Нижегородский государственный университет им. Н.И.Лобачевского

Факультет Вычислительной математики и кибернетики

Образовательный комплекс

Введение в методы параллельного программирования

Лабораторная работа 1.

Параллельные алгоритмы матрично-векторного умножения

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

Содержание

Упражнения:

- Постановка задачи
- Реализация последовательного алгоритма умножения матрицы на вектор
- Способы распределения данных
- Разработка параллельного алгоритма, основанного на разделении матрицы по строкам
- Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам
- Разработка параллельного алгоритма, основанного на блочном разделении матрицы

Упражнение 1: 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***

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

Упражнение 1: Постановка задачи

```
// Serial algorithm of matrix-vector multiplication
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)*

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Поэтапная разработка последовательного алгоритма:**
  - Задание 1 – Создание проекта **MatrixVectorMult**
  - Задание 2 – Определение размеров объектов и ввод данных
  - Задание 3 – Завершение процесса вычислений
  - Задание 4 – Реализация умножения матрицы на вектор
  - Задание 5 – Проведение вычислительных экспериментов

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 1** – Создание проекта **MatrixVectorMult**:
  - Переменные, которые будут использоваться в программе:

```
double* pMatrix;  // Initial matrix
double* pVector;  // Initial vector
double* pResult;  // Result vector
int Size;         // Sizes of initial matrix and vector
```

  - Вывод начального сообщения и ожидание нажатия любой клавиши перед завершением выполнения приложения:

```
printf ("Serial matrix-vector multiplication program\n");
getch();
```

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 2** – Определение размеров объектов и ввод данных:…
  - Для задания исходных данных реализуем функцию *ProcessInitialization*:
    - определяет размеры матрицы и вектора,
    - выделяет память для всех объектов, участвующих в умножении (*pMatrix, pVector* и *pResult*),
    - задает значения элементов исходных матрицы и вектора

```
