Compressed-LEAF bounding volume hierarchies

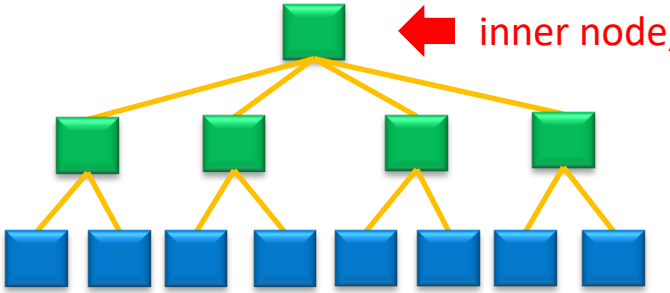
Carsten Benthin, Ingo Wald, Sven Woop, Attila Áfra

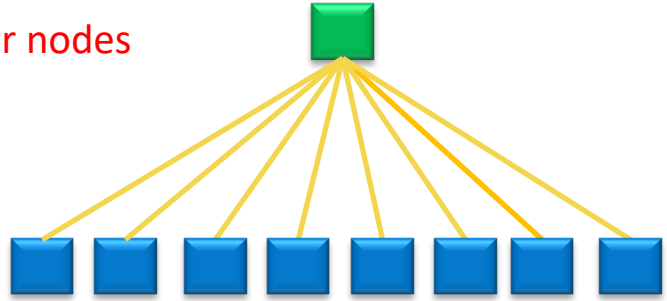
GOAL

“Brain-dead simple approach to make compressed wide BVHs more efficient”

Bounding Volume hierarchies (BVHs)

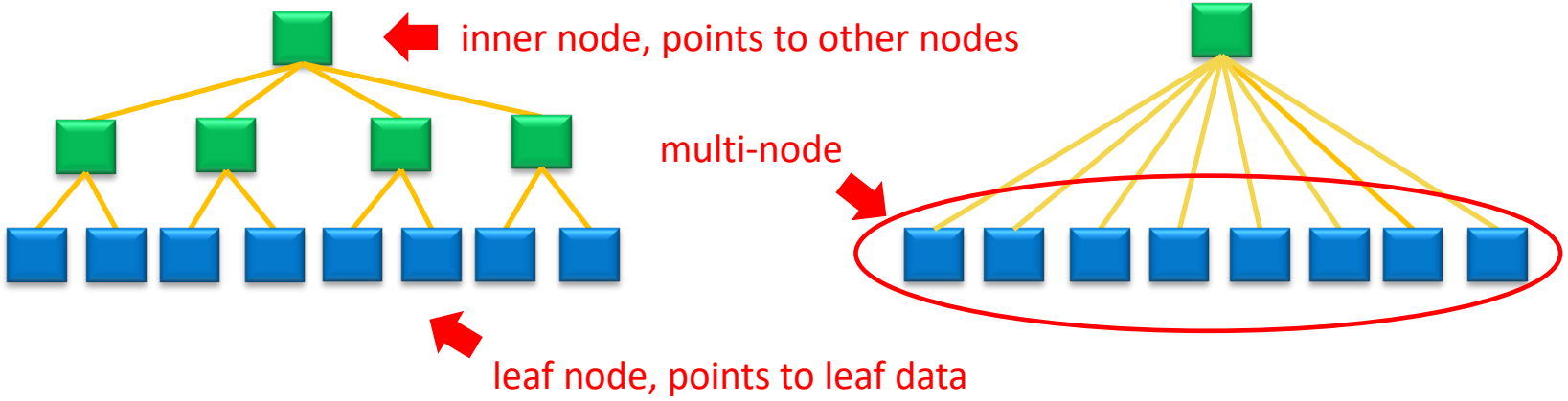
- BVH node = axis-aligned bounding boxes (AABB) + pointer
- 32-bit single precision floating point
- N-wide BVHs (N = 2,4,8)





Bounding Volume hierarchies (BVHs)

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



MULTI-NODE LAYOUT

- 6 * 8 * sizeof(float) + 8 * sizeof(void*) = 256 bytes

X_{min_0}	X_{min_1}	X_{min_2}	X_{min_3}	X_{min_4}	X_{min_5}	X_{min_6}	X_{min_7}
X_{max_0}	X_{max_1}	X_{max_2}	X_{max_3}	X_{max_4}	X_{max_5}	X_{max_6}	X_{max_7}
Y_{min_0}	Y_{min_1}	Y_{min_2}	Y_{min_3}	Y_{min_4}	Y_{min_5}	Y_{min_6}	Y_{min_7}
Y_{max_0}	Y_{max_1}	Y_{max_2}	Y_{max_3}	Y_{max_4}	Y_{max_5}	Y_{max_6}	Y_{max_7}
Z_{min_0}	Z_{min_1}	Z_{min_2}	Z_{min_3}	Z_{min_4}	Z_{min_5}	Z_{min_6}	Z_{min_7}
Z_{max_0}	Z_{max_1}	Z_{max_2}	Z_{max_3}	Z_{max_4}	Z_{max_5}	Z_{max_6}	Z_{max_7}
ptr_0	ptr_1	ptr_2	ptr_3	ptr_4	ptr_5	ptr_6	ptr_7

