Introduction to CUDA Programming on NVIDIA GPUs
Mike Giles

Practical 5: Tensor cores, cuBLAS and other libraries

This practical introduces the cuBLAS library and its use of Tensor Cores which were introduced in Compute Capability 7.0, the Volta generation of GPUs.

The first code `tensorCUBLAS` demonstrates the huge performance benefit of using the Tensor Cores, and the ease-of-use through the cuBLAS library. The second code `simpleTensorCoreGEMM` is a sample code from NVIDIA's Github repository which illustrates the more challenging programmatic use of Tensor Cores within a CUDA kernel.

1. Compile both applications, `tensorCUBLAS` and `simpleTensorCoreGEMM`, by typing `make`.

2. The tensorCUBLAS code performs a single precision general matrix-matrix multiplication operation. It contains three examples. The first does not use tensor cores, the second demonstrates the TF32 tensor cores on Ampere, and the final example implements a mixed precision GEMM using fp16 inputs. Run the `tensorCUBLAS` code.

3. Read through the source file to see how the library routines are used, referring to the online documentation to aid understanding of cuBLAS.

4. Comment out the TF32 example, reduce the number of repetitions to 3, and then run the code for matrix sizes n=32, 64, 128, 256, 512, 1024, noting a representative time for each run (where n is the size of a row or column, the matrices in this example are square).

5. Plot the timings against n for both examples. Can you describe the scaling behavior? is it what you expect?

6. Run the second code, `simpleTensorCoreGEMM`. It is possible it will report errors between the results obtained by directly using the tensor cores and those obtained from `cuBLAS`.

   If so, this is due to the low accuracy achieved with half precision floating point arithmetic. Increase the acceptable relative error by changing the variable `eps` from `1e-4` to `1e-3`. Re-compile and re-run the code and it should pass the tests and report the execution times for the two versions.

7. Read through the source code `simpleTensorCoreGEMM.cu` to try to understand what it is doing, noting in particular that

   - each warp computes an 16×16 block (or "tile") of the output matrix
   - each thread block has 128×4 threads, which corresponds to a 4×4 collection of warps, which together compute a 64×64 block of the output matrix
   - each warp has to work out the correct offsets for the blocks in its input and output matrices

8. If you are interested in cuBLAS, you might like to write your own code to do something of interest to you, or investigate some of the NVIDIA sample codes for other libraries in Section 4 of

   `https://github.com/NVIDIA/cuda-samples`

   Otherwise, you might prefer to move on to one of the later practicals, or go back to earlier practicals and complete some of the additional tasks you may not have had time to do.