Mesh shading implementation in Mesa

Implementing VK_EXT_mesh_shader in RADV

Timur Kristóf

2022

VALVE Linux Open-Source Graphics Drivers Group

XDC 2022

Table of Contents

1. Mesh shading recap

2. Mesa mesh shading implementation

3. Demo

Mesh shading recap

Mesh shading

The good

New programming model that enables efficient geometry processing for highly detailed scenes.

The bad

May be difficult to integrate and achieve better perf than the traditional pipeline.

The ugly

API is very low-level and vendor-specific tweaks are necessary for optimimum performance.

Mesh shading programming model

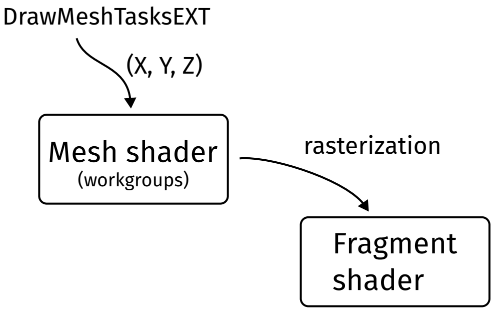
- Compute-like
- Creates vertices and primitives
- Eliminates fixed-function bottlenecks (IA, tess.)
- Very low level

Mesh shading programming model

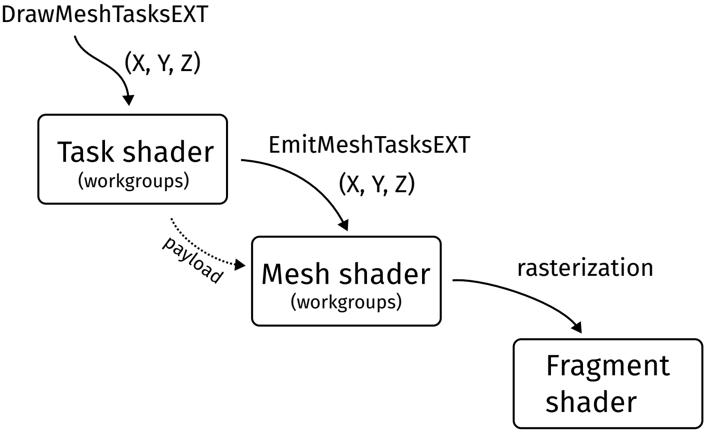
- Not (yet?) suitable for tiling GPUs

Overview of a mesh shading pipeline

Mesh shading pipeline (not recommended)



Mesh shading pipeline



DrawMeshTasksEXT

(X, Y, Z)

Task shader
(workgroups)

EmitMeshTasksEXT

(X, Y, Z)

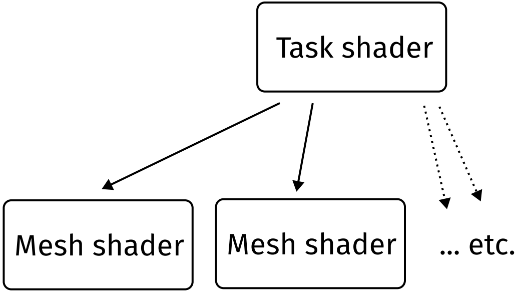
payload

Mesh shader
(workgroups)

rasterization

Fragment shader

Mesh shading pipeline



per-meshlet processing

1 dispatch ~ 1 mesh (all meshlets)
1 workgroup ~ group of meshlets
1 invocation ~ 1 meshlet (typical)

per-vertex/per-primitive processing

1 dispatch ~ group of meshlets
1 workgroup ~ 1 meshlet
1 invocation ~ 1/2 vertices/primitives

New shader stages

Task shader

How many mesh shader workgroups do you need?
Optional "payload" output.

Mesh shader

Uses a compute-like programming model to feed the rasterizer directly.

Typical uses of mesh shading

Meshlets

During asset building, split your geometry into a smaller cluster of primitives: "meshlets".

Procedural geometry

Generate geometry on the fly according to a mathematical formula without loading any data from memory.

What can you do in a task shader?