// Function for process intialization
void ProcessInitialization (double* &pMatrix,
  double* &pVector, double* &pResult, int &Size);
```

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 2** – Определение размеров объектов и ввод данных:…
  - Определение значений элементов исходных матрицы и вектора по шаблону:

$$pMatrix = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 \end{pmatrix}, \; pVector = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 2** – Определение размеров объектов и ввод данных:…
  - Функция для простого определения значений элементов исходных матрицы и вектора:

```
// Function for simple data initialization
void DummyDataInitialization (double* pMatrix,
  double* pVector, int Size)
```

  - Функция для задания значений элементов матрицы и вектора при помощи датчика случайных чисел:

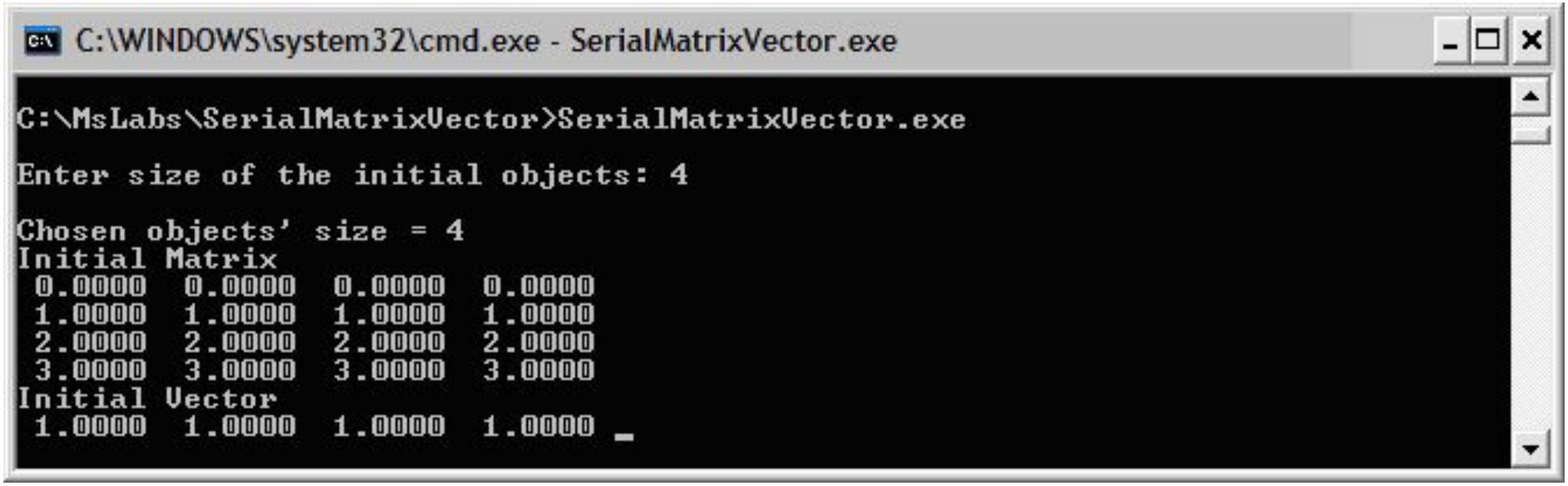
```
// Function for random initialization of object elements
void RandomDataInitialization (double* pMatrix,
  double* pVector,int Size)
```

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 2** – Определение размеров объектов и ввод данных:…
  - Для контроля правильности задания исходных данных необходимо разработать:
    - Функцию для форматированной печати матрицы *PrintMatrix* (входные параметры - матрица *pMatrix*, количество строк *RowCount* и количество столбцов *ColCount*),
    - Функцию для форматированной печати вектора *PrintVector* (входные параметры - вектор *pVector* и количество элементов в векторе *Size*)

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 2** – Определение размеров объектов и ввод данных:
  - Контроль правильности выполнения этапа задания исходных данных:



Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 3** – Завершение процесса вычислений:
  - Функция для корректного завершения процесса вычислений *ProcessTermination*:
    - Освобождает память, выделенную в ходе выполнения программы,
    - Входные параметры - матрица *pMatrix* и векторы *pVector* и *pResult*

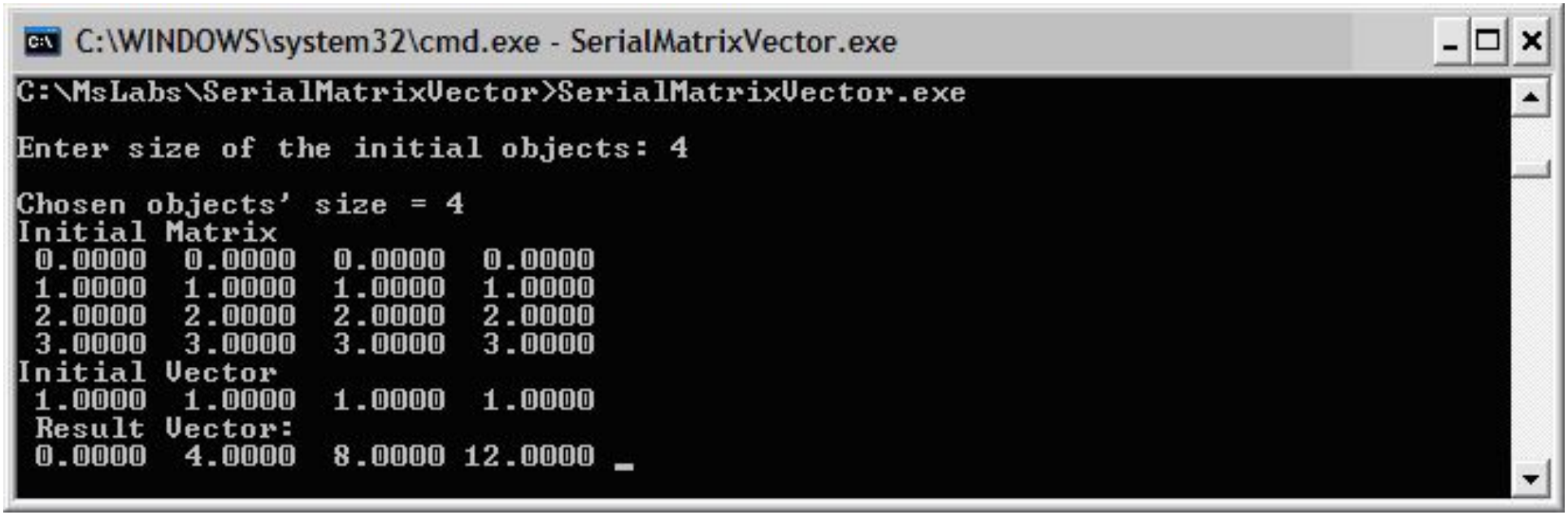
Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 4** – Реализация умножения матрицы на вектор:…
  - Функция *ResultCalculation* выполняет умножение матрицы на вектор в соответствии с алгоритмом, изложенным в упражнении 1:

```
// Function for matrix-vector multiplication
void ResultCalculation (double* pMatrix, double* pVector,
  double* pResult, int Size) {
  int i, j;  // Loop variables
  for (i=0; i<Size; i++) {
    pResult[i] = 0;
   for (j=0; j<Size; j++)
     pResult[i] += pMatrix[i*Size+j]*pVector[j];
  }
}
```

Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор…

- **Задание 4** – Реализация умножения матрицы на вектор:



Упражнение 2: Реализация последовательного алгоритма умножения матрицы на вектор

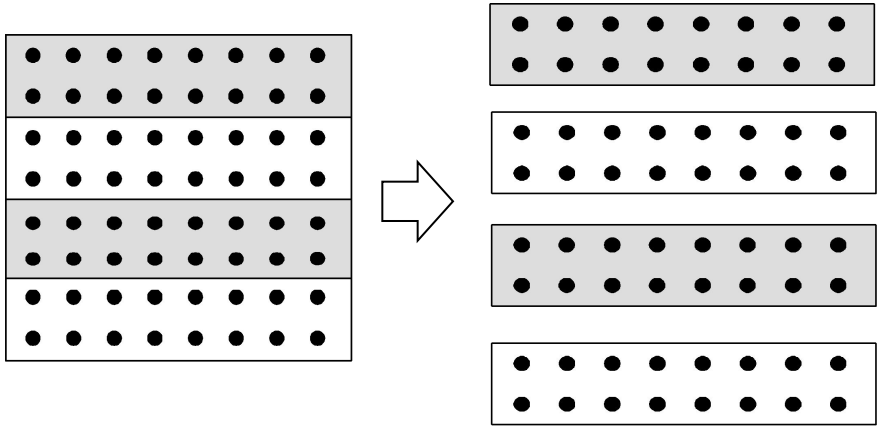
- **Задание 5** – Проведение вычислительных экспериментов:
  - Замените вызов функции *DummyDataInitialization* на *RandomDataInitialization* в функции *ProcessInitialization*,
  - Добавьте вычисление и вывод времени выполнения умножения,
  - Измерьте времена работы алгоритма умножения матрицы на вектор при различных количествах исходных данных,
  - Заполните таблицу результатов вычислений

Размер матрицы	Время выполнения алгоритма (с)
1000	
2000	
…	
10000	

Упражнение 3: Способы распределения данных…

Непрерывное (последовательное) распределение

горизонтальные полосы



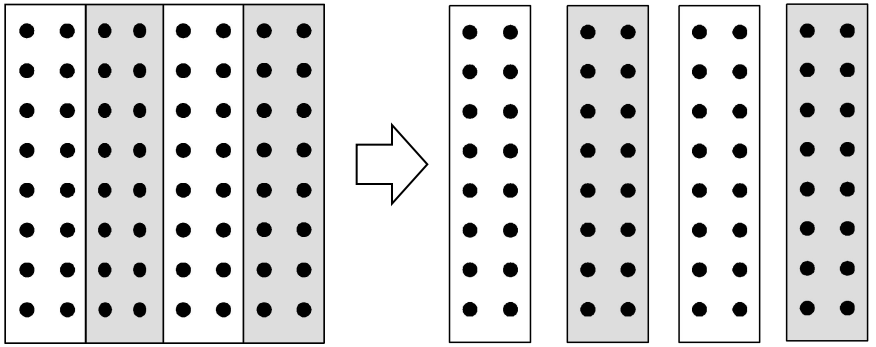
$$A = (A_0, A_1, ..., A_{p-1})^T,$$

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

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

$(a_i, 0 \le i < m, - \textit{строки матрицы } A)$

вертикальные полосы



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

