University of Nizhni Novgorod
Faculty of Computational Mathematics & Cybernetics

Introduction to Parallel Programming

Section 9.

Parallel Methods for Solving Linear Systems

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

Contents

Problem Statement…

A *linear equation* with ***n*** unknown variables

$$a_0 x_0 + a_1 x_1 + \ldots + a_{n-1} x_{n-1} = b$$

A finite set of ***n*** linear equations is called a *system of linear equations* or a *linear system*

$$\begin{aligned}
a_{0,0} x_0 \quad & + a_{0,1} x_1 \quad && + \ldots + a_{0,n-1} x_{n-1} \quad && = b_0 \\
a_{1,0} x_0 \quad & + a_{1,1} x_1 \quad && + \ldots + a_{1,n-1} x_{n-1} \quad && = b_1 \\
& \ldots \\
a_{n-1,0} x_0 \quad & + a_{n-1,1} x_1 \quad && + \ldots + a_{n-1,n-1} x_{n-1} \quad && = b_{n-1}
\end{aligned}$$

or in the matrix form:

$$Ax = b$$

Problem Statement

- The *solution* of the linear system is the values of the unknown vector *x*, that satisfy every equation in the linear system for the given matrix ***A*** and vector ***b***

The Gauss method – sequential algorithm…

- The *main idea of the method* is by using the equivalent operations to transform a dense system into an upper triangular system, which can be easily solved
- *Equivalent transformations:*
  - Multiplying an equation by a nonzero constant,
  - Interchanging two equations,
  - Adding an equation multiplied by a constant to another equation.

The Gauss method – sequential algorithm…

- On the first stage of the algorithm, which is called *the Gaussian elimination*, the initial system of linear equations is transformed into an upper triangular system by the sequential elimination of unknowns:

$$U x = c, \qquad U = \begin{pmatrix} u_{0,0} & u_{0,1} & \dots & u_{0,n-1} \\ 0 & u_{1,1} & \dots & u_{1,n-1} \\ & & \dots & \\ 0 & 0 & \dots & u_{n-1,n-1} \end{pmatrix}$$

- On the second stage of the algorithm, which is called *the back substitution*, the values of the variables are calculated

The Gauss method – sequential algorithm…

❑ Gaussian elimination:

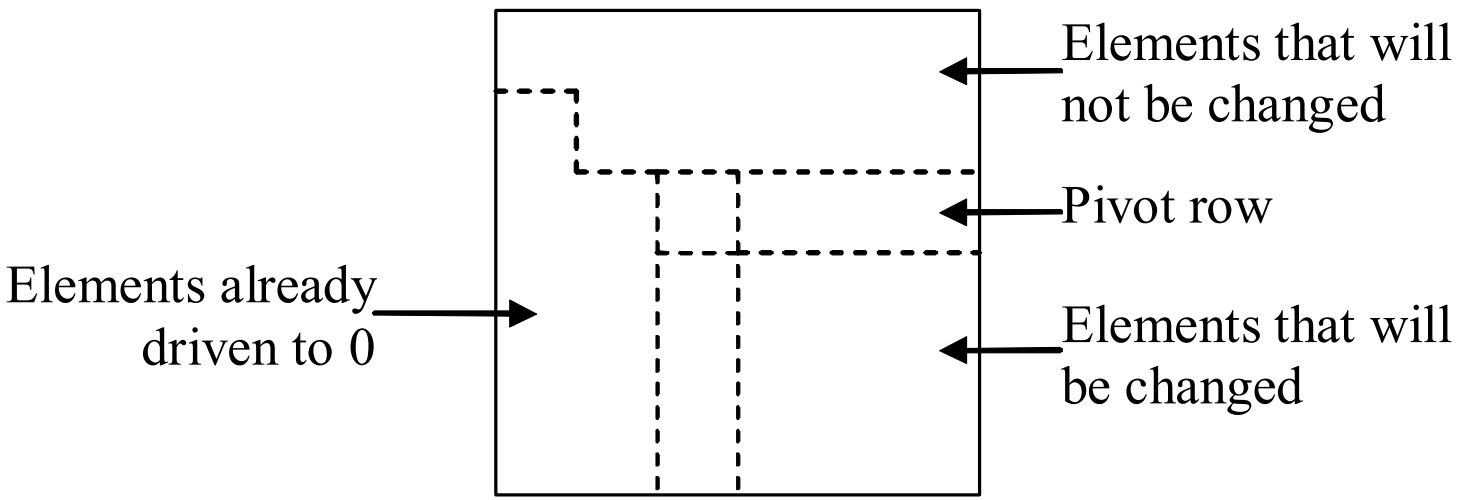
– At step *i, 0≤ i<n-1,* of the algorithm the nonzero elements below the diagonal in column *i* are eliminated by replacing each row *k*, where *i< k≤ n-1,* with the sum of the row *k* and the row *i* multiplied by *the value* ($-a_{ki}/a_{ii}$),