- Coarse per-meshlet culling
- LOD selection
- Geometry amplification
- Replacement for compute pre-pass

What else can you do in a mesh shader?

- Per-triangle culling
- Procedural generation of vertices and primitives

Mesa mesh shading implementation

In the beginning... (September 2021)

- NV_mesh_shader (no EXT)
- No test cases (no CTS)
- No users/apps (just an NV sample)

RADV mesh shading progress

- Oct-Dec 2021:
  NV_mesh_shader mesh-only pipelines + VRS
- Mar-Jun 2022:
  Task shaders
- Aug 2022:
  EXT_mesh_shader

Mesh shaders HW vs. programming model

VALVE

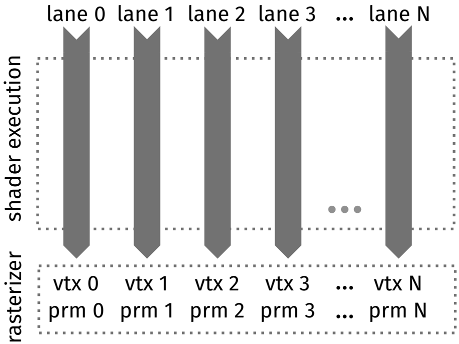
Where are outputs stored?

- NVidia: shared memory
- Intel: URB memory
- AMD: export space

AMD ”NGG” HW limitations

- 1 SIMD lane: up to 1 vertex + 1 primitive
- Up to 32K shared memory per workgroup
- 1D workgroup ID, etc.

AMD ”NGG” flow



lane 0 lane 1 lane 2 lane 3 ... lane N

shader execution

rasterizer

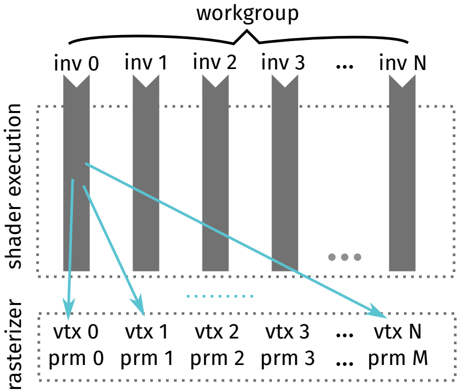
vtx 0 vtx 1 vtx 2 vtx 3 ... vtx N

prm 0 prm 1 prm 2 prm 3 ... prm N

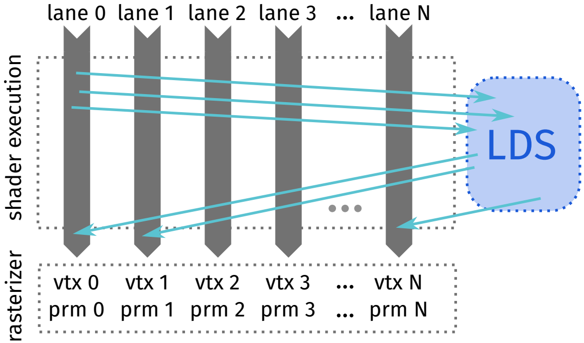
MS programming model requirements

- Any invocation can write any vertex/primitive
- Up to 48K shared memory per workgroup
- 3D workgroup ID, etc.

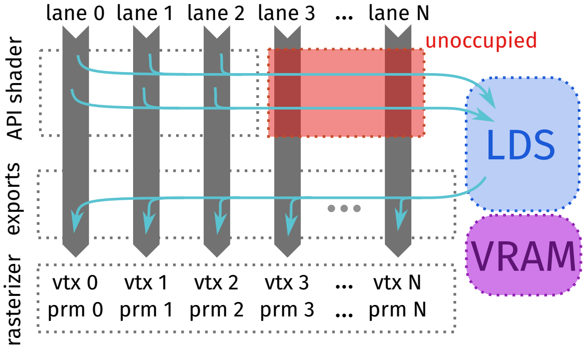
MS programming model requirements



We can implement that using LDS (+ VRAM)



Workgroup size vs. meshlet size



Task shader implementation

Task shader requirements

- Dispatch mesh shader workgroups
- Optional payload output up to 16K
- Task/Mesh should run in parallel

