2023 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)

Efficient Differentially Private Tensor Factorization in the Parallel and Distributed Computing Paradigm

1st Feng Zhang
School of Computer Science
China University of Geosciences
Wuhan, China
jeff.f.zhang@gmail.com

2nd Hao Wang
School of Computer Science
China University of Geosciences
Wuhan, China
2056901285@qq.com

3rd Ruixin Guo
School of Computer Science
China University of Geosciences
Wuhan, China
1285019980@qq.com

4th Erkang Xue
School of Computer Science
China University of Geosciences, Wuhan
Wuhan, China
957204464@qq.com

5th Guangzhi Qu
Department of Computer Science and Engineering
Oakland University
Rochester, USA
gqu@oakland.edu

***Abstract*—Tensor factorization plays a fundamental role in multiple areas of AI research. Nevertheless, it encounters significant challenges related to privacy breaches and operational efficiency. In this study, we propose a novel approach that addresses both of these issues simultaneously by integrating differential privacy with parallel and distributed computing. To accommodate diverse scenarios, we introduce two models: DPTF-SVRG and ADMM-DPTF, each leveraging specific techniques. DPTF-SVRG is designed for single-GPU environments and utilizes a unique strategy to reduce gradient variance, enabling faster convergence compared to SGD. Moreover, it achieves parallelism on the GPU through a lock-free asynchronous approach. On the other hand, ADMM-DPTF utilizes distributed ADMM to parallelize DPTF-SVRG, enabling multi-GPU parallelism. Experimental results demonstrate that our algorithms outperform existing benchmarks while maintaining differential privacy.**

***Index Terms*—Tensor Factorization, Differential Privacy, Parallel and Distributed Computing, Machine Learning**

I. Introduction

Efficiency is a significant challenge in tensor factorization due to the large size of tensors. To address this challenge, parallel and distributed computing has emerged as a crucial solution [1]. However, in this computing paradigm, privacy protection becomes another key concern, as not all participating parties are willing to disclose their private data [2].

To date, there has been a lack of research that addresses both efficiency and privacy concerns simultaneously. This existing gap in the literature has motivated our study to propose a novel approach. We propose the utilization of differential privacy (DP) to ensure privacy protection while achieving efficiency in both single-GPU and multi-GPU parallel computing scenarios:

1) We propose a parallel algorithm named DPTF-SVRG (differentially private tensor factorization with stochastic variance reduced gradient) for performing differentially private tensor factorization on a single-GPU platform. Our algorithm employs a gradient variance reduction technique to accelerate the convergence of stochastic gradient descent (SGD). Furthermore, we have incorporated an asynchronous parallelism mechanism using a lock-free scheme, which facilitates efficient parallel computation on the GPU.

2) We propose ADMM-DPTF, a multi-GPU parallel algorithm for differentially private tensor factorization, achieved by integrating distributed alternating direction method of multipliers (ADMM) with DPTF-SVRG. ADMM-DPTF parallelizes the computation across multiple GPUs using a lock-free scheme within each GPU, while employing a quantized technique to enhance inter-GPU communication. This approach enables efficient and scalable tensor factorization, preserving privacy while leveraging the power of distributed computing.

II. Related Work

A. Parallel Tensor Factorization

Traditional parallel and distributed techniques have been instrumental in addressing the efficiency concern in matrix factorization. However, the application of such techniques to tensors has proven to be challenging [3].

Yu et al. [4] introduced DS-ADMM, an ADMM-based model for distributed matrix factorization. Zhang et al. [5] further optimized DS-ADMM and proposed DS-ADMM++, which incorporates quantized techniques and DP. However, it is important to note that these approaches are specific to matrices and do not directly address tensor factorization.

Fanhua Shang and Liu et al. [6], [7] introduced two ADMM models to address the low-rank tensor completion problem. Subsequently, their work has spurred numerous follow-up studies [8], [9]. However, it is worth noting that all of these works have overlooked the issue of privacy protection.

B. Differential Privacy

DP has emerged as a promising approach for privacy-preserving machine learning and has received significant attention in the past decade [10], [11]. However, when DP is applied to matrix factorization, the challenge of error accumulation arises. Hua et al. [12] and Liu et al. [13] proposed pioneering approaches to tackle this challenge. But these works they all focus on matrices rather than tensors.

Several studies have claimed to implement differentially private tensor factorization [14], [15]. However, these studies have failed to address the challenge of performance enhancement.

In contrast, our study not only leverages the ADMM for parallelization but also introduces an innovative differentially private strategy to ensure privacy preservation. What further distinguishes our work is the comprehensive integration of these techniques into unified models.

III. Preliminaries

A. CP Decomposition

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ denote a third-order tensor, and $\boldsymbol{A} \in \mathbb{R}^{I\times R}$, $\boldsymbol{B} \in \mathbb{R}^{J\times R}$, $\boldsymbol{C} \in \mathbb{R}^{K\times R}$ represent the three factor matrices, respectively. Here, R corresponds to the number of latent factors.

