HiCOO: A Hierarchical Sparse Tensor Format for Tensor Decompositions

Jiajia Li, Jimeng Sun, Richard Vuduc
Georgia Institute of Technology, Georgia, USA
jiajiali@gatech.edu

Background

Tensor decompositions is a set of unsupervised methods to analyze and extract knowledge from tensors, which is widely used in healthcare analytics, image processing, machine learning, and social network analytics.

Sparse tensors

A tensor of order N is an N-way array, which provides a natural input representation of a multiway dataset. Many real-world tensors are sparse and have specific features. To discover useful knowledge, an efficient sparse format are critical to algorithm performance and scalability.

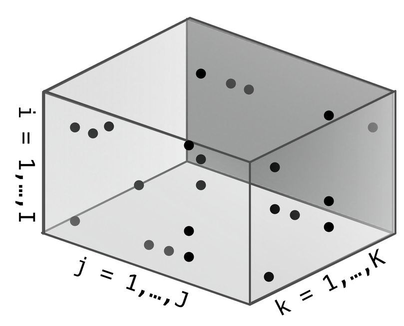
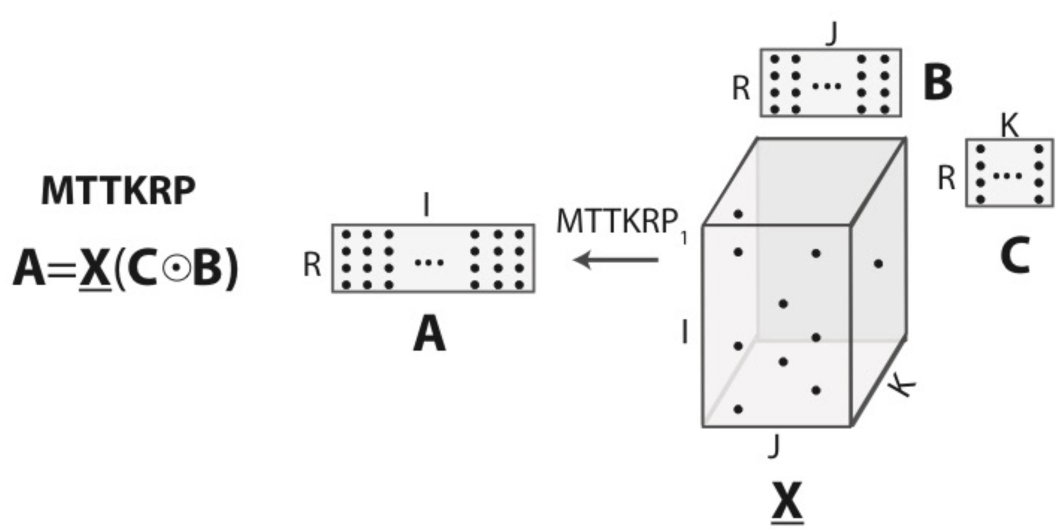


Fig.1: A third-order sparse tensor

Matricized Tensor Times Khatri-Rao Product (MTTKRP)

MTTKRP operation is the most expensive computational kernel of CANDECOMP/PARAFAC decomposition (CPD), a popular tensor decomposition algorithm.



Contributions

- We compare and summarize COO, CSF, and F-COO formats from storage space, the number of floating-point operations, memory traffic aspects, to motivate this work.
- We propose a new sparse tensor format, Hierarchical COOrdinate (HiCOO), 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.
- 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.
- 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.

HiCOO Format

HiCOO format blocks sparse sub-tensors and is represented by two-level indices with less bits. It is an extension of Compressed Sparse Blocks (CSB) format [1]. However, we store blocks in a sparse pattern instead of a dense long array, thus block indices also use less bits.

i	j	k	val
0	0	0	1
0	1	0	2
1	0	0	3
1	0	2	4
2	1	0	5
3	0	1	6
2	2	2	7
3	3	2	8

(a) COO

	bptr	bi	bj	bk	ei	ej	ek	val
B0	0	0	0	0	0	0	0	1
					0	1	0	2
					1	0	0	3
B1	3	0	0	1	1	0	0	4
B2	4	1	0	0	0	1	0	5
					1	0	1	6
B3	6	1	1	1	0	0	0	7
					1	1	0	8

(b) HiCOO

$S_{hicoo} < S_{coo}$

block density

$\alpha < 0.53$ when N=3

Format	Index Space (Bits)	Update Time	MTTKRP Work (Flops)	MTTKRP Memory Access (Bytes)	MTTKRP Arithmetic Intensity (AI)
COO	$96nnz$	0	$3Rnnz$	$12Rnnz$	$1/4$
F-COO	$65nnz$	$\Omega(nnz)$	$3Rnnz$	$12Rnnz$	$1/4$
CSF	$32-128nnz$	$\Omega(nnz)$	$2Rnnz-4Rnnz$	$8Rnnz-16Rnnz$	$1/4$
HiCOO	$24-184nnz$	0	$3Rnnz$	$12Rmin\{\frac{1}{c_B},1\}nnz$	$max\{\frac{1}{4},\frac{c_B}{4}\}$

Average Fiber Size per Tensor Block $c_B = \frac{nnz_B}{B}$

MTTKRP Algorithms

Algorithm 1 COO-MTTKRP algorithm([29]).