Task shader implementation ideas

- Abuse the tessellator
- Use a compute pre-pass

Task shader implementation ideas

- Abuse the tessellator (fixed func bottleneck)
- Use a compute pre-pass (extra barrier)

Task shader implementation ideas

- Abuse the tessellator (Intel, NVidia)
- Use a compute pre-pass
- Do it in firmware (AMD)

Task + mesh implementation on AMD

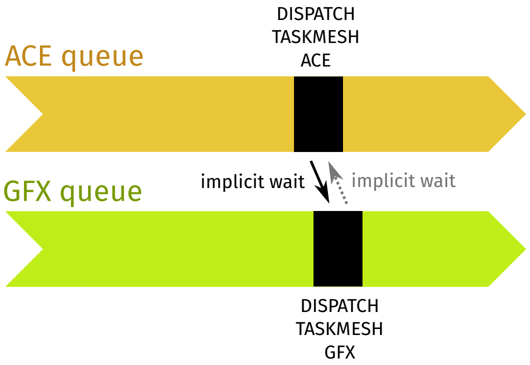
Task shaders are graphics shaders, but the HW needs them to be executed on a different HW queue.

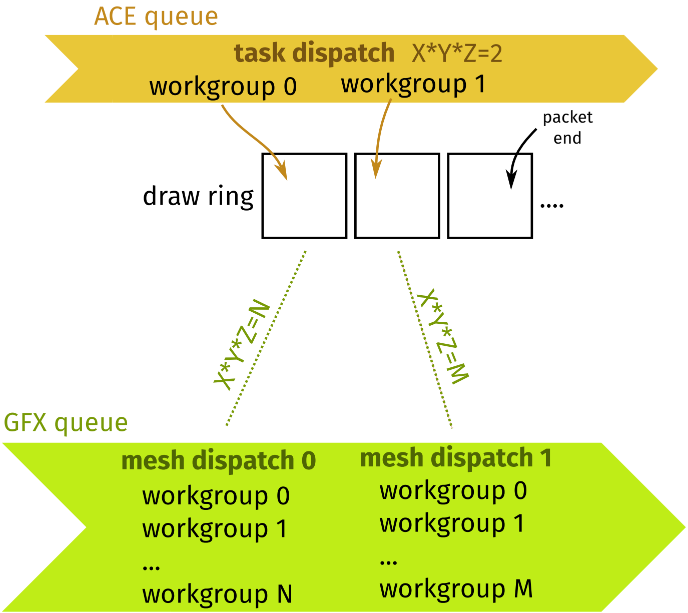
- Mesh shaders: GFX queue (graphics / general)
- Task shaders: ACE queue (async compute)

Task + mesh implementation on AMD

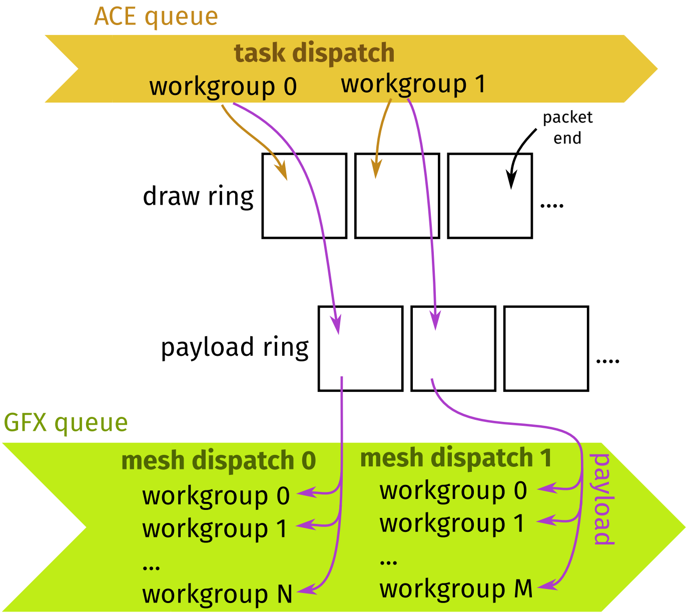
The difficulty is to pretend to the application that mesh/task are on the same queue.

