



University of Nizhni Novgorod  
Faculty of Computational Mathematics & Cybernetics

# *Introduction to Parallel Programming*

Section 15.

*Parallel Laboratory (ParaLab).*

*The Software System for Learning and Investigations  
of Parallel Computational Methods*

The Microsoft logo, consisting of the word "Microsoft" in white sans-serif font on a blue rectangular background.

**Microsoft**

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

# Contents

---

- ❑ System Overview
- ❑ Simulating the Parallel Computer
- ❑ Stating the Problem
- ❑ Choosing the Method
- ❑ Setting the Graphical Views
- ❑ Running the Experiments
- ❑ Accumulating and Analyzing the Results
- ❑ Saving the Data
- ❑ Summary



# Introduction

---

## ❑ **Parallel Laboratory (ParaLab)** provides:

- *Simulating multiprocessor computer systems* with various communication network topologies,
- *Executing* the numerous computational experiments for parallel solving the various time-consuming problems on the different parallel computer systems in the simulation mode,
- *Obtaining the visual views of the computational processes and data communication operations* which can be arisen in parallel computations,
- *Evaluating the efficiency estimations* of the parallel computational methods being studied

❑ In general ParaLab can be described as *an integrated software environment for studying and investigating* the parallel algorithms for solving time-consuming computational problems



# System Overview...

---

- ❑ **ParaLab** is a software system, which allows the user in a friendly environment to study and investigate parallel computations using only a single sequential computer – in addition all executed parallel operations are many-sided illustrated by a great number of graphic views and diagrams



# System Overview...

---

## □ ParaLab allows...

- To simulate a parallel computer system by choosing the *network topology* and its parameters, the *number of processors* and the *processor performance*,
- To state the *computational problem* to be solved and to set the problem *parameters*,
- To select the *parallel method* for solving the problem, that was formulated,
- To tune the *graphic views and diagrams* that illustrate parallel computations by setting the desirable rate of demonstration, visualization modes of communication operations, iterations at which calculations has to be visualized



# System Overview...

---

## ❑ ParaLab allows...

- To *execute the experiment* for parallel solving of stated problem:
  - several different experiments with various types of multiprocessor systems, problems or parallel methods can be taken for carrying out – all these experiments can be executed simultaneously in shared time mode,
  - a series of experiments, which sometimes require long computations, can be executed in the automatic mode – all calculated results can be accumulated in the "*experiment log*", which makes possible to analyze the obtained data



# System Overview...

---

## ❑ ParaLab allows:

- To *accumulate and analyze the results of the executed experiments*:
  - The accumulated data can be visualized in the several graphic forms - these graphs allow to demonstrate various dependences which can characterize parallel computations (execution time, speed up, efficiency) with respect to the problem complexity or the parameters of the computer system,
  - The required dependencies are evaluated for any combination of parameters of the computer system, the problem and the method by means of the Hockney model



# System Overview...

---

- ❑ All experiment data can be extracted from the experiment log to restore the previous state of the experiment – that allows to repeat the experiment again. Also it allows to continue computations from the suspended state
- ❑ All accumulated data, including parameters of the simulated computer system, the problem statement, the chosen method, the experiment log, can be saved in the file and then can be loaded for the next use





# System Overview

---

***The study and investigation provided by ParaLab allow the students to master the parallel computations and assist them to obtain the understanding how to design parallel algorithms for solving complex time-consuming problems in various areas of applications***



# Simulating the Parallel Computer...

---

- ❑ To simulate the computer system, it is necessary to determine:
  - the network topology,
  - the number of processors,
  - the performance of each processor,
  - the parameters of the communication network (latency, bandwidth, data transmission technique to be used).
- ❑ It should be noted that the computer system is assumed to be homogeneous, i.e. all the processors have the same performance, and all the communication channels have the same characteristics



# Simulating the Parallel Computer...

## Choosing the Network Topology...

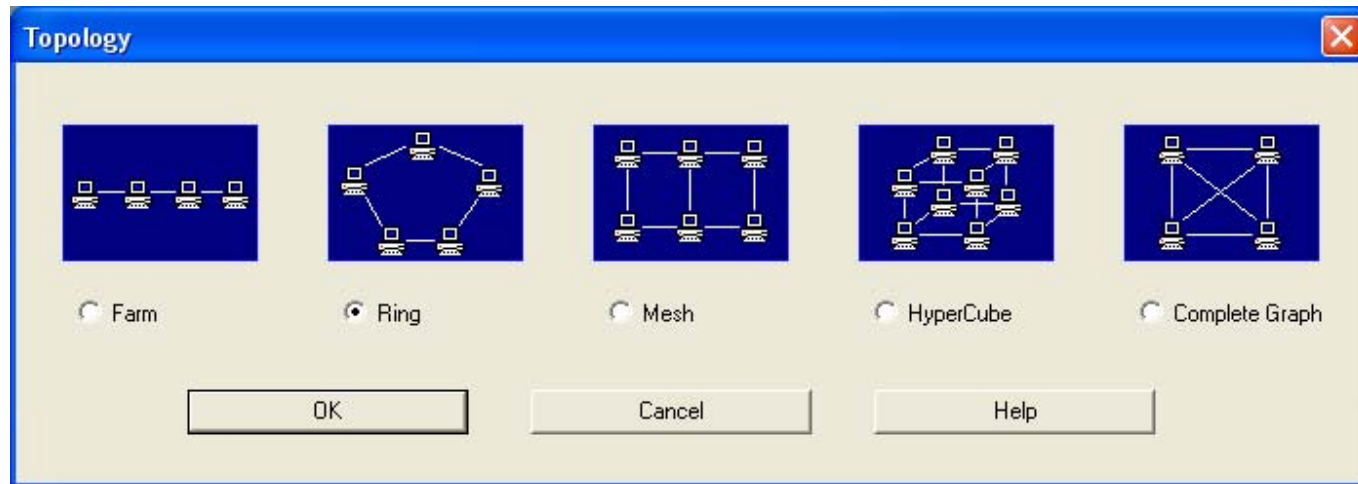
- ❑ The *network topology* is determined by the structure of communication links among the processors of the computer system
- ❑ ParaLab provides the following network topologies:
  - **Completely-connected graph or clique** – here there is a direct communication link between any pair of processors,
  - **Linear array or farm** – here each processor is connected by communication links only with two neighbors (with the previous and the following ones),
  - **Ring** – may be obtained from the linear array by connecting the first and the last linear array processors,
  - **Mesh** – the graph of communication links in this system forms a rectangular two-dimensional grid,
  - **Hypercube** – is a particular case of the N-dimensional grid structure; there are only two processors along each dimensionality of the grid



# Simulating the Parallel Computer...

## Choosing the Network Topology:

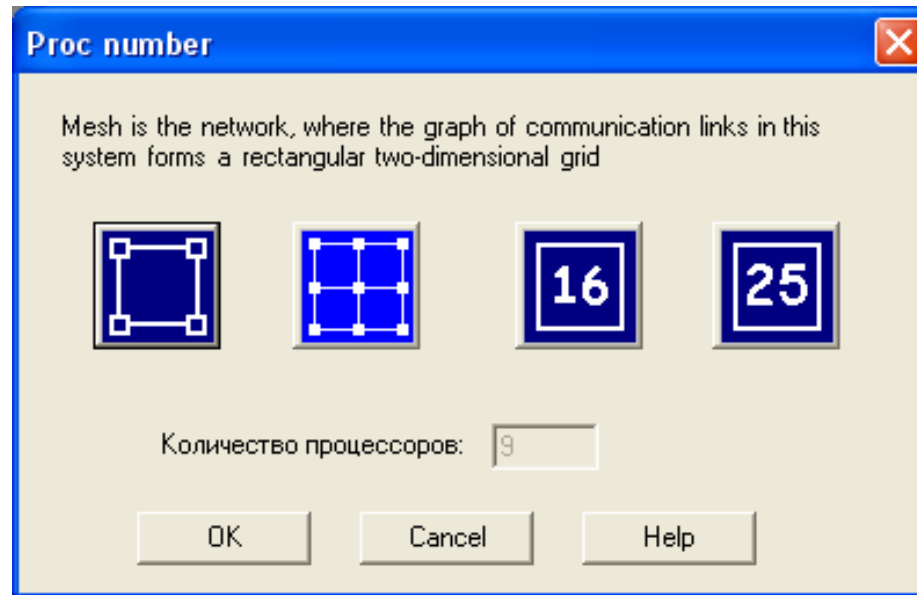
- ❑ To choose the desirable network topology:
  - Select the command **Topology** of the main menu **System**,
  - Point the desirable topology in the dialog window **Topology** by one click of the left mouse button,
  - Press **OK** to confirm the choice and **Cancel** to return to the main menu without selecting a new topology



