


Московский государственный университет им. М.В. Ломоносова


Введение в методы параллельных вычислений

Разработчик:
 А.В. Старченко, д.ф.-м.н., профессор
 E-mail: starch@math.tsu.ru
 Томский государственный университет

Направление 010300.68
 «Фундаментальная информатика и информационные технологии»

Проект комиссии Президента по модернизации и техническому развитию экономики России
 «Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения»

Суперкомпьютерный консорциум университетов России



Разработка курса выполнена в рамках Проекта комиссии Президента РФ по модернизации и техническому развитию экономики России «Создание системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения»


Применение потенциала суперкомпьютерных технологий (СКТ) как значимой составляющей инновационного развития страны является задачей государственной важности, относится к приоритетному направлению и находится под постоянным контролем Президента и Правительства России. Одним из сдерживающих факторов развития страны в этом направлении является острая нехватка высококвалифицированных кадров в области СКТ, поскольку подготовка таких специалистов сейчас отсутствует как элемент системы высшего профессионального образования.

Стратегической целью проекта является создание национальной системы подготовки высококвалифицированных кадров в области суперкомпьютерных технологий и специализированного программного обеспечения.

<http://hpc-education.ru>

© Московский государственный университет © Томский государственный университет Старченко А.В.

Суперкомпьютерный консорциум университетов России




Содержание курса

- Введение
- Рекуррентные формулы
- Параллельные вычисления определенных и кратных интегралов
- Умножение матрицы на вектор. Умножение матриц
- Прямые методы решения систем линейных уравнений на многопроцессорных системах Организация межпроцессорных обменов
- Трехдиагональные системы. Параллельная реализация прямых методов решения систем линейных уравнений
- Параллельная реализация итерационных методов решения СЛАУ
- Параллельная реализация быстрого преобразования Фурье

© Московский государственный университет © Томский государственный университет Старченко А.В.

Суперкомпьютерный консорциум университетов России




Содержание лекции

- Итерационные методы решения системы линейных уравнений с плотно заполненной матрицей
- Метод Якоби
 - Условия сходимости и критерии завершения итерационного процесса
 - Построение параллельной версии алгоритма на основе декомпозиции данных
 - Оценка ускорения и эффективности параллельного метода Якоби
- Метод Зейделя и метод SOR
 - Покомпонентная и матрично-векторная формы записи
 - Построение параллельной версии алгоритмов на основе асинхронной или хаотической релаксации
- Метод сопряженных градиентов

© Московский государственный университет © Томский государственный университет Старченко А.В.

Суперкомпьютерный консорциум университетов России




Итерационные методы решения СЛАУ с плотно заполненной матрицей

- Итерационные методы для решения линейных систем вида $Ax=b$ с невырожденной квадратной матрицей начинают расчет с заданного начального приближения x_0 и выполняют его последовательное улучшение до тех пор, пока приближенное решение не будет найдено с требуемой точностью.
- По теории возможно бесконечное число итераций для достижения точного решения. На практике итерации заканчиваются, когда норма невязки $r_k=b-Ax_k$ или другая мера ошибки не станет малой.

© Московский государственный университет © Томский государственный университет Старченко А.В.

Суперкомпьютерный консорциум университетов России



Итерационные методы решения СЛАУ с плотно заполненной матрицей

- Итерационные методы обычно используются для решения систем уравнений, количество которых слишком велико, чтобы их можно было обработать на современном компьютере прямыми методами.
- Кроме того, эти методы практически незаменимы при решении больших и плохообусловленных систем, поскольку строятся таким образом, чтобы погрешность метода во время поиска решения не накапливалась.
- Последовательные приближения к решению в таких методах обычно генерируются выполнением умножения матрицы на вектор.
- Итерационные методы не гарантируют получения решения для любой системы уравнений. Однако, когда они дают решение, то оно получается с меньшими затратами, чем прямыми методами.

© Московский государственный университет © Томский государственный университет Старченко А.В.



