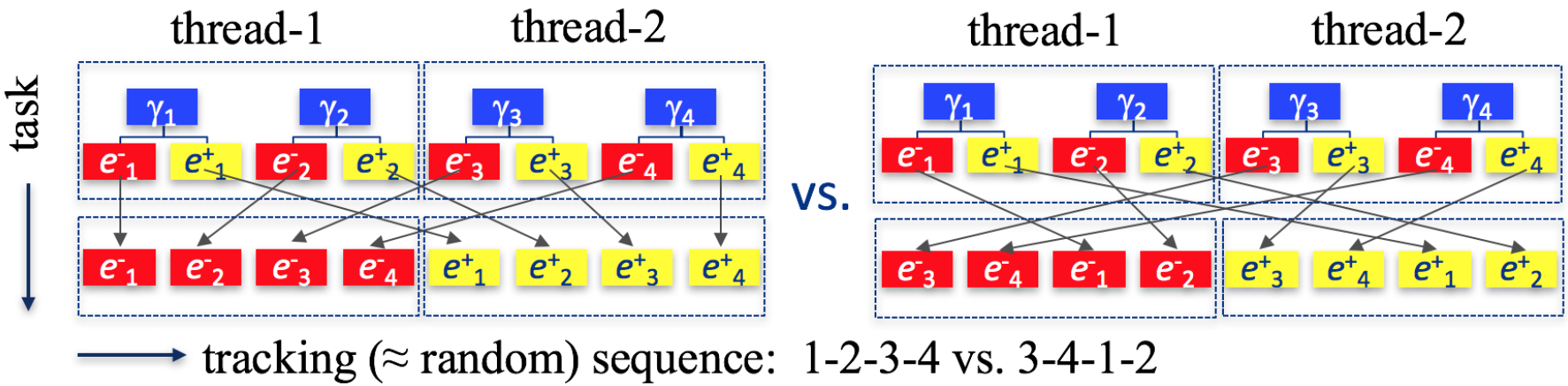
Reproducibility of concurrent particle transport simulation

Soon Yung Jun, Philippe Canal

Sept. 13, 2019
PDS Geant R&D Retreat

Introduction

- GeantV physics processes use scalar/vector pRNG under concurrent tasks - VecRNG provides parallel pseudo random number generation (pRNG) for both SIMD and SIMT workflows
- Reproducibility is a stringent requirement for HEP simulation
  - the execution order of threads is fragile and unpredictable (example: pair-production and tracking particles by the type)



  - how to maintain absolute reproducibility of random number sequences for concurrent simulation tasks?

Reproducibility under Concurrent Tasks

- Challenges for track-level parallelism under concurrent simulation work flows of GeantV
  - events are mixed and track processing order is not deterministic
  - maintain reproducibility between different modes and repeatability within the same mode
  - utilize vectorized pRNG efficiently keeping reproducibility
- Strategy: a track owns a pRNG state (or object)
  - generate output variate and update the given state in a thread-independent way

