HiCOO: A Hierarchical Sparse Tensor Format for Tensor Decompositions

Jiajia Li, Jimeng Sun, Richard Vuduc
Computational Science and Engineering, Georgia Institute of Technology
Atlanta, GA 30332
jiajiali@gatech.edu

ABSTRACT

This paper proposes a new Hierarchical COOrdinate (HiCOO) format for sparse tensors, which compresses its indices to units of sparse tensor blocks. HiCOO format does not favor one tensor mode over the others, thus can be used as a replacement of the traditional COOrdinate (COO) format. In this paper, we use HiCOO format for the Matriced Tensor Times Khatri-Rao Product (Mttkrp) operation, the most expensive computational core in the popular CANDECOMP/PARAFAC decomposition, then accelerate it on multicore CPU architecture using two parallel strategies for irregular shaped tensors. Parallel Mttkrp using HiCOO format achieves up to 3.5× (2.0× on average) speedup over COO format and up to 4.3× (2.2× on average) speedup over CSF format.

1 INTRODUCTION

A tensor of order N is an N-way array, which can provide a natural input representation of a multiway dataset. This tensor is sparse if it consists of mostly zero entries. [1] There are several techniques for analyzing and mining a dataset in the tensor form [7, 8, 12, 13, 15, 17, 19, 21], which have been applied in a variety of domains, including healthcare [9, 22], natural language processing [10], machine learning [1, 2], and social network analytics [18], among others.

This paper concerns performance enhancement techniques for one of the most popular of such tensor methods, the CANDECOMP/PARAFAC decomposition (CPD) [12]. The runtime of a typical CPD on an N^{th}-order tensor is dominated by the evaluation of N *matricized tensor times Khatri-Rao product* (Mttkrp) operations [4, 6, 10, 11, 14, 16, 21]. Some tensor formats have been proposed to compress the storage space for general sparse tensors. Compressed Sparse Fiber (CSF) format proposed in [21] is a hierarchical, fiber-centric format that effectively extends the Compressed Sparse Row (CSR) format of sparse matrices to sparse tensors. A recently proposed Flagged-COOrdinate (F-COO) format uses two flag arrays, bit-flag and start-flag, to replace index mode(s), thus its construction highly depends on the mode performing Mttkrp. Both CSF and F-COO formats are mode-sensitive because the same tensor operation in different modes need to store separate CSF/F-COO representation for each mode. [2] Therefore, neither CSF nor F-COO can replace COO format in sparse tensor applications.

[1] For instance, an $I \times I \times I$ tensor is sparse if its number of non-zero elements, m, satisfies $m \ll I^3$. Indeed, one typically expects $m = \mathcal{O}(I)$.

[2] For a third-order tensor, using less than three CSF representations to support all the three Mttkrps is possible but with some performance payoff.

We propose HiCOO format to fulfill this need by efficiently compressing a sparse tensor, maintaining the "mode-symmetric" property [3], and conserving all tensor information as in COO format. In this work, we make the following main contributions:

- We propose a new sparse tensor format, Hierarchical COOrdinate (HiCOO[3]), which largely compresses tensor indices in units of sparse tensor blocks under Z-Morton order for tensors with appropriate block locality. Since HiCOO format preserves all information of a sparse tensor, only one HiCOO representation is needed in tensor algorithms. (§ 3)
- We further accelerate the Matriced Tensor Times Khatri-Rao Product (Mttkrp) operation using HiCOO format on multicore CPU architecture. With a bulk scheduler and two parallel strategies for irregular shaped tensors, parallel Mttkrp exhibits better thread scalability. (§ 4)
- Overall, parallel Mttkrp in a single mode using HiCOO format achieves up to 3.5× (2.0× on average) speedup over COO format and up to 4.3× (2.2× on average) speedup over CSF format. (§ 5)

2 BACKGROUND

The order N of an N^{th}-order tensor is sometimes referred to as the number of *modes* or *dimensions*. Higher-order tensors ($N \geq 3$) are denoted by bold capital calligraphic letters, e.g., $\mathcal{X}$. A scalar element at position (i, j, k) of a tensor $\mathcal{X}$ is x_{ijk}.

