Vulkanised 2026

The 8th Vulkan Developer Conference
San Diego, USA | February 9-11, 2026

Vulkan Debugging on Linux

Emma Anholt, Igalia S.L.

Intros

Me

- 20+ years in the Mesa project
- Major work on Intel, Broadcom (Raspberry Pi), and Qualcomm
- Core work in shader compilers, GL state and resource management, formats, utilities, testing and conformance.
- Support tools: deqp-runner, shader-db, traces-db, gpu-trace-perf

Mesa project

- Open source, conformant Vulkan drivers:
  - Intel, AMD, NVIDIA, Qualcomm, ARM, Raspberry Pi, Apple (asahi), Apple (Metal), virtio-venus, and CPU (lavapipe)
- In progress:
  - Imagination, Microsoft
- Ships with Linux distributions

Mesa contributors

- IHVs: ARM, AMD, Imagination, Intel, Raspberry Pi, Qualcomm
- ISVs: Collabora, LunarG, Global Logic, Google, Igalia, Microsoft, Red Hat, Valve

You should develop against Mesa

Why Mesa?

- Ability to debug
- Visibility into how the hardware gets driven
- Access to the engineers
- Turnaround time for fixes
- Someone will run your software against it anyway

gdb just works

```
Thread 1 "coopmat_int8" received signal SIGSEGV, Segmentation fault.
0x00007ffff5a82656 in brw_from_nir_emit_cs_intrinsic (ntb=..., instr=0x5555557741d8) at
../src/intel/compiler/brw_from_nir.cpp:4846
4846                assert(nval[0].u32 == nval[j].u32);
(gdb) bt
#0  0x00007ffff5a82656 in brw_from_nir_emit_cs_intrinsic (ntb=..., instr=0x5555557741d8) at
../src/intel/compiler/brw_from_nir.cpp:4846
[…]
#8  0x00007ffff32f834e in anv_shader_compile_cs (device=0x55555569e550, mem_ctx=0x5555556ecb20,
shader_data=0x5555556f71c0, error_str=0x7fffffffcdd0)
    at ../src/intel/vulkan/anv_shader_compile.c:954
[...]
#12 0x00007ffff5084bc1 in vk_common_CreateComputePipelines (_device=0x55555569e550, pipelineCache=0x0,
createInfoCount=1, pCreateInfos=0x7fffffffdb50,
    pAllocator=0x0, pPipelines=0x555555656ec8) at ../src/vulkan/runtime/vk_pipeline.c:2283
```


Debugging your rendering

First, turn off features

- Driver-specific environment vars/android props
- Disable texture compression
- Disable HiZ/LRZ
- Add full cache flushes and syncs per draw

KHR_pipeline_executable_properties

Disassembly

Disassembly type: KHR_pipeline_executable_properties

```
impl main {
    con block b0:         // preds:
    div 32       %0 = @load_vector_arg_amd (base=12, arg_upper_bound_u32_amd=0)
    div 32       %1 = @load_vector_arg_amd (base=11, arg_upper_bound_u32_amd=0)
    con 32       %2 = @load_scalar_arg_amd (base=1, arg_upper_bound_u32_amd=0)
    con 32       %3 = load_const (0x00000000 = 0.000000)
    con 32       %4 = load_const (0xffff8000 = -32768 = 4294934528)
    con 64       %5 = pack_64_2x32_split %2, %4 (0xffff8000)
    con 32x8     %6 = @load_global_amd (%5, %3 (0x0)) (base=0, access=readonly|reorderable|smem-amd, align_mul=32, a
    con 32       %7 = load_const (0x40800000 = 4.000000)
    div 32       %8 = fmul %1, %7 (4.000000)
    div 32       %9 = fmul %0, %7 (4.000000)
    con 32      %10 = load_const (0xc0000000 = -2.000000)
    div 32      %11 = fadd %8, %10 (-2.000000)
    div 32      %12 = fadd %9, %10 (-2.000000)
    div 32      %13 = f2i32 %12
    div 32      %14 = f2i32 %11
    div 32x2    %15 = vec2 %14, %13
    div 32x4    %16 = (float32)txf %6 (texture_handle), %15 (coord), %3 (0x0) (lod), 0 (texture), skip_helpers, 2D
    con 32      %17 = load_const (0xbf800000 = -1.000000)
    div 32      %18 = fadd %8, %17 (-1.000000)
    div 32      %19 = f2i32 %18
    div 32x2    %20 = vec2 %19, %13
    div 32x4    %21 = (float32)txf %6 (texture_handle), %20 (coord), %3 (0x0) (lod), 0 (texture), skip_helpers, 2D
    div 32      %22 = f2i32 %8
    div 32x2    %23 = vec2 %22, %13
    div 32x4    %24 = (float32)txf %6 (texture_handle), %23 (coord), %3 (0x0) (lod), 0 (texture), skip_helpers, 2D
    con 32      %25 = load_const (0x3f800000 = 1.000000)
    div 32      %26 = fadd %8, %25 (1.000000)
    div 32      %27 = f2i32 %26
    div 32x2    %28 = vec2 %27, %13
    div 32x4    %29 = (float32)txf %6 (texture_handle), %28 (coord), %3 (0x0) (lod), 0 (texture), skip_helpers, 2D
```


