Performance comparison for NVIDIA CUDA and Intel Xeon Phi

STUDENT: PETRA LONCAR

FESB, UNIVERSITY OF SPLIT

May, 2016

Contents

- Introduction
- NVIDIA CUDA
- Intel Xeon Phi
- Conclusion
- tCSC 2016

Introduction

- today’s engineering and scientific problems and challenges require ever more computer power to be able to successfully process large amounts of data and solve the problems of modeling and simulation
- image processing, signal processing, physics simulation and financial calculation algorithms are accelerated by parallel data processing
- platform for parallel computing are based on clusters of multicore nodes
- GPU NVIDIA CUDA and Intel Xeon Phi coprocessor

NVIDIA CUDA

- CUDA - platform for parallel computing and programming model invented by NVIDIA
- designed to solve complex computational problems more efficiently than a CPU alone
- SIMT (Single-Instruction, Multiple-Thread) based model that enables the execution of blocks of threads
- supports the use of programming languages C, C++, Fortran, OpenACC
- CUDA C provides a set of extensions to the C language and the runtime library
- kernel - code running on the GPU with installed CUDA
- kernel is defined using declaration __global__

- thread within a block share data through shared memory and synchronize their execution by calling function __syncthread()
- using shared memory, a global memory bandwidth is saved
- threads within a block share data through the shared memory and are synchronized to coordinate their communication and memory accesses
- thread of different blocks are independent of each other and communicate via slower global memory
- serial code running is executed by thread on the CPU (host); a parallel code, the kernel, is executed on a GPU (device)
- GPU memory is allocated as a linear memory using cudaMalloc() function and de-allocated by cudaFree() function
- cudaMemcpy() function allows transfer of data between the memory of the CPU and GPU
- nvcc compiler simplifies compiling C code and kernel code in binary code for execution on the GPU

Intel Xeon Phi

- Intel Xeon Phi coprocessor is based on the MIC architecture
- offers efficient scaling, use of vectors, local memory
- the greatest degree of parallelism can be achieved by simultaneous application of vectorization using a vector instruction and scaling on a large number of coprocessor's core
- more than 50 cores and at least 8 GB of RAM
- based on Linux operating system
- allows the use of 64-bit x86 commands based on Single Instruction, Multiple Data (SIMD) model
- applications should use the Intel Xeon processor and Intel Xeon Phi coprocessor

- two computing modes:
  - offload
  - native
- advantage of offload mode - performance benefits from the use of processor (host) memory and coprocessor (device) memory devices coprocessors
- offload mode is supported by Intel's C compiler (ICC) and pragmas to control Intel Xeon Phi threads
- high-level programming languages: C, C++, Fortran
- programming models: OpenMP, Intel Cilk Plus, Intel Threading Building Block (TBB)

Conclusion

- timing results are compared for both platforms
- on CUDA, the algorithm that takes advantage of shared memory is more complex, but execution time is shorter
- benefit obtained by using shared memory increases as the data size grows
- CUDA lack is the limited memory
- main limitation of coprocessor is limited memory
- the offload mode is frequently used because it can use the host processor memory
- to get a good performance out of the coprocessor, programmer needs to utilize all cores and hardware threads and utilize vectorization
- significant expertise, time, deep knowledge of the target hardware

tCSC 2016

- role of compilers and flags
- quality test
- memory architectures and technologies
- threads management
- data storage

Thank you for your attention!