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»

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.

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

Содержание лекции

- Метод исключения Гаусса. Обзор
- LU-факторизация, последовательный алгоритм
- Построение параллельного алгоритма LU-разложения
- Параллельный алгоритм LU-разложения при 1D строковом укрупнении
- Особенности параллельного алгоритма. Организация опережающей рассылки
- Параллельные алгоритмы LU-разложения при 1D столбцовом и 2D укрупнении
- Проблема частичного выбора главного элемента
- Решение треугольных систем, последовательный алгоритм
- Построение параллельных алгоритмов решения треугольных систем

Метод исключения Гаусса

- Рассмотрим систему линейных алгебраических уравнений

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1; \\ &a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2; \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n. \end{aligned} \qquad (*)$$

- с невырожденной плотно заполненной матрицей ***A*** размера ***n*n***.
- Суть метода исключения Гаусса состоит в приведении эквивалентными преобразованиями системы (*) к системе с треугольной или диагональной матрицей.

Метод исключения Гаусса

- LU-теорема: Пусть дана квадратная матрица ***A*** порядка ***n***, и пусть A_k – главный минор матрицы, составленный из первых ***k*** строк и столбцов. Предположим, что $det(A_k) \neq 0$ для ***k=1,2,…,n***. Тогда существуют единственная нижняя треугольная матрица ***L*** с единичными элементами на главной диагонали и единственная верхняя треугольная матрица ***U*** такие, что ***A=LU***.
- Решая систему с нижней треугольной матрицей ***Ly=b*** прямой подстановкой, получим вектор ***y***. Затем, решая систему с верхней треугольной матрицей ***Ux=y*** обратной подстановкой, находим решение исходной системы уравнений ***x***.

LU-разложение

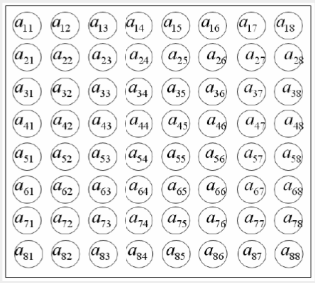
- Псевдокод:
  - ***do k=1,n-1***
    - ***do i=k+1,n***
      - $l_{ik} = a_{ik}/a_{kk}$
    - ***end do***
    - ***do j=k+1,n***
      - ***do i=k+1,n***
        - $a_{ij} = a_{ij} - l_{ik}*a_{kj}$
      - ***end do***
    - ***end do***
  - ***end do***
- В методе исключения Гаусса выполняется около $n^3/3$ парных операций сложения и умножения, а также $n^2/2$ делений.

$$T_1 \approx (t_{mult} + t_{add}) \cdot n^3 / 3$$

- Индексы ***i,j,k*** в циклах могут быть взяты в любом порядке: формы ***jki*** и ***kji*** – столбцово ориентированы; формы ***kij*** и ***ikj*** – строково ориентированы.

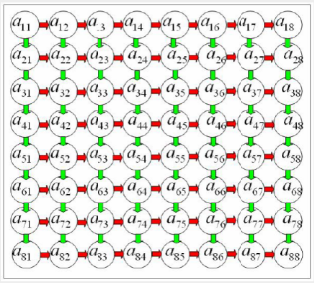
Построение параллельного алгоритма LU-разложения

- Декомпозиция. Выберем в качестве мелкозернистых фундаментальных подзадач следующее: для каждого ***i,j*** такая подзадача содержит a_{ij}, вычисляет и запоминает u_{ij} при $i \le j$ или l_{ij} при $i > j$. Все такие фундаментальные задачи образуют двумерный массив n^2 подзадач.



