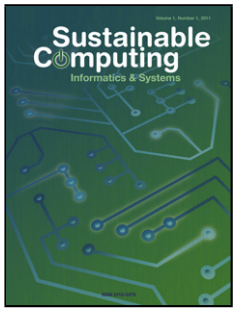

Title: A Survey on Techniques for Cooperative CPU-GPU Computing

Authors: Raju K, Niranjan N. Chiplunkar

PII: S2210-5379(17)30331-1
DOI: https://doi.org/10.1016/j.suscom.2018.07.010
Reference: SUSCOM 266

To appear in:

Received date: 30-8-2017
Revised date: 18-4-2018
Accepted date: 21-7-2018

Please cite this article as: K R, Chiplunkar NN, A Survey on Techniques for Cooperative CPU-GPU Computing, *Sustainable Computing: Informatics and Systems* (2018), https://doi.org/10.1016/j.suscom.2018.07.010

A Survey on Techniques for Cooperative CPU-GPU Computing

Raju K
Dept. of CS& E
NMAMIT, Nitte, Karkala
Karnataka, India
rajuk@nitte.edu.in

Niranjan N Chiplunkar
Dept. of CS& E
NMAMIT, Nitte, Karkala
Karnataka, India
nchiplunkar@nitte.edu.in

Highlights:

- Effective utilization of resources in CPU-GPU systems improves the performance and energy efficiency.
- Researches focus on collaboratively executing single or multiple tasks on CPU and GPU.
- Runtimes for portability of code and task allocation schemes improve the possibility of collaborative execution.
- Various research works in the area of cooperative CPU-GPU heterogeneous computing are reviewed and the research gaps are outlined.

Abstract;
Graphical Processing Unit provides massive parallelism due to the presence of hundreds of cores. Usage of GPUs for general purpose computation (GPGPU) has resulted in execution speedup and energy efficiency. In addition, the modern CPUs also possess multiple cores, with enormous computational power. In general, the current programming frameworks for CPU-GPU heterogeneous computing system do not facilitate the efficient utilization of computational resources so as to further improve the performance and reduce the energy consumption. Several researches have been carried out to improve the resource utilization and reduce the energy consumption of heterogeneous systems by cooperatively performing the computation on both multicore CPUs and GPUs. In this survey paper we review the various techniques available for CPU-GPU cooperative computing.

***Index Terms*—Heterogeneous System; CPU-GPU; Multicore CPU; Cooperative execution; Task allocation; Energy saving**

I. INTRODUCTION

As the clock frequency and power consumption of unicore processors reached its peak, processor architects switched to the concept of multicore processors. A multicore processor is one in which two or more execution cores are placed in a single chip. The operating system perceives each of these execution cores as a discrete physical processor with all the associated execution resources. The main objectives of multi-core architecture are to reduce power consumption, and enable efficient simultaneous processing of multiple tasks.

OpenMP [82] is an application programming interface that provides the developers with a simple and flexible interface making the development of parallel applications on multicore processors easier. OpenMP is based on the concept of multi-threading wherein a master thread forks several slave threads and the work is distributed amongst them. The threads run concurrently and are scheduled to run on different processor cores.

The Graphics Processing Unit is a co-processor, originally developed to accelerate graphical applications. Modern GPUs consists of hundreds of cores and capable of processing thousands of threads. The cores of a GPU execute the same instruction sequence in lockstep but possibly on different data elements. GPU computing has become one of the interesting areas of research due to the suitability of the architecture for executing applications that exhibit massive parallelism.

Though the primary reason of introducing GPU was for graphical purposes, it is now being used for general purpose parallel computing. During 2006-2007, a massively parallel architecture was introduced by NVIDA called as "CUDA" which revolutionized the concept of GPGPU programming. CUDA C/C++ [80] is an extension of the C/C++ programming languages