A Portable Implementation of RANLUX++

Jonas Hahnfeld, Lorenzo Moneta

May 18, 2021

Random Number Generators in High Energy Physics

- Generators with excellent quality and statistical properties:
  - MIXMAX (1991 / 2015)
  - RANLUX (1993 / 1994)

- RANLUX: subtract-with-borrow generator with simple recursion
  - Waste intermediate states to decorrelate generated numbers

  $\rightarrow$ Luxury level: higher quality with longer computing time

- RANLUX++: use the equivalent Linear Congruential Generator (LCG)
  - Avoid computing unneeded intermediate results, much higher quality "for free"
  - Advantage: fast skipping of numbers / "jumping" in generated sequence

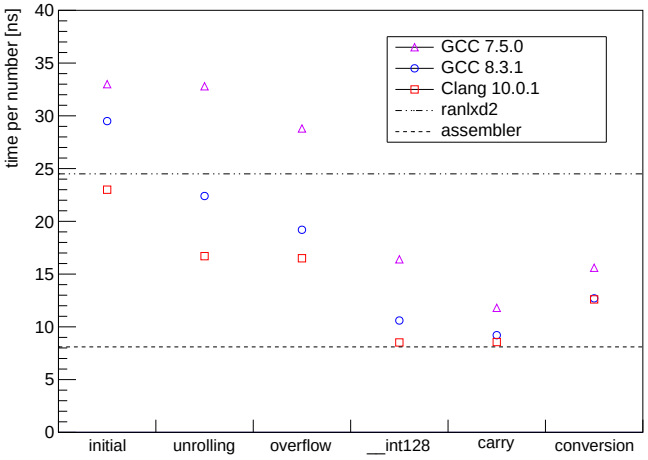
Portable RANLUX++ for ROOT

- RANLUX++: requires large state and multipliers (576 bits)
  - Shown to be profitable by A. Sibidanov in 2017
  - Arithmetic operations implemented in assembler for x86 architecture
  - For ROOT data analysis framework: portable implementation with standard C++

- Include a fix to avoid bias in generated numbers (not equally distributed)
  - Reported and solution proposed by M. Lüscher

  $\Rightarrow$ Convert LCG state back to RANLUX numbers

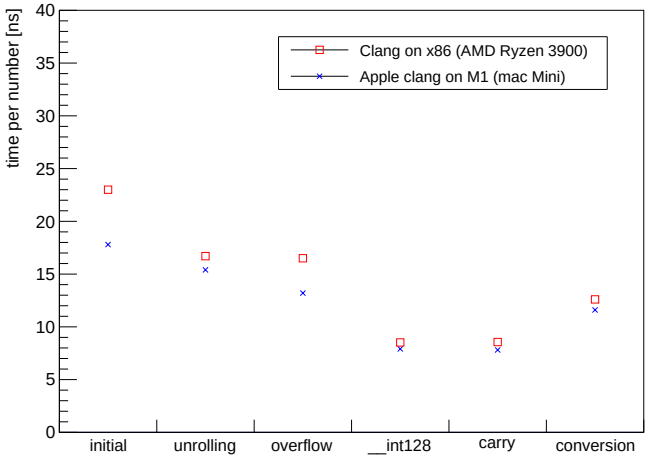
Optimization on x86



- AMD Ryzen 3900, produce `double` precision numbers
- Baselines:
  - assembler implementation by Sibidanov (bottom line)
  - `ranlxd2` by Lüscher
- Last column: conversion back to RANLUX numbers

Portability - Apple M1

- Implementation works on new architecture
- Optimizations give similar benefits



Portability - Nvidia GPUs

- Portable code can be reused with minor modifications:
  - Remove the dependency on ROOT's interface `TRandomEngine`
  - Hardcode the luxury level `p = 2048` (recommended value)
  - Add annotations `__host__ __device__`
  - Disable type `__int128` on the device

- Acceptable performance on the GPU (11.7 seconds for 10^{11} numbers)
  - Condition: threads must advance state at the same time
  - Slower than default generator in cuRAND (XORWOW, 3.1 seconds)
  - But: much better properties, already used in AdePT (MC simulation on GPUs)

Conclusion

- Portable implementation of RANLUX++
  - No assembler, only standard C++
  - Included in ROOT data analysis framework
- Portable optimizations on x86
  - Reached very competitive performance
- Tested on Apple M1 and Nvidia GPUs
- Work to get into GNU Scientific Library (GSL) and C++ standard