计算机学院

并行程序设计实验报告

高斯消去并行优化

姓名：赵廷枫

学号：2112529

专业：计算机科学与技术

2023 年 7 月 1 日

目录

1 问题描述

1.1 普通高斯消去

高斯消元（Gaussian elimination）是一种用于解线性方程组的基本算法，可以解决线性方程组求解中很多重要问题 [?]，适用于任何数量的未知数。其基本思想是通过一系列的元素操作，将一个线性系统转化为一个更易解的等价系统。

高斯消去法一般包括两个步骤：消元和回代

- 将系数矩阵转化为阶梯形矩阵（Row-Echelon form）。在这个阶段，算法从上到下逐行处理，通过行变换操作（交换、缩放、行相加），使得每一行下方的行在该行的主元位置（通常为对角线上的元素）左边的元素都变为 0.
- 在得到阶梯形矩阵后，算法从下到上进行回代，通过行变换将主元上方的元素都变为 0，从而得到行简化阶梯形矩阵（Row-Reduced Echelon form）.

高斯消去法具有极为广泛的应用，高斯消去法衍生出来的很多相关的算法也都具备广泛的应用前景。比如线性最小二乘问题，在统计学和机器学习中，线性最小二乘问题是一种常见的优化问题，高斯消元法可以用于求解此类问题。再比如在电路分析中，基尔霍夫定律等会形成线性方程组，可以用高斯消元法求解。

由此可见，高斯消去是一项非常基础的运算，如果我们能够对其进行加速，可以从根本上提升很多问题的计算效率。就像是矩阵乘法的加速抑或是卷积运算求积的加速，这些都是极为基础的运算，因此对高斯消去的过程进行并行加速是极为必要且重要的。

基于高斯消去过程的特点，我们尝试对高斯消去进行并行优化：SIMD 优化、Pthread 优化、OpenMP 优化、MPI 优化、GPU 编程、OneAPI 编程。

1.2 普通高斯消去主要难题

通过对高斯消去两个主要过程的分析，可以观察到高斯消去在并行化的处理过程中主要面临下面几个难题：

- **数据依赖性**：高斯消去法包含大量的数据依赖，这意味着某些计算必须等待其他计算完成。这种数据依赖性往往限制了并行性的潜力。
- **负载均衡**：高斯消去法在进行过程中，每一步所需要处理的数据量都在不断减少。这使得在并行环境中保持所有处理器的负载均衡变得困难。
- **通信开销**：在并行计算中，处理器之间的数据交换和同步通常需要显著的时间开销。在高斯消去法中，这个问题尤其明显，因为每一步的计算都可能需要访问和更新公共的数据结构。
- **数据局部性**：高斯消去法的数据访问模式可能导致较低的缓存利用率和较高的内存访问延迟。

在接下来具体的实现和分析中，也会重点优化和分析上述四个部分，从而实现高斯消去的最优化处理。

1.3 Grobner 基

Grobner 基理论的形成，经历了几十年的时间。1927 年，F.S.Macauly 为了研究理想的某些不变量，将全序的概念引入到由多变元多项式环中单项式全体组成的集合内。随后，1964 年，H.Hironaka 在研

究奇性分解时，引入了多变元多项式的除法算法。1965 年，奥地利数学家 B.Buchberger 使用除法算法系统地研究了域上多变元多项式环的理想生成元问题，在单项式的集合中引入了保持单项式的乘法运算的全序 (称为项序)，以保证多项式相处后所得余多项式的唯一性。B.Buchberger 引入了 S-多项式，使得对多项式环中的任一给定的理想，从它的一组生成元出发，可以计算得到一组特殊的生成元，也就是 Grobner 基。并且，Buchberger 设计了计算多元多项式理想的 Grbner 基算法 (称为 Buchberger 算法)，提出了优化该算法的若干准则 (Buchberger 第一、第二准则)。Grobner 基方法是求解非线性代数系统的一种非数值代的代数方法。其基本思想是，在原非线性多项式代数系统所构成的多项式环中，通过对变量和多项式的适当排序，对原系统进行约简，最后生成一个与原系统的等价且便于直接求解的标准基 (Grobner 基)。Grobner 基的理论和算法提供了一种标准的方法，能够很好地解决可以被表示成多变元多项式集合中的项的形式所构成的很多问题，在代数几何、交换代数和多项式理想理论、偏微分方程编码理论、统计学、非交换代数系统理论等有广泛的应用。

