Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Образовательный комплекс

Введение в методы параллельного программирования

Раздел 7.

Параллельные методы умножения матрицы на вектор

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

Содержание

- Постановка задачи
- Способы распределения данных
- Последовательный алгоритм
- Алгоритм 1 – ленточная схема, разделение матрицы по строкам
- Алгоритм 2 – ленточная схема, разделение матрицы по столбцам
- Алгоритм 3 – блочная схема
- Заключение

Введение

- Матрицы и матричные операции широко используются в разных областях приложений науки и техники
- Матричные вычисления требуют выполнения вычислительно-трудоемких расчетов
- Матричные операции предоставляют прекрасную возможность для демонстрации многих приемов и методов параллельного программирования

Матричные операции представляют собой классическую область применения параллельных вычислений

Постановка задачи

Умножение матрицы на вектор

$$c = A \cdot b$$

или

$$\begin{pmatrix} c_0 \\ c_1 \\ c_{m-1} \end{pmatrix} = \begin{pmatrix} a_{0,0}, & a_{0,1}, & \ldots, & a_{0,n-1} \\ & & \ldots & \\ a_{m-1,0}, & a_{m-1,1}, & \ldots, & a_{m-1,n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_{n-1} \end{pmatrix}$$

↳ Задача умножения матрицы на вектор может быть сведена к выполнению ***m*** независимых операций умножения строк матрицы ***A*** на вектор ***b***

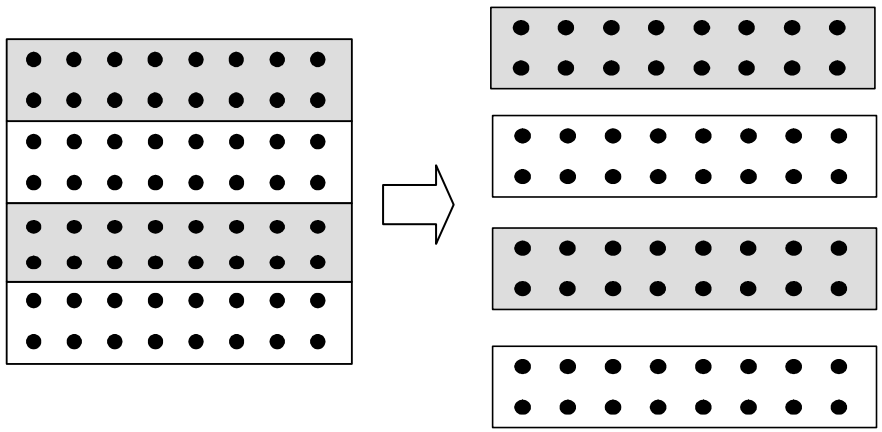
$$c_i = (a_i, b) = \sum_{j=1}^{n} a_{ij} b_j, \quad 0 \le i < m$$

В основу организации параллельных вычислений может быть положен ***принцип распараллеливания по данным***

Способы распределения данных: *ленточная схема*

Непрерывное (последовательное) распределение

горизонтальные полосы



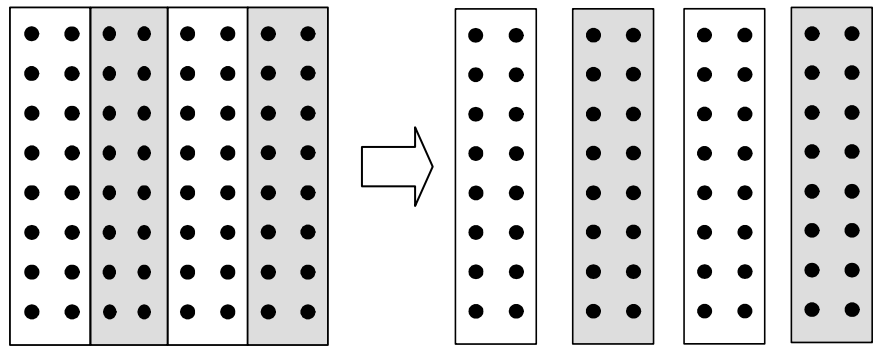
$$A = (A_0, A_1, ..., A_{p-1})^T,$$

$$A_i = (a_{i_0}, a_{i_1}, ..., a_{i_{k-1}}),$$

$$i_j = ik + j, 0 \le j < k, k = m / p$$

(a_i, $0 \le i < m$, – *строки матрицы* A)

вертикальные полосы



$$A = (A_0, A_1, ..., A_{p-1}),$$

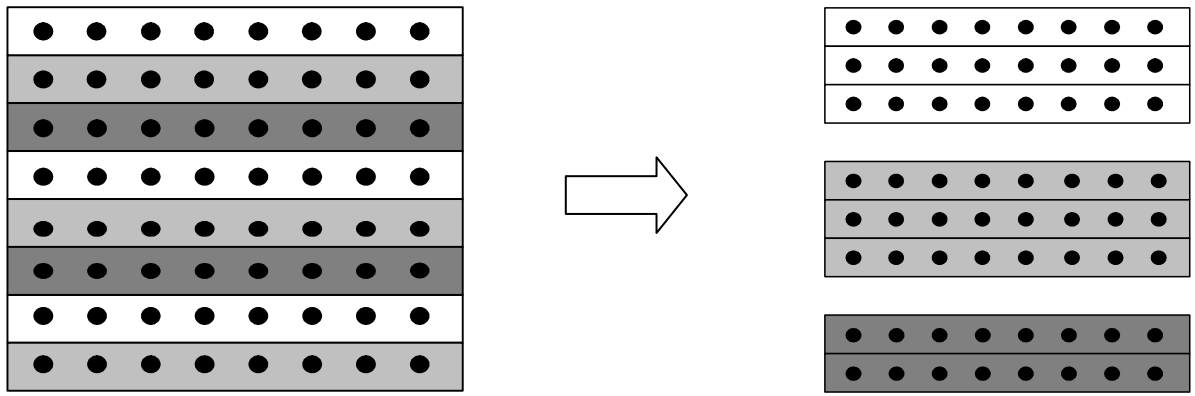
$$A_i = (\alpha_{i_0}, \alpha_{i_1}, ..., \alpha_{i_{k-1}}) \ ,$$

$$i_j = il + j, 0 \le j < l, l = n / p$$

(α_i, $0 \le i < m$, – *столбцы матрицы* A)