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \frac{1}{2}(&\sum_{(i,j,k)\in\Omega}(x_{ijk} - <\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k)^2 \\ &+ \lambda_1\|\boldsymbol{A}\|^2 + \lambda_2\|\boldsymbol{B}\|^2 + \lambda_3\|\boldsymbol{C}\|^2),\end{aligned} \tag{1}$$

where Ω represents the index set of non-nulls entries in the tensor $\boldsymbol{\mathcal{X}}$. The term $<\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k> = \sum_{r=1}^{R} a_{ir}b_{jr}c_{kr}$ denotes the inner product between $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_k$. Here, $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_k$ represent the i-th row of $\boldsymbol{A}$, the j-th row of $\boldsymbol{B}$, and the k-th row of $\boldsymbol{C}$, respectively; λ_1, λ_2, λ_3 are the regularization parameters.

B. Stochastic Variance Reduced Gradient

SVRG [16] is a technique designed to optimize SGD by mitigating the issue of gradient variance. The iteration rules for SVRG are as follows:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \tau(\nabla f_i(\boldsymbol{x}_t) - \nabla f_i(\tilde{\boldsymbol{x}}) + \boldsymbol{u}), \tag{2}$$

where $\tilde{\boldsymbol{x}}$ can be set to $\boldsymbol{x}_t$ or randomly chosen as $\boldsymbol{x}_s$ ($s = 0, 1, ..., t-1$), $\boldsymbol{u}$ represents the expectation of $\nabla f_i(\tilde{\boldsymbol{x}})$, according to the theory of paper [16], SVRG can achieve faster convergence than SGD.

C. Differential privacy [17]

Definition 1. (ϵ-DP) . D and D' represet two neighboring data sets. A random algorithm $\mathcal{A}$ is considered to satisfy ϵ-DP if, for any output S generated by $\mathcal{A}$, the following condition holds:

$$Pr[\mathcal{A}(D) = S] \le e^{\epsilon} \times Pr[\mathcal{A}(D') = S] \tag{3}$$

where ϵ represents privacy budget. A smaller value of ϵ indicates a higher level of privacy preservation but potentially reduced data utility.

IV. Single-GPU Parallelism

A. The Mathematics of the Algorithm

We define the objective function for the differentially private tensor factorization problem as:

$$\begin{aligned}\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \frac{1}{2}(&\sum_{(i,j,k)\in\Omega}(x_{ijk} - <\boldsymbol{a_i}, \boldsymbol{b_j}, \boldsymbol{c_k}>)^2 \\ &+ \lambda_1\|\boldsymbol{A}\|^2 + \lambda_2\|\boldsymbol{B}\|^2 + \lambda_3\|\boldsymbol{C}\|^2) \\ &+ \sum_{(i,j,k)\in\Omega}\eta_{ijk} <\boldsymbol{a_i}, \boldsymbol{b_j}, \boldsymbol{c_k}> .\end{aligned} \tag{4}$$

where $\eta_{ijk} \in \eta$ is the Laplacian noise corresponding to x_{ijk}, and $\boldsymbol{\eta}$ is a Laplacian noise matrix, $Lap(\Delta/\epsilon)$ (Δ is the sensitivity, and ϵ is the DP budget). Similar to Zhang et al.'s work [5], it is easy to prove that the derived $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ satisfy ϵ-DP.

We speed up differentially private tensor factorization by reducing the gradient variance. We can do this because the objective function is equivalent to the one with regularized variance reduced gradient.

Let:

$$f_{ijk}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \frac{1}{2}(x_{ijk} - <\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k>)^2, \tag{5}$$

$$f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \sum_{(i,j,k)\in\Omega} f_{ijk}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}), \tag{6}$$

$$\begin{aligned}g(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = &\frac{\lambda_1}{2}\|\boldsymbol{A}\|^2 + \frac{\lambda_2}{2}\|\boldsymbol{B}\|^2 + \frac{\lambda_3}{2}\|\boldsymbol{C}\|^2 \\ &+ \sum_{(i,j,k)\in\Omega}\eta_{ijk} <\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k>,\end{aligned} \tag{7}$$

Then Eq. (4) can be rewritten as:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} J(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) + g(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}), \tag{8}$$

To solve Eq. (8), we randomly select a f_{ijk} to calculate the gradient at each iteration. Each f_{ijk} only depends on $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_k$, instead of the entire factor matrix $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. Let $\tilde{\boldsymbol{A}}$, $\tilde{\boldsymbol{B}}$, $\tilde{\boldsymbol{C}}$ be the previous $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ in the iteration (see Eq. (2)), respectively. And let $\boldsymbol{U}_A$, $\boldsymbol{U}_B$, and $\boldsymbol{U}_C$ be the mean values of the partial derivatives $\frac{\partial f(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}},\tilde{\boldsymbol{C}})}{\partial \tilde{\boldsymbol{A}}}$, $\frac{\partial f(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}},\tilde{\boldsymbol{C}})}{\partial \tilde{\boldsymbol{B}}}$, and $\frac{\partial f(\tilde{\boldsymbol{A}},\tilde{\boldsymbol{B}},\tilde{\boldsymbol{C}})}{\partial \tilde{\boldsymbol{C}}}$ of all non-nulls in $\boldsymbol{\mathcal{X}}$, respectively, i.e.:

$$\boldsymbol{U}_A(i,:) = \frac{1}{|\Omega(i,:,:)|}\sum_{(i,j,k)\in\Omega(i,:,:)} -\tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{b}}_j * \tilde{\boldsymbol{c}}_k, \tag{9}$$

$$\boldsymbol{U}_B(j,:) = \frac{1}{|\Omega(:,j,:)|}\sum_{(i,j,k)\in\Omega(:,j,:)} -\tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i * \tilde{\boldsymbol{c}}_k, \tag{10}$$

$$\boldsymbol{U}_C(k,:) = \frac{1}{|\Omega(:,:,k)|}\sum_{(i,j,k)\in\Omega(:,:,k)} -\tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i * \tilde{\boldsymbol{b}}_j, \tag{11}$$

where $\tilde{\varepsilon}_{ijk} = x_{ijk} - <\tilde{\boldsymbol{a}}_i, \tilde{\boldsymbol{b}}_j, \tilde{\boldsymbol{c}}_k>$. $*$ denotes the Hadamard product.

Then the iterative updates of differential private tensor factorization with variance reduced gradient are:

$$\begin{aligned}\boldsymbol{a}_i = \boldsymbol{a}_i + \tau(\varepsilon_{ijk}\boldsymbol{b}_j * \boldsymbol{c}_k - \lambda_1\boldsymbol{a}_i - \eta_{ijk}\boldsymbol{b}_j * \boldsymbol{c}_k \\ - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{b}}_j * \tilde{\boldsymbol{c}}_k - \boldsymbol{U}_A(i,:)),\end{aligned} \tag{12}$$

$$\begin{aligned}\boldsymbol{b}_j = \boldsymbol{b}_j + \tau(\varepsilon_{ijk}\boldsymbol{a}_i * \boldsymbol{c}_k - \lambda_2\boldsymbol{b}_j - \eta_{ijk}\boldsymbol{a}_i * \boldsymbol{c}_k \\ - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i * \tilde{\boldsymbol{c}}_k - \boldsymbol{U}_B(j,:)),\end{aligned} \tag{13}$$

$$\begin{aligned}\boldsymbol{c}_k = \boldsymbol{c}_k + \tau(\varepsilon_{ijk}\boldsymbol{a}_i * \boldsymbol{b}_j - \lambda_3\boldsymbol{c}_k - \eta_{ijk}\boldsymbol{a}_i * \boldsymbol{b}_j \\ - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i * \tilde{\boldsymbol{b}}_j - \boldsymbol{U}_C(k,:)).\end{aligned} \tag{14}$$

Algorithm 1 DPTF-SVRG

```
Input: Sparse tensor $\mathcal{X}$, learning step $\tau$, regularization parameter $\lambda$, the number of non-nulls $n$
Output: $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$
1: Initialize the factor matrix $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$
2: for $s = 0$ to $S - 1$ do
3:   Let $\tilde{\boldsymbol{A}} = \boldsymbol{A}, \tilde{\boldsymbol{B}} = \boldsymbol{B}, \tilde{\boldsymbol{C}} = \boldsymbol{C}$
4:   Initialize $\boldsymbol{U}_A, \boldsymbol{U}_B, \boldsymbol{U}_C$ to 0
5:   Generate Laplacian noise $\boldsymbol{\eta}$
6:   for $i = 0$ to $n - 1$ parallel with lock do
7:     Let $(i, j, k)$ is the element for the $i$-th thread
8:     Update $\boldsymbol{U}_A(i,:)$ by (9)
9:     Update $\boldsymbol{U}_B(j,:)$ by (10)
10:    Update $\boldsymbol{U}_C(k,:)$ by (11)
11:  end for
12:  for $i = 0$ to $n - 1$ parallel do
13:    Let $(i, j, k)$ is the element for $i$-th thread
14:    Compute $\varepsilon_{ijk} = x_{ijk} - < \boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k >$
15:    Update $\boldsymbol{a}_i$ by (12)
16:    Update $\boldsymbol{b}_j$ by (13)
17:    Update $\boldsymbol{c}_k$ by (14)
18:  end for
19: end for
```

B. Parallel DPTF-SVRG on Single-GPU

Eqs. (12)-(14) show that each x_{ijk} updates three vectors: $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_k$. But if two non-nulls, x_{ijk} and x_{ixy}, are calculated simultaneously, we need to read and write the vector $\boldsymbol{a}_i$ at the same time, leading to conflicts. We adopt a lock-free scheme like Hogwild! [18] to handle such conflicts. We call the single-GPU parallel differentially private tensor factorization algorithm DPTF-SVRG, shown as Algorithm 1.

