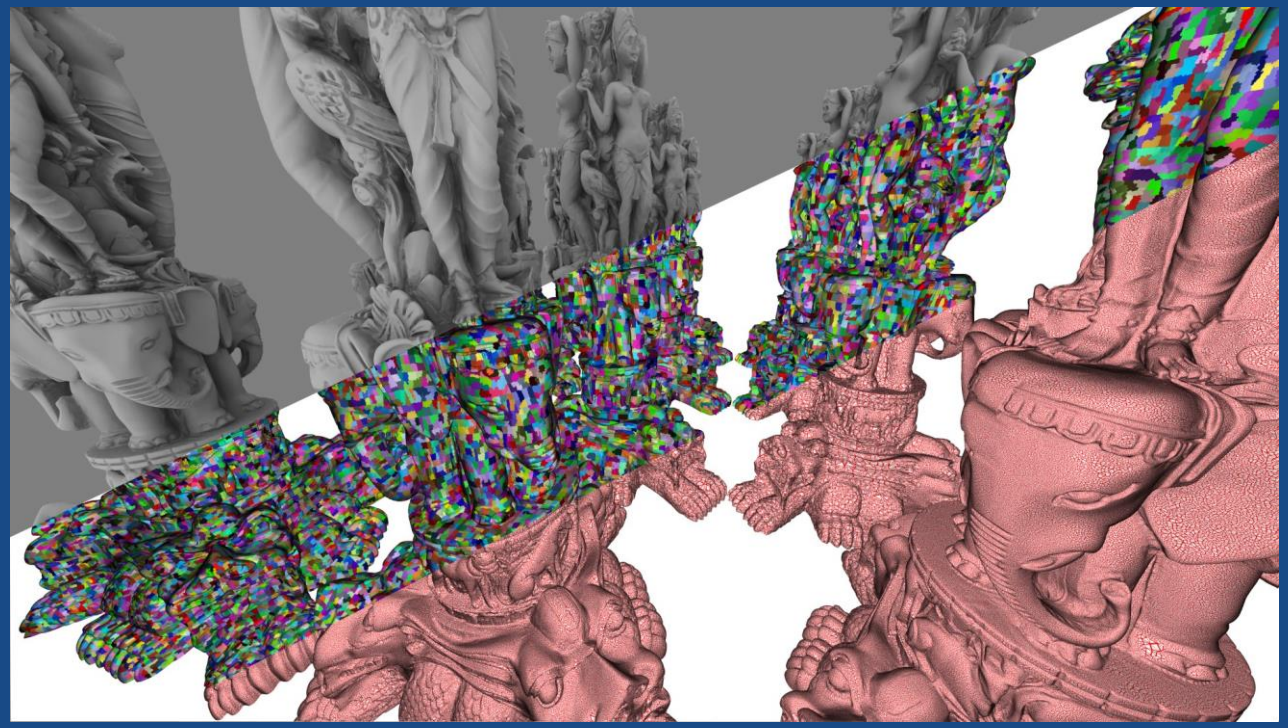


High Performance Graphics 2023

Real-Time Ray Tracing of Micro-Poly Geometry with Hierarchical Level of Detail

Carsten Benthin, Christoph Peters

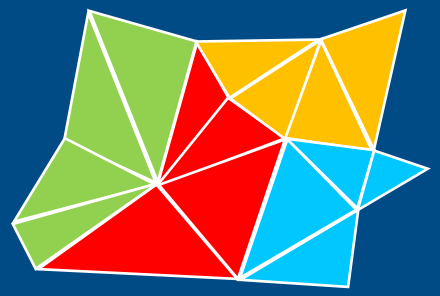
intel®

Nanite: Extreme Geometric Complexity in Real-Time

Nanite A Deep Dive Siggraph 2021

Nanite in a Nutshell

- Groups triangles into geometry clusters (≤ 128 triangles)
  - Lossy compression

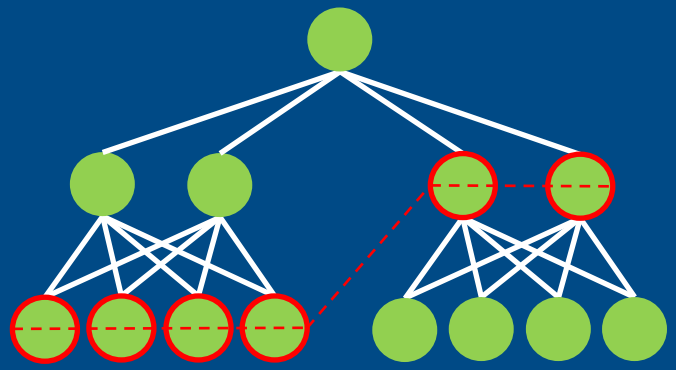


Nanite in a Nutshell

- Groups triangles into geometry clusters (≤ 128 triangles)
  - Lossy compression
- Cluster-based hierarchical LOD
  - Merge and split ➔ DAG (Directed-Acyclic-Graph)

