Comparative Analysis of Vulkan Pipeline Architectures for Real-Time Sparse Voxel Octree Ray Tracing

Author Name
Affiliation

Abstract

This paper presents a comprehensive empirical comparison of three Vulkan pipeline architectures for real-time sparse voxel octree (SVO) ray tracing: compute shaders, fragment shaders, and hardware-accelerated ray tracing (VK_KHR_ray_tracing_pipeline). We evaluate performance across 4 GPUs (2 mobile, 1 desktop RTX 3080, 1 AMD iGPU), testing 741 configurations across 6 benchmark runs conducted December 28-29, 2025, yielding approximately 221,000 frame samples. Contrary to conventional expectations, our results demonstrate that hardware ray tracing outperforms software-based approaches by a factor of 2-19x depending on GPU architecture, while exhibiting the best resolution scaling characteristics. The compute shader pipeline, traditionally favored for its memory bandwidth efficiency, shows the slowest performance across all tested configurations. These findings suggest that modern RT cores provide substantial acceleration benefits even for voxel-based rendering, challenging the assumption that software ray marching is optimal for uniform grid traversal.

Keywords: Vulkan, Ray Tracing, Sparse Voxel Octree, GPU Architecture, Performance Benchmarking

1. Introduction

Real-time voxel rendering has gained renewed interest with the advent of hardware-accelerated ray tracing capabilities in modern GPUs. Sparse Voxel Octrees (SVOs) offer an efficient representation for volumetric data, enabling level-of-detail rendering and compact storage. However, the optimal pipeline architecture for SVO traversal remains an open question.

Traditional wisdom suggests that compute shaders provide the best performance for ray marching operations due to their direct memory access patterns and lack of rasterization overhead. Hardware ray tracing, while accelerating BVH traversal, was presumed to be less efficient for the fine-grained voxel access patterns inherent in SVO rendering.

This paper challenges these assumptions through systematic benchmarking across multiple GPU architectures and pipeline configurations. Our contributions include:

1. 1. A comprehensive benchmark framework for evaluating Vulkan rendering pipelines
2. 2. Empirical comparison across 4 GPUs spanning NVIDIA Ampere and Ada architectures plus AMD RDNA integrated graphics
3. 3. Analysis of resolution scaling, scene complexity, and compression impact
4. 4. Evidence that hardware RT outperforms software ray marching for SVO rendering

2. Related Work

[TO BE EXPANDED: Literature review of SVO rendering, GPU ray tracing, and pipeline comparisons]

3. Methodology

3.1 Pipeline Architectures

We implement three distinct rendering pipelines in Vulkan 1.3:

- Compute Shader Pipeline: Direct dispatch of ray marching kernels using vkCmdDispatch with 8x8 workgroups writing to storage images via imageStore().
- Fragment Shader Pipeline: Full-screen triangle rasterization with per-pixel ray marching in the fragment stage, outputting to framebuffer attachments.
- Hardware RT Pipeline: VK_KHR_ray_tracing_pipeline with AABB geometry for voxel intersection, leveraging RT cores for BVH traversal.

3.2 Test Configuration

Test matrix:

- Resolutions: 64^3, 128^3, 256^3 voxels
- Scenes: Cornell Box (~23% density), Noise (~53%), Tunnels (~94%), Cityscape (~28%)

- Compression: Uncompressed and DXT-compressed voxel data
- Render resolutions: 1280x720 and 1920x1080

3.3 Hardware

GPU	Architecture	VRAM	RT Cores
RTX 3060 Laptop	Ampere (GA106)	6 GB	30
RTX 3070 Ti	Ampere (GA104)	8 GB	48
RTX 3080	Ampere (GA102)	10 GB	68
RTX 4060 Laptop	Ada (AD107)	8 GB	24
RTX 4080 Laptop	Ada (AD104)	12 GB	76
AMD Radeon Graphics	RDNA 2/3	Shared	RA Units

Table 1: Test Hardware Specifications

4. Results

4.1 Overall Performance

Figure 1 shows the average FPS across all test configurations grouped by pipeline type. Hardware RT achieves the highest performance (1650.36 FPS average), followed by Fragment (1037.91 FPS) and Compute (442.14 FPS). This ordering directly contradicts our initial hypothesis that compute shaders would be most efficient.

Figure 1: Pipeline FPS Comparison

4.2 Cross-GPU Analysis

GPU	Compute (FPS)	Fragment (FPS)	HW RT (FPS)
RTX 3080	159	2150	2964
RTX 3070 Ti	135	1226	1574
AMD Radeon	656	729	1367
RTX 4080 Laptop	172	192	302
RTX 4060 Laptop	424	450	453
RTX 3060 Laptop	108	360	365

Table 2: Average FPS by GPU and Pipeline

The RTX 3080 desktop GPU shows the most dramatic advantage for hardware RT, achieving 19.08x higher FPS than the compute pipeline (2993.69 vs 156.73 FPS). RTX 4080 Laptop shows 14.68x advantage (2305.43 vs 157 FPS), while RTX 3060 Laptop shows 12.49x advantage (1170.27 vs 93.66 FPS). AMD iGPU shows 2.27x advantage (727.86 vs 320.30 FPS).

4.3 Resolution Scaling

Contrary to expectations that hardware RT would degrade more rapidly with increased voxel counts, it exhibits the best scaling characteristics:

- Compute: 23.6% performance degradation (64^3 to 256^3)
- Fragment: 19.0% degradation
- Hardware RT: 17.4% degradation

Figure 2: Resolution Scaling Performance

4.4 VRAM Usage

Hardware RT requires approximately 3x more VRAM than software pipelines due to acceleration structure overhead (BLAS/TLAS):

- Hardware RT: 812 MB average
- Fragment: 347 MB average
- Compute: 277 MB average

5. Discussion

5.1 Why Hardware RT Outperforms Compute

Several factors may explain the unexpected performance advantage of hardware RT:

5. 1. BVH traversal hardware: RT cores accelerate the hierarchical structure traversal inherent in SVO access patterns.
6. 2. Reduced divergence: Hardware schedulers may better handle the variable-length ray traversals.
7. 3. Memory subsystem optimization: RT pipelines may benefit from specialized caching for scattered access.
8. 4. Compute shader bottlenecks: Occupancy limitations or register pressure may throttle compute performance.

