Heidelberg University
Institute for Computer Engineering (ZITI)

Master of Science Computer Engineering

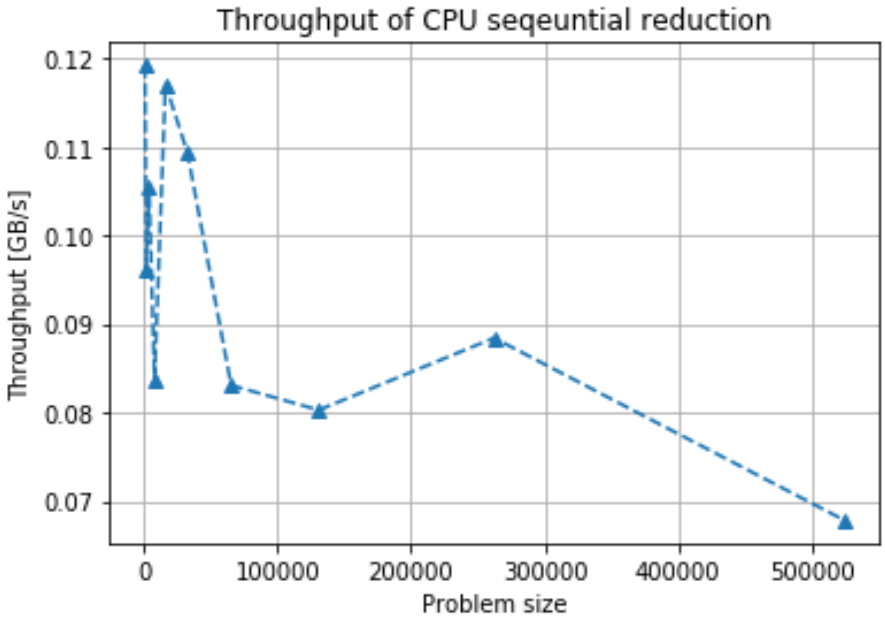
GPU Computing

Exercise 6

gpucomp03

Benjamin Maier
Daniel Barley
Laura Nell

Due date January 12th, 09:00



6.1 Reading

Roofline: An Insightful Visual Performance Model for Multicore Architectures

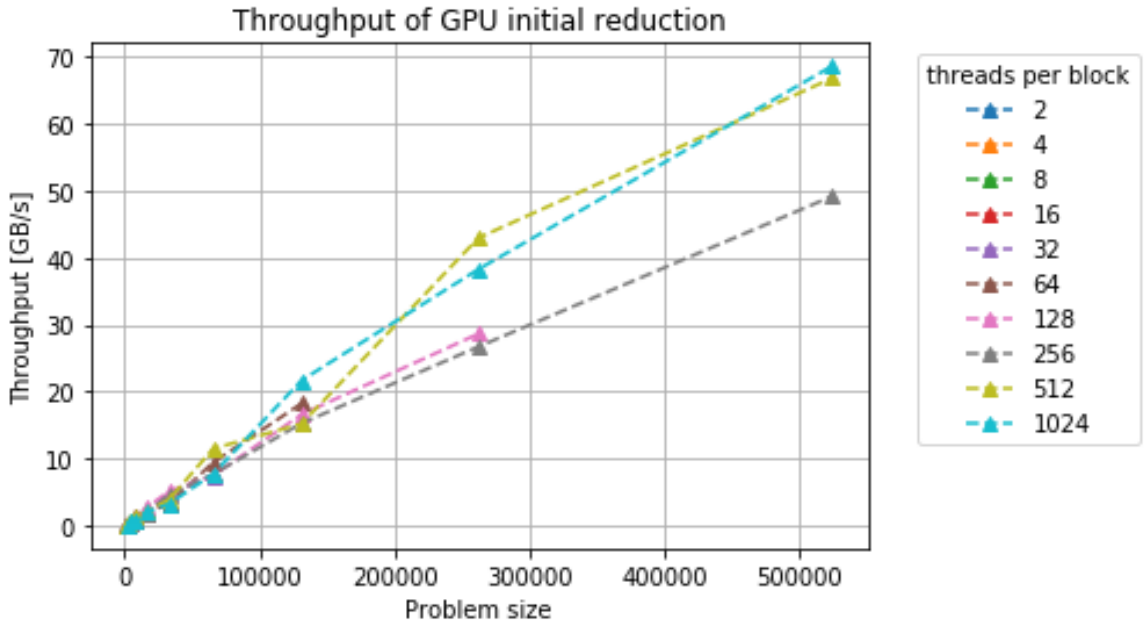
To address the increasing variety in microprocessors, the Roofline model was introduced to offer cross-platform understandable performance guidelines using bound and bottleneck analysis. It ties together floating-point performance, operational intensity and memory performance in a 2D-graph. After explaining the function of the Roofline model, the paper demonstrates its utility on four diverse multicore computers, then optimizing four floating-point kernels taken from the Seven Dwarfs before clearing up some fallacies.

The Roofline model sets an upper bound on performance depending on the kernel's operational intensity, thus showing if performance of the kernel is compute-bound or memory-bound and, by adding ceilings, which optimizations may be necessary.
Finally, the ridge point is a better predictor for performance than clock rate or peak performance and indicates when a computer is imbalanced.

The Roofline model is still used today to check for a kernel's peak performance, although one has to especially consider to choose the right metric. Therefore, we accept the given paper and its content.

6.2 Reduction - CPU sequential version

To start with the reduction exercise a sequential cpu version has been implemented. The bandwidth is shown in the figure 6.2. As one can see the version does not scale with the problem size, which might occurs from bad caching effects.



6.3 Reduction - GPU parallel initial version

For the next exercise we implemented a simple reduction alogirthm on the gpu. The bandwith plot is shown in figure 6.3. However, due to the parallelization, the throughput scales well with the problem size, where the best results can be achieved with a high amount of blocks per thread.

6.4 Reduction - GPU parallel optimized version

To optimize the gpu implementation we used a full loop unroll technique and doing the first reduction during the load. This should be the same idea as reduction version 6 from the lecture. The bandwith is shown in 6.4. However, the "optimized" version is not as good as the initial version. Sadly we were unable to discover the error. Maybe some more measurements would vanish some statistically side effects.

6.5 Willingness to present

Hereby, we declare our will to present the results shown in the former sections.


Throughput of GPU optimized reduction
Throughput [GB/s]
Problem size
0
10
20
30
40
50
0
100000
200000
300000
400000
500000
threads per block
2
4
8
16
32
64
128
256
512
1024