Nanite in a Nutshell

- Rendering clusters
  - Select LOD clusters by (view dependent) cut through DAG
  - Reduce #clusters in subset by frustum + occlusion culling
  - Decompress and (SW-)rasterize triangles in remaining clusters
  - ~20M triangles/frame



Issue

- Cluster-based hierarchical LOD + HW-accelerated ray tracing?
  - TLAS/BLAS API restriction

Issue

- Cluster-based hierarchical LOD + HW-accelerated ray tracing?
  - TLAS/BLAS API restriction
- OPTION 1: No per-frame LOD, put fixed geometry resolution into BLAS
  - GPU memory vs. BLAS memory footprint (less dense per triangle than cluster)
  - Scene updates
  - Geometry aliasing

Issue

- Cluster-based hierarchical LOD + HW-accelerated ray tracing?
  - TLAS/BLAS API restriction

- OPTION 2: Apply LOD per frame, put decompressed triangles into BLAS
  - Mesh topology changes ➔ full BVH rebuild
  - BVH rebuild perf too slow, e.g. 400 MTriangles/s for 20M triangles ➔ 50ms ☹

Issue

- Cluster-based hierarchical LOD + HW-accelerated ray tracing?
  - TLAS/BLAS API restriction

Our approach addresses this

- OPTION 2: Apply LOD per frame, put decompressed triangles into BLAS
  - Mesh topology changes ➔ full BVH rebuild
  - BVH rebuild perf too slow, e.g. 400 MTriangles/s for 20M triangles ➔ 50ms ☹

Our Approach

- Preprocessing phase (CPU)
- Per-frame phase (GPU)

Preprocessing Phase (CPU)

- Initial cluster generation
- Creating a hierarchy over clusters
- Lossy compression of cluster data

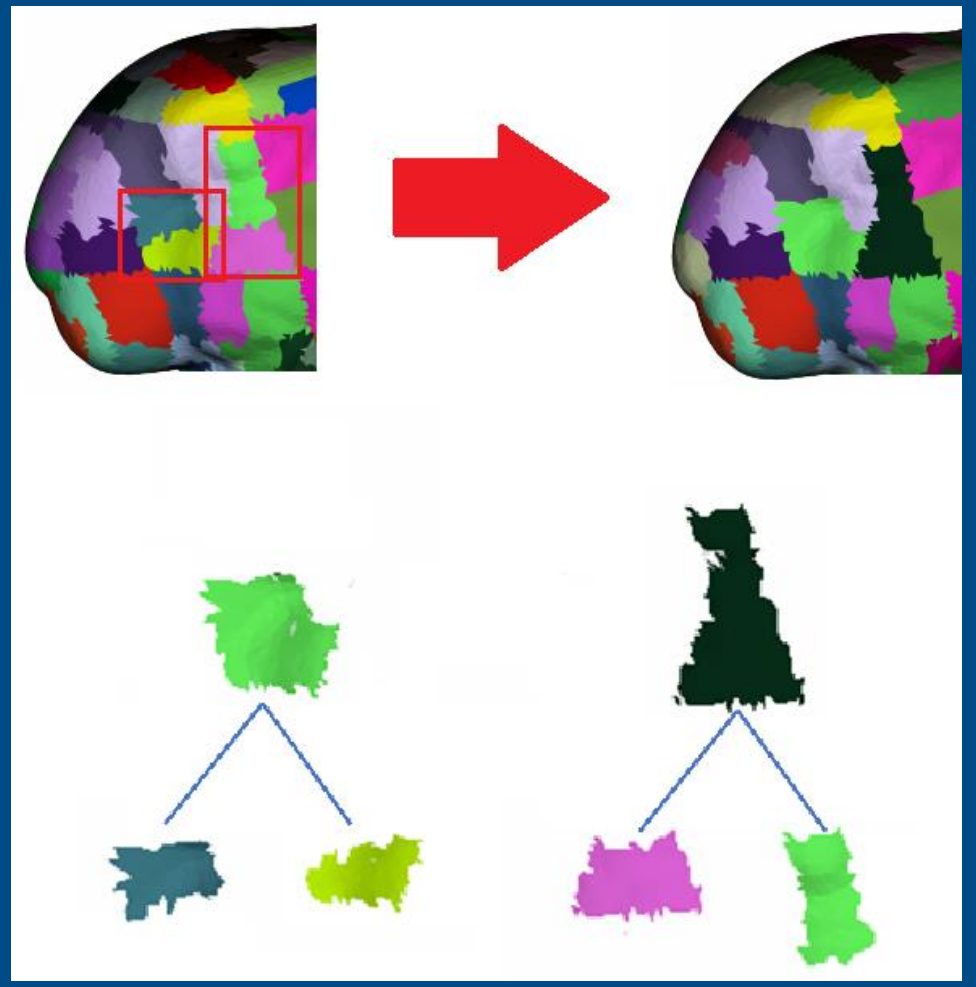
Quite similar to Nanite but more tailored towards ray tracing

Cluster Generation

- Convert triangles into quads (triangle pairs)
- Build BVH over all quads
- Extract clusters by top-down traversal
  - Subtree has ≤ 128 quads (256 triangles)
  - Reduces cluster’s AABB overlap

