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 GPU PROGRAMMING

Bachelor program "Fundamental Informatics and Information Technologies"

Specialization

General part

Professional block

Elective disciplines

Б3.В.ДВ.9

Degree

Bachelor

Education form

Full-time attendance

Nizhni Novgorod

2014

1. Course objective

This course is an introduction to GPU programming. It considers most popular technologies of GPU programming, GPU architecture, thread execution and memory hierarchy, basic optimization principles. We focus on CUDA and present basic features of CUDA C programming language as well as optimized CUDA libraries CUBLAS, CUFFT and CURAND. Theoretical materials are accompanied by a wide set of examples and practical problems.

The main objective of the course is to introduce basic principles and acquire skills of GPU programming.

It involves solving the following **problems**:

1. Overview of GPU programming technologies.
2. Study of GPU architecture.
3. Getting familiar with CUDA C programming languages. Mastering writing kernels and device functions, workload distribution, and data exchanges between host and device memory.
4. Getting acquainted with examples of implementation and optimization of applications on GPU.
5. Using optimized CUDA libraries: CUBLAS, CUFFT, CURAND.

2. Course position in the bachelor program

The course is developed for the 4-th year students and is given in the 8-th semester. This course belongs to the elective disciplines of the professional block.

The course is oriented on students with basic skills of C/C++ program development and familiar with basic concepts of parallel programming. Experience with OpenMP, TBB and MPI is not required, but is beneficial. All necessary information of GPU architecture and programming techniques is included in the course materials. The course is oriented on students with basic skills of C/C++ program development and familiar with basic concepts of parallel programming. Experience with OpenMP, TBB and MPI is not required, but is beneficial. All necessary information of GPU architecture and programming techniques is included in the course materials.

The course materials were developed and translated to English in 2014 as part of the "5-100 project" by 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 GPUs and be able to use it for applied applications.

Will know architecture, programming and optimization techniques of GPU.

Will Be Able To use programming and optimization techniques of GPU for applied applications development.

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

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	General purpose computing on GPU	8	1	2	–	–	1	Test
2	CUDA C.		2	2	–	–	1	Test
3	Vector addition		3	–	–	2	1	Test
4	Numerical integration of heat equation		4	–	–	2	1	Test
5	CUDA thread execution and memory hierarchy		5	2	–	–	2	Test
6	Optimization of CUDA applications		6	2	–	–	2	Test
7	Matrix multiplication		7	–	–	2	2	Test

8	CUDA libraries		8	2	–	–	2	Test
9	CUDA Libraries. Minimal residual method. Convolution		9	–	–	2	2	Test
10	Monte Carlo integration and option pricing		10	–	–	2	2	Test
	TOTAL:			**10**	–	**10**	**16**	**Final assessment form – exam**

4.2. Course description

Course content is as follows:

1. General purpose computing on GPU

General purpose computing on GPU (GPGPU). Heterogeneous computing. Reasons why computing on GPU became so popular. Overview of programming technologies that can be used for GPGPU.

2. CUDA C

Basics of CUDA C programming language. Function qualifiers. Inline variables. General concepts of CUDA: thread, block, kernel. Creating simple CUDA programs.

3. Vector addition

Simple implementation of vector addition and saxpy using CUDA. Data alignment and workload distribution.

4. Numerical integration of heat equation

Simple implementation of numerical integration of 2D heat equation using explicit scheme. Data alignment and workload distribution.

5. CUDA thread execution and memory hierarchy

GPU architecture. Thread execution, warps, SIMT. Memory hierarchy: global, local and shared memory, registers.

6. Optimization of CUDA applications

Optimization of CUDA applications. Optimization of parallel reduction implementation.

7. Matrix multiplication

Implementation and analysis of naive matrix multiplication algorithm. Optimized block algorithm using shared memory.

8. CUDA libraries

CUDA libraries. CUBLAS. CUFFT. CURAND. Using to develop and optimize applications.

9. CUDA Libraries. Minimal residual method. Convolution

Application of CUBLAS and CUFFT libraries to solve practically important problems. Implementation of minimum residual method for iterative SLA solving. Convolution of complex signals.

10. Monte Carlo integration and option pricing

Application of CURAND library for Monte Carlo method applied to two problems: integration and option pricing.

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 architectural differences between CPUs and GPUs.

2. Enlist main GPU programming technologies.
3. Find an example that proves that load balancing is required for efficient heterogeneous computing.

Module 2

4. What are three main components of CUDA C extensions?
5. Enlist CUDA C keywords learned from this lecture and explain their meaning.
6. Write a kernel for matrix-vector multiplication and code to invoke it.

Module 3

7. Implement axpy operation for complex numbers.
8. Modify axpy kernel so that it will work for any amount of block and threads per block.

Module 4

9. Create implementation of the kernel using 1D indexes.
10. Implement a version with non-stationary boundary condition.

Module 5

11. What are the ways of communication between threads in the same block and in different blocks?
12. What are main differences between global and shared memory?
13. Describe how conditional statements are executed in SIMT.

Module 6

14. Describe main optimization techniques for CUDA applications.
15. Which factors are important in evaluation how suitable is an algorithm for GPU?

Module 7

16. Modify block algorithm so that each thread computes several elements instead of 1.
17. Empirically find optimal block size.
18. Implement Strassen matrix multiplication algorithm using the developed block algorithm implementation for small enough matrices. Compare performance of Strassen and block algorithms depending on matrix size.

Module 8

19. Find out which functionality does CUBLAS provide. Compare it with other BLAS implementations.
20. Find out which functionality does CUFFT provide. Compare it with other FFT implementations.
21. Find out which functionality does CURAND provide. Compare it with other PRNG implementations.

Module 9

22. Modify implementation of minimal residual method so that stop condition is $||Ax - b|| < \varepsilon$.
23. Implement convolution algorithm via FFT on CPU using libraries fftw or Intel MKL. Compare performance with GPU implementation.

Module 10

24. Generalize our Monte Carlo integration for integrals of arbitrary dimension.
25. Implement three versions of Monte Carlo option pricing on GPU: naive, using CURAND external, using CURAND internal. Compare performance of these versions.
26. Modify implementation to compute prices of several options with different parameters.
27. Create multi-GPU implementation for multi-option case, each GPU handles one or several options.

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. Sanders J., Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming. – Addison-Wesley Professional, 2010. – 312 p.
2. Farber R. CUDA Application Design and Development. – Morgan Kaufmann, 2011. – 336 p.
3. NVIDIA CUDA C Programming Guide. [http://docs.nvidia.com/cuda/cuda-c-programming-guide/].

7.2 Further reading

4. GPU Computing Gems Emerald Edition, ed. Wen-mei W. Hwu. – Morgan Kaufmann, 2011. – 886 p.
5. NVIDIA CUDA C Best Practices Guide [http://docs.nvidia.com/cuda/cuda-c-best-practices-guide#axzz3JRcPurfI].

6. NVIDIA CUBLAS Documentation [http://docs.nvidia.com/cublas/index.html#axzz3JRcPurfI].
7. NVIDIA CUFFT Documentation [http://docs.nvidia.com/cufft/index.html#axzz3JRcPurfI].
8. NVIDIA CURAND Documentation [http://docs.nvidia.com/curand/index.html#axzz3JRcPurfI].

8 Course support

The following software and hardware are used during course study:

Hardware

"Lobachevsky" supercomputer with GPU is used.

Software

NVIDIA CUDA (CUDA C compiler, CUDA profiler) is used.

9 Authors

The course is developed by S. Bastrakov.

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