SIGGRAPH2015

Xroads of Discovery

SIGGRAPH2015
Xroads of Discovery

The 42nd International Conference and Exhibition
on Computer Graphics and Interactive Techniques

VALVE

Porting Source 2 to Vulkan

Dan Ginsburg
Valve

Summary

- Source 2 Overview
- Porting to Vulkan
  - Shaders and Pipelines
  - Command Buffers
  - Memory Management
  - Descriptor Sets

Source 2 Overview

Source 2

- OpenGL, DX9, DX11, Vulkan
- Windows, Linux, Mac
- Dota 2 Reborn

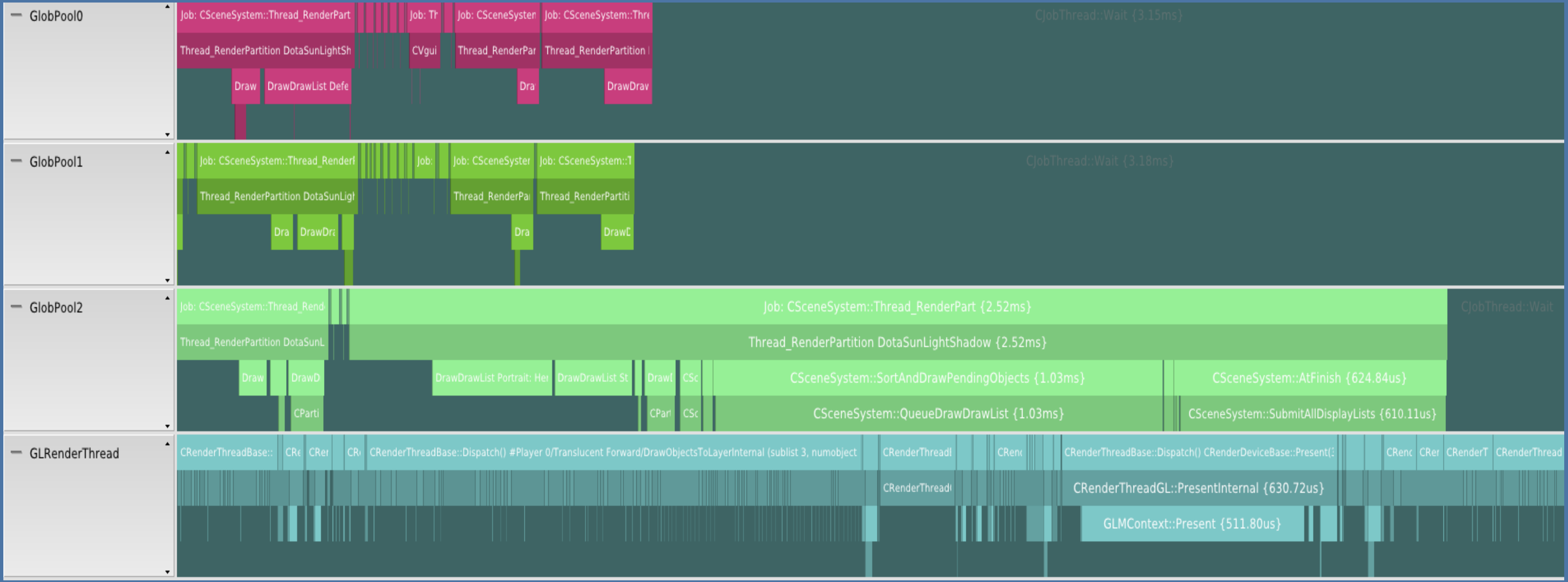
Heavy Damage

Use a Salve to heal. Taking damage will cancel the healing effect

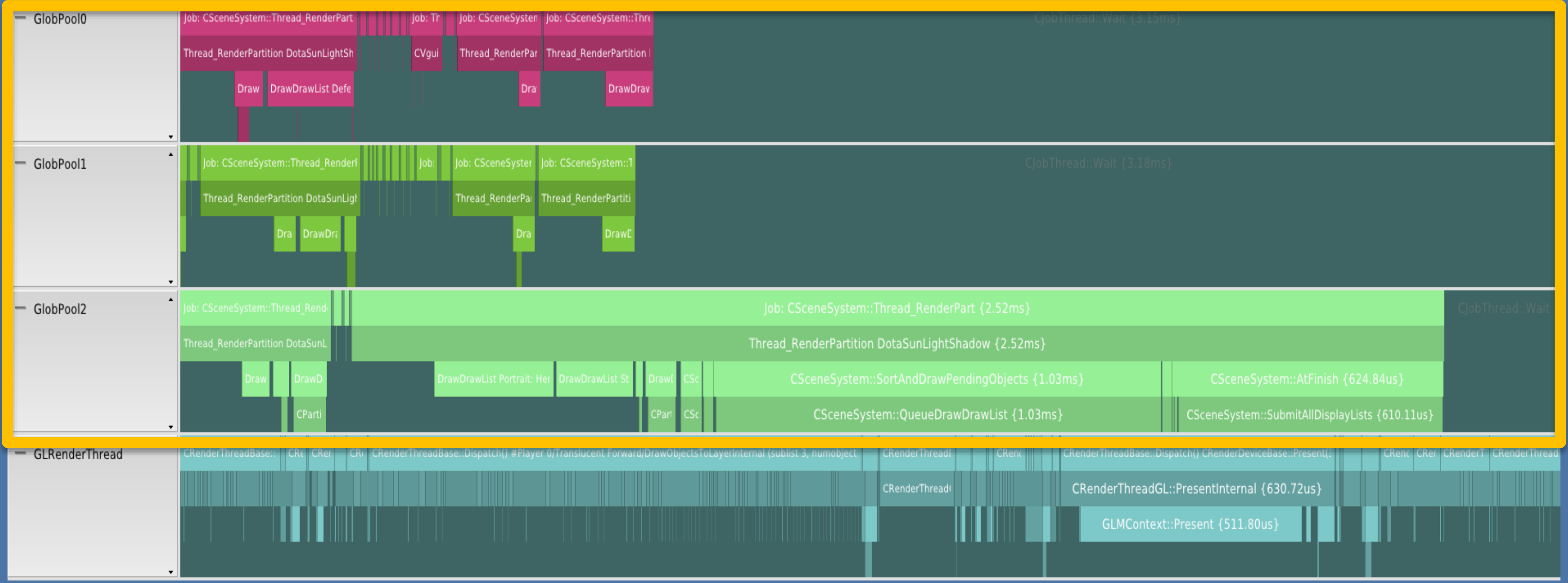
Source 2 Rendering

- DX11-like rendersystem abstraction
- Multithreaded
  - DX9/GL: software command buffers
  - DX11: deferred contexts
  - Single submission thread

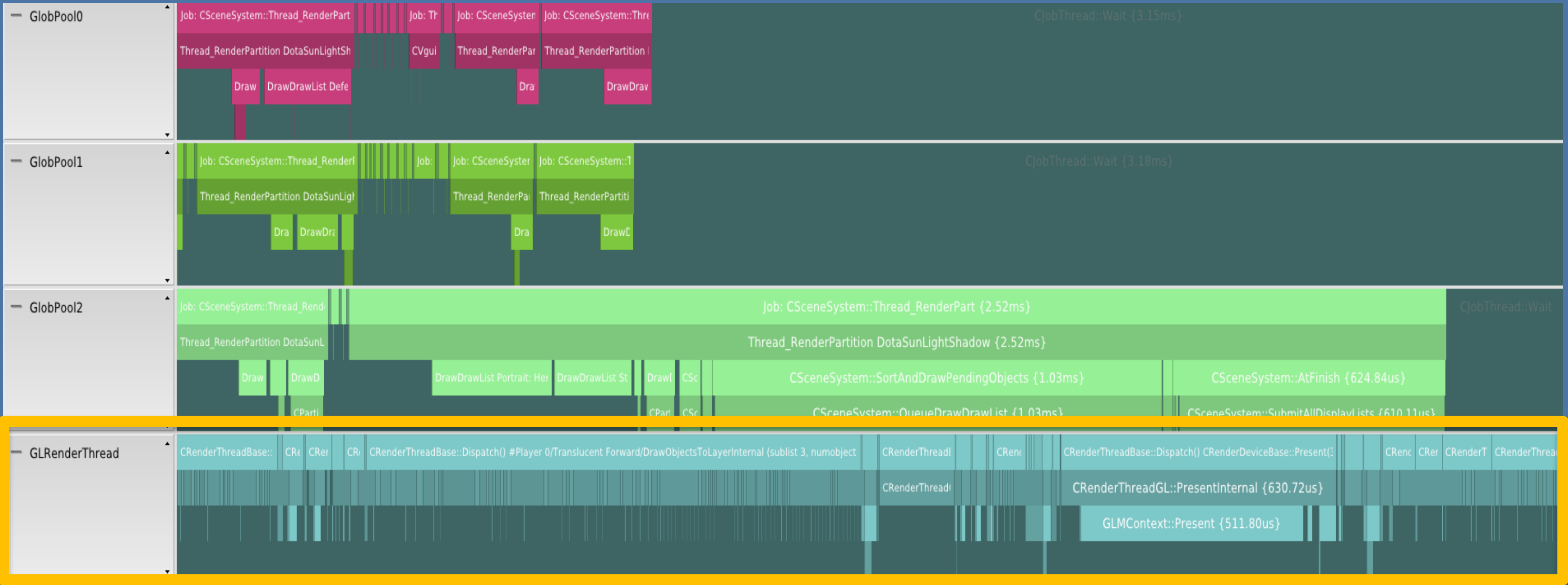
Source 2 Rendering (GL)