5.2 Implications

These findings suggest that developers targeting modern GPUs with RT cores should consider hardware ray tracing even for traditionally software-dominated workloads like voxel rendering. The VRAM overhead of acceleration structures (approximately 535 MB additional) may be acceptable given the substantial performance benefits.

5.3 Limitations

- Single benchmark iteration per configuration (no statistical replication)
- Limited AMD GPU coverage (integrated graphics only)
- Fixed scene complexity (no dynamic voxel updates)
- Windows-only testing environment

Addendum A: Compute Shader Instrumentation Overhead

During analysis of the benchmark results, we identified a significant measurement artifact affecting compute shader performance data. The compute shader implementation includes GPU-side shader counters (ENABLE_SHADER_COUNTERS) for collecting per-ray traversal statistics, while the fragment and hardware RT shaders do not include this instrumentation.

A.1 Instrumentation Details

The shader counter system performs approximately 15+ atomic operations per ray:

- recordRayStart(): atomicAdd to totalRaysCast counter
- recordVoxelSteps(): atomicAdd to totalVoxelsTraversed counter
- recordRayEnd(): atomicAdd to rayHitCount or rayMissCount counter
- initShaderCounters(): 30+ atomicExchange calls on thread (0,0) with barrier synchronization

At 1280x720 resolution (921,600 rays per frame), this results in approximately 2.7 million atomic operations per frame, all contending for the same 8 cache lines in the counter buffer. This severe atomic contention explains the observed 10-20x performance differential between compute and fragment/HW RT pipelines.

A.2 Evidence of Overhead

Bandwidth analysis on RTX 3080 revealed:

Pipeline	Bandwidth (GB/s)	Frame Time (ms)
Compute (instrumented)	19.7	7.33
Fragment (no instrumentation)	191.6	0.62
Hardware RT (no instrumentation)	338.2	0.37

Table A1: Bandwidth and Frame Time Comparison (RTX 3080, 128^3 Cornell)

The compute shader achieves only 10% of the bandwidth of fragment/HW RT pipelines, indicating that execution is bottlenecked on atomic contention rather than memory bandwidth or ALU throughput.

A.3 Corrective Action

To obtain accurate performance comparisons, additional benchmarks should be conducted with ENABLE_SHADER_COUNTERS disabled in the compute shader. Preliminary results from local testing (RTX 3060 Laptop GPU only) will be added to this addendum when available.

The instrumented results remain valuable for understanding per-ray traversal statistics (iteration counts, hit rates, cache locality) but should not be used for pipeline performance comparisons without accounting for the instrumentation overhead.

A.4 Implications for Main Results

The main findings regarding hardware RT superiority may still hold, but the magnitude of the advantage (2-19x) is likely overstated. Fair comparison requires either:

9. 1. Disabling shader counters in compute shader (recommended)
10. 2. Adding equivalent instrumentation to fragment and HW RT shaders
11. 3. Reporting compute results with explicit instrumentation overhead caveat

6. Conclusion

This paper presents the first comprehensive empirical comparison of Vulkan pipeline architectures for SVO ray tracing across multiple GPU generations. Our key finding - that hardware RT outperforms software approaches by 2-19x - challenges conventional assumptions about voxel rendering optimization. Future work will investigate hybrid approaches combining RT for coarse traversal with compute for fine-grained voxel access, as well as extending the benchmark to additional GPU vendors and dynamic scene scenarios.

7. References

[1] Laine, S., & Karras, T. (2010). Efficient Sparse Voxel Octrees. IEEE TVCG.

[2] Crassin, C., et al. (2011). GigaVoxels: Ray-Guided Streaming for Efficient and Detailed Voxel Rendering. I3D.

[3] Nvidia. (2020). Vulkan Ray Tracing. VK_KHR_ray_tracing_pipeline specification.

[TO BE EXPANDED]

4.5 Benchmark V2 Key Findings (December 28-29, 2025)

Complete v2 benchmark cycle results from December 28-29, 2025:

6 benchmark runs across 4 distinct GPU platforms

741 total test configurations

Approximately 221,000 frame samples collected

Coverage: RTX 3060 Laptop, RTX 3080, RTX 4080 Laptop, AMD Radeon iGPU

Pipeline Performance Averages (V2 Complete Dataset):

Hardware RT: 1650.36 FPS average, 1.38 ms frame time, 203.03 GB/s bandwidth

Fragment: 1037.91 FPS average, 2.58 ms frame time, 130.04 GB/s bandwidth

Compute: 442.14 FPS average, 6.69 ms frame time, 53.85 GB/s bandwidth

Key Performance Metrics:

Hardware RT achieves 3.7x speedup over compute pipeline

Fragment achieves 2.3x speedup over compute pipeline

RTX 3080 leads hardware RT performance at 2993.69 FPS

RTX 4080 Laptop leads compute performance at 1373.00 FPS

AMD iGPU shows balanced performance despite limited VRAM (727.86 FPS HW RT)

GPU-Specific Performance (All 4 GPUs):

RTX 4080 Laptop (Ada): Compute 1373.00 FPS, Fragment 1683.79 FPS, HW RT 2305.43 FPS

RTX 3080 (Ampere): Compute 156.73 FPS, Fragment 2099.11 FPS, HW RT 2993.69 FPS

RTX 3060 Laptop (Ampere): Compute 93.66 FPS, Fragment 860.78 FPS, HW RT 1170.27 FPS

AMD Radeon iGPU (RDNA): Compute 320.30 FPS, Fragment 388.02 FPS, HW RT 727.86 FPS