– All the necessary calculations are determined by the equations:

$$a'_{kj} = a_{kj} - (a_{ki} / a_{ii}) \cdot a_{ij},$$
$$b'_k = b_k - (a_{ki} / a_{ii}) \cdot b_i,$$
$$i \le j \le n-1, i < k \le n-1, 0 \le i < n-1$$

The Gauss method – sequential algorithm…

- Scheme of data at the *i*-th iteration of the Gaussian elimination



The Gauss method – sequential algorithm

❑ Back substitution

After the matrix of the linear system was transformed to the upper rectangular type, it becomes possible to calculate the unknown variables:

- We can solve the last equation directly, since it has only a single unknown x_{n-1} ,
- After we have determined the x_{n-1} , we can simplify the other equations by substituting the value of x_{n-1} ,
- Then the equation ***n-2*** has only the single unknown x_{n-2} and can be solved and so on.

The calculations of the back substitution can be represented as follows:

$$x_{n-1} = b_{n-1} / a_{n-1,n-1},$$

$$x_i = (b_i - \sum_{j=i+1}^{n-1} a_{ij} x_j) / a_{ii}, \quad i = n-2, n-3, ..., 0$$

The Gauss method – parallel algorithm…

- **Basic subtasks:**
  - All calculations are the uniform row operations of the matrix ***A***,
  - Data parallelism can be exploited to design parallel computations for the Gauss algorithm,
  - The calculations, that corresponds to one equation of the linear system, can be chosen as ***the basic subtask***.

The Gauss method – parallel algorithm…

❑ Analysis of Information Dependencies…

– Every iteration *i* of the Gaussian elimination algorithm consists of:

- **Choosing the pivot row** – all subtasks *k* (*k*≥*i)* have to exchange their matrix *A* values from the column with the eliminated variable x_i to find the row with absolute maximum value, the corresponding row becomes the pivot for the current iteration,
- **Sending** the pivot row to all of the subtasks, which number *k* is greater than *i* (*k*>*i*),
- **Subtracting** the pivot row from rows of the subtasks *k (k>i)* (eliminating the unknown x_i).

The Gauss method – parallel algorithm…

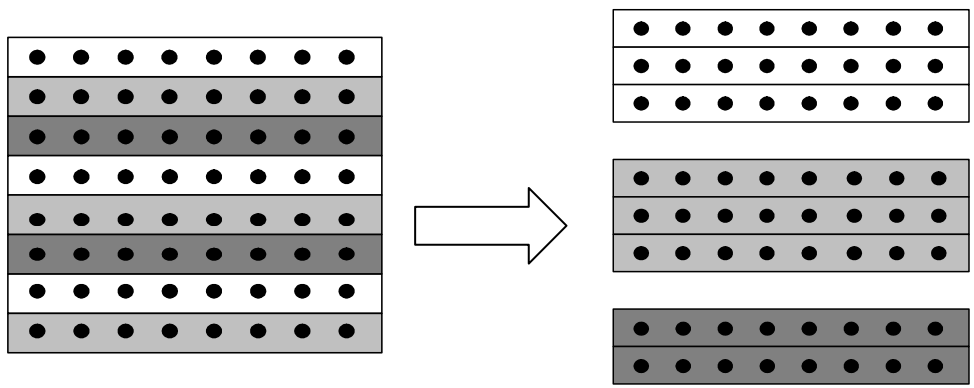
❑ Analysis of Information Dependencies:

– While executing the iterations of the back substitution, the subtasks perform necessary actions to obtain the values of the unknowns:

- After the subtask i, $0 \le i < n-1$, have calculated the value of the unknown x_i, this value must be sent to all the subtasks k, $k < i$,
- Then all the subtasks substitute the received value to their equation of the linear system and adjust their value of b.