Способы распределения данных: *ленточная схема*

Чередующееся (цикличное) горизонтальное разбиение

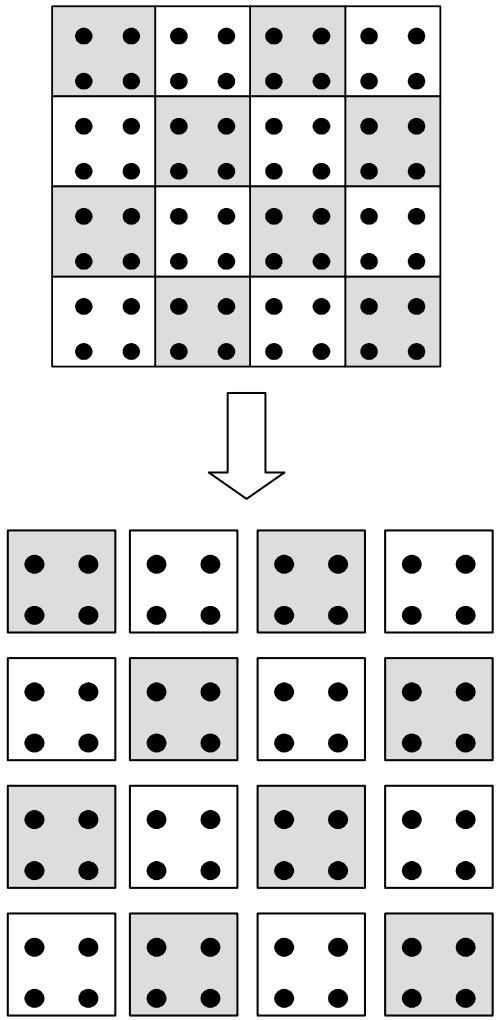


$$A = (A_0, A_2, ..., A_{p-1})^T,$$

$$A_i = (a_{i_0}, a_{i_1}, ..., a_{i_{k-1}}),$$

$$i_j = i + jp, 0 \le j < k, k = m / p$$

Способы распределения данных: *блочная схема*



$$A = \begin{pmatrix} A_{00} & A_{02} & \dots A_{0q-1} \\ & \dots & \\ A_{s-11} & A_{s-12} & \dots A_{s-1q-1} \end{pmatrix},$$

$$A_{ij} = \begin{pmatrix} a_{i_0 j_0} & a_{i_0 j_1} & \dots a_{i_0 j_{l-1}} \\ & \dots & \\ a_{i_{k-1} j_0} & a_{i_{k-1} j_1} & a_{i_{k-1} j_{l-1}} \end{pmatrix},$$

$$i_v = ik + v, 0 \le v < k, k = m / s$$

$$j_u = jl + u, 0 \le u \le l, l = n / q$$

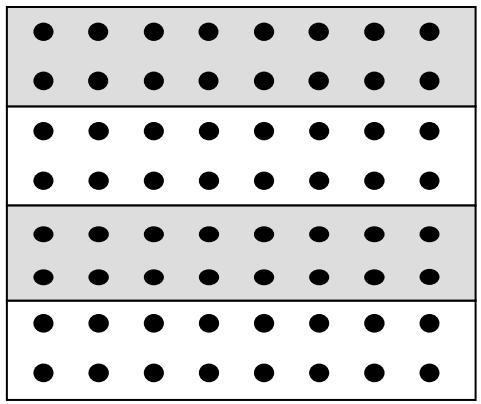
Последовательный алгоритм

```
// Алгоритм 7.1
// Последовательный алгоритм умножения матрицы на вектор
for ( i = 0; i < m; i++ ) {
  c[i] = 0;
  for ( j = 0; j < n; j++ ) {
    c[i] += A[i][j]*b[j]
  }
}
```

- Для выполнения матрично-векторного умножения необходимо выполнить ***m*** операций вычисления скалярного произведения
- Трудоемкость вычислений имеет порядок *O(mn)*.

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Распределение данных** – ленточная схема (разбиение матрицы по строкам)

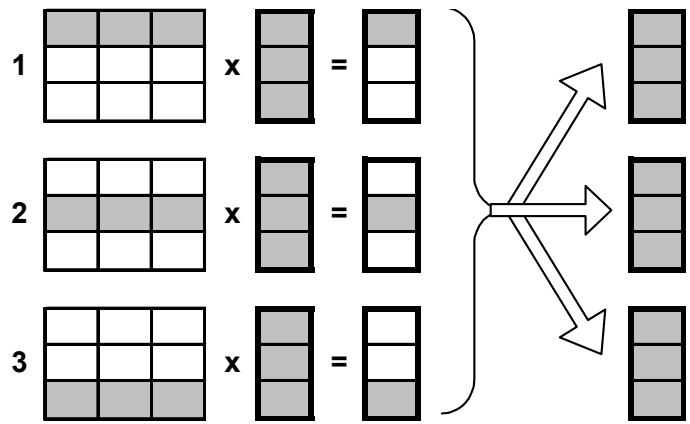


- **Базовая подзадача** - операция скалярного умножения одной строки матрицы на вектор

