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

Parallel Programming for Shared Memory Systems

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.В.ОД.8

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Purposes of mastering the discipline

The purpose of this course is mastering a set of skills and knowledge required for successful start of professional activities in the field of parallel programming on shared memory systems.

The course incorporates both all theory of parallel computing using Intel Threading Building Blocks (TBB), and practical knowledge and skills of TBB-based parallel programming.

The practice covers all parallel programming stages including sequential implementation for comparison purposes, parallel implementation and its analysis. Academic activities are based on test problems which do not require any special knowledge in any application except for the information represented in the course.

The main course objective is to study basic notions of TBB library and its use for parallel programming on shared memory systems.

The following tasks can be identified:

1. Mastering the use of TBB on shared memory systems.
2. Studying general approaches to implementation of TBB-based algorithms.
3. Discussion (in the course of practice) of efficient implementation examples of studied algorithms.

The course is intended for engineers, teaching staff, scientist and both graduate and post-graduate students.

2. The course in the undergraduate program

This course is intended for fourth year students and is taught in the 7^{th} semester. This is an elective course of the professional block.

It is intended for students familiar with fundamentals of C++ for program development. Experience in the use of OpenMP will be useful for the course purposes but is not mandatory. Students must have basic knowledge of mathematics corresponding to that of 2^{nd} and 3^{rd} academic years.

3. Requirements to discipline mastery

Learning 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)
- ability to understand, develop and use state-of-the-art information technologies (PC4)
- research and development capabilities (PC5)
- analytical capabilities (PC8)

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

Know the contents and capabilities of the TBB library; methods of parallel program development for shared memory systems.

Be able to correctly use the TBB-based algorithms and classes for parallel programming.

Possess the skills of parallel program development and debugging using TBB.

4. Discipline structure and contents

The total discipline complexity is 16 hours which include 12 hours of lectures, 4 hours of practice and 16 hours of independent work. Form of testing - credit.

4.1. Discipline structure

Sl. №	Section Disciplines	Semester	Semester week	Types of academic work including students' independent work and its intensity (in hours)				Forms of progress control *(per semester week)* Mid-term examination *(per semester)*
				Lectures	Practical	Practice	Ind. practice	
1	Introduction to TBB	6	1-2	2	-	-	2	Test
2	Parallelizing simple loops		3-4	2	-	-	2	Test
3	Parallel matrix-vector multiplication		5-6	-	-	2	2	Test
4	Parallelizing complex loops		7-8	2	-	-	2	Test
5	Task-based programming		9-10	2	-	-	2	Test
6	Parallel computation of the fast Fourier transform		11-12	-	-	2	2	Test
7	TBB synchronization primitives		13-14	2	-	-	2	Test
8	Flow graph		15-16	2	-	-	2	Test
	TOTAL:			12	-	4	16	Form of testing - credit.

4.2. Contents of discipline sections

The course has the following contents:

1. Introduction to TBB

This lecture describes the fundamentals of the TBB library, its contents and specifics of TBB-based programming. It describes the ways to initialize and terminate the library and to operate TBB together with OpenMP. The lecture also reviews the mechanisms of time measurement and dynamic memory allocation.

2. Parallelizing simple loops

The lecture describes the tbb::parallel_for template function that enables implementation of parallelized simple loops as illustrated by the matrix-vector multiplication problem. It reviews possible function variants and its parameters.

3. Parallel matrix-vector multiplication

This practice formulates the matrix-vector multiplication problem. It features a sequential algorithm implementation. The practice also reviews variations of parallel algorithm implementation and offers a parallel_for-based algorithm implementation.

4. Parallelizing complex loops

The lecture describes the tbb::parallel_reduce template function that enables implementation of parallelized simple loops as illustrated by the scalar product problem. It also describes structures that enable implementation of parallel programs involving loops where the number of iterations is not known in advance, sorting and pipelined computing.

5. Task-based programming

The lecture describes the task mechanism which is the most flexible one for parallel programming purposes. It reviews the methods for scheduling and synchronization of task execution. The queens problem illustrates a parallel implementation of recursive algorithms based on tasks.

6. Parallel computation of the fast Fourier transform

The practice formulates the problem of the fast Fourier transform computation. It features a sequential algorithm implementation. It describes parallel algorithm implementation versions and implements the iterative algorithm version based on parallel_for and the recursive version based on tasks.

7. TBB synchronization primitives

This lecture deals with TBB synchronization primitives. It describes various implementations of mutexes and readers/writers mutexes. It describes the atomic template class whose methods are atomic. The lecture also deals with threadsafe containers that may be useful for implementing a wide range of parallel algorithms.

8. Flow graph

This lecture describes parallel application development based on representing an algorithm as a flow graph. It contains visible examples demonstrating the reviewed approach. Main types of graph nodes are reviewed in detail.

5. Educational technology

The educative process is based on lectures, practices and extramural independent work. Lectures and practices are MS PowerPoint-aided.

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 and answering self-test questions. Independent work may take place both in class and at home.

