Compressed-LEAF bounding volume hierarchies

Carsten Benthin, Ingo Wald, Sven Woop, Attila Áfra
GOAL

“Brain-dead simple approach to make compressed wide BVHs more efficient”
Bounding Volume hierarchies (BVHs)

- BVH node = axis-aligned bounding boxes (AABB) + pointer
- 32-bit single precision floating point
- N-wide BVHs (N = 2,4,8)
Bounding Volume hierarchies (BVHs)

- Axis-aligned bounding boxes (AABB)
- 32-bit single precision floating point
- N-wide BVHs (N = 2,4,8)

**Diagram:**
- Inner node, points to other nodes
- Leaf node, points to leaf data
- Multi-node
MULTI-NODE LAYOUT

- $6 \times 8 \times \text{sizeof(float)} + 8 \times \text{sizeof(void*)} = 256$ bytes

<table>
<thead>
<tr>
<th></th>
<th>$X_{\text{min}0}$</th>
<th>$X_{\text{min}1}$</th>
<th>$X_{\text{min}2}$</th>
<th>$X_{\text{min}3}$</th>
<th>$X_{\text{min}4}$</th>
<th>$X_{\text{min}5}$</th>
<th>$X_{\text{min}6}$</th>
<th>$X_{\text{min}7}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{\text{max}0}$</td>
<td>$X_{\text{max}1}$</td>
<td>$X_{\text{max}2}$</td>
<td>$X_{\text{max}3}$</td>
<td>$X_{\text{max}4}$</td>
<td>$X_{\text{max}5}$</td>
<td>$X_{\text{max}6}$</td>
<td>$X_{\text{max}7}$</td>
<td></td>
</tr>
<tr>
<td>$Y_{\text{min}0}$</td>
<td>$Y_{\text{min}1}$</td>
<td>$Y_{\text{min}2}$</td>
<td>$Y_{\text{min}3}$</td>
<td>$Y_{\text{min}4}$</td>
<td>$Y_{\text{min}5}$</td>
<td>$Y_{\text{min}6}$</td>
<td>$Y_{\text{min}7}$</td>
<td></td>
</tr>
<tr>
<td>$Y_{\text{max}0}$</td>
<td>$Y_{\text{max}1}$</td>
<td>$Y_{\text{max}2}$</td>
<td>$Y_{\text{max}3}$</td>
<td>$Y_{\text{max}4}$</td>
<td>$Y_{\text{max}5}$</td>
<td>$Y_{\text{max}6}$</td>
<td>$Y_{\text{max}7}$</td>
<td></td>
</tr>
<tr>
<td>$Z_{\text{min}0}$</td>
<td>$Z_{\text{min}1}$</td>
<td>$Z_{\text{min}2}$</td>
<td>$Z_{\text{min}3}$</td>
<td>$Z_{\text{min}4}$</td>
<td>$Z_{\text{min}5}$</td>
<td>$Z_{\text{min}6}$</td>
<td>$Z_{\text{min}7}$</td>
<td></td>
</tr>
<tr>
<td>$Z_{\text{max}0}$</td>
<td>$Z_{\text{max}1}$</td>
<td>$Z_{\text{max}2}$</td>
<td>$Z_{\text{max}3}$</td>
<td>$Z_{\text{max}4}$</td>
<td>$Z_{\text{max}5}$</td>
<td>$Z_{\text{max}6}$</td>
<td>$Z_{\text{max}7}$</td>
<td></td>
</tr>
<tr>
<td>$\text{ptr}_0$</td>
<td>$\text{ptr}_1$</td>
<td>$\text{ptr}_2$</td>
<td>$\text{ptr}_3$</td>
<td>$\text{ptr}_4$</td>
<td>$\text{ptr}_5$</td>
<td>$\text{ptr}_6$</td>
<td>$\text{ptr}_7$</td>
<td></td>
</tr>
</tbody>
</table>

float

void*
MULTI-NODE LAYOUT

- $6 \times 8 \times \text{sizeof(float)} + 8 \times \text{sizeof(void*)} = 256$ bytes

