Weekly Research Progress Report

Student: 从 兴 **Date:** 2023/12/21 - 2024/1/16

List of accomplishments this week

1. 整理之前所有的周报以及相关论文，共享到北航云盘

2. 建立一个项目专门的 github 仓库，仓库地址为：https://github.com/BUAAer-xing

3. 梳理 2023 年所学到的知识，整理所总结的笔记，并利用 github 搭建了一个博客（博客地址为：https://buaaer-xing.github.io），记录每一篇论文的笔记、周报和相关技术。

4. 阅读 HiCOO 篇论文

Paper summary

Name：HiCOO-Hierarchical Storage of Sparse Tensors

Motivation：数据张量通常是稀疏的，因此并不需要显式存储或操作，而如何选择稀疏张量的数据结构对于后面的计算至关重要，目前，影响数据结构选择的两个关键问题是紧凑性和模式定向，如果要求结构是紧凑的，则需要对固定方向上的坐标进行压缩，类似于 csr 格式，但是大多数进行张量计算时，需要按照多个方向进行遍历张量，为了保证效率，需要存储多个方向压缩的副本，但这样虽然结构紧凑了，存储空间却大了，因此采用 coo 格式进行存储可以对多个方向的遍历计算具有一定的通用性，但 coo 格式不紧凑，为此，该团队想要提出一种新的存储张量的数据结构，来实现更加通用并且紧凑的存储。

Solution：该团队的解决方法是提出一个层级坐标格式–HiCOO，类似于 cache 中的组相联映射，利用固定大小的块对整个张量中的非零元素进行分块处理，使得多个非零元的坐标统一可以由一个大块的坐标来进行表示，然后用极少 bit 来指示每个小块中非零元所在位置，这样就使结构更加紧凑，同时可以借助公式，实现从 HiCOO 到 COO 格式的快速转化。