float

void*

MULTI-NODE LAYOUT

- 6 * 8 * sizeof(float) + 8 * sizeof(void*) = 256 bytes

X_{min_0}	X_{min_1}	X_{min_2}	X_{min_3}	X_{min_4}	X_{min_5}	X_{min_6}	X_{min_7}
X_{max_0}	X_{max_1}	X_{max_2}	X_{max_3}	X_{max_4}	X_{max_5}	X_{max_6}	X_{max_7}
Y_{min_0}	Y_{min_1}	Y_{min_2}	Y_{min_3}	Y_{min_4}	Y_{min_5}	Y_{min_6}	Y_{min_7}
Y_{max_0}	Y_{max_1}	Y_{max_2}	Y_{max_3}	Y_{max_4}	Y_{max_5}	Y_{max_6}	Y_{max_7}
Z_{min_0}	Z_{min_1}	Z_{min_2}	Z_{min_3}	Z_{min_4}	Z_{min_5}	Z_{min_6}	Z_{min_7}
Z_{max_0}	Z_{max_1}	Z_{max_2}	Z_{max_3}	Z_{max_4}	Z_{max_5}	Z_{max_6}	Z_{max_7}
ptr_0	ptr_1	ptr_2	ptr_3	ptr_4	ptr_5	ptr_6	ptr_7

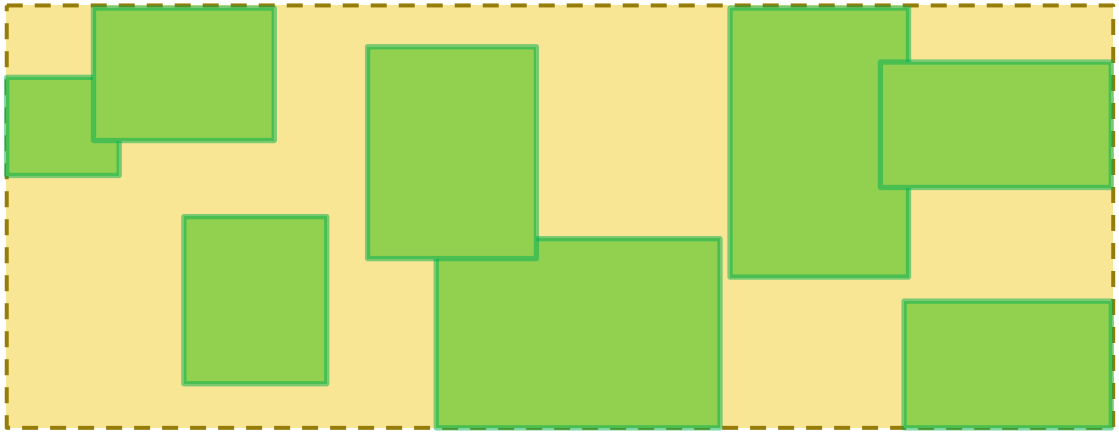
float

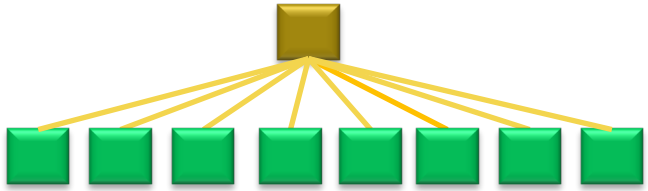
void*

"BVH Compression"
Mahovski et al.
Segovia et al.
Keely et al.
Wald et al.
Ylitie et al.
…

COMPRESSED MULTI-NODES

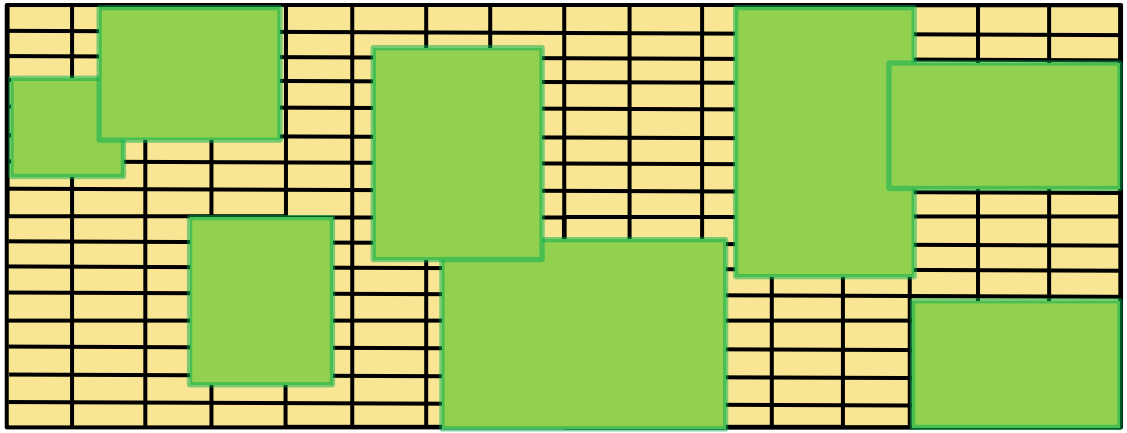
- Express children AABBs relative to the parent AABB