The most expensive computation kernel of the CP decomposition is the Matriced Tensor Times Khatri-Rao Product (Mttkrp). For an N^{th}-order tensor $\mathcal{X}$ and given matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}$, the mode-$n$ Mttkrp is

$$\tilde{\mathbf{A}}^{(n)} \leftarrow \mathbf{X}_{(n)} \left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)} \right), \quad (1)$$

where $\mathbf{X}_{(n)}$ is the mode-n matricization (or unfolding) of tensor $\mathcal{X}$, and $\odot$ is the Khatri-Rao product. Matricization reshapes a tensor into an equivalent matrix by arranging all mode-n fibers to be the columns of a matrix. (Readers may refer to Kolda and Bader's survey for more details [12].) The Khatri-Rao product is a "matching column-wise" Kronecker product between two matrices. Given matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$ and $\mathbf{B} \in \mathbb{R}^{J \times R}$, their Khatri-Rao product is denoted by $\mathbf{C} = \mathbf{A} \odot \mathbf{B}$ where $\mathbf{C} \in \mathbb{R}^{(IJ) \times R}$, or

$$\mathbf{C} = \mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_R \otimes \mathbf{b}_R], \quad (2)$$

where $\mathbf{a}_r$ and $\mathbf{b}_r$, $r = 1, \ldots, R$, are columns of $\mathbf{A}$ and $\mathbf{B}$.

For a sparse Mttkrp operation which refers to an Mttkrp with sparse tensors and dense factor matrices, its computation is directly made on nonzero entries without the matricization process. By using the most popular COO format, the Mttkrp

[3] pronounced as Haiku

algorithm multiplies every nonzero entry $x(i,j,k)$ with the element-wise product of row-j of $\mathbf{B}$ and row-k of $\mathbf{C}$, and then sum-reduces the rows to row-i of $\mathbf{A}$. The matrix rows are irregularly accessed because of the sparsity of tensor $\mathcal{X}$.

3 HICOO FORMAT

HiCOO format is an extension of Compressed Sparse Blocks (CSB) format [5]. However, we store blocks in a sparse pattern instead of a dense long array. Analogous to element indices inside a block, block indices also use less bits than the indices of COO format to get further compression.

COO format (figure 1(a)) stores every nonzero value along with a tuple of indices, while HiCOO format (figure 1(b)) stores nonzeros in a block-wise pattern. bi, bj, bk index sparse blocks, and ei, ej, ek index nonzero elements within a block. $bptr$ stores the beginning locations of blocks. Tensor blocks are sorted in Z-Morton order. For an N^{th}-order sparse, cubical tensor $\mathcal{X} \in \mathbb{R}^{I \times \cdots \times I}$, COO format needs

$$S_{coo} = N \times nnz \times logI \tag{3}$$

bits to store all indices, where nnz is the number of nonzeros, and $logI$ is index bit-length. Suppose n_b $B \times B \times B$ blocks in $\mathcal{X}$,

$$S_{hicoo} = n_b \times lognnz + N \times n_b \times log\frac{I}{B} + N \times nnz \times logB. \tag{4}$$

Assume $logI = 32$ (int), $lognnz = 64$ (long int), $logB = 8$, and $N = 3$. We set block density $\alpha = \frac{n_b}{nnz}$, thus $S_{hicoo} < S_{coo}$ when $\alpha < 0.53$. If a tensor has more than 2 nonzeros per block, HiCOO takes less space than COO.

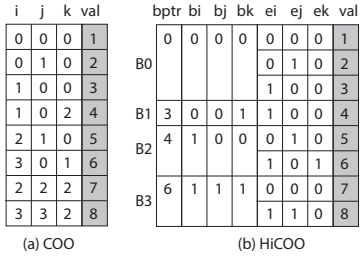


Figure 1: COO and HiCOO (in $2 \times 2 \times 2$ blocks) formats.

4 MTTKRP ALGORITHMS

Sequential Algorithm. Sequential HiCOO-Mttkrp algorithm is analogous to COO-Mttkrp algorithm by calculating the tensor index $i(j,k)$ from bi (bj, bk) and ei (ej, ek). Matrices are tiled by corresponding tensor blocks.

Parallel Algorithm. When updating matrix $\mathbf{A}$, multiple threads may write to the same row-i of $\mathbf{A}$. We integrate blocks into larger *bulks*, then schedule bulks according to their write dependence in Mttkrp to avoid these write conflicts. According to irregular tensor shapes, we choose between privatization and direct parallelization strategies. If the number of dependent bulks is larger than that of independent bulks, we use privatization strategy which computes partial matrix data thread-locally in parallel and then does a parallel reduction to get the final results; otherwise, we directly parallelize bulks by updating different matrix regions without write conflicts.

