The Ministry of Education and Science of the Russian Federation

Lobachevsky State University of Nizhni Novgorod

Computing Mathematics and Cybernetics faculty

The competitiveness enhancement program
of the Lobachevsky State University of Nizhni Novgorod
among the world's research and education centers

Strategic initiative

Achieving leading positions in the field of supercomputer technology and high-performance computing

Computational mathematics and cybernetics faculty

APPROVED

Dean of the Computational mathematics
and cybernetics faculty

________________ V. Gergel

"_____"__________________2014.

Course program

Introduction to parallel programming

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.Б12

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Purposes of mastering the discipline

The need for solving complex applied problems requiring large-scale computations and essentially limited maximum performance of classical computers (von Neumann machines) gave rise to multiprocessor systems. The use of such systems helps to a large extent increase the computer performance at any existing level of hardware development. However, this requires parallel generalization of the traditional sequential pattern of problem solving. Thus, for multiprocessor systems, numerical methods should be developed as systems of parallel interacting processes allowing for running on independent processors. The applicable algorithmic languages and systemware must enable development of parallel programs and ensure synchronization and mutual exclusion of asynchronous processes etc.

Supercomputer technologies and high performance computing based on parallel computers are becoming an increasingly important technical development factor; they are becoming omnipresent.

The mentioned issues form the subject matter of this course. The purpose of this course is to study mathematical models, methods and technologies of parallel programming for multicore and multiprocessor computers to the extent ensuring a successful start in the field of parallel programming. The proposed set of knowledge and skills forms a theoretical basis for method of complex program development and includes such topics as purposes and objectives of parallel data processing , construction principles for parallel computing systems, modeling and analysis of parallel computations, development principles for parallel programs and algorithms, systems for parallel programming and parallel numerical algorithms for solving standard computing mathematics problems.

2. Discipline in the concentration program

This course is intended for third year students and is taught in the 5th and 6th semesters. This is a base discipline of the professional curriculum.

Being originally graduate-oriented, this course is also intended for teachers and post-graduates in the field of parallel programming. This course involves basic knowledge and skills in structural, modular and (in some sections) object-oriented programming. The basic programming language is C/C++. Many ideas may be successfully used for another programming language that supports parallelism, especially Fortran.

3. Requirements to discipline mastery

Learning this discipline involves forming of the following competencies:

- ability to acquire, generalize and analyze the information (GC1) to the following extent:
    - ability to speak and write in a logical, reasoned and clear way;
- capacity for intellectual, cultural, moral, physical and professional development and self-improvement (GC2) to the following extent:
    - ability to persistently pursue objective subject to moral and legal norms and obligations;
    - capacity for continuous improvement of their professional and cultural level;
- ability to understand and put into practice the information theory as the fundamentals of information technologies (PC1)
- readiness to join the professional community (PC2)

- Ability to understand and use the existing mathematical tools and laws of natural science (PC3)
- Ability to understand, develop and use state-ot-the-art information technologies (PC4)
- R&D capabilities (PC5)
- Management capabilities (PC6)
- Design capabilities (PC7)
- Analytical capabilities (PC8)

<u>As a result, students should:</u>

Know parallel computation methods for solving problems of computing mathematics (matrix computations and partial equations), basic approaches to parallel programming, parallel program development technologies, their basic functionality and basic principles of use.

Be able to construct parallel program execution models, evaluate parallel computing efficiency, analyze computation complexity and possibility of algorithm parallelization, use parallel program general development schemes to implement their own algorithms, evaluate main parameters of the resulting parallel programs such as speedup, efficiency and scalability. Be able to develop, debug and optimize parallel programs for traditional and heterogeneous clusters/HPC systems.

Have parallel program development and debugging skills for MPI and OpenMP-based clusters and multicore systems.

4. Discipline structure and contents

The total discipline intensity is 8 credits/288 hours. Form of testing for the 5th and 6th semesters: examinations

4.1. Discipline structure

	Section	**Semester**	**Semester week**	**Types of academic work including students' independent work and its intensity (in hours)**				**Forms of progress control (per semester week)**
№	**Disciplines**			Lectures	Practice	Lab.	ndependen t work.	**Mid-term examination (per semester)**
1	Introduction	5	1	2	-	-	2	
2	Basic Notions and Definitions		2	2	-	-	4	
3	Computation models and methods of efficiency analysis		3-4	6	-	-	6	

