Vulkanised 2025

The 7th Vulkan Developer Conference
Cambridge, UK | February 11-13, 2025

Vulkan Best Practices for Mobile Development

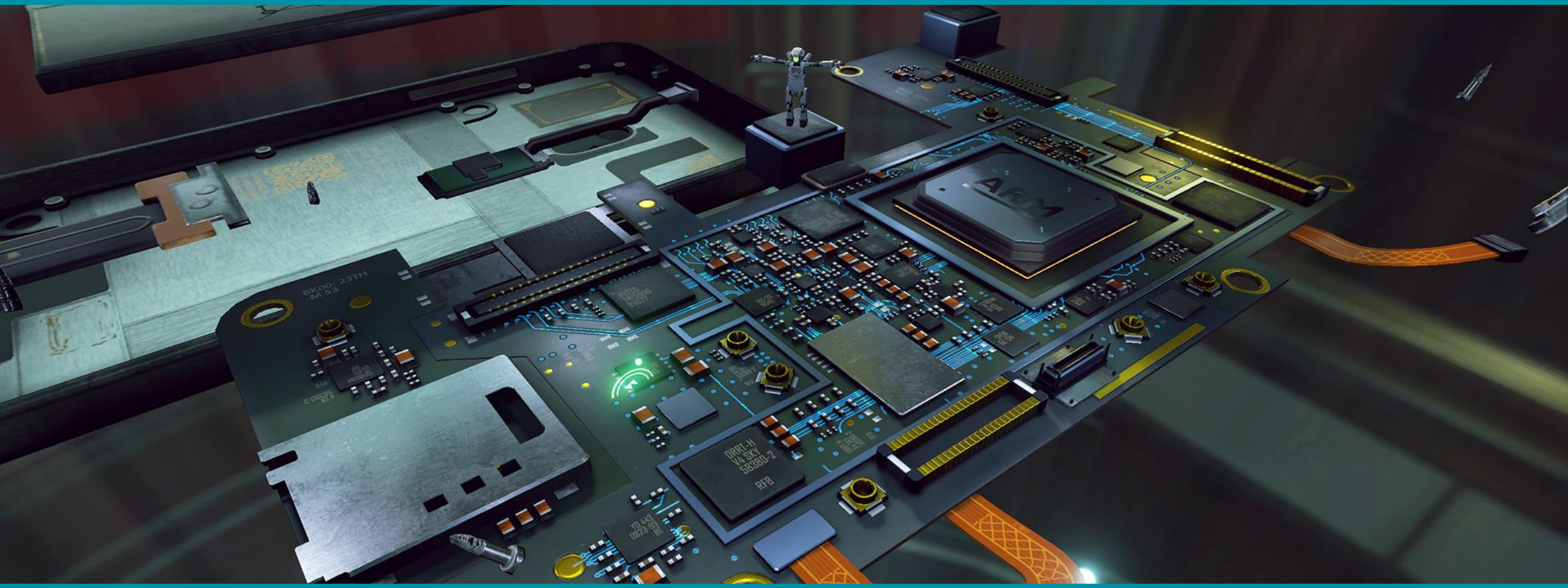
Pete Harris, Arm

Topics …

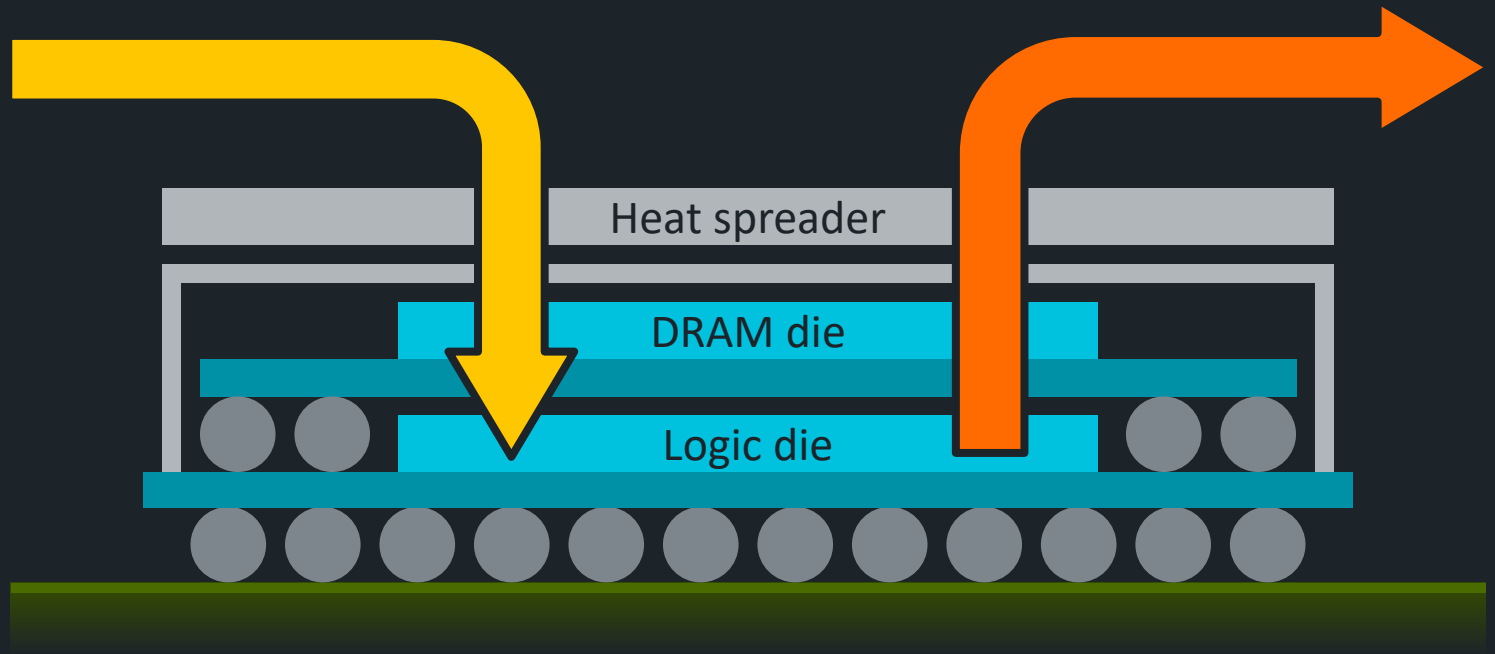
- Mobile systems
- Tile-based GPUs
- Vulkan API best practices
- Arm GPU new tech highlights

