Development of mixed-precision matrix multiplication kernels for GPUs

Themes: Matrix computations, high performance computing, mixed precision, C++, CUDA.

Duration: Internship

Starting date: March 2024

Institution: LISN (Laboratoire Interdisciplinaire des Sciences du Numérique) and Université Paris-Saclay

Supervisors: Oguz Kaya (www.oguzkaya.com, oguz.kaya@universite-paris-saclay.fr),
Joel Falcou (joel.falcou@universite-paris-saclay.fr),
Nicolas Gac (nicolas.gac@universite-paris-saclay.fr),
Alexis Aune (alexis.aune@universite-paris-saclay.fr),

Funding: ANR JCJC SELESTE

Context

In the last few years, the tensor processing unit (TPU) performance of GPUs has improved significantly. These TPUs, for example, Nvidia's Tensor-Cores, expose immense performance in computing low-precision matrix multiplications (FP16, BF16, INT8, and FP8 GEMMs). This has opened up a new field of research called reduced-precision arithmetic, focused on tapping into the performance offered by modern hardware on low-precision operations when possible, be it to reduce the memory footprint of an application (for example ML models), or increase the performance of a high-precision operation by emulating it using multiple lower-precision operations. Reduced-precision arithmetic is a part of a broader field called mixed-precision arithmetic.

Mixed-precision arithmetic is focused on using multiple precisions in a single operation to make the best use of the underlying hardware. A standard operation in this field is the scale-gemm, which consists in scaling two input matrices to a different precision type before executing a GEMM. This can be used, for example, to achieve faster GEMMs by scaling two FP64 input matrices into FP16 before multiplying them at the cost of reduced precision. This can also be used to reduce the inherent loss of precision of a GEMM operation by executing it with higher precision than the input matrices, for example, scaling two FP16 matrices into FP32 and then performing the GEMM.

The naive implementation of the scale-gemm consisting of two discrete steps, generates a lot of memory reads and writes before the GEMM. Due to the growing gap between the compute performance and the memory and cache bandwidth on recent GPUs, such an approach will be inefficient at best and completely memory-bound at worst. In many cases, the scale-gemm operation is followed by another scale operation, incurring even more memory operations after the GEMM has completed.

Objectives

The goal of this internship is to develop a high-performance CUDA kernel to perform the scale-gemm operation while avoiding memory accesses as much as possible in order to make better use of the GPU computing power. A strategy would be to scale the input and the output of the operation in-line during the GEMM, as the matrices elements are requested. However, modern Tensor-Cores are especially powerful, and special care has to be taken not to starve them. To do so, the data scaling and the GEMM operation could be done in an asynchronous or pipelined fashion, making use of the scheduling capabilities of modern GPUs.

Expected results

In the end, we aim to develop a "drop-in" kernel for the scale-gemm and scale-gemm-scale operation, enabling performance gains when this type of operation is used.

At the end of the internship, there will be a possibility for a PhD. thesis or a 2-year HPC engineering position, to be financed by the Numpex project.

Collaborations

The internship will be conducted within the ParSys group at LISN.

Funding

ANR JCJC SELESTE

Qualifications

Required qualifications:

- Experience with C/C++.
- Experience with CUDA.
- Having completed at least one parallel/high performance computing course.
- Competence and interest in programming and algorithms in general.

Preferred qualifications:

- Experience with modern C++ (C++14 and above).
- Background in linear algebra.
- Experience with different parallel programming paradigms.

References

[1] Joel Falcou and Jocelyn Serot. Eve, an object oriented simd library. *Scalable Computing: Practice and Experience*, 6(4), 2005.

[2] Massimiliano Fasi, Nicholas J Higham, Florent Lopez, Theo Mary, and Mantas Mikaitis. Matrix multiplication in multiword arithmetic: error analysis and application to gpu tensor cores. *SIAM Journal on Scientific Computing*, 45(1):C1–C19, 2023.