$$c_i = (a_i, b) = \sum_{j=1}^{n} a_{ij} b_j , \quad 0 \le i < m$$

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

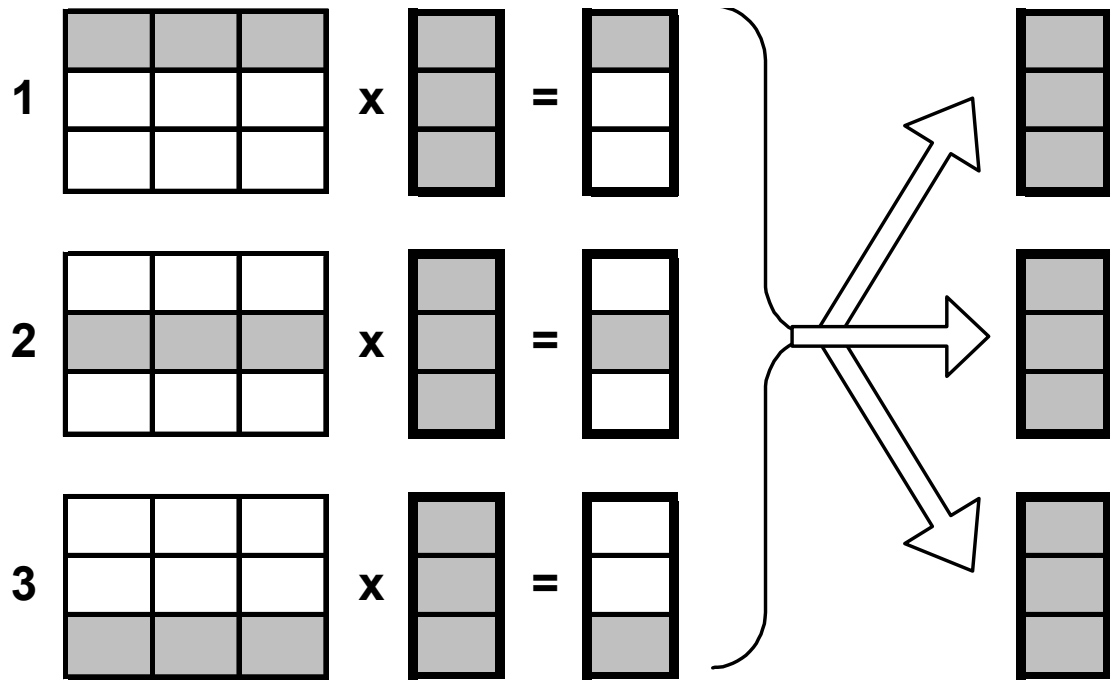
- **Выделение информационных зависимостей**
  - Базовая подзадача для выполнения вычисления должна содержать строку матрицы ***A*** и копию вектора ***b***. После завершения вычислений каждая базовая подзадача будет содержать один из элементов вектора результата ***c***



  - Для объединения результатов расчетов и получения полного вектора ***c*** на каждом из процессоров вычислительной системы необходимо выполнить операцию обобщенного сбора данных

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- Схема информационного взаимодействия



Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Масштабирование и распределение подзадач по процессорам**
  - Если число процессоров ***p*** меньше числа базовых подзадач ***m (p<m)***, базовые подзадачи могут быть укрупнены с тем, чтобы каждый процессор выполнял несколько операций умножения строк матрицы ***A*** и вектора ***b***. В этом случае, по окончании вычислений каждая базовая подзадача будет содержать набор элементов результирующего вектора ***c***
  - Распределение подзадач между процессорами вычислительной системы может быть выполнено с учетом возможности эффективного выполнения операции обобщенного сбора данных

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

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

$$S_p = \frac{n^2}{n^2 / p} = p \qquad E_p = \frac{n^2}{p \cdot (n^2 / p)} = 1$$

Разработанный способ параллельных вычислений позволяет достичь идеальных показателей ускорения и эффективности

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

❑ Анализ эффективности (уточненные оценки)

- Время выполнения параллельного алгоритма, связанное непосредственно с вычислениями, составляет

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

- Длительность выполнения операции сбора данных при использовании модели Хокни может быть определена при помощи следующего выражения:

$$T_p(comm) = \sum_{i=1}^{\lceil \log_2 p \rceil} (\alpha + 2^{i-1} w \lceil n/p \rceil / \beta) = \alpha \lceil \log_2 p \rceil + w \lceil n/p \rceil (2^{\lceil \log_2 p \rceil} - 1)/\beta$$

Общее время выполнения параллельного алгоритма составляет

$$T_p = (n/p) \cdot (2n-1) \cdot \tau + \alpha \cdot \log_2 p + w(n/p)(p-1)/\beta$$

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Программная реализация…**
  - Главная функция программы
    - Реализует логику работу программы,
    - Последовательно вызывает необходимые подпрограммы.

Программа

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Программная реализация …**
  - Функция **ProcessInitialization:**
    - Определяет исходные матрицу A и вектор b
    - Значения элементов матрицы A и вектора b определяются при помощи функции RandomDataInitialization.

Программа

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Программная реализация**
  - Функция **DataDistribution**:
    - Рассылает вектор *b*,
    - Распределяет строки матрицы *A* между процессорами вычислительной системы.

Программа

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

- **Программная реализация**
  - Функция **ParallelResultCaculation**:
    - Выполняет умножение строк матрицы A, которые были распределены на данный процесс, на вектор,
    - Получает блок результирующего вектора *c* .

Программа

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

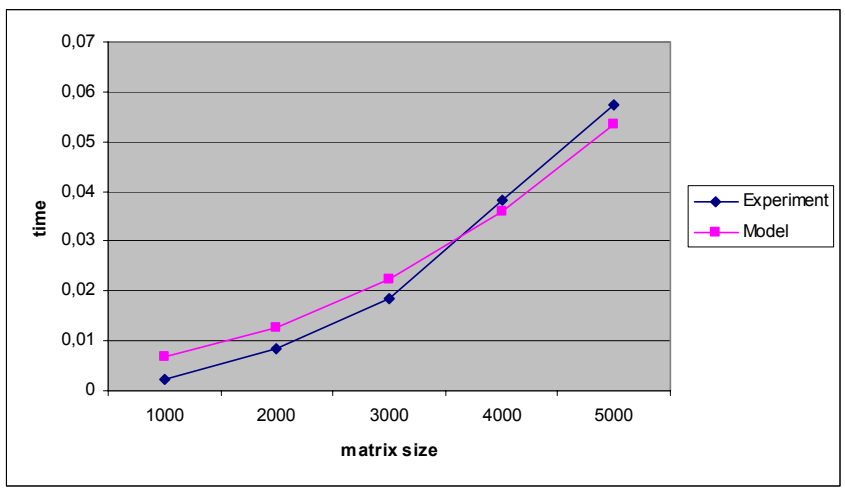
- **Программная реализация**
  - Функция **ResultReplication:**
    - Объединяет блоки результирующего вектора c, которые были получены на разных процессах,
    - Рассылает результирующий вектор на все процессы.