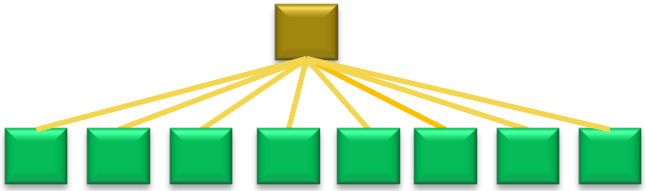




COMPRESSED MULTI-NODES

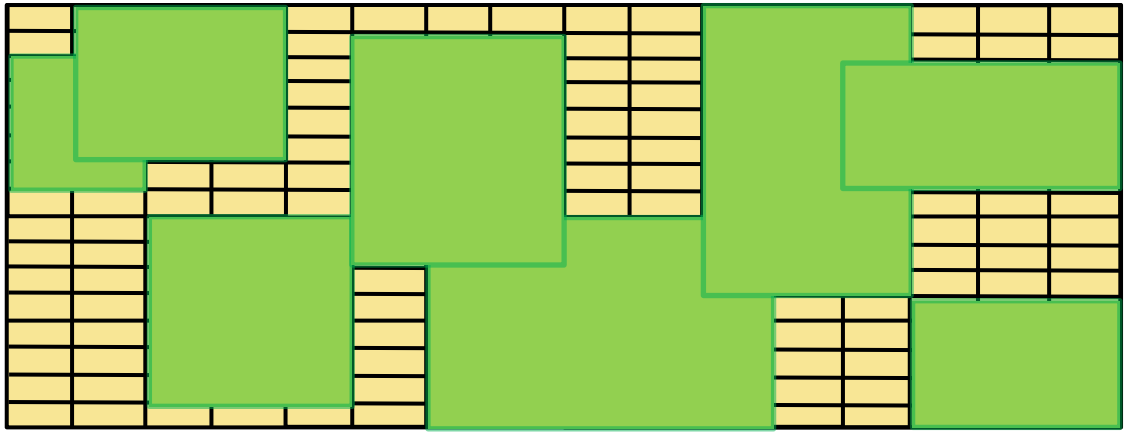
- Express children AABBs relative to the parent AABB
- Use lower precision (4/8/16-bit integer) for encoding

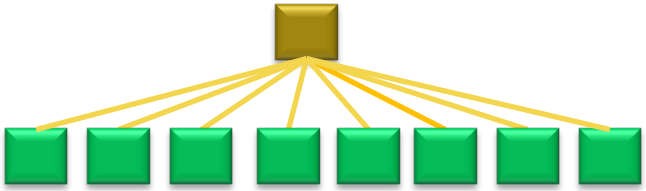




COMPRESSED MULTI-NODES

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



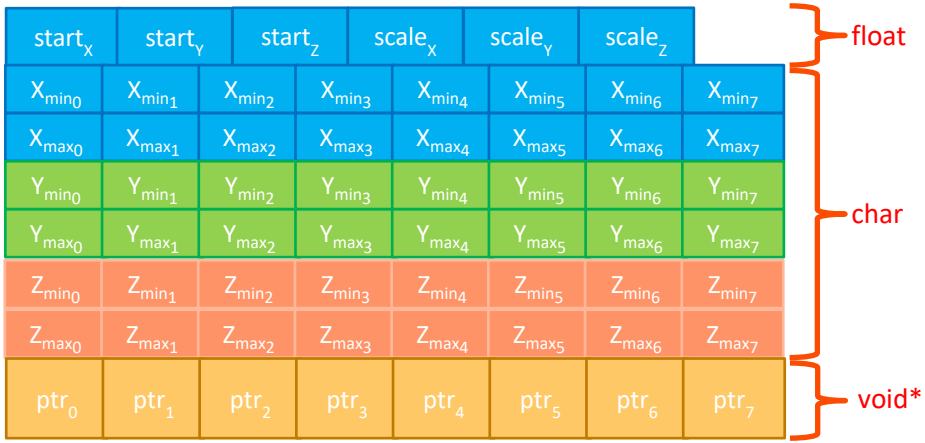


COMPRESSED MULTI-NODES

- Decompression
  - $start_X = parent.min_X$
  - $scale_X = (parent.max_X - parent.min_X) / 255.f$
  - $D_X = start_X + (float)i_X * scale_X$
- 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 $D_k = start_k + (float)i_k * scale_k$ (using SIMD)
  - ~24 SIMD instructions overhead (more than the actual ray-slabs test)

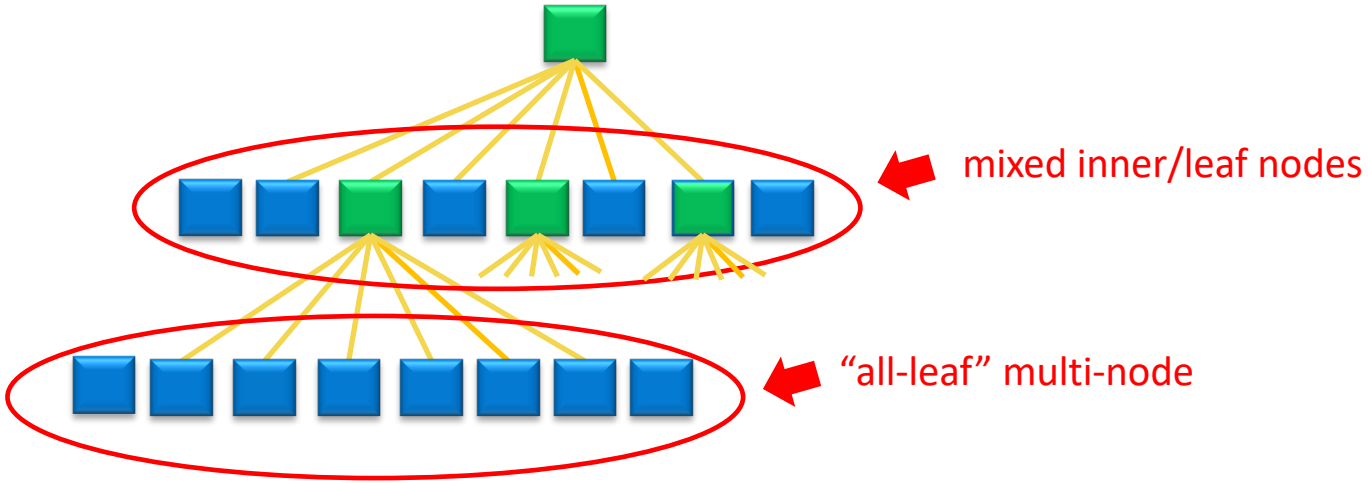
➔ ~10-20% total performance impact (diffuse path tracing) ☹

