arm

Web 进化论 – 2024年度大会

Vulkan synchronization for WebGPU

Nathan Li, Arm
May 28th, 2024

WebGPU

- Successor to WebGL
- API for accessing the GPU ... on the web!

arm

Differences to WebGL

+ No global state
+ Command buffers
+ Pipelines
+ Bind groups
+ Render/compute passes
+ Compute shaders

WebGPU

VS

WebGL™

arm

Differences to Vulkan

- Some features/extensions not available (yet)
  - VRS
  - Raytracing
  - Bindless
  - Geometry shaders
  - Etc.
- No explicit memory management
- No explicit synchronization

VS

arm

No explicit synchronization?

- So the browser is responsible
  - Rather, the WebGPU implementation in your browser

- In Chromium, that's Dawn
  - https://dawn.googlesource.com/dawn

- Implements WebGPU on top of Vulkan
  - (Also DirectX 12, Metal and GLES)

Dawn

arm

Synchronization in Dawn

+ User submits:
  - GPUCommandBuffer
  - Without synchronization

+ We need:
  - VkCommandBuffer
  - With pipeline barriers

**arm**

When recording GPUCommandEncoder

```
pass = cmd.beginRenderPass(mainPass)
pass.setPipeline(meshPipeline)
pass.setBindGroup(cameraMatricesBindGroup)
pass.setBindGroup(textureBindGroup)

// Draw table
pass.setVertexBuffer(tableVertices)
pass.setBindGroup(tableBindGroup)
pass.draw(123)

// Draw chair
pass.setVertexBuffer(chairVertices)
pass.setBindGroup(chairBindGroup)
pass.draw(234)
```

Sync scope

Texture	Usage

Buffer	Usage

arm
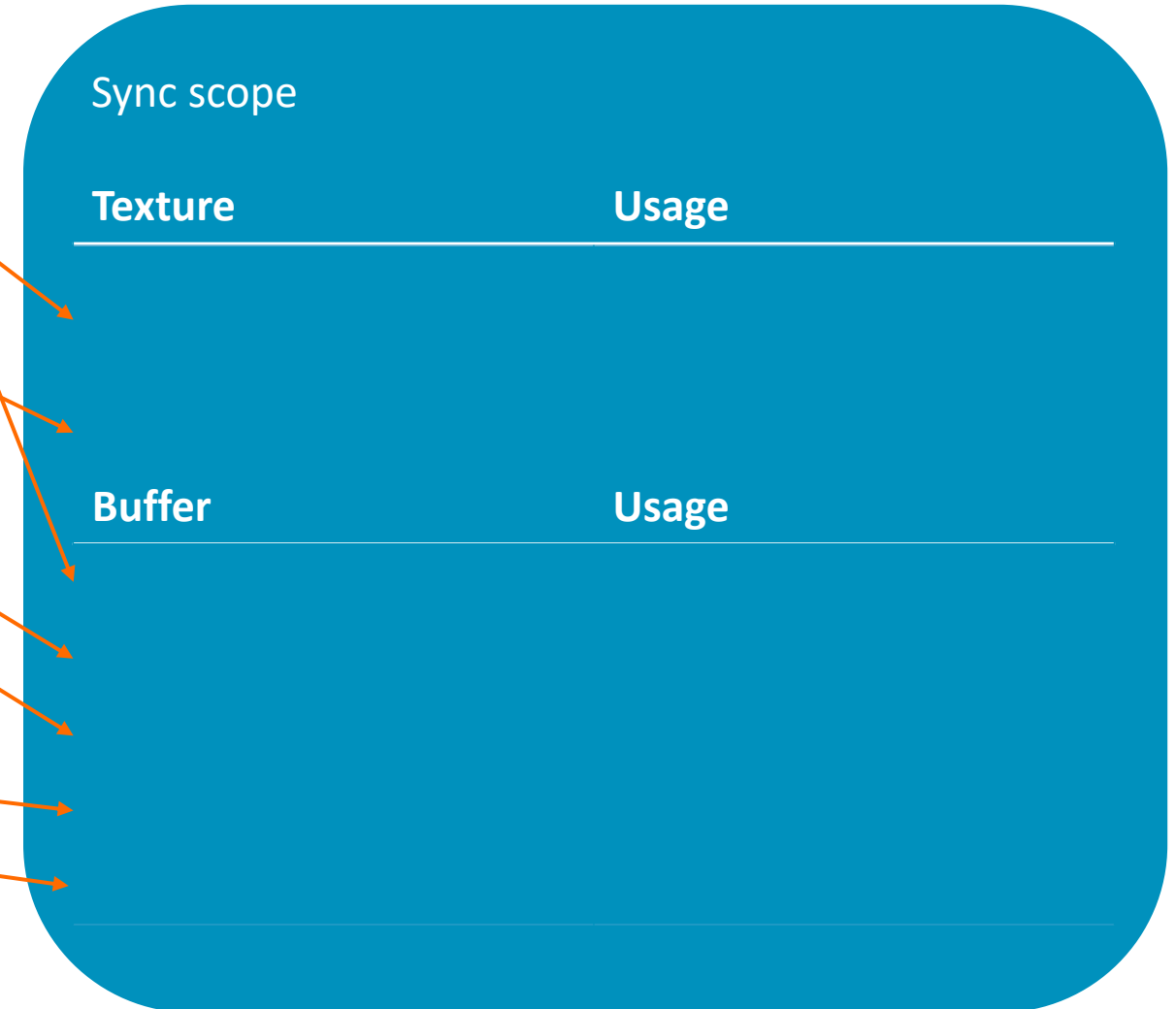
When submitting GPUCommandBuffer

```
void CommandBuffer::BeginRenderPass
  (syncScope) {
    for (auto t : syncScope.textures) {
        t.texture->RecordPipelineBarrier(
            t.usage);
    }

    for (auto b : syncScope.buffers) {
        b.buffer->RecordPipelineBarrier(
            b.usage);
    }

    vkCmdBeginRenderPass(...);
}
```