Task + mesh implementation on AMD



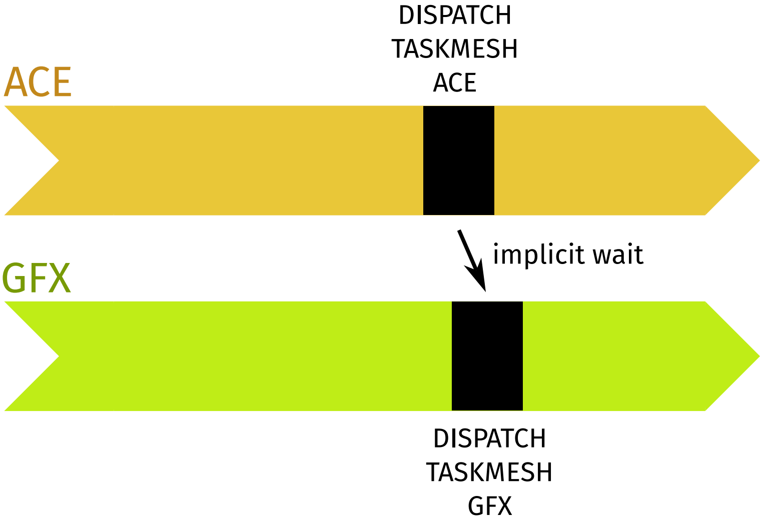

ACE queue
task dispatch X*Y*Z=2
workgroup 0
workgroup 1
packet end
draw ring
....
X*Y*Z=N
X*Y*Z=M
GFX queue
mesh dispatch 0
workgroup 0
workgroup 1
...
workgroup N
mesh dispatch 1
workgroup 0
workgroup 1
...
workgroup M

VALVE

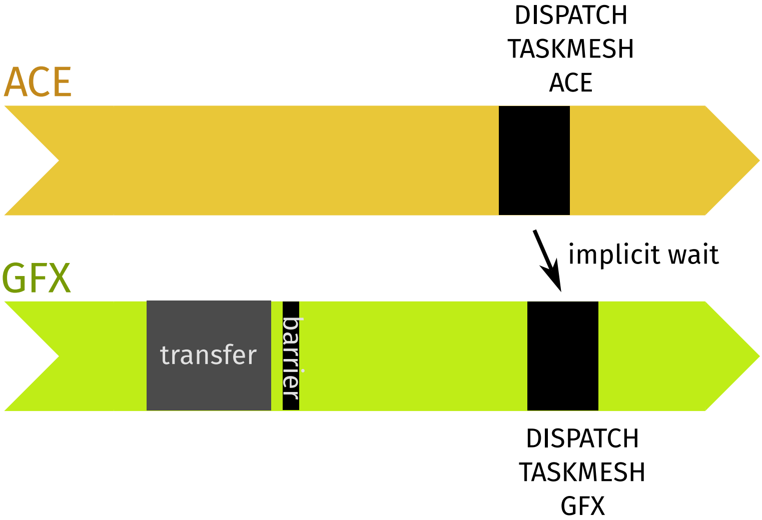

ACE queue
task dispatch
workgroup 0
workgroup 1
packet end
draw ring
....
payload ring
....
GFX queue
mesh dispatch 0
workgroup 0
workgroup 1
...
workgroup N
mesh dispatch 1
workgroup 0
workgroup 1
...
workgroup M
payload