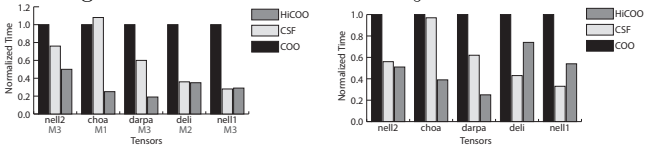
5 EVALUATION AND ANALYSIS

The experiments are tested on a Intel Xeon CPU E5-2650 platform consisting 24 physical cores with the gcc 5.4.1 compiler. Sparse tensors from the FROSTT dataset [20] are shown in table 1. We use 32-bit integers for indices and single-precision floating points for values.

Table 1: Description of sparse tensors.

Dataset	Order	Dimensions	NNZ	Density
nell2	3	$12K \times 9K \times 29K$	77M	2.4×10^{-5}
choa	3	$712K \times 10K \times 767$	27M	5.0×10^{-6}
darpa	3	$22K \times 22K \times 24M$	28M	2.4×10^{-9}
deli	3	$533K \times 17M \times 2.5M$	140M	6.1×10^{-12}
nell1	3	$3M \times 2M \times 25M$	144M	9.1×10^{-13}

Figure 2 shows the normalized execution times of parallel Mttkrps in COO, CSF (from Splatt[4]), and HiCOO formats in a single mode and all three modes separately on third-order tensors. Y-axis shows the execution time normalized to that of COO-Mttkrp. [5] Parallel HiCOO-Mttkrp in a single mode achieves up to 3.5× (2.0× on average) speedup over COO format and 4.3× (2.2× on average) speedup over CSF format. Regards to the sum of the execution time of all the three modes, parallel HiCOO-Mttkrp overperforms COO- and CSF-Mttkrp by up to 3.0× and 2.5× respectively for all but tensors *deli* and *nell1* for CSF-Mttkrp. HiCOO takes less space than COO and CSF for all but tensors *deli* and *nell1* for COO, because their block density ($\alpha \approx 1$) are much larger than our threshold 0.53 in § 3.



(a) Single mode (b) All modes

Figure 2: MTTKRP performance comparison.

Table 2: Sparse tensor space comparison.

tensors	COO (MB)	CSF (MB)	HiCOO (MB)
choa	411	666	192
darpa	434	958	308
nell2	1150	1850	546
deli	2090	4120	3490
nell1	2140	4430	3620

ACKNOWLEDGMENT

We thank Nick Liu and Srinivas Eswar for their comments. This material is based upon work supported by the U.S. National Science Foundation (NSF) Award Number 1533768. This work has also been funded in part by an Academic Alliances LDRD from Sandia National Laboratories. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

[4] An optimized Mttkrp algorithm is introduced in Splatt [21]. We configure Splatt with "ALLMODE" setting (storing all N CSF representations) for its best performance.

[5] No comparison with F-COO format is because it was implemented on GPUs.

REFERENCES

[1] Martín Abadi and others. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. (2015). Software available from tensorflow.org.

[2] Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. 2014. Tensor Decompositions for Learning Latent Variable Models. *J. Mach. Learn. Res.* 15, 1 (Jan. 2014), 2773–2832.

[3] Brett W. Bader and Tamara G. Kolda. 2007. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing* 30, 1 (December 2007), 205–231. DOI: https://doi.org/10.1137/060676489

[4] M. Baskaran, B. Meister, N. Vasilache, and R. Lethin. 2012. Efficient and scalable computations with sparse tensors. In *High Performance Extreme Computing (HPEC), 2012 IEEE Conference on*. 1–6. DOI:https://doi.org/10.1109/HPEC.2012.6408676

[5] Aydin Buluç, Jeremy T. Fineman, Matteo Frigo, John R. Gilbert, and Charles E. Leiserson. 2009. Parallel Sparse Matrix-vector and Matrix-transpose-vector Multiplication Using Compressed Sparse Blocks. In *Proceedings of the Twenty-first Annual Symposium on Parallelism in Algorithms and Architectures (SPAA '09)*. ACM, New York, NY, USA, 233–244. DOI:https://doi.org/10.1145/1583991.1584053

[6] Joon Hee Choi and S. Vishwanathan. 2014. DFacTo: Distributed Factorization of Tensors. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger (Eds.). Curran Associates, Inc., 1296–1304.

