Technische Universität München
Lehrstuhl Informatik V
M. Bader, A. Pöppl

WS 2016/17
Worksheet 2
November 23rd 2016

Tutorial: HPC - Algorithms and Applications WS 16/17

Complete the following assignments (alone or in a group), and send your source code via e-mail to `poeppl@in.tum.de` until Sunday, November, 27th 2016.

Worksheet 2: Roofline Model, Profiling and Coalesced Access

T2.1: Roofline Model and Hardware Profiling

a) Create a roofline model for the matrix multiplication kernels.

- 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 MAC Cluster (your own system) and draw the respective roofline into the graph.
- Include a ceiling into the model: Draw a line that represents the peak performance for uncoalesced memory access
- Find the computational intensity for basic matrix multiplication and tiled matrix multiplication kernels and mark the measured performance with a point in the graph for $\texttt{TILE\_SIZE} = 4, 8, 16, 32$. Which performance optimization for the kernel seems to be the most feasible? Maximize 1) operational intensity, 2) memory bandwidth or 3) floating point performance?

T2.2: Coalesced memory access

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

   - Analyze the tiled kernel for possible uncoalesced accesses to global memory and shared memory
   - Implement the fuction `matrixMultKernel_coalesced` by modifying the tiled kernel accordingly.

b) Measure the kernel performance and add the result to the roofline model from T2.1.

H2.1: 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) Measure the kernel's performance and add the result to the roofline model from T2.1.

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