Practice is assessed on the basis of e-tests. At the end of the course, students take a pass/fail test

6.1 Progress control

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

Final assessment is based on the final test results. This test includes questions from all course sections.

6.2 Self test

Section 1

1. On which operation systems does the TBB run?
2. Which factors affect the application runtime?
3. How does one initialize the TBB library?
4. How does one complete TBB operation?
5. How to initialize TBB for a certain number of threads in the TBB program?
6. Can TBB be used together with OpenMP?
7. What should be done to change the number of threads in a TBB program?

Section 2

8. How is for loop parallelized in the TBB library?
9. What is a Functor?
10. What is an Range?
11. What is the essence of grainsize?
12. How does parallel_for of the TBB library work?
13. What functor methods have to be implemented when parallel_for is used?
14. How is the grainsize value selected?
15. How is computation scheduling effected in parallel_for?
16. What are the scheduling strategies and in what ways are they different?

Section 3

17. How to assemble a TBB-based application?
18. What is the computational complexity of a matrix-vector multiplication algorithm (n is the matrix size)?
19. What are the main ways to allocate effort to threads for solving the problem of matrix-vector multiplication?
20. What is the total number of scalar operations in the problem of matrix-vector multiplication?
21. What TBB algorithm and class are the best for implementation of the parallel matrix-vector multiplication algorithm?

Section 4

22. What will you used to parallelize the for loop with the fixed number of iterations and reduction?
23. How is computation scheduling effected in parallel_reduce?
24. What is the difference between parallel_for and parallel_reduce?
25. What problems are parallel_for and parallel_reduce intended for?
26. What algorithms are best implemented using the tbb::pipeline class?
27. What algorithms are best implemented using the tbb::do function?

Section 5

28. What is a TBB logical task?
29. What are the special aspects of parallel algorithm development using logical tasks?
30. What structure will you use to parallelize the recursive function?
31. What memory type should be allocated to the logic task and why?

Section 6

32. What is the computational complexity of the DFT algorithm?
33. What is the computational complexity of the FFT algorithm?
34. What is the difference between DFT and FFT?
35. What is the essence of bit reversal?
36. Is the bit reversal stage indispensable?
37. What TBB algorithm and class are the best for implementation of the FFT algorithm?

Section 7

38. What is the mutual exclusion problem?
39. What is data race?
40. What synchronization primitives are implemented in TBB?
41. In what way do TBB threadsafe differ from STL containers?
42. Why the size method return a signed integer in the tbb::concurrent_queue class?

Section 8

43. What is a TBB flow graph?
44. What are TBB flow graph node?
45. What are TBB flow graph edges?
46. What node types exist in TBB? In what ways do they differ?

6.4 Grading criteria

Perfect	more than 95% test questions answered correctly

Excellent	80-95% test questions answered correctly
Very good	70-79% test questions answered correctly
Good	60-69% test questions answered correctly
Satisfactory	50-59% test questions answered correctly
Unsatisfactory	25-49% test questions answered correctly
Poor	less than 25% test questions answered correctly

7. Teaching, learning and information support

a) Basic references

1. Intel® Threading Building Blocks Home Page: https://www.threadingbuildingblocks.org/
2. Intel® Threading Building Blocks Reference Manual: https://software.intel.com/en-us/node/506130
3. Intel® Threading Building Blocks User Guide: https://software.intel.com/en-us/node/506045
4.

b) Additional references:

5. Andrews, G. R. (2000). Foundations of Multithreaded, Parallel, and Distributed Programming. – Reading, MA: Addison-Wesley.
6. Quinn, M. J. (2004). Parallel Programming in C with MPI and OpenMP. – New York, NY: McGraw-Hill.
7. Gonzalez R. C., Woods R. E. (2007). Digital Image Processing. - Prentice-Hall.
8. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein C. (2009). Introduction to Algorithms, 3rd Edition. – The MIT Press.
9. Kumar V., Grama, A., Gupta, A., Karypis, G. (1994). Introduction to Parallel Computing. - The Benjamin/Cummings Publishing Company, Inc. (2nd edn., 2003)

1. Inventory

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

Hardware

Multicore computers

Software

MicrosoftWindowsXP/7,Linux or Mac OS X

Code editor — Notepad++, SublimeText, Eclipse or Emacs

TBB library

9. Authors

The course has been developed under the supervision of Aleksey A. Sidnev, master of IT, teaching assistant at the Software Department (Faculty of Computing Mathematics and Cybernetics, UNN)

Author: A. A. Sidnev_____Assistant

The syllabus has been reviewed in the meeting of the Software Department of the Computing Mathematics and Cybernetics Faculty

_____ _____ 2014; record No. _____

R. G. Strongin _____ Professor, Doctor of Physics and Mathematics

The syllabus has been approved in the meeting of the Computing Mathematics and Cybernetics Faculty Methodological Committee.

_____ _____ 2014; record No. _____

The head of the commission _____ Natalia Shestakova