Mobile system essentials

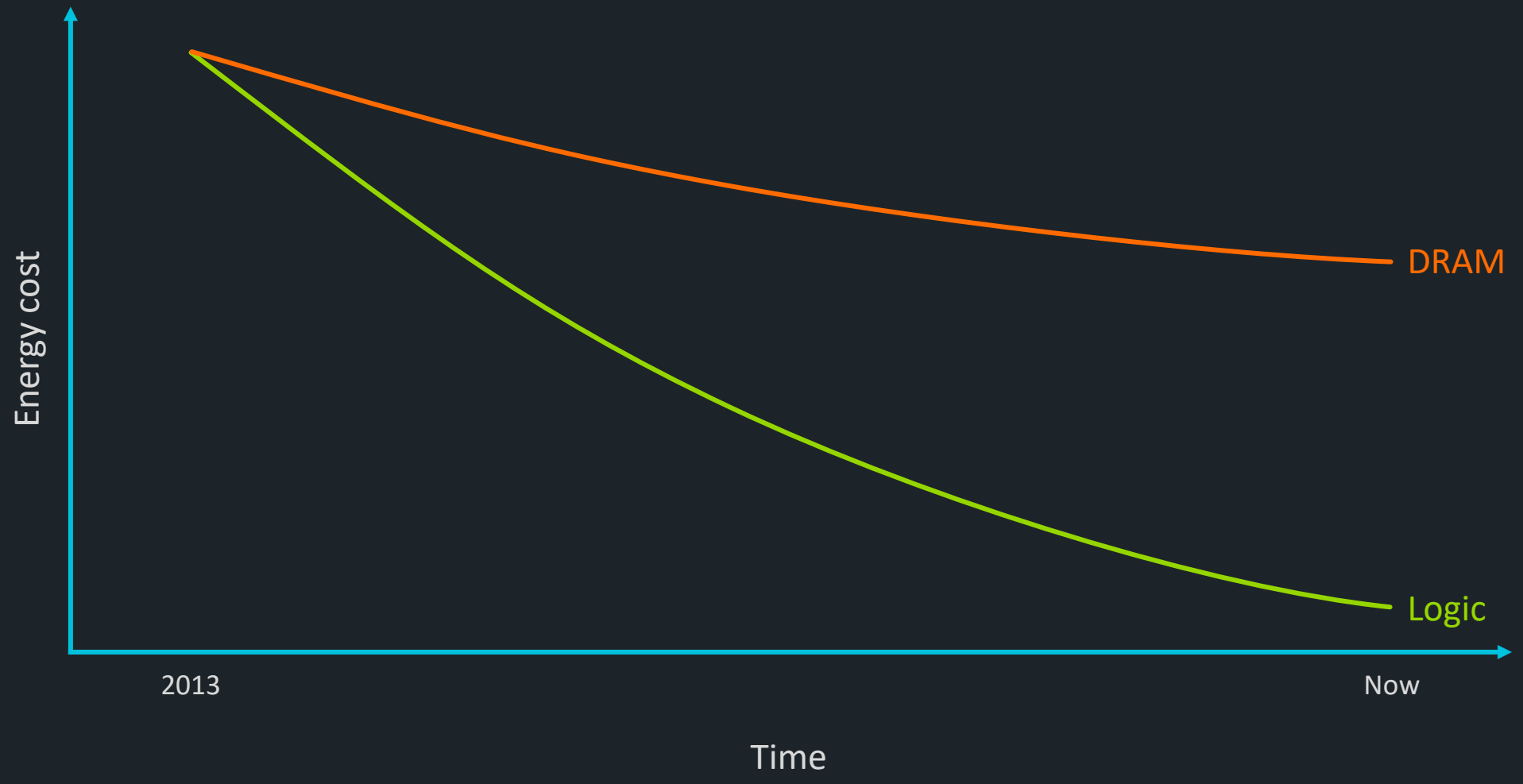
Mobile systems

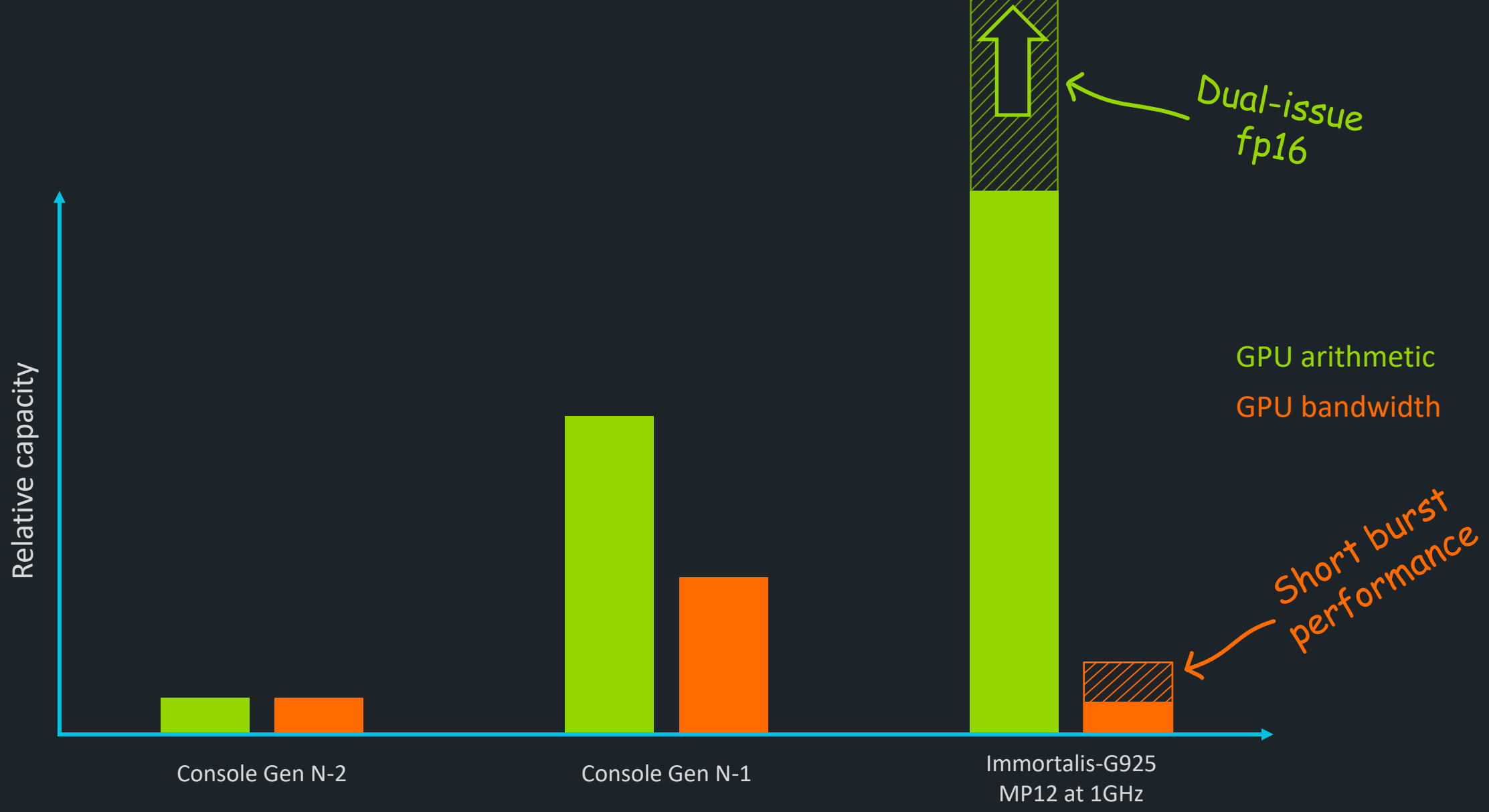


1 Watt in = 1 Watt out



3-6 Watt  budget

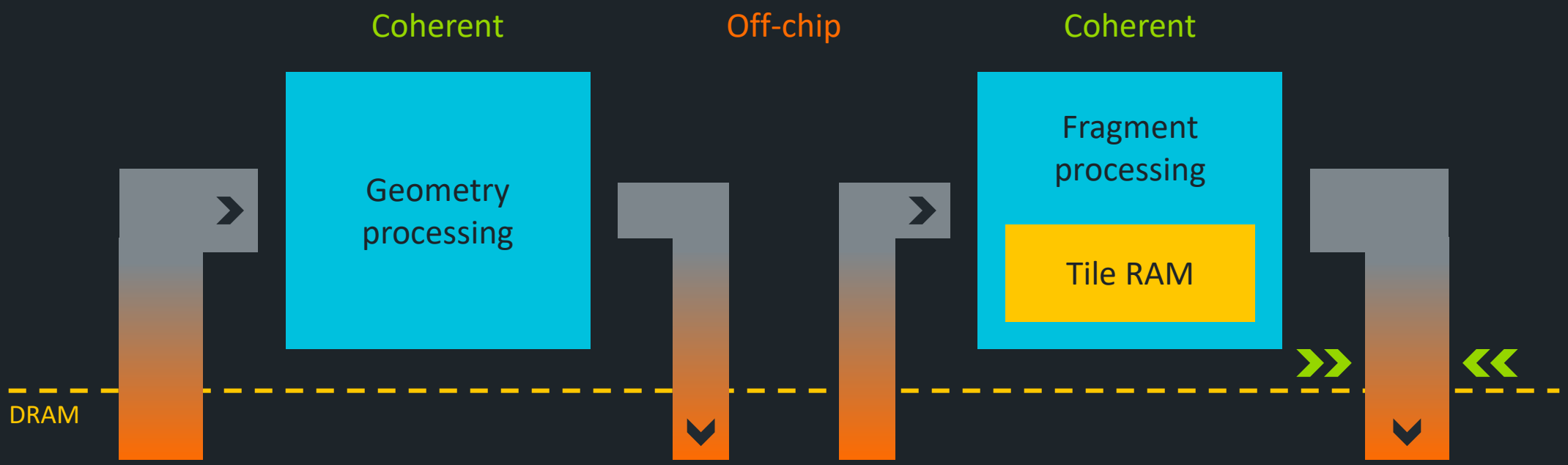



Dual-issue fp16
GPU arithmetic
GPU bandwidth
Short burst performance
Relative capacity
Console Gen N-2
Console Gen N-1
Immortalis-G925
MP12 at 1GHz

