Technische Universität München
Lehrstuhl Informatik V
M. Bader, O. Meister

WS 2013/14
Worksheet 2
November, 13th 2013

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, November, 24th 2013.

Worksheet 2: Roofline Model, Profiling and Coalesced Access

Assignment 1: Roofline Model and Hardware Profiling

Create a roofline model for the matrix multiplication kernels.

a) Create a graph with two logarithmic axes for operational intensity (x-axis) and floating point performance (y-axis). Look up peak Flop/s and memory bandwidth of the hardware you use (chair machine: NVidia Quadro NVS 290) and draw the respective roofline into the graph.

b) Include ceilings into the model: Draw a line that represents the peak performance for uncoalesced memory access and the peak performance for serialized kernel code.

c) Draw lines that represent the basic matrix multiplication and tiled matrix multiplication kernels and mark the measured performance with a point in the graph. You may assume that `TILE_SIZE` $= 32$. Which performance optimization for the kernel seems to be the most feasible? Maximize 1) operational intensity, 2) memory bandwidth or 3) floating point performance?

d) Use the cuda profiler to check the performance of the kernels implemented so far. Find the bottleneck by using appropriate performance counters.

Assignment 2: Coalesced memory access

Reduce the number of memory accesses in the matrix multiplication kernel by using coalesced accesses to global memory.

a) What has to be changed in the tiled kernel in order to gain coalesced accesses to global memory and shared memory?

b) Implement the fuction `matrixMultKernel_coalesced` by modifying the memory access pattern of the tiled kernel accordingly.

c) Measure the kernel's performance and add the result to the roofline model from assignment 1.

Assignment 3: Prefetching

Overlap computation and memory access in the matrix multiplication kernel in order to hide global memory latency.

a) Implement `matrixMultKernel_overlapped` with the following steps:

   i) Load the first tile into registers.

   ii) For all tiles except the last one: copy current tile from the registers to shared memory, load the next tile into registers and compute the current tile with the data in shared memory.

   iii) Compute the last tile with the data in the registers.

b) Again, measure the kernel's performance and add the result to the roofline model.

c) (optional) Try to reach peak performance! Possible optimizations: vectorization, loop unrolling, adjustment of thread granularity, ...