

Лекция 7

Параллельное решение СЛАУ методом Гаусса

Курносов Михаил Георгиевич

E-mail: mkurnosov@gmail.com

WWW: www.mkurnosov.net

Курс «Параллельные вычислительные технологии»

Сибирский государственный университет телекоммуникаций и информатики (г. Новосибирск)

Осенний семестр, 2017

Система линейных алгебраических уравнений (СЛАУ)

- Дана система линейных алгебраических уравнений

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

$$Ax = b$$

- Требуется найти решение – неизвестные x_1, x_2, \dots, x_n

Решение СЛАУ методом Гаусса

- **Метод Гаусса** (Gaussian elimination, row reduction) – метод последовательного исключения переменных
- **Шаги метода Гаусса:**
 1. **Прямой ход** (elimination) – СЛАУ приводится к треугольной форме путем элементарных преобразований (вычислительная сложность $O(n^3)$)
 2. **Обратный ход** (back substitution) – начиная с последних, находятся все неизвестные системы (вычислительная сложность $O(n^2)$)

Решение СЛАУ методом Гаусса

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_1 - x_2 + 2x_3 = 5 \\ 2x_1 - x_2 - x_3 = -3 \end{cases}$$

■ Прямой ход метода Гаусса

$$\begin{aligned} x_1 + x_2 + x_3 &= 6 \\ x_1 - x_2 + 2x_3 &= 5 \\ 2x_1 - x_2 - x_3 &= -3 \end{aligned}$$

x_1

→

$$\begin{aligned} x_1 + x_2 + x_3 &= 6 \\ -2x_2 + x_3 &= -1 \\ -3x_2 - 3x_3 &= -15 \end{aligned}$$

- умножили первое уравнение на 1 и вычли из второго

- умножили первое уравнение на 1 и вычли из второго

$$\begin{aligned} x_1 + x_2 + x_3 &= 6 \\ -2x_2 + x_3 &= -1 \\ -3x_2 - 3x_3 &= -15 \end{aligned}$$

x_2

→

$$\begin{aligned} x_1 + x_2 + x_3 &= 6 \\ x_2 - \frac{1}{2}x_3 &= \frac{1}{2} \\ -\frac{9}{2}x_3 &= -\frac{27}{2} \end{aligned}$$

- разделили на -2

- умножили второе уравнение на -3 и вычли из третьего

Решение СЛАУ методом Гаусса

$$\begin{cases} x_1 + x_2 + x_3 = 6 \\ x_1 - x_2 + 2x_3 = 5 \\ 2x_1 - x_2 - x_3 = -3 \end{cases}$$

- Обратный ход метода Гаусса

$$\begin{array}{lcl} \begin{array}{l} x_1 + x_2 + x_3 = 6 \\ x_2 - \frac{1}{2}x_3 = \frac{1}{2} \\ x_3 = 3 \end{array} & \rightarrow & \begin{array}{l} x_1 + x_2 + x_3 = 6 \\ x_2 = 2 \\ x_3 = 3 \end{array} & \rightarrow & \begin{array}{l} x_1 = 1 \\ x_2 = 2 \\ x_3 = 3 \end{array} \end{array}$$

Последовательная реализация метода Гаусса

```
int main(int argc, char *argv[])
{
    int n = 3000;
    double t = wtime();
    double *a = malloc(sizeof(*a) * n * n); // Матрица коэффициентов
    double *b = malloc(sizeof(*b) * n);     // Столбец свободных членов
    double *x = malloc(sizeof(*x) * n);     // Известные

    for (int i = 0; i < n; i++) {           // Инициализация
        srand(i * (n + 1));
        for (int j = 0; j < n; j++)
            a[i * n + j] = rand() % 100 + 1;
        b[i] = rand() % 100 + 1;
    }

    #if 0
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++)
            printf("%12.4f ", a[i * n + j]);
        printf(" | %12.4f\n", b[i]);
    }
    #endif
}
```

