Разработка параллельных программ на основе MPI для решения задач линейной алгебры

Исполнители проекта:
Блинова Екатерина
Сидоров Андрей
ФПМИ, 1-й курс

Руководитель проекта:
Черникова Анна

План доклада:

- Решение СЛАУ методом Гаусса.
- Использование MPI. (Параллельная реализация метода Гаусса)

Цели:

- Решить СЛАУ методом Гаусса.
  - Теоретическое освоение методов решения систем линейных алгебраических уравнений прямыми методами.
  - Алгоритм решения СЛАУ методом Гаусса.
  - Параллельная реализация решения СЛАУ методом Гаусса

Метод Гаусса.

Требуется найти решение системы линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + \ldots + a_{1n}x_n = b_1 \\ \ldots \\ a_{m1}x_1 + \ldots + a_{mn}x_n = b_m \end{cases} \iff Ax = b, \quad A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \ldots & & \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}. \quad (1)$$

- Матрица A называется основной матрицей системы, b — столбцом свободных членов.
- Тогда согласно свойству элементарных преобразований над строками основную матрицу этой системы можно привести к ступенчатому виду(эти же преобразования нужно применять к столбцу свободных членов):

$$\begin{cases} \alpha_{1j_1}x_{j_1} + \alpha_{1j_2}x_{j_2} + \ldots + \alpha_{1j_r}x_{j_r} + \ldots + \alpha_{1j_n}x_{j_n} = \beta_1 \\ \alpha_{2j_2}x_{j_2} + \ldots + \alpha_{2j_r}x_{j_r} + \ldots + \alpha_{2j_n}x_{j_n} = \beta_2 \\ \ldots \\ \alpha_{rj_r}x_{j_r} + \ldots + \alpha_{rj_n}x_{j_n} = \beta_r \\ 0 = \beta_{r+1} \\ \ldots \\ 0 = \beta_m \end{cases}, \qquad \alpha_{1j_1}, \ldots, \alpha_{rj_r} \neq 0.$$

Алгоритм метода Гаусса

Алгоритм решения СЛАУ методом Гаусса подразделяется на два этапа.

На первом этапе осуществляется так называемый **прямой ход**, когда путём элементарных преобразований над строками систему приводят к ступенчатой или треугольной форме, либо устанавливают, что система несовместна.

На втором этапе осуществляется так называемый **обратный ход**, суть которого заключается в том, чтобы выразить все получившиеся базисные переменные через небазисные и построить фундаментальную систему решений, либо, если все переменные являются базисными, то выразить в численном виде единственное решение системы линейных уравнений.

$$
\begin{aligned}
(2) \to (2) - (1) \cdot \left(\tfrac{a_{21}}{a_{11}}\right) \quad &: \quad a'_{22} \cdot x_2 + a'_{23} \cdot x_3 + \ldots + a'_{2n} \cdot x_n = b'_2 \\
(3) \to (3) - (1) \cdot \left(\tfrac{a_{31}}{a_{11}}\right) \quad &: \quad a'_{32} \cdot x_2 + a'_{33} \cdot x_3 + \ldots + a'_{3n} \cdot x_n = b'_3 \\
&\ldots \\
(m) \to (m) - (1) \cdot \left(\tfrac{a_{m1}}{a_{11}}\right) \quad &: \quad a'_{m2} \cdot x_2 + a'_{m3} \cdot x_3 + \ldots + a'_{mn} \cdot x_n = b'_n \\
(3) \to (3) - (2) \cdot \left(\tfrac{a'_{32}}{a'_{22}}\right) \quad &: \quad a''_{33} \cdot x_3 + \ldots + a''_{3n} \cdot x_n = b''_3 \\
&\ldots \\
(m) \to (m) - (m-1) \cdot \left(\frac{a^{(m-2)}_{m,n-1}}{a^{(m-2)}_{m-1,n-1}}\right) \quad &: \quad a^{(m-1)}_{mm} \cdot x_m + \ldots + a^{(m-1)}_{mn} \cdot x_n = b^{(m-1)}_m
\end{aligned}
$$

Распределение данных

В предложенном алгоритме исходная матрица коэффициентов A и вектор правых частей F разрезаны горизонтальными полосами, как показано на рис. Каждая полоса загружается в соответствующий компьютер: нулевая полоса – в нулевой компьютер, первая полоса – в первый компьютер, и т. д., последняя полоса – в p1-1 компьютер. Здесь, в примере, каждая ветвь генерирует свои части матрицы коэффициентов A и вектор правых частей F.