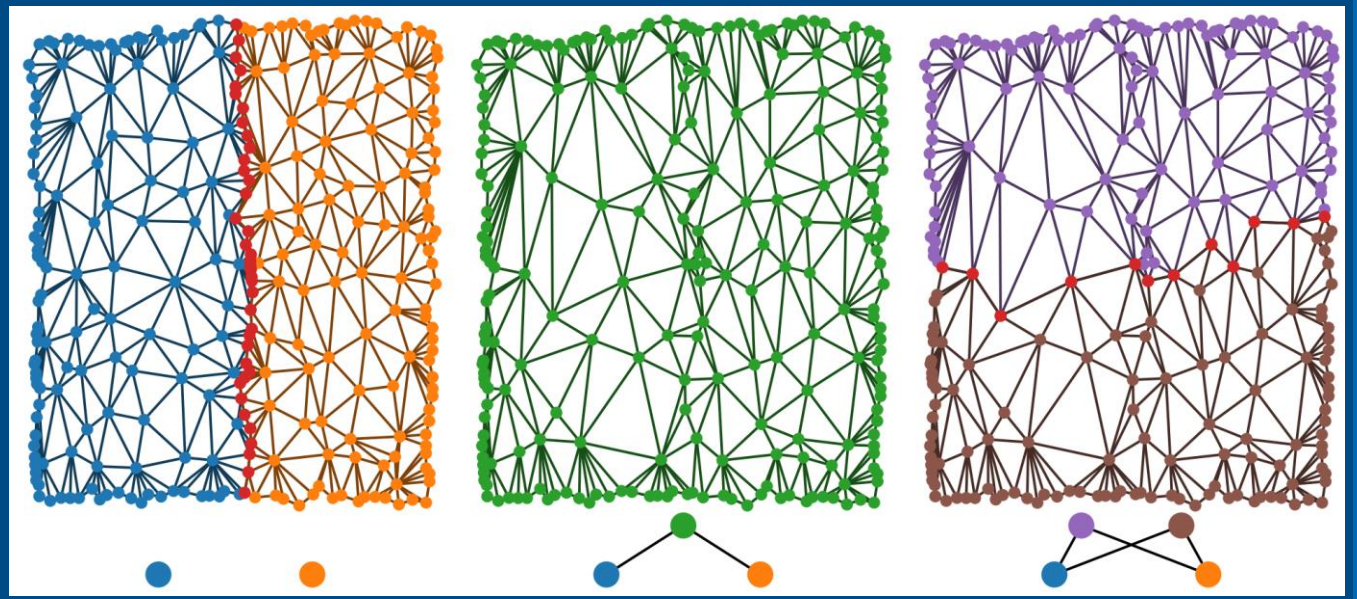
Cluster Hierarchy

- SAH-based cluster merging
  - Iterative bottom-up BVH builder (PLOC)
  - Select pairs with minimal merged AABB area
- Simplify geometry in merged cluster
  - Preserving boundary edges
- Merging would create binary tree but…



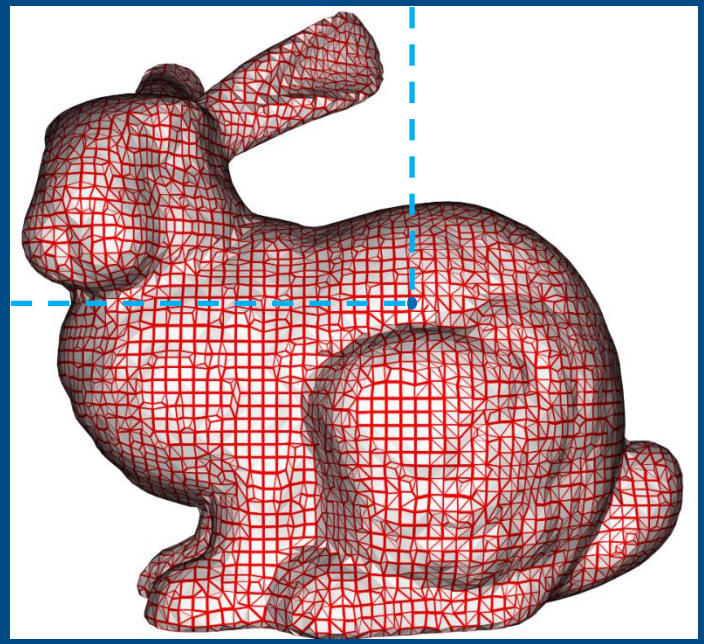
Cluster Merges Can Fail

- Too many boundary edges
  - ➔Simplification fails (#triangles after simplification too high)
  - ➔Split merged cluster
  - ➔Binary DAG
- DAG can have multiple roots



Lossy Compression of Cluster Data

- Cluster contains a lossy compressed quad mesh
  - 16bit vertex quantization with respect to object's bounding box
  - 8 bit vertex indices
- 4-6 bytes per triangle
- 165-222 MTriangles / GB of memory
- Watertight within objects but not across