Sync scope

Texture	Usage
Framebuffer	ColorAttachment
DepthBuffer	DepthAttachment
BurntWoodTexture	TextureBinding

Buffer	Usage
CameraMatrices	UniformBuffer
TableVertices	VertexBuffer
TableUniforms	UniformBuffer
ChairVertices	VertexBuffer
ChairUniforms	UniformBuffer

arm

Recording a barrier

```
                                              Example
void Texture::RecordPipelineBarrier(
    TextureUsage usage)                       ColorAttachment
{

    VkImageMemoryBarrier barrier = {};
    [...]
    barrier.accessMask = GetAccessMask(usage);   VK_ACCESS_COLOR_ATTACHMENT_WRITE_BIT
    barrier.imageLayout = GetImageLayout(usage); VK_IMAGE_LAYOUT_COLOR_ATTACHMENT_OPTIMAL

    VkPipelineStageFlags stages =
        GetPipelineStage(usage);              VK_PIPELINE_STAGE_COLOR_ATTACHMENT_OUTPUT_BIT

    vkCmdPipelineBarrier(...);
}
```

arm

Recording a barrier

```
[...]
barrier.srcAccessMask = GetAccessMask(mLastUsage);
barrier.oldImageLayout = GetImageLayout(mLastUsage);

VkPipelineStageFlags srcStage = GetPipelineStage(mLastUsage);

mLastUsage = usage;

[...]
```

© 2024 Arm

arm

Recording a barrier

+ `GetPipelineStage(usage)`
  - The most common way to do Vulkan synchronization?

+ Used in:
  - Dawn
  - Other open source WebGPU implementation
  - Vulkan applications
  - Commercial game engines

+ Let's hope there are no issues with it...