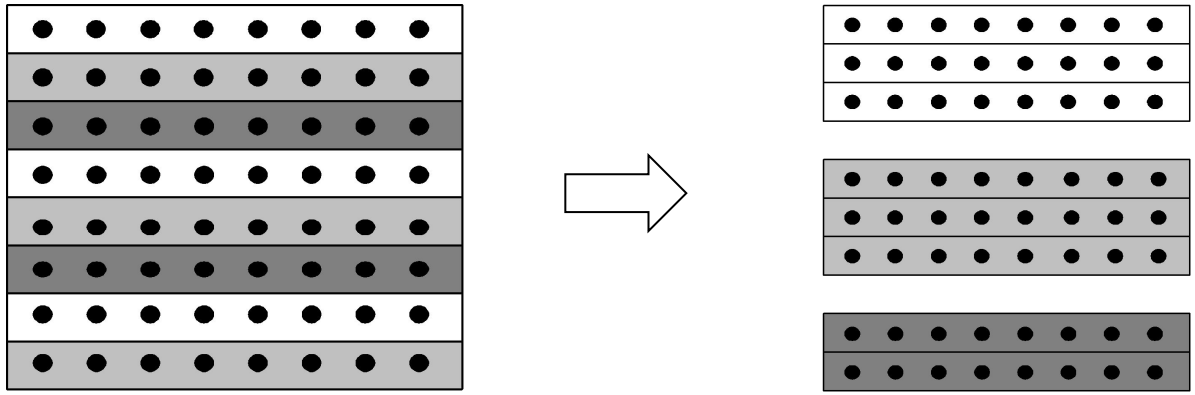
$$A_i = (\alpha_{i_0}, \alpha_{i_1}, ..., \alpha_{i_{k-1}}),$$

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

$(\alpha_i, 0 \le i < m, - \textit{столбцы матрицы } A)$

Упражнение 3: Способы распределения данных…

Чередующееся (цикличное) горизонтальное разбиение

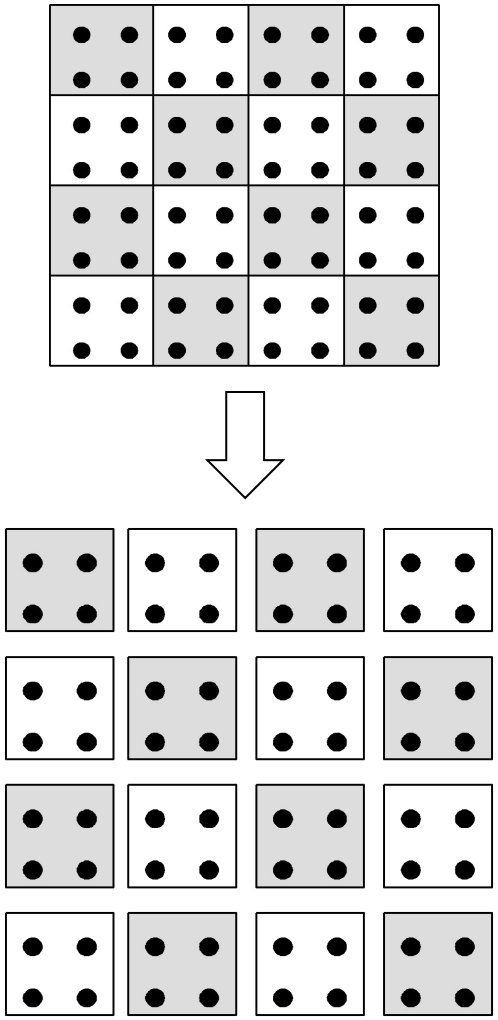


$$A = (A_0, A_2, ..., A_{p-1})^T,$$

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

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

Упражнение 3: Способы распределения данных



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

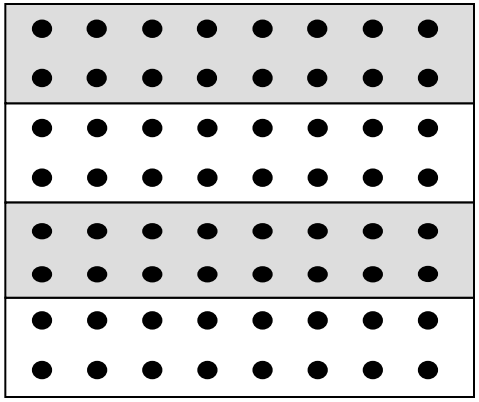
$$A_{ij} = \begin{pmatrix} a_{i_0 j_0} & a_{i_0 j_1} & \dots a_{i_0 j_{l-1}} \\ & \dots & \\ a_{i_{k-1} j_0} & a_{i_{k-1} j_1} & a_{i_{k-1} j_{l-1}} \end{pmatrix},$$

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

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

Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

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

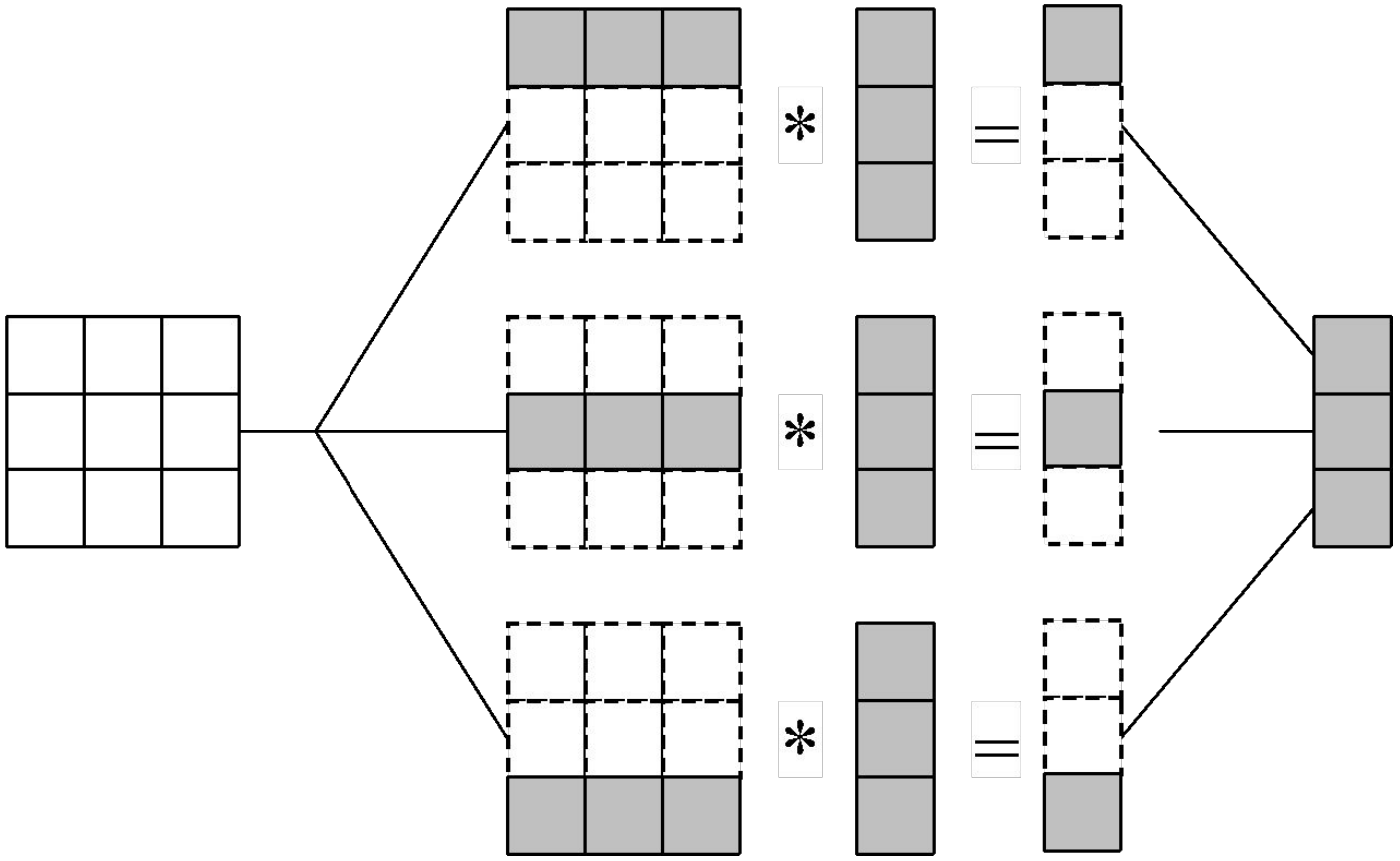


- **Базовая подзадача** - операция скалярного умножения одной строки матрицы на вектор

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

Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

- Схема параллельного выполнения:



Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

- **Программная реализация:**