V. Multi-GPU parallelism

A. Distributed DPTF Based on ADMM

We extend matrices to tensors so that differentially private tensor factorization may work in distributed systems. We adopt a data parallelism strategy similar to DS-ADMM [4]. Assuming there are P nodes, we randomly divide the 1st order to different nodes and split the entire tensor into multiple small tensors, i.e., $\boldsymbol{\mathcal{X}}^p \in \mathbb{R}^{I^p \times J \times K}$, $p = 1, ..., P$. We denote the local factor matrices as $\boldsymbol{A}^p$, $\boldsymbol{B}^p$, and $\boldsymbol{C}^p$, respectively, and set up two global factor matrices $\bar{\boldsymbol{B}}$ and $\bar{\boldsymbol{C}}$. The global factor matrices $\bar{\boldsymbol{B}}$, $\bar{\boldsymbol{C}}$ and the local factor matrices $\boldsymbol{B}^p$, $\boldsymbol{C}^p$ have the same size. And they contain all the columns.

The differentially private tensor factorization problem based on distributed ADMM can be formulated as:

$$\begin{aligned}&\min_{\substack{\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \\ \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}}} \frac{1}{2}(\sum_{p=1}^{P}\sum_{(i,j,k)\in\Omega^p}(x_{ijk} - < \boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p >)^2 + \lambda_1\|\boldsymbol{A}^p\|^2 \\ &+ \lambda_2\|\boldsymbol{B}^p\|^2 + \lambda_3\|\boldsymbol{C}^p\|^2) + \sum_{(i,j,k)\in\Omega^p}\eta_{ijk} < \boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p > \\ &s.t.\ \boldsymbol{B}^p - \bar{\boldsymbol{B}} = 0, \boldsymbol{C}^p - \bar{\boldsymbol{C}} = 0,\end{aligned} \tag{15}$$

where Ω^p is the non-null elements assigned to node p.

Please note that Eq. (15) precisely formulates differential private tensor factorization as a distributed ADMM problem and can be solved with the distributed ADMM paradigm [19].

We define

$$f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) = \sum_{p=1}^{P} f^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p), \tag{16}$$

where

$$f^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) = \sum_{(i,j,k)\in\Omega^p} \hat{f}_{ijk}(\boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p), \tag{17}$$

and

$$\begin{aligned}\hat{f}_{ijk}(\boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p) = \frac{1}{2}((x_{ijk} - < a_i^p, b_j^p, c_k^p >)^2 + \lambda_1 a_i^p (a_i^p)^T \\ + \lambda_2 b_j^p (b_j^p)^T + \lambda_3 c_k^p (c_k^p)^T) + \eta_{ijk} < a_i^p, b_j^p, c_k^p > .\end{aligned} \tag{18}$$

The augmented Lagrangian function of Eq. (15) is

$$\begin{aligned}L(\boldsymbol{A}, &\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{O}_B, \boldsymbol{O}_C, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}) = \\ &f(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}) + l(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{O}_B, \boldsymbol{O}_C, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}),\end{aligned} \tag{19}$$

where

$$l(\boldsymbol{B}, \boldsymbol{C}, \boldsymbol{O}_B, \boldsymbol{O}_C, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}) = \sum_{p=1}^{P} l^p(\boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \boldsymbol{\Theta}_B^p, \boldsymbol{\Theta}_C^p), \tag{20}$$

$$\begin{aligned}l^p(\boldsymbol{B}^p, \boldsymbol{C}^p, &\bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \boldsymbol{\Theta}_B^p, \boldsymbol{\Theta}_C^p) \\ &= \frac{\rho}{2}\|\boldsymbol{B}^p - \bar{\boldsymbol{B}}\|^2 + tr((\boldsymbol{\Theta}_B^p)^T(\boldsymbol{B}^p - \bar{\boldsymbol{B}})) \\ &+ \frac{\rho}{2}\|\boldsymbol{C}^p - \bar{\boldsymbol{C}}\|^2 + tr((\boldsymbol{\Theta}_C^p)^T(\boldsymbol{C}^p - \bar{\boldsymbol{C}})),\end{aligned} \tag{21}$$

where ρ is a hyperparameter, and $\boldsymbol{O}_B = \boldsymbol{\Theta}_B^p$, $\boldsymbol{O}_C = \boldsymbol{\Theta}_C^p$ are the Lagrange multipliers.

Let

$$\begin{aligned}&L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \boldsymbol{\Theta}_B^p, \boldsymbol{\Theta}_C^p) \\ &\quad = f^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) + l^p(\boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}, \boldsymbol{\Theta}_B^p, \boldsymbol{\Theta}_C^p)\end{aligned} \tag{22}$$