The Gauss method – parallel algorithm…

- **Scaling and Subtasks Distributing among the Processors…**
  - In case when the number of processors ***p*** is less than the number of the rows (***p<n)***, the basic subtasks can be aggregated in such a way that each processor would process several equations of the linear system



*The usage of **Rowwise Cyclic-Striped Decomposition** allows to achieve best characteristics of calculation load balancing among the subtasks*

The Gauss method – parallel algorithm…

- **Scaling and Subtasks Distributing among the Processors:**
  - The main form of communication interactions of the subtasks is the one-to-all broadcast,
  - As a result, for the effective implementation of data transmission operations between the basic subtasks the topology of the data transmission network must have the *a hypercube* or *the complete graph* structure.

The Gauss method – parallel algorithm…

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

$$S_p = \frac{(2n^3/3 + n^2)}{\frac{1}{p}\sum_{i=2}^{n}(3i + 2i^2)}, \quad E_p = \frac{(2n^3/3 + n^2)}{\sum_{i=2}^{n}(3i + 2i^2)}$$

Balancing of the calculation load among the processors, in general, is enough uniform

The Gauss method – parallel algorithm…

- **Efficiency Analysis** (detailed estimates)…
  - **Time of parallel algorithm execution, that corresponds to the processor calculations, consists of:**
    - **Time of the Gaussian elimination** (*n-1* iterations):
      - the first step is to chose the maximum value in the column with the eliminated variable,
      - the subtraction of the pivot row from all of the rest rows of matrix *A* stripe:

$$T_p^1 = \sum_{i=0}^{n-2} \left[\frac{(n-i)}{p} + \frac{(n-i)}{p} \cdot 2(n-i) \right] = \frac{1}{p} \sum_{i=0}^{n-2} \left[(n-i) \cdot +2(n-i)^2 \right]$$

    - **Time of the back substitution (*n-1* iteration):**
      - The adjusting of the vector *b* elements after the broadcast of the calculated value of the unknown variable:

$$T_p^2 = \sum_{i=0}^{n-2} 2 \cdot (n-i) / p$$

The Gauss method – parallel algorithm…

- **Efficiency Analysis** (detailed estimates)…

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

- **the Gaussian elimination:**
  - To determine the pivot row the processors exchange the local maximum values of the column with the eliminated unknown (*MPI_Allreduce*):

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

  - To broadcast of the pivot row:

$$T_p^2(comm) = (n-1)\cdot \log_2 p \cdot (\alpha + wn/\beta)$$

- **Back Substitution:**
  - At each iteration the processor broadcasts the calculated element of the result vector to all of the rest processors:

$$T_p^3(comm) = (n-1)\cdot \log_2 p \cdot (\alpha + w/\beta)$$

The Gauss method – parallel algorithm…

- **Efficiency Analysis** (detailed estimates):

Total time of parallel algorithm execution is:

$$T_p = \frac{1}{p}\sum_{i=2}^{n}(3i + 2i^2)\,\tau + (n-1)\cdot\log_2 p\cdot(3\alpha + w(n+2)/\beta)$$

The Gauss method – parallel algorithm…

❑ Description of the parallel program sample…

– **The *main* function**. Performs the main actions of the algorithm by calling the necessary functions sequentially:

- ***ProcessInitialization*** initializes the data of the problem,
- ***ParallelResultCalculation*** carries out the parallel Gauss algorithm,
- ***ResultCollection*** gathers the blocks of the result vector from the processors,
- ***ProcessTermination*** outputs the results of problem solving and deallocates the memory, which was used to store the necessary data.