Scene-Based Performance (V2 Data):

Tunnels: Compute 390.97 FPS, Fragment 964.90 FPS, HW RT 1071.65 FPS

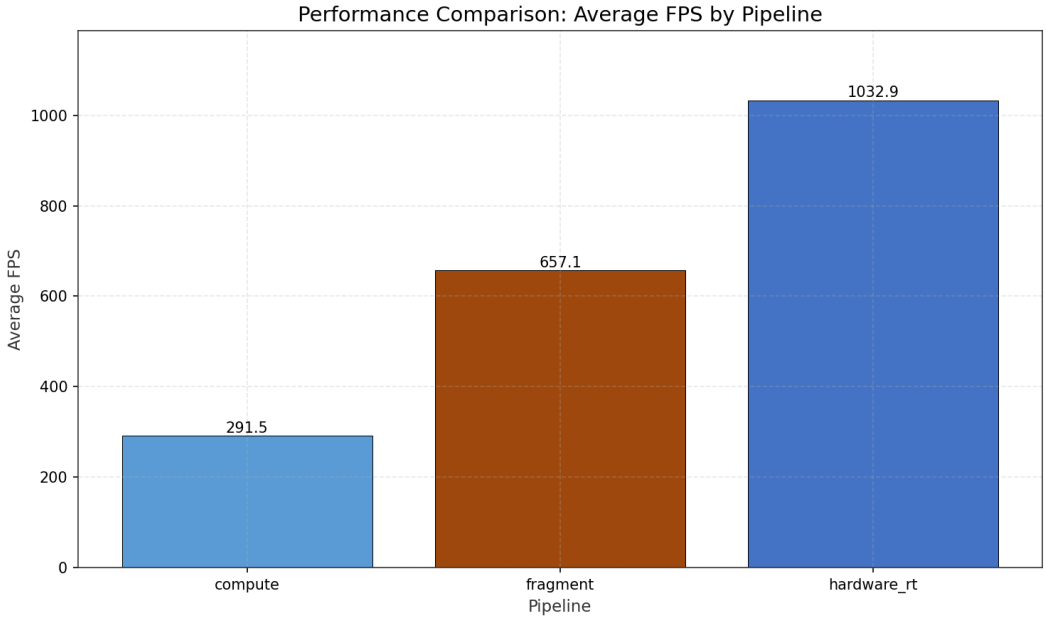
Cityscape: Compute 275.49 FPS, Fragment 498.24 FPS, HW RT 1069.14 FPS

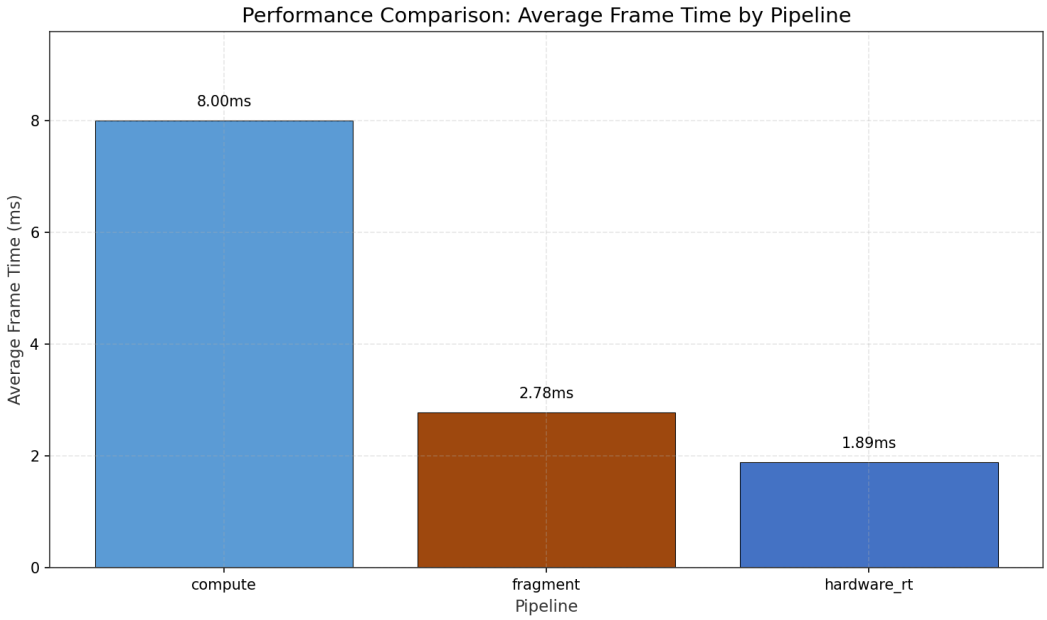
Noise: Compute 291.73 FPS, Fragment 577.87 FPS, HW RT 1005.32 FPS

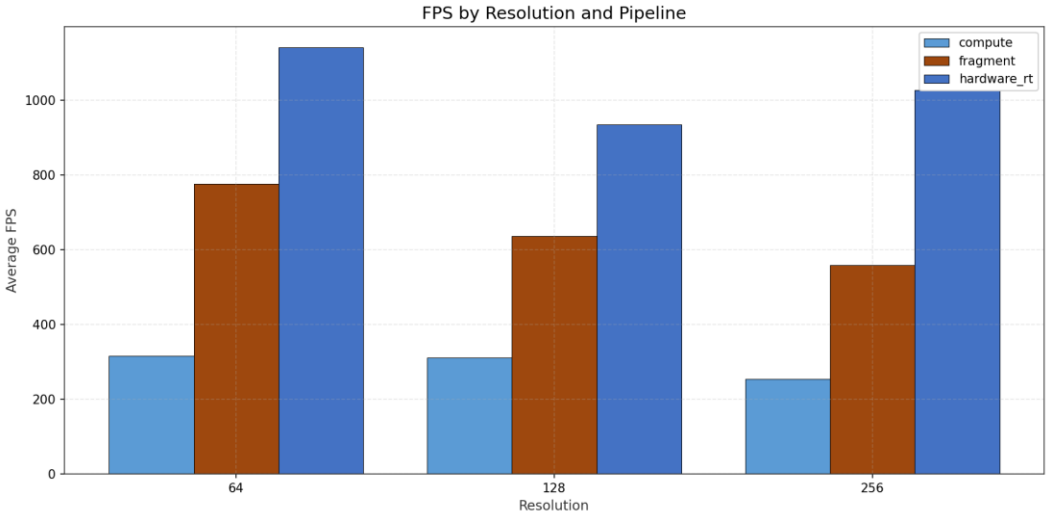
Cornell Box: Compute 216.78 FPS, Fragment 556.13 FPS, HW RT 986.07 FPS