1.4 特殊高斯消元

在布尔 Grobner 基的计算中，提出了一种特殊的高斯消元方法: 消元子模式的高斯消元算法。数据特点:

- 111
- 111
- 111
- 111
- 111

运算过程:

- 111
- 111
- 111
- 111
- 111

1.5 特殊高斯消元的主要难题

- 实验数据给的是稀疏矩阵 (并且存放于磁盘文件)，如何读取数据并将稀疏矩阵转换为稠密矩阵，以及用什么数据类型存放稠密矩阵。
- 受数据类型的限制，如何判断某行的消元子是否为空，如何获取被消元行的首项所在位置。
- 如何设计算法使得其易于并行化。
- 如何设计分批操作，以及批次覆盖不到的部分的操作

2 期末报告相关说明

2.1 文章结构说明

本文主要集中于普通高斯消去的实验和特殊高斯消去的实验，所以主要分为两大章节：普通高斯消去和特殊高斯消去。其中，在每一个章节中，我们分别讨论了并行程序设计与分析这门课程中讲解的主要内容：**SIMD 优化、Pthread 优化、OpenMP 优化、MPI 优化、GPU 编程、OneAPI 编程**。

在普通高斯消去部分，因为我们前面的实验已经完成了相关的工作，所以在此报告中主要做一些整合以及一些深入的分析，比如在适合的部分**分析汇编代码**以及**使用 Vtune 性能分析工具**对程序进行全面分析，从而剖析加速效果产生的底层机制与深层次原因。

在特殊高斯消去部分，我们则侧重于分析具体**代码中的核心步骤**，分析**并行思想在其中的运用以及具体的实现方式**，还会对实验相关数据进行分析，从而得到**不同的并行机制对特殊高斯消去的加速效果**，并尝试探讨加速效果深层次的原因。

2.2 实验平台环境配置

表 1: 本机 X86 平台

Operating System	windows 10
Collector Type	Event based sampling driver
CPU Name	Intel(R) microarchitecture code named Tigerlake
CPU Frequence	3.1GHz
Logical CPU count	8
L1 cache	320 KB
L2 cache	5.0MB
L3 cache	8.0MB

表 2: 鲲鹏 arm 平台

Architecture	aarch64
CPU(s)	96
cores per socket	48
CPU MAX MHZ	2600.0000
CPU MIN MHZ	200.0000
L1d cache	64k
L1i cache	64k
L2 cache	5512k
L3 cache	49152k

表 3: GPU 相关参数

Architecture	aarch64
GPU Name	NVIDIA GeForce RTX 4060
MAX MHZ	1665
Average MHZ	1410
Memory	8GB
Memory Ratio	14 Gbps
Floating Point Performance	16.20 TFLOPs

2.3 实验源代码仓库地址

源码链接：https://github.com/2112529/Parallel/tree/main/Final_work

说明：本报告中涉及到的所有代码都在这一个库中，因为后面章节的篇目较多，所以不再一一标明源代码仓库地址，只在本节说明，具体的每一部分的源代码请根据仓库中的文件夹名称对应相应的模块进行查找。

3 普通高斯消去

3.1 SIMD 优化

3.2 Pthread 优化

3.3 OpenMP 优化

3.4 MPI 优化

3.5 GPU 编程

3.6 OneAPI 编程

4 特殊高斯消去

4.1 SIMD 优化

4.2 Pthread 优化

4.3 OpenMP 优化

4.4 MPI 优化

4.5 GPU 编程

4.6 OneAPI 编程

5 课程总结

并行程序设计与分析这门课程介绍了很多并行优化的机制，对其总结如下：

- SIMD:
- Pthread:
- OpenMP:
- MPI:
- GPU:
- OneAPI:

总结而言，受益匪浅。