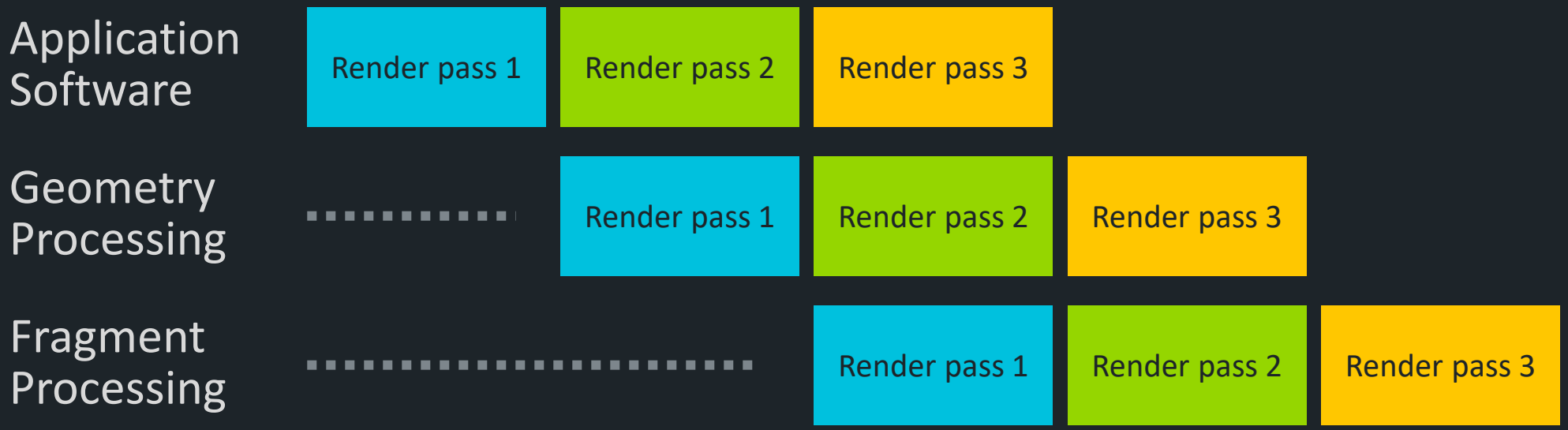
arm

Tile-based GPU essentials

Traditional tile-based rendering



Traditional tile-based scheduling

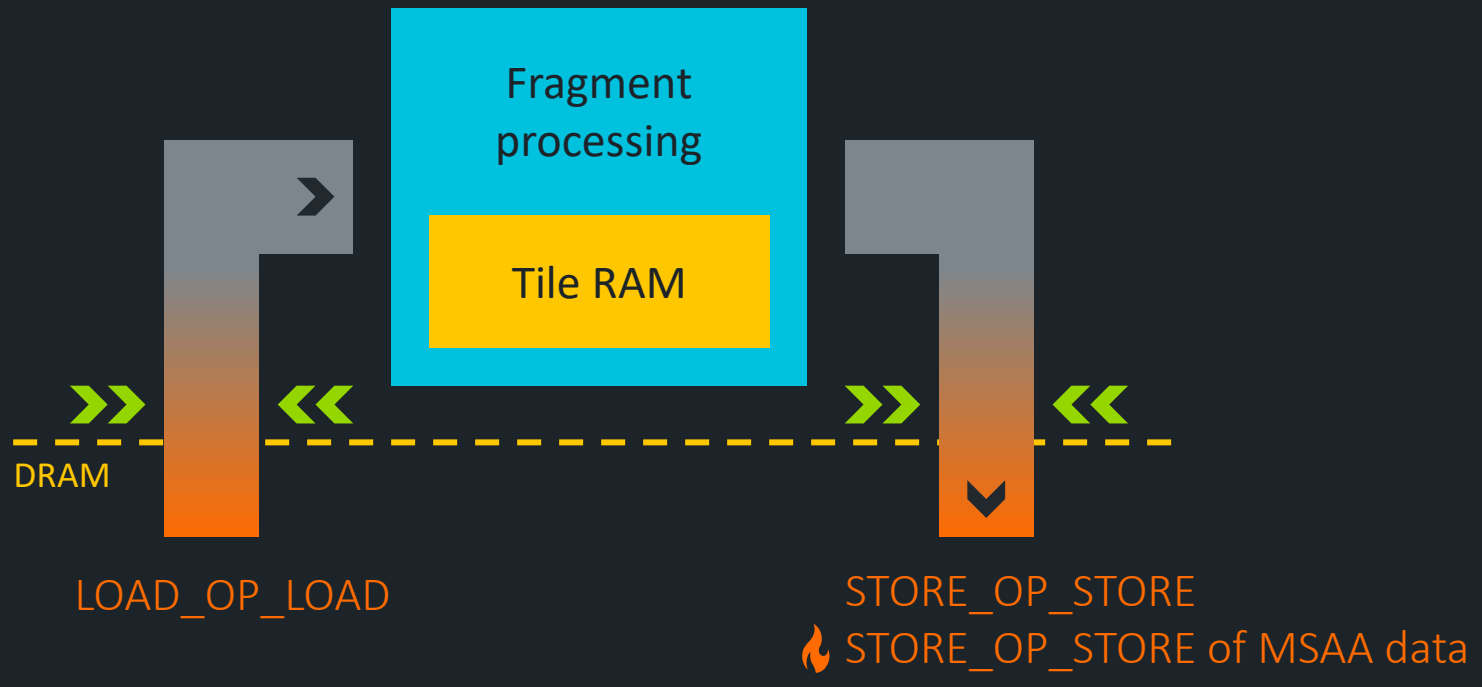


Essential API best practices

Efficient render passes

Optimize for tile memory life cycle

Goal: Minimize tile memory use of DRAM



Efficient render passes

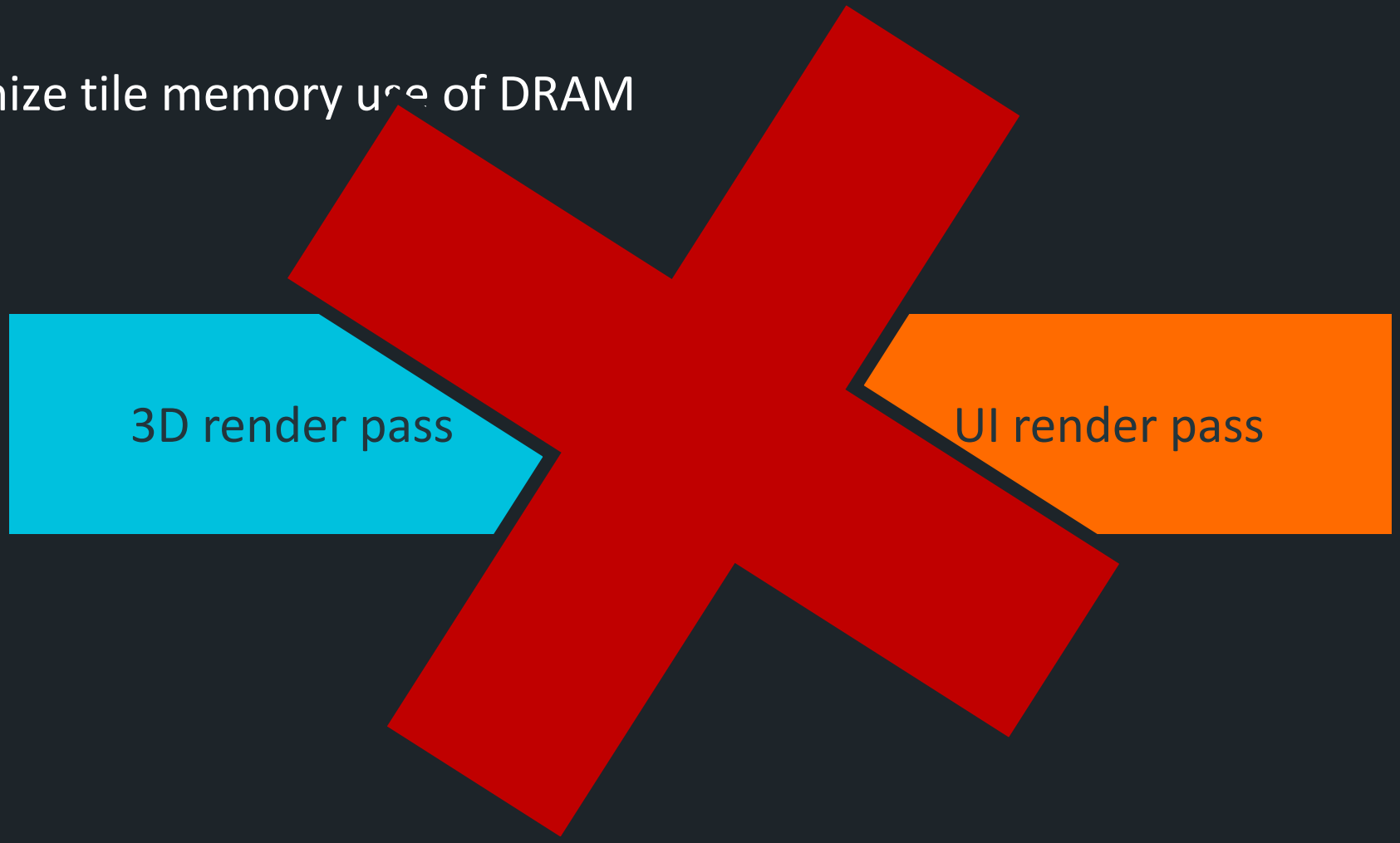
Optimize for tile memory life cycle

Goal: Minimize tile memory use of DRAM

Efficient render passes

Optimize for tile memory life cycle

Goal: Minimize tile memory use of DRAM



Efficient render passes

Optimize for tile memory life cycle

Goal: Minimize tile memory use of DRAM

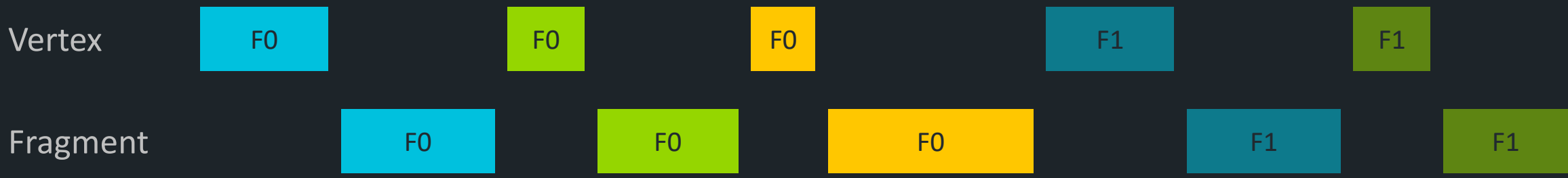
3D and UI render pass

Efficient scheduling

Optimize for render pass overlap

Bad: Pipeline barriers force render pass serialization

- srcStage = BOTTOM_OF_PIPE
- dstStage = TOP_OF_PIPE

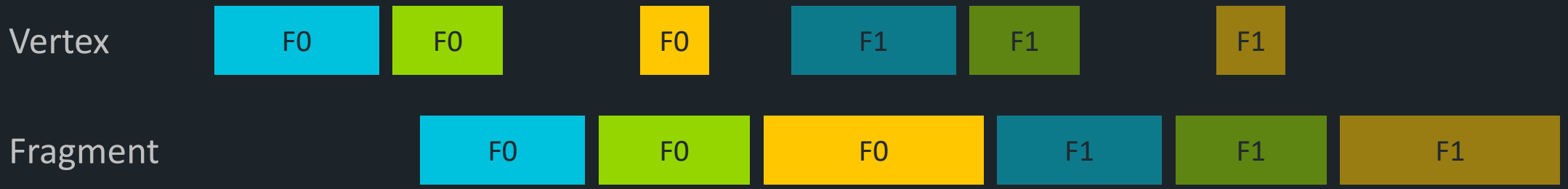


Efficient scheduling

Optimize for render pass overlap

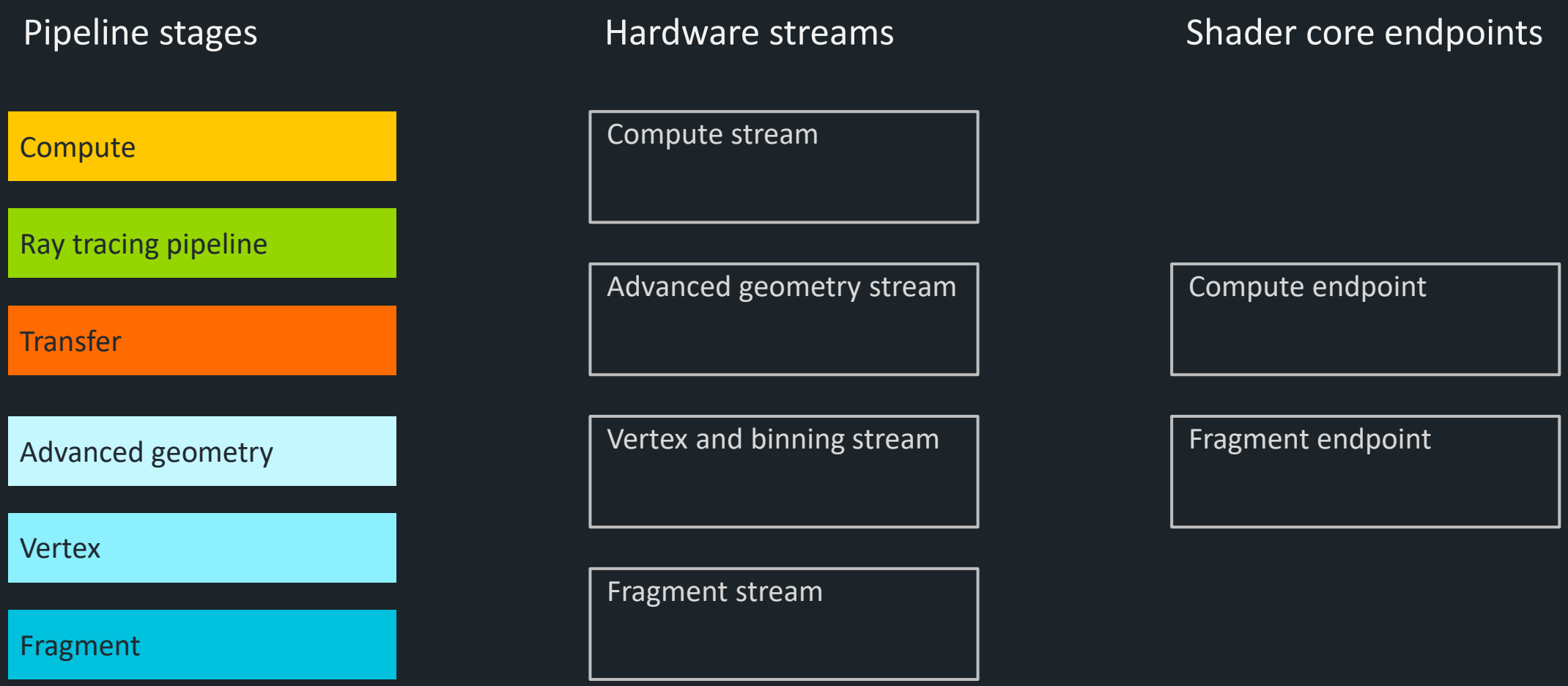
Good: Pipeline barriers allow parallel render pass processing

