



南開大學
Nankai University

计算机学院
并行程序设计第 6 次作业

高斯消去法的 CUDA 并行化

姓名：丁屹
学号：2013280
专业：计算机科学与技术

2022 年 6 月 15 日

目录

| | |
|-------------------------|----------|
| 1 问题描述 | 2 |
| 2 OpenMP 算法设计 | 3 |
| 2.1 测试用例的确定 | 3 |
| 2.2 实验环境和相关配置 | 3 |
| 2.3 算法设计 | 3 |
| 2.3.1 默认平凡算法 | 3 |

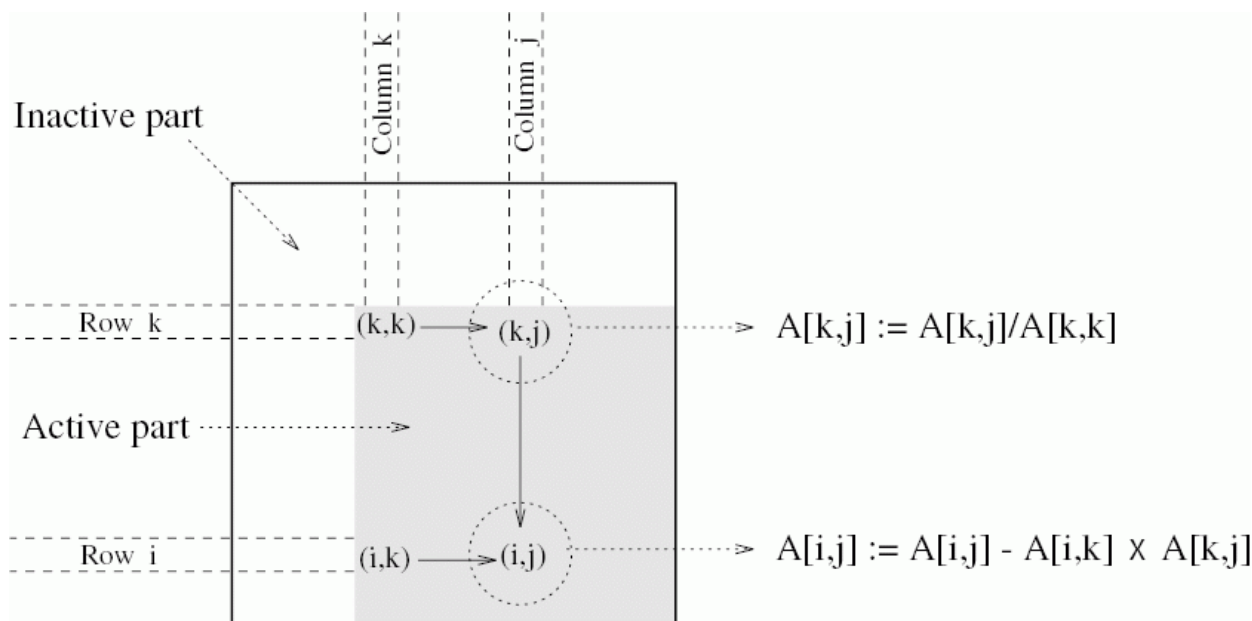


图 1.1: 高斯消去法示意图

1 问题描述

高斯消去的计算模式如图 1.1 所示，在第 k 步时，对第 k 行从 (k, k) 开始进行除法操作，并且将后续的 $k + 1$ 至 N 行进行减去第 k 行的操作，串行算法如下面伪代码所示。

Algorithm 1 普通高斯消元算法伪代码

```

1: function LU
2:   for  $k := 0$  to  $n$  do
3:     for  $j := k + 1$  to  $n$  do
4:        $A[k, j] := A[k, j] / A[k, k]$ 
5:     end for
6:      $A[k, k] := 1.0$ 
7:     for  $i := k + 1$  to  $n$  do
8:       for  $j := k + 1$  to  $n$  do
9:          $A[i, j] := A[i, j] - A[i, k] * A[k, j]$ 
10:      end for
11:       $A[i, k] := 0$ 
12:    end for
13:  end for
14: end function

```

观察高斯消去算法，注意到伪代码第 4, 5 行第一个内嵌循环中的 $A[k, j] := A[k, j] / A[k, k]$ 以及伪代码第 8 9 10 行双层 *for* 循环中的 $A[i, j] := A[i, j] - A[i, k] \times A[k, j]$ 都是可以进行向量化的循环。可以通过 CUDA 对这两步进行并行优化。

2 OpenMP 算法设计

源码链接: <https://github.com/ArcanusNEO/Parallel-Programming/tree/master/6>

2.1 测试用例的确定

由于测试数据集较大, 不便于各个平台同步, 所以采用固定随机数种子为 12345687 的 mt19937 随机数生成器。经过实验发现不同规模下, 所有元素独立生成, 限制大小在 $[0, 100]$, 能够生成可以被正确消元的矩阵。

代码如下:

测试数据集生成器

```
1 uniform_real_distribution<float> dist(0, 100);
2 mt19937 mt(12345687);
3 int n;
4 istream iss(argv[1]);
5 iss >> n;
6 cout << n << endl;
7 for (int i = 1; i <= n; ++i)
8     for (int j = 1; j <= n; ++j) cout << dist(mt) << " \n"[j == n];
```

2.2 实验环境和相关配置

实验在本地 Arch Linux x86_64 平台完成, 使用 cmake 构建项目;

- nVIDIA GPU 型号: NVIDIA Corporation TU106M [GeForce RTX 2060 Max-Q]
- 驱动版本: Driver Version: 515.48.07
- CUDA 版本: CUDA Version: 11.7

2.3 算法设计

2.3.1 默认平凡算法

使用一维数组模拟矩阵, 避免改变矩阵大小时第二维不方便调整、必须设成最大值的问题, 可以减少 cache 失效;

使用 `#define matrix(i, j) arr[(i) * n + (j)]` 宏, 增强可读性;

平凡算法

```
1 #define matrix(i, j) arr[(i) * n + (j)]
2 void func(int& ans, float arr[], int n) {
3     for (int k = 0; k < n; ++k) {
4         for (int j = k + 1; j < n; ++j) matrix(k, j) = matrix(k, j) / matrix(k, k);
5         matrix(k, k) = 1.0;
6         for (int i = k + 1; i < n; ++i) {
7             for (int j = k + 1; j < n; ++j)
8                 matrix(i, j) = matrix(i, j) - matrix(i, k) * matrix(k, j);
```

```
9      matrix(i, k) = 0;
10    }
11  }
12  #undef matrix
13 }
```