University of Nizhni Novgorod
Faculty of Computational Mathematics & Cybernetics

Introduction to Parallel Programming

Section 7. *Parallel Methods for Matrix-Vector Multiplication*

Gergel V.P., Professor, D.Sc.,
Software Department

Contents

Introduction

- Matrix calculations are widely used in various scientific and engineering applications
- Matrix operations usually take time-consuming calculations
- Matrix operations give a good opportunity to demonstrate wide range of parallel methods and techniques

Being highly time-consuming, matrix computations are the typical area of parallel computations

Problem Statement

Matrix-vector multiplication

$$c = A \cdot b$$

or

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

- Matrix-vector multiplication can be reduced to ***m*** inner products of matrix *A* rows and vector *b*

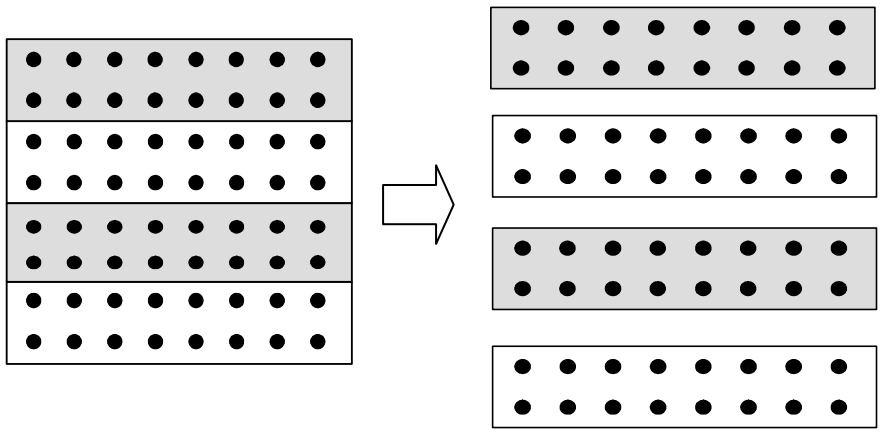
$$c_i = (a_i, b) = \sum_{j=0}^{n-1} a_{ij} b_j, \quad 0 \le i < m$$

Data parallelism can be exploited to design parallel computations

Data Decomposition Schemes : *Striped Decomposition…*

Block-striped Decomposition

Rowwise Block Striping



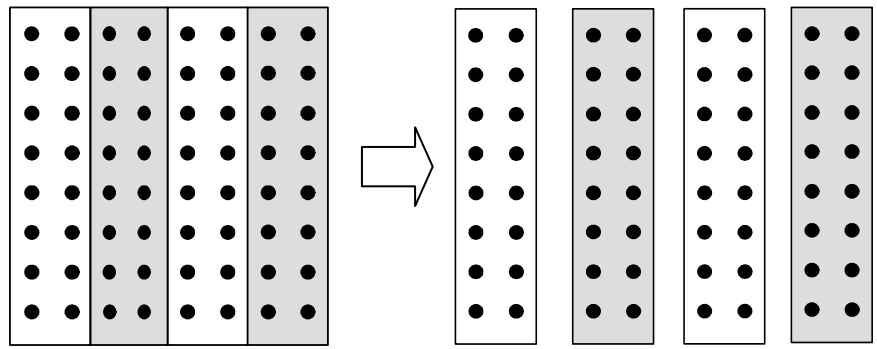
$$A = (A_0, A_1, ..., A_{p-1})^T,$$

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

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

$(a_i, 0 \le i < m, - \ matrix \ A \ rows)$

Columnwise Block Striping



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

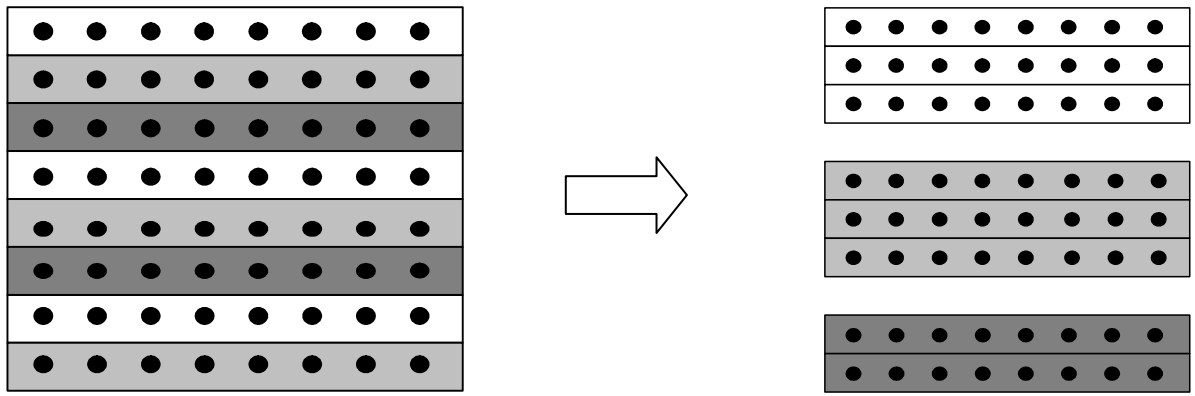
$$A_i = (\alpha_{i_0}, \alpha_{i_1}, ..., \alpha_{i_{k-1}}) \ ,$$