Related to us：为新的数据结构存储方式开拓新的思路

Work summary

1. 整理周报以及相关论文，共享到北航云盘

在北航云盘中主要保存阅读过的相关论文、提交的周报以及和实验室项目相关的文档。

图 1 共享到北航云盘的资料

2. 建立项目专门的 Github 仓库

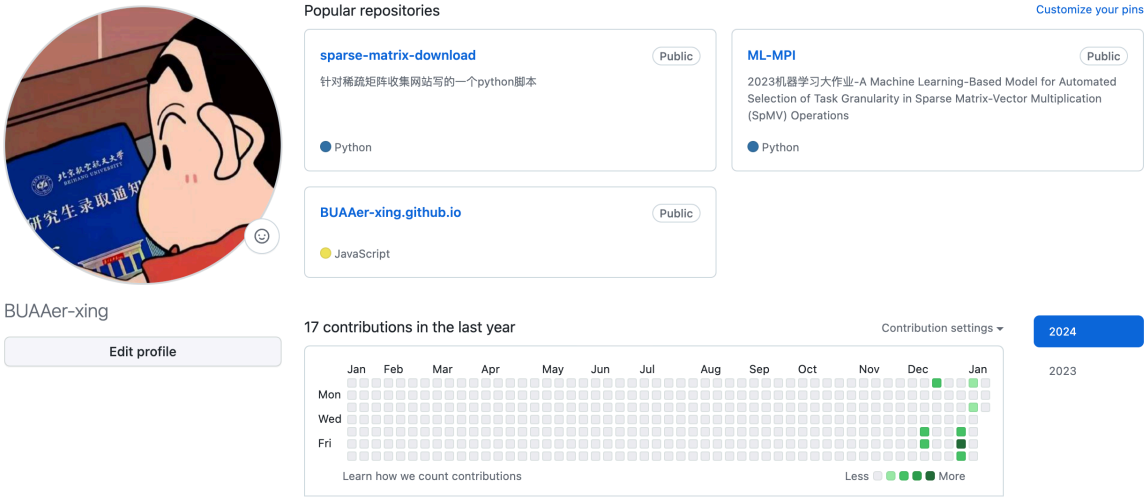

图 2 硕士研究生阶段的 github 仓库

仓库中上传了从稀疏矩阵网站获取资源的脚本、上次 ML 课程的源代码以及博客搭建的源代码。

3. 利用 github 搭建博客

博客地址：https://buaaer-xing.github.io

梳理 2023 年所学到的知识，整理所总结的笔记，并利用 github 搭建了一个博客，记录每一篇论文的笔记、周报和相关技术。方便以后的复习以及知识的继承。

图 3 各个论文的阅读笔记

4. HiCOO 论文

(1) 提出的背景

在面向张量的数据分析中，**一个核心问题是选择什么样的数据结构来组织张量**，数据张量通常是稀疏的，即大部分条目为零，并不需要显式存储或操作。因此，选择稀疏张量数据结构的问题类似于经典问题之一：如何选择稀疏矩阵格式。

影响格式选择的两个关键问题是 **紧凑性**和 **模式定向**。

- 紧凑性指的是保持总字节数较小。
- 模式定向是指一种格式是否偏好以特定顺序迭代张量模式。比如，二维矩阵的 CSR 压缩存储（行：mode-1，列：mode-2）特定顺序为 1<2，而在 CSC 压缩存储中，特定顺序为 2<1。

扩展到张量中，有 CSF(compressed sparse fiber)，因为稀疏张量分析方法在同一计算过程中可能需要对某个方向进行**多次迭代**。对于固定的存储格式，在一种方向上迭代可能很快，但在方向切换时可能变慢。**对于低阶（小 N）张量（例如矩阵），存储多个矩阵副本以缓解这种影响可能是可行的**。例如，可以将矩阵以 CSR 和 CSC 格式进行存储，并在需要时使用适当的格式，因此阶数不再重要。但**对于张量，随着阶数 N 的增长，多副本策略可能会变得不可行**。

稀疏张量的最简单、也或许是最受欢迎的存储格式是 COO 格式。**COO 存储不是模式特定的，而是通用模式**。但是，这种格式（COO）不如 CSF 等格式紧凑，因为 CSF

可以利用方向特异性来减少每个非零项的平均索引元数据量（即 ik 值）。就是压缩横坐标的存储或者压缩纵坐标的存储空间。

所以该文章提出了一种新的格式-HiCOO，试图实现既紧凑又通用的张量存储格式。

(2) 对三种格式进行理论分析

根据紧凑性和模式取向的标准，以及它们对张量计算（比如 MTTKRP）的预期行为，对 COO、CSF 和 F-COO 格式进行了比较和分析。

TABLE II
THE ANALYSIS OF TENSOR FORMATS AND THEIR MTTKRP ALGORITHMS FOR A THIRD-ORDER TENSOR ($N = 3$) WITH M NONZERO ENTRIES. THE WORD SIZE PARAMETERS ARE $\beta_{\text{INT}} = 32$, $\beta_{\text{LONG}} = 64$, $\beta_{\text{BYTE}} = 8$, AND $\beta_{\text{FLOAT}} = 32$ BITS FOR SINGLE-PRECISION FLOATING-POINT VALUES AND DISCARDING INSIGNIFICANT ITEMS.

	Data Structure		MTTKRP Behavior		
Format	Index Space (Bits)	Update Needed?	Work (Flops)	Memory Access (Bytes)	Arithmetic Intensity (AI)
COO	$96M$	NO	$3MR$	$12MR$	$1/4$
F-COO	$65M$	YES	$3MR$	$12MR$	$1/4$
CSF	$[32M, 128M]$	YES[1]	$[2MR, 4MR]$	$[8MR, 16MR]$	$1/4$
HiCOO	$[24M, 184M]$	NO	$3MR$	$min\{\frac{12}{c_b}, 12\}MR$	$max\{\frac{1}{4}, \frac{c_b}{4}\}$

[1] MTTKRPs in all tensor modes can use less CSF representations than tensor order or even only one CSF representation with performance payoff.

