Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Образовательный комплекс

Введение в методы параллельного программирования

Лекция 11.

Параллельные методы решения систем линейных уравнений

Гергель В.П., профессор, д.т.н.
Кафедра математического обеспечения ЭВМ

Содержание

Постановка задачи

Линейное уравнение с *n* неизвестными

$$a_0 x_0 + a_1 x_1 + \ldots + a_{n-1} x_{n-1} = b$$

Множество *n* линейных уравнений называется *системой линейных уравнений* или *линейной системой*

$$\begin{aligned}
a_{0,0} x_0 \quad & + a_{0,1} x_1 \quad && + \ldots + a_{0,n-1} x_{n-1} \quad && = b_0 \\
a_{1,0} x_0 \quad & + a_{1,1} x_1 \quad && + \ldots + a_{1,n-1} x_{n-1} \quad && = b_1 \\
& \ldots \\
a_{n-1,0} x_0 \quad & + a_{n-1,1} x_1 \quad && + \ldots + a_{n-1,n-1} x_{n-1} \quad && = b_{n-1}
\end{aligned}$$

В матричной форме:

$$Ax = b$$

Постановка задачи

- Под *задачей решения системы линейных уравнений* для заданных матрицы *A* и вектора *b* понимается нахождение значения вектора неизвестных *x*, при котором выполняются все уравнения системы.

Метод Гаусса – последовательный алгоритм…

- *Основная идея метода* - приведение матрицы *A* посредством эквивалентных преобразований к треугольному виду, после чего значения искомых неизвестных могут быть получены непосредственно в явном виде
- *Эквивалентные преобразования*:
  - Умножение любого из уравнений на ненулевую константу,
  - Перестановка уравнений,
  - Прибавление к уравнению любого другого уравнения системы.

Метод Гаусса – последовательный алгоритм…

На первом этапе - *прямой ход* метода Гаусса - исходная система линейный уравнений при помощи последовательного исключения неизвестных приводится к верхнему треугольному виду

$$Ux = c, \qquad U = \begin{pmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,n-1} \\ 0 & u_{1,1} & \dots & u_{1,n-1} \\ & & \dots & \\ 0 & 0 & \dots & u_{n-1,n-1} \end{pmatrix}$$

На *обратном ходе* метода Гаусса (второй этап алгоритма) осуществляется определение значений неизвестных

Метод Гаусса – последовательный алгоритм…

Прямой ход

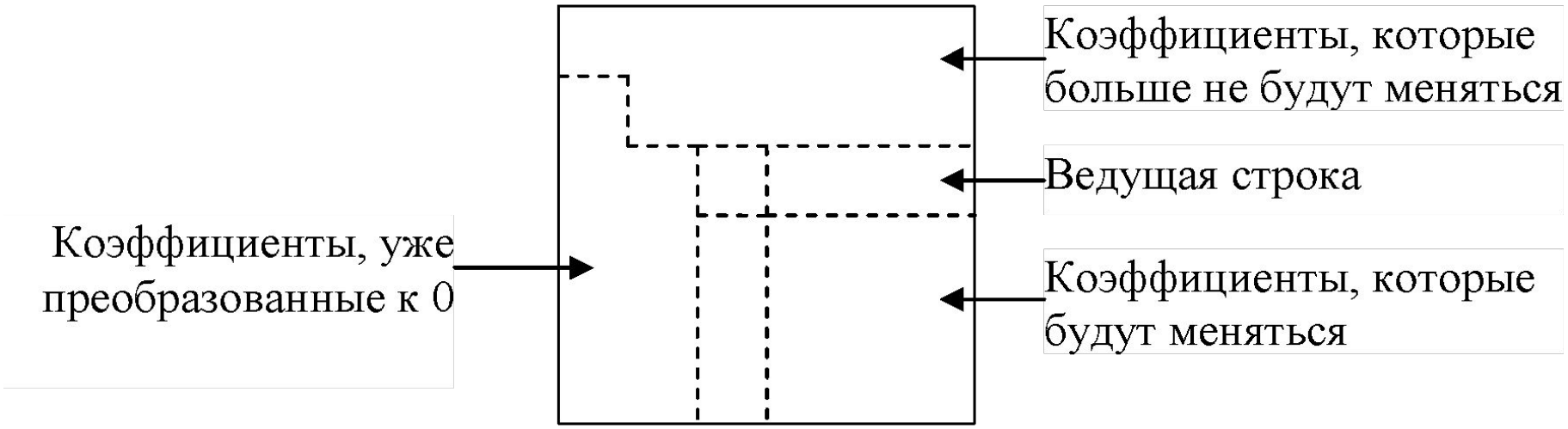
На итерации i, $0 \le i < n-1$, метода производится исключение неизвестной i для всех уравнений с номерами k, больших i (т.е. $i < k \le n-1$,). Для этого из этих уравнений осуществляется вычитание строки i, умноженной на константу (a_{ki}/a_{ii}) с тем, чтобы результирующий коэффициент при неизвестной x_i в строках оказался нулевым.