Последовательная реализация метода Гаусса

```
// Прямой ход -- O(n^3)
for (int k = 0; k < n - 1; k++) {
    // Исключение x_i из строк k+1...n-1
    double pivot = a[k * n + k];
    for (int i = k + 1; i < n; i++) {
        // Из уравнения (строки) i вычитается уравнение k
        double lik = a[i * n + k] / pivot;
        for (int j = k; j < n; j++)
            a[i * n + j] -= lik * a[k * n + j];
        b[i] -= lik * b[k];
    }
}

// Обратный ход -- O(n^2)
for (int k = n - 1; k >= 0; k--) {
    x[k] = b[k];
    for (int i = k + 1; i < n; i++)
        x[k] -= a[k * n + i] * x[i];
    x[k] /= a[k * n + k];
}
```

$$\begin{aligned}a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 &= b_1 \\a'_{22}x_2 + \dots + a'_{25}x_5 &= b'_2 \\a'_{32}x_2 + \dots + a'_{35}x_5 &= b'_3 \\a'_{42}x_2 + \dots + a'_{45}x_5 &= b'_4 \\a'_{52}x_2 + \dots + a'_{55}x_5 &= b'_5\end{aligned}$$



$$\begin{aligned}x_1 &= \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}} \\x_2 &= \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}} \\x_3 &= \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}} \\x_4 &= \frac{b'_4 - a'_{45}x_5}{a'_{44}} \\x_5 &= \frac{b'_5}{a'_{55}}\end{aligned}$$

Последовательная реализация метода Гаусса

```
// ОТЛАДКА: Сравнение результатов с GNU Scientific Library (GSL) -- #include <gsl/gsl_linalg.h>
for (int i = 0; i < n; i++) {
    srand(i * (n + 1));
    for (int j = 0; j < n; j++)
        a[i * n + j] = rand() % 100 + 1;
    b[i] = rand() % 100 + 1;
}

gsl_matrix_view gsl_a = gsl_matrix_view_array(a, n, n);
gsl_vector_view gsl_b = gsl_vector_view_array(b, n);
gsl_vector *gsl_x = gsl_vector_alloc(n);
int s;
gsl_permutation *p = gsl_permutation_alloc(n);
gsl_linalg_LU_decomp(&gsl_a.matrix, p, &s);
gsl_linalg_LU_solve(&gsl_a.matrix, p, &gsl_b.vector, gsl_x);

printf ("GSL X[%d]: ", n);
for (int i = 0; i < n; i++)
    printf("%f ", gsl_vector_get(gsl_x, i));
printf("\n");
```

```
$ gcc ... -lgsl -lgslcblas -lm
```


Последовательная реализация метода Гаусса

```
// Сравнение векторов
for (int i = 0; i < n; i++) {
    if (fabs(x[i] - gsl_vector_get(gsl_x, i)) > 0.0001) {
        fprintf(stderr, "Invalid result: elem %d: %f %f\n", i, x[i], gsl_vector_get(gsl_x, i));
        break;
    }
}
gsl_permutation_free(p);
gsl_vector_free(gsl_x);
#endif

free(b);
free(a);
t = wtime() - t;
printf("Gaussian Elimination (serial): n %d, time (sec) %.6f\n", n, t);
#if 0
printf("X[%d]:      ", n);
for (int i = 0; i < n; i++)
    printf("%f ", x[i]);
printf("\n");
#endif
free(x);
return 0;
}
```

Параллельный метод Гаусса

Версия 1

- Каждый процесс хранит в своей памяти одну строку матрицы – одно уравнение
- Требуется n процессов

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 1}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 2}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 3}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 4}$$

Прямой ход

- Процесс 0 передает свою строку 1 всем
- Процессы 1.. P -1 исключают x_1 из своих уравнений
- Процесс 1 передает свою строку 2 всем
- Процессы 2.. P -1 исключают x_2 из своих уравнений
- ...
- Процесс P -2 передает свою строку n -1 всем
- Процесс P -1 исключают x_{n-1} из своего уравнения

