<u>Тема №3:</u> Параллельных алгоритмов для задач линейной алгебры Вопрос:

1. Схемы алгоритмов задач линейной алгебры

2. Алгоритмы умножения матрицы на матрицу и их реализация на структурах типа: кольцевая, 2D (решетка), 3D (куб)

Упражнения и задания к теме №3

1. Схемы алгоритмов задач линейной алгебры

Параллельные алгоритмы используются для решения следующих задач: умножение матрицы на матрицу, задача Дирихле, решение систем линейных алгебраических уравнений (СЛАУ) методом Гаусса и методом простой итерации и другие. В простом варианте сетевого задачи (задача Дирихле) шаг сетки в пространстве вычислений одинаков и не меняется в процессе вычислений. При динамическом изменении шага сетки надо было бы решать задачу параллельного программирования, как перебалансирование вычислительного пространства между компьютерами, для выравнивания вычислительной нагрузки компьютеров.

<u>Особенности алгоритмов:</u>

1. Задачи относятся к задач, *розпаралелюються крупнозернистыми методами.*

2. Для представления алгоритмов используется SPMD-модель вычислений (*распараллеливания по данным).*

3. *Однородное распределения данных* по компьютерам - основа для хорошего баланса времени исчисления и времени, затрачиваемого на взаимодействии ветвей параллельной программы. Такое распределение используется с целью обеспечения *равенст объемов частей данных,* распределяемых и
соответствия нумерации частей данных, распределяемых *нумерации компьютеров в системе.*

4. Исходными данными рассмотренных алгоритмов является *матрицы, векторы и 2D (двумерный) пространство вычислений.*

5 В рассмотренных алгоритмах применяются такие способы однородного распределения данных: *горизонтальными полосами, вертикальными полосами и циклическими горизонтальными полосами.*

При распределении горизонтальными полосами матрица, вектор или *2D* пространство "разрезается" на полосы по строкам. пусть *M* - количество строк матрицы, количество элементов вектора или количество строк узлов *2D* пространства, *P* - количест виртуальных компьютеров в системе, $C1 = M / P$ - целая часть от деления, $C2 = M / P$ - дробная часть от деления. Данные разрезаются на *P* полос. первые ($P-C2$) полос имеют по *C1* строк, а другие *C2* полосы имеют по $C1 + 1$ строк.

Полосы данных распределяются по компьютерам следующим образом. Первая полоса помещается в компьютер с номером 0, вторая полоса - в компьютер 1, и т.д. Такое распределение полос по компьютерам учитывается в параллельном алгоритме. Распределение вертикальными полосами аналогичен предыдущему, только в распределении участвуют столбцы матрицы или столбцы узлов *2* D пространства. При распределении циклическими горизонтальными полосами данные разрезаются на количество полос значительно больше, от количества компьютеров. И чаще всего полоса состоит из одной строки. Первая полоса загружается в компьютер 0, вторая - в компьютер 1, и т.д., затем, *P-первых* полоса снова в компьютер 0, *P-а* полоса в компьютер 1, и т.д.

Однако, только однородность распределения данных еще недостаточно для эффективного выполнения алгоритма. Эффективность алгоритмов зависит и от способа распределения данных. Различными способами представления данных ведет, соответственно, и к различной организации алгоритмов, обрабатывающих эти данные.

Точные значения эффективности конкретного параллельного алгоритма могут быть определены на конкретной вычислительной системе на некотором наборе данных. То есть, эффективность параллельных алгоритмов зависит, во-первых, от *вычислительной системы,* на которой выполняется задача, а, во-вторых, от *структуры самих алгоритмов.* Она определяется как отношение времени реализации параллельного алгоритма задачи до времени реализации последовательного алгоритма этой же задачи. Эффективность можно измерять и соотношением между временем, затраченным на обмен данными между процессами, и общим временем вычислений. Заметим, что эффективность алгоритмов, использующих глобальный обмен данными, снижается с увеличением количества параллельных веток. То есть с увеличением количества компьютеров в системе, скорость выполнения глобальной операции обмена будет падать. К таким задачам можно отнести, например, задачу решения СЛАУ итерационными методами. Эффективность алгоритмов, в которых обмен данными осуществляется только локально, будет неизменной с увеличением количества параллельных веток.