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

$(\alpha_i, 0 \le i < m, - \ matrix \ A \ columns)$

Data Decomposition Schemes : *Striped Decomposition*

Rowwise Cyclic-Striped Decomposition

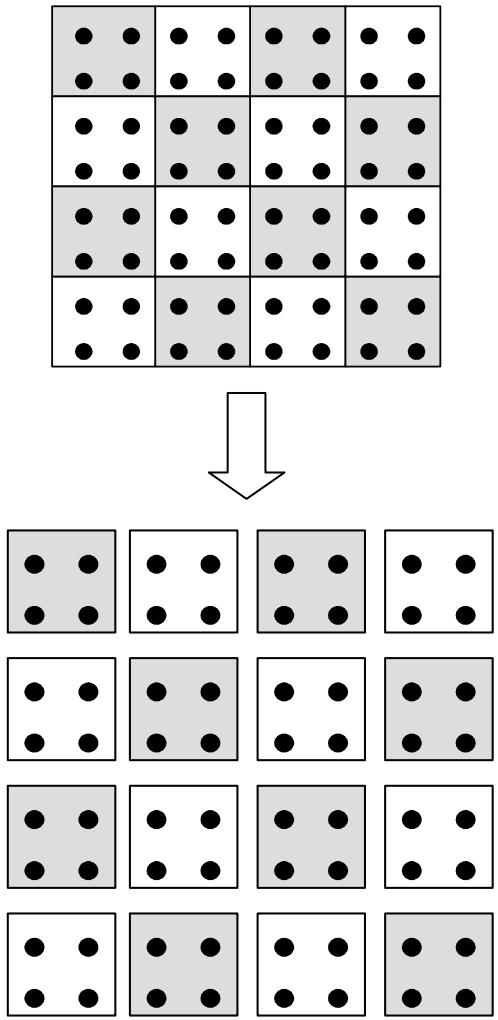


$$A = (A_0, A_2, ..., A_{p-1})^T,$$

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

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

Data Decomposition Schemes: *Checkerboard Decomposition*



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

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

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

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

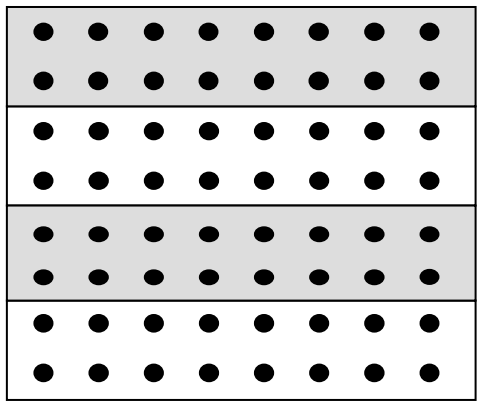
Sequential Algorithm

```
// Algorithm 7.1
// Sequential 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]
  }
}
```

- Matrix-vector multiplication consists of ***m*** inner products
- The algorithm's complexity is *O(mn)*

Algorithm 1: *Rowwise Block-Striped Decomposition*…

- **Data distribution** – rowwise block-striped decomposition



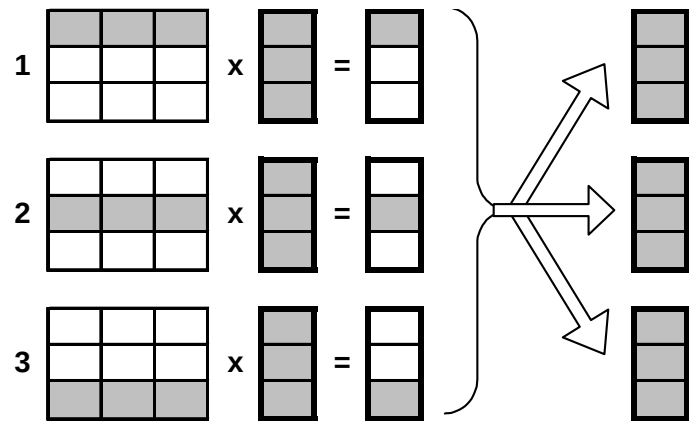
- **Basic subtask** – inner product of matrix ***A*** row and vector ***b***

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

Algorithm 1: *Rowwise Block-Striped Decomposition*…

❑ Analysis of Information Dependencies

– To perform the basic subtask of inner product the processor must hold the corresponding row of matrix ***A*** and the copy of vector ***b***. After computing each basic subtask determines one of the elements of the result vector ***c***,