4	OpenMP-Based Parallel Programming		5-6	4	-	3	4	
5	Principles of Parallel Method Development.		7-8	4	-	3	10	
6	OpenMP-Based Parallel Methods for Matrix-Vector Multiplication		9-12	6	-	4	10	
7	OpenMP-Based Parallel Methods for Matrix Multiplication		13-16	6	-	6	10	
8								Examination (36 hours)
9	MPI-Based Parallel Programming. Basic operations.	6	1	3		2	4	
10	MPI-Based Parallel Programming. Collective operations.		2	4	-	4	6	
11	MPI-Based Parallel Methods for Matrix-Vector Multiplication		3-6	6	-	6	6	

12	MPI-Based Parallel Methods for Matrix Multiplication		7-10	6	-	6	10	
13	OpenMP-Based Parallel Programming (continued)		11	4		2	4	
14	Parallel Methods for Solving Systems of Partial Differential Equations		12-13	6		8	4	
15	Parallel Computing at Systems With Shared and Distributed Memory		14-16	4			4	
16	Parallel Computation Modeling and Analysis		17-18	5		4	4	
17	Estimation of Communication Complexity for Parallel Algorithms.		19	4		4	4	
18	Final examination							Examination (36 hours)
	TOTAL:			72	0	52	92	

4.2. Contents of discipline sections

The course consists of several sections with lectures and practical classes dedicated to studying parallel programming models, methods and technologies. This course is oriented to the use of shared memory multicore systems, cluster systems and supercomputers based on traditional CPUs.

The course contains educational materials sufficient for a successful start in the field of parallel programming for modern HPC systems. As part of this approach, lectures account for a significant part of this course and has the following contents:

Lecture 1. Introduction.

The lecture emphasizes the importance of parallel computations and describes the general course structure.

Lecture 2. Basic Notions and Definitions

The lecture introduces the notion of parallel computations. It describes basic efficiency parameters and demonstrates applicability of such parameters by the example of the number sequence summation problem.

Lecture 3. OpenMP-Based Parallel Programming.

The lecture gives a review of the OpenMP technology. It describes OpenMP clauses and their main parameters.

Lecture 4. Principles of Parallel Method Development.

The lecture describes parallel method development stages and gives development examples.

Lectures 5-6. OpenMP-Based Parallel Methods for Matrix-Vector Multiplication

The lecture is dedicated to the basic parallel methods for matrix-vector multiplication for systems with shared memory.

Lectures 7-8. OpenMP-Based Parallel Methods for Matrix Multiplication

The lecture is dedicated to the basic parallel methods for matrix multiplication for systems with shared memory.

Lecture 9. MPI-Based Parallel Programming. Basic operations.

The lecture gives basic notions and definitions related to MPI. It also lists the minimum set of functions required for parallel program development.

Lecture 10. MPI-Based Parallel Programming. Collective operations.

The lecture describes an extended set of operations ensuring more efficient data exchange between computation processes. It also gives an application example for collective operations.

Lectures 11-12. MPI-Based Parallel Methods for Matrix-Vector Multiplication

The lecture is dedicated to the basic parallel methods for matrix-vector multiplication for systems with distributed memory.

Lectures 13-14. MPI-Based Parallel Methods for Matrix Multiplication

The lecture is dedicated to the basic parallel methods for matrix multiplication for systems with distributed memory.

Lecture 15. OpenMP-Based Parallel Programming (continued)

The lecture describes the OpenMP library and the environment variables that can be used for OpenMP runtime environment setup.

Lecture 16. Parallel Methods for Solving Systems of Partial Differential Equations

Lectures 17-18. Parallel Computing at Systems with Shared and Distributed Memory

This lecture describes a classification of computer systems and introduces the notion of multithread and multicore-based parallelism and the notion of clusters. It discusses the definition and types of computer system topologies.

Lecture 19. Parallel Computation Modeling and Analysis

The lecture is dedicated to the basic theory of parallel computation modeling and analysis. It introduces the notion of "operations-operands" graph and describes a number ways to estimate parallel method efficiency. Parallel computation modeling and analysis is illustrated by π computation and the finite difference method.

Lecture 20. Estimation of Communication Complexity for Parallel Algorithms.

The lecture is dedicated to the issues of interprocessor communication time estimation. It compares the runtimes for models and experiments.

