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

PROGRAMMING AND OPTIMIZATION FOR INTEL XEON PHI

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.В.ДВ.8

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Course objective

On the one hand, programming for Intel Xeon Phi is essentially similar to programming for traditional processors. We still should efficiently use vector instruction sets (SIMD, data parallelism), provide uniform as possible thread workload (core-level parallelism), and efficiently work with the memory (taking in account multilayer memory organization).

The methods to achieve this goal are also somewhat standard. Thus, to enable vectorization we may use high performance libraries, capabilities of optimizing compilers, assisting it with compiler options and directives, use Intel Cilk Plus, employ intrinsics and inline assembly if necessary. At the same time wider registers, vector FMA, enhanced set of operations with memory and other features of vector extensions of Xeon Phi instruction set are capable to significantly speed up the computations. For core-level parallelism we may use well-established parallel programming technologies (MPI, OpenMP, Intel Cilk Plus and others). In this case the choice of appropriate number of processes and threads can significantly influence the performance. The basic methods of improving memory efficiency are also rather standard. Thus, the main concerns are efficient cache utilization, avoiding bus overloading, and minimizing exchanges with host memory. Overall, optimization for Intel CPUs generally favors Xeon Phi as well and vice versa. Nevertheless it is not the same. In the lectures and practical works of this course we demonstrate on different examples how various optimization techniques influence the run time on the host and coprocessor. This course considers development and optimization of software that efficiently utilize manycore architectures by the example of Intel Xeon Phi.

The main objective of the course is to study basic principles and acquire skills to develop programs that efficiently utilize Intel Xeon Phi.

It involves solving the following **problems**:

1. Exploration of Intel Xeon Phi architecture and the main mechanisms that influence application performance.

2. Study of Intel Xeon Phi programming models and the corresponding system-level software. Mastering development, building and launching applications on Intel Xeon Phi.

3. Study of principles and features of applying parallel programming technologies for development and optimization of computational applications for Xeon Phi, including SIMD, OpenMP and Cilk Plus.

4. Mastering optimization and vectorization of computational loops, enhancing memory efficiency, load balancing.

5. Getting acquainted with examples of successful optimization of applications that are initially not fully suited for efficient utilization of Intel Xeon Phi.

2. Course position in the bachelor program

The course is developed for the 4-th year students and is given in the 7-th semester. This course belongs to the elective disciplines of the professional block.

The course is oriented on the students with basic skills of C/C++ program development and familiar with parallel programming using OpenMP and MPI. Experience with TBB and MKL as well as other components of Intel Parallel Studio XE will be useful throughout the course, but is not required. All necessary information of using Intel Parallel Studio XE is included in the course materials. The students are supposed to have basic mathematical knowledge corresponding to 2-3 years of university.

The course is developed in 2014 as part of the "5-100 project" by Iosif Meyerov and Sergey Bastrakov.

3. Learning outcomes and requirements

In the framework of this course, the following **competencies** are formed:

- Possessing the general culture of thinking, the ability to perceive, compile and analyze information (General Competency 1 - GC1). Students will be able to:
  - construct oral and written arguments in a logical and clear manner.
- The ability of intellectual, cultural, moral, physical, and professional self-development and self-improvement (GC 2). Students will be able to:
  - constantly improve their professional and cultural level.
- The ability to understand and apply in practice the theory of information as a fundamental scientific basis of information technology (Professional Competency 1 – PC 1). Students will be able to:
  - understand the content side of the information process, know the techniques for sending, receiving, processing, analyzing and storing data.
- The ability to understand, develop and apply modern information technology (PC 4). Students will be able to:
  - understand the concepts and implement the functionality of the following core technologies:
    - at the level of technological literacy:
      - computer systems architecture;
    - at the level of in-depth knowledge:

        - basic programming;
        - parallel and distributed computing.
    - develop and use professionally the software for supporting information systems and processes, to be able to use modern instrumental computing equipment.

- The ability to conduct research (PC 5). Students will be able to:
    - develop new algorithmic and methodological and technological solutions;
    - collect, process and interpret the data of modern research necessary to develop approaches, decisions and conclusions on appropriate scientific and professional issues.
- The ability to conduct analytical activities (PC 8). Students will be able to:
    - analyze and select modern technologies and methodologies for implementing an information system.