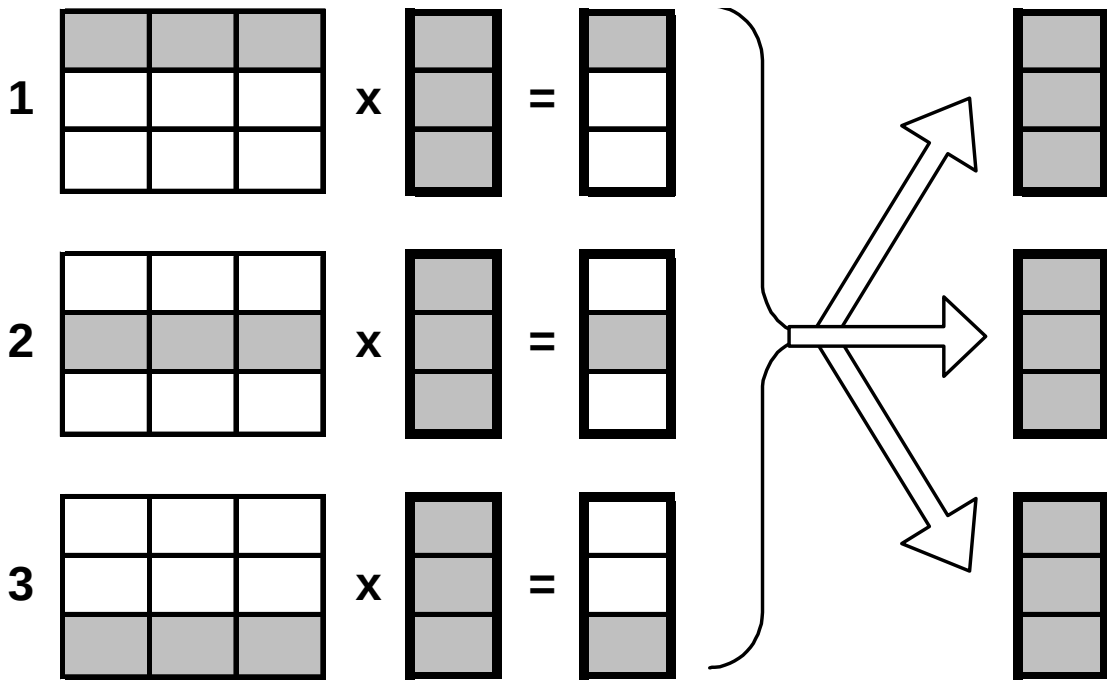


– To combine the computation results

and to obtain the vector ***c*** on each processor of the system, it is necessary to execute the gather and broadcast (*Allgather*) operations

Algorithm 1: *Rowwise Block-Striped Decomposition…*

❑ Scheme of Information Dependences



Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Aggregating and Subtasks Distributing among the Processors**
  - In case when the number of processors ***p*** is less than the number of basic subtasks ***m***, we can combine the basic subtasks in such a way that each processor would execute several inner products of matrix ***A*** row and vector ***b***. In this case after the completion of computation, each aggregated basic subtask determines several elements of the result vector ***c***,
  - Subtasks distribution among the processors of the computational system have to meet the requirements of effective All-gather operation execution

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Efficiency Analysis:**
  - Speed-up and Efficiency generalized estimates

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

Developed method of parallel computations allows to achieve ideal speed-up and efficiency characteristics

Algorithm 1: *Rowwise Block-Striped Decomposition…*

❑ Efficiency Analysis (detailed estimates):

- Time of parallel algorithm execution, that corresponds to the processor calculations:

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

- Time of All-gather operations can be obtained by means of the Hockney model:

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

Total time of parallel algorithm execution is

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

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Description of the parallel program sample…**
  - The main program function
    - realizes the logic of the algorithm operations,
    - sequentially calls out the necessary subprograms.

Code

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Description of the parallel program sample…**
  - Function **ProcessInitialization:**
    - defines the initial data for matrix A and vector b
    - the values for matrix A and vector b are formed in function RandomDataInitialization.

Code

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Description of the parallel program sample**
  - Function **DataDistribution**:
    - pushes out vector *b*,
    - distributes the rows of initial matrix *A* among the processes of the computational system.

Code

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Description of the parallel program sample**
  - Function **ParallelResultCaculation**:
    - performs the multiplication of the matrix rows, which are at a given moment distributed to a given process, by a vector,
    - forms the block of the result vector *c* .

Code

Algorithm 1: *Rowwise Block-Striped Decomposition…*

- **Description of the parallel program sample**
  - Function **ResultReplication:**
    - unites the blocks of the result vector *c*, which have been obtained on different processors,
    - replicates the result vector to all the computational system processes.

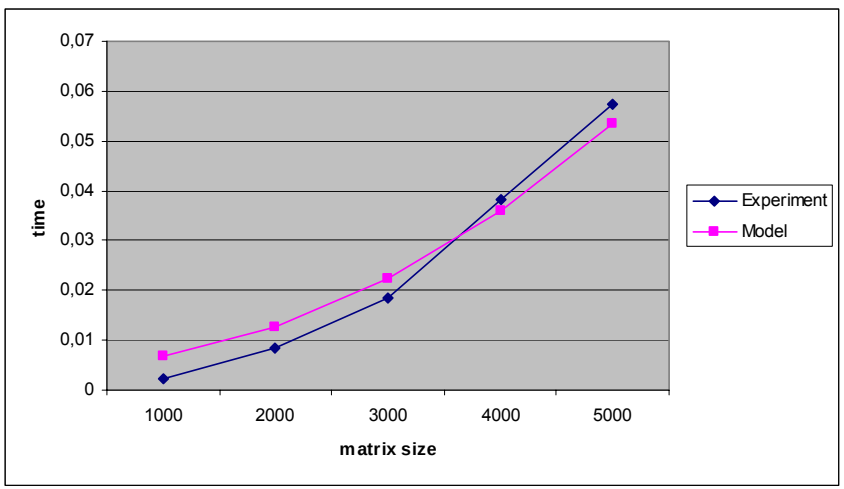
Code

Algorithm 1: *Rowwise Block-Striped Decomposition…*

❑ Results of computational experiments…

– Comparison of theoretical estimations and results of computational experiments

Matrix Size	2 processors		4 processors		6 processors	
	Model	Experiment	Model	Experiment	Model	Experiment
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

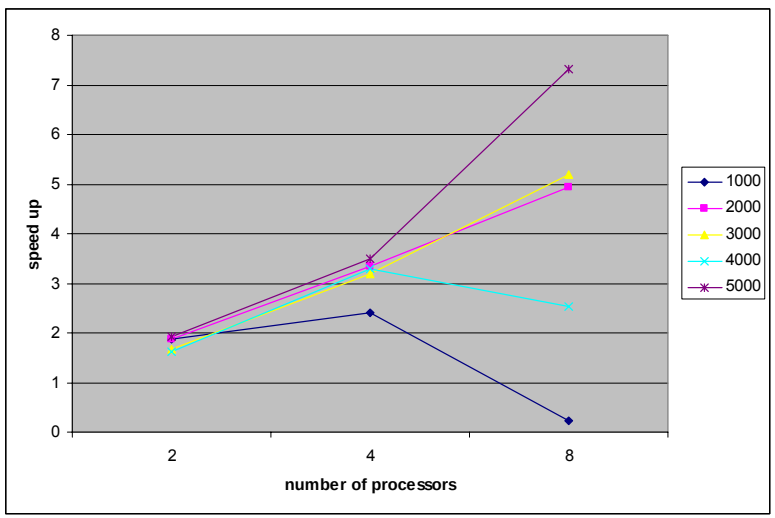


Algorithm 1: *Rowwise Block-Striped Decomposition*

❑ Results of computational experiments:

