What does it mean to work at Stream HPC?

High performance computing on many-core environments and low-level optimisations are very important concepts in large scientific projects nowadays. Stream HPC is one of the market's more prominent clubs and is substantially expanding. As we often get asked how it is to work at the company, we'd like to give you a little peak into our kitchen.

Stream HPC's DNA

To understand our DNA, you'd need to know how the company got started. In 2010 the company was born from the deep boredom that was born from within the corporate IT workspace. Stream's founder Vincent Hindriksen had to maintain a piece of software that was often failing to process the daily reports. After documenting the internals and algorithms of the code by interviewing the key people and some reverse engineering, it was a lot easier to create effective solutions for the bugs within the software. After fixing a handful of bugs, there was simply a lot less to do except reading books and playing online games.

To avoid becoming a master in Sudoku, he spent the following three weeks in rewriting all the code, using the freshly produced documentation. 2.5 hours needed to process the data was reduced to 19 seconds ? yes, the kick for performance optimisation was already there. For some reason it took well over 6 months to port the proof-of-concept, which was simply unbearable as somebody had to make sure the old code was maintained for 40 hours a week.

The reason to start the company was simple: to make intelligent use of time and provide software that is engineered for performance and maintainability. Lots of exciting projects for fantastic clients followed in the next 8 years that allowed us to broaden our expertise and build up confidence. GPUs were there at just the right time ? without GPUs it would have probably been performance engineering on CPUs.

Projects we do

Today we work on plenty of exciting projects and discover new markets. Here are a few ongoing projects:

For an embedded project we're working on signal processing. Implementing an extremely fast feedback-loop to reduce the signal-to-noise ratio in a high bandwidth environment. The interesting part here is that the used GPU has special extensions, and the code and the accompanying hardware will be used world-wide.

For a space agency we're creating a faster version of spacecraft simulations, with the goal to make interaction with the simulator close to real-time. Just imagine telling your partner that you work for a space agency.

For a medical company we're making sure the image stream can be processed while the camera operates at a continuous rate of 600 Hz. Here there is another balancing problem: getting this performance, while keeping the required power limited.

The one public project we're doing is for AMD. We build high-performance libraries for them, which can be found on Gitlab. Recently we made rocPRIM and before that rocRAND. This is very interesting to do, especially because these libraries are being used to power well-known projects in many computer science fields, like TensorFlow in artificial intelligence.

Upcoming is work on compilers. How to make an LLVM compiler make sure a certain GPU will work as fast as possible when using different high level parallel languages? We have done some work for MESA, where we focused on solving bugs for certain games.

Who we are

We're a close-knitted group of motivated individuals, who get a kick out of performance optimisations and are experienced in programming GPUs. Every day we have discussions on performance. Finding out why certain hardware behaves in a certain manner when a specific computing load is applied. For instance why certain code is not as fast as theoretically promised, and then finding the bottlenecks by analyzing the device and finding solutions for removing those bottlenecks. As a team we make better code than we could ever do as individuals.

Quality is important for everybody on the team. This has a simple reason: we cannot speedup code that is of low quality. This is also why we don't use many tools that automatically do magic, as these often miss many important improvements and don't improve the code quality. We don't expect AI to dully replace us soon, but once it's possible we'll probably be part of that project ourselves.

Computer science in general is evolving at a fast rate and therefore learning, is an important part of the job. Reading papers, finding new articles, discussing future hardware architectures and how they would affect performance, is very important. With every project, we have to gather as much data as possible using scientific publications, interesting blog posts and code repositories in order to be on the bleeding edge of technology for our project. Why use a hammer to speedup code, when you don't know which hammer to use best?

And we share and appreciate humor!

Pull-style management and log-style documentation

We use Gitlab and Mattermost to share code and have discussions. This makes it possible to keep good track of each project - searching for what somebody said or coded two years ago is quite easy. Using modern tools has changed the way we work a lot, thus we have questioned and optimised everything that was presented as "good practise". Most notable are the management and documentation style.

