**Task Management for Irregular Workloads on the GPU**

Stanley Tzeng, Anuj Patney, and John D. Owens

---

### Motivation

We explore software mechanisms for managing irregular workloads on graphics processing units. Traditional GPU programming guidelines teach us how to efficiently program the GPU for data parallel pipelines with regular input and output. We present a strategy for solving task parallel workloads that can handle irregular workloads on the GPU.

### Problem

**Warp Size Work Granularity**

- **Problem:** We want to emulate task level parallelism on the GPU without loss in efficiency.
- **Solution:** We choose block sizes of 32 threads/block. Choosing this size has two major advantages for us:
  - Removes messy synchronization barriers. This is especially important when used in conjunction with Persistent Threads.
  - One can view each block as a MIMD thread where the execution is independent of other blocks, maintaining the efficiency of SIMD execution while giving us MIMD style granularity. We call these blocks processors. Several processors can reside on a SM.

**Uberkernels**

- **Problem:** Want to eliminate global kernel barriers for better processor utilization.
- **Solution:** Use the Uberkernel programming model. Uberkernels pack multiple execution routes into one kernel, effectively going through multiple stages of the pipeline within one kernel. When can pack multiple kernels into one kernel, we eliminate the explicit barrier and overhead between kernel launches.

### References

- **BLOOMER R. D., LEIDERSON C. E.:** Scheduling multithreaded computations by work stealing. Journal of the ACM 46, 5 (Sept. 1999), 720–748.
- **CERDA-NANCY D., TISGAS F.:** On dynamic load-balancing on graphics processors. In Graphics Hardware 2008 (June 2008), pp. 57–64.

### Results

- **Examples:** This graph shows our task donation memory scheme in terms of idle iterations. We define an idle iteration as any one processor either waiting for a lock or waiting for other processors to finish before it can terminate. Our scheme is roughly the same as a previous task stealing scheme, but it uses less memory. For more experiments and results, please see our paper.

---

### Conclusion

We have demonstrated how to build a system on the GPU that can deal with irregular workloads on a task size granularity. With our system we built a Reyes rendering pipeline which can achieve real-time frame rates. Novelty:

- Our work is the first to combine a work donation approach for workflow management with Uberkernels and persistent thread programming styles to exploit task parallelism and handle irregular workloads.

**Future Work:**

In the future we hope to explore how newer GPU hardware can help our hardware. AMDs are known to have much faster atomics than its predecessor. We hope that these advancements in hardware will allow us to explore further abstractions and models for designing different pipelines on the GPU.