RANDOM NUMBERS GENERATORS FOR GEANTV: NEEDS AND EMERGING OPTIONS

JOHN APOSTOLAKIS, CERN

Overview

- Particle Transport Monte Carlo & PRNGs
- Requirements for PRNGs
- Reproducibility & fine-grained parallelism
  - Extra requirements for PRNGs
- Options for PRNGs
- Summary

Particle Transport & PRNGs

- Almost every ‘decision’ requires the use a random number
  - Deciding which interaction occurs (e.g. absorption or scattering?)
  - Generating a secondary particle requires sampling values from probability distribution functions
- Random numbers are a **vital** part of particle transport Monte Carlo
  - Consume O(3%) of CPU time
  - Any serious error (bad seeding, wrong PRNG or other) would waste 10^4-10^6 US$ of CPU time!

Needs of Particle Transport

- Good statistical properties for the values
- Stream of reliable, portable random numbers are critical
  - Large period - 30 * 10^6 steps/event * 10^{10} events/year
  - Low correlation for the full sub-sequence of a stream
- Computing performance - is 3-10% of CPU time
    - but RANLUX @ highest ‘luxury’ can be > 10%
  - it matters - seek to make it < 2%, if possible
- Reproducibility/portability between operating systems & CPU arch.

What does GeantV mean for PRNGs?

- ‘Simple’ mode: each task/thread uses an ‘independent’ stream of the PRNG
  - Minimum concurrent (current configuration):
    - N_threads = O(100)
- Reproducible ‘mode’: each track has a PRNG state
  - the number of concurrent PRNG is much larger
  - N_PRNG = number of tracks in flight = 10^5-10^8

PRNG quality & affordability

- Some say “even lower-quality PRNGs could be **adequate** for particle transport” as there are many interactions, many decisions, …
- But using a **highest-available quality** PRNG is important:
  - correlations in the initial variates of a stream could lead to same results for the key first interactions & compromise quality of results
  - it is an **insurance policy** - if we can afford it.
- As Fred James said: “we should seek the best PRNG we can afford” !
- So we should seek a fast, mathematically-motivated ‘excellent’ PRNG