Все необходимые вычисления определяются при помощи соотношений:

$$a'_{kj} = a_{kj} - (a_{ki} / a_{ii}) \cdot a_{ij}, \quad i \le j \le n-1, i < k \le n-1, 0 \le i < n-1$$
$$b'_k = b_k - (a_{ki} / a_{ii}) \cdot b_i,$$

Метод Гаусса – последовательный алгоритм…

- Общая схема состояния данных на *i*-ой итерации прямого хода алгоритма Гаусса



Метод Гаусса – последовательный алгоритм

Обратный ход

После приведения матрицы коэффициентов к верхнему треугольному виду становится возможным определение значений неизвестных:

- Из последнего уравнения преобразованной системы может быть вычислено значение переменной x_{n-1},
- Из предпоследнего уравнения становится возможным определение переменной x_{n-2} и т.д.

В общем виде, выполняемые вычисления при обратном ходе метода Гаусса могут быть представлены при помощи соотношений:

$$x_{n-1} = b_{n-1} / a_{n-1,n-1},$$

$$x_i = (b_i - \sum_{j=i+1}^{n-1} a_{ij} x_j) / a_{ii}, \quad i = n-2, n-3, \ldots, 0$$

Метод Гаусса – параллельный алгоритм…

- **Определение подзадач**
  - Все вычисления сводятся к однотипным вычислительным операциям над строками матрицы коэффициентов системы линейных уравнений,
  - Следовательно, в основу параллельной реализации алгоритма Гаусса может быть положен принцип распараллеливания по данным,
  - В качестве *базовой подзадачи* примем все вычисления, связанные с обработкой одной строки матрицы *A* и соответствующего элемента вектора *b*.

Метод Гаусса – параллельный алгоритм…

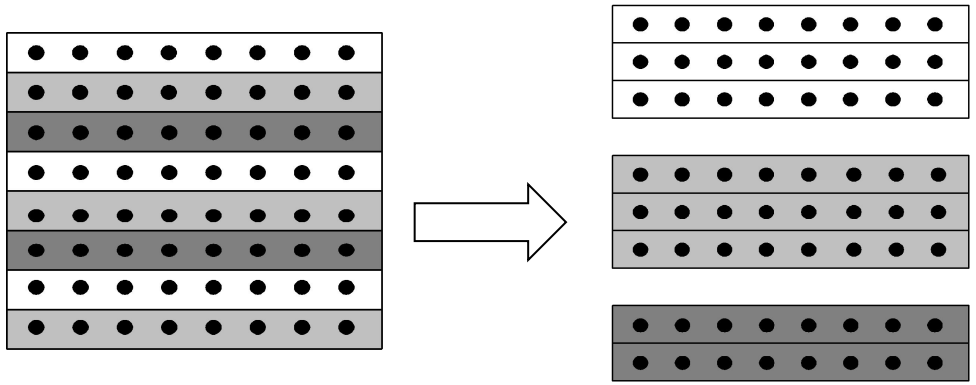
- **Выделение информационных зависимостей…**
  - Каждая итерация *i* выполнения **прямого хода** алгоритма Гаусса включает:
    - **Выбор ведущей строки**, для выполнения которого подзадачи с номерами *k, k>i,* должны обменяться своими элементами при исключаемой переменной x_i для нахождения максимального по абсолютной величине значения. Строка, которой принадлежит выбранное значение, выбирается в качестве *ведущей строки* для выполняемой итерации метода,
    - **Рассылку** выбранной ведущей строки матрицы *A* и соответствующего элемента вектора *b* всем подзадачам с номерами *k, k>i*,
    - **Вычитание** строк для всех подзадачи *k (k>i)*, обеспечивая тем самым исключение соответствующей неизвестной x_i.

Метод Гаусса – параллельный алгоритм…

- **Выделение информационных зависимостей**
  - При выполнении **обратного хода** метода Гаусса подзадачи выполняют необходимые вычисления для нахождения значения неизвестных:
    - Как только какая-либо подзадача i, $0 \le i < n-1$, определяет значение своей переменной x_i, это значение должно быть разослано всем подзадачам с номерами k, $k < i$,
    - Далее подзадачи подставляют полученное значение новой неизвестной в линейное уравнение, представленное строкой матрицы A, расположенной в данной подзадаче, и выполняют корректировку значений для элементов вектора b.