GlobPool0
Job: CSceneSystem::Thread_RenderPart
Job: Th
Job: CSceneSysten
Job: CSceneSystem::Thre
Thread_RenderPartition DotaSunLightSh
CVgui
Thread_RenderPar
Thread_RenderPartition
Draw
DrawDrawList Defe
Dra
DrawDrav
CJobThread::Wait {3.15ms}
GlobPool1
Job: CSceneSystem::Thread_RenderP
Job:
Job: CSceneSyster
Job: CSceneSystem::T
Thread_RenderPartition DotaSunLigh
Thread_RenderPa
Thread_RenderPartiti
Dra
DrawDr
Dra
DrawD
CJobThread::Wait {3.18ms}
GlobPool2
Job: CSceneSystem::Thread_Rend
Job: CSceneSystem::Thread_RenderPart {2.52ms}
CJobThread::Wait
Thread_RenderPartition DotaSunL
Thread_RenderPartition DotaSunLightShadow {2.52ms}
Draw
DrawD
DrawDrawList Portrait: Her
DrawDrawList St
DrawL
CSc
CSceneSystem::SortAndDrawPendingObjects {1.03ms}
CSceneSystem::AtFinish {624.84us}
CParti
CPar
CSc
CSceneSystem::QueueDrawDrawList {1.03ms}
CSceneSystem::SubmitAllDisplayLists {610.11us}
GLRenderThread
CRenderThreadBase::
CRe
CRer
CR
CRenderThreadBase::Dispatch() #Player 0/Translucent Forward/DrawObjectsToLayerInternal (sublist 3, numobject
CRenderThread
CRenc
CRenderThreadBase::Dispatch() CRenderDeviceBase::Present(
CRenc
CRer
CRenderT
CRenderThread
CRenderThread
CRenderThreadGL::PresentInternal {630.72us}
GLMContext::Present {511.80us}

Source 2 Rendering (GL)


GlobPool0
Job: CSceneSystem::Thread_RenderPart
Job: Th
Job: CSceneSysten
Job: CSceneSystem::Thre
CJobThread::Wait {3.15ms}
Thread_RenderPartition DotaSunLightSh
CVgui
Thread_RenderPar
Thread_RenderPartition
Draw
DrawDrawList Defe
Dra
DrawDraw
GlobPool1
Job: CSceneSystem::Thread_Render
Job:
Job: CSceneSyster
Job: CSceneSystem::T
CJobThread::Wait {3.18ms}
Thread_RenderPartition DotaSunLigh
Thread_RenderPar
Thread_RenderPartiti
Dra
DrawDra
Dra
DrawD
GlobPool2
Job: CSceneSystem::Thread_Rend
Job: CSceneSystem::Thread_RenderPart {2.52ms}
CJobThread::Wait
Thread_RenderPartition DotaSunL
Thread_RenderPartition DotaSunLightShadow {2.52ms}
Draw
DrawD
DrawDrawList Portrait: Her
DrawDrawList St
DrawL
CSc
CSceneSystem::SortAndDrawPendingObjects {1.03ms}
CSceneSystem::AtFinish {624.84us}
CParti
CPart
CSc
CSceneSystem::QueueDrawDrawList {1.03ms}
CSceneSystem::SubmitAllDisplayLists {610.11us}
GLRenderThread
CRenderThreadBase::
CRe
CRer
CR
CRenderThreadBase::Dispatch() #Player 0/Translucent Forward/DrawObjectsToLayerInternal (sublist 3, numobject
CRenderThread
CRenc
CRenderThreadBase::Dispatch() CRenderDeviceBase::Present(
CRenc
CRer
CRenderT
CRenderThread
CRenderThread
CRenderThreadGL::PresentInternal {630.72us}
GLMContext::Present {511.80us}

Source 2 Rendering (GL)


GlobPool0
Job: CSceneSystem::Thread_RenderPart
Job: Th
Job: CSceneSysten
Job: CSceneSystem::Thre
Thread_RenderPartition DotaSunLightSh
CVgui
Thread_RenderPar
Thread_RenderPartition
Draw
DrawDrawList Defe
Dra
DrawDrav
CJobThread::Wait {3.15ms}
GlobPool1
Job: CSceneSystem::Thread_Renderf
Job:
Job: CSceneSyster
Job: CSceneSystem::T
Thread_RenderPartition DotaSunLigt
Thread_RenderPa
Thread_RenderPartiti
Dra
DrawDr
Dra
DrawD
CJobThread::Wait {3.18ms}
GlobPool2
Job: CSceneSystem::Thread_Rend
Job: CSceneSystem::Thread_RenderPart {2.52ms}
CJobThread::Wait
Thread_RenderPartition DotaSunL
Thread_RenderPartition DotaSunLightShadow {2.52ms}
Draw
DrawD
DrawDrawList Portrait: Her
DrawDrawList St
Drawl
CSc
CSceneSystem::SortAndDrawPendingObjects {1.03ms}
CSceneSystem::AtFinish {624.84us}
CParti
CPart
CSc
CSceneSystem::QueueDrawDrawList {1.03ms}
CSceneSystem::SubmitAllDisplayLists {610.11us}
GLRenderThread
CRenderThreadBase::
CRe
CRer
CR
CRenderThreadBase::Dispatch() #Player 0/Translucent Forward/DrawObjectsToLayerInternal (sublist 3, numobject
CRenderThread
CRen
CRenderThreadBase::Dispatch() CRenderDeviceBase::Present(
CRenc
CRer
CRenderT
CRenderThrea
CRenderThread
CRenderThreadGL::PresentInternal {630.72us}
GLMContext::Present {511.80us}

Source 2 Vulkan Port

- Started with GL and DX11 renderer
  - DX11 deferred contexts mapped well to Vulkan command buffers
  - Leveraged GLSL shader conversion

Shaders and Pipelines

Porting Shaders to Vulkan

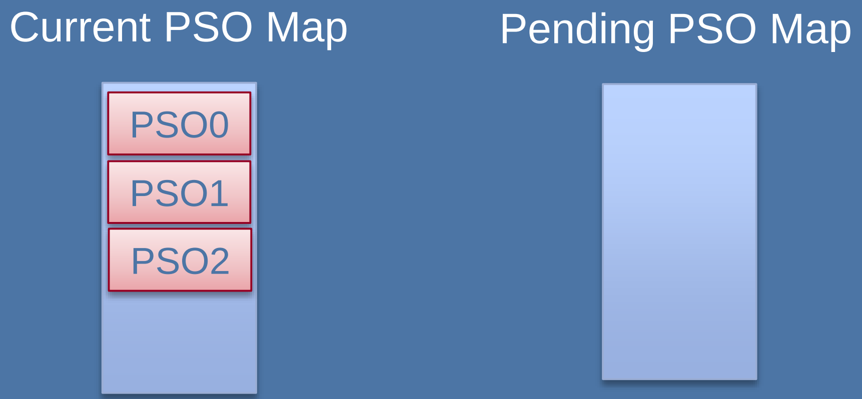
- HLSL -> GLSL
  - See: *Moving Your Games to OpenGL, Steam Dev Days 2014*
- GLSL -> SPIR-V
  - Descriptor set layout qualifiers to GLSL
  - Open source glslang SPIR-V backend
  - SPVremapper for compression
  - *https://github.com/KhronosGroup/glslang*

Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing

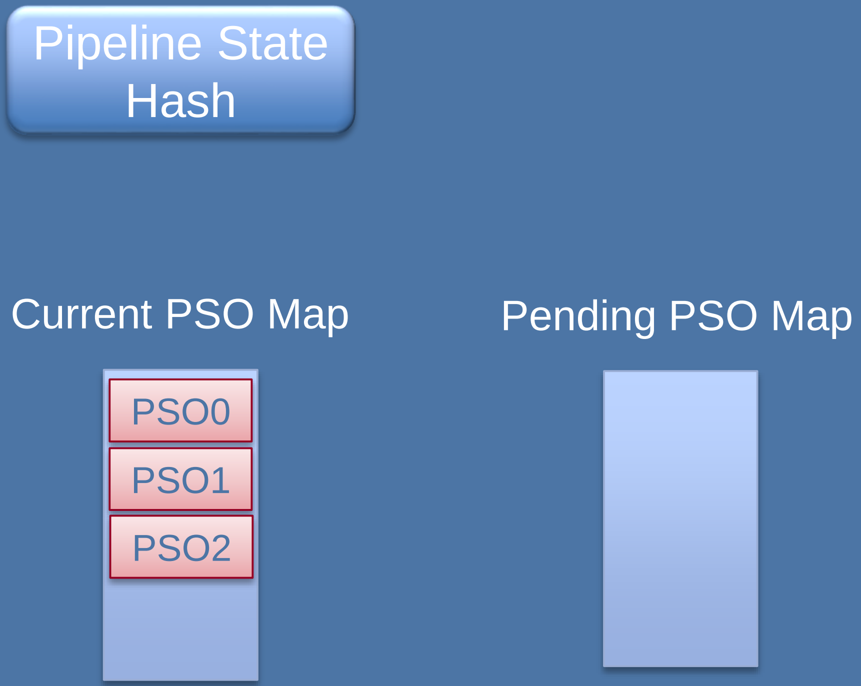
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



Pipeline State Hash

Current PSO Map

Pending PSO Map

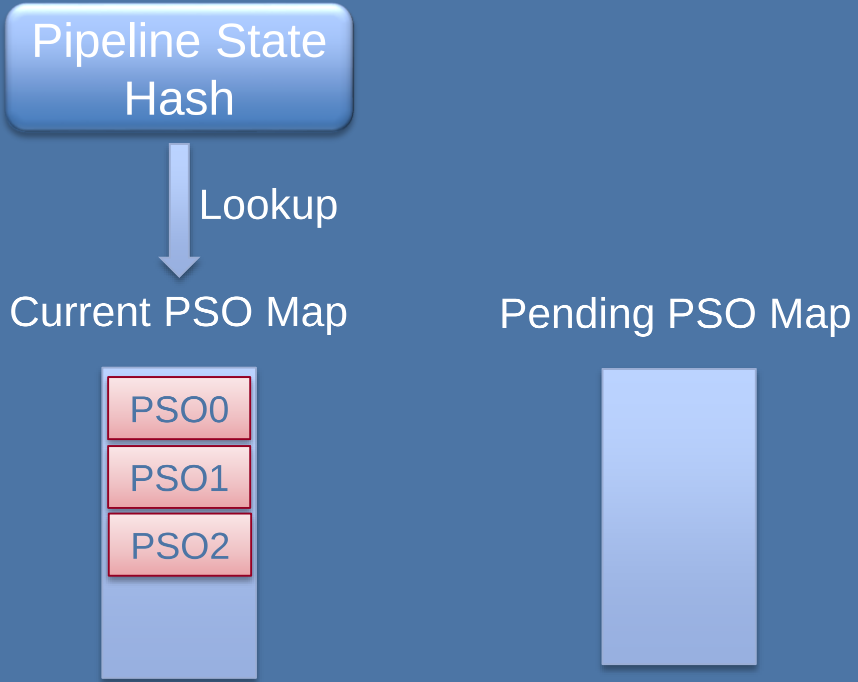
PSO0

PSO1

PSO2

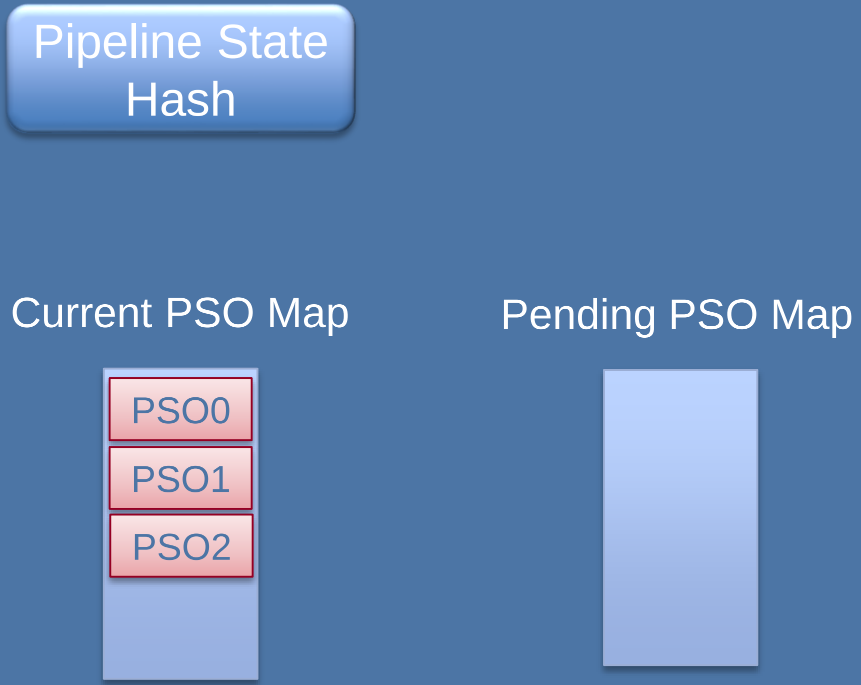
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



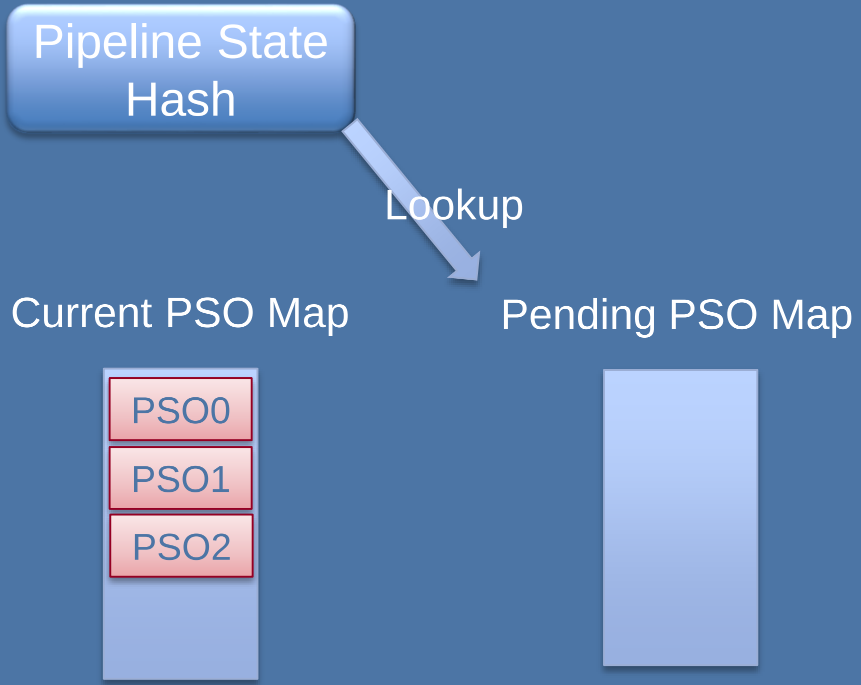
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



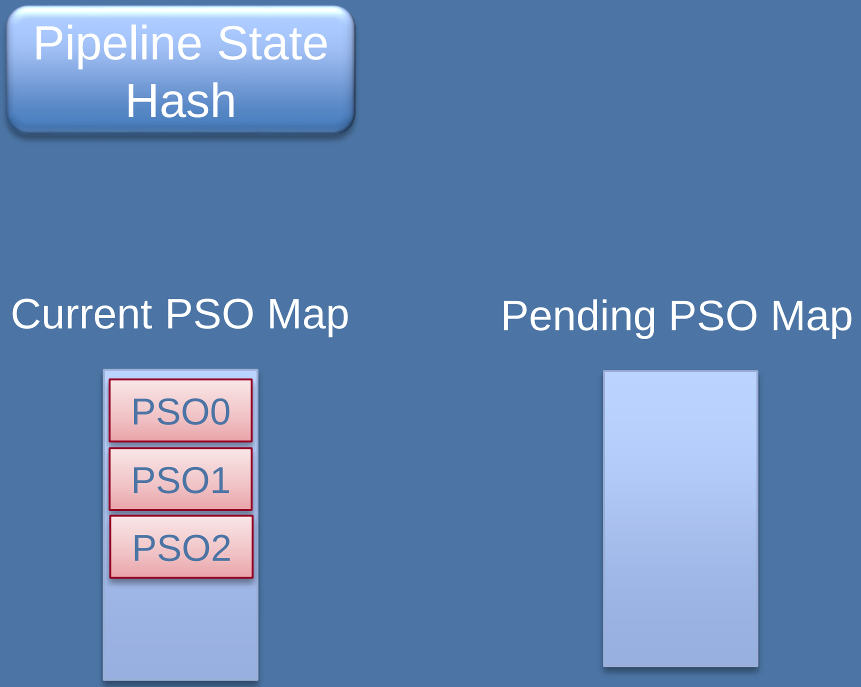
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