```
// Function for calculating matrix-vector multiplication
void ParallelResultCalculation(double* pMatrix, double* pVector,
  double* pResult, int Size) {
  int i, j;  // Loop variables
#pragma omp parallel for private (j)
  for (i=0; i<Size; i++) {
    pResult[i] = 0;
    for (j=0; j<Size; j++)
      pResult[i] += pMatrix[i*Size+j]*pVector[j];
  }
}
```

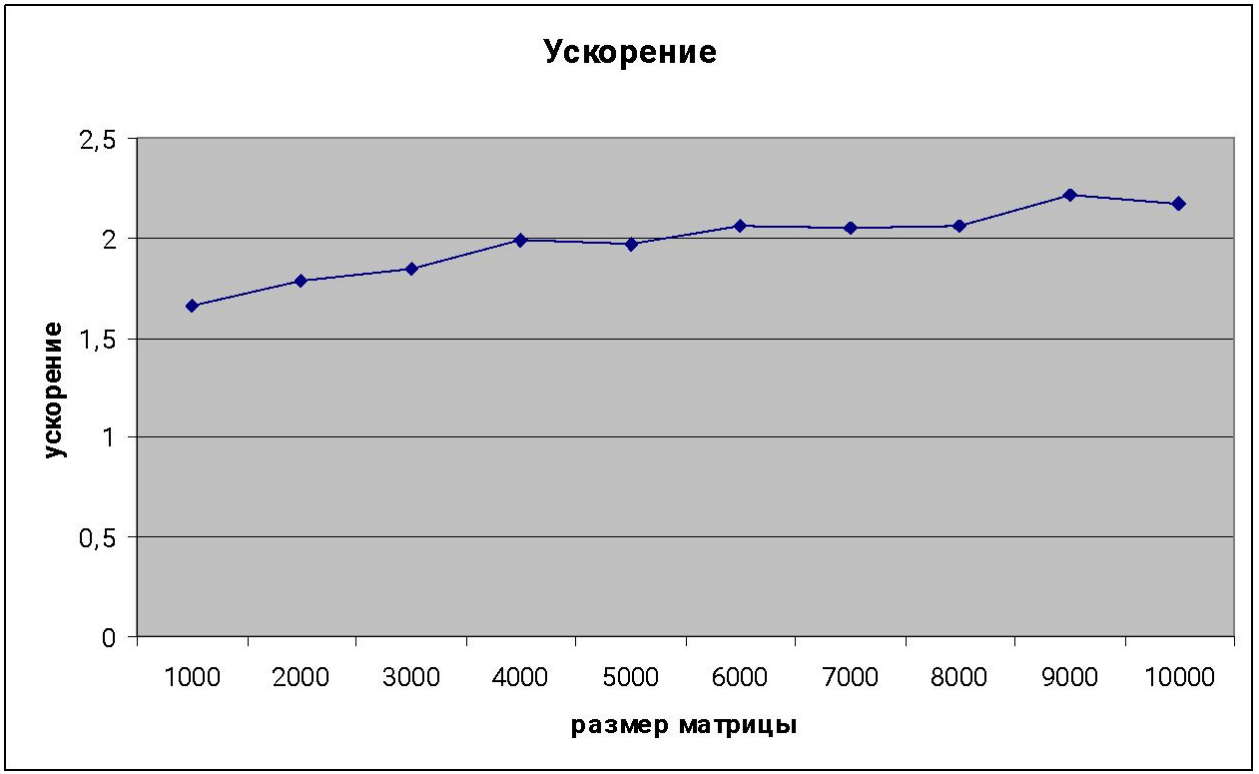
Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

- **Проведение вычислительных экспериментов:**
  - Добавьте вычисление и вывод времени выполнения параллельного алгоритма умножения матрицы на вектор,
  - Проведите вычислительные эксперименты,
  - Измерьте времена работы умножения матрицы на вектор при различных количествах исходных данных
  - Определите получаемое ускорение,
  - Заполните таблицу результатов вычислений

Размер матрицы	Время выполнения последовательного алгоритма	Время выполнения параллельного алгоритма	Ускорение