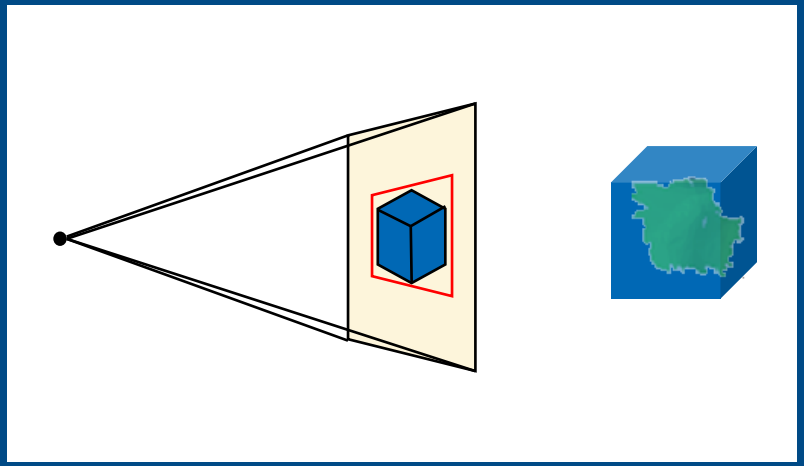


Per-Frame Phase (GPU)

- LOD cluster selection
- Cluster decompression and per cluster BVH build
- Cluster BVH fusing

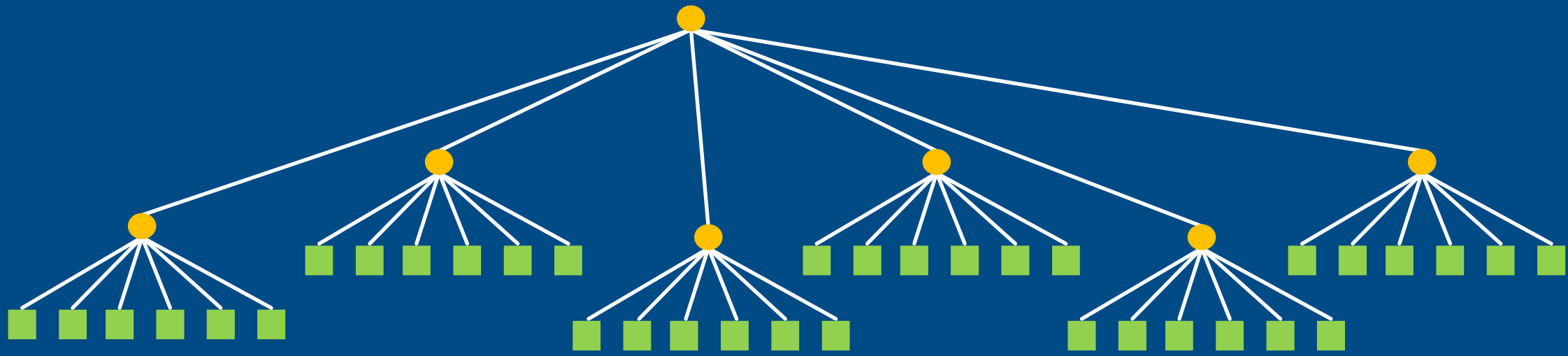
LOD Cluster Selection

- DAG top-down traversal starting from roots
- For each traversal step
  - Project cluster's AABB on image plane
  - Stop and select cluster if diagonal projection's 2D AABB < threshold
- Clusters outside view frustum
  - Cannot cull, need them for secondary rays
  - Assigned coarse LOD level



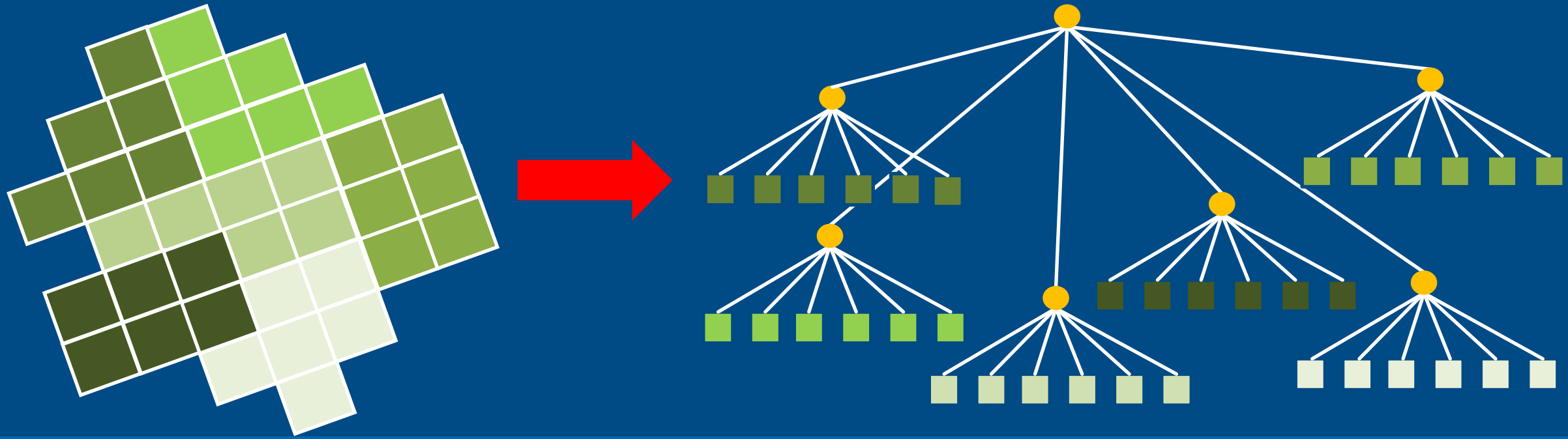
Cluster Decompression and Per Cluster BVH Build

