How we sped up a flooding simulation 35 times (from 32-core CPU to multi-GPU)

Hampstead flooding

How water moves through an area given a certain pace of instream, can be fully simulated. We got a request to make such simulation faster, as it took already too much time to do moderate simulations. As the customer wanted to be able to have more details, larger areas and more alternative situations computed, the current performance did not suffice.

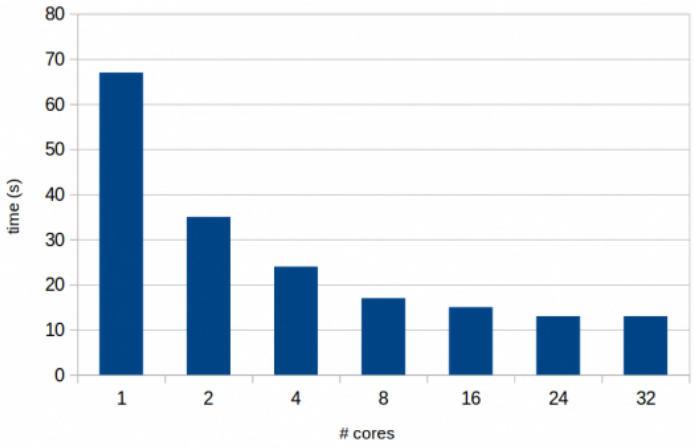
The code was already ported to MPI to scale to 8 cores. This code was used as a base for creating our optimised GPU-code. Using **a single GPU** we managed to get an 44 to 58 times speedup over single core CPU, which **is 5 to 7 times faster than MPI on 8 to 32 CPU cores**.

For larger experiments we could increase the performance advantage over MPI-code from 7 times to **a total of 35 times, using multiple GPUs**.
We solved both the weak-scaling problem and the mapping on GPUs
If you add the 9x speedup of the initial performance-optimisation, the total is over 2600x. What could be done in a year, now can be done in 3.5 hours. This clearly shows the importance of **software performance engineering**. Most code already had some optimisations applied (just like here) and 5 to 7 times speedup is quite achievable.

Read below for some more details.

Initial port to MPI

A small experiment already took 10 minutes using the original code. Others before us optimised the code to run that same experiment in 67 seconds, a good 9x speedup. They used MPI to further speed up the code on multi-core CPUs. The code scaled well to 8 cores, but not hardly gave more speedup on 32 cores (2x 16 core Xeon) due to communication overhead. Also with bigger experiments, as the amount of computations done by one core has the same limits.



Scaling of the flooding MPI-code from 1 to 32 cores.

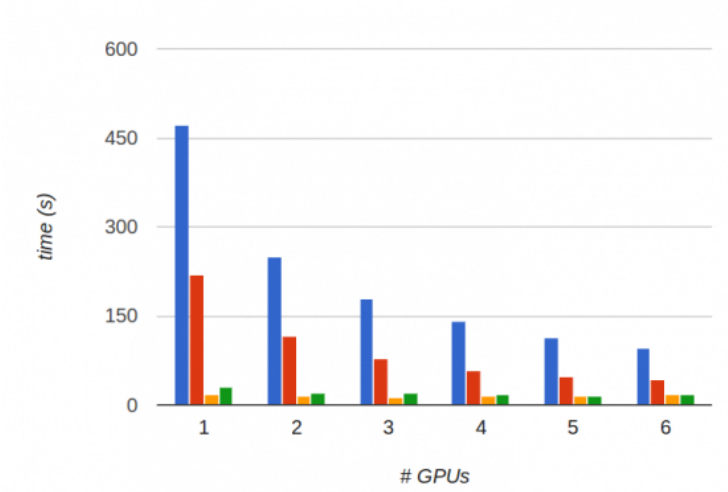
With this weak scaling after 8 cores, it was clear we needed to do more than just do a quick port from MPI to OpenCL. As the optimisations were well-documentend and thus we did not needed to fully reinvent the wheel here, we could focus on specific optimisations.

Our port to OpenCL

As the experiment was bandwidth limited, we chose the dual-GPU **FirePro S9300X2** for its fast HBM memory (which has a staggering 500 GB/s per GPU). Focusing on the communication-overhead, we could find several GPU-specific optimisations to get to 44 to 58 time speedups - 5 to 7 times faster than MPI on 32 CPU-cores.
Based on GPU-usage measurements we expect even larger speedups for larger experiments.
We ran the OpenCL-code on CPUs, but mostly due to GPU-specific optimisations we did not see any extra speedup on the CPU.

Multi-GPU

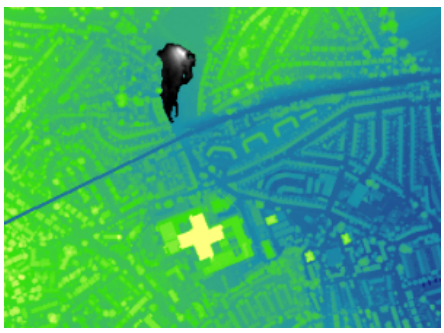
With three dual-GPU FirePro S9300X2's available in the server, we had 6 high-end GPUs to scale to. To get good scaling, extra work was done on reducing the communication-overhead. The focus was on larger sized experiments, as the speedup of smaller experiments did not scale at all.



Scaling of different sized experiments from 1 to 6 FirePro GPUs.

Scaling to the 6 GPUs gave a 5x speedup over one GPU, where the scaling after 4 GPUs got weaker. A combination of using better scaling techniques and using the FirePro's fast HBM memory gave a total speedup of 295 times over a single threaded program and 35 times over MPI on 32 cores.
What took over a month on a high-end Xeon server now takes only a day on the FirePro-server

The need for larger simulations



With rising sea levels you understand this software is important. With increasing demand for larger sized simulations, time-reduction of 35 times for such experiments was much needed.

We can bring you in contact with our client if you need to have large scale flooding-experiments computed. Your simulation can be computed on StreamHPC's servers.

If you want to have your river managed, lake restored or wetland protected and have a need for larger simulations, mail us and we'll bring you in contact with our customer.

Hopefully we have shown you what we are capable of, if it comes to speeding up simulation software. If you have such code to have scaled up like this flooding simulation, call us to discuss.