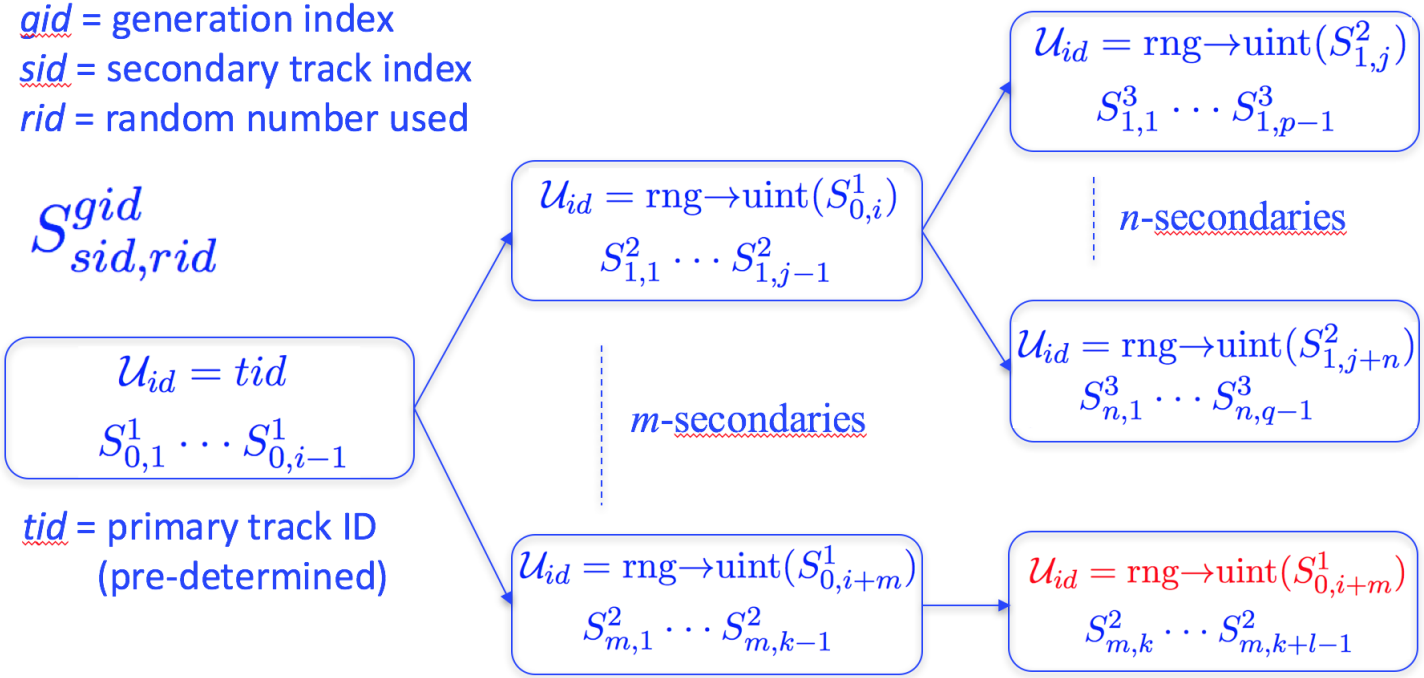
```
r1 = rng->Uniform(track.RngState());
```

  - assign a unique sequence (stream) to each track in a collision resistant way (example: initialize the random state of a new secondary track with the random state of the parent)

```
index = rng->UniformIndex(track.RngState());
secondaryTrack.InitializeState(index);
```

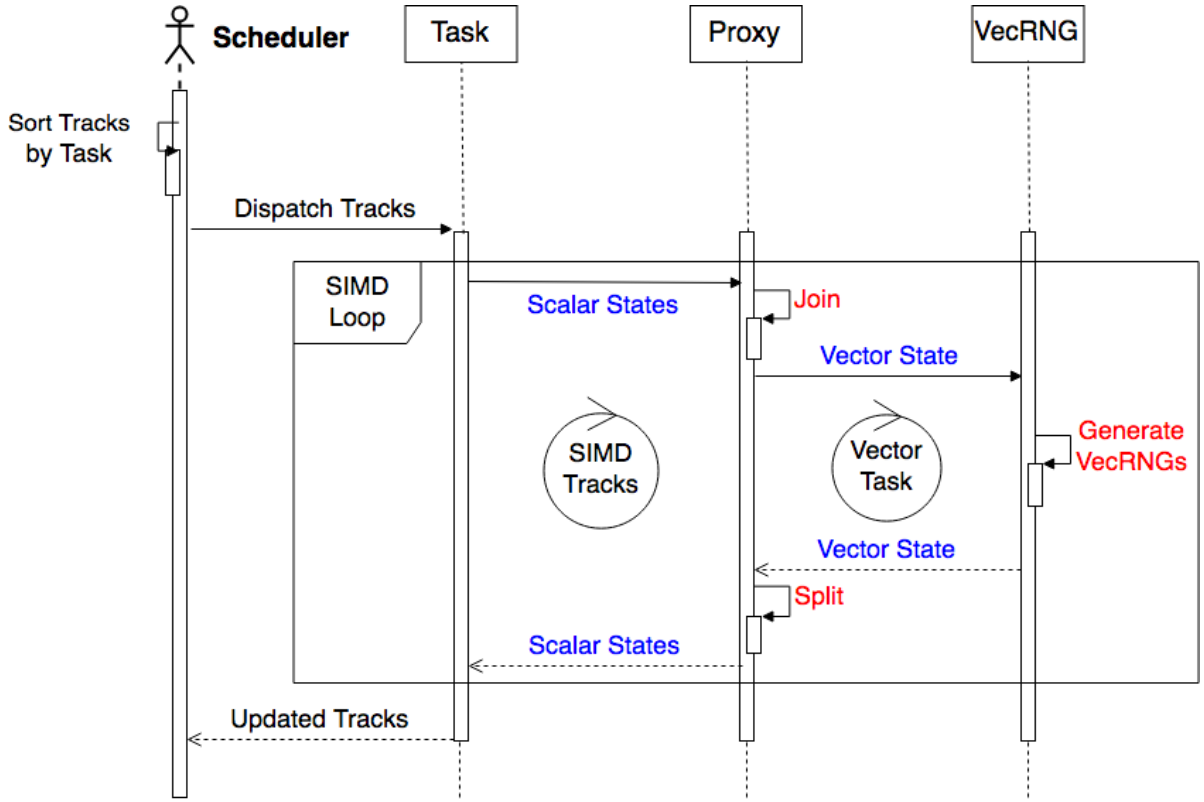
Reproducibility: Stream Assignment (A Solution)