<table>
<thead>
<tr>
<th>X_{min0}</th>
<th>X_{min1}</th>
<th>X_{min2}</th>
<th>X_{min3}</th>
<th>X_{min4}</th>
<th>X_{min5}</th>
<th>X_{min6}</th>
<th>X_{min7}</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_{max0}</td>
<td>X_{max1}</td>
<td>X_{max2}</td>
<td>X_{max3}</td>
<td>X_{max4}</td>
<td>X_{max5}</td>
<td>X_{max6}</td>
<td>X_{max7}</td>
</tr>
<tr>
<td>Y_{min0}</td>
<td>Y_{min1}</td>
<td>Y_{min2}</td>
<td>Y_{min3}</td>
<td>Y_{min4}</td>
<td>Y_{min5}</td>
<td>Y_{min6}</td>
<td>Y_{min7}</td>
</tr>
<tr>
<td>Y_{max0}</td>
<td>Y_{max1}</td>
<td>Y_{max2}</td>
<td>Y_{max3}</td>
<td>Y_{max4}</td>
<td>Y_{max5}</td>
<td>Y_{max6}</td>
<td>Y_{max7}</td>
</tr>
<tr>
<td>Z_{min0}</td>
<td>Z_{min1}</td>
<td>Z_{min2}</td>
<td>Z_{min3}</td>
<td>Z_{min4}</td>
<td>Z_{min5}</td>
<td>Z_{min6}</td>
<td>Z_{min7}</td>
</tr>
<tr>
<td>Z_{max0}</td>
<td>Z_{max1}</td>
<td>Z_{max2}</td>
<td>Z_{max3}</td>
<td>Z_{max4}</td>
<td>Z_{max5}</td>
<td>Z_{max6}</td>
<td>Z_{max7}</td>
</tr>
<tr>
<td>ptr_0</td>
<td>ptr_1</td>
<td>ptr_2</td>
<td>ptr_3</td>
<td>ptr_4</td>
<td>ptr_5</td>
<td>ptr_6</td>
<td>ptr_7</td>
</tr>
</tbody>
</table>

“BVH Compression”
Mahovski et al.
Segovia et al.
Keely et al.
Wald et al.
Ylitie et al.
...

float

void*
COMPRESSED MULTI-NODES

- Express children AABBs relative to the parent AABB
COMPRESSED MULTI-NODES

• Express children AABBs relative to the parent AABB

• Use lower precision (4/8/16-bit integer) for encoding
COMPRESSED MULTI-NODES

• Express children AABBs relative to the parent AABB
• Use lower precision (4/8/16-bit integer) for encoding
COMPRESSED MULTI-NODES

- Decompression
  - $\text{start}_X = \text{parent.min}_X$
  - $\text{scale}_X = (\text{parent.max}_X - \text{parent.min}_X) / 255.f$
  - $D_X = \text{start}_X + (\text{float})i_X \times \text{scale}_X$

- Implementation
  - Maps well to 8-wide SIMD
  - Decompress all 8 children AABBs in parallel
  - Compressed multi-node BVH = Quantized BVH (QBVH)
COMPRESSED MULTI-NODE LAYOUT

• Uncompressed: 6 * 8 * sizeof(float) + 8 * sizeof(void*) = 256 bytes
• Compressed: 6 * sizeof(float) + 6 * 8 * sizeof(char) + 8 * sizeof(void*) = 136 bytes
ISSUES WITH QBVHs

- Coarser bounds
  - +2-4% traversal steps
  - +4-10% intersection steps
- Costly decompression
  - 6 times $D_k = \text{start}_k + (\text{float})i_k \times \text{scale}_k$ (using SIMD)
  - ~24 SIMD instructions overhead (more than the actual ray-slabs test)

→ ~10-20% total performance impact (diffuse path tracing) 😞
MEH, I WANT MY PERFORMANCE BACK...

