Technische Universität Kaiserslautern
Fachbereich Informatik
RHRK
Dr. habil. Josef Schüle

High Performance Computing with GPUs
Exercise Sheet No. 5

Examples for measuring performance may be found in Sheet 2 Part B, the occupancy and ILP/TLP exercise.

Exercise 1: Multiplying two matrices

Your task is to use the `cublas` library for multiplying two rectangular matrices and to compare the performance of the library call with your kernel from exercise 3. The corresponding library function is called `cublasSgemm` and you should easily find an example of its usage in the internet.

The only functions really required are

```
#include "cublas_v2.h"
cublasCreate(&handle);
cublasSgemm(handle,...);
cublasDestroy(handle);
```

Please bear in mind the layout of the matrices - `cublas` is written in column-major-order, i.e. transposed to the usual C programming style. You may look at options for specifying `cublasOperation_t`.

Start with a small matrix preset with random numbers there you compare the CPU and the GPU output. If everything is ok, you may continue with larger ones (s. below).

Some rules: You want to calculate something like

$$C = \alpha C + \beta A * B$$

Matrix C (m× n) is the result and transposed in the cublas library. The dimensions of A and B have to match, that is A (m× k) and B (k× n). cublas actually does:

$$C^T = \alpha C^T + \beta B^T * A^T$$

Thus, interchanging A and B does the trick and you need no transposition for this task.

Exercise 2: Transpose a matrix

Please write now your own kernel for the transposition of a rectangular matrix. Compare different variations of the transposition as outlined in the lecture (padding , shared memory, . . .)

Exercise 3: Benchmarks Perform a set of tests and comparisons:

- Different sizes. Try 1000, 2000, 4000. Problem complexity is $2n^3$. Look at the scaling of your program.
- Different patterns. Compare $C = AB$, $C = A^T B$, $C = AB^T$ and $C = C + AB$.