4.3. Practice topics

Practice topic
Creation of multithreaded programs as illustrated by the Pi computation problem and scalar vector multiplication using OpenMP
Solving the "Producer-Consumer" problem.
OpenMP-based computation of scalar products of vectors with manual job assignment.
OpenMP-based computation of scalar products of vectors with OpenMP-based job assignment. Experiments involving various schedule versions. Efficiency measurements
Solving a matrix-vector multiplication problem using OpenMP. Implementation and comparison of various data distribution schemes
Parallel MPI-based program "Hello World". Launch of MPI-based programs
MPI-based computation of scalar products of vectors with manual job assignment. Efficiency measurements
MPI-based Pi computation MPI-based parallel sorting
Solving matrix multiplication problem based on MPI

5. Educational technology

Teaching is based on such educational technologies as lectures, practical classes, seminars (problem-oriented, design, discussion, training and organizational ones), extramural independent work, preparation of reports and course papers. This involves the use of project method, information technologies, testing, e-learning tools and web browsing. Lectures include MS Powerpoint presentations. PC presentations and materials are used for the purpose of practice.

6. Teaching and learning support of students' independent work. Grading tools for routine progress control and mid-term proficiency examinations.

Independent work consists in familiarization with theory based on textbooks and monographs indicated in the references list, solving practical problems, preparation for seminars, reports, presentations and e-tests in the course of learning, and answering self-test questions. Independent work may take place both in reference rooms and at home.

Practice assessment has the form of e-tests and credit tasks.

Formative assessment involves periodic tests. Test questions are taken from the self-test question list (see below).

Final assessment is based on the results of task completion. Such tasks involve implementing sequential and parallel versions of a classic algorithm (see below) based on various parallel programming technologies. This also includes computational experiments for test problems, scalability evaluation, comparative performance analysis and final report preparation.

6.1 Practice

Each practical class involves the following activities:

1. Literature search and studying of the algorithm proposed for implementation.
2. Sequential algorithm implementation in C/C++. Testing, debugging, solving test problems, efficiency analysis.
3. Development of the parallel method version based on the following: MPI, OpenMP, MPI + OpenMP. Testing, debugging, solving test problems, efficiency analysis.
4. Comparative analysis of performance/efficiency.
5. Activity report preparation.

List of algorithms proposed for implementation:

Matrix multiplication

1. Use of horizontal partitioning
2. Cannon’s algorithm
3. Fox algorithm
4. DNS algorithm

Matrix transposing

Solving linear system

1. Gaussian method
2. Iterative methods (Jacobi method)
3. Iterative methods (Seidel method)

Sorting

1. Bubble sorting
2. Quick sorting

For each section, the set of algorithms may be modified/extended at the teacher’s discretion.

6.2. Self test

1. How are the concepts “speedup” and “efficiency” defined?
2. Is it possible to attain superlinear speedup?
3. What is the contradictoriness of the speedup and efficiency characteristics?
4. What is scalable algorithm? Give examples of methods with various scalability levels
5. What are the main ways to ensure parallelism?
6. In what way can parallel computer systems differ?
7. What is the basis of Flynn's taxonomy?
8. What is the principle of subdviding multiprocessor systems into multiprocessors and multicomputers?
9. What system classes exist for multiprocessors?

10. What are advantages and disadvantages of symmetric multiprocessors?
11. What system classes exist for multicomputers?
12. What are advantages and disadvantages of cluster systems?
13. What data communication network topologies are most widely used to construct multiprocessor systems?
14. What are the specific features of data communication networks for clusters?
15. What are the main characteristics of the data communication networks?
16. What system platforms may be used for the purposes of building clusters?
17. How is the “operations-operands” model defined?
18. How is the schedule for the distribution of computations among processors defined?
19. How is the time of parallel algorithm execution defined?
20. What schedule is optimal?
21. How can the minimum possible time of problem solving be defined?
22. What is a paracomputer? What can this concept be useful for?
23. What estimates should be used as the characteristics of the sequential problem solving time?
24. How to define the minimum possible time of parallel problem solving according to “op-erands-operations” graph?
25. What dependences may be obtained for parallel problem solving time if the number of processor being used is increased or decreased?
26. What number of processors corresponds to the parallel algorithm execution time comparable in the order with the estimates of minimum possible time of problem solv-ing?
27. How are the concepts “speedup” and “efficiency” defined?
28. Is it possible to attain superlinear speedup?
29. What is the contradictoriness of the speedup and efficiency characteristics?
30. How is the concept of computation cost defined?
31. What is the concept of the cost-optimal algorithm?
32. What does the problem of parallelizing a sequential algorithm of the numeric values summation lie in?
33. What is the essence of the summation cascade scheme? What is the aim of considering the modified version of the scheme?
34. What is the difference between the speedup and efficiency characteristics for the discussed versions of the summation cascade scheme?
35. What is the parallel algorithm of all the partial sums computation of a numeric value sequence?
36. How is Amdahl’s law formulated? Which aspect of parallel computation does it allow to take into account?
37. What algorithm is a scalable one? Give examples of methods with various scalability levels
38. What is meant by “process” and “thread”? Describe their similarities and differences.
39. Give a general characteristic of parallel programs as a system of threads run in parallel.
40. What basic assumptions can be made as to the character of temporal relations between the executed command sequences of different threads?
41. What determines increased complexity of parallel programming?
42. What does the problem of thread mutual exclusion consist in? What are the main requirements to the methods for solving this problem?
43. What is the disadvantage of lockstep synchronization for the purpose of thread mutual exclusion?