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…

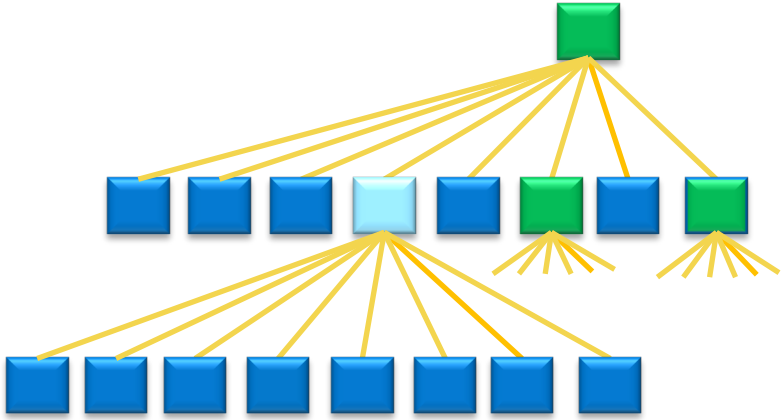
- What's a "leaf" multi-node in a 8-wide BVH?



- Only compress "all-leaf" multi-nodes

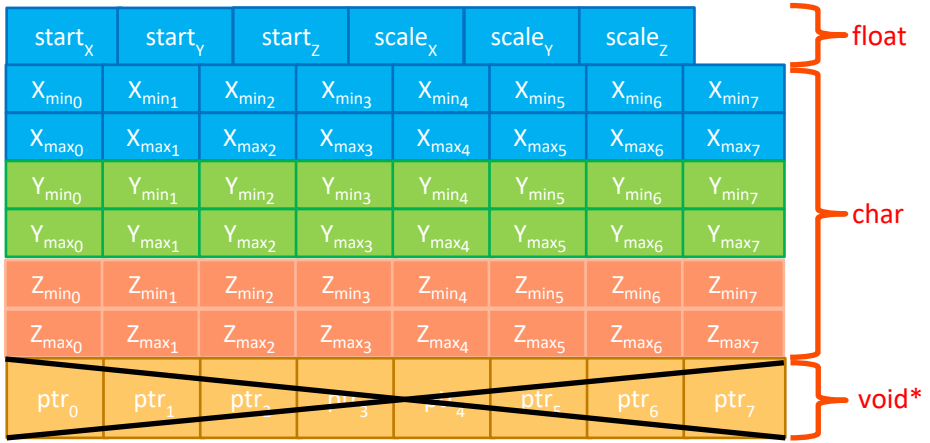
COMPRESSED-LEAF BOUNDING VOLUME HIERARCHIES (CLBVH)

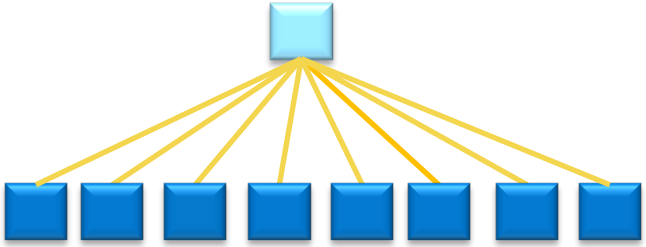
- CLBVH = regular BVH with compressed "all-leaf" multi-nodes
- Tweak top-down BVH builder to primarily generate "all-leaf" multi-nodes
- Mark them as "special leaves" (need 1 additional bit)
- Keep top-down BVH traversal unchanged ☺
- Two types of leaves
  - Regular leaves
  - "All-leaf" multi-nodes



FURTHER COMPRESSION…

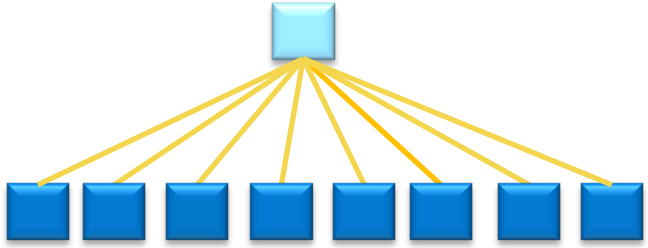
- Store referenced primitive data directly behind “all-leaf” multi-nodes
- Get rid of explicit pointers to primitive data
- Pointer is given implicitly by children index





