Vectorization Studies of Random Number Generators on Intel's Haswell Architecture

19 Aug 2014

Yigit Demirag (PH/SFT)
Dr. Sandro Wenzel (PH/SFT)
Daniel Funke (PH/SFT)

CERN openlab

Background image: Shutterstock

Background-1

- **Random Number Generators** are widely used at CERN, especially Monte Carlo Simulations in GEANT4, ROOT, etc.
- PRNGs are deterministic algorithms generating pseudorandom numbers.
- **Main Goal :** To study vectorization of selected Counter Based Random Number Generators (CBRNG), while comparing AVX2 and SSE4.2 performances using vector library and autovectorization.

  uint_64t someRandomNumber **= CBRNG(uint64_t** key**, uin64_t** counter**)**
- Philox and Threefry are 64-bit CBRNGs proposed by K.Salmon at 2011 [1]

[1] K.Salmon. Parallel Random Numbers: As Easy as 1, 2, 3. ACM 2011

Background-2

Philox	Threefry	XorShift[2]
• SP - Network • S box is a simple Feistel function with 72 rounds **64-bit [XOR, MUL]** $L' = B_k(R) = \mathrm{mullo}(R, M)$ $R' = F_k(R) \oplus L = \mathrm{mulhi}(R, M) \oplus k \oplus L.$ • 16 rounds 64-bit ADD with constant to key.	• SP - Network • 21 rounds of 64-bit [ADD,XOR,MUL,AND] • ADD and AND are done with constants.	• Recursive algorithm • Consists of bitshift and 64-bit XOR. • Outputs are seeds for next function. • Proof-of-concept work

[2] Marsaglia, G. (2003) Xorshift RNGs. Journal of Statistical Software vol. 8 issue 14.

Contribution-1

- 64-bit CBRNGs **Philox**, **Threefry** and **XorShift** are implemented.
- To be able to test auto-vectorization by GCC and ICC, input data structure is transformed from **AoS to SoA**.

Steps :

1. Find hot loops/basic blocks to vectorize (Vtune,SDE)
2. Make sure arrays are aligned.
3. Check data dependencies (ie. algorithm is vectorizable)
4. Use __restrict__ keyword in C99 to tell compiler there is no pointer aliasing
5. Make sure only calling functions are inlined (void __attribute__ ((always_inline)))
6. Try auto-vectorization with (-O3 -xAVX2 -vec-report2)
7. If doesn't vectorize try #pragma vector always, then #pragma ivdep.
8. Increase performance by unrolling the loops by #pragma unroll(N)

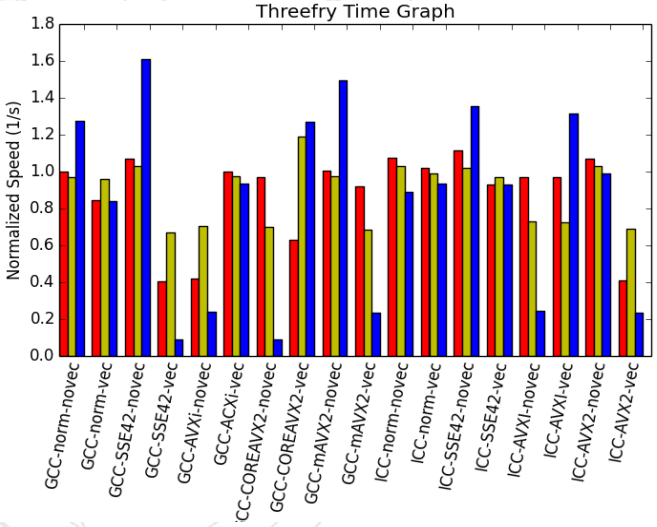
Contribution-2

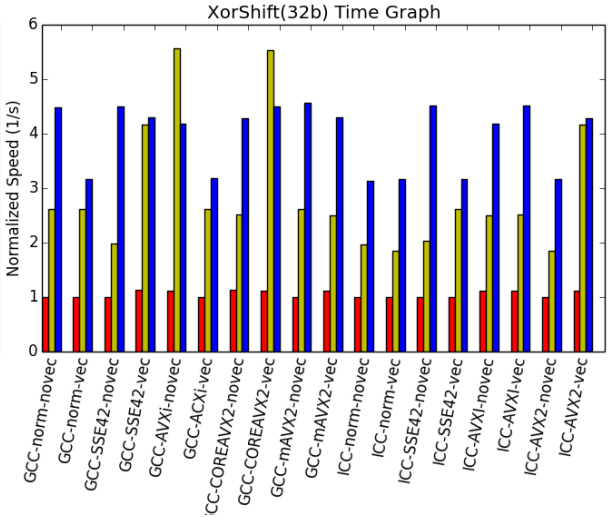
- Benefited from Agner Fog's Vectorization Class that encapsulates the **intrinsic** type to vectorize **Threefry** and **Xorshift** RNGs[3].
- Tested different optimization levels (-O2, -O3), compiler options for vectorization and different architectures (SSE4.2, AVX1, AVX2)
- Wrote a script that generates different executables for different architectures, sets governor frequency to 3.0, pins tasks to only one CPU and generates performance plots.

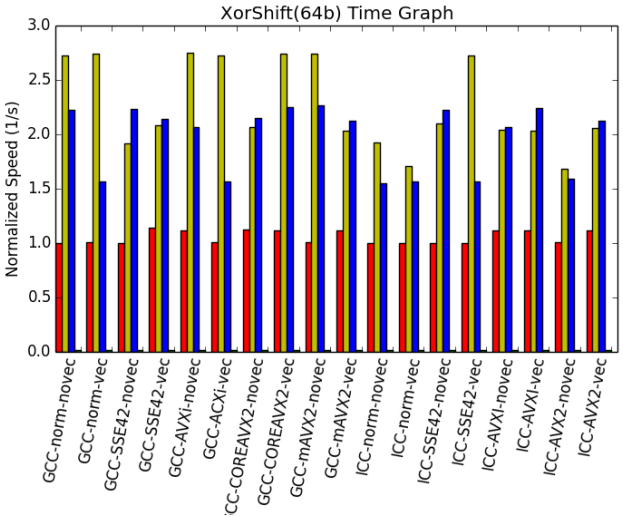
[3] http://www.agner.org/optimize/

PRELIMINARY RESULTS

The Project Impact







Normal D.Structure | SoA implementation | Fog's Library

Background image: Shutterstock

Thank You!

yigit.demirag@cern.ch