- Reproducibility of the random forest of tracking trees under task-level parallelism: S= random state of the track
  - seed = run ID $\otimes$ event ID
  - unique stream ID, $\mathcal{U}_{id}$ = rng→UniformIndex(S) $\equiv$ rng→uint(S) (ex: output of Random123 is a collision-resistant AES/ARS hash)



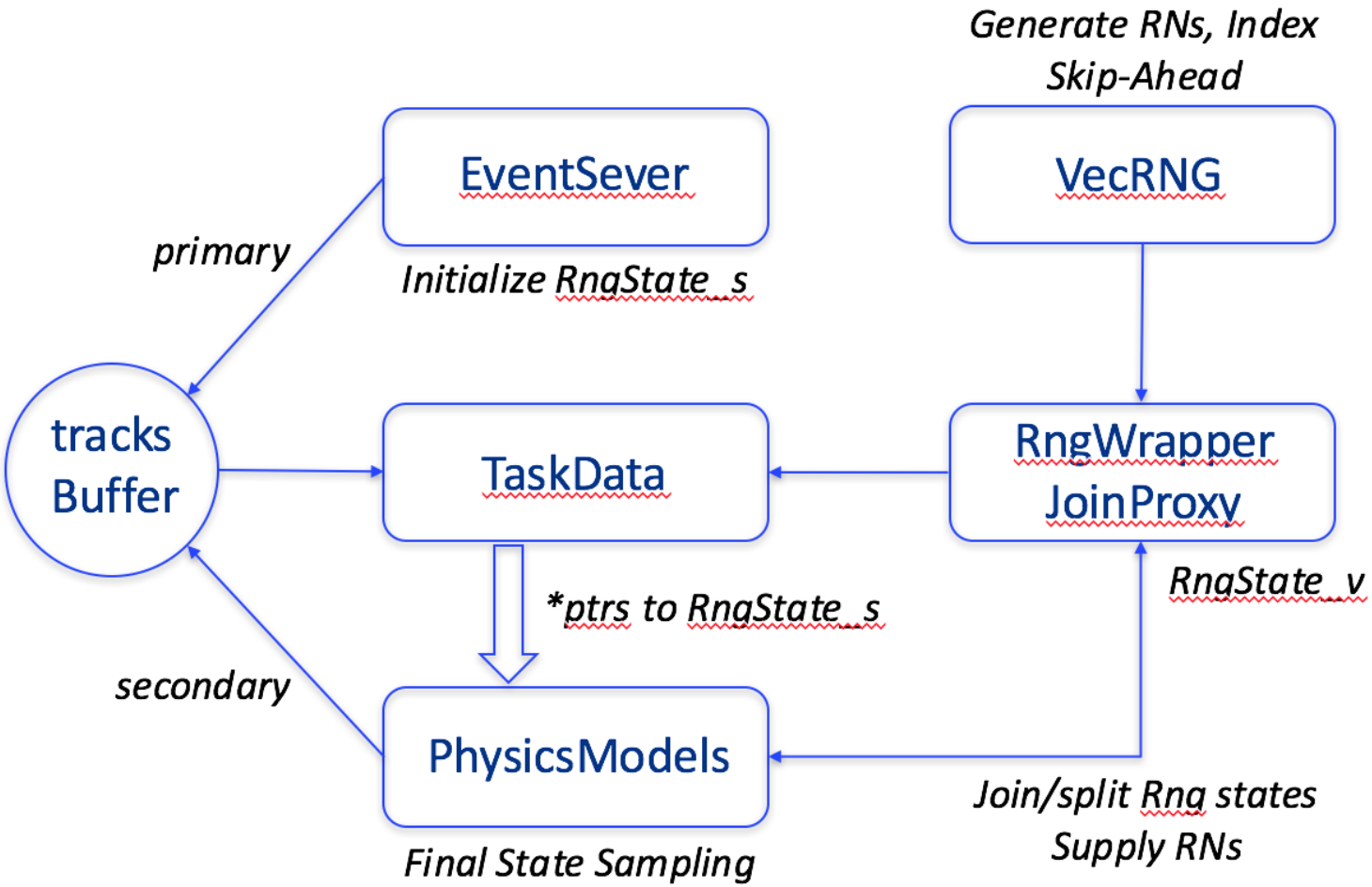
Reproducibility: Strategies for Vector Tasks