Input: A third-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, dense factors $\mathbf{B} \in \mathbb{R}^{J\times R}, \mathbf{C} \in \mathbb{R}^{K\times R}$;
Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in R^{I\times R}$;
$\triangleright\ \tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{C}\odot\mathbf{B})$

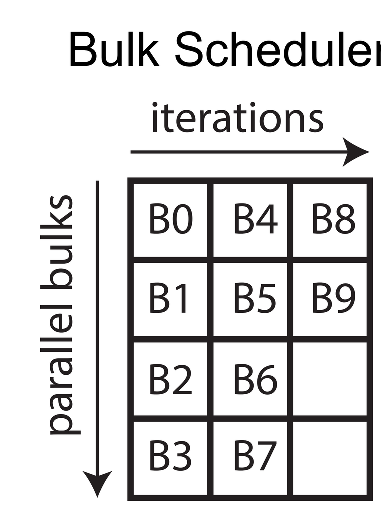
for $x = 1,\dots,nnz$ **do**
  $i = inds(x,1)$, $j = inds(x,2)$, $k = inds(x,3)$;
  for $r = 1,\dots,R$ **do**
    $\tilde{A}(i,r) += val(x)C(k,r)B(j,r)$
  end for
end for
return $\tilde{\mathbf{A}}$;

Algorithm 2 Sequential HiCOO-MTTKRP algorithm.

Input: A third-order HiCOO sparse tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, dense factors $\mathbf{B} \in \mathbb{R}^{J\times R}, \mathbf{C} \in \mathbb{R}^{K\times R}$, block size $B\times B\times B$;
Output: Updated dense factor matrix $\tilde{\mathbf{A}} \in R^{I\times R}$;
$\triangleright\ \tilde{\mathbf{A}} \leftarrow \mathcal{X}_{(1)}(\mathbf{C}\odot\mathbf{B})$

for $b = 0,\dots,n_b$ **do**
  $bi = binds(b,1)$, $bj = binds(b,2)$, $bk = binds(b,3)$;
  $\mathbf{A}_b = \mathbf{A} + bi*B*R$; $\mathbf{B}_b = \mathbf{B} + bj*B*R$; $\mathbf{C}_b = \mathbf{C} + bk*B*R$;
  for $x = bptr[b],\dots,bptr[b+1]-1$ **do**
    $ei = einds(x,1)$, $ej = einds(x,2)$, $ek = einds(x,3)$
    for $r = 1,\dots,R$ **do**
      $\tilde{A}_b(ei,r) \leftarrow val(x)C_b(ek,r)B_b(ej,r)$
    end for
  end for
end for
return $\tilde{\mathbf{A}}$;

- Sequential HiCOO MTTKRP
  - Smaller memory footprints
  - Better data locality
  - Matrix tiling
- Parallel HiCOO MTTKRP
  - Schedule larger blocks for irregular shaped tensors.
  - Two strategies: direct parallelization and privatization



Results

We test our algorithms on a Intel Xeon CPU E5-2650 platform with 24 physical cores. The sparse tensors are from FROSTT dataset [4]. We use 32-bit integers for indices, 64-bit integers for nonzero pointers, and 32-bit single-precision floating points for values. The block size is set to 128.

