第三次作业说明

姚杰男

2023 年 11 月 29 日

1 题目背景

稀疏矩阵-稀疏矩阵乘法 (SpGEMM)，是在当前科学计算应用里十分常用和重要的一个计算核心。

$C = A*B$, 其中，A 和 B 都是稀疏矩阵，相乘得到稀疏矩阵 C。SpGEMM 作为第三层稀疏 BLAS 中的关键操作，在稀疏线性解法器、图处理框架以及机器学习算法等多个场合都具有广泛的应用。

稀疏矩阵有多种不同的存储格式，而本次我们给出的输入矩阵默认以 CSR(Compressed Sparse Rows) 格式存储，如下图所示。

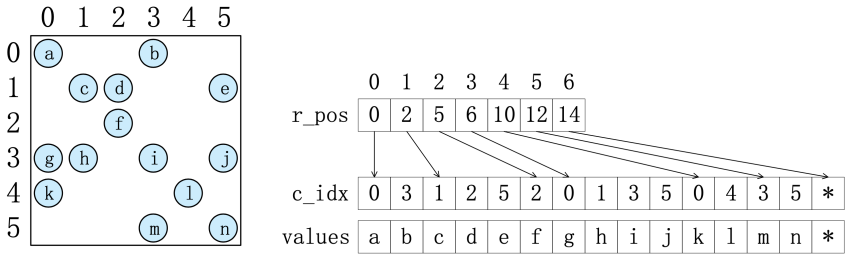

图 1: CSR 数据结构示意图

矩阵的非零元分布由 3 个数组表示：

1.r_pos[i] 表示第 i 行第一个非零元的位置，r_pos[i+1] - r_pos[i] 表示第 i 行非零元的数目。

2.c_idx[j] 表示第 j 个非零元所在原矩阵的列下标。

3.value[j] 表示第 j 个非零元的数值。矩阵的行号和列号下标均从 0 开始，非零元也是从 0 开始编号。

2 问题描述

2.1 要求

1. 独立完成代码实现与优化。

2. 提交文件夹命名格式为学号 + 姓名 +h3, 如: 2023000000_name_h3, 其中包含文件夹 spgemm 和报告 report.pdf。

3. 注意 DDL，以网络学堂发布的为准。

2.2 任务

在 **GPU 单卡**上，对**单精度**稀疏矩阵-稀疏矩阵乘法进行并行实现和性能优化。

解压 homework3.zip 后，其中的 spgemm 目录即是稀疏矩阵-稀疏矩阵乘法的基础代码。

gemm-optimized.c 为本次作业需要优化的文件。run.sh 为单个矩阵测试脚本，run_all.sh 为最终测试脚本。

程序运行方式参考 run.sh 和 run_all.sh。

2.3 要求

1. 本次选取的矩阵均为方阵, spgemm 计算的是矩阵的平方 $C = A * A$。已公开的测试矩阵共 7 个，主要为中等规模的矩阵，需要通过正确性验证，相对精度误差小于 1e-6 即可。另有部分未公开的测试矩阵在提交作业进行评估时会一同参与最终的性能测试，请大家注意代码的可扩展性和灵活性，不建议假定相乘的两个矩阵为相同矩阵来进行访存和计算过程的简化。

2. 测试结果应分别包括预处理时间（包括矩阵布局转换，重新排序等）和 SpGEMM 的实际计算时间，请在报告中分别体现。

2.4 评分

2.4.1 稀疏矩阵-稀疏矩阵乘

1. 通过正确性检测 (25%)

2. 评测 SpGEMM 的性能结果，主要按照全部测试矩阵几何平均的实际计算的性能作为评测结果，结合预处理耗时，进行打分 (40%)

3. **详细描述**在实现 SpGEMM 中采取的优化手段，代码对应的部分，以及对应的实验结果，解释目前取得的性能结果 (35%)。

2.5 提示

1. 部分测试样例中每行的非零元分布不均，若按照行划分并行任务，可能会导致负载不均衡。

2. CSR 不一定是最优的矩阵存储格式，可以在预处理过程中做转换。

3. 可以使用性能工具分析结果，鼓励大家用性能模型分析性能行为，帮助开展性能优化。

4. 有进一步的问题请与助教和老师及时交流。

3 参考资料

SpGEMM 在 GPU 上的主要优化工作如下：

TileSpGEMM [1] 是近年 PPoPP 的工作。早些时间的工作还有 bhSPARSE [2], RMERGE [3], nsparse [4], Register_based_SpGEMM [5] 以及 SpECK [6]。

大家可以阅读以上的文献，也可以在网上自行查找其他的相关文献。

参考文献

[1] Yuyao Niu, Zhengyang Lu, Haonan Ji, Shuhui Song, Zhou Jin, and Weifeng Liu. Tilespgemm: a tiled algorithm for parallel sparse general matrix-matrix multiplication on gpus. In *Proceedings of the 27th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 90–106, 2022.

[2] Weifeng Liu and Brian Vinter. An efficient gpu general sparse matrix-matrix multiplication for irregular data. In *2014 IEEE 28th international parallel and distributed processing symposium*, pages 370–381. IEEE, 2014.

[3] Felix Gremse, Andreas Hofter, Lars Ole Schwen, Fabian Kiessling, and Uwe Naumann. Gpu-accelerated sparse matrix-matrix multiplication

by iterative row merging. *SIAM Journal on Scientific Computing*, 37(1):C54–C71, 2015.

[4] Yusuke Nagasaka, Akira Nukada, and Satoshi Matsuoka. High-performance and memory-saving sparse general matrix-matrix multiplication for nvidia pascal gpu. In *2017 46th International Conference on Parallel Processing (ICPP)*, pages 101–110. IEEE, 2017.

[5] Junhong Liu, Xin He, Weifeng Liu, and Guangming Tan. Register-aware optimizations for parallel sparse matrix–matrix multiplication. *International Journal of Parallel Programming*, 47(3):403–417, 2019.

[6] Mathias Parger, Martin Winter, Daniel Mlakar, and Markus Steinberger. Speck: Accelerating gpu sparse matrix-matrix multiplication through lightweight analysis. In *Proceedings of the 25th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 362–375, 2020.