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.

```c
uint_64t someRandomNumber = CBRNG(uint64_t key, uint64_t counter)
```

Philox and Threefry are 64-bit CBRNGs proposed by K.Salmon at 2011 [1]

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

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)`
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.

The Project Impact

PRELIMINARY RESULTS

Threafry Time Graph

XorShift(32b) Time Graph

XorShift(64b) Time Graph

<table>
<thead>
<tr>
<th>Normal D. Structure</th>
<th>SoA implementation</th>
<th>Fog’s Library</th>
</tr>
</thead>
</table>

19/08/2014

Yigit Demirag
Thank You!

yigit.demirag@cern.ch