Метод Гаусса – параллельный алгоритм…

- **Масштабирование и распределение подзадач по процессорам…**
  - В случае, когда размер матрицы, описывающей систему линейных уравнений, оказывается большим, чем число доступных процессоров (т.е., $n>p$), базовые подзадачи можно укрупнить, объединив в рамках одной подзадачи несколько строк матрицы.



*Использование **циклического способа** формирования полос позволяет обеспечить лучшую балансировку вычислительной нагрузки между подзадачами*

Метод Гаусса – параллельный алгоритм…

- **Масштабирование и распределение подзадач по процессорам**
  - Основным видом информационного взаимодействия подзадач является операция передачи данных от одного процессора всем процессорам вычислительной системы,
  - Как результат, для эффективной реализации требуемых информационных взаимодействий между базовыми подзадачами, топология сети передачи данных должны иметь структуру *гиперкуба* или *полного графа*.

Метод Гаусса – параллельный алгоритм…

- **Анализ эффективности**
  - Общая оценка показателей ускорения и эффективности

$$S_p = \frac{(2n^3/3 + n^2)}{\frac{1}{p}\sum_{i=2}^{n}(3i + 2i^2)}, \quad E_p = \frac{(2n^3/3 + n^2)}{\sum_{i=2}^{n}(3i + 2i^2)}$$

Балансировка вычислительной нагрузки между процессорами, в целом, является достаточно равномерной

Метод Гаусса – параллельный алгоритм…

❑ Анализ эффективности (уточненные оценки)…

- Время выполнения параллельного алгоритма, связанное непосредственно с вычислениями, состоит из:

• **Времени выполнения прямого хода алгоритма Гаусса** (*n-1* итерация).

▪ выбор максимального значения в столбце с исключаемой неизвестной,

▪ вычитание ведущей строки из каждой строки оставшейся части полосы матрицы *A*

$$T_p^1 = \sum_{i=0}^{n-2}\left[\frac{(n-i)}{p} + \frac{(n-i)}{p}\cdot 2(n-i)\right] = \frac{1}{p}\sum_{i=0}^{n-2}\left[(n-i)\cdot +2(n-i)^2\right]$$

• **Времени выполнения обратного хода алгоритма Гаусса** (*n-1* итерация).

▪ обновление значения правых частей после рассылки вычисленного значения очередной неизвестной

$$T_p^2 = \sum_{i=0}^{n-2} 2\cdot(n-i)/\,p$$

Метод Гаусса – параллельный алгоритм…

- **Анализ эффективности** (уточненные оценки)…

  - Оценка затрат на выполнение операций передачи данных между процессорами:

    - **При выполнении прямого хода алгоритма Гаусса**
      - для определения ведущей строки процессоры обмениваются локально найденными максимальными значениями в столбце с исключаемой переменной (*MPI_Allreduce*)

$$T_p^1(comm) = (n-1)\cdot \log_2 p \cdot (\alpha + w/\beta)$$

      - рассылка выбранной ведущей строки

$$T_p^2(comm) = (n-1)\cdot \log_2 p \cdot (\alpha + wn/\beta)$$

    - **При выполнении обратного хода алгоритма Гаусса**
      - на каждой итерации осуществляется рассылка между всеми процессорами вычисленного значения очередной неизвестной

$$T_p^3(comm) = (n-1)\cdot \log_2 p \cdot (\alpha + w/\beta)$$

Метод Гаусса – параллельный алгоритм…

- **Анализ эффективности** (уточненные оценки)

Общее время выполнения параллельного алгоритма:

$$T_p = \frac{1}{p}\sum_{i=2}^{n}(3i + 2i^2)\,\tau + (n-1)\cdot \log_2 p \cdot (3\alpha + w(n+2)/\beta)$$

Метод Гаусса – параллельный алгоритм…

Программная реализация…

– **Главная функция программы *main***. Реализует логику работы алгоритма, последовательно вызывает необходимые подпрограммы:

- ***ProcessInitialization*** определяет исходные данные решаемой задачи,
- ***GaussianElimination*** выполняет прямой ход метода Гаусса,
- ***BackSubstitution*** реализует обратный ход метода Гаусса,
- ***ResultCollection*** осуществляет сбор со всех процессов отдельных частей вычисленного вектора неизвестных,
- ***ProcessTermination*** выполняет необходимый вывод результатов решения задачи и освобождает всю ранее выделенную память для хранения данных.