KHR_pipeline_executable_properties

Disassembly type: KHR_pipeline_executable_properties

```
1909    ; Final Assembly
1910
1911    BB0:
1912        s_setprio 2                                             ; bfb50002
1913        s_movk_i32 s3, 0x8000                                   ; b0038000
1914        s_load_b256 s[0:7], s[2:3], null                        ; f40c0001 f8000000
1915        v_mul_f32_e32 v1, 4.0, v1                               ; 100202f6
1916        v_mul_f32_e32 v2, 4.0, v2                               ; 100404f6
1917        s_delay_alu instid0(VALU_DEP_2)                         ; bf870002
1918        v_add_f32_e32 v0, -2.0, v1                              ; 060002f5
1919        v_add_f32_e32 v6, -1.0, v1                              ; 060c02f3
1920        v_cvt_i32_f32_e32 v7, v1                                ; 7e0e1101
1921        v_add_f32_e32 v1, 1.0, v1                               ; 060202f2
1922        v_add_f32_e32 v3, -2.0, v2                              ; 060604f5
1923        v_add_f32_e32 v8, -1.0, v2                              ; 061004f3
1924        v_cvt_i32_f32_e32 v9, v2                                ; 7e121102
1925        v_add_f32_e32 v2, 1.0, v2                               ; 060404f2
1926        v_cvt_i32_f32_e32 v4, v0                                ; 7e081100
1927        v_cvt_i32_f32_e32 v6, v6                                ; 7e0c1106
1928        v_cvt_i32_f32_e32 v1, v1                                ; 7e021101
1929        v_cvt_i32_f32_e32 v5, v3                                ; 7e0a1103
1930        v_cvt_i32_f32_e32 v8, v8                                ; 7e101108
1931        v_cvt_i32_f32_e32 v2, v2                                ; 7e041102
1932        s_waitcnt lgkmcnt(0)                                    ; bf89fc07
1933        s_clause 0xf                                            ; bf85000f
1934        image_load v[12:15], v[4:5], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f84 00000c04
1935        image_load v[16:19], [v6, v5], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f85 00001006 00000005
1936        image_load v[20:23], [v7, v5], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f85 00001407 00000005
1937        image_load v[24:27], [v1, v5], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f85 00001801 00000005
1938        image_load v[28:31], [v4, v8], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f85 00001c04 00000008
1939        image_load v[32:35], [v6, v8], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f85 00002006 00000008
1940        image_load v[36:39], v[7:8], s[0:7] dmask:0xf dim:SQ_RSRC_IMG_2D unorm ; f0000f84 00002407
```


KHR_pipeline_executable_properties

Disassembly type: KHR_pipeline_executable_properties

```
37
38    ---- NIR Shader(s) ----
39
40    ; The optimized NIR shader(s)
41
42    shader: MESA_SHADER_FRAGMENT
43    source_blake3: {0xe3c8bb2a, 0xbdea236b, 0xd63651bc, 0x99edb237, 0xe4a1f60d, 0xdf568539, 0xc4b02f8b, 0x45574bdc}
44    prev_stage: MESA_SHADER_VERTEX
45    num_textures: 1
46    outputs_written: 4
47    system_values_read: 0x00000000'00000000'00000000'00080000
48    float_controls_execution_mode: 0x00000010
49    api_subgroup_size: 64
50    max_subgroup_size: 64
51    min_subgroup_size: 64
52    bit_sizes_float: 0x20
53    bit_sizes_int: 0x21
54    separate_shader: true
55    known_interpolation_qualifiers: true
56    flrp_lowered: true
57    origin_upper_left: true
```