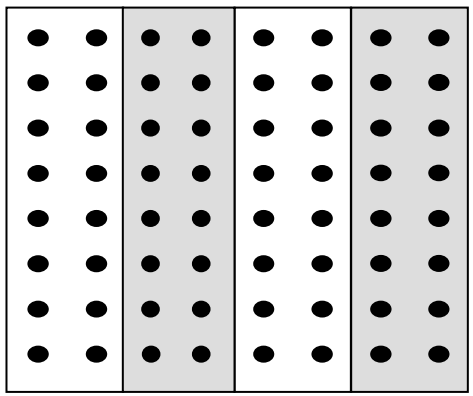
– Speed-up

Matrix Size	Sequential Algorithm	Parallel Algorithm					
		2 processors		4 processors		8 processors	
		Time	Speed Up	Time	Speed Up	Time	Speed Up
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

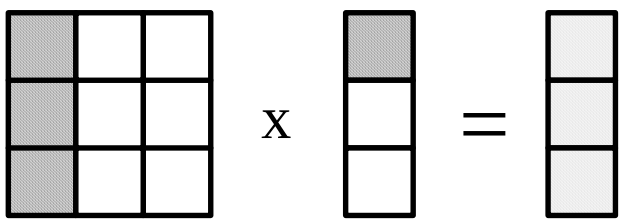


Algorithm 2: *Columnwise Block-Striped Decomposition…*

- **Data distribution** – columnwise block-striped decomposition



- **Basic subtask** – multiplication of matrix ***A*** column to one of the vector ***b*** elements



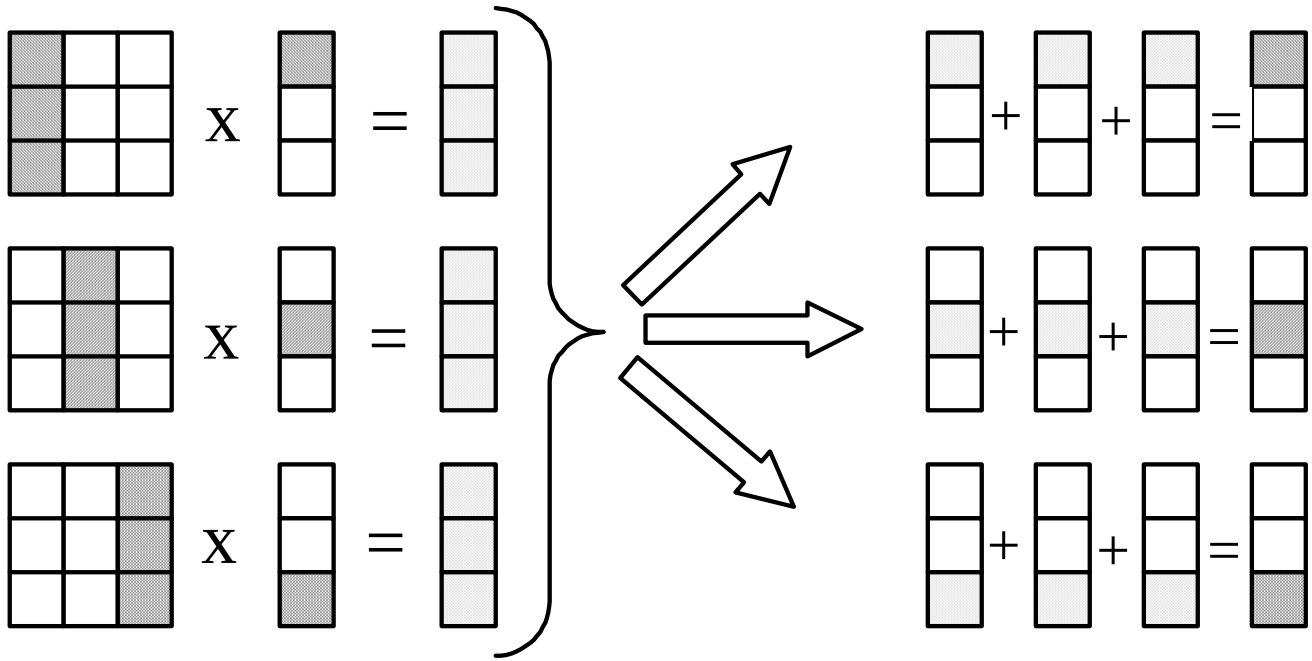
Algorithm 2: *Columnwise Block-Striped Decomposition…*

❑ Analysis of Information Dependencies

- To perform the calculations ***j***-th basic subtask must hold ***j***-th column of ***A*** matrix and ***j***-th elements of ***b*** and ***c*** vectors, i.e. b_j and c_j,
- At the time of computations ***j***–th subtask performs the multiplication of it's A matrix column by b_j element and calculates the ***c'(j)*** vector of partial results ($c'_i(j) = a_{ij} b_j$, $0 \le i < n$),
- To obtain the result vector ***c*** subtasks should exchange their partial results and sum obtained data ($c_i = \sum_{j=0}^{n-1} c'_i(j), 0 \le i < n$)