Реализация распределения данных

- При прямом ходе матрица приводится к треугольному виду последовательно по компьютерам. Вначале к треугольному виду приводятся строки в нулевом компьютере, при этом нулевой компьютер последовательно, строка за строкой, передает свои строки остальным компьютерам, начиная с первого. Затем к треугольному виду приводятся строки в первом компьютере, передавая свои строки остальным компьютерам, начиная со второго, т.е. компьютерам с большими номерами, и т. д.

После прямого хода полосы матрицы A в каждом узле будут иметь вид:

0

```
1$$$$$$$$$$$$$$$
01$$$$$$$$$$$$$$
001$$$$$$$$$$$$$
0001$$$$$$$$$$$$
```

1

```
00001$$$$$$$$$$
000001$$$$$$$$$
0000001$$$$$$$$
00000001$$$$$$$
```

2

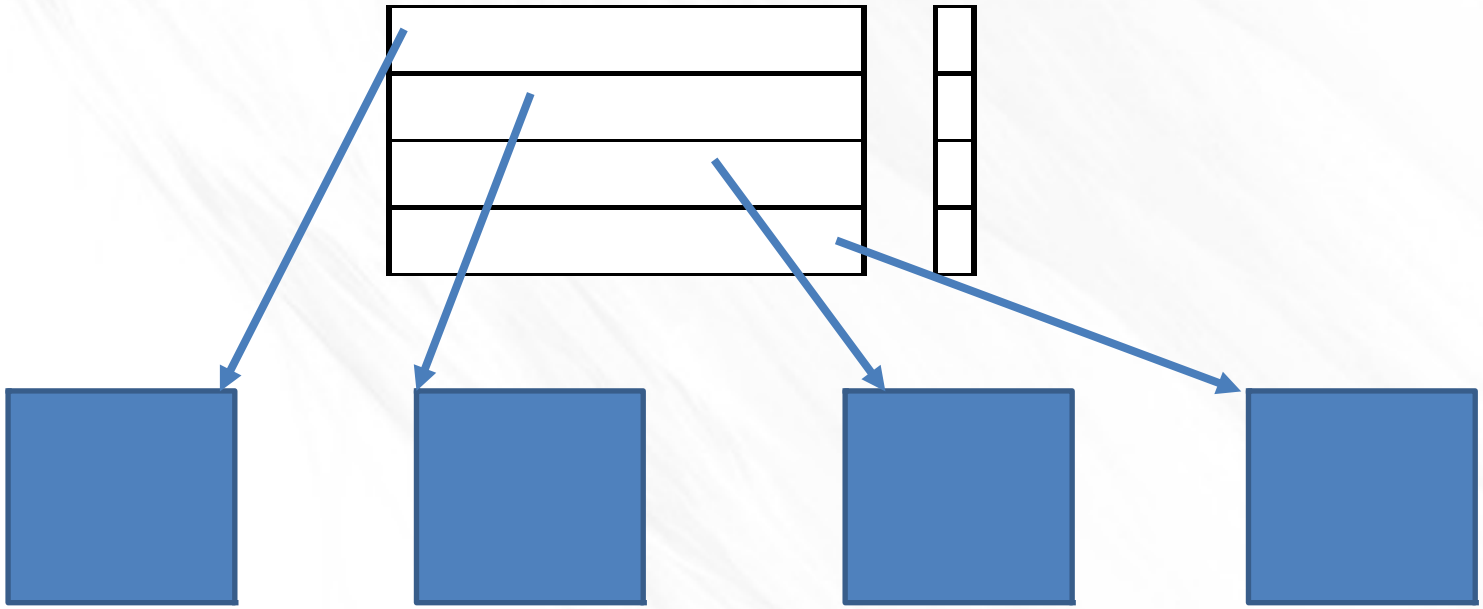
```
000000001$$$$$$
0000000001$$$$$
00000000001$$$$
000000000001$$$
```