- srcStage = BOTTOM_OF_PIPE
- dstStage = FRAGMENT



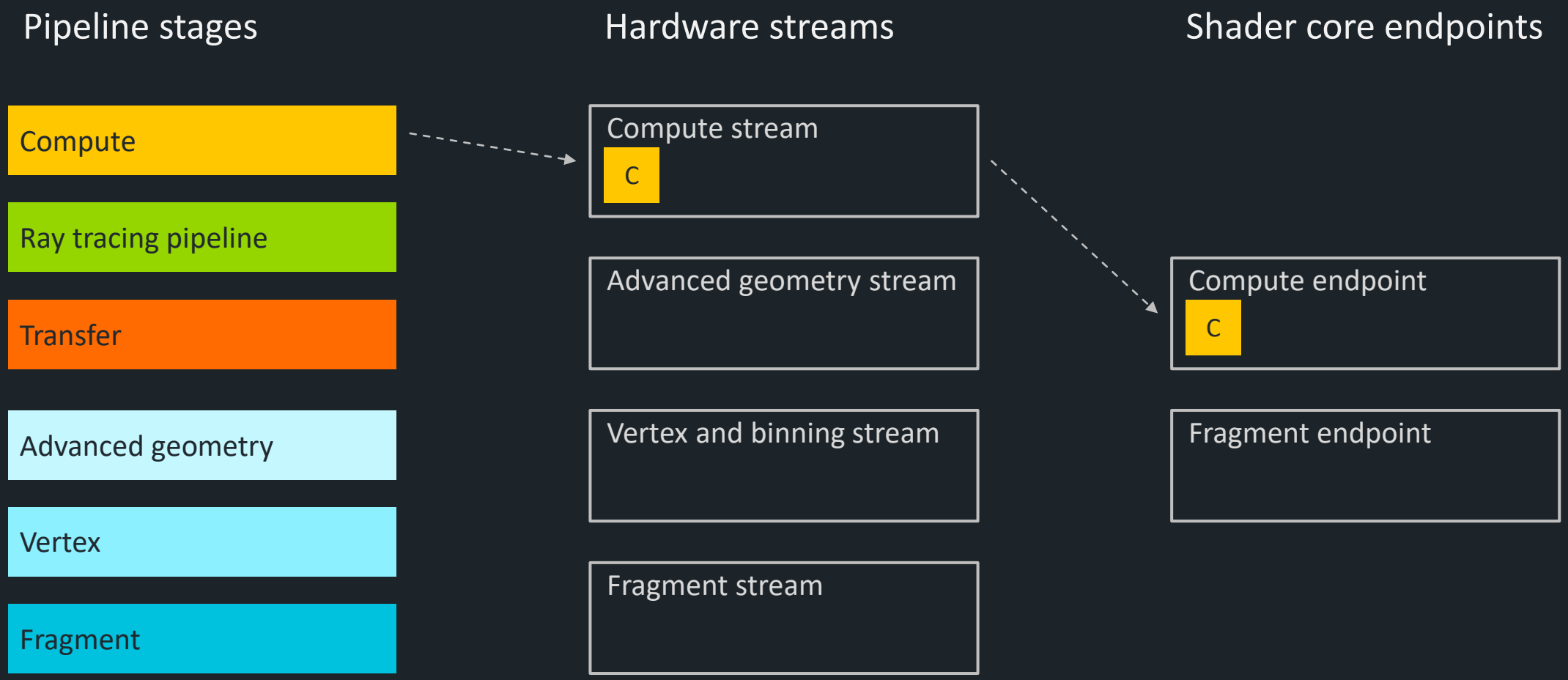
Efficient scheduling

The bigger picture …



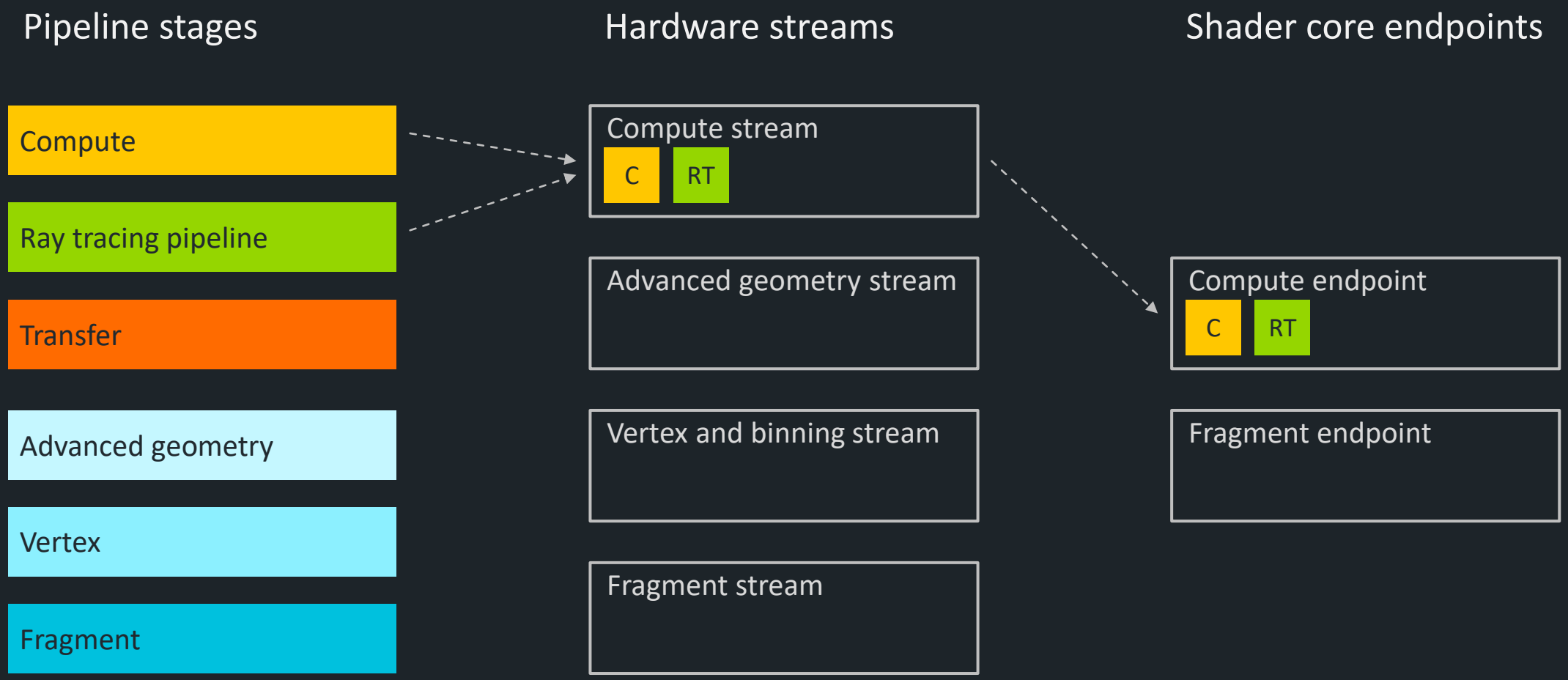
Efficient scheduling

The bigger picture …



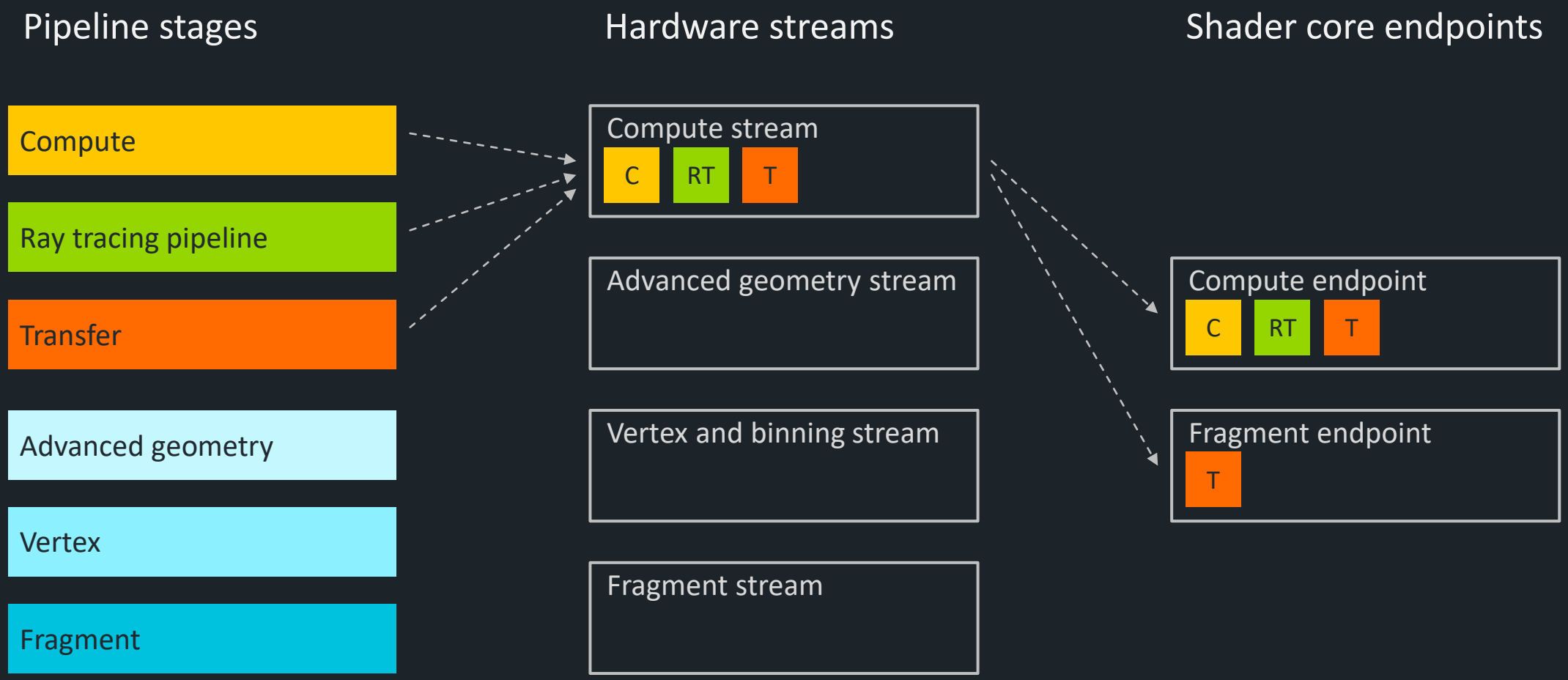
Efficient scheduling

The bigger picture …



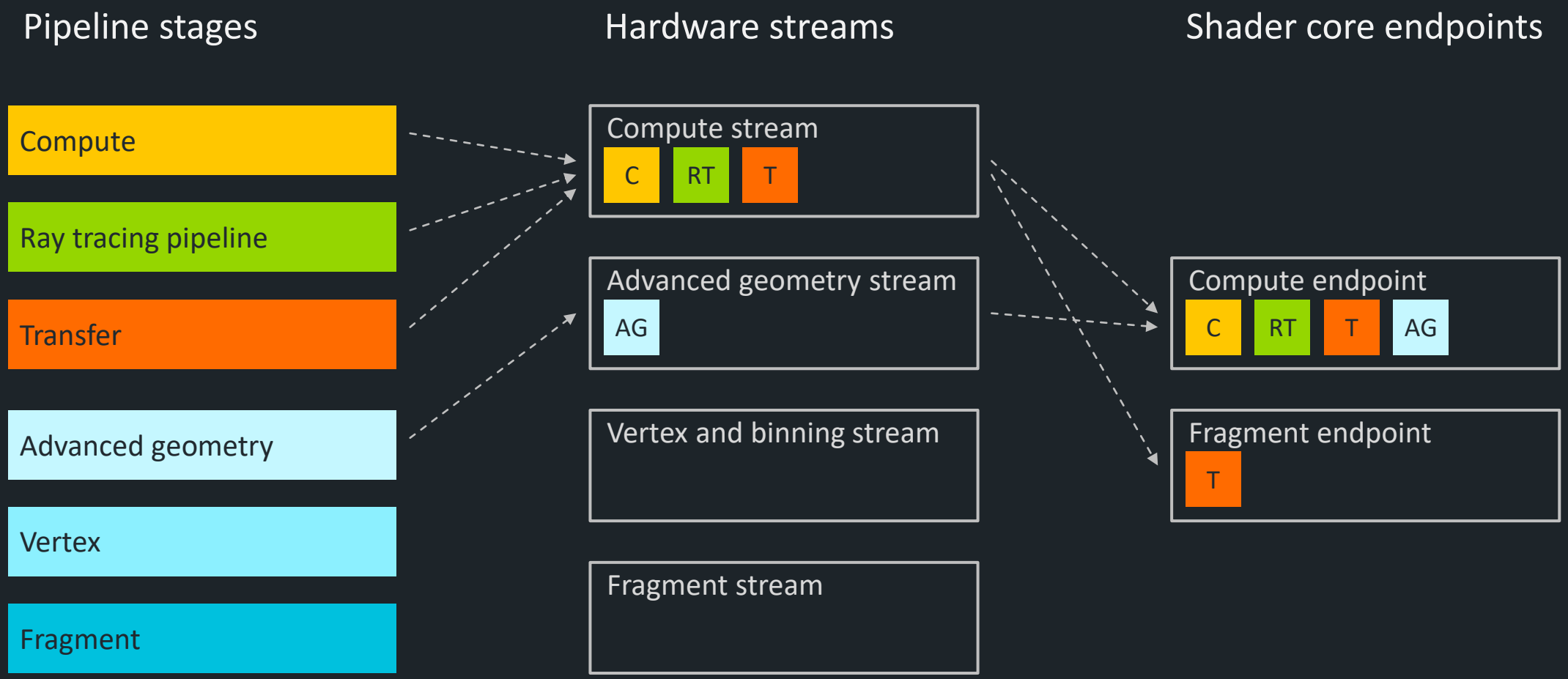
Efficient scheduling

The bigger picture …