Итерационные методы решения СЛАУ с плотно заполненной матрицей

- Итерационные методы для решения линейных систем обычно включают следующие базовые операции линейной алгебры (уровня 1 или 2):
 - линейная комбинация векторов (*saxpy*);
 - скалярное произведение векторов (*dot*);
 - умножение матрицы на вектор (*gaxpy*);
 - решение систем с треугольными матрицами (*trsv*).
- При параллельной реализации таких операций обычно производится декомпозиция данных и операций по числу используемых параллельных процессов, которая сопровождается передачей данных между процессами для обеспечения локальных вычислений.



Метод Якоби

- Получая начальное приближение x_0 , метод Якоби рассчитывает новое приближение к точному решению для каждой компоненты по следующей формуле ($a_{ii} \neq 0, i = 1, 2, \dots, n$):

$$x_i^{(k+1)} = \frac{b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j^{(k)}}{a_{ii}}, i = 1, \dots, n; k = 0, 1, \dots; x_k = \begin{pmatrix} x_1^{(k)} \\ x_2^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix}$$



Метод Якоби

- Если обозначить за D диагональную матрицу, образованную диагональными элементами A , а за L и U нижнюю и верхнюю треугольные матрицы вида

$$L = \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & 0 \end{pmatrix} \quad U = \begin{pmatrix} 0 & a_{12} & \dots & a_{1n} \\ 0 & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}$$

- то метод Якоби можно записать как

$$x_{k+1} = -D^{-1}(L+U)x_k + D^{-1}b; k = 0, 1, \dots$$



Метод Якоби

- Для корректности метода Якоби требуются ненулевые диагональные элементы матрицы, что можно обеспечить перестановкой строк или столбцов такой матрицы, если это необходимо.
- При реализации метода Якоби на компьютере в памяти должны храниться все компоненты векторов x_{k+1} и x_k , поскольку ни одна из компонент не может быть переписана до тех пор, пока новое приближение не будет получено.
- Кроме того, следует обратить внимание, что все компоненты следующего приближения x_{k+1} могут быть рассчитаны одновременно и независимо, что открывает большие возможности при построении параллельной версии алгоритма.



Условия сходимости и критерии завершения итерационного процесса

- Если матрица A имеет строгое диагональное преобладание, что часто имеет место в практических задачах,

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, i = 1, \dots, n;$$

то метод Якоби сходится, хотя сходимость может быть очень медленной.

- В качестве критерия завершения итерационного процесса можно использовать следующее условие:

$$\|x_{k+1} - x_k\| = \max_i |x_i^{(k+1)} - x_i^{(k)}| < \varepsilon \text{ (and)} \|Ax_{k+1} - b\| < \varepsilon.$$



Метод Якоби

- Пользуясь формулой

$$x_i^{(k+1)} = \frac{b_i - \sum_{\substack{j=1 \\ j \neq i}}^n a_{ij} x_j^{(k)}}{a_{ii}}, i = 1, \dots, n; k = 0, 1, \dots,$$

- оценим временные затраты на выполнение одной итерации последовательного алгоритма метода Якоби:

$$T_1 \approx n \cdot ((n-1) \cdot (t_{add} + t_{mult}) + t_{add} + t_{div}) \approx 2 \cdot n^2 \cdot t_a; (n \gg 1).$$



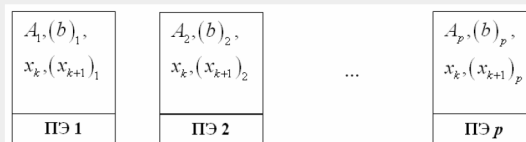
Построение параллельного алгоритма на основе декомпозиции данных

- На этапе декомпозиции в качестве фундаментальной мелкозернистой подзадачи выберем задачу расчета одной компоненты вектора x_{k+1} .
- При проектировании коммуникаций между подзадачами становится очевидным, что они не требуются, поскольку расчеты каждой компоненты $x_i^{(k+1)}$ на одной итерации ведутся независимо и одновременно. Однако необходимо контролировать выполнение условия завершения итераций, для чего нужно располагать всеми компонентами вектора x_{k+1} , A , b , x_k .
- Таким образом, потребуются сборка x_{k+1} либо на одном или на всех процессорных элементах и принятие решения о продолжении или завершении итерационного процесса.