44. Give examples of incorrect solution of the thread mutual exclusion problem when mutual exclusion is lost.
45. Give examples of incorrect solution of the thread mutual exclusion problem when thread interlocking is possible.
46. Give examples of incorrect solution of the thread mutual exclusion problem when infinite delay of access to critical sections is possible.
47. How is the Dekker's algorithm used for solving the problem of thread mutual exclusion?
48. What does the concept of Dijkstra semaphores consist in? Give an example of solving the problem of mutual exclusion of threads using semaphores.
49. What is the concept of Hoare’s monitors? Give an example of solving the problem of mutual exclusion of threads using monitors.
50. What does the problem of thread synchronization consist in?
51. How is the problem of thread synchronization solved using conditional variables?
52. How is the problem of thread synchronization solved using barrier synchronization?
53. What does the problem of thread interlocking consist in? Indicate the required conditions for forming deadlocks.
54. What does the “thread-resource” graph-based parallel model consist in?
55. How can thread number optimization improve parallel program efficiency?
56. How can minimization of thread communication improve parallel program efficiency?
57. How can memory operation optimization improve parallel program efficiency?
58. How can the use of parallel method libraries improve parallel program efficiency?
59. What computer platforms range with shared memory systems?
60. What approaches to parallel software development are used?
61. What is the essence of OpenMP basics?
62. Why is it important to standardize parallel program development tools?
63. What are the main advantages of OpenMP?
64. What is a parallel program in terms of OpenMP?
65. What is meant by the notion of thread?
66. What problem arise when shared data are used by parallel threads?
67. What writing format is used for OpenMP clauses?
68. What is the purpose of the parallel clauses?
69. What is meant by the parallel program segment, region and section?
70. What minimum set of OpenMP clauses is necessary for parallel software development?
71. How can one determine runtime for an OpenMP-based software?
72. How does one parallelize loops in OpenMP? What are the conditions of loop parallelization?
73. Which OpenMP capabilities manage allocation of loop iteration to threads?
74. How is the order of iterations determined for parallelized loops in OpenMP?
75. What are the rules of computation synchronization for parallelized loops in OpenMP?
76. How can one parallelize program code segments with low computational complexity?
77. How are shared and local thread variables defined?
78. What is meant by reduction?
79. What ways to ensure mutual exclusions can be used in OpenMP?
80. What is meant by an atomic operation?
81. How are critical sections determined?
82. What operations are offered by OpenMP for semaphore variables (locks)?
83. In what cases should we apply barrier synchronization?

84. How is task paralellism ensured in OpenMP (sections directive)?
85. How are single-thread parts of parallel fragments identified (single and master directives)?
86. How is memory state synchronized (using flush directive)?
87. How are persistent local variables of threads used (threadprivate and copyin directives)?
88. What tools does OpenMP have to manage the number of created threads?
89. What is meant by dynamic thread creation mode?
90. How is parallel segment nesting managed?
91. How can one ensure the program code uniqueness for both sequential and parallel program versions?
92. What compilers ensure OpenMP support?
93. What minimum set of operations of sufficient for the organization of parallel computations in the distributed memory systems?
94. Why is it important to standardize message passing tools?
95. How can a parallel program be defined?
96. What is the difference between the concepts of “process” and “processor”?
97. What minimum set of MPI functions is necessary for parallel software development?
98. How are the messages being passed described?
99. How do we organize the reception of messages from specific processes?
100. How can one determine runtime for MPI-based software?
101. What is the difference between point-to-point and collective data transmission operations?
102. Which MPI function provides transmitting data from a process to all the processes?
103. What is meant by reduction?
104. In what cases should we apply barrier synchronization?
105. What is a deadlock? In what cases does the function of the simultaneous transmission/reception guarantee the absence of deadlock situations?
106. What collective data transmission operations are supported in MPI?
107. What is meant by the derived data type in MPI?
108. What ways of type construction are implemented in MPI?
109. In what situations may data packing and unpacking be used?
110. What is a communicator in MPI?
111. What is a virtual topology in MPI?
112. What types of virtual topologies are supported in MPI?
113. What may virtual topologies appear to be useful for?