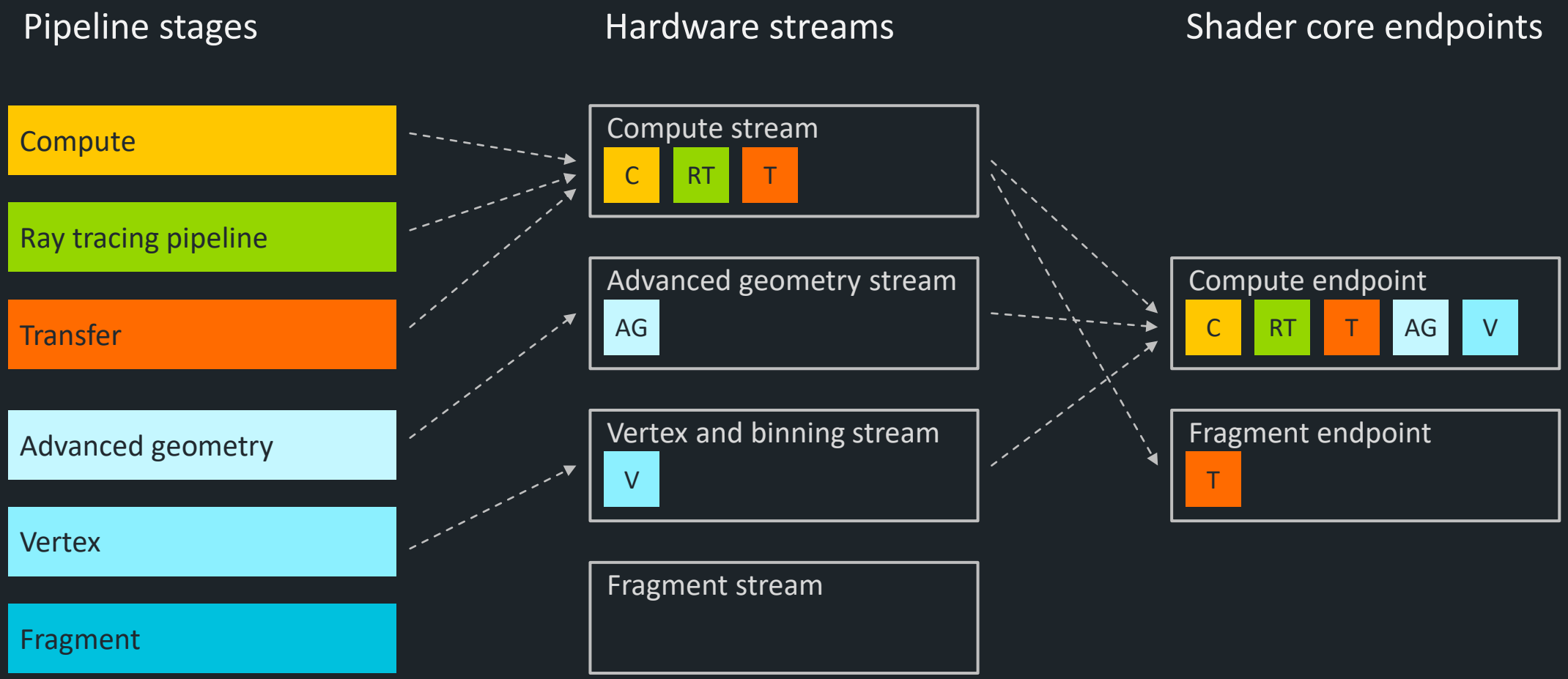
Efficient scheduling

The bigger picture …



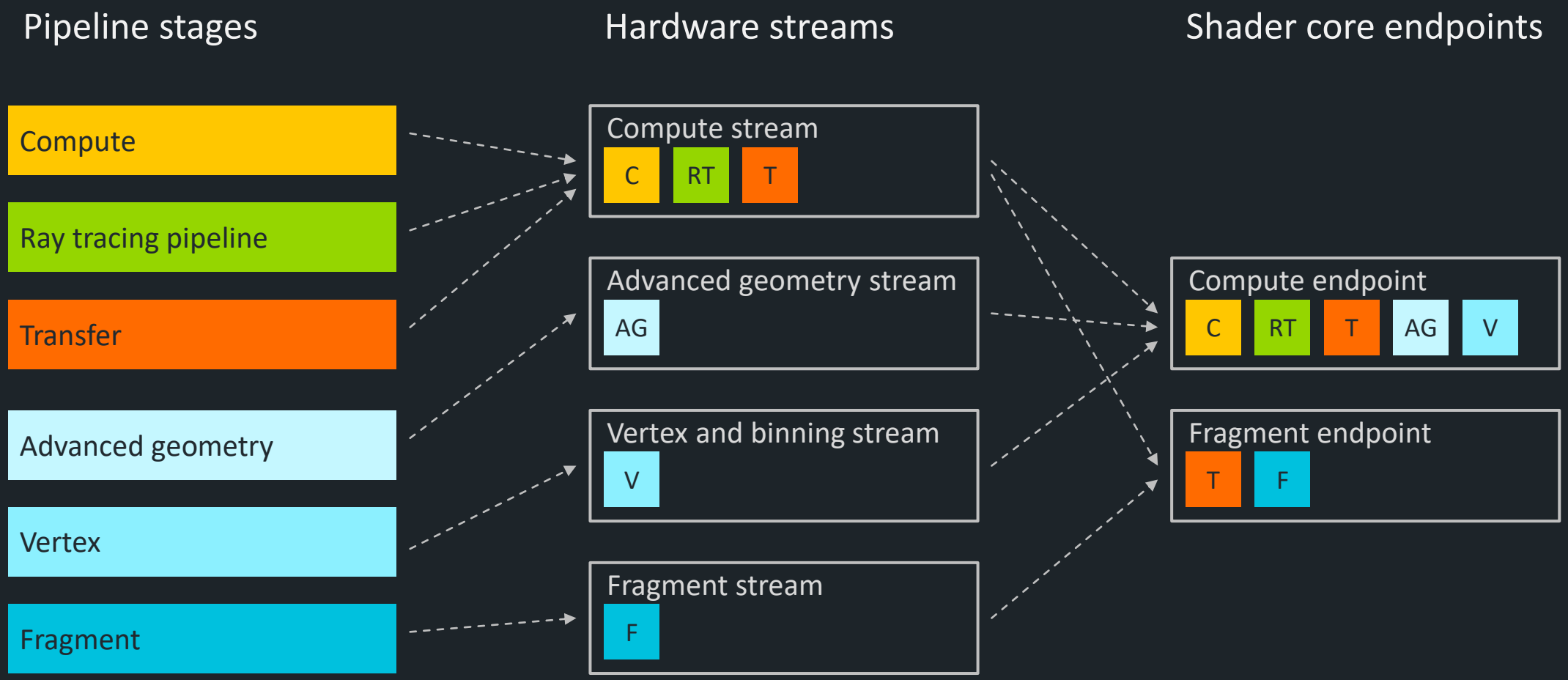
Efficient scheduling

The bigger picture …



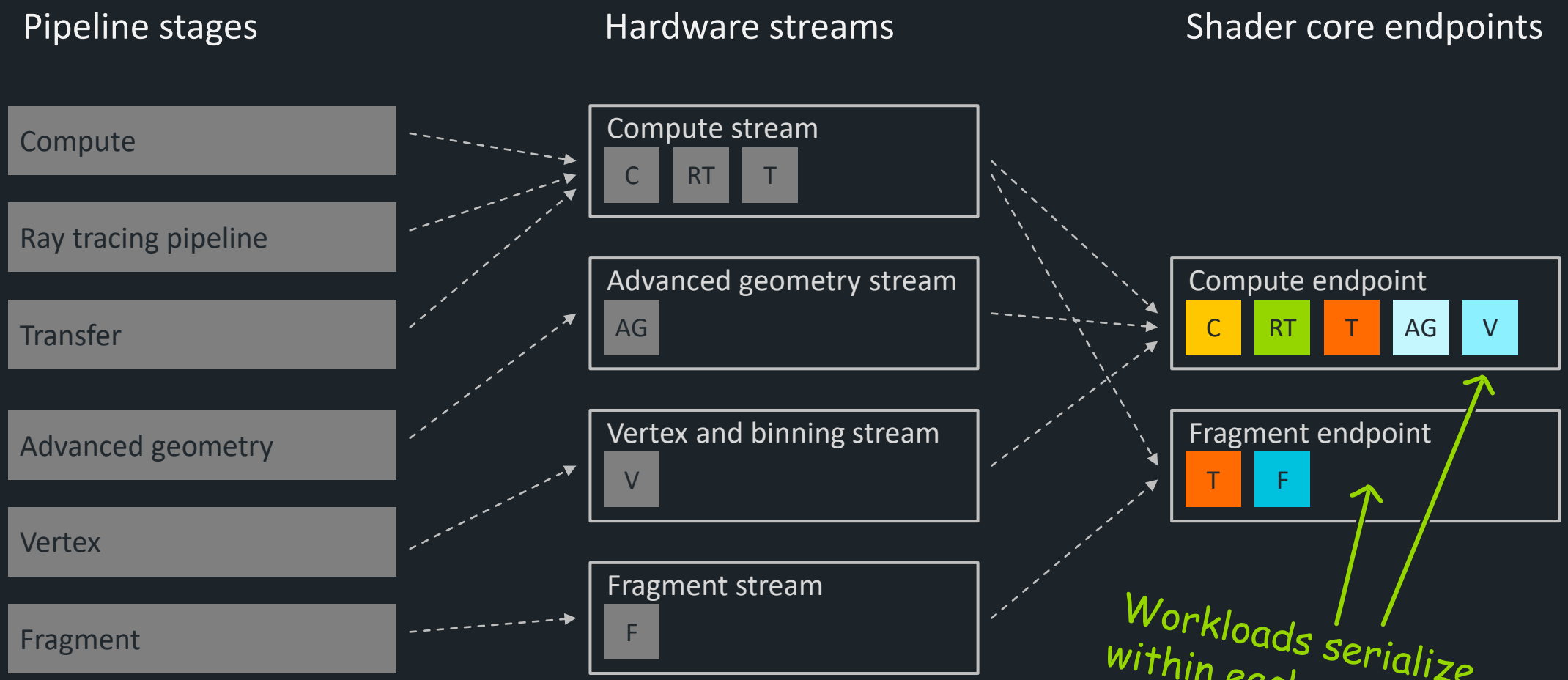
Efficient scheduling

The bigger picture …



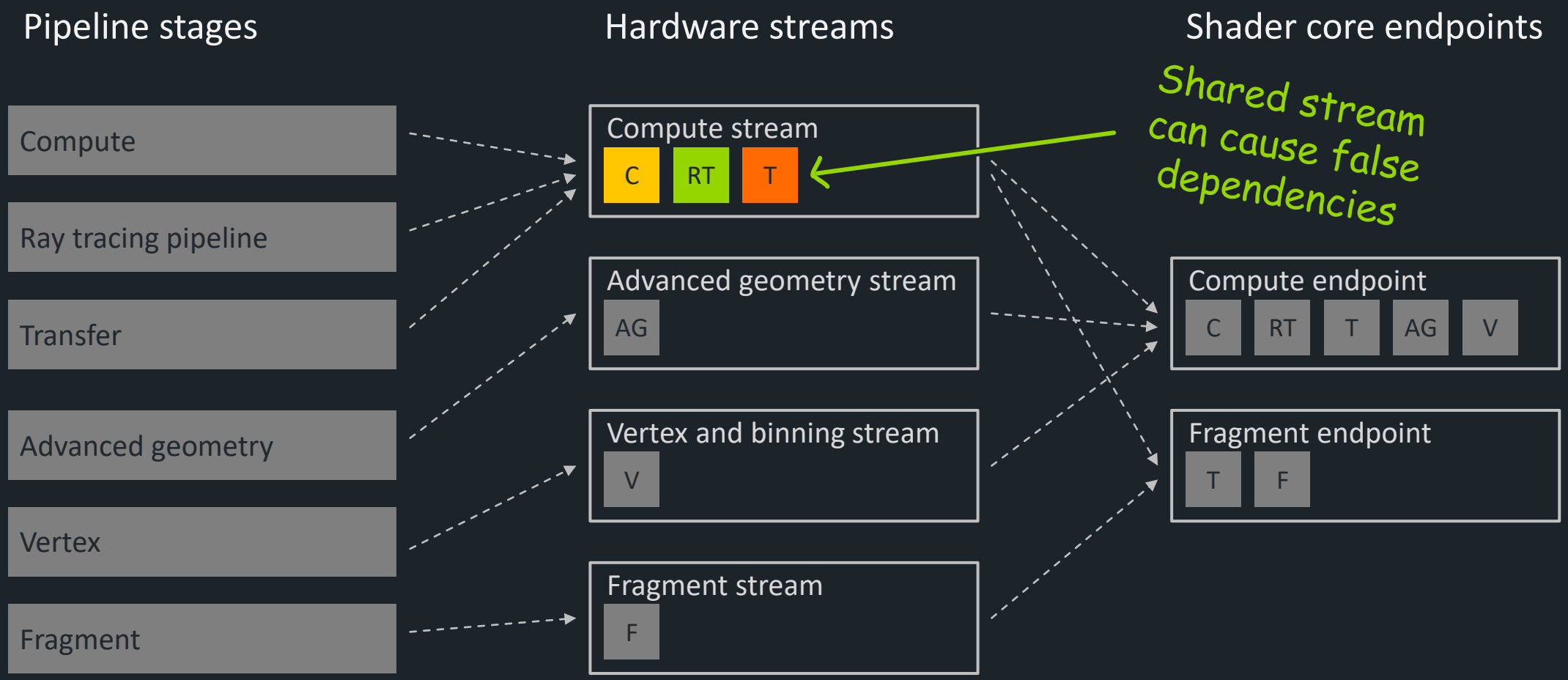
Efficient scheduling

The bigger picture …



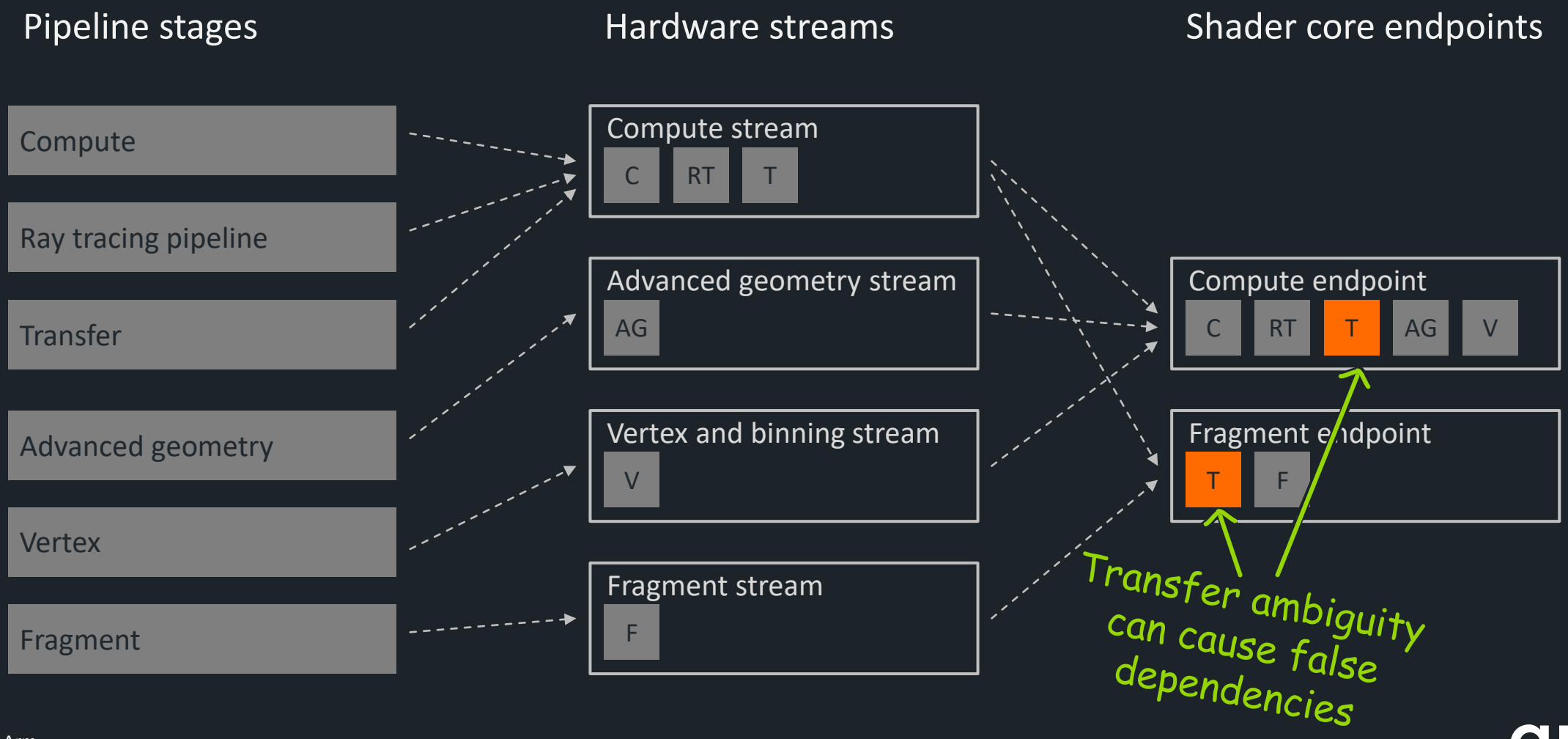
Efficient scheduling

