RANDOM NUMBERS FOR PARTICLE TRANSPORT MONTE CARLO: NEW USE CASES AND REQUIREMENTS

JOHN APOSTOLAKIS, CERN

Overview

- Particle / Radiation Transport Monte Carlo
  - Beyond one run and one thread
  - Requirements for fine-grained parallel transport
- Update 2016
- Summary

LHC & Particle Transport

- LHC experiments are using running detector simulation - mainly Geant4
  - around 200,000 CPU cores at any time
  - almost every single day of each year!
- Random numbers are a vital part of particle transport Monte Carlo
  - Consume 3-10% of CPU time
  - Any serious error (bad seeding, wrong PRNG or other) would waste 10^4-10^6 US$ of CPU time!

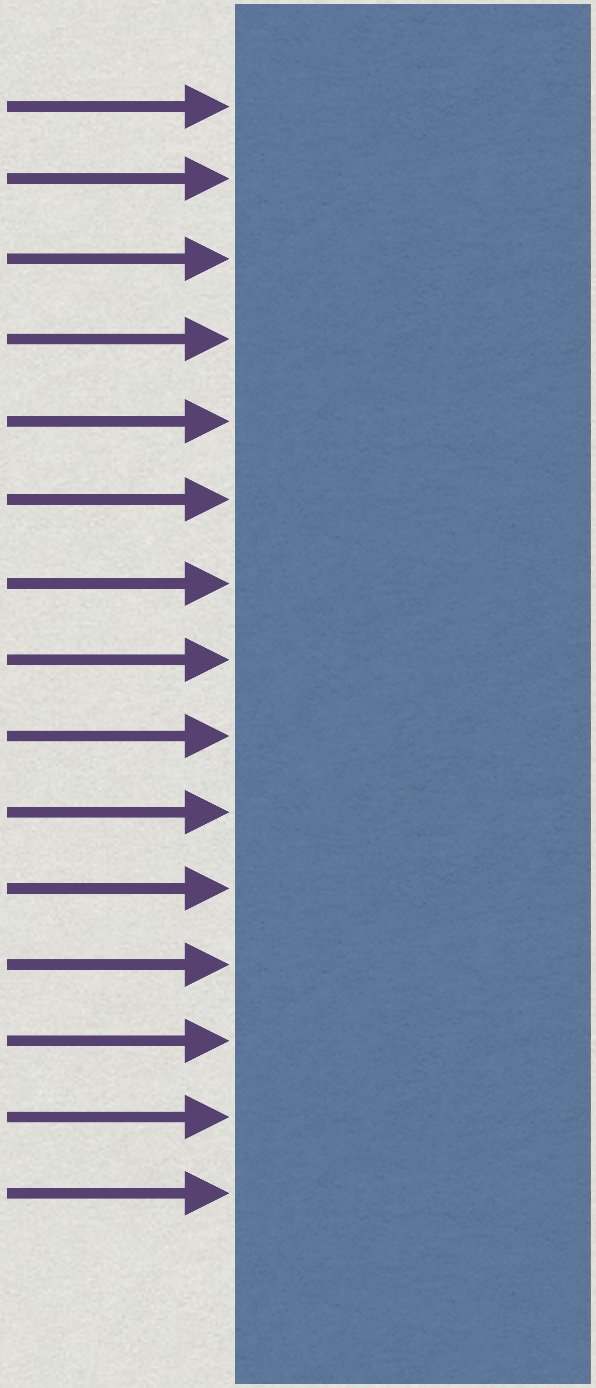
Particle Transport

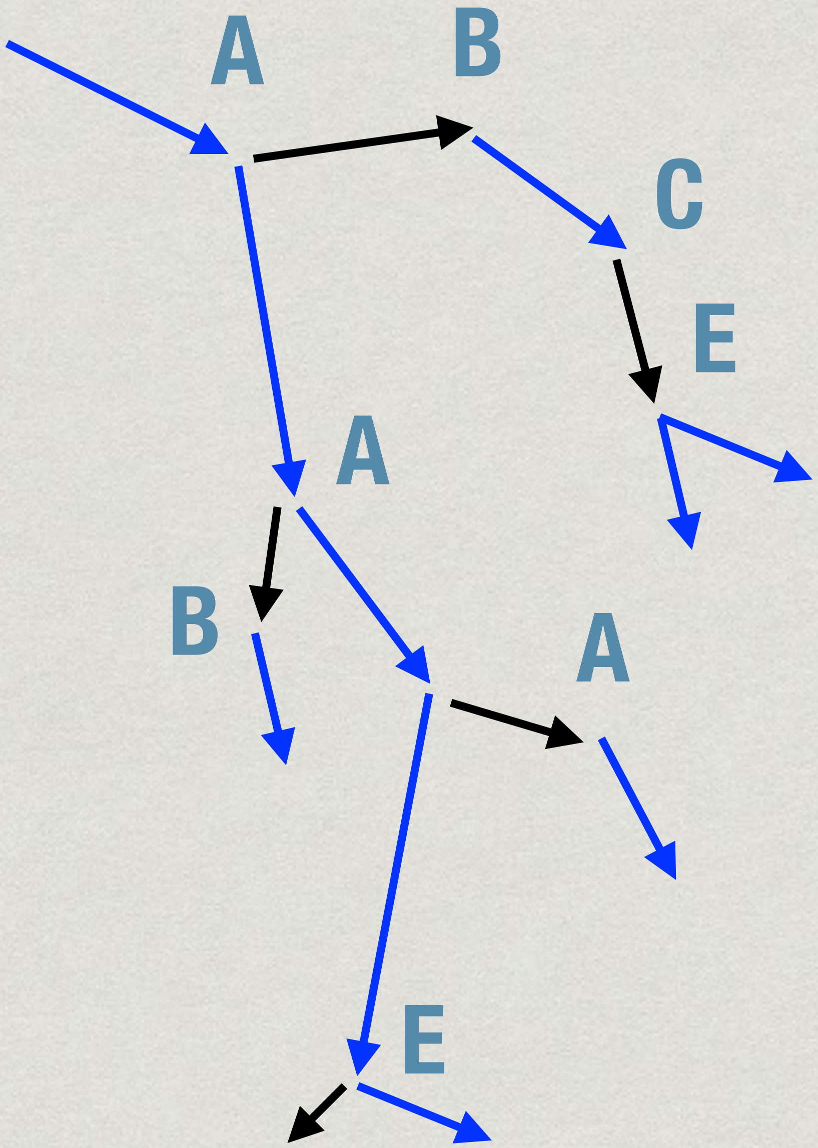
- Used to simulate interaction in/with detectors (HEP, Medical), facilities (accelerators), and even planet's atmosphere
- Particles undergo interactions - microscopic or 'effective'
  - can create new, secondary particles (cascade = tree of particles)
- Many 'decision' depends on using a random number
  - Deciding which interaction (e.g. absorption or scattering?)
  - Generating a secondary particle depends on sampling value from a probability distribution function

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 @ Luxury Level=5 can be > 10%
  - it matters - so we should seek to make it < 2-3%, if possible
- Reproducibility/portability between operating systems & CPU arch.

Example





Parallelism

- Clusters: Using from O(100) to O(1000) on a site
- Grids - taking part of the time of many more cores O(100,000) for one application or one experiment
- Inside one system - SIMD instructions in CPU, multiple cores (desktop or accelerator)
- Parallelism can be used at many levels of granularity
  - Job - different CPUs using batch processing or grid
  - Event parallelism - choice for multi-threading
  - Track parallelism - for primary tracks (or finer ?)

Job parallelism