1000			
2000			
…			
10000			

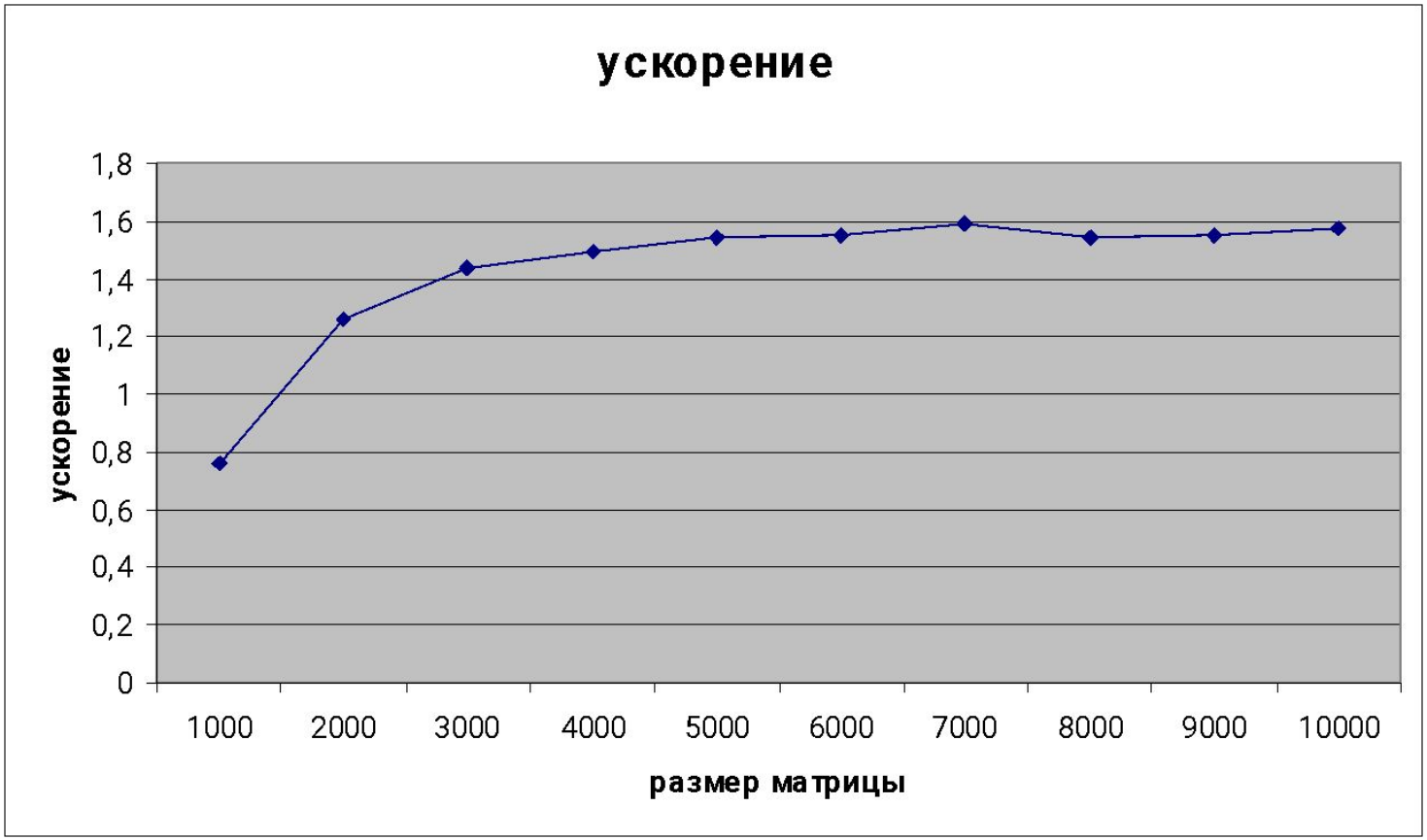
Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

- Desk-side T-Forge Mini cluster
  - AMD Opteron® 275 2.2 GHz dual core processors
  - RAM 4 GB



Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам…

- Intel Core 2 CPU 6300 1.86 GHz, 2 GB RAM

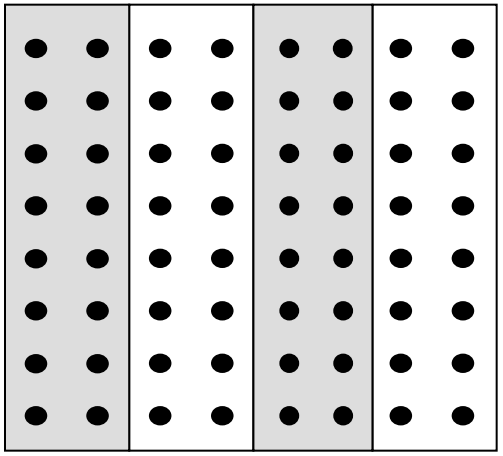


Упражнение 4: Разработка параллельного алгоритма, основанного на разделении матрицы по строкам

- Pentium D 820 2.8 GHz, 2 GB RAM

Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

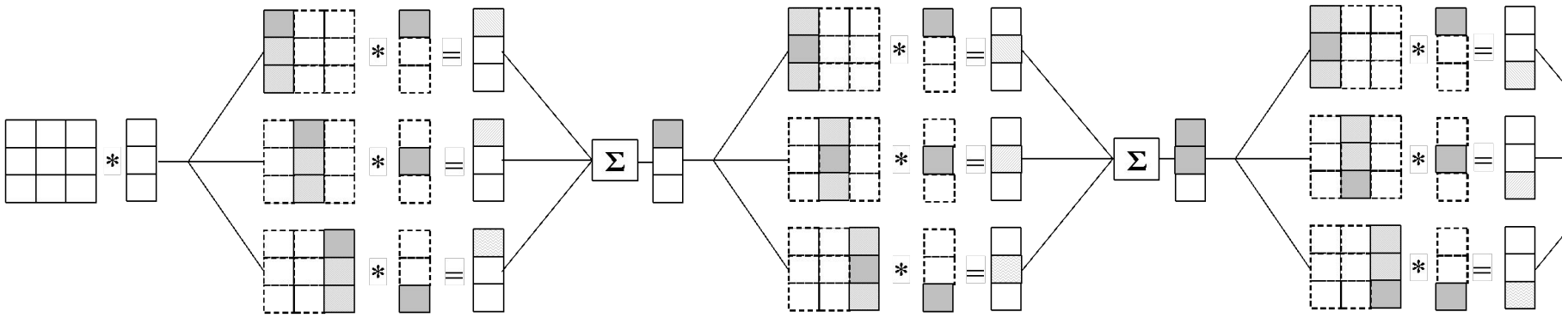
- **Распределение данных** – ленточная схема (разбиение матрицы по столбцам)



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

Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

- Схема параллельного выполнения:



Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

- **Программная реализация:**