Программа

Алгоритм 1: ленточная схема (разбиение матрицы по строкам)…

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

Размер матрицы	2 процессора		4 процессора		6 процессора	
	Модель	Эксперимент	Модель	Эксперимент	Модель	Эксперимент
1000	0,0069	0,0021	0,0108	0,0017	0,0152	0,0175
2000	0,0132	0,0084	0,0140	0,0047	0,0169	0,0032
3000	0,0235	0,0185	0,0193	0,0097	0,0196	0,0059
4000	0,0379	0,0381	0,0265	0,0188	0,0233	0,0244
5000	0,0565	0,0574	0,0359	0,0314	0,0280	0,0150

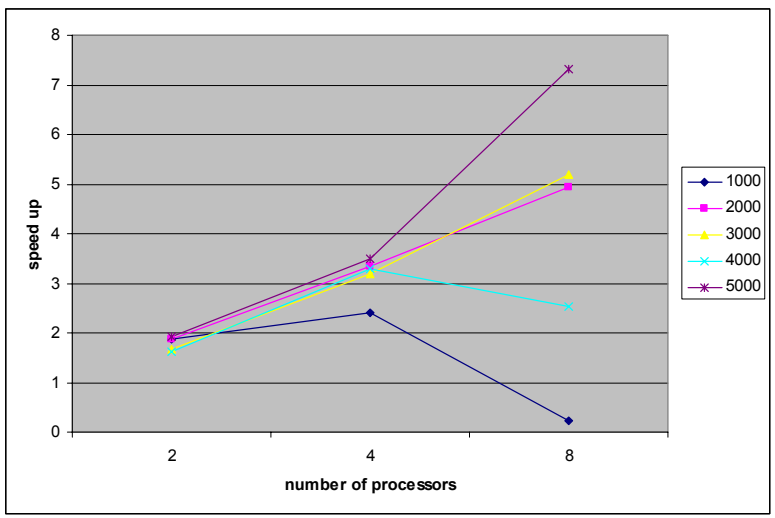


Алгоритм 1: ленточная схема (разбиение матрицы по строкам)

❑ Результаты вычислительных экспериментов

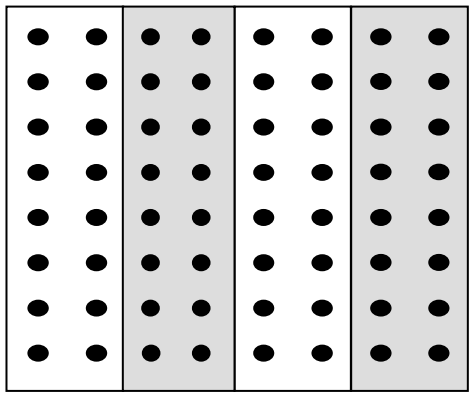
– Ускорение вычислений

Размер матрицы	Последовательный алгоритм	Параллельный алгоритм					
		2 процесса		4 процесса		8 процесса	
		Время	Ускорение	Время	Ускорение	Время	Ускорение
1000	0,0041	0,0021	1,8798	0,0017	2,4089	0,0175	0,2333
2000	0,016	0,0084	1,8843	0,0047	3,3388	0,0032	4,9443
3000	0,031	0,0185	1,6700	0,0097	3,1778	0,0059	5,1952
4000	0,062	0,0381	1,6263	0,0188	3,2838	0,0244	2,5329
5000	0,11	0,0574	1,9156	0,0314	3,4993	0,0150	7,3216

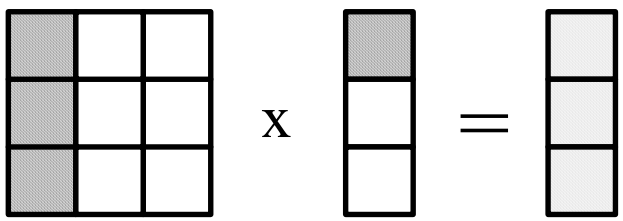


Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

- **Распределение данных** – ленточная схема (разбиение матрицы по столбцам)



- **Базовая подзадача** - операция умножения столбца матрицы *A* на один из элементов вектора *b*

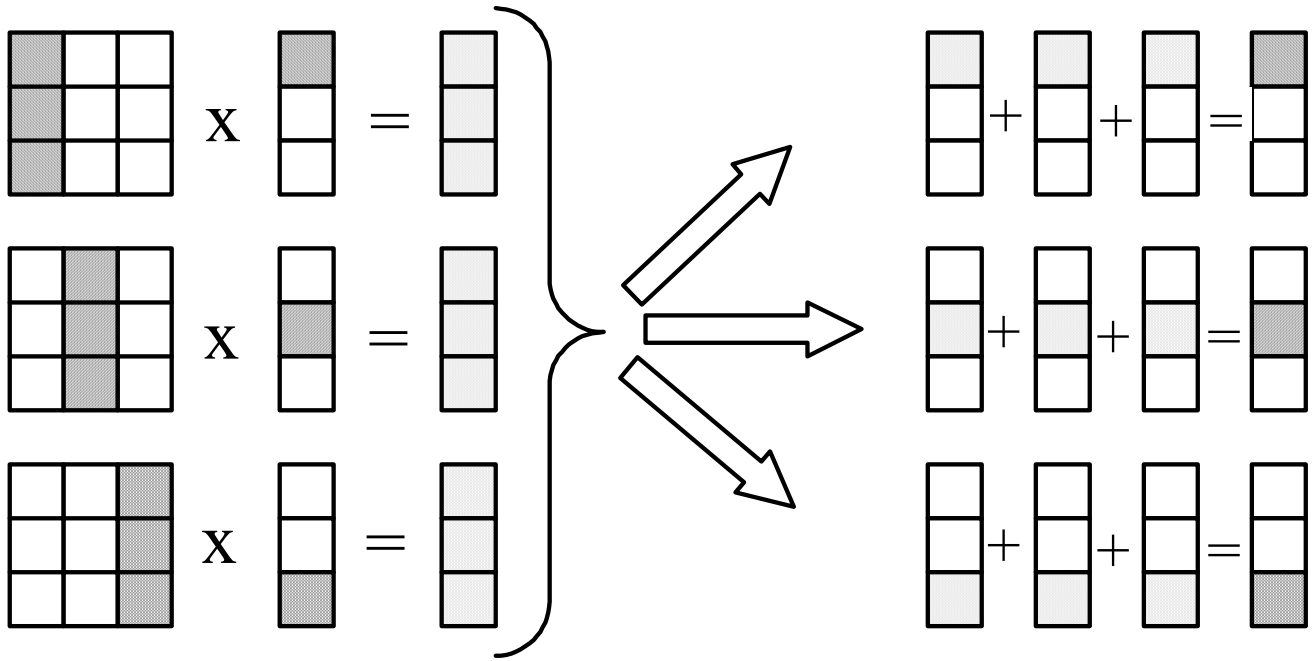


Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

- **Выделение информационных зависимостей**
  - Базовая подзадача для выполнения вычисления должна содержать должна содержать ***i***-й столбец матрицы ***A*** и ***i***-е элементы b_i и c_i векторов ***b*** и ***c***
  - Каждая базовая задача ***i*** выполняет умножение своего столбца матрицы ***A*** на элемент b_i, в итоге в каждой подзадаче получается вектор ***c'(i)*** промежуточных результатов
  - Для получения элементов результирующего вектора ***c*** подзадачи должны обменяться своими промежуточными данными

Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

- Схема информационного взаимодействия



Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

- **Масштабирование и распределение подзадач по процессорам**
  - В случае, когда количество столбцов матрицы ***n*** превышает число процессоров ***p***, базовые подзадачи можно укрупнить, объединив в рамках одной подзадачи несколько соседних столбцов (в этом случае исходная матрица ***A*** разбивается на ряд вертикальных полос). В этом случае, по окончании вычислений и проведения операции обмена каждая базовая подзадача будет содержать набор элементов результирующего вектора ***c***
  - Распределение подзадач между процессорами вычислительной системы может быть выполнено с учетом возможности эффективного выполнения операции обмена элементами векторов частичных результатов

Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

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

$$S_p = \frac{n^2}{n^2 / p} = p \qquad E_p = \frac{n^2}{p \cdot (n^2 / p)} = 1$$

Разработанный способ параллельных вычислений позволяет достичь идеальных показателей ускорения и эффективности

Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

❑ Анализ эффективности (уточненные оценки)

- Время выполнения параллельного алгоритма, связанное непосредственно с вычислениями, составляет

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

- Операция обобщенной передачи данных может быть реализована двумя способами:

  • каждый процессор последовательно передает свои данные всем остальным процессорам вычислительном системы, длительность такой операции при использовании модели Хокни:

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

  • в случае, когда топология вычислительной сети может быть представлена в виде гиперкуба, операция обобщенной передачи может быть выполнена за $log_2 p$ шагов :

$$T_p^2(comm) = \lceil \log_2 p \rceil (\alpha \cdot + w(n/2)/\beta)$$

Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

- **Анализ эффективности** (уточненные оценки)

- Общее время выполнения параллельного алгоритма при использовании первого способа реализации операции обобщенной передачи данных составляет:

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

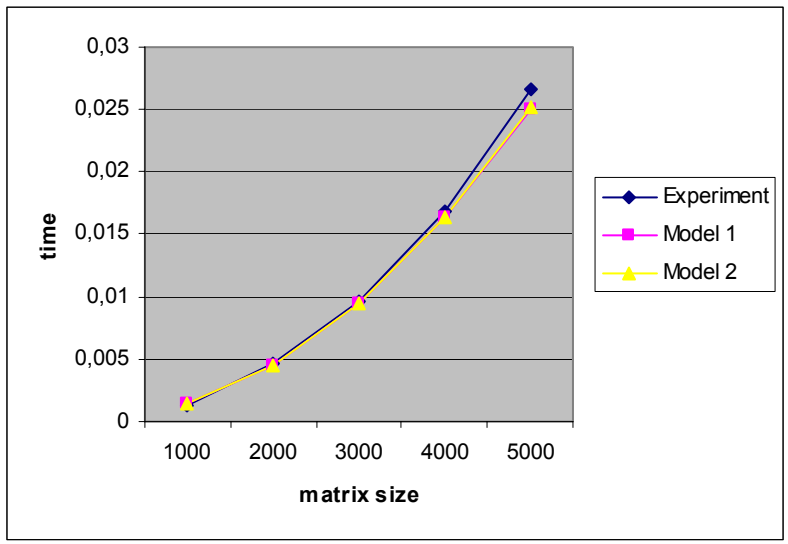
- При использовании второго способа реализации операции обобщенной передачи данных, общее время выполнения алгоритма составляет:

$$T_p^2 = [n \cdot (2 \cdot \lceil n/p \rceil - 1) + n] \cdot \tau + \lceil \log_2 p \rceil (\alpha \cdot + w(n/2)/\beta)$$

Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)…

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

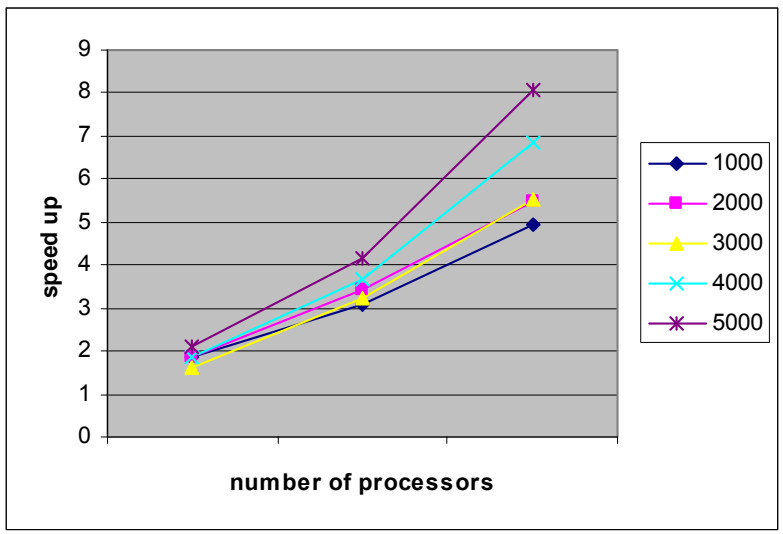
Размер матрицы	2 процесса			4 процесса			8 процесса		
	Модель 1	Модель 2	Эксперимент	Модель 1	Модель 2	Эксперимент	Модель 1	Модель 2	Эксперимент
1000	0,0021	0,0021	0,0022	0,0014	0,0013	0,0013	0,0015	0,0011	0,0008
2000	0,0080	0,0080	0,0085	0,0044	0,0044	0,0046	0,0031	0,0027	0,0029
3000	0,0177	0,0177	0,019	0,0094	0,0094	0,0095	0,0056	0,0054	0,0055
4000	0,0313	0,0313	0,0331	0,0162	0,0163	0,0168	0,0091	0,0090	0,0090
5000	0,0487	0,0487	0,0518	0,0251	0,0251	0,0265	0,0136	0,0135	0,0136



Алгоритм 2: ленточная схема (разбиение матрицы по столбцам)