Параллельная реализация метода Якоби

- Рассмотрим схему укрупнения мелкозернистых фундаментальных подзадач, при которой каждый укрупненный блок содержит по n/p таких подзадач.



- Распределение исходных данных по процессорным элементам



Параллельная реализация метода Якоби

- для $k=0, 1, 2, \dots$ выполнять на каждом ПЭμ:
 - {1} расчет $(x_{k+1})_\mu$
 - {2} расчет $\|(x_{k+1})_\mu - (x_k)_\mu\|$ и $\|(b)_\mu - A_\mu x_k\|$;
 - {3} пересылка полученных значений норм вместе с векторами $(x_{k+1})_\mu$ на остальные ПЭ;
 - {4} обновление вектора последнего приближения и проверка каждым ПЭμ выполнения условия завершения итераций. В случае, если условие выполнено, то выход из цикла.



Параллельная реализация метода Якоби

- Оценка временных затрат разработанного параллельного алгоритма на одной итерации без учета этапа инициализации:

$$T_p \approx 2 \frac{n^2}{p} t_a + 4 \frac{n}{p} t_a + (p-1) \left(\frac{n}{p} + 2 \right) t_{comm};$$

{1}
{2}
{3}

$$\left(\frac{n}{p} \gg 1 \right) \Rightarrow T_p \approx 2 \frac{n^2}{p} t_a + (p-1) \frac{n}{p} t_{comm}.$$



Оценка ускорения и эффективности параллельного метода Якоби

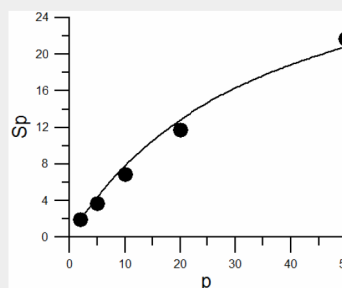
$$S_p = \frac{T_1}{T_p} \approx \frac{2n^2 t_a}{2 \frac{n^2}{p} t_a + (p-1) \frac{n}{p} t_{comm}} = \frac{p}{1 + \frac{(p-1) \cdot \kappa}{2n}};$$

$$E_p = \frac{S_p}{p} \approx \frac{1}{1 + \frac{(p-1) \cdot \kappa}{2n}}; \quad E_p \rightarrow 1, \quad \frac{n}{\kappa} \rightarrow \infty.$$

$$\kappa = t_{com} / t_a \gg 1.$$



Оценка ускорения и эффективности параллельного метода Якоби



- Ускорение параллельного метода Якоби при решении СЛАУ с плотно заполненной матрицей ($n=5000$).
- Линия – оценка S_p по формуле при $\kappa=140$.
- Значки – расчет на кластере ТГУ СКИФ Cyberia.



Метод Зейделя

- Покомпонентная форма записи метода Зейделя имеет вид ($a_{ii} \neq 0, i = 1, 2, \dots, n$):

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)}}{a_{ii}}, i = 1, 2, \dots, n;$$

- или

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} - \sum_{j=i+1}^n a_{ij} x_j^{(k+1)}}{a_{ii}}, i = n, n-1, \dots, 1.$$



Метод Зейделя

- Матрично-векторная форма записи имеет вид:

$$x_{k+1} = (D + L)^{-1} (b - Ux_k), k = 0, 1, 2, \dots;$$

$$x_{k+1} = (D + U)^{-1} (b - Lx_k), k = 0, 1, 2, \dots$$

- Метод Зейделя сходится при любом начальном приближении и любой правой части СЛАУ b , если матрица A имеет строгое диагональное преобладание или линейный оператор, которому соответствует матрица, самосопряжен и положителен ($A=A^*, A>0$).



Метод Зейделя. Асинхронный подход

- Рассмотренные формулы метода Зейделя являются рекуррентными, т.е. для расчета $x_i^{(k+1)}$ нужно сначала рассчитать, скажем, компоненты $x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_{i-1}^{(k+1)}$.
- Для преодоления этой ситуации в практике параллельных вычислений часто используют так называемую **асинхронную** или **хаотическую релаксацию**, когда при получении компонент вектора приближенного решения x_{k+1} используют последние значения компонент вектора x , имеющиеся на каждом процессорном элементе.