arm

The issue with GetPipelineStage(usage)

```
void Texture::RecordPipelineBarrier(
    TextureUsage usage)
{

    VkImageMemoryBarrier barrier = {};
    [...]
    barrier.accessMask = GetAccessMask(usage);
    barrier.imageLayout = GetImageLayout(usage);


    VkPipelineStageFlags stage =
        GetPipelineStage(usage);

    vkCmdPipelineBarrier(...);
}
```
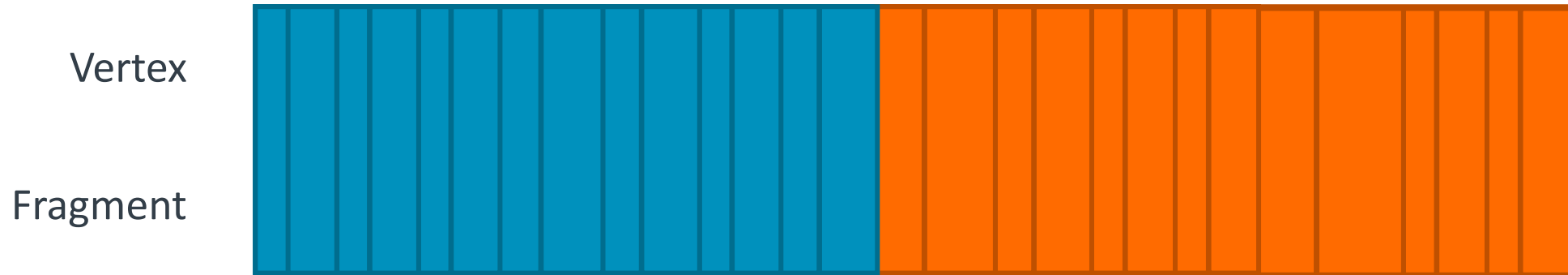
Example

TextureBinding

VK_ACCESS_SHADER_READ_BIT

VK_IMAGE_LAYOUT_SHADER_READ_ONLY_OPTIMAL

VK_PIPELINE_STAGE_FRAGMENT_SHADER_BIT |
VK_PIPELINE_STAGE_VERTEX_SHADER_BIT |
VK_PIPELINE_STAGE_COMPUTE_SHADER_BIT |
VK_PIPELINE_STAGE_GEOMETRY_SHADER_BIT |
VK_PIPELINE_STAGE_TESSELLATION_CONTROL_SHADER_BIT |
VK_PIPELINE_STAGE_TESSELLATION_EVALUATION_SHADER_BIT |
VK_PIPELINE_STAGE_TASK_SHADER_BIT_EXT |
VK_PIPELINE_STAGE_MESH_SHADER_BIT_EXT |
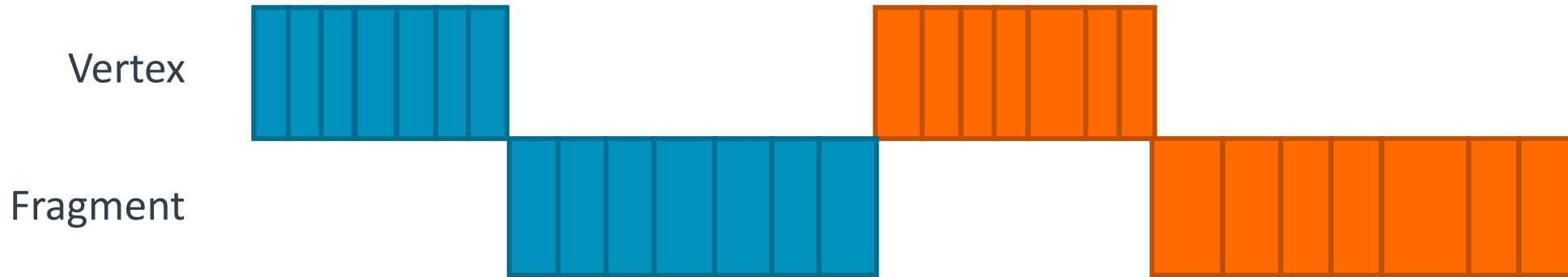VK_PIPELINE_STAGE_RAY_TRACING_SHADER_BIT_KHR