The Gauss method – parallel algorithm…

❑ Description of the parallel program sample…

– **The *main* function**. This function uses the arrays:

- ***pParallelPivotPos*** determines the positions of the rows, that have been chosen as pivot rows at the iterations of the Gaussian elimination algorithm; as a result, elements of this array determines the order of the back substitution algorithm iterations (this array is global, every changing of its elements has to be accompanied by the communication operation of sending the changed data to all of the rest processors),
- ***pProcPivotIter*** determines the numbers on the Gaussian elimination iterations, that uses the rows of the current processor as pivots; zero value of the array element means that the corresponding row have to be processed during the Gaussian elimination iteration (each processor has the local array pProcPivotIter).

Code

The Gauss method – parallel algorithm…

❑ Description of the parallel program sample…

– **The function *ParallelResultCalculation***. This function uses the arrays:

• This function contains the calls of the functions for executing the Gauss algorithm stages

Code

The Gauss method – parallel algorithm…

- **Description of the parallel program sample…**
  - The function ***ParallelGaussianElimination*** executes the parallel variant of the Gaussian elimination algorithm

Code

  - The function ***ParallelEliminateColumns*** carries out the subtraction of the pivot row from the rows, that are held by the same processor and haven't been used as pivots yet (the corresponding elements of the array *pProcPivotIter* are equal to zero)

The Gauss method – parallel algorithm…

- **Description of the parallel program sample:**
  - The function ***BackSubstitution*** implements the parallel variant of the back substitution algorithm.

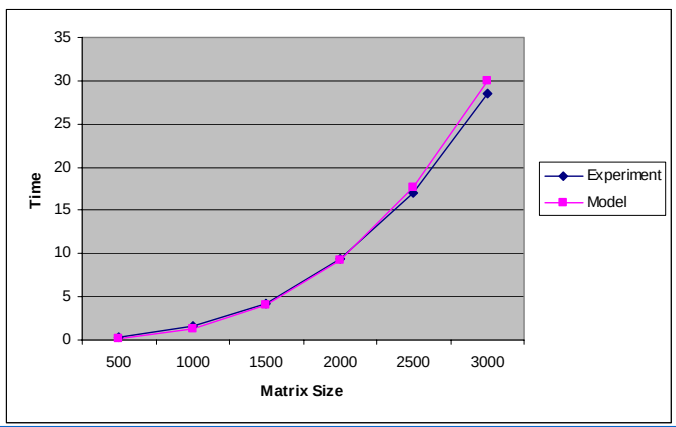
Code

The Gauss method – parallel algorithm…

❑ Results of computational experiments…

– Comparison of theoretical estimations and results of computational experiments

Matrix Size	2 processors		4 processors		8 processors	
	Model	Experiment	Model	Experiment	Model	Experiment
500	0,2393	0,3302	0,2819	0,5170	0,3573	0,7504
1000	1,3373	1,5950	1,1066	1,6152	1,1372	1,8715
1500	4,0750	4,1788	2,8643	3,8802	2,5345	3,7567
2000	9,2336	9,3432	5,9457	7,2590	4,7447	7,3713
2500	17,5941	16,9860	10,7412	11,9957	7,9628	11,6530
3000	29,9377	28,4948	17,6415	19,1255	12,3843	17,6864

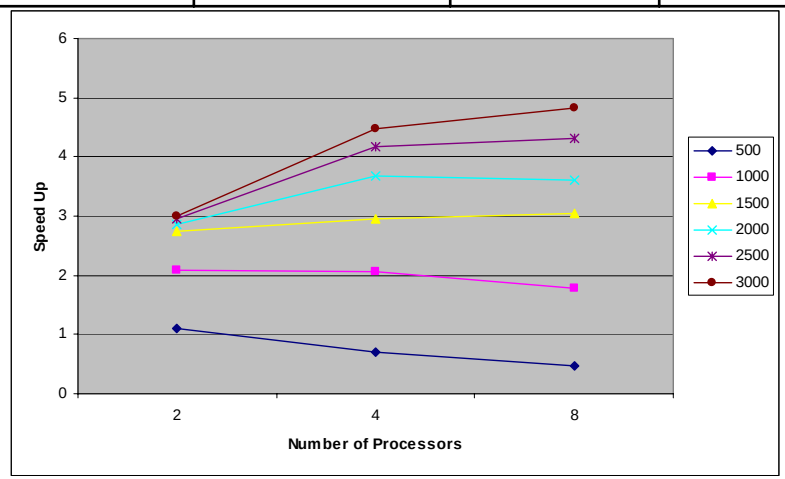


The Gauss method – parallel algorithm

❑ Results of computational experiments:

– Speedup

Matrix Size	Serial Algorithm	Parallel Algorithm					
		2 processors		4 processors		8 processors	
		Time	Speed Up	Time	Speed Up	Time	Speed Up
500	0,36	0,3302	1,0901	0,5170	0,6963	0,7504	0,4796
1000	3,313	1,5950	2,0770	1,6152	2,0511	1,8715	1,7701
1500	11,437	4,1788	2,7368	3,8802	2,9474	3,7567	3,0443
2000	26,688	9,3432	2,8563	7,2590	3,6765	7,3713	3,6204
2500	50,125	16,9860	2,9509	11,9957	4,1785	11,6530	4,3014
3000	85,485	28,4948	3,0000	19,1255	4,4696	17,6864	4,8333



The Conjugate Gradient Method…

- **Iterative Methods for Solving Systems of Linear Equations:**
  - Come up with a series of approximations x_0, x_1,…, x_k,…, to the value of the solution x^* of the system $Ax=b$,
  - Each following approximation gives the estimation of the solution value with the decreasing approximation error,
  - The estimation for the exact solution can be obtained with any definite accuracy.

The conjugate gradient method *– one of the well known iterative algorithms for solving systems of linear equations*

The Conjugate Gradient Method…

- The conjugate gradient method can be applied to solve the system of linear equations with the symmetric and positive definite matrix:
  - A $n \times n$ matrix $\boldsymbol{A}$ is *symmetric* if it is equal to its transposed matrix, i.e. $A=A^T$,
  - A $n \times n$ matrix $\boldsymbol{A}$ is *positive definite* if for every nonzero vector $\boldsymbol{x}$ and its transpose x^T, the product $\boldsymbol{x^T A x > 0}$
- If calculation errors are ignored, the conjugate gradient method is guaranteed to converge to a solution in $\boldsymbol{n}$ or fewer iterations

The Conjugate Gradient Method – Sequential Algorithm…

- If *A* is symmetric and positive definite, then the function

$$q(x) = \frac{1}{2} x^T \cdot A \cdot x - x^T b + c$$

has a unique minimizer that is the solution of the system *Ax=b*

The conjugate gradient method is one of many iterative algorithms that solve Ax=b by minimizing q(x)

The Conjugate Gradient Method – Sequential Algorithm…

- An ***iteration*** of the conjugate gradient method is of the form:

$$x^k = x^{k-1} + s^k d^k$$

where

- x^k – a new approximation of x,
- x^{k-1} – a approximation of x, which was computed on the previous step,
- s^k – a scalar step size,
- d^k – a direction vector.

The Conjugate Gradient Method – Sequential Algorithm…

Before the first iteration, x^0 and d^0 are both initialized to the zero vector and g^0 is initialized to $-b$.

Step 1: Compute the gradient

$$g^k = A \cdot x^{k-1} - b$$

Step 2: Compute the direction vector

$$d^k = -g^k + \frac{\left((g^k)^T, g^k\right)}{\left((g^{k-1})^T, g^{k-1}\right)} d^{k-1}$$

