Universität Heidelberg
Institute for Computer Engineering (ZITI)

Master of Science Computer Engineering

GPU Computing

Exercise 6

Group gpu04
Altuntop, Ekrem
Pingitzer, Danny
Junge, Andreas G.

Due date December 11th, 09:00

6 Exercise

6.1 Reading

Read the following paper and provide review as explained in the first lecture (see slides):

Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: an insightful visual performance model for multicore architectures. Commun. ACM 52, 4 (April 2009), 65-76.

The article "Roofline: an insightful visual performance model for multicore architectures." by Samuel Williams, Andrew Waterman, and David Patterson is presenting a performance model, "Roofline", to estimate performance of a kernel on a certain system (multicore, accelerators). The proposed Roofline model ties together floating-point performance, operational intensity, and memory performance in a 2D graph. Furthermore it can show inherent hardware limitations for a given kernel and unveil benefit of optimizations.
It relies on the concept of Operational Intensity, this is the ratio of total floating-point operations to total data movement (bytes).
The performance in the model is bound by the peak flop rate and the streaming bandwidth, so basically the memory and compute boundness of a system. At the ridge point, the intersection of the diagonal (bandwith) and the horizontal (peak flop) roof, the x-coordinate is the minimum operational intensity required to achieve maximum performance and also a hint to the level of difficulty to achieve peak performance.
Since the model also takes regard to the memory boundness, we can also estimate, how hardware changes will affect the performance of our kernels.

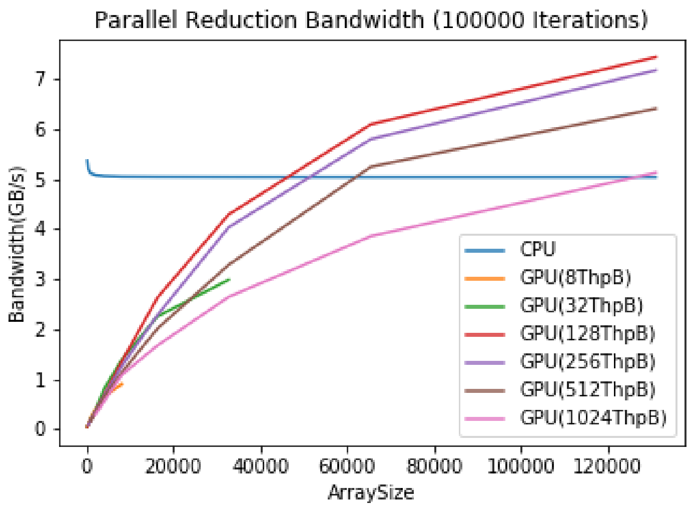
I myself am not an expert in performance evaluation models, but the proposed model seems reasonable. It fits in the current computing era better than many older models, and is widely accepted in the HPC community. Still, older models, such as Amdahl's Law are still relevant.

Review: Accept.

6.2 CPU sequential

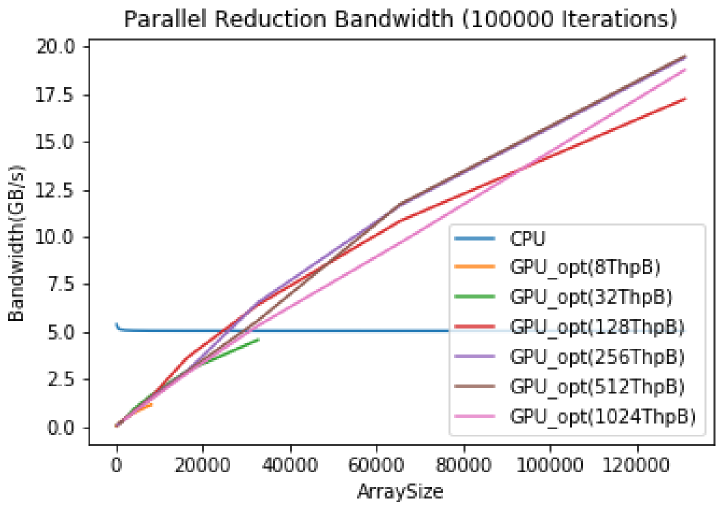
The bandwidth of the sequential CPU version is mostly very constant over a large range of data sizes. There seems to be a very small drop-off the larger the array gets and there is a small performance boost for very small matrices. (Figure 1,2 and 3)

6.3 GPU Parallel Initial Version

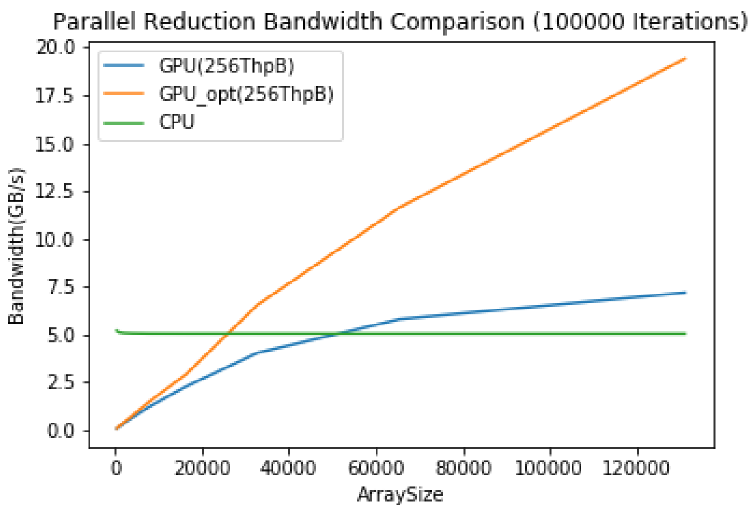


The performance of the initial GPU version is worse than the CPU version for small arrays, but scales better for larger array sizes. The scaling is clearly below linear. There is a large difference for ThpB counts with 128 being the best and 1024 being the worst. 32 and 8 ThpB could not be measured very far, because the second kernel call blocksize needs to be smaller than 1024.

6.4 Optimized Parallel Version



The optimized parallel version has much better performance, it almost scales linearly up to 100000 elements, there is much less of a variance between the ThpB curves, and it outperforms the CPU slightly earlier. The optimal threadcount seems to be between 256 and 512 threads.



Both GPU programs start at similar points for small gridsizes but diverge rapidly.