Метод Гаусса – параллельный алгоритм…

- **Программная реализация…**
  - **Главная функция программы *main***. Использует массивы:
    - ***pPivotPos*** определяют номера строк матрицы, выбираемых в качестве ведущих, по итерациям прямого хода метода Гаусса – определяет далее порядок выполнения итераций для обратного хода (массив является глобальным и любое его изменение требует выполнения операции рассылки измененных данных),
    - ***pProcPivotIter*** определяют номера итераций прямого хода метода Гаусса, на которых строки процесса использовались в качестве ведущих – нулевое значение элемента означает, что соответствующая строка должна обрабатываться при исключении неизвестных (массив является локальным для каждого процесса).

Программа

Метод Гаусса – параллельный алгоритм…

- **Программная реализация…**
  - Функция ***GaussianElimination*** выполняет параллельный вариант прямого хода алгоритма Гаусса

Программа

  - Функция ***EliminateRows*** проводит вычитание ведущей строки из строк процесса, которые еще не использовались в качестве ведущих (т.е. для которых элементы массива *pProcPivotIter* равны нулю)

Метод Гаусса – параллельный алгоритм…

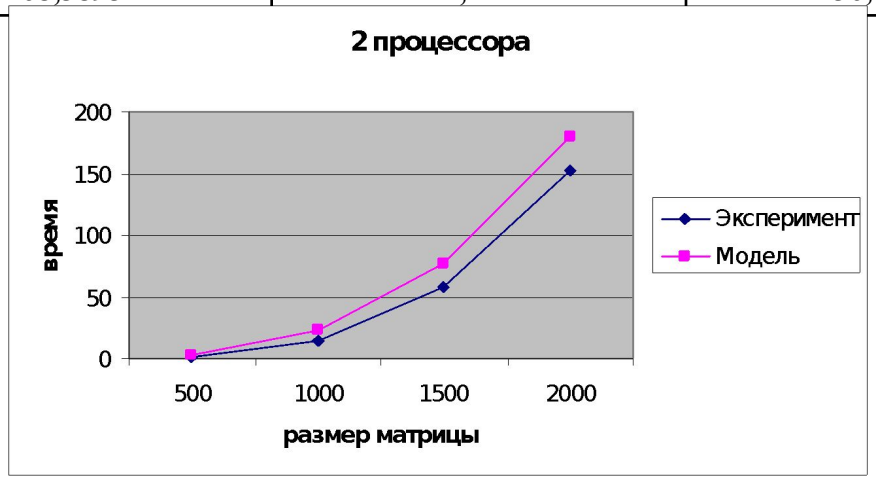
- **Программная реализация…**
  - Функция ***BackSubstitution*** реализует параллельный вариант обратного хода Гаусса.

Программа

Метод Гаусса – параллельный алгоритм…

- **Результаты вычислительных экспериментов**
  - Сравнение теоретических оценок и экспериментальных данных

Размер системы	2 процессора		4 процессора	
	T_p(модель)	T_p^*	T_p(модель)	T_p^*
500	1,2558	1,3444	0,9376	1,0573
1000	8,8879	10,1637	5,2025	6,3733
1500	29,1734	32,8611	15,9326	19,9917
2000	68,3893	77,4121	36,2665	45,9435

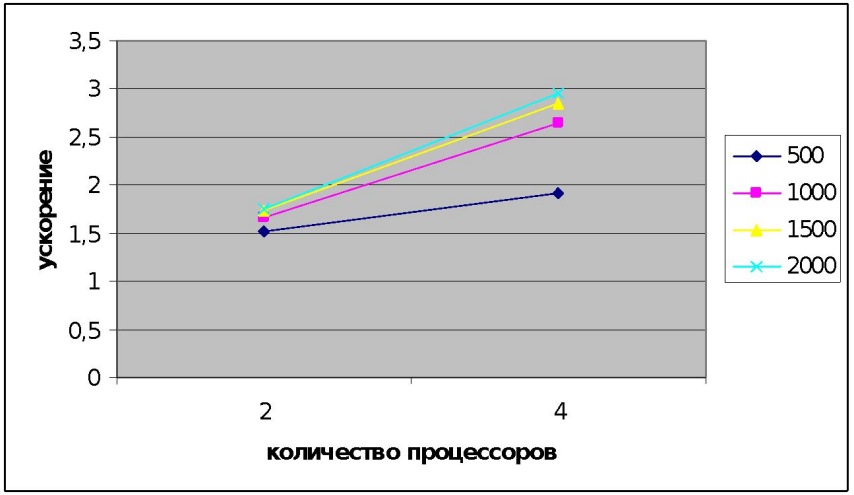


Метод Гаусса – параллельный алгоритм

Результаты вычислительных экспериментов

– Ускорение вычислений

Размер системы	Последовательный алгоритм	Параллельный алгоритм			
		2 процессора		4 процессора	
		Время	Ускорение	Время	Ускорение
500	2,025	1,3444	1,5062	1,0573	1,9152
1000	16,8001	10,1637	1,6529	6,3733	2,6360
1500	56,9628	32,8611	1,7334	19,9917	2,8493
2000	135,547	77,4121	1,7509	45,9435	2,9503