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



$start_X$	$start_Y$	$start_Z$	$scale_X$	$scale_Y$	$scale_Z$		
X_{min_0}	X_{min_1}	X_{min_2}	X_{min_3}	X_{min_4}	X_{min_5}	X_{min_6}	X_{min_7}
X_{max_0}	X_{max_1}	X_{max_2}	X_{max_3}	X_{max_4}	X_{max_5}	X_{max_6}	X_{max_7}
Y_{min_0}	Y_{min_1}	Y_{min_2}	Y_{min_3}	Y_{min_4}	Y_{min_5}	Y_{min_6}	Y_{min_7}
Y_{max_0}	Y_{max_1}	Y_{max_2}	Y_{max_3}	Y_{max_4}	Y_{max_5}	Y_{max_6}	Y_{max_7}
Z_{min_0}	Z_{min_1}	Z_{min_2}	Z_{min_3}	Z_{min_4}	Z_{min_5}	Z_{min_6}	Z_{min_7}
Z_{max_0}	Z_{max_1}	Z_{max_2}	Z_{max_3}	Z_{max_4}	Z_{max_5}	Z_{max_6}	Z_{max_7}
Leaf-Data$_0$	Leaf-Data$_1$	Leaf-Data$_2$	Leaf-Data$_3$	Leaf-Data$_4$	Leaf-Data$_5$	Leaf-Data$_6$	Leaf-Data$_7$

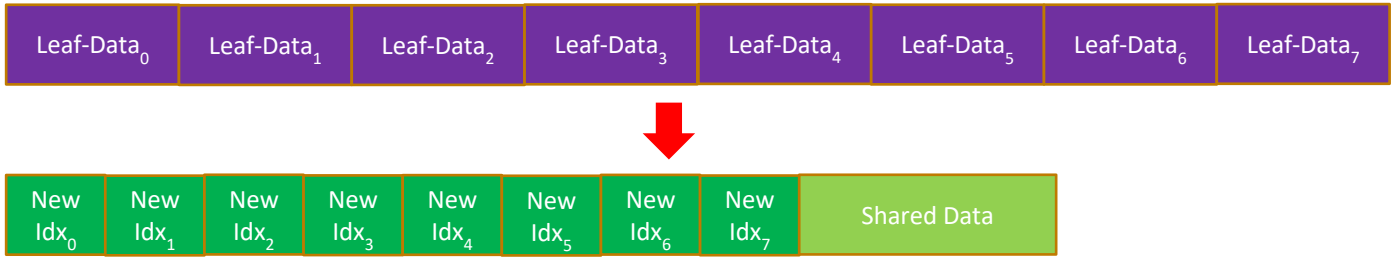
float

char

Reduces size to 72 bytes only

EVEN FURTHER COMPRESSION… CLBVH (compact)

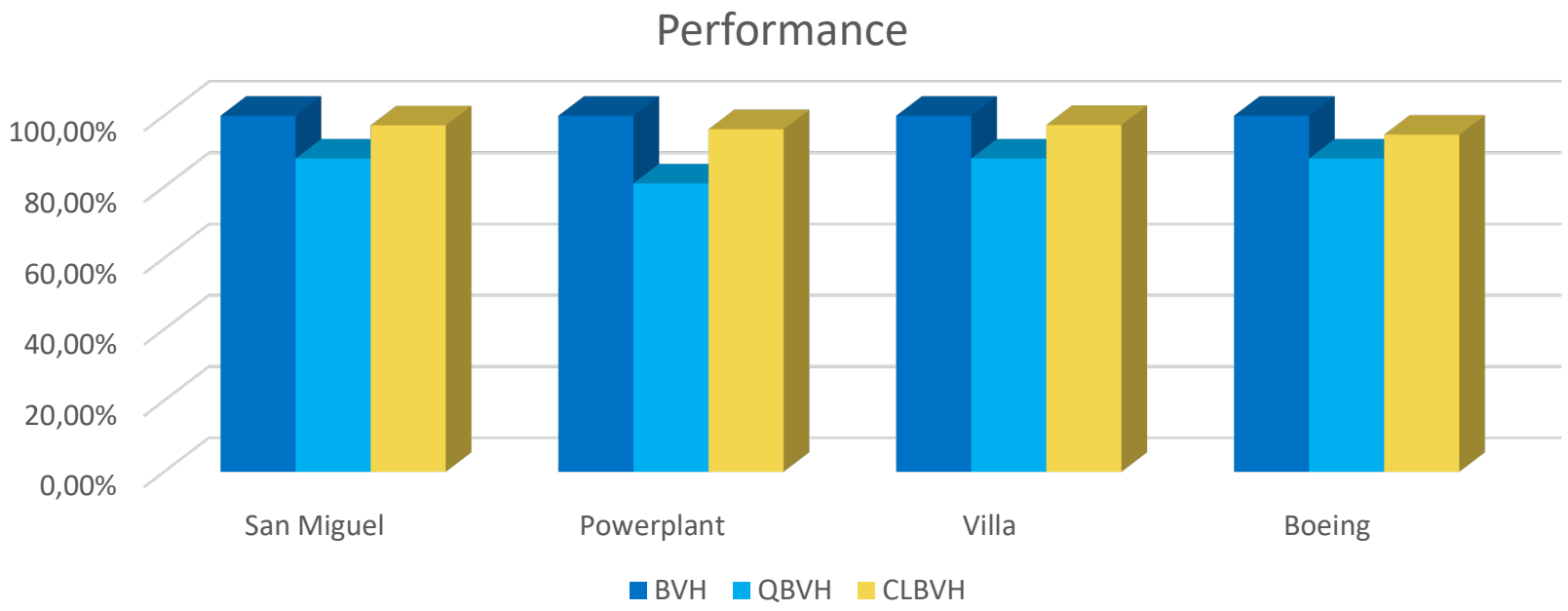
- Extract shared features in geometry data in “all-leaf” multi-node
- Object IDs, vertices, vertex indices, shader IDs, etc.
- Re-index data with more compact “indices”
- Similar to small treelets
- Reduces size of leaf data by 15-25% (lossless compression)