As the result of education graduates from the course:

Graduates from the course will have knowledge of architecture and programming techniques of Intel Xeon Phi and be able to.

Will know architecture, programming and optimization techniques of Intel Xeon Phi.

Will Be Able To use programming and optimization techniques of Intel Xeon Phi for applied applications development.

Can use system software (compiler, profiler, high performance computing libraries) for software development for Intel Xeon Phi.

4. Course outline

The course consists of 1 credit, 36 hours, including 10 lecture hours and 10 practice hours. Practice classes can be held as lab works (students carry out assignments step-by-step under supervision) or master class (supervisor demonstrates and explains step-by-step solutions). 16 hours are allocated for individual work. The authors encourage additional work.

4.1. Course outline

Course outline is as follows:

#	Module	Semester	Week	Module type				Assessment
				Lecture	Seminar	Lab	Individual work	
1	Intel Xeon Phi architecture	6	1	2	–	–	1	Test
2	Program execution on Intel Xeon Phi. Models of Intel Xeon Phi programming.		2	2	–	–	1	Test
3	Building and launching applications on Intel Xeon Phi		3-4	–	–	2	2	Test
4	Vector extensions of Intel Xeon Phi		4-5	2	–	–	1	Test
5	Optimization of applications for Intel Xeon Phi using Intel C/C++ Compiler. Vectorization		6-7	–	–	2	2	Test
6	Elements of optimization of applications for Intel Xeon Phi. Intel C/C++ Compiler		7-8	2	–	–	2	Test
7	Optimization of prime factorization. Vectorization and load balancing		9-10	–	–	2	2	Test
8	Step-by-step optimization of European option pricing		11-12	–	–	2	2	Test
9	Elements of optimization of applications for Intel Xeon Phi: Intel MKL, Intel VTune Amplifier XE		13-14	2	–	–	1	Test
10	Optimization of matrix multiplication. Enhancing memory efficiency		15-17	–	–	2	2	Test
	TOTAL:			**10**	–	**10**	**16**	**Final assessment form – exam**

4.2. Course description

Course content is as follows:

1. Intel Xeon Phi architecture

Architecture and programming model of Intel Xeon Phi. Host, Host OS, Micro Operating System, Intel Manycore Platform Software Stack. Coprocessor architecture, main features of its components. The principle of core pipeline operation. Memory hierarchy and the corresponding performance aspects. A brief overview of instruction code.

2. Program execution on Intel Xeon Phi. Models of Intel Xeon Phi programming.

Principles of program execution and models of organization of computations using Intel Xeon Phi coprocessors. System software that responsible for program execution on Xeon Phi. Intel Manycore Platform Software Stack and symmetric communication interface. Offload, coprocessor-only and symmetric models. The ways to develop applications for Xeon Phi.

3. Building and launching applications on Intel Xeon Phi

Porting applications to Intel Xeon Phi. Examples. Programming models and ways of building and running applications. Overview of programming models. Examples of running application in Offload mode, Coprocessor-only mode and Symmetric mode.

4. Vector extensions of Intel Xeon Phi

Introduction about vector extensions and SIMD paradigm. Features of vector extensions on Xeon Phi. Main data types and registers. Main operations and extended support for mathematical functions. Access to vector extensions from high level programming languages. Ways of vectorization. Vectorization of mathematical functions.

5. Optimization of applications for Intel Xeon Phi using Intel C/C++ Compiler. Vectorization

Basic topics of vectorization. Main ways to use vectorization on Intel Xeon Phi coprocessor. Vectorization techniques on examples: restrict keyword and #pragma ivdep to guarantee there are no potential dependencies, explicit vectorization using #pragma simd, Array notation and Elemental functions. Vectorization of loops that include mathematical routines.

6. Elements of optimization of applications for Intel Xeon Phi. Intel C/C++ Compiler

Optimization techniques for Intel Xeon Phi. Hardware details, programming models and vectorization capabilities. New elements of the offload mode that handle memory. Explicit and implicit memory management. Additional aspects of vectorization. Auto vectorization, compiler directives, Array notation and Elemental functions. Loop profiler and optimization reports. Load balancing in parallel applications. Examples.