Обратный ход

- Процесс P -1 вычисляет x_n и передает всем
- Процесс P -2 вычисляет x_{n-1} и передает всем
- ...
- Процесс 1 вычисляет x_2 и передает всем
- Процесс 1 вычисляет x_1 и передает всем

Параллельный метод Гаусса

Версия 1

- Каждый процесс хранит в своей памяти одну строку матрицы – одно уравнение
- Требуется n процессов

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 1}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 2}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 3}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 5}$$

Процессы неравномерно загружены
вычислениями – после передачи строки
они выбывают из вычислений

Прямой ход

- Процесс 0 передает свою строку 1 всем
- Процессы 1.. P -1 исключают x_1 из своих уравнений
- Процесс 1 передает свою строку 2 всем
- Процессы 2.. P -1 исключают x_2 из своих уравнений
- ...
- Процесс P -2 передает свою строку n -1 всем
- Процесс P -1 исключают x_{n-1} из своего уравнения

Обратный ход

- Процесс P -1 вычисляет x_n и передает всем
- Процесс P -2 вычисляет x_{n-1} и передает всем
- ...
- Процесс 1 вычисляет x_2 и передает всем
- Процесс 1 вычисляет x_1 и передает всем

Параллельный метод Гаусса

Версия 2

- Каждый процесс хранит в своей памяти горизонтальную полосу из n / P смежных строк
- Требуется P процессов

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 0}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 1}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 1}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 2}$$

**Схема прямого и обратного ходов
такая же, как в версии 1**

**Процессы неравномерно загружены
вычислениями – после передачи строки
они выбывают из вычислений**

Параллельный метод Гаусса

Версия 3 – циклическое распределение строк матрицы

- Каждый процесс хранит в своей памяти порядка n / P строк, векторы $b[n]$ и $x[n]$
- Строки распределены по циклической схеме для выравнивания вычислительной загрузки процессов

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 1}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 2}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 0}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 1}$$

Параллельный метод Гаусса

Устранение x_1 (процесс, содержащий строку 1 рассылает её)

$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1$	Процесс 0 рассылает строку 1
$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2$	Процесс 1: вычитает принятую строку 1 из своей строки
$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3$	Процесс 2: вычитает принятую строку 1 из своей строки
$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4$	Процесс 0: вычитает принятую строку 1 из своей строки
$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5$	Процесс 1: вычитает принятую строку 1 из своей строки



$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1$	Процесс 0 рассылает строку 1
$a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2$	Процесс 1: вычитает принятую строку 1 из своей строки
$a'_{32}x_2 + \dots + a'_{35}x_5 = b'_3$	Процесс 2: вычитает принятую строку 1 из своей строки
$a'_{42}x_2 + \dots + a'_{45}x_5 = b'_4$	Процесс 0: вычитает принятую строку 1 из своей строки
$a'_{52}x_2 + \dots + a'_{55}x_5 = b'_5$	Процесс 1: вычитает принятую строку 1 из своей строки

Параллельный метод Гаусса

Устранение x_2 (процесс, содержащий строку 2 рассылает её)

$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1$ Процесс 0: строка не изменяется

$a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2$ **Процесс 1: рассылает строку 2**

$a'_{32}x_2 + \dots + a'_{35}x_5 = b'_3$ Процесс 2: вычитает принятую строку 2 из своей строки

$a'_{42}x_2 + \dots + a'_{45}x_5 = b'_4$ Процесс 0: вычитает принятую строку 2 из своей строки

$a'_{52}x_2 + \dots + a'_{55}x_5 = b'_5$ Процесс 1: вычитает принятую строку 2 из своей строки



$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1$ Процесс 0: строка не изменяется

$a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2$ **Процесс 1: рассылает строку 2**

$a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3$ Процесс 2: вычитает принятую строку 2 из своей строки

$a'_{43}x_3 + \dots + a'_{45}x_5 = b'_4$ Процесс 0: вычитает принятую строку 2 из своей строки

$a'_{53}x_3 + \dots + a'_{55}x_5 = b'_5$ Процесс 1: вычитает принятую строку 2 из своей строки