The bigger picture …



Efficient scheduling

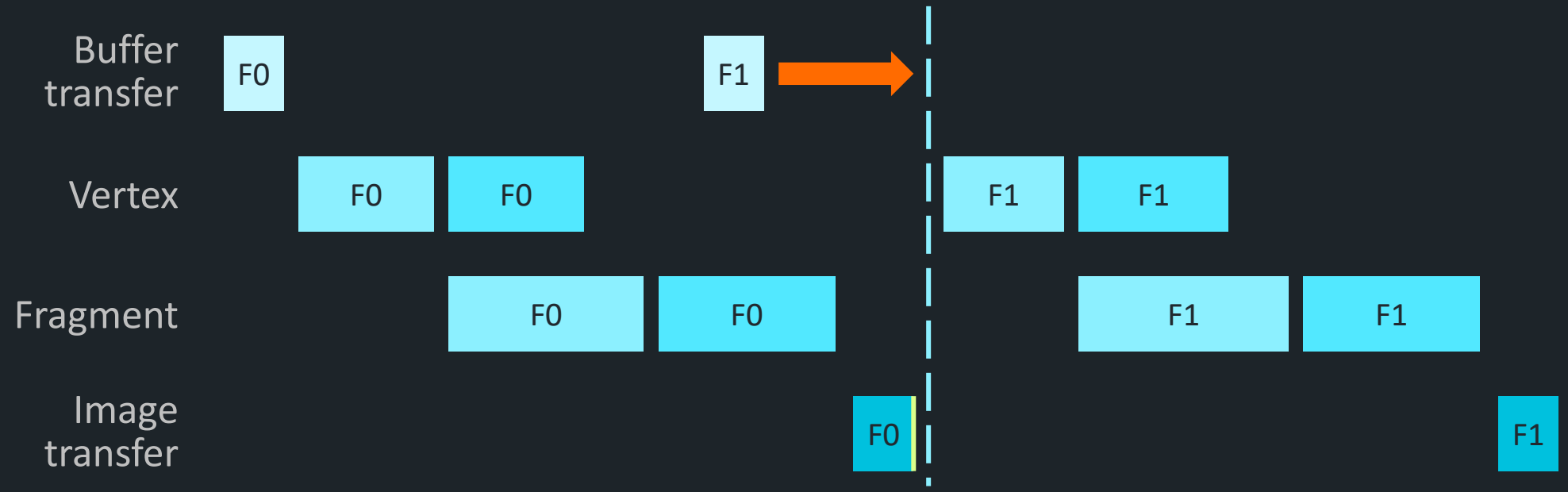
The bigger picture …



Efficient scheduling

Minimize false dependencies

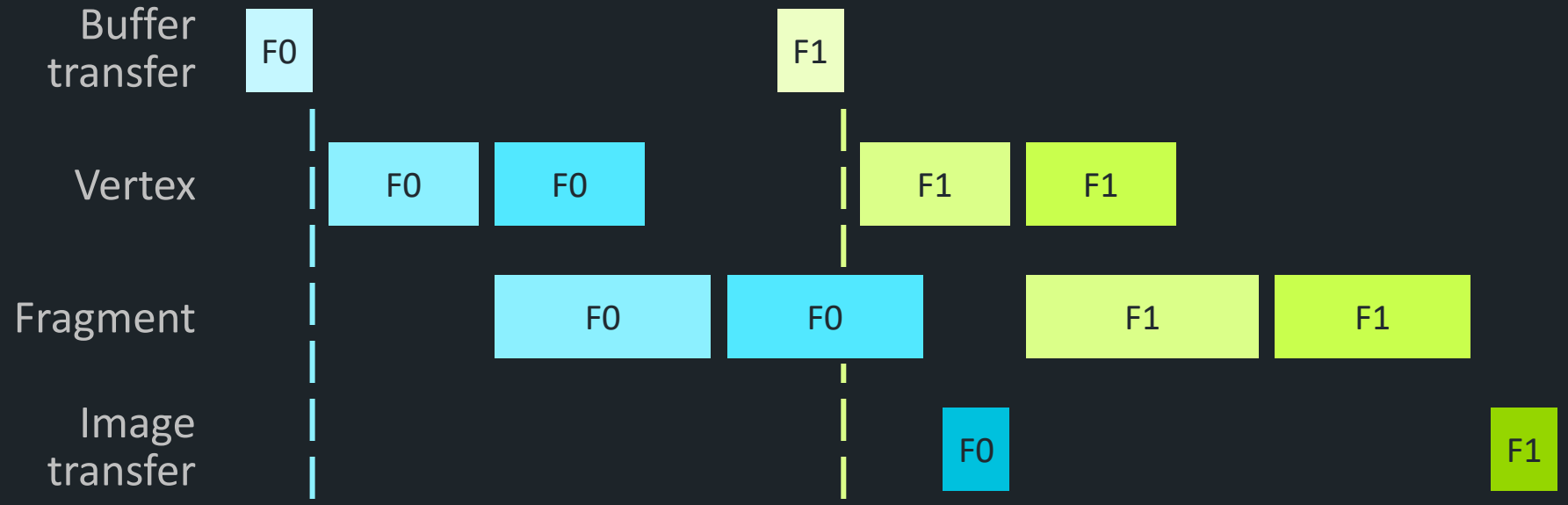
Bad: Single VkQueue causing false cross-frame dependencies



Efficient scheduling

Minimize false dependencies

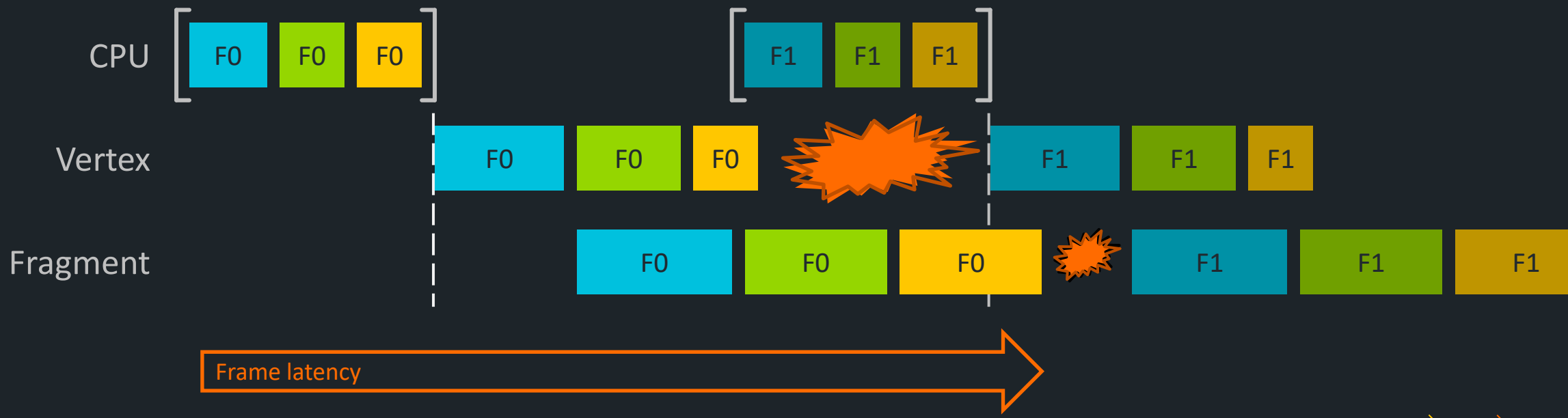
Good: Alternating VkQueues avoids the false dependency