- Target ray tracing HW (Intel Arc Series)
  - 6-wide quantized BVH = QBVH6 (64 bytes), quad/triangle-pair per leaf (64 bytes)
- QBVH6 is ~5x larger than lossy compressed cluster

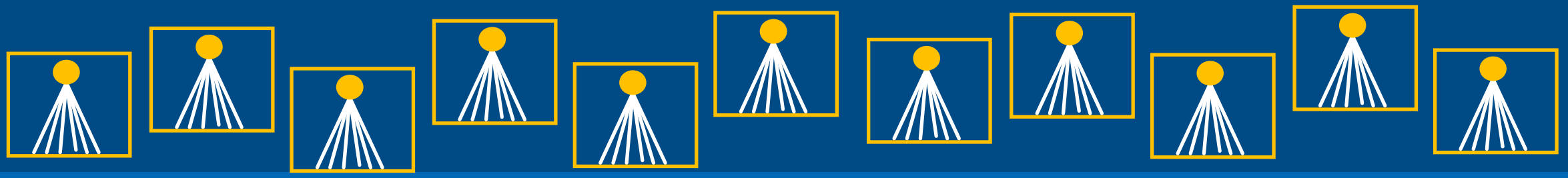


Cluster Decompression and Per Cluster BVH Build

- Decompresses cluster (≤ 128 quads) and directly convert into a QBVH6
  - Load cluster ➔ decompress into SLM ➔ convert to QBVH6 ➔ write out to memory
  - Omits unnecessary and memory bandwidth heavy intermediate steps