Step 3: Compute the step size

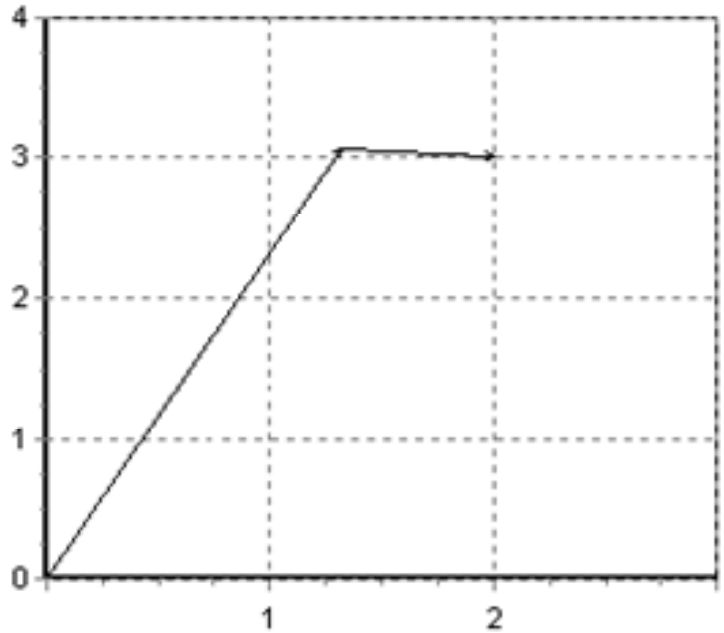
$$s^k = \frac{\left(d^k, g^k\right)}{(d^k)^T \cdot A \cdot d^k}$$

Step 4: Compute the new approximation of x

$$x^k = x^{k-1} + s^k d^k$$

The algorithm's complexity is $T_1 = 2n^3 + 13n^2$

The Conjugate Gradient Method – Sequential Algorithm



Iterations of the conjugate gradient method for solving the system of two linear equations:

$$\begin{cases} 3x_0 - x_1 = 3 \\ -x_0 + 3x_1 = 7 \end{cases}$$

The Conjugate Gradient Method – Parallel Algorithm…

❑ Parallel Computation Design

- Iterations of the conjugate gradient method can be executed only in sequence, so the most advisable approach is to parallelize the computations, that are carried out at each iteration,
- The most time-consuming computations are the multiplication of matrix ***A*** by the vectors ***x*** and ***d***,
- Additional operations, that have the lower computational complexity order, are different vector processing procedures (inner product, addition and subtraction, multiplying by a scalar).

While implementing the parallel conjugate gradient method, it can be used parallel algorithms of matrix-vector multiplication, that was described in the section 7 in detail

The Conjugate Gradient Method – Parallel Algorithm…

❑ Efficiency Analysis…

(using the parallel algorithm of matrix-vector multiplication, which is based on rowwise block-striped matrix decomposition, vectors are copied to all processors)

– The computation complexity of the parallel algorithm of matrix-vector multiplication based on the rowwise block-striped matrix decomposition:

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

– As a result, when all vectors are copied to all processors, the total calculation complexity of the parallel conjugate gradient method is equal to:

$$T_p = n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)$$

The Conjugate Gradient Method – Parallel Algorithm…

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

$$S_p = \frac{2n^3 + 13n^2}{n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)}, \quad E_p = \frac{2n^3 + 13n^2}{p \cdot n\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right)}$$

Balancing of the calculation load among the processors, in general, is enough uniform

The Conjugate Gradient Method – Parallel Algorithm…

- **Efficiency Analysis** (detailed estimates):
  - Time of the data communication operations can be obtained by means of the Hockney model (section 7):

$$T_p(comm) = 2n\left(\alpha \cdot \lceil \log p \rceil + w(n/p)(p-1)/\beta\right)$$

  - Total time of the parallel conjugate gradient algorithm execution is:

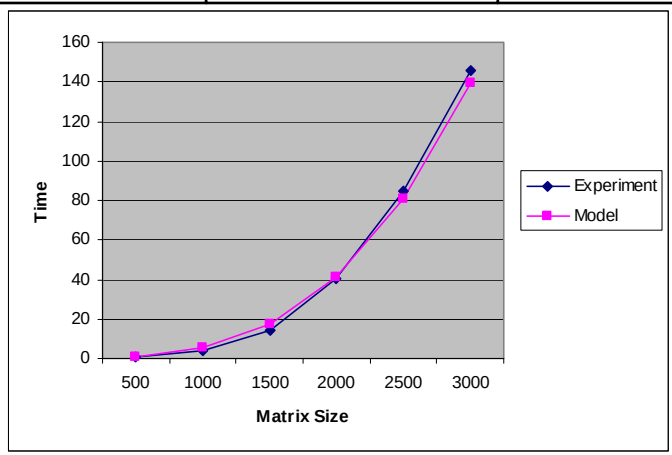
$$T_p = n \cdot \left[\left(2\lceil n/p \rceil \cdot (2n-1) + 13n\right) \cdot \tau + 2\left(\alpha \cdot \lceil \log p \rceil + w(n/p)(p-1)/\beta\right)\right]$$