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}

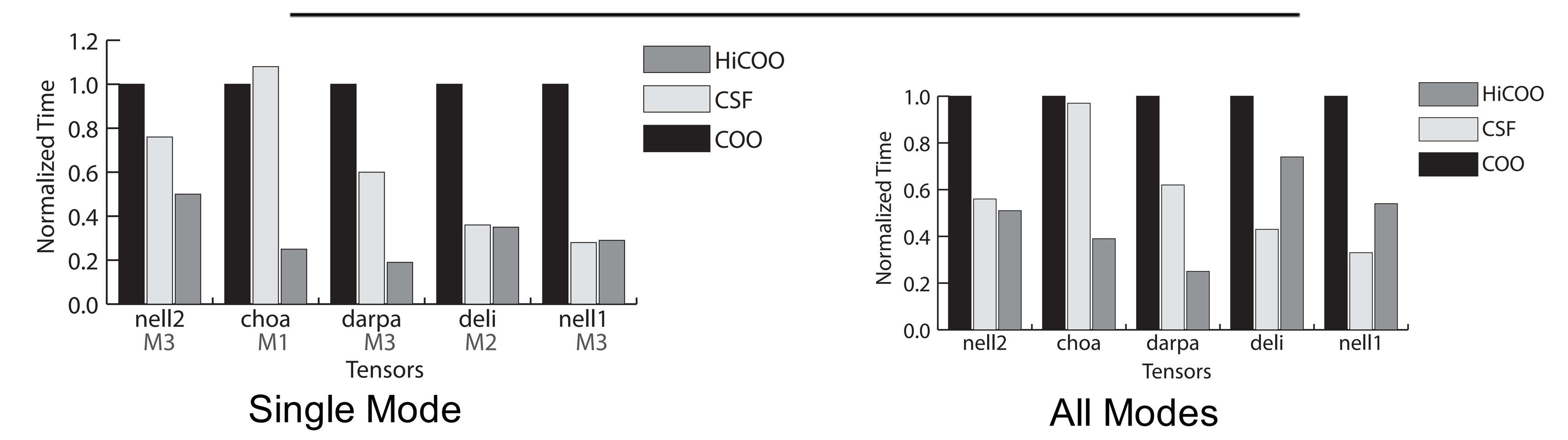


Fig.2: MTTKRP performance comparison

Table 2: Sparse tensor space comparison.

tensors	COO (MB)	CSF (MB)	HiCOO (MB)
choa	411	666	192
1998DARPA	434	958	308
nell2	1150	1850	546
nell1	2140	4430	3620
delicious	2090	4120	3490

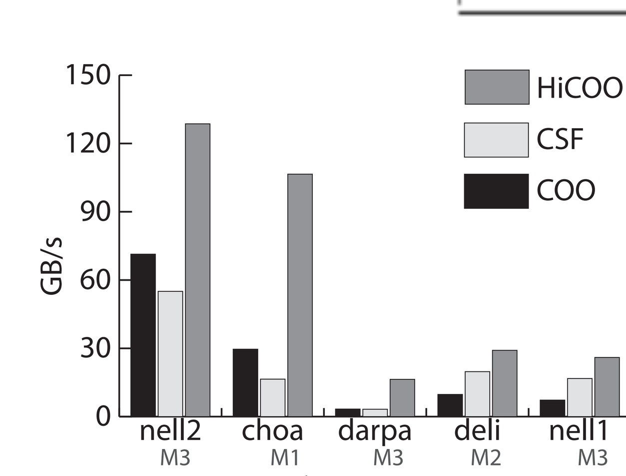


Fig. 3: The bandwidth of MTTKRPs

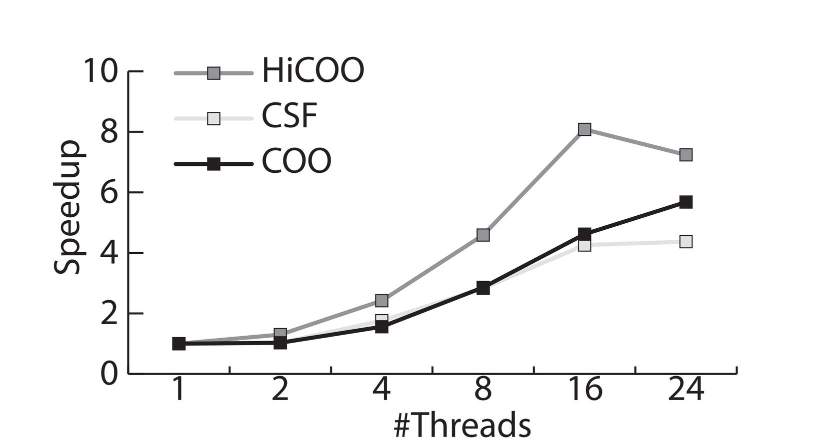


Fig. 4: Thread scalability on tensor "nell2"

Conclusion

Future, we will implement Tensor-Times-Matrix (TTM) operation using HiCOO and also accelerate HiCOO-MTTKRP and HiCOO-TTM algorithms on GPUs.

References

[1] Aydin Buluc, et al. 2009. Parallel Sparse Matrix-vector and Matrix-transpose-vector Multiplication Using Compressed Sparse Blocks. SPAA '09. ACM, New York, NY, USA, 233–244.

[2] Shaden Smith, et al. 2015. SPLATT: Efficient and Parallel Sparse Tensor-Matrix Multiplication. IPDPS'15.

[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.

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

Acknowledgement: This project is 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.