# Simulating the Parallel Computer...

## Setting the Number of Processors:

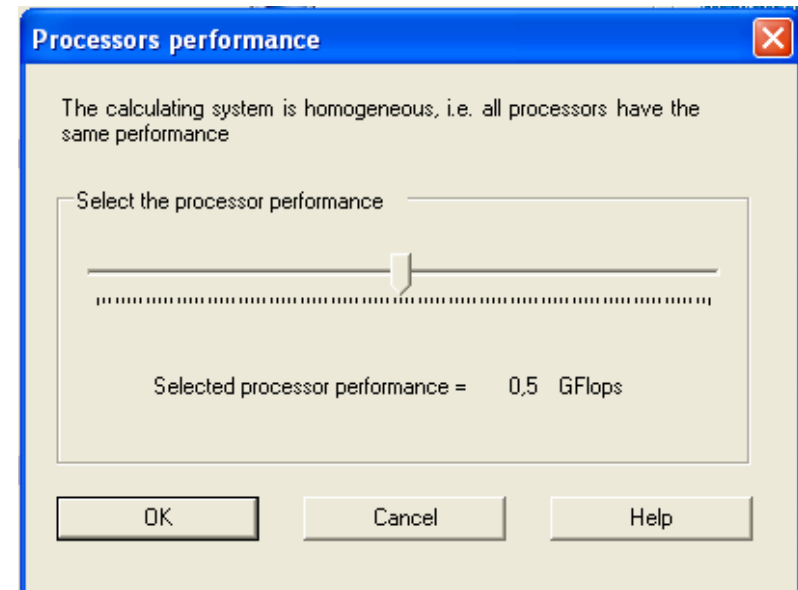
- ❑ To set the desirable number of processors, it is necessary to select the command **Number of processors** of the main menu **System**, then to point the icon with the needed number of processors by clicking the left mouse button



# Simulating the Parallel Computer...

## Setting the Processor Performance:

- ❑ The processor performance is set by the number of floating point operations per second – **flops**
- ❑ It is assumed that the performance of all processors are identical and the execution time for all processor instructions is equal
- ❑ To set the performance of the processors the command **Processor Performance** of the main menu **System** has to be selected; then at the dialog window **Processor Performance** the desirable value of the performance can be set by the track bar



# Simulating the Parallel Computer...

## Setting Network Parameters...

*The time of transmitting data among the processors determines the **communication overhead** of the parallel algorithm execution on a multiprocessor system*

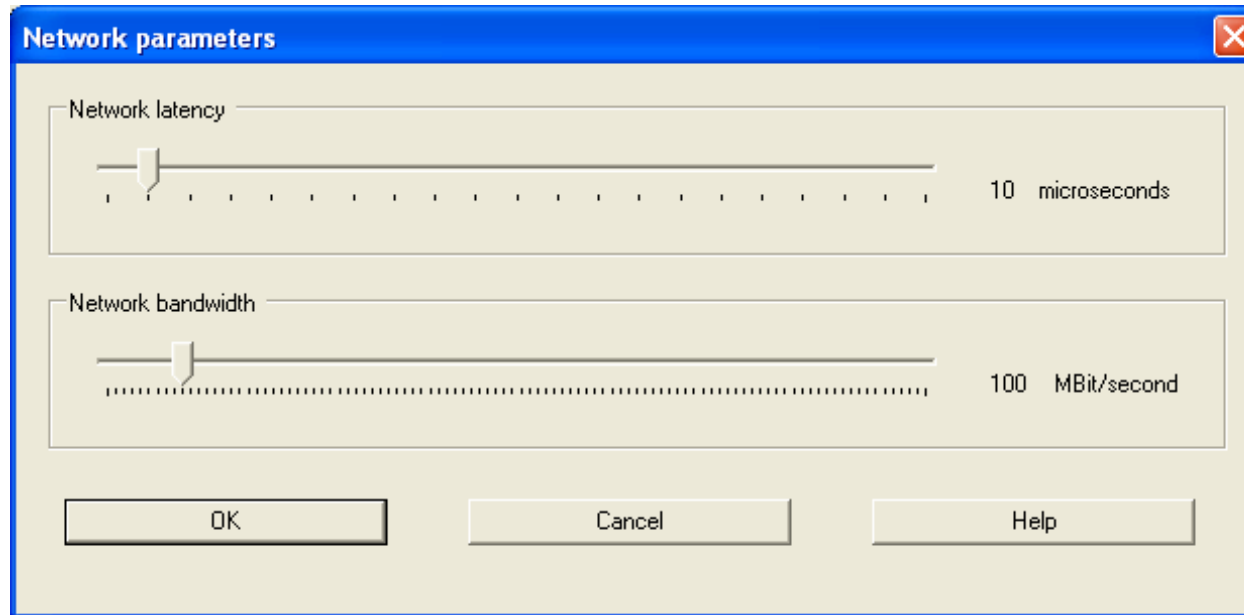
- Parameters that can be used to evaluate communication overhead:
  - *Latency* ( $\alpha$ ) - is the network initialization time, which includes the duration of preparing a message for transmission, as well as the duration of the searching for the route in the network, etc.,
  - *Network bandwidth* ( $\beta$ ) - is defined as the maximum amount of data, which can be transmitted in a certain unit of time over a data transmission channel. This parameter is measured, for instance, in bits per second



# Simulating the Parallel Computer...

## Setting Network Parameters...

- ❑ To set the parameters of the network, the command **Network parameters** of the menu **System** has to be selected; then the desirable values can be set by the track bars at the dialog window **Network parameters**





# Simulating the Parallel Computer...

---

## Setting Network Parameters...

- There are two well-known communication methods, that are implemented in ParaLab: *store-and-forward routing (SFR)* and *cut-through routing (CTR)*



# Simulating the Parallel Computer...

## Setting Network Parameters...

- ❑ *Store-and-forward routing* - the processor, which contains the source message, prepares all the amount of data for transmission, determines the transit processor, via which the data may be delivered to the target processor, and starts the data transmission operation. The processor, which is to receive the message, performs, first of all, the receiving of all the transmitted data, and only then starts to transmit the data further along the route
- ❑ As a result the communication time  $T$  for transmitting the message of size  $m$  along the route of length  $l$  is determined by the following expression (in accordance with the Hockney model):

$$T = (\alpha + m / \beta) \cdot l$$



# Simulating the Parallel Computer...

## Setting Network Parameters...

- ❑ *Cut-through routing* – in this case the transmitted message is subdivided onto several *packets*; the transit processor may send transmitted data along the chosen route immediately after the receiving a packet without waiting for the termination of receiving all the message packets:

- The number of the transmitted packets is equal to the following:

$$n = \left\lceil m / (V - V_0) \right\rceil + 1,$$

where  $V$  is the packet size, and the value  $V_0$  determines the amount of the control data transmitted in each packet (*the packet header*),

- As a result, in this case the communication time is equal to the following:

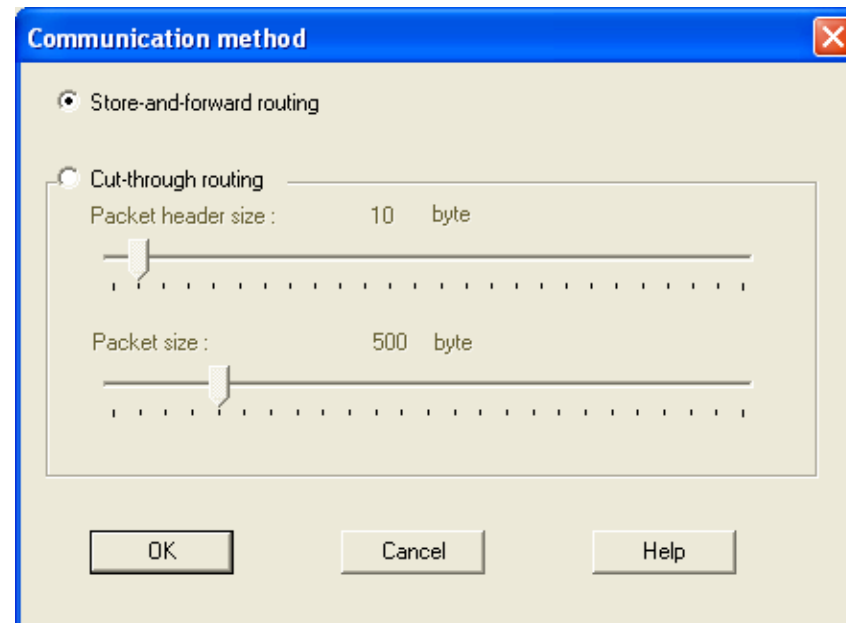
$$T = \alpha + \frac{V}{\beta} \left( l + \left\lceil \frac{m}{V - V_0} \right\rceil \right) = \alpha + \frac{V}{\beta} (l + n - 1)$$



# Simulating the Parallel Computer

## Setting Network Parameters:

- ❑ To choose the data communication method to be used in the computational experiments and to set the parameters, select the command **Communication Method** of the menu **System**; then set the desirable values with the help of the track bars at the dialog window **Communication Method**



# Stating the Problem...

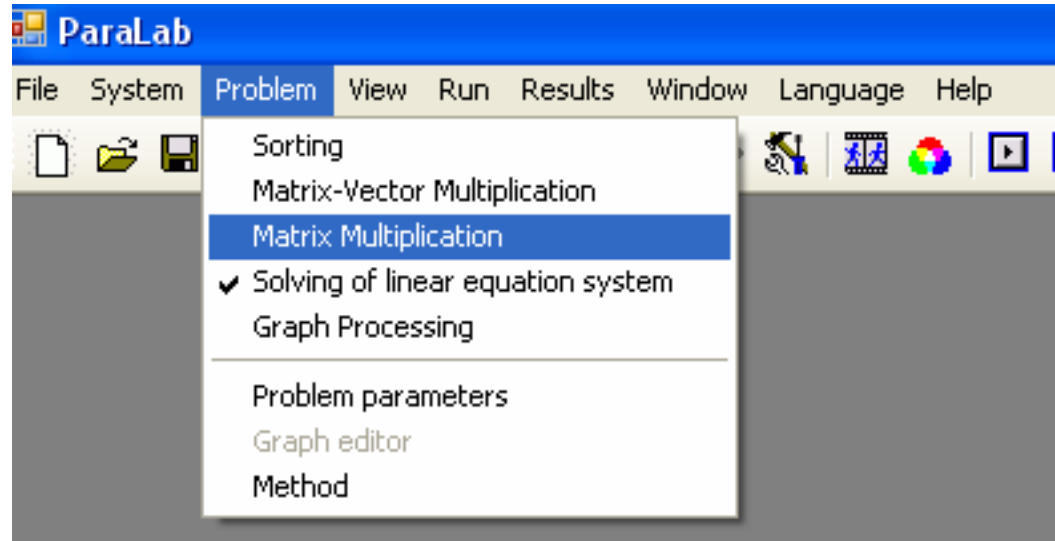
---

- ❑ To demonstrate parallel computations ParaLab includes well-known parallel algorithms applied to solving some typical time-consuming problems, that arise in various research and industrial applications:
  - Parallel sorting methods,
  - Parallel methods for matrix calculations (matrix-vector multiplication and matrix multiplication),
  - Parallel methods for solving linear systems,
  - Parallel methods for graph computations.



# Stating the Problem...

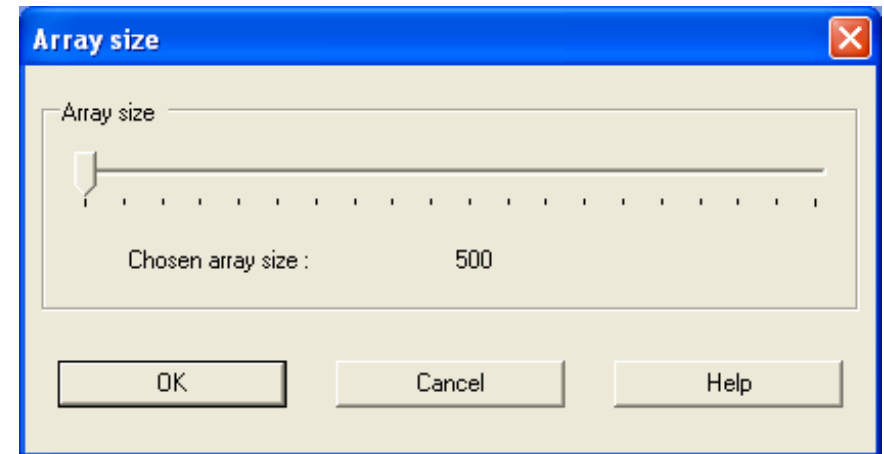
- ❑ To choose a problem select the menu **Problem**; then set the desirable problem by one click on the menu items **Sorting**, **Matrix-Vector Multiplication**, **Matrix multiplication**, **Solving of linear system**, **Graph processing**. The selected problem becomes the current one at the active window



# Stating the Problem...

## Setting the Problem Size

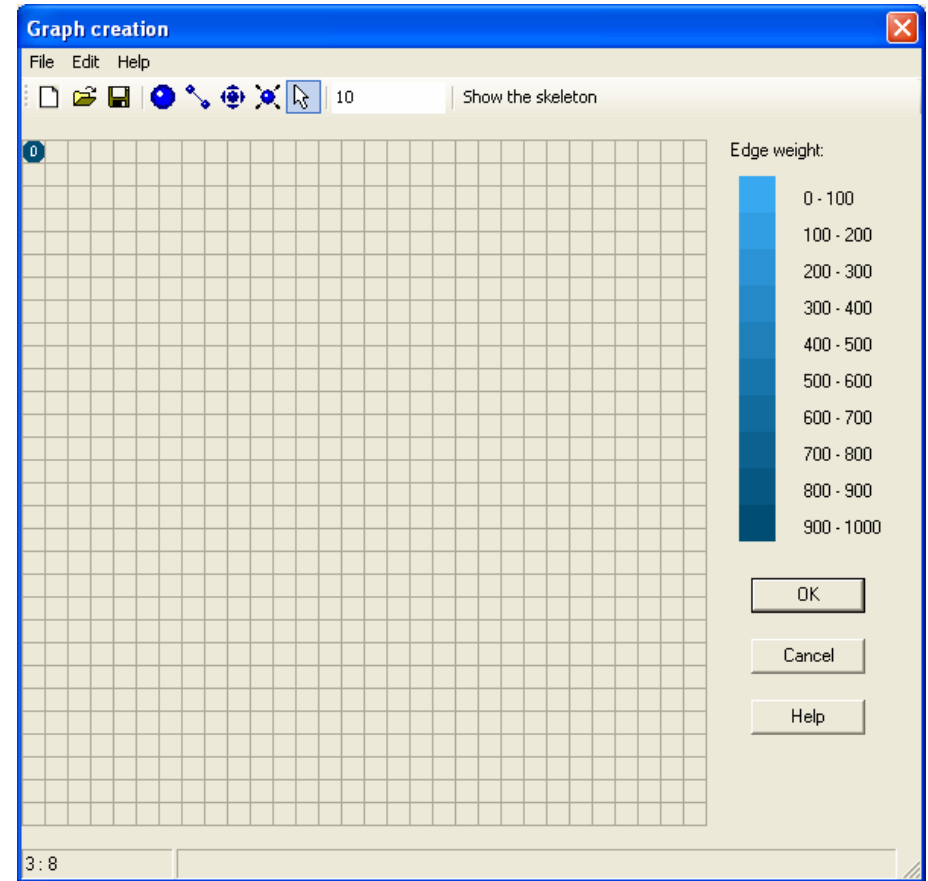
- ❑ The complexity of the selected problem can be regulated by setting the desirable size of initial data  **$N$** :
  - For the sort problem  $N$  is the number of the sorted values,
  - For the matrix operations and the problem of solving a system of linear equations  $N$  is the order of the matrices,
  - For the graph computation problem  $N$  is the number of graph vertices.
- ❑ To select the problem size it is necessary to select the command **Problem parameters** of the menu **Problem**, then to set the desirable value by the track bar at the dialog window