Метод Зейделя. Асинхронный подход

- Асинхронная релаксация может быть эффективной, однако порядок выполнения вычислительной работы в ней отличается от организации расчетов в последовательном методе Зейделя, что существенно усложняет анализ сходимости полученного параллельного метода решения систем линейных алгебраических уравнений.
- При решении СЛАУ с плотно заполненной матрицей можно воспользоваться матрично-векторной формой записи метода Зейделя, согласно которой расчет компонент вектора x_{k+1} может вестись на каждой итерации при решении системы с верхне- или нижнетреугольной матрицей. В предыдущей главе было отмечено, что для таких систем параллельные методы оказываются лишь умеренно эффективными.



Метод SOR

- Метод релаксации (метод SOR (**s**uccessive **o**ver **r**elaxation)) имеет вид: $a_{ii} \neq 0, i = 1, 2, \dots, n$

$$x_i^{(k+1)} = (1 - \omega)x_i^{(k)} + \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^n a_{ij} x_j^{(k)} \right];$$

- Матрично-векторный вид:

$$x_{k+1} = (D + \omega L)^{-1} [(1 - \omega)D - \omega U] x_k + \omega (D + \omega L)^{-1} b;$$

- Итерации метода SOR сходятся при любом начальном приближении, если A - симметричная положительно определенная матрица и параметр релаксации $\omega \in (0, 2)$.



Метод SOR

- Параллельная реализация метода SOR для линейных систем с плотно заполненной матрицей осуществляется аналогично распараллеливанию метода Зейделя, т.е. на каждой итерации решаются параллельным методом треугольные системы вида

$$(D + \omega L)x_{k+1} = [(1 - \omega)D - \omega U]x_k + \omega b, k = 0, 1, 2, \dots$$

- или используются асинхронные или хаотические релаксации.



Метод SOR

- Применение асинхронного подхода в качестве способа распараллеливания метода Зейделя или метода SOR ведет, как правило, к увеличению общего числа итераций при заданной точности вычислений по сравнению с соответствующими последовательными алгоритмами, причем при увеличении степени декомпозиции исходных данных (с ростом числа используемых процессорных элементов) количество итераций повышается.
- Теоретические оценки ускорения и эффективности параллельных алгоритмов методов Зейделя и метода SOR, базирующихся на асинхронном подходе, провести намного сложнее, чем для метода Якоби, для которого общее число итераций в последовательной и параллельной версиях алгоритма совпадают.



Метод сопряженных градиентов

- Если матрица $A \in \mathbb{R}^{n \times n}$ является симметричной и положительно определенной, то решение СЛАУ $Ax = b$ можно найти используя метод сопряженных градиентов.
- **Теорема.** Если матрица A удовлетворяет указанным выше свойствам, а p_0, p_1, \dots, p_{n-1} - ненулевые линейно независимые векторы, удовлетворяющие условию $(p_i, Ap_j) = 0, i \neq j$,
- то при любом начальном приближении x_0 итерационные приближения $x_{k+1} = x_k - \alpha_k p_k$, $\alpha_k = -(p_k, r_k) / (p_k, Ap_k), r_k = b - Ax_k$
- сходятся к точному решению не более чем за n шагов.



Метод сопряженных градиентов

- - выбрать x_0 , вычислить $r_0 = b - Ax_0$, положить $p_0 = r_0$,
 - рассчитать (r_0, r_0) ,
 - для $k = 0, 1, 2, \dots$
- $$\alpha_k = -(r_k, r_k) / (p_k, Ap_k), \quad (1)$$
- $$x_{k+1} = x_k - \alpha_k p_k, \quad (2)$$
- $$r_{k+1} = r_k + \alpha_k Ap_k, \quad (3)$$
- рассчитать (r_{k+1}, r_{k+1}) и если $\sqrt{(r_{k+1}, r_{k+1})} \geq \varepsilon$, продолжать
- $$\beta_k = (r_{k+1}, r_{k+1}) / (r_k, r_k), \quad (5)$$
- $$p_{k+1} = r_{k+1} + \beta_k p_k. \quad (6)$$