- **Результаты вычислительных экспериментов**
  - Ускорение вычислений

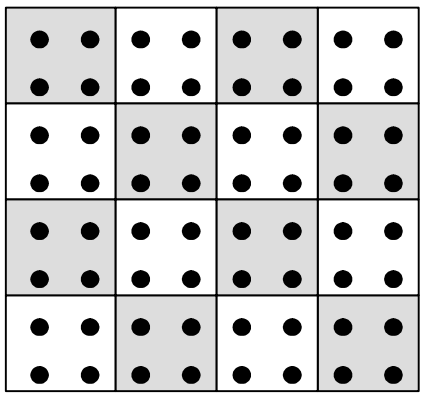
Размер матрицы	Последовательный алгоритм	2 процессора		4 процессора		8 процессора	
		Время	Ускорение	Время	Ускорение	Время	Ускорение
1000	0,0041	0,0022	1,8352	0,0132	0,3100	0,0008	4,9409
2000	0,016	0,0085	1,8799	0,0046	3,4246	0,0029	5,4682
3000	0,031	0,019	1,6315	0,0095	3,2413	0,0055	5,5456
4000	0,062	0,0331	1,8679	0,0168	3,6714	0,0090	6,8599
5000	0,11	0,0518	2,1228	0,0265	4,1361	0,0136	8,0580



Алгоритм 3: блочная схема…

- **Распределение данных** – блочная схема

  предполагается, что количество процессоров ***p=s·q*** , количество строк матрицы является кратным ***s***, а количество столбцов – кратным ***q***, то есть ***m=k·s*** и ***l=n·q***.



$$A = \begin{pmatrix} A_{00} & A_{02} & \dots A_{0q-1} \\ & \dots & \\ A_{s-11} & A_{s-12} & \dots A_{s-1q-1} \end{pmatrix} \quad A_{ij} = \begin{pmatrix} a_{i_0 j_0} & a_{i_0 j_1} & \dots a_{i_0 j_{l-1}} \\ & \dots & \\ a_{i_{k-1} j_0} & a_{i_{k-1} j_1} & a_{i_{k-1} j_{l-1}} \end{pmatrix}$$

$$i_v = ik + v,\ 0 \le v < k,\ k = m / s$$
$$j_u = jl + u,\ 0 \le u \le l,\ l = n / q$$

Алгоритм 3: блочная схема…

- **Базовая подзадача** определяется на основе вычислений, выполняемых над матричными блоками:
  - Подзадачи нумеруются индексами *(i, j)* располагаемых в подзадачах матричных блоков
  - Подзадачи выполняют умножение содержащегося в них блока матрицы ***A*** на блок вектора ***b***

$$b(i,j) = (b_0(i,j),\ldots,b_{l-1}(i,j))^T, \quad b_u(i,j) = b_{j_u},\ j_u = jl+u,\ 0 \le u < l,\ l = n/q$$

  - После перемножения блоков матрицы ***A*** и вектора ***b*** каждая подзадача *(i,j)* будет содержать вектор частичных результатов *c'(i,j)*,

$$c'_v(i,j) = \sum_{u=0}^{l-1} a_{i_v j_u} b_{j_u},\ i_v = ik+v,\ 0 \le v < k,\ k = m/s,$$
$$j_u = jl+u,\ 0 \le u \le l,\ l = n/q$$

Алгоритм 3: блочная схема…

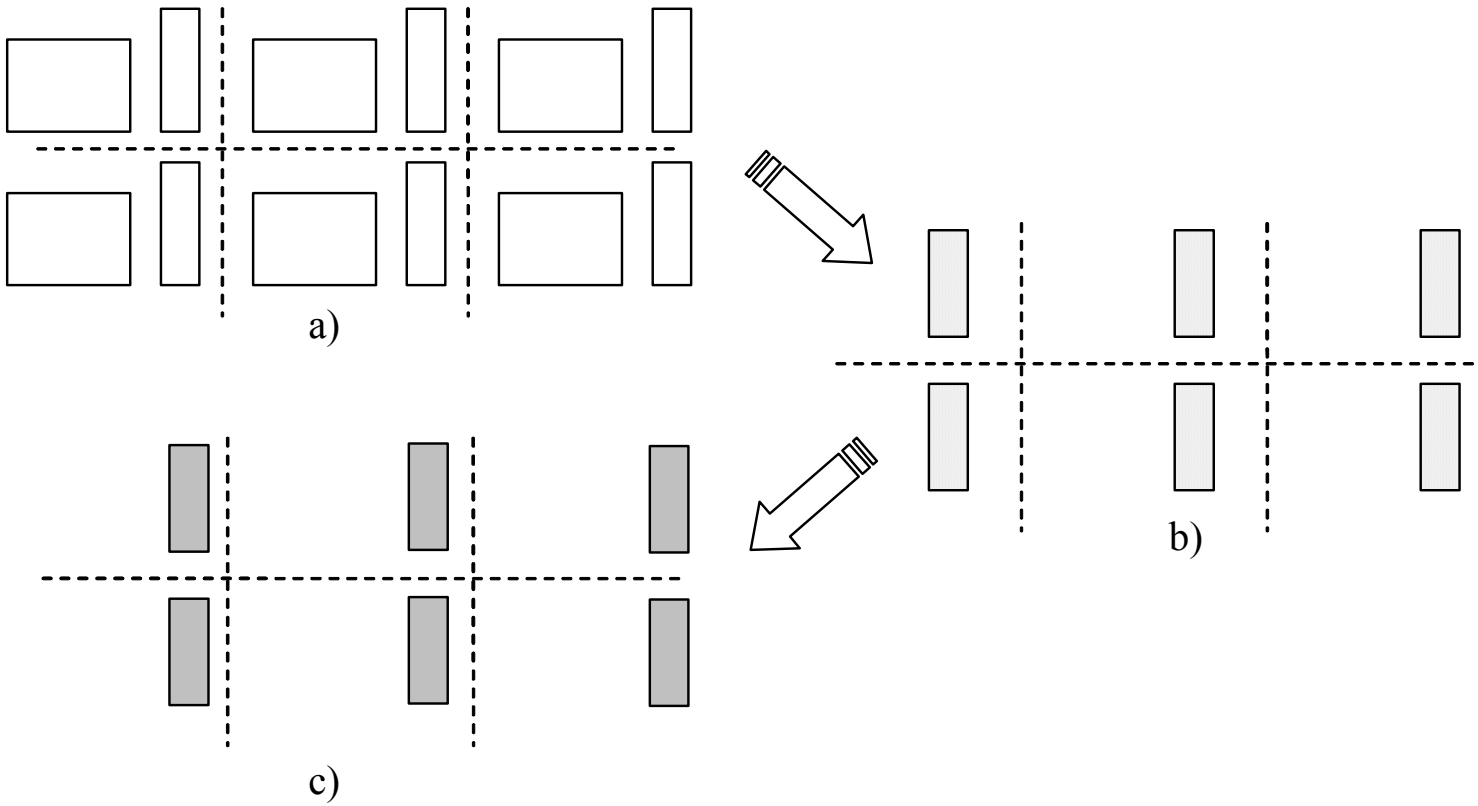
- **Выделение информационных зависимостей**
  - Поэлементное суммирование векторов частичных результатов для каждой горизонтальной полосы (*редукция*) блоков матрицы ***A*** позволяет получить результирующий вектор ***c***

$$c_\eta = \sum_{j=0}^{q-1} c'_\nu (i,j),\ 0 \le \eta < m,\ i = \eta / s,\ \nu = \eta - i \cdot s$$