Метод сопряженных градиентов…

- **Итерационные методы решения систем линейных уравнений**
  - к искомому точному решению x^* системы $Ax=b$ строится последовательность приближенных решений $x_0, x_1, \ldots, x_k, \ldots$,
  - каждое очередное приближение дает оценку точного решения со все уменьшающейся погрешностью,
  - оценка точного решения может быть получена с любой требуемой точностью

Метод сопряженных градиентов *- один из наиболее известных итерационных методов решения систем линейных уравнений*

Метод сопряженных градиентов…

- Метод сопряженных градиентов может быть применен для решения системы линейных уравнений с симметричной, положительно определенной матрицей
  - Матрица A является *симметричной*, если она совпадает со своей транспонированной матрицей, т.е. $A=A^T$,
  - Матрица A называется *положительно определенной*, если для любого вектора x справедливо: $x^TAx>0$.
- После выполнения n итераций метода сопряженных градиентов (n есть порядок решаемой системы линейных уравнений), очередное приближение x^n совпадает с точным решением.

Метод сопряженных градиентов – последовательный алгоритм...

- Если матрица *A* симметричная и положительно определенная, то функция

$$q(x) = \frac{1}{2} x^T \cdot A \cdot x - x^T b + c$$

имеет единственный минимум, который достигается в точке *x**, совпадающей с решением системы линейных уравнений.

Метод сопряженных градиентов является одним из широкого класса итерационных алгоритмов, которые позволяют найти решение путем минимизации функции q(x)

Метод сопряженных градиентов – последовательный алгоритм...

- ***Итерация метода*** сопряженных градиентов состоит в вычислении очередного приближения к точному решению

$$x^k = x^{k-1} + s^k d^k$$

где

x^k – очередное приближение,

x^{k-1} – приближение, построенное на предыдущем шаге,

s^k – скалярный шаг,

d^k – вектор направления

Метод сопряженных градиентов – последовательный алгоритм...

Перед выполнением первой итерации вектора x_0 и d_0 полагаются равными нулю, а для вектора g_0 устанавливается значение равное $-b$.

Шаг 1: Вычисление градиента

$$g^k = A \cdot x^{k-1} - b$$

Шаг 2: Вычисление вектора направления

$$d^k = -g^k + \frac{\left((g^k)^T, g^k\right)}{\left((g^{k-1})^T, g^{k-1}\right)} d^{k-1}$$

Шаг 3: Вычисление величины смещения по выбранному направлению

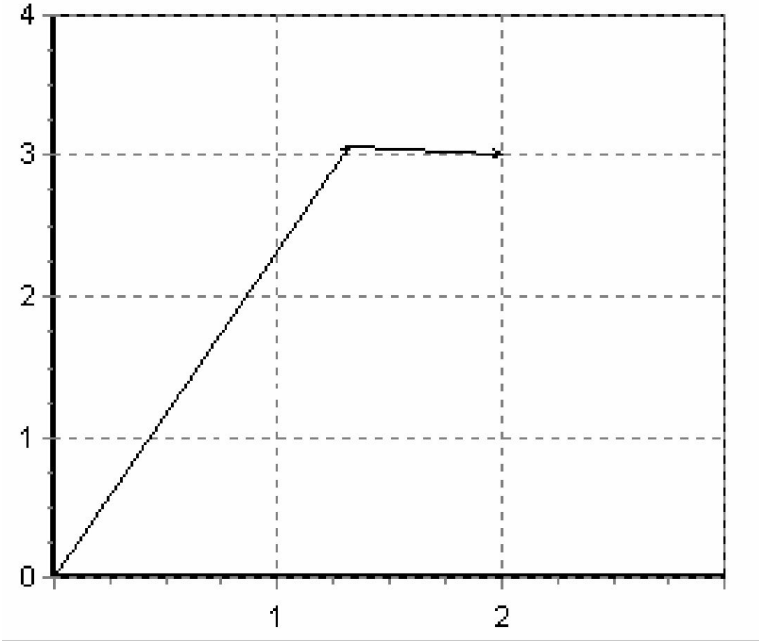
$$s^k = \frac{\left(d^k, g^k\right)}{(d^k)^T \cdot A \cdot d^k}$$

Шаг 4: Вычисление нового приближения

$$x^k = x^{k-1} + s^k d^k$$

Вычислительная сложность алгоритма $T_1 = 2n^3 + 13n^2$

Метод сопряженных градиентов – последовательный алгоритм...



