Using Integrated Processor Graphics to Accelerate Concurrent Data and Index Structures

Poster Summary

JOEL FUENTES, University of California, Irvine

Graphics applications are used in every personal electronic device, such as laptops, tables, smartphones, etc. Most of these devices have, in a form of System on a Chip (SoC), an integrated GPU (iGPU) that performs graphics tasks. Furthemore, the main chip manufactures have improved considerably their last generation of iGPUs, with compute capabilities that approach TeraFLOPS performance. In this project we study concurrent and index structures that can benefit from the iGPU architecture to accelerate their processing. Our special focus are data and index structures that are commonly used by applications such as databases, search engines, file systems, and so on. We propose a new lock-free Skiplist, which is a concurrent data structure that supports search and update operations. We also introduce a general algorithm to construct and query compressed index structures based on *rank* and *select* operations, which was tested with three well-known index structures: K^2-tree, Wavelet tree and Suffix tree. The motivation of this work is to exhibit that concurrent data and index structures can benefit from the iGPU architecture and computational model, and important performance gains and energy savings can be obtained when running data structure workloads on the iGPU instead of CPU cores.

Skiplists are popular in concurrent algorithms, as they offer a probabilistic alternative to balanced search trees without costly balancing operations and still providing $O(logn)$ for search and update operations. We take advantage of SIMD16 capabilities of the GenX architecture to design a Skiplist based on chunked lists. So instead of having nodes and a list of individual updates when inserting or deleting keys, entire chunks are updated using SIMD16 atomic operations. Thus, the parallelism of this data structure is achieved at thread and instruction level (data-parallel through SIMD). A detailed proof of correctness using linearization points establishes that the proposed Skiplist guarantees the lock-freedom property for the insert and delete operations, and wait-free property for its search operation.

Experimental tests were carried out in order to measure the performance of combined concurrent operations using the proposed Skiplist. Additionally, similar operations were performed on state-of-the-art concurrent Skiplist for CPU. The proposed Skiplist for iGPU (CMSL) achieves higher number of operations for all the scenarios, being its best performance when all the operations are search, with 4.3x speedup. When all the operations are updates, it still presents 1.5-2x speedup. We also measured the energy savings that are achieved by CMSL on iGPU, showing up to 300% of energy savings over the same experiments on CPU cores.

On the other hand, index structures are auxiliary structures designed to speed up the search for records in different applications, e.g. databases use different types of index structures to answer quickly different types of queries. While the search in these structures is fast, constructing the index structures is a time-consuming task. We propose a general algorithm (named CMIB) to construct compressed index structures as well as algorithms to perform the actual queries on iGPU. Most of the index structures are in a tree-like form. As such, there exist dependencies between levels that prevent parallelism through levels. So they are built level by level in top-down or bottom-up manner. CMIB considers this restriction but takes advantage of large levels that are parallelized by threads, where every thread also performs

Author's address: Joel Fuentes, University of California, Irvine, P.O. Box 1212, Irvine, USA, 43017-6221, joel.fuentes@uci.edu.

data-parallel operations through SIMD instructions. In general, the logic behind CMIB is an iterative algorithm that dispatches iGPU kernels depending on the input size. If there are pending levels after every iteration, the algorithm starts over dispatching a new kernel until all the levels are processed.

Three well-known index structures were selected to test the CMIB algorithm: K^2-tree which represents webgraphs in compressed space, Wavelet tree which is used to index text/string and Suffix tree that is also used for text/strings. We measured and compared the performance of CMIB with sequential and multi-threaded implementations for CPU. CMIB delivers less construction time for all the scenarios where the number of edges is greater than 100 thousand. This validates the general logic behind of CMIB, which will only dispatch iGPU kernels if the input size is big enough. In general, the greater the input size is, greater is the difference of construction time between iGPU and CPU. In summary, CMIB achieved 7x speedup when constructing the K^2-tree of size 60 millions edges. In terms of concurrent queries, it achieves 11x speedup over the CPU with same structure size.