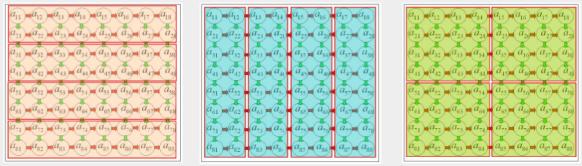
Построение параллельного алгоритма LU-разложения

- Проектирование коммуникаций.
  - ***do k=1,min(i,j)-1***
    - ***получить*** a_{kj}
    - ***получить*** l_{ik}
    - $a_{ij} = a_{ij} - l_{ik}*a_{kj}$
  - ***end do***
  - ***if*** $i \le j$ ***then***
    - ***разослать*** a_{ij} ***подзадачам (k,j)***, ***k=i+1,…,n***
  - ***else***
    - ***получить*** a_{jj}
    - $l_{ij} = a_{ij}/a_{jj}$
    - ***разослать*** l_{ij} ***подзадачам (i,k)***, ***k=j+1,…,n***
  - ***end if***



Построение параллельного алгоритма LU-разложения

- Укрупнение. Для имеющегося массива фундаментальных подзадач могут быть использованы:
  - Одномерное укрупнение (в блоки объединяются по ***n/p*** строк или столбцов)
  - Двумерное укрупнение (одна составная подзадача включает n^2/p фундаментальных подзадач)

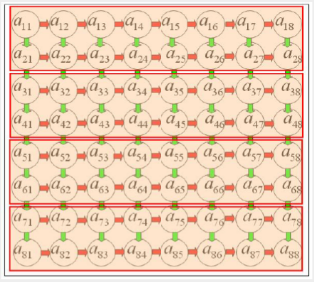


n=8 p=4

Построение параллельного алгоритма LU-разложения

- Планирование вычислений. Распределение составных подзадач по процессорным элементам производится с учетом архитектуры многопроцессорной вычислительной системы.

1D строковое укрупнение



- В этом случае нет необходимости рассылать множители l_{ij} по строкам, т.к. любая строка заданной матрицы целиком принадлежит одной составной подзадаче.
- После пересчета элементов в строке необходима рассылка полученных значений, чтобы обеспечить вычисления для каждой строки нижней составной подзадачи.

Параллельная реализация LU-разложения

```
do k = 1,n-1
   if k ∈ myrows then
    переслать {a_kj: k≤j≤n} другим
    составным подзадачам
   else
     получить {a_kj: k≤j≤n}
   end if
   do i ∈ myrows, i>k
      l_ik = a_ik/a_kk
   end do
   do j = k+1,n
      do i ∈ myrows, i>k
         a_ij = a_ij – l_ik*a_kj
      end do
    end do
end do
```

- Здесь ***myrows*** – множество индексных значений номеров строк, распределенных одной составной подзадаче.
- В этом алгоритма на каждом шаге k каждой задаче требуется $(n-k)^2/p$ флопов. В итоге

$$T_p^{calc} \approx (t_{mult} + t_{add})n^3 / 3p;$$

- Количество данных, пересылаемых на каждом шаге – $(n-k)$. Тогда

$$T_p^{comm} \approx t_{comm} n^2 / 2;$$

Особенности параллельной реализации LU-разложения

- Каждая составная подзадача считается выполненной, как только заканчивается обработка последней строки распределенной части матрицы ***A***. Если составная подзадача объединяет строки матрицы ***A*** с последовательной нумерацией, то обрабатывающий строку процессорный элемент завершает свою работу задолго до окончания общего вычислительного процесса!
- Обработка строк в порядке увеличения их номера ***i*** требует существенно меньшего объема вычислительной работы (по сравнению, например, с обработкой строки с номером 2), что ведет к неравномерной загрузке процессоров.

Особенности параллельной реализации LU-разложения

- Балансировка загрузки процессоров может быть улучшена, если строки матрицы ***A*** для составных подзадач объединять циклически, т.е. строка ***i*** назначается составной подзадаче (процессорному элементу) с номером ***i mod p, i=1,…,n***.

строки:	строки:	…	строки:	…	строки:
1	*2*		*m*		*p*
p+1	*p+2*		*p+m*		*2p*
…	…		…		…
(k-1)p+1	*(k-1)p+2*		*(k-1)p+m*		*kp*
ПЭ 1	ПЭ 2		ПЭ *m*		ПЭ *p*

Особенности параллельной реализации LU-разложения

- Также возможно использовать другие способы распределения данных, например, применение слоистой схемы с отражениями. Для этой схемы первые строк матрицы ***A*** распределяются между процессорными элементами в естественном порядке, следующие строк размещаются в обращенном порядке и т.д.