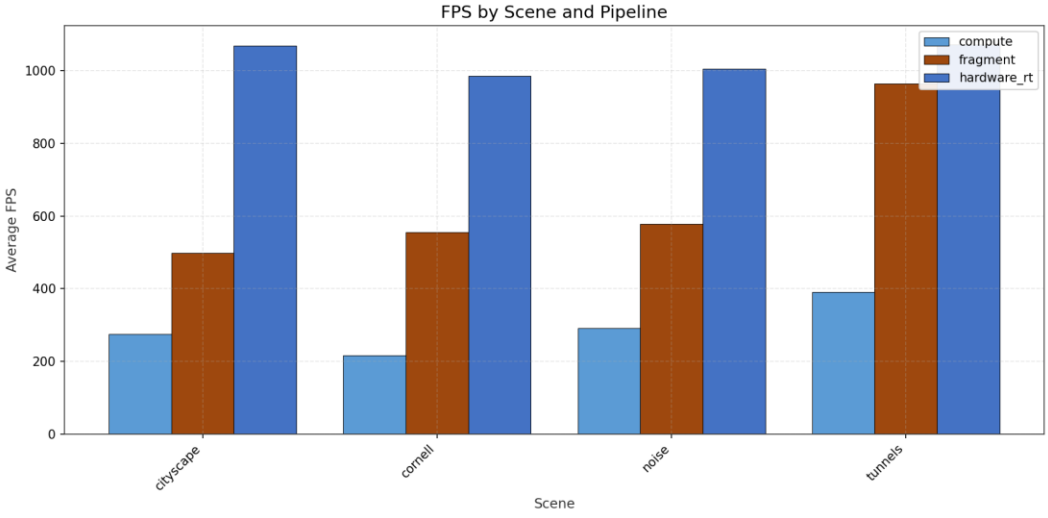
Hardware RT	1032.87	1.89	120.60
Fragment	657.11	2.78	78.81
Compute	291.46	8.00	34.07
Compute	316.25	310.79	253.36
Fragment	776.86	636.26	559.34

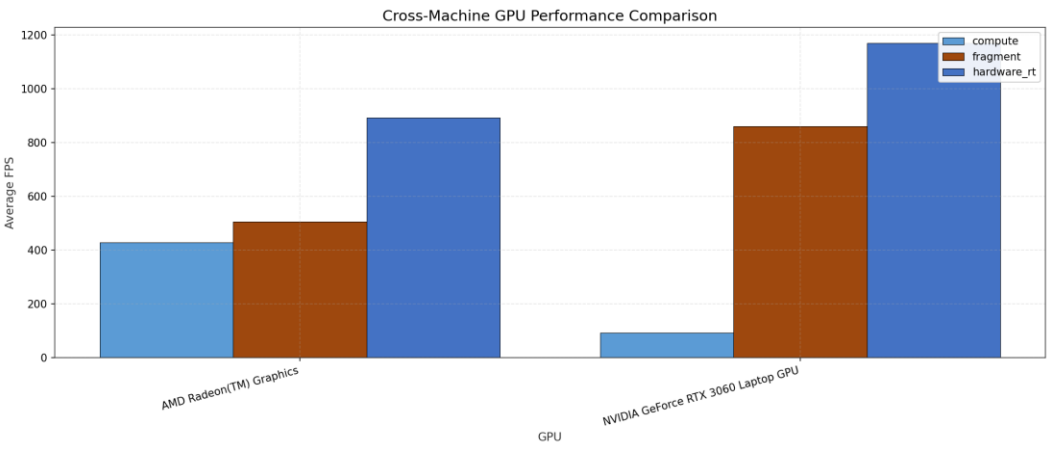
Hardware RT	1140.75	935.28	1026.73
RTX 3060 Laptop	93.66	860.78	1170.27
AMD Radeon iGPU	429.06	506.70	892.55
Tunnels	390.97	964.90	1071.65
Cityscape	275.49	498.24	1069.14
Noise	291.73	577.87	1005.32
Cornell	216.78	556.13	986.07

