Sparta: High-Performance, Element-Wise Sparse Tensor Contraction on Optane-based Heterogeneous Memory

Jiawen Liu*, Jie Ren*, Roberto Gioiosa♮, Dong Li*, Jiajia Li♮
{jliu265,jren6,dli35}@ucmerced.edu,{roberto.gioiosa,Jiajia.Li}@pnnl.gov
* University of California, Merced
♮ Pacific Northwest National Laboratory

1 Motivation and Contribution

Tensors, especially those high-dimensional sparse tensors are attracting increasing attentions, because of their popularity in many applications. High-order sparse tensors have been studied well in tensor decompositions on various hardware platforms with a focus on the product of a sparse tensor and a dense matrix or vector. Nevertheless, the two-sparse-tensor contraction (SpTC), foundation for a spectrum of applications, such as quantum chemistry, quantum physics and deep learning, are still lack of sufficient research, especially with element-/pair-wise sparsity. In essence, SpTC, a high-order extension of sparse matrix-matrix multiplication (SpGEMM), multiplies two sparse tensors along with their common dimensions. Efficient SpTC introduces multiple challenges.

First, the size and non-zero pattern of the output tensor are unknown before computation. Thus, memory allocation for the output tensor is difficult. Unlike operations such as a sparse tensor multiplying a dense matrix/vector where the size of the output data is predictable, the output tensor of an SpTC is usually sparse and the non-zero pattern (e.g., the number of non-zero elements and their distribution) is unpredictable before the actual computation. Sparse data objects and unpredictable output size also exist in SpGEMM. Two popular approaches have been proposed to solve these issues for SpGEMM while are not efficient for SpTC. The first approach, using an extra symbolic phase to predict the accurate output size and non-zero pattern, suffers from expensive pre-processing and is unaffordable in a dynamic sparsity environment. This issue is especially severe in SpTC, because an SpTC with the exactly same input is usually computed only once in a long sequence of tensor contractions. However, with the symbolic approach, every SpTC is attached to both the symbolic phase and SpTC computation, which is very expensive, especially for large applications. The second approach makes a loose upper-bound prediction on the memory consumption of the output tensor, which wastes memory space. On the other hand, a tight prediction for SpTC of high-order tensors is very difficult because the more contract dimensions of SpTC make the prediction less accurate based on the existing prediction algorithms.

Second, irregular memory accesses along with multi-dimensional index search to the second input tensor and accumulator introduce performance problems. Similar to SpGEMM, SpTC has indirect memory accesses to the second input tensor, caused by the non-zero indices of the first input tensor. Take an SpGEMM $\mathbf{C} = \mathbf{A} \times \mathbf{B}$ as an example. A non-zero $\mathbf{A}(0,1)$ gets, e.g. $\mathbf{B}(1,1)$, to perform multiplication; while $\mathbf{A}(0,10)$ computes with, e.g. $\mathbf{B}(10,2)$. Those irregular memory accesses of $\mathbf{B}$ and the sparse accumulator, which happen more often with the high-dimensional tensors, are not cache friendly. In addition, index search and accumulator, which is used to address irregular memory accesses in SpTC, is more expensive than that in SpGEMM. Our evaluation shows that they takes 54% of SpTC performance on average.

Third, massive memory consumption caused by large input and output tensors and intermediate results creates pressure on the traditional DRAM-based machine. Sparse tensors from real-world applications easily consume a few to dozens of GB memory, while the output tensor could be even larger, because it might contain more non-zero elements than any of the input sparse tensor. The intermediate results could be large as well, especially for multi-threading environment where each thread has its own intermediate results. Compared to the well-studied "sparse tensor times dense matrices/vectors", SpTC results in substantial memory consumption easily, which can be beyond typical DRAM capacity (up to a few hundreds of GB) on a single machine. However, expanding DRAM capacity is not cost effective, while adding cheap but much slower SSD causes significant performance drop. This memory capacity problem is becoming more serious in those HPC applications with increasing dimension size in tensors.

To address the first two challenges, we propose Sparta with performance optimizations executed in five stages: input processing, index search, accumulation, writeback, and output sorting. In particular, we employ dynamic arrays to accurately allocate memory space for the accumulator and output tensor to avoid the unknown output challenge. For multi-threading, we introduce a thread-private, dynamic object to store the output tensor from each thread for better parallelization. To address the irregular memory access challenge, we perform permutation and sorting on input sparse

tensors before computation, thus significantly improve temporary locality of non-zeros in the first input tensor and spacial locality of non-zeros in the second input tensor. Furthermore, we adopt a hash table-based approach based on a large-number representation for the second tensor and accumulator to significantly speed up the process of multi-dimensional search in SpTC. With the above optimizations, Sparta substantially outperforms the traditional SpTC algorithm extended from SpGEMM. By evaluating real data from quantum chemistry and physics, our element-wise Sparta beats their block-wise sparse algorithms (calling BLAS for dense block computations) by 7.1× on average.