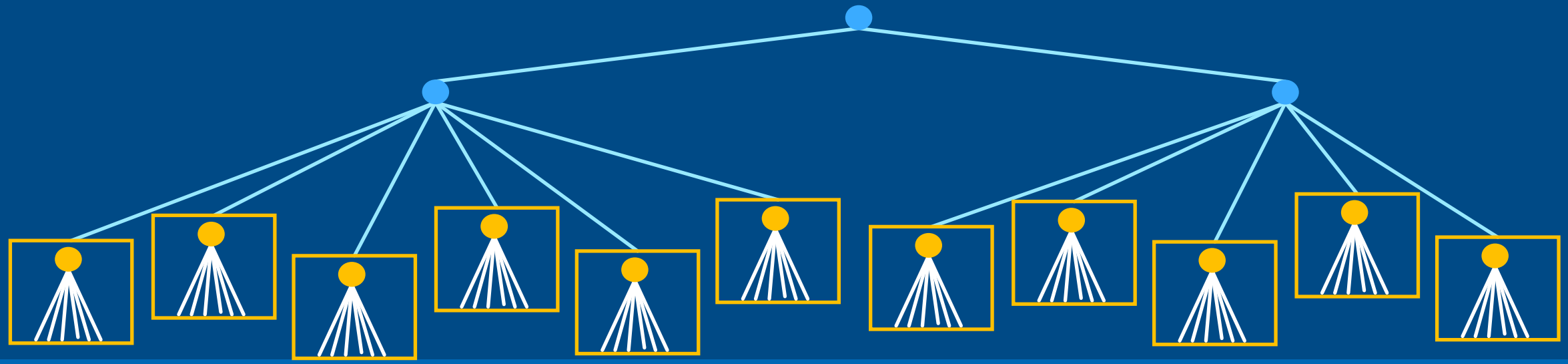


Cluster BVH Fusing



Cluster BVH Fusing

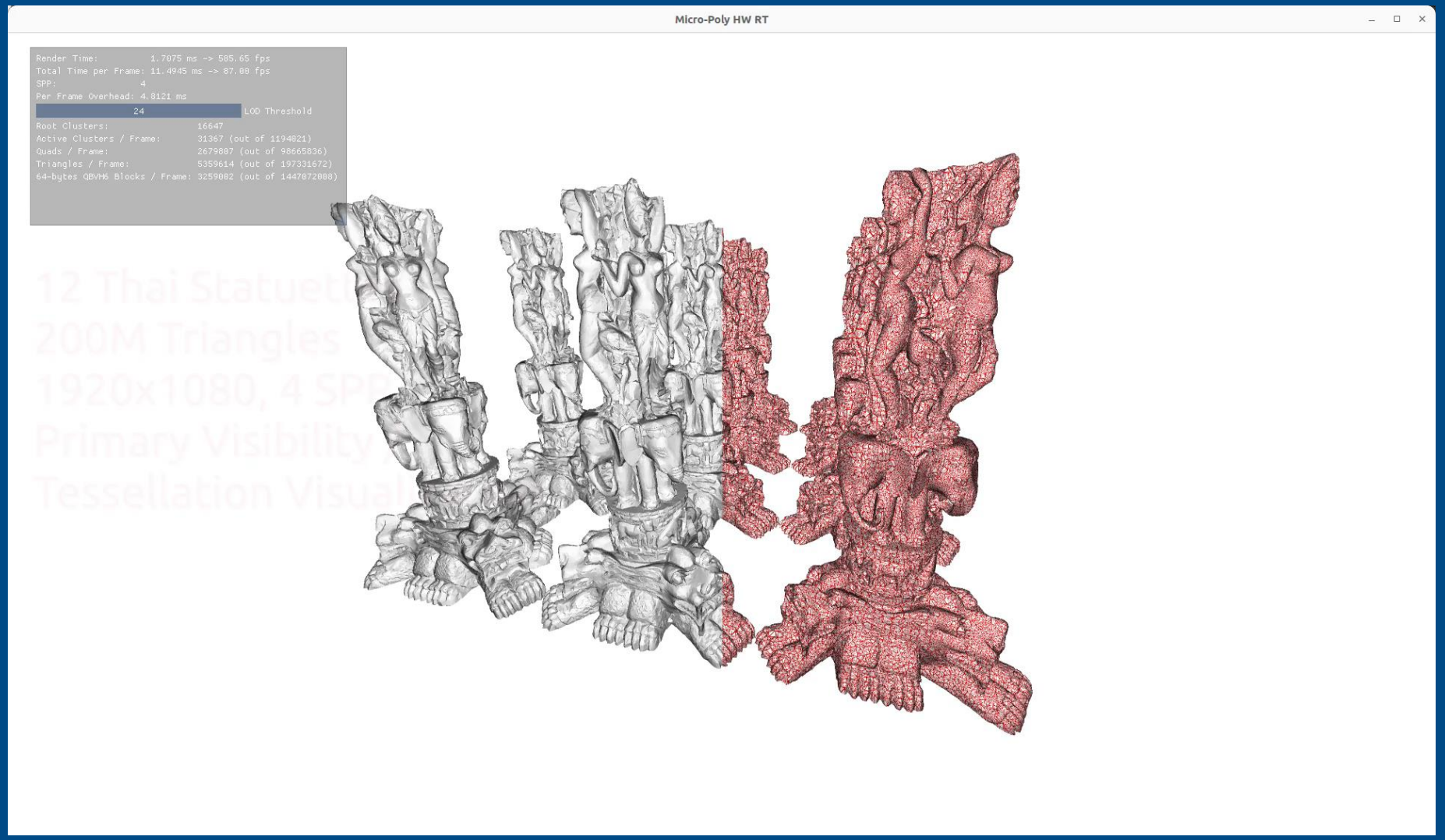
- Simply build top-level QBVH6 over selected cluster QBVH6s
  - Result is single QBVH6 (BLAS)
- Full build makes adding/removing new clusters trivial (e.g. streaming)