NIR compiler debug

```
# MESA_SHADER_CACHE_DISABLE=1 NIR_DEBUG=print renderdoccmd replay […]
nir_lower_vars_to_ssa
shader: MESA_SHADER_FRAGMENT
source_blake3: {0xe3c8bb2a, 0xbdea236b, 0xd63651bc, 0x99edb237, […]}
[…]
    32     %162 = deref_var &ps_frag_coord (system vec4)
    32x4   %163 = @load_deref (%162) (access=none)
[…]
nir_propagate_invariant
nir_merge_clip_cull_distance_vars
nir_lower_doubles
ac_nir_lower_sin_cos
nir_lower_system_values
shader: MESA_SHADER_FRAGMENT
source_blake3: {0xe3c8bb2a, 0xbdea236b, 0xd63651bc, 0x99edb237, […]}
[…]
    32x4   %141 = @load_frag_coord
```

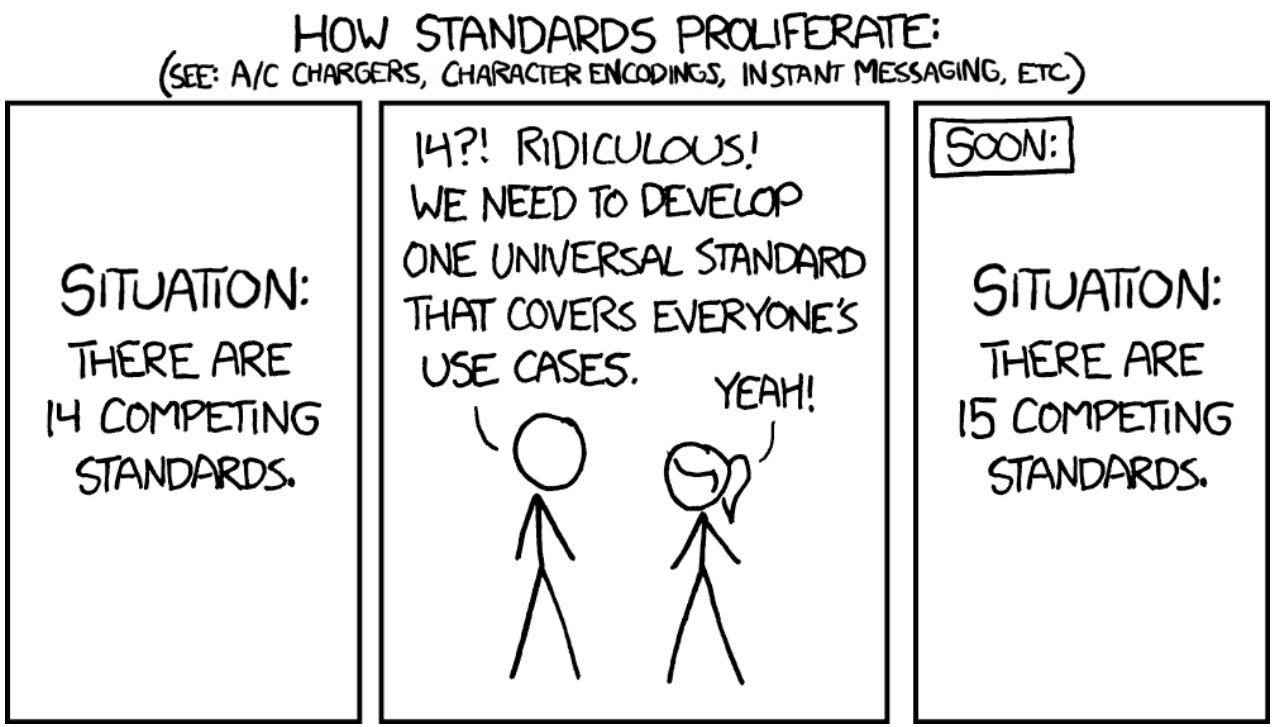

GPU hangs

- AMD: devcore, UMR
- Intel: devcore, I915_DUMP_CMD=1
- Qualcomm: devcore, TU_DEBUG=rd
- Broadcom: V3D_DEBUG=clif