Algorithm 2: *Columnwise Block-Striped Decomposition…*

❑ Scheme of Information Dependences



Algorithm 2: *Columnwise Block-Striped Decomposition…*

- **Aggregating and Subtasks Distributing among the Processors**
  - In case when the number of matrix ***A*** columns ***n*** is greater than the number of processors ***p***, we can combine the basic subtasks in such a way that each subtask would contain several columns of matrix ***A*** (in this case the matrix is decomposed into the vertical strips). After the completion of computation and data passing procedure, each aggregated basic subtask determines partial results of each element of vector ***c***,
  - Subtasks distribution among the processors of the system have to meet the requirements of effective execution of partial results exchanging operation

Algorithm 2: *Columnwise Block-Striped Decomposition…*

- **Efficiency Analysis:**
  - Speed-up and Efficiency generalized estimates

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

Developed method of parallel computations allows to achieve ideal speed-up and efficiency characteristics

Algorithm 2: *Columnwise Block-Striped Decomposition…*

❑ Efficiency Analysis (detailed estimates)…

- Time of parallel algorithm execution, that corresponds to the processor calculations, is

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

- Data passing needed during computation can be carried out in two ways:

  • Every process pass it's data successively to the other processes - time of this operation can be obtained by means of the Hockney model:

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

  • When the network topology can be represented as a hypercube, the data passing operation can be performed in $log_2 p$ steps:

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

Algorithm 2: *Columnwise Block-Striped Decomposition…*

❑ Efficiency Analysis (detailed estimates):

- The total time of the parallel algorithm execution with the first type of the data passing implementation is:

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

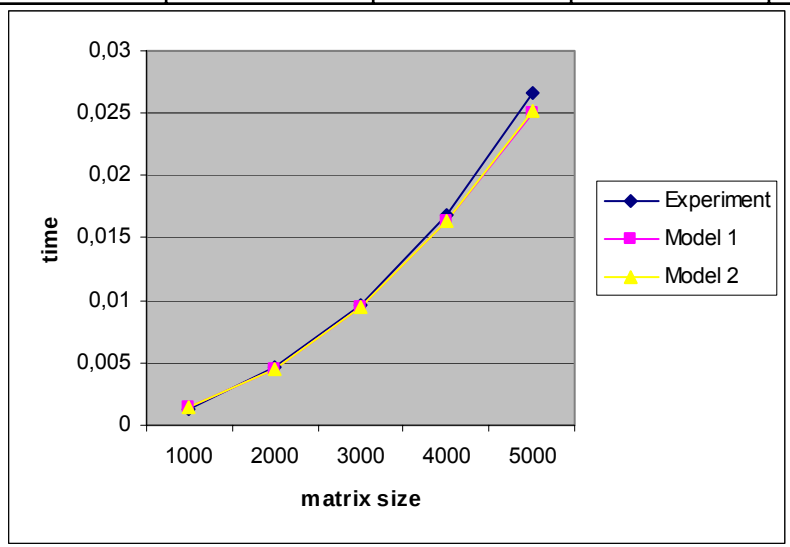
- When the second way of data passing is implemented, the total time of the parallel algorithm is:

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

Algorithm 2: *Columnwise Block-Striped Decomposition…*

- **Results of computational experiments...**
  - Comparison of theoretical estimations and results of computational experiments

Matrix Size	2 processors			4 processors			8 processors		
	Model 1	Model 2	Experiment	Model 1	Model 2	Experiment	Model 1	Model 2	Experiment
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

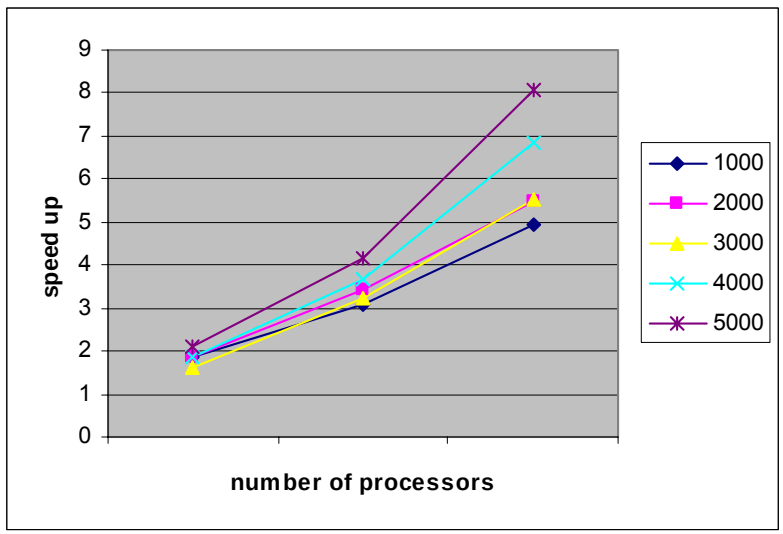


Algorithm 2: *Columnwise Block-Striped Decomposition*

❑ Results of computational experiments:

– Speed-up

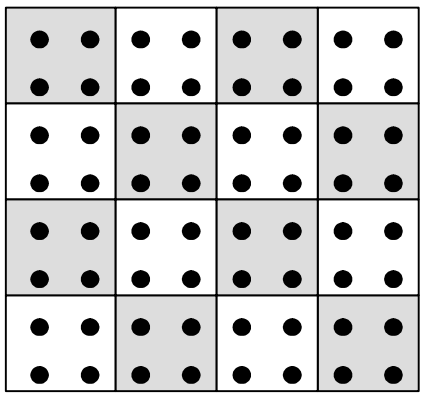
Matrix Size	Sequential Algorithm	2 processors		4 processors		8 processors	
		Time	Speed up	Time	Speed up	Time	Speed up
1000	0,0041	0,0022	1,8352	0,0132	0,3100	0,0008	4,9409
2000	0,016	0,0085	1,8799	0,0046	3,4246	0,0029	5,4682
3000	0,031	0,019	1,6315	0,0095	3,2413	0,0055	5,5456
4000	0,062	0,0331	1,8679	0,0168	3,6714	0,0090	6,8599
5000	0,11	0,0518	2,1228	0,0265	4,1361	0,0136	8,0580



Algorithm 3: *Checkerboard Decomposition…*

- **Data distribution** – checkerboard scheme

  Let the number of processors $p=s \cdot q$, the number of rows of matrix A is divisible by s, the number of columns is divisible by q, i.e. $m=k \cdot s$ и $l=n \cdot q$.