Метод сопряженных градиентов

- Из формул (1)-(6) видно, что на каждой итерации метода производится **одно** умножение матрицы на вектор в (1), **два** скалярных произведения в (1) и (4), **три** линейных комбинации векторов $saxru$ ((2), (3), (6)), а также несколько скалярных операций.
- При $n \gg 1$ временные затраты на проведение одной итерации метода сопряженных градиентов можно оценить как $T_1 \approx (2 \cdot n^2 + 2 \cdot (2 \cdot n) + 3 \cdot (2 \cdot n)) \cdot t_a \approx 2 \cdot n^2 \cdot t_a$.
- Таким образом, по вычислительной трудоемкости выполнение одной итерации метода сопряженных градиентов аналогично выполнению итерации метода Якоби.



Параллельная реализация CG

- В параллельной реализации как данные (векторы b, r_k, p_k, x_k и матрица A), так и основные операции (скалярные произведения, линейные комбинации векторов $saxru$, умножение матрицы на вектор) разделяются на p используемых укрупненных подзадач.
- В дополнение к коммуникациям (межпроцессорным обменам данных), требуемым для осуществления этих основных операций, необходимы также дополнительные коммуникации для проверки сходимости вычислительного процесса (выполнение редукции sum или max при вычислении норм $\|r_k\|$ и $\|x_{k+1} - x_k\|$).



Параллельная реализация CG

- Декомпозиция векторов обычно осуществляется равномерно между p подзадачами (процессорными элементами), причем каждая подзадача содержит компоненты векторов с одним и тем же множеством индексов, например на ПЭМ ($\mu=1, \dots, p$) распределяются компоненты векторов b, r_k, p_k, x_k с номерами индексов $i=1+(\mu-1) \cdot n/p, \dots, \mu \cdot n/p$ ($\mu=1, \dots, p$).
- Поэтому при выполнении операции линейной комбинации векторов $saxru$ не требуются коммуникации (межпроцессорная передача данных), в то время как при вычислении скалярных произведений они просто необходимы.



Параллельная реализация CG

- Декомпозиция матрицы A может осуществляться по строкам, по столбцам или на блоки.
- Для плотно заполненной матрицы декомпозиция на строчные блоки $A\mu$ ($\mu=1, \dots, p$) обеспечивает вычисление произведения $A\mu \cdot p_k$ без межпроцессорных коммуникаций.
- В то же время, коммуникации потребуются на следующем этапе, когда необходимо получить остальные компоненты произведения Ap_k для вычисления скалярного произведения (p_k, Ap_k) .



Параллельная реализация CG

- При распределении строчных блоков из n/p строк матрицы A и по n/p компонентом векторов по подзадачам коммуникации требуются для операций (1) и (4)

- выбрать x_0 , вычислить $r_0 = b - Ax_0$, положить $p_0 = r_0$,
 - рассчитать (r_0, r_0) ,
 - для $k = 0, 1, 2, \dots$

$$\alpha_k = -(r_k, r_k) / (p_k, Ap_k), \quad (1)$$

$$x_{k+1} = x_k + \alpha_k p_k, \quad (2)$$

$$r_{k+1} = r_k + \alpha_k Ap_k, \quad (3)$$

$$\text{рассчитать } (r_{k+1}, r_{k+1}) \text{ и если } \sqrt{(r_{k+1}, r_{k+1})} \geq \varepsilon, \text{ продолжить} \quad (4)$$

$$\beta_k = (r_{k+1}, r_{k+1}) / (r_k, r_k), \quad (5)$$

$$p_{k+1} = r_{k+1} + \beta_k p_k. \quad (6)$$



Параллельная реализация CG

- При проведении оценок ускорения и эффективности построенного параллельного алгоритма нужно иметь в виду, что объем вычислительной работы, выполняемой всеми подзадачами, в сумме совпадает с объемом вычислений в последовательном алгоритме метода сопряженных градиентов.
- На (1) и (4) шагах параллельного алгоритма требуется сборка векторов p_k, Ap_k и r_{k+1} на каждом ПЭ (пересылка каждым ПЭ остальным по n/p чисел) с последующим вычислением скалярных произведений.
- И хотя в этом случае производится дублирование вычислений скалярных произведений, однако не требуется пересылки полученных результатов операции dot.