Параллельный метод Гаусса

Устранение x_3 (процесс, содержащий строку 3 рассылает её)

$$\begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 & \text{Процесс 0: строка не изменяется} \\ a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2 & \text{Процесс 1: строка не изменяется} \\ a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3 & \text{Процесс 2: рассылает строку 3} \\ a'_{43}x_3 + \dots + a'_{45}x_5 = b'_4 & \text{Процесс 0: вычитает принятую строку 3 из своей строки} \\ a'_{53}x_3 + \dots + a'_{55}x_5 = b'_5 & \text{Процесс 1: вычитает принятую строку 3 из своей строки} \end{array}$$



$$\begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 & \text{Процесс 0: строка не изменяется} \\ a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2 & \text{Процесс 1: строка не изменяется} \\ a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3 & \text{Процесс 2: рассылает строку 3} \\ a'_{44}x_4 + a'_{45}x_5 = b'_4 & \text{Процесс 0: вычитает принятую строку 3 из своей строки} \\ a'_{54}x_4 + a'_{55}x_5 = b'_5 & \text{Процесс 1: вычитает принятую строку 3 из своей строки} \end{array}$$

Параллельный метод Гаусса

Устранение x_4 (процесс, содержащий строку 4 рассылает её)

$$\begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 & \text{Процесс 0: строка не изменяется} \\ a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2 & \text{Процесс 1: строка не изменяется} \\ a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3 & \text{Процесс 2: строка не изменяется} \\ a'_{44}x_4 + a'_{45}x_5 = b'_4 & \text{Процесс 0: рассылает строку 4} \\ a'_{54}x_4 + a'_{55}x_5 = b'_5 & \text{Процесс 1: вычитает принятую строку 4 из своей строки} \end{array}$$



$$\begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 & \text{Процесс 0: строка не изменяется} \\ a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2 & \text{Процесс 1: строка не изменяется} \\ a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3 & \text{Процесс 2: строка не изменяется} \\ a'_{44}x_4 + a'_{45}x_5 = b'_4 & \text{Процесс 0: рассылает строку 4} \\ a'_{55}x_5 = b'_5 & \text{Процесс 1: вычитает принятую строку 4 из своей строки} \end{array}$$

Параллельный метод Гаусса

Обратный ход – инициализация x_i

Каждый процесс инициализирует $x[i]$ значениями $b[i]$, хранящимися в его памяти

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0: } x_1 = b'_1, \quad x_{2-5} = 0$$

$$a'_{22}x_2 + \dots + a'_{25}x_5 = b'_2 \quad \text{Процесс 1: } x_2 = b'_2, \quad x_{1,3-5} = 0$$

$$a'_{33}x_3 + \dots + a'_{35}x_5 = b'_3 \quad \text{Процесс 2: } x_3 = b'_3, \quad x_{1-2,3-5} = 0$$

$$a'_{44}x_4 + a'_{45}x_5 = b'_4 \quad \text{Процесс 0: } x_4 = b'_4, \quad x_{1-3,5} = 0$$

$$a'_{55}x_5 = b'_5 \quad \text{Процесс 1: } x_5 = b'_5, \quad x_{1-4} = 0$$

Параллельный метод Гаусса

Обратный ход – инициализация x_i

Каждый процесс инициализирует $x[i]$ значениями $b[i]$, хранящимися в его памяти

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = b'_1$, $x_{2-5} = 0$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: $x_2 = b'_2$, $x_{1,3-5} = 0$

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: $x_3 = b'_3$, $x_{1-2,3-5} = 0$

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: $x_4 = b'_4$, $x_{1-3,5} = 0$

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: $x_5 = b'_5$, $x_{1-4} = 0$

Параллельный метод Гаусса

Обратный ход – вычисление x_5

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = b'_1 - a'_{15}x_5$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: $x_2 = b'_2 - a'_{25}x_5$