7. Optimization of prime factorization. Vectorization and load balancing

Non-standard techniques for boosting performance of applications of Xeon Phi. Simple algorithm for prime factorization. Implementation and porting to Xeon Phi. Efficiency of parallel implementation and load balancing issues. Optimization aspects. Hybrid CPU + Xeon Phi computational scheme.

8. Step-by-step optimization of European option pricing

A model and basic concepts of a financial market. Descriptions of the option pricing problem. Basic implementation and its performance analysis. Performance optimization in a step-by-step fashion: eliminate unnecessary type casts, carry out invariants, perform mathematical transforms that replace heavy math routines with lighter ones, vectorize and parallelize, perform warm-up to reduce overhead on thread creation, try reducing precision of floating-point operations, utilize streaming stores.

9. Elements of optimization of applications for Intel Xeon Phi: Intel MKL, Intel VTune Amplifier XE

Intel MKL on Xeon Phi: offloading computations in Automatic offload and Compiler assisted offload. Intel Amplifier: general overview, running on Xeon Phi, important performance evaluation metrics.

10. Optimization of matrix multiplication. Enhancing memory efficiency

Naive implementation of matrix multiplication. Order of loops and its influence to the performance on Xeon Phi. Alignment and compiler directives. Block matrix multiplication algorithms.

5. Learning technologies

During course we use the following learning technologies: lectures, lab works, individual work, assessment techniques. Powerpoint presentations for all lectures and practical lessons are used.

6. Individual work and assessment techniques

Individual work consists of mastering theoretical and practical material according to the given references, solving practical problems, and answering on the given questions. Individual work can be done in both classes and at home.

Control of individual work is performed by electronic tests. In the end of the program there is a final test.

6.1 Assessment forms

Monitoring of progress in studies is performed by tests in class that consist of assignments from the list of questions and practical problems (given below).

The final attestation is done based on the results of the final test. This test includes questions from all sections of the course.

6.2 Individual work: Questions and Practical problems

Module 1

1. Distinguish main architecture features of Intel Xeon Phi compared to multicore CPUs.
2. Describe operating of core pipeline of Xeon Phi.
3. Formulate principles of memory organization of Xeon Phi.
4. Find out main differences in vector extensions of Xeon Phi from AVX instruction set.
5. Name features of Xeon Phi architecture that are most performance influencing.

Module 2

6. Implement dot product using various parallel programming technologies.
7. Implement dense matrix-vector multiplication using various parallel programming technologies.
8. Implement multiplication of a sparse matrix on a dense vector using various parallel programming technologies.

   For each problem build and run your application on Xeon Phi, analyze the correctness. Compare execution time on host and coprocessor.

Module 3

9. Implement matrix-vector multiplication in Offload mode.
10. Implement matrix-vector multiplication in coprocessor-only mode. Suppose there is only one coprocessor.
11. Implement matrix-vector multiplication in symmetric mode. Provide two levels of parallelism: simultaneous computation of row-vector products and parallel computation of each row-vector product.

    For each problem build and run your application on Xeon Phi, analyze the correctness. Compare execution time on host and coprocessor.

Module 4

12. List the main technical characteristics of Xeon Phi vector extensions.
13. List the main Xeon Phi vector instruction groups.

14. List the main ways of vectorization in C and Fortran programming languages. Formulate advantages and disadvantages of each way.
15. Implement dense matrix-vector product on Xeon Phi. Ensure vectorization using different ways. Analyze the correctness and performance compared to scalar code.

Module 5

16. Implement scalar and all vectorized versions of the first example considered for Intel Xeon Phi. Compare performance of all versions, explain the results.
17. Implement dense matrix-vector product. Find out details of vectorization using vectorization report, assist vectorization if necessary. Does vectorization depend on row-major or column-major matrix storage?
18. Perform a study similar to the previous assignment for dense matrix-matrix product computed by definition.
19. Study the considered example on loops with calls to mathematical routines. Find the minimum number of iterations which allows VML to outperform SVML.

