Dominique Pennington

Research Directory: Craig Tanis

Accelerating Ray Tracers

Introduction

Arguably, the three main sources of limitation for computational performance by way of CPU clock speeds are power consumption, heat production, and propagation delay. The impressive speeds to which we have pushed CPU clocks have turned what were once theoretical concerns into reality of hardware development. As more megahertz are produced there is an increasingly direct correlation to power consumption and unmanageable heat production. Additionally, propagation delays of the chipset are now growing longer than the clock speeds themselves.

The industry has begun a shift away from the pursuit of clock speed and toward a trend of providing more processing cores at relatively slower clock speeds with the intent of dividing the labor of processing among a distinct set of cores which can work together to complete given task.[1] This paradigm requires code that is more parallel in nature in order to better suit the shift.

As the performance of single threaded applications continues to see less benefit from general-purpose CPU innovations, it becomes necessary not only parallelize software, but hardware as well. Hardware accelerators take this focus on parallelism to an extreme. An accelerator or “coprocessor” is not a piece of general-use hardware, but a microarchitecture specifically designed to more efficiently handle compute-intensive tasks which are often highly parallel and vectorized in nature.

The objective of this research is to compare the performance of an OpenACC parallelized Ray Tracer using two competing models of hardware acceleration. We hope draw out dissimilarities between the two architectures and their compilers for similarly complex and demanding algorithmic applications.

Background

Ray Tracing

Ray tracing is a rendering process by which a complex three dimensional scene is generated by a pseudo-realistic simulation of light interaction upon objects. The scene is comprised of the objects lying in front of a "camera" with a viewplane orthogonal to the heading direction of the camera's viewing frustum. A large number of rays are cast through each pixel of the viewplane toward the scene. Upon intersection with an object, a resultant ray leaves the object and the object's properties are stored. The stored properties will later be combined with further intersected object's properties to determine the final color of the pixel.[2]

The Ray Tracing algorithm is known as being "embarrassingly parallel" as it takes little effort to refactor code in order to separate the algorithm into multiple tasks which can be easily distributed to run in parallel, thereby optimizing it for running on accelerator/coprocessor architectures due to their parallel nature.

OpenACC

OpenACC is a set of direct-based extensions to standard Fortran, C, and C++ that enables offloading of data, compute-intensive loops, and regions of code from a CPU host to an attached accelerator device. The API enables a "one solution, many application" approach to parallelized applications. The API allows for a single implementation to be used across many different architectures, whether they be APUs, CPUs, CAPS, or GPUs. [3]

Through the insertion of compiler directives, a developer can command the compiler to map compute-intensive loops to parallel or vector execution units that will be optimized for the respective architecture. [4]

The focus of our parallelization directives will be on the identified regions of serial bottleneck in order to offload the most compute-intensive operations to the system's available hardware accelerator.

Proposed Research

Microarchitectures

The first architecture we have chosen is the the ARK/Skylake 6th Generation Intel® Core™ Processor. The Skylake was released in 2015 as a consumer level general purpose CPU. It was chosen in order to give contrast of our experimental groups by serving as the scientific control of a non-accelerated consumer-based system.

The non-GPU based accelerator we will research is the Intel MIC Accelerator (Many Integrated Core Architecture) Intel® Xeon Phi™ Coprocessor. The Xeon Phi architecture focuses primarily on highly-parallel, vector-intensive, and memory bound code, by which a large number of onboard Intel cores deliver a higher aggregate performance for highly-parallel code. [5]The Intel Phi will serve as the first architecture in the experimental group demonstrating the effects of OpenACC on code optimized for a CPU-based hardware accelerator.

The GPU-Based Nvidia Tesla "Kepler" K-20 is an accelerator built for scientific high-performance computing (HPC) workstations that focuses on maximizing the power efficiency and performance of computation on highly-parallelized code while increasing the programmability of the architecture as compared to their previous designs. The headline features of the Kepler include Dynamic Parallelism (enabling the GPU to dynamically spawn new threads without host CPU interaction) and Hyper-Q (allowing multiple CPU cores to launch work on a single GPU, decreasing CPU idle time and increasing GPU usage).[6] The Nvidia Kepler will serve as the second architecture in the experimental group demonstrating the effects of OpenACC on code optimized for a GPU-based hardware accelerator.

Experimental

The experiment will be to build a serial Ray Tracer and identify regions of the algorithm which are particularly compute-intensive - by nature of the serial approach. Once bottlenecks are identified focus will shift toward efforts of parallelizing those particularly slow regions by way of OpenACC. We will then apply the OpenACC parallelization of the algorithm to the various architectures in order to determine which platform is better suited for an OpenACC parallelized Ray Tracing application.

Conclusion

The primary implications of this research are in the fields of computer graphics and high performance computing. We hope to speak on the subject of high performance computing through the language of computer graphics.

The final project will take the form of a bound thesis between 30 and 50 pages as well as a technical demonstration in video form.

References

[1] Dongarra, J., Gannon, D., Fox, G., & Kennedy, K. (n.d.). The Impact of Multicore on Computational Science Software ... Retrieved March, 2016, from http://www.researchgate.net/

[2] Nery, A., Nedjah, N., & Franca, F. (2010, November). Massively Parallel Identification of Intersection Points for ... Retrieved March, 2016, from http://www.researchgate.net/publication/221312973_Massively_Parallel_Identification_of_Intersection_Points_for_GPGPU_Ray_Tracing

[3] Openacc-standard. (2013). OpenACC API. Retrieved March, 2015, from http://www.cray.com/

[4] OpenACC-standard. (n.d.). How does OpenACC work? Retrieved March 2016, from http://www.openacc.org/

[5] Intel. (2013). *The Intel Xeon Phi Product Family* [Product Brief]. Santa Clara, CA: Intel Corporation.

[6] NVIDIA. (2012). *NVIDIA Kepler GK110* [Pamphlet]. Santa Clara, CA: Nvidia.