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: $x_3 = b'_3 - a'_{35}x_5$

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: $x_4 = b'_4 - a'_{45}x_5$

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: рассылает x_5

Параллельный метод Гаусса

Обратный ход – вычисление x_4

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = b'_1 - a'_{15}x_5 - a'_{14}x_4$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: $x_2 = b'_2 - a'_{25}x_5 - a'_{24}x_4$

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: $x_3 = b'_3 - a'_{35}x_5 - a'_{34}x_4$

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: рассылает x_4

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: значение x_5 найдено

Параллельный метод Гаусса

Обратный ход – вычисление x_3

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = b'_1 - a'_{15}x_5 - a'_{14}x_4 - a'_{13}x_3$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: $x_2 = b'_2 - a'_{25}x_5 - a'_{24}x_4 - a'_{23}x_3$

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: рассылает x_3

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: значение x_4 найдено

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: значение x_5 найдено

Параллельный метод Гаусса

Обратный ход – вычисление x_2

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = b'_1 - a'_{15}x_5 - a'_{14}x_4 - a'_{13}x_3 - a'_{12}x_2$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: рассылает x_2

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: значение x_3 найдено

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: значение x_4 найдено

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: значение x_5 найдено

Параллельный метод Гаусса

Обратный ход – вычисление x_1

$$x_1 = \frac{b'_1 - a'_{12}x_2 - a'_{13}x_3 - a'_{14}x_4 - a'_{15}x_5}{a'_{11}}$$

Процесс 0: $x_1 = (b'_1 - a'_{15}x_5 - \dots - a'_{12}x_2)/a'_{11}$

$$x_2 = \frac{b'_2 - a'_{23}x_3 - a'_{24}x_4 - a'_{25}x_5}{a'_{22}}$$

Процесс 1: значение x_2 найдено

$$x_3 = \frac{b'_3 - a'_{34}x_4 - a'_{35}x_5}{a'_{33}}$$

Процесс 2: значение x_3 найдено

$$x_4 = \frac{b'_4 - a'_{45}x_5}{a'_{44}}$$

Процесс 0: значение x_4 найдено

$$x_5 = \frac{b'_5}{a'_{55}}$$

Процесс 1: значение x_5 найдено

Параллельный метод Гаусса

```
int main(int argc, char *argv[])
{
    int n = 3000;
    int rank, commsize;
    MPI_Init(&argc, &argv);
    double t = MPI_Wtime();
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &commsize);

    int nrows = get_chunk(n, commsize, rank);
    int *rows = malloc(sizeof(*rows) * nrows);    // Номера локальных строк

    // Матрица дополнена столбцом для вектора b
    double *a = malloc(sizeof(*a) * nrows * (n + 1));
    double *x = malloc(sizeof(*x) * n);
    double *tmp = malloc(sizeof(*tmp) * (n + 1));
```

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 1}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 2}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 0}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 1}$$

Процесс 0: rows[2] = {0, 3}

Процесс 1: rows[2] = {1, 4}

Процесс 2: rows[1] = {2}

Параллельный метод Гаусса

```
int get_chunk(int total, int commsize, int rank)
{
    int n = total;
    int q = n / commsize;
    if (n % commsize)
        q++;
    int r = commsize * q - n;

    /* Compute chunk size for the process */
    int chunk = q;
    if (rank >= commsize - r)
        chunk = q - 1;
    return chunk;
}
```

$$a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 = b_1 \quad \text{Процесс 0}$$

$$a_{21}x_1 + a_{22}x_2 + \dots + a_{25}x_5 = b_2 \quad \text{Процесс 1}$$

$$a_{31}x_1 + a_{32}x_2 + \dots + a_{35}x_5 = b_3 \quad \text{Процесс 2}$$

$$a_{41}x_1 + a_{42}x_2 + \dots + a_{45}x_5 = b_4 \quad \text{Процесс 0}$$

$$a_{51}x_1 + a_{52}x_2 + \dots + a_{55}x_5 = b_5 \quad \text{Процесс 1}$$

get_chunk(5, 3, rank)