Module 6

20. Compare explicit and implicit data transfers in offload mode.
21. Explain how does #pragma simd directive work and when is it used.
22. Overview main options of #pragma simd directive. Describe advantages and disadvantages of this way of vectorization.
23. Describe Array notation and its advantages for vectorization.
24. Describe Elemental functions and its advantages for vectorization.
25. How to use compiler vector reports?
26. What is data alignment and why is it used?
27. How to vectorize an external loop using compiler directives?
28. How to use loop profiler?
29. How to provide thread mapping to cores on Intel Xeon Phi?

Module 7

30. Adjust number of OpenMP thread for the maximum performance.
31. Check if performance of our implementation increases with odd chunk size.
32. Study ways of algorithmic optimization of prime factorization.
33. Study ways of further optimization of our implementation.
34. Find the optimal distribution of workload between CPU and Xeon Phi in the hybrid scheme.

Module 8

35. Compare AoS and SoA data layouts in Black-Scholes computation.

36. Modify our implementation to computing Call and Put option prices. Experiment with different versions of the code and evaluate influence of our optimizations on performance on CPU and Xeon Phi in this case. Prove that memory bus on Xeon Phi is overloaded if many threads are used.
37. Try different amount of threads on Xeon Phi. Find the optimal configuration.
38. Explain speedup from vectorization on CPU and Xeon Phi.

Module 9

39. Describe, in which cases MKL can be used and why?
40. What are features of MKL on coprocessor?
41. Describe ways of using MKL on coprocessor, advantages and disadvantages of each way.
42. What kind of information useful for optimization can one retrieve using Amplifier?
43. What are analysis types of Amplifier? Describe the difference between them.
44. Which role does a profiler play in the process of software optimization?
45. What are features of code profiling on coprocessor?

Module 10

46. Create a memory access map for the naive version for 8-way associative cache, cache lane is 64 bytes, cache size is 32 KB. Explain why the program is slow for N = 1024.
47. Find optimal block sizes for square and rectangular blocks for block matrix multiplication.
48. Find optimal number of OpenMP threads for block matrix multiplication depending of matrix and block sizes.
49. Investigate how warm-up affects computational time of all implemented versions.

6.4 Assessment criteria

Perfect	Correct answers on >95% of the number of test questions
Excellent	Correct answers on 80-95% of the number of test questions
Very good	Correct answers on 70-79% of the number of test questions
Good	Correct answers on 60-69% of the number of test questions
Satisfactory	Correct answers on 50-59% of the number of test questions
Unsatisfactory	Correct answers on 25-49% of the number of test questions
Bad	Correct answers on <25% of the number of test questions

7. References

7.1 Main literature

1. Jeffers J., Reinders J. Intel Xeon Phi Coprocessor High Performance Programming. – Morgan Kaufmann, 2013. – 432 p.
2. Intel Xeon Phi Coprocessor System Software Developers Guide. [http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-system-software-developers-guide].
3. Nguyen Loc Q et al. Intel Xeon Phi Coprocessor Developer's Quick Start Guide. [http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide].
4. Rahman R. Intel Xeon Phi Core Micro-architecture [http://software.intel.com/en-us/articles/intel-xeon-phi-core-micro-architecture].
5. Jeffers J., Reinders J. High Performance Parallelism Pearls. Multicore and Many-core Programming Approaches. – Morgan Kaufmann, 2014. – 600p.
6. Intel Developer Zone [http://software.intel.com/en-us/mic-developer].

7.2 Further reading

7. Intel Math Kernel Library Documentation [http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation]
8. Intel Xeon Phi Coprocessor. Performance Monitoring Units Documentation [http://software.intel.com/sites/default/files/forum/278102/intelr-xeon-phitm-pmu-rev1.01.pdf]
9. Krishnai R. Data Alignment to Assist Vectorization: [http://software.intel.com/en-us/articles/data-alignment-to-assist-vectorization].
10. Mackay D. Optimization and Performance Tuning for Intel Xeon Phi Coprocessors, Part 1: Optimization Essentials, 2012. [http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-1-optimization]
11. Reinders J. An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors. [http://software.intel.com/en-us/blogs/2012/11/14/an-overview-of-programming-for-intel-xeon-processors-and-intel-xeon-phi].
12. Reinders J. An Overview of Programming for Intel Xeon processors and Intel Xeon Phi coprocessors. [http://software.intel.com/en-us/blogs/2012/11/14/an-overview-of-programming-for-intel-xeon-processors-and-intel-xeon-phi].
13. Belinda L. Intel VTune Amplifier XE video tutorial 5: Using the command line, 2013 [http://software.intel.com/ru-ru/videos/intel-vtune-amplifier-xe-video-tutorial-5-using-the-command-line]