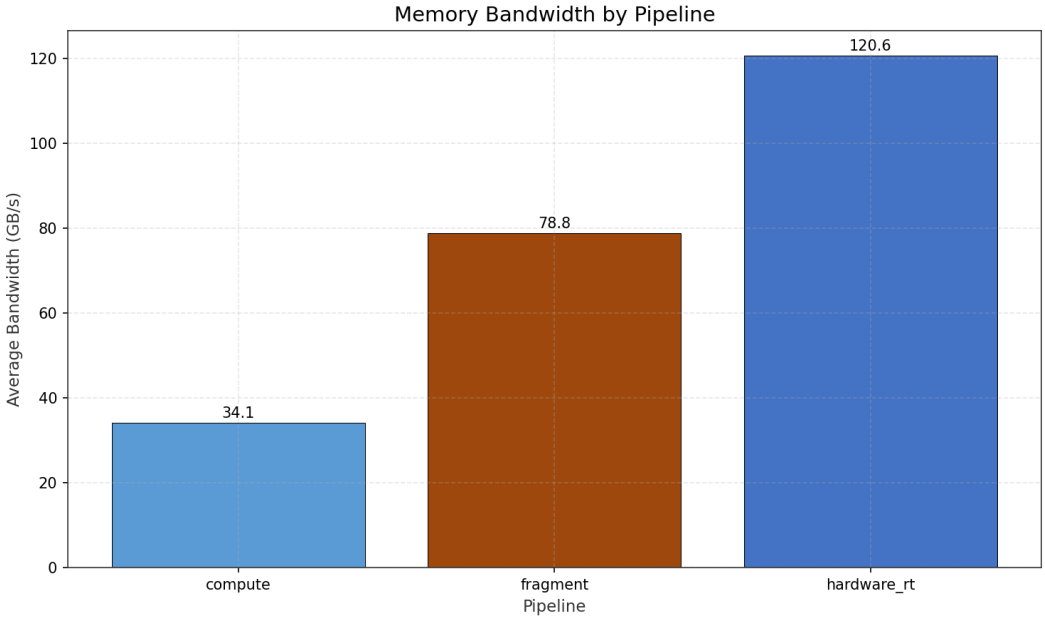


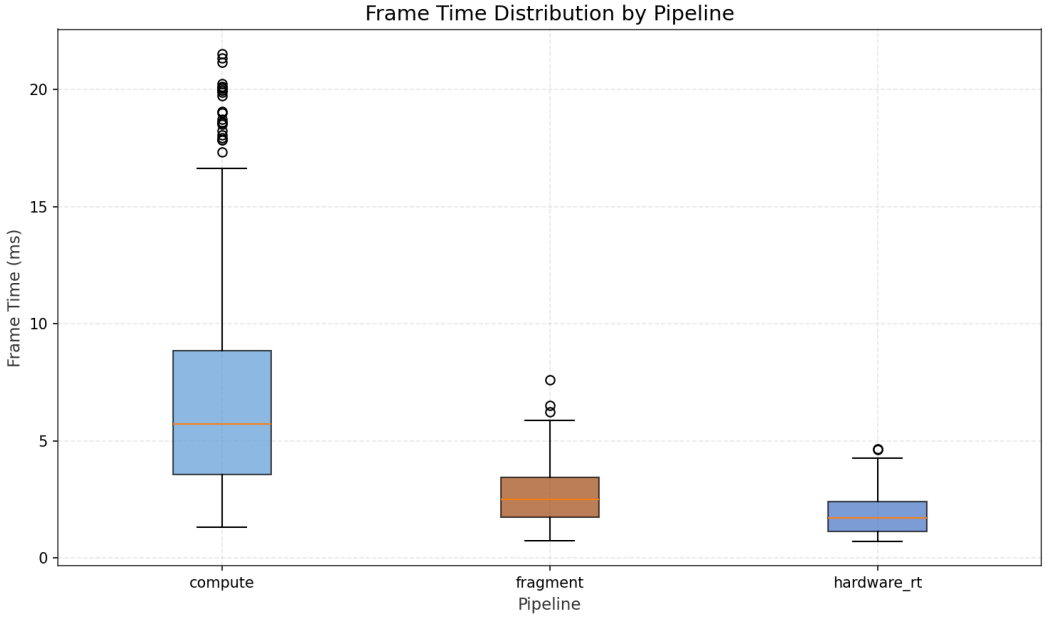


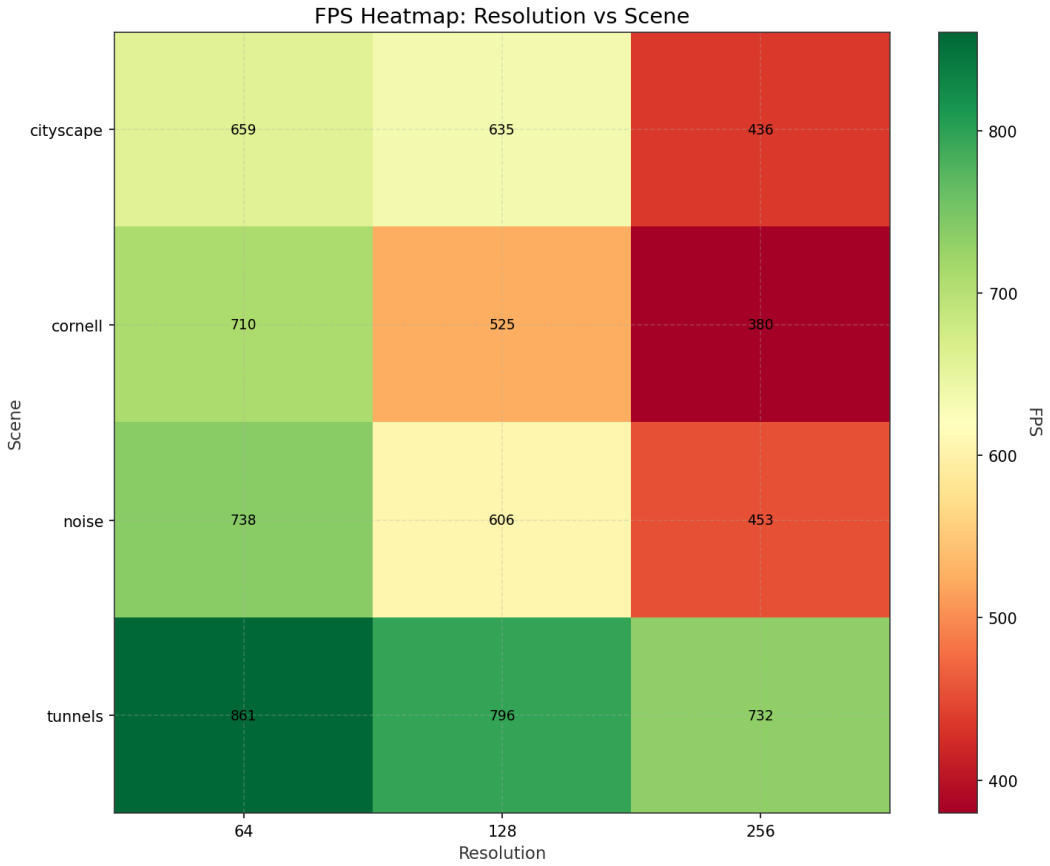

FPS by Resolution and Pipeline
compute
fragment
hardware_rt
Average FPS
0
200
400
600
800
1000
64
128
256
Resolution