Observations

• For a binary BVH half the nodes are leaf nodes, >80% for 8-wide BVH
• Ray traverses multiple inner multi-nodes per leaf
• Most of the overhead comes from decompressing inner multi-nodes

General Idea

• Only compress “leaf” multi-nodes 😊
There’s A LITTLE PROBLEM...

- What’s a “leaf” multi-node in a 8-wide BVH?

- Only compress “all-leaf” multi-nodes
COMPRESSED-LEAF BOUNDING VOLUME HIERARCHIES (CLBVH)

- CLBVH = regular BVH with compressed “all-leaf” multi-nodes
- Tweak top-down BVH builder to primarily generate “all-leaf” multi-nodes
- Mark them as “special leaves” (need 1 additional bit)
- Keep top-down BVH traversal unchanged
- Two types of leaves
  - Regular leaves
  - “All-leaf” multi-nodes
FURTHER COMPRESSION...

- Store referenced primitive data directly behind “all-leaf” multi-nodes
- Get rid of explicit pointers to primitive data
- Pointer is given implicitly by children index
FURTHER COMPRESSION…

- Store referenced primitive data directly behind “all-leaf” multi-nodes
- Get rid of explicit pointers to primitive data
- Pointer is given implicitly by children index

Reduces size to 72 bytes only
EVEN FURTHER COMPRESSION... CLBVH (compact)

- Extract shared features in geometry data in “all-leaf” multi-node
- Object IDs, vertices, vertex indices, shader IDs, etc.
- Re-index data with more compact “indices”
- Similar to small treelets
- Reduces size of leaf data by 15-25% (lossless compression)
Results

- Evaluation done in special version of Embree 3.x
- Dual-socket Intel® Xeon® Platinum, 2x28 cores, 96 GB memory
- Four scenes (10M-350M triangles), diffuse path tracing (up to 8 bounces)
- “Best speed” vs. “least memory” mode
Results (best SPEED MODE)

From 10-20% performance reduction down to 2-4%
• BVH memory reduction of 42-46%
**ResultS (BEST SPEED MODE)**

- Only 8-10% total memory reduction
ResultS (LEAST MEMORY MODE)

• From 7-15% performance reduction down to 0-3%
Results (LEAST MEMORY MODE)

- Similar BVH memory reduction of 42-46%
Results (LEAST MEMORY MODE)

- CLBVH reduces total memory by 16-24%, CLBVH (compact) by 26-37%
Summary & OUTLOOK

• Compared to QBVHs, our CLBVHs:
  • Are able to recover most of the performance slow-down
  • Provide similar BVH size reductions
  • Easy to integrate into existing ray tracing engines (impacts only leaf code)
  • Even allows for (lossless) leaf data compression

• Embree already uses a variant of CLBVHs for hair/curve primitives
• GPU, custom HW
Questions?
Legal

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document. Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade. This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks

Benchmark results were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown". Implementation of these updates may make these results inapplicable to your device or system.

Intel, Xeon and the Intel logo are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others

© Intel Corporation.
Acronym List

• Application Programming Interface (API)
• Bounding Volume Hierarchy (BVH)
• Independent Software Vendor (ISV)
• Instruction Set Architecture (ISA)
• Intel® Advanced Vector Extensions (Intel® AVX)
• Intel® Advanced Vector Extensions 2 (Intel® AVX2)
• Intel® Advanced Vector Extensions 512 (Intel® AVX-512)
• Intel® SPMD Program Compiler (Intel® SPC)
• Intel® Streaming SIMD Extensions (Intel® SSE)
• Intel® Threading Building Blocks (Intel® TBB)
• Non-Uniform Rational Basis Spline (NURBS)
• Single Instruction, Multiple Data (SIMD)
• Single Program, Multiple Data (SPMD)
• Surface Area Heuristic (SAH)