The update rules for ADMM are

$$\begin{aligned}&\boldsymbol{A}_{t+1}^p, \boldsymbol{B}_{t+1}^p, \boldsymbol{C}_{t+1}^p \\ &= \arg\min_{\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p} L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}_t, \bar{\boldsymbol{C}}_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t),\end{aligned} \tag{23}$$

$$\bar{\boldsymbol{B}}_{t+1}, \bar{\boldsymbol{C}}_{t+1} = \arg\min_{\bar{\boldsymbol{B}},\bar{\boldsymbol{C}}} L(\boldsymbol{A}_{t+1}, \boldsymbol{B}_{t+1}, \boldsymbol{C}_{t+1}, (\boldsymbol{O}_B)_t, (\boldsymbol{O}_C)_t, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}), \tag{24}$$

$$(\boldsymbol{\Theta}_B^p)_{t+1} = (\boldsymbol{\Theta}_B^p)_t + \rho(\boldsymbol{B}_{t+1}^p - \bar{\boldsymbol{B}}_{t+1}), \tag{25}$$

$$(\boldsymbol{\Theta}_C^p)_{t+1} = (\boldsymbol{\Theta}_C^p)_t + \rho(\boldsymbol{C}_{t+1}^p - \bar{\boldsymbol{C}}_{t+1}). \tag{26}$$

The update rules of $\bar{\boldsymbol{B}}$ and $\bar{\boldsymbol{C}}$ are

$$\bar{\boldsymbol{B}}_{t+1} = \frac{1}{P}\sum_{p=1}^{P} \boldsymbol{B}_{t+1}^p, \tag{27}$$

$$\bar{\boldsymbol{C}}_{t+1} = \frac{1}{P}\sum_{p=1}^{P} \boldsymbol{C}_{t+1}^p, \tag{28}$$

Algorithm 2 ADMM-DPTF

```
Input: sparse tensor $\boldsymbol{\mathcal{X}}$, learning step $\tau$, regularization parameter $\lambda$, $\rho$, the number of non-nulls $n$, the number of GPUs $P$
Output: $\boldsymbol{A}, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}$
  Data is split using Algorithm 2 and distributed to P GPUs
  for $p = 0$ to $P-1$ multi-GPUs parallel do
    Initialize parameter $\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}$
    Let $\boldsymbol{\Theta}_B^p = 0, \boldsymbol{\Theta}_C^p = 0$
    for $s = 0$ to $S-1$ do
      Let $\tilde{\boldsymbol{A}}^p = \boldsymbol{A}^p, \tilde{\boldsymbol{B}}^p = \boldsymbol{B}^p, \tilde{\boldsymbol{C}}^p = \boldsymbol{C}^p$
      Initialize $\boldsymbol{U}_A^p = 0, \boldsymbol{U}_B^p = 0, \boldsymbol{U}_C^p = 0$
      Generate Laplacian noise $\boldsymbol{\eta}$
      for $i = 0$ to $n-1$ sigle-GPU parallel with lock do
        Let $(i,j,k)$ is the element for $i$-th thread
        Update $\boldsymbol{U}_A^p(i,:)$ by (9)
        Update $\boldsymbol{U}_B^p(j,:)$ by (10)
        Update $\boldsymbol{U}_C^p(k,:)$ by (11)
      end for
      for $i = 0$ to $n-1$ sigle-GPU parallel do
        Let $(i,j,k)$ is the element for i-th thread
        Calculate $\varepsilon_{ijk} = x_{ijk} - <\boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p>$
        Update $\boldsymbol{a}_i^p$ by (32)
        Update $\boldsymbol{b}_j^p$ by (33)
        Update $\boldsymbol{c}_k^p$ by (34)
      end for
      Update $\bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}$ by (27), (28)
      Update $\boldsymbol{\Theta}_B^p, \boldsymbol{\Theta}_C^p$ by (25), (26)
    end for
  end for
```

B. Gradient Variance Reduction

To employ gradient variance reduction to solve Eq. (23), we first transform L^p into its objective function

$$\min_{\substack{\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p\\ \bar{\boldsymbol{B}},\bar{\boldsymbol{C}}}} J(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}) = f^s(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) + g(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}), \tag{29}$$

data set	MovieLens 20M	Netflix
non-nulls #	20000263	100480507
size	138493 × 27278 × 25	17770 × 480189 × 2
range	1 - 5	0.5 - 5.0
λ	0.05	0.05
ρ	0.05	0.05
τ	0.01	0.01

TABLE I: Dataset and Parameter Setting

where

$$f^s(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) = \sum_{(i,j,k)\in\Omega^p} f_{ijk}^{SVRG}(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p),$$
$$f_{ijk}^{SVRG}(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) = \frac{1}{2}(x_{ijk} - <\boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k>)^2, \tag{30}$$

$$\begin{aligned} g(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \bar{\boldsymbol{B}}, \bar{\boldsymbol{C}}) &= \frac{\lambda_1}{2}\|\boldsymbol{A}^p\|^2 + \frac{\lambda_2}{2}\|\boldsymbol{B}^p\|^2 + \frac{\lambda_3}{2}\|\boldsymbol{C}^p\|^2 \\ &+ \sum_{(i,j,k)\in\Omega} \eta_{ijk} <\boldsymbol{a}_i^p, \boldsymbol{b}_j^p, \boldsymbol{c}_k^p> \\ &+ \frac{\rho}{2}\|\boldsymbol{B}^p - \bar{\boldsymbol{B}}\|^2 + tr((\boldsymbol{\Theta}_B^p)^T(\boldsymbol{B}^p - \bar{\boldsymbol{B}})) \\ &+ \frac{\rho}{2}\|\boldsymbol{C}^p - \bar{\boldsymbol{C}}\|^2 + tr((\boldsymbol{\Theta}_C^p)^T(\boldsymbol{C}^p - \bar{\boldsymbol{C}})). \end{aligned} \tag{31}$$

