Sparsity-Aware Distributed Tensor Decomposition

ABSTRACT

Tensors are used by a wide range of applications as data structures to model multi-dimensional data. Tensor decomposition is a class of methods for latent data analytics. This work presents a sparsity-aware distributed tensor decomposition for distributed memory systems, which addresses the performance issues of irregular tensors, i.e., irregularly shaped and non-uniformly distributed non-zero values. Such tensors make performance optimization particularly challenging, due to the inherent trade-off between computation and communication and lack of information for balanced distribution. We optimize the CANDECOMP/PARAFAC decomposition, a popular low-rank tensor decomposition used in a wide variety of applications. We reveal the bottlenecks in the state-of-the-art design, and propose optimization strategies to address the bottlenecks. Results on real systems show our proposed sparsity-aware distributed CANDECOMP/PARAFAC decomposition outperforms the state-of-the-art distributed SPLATT library by up to 2.3× on 64 CPU nodes.

1 INTRODUCTION

Tensor decomposition is a class of tensor methods to data analytics, low-rank approximation, and so on. CANDECOMP /PARAFAC decomposition (CPD) is one of the most popular tensor decompositions, widely used in (healthcare, social network, brain signal, electrical grid) data analytics, machine learning, recommendation systems, and other domains. Sparse tensor decomposition brings more challenges for performance improvement.

Sparse CPD has been studied in distributed memory environment to obtain impressive performance improvement [1, 2]. However, we observe the state-of-the-art implementation [2] suffers some performance bottlenecks due to the imbalance between computation and communications, or within either. Our research surrounds the following research questions. First, how to configure MPI processes for a sparse CPD as it is hard to measure its nonzero distribution and features for best trade-off between computation and communication? Second, computation cost is not always dominated by sparse tensors computation, but could also be other computations. How to balance the trade-off between them? Third, no execution overlap between computation and communication. Pursuing execution overlap needs to deeply investigate and split some of the stages.

The main claimed contributions of this work are:

- Our work investigates the state-of-the-art implementation and observes five performance issues.
- We propose three strategies to solve the process configuration challenge, data distribution options and overlapping computation and communication.
- Our method scales well when using up to 64 24-core nodes and obtains up to 2.3× performance improvement over distributed SPLATT library [2].

2 BACKGROUND

CPD CANDECOMP/PARAFAC decomposition (CPD) factorizes a tensor into a sum of component rank-one tensors.

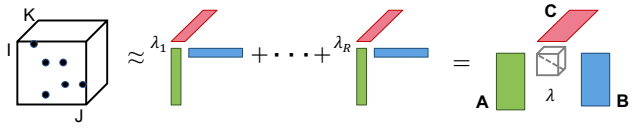

Figure 1: CPD for a 3rd-order sparse tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$.

Data decomposition and distribution. The medium-grained algorithm decomposes the tensor $\mathcal{X}$ and each factor matrix $\mathbf{A}^{(n)}$ in a non-overlapping fashion and distributes to processes.

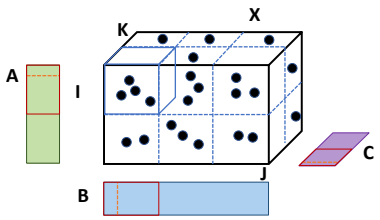

Figure 2: Tensor and matrix distribution on a $2 \times 3 \times 2$ process grid.

3 OBSERVATIONS

This section highlights five observations distilled from an extensive evaluation of the state-of-the-art medium-grained CPD implementation in SPLATT, the Surprisingly ParalleL spArse Tensor Toolkit [2].

OBS 1: *Both computation and communication have non-negligible costs and the dominance varies with tensors.*

OBS 2: *Computation cost is not always dominated by sparse tensor computation, but also dense matrix computations.*

OBS 3: *The communication bottleneck is mostly transforming matrices.*

OBS 4: *Different load imbalance factors influence computation and communication overhead.*

OBS 5: *There is no execution overlap between computation and communication.*

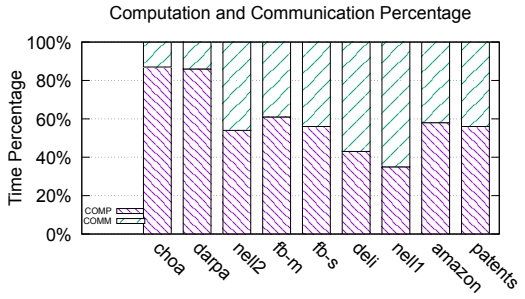

Figure 3: Either computation or communication can dominate.