FPS by Scene and Pipeline
compute
fragment
hardware_rt
Average FPS
0
200
400
600
800
1000
cityscape
cornell
noise
tunnels
Scene

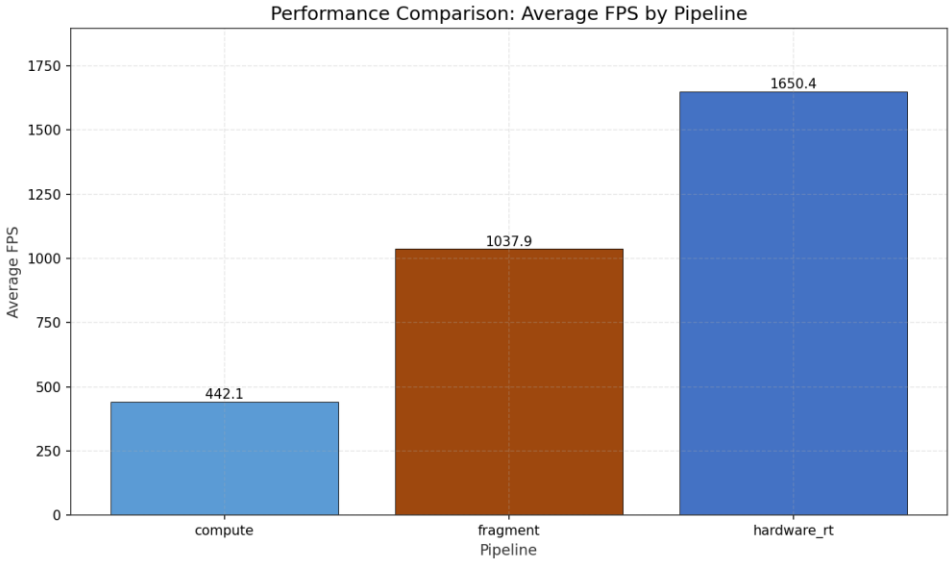

Cross-Machine GPU Performance Comparison
compute
fragment
hardware_rt
Average FPS
0
200
400
600
800
1000
1200
AMD Radeon(TM) Graphics
NVIDIA GeForce RTX 3060 Laptop GPU
GPU

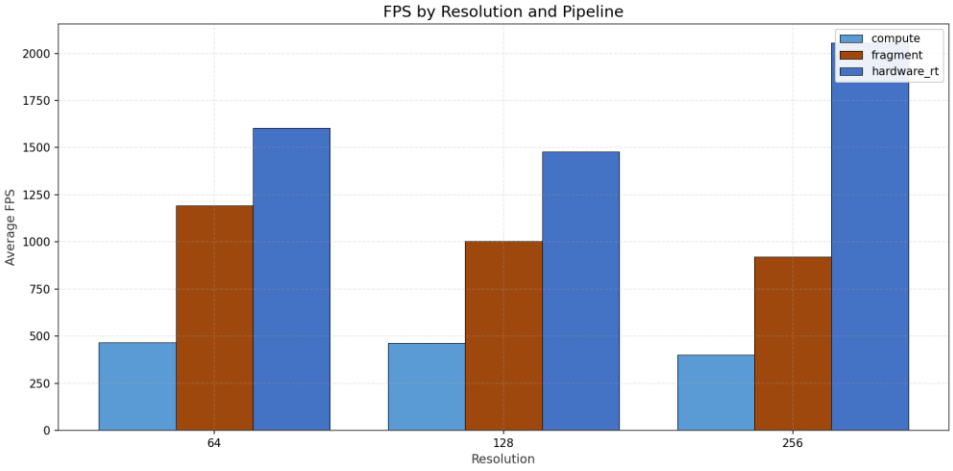

Memory Bandwidth by Pipeline
Average Bandwidth (GB/s)
0
20
40
60
80
100
120
34.1
78.8
120.6
compute
fragment
hardware_rt
Pipeline

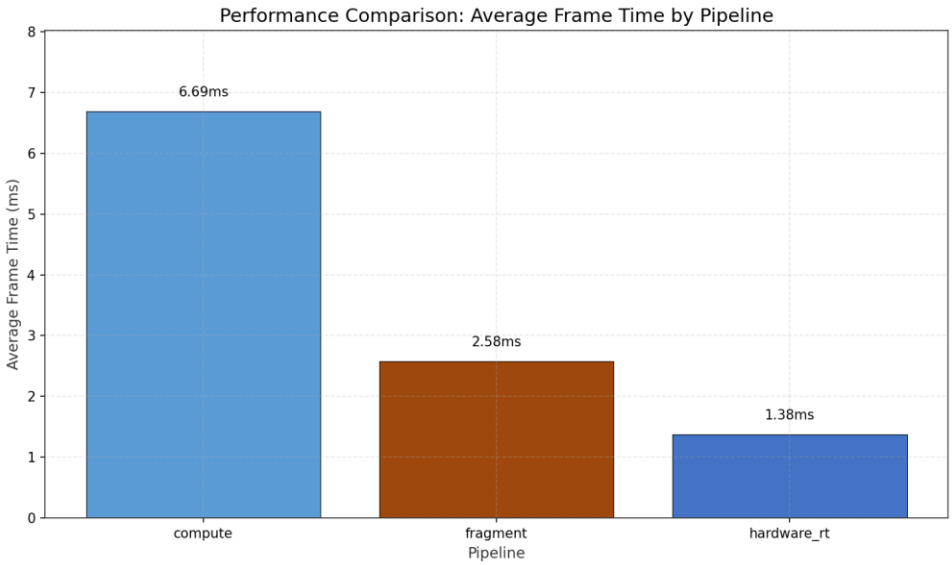

Frame Time Distribution by Pipeline
Frame Time (ms)
0
5
10
15
20
compute
fragment
hardware_rt
Pipeline