Synchronizing the two queues

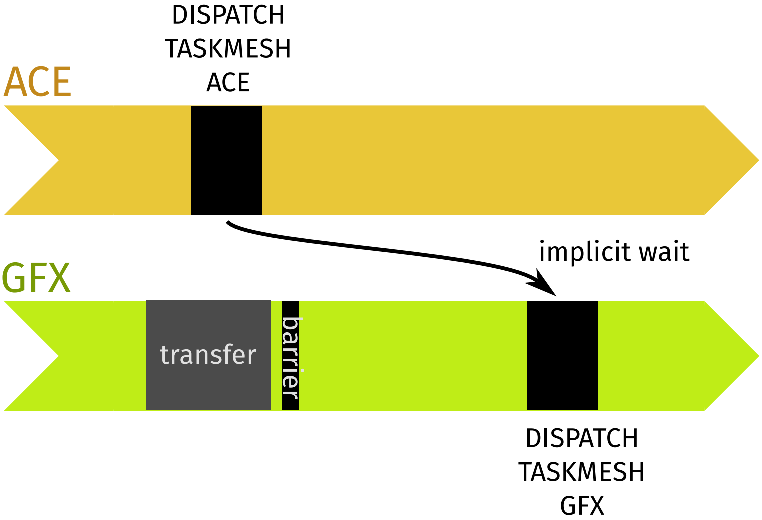
ACE+GFX without synchronization



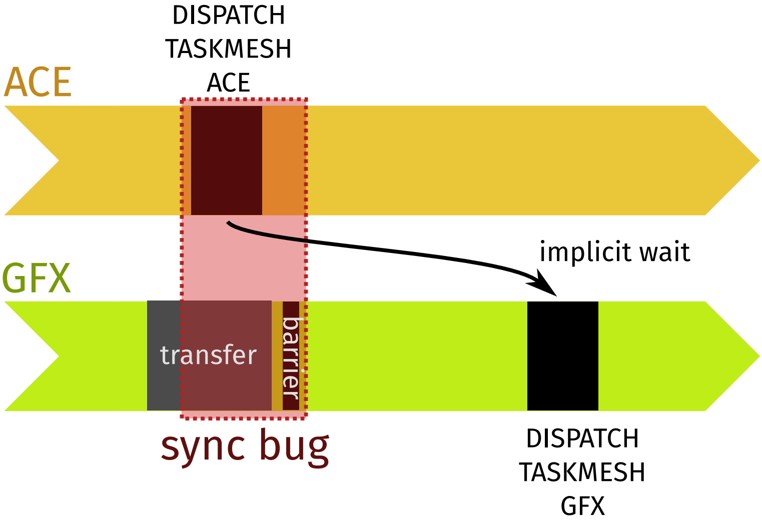
What happens if you have a barrier?



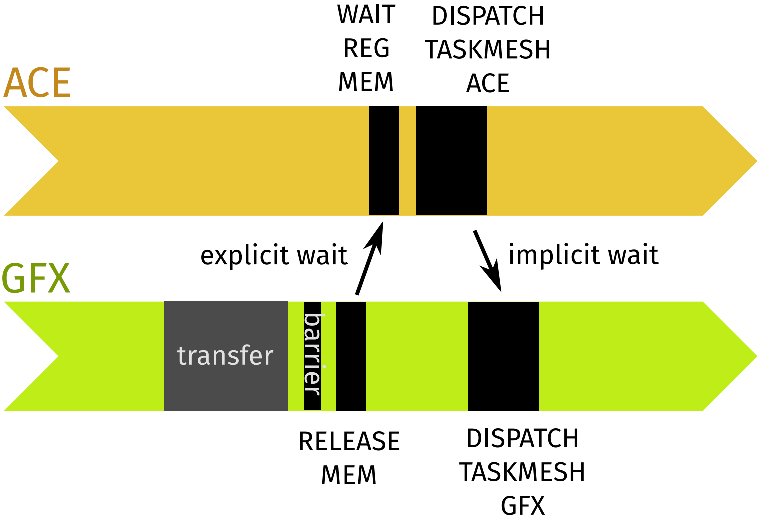
What happens if you have a barrier?



What happens if you have a barrier?