Итерации метода сопряженных градиентов при решении системы линейных уравнений второго порядка:

$$\begin{cases} 3x_0 - x_1 = 3 \\ -x_0 + 3x_1 = 7 \end{cases}$$

Метод сопряженных градиентов – параллельный алгоритм...

Организация параллельных вычислений

- Выполнение итераций метода осуществляется последовательно, следовательно наиболее целесообразный подход состоит в распараллеливании вычислений, реализуемых в ходе выполнения отдельных итераций,
- Основные вычисления, выполняемые в соответствии с методом, состоят в умножении матрицы *A* на вектора *x* и *d,*
- Дополнительные вычисления, имеющие меньший порядок сложности, представляют собой различные операции обработки векторов (скалярное произведение, сложение и вычитание, умножение на скаляр).

При организации параллельных вычислений может быть полностью использован материал, изложенный в разделе "Параллельные методы умножения матрицы на вектор"

Метод сопряженных градиентов – параллельный алгоритм...

Анализ эффективности…

(при использовании параллельного алгоритма умножения матрицы на вектор при ленточном горизонтальном разделении матрицы и при полном дублировании всех обрабатываемых векторов)

– Вычислительная сложность параллельных операций умножения матрицы на вектор при использовании схемы ленточного горизонтального разделения матрицы

$$T_p(calc) = 2n\lceil n/p \rceil \cdot (2n-1)$$

- Как результат, при условии дублирования всех вычислений над векторами общая вычислительная сложность параллельного варианта метода сопряженных градиентов является равной

$$T_p = n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)$$

Метод сопряженных градиентов – параллельный алгоритм...

- **Анализ эффективности…**
  - Общая оценка показателей ускорения и эффективности

$$S_p = \frac{2n^3 + 13n^2}{n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)}, \quad E_p = \frac{2n^3 + 13n^2}{p \cdot n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)}$$

Балансировка вычислительной нагрузки между процессорами является достаточно равномерной

Метод сопряженных градиентов – параллельный алгоритм...

- **Анализ эффективности** (уточненные оценки)
  - Коммуникационная сложность рассматриваемых параллельных вычислений (см. раздел 7)

$$T_p(comm) = 2n(\alpha \cdot \lceil \log p \rceil + w(n/p)(p-1)/\beta)$$

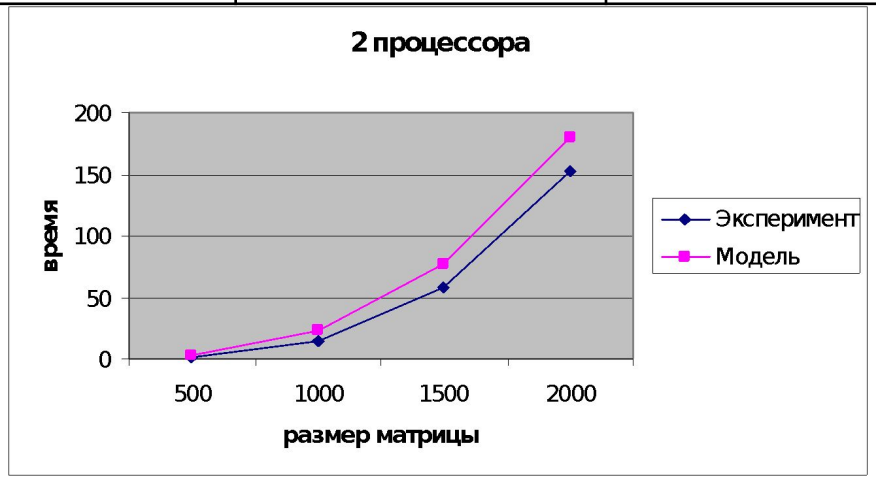
  - Общее время выполнения параллельного варианта метода сопряженных градиентов для решения систем линейных уравнений

$$T_p = n \cdot [(2\lceil n/p \rceil \cdot (2n-1) + 13n) \cdot \tau + 2(\alpha \cdot \lceil \log p \rceil + w(n/p)(p-1)/\beta)]$$

Метод сопряженных градиентов – параллельный алгоритм...