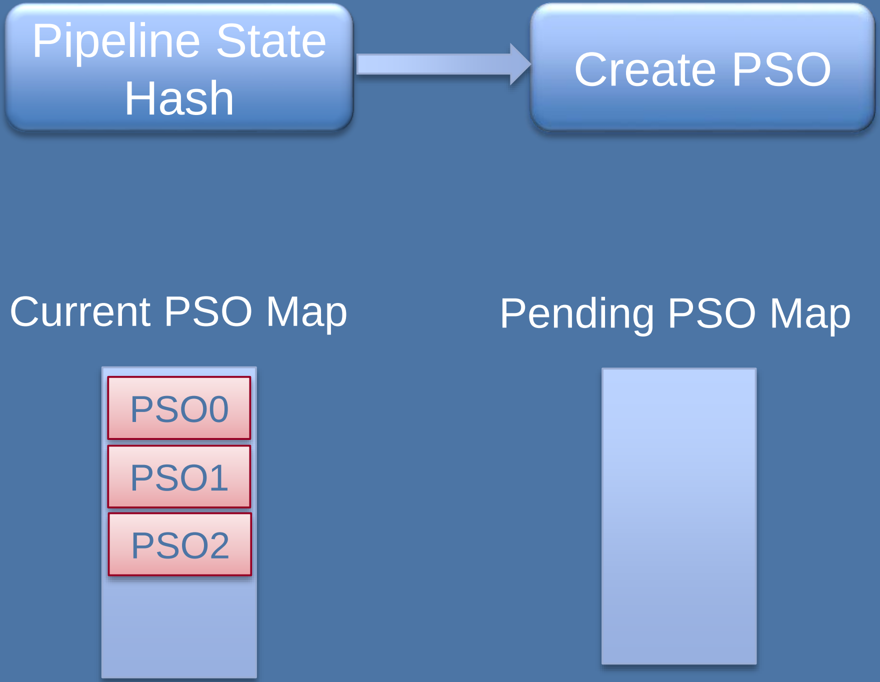
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



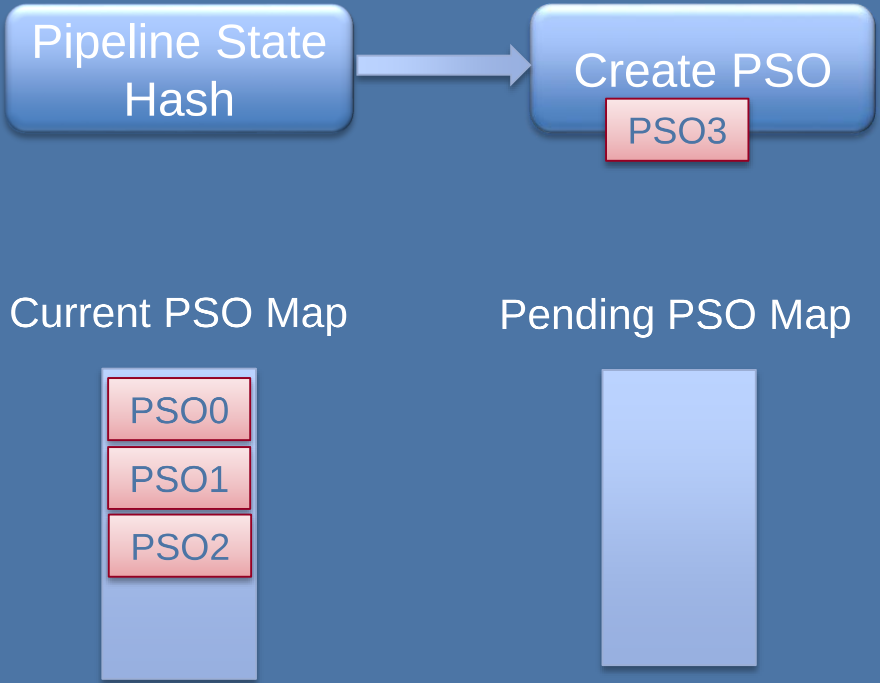
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - – PSO Map
  - – Pending and current
    - Reduces mutexing



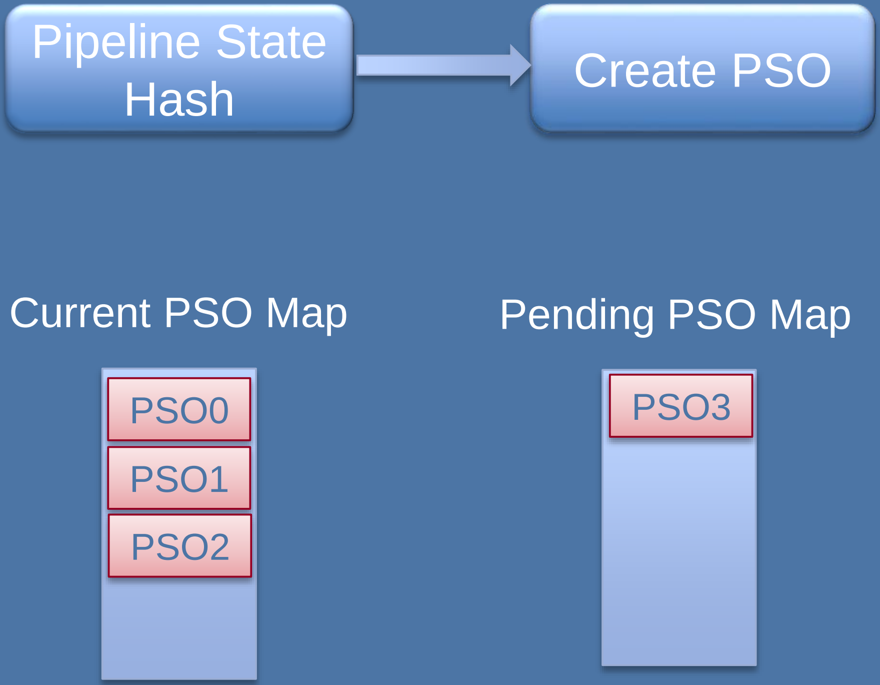
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



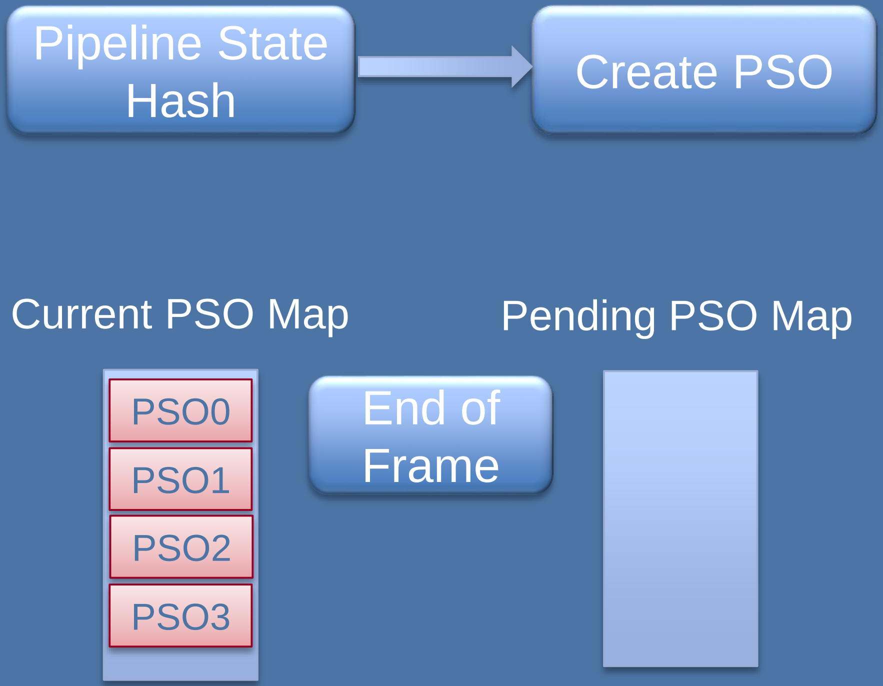
Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



Pipeline State Objects (PSOs)

- Each thread caches pipeline state
- Global pipeline manager
  - PSO Map
  - Pending and current
    - Reduces mutexing