14. Best Known Methods for Using OpenMP on Intel Many Integrated Core (Intel MIC) Architecture, Volume 1a, January 29, 2013.
15. Cepeda S. Optimization and Performance Tuning for Intel Xeon Phi Coprocessors, Part 2: Understanding and Using Hardware Events, 2012 [http://software.intel.com/en-us/articles/optimization-and-performance-tuning-for-intel-xeon-phi-coprocessors-part-2-understanding]
16. Fourestey G. Intel Xeon Phi Programming Models [https://hpcforge.org/plugins/mediawiki/wiki/pracewp8/images/6/68/XeonPhi.pdf]
17. Green R.W. Effective Use of the Intel Compiler's Offload Features. [http://software.intel.com/en-us/articles/effective-use-of-the-intel-compilers-offload-features]
18. Green R.W. Outer Loop Vectorization via Intel Cilk Plus Array Notations. [http://software.intel.com/en-us/articles/outer-loop-vectorization-via-intel-cilk-plus-array-notations]
19. Green R.W. Overview of Vectorization Reports and new vec-report6: [http://software.intel.com/en-us/articles/overview-of-vectorization-reports-and-new-vec-report6]
20. Green R.W. Vectorization Essentials. [http://software.intel.com/en-us/articles/vectorization-essential]
21. Intel and Third Party Tools and Libraries available with support for Intel® Xeon Phi™ Coprocessor [http://software.intel.com/en-us/articles/intel-and-third-party-tools-and-libraries-available-with-support-for-intelr-xeon-phitm].
22. Intel Compiler Documentation. Thread Affinity Interface [http://software.intel.com/sites/products/documentation/studio/composer/en-us/2011Update/compiler_c/optaps/common/optaps_openmp_thread_affinity.htm
23. Intel Corporation. A case study comparing AoS (Arrays of Structures) and SoA (Structures of Arrays) data layouts for a compute-intensive loop run on Intel® Xeon® processors and Intel® Xeon Phi™ product family coprocessors. [http://software.intel.com/en-us/articles/a-case-study-comparing-aos-arrays-of-structures-and-soa-structures-of-arrays-data-layouts]
24. Intel Corporation. Advanced Intel Xeon Phi Coprocessor Workshop, Extracting Vector Performance with Intel Compilers, September 2012.
25. Intel Corporation. Advanced Intel Xeon Phi Coprocessor Workshop, Intel Math Kernel Library 11.0. Support for Intel Xeon Phi Coprocessor, September 2012.
26. Intel Corporation. Advanced Intel Xeon Phi Coprocessor Workshop, Performance Tuning for Intel Xeon Phi Coprocessors, September 2012.

27. Intel Corporation. Beginning Intel Xeon Phi Coprocessor Workshop, Offload Compilation, September 2012.
28. Intel Corporation. Beginning Intel Xeon Phi Coprocessor Workshop, Optimization, September 2012.
29. Intel Corporation. Intel® C++ Compiler XE 13.1 User and Reference Guides: User-mandated or SIMD Vectorization [http://software.intel.com/sites/products/documentation/doclib/iss/2013/compiler/cpp-lin/GUID-42986DEF-8710-453A-9DAC-2086EE55F1F5.htm].
30. Intel Corporation. Intel® C++ Compiler XE 13.1 User and Reference Guides: SIMD [http://software.intel.com/sites/products/documentation/studio/composer/en-us/2011Update/compiler_c/cref_cls/common/cppref_pragma_simd.htm].

8 Course support

The following software and hardware are used during course study:

Hardware

"Lobachevsky" supercomputer with the Intel Xeon Phi coprocessors is used.

Software

Intel Parallel Studio XE (C/C++ Compiler for Intel Xeon Phi, Intel Amplifier, Intel MKL) is used.

9 Authors

The course was developed by I.B. Meyerov and S.I. Bastrakov.

Associate prof. _____ Iosif Meyerov

Junior researcher _____ Sergey Bastrakov

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