arm

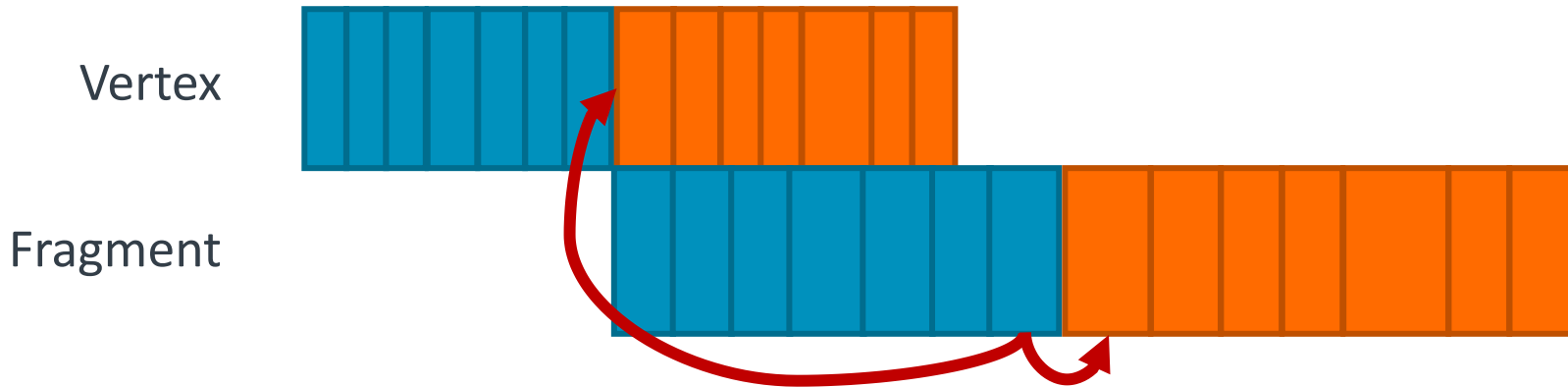# A simple frame

Vertex

Fragment

arm

... on a tiled GPU

Vertex

Fragment

arm

... on a tiled GPU

Vertex

Fragment

© 2024 Arm

arm

... on a tiled GPU

Vertex

Fragment

© 2024 Arm

arm
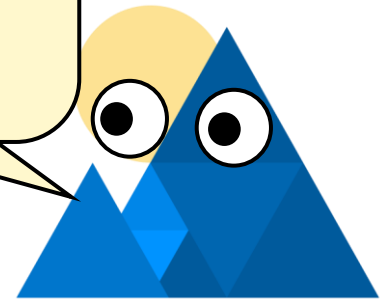
Hold it!


```
srcStages = LATE_FRAGMENT_TESTS
dstStages = FRAGMENT_SHADER | VERTEX_SHADER | COMPUTE_SHADER
```

Hold it!
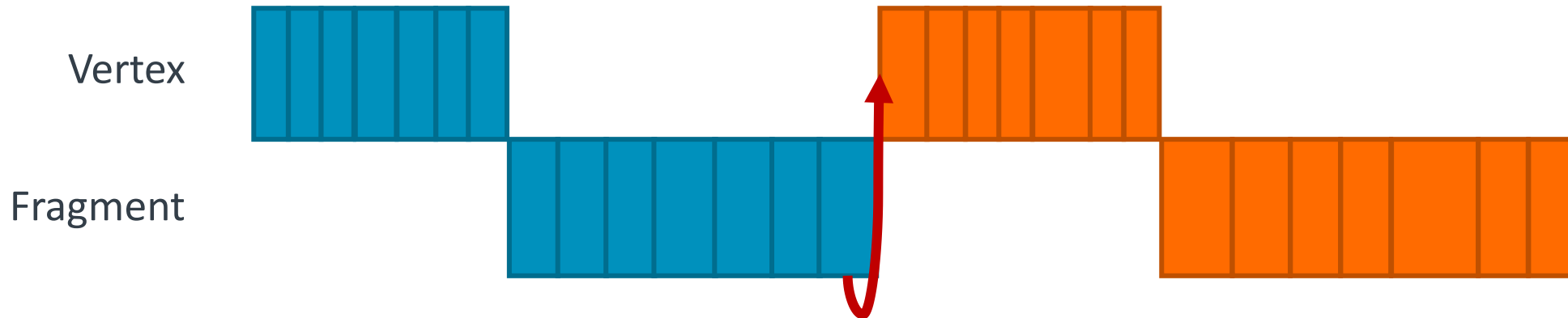

```
srcStages = LATE_FRAGMENT_TESTS
dstStages = FRAGMENT_SHADER | VERTEX_SHADER | COMPUTE_SHADER
```

arm

The problem

+ `GetPipelineStage(usage)`
  - Can't tell if TextureBinding is Vertex, Fragment or Compute

