Teaching Parallel Programming for Time-Efficient Computer Applications

Abstract

Academic research and engineering challenge both require high performance computing (HPC), which can be achieved through parallel programming. The existing curricula of most universities do not properly address the major transition from single-core to multicore systems and sequential to parallel programming. They focus on applying application program interface (API) libraries and open multiprocessing (OpenMP), message passing interface (MPI), and compute unified device architecture (CUDA)/GPU techniques. This approach misses the goal of developing students' long-term ability to solve real-life problems by thinking in parallel. In this article, a novel approach is proposed to teach parallel computing that will prepare computer application developers for present and future computation challenges. Using multicore/manycore architecture and popular challenging problems from areas like computer science, proposed approach teaches how to analyze and develop efficient solutions for the problems. As preliminary work, some multithreaded parallel programs are introduced to computer science and engineering students. Based on the feedbacks from information technology (IT) professionals and Student Outcomes Assessment Reports, proposed approach has potential to provide adequate knowledge so that students can fulfill the growing industry
demands for HPC. Based on the Steady State Heat Equation experiment, CUDA/GPU parallel programming may achieve up to 241x speed up factor while simulating heat transfer on a 5000x5000 thin surface.

References

- Multicore Programming Education. 2009. Workshop on Directions in Multicore Programming Education. Washington DC.
- Brown, R. 2010. A strategy for injecting parallel computing education throughout the computer science curriculum. NSF CCLI Award 0942190.

Index Terms

Computer Science Distributed Systems

Keywords

CUDA/GPU technology; multicore architecture; OpenMP; Open MPI; parallel programming;