Understanding the Efficiency of Ray Traversal on GPUs – Kepler and Fermi Addendum

Timo Aila, Samuli Laine, Tero Karrass
NVIDIA Research

Abstract

- This poster is an addendum to the HPG2009 paper "Understanding the Efficiency of Ray Traversal on GPUs" [AL09].
- We cover the performance optimization of traversal and intersection kernels for Fermi and Kepler architectures [NV110, NV112a].
- We discuss two esoteric instructions, present in both Fermi and Kepler, and show how they can be used for faster acceleration structure traversal.
- We cover the performance optimization of traversal and intersection kernels for Tesla, Fermi, and Kepler.

Implications of memory architecture

- **Fermi**
  - Has L1 and L2 caches
  - L1 services only one cache line per clock
  - Fetch instruction replayed until all threads of a warp serviced
  - Divergent accesses bottlenecked by L1 → SM
  - Even with high hit rate and abundant memory bandwidth, texture units are not fast enough to handle all traffic
  - Texture units are not fast enough to handle all traffic
  - L1 is useful only for coherent, low-priority accesses
  - Traverse work that needs coherence
  - Split workload between L1 and texture

- **Kepler**
  - Significant upgrade to FLOPS and texture units
  - Same L1 and L2
  - L1 is useful only for coherent, low-priority accesses
  - Rays, traversal stacks
  - Fetches through texture cache are divergence-tolerant
  - Latency is high, dependent fetches should be avoided
  - Speculative traversal is useful again
  - It is beneficial to replace terminated rays when SIMD utilization drops below a threshold
  - As speculated in [AL09]
  - We use a threshold of 60%

Performance and scalability

- Relative average performance (MRays/sec) of primary, ambient occlusion, and diffuse rays in our four test scenes on GTX285, GTX480, and GTX680, plotted against the relative memory bandwidth and peak FLOPS.
- Ray tracing performance continues to follow peak flops very closely, while memory bandwidth has increased at a much slower rate.
- Interestingly, diffuse rays seem to scale even better than primary rays, but that is an artifact caused by our Kepler-specific optimizations that favor incoherent rays (See Table 2).

Table 1: Performance measurements in MRays/sec for Tesla (GTX285), Fermi (GTX480) and Kepler (GT680) using the setup from [AL09]. The San Miguel scene is from PRTF. The scaling between generations is visualized above.

<table>
<thead>
<tr>
<th>Ray type</th>
<th>Tesla AL09</th>
<th>Fermi</th>
<th>Kepler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary</td>
<td>142.2</td>
<td>272.1</td>
<td>432.6</td>
</tr>
<tr>
<td>AD</td>
<td>134.5</td>
<td>284.1</td>
<td>515.2</td>
</tr>
<tr>
<td>Diffuse</td>
<td>60.9</td>
<td>126.1</td>
<td>254.8</td>
</tr>
</tbody>
</table>

Table 2: Summary of differences in traversal and intersection kernels for Tesla, Fermi, and Kepler.

<table>
<thead>
<tr>
<th></th>
<th>Tesla</th>
<th>Fermi</th>
<th>Kepler</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data fetches</td>
<td>Fetch nodes through texture, triangles from (uncached) global memory.</td>
<td>Fetch nodes through L1, triangles via texture. L1 is a bottleneck with incoherent rays but texture is not fast enough to fetch everything.</td>
<td>Fetch all node and triangle data via texture, and avoid dependent fetches whenever possible. L1 is slow and beneficial only for traversal stacks and ray fetches.</td>
</tr>
<tr>
<td>Persistent threads</td>
<td>Doubles performance.</td>
<td>Not beneficial due to a better hardware work distributor.</td>
<td>Can replace all terminated rays once fewer than 60% of the warp’s lanes have work. Favors incoherent rays, +10%.</td>
</tr>
<tr>
<td>Speculative traversal</td>
<td>Nearly always useful.</td>
<td>A small win for incoherent rays and large scenes.</td>
<td>One or two-slot postpone buffer is beneficial for incoherent rays.</td>
</tr>
</tbody>
</table>

| VMIN, VMAX    | Not available. | A trivial +10%. | Less useful than on Fermi because of lower throughput, +5%. |

Summary

- Our recommendations are summarized in Table 2
- Extended version is available as technical report NVR-2012-02 at http://research.nvidia.com
- Optimized kernels are available at http://code.google.com/p/understanding-the-efficiency-of-ray-traversal-on-gpus/

References