+ `GetPipelineStage(usage, shaderStages)`
  - Trivial to determine

© 2024 Arm

arm

Tracking shader stages

Texture	Usage
Framebuffer	ColorAttachment
DepthBuffer	DepthAttachment
BurntWoodTexture	TextureBinding

Buffer	Usage
CameraMatrices	UniformBuffer
TableVertices	VertexBuffer
TableUniforms	UniformBuffer
ChairVertices	VertexBuffer
ChairUniforms	UniformBuffer

Sync scope

arm

Tracking shader stages

Texture	Usage	Shader stages
Framebuffer	ColorAttachment	None
DepthBuffer	DepthAttachment	None
BurntWoodTexture	TextureBinding	Fragment
Buffer	**Usage**	**Shader stages**
CameraMatrices	UniformBuffer	Vertex
TableVertices	VertexBuffer	None
TableUniforms	UniformBuffer	Vertex
ChairVertices	VertexBuffer	None
ChairUniforms	UniformBuffer	Vertex

Sync scope

arm

How to find the shader stages?

- Explicit tagging? Shader analysis?

- User already told us

- `BindGroupLayoutEntry.visibility`
  - Mask of all shader stages the resource is used in.

© 2024 Arm

arm

Recording a barrier
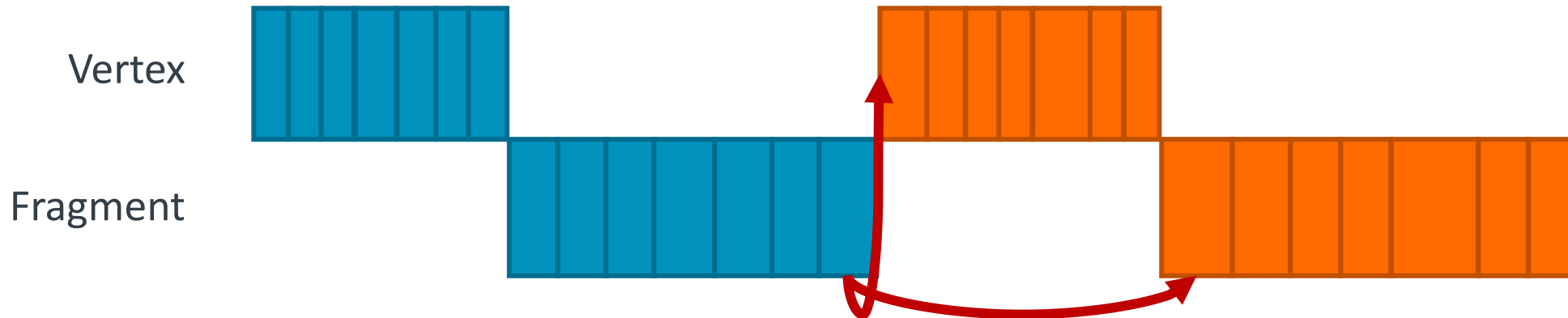
+ From
  - `GetPipelineStage(usage)`