строки:	строки:	…	строки:	…	строки:
1	*2*		*m*		*p*
2p	*2p-1*		*2p-m*		*p+1*
…	…		…		…
(k-1)p+1	*(k-1)p+2*		*(k-1)p+m*		*kp*
ПЭ 1	ПЭ 2		ПЭ *m*		ПЭ *p*

Особенности параллельной реализации LU-разложения

- Производительность параллельных вычислений может быть увеличена за счет совмещения коммуникационных операций и вычислений.
- Пусть для циклической схемы распределения данных процесс обработки строк ведется в порядке увеличения их номера.

строки:	строки:	…	строки:	…	строки:
1	*2*		*m*		*p*
p+1	*p+2*		*p+m*		*2p*
…	…		…		…
(k-1)p+1	*(k-1)p+2*		*(k-1)p+m*		*kp*
ПЭ 1	ПЭ 2		ПЭ *m*		ПЭ *p*

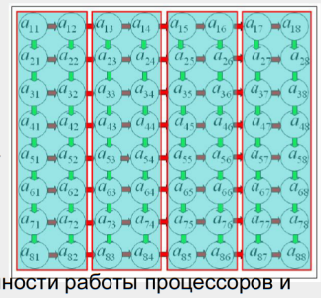
Особенности параллельной реализации LU-разложения

- Наиболее эффективным образом можно организовать параллельные вычисления следующим образом: на ***m***-ом шаге только что обработанная (***m+1***)-ая строка рассылается сразу после того, как закончится ее обработка.
- Такая стратегия осуществления параллельных вычислений называется *опережающей рассылкой*.

строки:	строки:	…	строки:	…	строки:
1	*2*		*m*		*p*
p+1	*p+2*		*p+m*		*2p*
…	…		…		…
(k-1)p+1	*(k-1)p+2*		*(k-1)p+m*		*kp*
ПЭ 1	ПЭ 2		ПЭ *m*		ПЭ *p*

Параллельные алгоритмы LU-разложения при 1D столбцовом и 2D укрупнении

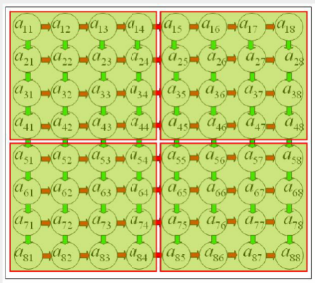
- При 1D столбцовом укрупнении нет необходимости в пересылке строк матрицы ***A***.
- Однако при таком способе распределения данных нет явного параллелизма при вычислении множителей l_{ij} и модификации элементов в столбцах матрицы.



Для повышения согласованности работы процессоров и балансировки загрузки можно использовать циклическое распределение столбцов матрицы.

Параллельные алгоритмы LU-разложения при 1D столбцовом и 2D укрупнении

- В 2D укрупнении алгоритм параллельной *LU*-факторизации совмещает свойства рассмотренных 1D схем укрупнения.
- Требуется горизонтальная и вертикальная рассылка строк матрицы ***A*** и столбцов множителей l_{ij} соответственно.



$$T_p \approx (t_{mult} + t_{add})n^3 / 3p + t_{comm} n^2 / \sqrt{p}.$$

Частичный выбор главного элемента

- Чтобы обеспечить корректность и устойчивость *LU*-разложения на практике применяется схема частичного выбора главного элемента, в результате которой в непреобразованной части матрицы производится перестановка строк таким образом, чтобы в ведущем столбце находился наибольший по модулю элемент матрицы.
- При такой стратегии множители по абсолютной величине не будут превышать единицы, что способствует уменьшению влияния ошибки округления на конечный результат.

Частичный выбор главного элемента

- При 1D столбцовой декомпозиции поиск главного элемента не требует обмена данными между процессорными элементами, – он ведется каким-то одним процессором. Однако, в этом случае остальные процессоры простаивают. Выход из такой ситуации может быть обеспечен путем опережающего вычисления и рассылки.
- На ***k***-ом шаге поиск главного элемента в ***k+1***-ом столбце сразу же начинается после модификации этого столбца.
- Как только будет найдена ведущая строка, эта информация сразу же передается другим процессорным элементам, которые могут параллельно выполнить перестановку строк.