[7] Andrzej Cichocki. 2014. Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions. *CoRR* abs/1403.2048 (2014).

[8] Lieven De Lathauwer and Dimitri Nion. 2008. Decompositions of a Higher-Order Tensor in Block Terms—Part III: Alternating Least Squares Algorithms. *SIAM J. Matrix Anal. Appl.* 30, 3 (2008), 1067–1083. DOI:https://doi.org/10.1137/070690730 arXiv:http://dx.doi.org/10.1137/070690730

[9] Joyce C. Ho, Joydeep Ghosh, and Jimeng Sun. 2014. Marble: High-throughput Phenotyping from Electronic Health Records via Sparse Nonnegative Tensor Factorization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '14)*. ACM, New York, NY, USA, 115–124. DOI:https://doi.org/10.1145/2623330.2623658

[10] U. Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. GigaTensor: Scaling Tensor Analysis Up by 100 Times—Algorithms and Discoveries. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, New York, NY, USA, 316–324. DOI:https://doi.org/10.1145/2339530.2339583

[11] Oguz Kaya and Bora Uçar. 2015. Scalable Sparse Tensor Decompositions in Distributed Memory Systems. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '15)*. ACM, New York, NY, USA, Article 77, 11 pages. DOI:https://doi.org/10.1145/2807591.2807624

[12] Tamara G. Kolda and Brett W. Bader. 2009. Tensor Decompositions and Applications. *SIAM Rev.* 51, 3 (2009), 455–500.

[13] Jiajia Li, Casey Battaglino, Ioakeim Perros, Jimeng Sun, and Richard Vuduc. 2015. An input-adaptive and in-place approach to dense tensor-times-matrix multiply. In *ACM/IEEE Supercomputing (SC '15)*. ACM, New York, NY, USA.

[14] J. Li, J. Choi, I. Perros, J. Sun, and R. Vuduc. 2017. Model-Driven Sparse CP Decomposition for Higher-Order Tensors. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 1048–1057. DOI:https://doi.org/10.1109/IPDPS.2017.80

[15] Jiajia Li, Yuchen Ma, Chenggang Yan, and Richard Vuduc. 2016. Optimizing Sparse Tensor Times Matrix on Multi-core and Many-core Architectures. In *ACM/IEEE the Sixth Workshop on Irregular Applications: Architectures and Algorithms*. IEEE, Salt Lake City, Utah, USA.

[16] Bangtian Liu, Chengyao Wen, Anand D. Sarwate, and Maryam Mehri Dehnavi. 2017. A Unified Optimization Approach for Sparse Tensor Operations on GPUs. *CoRR* abs/1705.09905 (2017). http://arxiv.org/abs/1705.09905

[17] Alexander Novikov, Dmitry Podoprikhin, Anton Osokin, and Dmitry Vetrov. 2015. Tensorizing Neural Networks. *CoRR* abs/1509.06569 (2015).

[18] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2012. ParCube: Sparse Parallelizable Tensor Decompositions. In *Proceedings of the 2012 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I (ECML PKDD'12)*. Springer-Verlag, Berlin, Heidelberg, 521–536. DOI:https://doi.org/10.1007/978-3-642-33460-3_39

[19] Ioakeim Perros, Robert Chen, Richard Vuduc, and Jimeng Sun. 2015. Sparse Hierarchical Tucker Factorization and its Application to Healthcare. *IEEE International Conference on Data Mining (ICDM)* (2015).

[20] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. 2017. FROSTT: The Formidable Repository of Open Sparse Tensors and Tools. (2017). http://frostt.io/

[21] Shaden Smith, Niranjay Ravindran, Nicholas Sidiropoulos, and George Karypis. 2015. SPLATT: Efficient and Parallel Sparse Tensor-Matrix Multiplication. In *Proceedings of the 29th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*.

[22] Yichen Wang, Robert Chen, Joydeep Ghosh, Joshua C. Denny, Abel Kho, You Chen, Bradley A. Malin, and Jimeng Sun. 2015. Rubik: Knowledge Guided Tensor Factorization and Completion for Health Data Analytics. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1265–1274. DOI:https://doi.org/10.1145/2783258.2783395