图 4 几种存储格式的对比分析结果

- **COO 格式**：将每个非零值与其所有位置索引一起存储
- **CSF 格式**：CSF（压缩稀疏 fiber）是一种分层的、以 fiber 为中心的格式，有效地将 CSR 矩阵格式推广到张量。CSF 将非零值组织成树形结构。每个级别对应一个

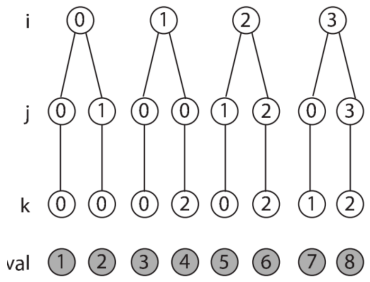

图 5 CSF-1 格式举例

张量模式，每个非零值都是从根节点到叶子节点的路径。由于路径暗示了枚举非零值的模式顺序，CSF 具有很强的模式特异性。对于一个 N 阶张量，如果需要在每个模式中迭代多次进行张量操作以获得最佳性能，则需要 N 个 CSF 树，如 [35] 所述。类比的情况是，在计算中既需要并行矩阵-向量乘法又需要矩阵转置-向量乘法，并且没有写冲突；一种简单快速的方法是按照前人的研究，将矩阵同时存储为 CSR 和 CSC 格式，但这会使空间成本加倍。然而，存储多个 CSF 树会消耗大量额外的存储空间。

- **F-COO 格式**：像 COO 一样，F-COO 存储非零张量元素。然而，F-COO 不受负载不平衡的影响，并且在不同模式的稀疏张量上运行时能保持最大的平行度。而且，

与 COO 类似，F-COO 对底层稀疏张量结构的不规则性不敏感。F-COO 使用两个

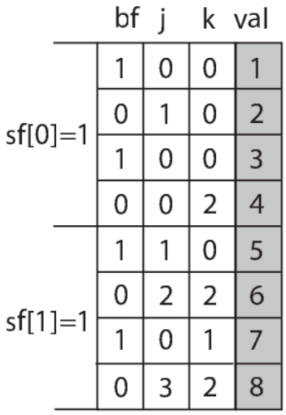

图 6 F-COO 格式

标志数组，即位标志（bf）和起始标志（sf）。

- bf 数组用于表示索引模式的任何更改，从而显示计算已切换到另一个 fiber（在 SpTTM 中）或切换到另一个 slice（在 SpMTTKRP 中）。
- F-COO 还配备了一个起始标志（sf），用于指示新的开始是否在当前分区内。

(3)HiCOO 格式的转换过程

首先总结 HiCOO 格式的特点：**就是以稀疏张量块为单位压缩张量索引，并利用较短的整数类型来表示块内的偏移量，从而实现整体的紧凑性并保持 COO 本身的通用性。**