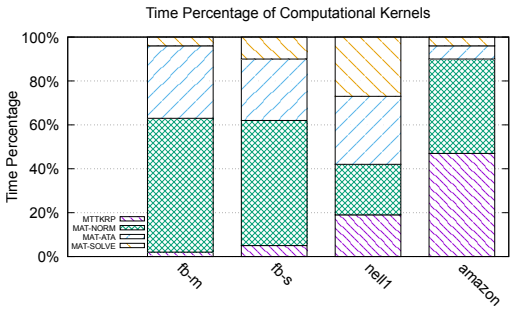

Figure 4: Multiple main computational kernels.

4 OPTIMIZATION

Prediction-based Grid Configuration The performance difference could be up to 3.1× on 32 processes among different grids. The state-of-the-art work [2] considers tensor dimension sizes but disregard the non-uniformity of sparsity and the matrix distribution. We take such irregularity into consideration in our prediction. It accurately estimates nonzero imbalance and chooses from several near-optimal candidates.

Algorithm 1 Our grid configuration strategy with $n_p = 1$.

Require: Number of processes P, tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$;
Ensure: Grid configuration G;
1: ▷ Ordered from largest to smallest
2: $primes$ = getPrimes(P);
3: avgI = $(I_1 + I_2 + I_3)/3$
4: **for** pr in $primes[0:-1]$ **do**
5: Assign pr to the largest mode I_n
6: $I_n -=$ avgI
7: **end for**
8: Assign $primes[-1]$ to every mode, generate three grid candidates.
9: Predict r_{nnz} of three candidates under different tensor distribution strategies
10: G = candidates with min(r_{nnz})
11: **Return** G;

Tensor Dimensions-Oriented Data Distribution The first strategy (Fig. 5 "#Nonzero Balance"), from state-of-the-art [1, 2], only targets nonzero balance; while the second strategy (Fig. 5 "Matrix Balance"), inspired from dense tensor decomposition, only targets matrix balance.We introduce a new strategy based on balanced matrix sizes that also considers nonzero distribution, thus balance both dense matrix and sparse tensor related computations.

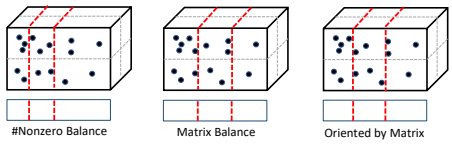

Figure 5: Our distribution ("Oriented by Matrix") balances both dense matrix and sparse tensor related computations

5 EXPERIMENTAL RESULTS

We perform experiments on a Linux-based Intel Xeon CPU E5-2670 v3 multicore server platform with 24 physical cores distributed on two sockets, each with 2.3 GHz frequency. We use the sparse tensors from real-world applications, that appear in Table 1.

Table 1: Description of sparse tensors.

Tensors	Dimensions	#Nnzs	Density
choa	$712K \times 10K \times 767$	27M	5.0×10^{-6}
darpa	$22K \times 22K \times 24M$	28M	2.4×10^{-9}
nell2	$12K \times 9K \times 29K$	77M	2.4×10^{-5}
fb-m	$23M \times 23M \times 166$	100M	1.1×10^{-9}
fb-s	$39M \times 39M \times 532$	140M	1.7×10^{-10}
deli	$533K \times 17M \times 2.5M$	140M	6.1×10^{-12}
nell1	$2.9M \times 2.1M \times 25M$	144M	9.1×10^{-13}
amazon	$4.8M \times 1.8M \times 1.8M$	1742M	1.1×10^{-10}
patents	$46 \times 239K \times 239K$	3597M	1.4×10^{-3}

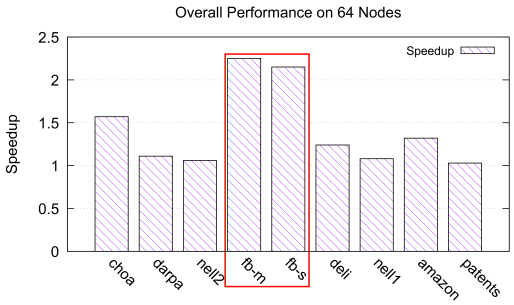

Figure 6: Our approach obtains higher speedup for irregular tensors (*fb-m, fb-s*).

REFERENCES

[1] 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* (Austin, Texas) *(SC '15)*. ACM, New York, NY, USA, Article 77, 11 pages. https://doi.org/10.1145/2807591.2807624

[2] Shaden Smith and George Karypis. 2016. A Medium-Grained Algorithm for Distributed Sparse Tensor Factorization. In *Parallel and Distributed Processing Symposium (IPDPS), 2016 IEEE International*. IEEE.