Public Abstract
First Name:Da
Middle Name:
Last Name:Li
Adviser's First Name:Michela
Adviser's Last Name:Becchi
Co-Adviser's First Name:
Co-Adviser's Last Name:
Graduation Term:SS 2016
Department:Engineering
Degree:PhD
Title:Facilitating Emerging Applications on Many-core Processors

Over the last decade, many-core Graphics Processing Units (GPUs) have been widely used to accelerate a variety of applications. Meanwhile, Intel has released its Xeon Phi Coprocessor, which is equipped with more than fifty x86 cores, each supporting four hardware threads. Despite their widespread use, many-core processors are still considered relatively difficult to program, in that they require the programmer to be familiar with both parallel programming and the hardware features of these devices.

Due to their massive computational powers, many-core processors have been successfully used to parallelize a wide variety of dense matrices and vectors based applications. These extensively investigated problems are mainly from linear algebra, stencil computations, image processing and so on. However, many established and emerging problems have not yet been fully explored. Some of these applications use irregular algorithms/operations (e.g., dynamic programming), while others are based on irregular data structures, such as graphs. It has been shown that these emerging applications do exhibit certain degree of static and runtime parallelism, but are relatively hard to parallelize.

My research focuses on addressing important issues related to the deployment of emerging applications on many-core processors. In particular, we proposed efficient GPU implementations for large-scale pairwise sequence alignment and integrated proposed GPU kernels into a hybrid MPI-CUDA framework for CPU-GPU clusters. we also targeted graph- or tree-based applications and proposed: (1) unifying programming interfaces for many-core processors (2) runtime support for efficient execution on irregular datasets and (3) compiler support for efficient mapping of applications onto hardware. Finally, we conducted a comprehensive study of performance, memory footprint and power consumption on various platforms and extended existing central processing units (CPU) only or graphic processing units (GPU) only CNNs learning methods to CPU-GPU cooperative ways. We also implemented a virtual memory and integrated into Caffe to facilitate training large CNN models with limited GPU memory.