Решение систем с треугольными матрицами ***Lx=b***

- Для системы с нижней треугольной матрицей Lx=b решение может быть получено в результате прямой подстановки

$$x_i = \left(b_i - \sum_{j=1}^{i-1} l_{ij} x_j \right) / l_{ii},$$

$$i = 1,\ldots,n;$$

$$T_1 \approx (t_{add} + t_{mult}) n^2 / 2$$

- Псевдокод этого алгоритма выглядит следующим образом:

```
do i = 1,n
  x_i = b_i/l_ii
  do j = i+1,n
    b_j = b_j – l_ji * x_i
  end do
end do
```


Решение систем с треугольными матрицами ***Ux=b***

- При решении системы с верхней треугольной матрицей требуется применять обратную подстановку

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij} x_j \right) / u_{ii},$$

$$i = n,\ldots,1;$$

$$T_1 \approx (t_{add} + t_{mult}) n^2 / 2$$

- Псевдокод имеет вид

```
do i = n,1
  x_i = b_i/u_ii
  do j = 1,i-1
    b_j = b_j – u_ji * x_i
  end do
end do
```

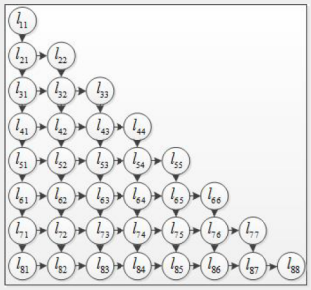
- Этот псевдокод полезен, если матрица U хранится в памяти компьютера по столбцам.

Параллельный алгоритм решения СЛАУ ***Lx=b*** (основной подход)

- Декомпозиция. В качестве фундаментальных мелкозернистых подзадач будем выбирать следующие:
  - Для ***i=2,…,n***; ***j=1,…,i-1 (i,j)***-ая подзадача запоминает l_{ij}, получает x_j и вычисляет произведение $l_{ij} \cdot x_j$
  - Для ***i=1,…,n (i,i)***-ая подзадача хранит значения l_{ii} и b_i, собирает произведения $l_{ij} \cdot x_j$, вычисляет сумму
  - $t_i = \sum_{j=1}^{i-1} l_{ij} x_j$
  - вычисляет и запоминает $x_i=(b_i-t_i)/l_{ii}$.
- Таким образом, декомпозиция дает двумерный треугольный массив n(n+1)/2 подзадач.

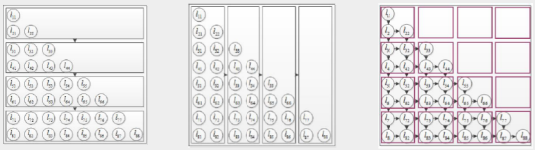
Параллельный алгоритм решения СЛАУ ***Lx=b*** (основной подход)

- Проектирование коммуникаций. При определении связанности функциональных подзадач можно заметить, что:
  - Для ***j=1,…,n-1 (j,j)***-ая подзадача одновременно передает значение x_j подзадачам ***(i,j), i=j+1,…,n***.
  - Затем производится передача произведений $l_{ij} \cdot x_j$ от подзадач ***(i,j)***, ***j=1,…,i-1*** подзадаче ***(i,i)***.



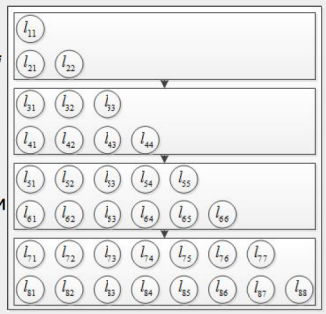
Параллельный алгоритм решения СЛАУ ***Lx=b*** (основной подход)

- Укрупнение. Имеющийся двумерный массив фундаментальных подзадач может быть объединен в ***p*** блоков следующим образом:
  - Одномерное укрупнение по ***n/p*** строк (или по ***n/p*** столбцов).
  - Двумерное укрупнение по ***n^2/p*** подзадач.
- Планирование вычислений осуществляется путем назначения укрупненных подзадач соответствующим процессорным элементам.



1D строковое укрупнение

- Такой способ укрупнения не требует передачи данных для вычисления суммы значений t_i по строкам, поскольку строка матрицы мелкозернистых подзадач целиком принадлежит укрупненной подзадаче.
- Заметим, что нет никакого параллелизма при вычислении этих сумм.
- Необходима передача вычисленных значений неизвестных x_i другим укрупненным подзадачам.