Процесс 0: n rows = 2

Процесс 1: n rows = 2

Процесс 2: n rows = 1

Параллельный метод Гаусса

```
// Инициализация как в последовательной версии
```

```
for (int i = 0; i < nrows; i++) {  
    rows[i] = rank + commsize * i;  
    srand(rows[i] * (n + 1));  
    for (int j = 0; j < n; j++)  
        a[i * (n + 1) + j] = rand() % 100 + 1;  
    // b[i]  
    a[i * (n + 1) + n] = rand() % 100 + 1;  
}
```

```
#if 0
```

```
    MPI_Recv(NULL, 0, MPI_INT, (rank > 0) ? rank - 1 : MPI_PROC_NULL, 0, MPI_COMM_WORLD,  
             MPI_STATUS_IGNORE); // Вывод в порядке: proc 0, 1, 2, ... P-1.  
    printf("Proc %d: ", rank);  
    for (int i = 0; i < nrows; i++)  
        printf("%d ", rows[i]);  
    printf("\n");  
    MPI_Ssend(NULL, 0, MPI_INT, rank != commsize - 1 ? rank + 1 : MPI_PROC_NULL, 0,  
             MPI_COMM_WORLD);
```

```
#endif
```

Параллельный метод Гаусса

```
// Прямой ход
int row = 0;
for (int i = 0; i < n - 1; i++) {
    // Исключаем x_i
    if (i == rows[row]) {
        // Рассылаем строку i, находящуюся в памяти текущего процесса
        MPI_Bcast(&a[row * (n + 1)], n + 1, MPI_DOUBLE, rank, MPI_COMM_WORLD);
        for (int j = 0; j <= n; j++)
            tmp[j] = a[row * (n + 1) + j];
        row++;
    } else {
        MPI_Bcast(tmp, n + 1, MPI_DOUBLE, i % commsize, MPI_COMM_WORLD);
    }

    // Вычитаем принятую строку из уравнений, хранящихся в текущем процессе
    for (int j = row; j < nrow; j++) {
        double scaling = a[j * (n + 1) + i] / tmp[i];
        for (int k = i; k < n + 1; k++)
            a[j * (n + 1) + k] -= scaling * tmp[k];
    }
}
```

Параллельный метод Гаусса

```
// Инициализация неизвестных
row = 0;
for (int i = 0; i < n; i++) {
    x[i] = 0;
    if (i == rows[row]) {
        x[i] = a[row * (n + 1) + n];
        row++;
    }
}
```

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{15}x_5 &= b_1 & \text{Процесс 0: } x_1 = b'_1, x_{2-5} &= 0 \\ a'_{22}x_2 + \dots + a'_{25}x_5 &= b'_2 & \text{Процесс 1: } x_2 = b'_2, x_{1,3-5} &= 0 \\ a'_{33}x_3 + \dots + a'_{35}x_5 &= b'_3 & \text{Процесс 2: } x_3 = b'_3, x_{1-2,3-5} &= 0 \\ a'_{44}x_4 + a'_{45}x_5 &= b'_4 & \text{Процесс 0: } x_4 = b'_4, x_{1-3,5} &= 0 \\ a'_{55}x_5 &= b'_5 & \text{Процесс 1: } x_5 = b'_5, x_{1-4} &= 0 \end{aligned}$$

Параллельный метод Гаусса

```
/* Обратный ход */
row = nrow - 1;
for (int i = n - 1; i > 0; i--) {
    if (row >= 0) {
        if (i == rows[row]) {
            x[i] /= a[row * (n + 1) + i];           // Передаем найденное x_i
            MPI_Bcast(&x[i], 1, MPI_DOUBLE, rank, MPI_COMM_WORLD);
            row--;
        } else
            MPI_Bcast(&x[i], 1, MPI_DOUBLE, i % commsize, MPI_COMM_WORLD);
    } else
        MPI_Bcast(&x[i], 1, MPI_DOUBLE, i % commsize, MPI_COMM_WORLD);

    for (int j = 0; j <= row; j++)                   // Корректировка локальных x_i
        x[rows[j]] -= a[j * (n + 1) + i] * x[i];
}

if (rank == 0)
    x[0] /= a[row * (n + 1)];                         // Корректировка x_0
MPI_Bcast(x, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
// Все процессы содержат корректный вектор x[n] решений
```