The Conjugate Gradient Method – Parallel Algorithm…

❑ Results of computational experiments…

– Comparison of theoretical estimations and results of computational experiments

Matrix Size	2 processors		4 processors		8 processors	
	Model	Experiment	Model	Experiment	Model	Experiment
500	1,3042	0,4634	0,6607	0,4706	0,3390	1,3020
1000	10,3713	3,9207	5,2194	3,6354	2,6435	3,5092
1500	34,9333	17,9505	17,5424	14,4102	8,8470	20,2001
2000	82,7220	51,3204	41,4954	40,7451	20,8822	37,9319
2500	161,4695	125,3005	80,9446	85,0761	40,6823	87,2626
3000	278,9077	223,3364	139,7560	146,1308	70,1801	134,1359

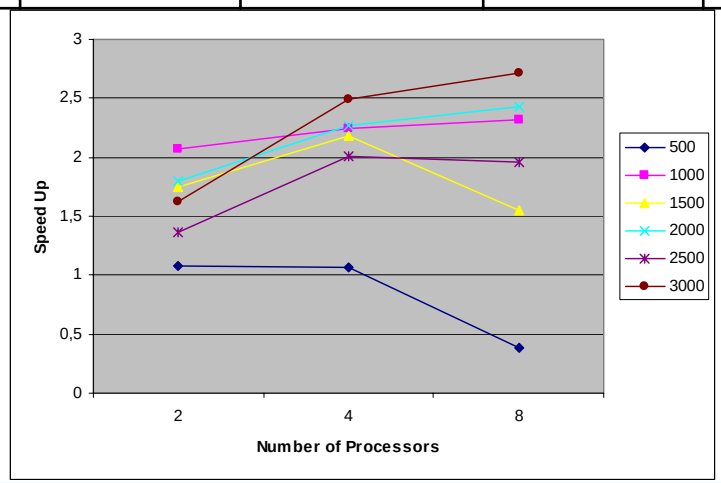


The Conjugate Gradient Method – Parallel Algorithm

❑ Results of computational experiments:

– Speedup

Matrix Size	Serial Algorithm	Parallel Algorithm					
		2 processors		4 processors		8 processors	
		Time	Speed Up	Time	Speed Up	Time	Speed Up
500	0,5	0,4634	1,0787	0,4706	1,0623	1,3020	0,3840
1000	8,14	3,9207	2,0761	3,6354	2,2390	3,5092	2,3195
1500	31,391	17,9505	1,7487	14,4102	2,1783	20,2001	1,5539
2000	92,36	51,3204	1,7996	40,7451	2,2667	37,9319	2,4348
2500	170,549	125,3005	1,3611	85,0761	2,0046	87,2626	1,9544
3000	363,476	223,3364	1,6274	146,1308	2,4873	134,1359	2,7097