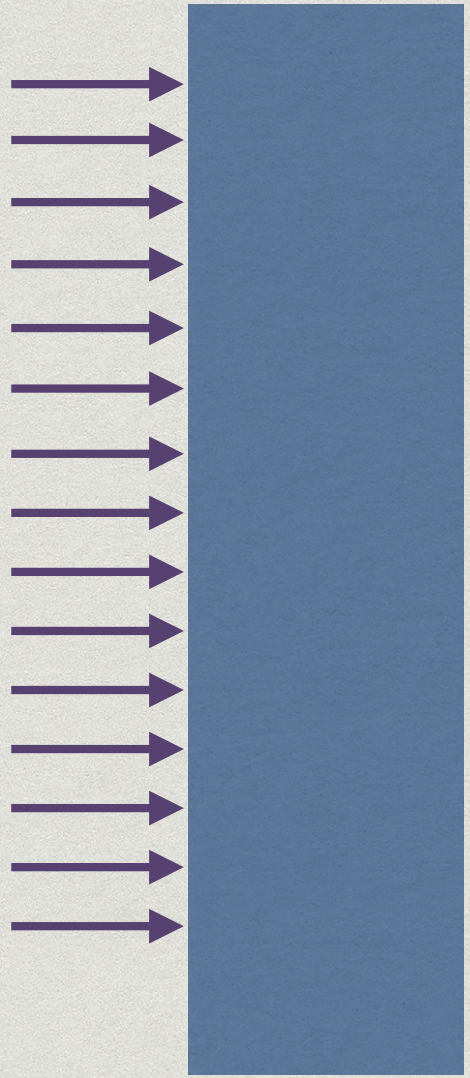
BEYOND A PRNG PER 'TASK'/THREAD

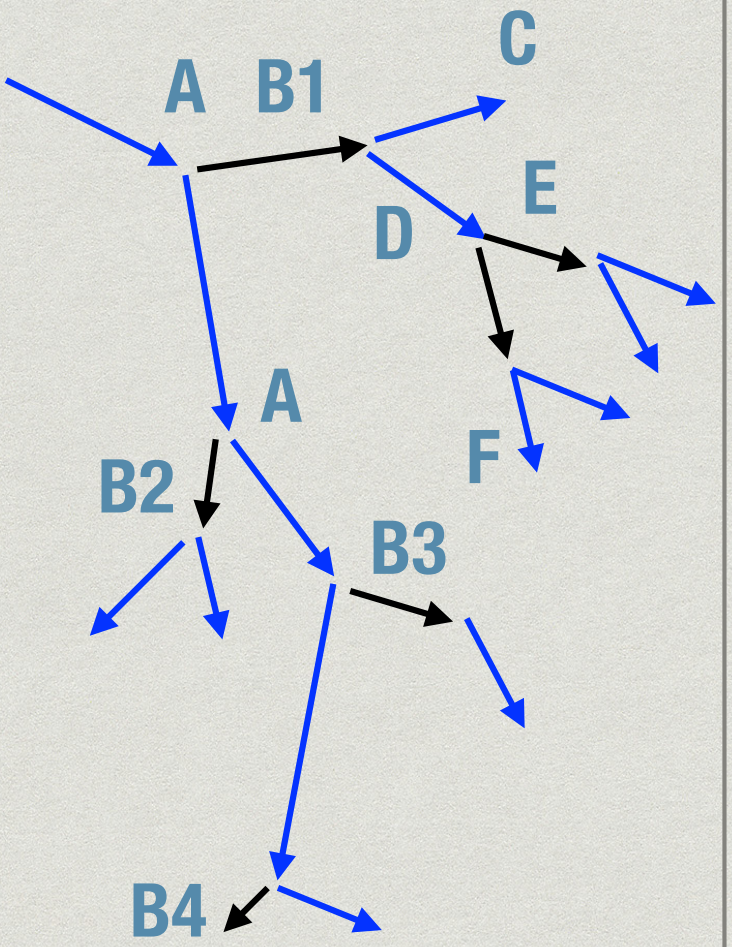
SEEDING FOR REPEATABLE TRACK-LEVEL PARALLELISM

Addressing reproducibility

- To meet **reproducibility** of simulation (without a fixed order of processing the tracks),
  - each track must use its own PRNG stream
  - a secondary particle must be given a unique, reproducible state (or seed) when it is created
  - $N_{streams} = N_{tracks\ in\ flight} = N_{baskets} * a_{average\_occupancy} \sim= 10^{6}\text{-}10^{8}$?
- Most tracks are low energy - **few random numbers needed** on average per stream => **initialisation time** is very important!
- Yet some tracks will have a **long 'life'** - many steps & interactions
  - e.g. muons, high energy electrons, some hadrons, low-E neutrons

Ordering





PRNG requirements 1/2

- Fast vectorised implementation
  - Parallelising over PRNG-streams
- Excellent* statistical properties
  - ‘no’ correlation of numbers within a stream
  - ‘no’ correlation between streams
- Efficient seeding from large integer (128-bit+ ?)
- Ability to use on new hardware types: GPU, MIC

PRNG requirements 2/2

- Amount of memory read & write per output number matters
  - Size of internal state is relevant for MT Geant4 and ‘regular’ Geant-V
- Size can be a critical issue
  - in the case of ‘**reproducible**’ mode, it contributes to total memory footprint - it must be smaller than the size of a track (1-2 Kbytes)
  - or if vector efficiency ‘requires’ copying of RNG state to use aligned vector instructions.
- Ideally internal state is of the order of one or two cache lines

PRNG options ?

- Traditional / existing simulation - **large state**
  - RANLUX - faster LUX=3 or better/costlier ‘5’ (state: 24 doubles)
    - Note: SSE implementation (ca 2002, Luscher) can produce 4 streams much faster
  - Mersenne twister RNG & variants (e.g. WELL) - state 507 integers
- Potential alternatives - **smaller state**
  - ‘Modern’ Linear Congruential Generators
- Using multiple streams there are **no guarantees of independence** - depending on very large period to make probability of collision very small

PRNG for fine grain (per track)

- Small state, but known or potential drawbacks
  - ‘Modern’ LCGs (question of correlations ?)
- PRNGs with the most potential for use
  - Random123 - PRNG ‘without state’ based on Cryptographic ‘technology’ - J. Salmon et al (state = counters)
  - New MIXMAX class of generators
    - Predictable qualities (de-correlation) backed by maths

