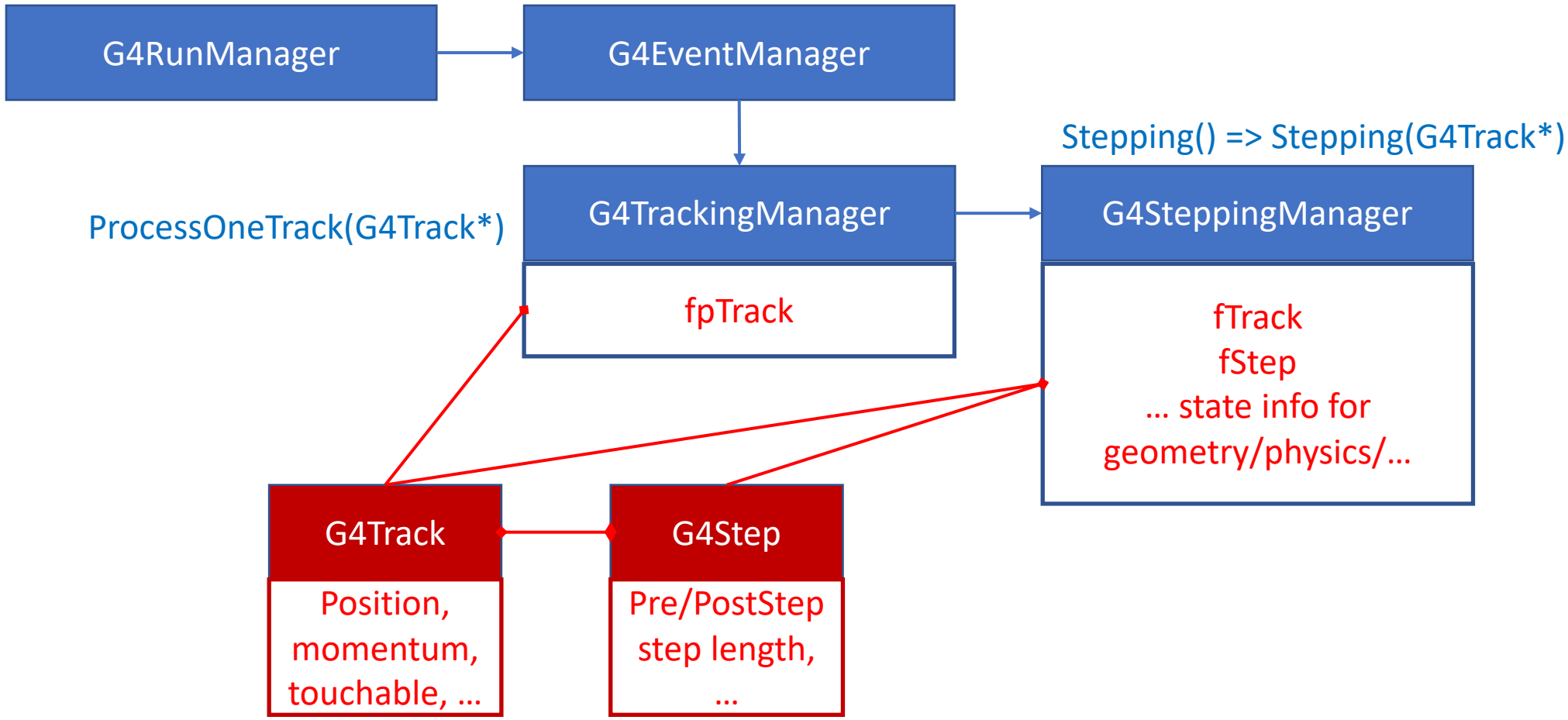
- Intercept charged tracks during transportation process
  - Basketize before FieldManager::ComputeStep
- After basketizer, copy track info relevant for field propagation to SOA
  - charge, position, momentum
- Dispatch to vectorized field propagator, then gather outputs to original tracks
- Stack tracks for further sequential processing
- Extension to MSC possible, but more complex, handling multiple basketizers needs GeantV-like scheduling
- User implications on track sequencing to be discussed



But how to pause/resume tracks with ~no overhead?

- Track/step state now embedded in the state of the stepping manager
  - Stepping interface advances the existing state
- Some mixture of the step state in physics (processes/models)
- Much lower or no mixture of step state in higher-level managers
  - Tracking manager, Event/Run managers
- Can we have a fully backward-compatible Geant4 (from user perspective) making the track/step state ‘volatile’ from managers
  - Handled explicitly in internal interfaces
- De-coupling state is a pre-requisite for any track/step-level parallelism attempt

Managers and state



Feasible? How long?

- Technically possible, but non-adiabatically
  - Requires synchronized changes in most managers
    - Working on a fork?
  - Proof of principle should not take more than a couple of months FTE
    - Changing signatures to pass externally track/step info
    - Cutting-out the state from managers/models
- User interfaces: no change in this phase
  - Also no expected performance benefits, just introducing a design that disentangles state from code

What after?

- Multiple specialized stacks straightforward
  - Grouping tracks by certain locality criteria
- Interrupt transport of a track and resume with no overhead
  - Gathering of “baskets” for some hotspots possible
- Sub-event parallelism possible below primary level
  - E.g. for use-cases where even a single primary can cause memory havoc
- Adopting track-level parallelism within and event will break the step sequencing
  - Some implications on user code
  - More important implications on reproducibility in MT/accelerator mode
- More concurrent events (owned by threads) needed to enhance locality
  - Brings more complexity to user code (need to manage tracks from mixed events)
  - A bit diferent PRNG approach for reproducibility (e.g. RNG state/engine per event)
    - We know how to do it…

Discussion

- Pro’s and con’s
- Technical issues
- Volunteers…