ResultS

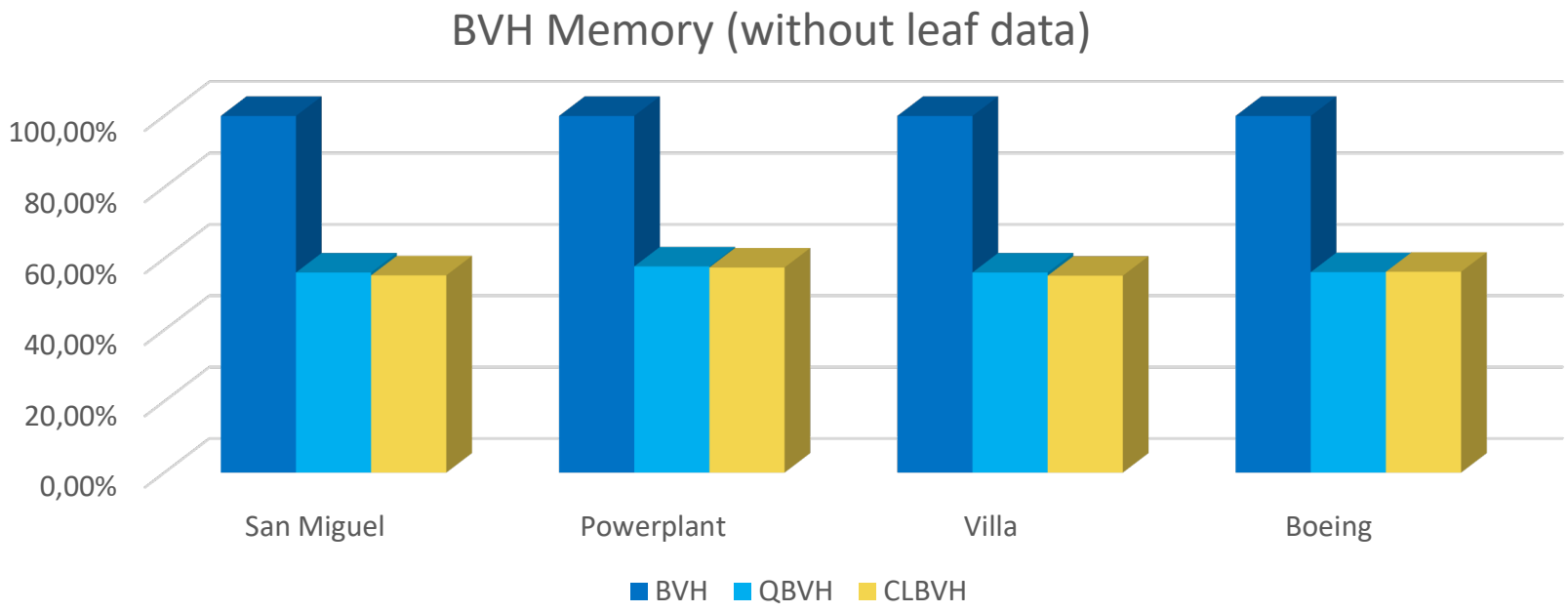
- Evaluation done in special version of Embree 3.x
- Dual-socket Intel® Xeon® Platinum, 2x28 cores, 96 GB memory
- Four scenes (10M-350M triangles), diffuse path tracing (up to 8 bounces)
- “Best speed” vs. “least memory” mode

ResultS (best SPEED MODE)



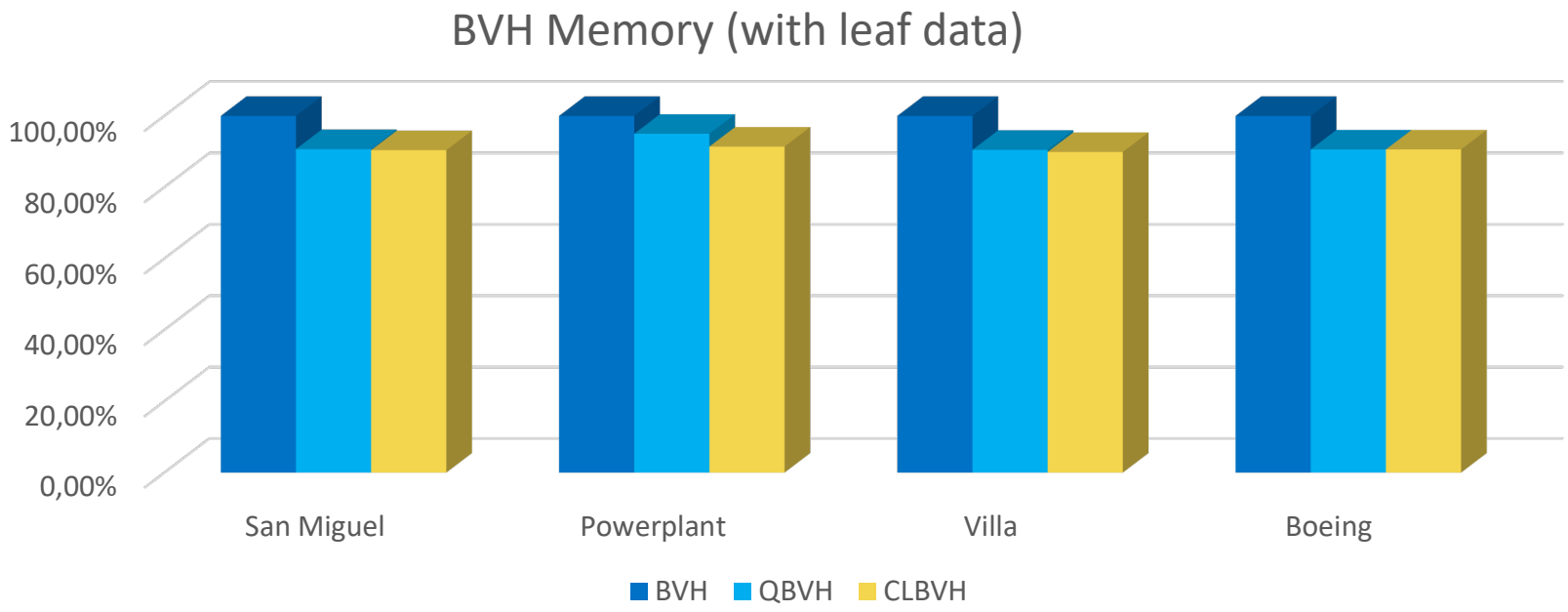
- From 10-20% performance reduction down to 2-4%

