Tutorial: HPC - Algorithms and Applications
WS 13/14

Complete the following assignments (alone or in a group), and send your source code via e-mail to meistero@in.tum.de until Sunday, December, 8th 2013.

Worksheet 3: Coalesced Access, Sparse Linear Algebra

Assignment 1: Recap on Coalesced Access

Let $n = 64; \; tx = \text{threadIdx.x}; \; ty = \text{threadIdx.y}; \; tz = \text{threadIdx.z}$; Are accesses to the array $\text{float } A$ in the following calls uncoalesced, partially coalesced or coalesced (chipset: NVidia Fermi, CUDA $\text{cc } \geq 2.0$)? Answer shortly.

a) $\text{float } f = A[tx];$

b) $\text{float } f = A[tx + 1];$

c) $\text{float } f = A[2 * tx];$

d) $\text{float } f = A[tx / 2];$

e) $\text{float } f = A[n * tx];$

f) $\text{float } f = A[ty];$

g) $\text{float } f = A[(tx * n + ty) * n + tz];$

h) $\text{float } f = A[(tz * n + ty) * n + tx];$

Assignment 2: CSR kernel

Write a CSR matrix-vector multiplication kernel for the PageRank example code.

a) Open kernels.cu, define grid and block size

b) Implement the $k_{csr\_mat\_vec\_mm}$ kernel:

i) Assign one thread to a matrix row
ii) Compute one row × vector product in a loop.

iii) Add the result to the output vector.

c) Compile the code using make. You will have to choose a suitable C compiler (gcc is the default in Makefile).

d) Try the kernel on a small matrix (mtx/my.mtx). The program output should be:

\[
\begin{align*}
    x_1 &= 3.602992 \times 10^{-1} \\
    x_2 &= 2.700142 \times 10^{-2} \\
    x_3 &= 6.238353 \times 10^{-2} \\
    x_4 &= 2.431707 \times 10^{-2} \\
    x_5 &= 4.042290 \times 10^{-1} \\
    x_6 &= 2.700142 \times 10^{-2} \\
    x_7 &= 2.431707 \times 10^{-2} \\
    x_8 &= 5.378454 \times 10^{-2} \\
    x_9 &= 1.666666 \times 10^{-2}
\end{align*}
\]

e) Next, test the kernel on a bigger matrix. Download for example flickr.mtx from http://www.cise.ufl.edu/research/sparse/MM/Gleich and run the PageRank algorithm on the matrix (if it’s too big choose a different matrix). Which page is the most relevant according to the algorithm?

**Assignment 3: Vectorized CSR kernel**

Write a vectorized CSR matrix-vector multiplication kernel for the PageRank example code.

a) Implement the kcsr2_mat_vec_mm kernel in kernels.cu:

   i) Assign one warp to a matrix row

   ii) Allocate a shared array vals[] for the partial results of a block

   iii) Compute one row × vector product in a loop. This time, parallelize the loop over all 32 threads in the warp. Take care that access to the arrays indices and data is coalesced.

   iv) Use a reduction of some kind (ideally: binary fan-in) to add up the partial sums in vals[] and add the output to the result vector.

b) Try the kernel on mtx/my.mtx again and check if the output is consistent with Assignment 2d.

c) Test both CSR kernels on the big matrix and measure execution times (time ./sparse mtx/flicker.mtx). How does performance compare?