Параллельная реализация CG

- Другая стратегия организации межпроцессорных коммуникаций для выполнения параллельного алгоритма заключается в следующем.
- Для осуществления операции Ap_k на шаге (1) производится сборка вектора p_k на каждом ПЭ.
- В вычислении скалярного произведения участвуют все ПЭ. Рассчитав $[(p_k, Ap_k)]_\mu$, полученное значение передается остальным ПЭ для проведения операции редукции sum.
- В итоге, в первом варианте межпроцессорной передачи данных на каждой итерации требуется каждому ПЭ пересылать по $3n/p$ чисел остальным, то во втором – по $n/p+2$, причем нет дублирования при вычислении dot.



Параллельная реализация CG

- Оценим временные затраты параллельного алгоритма на каждой итерации

$$T_p \approx 2 \frac{n^2}{p} t_a + (p-1) \left(\frac{n}{p} + 2 \right) t_{comm};$$

- Тогда при $n/p \gg 1$

$$S_p = \frac{T_1}{T_p} \approx \frac{p}{1 + \frac{(p-1)k}{2n}}; \quad k = \frac{t_{comm}}{t_a} \gg 1.$$

- Ускорение параллельного метода сопряженных градиентов совпадает с ускорением метода Якоби.



Предобусловленный метод сопряженных градиентов

- Скорость сходимости метода сопряженных градиентов на каждой итерации можно оценить по следующей формуле: $\|x_k - x^*\|_2 \leq 2\sqrt{v} a^k \|x_0 - x^*\|_2$, $a = (\sqrt{v}-1)/(\sqrt{v}+1)$,
- x^* - точное решение; $v = cond(A) = \|A\| \cdot \|A^{-1}\|$
- Если $v \sim 1, a \ll 1$ сходимость быстрая.
- Если $v \gg 1, a \sim 1$ сходимость медленная.
- Это наблюдение приводит к общему понятию предобуславливания матрицы A посредством преобразования конгруэнтности $\tilde{A} = SAS^T$, где S - невырожденная матрица, выбранная так, чтобы $cond(\tilde{A}) < cond(A)$.



Предобусловленный метод сопряженных градиентов

- Тогда система, которую нужно решить, запишется в виде $\tilde{A}\tilde{x} = \tilde{b}$; $\tilde{A} = SAS'$; $\tilde{x} = (S')^{-1}x$; $\tilde{b} = Sb$.
- Рассмотрим подход предобуславливания, основанный на использовании симметричной положительно определенной матрицы M , аппроксимирующей матрицу A , и $(S'S)^{-1} = M$.
- Обычно в качестве матрицы M для предобуславливания СЛАУ используется либо диагональная либо треугольная матрица. Основными требованиями выбора матрицы M является простота решения системы и чтобы матрица M хорошо аппроксимировала матрицу A .



Предобусловленный метод сопряженных градиентов

- выбрать x_0 , вычислить $r_0 = b - Ax_0$, решить систему $M\tilde{r}_0 = r_0$, положить $p_0 = \tilde{r}_0$,
- рассчитать (\tilde{r}_0, r_0) ,
- для $k = 0, 1, 2, \dots$
 - $\alpha_k = -(\tilde{r}_k, r_k) / (p_k, Ap_k)$, (1)
 - $x_{k+1} = x_k - \alpha_k p_k$, (2)
 - $r_{k+1} = r_k + \alpha_k Ap_k$, (3)
 - рассчитать (r_{k+1}, r_{k+1}) и если $\sqrt{(r_{k+1}, r_{k+1})} \geq \varepsilon$, продолжать (4)
 - решить систему $M\tilde{r}_{k+1} = r_{k+1}$, (5)
 - $\beta_k = (\tilde{r}_{k+1}, r_{k+1}) / (\tilde{r}_k, r_k)$, (6)
 - $p_{k+1} = \tilde{r}_{k+1} + \beta_k p_k$. (7)



Предобусловленный метод сопряженных градиентов

- Для предобуславливания систем часто используют:
 - Диагональные или блочно диагональные матрицы
 - Метод последовательной релаксации SSOR
 - Неполную факторизацию
 - Полиномиальные предобуславливатели.
- Поскольку при построении параллельных версий предобусловленного метода сопряженных градиентов не все предобуславливатели эффективно распараллеливаются, необходимо специальное внимание уделять их выбору.