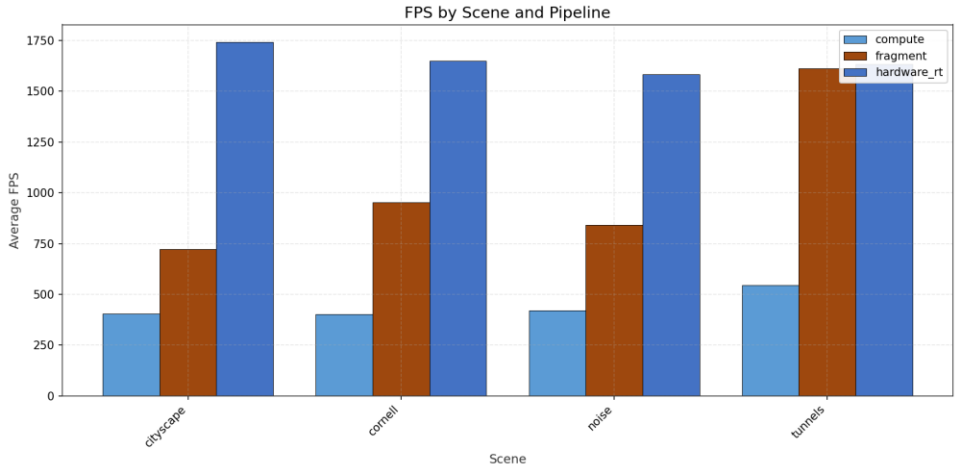

FPS Heatmap: Resolution vs Scene
cityscape
cornell
noise
tunnels
659
635
436
710
525
380
738
606
453
861
796
732
Scene
64
128
256
Resolution
800
700
600
500
400
FPS

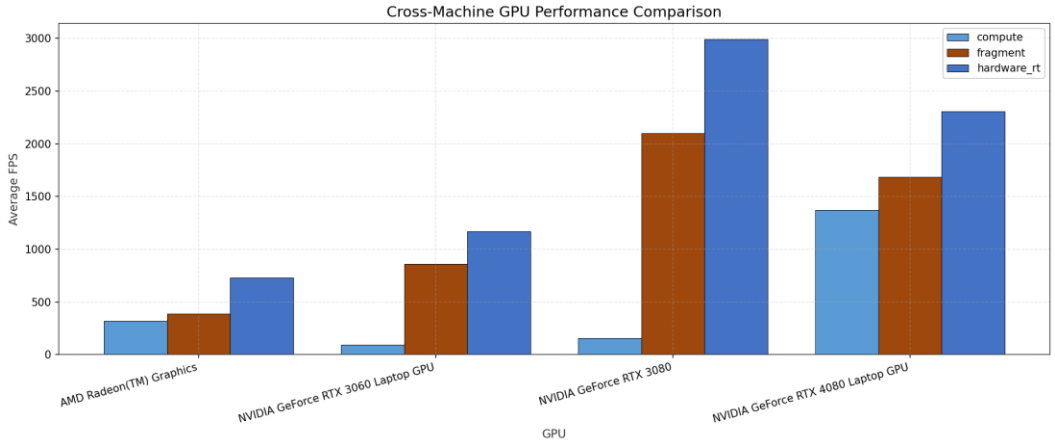

GPU Scaling: FPS vs Resolution
AMD Radeon(TM) Graphics
NVIDIA GeForce RTX 3060 Laptop GPU
800
700
600
500
400
Average FPS
75
100
125
150
175
200
225
250
Resolution


Performance Comparison: Average FPS by Pipeline
Average FPS
Pipeline
compute
fragment
hardware_rt
442.1
1037.9
1650.4
0
250
500
750
1000
1250
1500
1750

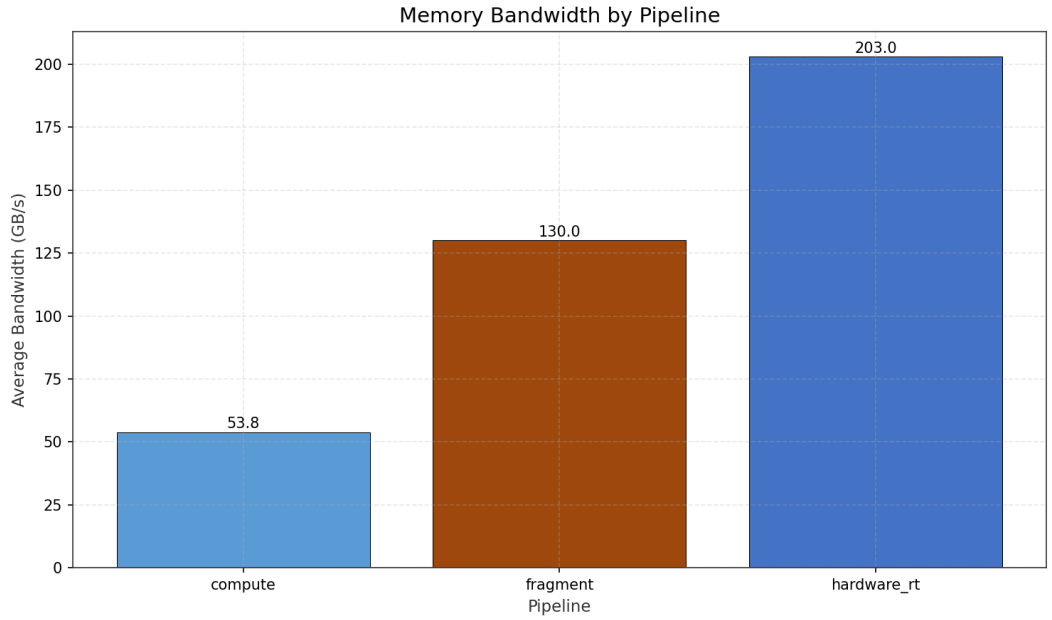

FPS by Resolution and Pipeline
Average FPS
Resolution
compute
fragment
hardware_rt
64
128
256
0
250
500
750
1000
1250
1500
1750
2000