6.3. Grading criteria

Perfect - complete mastery of basic and additional subject matter without errors or mistakes, ability to solve non-conventional problems; competencies (parts of competencies) related to the discipline have been mastered in an integrated manner exceeding obligatory requirements. A sustainable system of competences has been formed, relation to mastering other competences is evident;

Excellent - complete mastery of basic and additional subject matter without errors or mistakes, competencies (parts of competencies) have been mastered completely at a high level; a sustainable system of competencies has been formed;

Very good - sufficient mastery of basic subject matter with insignificant errors, ability to solve standard problems, competencies (parts of competencies) have been mastered completely;

Good - mastery of basic materials with noticeable errors, competencies (parts of competencies) have been mastered for the most part;

Satisfactory - minimal mastery of the subject matter with errors and mistakes, ability to solve principal problems, competencies (parts of competencies) have been mastered at the minimum level required to achieve basic learning objectives;

Unsatisfactory - insufficient knowledge of the subject matter, additional training required, competencies (parts of competencies) related to this discipline have not been mastered in a manner sufficient for achieving basic learning objectives;

Poor - no mastery of the subject matter; respective competencies have not been mastered.

7. Teaching, learning and information support

Recommended references:

1. Gergel V.P., Strongin R.G. Introduction to Parallel Computing on Multi-Processor Systems. UNN. Nizhni Novgorod, 2001, (2003 2-nd edition).
2. Gergel V.P. High performance computations of multiprocessor multicore systems. Moscow, 2010
3. Chandra R., Dagum L., Kohr D., Maydan D., McDonald J., Melon R. Parallel Programming in OpenMP. – Morgan Kaufmann Publishers, 2000.
4. Grama, A., Gupta, A., Kumar V. Introduction to Parallel Computing. – Harlow, England: Addison-Wesley, 2003.
5. Quinn M. J. Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill, 2004.
6. Buyya, R. High Performance Cluster Computing. Volume1: Architectures and Systems. Volume 2: Programming and Applications. - Prentice Hall PTR, Prentice-Hall Inc., 1999
7. Culler, D., Singh, J.P., Gupta, A. Parallel Computer Architecture: A Hardware/Software Approach. - Morgan Kaufmann., 1998
8. Group, W., Lusk, E., Skjellum, A. (1994). Using MPI. Portable Parallel Programming with the Message-Passing Interface. –MIT Press.
9. Roosta, S.H. Parallel Processing and Parallel Algorithms: Theory and Computation. Springer-Verlag,NY., 2000
10. Fengguang S., Tomov S., Dongarra J. Efficient Support for Matrix Computations on Heterogeneous Multi-core and Multi-GPU Architectures. – University of Tennessee Computer Science Technical Report, UT-CS-11-668, (also Lawn 250), 2011. [http://icl.cs.utk.edu/news_pub/submissions/lawn250.pdf]

8. Inventory

To conduct classes in this discipline, the following software and hardware is required.

Hardware

For MPI classes, a computing cluster/mini cluster is desirable. In absence of cluster systems, several PCs in the local network can be used.

For OpenMP classes, multicore PCs are required.

Software

For the purposes of this course, the following software is necessary:

- Microsoft Windows XP/7 or Linux
- Microsoft Visual Studio 2008/2010
- Intel Parallel Studio XE
- A MPI implementation (e. g. MPICH)

Author(s): Professor ____________ V.P. Gergel

Assistant ____________ Ye. A. Kozinov

Course program is discussed by Software Department members.

«______» ________________ 2014; Document # _______

The head of the Software Department, prof. __________________________ Roman Strongin

Course program is approved by methodical commission of Computational Mathematics and Cybernetics Faculty of UNN,

«______» ________________ 2014; Document # _______

The head of the commission _______________________________ Natalia Shestakova