Заключение

- Рассмотрены параллельные алгоритмы итерационных методов решения систем линейных уравнений с плотно заполненной матрицей: метода Якоби, Зейделя, верхней релаксации SOR, метода сопряженных градиентов CG.
- Алгоритмы основываются на декомпозиции данных и базовых операций линейной алгебры – dot, saxpy, gaxpy.
- Для алгоритмов, включающих расчеты по рекуррентным формулам (методы Зейделя, SOR, предобусловленный метод CG), предлагается использовать асинхронный подход.



Вопросы для обсуждения

- В чем заключается идея итерационных методов решения СЛАУ и преимущество их применения перед прямыми методами?
- Какие базовые операции линейной алгебры обычно включают итерационные методы и какими способами они распараллеливаются?
- Оцените временные затраты на выполнение одной итерации в методе Якоби.
- Дайте словесное описание параллельного алгоритма метода Якоби.
- Как следует организовать проверку условия завершения итерационного процесса в параллельном алгоритме метода Якоби?
- Нарисуйте схему распределения данных по процессорным элементам в параллельном алгоритме метода Якоби.
- Выполните оценку временных затрат в параллельном алгоритме метода Якоби. Оцените ускорение параллельного алгоритма метода Якоби.



Вопросы для обсуждения

- В чем заключается проблема распараллеливания метода Зейделя? Прокомментируйте с использованием формул метода.
- Что собой представляет асинхронная или хаотическая релаксация?
- В чем заключается проблема применения асинхронной релаксации?
- Опишите основные этапы метода сопряженных градиентов и оцените временные затраты на проведение одной итерации последовательного алгоритма.
- Каким образом осуществляется построение параллельной версии метода CG?
- Дайте оценку временных затрат одной итерации параллельного алгоритма метода CG.



Темы заданий для самостоятельной работы

- Оцените ускорение и эффективность асинхронного параллельного алгоритма метода Зейделя.
- Оцените ускорение и эффективность асинхронного параллельного алгоритма метода верхней релаксации.
- Написать MPI-программу параллельного алгоритма метода Якоби. Исследовать ее ускорение при решении систем с плотно заполненной матрицей.
- Написать MPI-программу параллельного алгоритма метода Зейделя. Исследовать ее ускорение при решении систем с плотно заполненной матрицей.



Основная литература

1. *Высокопроизводительные вычисления на кластерах.* – Томск: Изд-во Том. ун-та, 2008.
2. *Хокни Р., Джессхоуп К.* Параллельные ЭВМ. Архитектура, программирование и алгоритмы. М: Радио и связь, 1986.
3. *Ортега Дж.* Введение в параллельные и векторные методы решения линейных систем. М.: Мир, 1991.
4. <http://www.cse.illinois.edu/courses/cs554/notes/>
5. *Гергель В.П.* Теория и практика параллельных вычислений. – М.: Интернет-Университет Информационных технологий; БИНОМ. Лаборатория знаний, 2007.
6. *Деммель Дж.* Вычислительная линейная алгебра. – М.: Мир, 2001.



Следующая тема

- Введение
- Рекуррентные формулы
- Параллельные вычисления определенных и кратных интегралов
- Умножение матрицы на вектор. Умножение матриц
- Прямые методы решения систем линейных уравнений на многопроцессорных системах Организация межпроцессорных обменов
- Треугольные системы. Параллельная реализация прямых методов решения систем линейных уравнений
- Параллельная реализация итерационных методов решения СЛАУ
- Параллельная реализация быстрого преобразования Фурье