Pipeline State Hash

Create PSO

Current PSO Map

Pending PSO Map

PSO0

PSO1

PSO2

PSO3

End of Frame

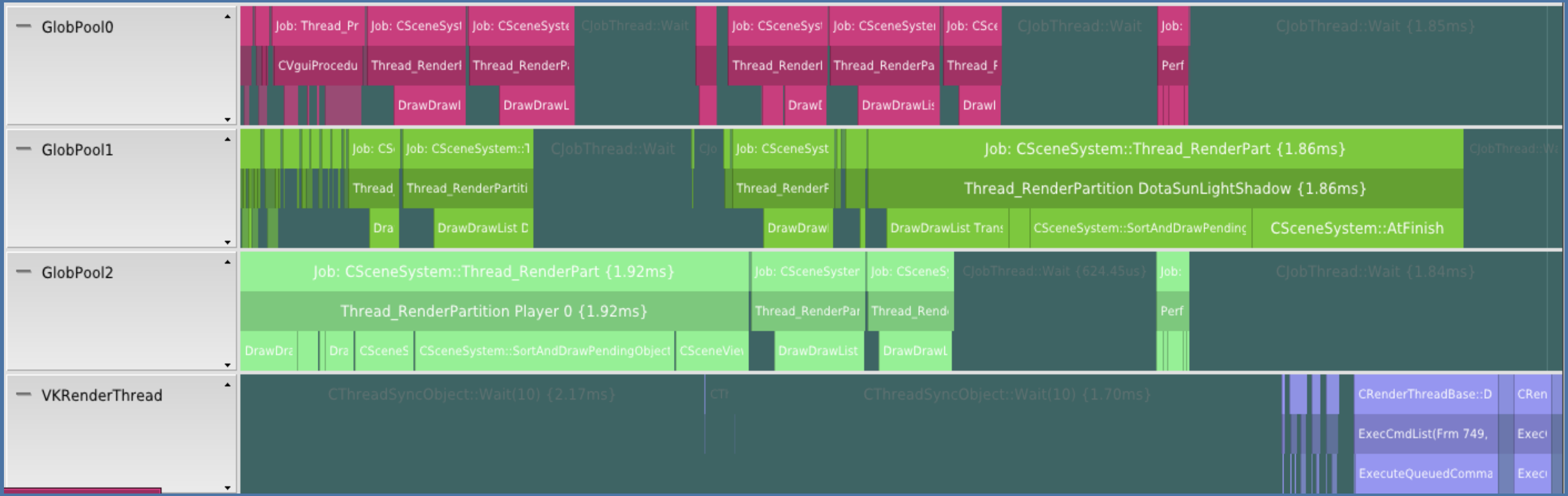
Command Buffers

Command Buffers

- Used where DX11 deferred contexts were used
- Each thread builds command buffer
- Single thread performs submission to queue

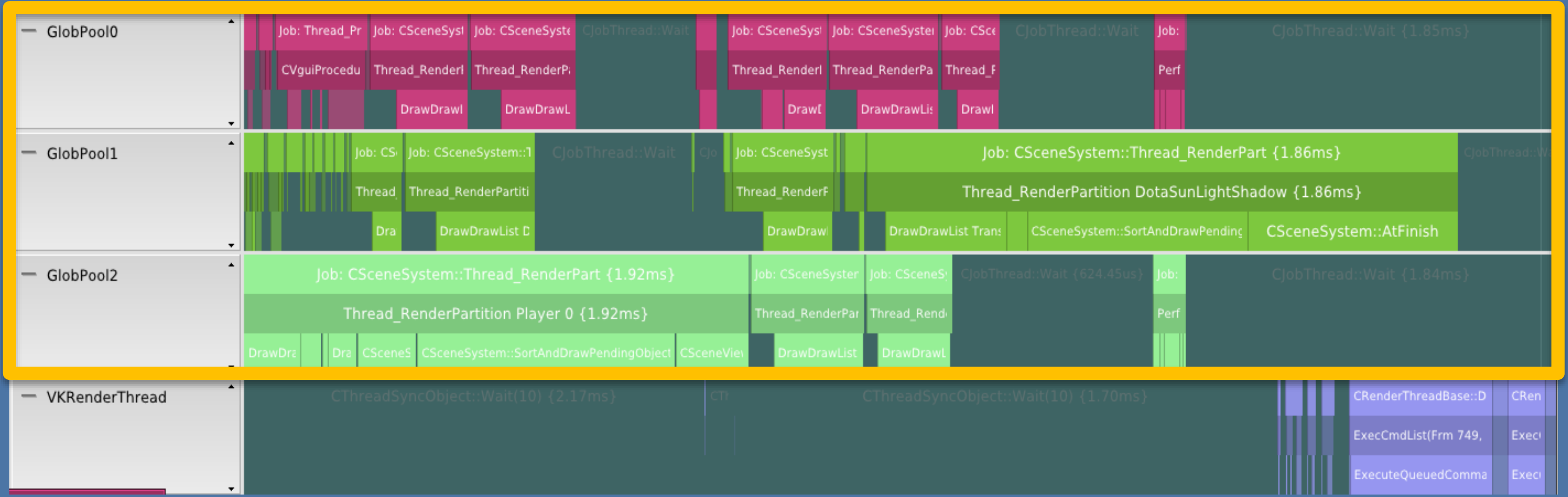
Command Buffers

- Used where DX11 deferred contexts were used
- Each thread builds command buffer
- Single thread performs submission to queue



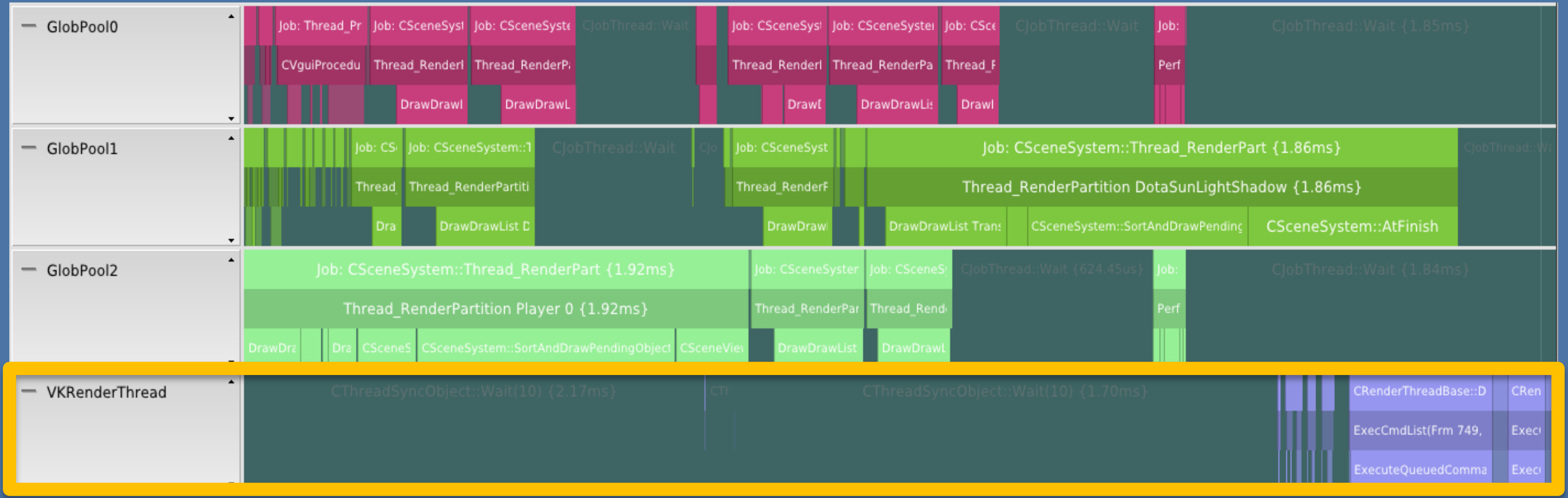
Command Buffers

- Used where DX11 deferred contexts were used
- Each thread builds command buffer
- Single thread performs submission to queue



Command Buffers

- Used where DX11 deferred contexts were used
- Each thread builds command buffer
- Single thread performs submission to queue