2. Алгоритмы умножения матрицы на матрицу и их реализация на структурах типа: кольцевая, 2D (решетка), 3D (куб)

Умножение матрицы на вектор и матрицы на матрицу являются базовыми макрооперации для многих задач линейной алгебры, например итерационных методов решения систем линейных уравнений и т.п. Поэтому приведены алгоритмы здесь можно рассматривать как фрагменты в алгоритмах этих методов. Рассмотрим три алгоритма умножения матрицы на матрицу. Разнообразие вариантов алгоритмов возникает из-за разнообразия вычислительных систем и разнообразия размеров задач. рассматриваются и *различные варианты загрузки данных* в систему: загрузка данных через один компьютер; и загрузки данных непосредственно каждым компьютером с дисковой памяти. Если загрузка данных осуществляется через один компьютер, то данные считываются этим компьютером с дисковой памяти, разрезаются на части, которые рассылаются другим компьютерам. Но данные могут быть подготовлены и заранее, то есть заранее разрезанные по частям и каждая часть записана на диск в виде отдельного файла со своим именем; затем каждый компьютер непосредственно считывает с диска, предназначенный для него файл.

Алгоритм 1- Перемножение матрицы на матрицу на кольцевой структуре

Заданы две выходные матрицы А и В. вычисляется произведение **С = А** x **В,** где **А -** матрица $n_1 \times n_2$, и **В -** матрица $n_2 \times n_3$. матрица результатов **С** имеет размер $n_1 \times n_3$. Выходные матрицы предварительно разрезанные на полосы, полосы записаны на дисковую память отдельными файлами со своими именами и доступны всем компьютерам. Матрица результатов возвращается в нулевое процесс.

Реализация алгоритма выполняется на кольце с p_1 компьютеров. Матрицы разрезанные как показано на рис. 7.1: матрица **А** разрезанная на p_1 горизонтальных полос, матрица **В** разрезанная на p_1 вертикальных полос, и матрица результата **С** разрезанная на p_1 полосы. Здесь предполагается, что в память каждого компьютера загружается и может находиться только одна полоса матрицы **А** и одна полоса матрицы **В.**

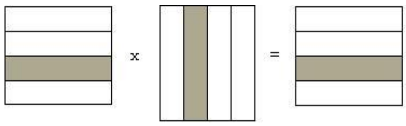


Рис. 3.1 *Разрезания данных для параллельного алгоритма произведения двух матриц при исчислении на кольце компьютеров. Выделенные полосы расположены в одном компьютере.*

Так как по условию в компьютерах находится по одной полосе матриц, то полосы матрицы **В (** или полосы матрицы **А)** необходимо "прокрутить" по кольцу компьютеров мимо полосы матрицы **А** (матрицы **В).** Каждый сдвиг полос вдоль кольца и соответствующая операция умножения приведена на рис.3.2 в виде отдельного шага. На каждом из таких шагов исчисляется только часть полосы. процесс i вычисляет на j-м шаге произведение i-й горизонтальной полосы матрицы **А** j-й вертикальной полосы матрицы **В,** произведение полученный в подматрицы (i, j) матрицы **С.**

Вычисление происходит в такой последовательности.

1. Каждый компьютер считывает с дисковой памяти соответствующую ему полосу матрицы А. Нулевая полоса должна считываться нулевым компьютером, первая полоса - первым компьютером и т.д., последняя полоса - считывается последним компьютером. На рис. 3.2 полосы матрицы **А** и **В** пронумерованы.

2. Каждый компьютер считывает с дисковой памяти соответствующую ему полосу матрицы **В.** В данном случае нулевая полоса должна считываться нулевым компьютером, первая полоса - первым компьютером и т.д., последняя полоса - считывается последним компьютером.

3. Вычислительный шаг 1. Каждый процесс вычисляет одну подматрицу произведения. Вертикальные полосы матрицы **В** сдвигаются вдоль кольца компьютеров.

4. Вычислительный шаг 2. Каждый процесс вычисляет одну подматрицу произведения. Вертикальные полосы матрицы **B** сдвигаются вдоль кольца компьютеров. И т.д.

5. Вычислительный шаг p1-1. Каждый процесс вычисляет одну подматрицу произведения. Вертикальные полосы матрицы **B** сдвигаются вдоль кольца компьютеров.

6. Вычислительный шаг p1. Каждый процесс вычисляет одну подматрицу произведения. Вертикальные полосы матрицы **B** сдвигаются вдоль кольца компьютеров.

7. Матрица **C** собирается в нулевом компьютере.