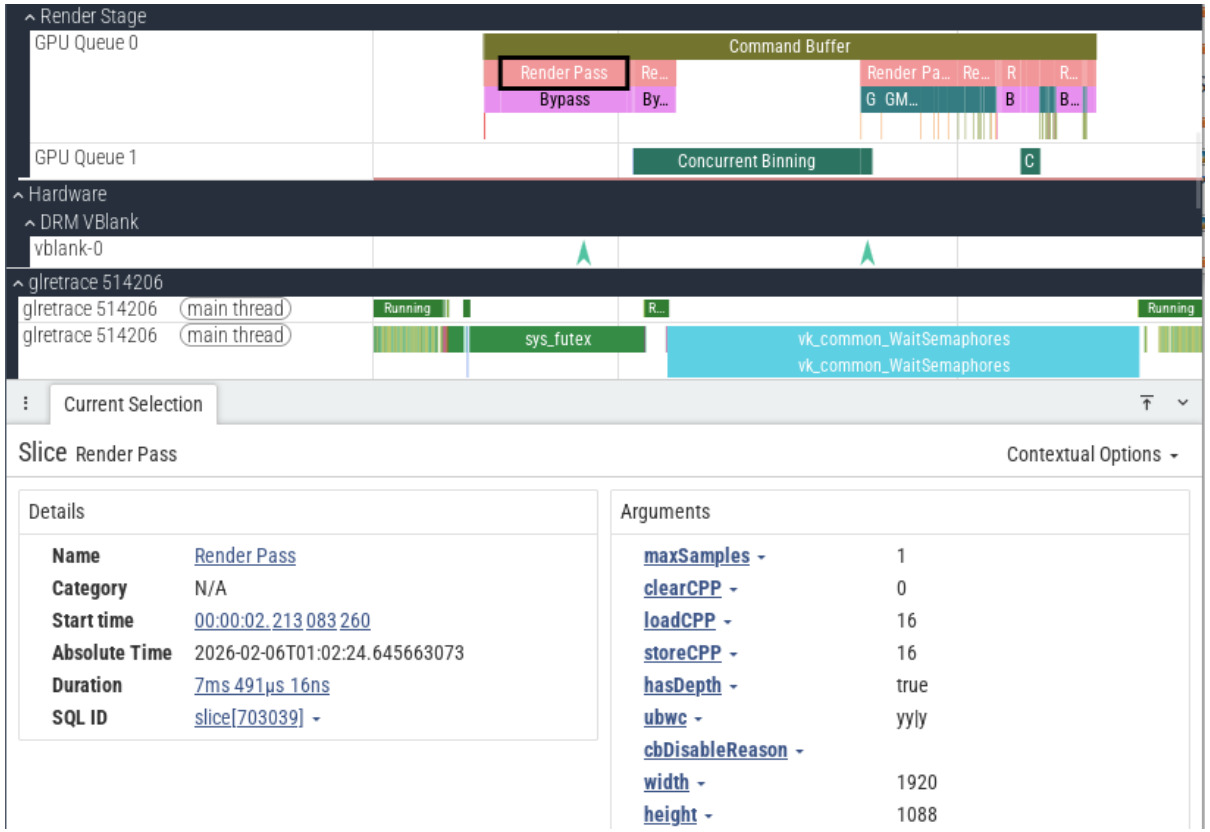
Performance debug

- VK_KHR_performance_query
  - Intel, AMD (a bit), Qualcomm, Raspberry Pi

Performance visualization



Perfetto tracing



Do your users care about Mesa?

- Steam hardware survey is 3.3% Linux, and has been growing year-over-year.
- High linux usage in workstations.
- Some android apps are shipping with Mesa.

Resources

- debug builds
- perfetto
- GPU hang debugging

Questions

Join us!

https://www.igalia.com/jobs