- Results of simulation should be reproducible (identical) between different modes (scalar vs. vector)
- Extension for deterministic vectorized sampling with vector rngs
  - Gather approach: generate scalar Rngs→ gather to a SIMD array
  - Proxy approach: join–states → generate VecRngs → split-state



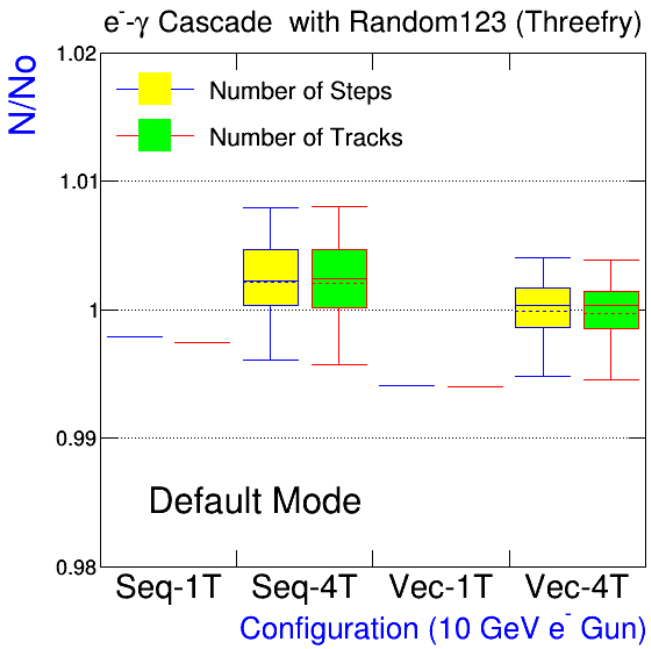
RngState workflow in GeantV

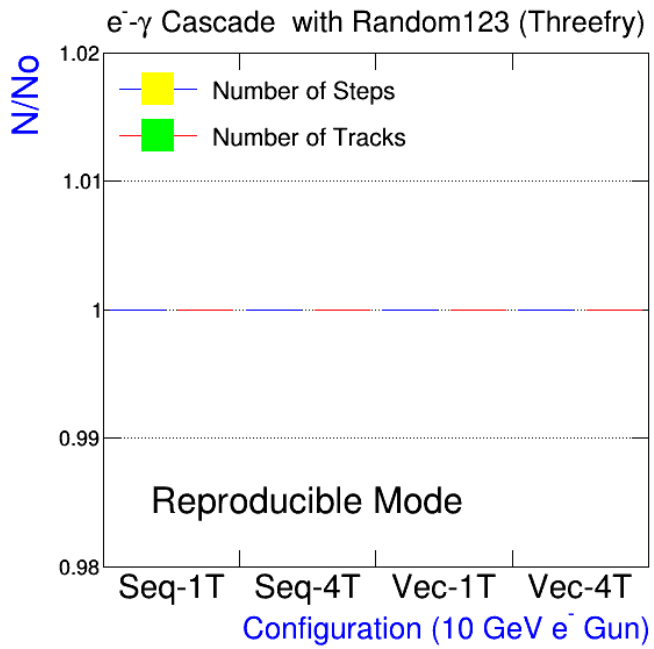
- A simplified picture of the RngState workflow for reproducibility