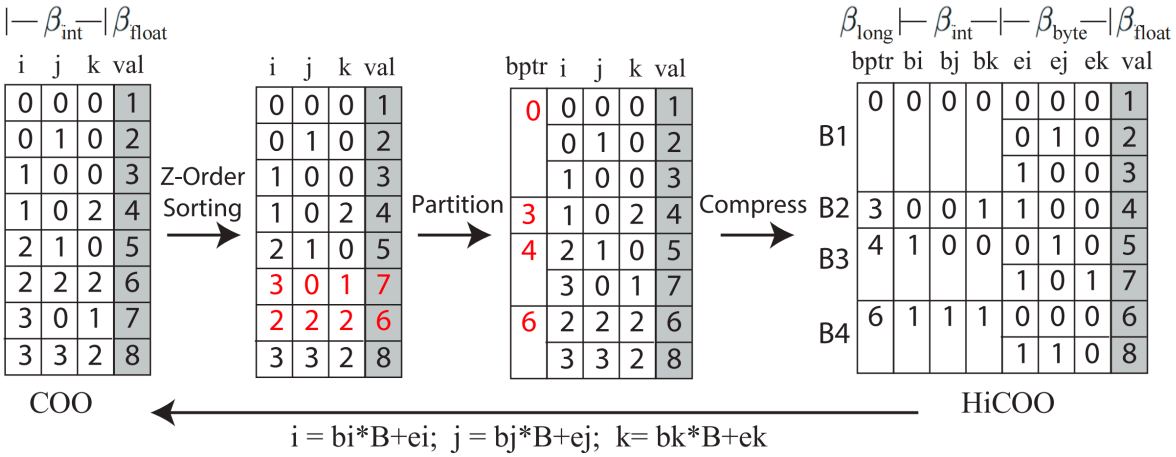

图 7 从 COO 格式转换为 HiCOO 格式的过程

从 COO 转换为 HiCOO 的三个步骤：

- **排序**
  - 使用快速排序的变体，按 **Z-Morton 顺序**对 COO 张量的所有非零值进行排序

* Morton 码是一种将多维空间中的坐标映射到一维空间的方法，它与 Z-Morton 曲线有关联。在 Morton 码中，**相邻的多维坐标被编码为相邻的一维索引**，这种编码方式使得**在多维空间中的数据能够以线性方式进行存储和访问**。
* Z-Morton 曲线和 Morton 码的特点使得它们在空间索引数据结构（比如四叉树、八叉树等）、并行计算、以及一些计算几何算法中有着广泛的应用，因为**它们能够有效地处理多维数据，并提供高效的访问方式**。

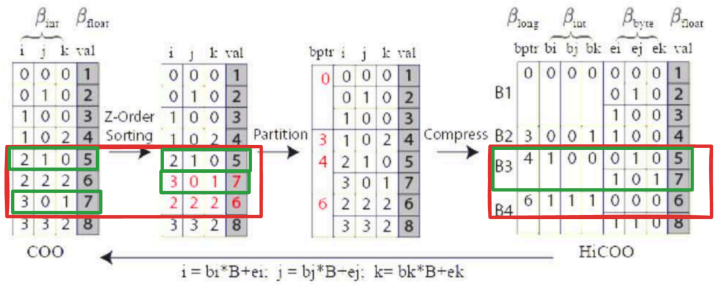

图 8 经过排序后的对比

– 进行完该步骤以后，可以使得相邻块的坐标在转换为 HiCOO 坐标格式时，也在相邻，为后面的压缩做准备。

- **分割**
  – 通过给定的块大小，对已知的大张量块进行划分、分割。
- **压缩**
  – 最后，将 COO 指标相应地压缩为**块指标**和**块内元素指标**。

同时，转换为 HiCOO 坐标格式后，之后并不需要再将其转换回 COO 格式，直接利用下方的公式，直接可以由 HiCOO 格式得到 COO 格式，从而使得 HiCOO 格式存储的矩阵直接参加运算。

$$i = b_i \cdot B + e_i$$

$$j = b_j \cdot B + e_j$$

$$k = b_k \cdot B + e_k$$

HiCOO 格式存储的 value 值仍然是 COO 格式下存储的 value 值。

说明：

- b_i, b_j, b_k 指示的是**块指针**，它的存储类型是 int 类型（64 位）
- e_i, e_j, e_k 指示的是**块内位置**，它的存储类型是 byte 类型（8 位）
- $bptr$ 数组中的元素的存储类型为 long 类型，它存储的是**每个块的开始位置**

综上所述，在 HiCOO 格式中，Z-Morton 排序为张量算法提供了更好的**数据局部性**，而压缩索引**节省了稀疏张量的存储空间**，同时也**减少了张量访问的存储带宽**。

(4) 加速 MTTKRP 操作

通过使用超级块调度器和两种并行化策略，在基于 HICOO 的多核 CPU 架构上加速 MTTKRP。将**一组小块分组成一个大而逻辑的超级块**。超级块中的块总是一起调度并分配给单个线程。在超级块中，以相同模式物理存储非零。

Next

读 linpack 的相关使用和源代码，在源代码的基础上进行相关论文实现的改进，观察效果。

继续阅读 Optimizing High-Performance Linpack for Exascale Accelerated Architectures 这篇论文。