Public Abstract

First Name:Kittisak
Middle Name:
Last Name:Sajjapongse
Adviser's First Name:Michela
Adviser's Last Name:Becchi
Co-Adviser's First Name:
Co-Adviser's Last Name:

Graduation Term:SS 2015
Department:Computer Engineering
Degree:PhD

Title:HIERARCHICAL SCHEDULING AND UNIFORM ACCESS PROGRAMMING FRAMEWORKS FOR HETEROGENEOUS CPU-GPU COMPUTING CLUSTERS

The advance of the GPU hardware architecture has made GPUs attractive devices for general- purpose computing. Modern GPUs are equipped with an increasing number of cores, a flexible memory hierarchy, and a large memory capacity. While the computational power of modern GPU devices has allowed their introduction in high-performance computing (HPC) clusters and the efficient processing of ever larger workloads, existing software components for HPC clusters still offer basic support for hardware heterogeneity and often cause performance limitations in the presence of GPU devices. In particular, two kinds of limitations are associated with these software components: runtime support and programmability. We found that these limitations are due to the fact that existing software frameworks for heterogeneous clusters treat GPUs as dedicated coprocessor devices.

In this dissertation, we propose two software frameworks for addressing the performance and hardware underutilization issues found in heterogeneous CPU-GPU clusters as well as increasing their programmability. Our frameworks provide a uniform view of compute resources and treat CPUs and GPUs equally as first-class resources, allowing efficient management of heterogeneous compute resources. First, we propose a hierarchical scheduling framework consisting of a node-level runtime and a cluster-level scheduler that provides abstraction of heterogeneous compute resources at different granularities. This hierarchical framework targets existing applications and does not require their modification. In the node-level runtime, we identify and design mechanisms, such as virtual GPUs, GPU virtual memory, dynamic load balancing and pre-emption, which are necessary to support efficient sharing and load balancing schemes for GPUs within a compute node. In the cluster-level scheduler, we introduce mechanisms to abstract compute nodes and perform load balancing in concert with the node-level runtime. Our hierarchical scheduling framework allows supporting different load balancing policies and does not require additional inputs (such as profiling information) from users. Second, we propose a programming framework based on a novel memory and execution model. Our memory model hides disjoint addressing spaces (corresponding to different CPUs, GPUs and compute nodes) and provides a view of a single virtual memory space that can be accessed by all compute resources in a heterogeneous cluster. Our execution model provides uniform access to compute resources and allows our framework to treat all CPUs and GPUs equally and to access data in the virtual memory space.