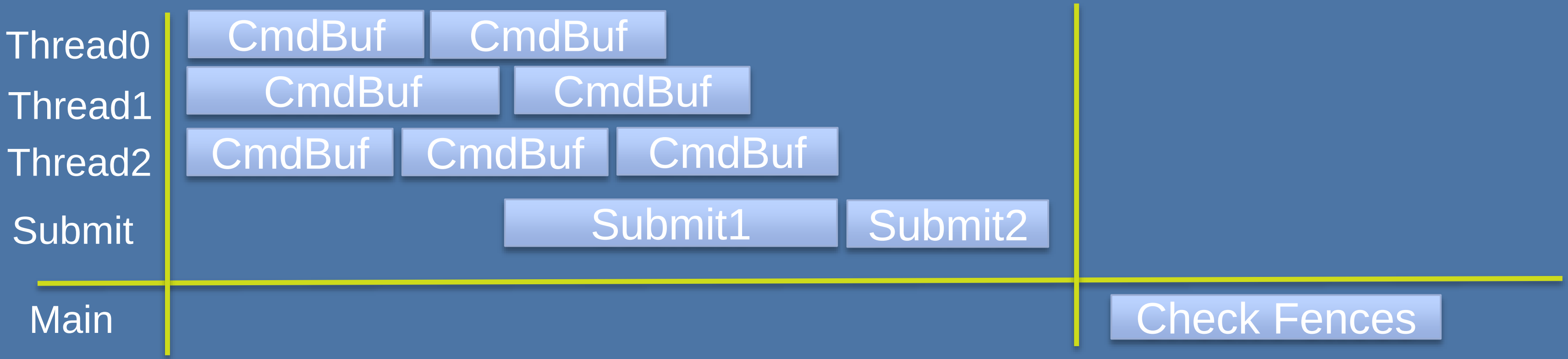
Command Buffers

- Recycled within per-thread pools

Thread 0 CommandPool

Thread 1 CommandPool

Thread 2 CommandPool



Thread0 CmdBuf CmdBuf

Thread1 CmdBuf CmdBuf

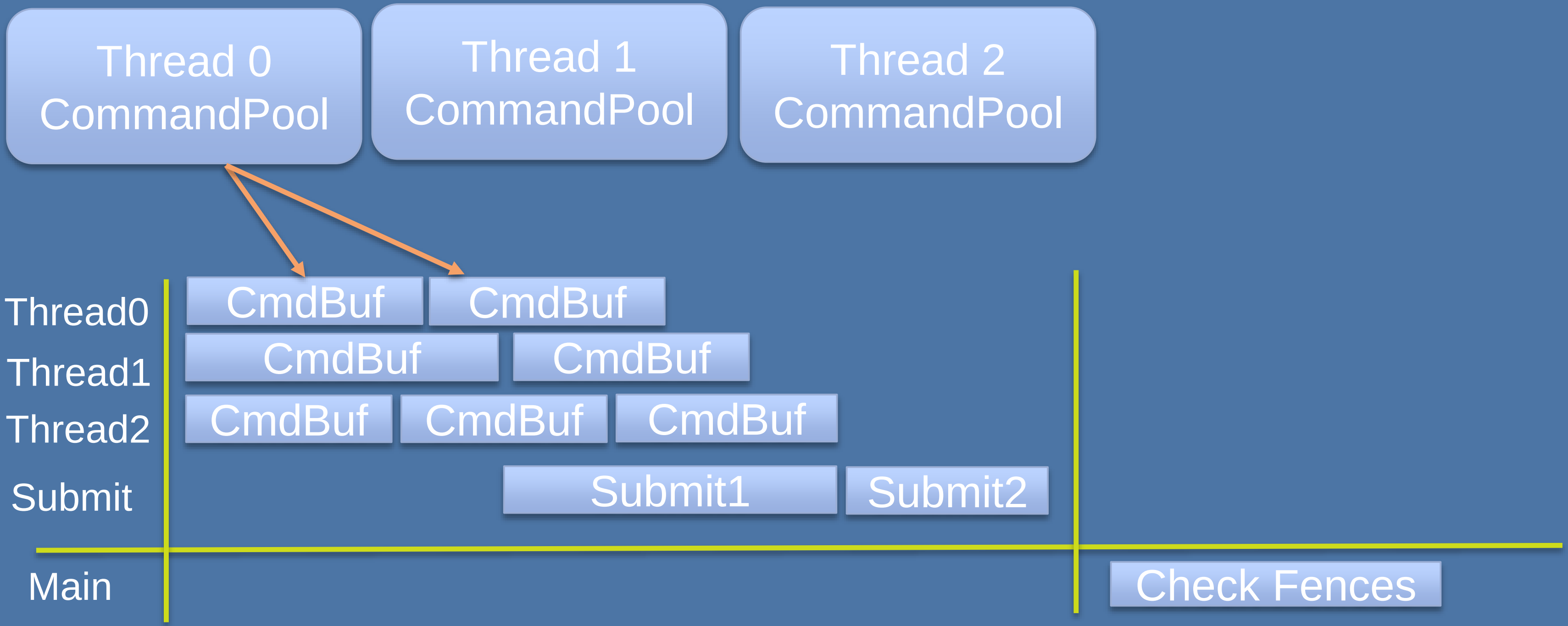
Thread2 CmdBuf CmdBuf CmdBuf

Submit Submit1 Submit2

Main Check Fences

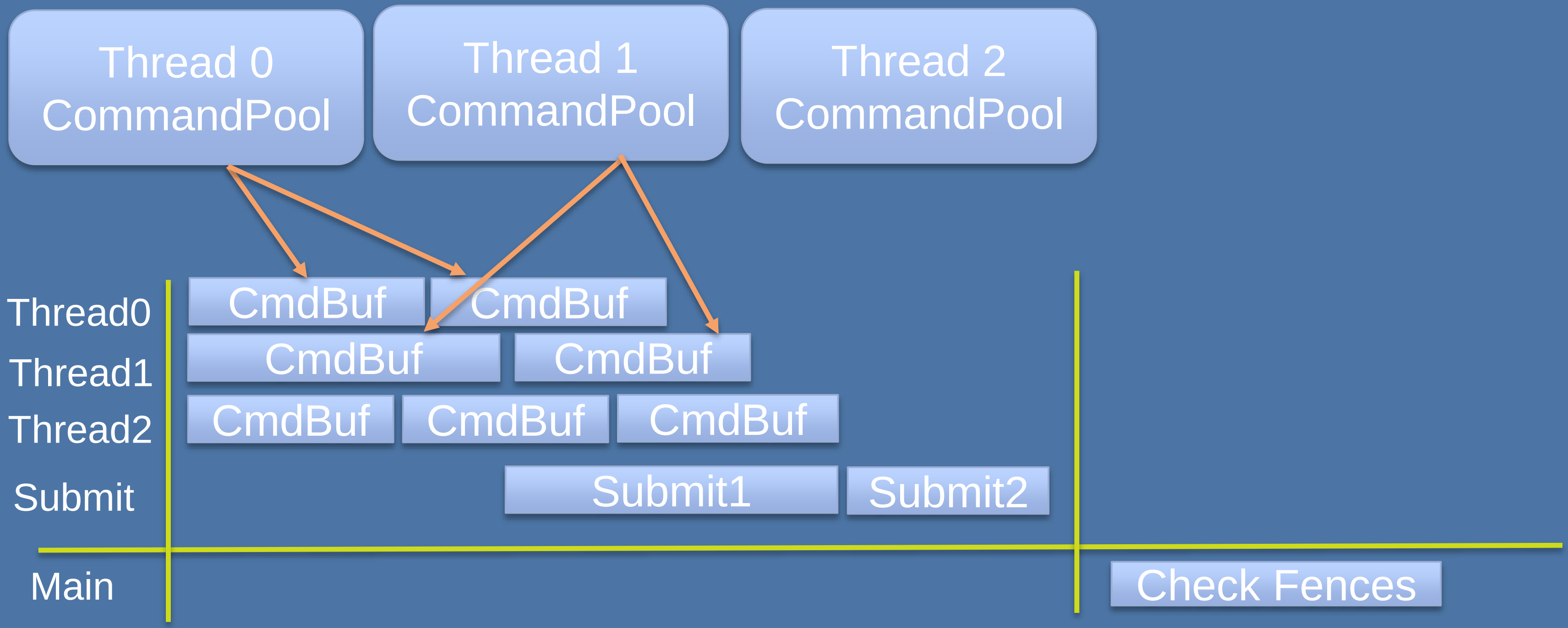
Command Buffers

- Recycled within per-thread pools



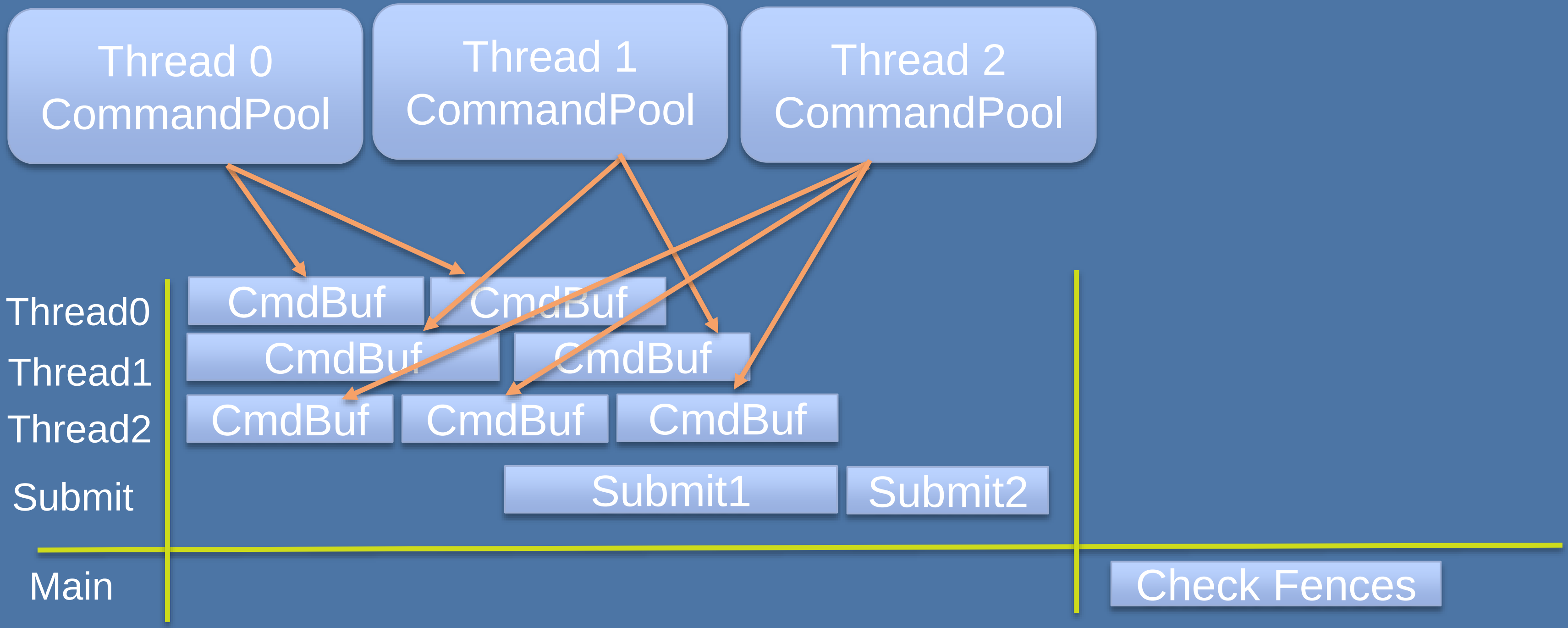
Command Buffers

- Recycled within per-thread pools



Command Buffers

- Recycled within per-thread pools



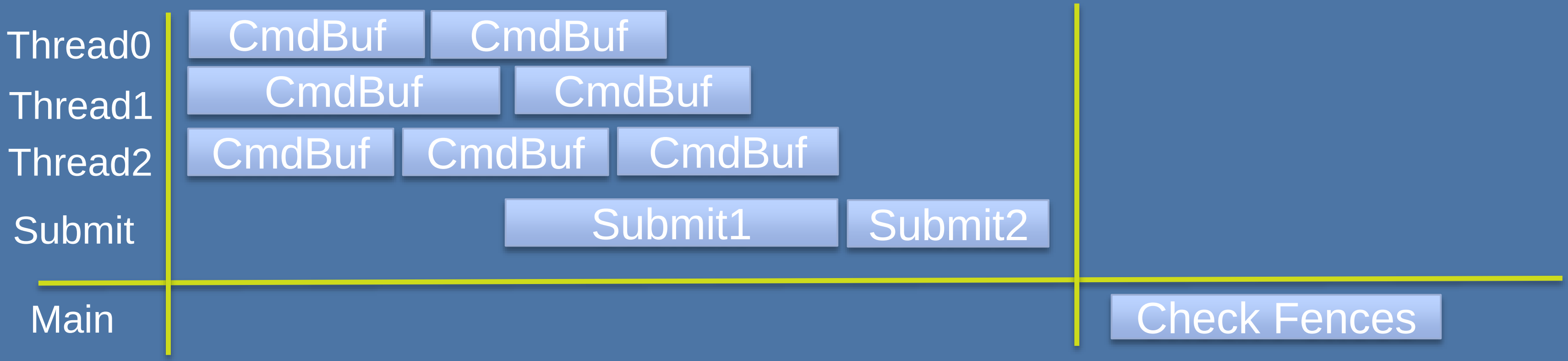
Command Buffers

- Recycled within per-thread pools

Thread 0 CommandPool

Thread 1 CommandPool

Thread 2 CommandPool



Thread0: CmdBuf, CmdBuf

Thread1: CmdBuf, CmdBuf

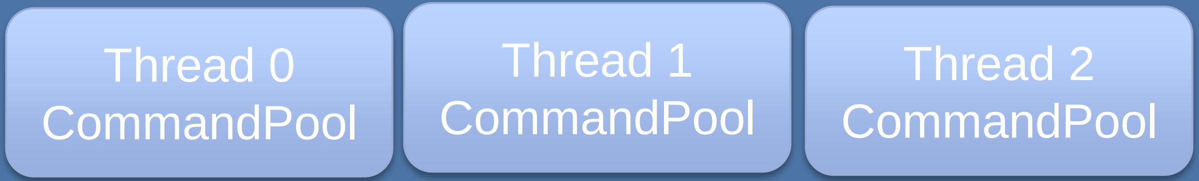
Thread2: CmdBuf, CmdBuf, CmdBuf

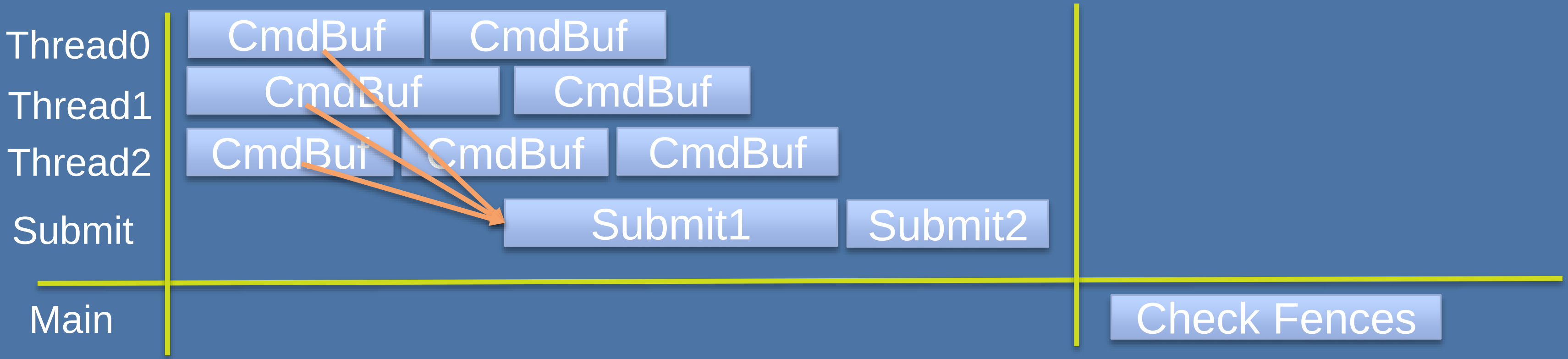
Submit: Submit1, Submit2

Main: Check Fences

Command Buffers

- Recycled within per-thread pools





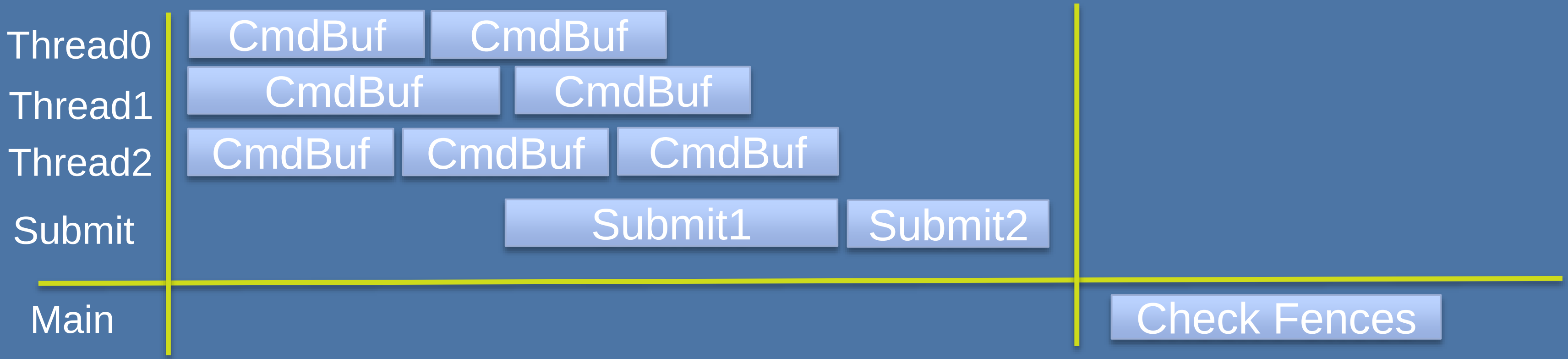
Command Buffers

- Recycled within per-thread pools

Thread 0 CommandPool

Thread 1 CommandPool

Thread 2 CommandPool



Thread0 CmdBuf CmdBuf

