Alexander Powell

10.12.2016

CS 680

“Automatic CPU-GPU Communication Management and Optimization” Critique

Summary

At a baseline level, most PCs with GPUs are able to achieve a few hundred GFLOPS. However, with real applications that are specially designed to use the GPU architecture, it’s not uncommon to achieve speedups somewhere between 4 to 100 times the baseline. However, it can often be difficult to parallelize this code for a CPU-GPU memory architecture. This paper presents the first fully automatic system for managing and optimizing CPU-GPU communication (called the CGCM). It consists of a run-time library and a set of compiler transformations that work together to both manage and optimize the communication. Communication management is often a problem for distributed memory systems. Optimizing communication can have a profound impact on program performance. At a high level, the goal of optimization is to avoid cyclic communication which causes the CPU to wait for the GPU to transfer memory and the GPU to wait for the CPU to send more work.

Pros & Cons

The method proposed is novel because it breaks new ground in two ways; it’s the first fully automatic communication *management* system as well as the first *optimization* system. Also, the authors referenced a lot of existing prior works, but I think it would have been more helpful to have that at the beginning instead of the end where they put it. I thought the writing style started out fine but towards the end they were using too much jargon and too many acronyms to keep track of sometimes, which made the paper rather hard to read.

Questions and Concerns:

1. Based on the paper or any other sources you’ve referenced, are there any downsides to this method of optimization and management and why?
2. On page 150 it says that most programs show performance improvements and none have worse performance using CGCM. After the CGCM optimization, is it possible for a program’s performance to decrease?