- **Результаты вычислительных экспериментов**
  - Сравнение теоретических оценок и экспериментальных данных

Размер системы	2 процессора		4 процессора	
	T_p(модель)	T_p^*	T_p(модель)	T_p^*
500	3,0633	1,3951	1,7029	1,2303
1000	22,9046	14,2252	11,9231	11,0989
1500	76,1431	57,9130	38,9704	41,9521
2000	179,3979	152,0721	91,1543	110,7042

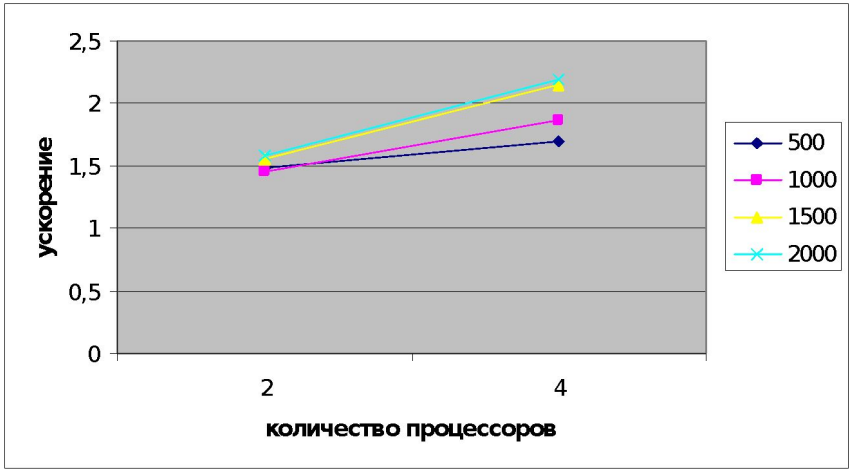


Метод сопряженных градиентов – параллельный алгоритм

Результаты вычислительных экспериментов

– Ускорение вычислений

Размер системы	Последовательный алгоритм	Параллельный алгоритм			
		2 процессора		4 процессора	
		Время	Ускорение	Время	Ускорение
500	2,0766	1,3951	1,4885	1,2303	1,6879
1000	20,6389	14,2252	1,4509	11,0989	1,8595
1500	90,2671	57,9130	1,5587	41,9521	2,1517
2000	241,6062	152,0721	1,5888	110,7042	2,1824