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

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

Algorithm 3: *Checkerboard Decomposition…*

- **Basic subtask** is based on the operations, that are carried out on the matrix blocks:
  - The indices *(i, j)* of the matrix block can be used to indicate subtasks,
  - Subtasks perform multiplication of the matrix ***A*** block and vector ***b*** block

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

  - After the multiplication of matrix ***A*** block and vector ***b*** each subtask *(i,j)* will hold the vector of partial results *c'(i,j)*,

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

Algorithm 3: *Checkerboard Decomposition…*

❑ Analysis of Information Dependencies:

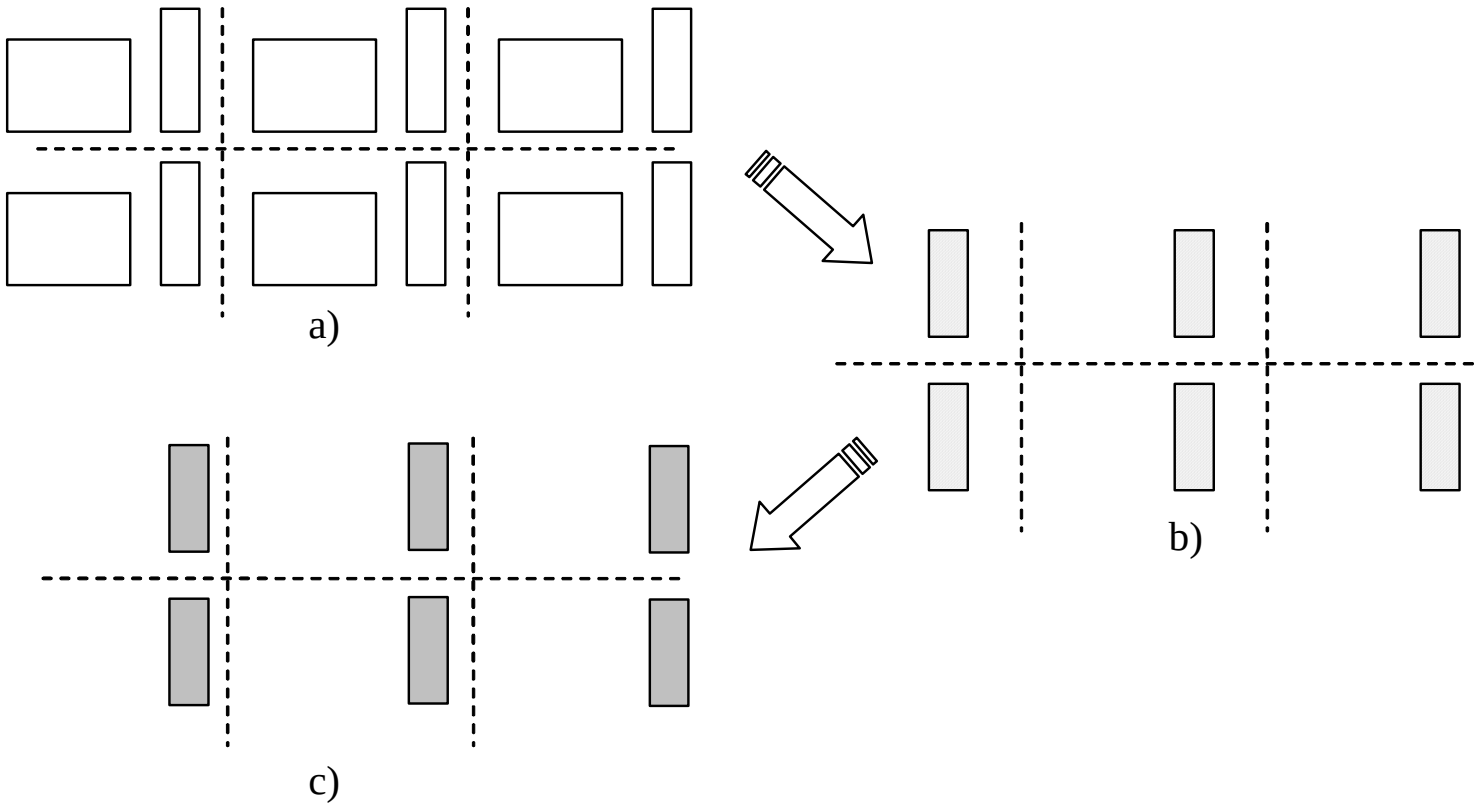
- Tasks in each row of the task grid perform a sum reduction on their block of the vector ***c***:

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

- Computations can be perform in such a way that after the sum reduction the result vector ***c*** will be distributed by blocks among the tasks in each column of the subtask grid,
- The information dependence between the basic subtasks takes place only on the stage of summing the partial results

Algorithm 3: *Checkerboard Decomposition…*

❑ Scheme of Information Dependences



Algorithm 3: *Checkerboard Decomposition…*

- **Aggregating and Subtasks Distributing among the Processors**
  - We can select matrix ***A*** block sizes so that the amount of basic subtasks will be equal to the number of processors ***p***, ***p=s·q***:
    - If the number ***s*** of blocks in horizontal order increases, the amount of iterations in the summing of the partial results grows,
    - If the vertical size ***q*** of task grid grows, the amount of passing data increases,
  - Subtasks distribution among the processors of the system have to meet the requirements of effective execution of sum reduction

