|  |  |
| --- | --- |
|  |  |

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”

**PROGRAMMING AND OPTIMIZATION   
FOR INTEL XEON PHI**

*Lecture 2. Program execution on Intel Xeon Phi.   
Computational models on Intel Xeon Phi*

Nizhni Novgorod

2014

# Objectives

The objective of this lecture is to give an idea of different ways of development and running applications on Intel Xeon Phi coprocessors.

# Abstract

This lecture considers principles of program execution and models of organization of computations using Intel Xeon Phi coprocessors. We describe system software that responsible for program execution on Xeon Phi. We briefly describe Intel Manycore Platform Software Stack and symmetric communication interface. Offload, coprocessor-only and symmetric models are overviewed. We discuss ways to develop applications for Xeon Phi.

# BRIEF OVERVIEW

The first segment of the lecture is devoted to system software responsible for program execution on Intel Xeon Phi. This software is oriented on high performance computing applications that take full advantage of simultaneous execution of hundreds of threads. The software is intended for systems with PCI Express bus working under Linux and Windows operating systems. Intel Xeon Phi coprocessor is a high performance manycore system implemented as a board plugged into PCI Express port. A host operating system uses a driver to communicate with coprocessor as a PCI Express device. Libraries provide basic functionality such as coprocessor location, data transfer between host and coprocessor, loading executables to Xeon Phi and application launching. Instrumental tools allow to control the settings of a coprocessor, retrieve the information about its state, refresh coprocessor memory, etc.; one can open a terminal in coprocessor and run programs using ssh. From the host operating system point of view coprocessor is a separate computational SMP-domain loosely connected to the main processors of the system. Since a coprocessor is integrated to the base system as PCI Express device, there are several ways of communication between the host and coprocessor, which allows several models of using the coprocessor. Xeon Phi supports standard high performance computing API: TCP/IP sockets, MPI, OpenCL. Additionally there are specialized interfaces to access features of the coprocessor, e.g. SCIF API (Symmetric Communication Interface API) provides interaction between a host system and a coprocessor and high level APIs use it for data transfers. Then we illustrate interconnections between different API implemented in MPSS (Intel MIC Architecture Manycore Platform Software Stack), describe the process of coprocessor start and OS loading. The coprocessor OS is based on a standard Linux kernel with minimum changes for a support of the new architecture and lack of some components that are standard for computational systems. The coprocessor OS provides standard capabilities such as process creation and termination, thread scheduling, management of memory, power and configuration, etc. Specific components of the coprocessor are controlled via special driver. We consider Symmetric Communication Interface (SCIF) which is a basic mechanism of interaction between host processors and the coprocessor and several coprocessors in one system. SCIF uses DMA mechanism on coprocessor side and mapping of physical memory to virtual address space of any process running of a host or coprocessor. Interaction between a pair of SCIF clients is based on direct access to memory of each other, in particular, two coprocessors communicate without using host memory.

The second segment of the lecture is devoted to execution models on Intel Xeon Phi. Xeon Phi architecture supports several modes of coprocessor usage that could be combined to achieve optimal performance depending on properties of a specific problem. A process can be running either on host OS or coprocessor OS; depending on the mode being used one can utilize only host processors, or only coprocessors, or both. Intel MPI library for Linux OS corresponds to MPI-2.1 and is based on MPICH2 and MVAPICH2 implementations. Intel is planning to provide full functionality of MPI implementation for all Intel Xeon and Intel Xeon Phi configurations in order to provide uniform environment for developers. Currently there is a lack of support of some functionality, namely, dynamic process control, MPI file IO, one-sided transfers to a passive receiver (receiver does not call MPI routines).

There are two modes of application execution: Offload mode and MPI mode.

In the offload mode one of the following approaches is used:

* MPI processes are running on Xeon processors of the host system, computational kernels can be running on Xeon Phi. This model is supported at C, C++, and Fortran Intel compilers for MIC, Intel MKL, and Intel MPI for Linux from version 4.0. Update 3.
* MPI processes are running on Xeon Phi coprocessors and are capable of running functions on host Xeon processors. This model is currently not supported by compilers and runtime libraries.

In the MPI mode the host system and each Xeon Phi coprocessor are considered as separate independent nodes, MPI processes are running on all devices in all possible combinations. There are three main models:

* Symmetric model – MPI processes are running on both host processors and coprocessors. This model is natural for heterogeneous cluster systems, but requires adequate load balance between processors and coprocessors. The following two models are special cases of this one.
* Coprocessor-only model – all MPI processes are running on coprocessors.
* Host-only model – all MPI processes are running on host processors, coprocessors are not utilized.

The lecture describes all models in details.

The third segment of the lecture discusses application development for Intel Xeon Phi. Essentially it requires the same expertise and skills as parallel programming for distributed memory multicore systems. One can make use of the following programming tools:

* Intel Parallel Studio XE 2013, Intel Cluster Studio XE 2013, Intel(R) SDK for OpenCL Applications XE 2013 Beta, gcc, etc.;
* Intel Math Kernel Library (Intel MKL), Intel Threading Building Blocks (Intel TBB), Intel Integrated Performance Primitive (Intel IPP), Intel MPI for Linux, MPICH2, Boost, etc.;
* Intel Debugger, gdb, totalview, profiling tools (part of Intel Parallel Studio), virtualization tools, such as xen, etc.

Building of heterogeneous applications is done on the host system. In the Offload mode all offload-blocks are compiled in two versions: for the host system and for the coprocessor. The compiler creates executables and/or libraries containing full code for processor and coprocessor. Xeon Phi coprocessor presence is checked in the runtime during first offload code execution. If a vacant Xeon Phi coprocessor is found, the binary executable is loaded to the coprocessor and proper libraries are initialized, then offload-code is executed. Otherwise host version of the code is executed. Thus an application works on systems with Xeon Phi coprocessors and without them.

Finally, we discuss several versions of array sum implementation for different execution models.

# FOR STUDENTS

Intel Xeon Phi architecture can be studied on materials of [6] (Chapter 8, Coprocessor Architecture).

# References

1. 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].
2. User and Reference Guide for the Intel® C++ Compiler [http://software.intel.com/en-us/compiler\_14.0\_ug\_c]
3. Reinders J. An Overview of Programming for Intel Xeon processors and Intel Xeon Phi copro-cessors. [http://software.intel.com/en-us/blogs/2012/11/14/an-overview-of-programming-for-intel-xeon-processors-and-intel-xeon-phi].
4. Loc Q Nguyen 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].
5. Intel Xeon Phi Coprocessor System Software Developers Guide. [http://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-system-software-developers-guide].
6. Jeffers J., Reinders J. Intel Xeon Phi Coprocessor High Performance Programming. Morgan Kaufmann, 2013.

# Individual work

1. Implement dot product using various parallel programming technologies.
2. Implement dense matrix-vector multiplication using various parallel programming technologies.
3. 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.