Sorting Large Data Sets with FPGA-Accelerated Samplesort

Han Chen, Sergey Madaminov, Michael Ferdman, Peter Milder
Stony Brook University
Email: {han.chen.2, peter.milder}@stonybrook.edu,{smadaminov, mferdman}@cs.stonybrook.edu

***Abstract*—Sorting is a fundamental operation in many applications such as databases, search, and social networks. Although FPGAs have been shown effective at sorting data sizes that fit on chip, systems that sort larger data sets by shuffling data on and off chip are typically bottlenecked by costly merge operations or data transfer time.**

We propose a new approach to sorting large data sets by accelerating the samplesort algorithm using a server with a PCIe-connected FPGA. Samplesort works by randomly sampling to determine how to partition data into approximately equal-sized non-overlapping "buckets," sorting each bucket, and concatenating the results. Although samplesort can partition a large problem into smaller ones that fit in the FPGA's on-chip memory, partitioning in software is slow. Our system uses a novel parallel hardware partitioner that is only limited in data set size by available FPGA hardware resources. After partitioning, each bucket is sorted using parallel sorting hardware. The CPU is responsible for sampling data, cleaning up any potential problems caused by variation in bucket size, and providing scalability by performing an initial coarse-grained partitioning when the input set is larger than the FPGA can sort.

We prototype our design using Amazon Web Services FPGA instances, which pair a Xilinx Virtex UltraScale+ FPGA with a high-performance server. Our experiments demonstrate a 17.1x speedup over GNU parallel sort when sorting 2^{23} key-value records and a speedup of 4.2x when sorting 2^{30} records.

At datacenter scale, there is a constant need for large-scale sorting operations, with performance and cost of running sort being a major consideration of application design. Most previous accelerator designs for sorting have focused on building high throughput hardware structures [1]. However, the amount of data that can be sorted in hardware by these approaches is limited by the on-chip memory size. Other approaches [2] aim to use hardware to accelerate merging sorted subsequences of data stored in off-chip memory. However prior approaches in this area have high hardware cost and limitations in the number of buckets that can be merged concurrently, necessitating many round-trips from/to off-chip DRAM.

In this work, we present a new PCIe-based FPGA accelerator design to implement the samplesort algorithm. The key to our design is a novel parallel hardware partitioner that partitions data into approximately equally-sized buckets of non-overlapping data (i.e., where all values in the ith bucket are guaranteed to be less than all values in the $(i+1)$th bucket, but values within a bucket are not necessarily sorted). The buckets are stored by the partitioner in the off-chip memory accessible to the FPGA, and then read back from off-chip memory by a parallel sorting network [1] that writes the final sorted sequence back to the host over PCIe. Our implementation overcomes a number of challenges, which we address through cooperation between the FPGA and software running on the host CPU. First, we use software to randomly sample data to determine how to split the data such that buckets have approximately equal size. Second, although the sampling approach produces equal-size buckets with high probability, our system detects and "fixes" oversized outlier buckets in the rare event when they occur. Lastly, the software system provides scalability; even with the efficient partitioning hardware design, the size of data that the FPGA can sort at once is still limited by the available FPGA resources. To allow sorting larger data sets, the CPU is used to perform an initial coarse-grained partitioning before sending data to the FPGA for further partitioning and sorting.

Our approach, based on partitioning, is fundamentally more scalable than prior approaches based on merging sorted subsequences. In the partitioning process, each record is independent, whereas the merging process must compare each value to those around it. This independence allows us to construct an inexpensive and high-throughput partitioner that can split data among a large number of buckets. For example, our prototype partitioner splits data into 2,048 buckets in a single pass over the data, while a 32-way merging network would require three round trips to memory to merge 2,048 buckets.

We implemented and tested a prototype of our sorter on an Amazon AWS F1 FPGA instance, demonstrating a 17.1x speedup over GNU parallel sort when sorting 2^{23} key-value records and a speedup of 4.2x when sorting 2^{30} records. In a cost comparison, we found that the higher per-hour cost of the FPGA instance is justified by the final speedup obtained, meaning that the FPGA-accelerated system is both faster and more cost effective. Finally, when compared to a much larger CPU system (with 64 hardware threads), we found that our FPGA sorter still yields speedups of 3.8x for 2^{23} records and 2.4x for 2^{30} records.

References

[1] M. Zuluaga, P. Milder, and M. Püschel, "Streaming Sorting Networks," *ACM Transactions on Design Automation of Electronic Systems*, vol. 21, no. 4, May 2016.

[2] M. Saitoh, E. A. Elsayed, T. V. Chu, S. Mashimo, and K. Kise, "A High-Performance and Cost-Effective Hardware Merge Sorter without Feedback Datapath," in *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Apr 2018.