Особенности параллельного алгоритма при 1D укрупнении

- Выполнение каждой укрупненной подзадачи заканчивается после решения последнего уравнения, принадлежащего этой подзадаче.
- При последовательном распределении уравнений по блокам укрупненная подзадача может быть выполнена задолго до того, как будет закончена вся вычислительная работа по решению треугольных систем.
- Расчет скалярных произведений для каждой строки мелкозернистых подзадач требует большего количества вычислений при увеличении ее номера.

Особенности параллельного алгоритма при 1D укрупнении

- Согласованность работы процессоров и балансировка их загрузки может быть улучшена за счет объединения строк в блоки в циклическом порядке, когда ***i***-ая строка подзадач назначается укрупненной подзадаче с номером ***i mod p***.

Особенности параллельного алгоритма при 1D укрупнении

- Для решения проблемы равномерной загрузки процессоров при укрупнении подзадач также могут быть использованы блочный циклический способ объединения строк или схема с отражениями.

Оценка ускорения алгоритма

- Время, необходимое для вычислений:

$$T_p^{comp} \approx n\left(t_{dev} + (t_{add} + t_{mult})\frac{n}{2p}\right) \approx (t_{add} + t_{mult})\frac{n^2}{2p}$$

- Время, затрачиваемое на передачу $\boldsymbol{x_i}$:

$$T_p^{comm} = t_{com}\frac{n(p-1)}{p}$$

- Ускорение:

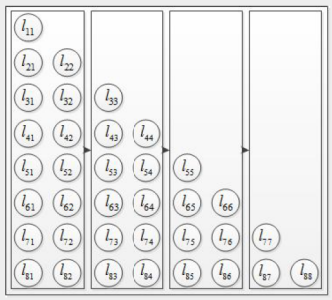
$$S_p = \frac{T_1}{T_p} \approx \frac{p}{1 + \frac{2t_{com}(p-1)}{(t_{add} + t_{mult})n}}$$

Повышение эффективности

- Повышение эффективности построенной параллельной программы может быть достигнуто при уменьшении доли временных затрат на обеспечение пересылки данных между вычислительными узлами.
- Одним из путей решения этой проблемы является использование стратегии опережающей рассылки, т.е. рассылка данных совмещается с вычислениями.
- На ***j***-ом шаге укрупненная подзадача, включающая ***j+1***-ое уравнение, может рассчитать значение $\boldsymbol{x_{j+1}}$ и немедленно передать его другим укрупненным подзадачам перед осуществлением окончательных вычислений с использованием $\boldsymbol{x_j}$.

1D столбцовое укрупнение

- Нет необходимости рассылать вычисленные значения неизвестных по вертикали, поскольку каждый необходимый столбец матрицы целиком принадлежит только одной укрупненной подзадаче.
- Для вычисления суммы произведений $\boldsymbol{\Sigma l_{ij} \cdot x_j}$ потребуются пересылки между укрупненными блоками по горизонтали.



Особенности параллельного алгоритма

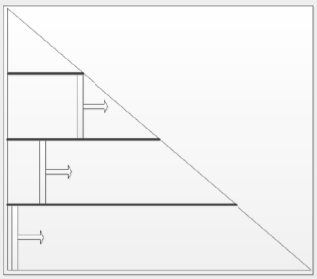
- Каждая укрупненная подзадача начинает выполняться только тогда, когда начнет вычисляться неизвестная $\boldsymbol{x_j}$ ее первого столбца.
- Если каждая укрупненная подзадача объединяет последовательные блоки столбцов, она может оставаться в бездействии большую часть вычислений.
- Кроме того, количество произведений, осуществляемых с одной компонентой вектора ***x***, уменьшается с увеличением номера столбца.
- Равномерная загрузка процессорных элементов может быть улучшена за счет назначения столбцов блокам по циклической схеме. Кроме того, могут использоваться другие схемы, например, блочно-циклическая или схема с отражениями.