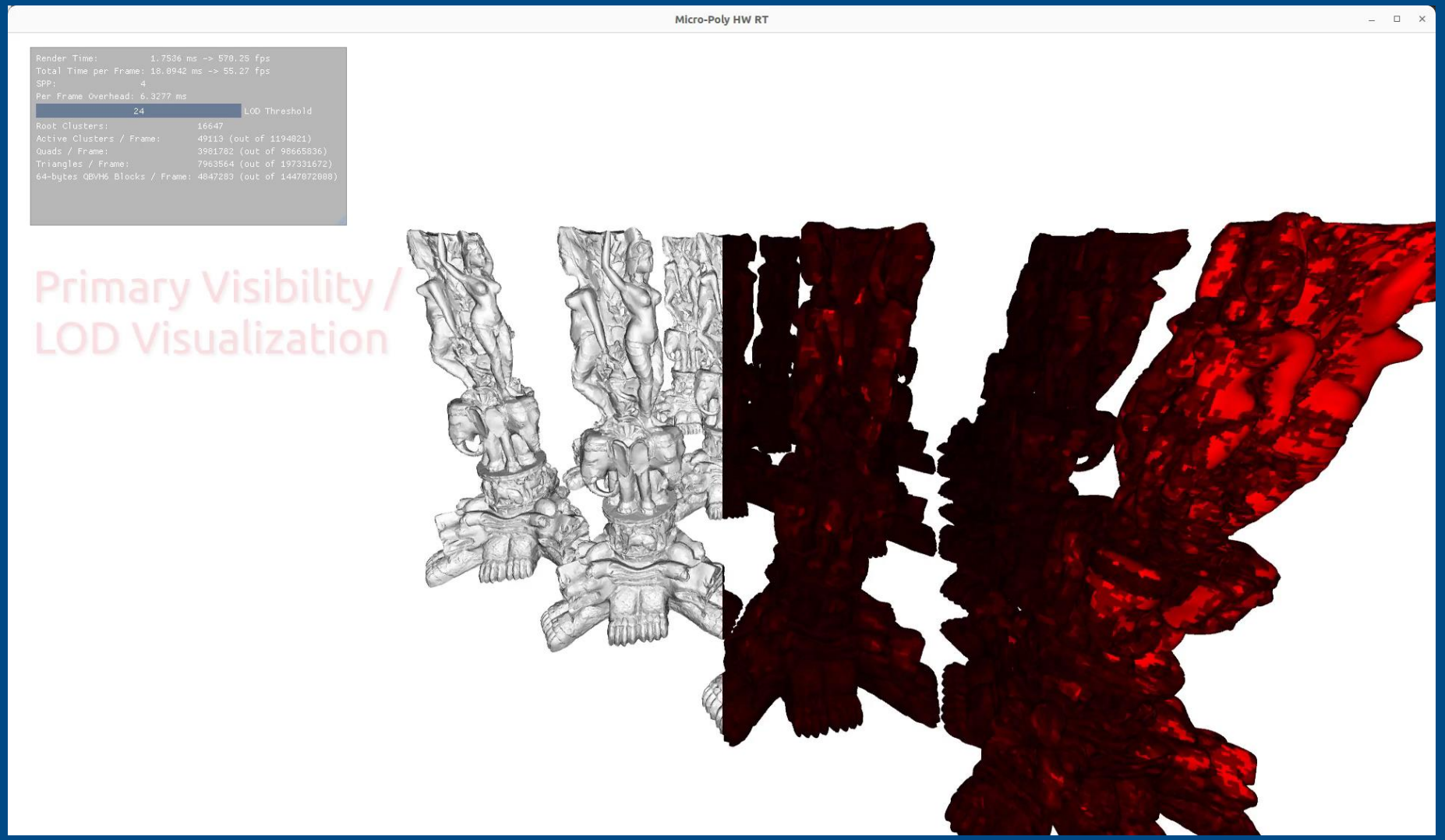
Results

- Intel Arc A770, 16 GB memory
- Modified Embree 4.0
  - Bypasses DXR/Vulkan API restrictions, e.g. compressed clusters as geometry type
  - Ray queries