ResultS (BEST SPEED MODE)



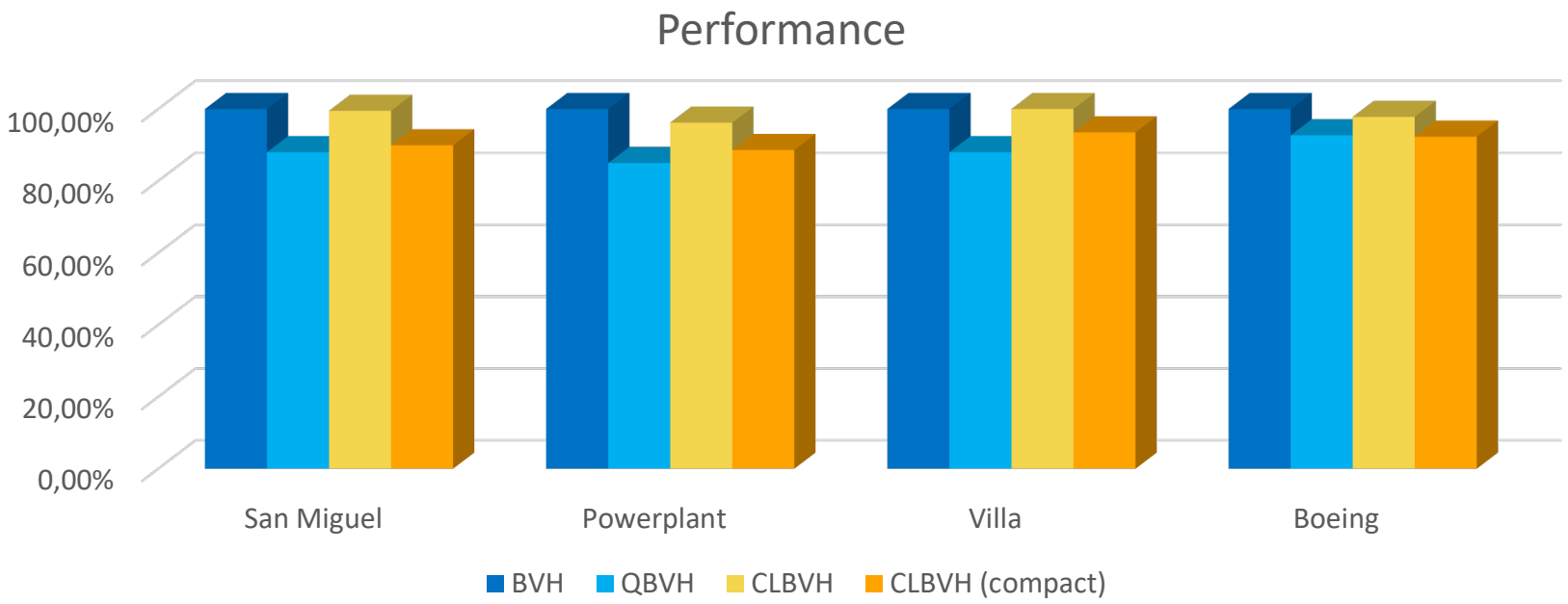
- BVH memory reduction of 42-46%

ResultS (BEST SPEED MODE)