# Stating the Problem...

## Graph Editor...

- ❑ In case of choosing the problem of graph processing ParaLab provides several ways to obtain the graph – it can be created by a random generator, edited, stored in the file and loaded from the file. All these possibilities are supported by the graph editor – to activate it execute the command **Graph Editor** of the menu **Problem**





# Stating the Problem

---

- ❑ **Graph Editor** provides the possibility of:
  - Creating a new graphs,
  - Editing the existing graphs:
    - Adding and removing vertices,
    - Adding and removing edges,
    - Moving vertices,
    - Setting the weight of the edges
  - Storing the graph in a file, and loading a graph from the file,
  - Forming a graph by a random generator



# Choosing the Method...

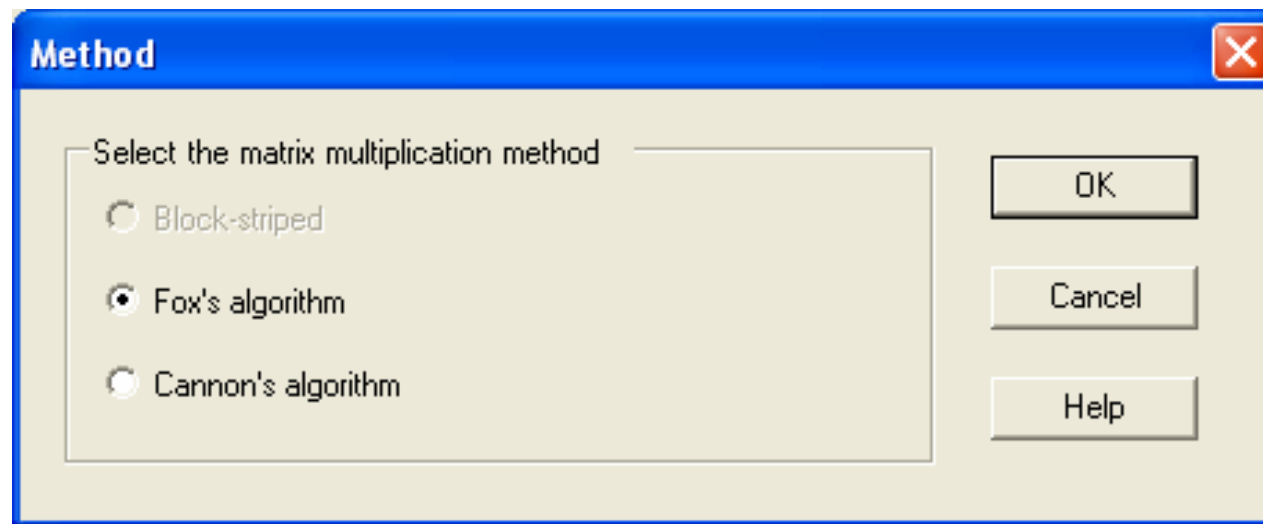
---

- ❑ For parallel solving the process of computations should be presented as a set of independent computational procedures, which may be distributed among processors and executed in parallel
- ❑ In general such computations can be implemented in the following way:
  - First of all the computational process has to be splitted into parts (*subtasks*), which may be executed simultaneously,
  - Then it is necessary to distribute the subtasks among the processors,
  - As a result, the subtasks can be executed in parallel; communications between subtasks, that are required during computations, can be provided by data transmission operations



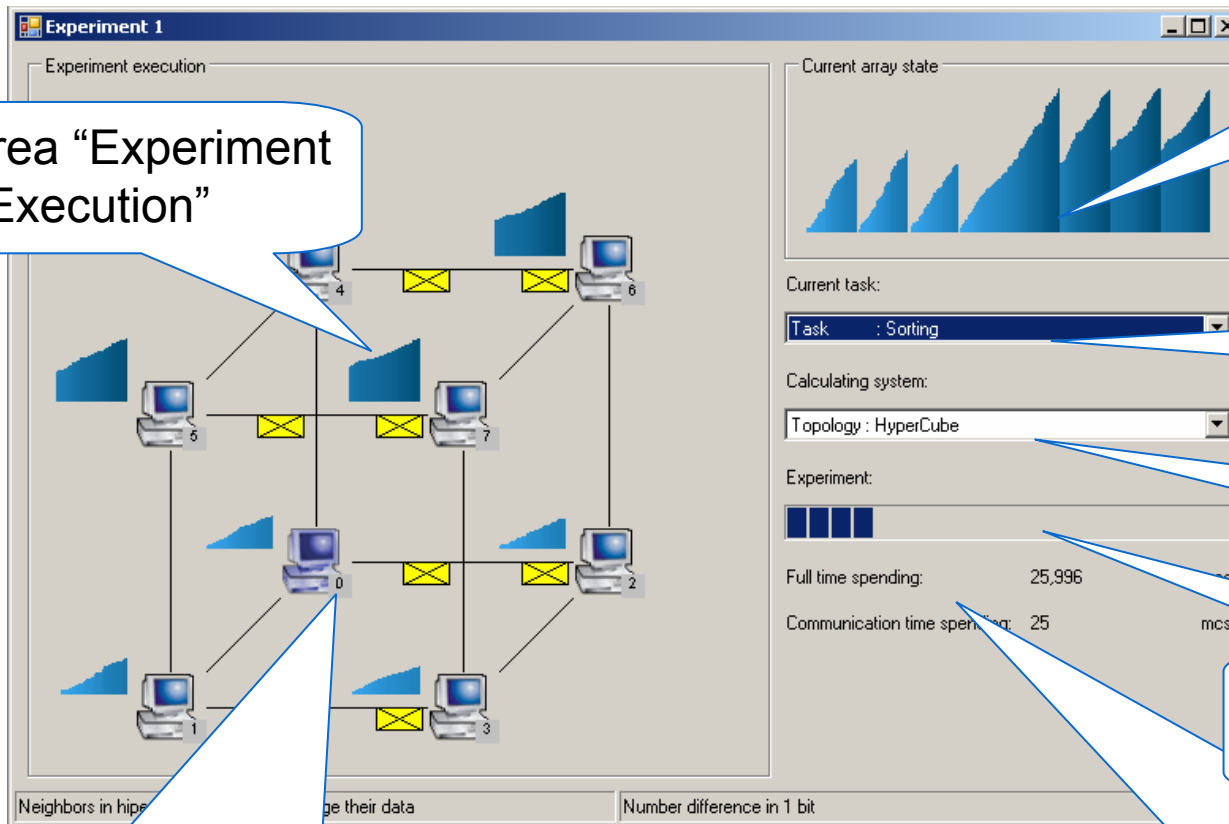
# Choosing the Method

- ❑ To choose the method, the command **Method** of the menu **Problem** has to be selected; then the desirable method can be set by choosing one from the available alternatives at the dialog window **Method**



# Setting the Graphical Views...

## ❑ The Window of the Computational Experiment



The Area "Experiment Execution"

The current state of the experiment results

Problem parameters

System parameters

The progress bar of the experiment execution

Active processor  
(icon is highlighted with the blue color)

Time characteristics of the experiment execution



# Setting the Graphical Views...

## □ The Area “Experiment Execution”...

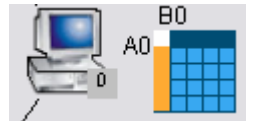
- In this window area the processors of the multiprocessor system are shown, they are connected by communication links according to the chosen topology,
- The processors are enumerated – to observe the processor number indicate the processor icon by the mouse pointer,
- One of the processors of the multiprocessor system is active, its icon is highlighted with the blue color,
- Calculations performed by one of the processors can be visualized – to activate this mode double click by the left mouse button on the processor icon, as a result the window **Demonstration of the processor calculations** would appear



