CPU Code modernisation - our hidden expertise

You've seen the speedups possible on GPUs. We secretly know that many of these techniques would also work on modern multi-core CPUs. If after the first optimisations the GPU still gets an 8x speedup, the GPU is the obvious choice. When it's 2x, would the better choice be a bigger CPU or a bigger GPU? Currently the GPU is chosen more often.

Now AMD, Intel and AMD have 28+ core CPUs, the answer to that question might now lean towards the CPU. With a CPU that has 32 cores and 256bit vector-computations via AVX2, each clock-cycle 32 double4 can be computed. A 16-core AVX1 CPU could work on 16 double2's, which is only a fourth of that performance. Actual performance compared to peak-performance is comparable to GPUs here.

AMD started with launching a 32-core CPU. Intel reacted immediately by hinting they will also launch a 32-core Xeon. Meanwhile IBM works on launching their quad-threaded 24-core Power9 CPU. Cavium is providing 64bit 64-core ARM processors, which also need many threads to keep them busy. Not only core-numbers increase, but the interconnect standards now all push for upgrades while HBM seeks a way outside GPUs.
CPUs have reborn.
We will discuss the advantages of these CPUs in upcoming blog posts.

Algorithm and CPU optimisations

Where performance optimisation (aka "code modernisation") often is described as "applying tricks". Having a specialisation of making easily-readable code that performs, we know better. With CPUs taking all that makes GPU fast, we now can also apply GPU-specific optimisations in the CPU-domain.

An example of where we used the CPU in a project was with NooSpheer. The initial goal was to use the GPU, but we found that the CPU was faster for that given algorithm. The 40,000x speedup was on a 8-core CPU with AVX, and the code is expected to run much faster on the CPUs described above.
?Stream is an elite, dependable and unique development outfit. We utilized Stream to achieve a **~40,000x speedup** for our quantum simulation software.
If your project demands **ultra fast design and robust implementation**, work with them.?

Jordan Ash, CEO Noospheer

The multi-core CPU is dead.

Long live the multi-multi-core CPU!]
In 2012 I wrote on CPUs with embedded GPUs. That was a big change and defined a complete new type of CPU. 20+ cores is not so big, but still it defines a new type of processor. This means a split: 4 or 8 cores for desktops, laptops and mobiles, and 20+ cores for

shared servers and HPC.

You might ask, why only now? This is due to better (and more accepted) virtualisation techniques and (thanks to GPUs) more code that's optimised for handling data-parallel workloads. Another reason is Intel's monopoly in the server-market, that is now heavily under attack by ARM, IBM and AMD.
These modern, 20+ core CPUs are very welcomed, as they make the CPU-GPU gap smaller.
As you might have guessed, programming a 28+ core CPU is different from programming a 4-core CPU, but it's very possible. We know that because we have a long experience in building scalable software, that can be optimised for both low-end quad-core CPUs and high-end 1000+ core GPUs. We're welcoming the multi-multi-core CPUs with open arms.