Saying an engineer hates documentation and being managed because he/she is lazy is simply false. It's because most management and documentation styles are far from optimal.

Pull-style management is where the tasks are written down by the team, based on the proposal. All these tasks are put into the task-list of the project, and then each team member picks the tasks that are a good fit. The last resort for the tasks that stay behind and have a deadline (being pushed) was only needed in a few cases.

All code (MR) is checked by one or two colleague, chosen by the one who wrote the code. More important are the discussions in advance, as the group can give more insight than any individual and one can get into the task well-prepared. The goal is not to get the job finished, but not having written the code where a future bug has been found.

All types of code can contain comments and Doxygen can create documentation automatically, so there is no need to copy functions into a Word-document. Log-style documentation was introduced, as git history and doxygen don't answer why a certain decision has been made. By writing down a logbook, a new member of the team can just read these remarks and fully understand why the architecture is how it is and what the limits are. We'll discuss this in more detail later.

These type of solutions describe how we work and differ from a corporate environment: no-nonsense and effective.

Amsterdam

Amsterdam is the economic centre of the Netherlands, a small country with 17 million inhabitants. It's the home of HPC-companies Bright Computing and ClusterVision, and has a large IT workforce that also feed the R&D demand of large international companies. As the number of companies settling here is still growing, Amsterdam is even planning to build a complete new city for 40 to 70 thousand people in the harbour area.

There are different sides of the city. When you think of Amsterdam as a tourist, you might think of the Anne Frank House, Gay Parade, Van Gogh Museum, the Red Light District, the canals, windmills and Tulips. If you would consider living here there are about the 180 different nationalities that live in the city, the 22 international schools and two universities, the vibrant night life and the many-villages-make-the-city atmosphere. Locals of all professions are fluent in English and there is a lively expat community.

You don't need to live in Amsterdam, as there are several cities and villages nearby with all unique identities. As the Dutch infrastructure is of high standard, Amsterdam is easy to reach via train (and car) from several nearby cities and villages. For instance taking the train from Haarlem to the office takes 9 to 13 minutes, Leiden or Utrecht half an hour. Want to live at the sea? Zandvoort to the office is 25 minutes.

Expats (both single and with family) say they found it easy to build up a social life. For Europeans it's very easy to move to Amsterdam, as there are no real borders in the EU.

Where do we fit in your career?

Each job should get you forward, when done at the right moment. Question is when Stream HPC is the right choice.

As you might have seen, we don't require a certain education. This is because a career is a sum, and an academic study can be replaced by various types of experience. The optimum is often both a study and the right type of experience. This means that for us a senior can be a student and a junior have 20 years in the field.

So what is the "right type of experience"? Let's talk about those who only have job-experience with CPUs. First being hooked by performance, as primary interest would be the first reason to get into HPC and GPGPU. Second being good at C and C++ programming. Third knowing algorithms and mathematics really well and can quickly apply them. Fourth being a curious and quick learner, which shows by you having experimented with GPUs. This is also exactly what we test and check during the application procedure.

During your job you'll learn anything around GPU-programming with a balance between theory and practise. Preparation is key in

how we work, and this you will develop in many circumstances.

Those who left Stream HPC have gotten very senior roles, from team lead to CTO. With Stream HPC growing in size, the growth opportunities within the company are increasing also. To quote one of the applicants: "Once you get in, you know you'll have a career boost". Honestly, we prefer to keep those the team likes to work with, instead of loosing them to another company.

The future

Would you like to work for a rapidly growing company of motivated GPU professionals in one of Europe's most attractive cities Amsterdam? We seek likeminded and motivated people who are like us: curious and friendly. If you liked what you read here, do check our open job positions:

GPU library developer]
CUDA/OpenCL GPU developer]
- LLVM compiler developer for GPUs (opens soon)

If you know people, who are bored in their current job please send them to streamhps/jobs.