# Setting the Graphical Views...

## The Area “Experiment Execution”...

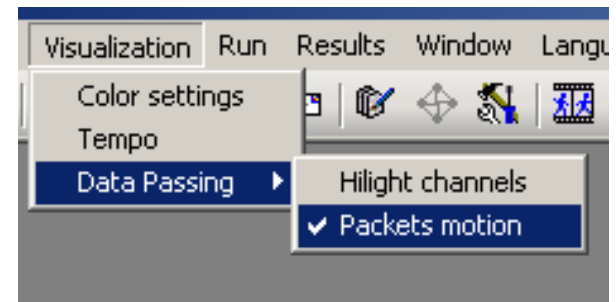
- ❑ The data located on a processor at any moment of the experiment execution is schematically shown at each processor:
  - **Sorting algorithms** - a parts of the sorted array are visualized near the processor icons. Each array element is shown by a vertical line. The line height and its color intensity depends on the value of the element - the higher and darker the line is the greater the element is,
  - **Matrix operations** and **Solving of linear systems** - a matrix frame is shown near every processor; the parts of the initial data located on the processor are given in different colors,
  - **Graph processing** - a subgraph, which consists of the vertices located on the processor, is displayed near each processor



# Setting the Graphical Views...

## The Area “Experiment Execution”...

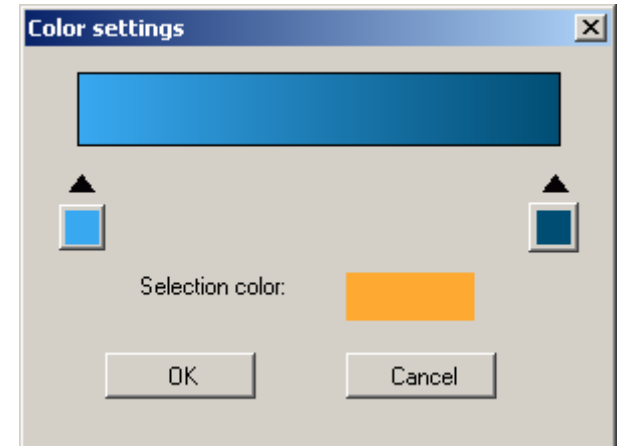
- ❑ In the course of carrying out the experiment the data exchange among the processors is also demonstrated in the area “Experiment execution”. It may be shown in two modes:
  - The **Channel View** mode– the line, along which the data is transmitted, is marked by the red color,
  - The **Packet View** mode– messages transmitted between processors are demonstrated by the moving envelope pictogram. When parallel matrix multiplication is carried out, the number of the transmitted block is shown in the envelope icon
- ❑ To set the method of displaying the processor communication, choose the alternative **Channel View** or **Packet View** of the command **Demo mode** from the menu **View**



# Setting the Graphical Views...

## The Area “Experiment Execution”:

- ❑ To display only certain iterations in the **Experiment Execution** area, the command **View Step** of the menu **View** should be selected (the command is accessible only when the problem of graph processing is the current one in the active window, in this case the algorithm executes a great number of identical iterations)
- ❑ To change the colors, which are used by ParaLab to visualize the parallel computations, the command **Colors** of the menu **View** has to be selected





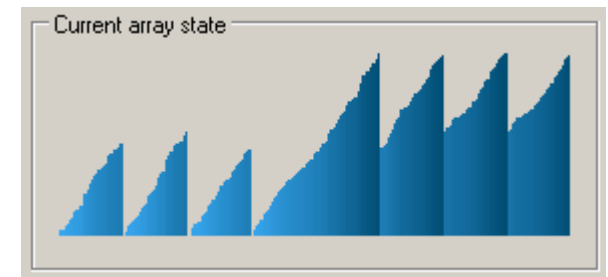
# Setting the Graphical Views...

## The Area “Current Array State” (in case of sorting):

- ❑ This area displays the sequence of the sorted array elements. As in the area “Experiment execution”, each element is displayed by a vertical line. The height and the intensity of color depends on the element value. The higher and darker the line is, the greater the element is
- ❑ The sorted data is distributed among processors by blocks. The result array is formed by blocks located on processors with the successively increasing numbers

$A = [A_0, A_1, \dots, A_p]$ ,  $A_i$  is the block, located on the processor  $i$

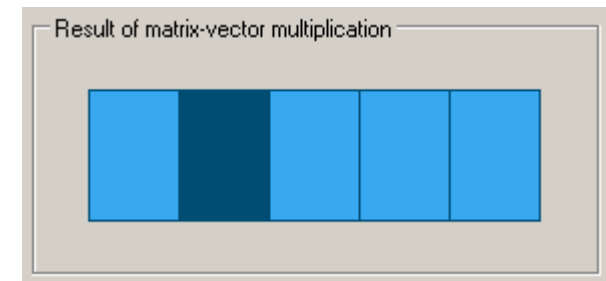
- ❑ After the execution of sorting, the array blocks on each processor has to be sorted and, besides, the elements located on the processor with a smaller number must not exceed the elements, located on the processor with a greater number



# Setting the Graphical Views...

## The Area “The Result of Matrix-Vector Multiplication”...

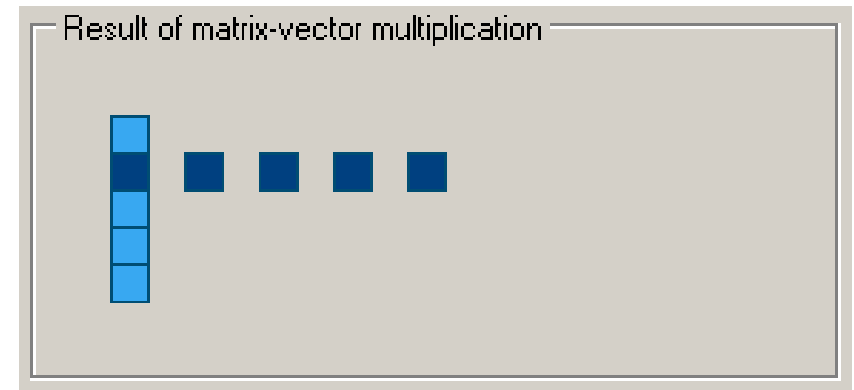
- ❑ Algorithm based on the *rowwise block-striped* matrix decomposition (the vector  $b$  is copied on each processor):
  - each processor computes a block of the result vector by multiplying a row of matrix  $A$  and the vector  $b$ ,
  - The block computed on the active processor is displayed by the dark blue color,
  - After completing the calculations, blocks of the result vector are accumulated on each processor (the color of all blocks become dark blue)



# Setting the Graphical Views...

## The Area “The Result of Matrix-Vector Multiplication”...

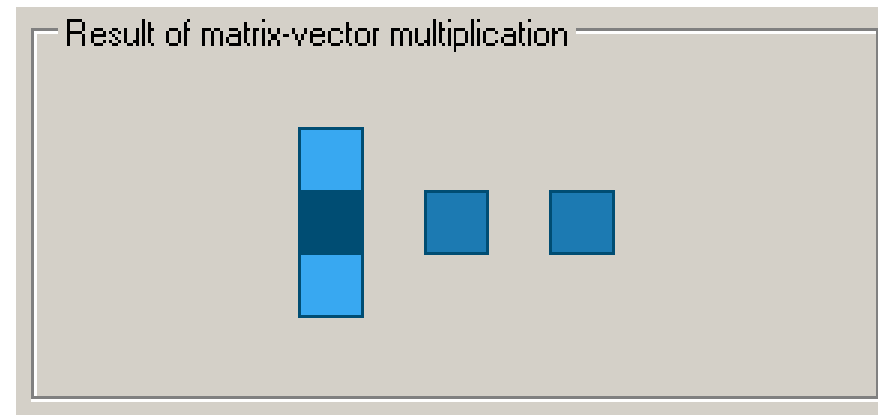
- ❑ Algorithm based on the *columnwise block-striped* matrix decomposition (the vector  $b$  is distributed among processors):
  - each processor computes the vector of partial results multiplying a matrix row strip by a block of the vector  $b$ ,
  - All the blocks of the resulting vector in the area **The result of matrix-vector multiplication** are highlighted by the light blue color. After completing the calculations, all the partial results are exchanged in such a way that blocks of the result vector can be computed. The block of the active processor is displayed in the area **The result of matrix-vector multiplication** by the dark blue color