Thread1 CmdBuf CmdBuf

Thread2 CmdBuf CmdBuf CmdBuf

Submit Submit1 Submit2

Main Check Fences

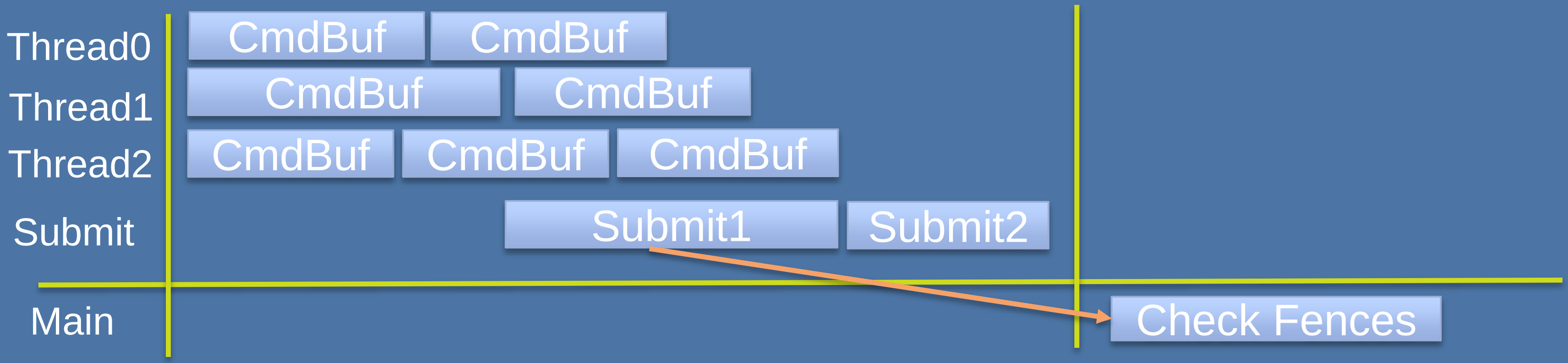
Command Buffers

- Recycled within per-thread pools

Thread 0 CommandPool

Thread 1 CommandPool

Thread 2 CommandPool



Thread0: CmdBuf, CmdBuf

Thread1: CmdBuf, CmdBuf

Thread2: CmdBuf, CmdBuf, CmdBuf

Submit: Submit1, Submit2

Main: Check Fences

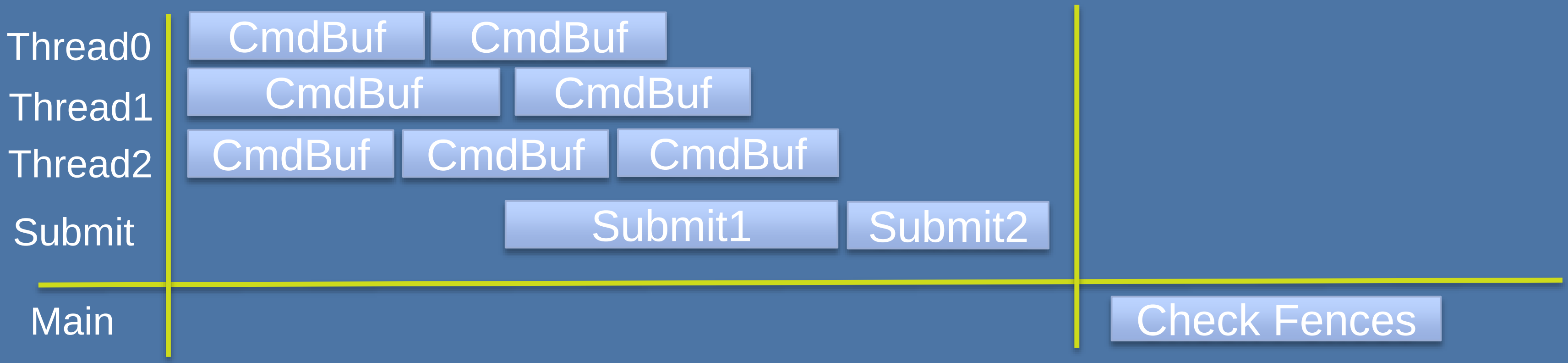
Command Buffers

- Recycled within per-thread pools

Thread 0 CommandPool

Thread 1 CommandPool

Thread 2 CommandPool



Thread0: CmdBuf, CmdBuf

Thread1: CmdBuf, CmdBuf

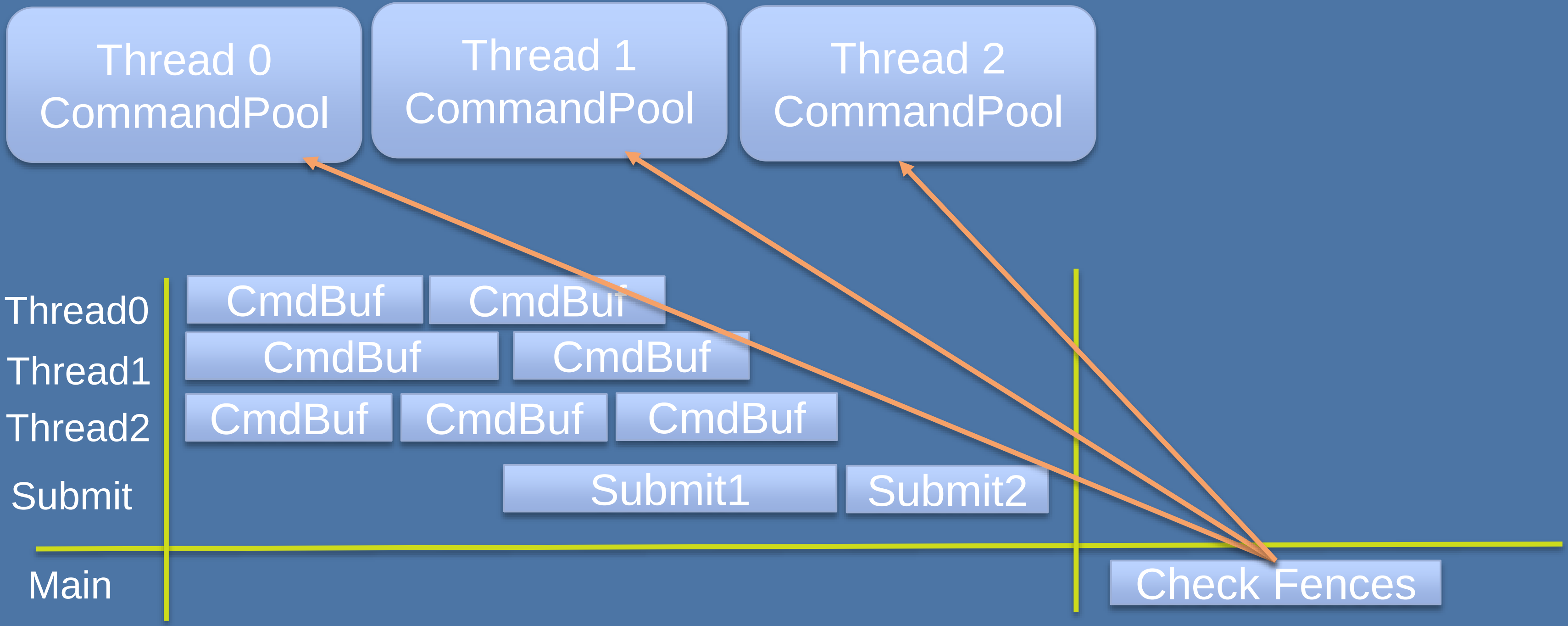
Thread2: CmdBuf, CmdBuf, CmdBuf

Submit: Submit1, Submit2

Main: Check Fences

Command Buffers

- Recycled within per-thread pools



Command Buffer Performance

- Submit in batches
  - vkQueueSubmit has cost on Windows
  - Faster to group submissions together
- Minimize number of command buffers
- Minimize memory referenced per command buffer
- Use VK_CMD_BUFFER_OPTIMIZE_ONE_TIME_SUBMIT_BIT
  - Optimize for one-time submission

Memory Management

General Strategies

- Pool resources together
  - Reduces memory reference count
