CS 5220 Project Proposal - Investigating Whippletree

Xiang Long (XL483)

Whippletree[1] is a recently-proposed programming model for scheduling dynamic workloads on GPUs. It is an evolution of the traditional *megakernel* concept for GPU programming. Instead of rendering tasks themselves being loaded directly onto stream processors, a persistent thread that acts as a manager of tasks permanently resides on each stream processor while pulling and executing the actual tasks from a software-implemented queue. This allows for programmer-defined scheduling of tasks depending on the resources currently free on the GPU, and also saves on the overheads of many consecutive small tasks being ignited on the stream processors. Whippletree is a software implementation of improvements built upon past research into megakernels, and the authors claim that it consistently outperforms two of the traditional non-megakernel scheduling methods, Time-Sliced Kernels and Dynamic Parallelism, especially where the workload is predominantly heterogeneous.

The goals of the project are as follows:

- Understand the methodology of Whippletree and learn to schedule GPU compute tasks using its programming model. The reference Whippletree implementation can be found at `http://www.icg.tugraz.at/project/parallel/downloads`.
- For some task that is suitable for parallel computation on a GPU, compare the performance of Whippletree scheduling against a common non-megakernel scheduling method.
  - The Whippletree paper[1] contains detailed performance comparison data between Whippletree and the Time-Sliced Kernels and Dynamic Parallelism methods. The examples tested were procedural city generation, Reyes rendering, and volume rendering with irradiance caching. It would be interesting to now compare the performance on examples that are not related to rendering but nevertheless are often performed on GPUs, such as hashing or video encoding. These tasks may have a very different heterogeneity profile to rendering and could offset some of the heterogeneous task optimizations that Whippletree aims for.
  - At the very least the project could take one of the examples given in the paper and attempt to replicate the performance results.

As the reference Whippletree implementation is written in CUDA, this project would require fluency in or learning to program with CUDA and access to a CUDA-supporting Nvidia GPU.

References

[1] Markus Steinberger, Michael Kenzel, Pedro Boechat, Bernhard Kerbl, Mark Dokter, and Dieter Schmalstieg. Whippletree: Task-based scheduling of dynamic workloads on the GPU. *ACM Trans. Graph.*, 33(6):228:1–228:11, November 2014.

`http://data.icg.tugraz.at/~dieter/publications/Schmalstieg_286.pdf`