# Setting the Graphical Views...

## The Area “The Result of Matrix-Vector Multiplication”:

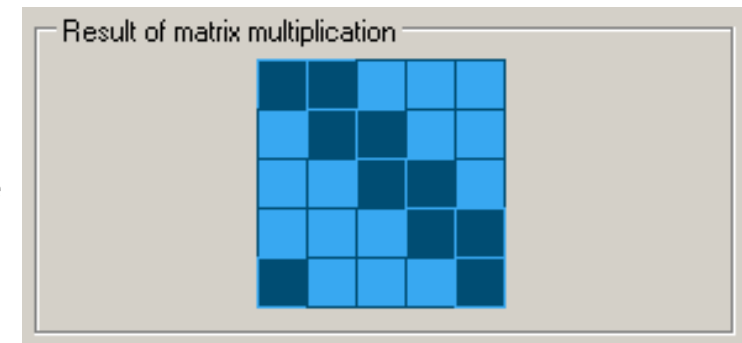
- ❑ Algorithm based on the *checkerboard* matrix decomposition:
  - Matrix  $A$  is distributed among the processors of the processor grid. Vector  $b$  is distributed among the processors, that form the columns of the processor grid,
  - After multiplying a block of matrix  $A$  by a block of vector  $b$ , each processor computes the block of the partial results. It is highlighted by the light blue color,
  - After exchanging blocks within each row of the processor grid, processors compute blocks of the result vector



# Setting the Graphical Views...

## The Area “Result of Matrix Multiplication”...

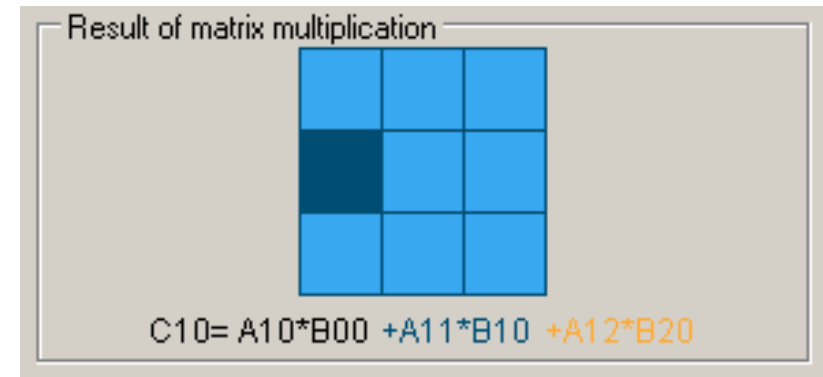
- The result matrix  $C$  is presented as a grid of rectangular blocks. Each processor is responsible for computing one block (the Fox and Cannon algorithms) or a row of blocks (the block-striped algorithm) of the result matrix  $C$ :
  - *The block-striped algorithm* - the blocks, which have already been computed, are marked by the dark blue color



# Setting the Graphical Views...

## The Area “Result of Matrix Multiplication”:

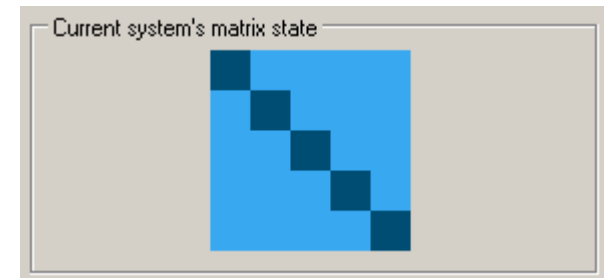
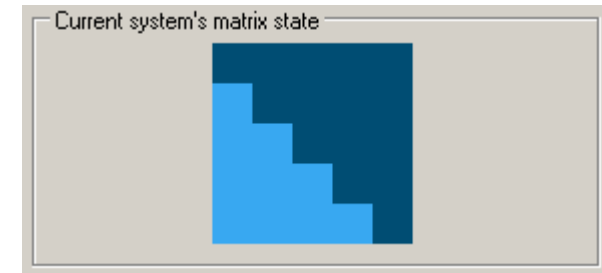
- In case of *the Fox or Cannon algorithms*, all the blocks of the result matrix  $C$  are computed simultaneously, and none of them can be computed until all the method iterations are performed,
- As a result the area **Result of Matrix Multiplication** is used only to visualize calculations of the active processor. The elements of the multiplication formula, which have already been computed, are written by the dark blue color, and the formula element, which is being computed at the current iteration, is marked by the red color



# Setting the Graphical Views...

## The Area “The Result of Solving the Linear Equation System”:

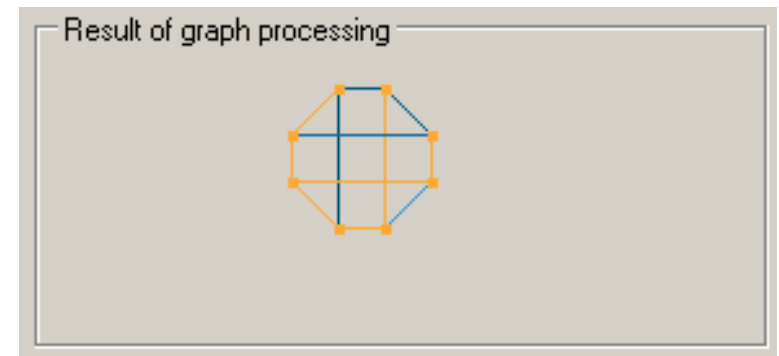
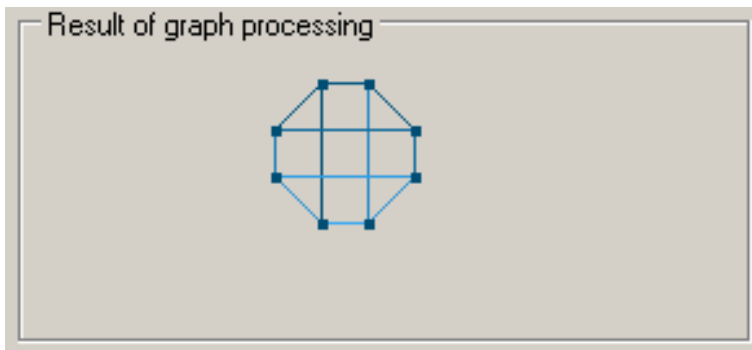
- This area shows the current state of the linear equation system in the course of the executing the Gauss elimination algorithm:
  - The dark blue color marks the non-zero elements, while the light blue color – the zero ones,
  - After completing the Gauss elimination all matrix elements located below the main diagonal are equal to zero,
  - After completing the back substitution all non-zero elements are located on the main diagonal



# Setting the Graphical Views...

## The Area “Result of Graph Processing”:

- ❑ This area shows the current state of the graph:
  - The graph edges are shown in different colors: the darker the color is, the greater the edge weight is,
  - In the process of the algorithm execution the red color is used to mark the vertices and the edges, which have already been included into the minimum spanning tree (the Prim algorithm) or into the tree of the shortest paths (the Dijkstra algorithm)





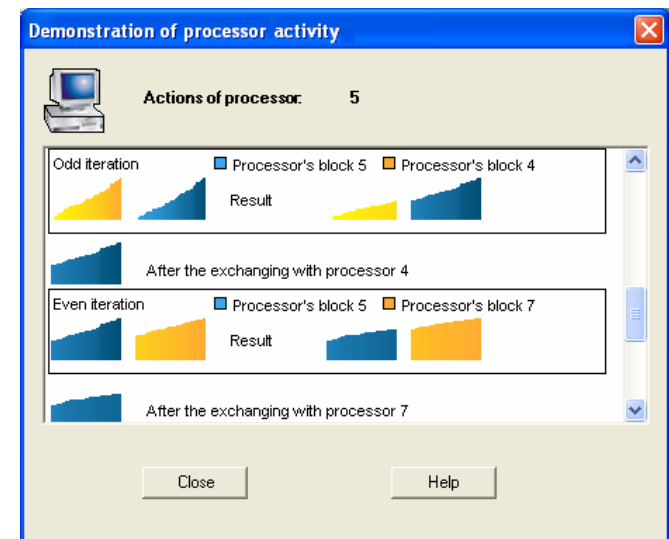
# Setting the Graphical Views

## Visualizing the Processor Calculations:

- ❑ ParaLab provides the possibility to visualize the computations executed by a single processor being selected in the separate window

### Demonstration of Processor Calculations:

- To choose a processor which calculations have to be demonstrated the command **Processor View** of the menu **View** has to be selected,
- The desirable processor can be selected by the double click of the left mouse button on the processor icon



# Running the Experiments...

- ❑ ParaLab provides different computation schemes for running the experiments. Problems can be solved:
  - In the ordinary sequential execution mode,
  - In the step-by-step mode (with suspending the calculations after each step),
  - In the time sharing mode for running all formulated experiments simultaneously.