+ To
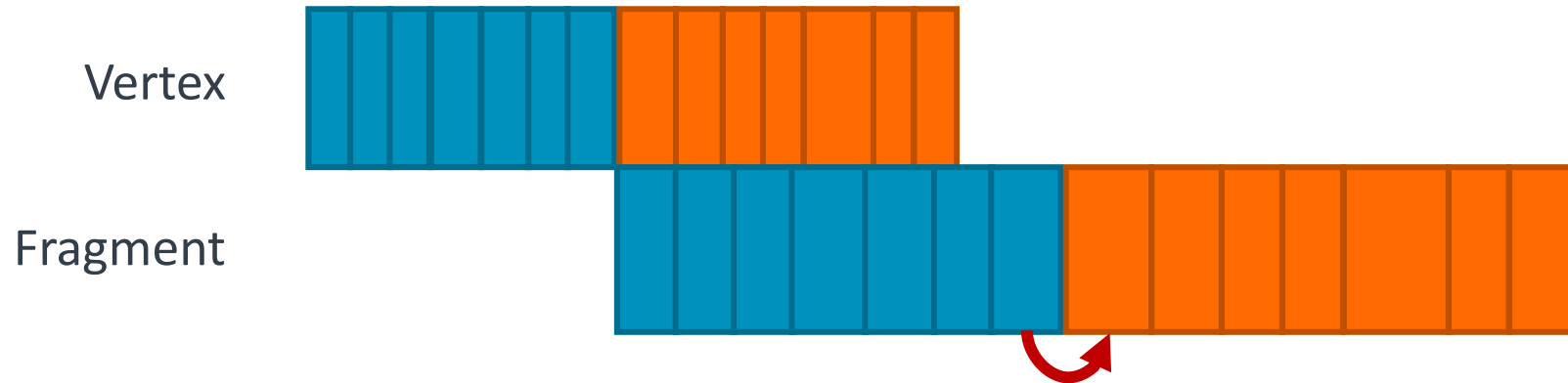 - `GetPipelineStage(usage, shaderStages)`

arm

Results


```
srcStages = LATE_FRAGMENT_TESTS
dstStages = FRAGMENT_SHADER | VERTEX_SHADER | COMPUTE_SHADER
```
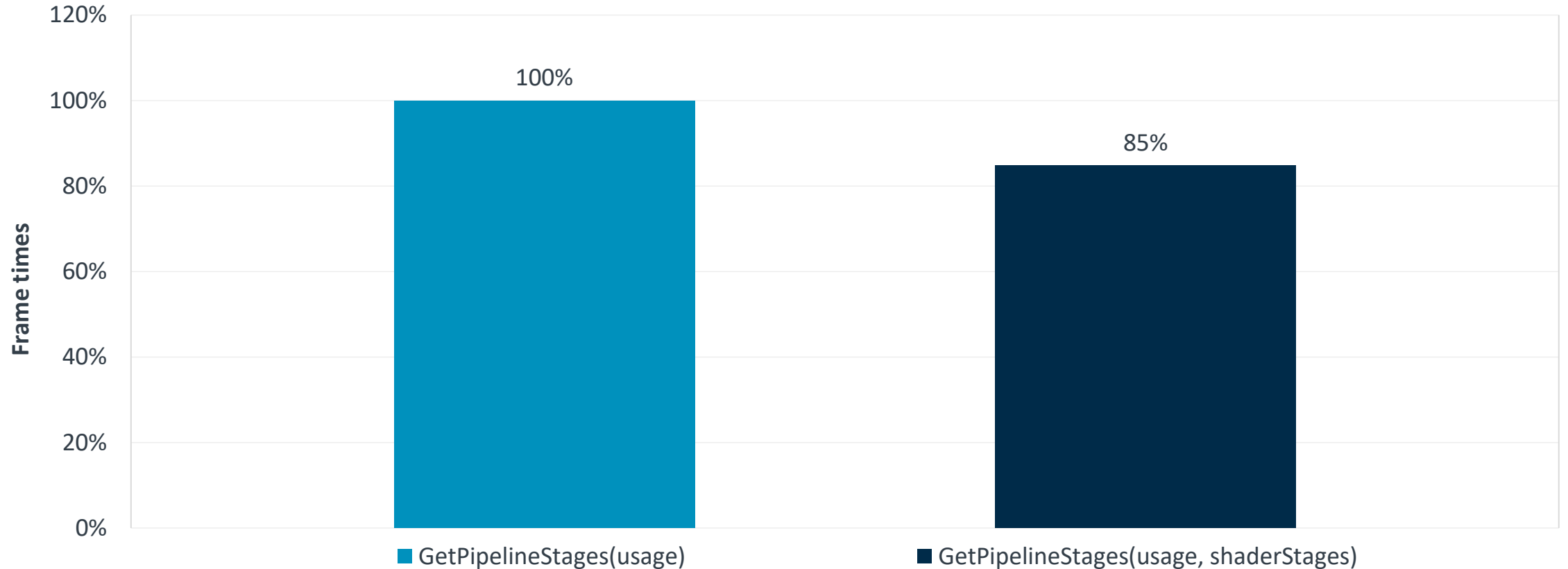
arm