Then the update rules for $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ are:

$$\begin{aligned} \boldsymbol{a}_i^p = \boldsymbol{a}_i^p + \tau(&\varepsilon_{ijk}\boldsymbol{b}_j^p * \boldsymbol{c}_k^p - \lambda_1\boldsymbol{a}_i^p \\ &- \eta_{ijk}\boldsymbol{b}_j^p * \boldsymbol{c}_k^p - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{b}}_j^p * \tilde{\boldsymbol{c}}_k^p - \boldsymbol{U}_A^p(i,:)), \end{aligned} \tag{32}$$

$$\begin{aligned} \boldsymbol{b}_j^p = \boldsymbol{b}_j^p + \tau(&\varepsilon_{ijk}\boldsymbol{a}_i^p * \boldsymbol{c}_k^p - \lambda_2\boldsymbol{b}_j^p - \rho(\boldsymbol{b}_j^p - \bar{\boldsymbol{b}}_j) - \theta_{\boldsymbol{B}_j}^p \\ &- \eta_{ijk}\boldsymbol{a}_i^p * \boldsymbol{c}_k^p - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i^p * \tilde{\boldsymbol{c}}_k^p - \boldsymbol{U}_B^p(j,:)), \end{aligned} \tag{33}$$

$$\begin{aligned} \boldsymbol{c}_k^p = \boldsymbol{c}_k^p + \tau(&\varepsilon_{ijk}\boldsymbol{a}_i^p * \boldsymbol{b}_j^p - \lambda_3\boldsymbol{c}_k^p - \rho(\boldsymbol{c}_k^p - \bar{\boldsymbol{c}}_k) - \theta_{\boldsymbol{C}_k}^p \\ &- \eta_{ijk}\boldsymbol{a}_i^p * \boldsymbol{b}_j^p - \tilde{\varepsilon}_{ijk}\tilde{\boldsymbol{a}}_i^p * \tilde{\boldsymbol{b}}_j^p - \boldsymbol{U}_C^p(k,:)), \end{aligned} \tag{34}$$

where the formula for calculating $\boldsymbol{U}_A^p$, $\boldsymbol{U}_B^p$, $\boldsymbol{U}_C^p$ are entirely consistent with those for the preceding formulas 9, 10, 11.

C. Parallel ADMM-DPTF on Multi-GPU

We implement multi-GPU parallel differentially private tensor factorization based on the DS-ADMM paradigm and the lock-free scheme. We call it ADMM-DPTF (ADMM based Differentially Private Tensor Factorization). Algorithm 2 shows its details.

VI. Experiment

A. Experiment Setup

1) Experiment Platform: Our experiments are based on the GPU partition of Tianhe 2. Five nodes are in the GPU partition, each with two K80 GPUs. The K80 GPU is a dual-GPU design with 24 GB GDDR5 memory.

2) Dataset: Table I shows the data set information and the corresponding parameter setting.

3) Parameter Setting: We initialize $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ with random numbers in (0,1) and set the rank R of the tensor to 10 and we set the maximum number of iterations to 100.

4) RMSE: The Root Mean Square Error (RMSE) is defined as

$$RMSE = \sqrt{\frac{1}{|\Omega|}\sum(x_{ijk} - < \boldsymbol{a}_i, \boldsymbol{b}_j, \boldsymbol{c}_k >)^2}, \quad (35)$$

where x_{ijk} is the non-null element in the test set, $\boldsymbol{a}_i$, $\boldsymbol{b}_j$, and $\boldsymbol{c}_k$ represent the three factor matrices, respectively.

5) Speedup: Speedup is a metric of the performance gain of a parallel algorithm compared to the corresponding serial algorithm when they deal with the same problem. It is defined as

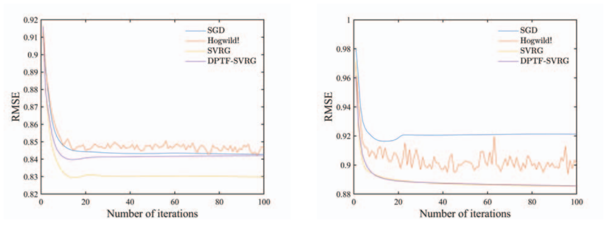
$$S_p = \frac{T_1}{T_p}. \quad (36)$$

Our experiments compare the speedup of the serial algorithm to the single-GPU parallel algorithm, and the speedup of the single-GPU to the multi-GPU.