Волновой алгоритм

- Параллелизм рассмотренных выше алгоритмов получается из внутреннего цикла, выполнение которого разбивается и распределяется по укрупненным задачам.
- Внешний цикл выполняется последовательно.
- Эти параллельные алгоритмы вычисляют только по одной компоненте решения ***x***, хотя для повышения эффективности эти последовательные шаги алгоритма могут обрабатываться конвейерным способом.
- Так называемые волновые алгоритмы используют параллелизм явно и для внешнего цикла, обрабатывая многочисленные компоненты решения одновременно.

Волновой алгоритм

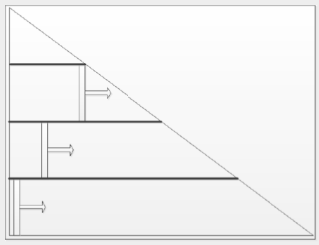
- Рассмотрим 1D столбцовое укрупнение.
- Подзадача, располагающая столбцом ***j***, рассчитывает только ***s*** компонент обновляемого вектора ***t*** и пересылает их подзадаче, имеющей столбец ***j+1***, перед продолжением работы со следующими ***s*** компонентами.



$$T_p \approx (t_{add} + t_{mult} + t_{comm} / s)(n^2 + np + s(s-1)p^2) / (2p)$$

Волновой алгоритм

- По мере того, как длина сегмента ***s*** увеличивается, затраты на передачу данных уменьшаются, однако растут затраты на вычисления.
- Волновой алгоритм может применяться и для 1D строкового укрупнения.



Заключение

- Рассмотрена адаптация алгоритма метода исключения Гаусса для решения систем линейных уравнений с плотно заполненной матрицей на многопроцессорные вычислительные системы с распределенной памятью.
- Для алгоритмов LU-факторизации и решения треугольных систем на основе общего подхода построены параллельные алгоритмы для 1D и 2D укрупнения.
- Выполнена оценка ускорения этих алгоритмов с учетом коммуникационных затрат.
- Рассмотрены подходы повышения производительности параллельных алгоритмов.

Вопросы для обсуждения

- Для чего нужна LU-факторизация матрицы ***A*** в методе исключения Гаусса?
- Укажите алгоритмическую сложность LU-факторизации.
- Что может быть выбрано в качестве мелкозернистой фундаментальной подзадачи в алгоритме LU-факторизации? Как связаны они между собой?
- Какие стратегии укрупнения могут использоваться при построении параллельного алгоритма? Их особенности.
- В чем заключаются недостатки 1D строкового укрупнения, в котором строки распределяются по порядковым номерам го укрупненным подзадачам?
- Что такое циклическая схема распределения подзадач?
- В чем заключаются особенности слоистой строчной схемы с отражениями?
- Для чего используется опережающая рассылка? В чем заключается ее преимущество?
- Где меньше коммуникационные затраты в параллельном алгоритме LU-факторизации: в 1D строковом укрупнении или в 2D укрупнении?

Вопросы для обсуждения

- Как реализуется частичный выбор главного элемента в параллельном алгоритме LU-факторизации?
- Как можно минимизировать коммуникационные затраты в параллельном алгоритме LU-факторизации?
- Какова алгоритмическая сложность решения систем с треугольными матрицами?
- Что можно выбрать в качестве мелкозернистой фундаментальной подзадачи при построении параллельного алгоритма? Как они связаны между собой?
- Какие стратегии укрупнения можно использовать при построении параллельного алгоритма решения систем с треугольными матрицами? Укажите особенности их реализации.
- В чем заключаются недостатки 1D строковой схемы укрупнения?
- Как можно повысить эффективность 1D строковой схемы укрупнения?
- В чем заключается преимущество волновых алгоритмов решения треугольных систем?

Темы заданий для самостоятельной работы

- Исследовать ускорение параллельных алгоритмов LU-факторизации, использующих 1D столбцовую и 2D декомпозиции.
- Построить параллельный алгоритм решения треугольных систем вида ***Ux=b*** и исследовать его ускорение.
- Написать MPI-программу LU-факторизации и исследовать ее масштабируемость. Сравнить результаты вычислительных экспериментов с теоретическим анализом
- Написать MPI-программу решения треугольной системы. Сравнить результаты с теоретическим анализом.

Основная литература

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

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