3

```
0000000000001$$
00000000000001$
000000000000001
```

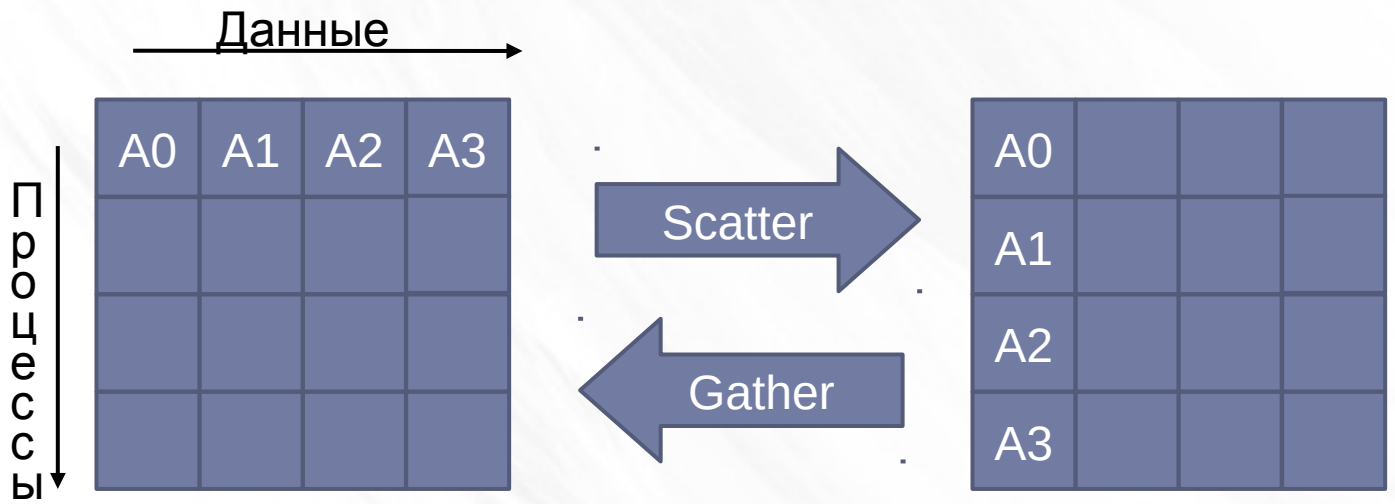
Особенность реализации

Особенностью этого алгоритма является то, что как при прямом, так и при обратном ходе, компьютеры, завершившие свою часть работы, переходят в состояние ожидания, пока не завершат эту работу другие компьютеры. Таким образом, вычислительная нагрузка распределяется по компьютерам неравномерно, не смотря на то, что *данные* изначально распределяются по компьютерам приблизительно одинаково.



Реализация

- Реализация заключается в равномерном распределении данных по всем компьютерам.
- Распределение происходит с помощью функции Scatter и после вычисления на каждом из компьютеров собираем информацию с помощью Gather



Результаты

Размер матрицы A*A	Время работы при последовательном алгоритме (секунд)	Время работа на 10 машинах (секунд)
500	0.5	1
1000	2	2
2000	14	7

Спасибо за внимание.

Коллективные взаимодействия

Коллективная связь осуществляет взаимодействия типа «один-ко-всем», «все-к-одному», «все-со-всеми». MPI имеет следующие функции коллективной связи:

1. Синхронизация (barrier) – синхронизирует все процессы группы

2. Глобальные функции связи

broadcast – передача данных от одного процесса группы к остальным (в т.ч. самому себе)

gather – сбор данных от всех процессов группы к одному процессу

scatter – разброс данных от одного процесса группы ко всем остальным (в т.ч. самому себе)