To address the third challenge, we explore the emerging persistent memory-based heterogeneous memory (HM). In particular, recent Intel Optane DC Persistent Memory Module (PMM) provides bandwidth and latency slightly inferior to that of DRAM but with only half of the price. PMM often pairs with a small DRAM to build HM, where frequently accessed data objects placed in DRAM and the rest resided in PMM with several TBs of large memory capacity. It is performance-critical to decide the placement of data objects of SpTC (input and output tensors and intermediate results) on PMM-based HM, to make best use of DRAM's high bandwidth and low latency without causing frequent data movement between PMM and DRAM. We first characterize memory read/write patterns associated with those data objects in SpTC, and reveal the performance sensitivity of SpTC to the placement of those data objects on PMM and DRAM. Sparta then prioritizes the data placement between DRAM and PMM statically based on our knowledge on the SpTC algorithm and characterization of data objects for best performance. Sparta effectively avoids unnecessary data movement suffered in the traditional application-agnostic solutions (such as hardware-managed DRAM caching or software-based page hotness tracking).

Sparta has been accepted into the prestigious conference, PPoPP 2021 [2] and is open-source[1]. Our main contributions are summarized as follows.

- We introduce the first, high-performance SpTC system for arbitrary-order element-wise sparse tensor contraction, named Sparta.
- We explore the emerging PMM-based HM to address memory capacity limitation suffered in the traditional tensor computations.
- Evaluated with 15 datasets, Sparta brings $28\times - 576\times$ speedup over the traditional SpTC with SPA. With our proposed algorithm- and memory heterogeneity-aware data management, Sparta brings extra performance improvement on HM built with DRAM and PMM over a state-of-the-art software-based data management solution, a hardware-based data management solution, and PMM-only by 30.7% (up to 98.5%), 10.7% (up to 28.3%) and 17% (up to 65.1%) respectively.

2 Data Placement on Optane-based Heterogeneous Memory

We characterize memory accesses of major data objects of Sparta, in terms of access patterns (sequential/random and read/write). We gain three interesting observations. First, performance difference between read and write matters a lot to performance of Sparta. For example, for data objects with sequential read-only access pattern, placing them on PMM causes ignorable performance loss. In contrast, placing data objects with sequential write-only on PMM causes large performance loss on PMM. Second, sequential and random accesses behaves very differently in performance. For instance, placing sequential read-only data objects on PMM causes negligible performance loss, while placing data objects with random read-only access pattern causes large performance loss on PMM. Third, the performance of Sparta is not sensitive to the placement of some data objects on PMM. For example, placing some data objects in Sparta on PMM has ignorable performance loss, because of the memory access patterns discussed in the above two observations.

Sparta shows that static data placement on heterogeneous memory can outperform dynamic data migration, which is unprecedented. In particular, some data objects are not sensitive to their position (either on PMM or DRAM), because of their read-only sequential memory access pattern. The dynamic data migration solutions unnecessarily migrate them to DRAM for high performance. Furthermore, dynamic data migration solutions cannot effectively capture memory access patterns of some data objects because memory access patterns associated with some data objects might frequently change across different execution stages in Sparta. Thus, the dynamic data migration solutions cause unnecessary data migration. Moreover, workload characterization and modeling employed in our static data placement solution can prioritize the data placement on DRAM to make best use of limited DRAM space.

Sparta is the first effort to provide high-performance, element-wise and arbitrary-order sparse tensor contraction on PMM-based Heterogeneous Memory. Sparta can be applied to software from diverse domains, including but not limited to NWChem for computational chemistry, ITensor for physics, and TensorNetwork for deep learning [1].

References

[1] Lingjie Li, Wenjian Yu, and Kim Batselier. Faster tensor train decomposition for sparse data. *arXiv preprint arXiv:1908.02721*, 2019.

[2] Jiawen Liu, Jie Ren, Roberto Gioiosa, Dong Li, and Jiajia Li. Sparta: High-performance, element-wise sparse tensor contraction on heterogeneous memory. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 2021.

[1] https://github.com/PASAUCMerced/sparta