Tutorial: HPC - Algorithms and Applications
WS 14/15

Complete the following assignments (alone or in a group), and send your source code via e-mail to meistero@in.tum.de until Sunday, November, 30th 2014.

Worksheet 3: Coalesced Access, Sparse Linear Algebra

T3.1: Recap on Coalesced Access

Consider this CUDA kernel call:

```c
dim3 grid(8, 8, 1); dim3 block(32, 32, 1);
kernel<<<grid, block>>>(A);

__global__ void kernel(float* A) {
    int tx = threadIdx.x, ty = threadIdx.y, tz = threadIdx.z;
    const int n = 64;
    float f;
    /* set value of f here */
}
```

For each of the following instructions, answer shortly if access to the array A is uncoalesced, partially coalesced or coalesced (chipset: NVidia Fermi, CUDA cc ≥ 2.0).

a) \( f = A[tx] \);
b) \( f = A[(tx * n + ty) * n + tz] \);
c) \( f = A[tx + 1] \);
d) \( f = A[ty] \);
e) \( f = A[2 * tx] \);
f) \( f = A[n * tx] \);
g) \( f = A[tx / 2 + 16] \);
h) \( f = A[ty * tx] \);
T3.2: CSR kernel

a) Write a CSR matrix-vector multiplication kernel for the PageRank example code.
   - Define grid and block size in kernels.cu
   - Implement the k_csr_mat_vec_mm kernel:
     i) Assign a matrix row to each thread
     ii) Compute the row $\times$ vector product in a loop.
     iii) Add the result to the output vector.
   - Compile the code using make. You will have to choose a suitable C compiler (gcc is the default in Makefile).

b) Try the kernel on a small matrix (mtx/my.mtx). The program output should be:
   
   $x_1 = 3.602992e-01$
   $x_2 = 2.700142e-02$
   $x_3 = 6.238353e-02$
   $x_4 = 2.431707e-02$
   $x_5 = 4.042290e-01$
   $x_6 = 2.700142e-02$
   $x_7 = 2.431707e-02$
   $x_8 = 5.378454e-02$
   $x_9 = 1.666666e-02$

c) Next, test the kernel on a bigger matrix. Download flickr.mtx or usroads.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?

H3.1: Vectorized CSR kernel

a) Write a vectorized CSR matrix-vector multiplication kernel for the PageRank example code.
   - Implement the k_csr2_mat_vec_mm kernel in kernels.cu:
     i) Assign a matrix row to each warp now
     ii) Allocate a shared array vals[] for the partial results of a block
     iii) Compute one row $\times$ 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 new kernel on mtx/my.mtx and check if the output is consistent with T3.2b

c) Test both CSR kernels on the big matrix from T3.2c and measure execution times (time ./sparse mtx/flicker.mtx). How does performance compare?