Learn about AMD's PRNG library we developed: rocRAND - includes benchmarks

When CUDA kept having a dominance over OpenCL, AMD introduced HIP - a programming language that closely resembles CUDA. Now it doesn't take months to port code to AMD hardware, but more and more CUDA-software converts to HIP without problems. The real large and complex code-bases only take a few weeks max, where we found that solved problems also made the CUDA-code run faster.

The only problem is that CUDA-libraries need to have their HIP-equivalent to be able to port all CUDA-software.

Here is where we come in. We helped AMD make a high-performance Pseudo Random Generator (PRNG) Library, called rocRAND. Random number generation is important in many fields, from finance (Monte Carlo simulations) to Cryptographics, and from procedural generation in games to providing white noise. For some applications it's enough to have some data, but for large simulations the PRNG is the limiting factor.

The library provides the most used PRNGs and QRNG (Quasi RNG) based on what we found on Github. Several you can find in cuRAND:

- XORWOW
- MRG32k3a
- Mersenne Twister for Graphic Processors (MTGP32)
- Philox (4x32, 10 rounds)
- Sobol32

If you're familiar with PRNGs, you see that from the most important families of generators there is an option. Now it's easy to port software that uses cuRAND. But that's not all.

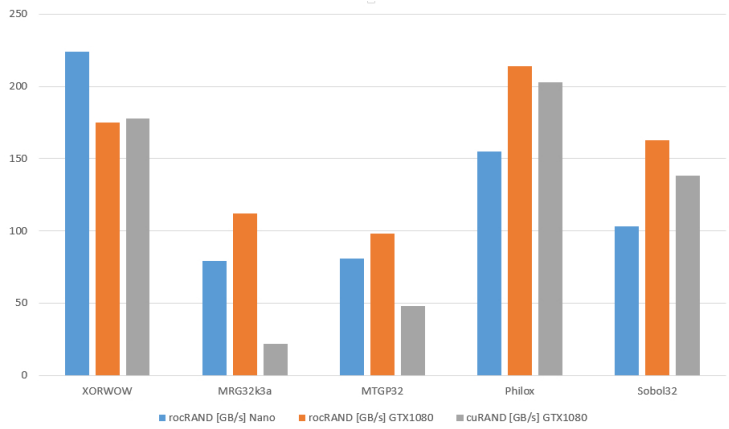
rocRAND is faster than cuRAND in most cases

rocRAND works on NVidia hardware too. And in most cases it's faster than cuRAND.

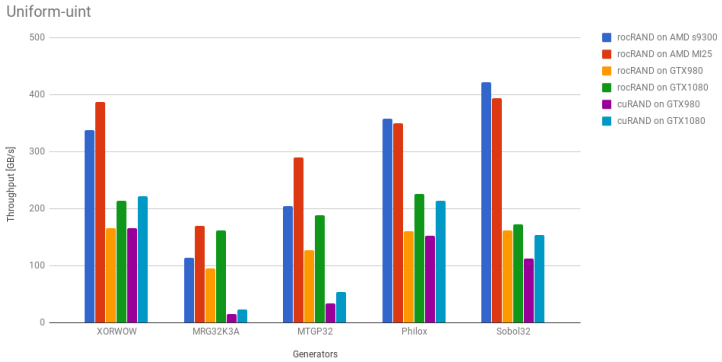
Here we compare rocRAND for normal-floats on the AMD Radeon Nano, rocRAND on the GTX 1080 and cuRAND on the GTX 1080. The professional grade GPUs, like the AMD MI25 are much faster - but this is just to show that the library written for AMD GPUs is faster than NVidia's own library.

Benchmarks

This is before the optimization-phase on AMD R6 Nano and Nvidia GTX1080 - rocRAND on par with cuRAND.



This is after the optimizations, where AMD GPUs get the upper hand due to higher bandwidth memory:



As you can see, it's preferable to also use the library for NVidia-only projects.

Doing your own benchmarks

On the Github of rocRAND you find instructions to benchmark the library on your own hardware. Do know that the library has been tuned for all recent AMD GPUs and Nvidia GTX GPUs, not Tesla GPUs. Also the code does not work on CPUs or Intel GPUs.

More on random numbers on our blog

Want to know more about Random numbers? We wrote about the subject before.

https://streamhpc.com/blog/2016-03-18/random-numbers-in-parallel-computing-generation-and-reproducibility-part-1/

https://streamhpc.com/blog/2016-08-17/random-numbers-parallel-computing-generation-reproducibility-part-2/

https://streamhpc.com/blog/2016-08-18/porting-code-that-uses-random-numbers/

Need a tailored RNG?

When you know the exact restrictions you have for your project, we can:

- further tune the library to be even **faster**, or

add **special characteristics** (i.e. less cyclic), or
port **other PRNGs** to the GPU.

We did not put these hacks in the official code, as we then could not guarantee a correct output for generic goals. In case you need a RND tailored for your specific needs, we are the team that can build it.

Get in touch with the GPU Library Specialists today.