Cost-based Workload Balancing for Ray Tracing on Multi-GPU Systems

Mario Rincón-Nigro* Zhigang Deng†

University of Houston

Keywords: Ray Tracing, Workload Balancing, Multi-GPU Computing

1 Introduction

Ray tracing is at the core of most techniques for creating realistic imagery. Parallel implementations of ray tracing handle the irregular workload through task systems. The strengths of static and dynamic scheduling strategies are complementary to each other. Static strategies do not incur in synchronization overhead while dynamic strategies generally provide computational times closer to the optimal scheduling. Hybrid strategies combining good static initialization and dynamic task assignation have been shown to be a better alternative than pure static and dynamic strategies [Heirich and Arvo 1998]. We experiment with a novel strategy for load balancing on multi-GPU systems. We obtain a quick estimate of the cost of traversing batches of rays over bounding volume hierarchies. The estimated costs are used to achieve a tighter assignation of tasks to processing units. Results suggest that cost-based initialization can enhance common balancing strategies and reduce rendering times.

2 Our Approach

We estimate the cost of processing each task by performing a *reduced traversal* of the rays over bounding volume hierarchies (BVHs). The reduced traversal of a ray does not return ray hits, but the number of primitive intersection tests (i.e. boxes and triangles) that need to be performed in order to compute the hit, and can be performed faster than a full trace operation. This works as current high performance implementations of ray tracing on GPUs (e.g., [Aila and Laine 2009]) rely on texture memory for caching BVH nodes during ray traversals. Triangle primitives on the other hand are not cached, for these are not requested nearly as often as the subset of nodes that were recently traversed. Performing the reduced traversal for every ray within a task results in excessive overhead, however. We reduce the estimation overhead by sampling the tasks. Coherent rays are sampled over a Z-curve, and diffuse rays are randomly sampled. Enhanced initialization of the tasks system can then be achieved by enforcing a scheduling in which the remaining task with the largest cost is assigned to the GPU with the least amount of work.

3 Results

The time taken by the reduced traversal was measured to be 42.5% and 43.5% of the full trace operation on a NVidia Tesla C1060 GPU, for coherent and diffuse rays, respectively. We empirically found that sampling 12.5% of rays results in overheads of 3.2% and 4.1% of the full trace operation, and estimation errors of 3.2% and 4.1%, for coherent and diffuse rays, respectively. Figure 1 shows the effect on the overall tracing times of using the estimated costs for initializing a static distributed queue, and a centralized queue. Experiments were performed on an AMAX machine with a Xeon E5520 processor, 8GB of RAM memory, and 3 NVIDIA Tesla C1060 GPUs. Our implementation of the centralized multi-GPU queue involves one kernel launch for each task in order to synchronize the GPUs. The best tracing times were measured for the static distribution with cost initialization. Intuitively one would expect the centralized queue to outperform the static strategy, but current general purpose GPU programing frameworks (e.g. CUDA and OpenCL) do not provide direct ways to avoid the kernel interruption required for synchronization between units.

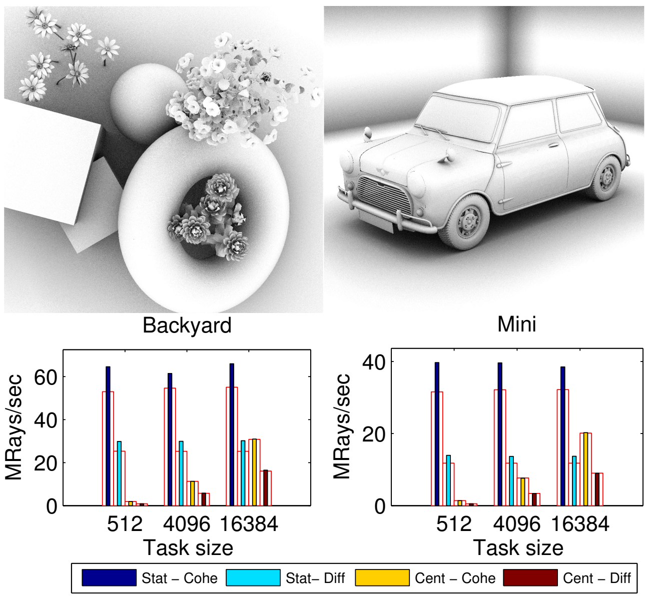

Figure 1: *Tracing times for cost-initializated (colored bars), and regular versions (background bars) of a static distributed queues task system and a centralized queue task system for various task sizes and two scenes (Backyard, 213802 tris; Mini, 234443 tris).*

4 Future Work

Global memory accesses are the bottleneck in current high performance GPU tracers. The reduced traversal for cost estimation works by taking advantage of this shortcoming. A more general approach for cost estimation needs to be investigated for use within parallel platforms other than current multi-GPU systems. Additionally, efficient synchronization mechanisms for the centralized queue in multi-GPU systems should be investigated for a complete evaluation.

References

AILA, T., AND LAINE, S. 2009. Understanding the efficiency of ray traversal on GPUs. In *Proc. High-Performance Graphics 2009*, 145–149.

HEIRICH, A., AND ARVO, J. 1998. A competitive analysis of load balancing strategies for parallel ray tracing. *The Journal of Supercomputing 12*, 1-2, 57–68.

*e-mail:mario.rincon.nigro@gmail.com

†e-mail:zdeng@cs.uh.edu.com