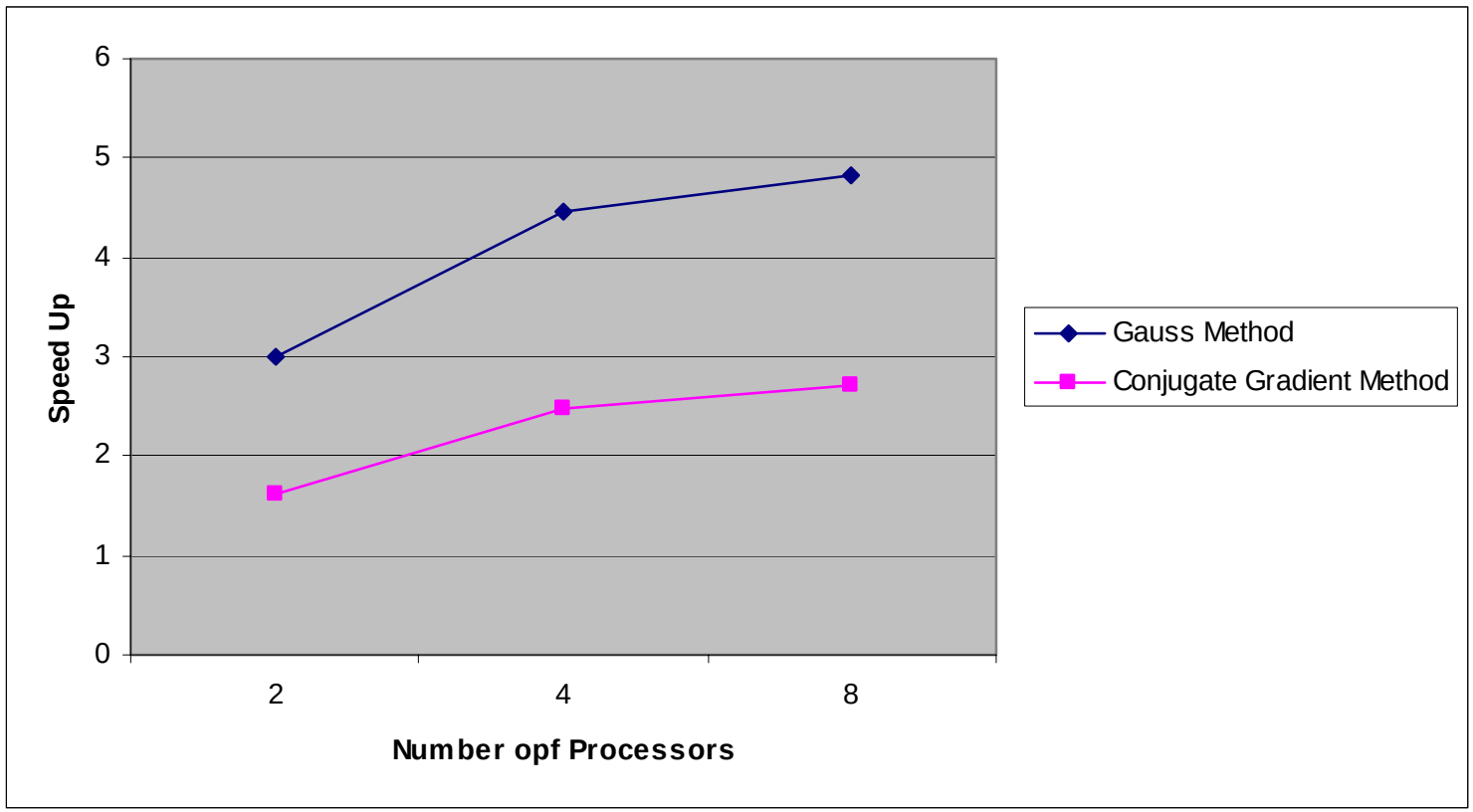


Summary…

- Two parallel algorithms for solving systems of linear equations are discussed:
  - The Gauss Method,
  - The Conjugate Gradient Method
- The parallel program sample for the Gauss algorithm is described
- Theoretical estimations and results of computational experiments show the greater efficiency of the Gauss method

Summary

- The speedup of the parallel algorithms for solving linear system of 3000 equations



Discussions

- What is the reason of such low estimations for speedup and efficiency of parallel algorithms?
- Is there a way to increase the speedup and efficiency characteristics?
- What algorithm has the greater computational complexity?
- What is the main advantage of the iterative methods for solving the linear systems?

Exercises

- Develop the parallel programs for the Gauss method based on columnwise block-striped decomposition
- Develop the parallel programs for the Jacobi and Zeidel method based on columnwise block-striped 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

- **Bertsekas, D.P., Tsitsiklis, J.N.** (1989). Parallel and distributed Computation. Numerical Methods. - Prentice Hall, Englewood Cliffs, New Jersey.
- **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 Sorting Methods**

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 in 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.