Dynamic Load Balancing for Heterogeneous Systems

Kenneth Lee, Kyle Morgan

***Abstract*—Leveraging maximimum performance from specialized hardware is one of the major problems facing computer scientists today. General-purpose computing on graphics processing units (GPGPU) is a technique that is becoming increasingly popular for handling computations typically performed by the CPU. However, many GPGPU algorithms place the entire workload on the GPU, leaving the CPU idle for the duration of the computation. This paper investigates the effect various load-balancing schemes across the CPU and GPU have on overall application performance. Additionally, the discrete and fused GPU architectures, and their impact on performance, will be investigated.**

I. Introduction

LEVERAGING the maximum performance from specialized hardware is one of the major challenges computer scientists face today. The unique hardware architecture of the Graphics Processing Unit (GPU) gives it a place as a massively parallel hardware accelerator. However, because of the difficulty in programming for this hardware, most of the effort is spent on trying to achieve performance from the GPU, while leaving the CPU unused. By also using the CPU to process a smaller part of the data we can achieve higher utilization of resources and therefore increase program performance.

GPU+CPU co-processing hardly a new idea. Jimenez et al. [4] present a dynamic library which can be used in a system to dynamically schedule tasks for GPU computation across various processes. The method describes only looks at optimizing usage of the GPU across the whole system, instead of optimizing a single applications performance. The StarPU system, on the other hand, presents a method of co-processing and load balancing on a per application basis [1]. They show that by using an appropriate model for the load balancing, they are able to achieve improvements in speed for applications which call for code acceleration multiple times. We intend to extend this work to improve the speed of each kernel execution by splitting the work of that kernel over the capable resources, similar to the schemes used by OpenMP. Both of these papers address data transfer times as a major cost of GPU computing. Using the new AMD Fusion architecture, which fuses the CPU and GPU onto a single die, we may be able to eliminate or greatly reduce this cost, which will impact the overhead of GPU+CPU co-processing.

We expect that by using GPU+CPU processing we will improve the performance of the application, when compared to CPU-only or GPU-only. We will measure performance of the application using a wall time clock. We also hypothesize that the fused GPU+CPU architecture will allow for lower overhead for load balancing.

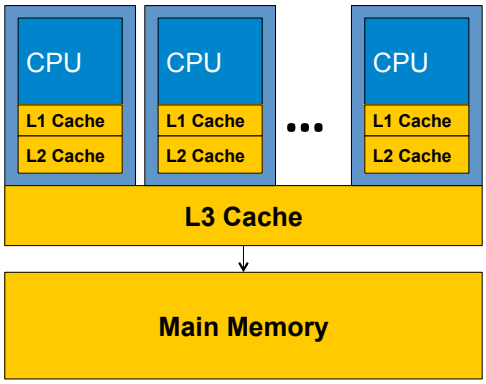

Fig. 1. CPU Architecture

II. Motivation

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

III. Architecture Overview

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum. Blah blah blah see figure 1.

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

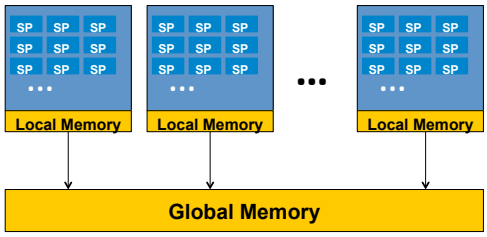

Fig. 2. GPU Architecture

IV. Conclusion

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

References

[1] C. Augonnet, S. Thibault, R. Namyst, and P.A. Wacrenier, "StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures" in *Proceedings of the 15th International Euro-Par Conference on Parallel Processing*, 2009, pp. 863-874.

[2] S. Che, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, S.H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing" in *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 44-54.

[3] A. Danalis, G. Marin, C. McCurdy, J.S. Meredith, P.C. Roth, K. Spafford, V. Tipparaju, and J.S. Vetter, "The Scalable Heterogeneous Computing (SHOC) benchmark suite" in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, 2010, pp. 63-74.

[4] V.J. Jimnez, L. Villanova, I. Gelado, M. Gil, G. Fursin, and N. Navarro, "Predictive Runtime Code Scheduling for Heterogeneous Architectures," in *Proceedings of the 4th International Conference on High Performance Embedded Architectures and Compilers*, 2009, pp. 19-33.

Appendix
Team Member Contributions

Kenneth Lee did A, B, C. Kyle Morgan did X, Y, Z.