```
// Function for calculating matrix-vector multiplication
void ParallelResultCalculation(double* pMatrix, double* pVector,
  double* pResult, int Size) {
  int i, j;  // Loop variables
  double IterGlobalSum = 0;
  for (i=0; i<Size; i++) {
    IterGlobalSum = 0;
#pragma omp parallel for reduction(+:IterGlobalSum)
    for (j=0; j<Size; j++)
      IterGlobalSum += pMatrix[i*Size+j]*pVector[j];
    pResult[i] = IterGlobalSum;
  }
}
```

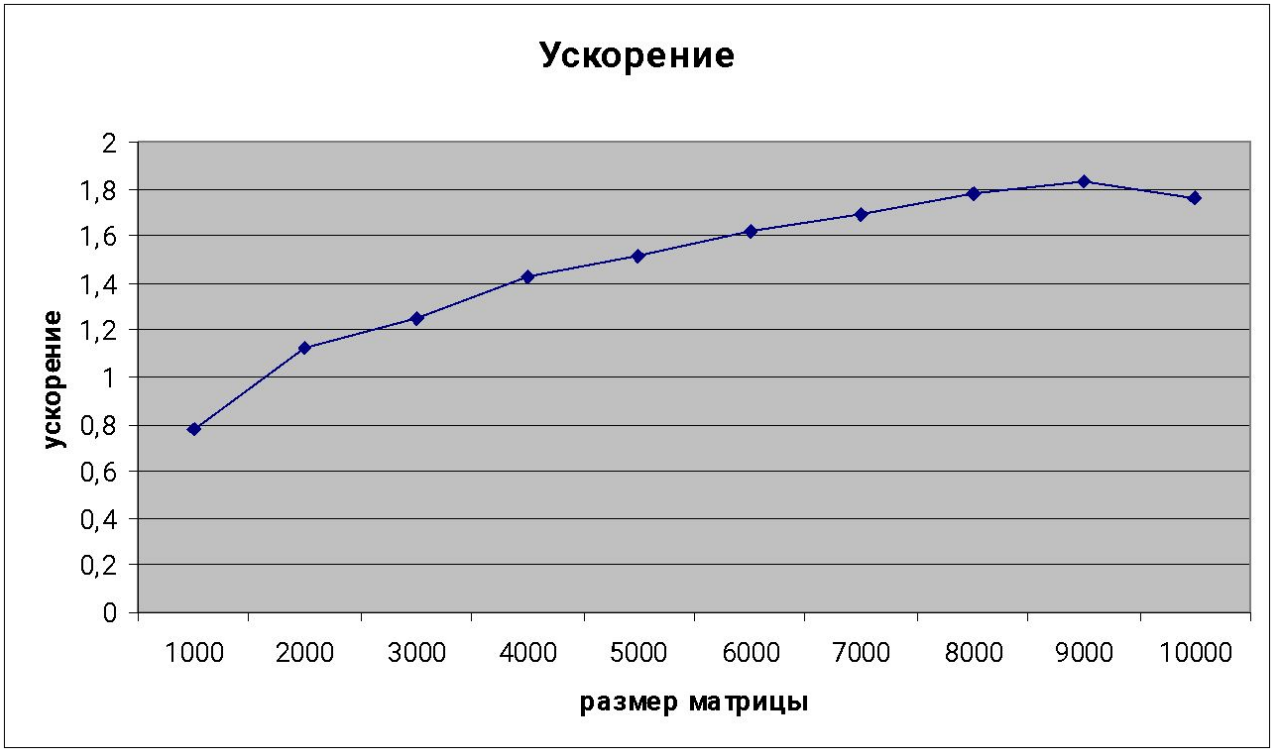
Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

- **Проведение вычислительных экспериментов:**
  - Добавьте вычисление и вывод времени выполнения параллельного алгоритма умножения матрицы на вектор,
  - Проведите вычислительные эксперименты,
  - Измерьте времена работы умножения матрицы на вектор при различных количествах исходных данных
  - Определите получаемое ускорение,
  - Заполните таблицу результатов вычислений

Размер матрицы	Время выполнения последовательного алгоритма	Время выполнения параллельного алгоритма	Ускорение
1000			
2000			
…			
10000			

Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

- Desk-side T-Forge Mini cluster
  - AMD Opteron® 275 2.2 GHz dual core processors
  - RAM 4 GB

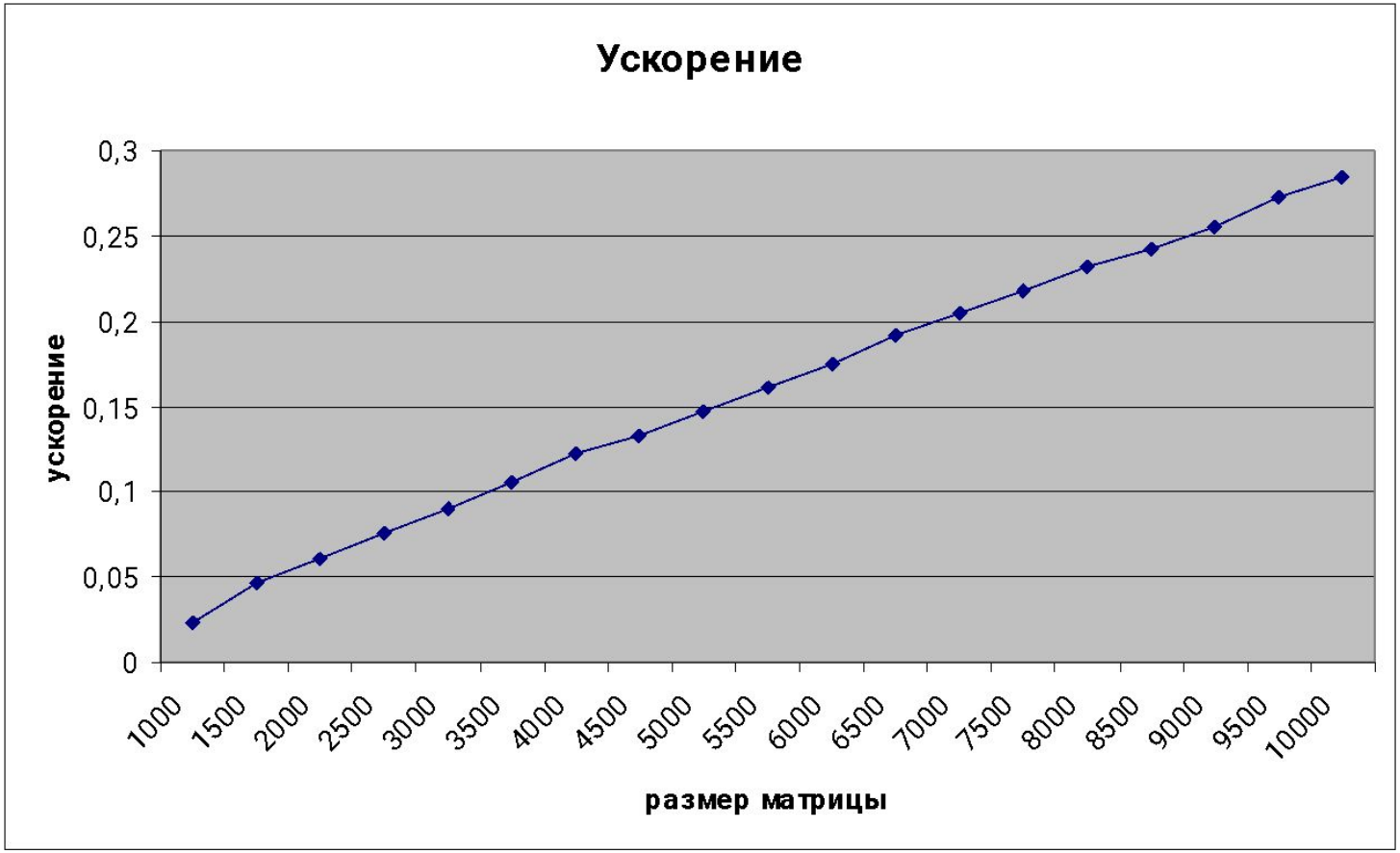


Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам…

- Intel Core 2 CPU 6300 1.86 GHz, 2 GB RAM

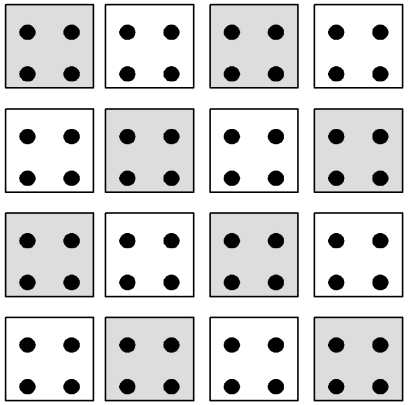
Упражнение 5: Разработка параллельного алгоритма, основанного на разделении матрицы по столбцам

- Pentium D 820 2.8 GHz, 2 GB RAM



Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