Reproducibility: Verification

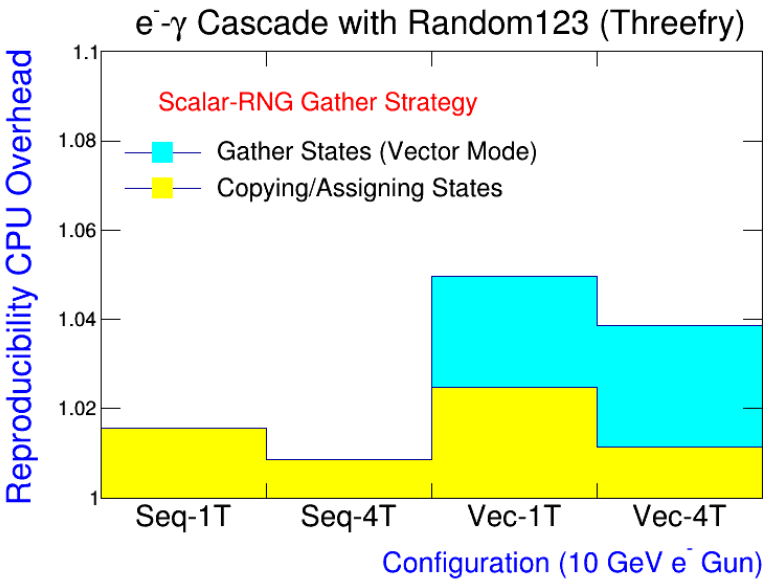
- Test setup with 10 GeV e^- with a subset of GeantV EM physics
  - passage through 50 layers of LAr-Pb calorimeter (TestEM3 [**?**])
  - $e^- - \gamma$ cascade (Bremsstrahlung, Ionisation and Compton)
  - 20 measurements (runs) of 1000 events, 10 e^-/event
  - configuration: (Sequential/Vector) $\otimes$ (1Thread/4Threads)
- The total number of steps (tracks) normalized to those of the first run of the reproducible Seq-1T mode, N_o ($\sim 5.6 \times 10^8$ steps)

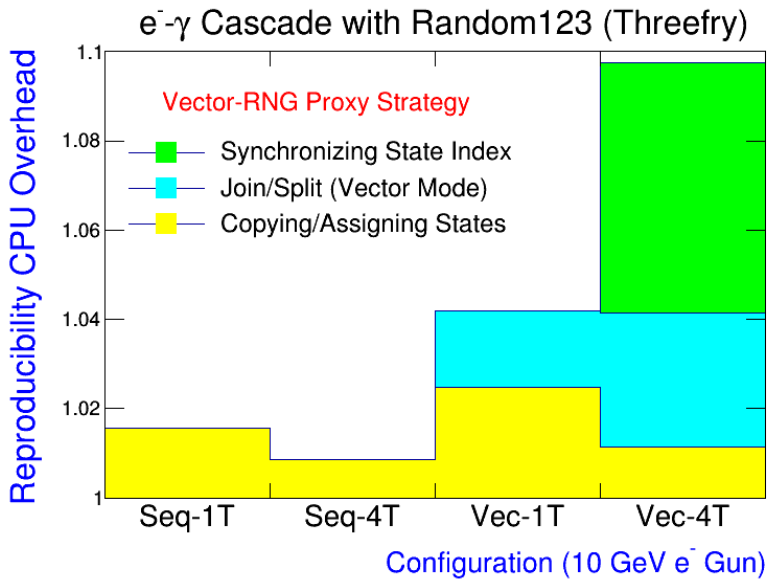




Reproducibility: Overhead

- Reproducibility introduces overhead due to
  - copying/assigning RNG states during workflows
  - gather RNGs to a SIMD array or join/split states
  - synchronizing the index of states in output (Random123 specific)
- Overhead: Time(Reproducibility)/Time(Fast Mode)





- the overhead of join/split of the proxy approach will be negligible if N(vector random numbers used) $\gg 1$ per join/split.

Summary

- Demonstrated reproducibility of propagating multiple particles in parallel in HEP event simulation with concurrent workflows.
  - For full reproducibility, it is important to develop common physics kernels for both scalar and vector tasks.
- The CPU overhead for full reproducibility is $\sim (5 - 10)\%$.
- Studied different strategies for efficient uses of vectorized pRNG.