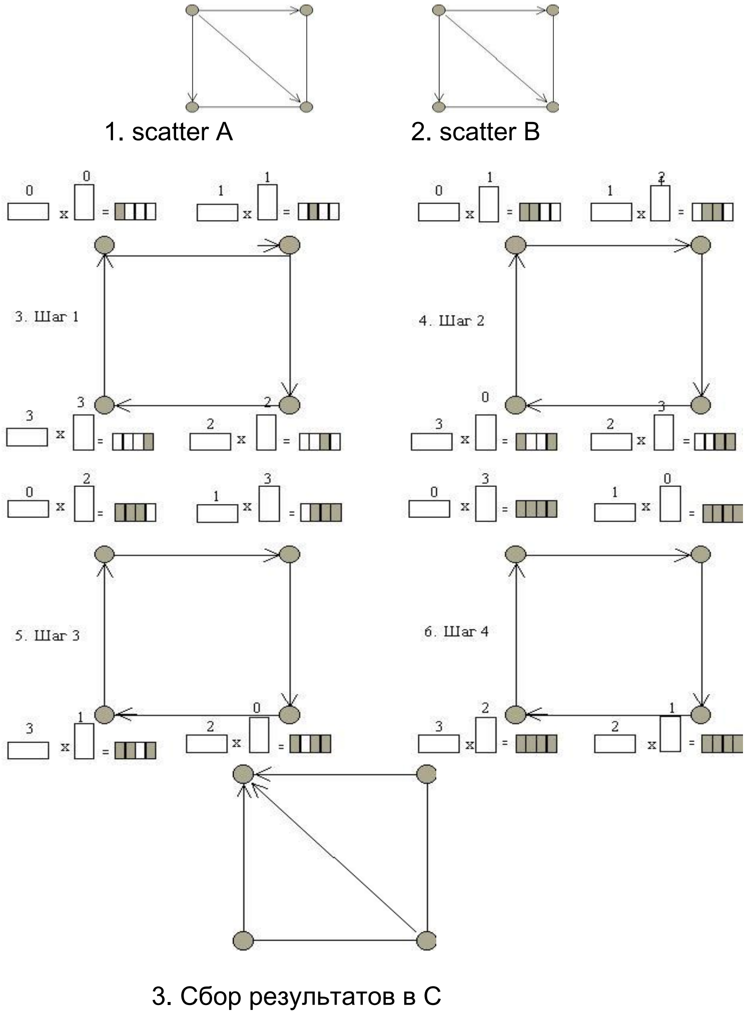


Рис. 3.2 *Стадии вычислений произведения матриц в кольце компьютеров.*

Если "прокручивать" вертикальные полосы матрицы **B,** то матрица **C** будет распределена горизонтальными полосами, а если "прокручивать" горизонтальные полосы матрицы **A,** то матрица **C** будет распределена вертикальными полосами.

Алгоритм характерен тем, что после каждого шага вычислений осуществляется обмен данными. пусть t_u, t_s, и t_p время операций, соответственно, умножение, сложение и пересылка одного числа в соседней компьютер. Тогда суммарное время операций умножений равно:

$$U = (n_1 * n_2) * (n_3 * n_2) * t_u,$$

суммарное время операций добавлений равна:

$$S = (n_1 * n_2) * (n_3 * (n_2\ 1)) * t_s,$$

суммарное время операций пересылок данных по всем компьютерах равна:

$$P = (n_3 * n_2) * (p_1 - 1) * t_p.$$

Общее время вычислений определим как:

$$T = (U + S + P) / p_1$$

Отношение времени "вычислений без обменов" к общему времени вычислений является величина:

$$K = (U + S) / (U + S + P).$$

Если время передачи данных большой по сравнению со временем вычислений, или каналы передачи медленные, то эффективность алгоритма будет не высока. Здесь не учитывается время начальной загрузки и выгрузки данных в память системы. В полосах матриц может быть разное количество строк, а разница в количестве строк между полосами - 1. При больших матрицах этим можно пренебречь.

При достаточных ресурсах памяти в системе лучше использовать алгоритм, в котором минимизированы обмены между компьютерами в процессе вычислений. Это достигается за счет дублирования некоторых данных в памяти компьютеров. В следующих двух алгоритмах используется этот подход.

алгоритм 2 - Перемножения матрицы на матрицу на2D решетке

вычисляется произведение *C = A x B,* где A - матрица n_1 **x** n_2**, и B - матрица** n_2 **x** n_3.

матрица результатов C имеет размер n_1 **x** n_3**. Выходные матрицы сначала доступны на нулевом процессе, и матрица** результатов возвращается в нулевое процесс.