Algorithm 3: *Checkerboard Decomposition…*

- **Efficiency Analysis:**
  - Speed-up and Efficiency generalized estimates:

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

Developed method of parallel computations allows to achieve ideal speed-up and efficiency characteristics

Algorithm 3: *Checkerboard Decomposition…*

❑ Efficiency Analysis (detailed estimates):

- The time of blocks multiplication is:

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

- The sum reduction can be executed in accordance with the cascade scheme. In this case communications includes $log_2 q$ data passing operations, each message has $w\lceil n/s \rceil$ size. As a result, the communication time can be estimate by means of the Hockney model:

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

Total time of parallel algorithm execution is:

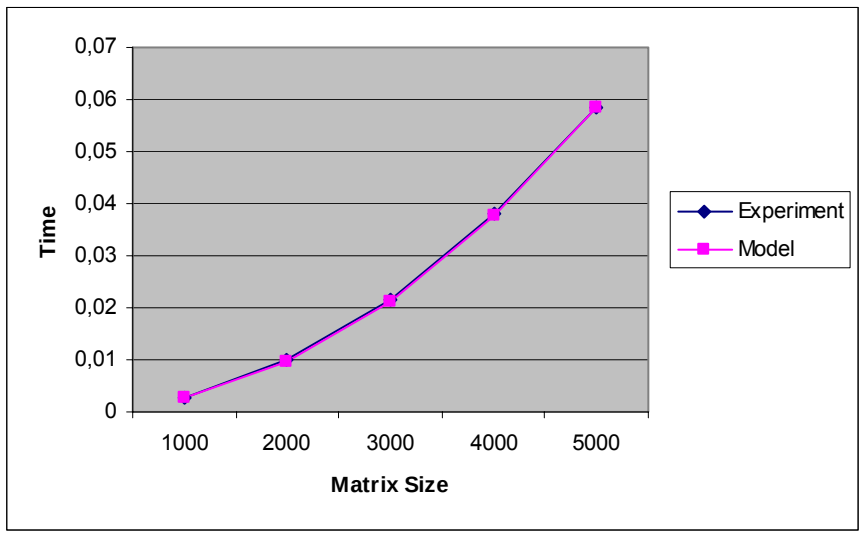
$$T_p = \lceil n/s \rceil \cdot (2 \cdot \lceil n/q \rceil - 1) \cdot \tau + (\alpha + w \lceil n/s \rceil / \beta) \lceil \log_2 q \rceil$$

Algorithm 3: *Checkerboard Decomposition…*

❑ Results of computational experiments…

– Comparison of theoretical estimations and results of computational experiments

Matrix Size	4 processors		9 processors	
	Model	Experiment	Model	Experiment
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

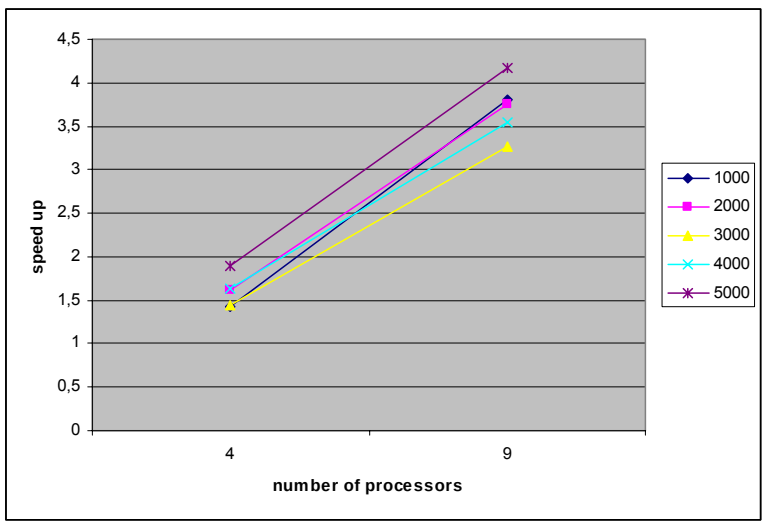


Algorithm 3: *Checkerboard Decomposition…*

❑ Results of computational experiments:

– Speed-up

Matrix Size	Sequential Algorithm	Parallel Algorithm			
		4 processors		9 processors	
		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

