Get ready for conversions of large-scale CUDA software to AMD hardware

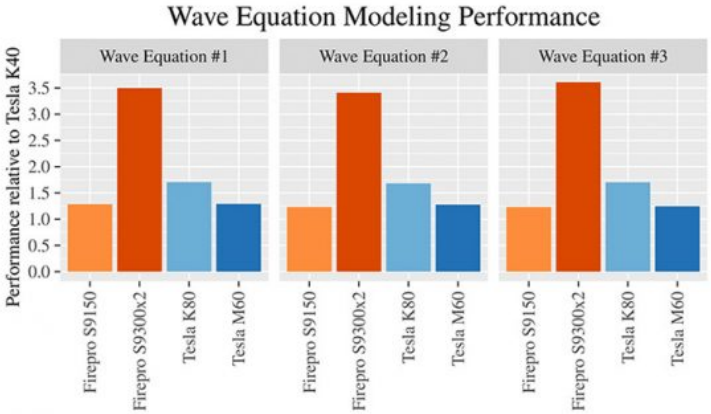
In the past years we have been translating several types of software to AMD, targeting OpenCL (and HSA). The main problem was that manual porting limits the size of the to-be-ported code-base.

Luckily there is a new tool in town. AMD now offers HIP, which converts over 95% of CUDA, such that it works on both AMD and NVIDIA hardware. That 5% is solving ambiguity problems that one gets when CUDA is used on non-NVIDIA GPUs. Once the CUDA-code has been translated successfully, software can run on both NVIDIA and AMD hardware without problems.

The target group of HIP are companies with older clusters, who don't want to pay the premium prices for NVIDIA's latest offerings. Replacing a single server with 4 Tesla K20 GPUs of 3.5 TFLOPS by 3 dual-GPU FirePro S9300X2 GPUs of 11 TFLOPS will give a huge performance boost for a competitive price.
The costs of making CUDA work on AMD hardware is easily paid for by the price difference, when upgrading a GPU-cluster.

But why AMD hardware?

While AMD does not promote their hardware as well as NVIDIA does, they have great offerings. For instance the 13.9 TFLOPS FirePro S9300X2 with dual GPU and very fast HBM memory, or the 5.2 TFLOPS FirePro S9170 with 32GB (!) of memory. The metric that serves them best is performance per Euro/Dollar - this is what got them a win at geoscience company CGG.



CGG is a leader in cutting-edge geoscience and recently conducted proprietary wave equation modelling benchmarking on several different GPU accelerators, including the new AMD FirePro? S9300 x2 GPU. As the complexity of the wave equation increased, the performance advantage also grew in favour of the AMD FirePro? S9300x2 GPU, to a point where it was 2x faster than any other card tested.

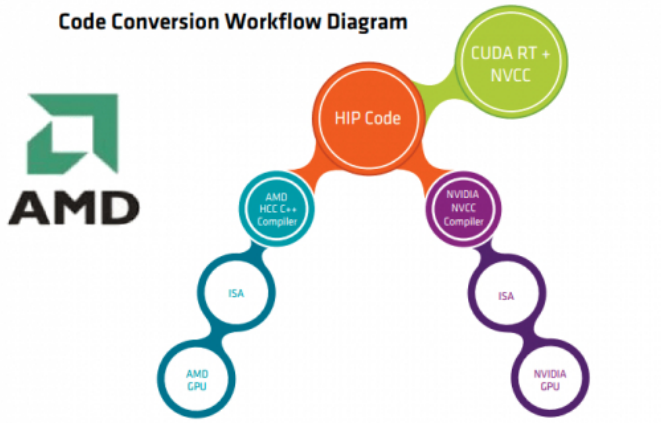
Knowing that the K80 and S9300X2 are dual-GPU cards, you see that the single-GPU S9150 (in the graph at 70% of K80 performance) is also doing very well in comparison.

With NVIDIA's latest offerings being very expensive and the positive news around AMD CPUs (Zen) and GPUs (Polaris, Vega, etc), interest in AMD is rising.

HIP makes it very cheap to test out legacy software on AMD hardware.

How does HIP work?

The below image explains it: CUDA gets converted to HIP and HIP gets compiled for the NVIDIA GPU with NVCC, and for the AMD GPU with their new C++ compiler HCC.



HIP can be seen as a fresh restart of CUDA, where inconveniences are removed - think of code that checks for __CUDA_ARCH__ to actually find out if there is double precision support. The language is very close to CUDA in its core features. Another choice would have been to fix all goals within the CUDA language, but that would have been an unbounded and moving target - now also both AMD and NVIDIA are primary targets of equal weight. Since HIP is open source, new and missing features can be added. See the project's contribution guidelines for more information.