Dataset	Algorithm	Description	Speedup
MovieLens	SGD	serial	1
	Hogwild!	sigle-GPU	22.99
	SVRG	serial	1
	DPTF-SVRG	sigle-GPU	15.73
Netflix	SGD	serial	1
	Hogwild!	sigle-GPU	21.13
	SVRG	serial	1
	DPTF-SVRG	sigle-GPU	16.09

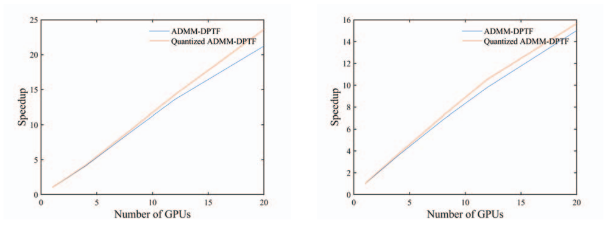
TABLE II: Comparison of DPTF-SVRG speedup

B. Experimental Results



(a) MovieLens 20M (b) Netflix

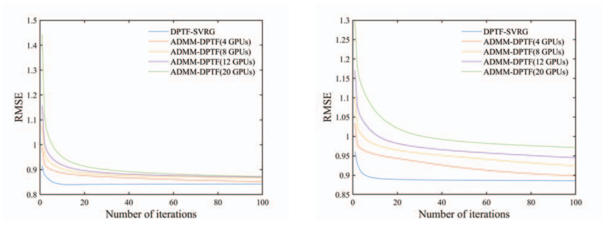
Fig. 1: RMSE varies with the number of iterations



(a) MovieLens 20M (b) Netflix

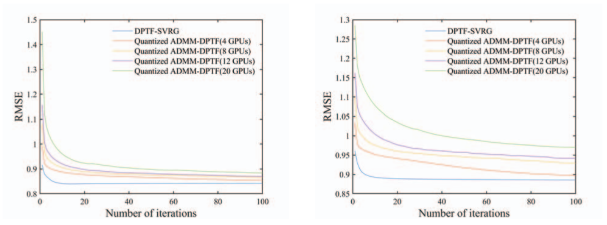
Fig. 2: Speedup comparison of ADMM-DPTF and quantized ADMM-DPTF

1) Accuracy of Prediction: Figure 1 shows the accuracy trend over the number of iterations for the four algorithms. Among them, SVRG performs the best on both datasets because it reduces the gradient variance during the update process so that it converges faster. However, Hogwild! obviously fluctuates during the decline. The fluctuation is because it adopts a lock-free parallel method, which may introduce errors in parameter updating.



(a) MovieLens 20M (b) Netflix

Fig. 3: RMSE of ADMM-DPTF



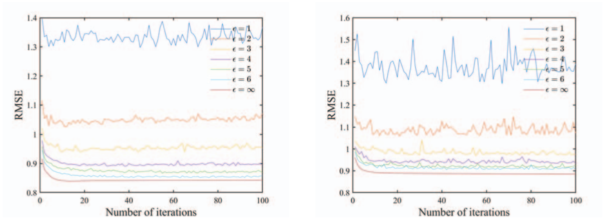
(a) MovieLens 20M (b) Netflix

Fig. 4: RMSE of Quantized ADMM-DPT

Figure 3 shows the accuracy curve of ADMM-DPTF with different numbers of GPUs. The figure shows that accuracy decreases when the number of GPUs increases. Such a trend indicates that in distributed computing, data on different GPUs are different. ADMM is used to synchronize the data on multiple GPUs, so it affects the parameter update.

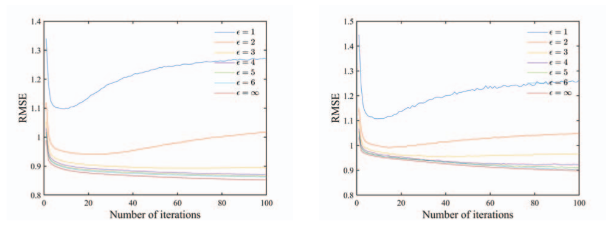
Figure 4 shows the accuracy curve of the quantized ADMM-DPTF under different numbers of GPUs. The curves of ADMM-DPTF and quantized ADMM-DPTF have little differences in the same data set, which shows that the quantization method hardly affects the accuracy.

2) Speedup of Parallel Algorithms: Table II shows the experimental results of this part. We calculated the elapsed



(a) MovieLens 20M (b) Netflix

Fig. 5: RMSE of DPTF-SVRG under various ϵ



(a) MovieLens 20M (b) Netflix

Fig. 6: RMSE of ADMM-DPTF under various ϵ

time when each algorithm executes 100 iterations. It can be seen that Hogwild! has a better acceleration effect than DPTF-SVRG, because the calculation of SVRG is more complicated: (1) It needs to calculate the global average gradient $\boldsymbol{U}_A, \boldsymbol{U}_B, \boldsymbol{U}_C$, and data reading and writing need to be locked, so it cannot be completely parallelized; (2) The update of its parameters is more complicated, and in the case of fewer registers on the GPU, its computation needs to access slower memory, which takes more time than Hogwild!.

3) Multi-GPU Speedup: We test the speedup of ADMM-DPTF with different numbers of GPUs and compare the speedup of ADMM-DPTF and quantized ADMM-DPTF.

