Parallel Computing with Multiple GPUs on a Single Machine to Achieve Performance Gains

Robert Gulde[1]
Georgia State University
robertgulde@motorola.com

Michael Weeks[2]
Georgia State University
sowen@cs.gsu.edu

Scott Owen[3]
Georgia State University
mweeks@cs.gsu.edu

Yi Pan[4]
Georgia State University
pan@cs.gsu.edu

1 Introduction

Using a technique similar to cluster based computing, this research demonstrates the use of multiple Graphics Processing Units (GPUs) to achieve enhanced performance on a single processor workstation. Past examples of performance gains achieved by cluster based computing in order to speed up the rendering process have been demonstrated. However, these approaches have neglected parallel processing capabilities through multiple graphics cards with hardware acceleration on a single CPU based computer. In this research a technique was developed to utilize multiple threads to drive hardware accelerated graphics cards.

Performance gains achieved by this technique demonstrated that GPUs on a single CPU system follow both Gustafson's Law of constant time as work load scales with the number of GPUs and Amdahl's Law of speed up achieved through work division across processing elements.

2 Exposition

The technique focused on in this research was to have individual threads load and execute the rendering pass within their own rendering and device contexts. This allows the individual threads to block while the graphics hardware executes OpenGL rendering calls. When a thread blocks on the graphics drivers, other threads then load and execute rendering instructions. The performance increase is due to running the GPUs simultaneously.

Figure 1 demonstrates the test system in which a main controlling thread initiates rendering threads and then starts the rendering pass via event signaling. Following this each of the threads loads and executes a Cg test a barrier by first signaling the main thread via a semaphore then blocking on the next event signaling algorithm. As each thread finishes, they synchronize on. Figure 2 demonstrates the mitigation of overhead effects on scaled workload for 4 GPUs as instruction count is increased.

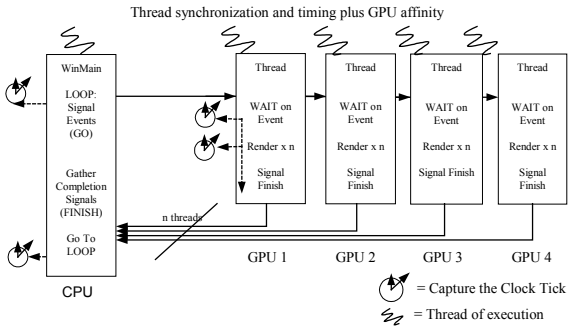

Figure 1. Synchronization of multiple rendering threads.

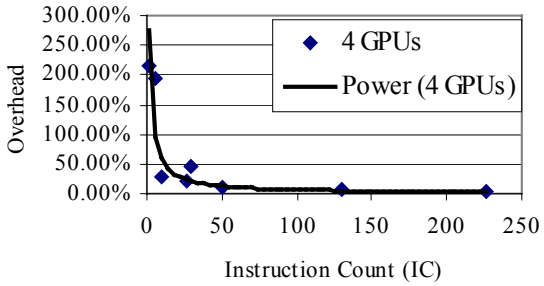

Figure 2. Effect of Overhead reduced as a function of the instruction count, for 4 GPUs.

Four nVidia FX5200 PCI based graphics cards were utilized in this research. Several algorithms and problem sizes were explored utilizing this test framework. Larger and more complex algorithms exhibited greater amounts of speed up or constant time results. It was found that four GPUs could achieve four times the work with only a 1.82% time penalty, while taking a single set of data and dividing the work by 4 GPUs resulted in a speed up of 355%.

3 Conclusions

This research demonstrates that 4 GPUs driven in parallel can achieve lower bound improvements of either 4 times the work with an 11% time penalty, or a speed up of 352% by dividing the work across 4 PCI based GPUs.