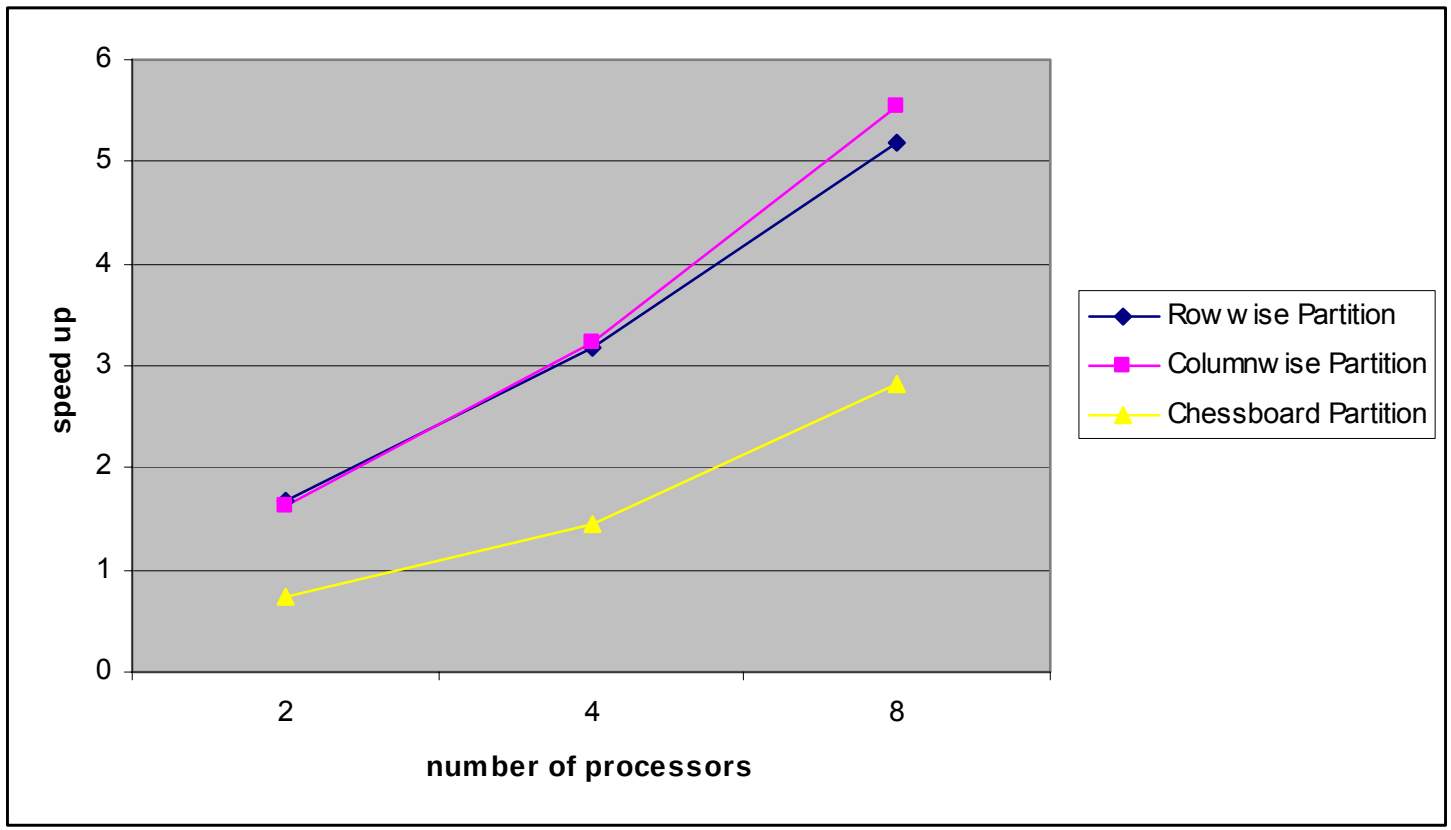


Summary…

- Various ways of matrix distribution among the processors have been described:
  - Striped rowwise/columnwise decomposition,
  - Checkerboard decomposition
- Three algorithms of matrix-vector multiplication have been designed, analyzed and benchmarked:
  - Algorithm 1 is based on rowwise block-striped matrix decomposition,
  - Algorithm 2 is based on columnwise block-striped matrix decomposition,
  - Algorithm 3 is based on checkerboard matrix decomposition
- Theoretical analysis allows to predict the speed-up and efficiency characteristics of parallel computations with sufficiently high accuracy

Summary

- All presented algorithms have nearly the same theoretical estimations for speed-up and efficiency characteristics



Discussions

- Why it is allowable to copy the vector to all processes while developing the parallel algorithms for matrix-vector multiplication?
- Which algorithm shows the best speed-up and efficiency characteristics?
- Can the utilization of cyclic-striped data decomposition influence on the time of algorithm execution?
- Which data passing operations are required for the parallel matrix-vector multiplication algorithms?

Exercises

- Develop the parallel program for matrix-vector multiplication based on columnwise block-striped decomposition
- Develop the parallel program for matrix-vector multiplication based on checkerboard decomposition
- Formulate the theoretical estimations for the execution time of these algorithms
- Execute programs. Compare the time of computational experiments and the theoretical estimations being obtained

References

- **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.

Next Section

- Parallel Methods for Matrix Multiplication

Author's Team

Gergel V.P., Professor, Doctor of Science in Engineering, Course Author

Grishagin V.A., Associate Professor, Candidate of Science in Mathematics

Abrosimova O.N., Assistant Professor (chapter 10)

Kurylev A.L., Assistant Professor (learning labs 4,5)

Labutin D.Y., Assistant Professor (ParaLab system)

Sysoev A.V., Assistant Professor (chapter 1)

Gergel A.V., Post-Graduate Student (chapter 12, learning lab 6)

Labutina A.A., Post-Graduate Student (chapters 7,8,9, learning labs 1,2,3, ParaLab system)

Senin A.V., Post-Graduate Student (chapter 11, learning labs on Microsoft Compute Cluster)

Liverko S.V., Student (ParaLab system)

About the project

The purpose of the project is to develop the set of educational materials for the teaching course “Multiprocessor computational systems and parallel programming”. This course is designed for the consideration of the parallel computation problems, which are stipulated in the recommendations of IEEE-CS and ACM Computing Curricula 2001. The educational materials can be used for teaching/training specialists in the fields of informatics, computer engineering and information technologies. The curriculum consists of **the training course “Introduction to the methods of parallel programming”** and **the computer laboratory training “The methods and technologies of parallel program development”**. Such educational materials makes possible to seamlessly combine both the fundamental education in computer science and the practical training in the methods of developing the software for solving complicated time-consuming computational problems using the high performance computational systems.

The project was carried out in Nizhny Novgorod State University, the Software Department of the Computing Mathematics and Cybernetics Faculty (http://www.software.unn.ac.ru). The project was implemented with the support of Microsoft Corporation.