- Simulation in the Grid has used job parallelism
- Typical: a job consists of 50-1000 events (collisions)
- Each job must use a separate stream of PRNG
  - Initial seed is pre-generated using job type & id
  - Full state of PRNG at start of each event may be stored with event output (for reproducibility)

Multi-threading

- One job uses many threads
  - multiple ‘actors’ or thread of control
  - share address space of a single process
  - constant data can be shared between threads (significant memory savings)
- Typically number of threads depends on hardware resources - number of cores, ideal threads/core

Event parallelism

- Natural evolutionary choice for MT is event processing - chosen by Geant4
  - the work unit for a thread is an event
- To obtain / ensure reproducibility of events, each event must have a predefined state or seed.
- Geant4 production releases included MT event parallelism since 10.0 (Dec. 2013)

Why fine-grained parallelism?

- LHC/HEP experiment must get 5-10x detector simulation from existing resources
- Today's CPU have vector/SIMD registers & instructions output / cycle - factor of 4-8x
- Traditional HEP & PT code uses few vectors & stumbles constantly in using modern CPUs (e.g. low intensity in instruction impact and almost no instruction re-use)
- Need to restructure algorithms to adapt to the most important element for performance - the memory cache hierarchy
- Exploring the potential of accelerator architectures (GPUs, Intel Xeon Phi(TM))

Fine-grained parallelism - does it change anything?

- Impact
  - Can no longer follow 1 track (electron, photon, neutron) at a time
  - Must **gather** similar **work** into groups - ‘baskets’
  - A major **revolution** was and is needed - transport must be completely reorganised from the ‘ground’ up
- R&D
  - explored in depth by the GeantV prototype (2012-2014) ;
  - now in development in GeantV project (geant.cern.ch)

How does it changes?