Параллельное выполнение алгоритма осуществляется на двумерной (2D) решетке компьютеров размером p_1 **x** p_2**. Матрицы разрезанные, как показано на рис. 3.3: матрица A разрезанная на** p_1
горизонтальных полос, матрица B разрезанная на p_2 **вертикальных полос, и матрица результата C**
разрезанная на p_1 **x** p_2 **подматрицы (или субматрици).**

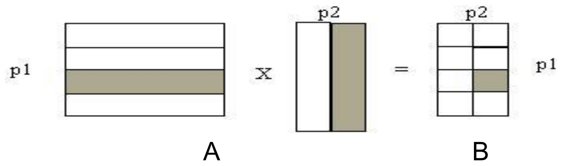


Рис. 3.3 *Разрезания данных для параллельного алгоритма произведения двух матриц при исчислении на 2D решетке компьютеров. Выделенные данные расположены в одном компьютере.*

Каждый компьютер (*i, j)* **вычисляет произведение** *i-* **и горизонтальной полосы матрицы A и** *j-* **и вертикальной полосы матрицы B, про получается в подматрицы (** *i, j)* **матрицы C.**

Последовательность стадий вычисления приведена на рис.3.4:

*1. Матрица **A** распределяется по горизонтальным полосам вдоль координаты (x, 0).*

*2. Матрица **B** распределяется по вертикальным полосам вдоль координаты (0, y).*

*3. Полосы **A** распространяются в измерении y.*

*4. Полосы **B** распространяются в измерении x.*

5. Каждый процесс вычисляет одну подматрицу произведения.

*7. Матрица **C** собирается с (x, y) плоскости.*

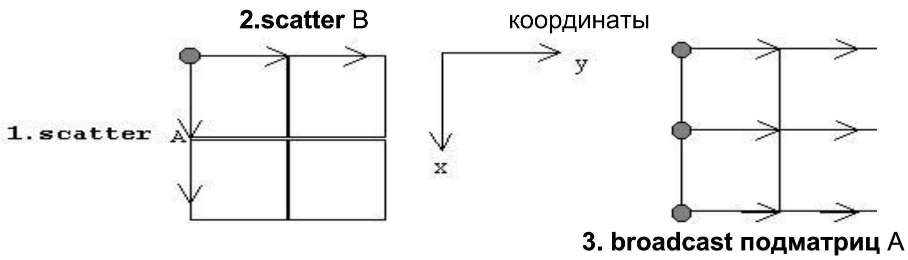
Осуществлять пересылку между компьютерами во время вычислений не нужно, потому что все полосы матрицы A пересекаютс со всеми полосами матрицы B в памяти компьютеров системы.

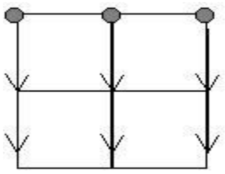
Этот алгоритм эффективнее предыдущего, так как непродуктивное время пересылок данных осуществляется только при загрузке данных в память компьютеров и при их выгрузке, и обмены данными в процессе вычислений отсутствуют. Поскольку время обменов равной нулю, а время загрузки и выгрузки здесь не учитывается, то общее время вычислений равна:

$$T = (U + S) / (p_1 * p_2)$$

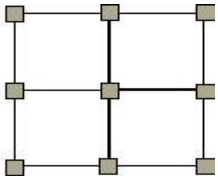
А отношение времени "вычислений без обменов" к общему времени вычислений является величина:

$$K = (U + S) / (U + S) = 1.$$

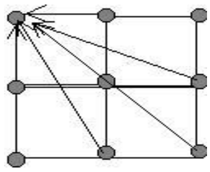




4. broadcast подматриц B



5. вычисления произведений (подматриц в C)



6. *сбор результатов в* C

Рис. 3.4 *Стадии вычисления произведения матриц в 2D параллельном алгоритме.*

алгоритм 3 - Перемножения матрицы на пространственной сетке компьютеров

Для больших матриц, при исчислении произведений может быть уменьшен путем применением алгоритма, осуществляет вычисления на 3-мерной (пространственной) сетке компьютеров.

В приведенном ниже алгоритме отображаются основные данные объемом $n_1 \times n_2 + n_2 \times n_3 + n_1 \times n_3$ на объемную сетку компьютеров размером $p_1 \times p_2 \times p_3$. Матрицы разрезанные, как показано на рис. 3.5 Матрица **A** разрезанная на $p_1 \times p_2$ субматрици, матрица ***B*** разрезанная на $p_2 \times p_3$ субматрици, и матрица **C** разрезанная на $p_1 \times p_3$ суб компьютер (*i, j, k)* вычисляет произведение субматрици (*i, j)* матрицы **A** и субматрици (*j, k)* матрицы **B.** Субматриця (*i, k)* матрицы **C** получается суммированием промежуточных результатов, вычисленных в компьютерах (*i, j, k), j = 0, ..., p2-1.*

Последовательность стадий вычисления приведена на рис. 3.6.