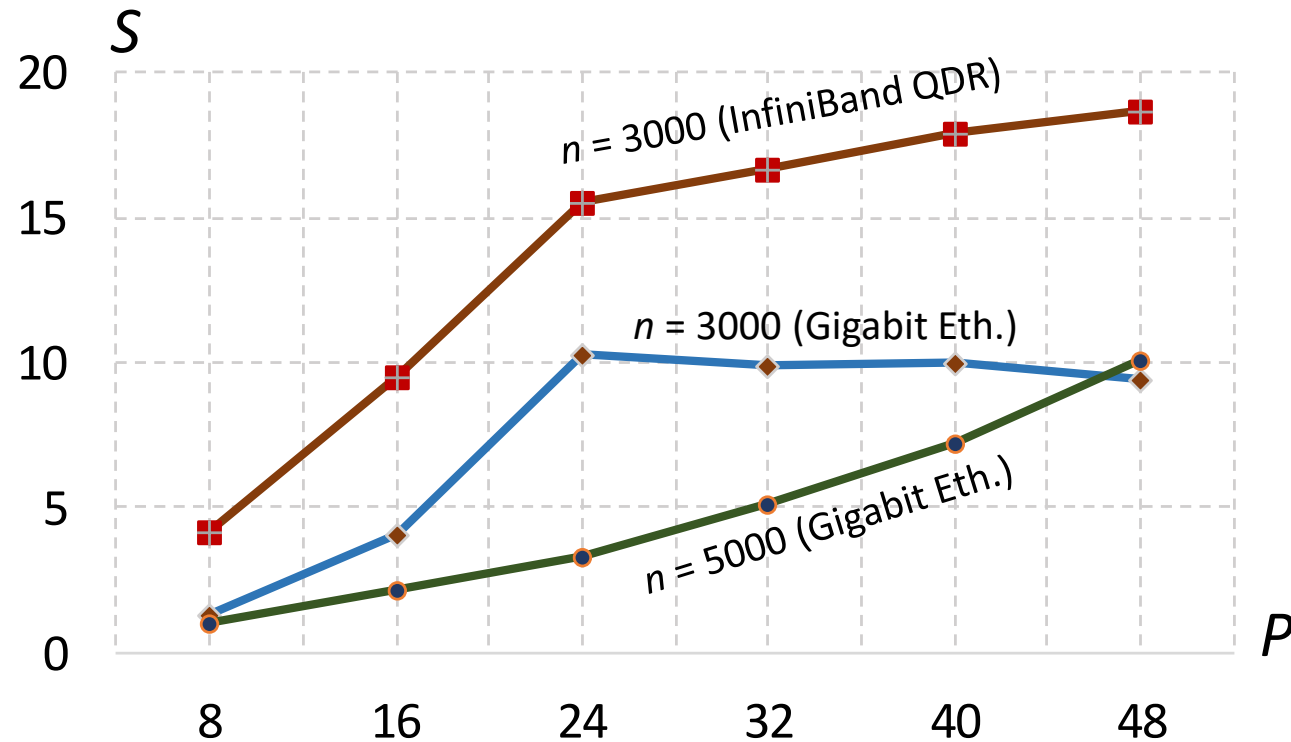
Параллельный метод Гаусса

```
free(tmp);
free(rows);
free(a);
t = MPI_Wtime() - t;

if (rank == 0) {
    printf("Gaussian Elimination (MPI): n %d, procs %d, time (sec) %.6f\n",
        n, commsize, t);
    #if 0
    printf("MPI X[%d]: ", n);
    for (int i = 0; i < n; i++)
        printf("%f ", x[i]);
    printf("\n");
    #endif
}

free(x);
MPI_Finalize();
return 0;
}
```

Анализ эффективности



Зависимость коэффициента S ускорения от числа P параллельных процессов:
кластеры Jet (Gigabit Ethernet) и Oak (InfiniBand QDR)