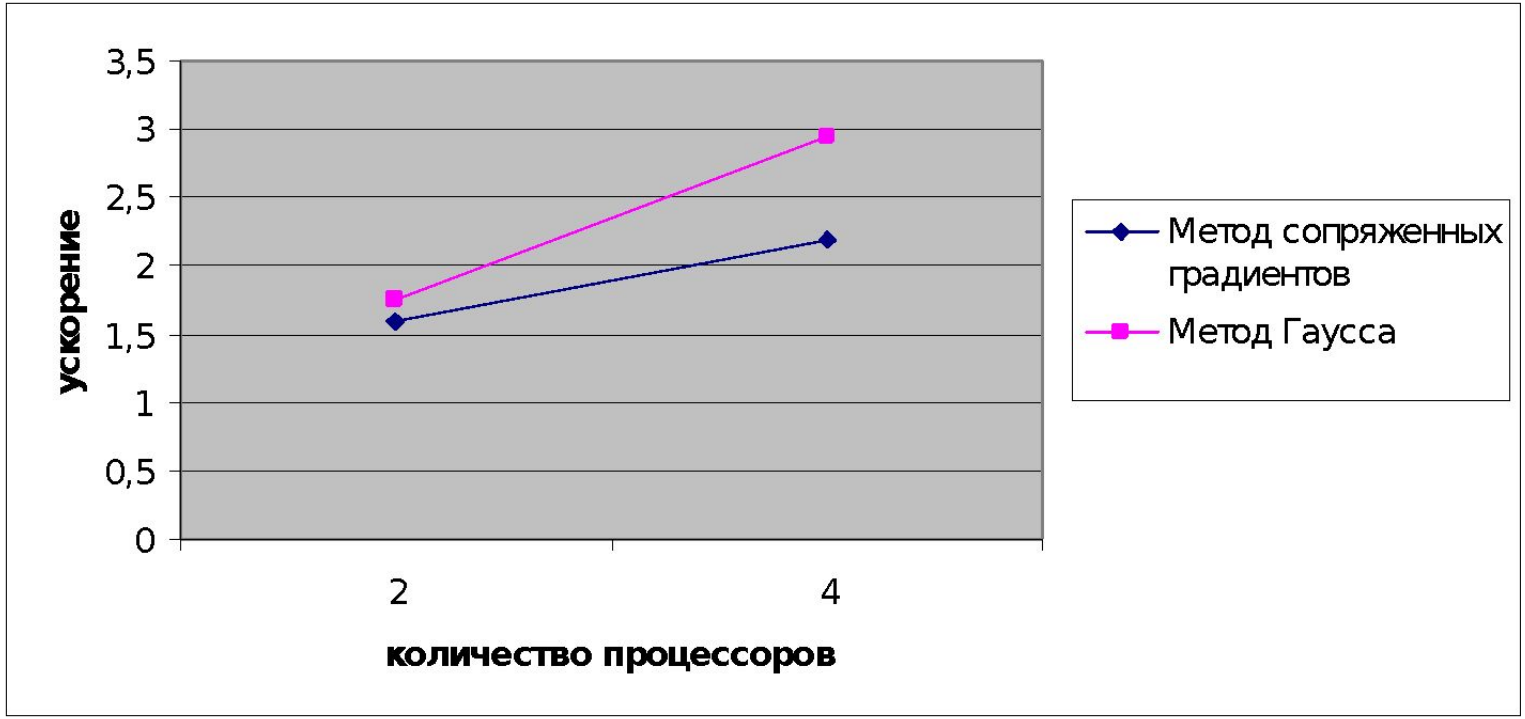


Заключение

- Рассмотрены два параллельных алгоритма решения систем линейных уравнений:
  - Метод Гаусса,
  - Метод сопряженных градиентов
- Представлена программная реализация метода Гаусса
- Теоретические оценки и результаты экспериментов показывают большую эффективность метода сопряженных градиентов

Заключение

- Ускорение параллельных алгоритмов решения системы линейных уравнений с размером матрицы 2000×2000



Вопросы для обсуждения

- Оцените, на каком этапе метода Гаусса (прямой и обратный ход) происходит большее снижение показателей эффективности.
- В чем причина столь низких показателей ускорения и эффективности параллельного алгоритма Гаусса?
- Существуют ли способ улучшения этих показателей?
- Какой из рассмотренных алгоритмов обладает большей вычислительной сложностью?
- В чем состоит основное преимущество итерационных методов решения систем линейных уравнений?

Темы заданий для самостоятельной работы

- Выполните разработку параллельного варианта метода Гаусса при вертикальном разбиении матрицы по столбцам
- Выполните разработку параллельных вариантов методов Якоби и Зейделя решения систем линейных уравнений
- Постройте теоретические оценки времени работы этих алгоритмов с учетом параметров используемой вычислительной системы. Проведите вычислительные эксперименты и сравните полученные результаты с ранее полученными теоретическими оценками

Литература

- **Гергель В.П.** (2007). Теория и практика параллельных вычислений. – М.: Интернет-Университет, БИНОМ. Лаборатория знаний.
- **Бахвалов** Н.С., Жидков Н.П., Кобельков Г.М (1987) Численные методы. – М.: Наука.
- **Самарский** А.А., Гулин А.В. (1989). Численные методы – М.: Наука.
- **Bertsekas**, D.P., Tsitsiklis, J.N. (1989). Parallel and distributed Computation. Numerical Methods. - Prentice Hall, Englewood Cliffs, New Jersey.
- **Kumar** V., Grama, A., Gupta, A., Karypis, G. (1994). Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)
- **Quinn**, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.

Следующая тема

- **Параллельные методы сортировки**

Авторский коллектив

Гергель В.П., профессор, д.т.н., руководитель

Гришагин В.А., доцент, к.ф.м.н.

Сысоев А.В., ассистент (раздел 1)

Лабутин Д.Ю., ассистент (система ПараЛаб)

Абросимова О.Н., ассистент (раздел 10)

Гергель А.В., аспирант (раздел 12)

Лабутина А.А., магистр (разделы 7,8,9, система ПараЛаб)

Сенин А.В. (раздел 11)

О проекте

Целью проекта является создание образовательного комплекса "Многопроцессорные вычислительные системы и параллельное программирование", обеспечивающий рассмотрение вопросов параллельных вычислений, предусматриваемых рекомендациями Computing Curricula 2001 Международных организаций IEEE-CS и ACM. Данный образовательный комплекс может быть использован для обучения на начальном этапе подготовки специалистов в области информатики, вычислительной техники и информационных технологий.

Образовательный комплекс включает **учебный курс "Введение в методы параллельного программирования" и лабораторный практикум "Методы и технологии разработки параллельных программ"**, что позволяет органично сочетать фундаментальное образование в области программирования и практическое обучение методам разработки масштабного программного обеспечения для решения сложных вычислительно-трудоемких задач на высокопроизводительных вычислительных системах.

Проект выполнялся в Нижегородском государственном университете им. Н.И. Лобачевского на кафедре математического обеспечения ЭВМ факультета вычислительной математики и кибернетики (http://www.software.unn.ac.ru). Выполнение проекта осуществлялось при поддержке компании Microsoft.