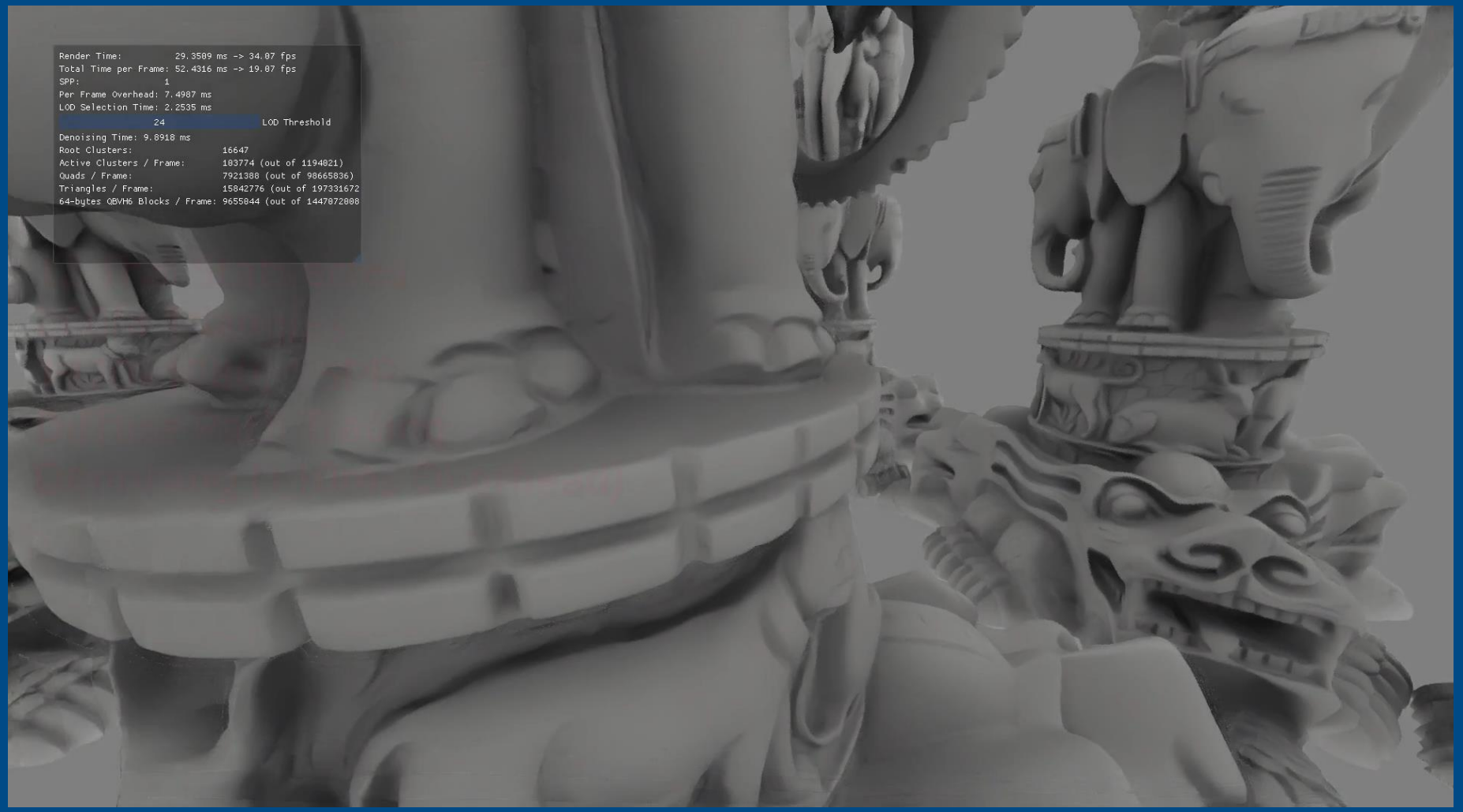
Tessellation/Cluster



LOD


Micro-Poly HW RT
Render Time: 1.7536 ms -> 570.25 fps
Total Time per Frame: 18.0942 ms -> 55.27 fps
SPP: 4
Per Frame Overhead: 6.3277 ms
24
LOD Threshold
Root Clusters: 16647
Active Clusters / Frame: 49113 (out of 1194021)
Quads / Frame: 3981782 (out of 98665836)
Triangles / Frame: 7963564 (out of 197331672)
64-bytes QBVH6 Blocks / Frame: 4847283 (out of 1447072008)
Primary Visibility / LOD Visualization

Path Tracing + Denoising



Per Frame Cost

	Thai	Rungholt	Landscape
Triangles	200 M	100 M	132 M
Triangles Per Frame	16 M	40 M	21 M
LOD Selection	1.7 ms	0.9 ms	1.2 ms
Decomp + Cluster QBVH6	2.4 ms	5.9 ms	2.6 ms
Fusing Cluster QBVH6s	1.6 ms	2.3 ms	2.0 ms
Total	5.7 ms	9.0 ms	5.8 ms
QBVH6 Build Perf	4.0 GTriangles/s	4.8 GTriangles/s	4.5 GTriangles/s

Per Frame Cost

	Thai	Rungholt	Landscape
Triangles	200 M	100 M	132 M
Triangles Per Frame	16 M	40 M	21 M
LOD Selection	1.7 ms	0.9 ms	1.2 ms
Decomp + Cluster QBVH6	2.4 ms	5.9 ms	2.6 ms
Fusing Cluster QBVH6s	1.6 ms	2.3 ms	2.0 ms
Total	5.7 ms	9.0 ms	5.8 ms
QBVH6 Build Perf	4.0 GTriangles/s	4.8 GTriangles/s	4.5 GTriangles/s

Per Frame Cost

	Thai	Rungholt	Landscape
Triangles	200 M	100 M	132 M
Triangles Per Frame	16 M	40 M	21 M
LOD Selection	1.7 ms	0.9 ms	1.2 ms
Decomp + Cluster QBVH6	2.4 ms	5.9 ms	2.6 ms
Fusing Cluster QBVH6s	1.6 ms	2.3 ms	2.0 ms
Total	5.7 ms	9.0 ms	5.8 ms
QBVH6 Build Perf	4.0 GTriangles/s	4.8 GTriangles/s	4.5 GTriangles/s

Build performance for QBVH6 with cluster hierarchy > 10x vs. QBVH6 over quads

Full Dynamic Content

```
Render Time:          0.7310 ms -> 1368.01 fps
Total Time per Frame: 12.1581 ms -> 82.25 fps
SPP:                  4
Per Frame Overhead: 4.6910 ms
LOD Selection Time: 1.2507 ms
                  24                  LOD Threshold
Root Clusters:                0
Active Clusters / Frame:      52001 (out of 0)
Quads / Frame:                131664 (out of 0)
Triangles / Frame:            263328 (out of 0)
64-bytes QBVH6 Blocks / Frame: 199144 (out of 0)
```

Fully Dynamic Content
52K Patches
4 SPP
Per-Frame
- Patch LOD selection
- Patch Tessellation
- Cluster Conversion

Conclusion

- Cluster-based BLAS construction extremely fast

Conclusion

- Cluster-based BLAS construction extremely fast
- Getting close to real-time hierarchical LOD with HW-accelerated RT

Conclusion

- Cluster-based BLAS construction extremely fast
- Getting close to real-time hierarchical LOD with HW-accelerated RT
- Lossy compressed cluster/mesh should be a new primitive type

Future Work

- Need more dense representations
  - Lossy compressed clusters (delta encoding + prediction)
  - HW-supported geometry representation inside the BLAS
- Need a standardized lossy compressed cluster/mesh primitive type
  - DXR/Vulkan API support

Questions?

Full Video


intel