  - организуем вычисления таким образом, чтобы при завершении расчетов вектор ***c*** располагался поблочно в каждой из вертикальных полос блоков матрицы ***A***
  - информационная зависимость базовых подзадач проявляется только на этапе суммирования результатов перемножения блоков матрицы ***A*** и блоков вектора ***b***

Алгоритм 3: блочная схема…

- Схема информационного взаимодействия



Алгоритм 3: блочная схема…

- **Масштабирование и распределение подзадач по процессорам**
  - Размер блоков матрицы ***A*** может быть подобран таким образом, чтобы общее количество базовых подзадач совпадало с числом процессоров ***p***, ***p=s·q***.
    - Большое количество блоков по горизонтали (***s***) приводит к возрастанию числа итераций в операции редукции результатов блочного умножения,
    - увеличение размера блочной решетки по вертикали (***q***) повышает объем передаваемых данных между процессорами.
  - При решении вопроса распределения подзадач между процессорами должна учитываться возможность эффективного выполнения операции редукции.

Алгоритм 3: блочная схема…

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

$$S_p = \frac{n^2}{n^2 / p} = p \qquad E_p = \frac{n^2}{p \cdot (n^2 / p)} = 1$$

Разработанный способ параллельных вычислений позволяет достичь идеальных показателей ускорения и эффективности

Алгоритм 3: блочная схема…

❑ Анализ эффективности (уточненные оценки)

- Общее время умножения блоков матрицы ***A*** и вектора ***b*** может быть определено как

$$T_p(calc) = \lceil n/s \rceil \cdot (2 \cdot \lceil n/q \rceil - 1) \cdot \tau$$

- Операция редукции данных может быть выполнена с использованием каскадной схемы и включает, тем самым, $log_2 q$ итераций передачи сообщений размера $w\lceil n/s \rceil$. Как результат, оценка коммуникационных затрат параллельного алгоритма при использовании модели Хокни может быть определена при помощи следующего выражения:

$$T_p(comm) = (\alpha + w \lceil n/s \rceil / \beta) \lceil \log_2 q \rceil$$

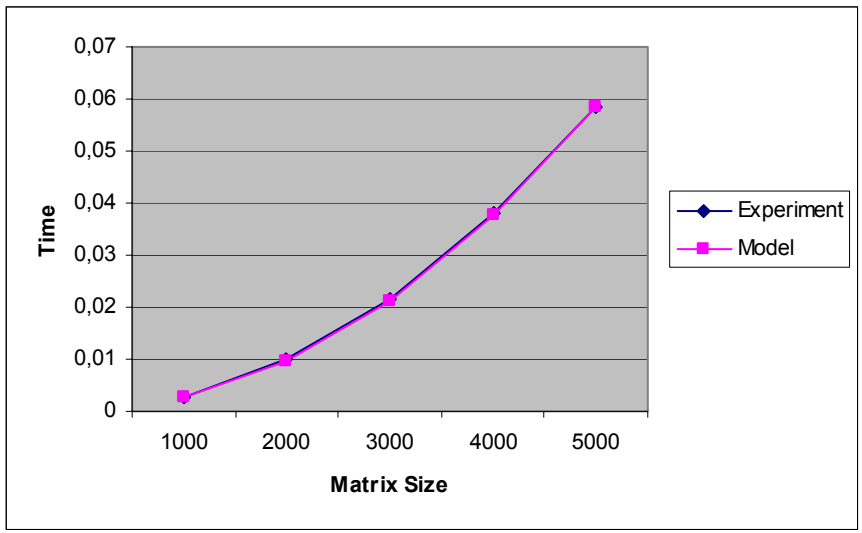
Общее время выполнения параллельного алгоритма составляет

$$T_p = \lceil n/s \rceil \cdot (2 \cdot \lceil n/q \rceil - 1) \cdot \tau + (\alpha + w \lceil n/s \rceil / \beta) \lceil \log_2 q \rceil$$

Алгоритм 3: блочная схема…

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

Размер матрицы	4 процессора		9 процессора	
	Модель	Эксперимент	Модель	Эксперимент
1000	0,0025	0,0028	0,0012	0,0010
2000	0,0095	0,0099	0,0043	0,0042
3000	0,0212	0,0214	0,0095	0,0095
4000	0,0376	0,0381	0,0168	0,0175
5000	0,0586	0,0583	0,0262	0,0263