Efficient scheduling

Get idle hardware working

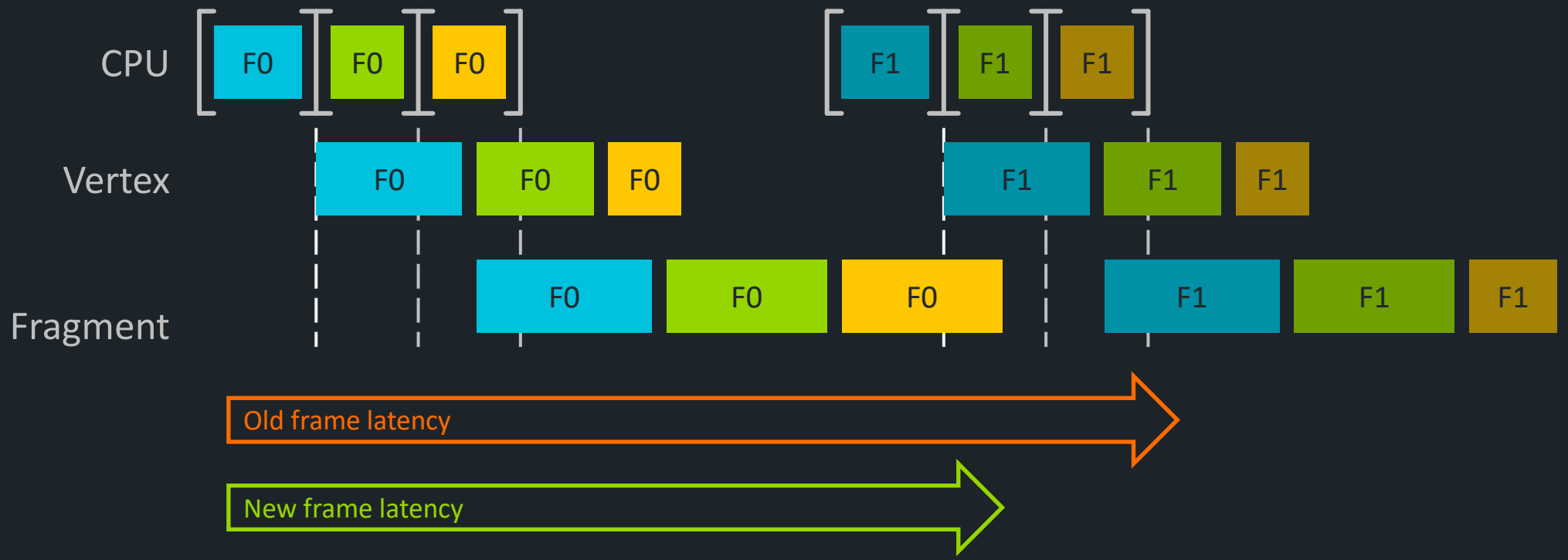
Bad: Single vkQueueSubmit per frame



Efficient scheduling

Get idle hardware working

Good: Multiple vkQueueSubmits per frame



Arm GPU new tech

Vulkan API support

Vulkan extension support

Common asks

- Debugging enhancements
  - VK_EXT_device_fault_report
  - VK_EXT_device_address_binding
- Limits enhancements:
  - Increased max buffer size limit
  - Increased descriptor count limit

Vulkan extension support

Dynamic rendering

- Dynamic rendering
  - VK_KHR_dynamic_rendering_local_read
  - VK_KHR_extended_dynamic_state
  - VK_KHR_extended_dynamic_state2
  - VK_KHR_extended_dynamic_state3

- **Caveats:** Beware of driver bugs in early releases

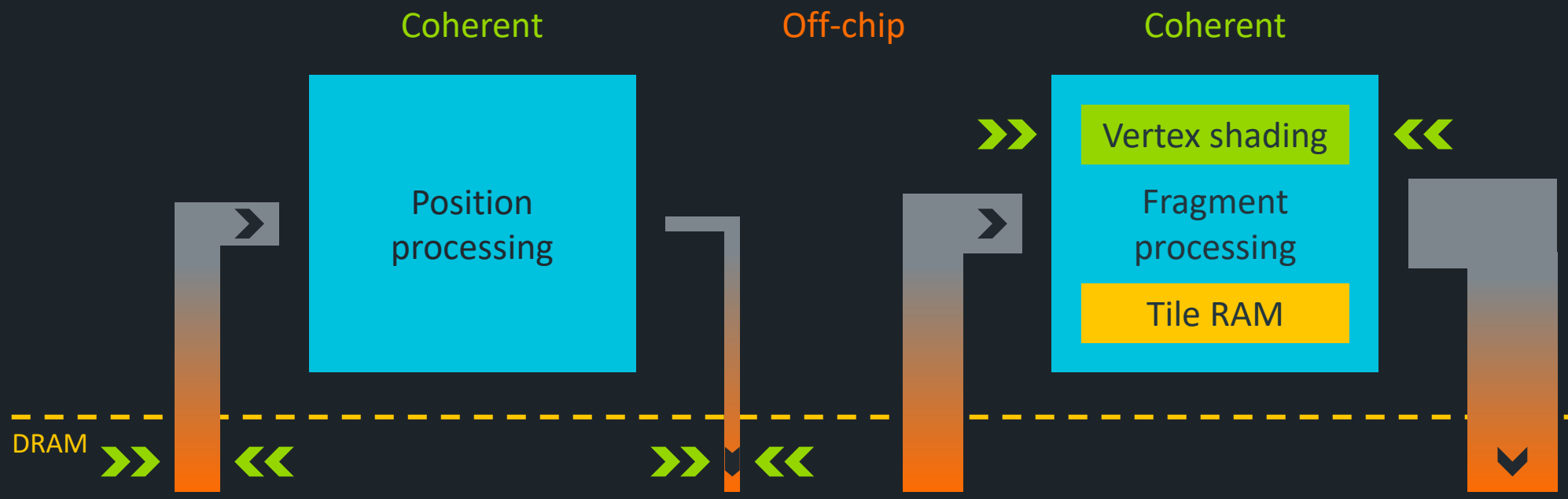
- **New:** Arm GPU Software Developer Errata Notice for Application Developers
  - developer.arm.com/documentation/SDEN-3735689