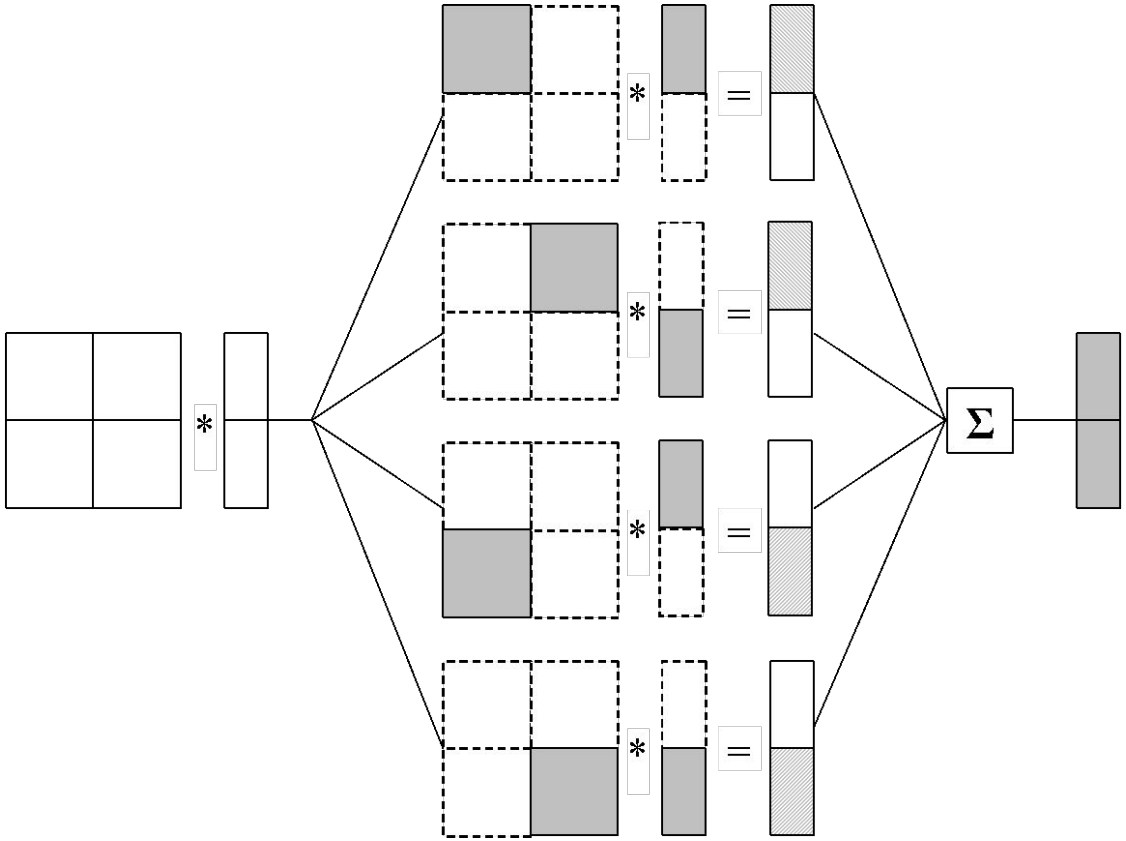
- **Распределение данных** – блочная схема



- **Базовая подзадача** – умножение блока матрицы *A* на блок вектора *b*

Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

- Схема параллельного выполнения:



Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

- **Программная реализация**
  - Code

Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

- **Проведение вычислительных экспериментов:**
  - Добавьте вычисление и вывод времени выполнения параллельного алгоритма умножения матрицы на вектор,
  - Проведите вычислительные эксперименты,
  - Измерьте времена работы умножения матрицы на вектор при различных количествах исходных данных
  - Определите получаемое ускорение,
  - Заполните таблицу результатов вычислений

Размер матрицы	Время выполнения последовательного алгоритма	Время выполнения параллельного алгоритма	Ускорение
1000			
2000			
…			
10000			

Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

- Desk-side T-Forge Mini cluster
  - AMD Opteron® 275 2.2 GHz dual core processors
  - RAM 4 GB

Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки…

- Intel Core 2 CPU 6300 1.86 GHz, 2 GB RAM

Упражнение 6: Разработка параллельного алгоритма, основанного на разделении матрицы на блоки

- Pentium D 820 2.8 GHz, 2 GB RAM

Заключение

- Рассмотрен три метода параллельного выполнения умножения матрицы на вектор
- Разработаны приложения, реализующие последовательный и параллельные алгоритмы умножения матрицы на вектор
- Проведены вычислительные эксперименты, проведено сравнение последовательного и параллельного алгоритмов