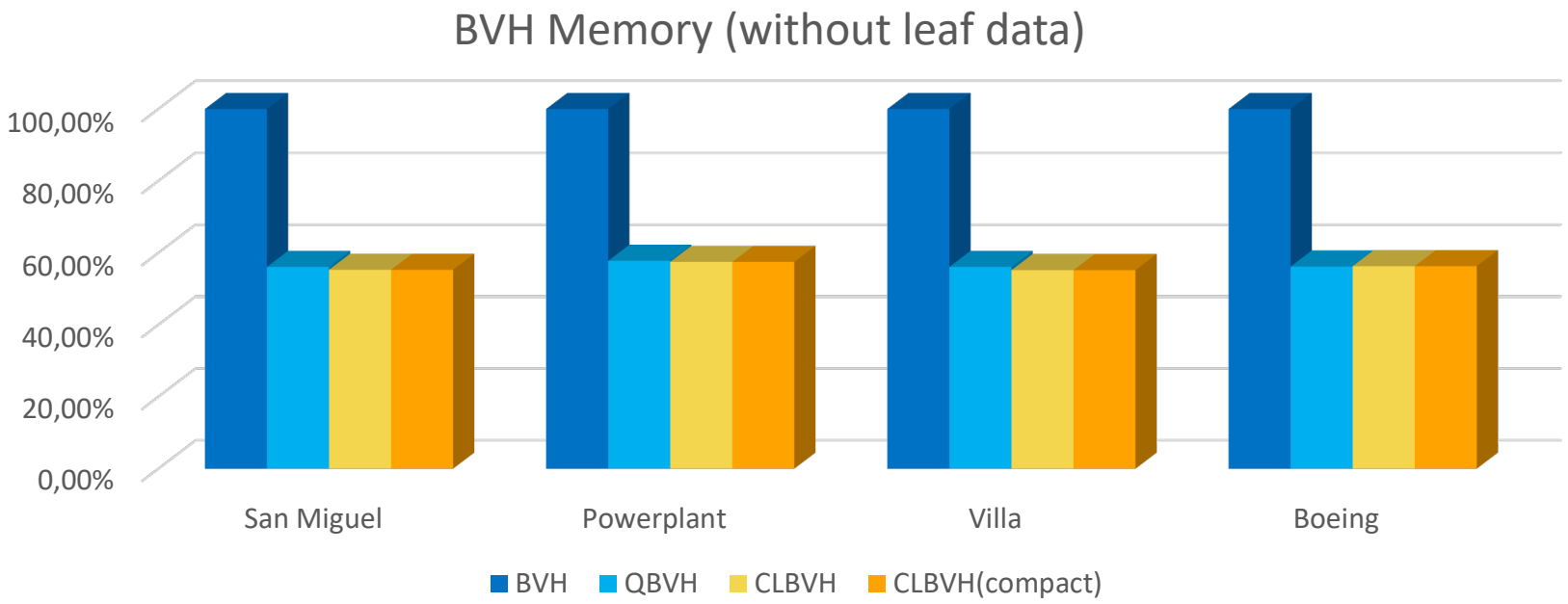
- Only 8-10% total memory reduction

ResultS (LEAST MEMORY MODE)



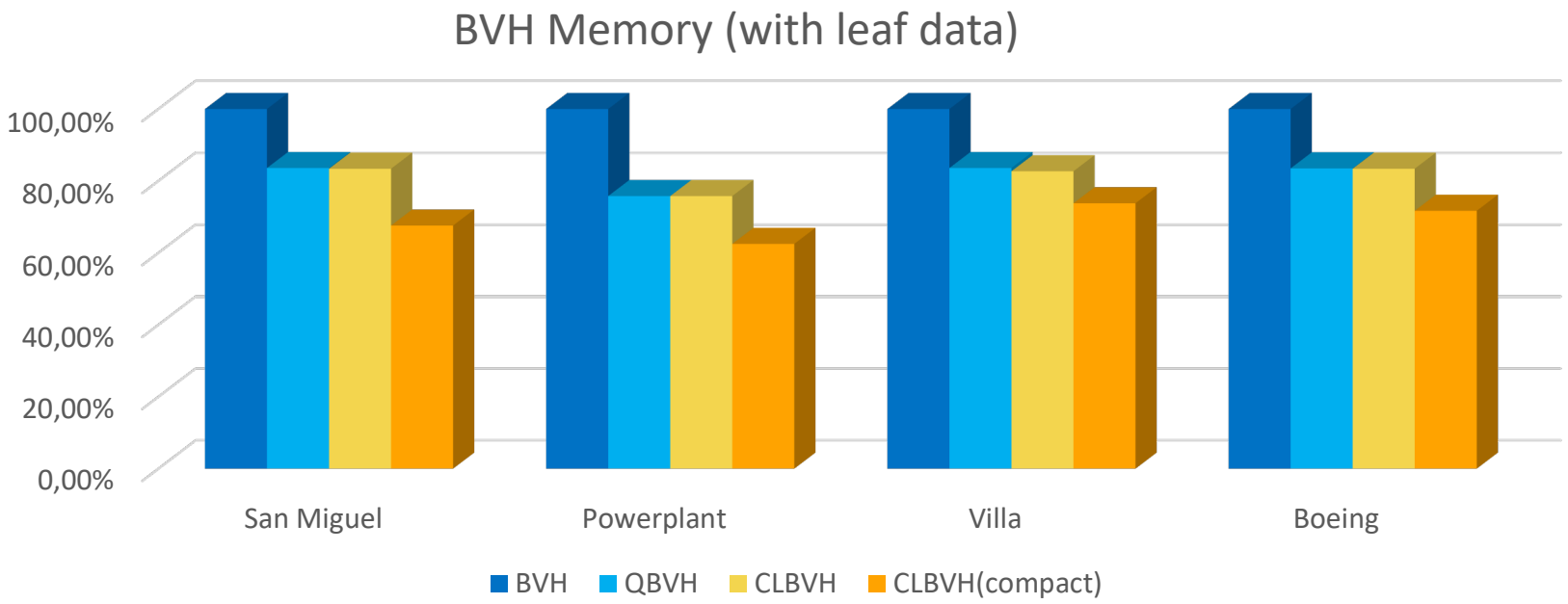
- From 7-15% performance reduction down to 0-3%

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

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)

intel®