Results


```
srcStages = LATE_FRAGMENT_TESTS
dstStages = FRAGMENT_SHADER
```

arm

Results

+ Measured on an Immortalis-G715 device

© 2024 Arm

arm

When submitting GPUCommandBuffer

Flashback

```
void CommandBuffer::BeginRenderPass
  (syncScope) {
    for (auto t : syncScope.textures) {
        t.texture->RecordPipelineBarrier(
            t.usage);
    }

    for (auto b : syncScope.buffers) {
        b.buffer->RecordPipelineBarrier(
            b.usage);
    }

    vkCmdBeginRenderPass(...);
}
```

Sync scope

Texture	Usage
Framebuffer	ColorAttachment
DepthBuffer	DepthAttachment
BurntWoodTexture	TextureBinding

Buffer	Usage
CameraMatrices	UniformBuffer
TableVertices	VertexBuffer
TableUniforms	UniformBuffer
ChairVertices	VertexBuffer
ChairUniforms	UniformBuffer

arm

Barrier merging

- One vkCmdPipelineBarrier per resource is a lot...
- vkCmdPipelineBarrier can contain many memory barriers
- Let's merge them into one!

arm

Barrier merging example

- First barrier: TRANSFER ⟶ VERTEX
- Second barrier: FRAGMENT ⟶ FRAGMENT
- Merged barrier: TRANSFER | FRAGMENT ⟶ VERTEX | FRAGMENT

arm

Better barrier merging

- Two vkCmdPipelineBarrier's
  - One for everything with VERTEX in its dstStageMask
  - One for everything else

- (Or synchronization2)

© 2024 Arm

arm

Conclusions

+ If you're doing:
  - `GetPipelineStage(usage)`

+ Consider:
  - `GetPipelineStage(usage, shaderStages)`

+ And save 15%

© 2024 Arm

arm

Resources

- Synchronization validation layers
  - See talk: Using Vulkan Synchronization Validation Effectively
- Vulkan Samples
  - Pipeline barriers sample
- Full details on the Dawn commit
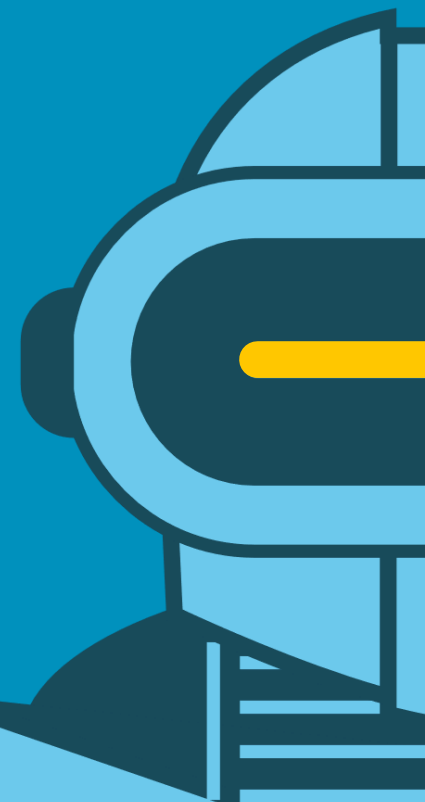 - https://dawn-review.googlesource.com/c/dawn/+/151340
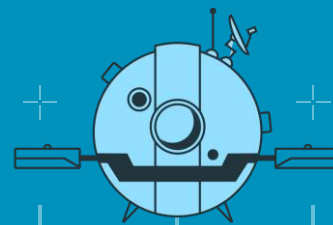
arm

arm Developer Program

Build the future on Arm

- Join our developer community
- Connect with like-minded developers
- Get fresh insights from Arm experts
- **Sign up:** arm.com/developerprogram

Find out more

- **Talk to us via Discord** ArmSoftwareDev
- **Docs & Resources:** arm.com/vulkan
- **Forums and blogs:** community.arm.com

Discord | ArmSoftwareDev

Twitter | @ArmSoftwareDev

YouTube | @ArmSoftwareDevelopers