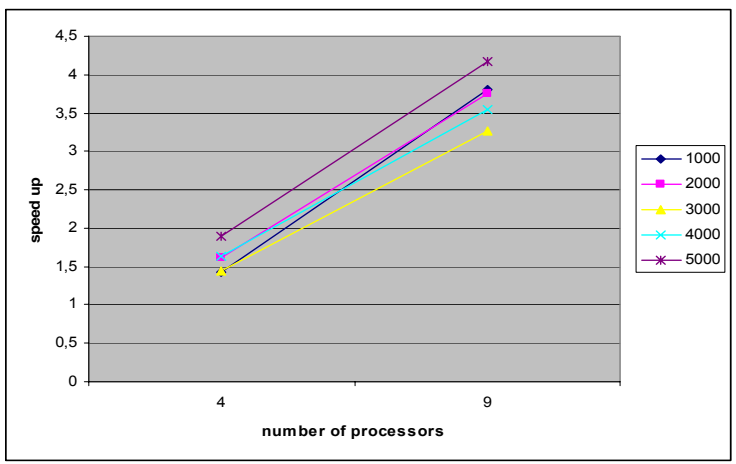


Алгоритм 3: блочная схема

❑ Результаты вычислительных экспериментов

– Ускорение вычислений

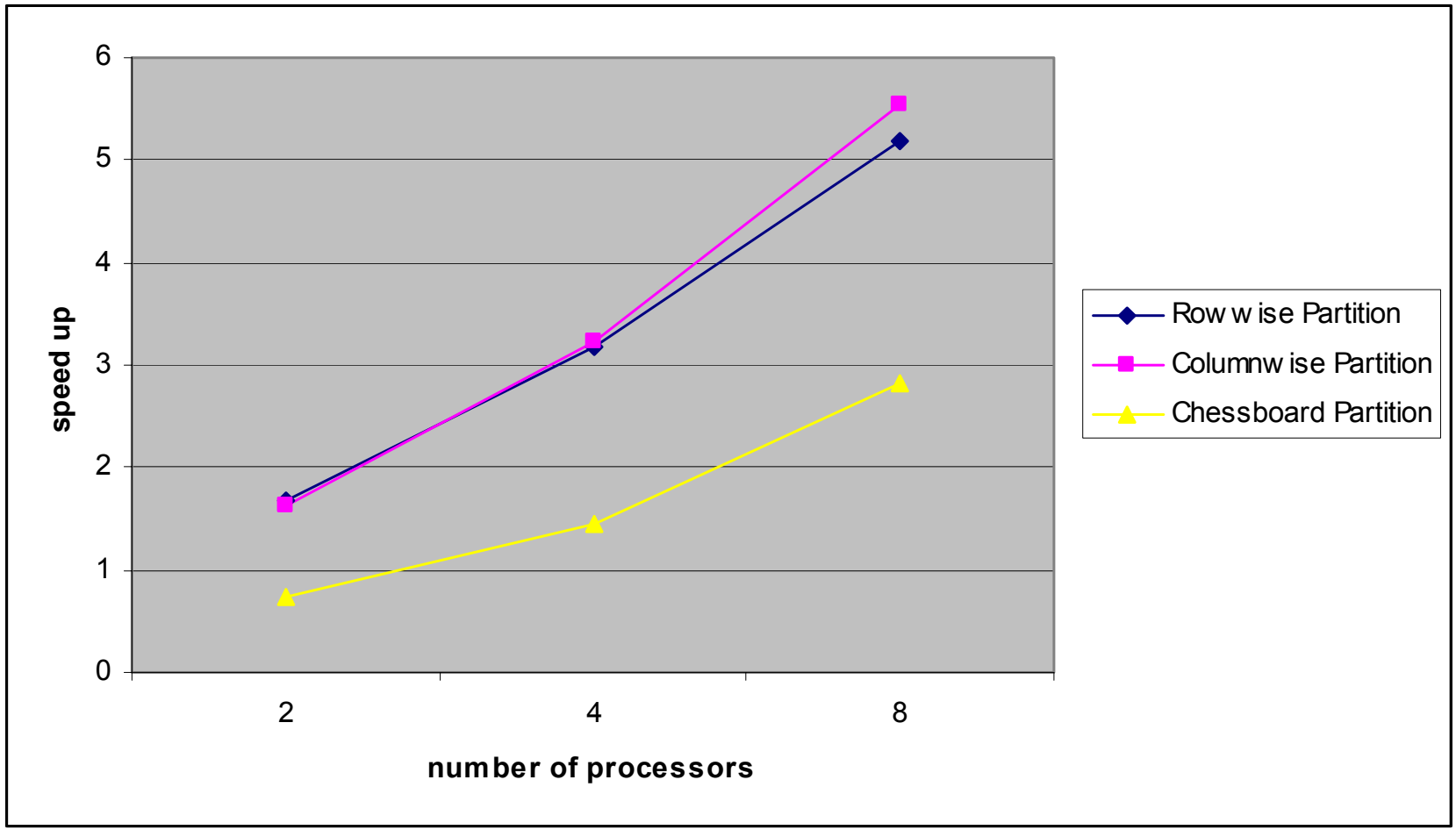
Размер матрицы	Последовательный алгоритм	Параллельный алгоритм			
		4 процессора		9 процессоров	
		Time	Speed Up	Time	Speed Up
1000	0,0041	0,0028	1,4260	0,0011	3,7998
2000	0,016	0,0099	1,6127	0,0042	3,7514
3000	0,031	0,0214	1,4441	0,0095	3,2614
4000	0,062	0,0381	1,6254	0,0175	3,5420
5000	0,11	0,0583	1,8860	0,0263	4,1755



Заключение

- Рассмотрены возможные схемы разделения данных между процессорами многопроцессорной вычислительной системы для параллельного выполнения матричных вычислений:
  - Ленточная схема (горизонтальное и вертикальное разбиение),
  - Блочная схема
- Описаны три возможных параллельных реализации одной из наиболее часто используемых матричных операций - умножения матрицы на вектор:
  - Алгоритм 1 – ленточное горизонтальное разбиение данных,
  - Алгоритм 2 – ленточное вертикальное разбиение данных,
  - Алгоритм 3 – блочное разбиение данных
- Теоретические оценки позволяют достаточно точно определить показатели эффективности параллельных вычислений

Сравнение алгоритмов



Вопросы для обсуждения

- Почему при разработке параллельных алгоритмов умножения матрицы на вектор допустимо копировать вектор на все процессоры?
- Какой из алгоритмов обладает наилучшими показателями ускорения и эффективности?
- Может ли использование циклической схемы разделения данных повлиять на время работы каждого из представленных алгоритмов?
- Какие операции передачи данных необходимы в алгоритмах умножения матрицы на вектор?

Темы заданий для самостоятельной работы

- Выполните реализацию параллельного алгоритма, основанного на ленточном разбиении матрицы на вертикальные полосы
- Выполните реализацию параллельного алгоритма, основанного на разбиении матрицы на блоки
- Постройте теоретические оценки времени работы этих алгоритмов с учетом параметров используемой вычислительной системы
- Проведите вычислительные эксперименты. Сравните результаты реальных экспериментов с полученными теоретическими оценками

Литература

- **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 матричного умножения**

Авторский коллектив

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

Гришагин В.А., доцент, к.ф.м.н.

Абросимова О.Н., ассистент (раздел 10)

Лабутин Д.Ю., ассистент (система ПараЛаб)

Курылев А.Л., ассистент (лабораторные работы 4, 5)

Сысоев А.В., ассистент (раздел 1)

Гергель А.В., аспирант (раздел 12, лабораторная работа 6)

Лабутина А.А., аспирант (разделы 7,8,9, лабораторные работы 1, 2, 3, система ПараЛаб)

Сенин А.В., аспирант (раздел 11, лабораторные работы по Microsoft Compute Cluster)

Ливерко С.В. (система ПараЛаб)

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.