Several experiments (*series*) can be executed sequentially in the same window in the automatic mode. At the time of starting the series, the parameter (the problem size or the number of processor) has to be selected – this parameter will be varied for every executed experiment in the series. All obtained results in the series of experiments can be stored in the experiment log for the next observation and analysis



# Running the Experiments...

---

## □ Sequential Mode:

- To run a computational experiment in this mode, the command **In the active window** of the menu **Run** should be executed,
- To suspend the executed experiment select the **Stop** command,
- To resume the previously suspended experiment the command **Continue** should be executed



# Running the Experiments...

---

## ❑ Step-by-step Mode:

- To start the experiment in the step-by-step mode the command **Step-by-step** of the menu **Run** has to be **selected**. The menu of this mode holds the commands:
  - The command **Step** that is used to execute the next iteration of the calculations,
  - The command **Run** that is used to start running the experiment without suspending after every iteration,
  - The command **Stop** that is used to terminate the step-by-step mode and return to the main menu



# Running the Experiments...

---

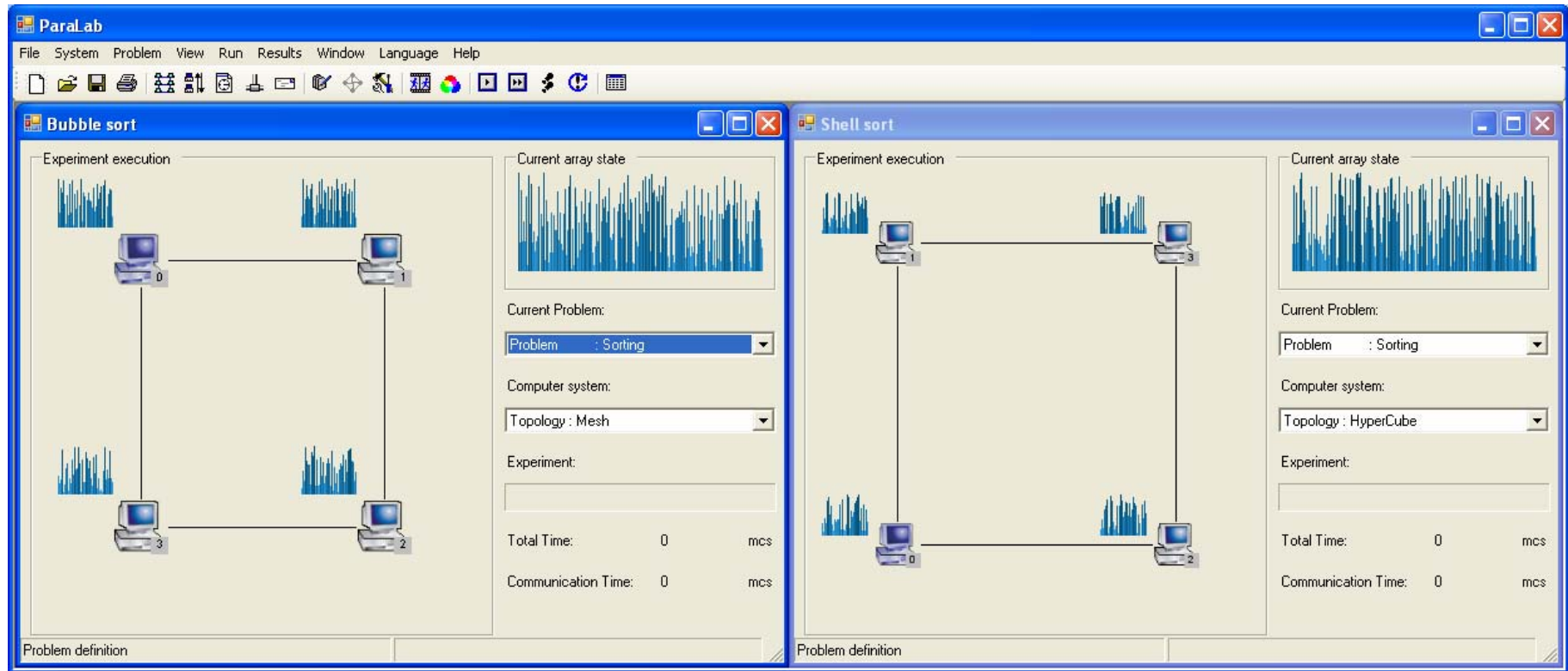
## □ Running a Set of Experiments Simultaneously...

- Running the experiments in this mode allows to visualize simultaneously the results of all the executed experiments,
- To run the computational experiments in all available windows in the time sharing mode (i.e. the next iteration is executed only after ending the current one in all available windows) the command **In all windows** of the menu **Run** has to be selected



# Running the Experiments...

## ❑ Running a Set of Experiments Simultaneously



# Running the Experiments...

---

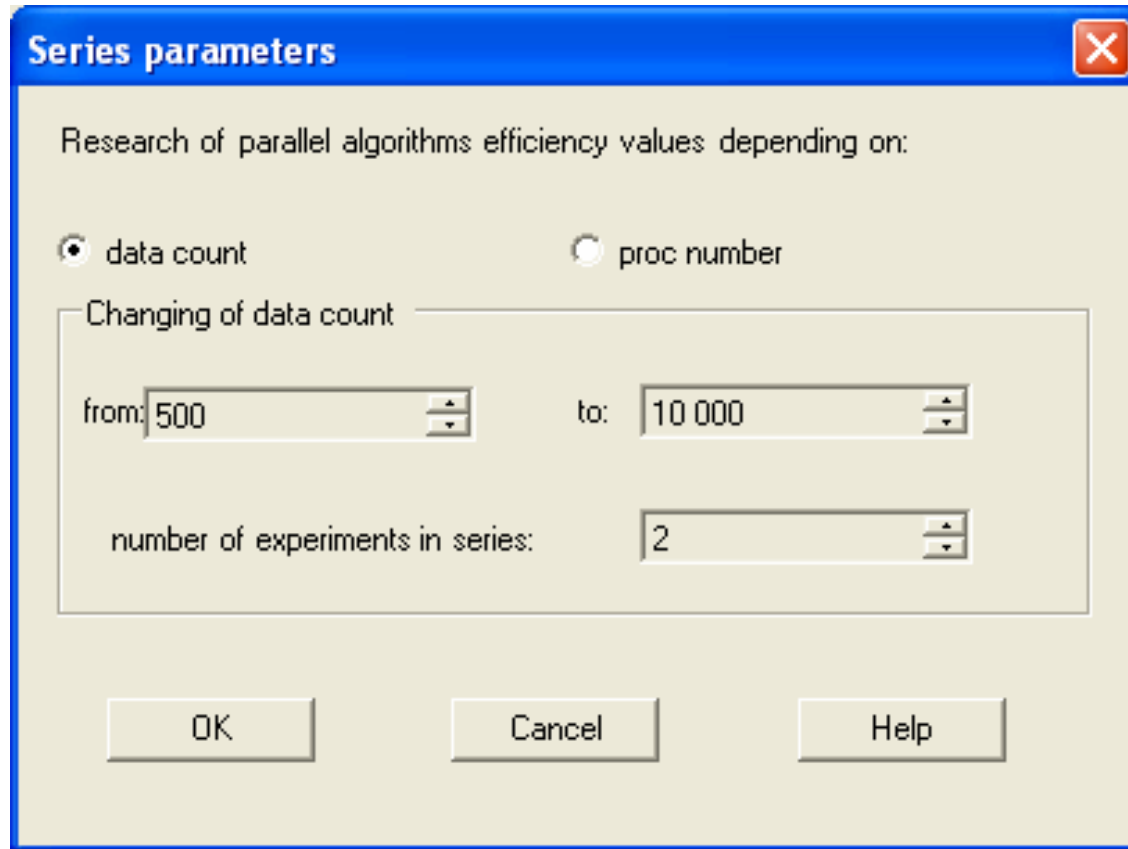
## Running the Series of Experiments...