Converting from HIP to CUDA is done with a header-file.

Developer tools are the same

As the conversion from HIP to CUDA works via a header file, the generated code is thus 100% CUDA-compatible. Tools like Visual Profiler and NSight therefore fully work on NVIDIA hardware.

For AMD there is CodeXL, which works with all AMD's drivers, languages and GPUs on both Windows and Linux.

Limitations

AMD hardware is not the same as NVIDIA hardware and therefore some optimisations need to be done to get good performance from the AMD GPUs. For instance NVIDIA has a warp-size of 32 and AMD hardware 64. If the code has been optimised with one GPU in mind, then this step would take more time - this is comparable to optimising CUDA for Pascal GPUs, when the code was written for the Tesla architecture.

HIP cannot be converted to OpenCL, due to limitations in the language and because OpenCL does not have a finished C++ implementation yet. This means that with HCC and HIP more hardware capabilities of AMD hardware can be used.

HIP and HCC are only supported by modern AMD GPUs based on Hawaii, Carrizo and Fiji architectures.

Is it yet another compute language?

Yes and no, or at least not at the moment. The prefixes of function names look like simple renames from CU to HIP - the 95% of the code that can easily be ported. As long as it stays this way, I'd say it's a flavour of CUDA where ambiguity has been removed.

I do hope NVIDIA embraces the initiative and starts supporting the improvements that HIP has introduced.

Unique advantages

When CUDA needs to be ported to future languages, most of the same problems need to be answered as during the porting to HIP. This means that HIP-code is more portable than CUDA.

HIP can be learned in a day, if you know CUDA.

The biggest advantage is that the porting can be done separate from optimisation, in contrast with OpenCL-to-CUDA porting.

Separating porting from performance optimisation

It is important to have a quick porting solution without any need for optimisation. This is because the question "What's the return on investment?" on porting the code to AMD needs to be answered as quick as possible. HIP brings down the costs of answering that question, especially for large code-bases.

The current situation: porting CUDA to OpenCL

Separating porting from optimisations is not easy when porting from CUDA to OpenCL. This is because OpenCL simply is different from CUDA in various ways. We ask a one-week effort (code-review, benchmarking, testing, trial-porting, etc) to be able to give an indication of how much time is needed to fully port code to OpenCL. For large scale code, the total project can take many months - remember our work on GROMACS.

Having two code-bases increases the maintenance costs. In most cases the CUDA-code is dropped for the optimised OpenCL-code to prevent this, with notable exceptions like GROMACS.

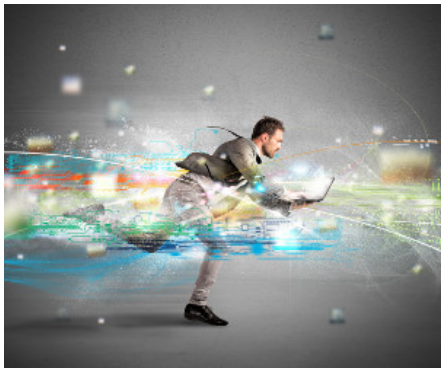
The new option: converting CUDA to HIP

By porting of CUDA-code to HIP, it is quicker to assess if AMD GPUs are worth the investment. **Within two days a report can be created on how much time it would take to port all code to HIP and what the expected performance will be**. In many cases it takes only a few days to a week to get a large code-base working on AMD hardware, quick optimisations can be implemented, and a first benchmark can be done. We will be sharing some real-world results on this blog later.

The maintenance costs also go down. General optimisations applied in the HIP code will have immediate result on all GPUs, making our work on performance optimisations more efficient.

StreamHPC's Services

We are heavily investing in HIP and offer the following services around HIP from today (between brackets the service under which it falls).



Assess possible performance improvement when using AMD FirePro (code review). We can tell you what improvement there is when upgrading to AMD hardware. If your code is better suited for another processor, we can also assess that.

Converting CUDA to HIP (porting). As we know the tools and hardware, we can do such porting much faster and with less mistakes.

Optimising performance (performance engineering). Making your code run faster on any NVIDIA or AMD GPU - our expertise since 2010.

Development environment setup (developer support). Assisting you in moving the development and build environments to HIP.

Automated Building + Benchmarking (developer support): Benchmarking selected GPUs to detect performance regression or improvement. You don't need to have all hardware and can focus on a few GPUs, and we can assist in preventing accidental regressions on certain devices by sending you daily performance reports.

For contacting sales or getting more information on HIP, AMD FirePro and our services, **call +31854865760** or send an email to info@streamhpc.com].