- No longer ‘follow’ one particle at a time
- The work is organised in vectors/baskets of particles
  - all particles in one type of volume
  - electrons in Fe (i.e. one material)
  - photons in PbWO4 (complex material)
- Each part of work is done on all tracks in a basket - in parallel
  - 5 or 17 photons undergo Compton in PbWO4
  - 27 or 127 particles arriving at a volume boundary are relocated

What does this mean for PRNGs?

- A larger number of concurrent PRNG streams
  - Minimum concurrent (current configuration):
    - N_threads = O(100)
- In a different ‘mode’ (reproducible) the number of concurrent PRNG is much larger:
  - N_PRNG = number of tracks in flight = 10^5-10^8 ?

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
  - if vector efficiency is marginal and requires copying of RNG state to use aligned vector instructions
  - 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)
- Ideally internal state is of the order of one or two cache lines

Emerging ideas

- To meet reproducibility of simulation
  - each track must use its own PRNG stream
  - a secondary particle must start with a unique, reproducible number (seed)
  - N_streams = N(tracks in flight)
    - = N(baskets) * A_average_occupancy ~= 10^6-10^8 ?
- Most tracks are low energy - **few random numbers needed on average per stream** => initialisation time is very important!
- These are ideas under development - J.A. & Sandro Wenzel

Overview - Update 2016

- Particle / Radiation Transport Monte Carlo
- Beyond one run and one thread
- Requirements for fine-grained parallel transport
- Summary

Topics

- Progress - use in Geant4
- Requirements for parallelism
  - (Re)seeding/splitting for event/track parallelism
  - Repeatability
- CPU Performance & potential for Vectorisation
- Loss of memory of initial state & de-correlation

Integration in Geant4

- Geant4 uses CLHEP library
  - MIXMAX “1.0” included in CLHEP 2.3.1.1
- Alex Howard will report some experience & issues

Requirements for parallelism

- Seeding for event parallelism
- Seeding for repeatable track-level parallelism

Vectorisation

- Potential of newest ‘vector’ CPUs for 4-8 size vectors operands of 64-bit each
    - i) for a single stream PRNG, i.e. for a sequential application
    - ii) for a multi-stream PRNG - each vector’s lane is used for a different track
- Q: Is it possible to use a different divisor (in place of $2^{61} - 1$) in order to make ‘best’ use of more vector FPUs?

‘Multi-stream’ PRNG: needs

- Deal with N ‘tracks’, with N= 1 - 64 (typically)
- Vectorise for N= (2,) 4, 8
- Each track
  - produces a set of secondaries
    - each secondary of which must be given a new state of the PRNG (to ensure repeatability of simulation)
  - can consume a different number of variates

‘Multi-stream’ PRNG

- Each track consumes a different number of variates
  - one photon can undergo Compton, which may need 7 variates, another may undergo photo-absorption and need 12 variates
- Since a track must not be affected by the batch in which it is processed,
  - it must not ‘know’ (be affected) about the other tracks which are being consumed

‘Multi-stream’use: implications

- So either the ‘vector’ implementation of the PRNG
  1. must allow different number of variates to be consumed (while vectorised), or
  2. all particles of a particular type must use the same number of variates within a step

The competition

- Traditional / existing simulation
  - RANLUX - with affordable LUX=3 or costly ‘5’
  - Merseinne twister RNG & variants
  - ‘Modern’ Linear Congruential Generators
- Fine grained simulation - large scale limit
  - Random123 - PRNG ‘without state’ based on Cryptographic ‘technology’ - J. Salmon et al

Personal view

- Some believe that even lower-quality PRNGs could be adequate for particle transport
  - there are many interactions, many decisions, …
- My view is that using a high-quality PRNG is important or even vital:
  - correlations in the first numbers can lead to same/similar results of interactions and reduce quality of results
  - it is an important insurance policy - if we can afford it.
- I agree with Fred that “we should seek the best PRNG we can afford” !
- So we should seek a fast, mathematically-motivated ‘excellent’ PRNG - and MIXMAX offers the chance to do this!

Summary / conclusion

- PRNG are critical for particle transport simulation
  - bad choices could invalidate runs of tens of thousands or even millions of CPU months!
- Moving to small-granularity parallelism means new challenges!
- A clear opportunity exists for high-quality PRNG (family)
  - efficient to implement, with a “small” state
  - vectorizable/SIMD & adaptable to GPUs (produce >10^3 streams!)
  - with ability to obtain very-many streams
- A clear opportunity for MIXMAX !