Properties of MIXMAX

- Theory based on Anosov C-systems ensures properties from math theory
  - predictable mixing, de-correlation and memory loss (G. Saviddy et al, 1990)
- A new state is obtained $\mathbf{x}_{j+1} = A\,\mathbf{x}_j$
  - The integer matrix A must fulfil two conditions (det(A) = 1 & no eigenvalues with $|e_i| = 1$). The period must be found to allow use as PRNG.
- Family $A(N_{rank}, s)$ with shortcut algorithm to compute $\mathbf{x}_{j+1}$ & method to find period (K. Saviddis). Extended family $A(N_{rank}, s, m)$ created with larger 'entropy'
  - Identified 'A' with N=17 with large 'entropy' which passes BigCrush test (state = index + (17+1) 64-bit numbers)
- Method for initialising by skipping from one initial state. Cost ~ N
- Method for generating non-colliding state at each interaction (by "splitting" for every new secondary) proposed. How to "split" to accommodate 10^7 tracks/ event, with some requiring many variates to be resolved.

Vectorisation: potential & issues

- Potential of newest ‘vector’ CPUs for 4-8 size vectors operands of 64-bit each
  - for a single stream PRNG, i.e. for a sequential application (or scalar mode)
  - for a multi-stream PRNG - each vector’s lane is used for a different track
- Each interaction produces a set of secondaries
  - for reproducibility each secondary of which must be given a new state of the PRNG
- Each interaction can consume a different number of variates (next slide)

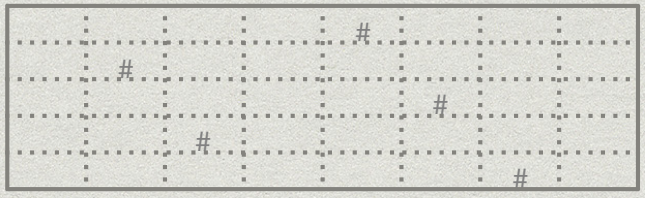
‘Multi-stream’ complications

- Physics processes can consume a different number of variates
  - one photon undergoing Compton may need 7 variates, another undergoing photo-absorption may need 9 variates
- Since a track must not be affected by the bucket in which it is processed,
  - it must not ‘know’ (be affected) about the other tracks which are being consumed ‘in parallel’.

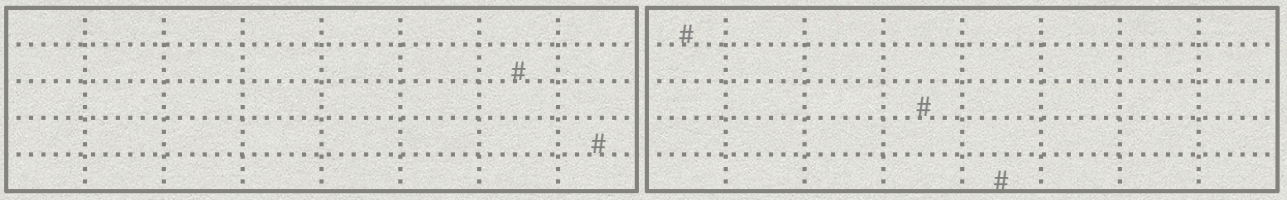
‘Multi-stream’use: implications

- So either the ‘vector’ implementation of the PRNG must allow different number of variates to be consumed (while vectorised).
- Potential solutions:
  1. ‘particle lockstep’: all tracks of a particle must use the same number of variates within a physics step (the maximum between them), or
  2. ‘de-coupled’: active tracks in a vector (‘lanes’) can be in a different
     - create a new set of variates together, putting results into (longer) working arrays
     - when a track is ‘stored’, only the active ‘window’ is stored (extra window of variates is discarded)

Before



After



Summary / conclusion

- PRNG are critical for particle transport simulation
  - bad choices could invalidate runs of thousands of CPU months!
- Moving to small-granularity parallelism means new challenges!
- Requirements for PRNGs for **fine-grain use** (to get **repeatability**)
  - efficient to implement, with a “small” state
  - vectorizable/SIMD & adaptable to GPUs (produce >10^3 streams!)
  - ability to obtain many streams & create state for each new tracks
- Some PRNGs meet some of the requirements: modern LCGs & Random123
  - a new family MIXMAX offers real promise to address all these needs

BACKUP SLIDES

Use of MIXMAX in Geant4

- Geant4 uses CLHEP library
  - MIXMAX “1.0” included in CLHEP 2.3.1.1 (Nov 2015)
- Updated MIXMAX 2.0 with new family and N=17 in Geant4 10.3-beta (June 2016)