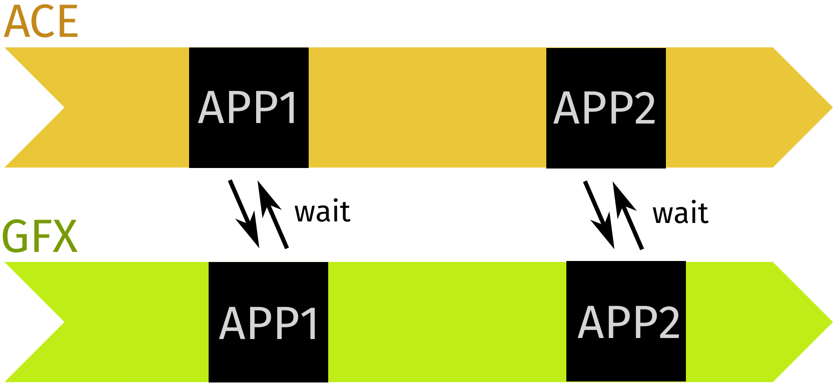


Solving barriers with task shaders

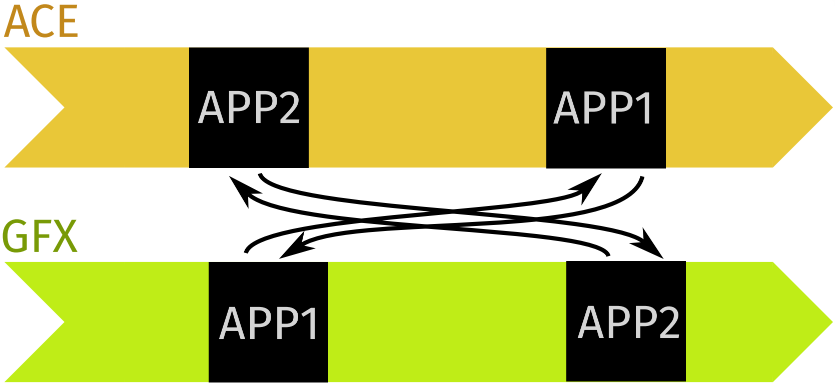


Multiple processes with task shaders

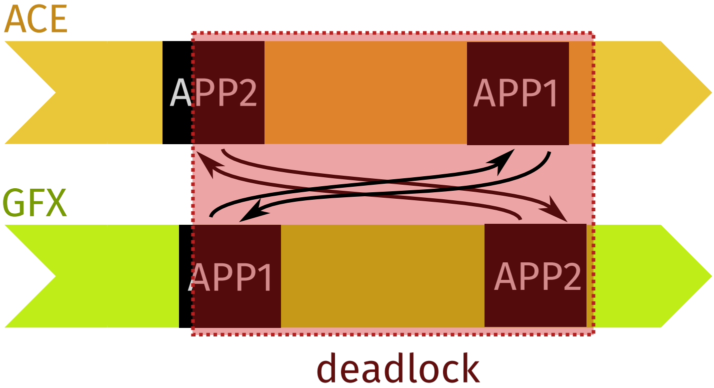
Optimal case with multiple processes



But the kernel doesn’t guarantee the ordering...



But the kernel doesn’t guarantee the ordering...



But the kernel doesn’t guarantee the ordering...

Solution: "gang submit"

- Submit to multiple queues at the same time
- Kernel schedules the jobs together
- No mixup between different apps

But the kernel doesn’t guarantee the ordering...

Solution: "gang submit"

- Not yet available in a released kernel
- Until then, `RADV_PERFTEST=ext_ms` (implemented with scheduled dependencies)

Where is the code?

Where is the code?

NIR lowering passes (backend specific)

- `ac_nir_lower_ngg`
- `nir_lower_task_shader`
- `ac_nir_lower_taskmesh_io_to_mem`

Where is the code?

RADV code

- `radv_pipeline`
- `radv_cmd_buffer`
- Major refactor in the submission code

Demo

Mesh shading demo

NVidia CAD scene demo
The scene contains nine cars, but the camera focuses on a single one, most others are fully outside frustum. The total scene has 32 M triangles and 16 K drawcalls.

Thanks

Questions, suggestions, discussion?

Mesh shading implementation in Mesa
Timur Kristóf

Venemo @ #dri-devel, #radeon, ...
https://timur.hu

https://github.com/Venemo/xdc2022_mesa_mesh_shading

Mesh shading implementation in Mesa

Implementing VK_EXT_mesh_shader in RADV

Timur Kristóf

2022

VALVE Linux Open-Source Graphics Drivers Group

XDC 2022