Arm GPU new tech

Deferred vertex shading

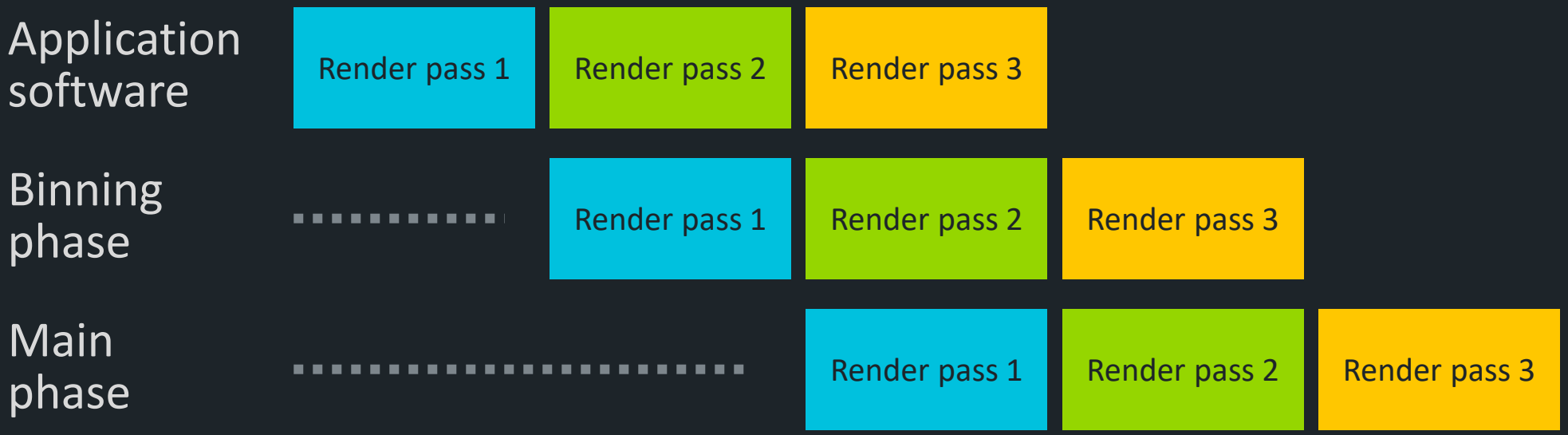
Deferred vertex shading

Introduced with Immortalis-G720 series



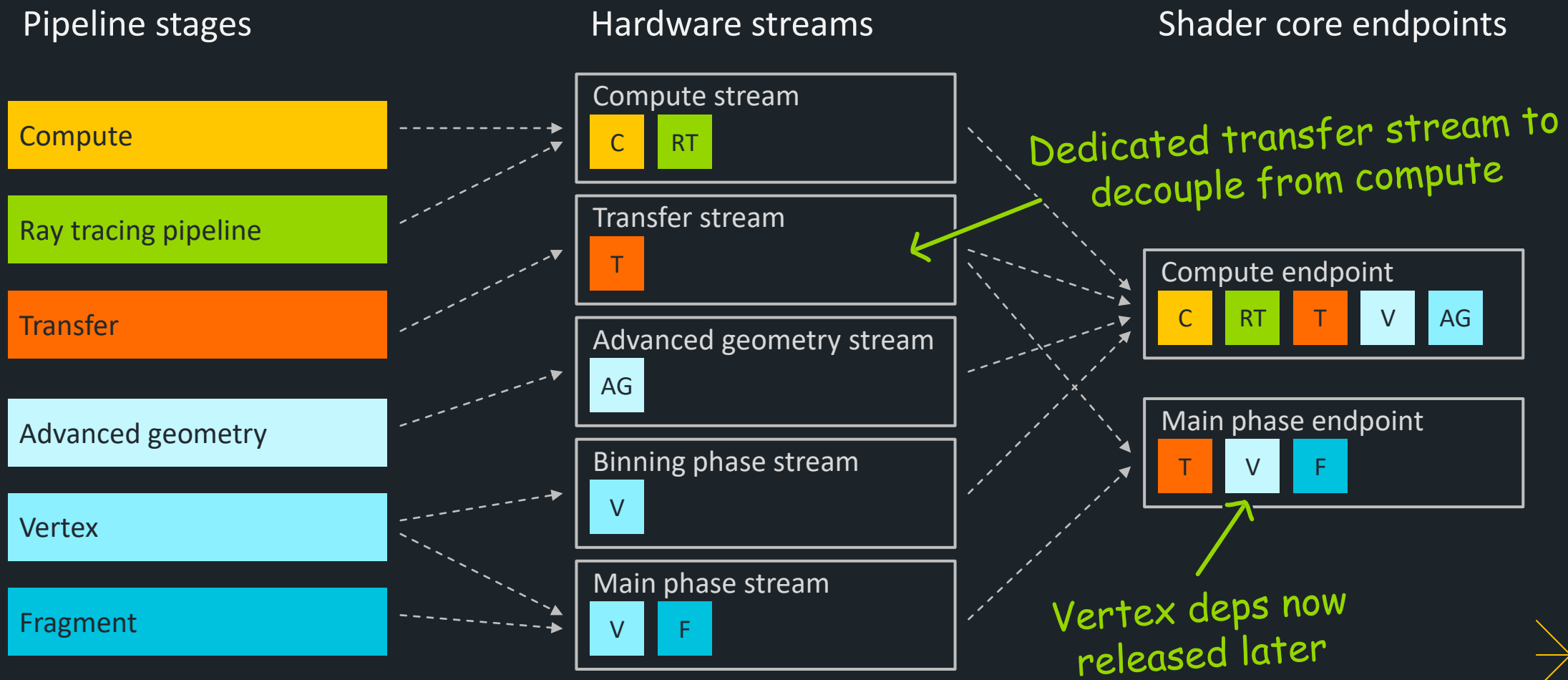
Deferred vertex shading

Introduced with Immortalis-G720 series



Efficient scheduling

The bigger picture …

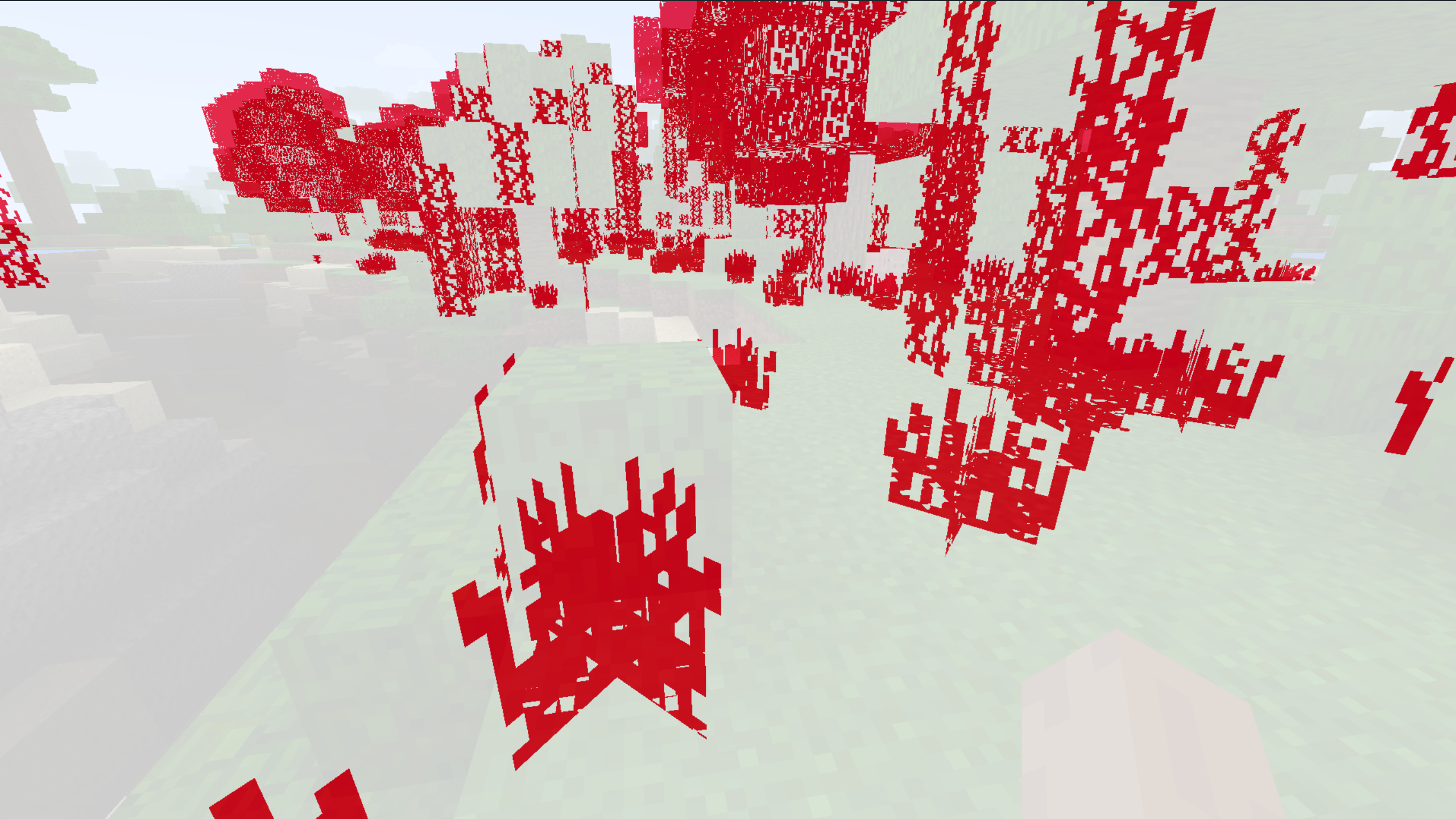


Arm GPU new tech

Fragment prepass HSR

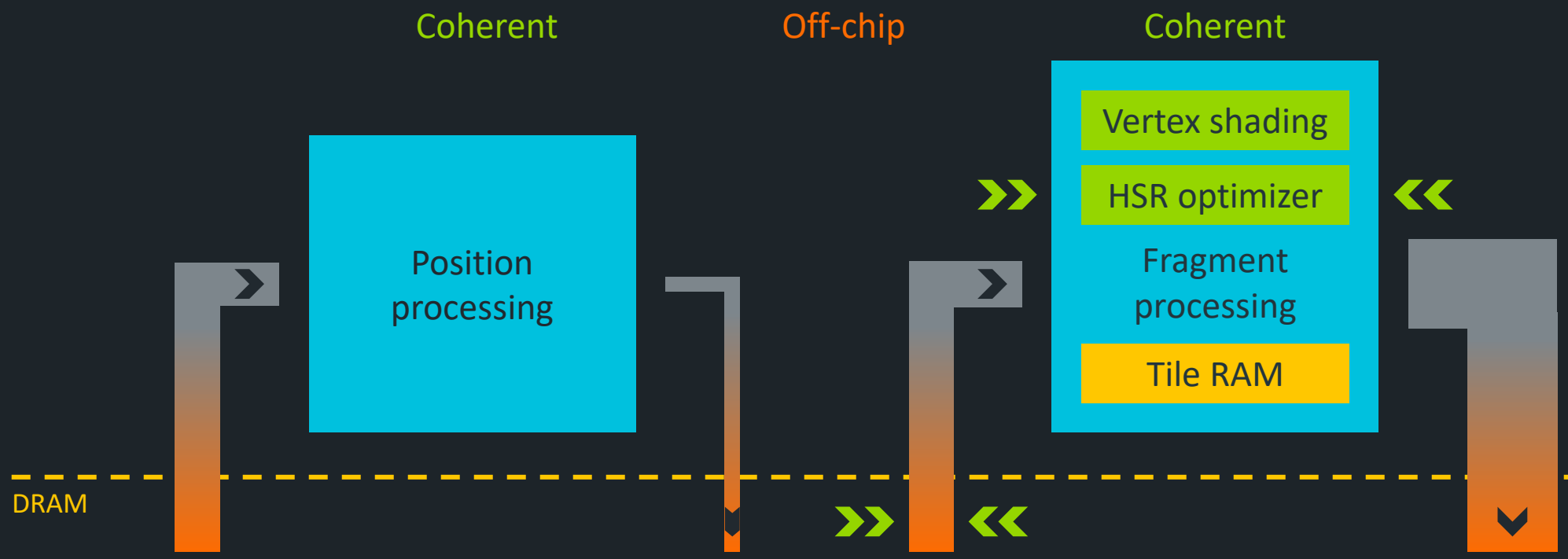
Hidden surface removal revisited

- Arm GPUs have had HSR since Mali-T620
  - Occluder selection based on draw state
  - Occlusion based on quad coverage
- … but it’s been getting less effective over time
  - Larger tiles = less primitive layering
  - Smaller triangles = fewer quads occluded
  - More alpha-testing = fewer occluders
- New approach needed!



Prepass hidden surface removal

Introduced with Immortalis-G925 series



Depth prepass concept

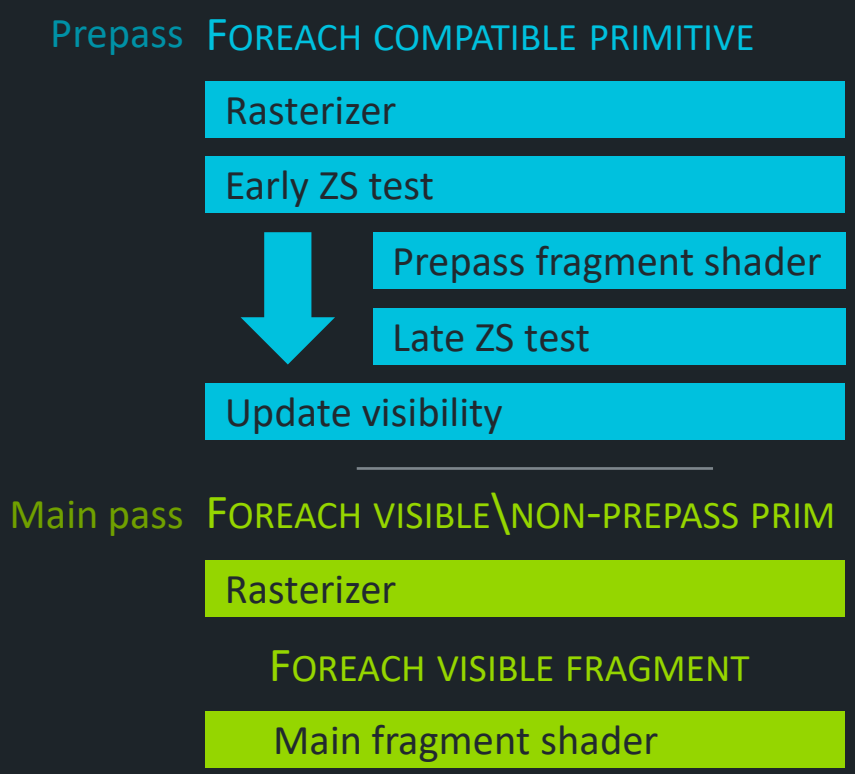
- A common app-driven optimization
  - Render opaque geometry depth-only
  - Render opaque geometry again and color if DEPTH_EQUAL
  - Render transparent geometry on top

Pixel perfect occlusion

Render geometry twice

Fragment prepass hidden surface removal

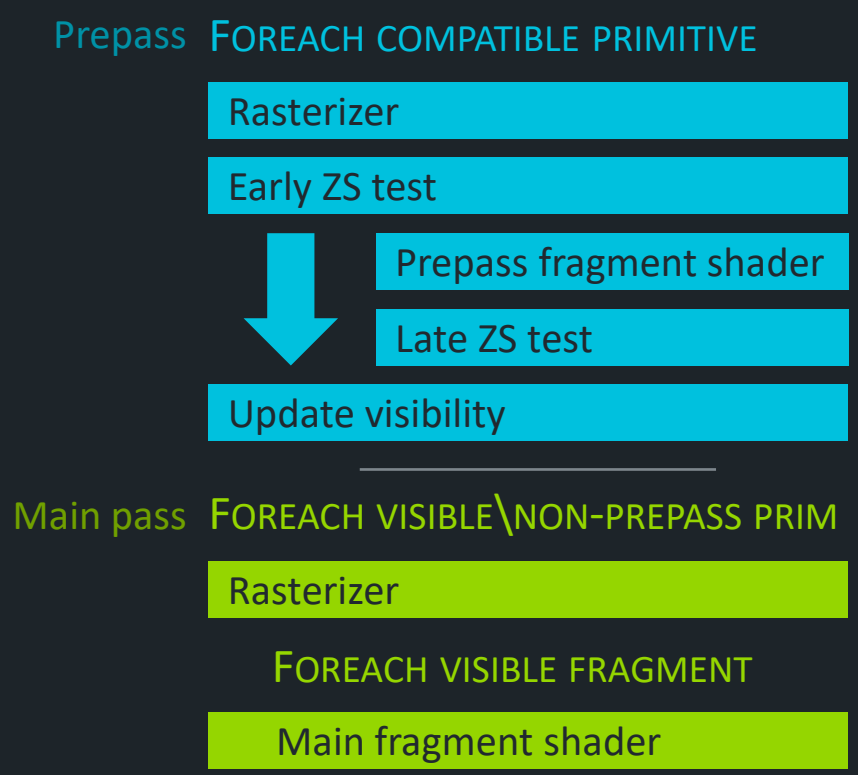
Introduced with Immortalis-G925 series



- Automated depth prepass in hardware!
- Prepass:
  - Processes compatible opaque primitives
  - Skips compatible transparent primitives
  - Terminates on first incompatible primitive
- Main pass:
  - Processes prepass primitives that are visible
  - Processes primitives that skipped the prepass
  - Processes primitives that were after the prepass

Fragment prepass hidden surface removal

Introduced with Immortalis-G925 series



- **Goal:** Avoid incompatible draws terminating prepass
  - **Bad:** Is non-opaque and writes ZS
    - Writes with alpha translucency
    - Writes subset of color attachments written earlier
  - **Bad:** Has read-modify-write side-effects
  - **Bad:** Modifies coverage, writes ZS, and uses layout(early)
  - **Bad:** Uses rasterizer coverage in the shader
    - Uses gl_SampleMask or equivalent
    - Uses gl_HelperInvocation or equivalent
    - Uses centroid interpolation
  - **Bad:** Uses helper framebuffer values cross-warp
- **Goal:** Put incompatible after all compatible draws

Summary!

- High performance mobile is increasingly about bandwidth efficiency
  - Entry-level devices still care about pure GPU performance
- Most mobile GPUs are tile-based
  - Make good use of loadOp, storeOp, and resolve attachments
  - Main API level impact is scheduling differences
- Most tile-based GPU best practices are portable to immediate mode GPUs
  - May not help, but shouldn’t be worse
- We have some major updates to make Arm GPUs more efficient
  - Mostly “just work”
  - API usage playing nicely can help to maximize benefits!

Thank You
Danke
Gracias
Grazie
谢谢
ありがとう
Asante
Merci
감사합니다
धन्यवाद
Kiitos
شكرًا
ধন্যবাদ
תודה