Figure 2 shows the speedup curve of ADMM-DPTF and quantized ADMM-DPTF. It can be seen that on MovieLens 20M, both ADMM-DPTF and quantized ADMM-DPTF have super-linear speedup. It also illustrates that the quantized ADMM-DPTF has better performance, and its speedup exceeds that of the algorithm without quantization. Because in the quantization method, the computational time required for compressing and decompressing data offsets the time saved in communicating the compressed data.

4) Privacy Protection: We test the tradeoff between accuracy and privacy protection under different DP budgets ϵ. We use budgets of 1-6 to test the algorithm. Figures 5 and 6 show the accuracy of DPTF-SVRG and ADMM-DPTF under different DP budgets ϵ.

It can be seen that the results on the two datasets are similar: the results fluctuate significantly when ϵ values are small, and the accuracy is low; but when ϵ values increase, the curves get closer to the curve without DP. And the curve of the ADMM-DPTF is smoother because it reduces the result to a node, and the reduction process may combine the parameters on each GPU device, which may also correct the parameters and make the curve smoother.

VII. Conclusion and Future Work

Our work utilizes the data parallelism strategy. Section II-A states that model parallelism is another route to implementing efficient tensor factorization. Current work in model parallelism is to approximate the rank of a tensor to get the optimized core tensor [8], [9]. Such efforts can improve accuracy since tensor factorization handles the same task with tensor complement, in which low rank is essential for accuracy. In future work, we will consider strategies to combine data parallelism and model parallelism to improve accuracy and efficiency further.

References

[1] J. Ma, Q. Zhang, J. Lou, L. Xiong, and J. C. Ho, "Communication efficient federated generalized tensor factorization for collaborative health data analytics," in *Proceedings of the International World World Web Conference 2021 (WWW'21)*, 2021, pp. 171—-182.

[2] J. Feng, L. T. Yang, Q. Zhu, and K.-K. R. Choo, "Privacy-preserving tensor decomposition over encrypted data in a federated cloud environment," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, pp. 857–868, 2020.

[3] S. E. Kurt, S. Raje, A. Sukumaran-Rajam, and P. Sadayappan, "Sparsity-aware tensor decomposition," in *Proceedings of the 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS'22)*, 2022, pp. 952–962.

[4] Z.-Q. Yu, X.-J. Shi, L. Yan, and W.-J. Li, "Distributed stochastic ADMM for matrix factorization," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014, pp. 1259–1268.

[5] F. Zhang, E. Xue, R. Guo, G. Qu, G. Zhao, and A. Y. Zomaya, "DS-ADMM++: A novel distributed quantized ADMM to speed up differentially private matrix factorization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 6, pp. 1289–1302, 2022.

[6] F. Shang, Y. Liu, and J. Cheng, "Generalized higher-order tensor decomposition via parallel admm," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI'14)*, 2014, pp. 1279–1285.

[7] Y. Liu, F. Shang, W. Fan, J. Cheng, and H. Cheng, "Generalized higher-order orthogonal iteration for tensor decomposition and completion," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS'14)*, 2014, pp. 1–9.

[8] Y. Liu, Z. Long, H. Huang, and C. Zhu, "Low CP rank and Tucker rank tensor completion for estimating missing components in image data," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 4, pp. 944–954, 2020.

[9] J. Yu, G. Zhou, W. Sun, and S. Xie, "Robust to rank selection: Low-rank sparse Tensor-ring completion," *IEEE Transactions on Neural Networks and Learning Systems*, no. Early Access, pp. 1–15, 2021.

[10] Y. Zhao and J. Chen, "A survey on differential privacy for unstructured data content," *ACM Computing Surveys*, vol. 54, no. 11s, p. 207:1–28, 2022.

[11] Z. Li, B. Ding, C. Zhang, N. Li, and J. Zhou, "Federated matrix factorization with privacy guarantee," *Proceedings of the VLDB Endowment*, vol. 15, no. 4, pp. 1–14, 2021.

[12] J. Hua, C. Xia, and S. Zhong, "Differentially private matrix factorization," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 1763–1770.

[13] Z. Liu, Y.-X. Wang, and A. Smola, "Fast differentially private matrix factorization," in *Proceedings of the 9th ACM Conference on Recommender Systems*, 2015, pp. 171–178.

[14] H. Imtiaz and A. D. Sarwate, "Distributed differentially private algorithms for matrix and tensor factorization," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 6, pp. 1449–1464, 2018.

[15] J. Wang, H. Han, H. Li, S. He, P. K. Sharma, and L. Chen, "Multiple strategies differential privacy on sparse tensor factorization for network traffic analysis in 5g," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 3, pp. 1939–1948, 2022.

[16] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proceedings of theAdvances in neural information processing systems (NIPS'13)*, 2013, pp. 26:1–9.

[17] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[18] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild!: A lock-free approach to parallelizing stochastic gradient descent," in *Proceedings of the Advances in neural information processing systems (NIPS'11)*, 2011, pp. 24:1–9.

[19] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.