- ❑ ParaLab provides the opportunity of running several experiments (*series*) sequentially in the same window in the automatic mode. To start running the experiments in this mode, it is necessary:
  - To choose the window, in which the experiments will be executed,
  - To set the number of experiments and select the parameter (the problem size or the number of processors), that will be varied in each experiment of the series
- ❑ The results of the experiments can be stored in the experiment log for the next observation and analysis



# Running the Experiments

## Running the Series of Experiments



**Series parameters**

Research of parallel algorithms efficiency values depending on:

☒ data count ☐ proc number

Changing of data count

from: 500 to: 10 000

number of experiments in series: 2

OK Cancel Help



# Accumulating and Analyzing the Results...

---

- ❑ Executing the experiments for studying various parallel algorithms of solving complicated computational problems requires in many cases long-continued computations
- ❑ Moreover a great number of experiments have to be performed to come to reliable conclusions
- ❑ ParaLab has various means for accumulating the results of the computational experiments and for presenting the data in the table and graphical forms, which are suitable for analysis



# Accumulating and Analyzing the Results...

## The General Experiment Results...

- ❑ ParaLab accumulates the experimental results automatically
  - ❑ For each experiment the following information is saved:
    - The date and the time when the experiment was executed,
    - The parameters of the computer system that was simulated,
    - The parameters of the problem that was solved,
    - The parameters of the method that was chosen,
    - The time of the experiment execution and the time of communication operations that were executed within the experiment.
- Data of needless experiments can be removed, it is possible to erase all accumulated results
- ❑ Every previously executed experiment can be restored in the active window to repeat or proceed with computations with the help of saved data



# Accumulating and Analyzing the Results...

## The General Experiment Results...

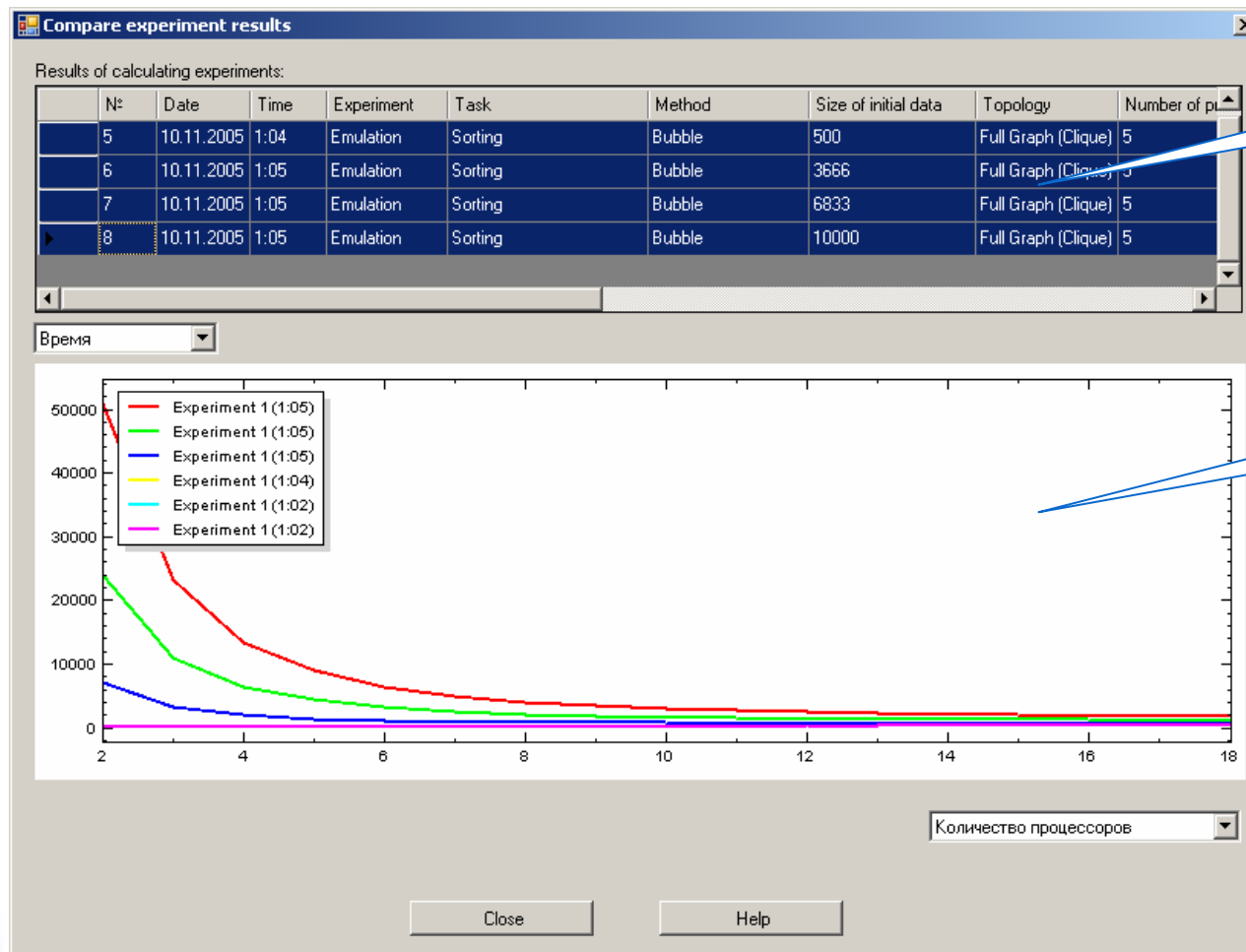


Table of Results

Graphical Board

# Accumulating and Analyzing the Results...

---

## The General Experiment Results:

- ❑ For dependencies that are shown in the graphical board user can select:
  - For the vertical axis – the time of the experiment execution or speedup of parallel computations,
  - For the horizontal axis – the problem size, the number of processor, the processor performance, the latency or the bandwidth of the network



# Saving the Data...

---

- ❑ The results of the experiments can be saved in the ParaLab archive. The data stored for the active window includes:
  - The parameters of the computer system (the topology type, the number of processors, the processor performance, the latency and the network bandwidth, the data transmission method),
  - The parameters of the stated problem and the chosen method,
  - The data of the experiment log



# Saving the Data

---

- ❑ To save the experiment results the command **Save** of the menu **File** has to be selected - the data are saved in the file with the name extension **.prl**
- ❑ The data saved in the ParaLab archive can be restored at any moment by the command **Open** of the menu **File**



# Summary...

- ❑ **Parallel Laboratory (ParaLab)** is *an integrated software environment for studying and investigating* the parallel algorithms for solving complicated time-consuming problems
- ❑ ParaLab provides:
  - *Simulating multiprocessor computer systems* with various data communication network topologies,
  - *Executing* the numerous computational experiments for parallel solving the various time-consuming problems on the different parallel computer systems in the simulation mode or by the remote access via Internet,
  - *Obtaining the visual views of the computational processes and data communication operations* which can be arisen in parallel computations,
  - *Evaluating the efficiency estimations* of the parallel computational methods being studied



# Summary

---

***The study and investigation provided by ParaLab,  
allow the students to master the parallel  
computations and assist them to obtain the  
understanding  
how to design parallel algorithms  
for solving complex time-consuming problems  
in various areas of applications***





# References

---

- ❑ **Gergel, V.P., Strongin, R.G.** (2001, 2003 - 2 edn.). Introduction to Parallel Computations. - N.Novgorod: University of Nizhni Novgorod (In Russian)
- ❑ **Kumar V., Grama A., Gupta A., Karypis G.** (1994). Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)



# 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)



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.