1. *Субматрици **A** распределяются в (x, y, 0) плоскости.*
2. *Субматрици **B** распределяются в (0, y, z) плоскости.*
3. *Субматрици **A** распространяются в измерении z.*
4. *Субматрици **B** распространяются в измерении x.*
5. *Каждый процесс вычисляет одну субматрицю.*
6. *Промежуточные результаты редуцируется в измерении y.*
7. *Матрица **C** собирается с (x, 0, z) плоскости.*

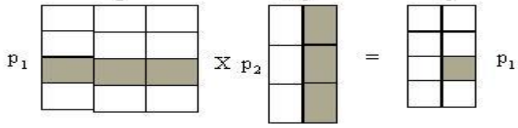
Алгоритм на предыдущий, но дополнительно разрезаются еще полосы матриц, и эти разрезанные полосы распределяются в третьем измерении *y.* В данном случае в каждом компьютере будут перемножуватися только части векторов строк матрицы **A** и части столбцов матрицы ***B.*** В результате будет только частичная сумма для каждого элемента результирующей матрицы **C.** Операция суммирования вдоль координаты *y* этих полученных частичных сумм для результирующих элементов и завершает вычисления матрицы ***C.***

Общее время вычислений в этом алгоритме равен:

$$T = (U + S) / (p_1 * p_2 * p_3)$$

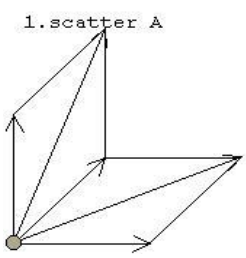
А отношение времени "вычислений без обменов" к общему времени вычислений является величина:

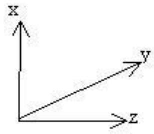
$$K = (U + S) / (U + S) = 1.$$

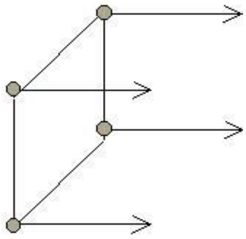


A B C

Рис. 3.5 *Разрезания данных для параллельного алгоритма произведения двух матриц при исчислении на пространственной сетке компьютеров.*

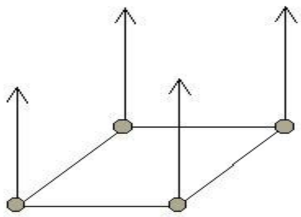


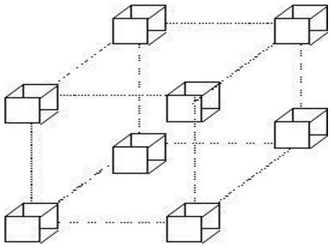




2. scatter B **3.** broadcast подматриц

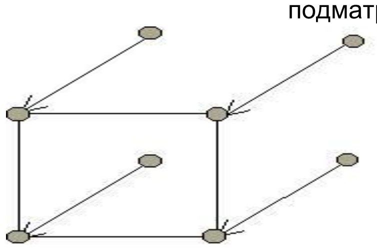
с A

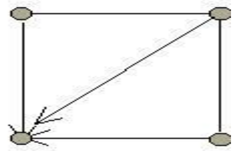




4. broadcast подматриц с B **5.** вычисление произведений

подматриц в C)





6. reduce (суммирования) произведений **7.** gather C (сбор

результатов)

Рис. 7.6. *Стадии вычислений в 3D параллельном алгоритме произведения матриц.*

<u>Упражнения и задания к теме №3</u>

1. Разработайте параллельный алгоритм вычисления величины $\sum_{i=1}^{N} BAC^{*}_{kk}$, Где А и В - одномерные массивы.

2. данные матрицы **А** и **В.** Разработайте алгоритм вычисления матрицы **С = А * В** - **В ° С.**

3. Данная матрица **А** и векторы *a* и *b*. Разработайте алгоритм вычисления матрицы **С =** *a* - **А *** *b.*

4. Возможна разное количество полос, на которые делятся матрицы **А** и **В** при исчислении произведения двух матриц на кольцевой структуре?

5. Приведите преимущества и недостатки организации перемножения двух матриц на структурах, которые описаны в данной теме?

Приведите преимущества и недостатки организации начальной загрузки при умножении двух матриц на структурах, которые описаны в данной теме?