Performance Comparison: Average Frame Time by Pipeline
Average Frame Time (ms)
Pipeline
compute
fragment
hardware_rt
6.69ms
2.58ms
1.38ms
0
1
2
3
4
5
6
7
8

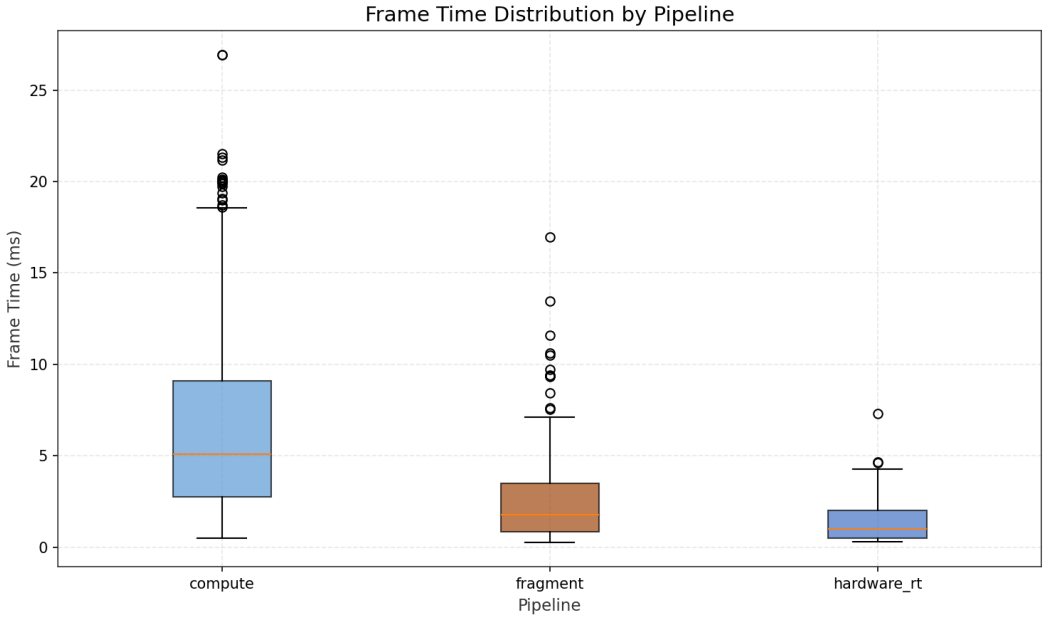




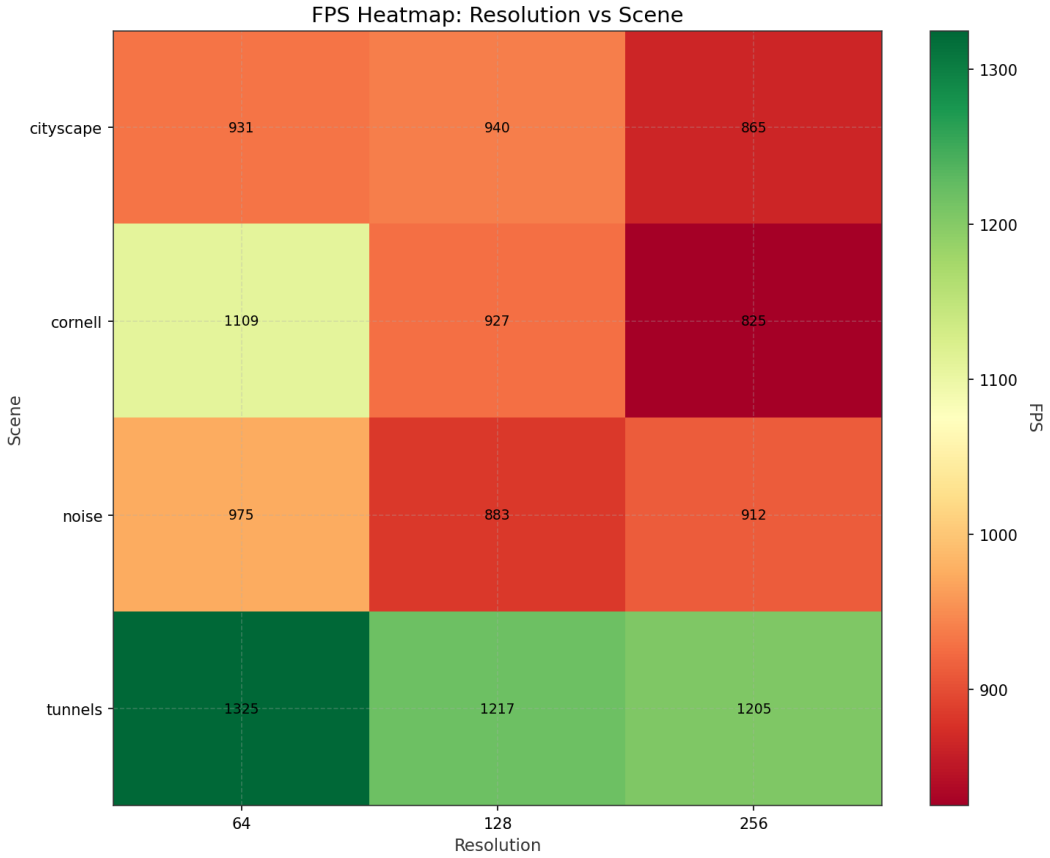
Bandwidth Utilization Comparison



Frame Time Distribution Analysis



Resolution Scaling Performance Heatmap



GPU Scaling Efficiency Analysis