- Use per-thread pools to reduce contention
- Recycle dynamic pools on frame boundaries

Resources

Static Resources	Dynamic Resources
• Global Pools • Device Only • Textures/Render Targets • 128MB Pools • VB/IB/CBs • 8MB Pools	• Per-Thread Pools • Host Visible (Persistently Mapped) • VB/IB/CBs • 8MB Pools

Dynamic Vertex/Index Buffers

- To update:
  - Grab new offset from per-thread pool
  - memcpy into pool
  - Bind VBs with: vkCmdBindVertexBuffers(..,buffer,offset)
  - Bind IBs with: vkCmdBindIndexBuffer(..,buffer,offset,..)
- Recycle pools when last GPU fence of frame retires

Dynamic Uniform Buffers

- Differences from VB/IBs:
  - UBOs are bound via descriptors
  - Use dynamic UBOs to avoid vkUpdateDescriptors
  - Pass UBO offset to vkCmdBindDescriptorSets

Dynamic Textures

- Staged in persistently mapped buffers
  - Recycled per-frame
- Copy with vkCmdCopyBufferToImage

Descriptor Sets

Descriptor Set - Ideal

- Allocate and bake descriptor sets up front
- Group sets by update frequency
- Only update changed sets

Descriptor Set - Reality

- Difficult to bake descriptors with DX11-like abstraction
- Our approach
  - Pre-allocate descriptor sets with fixed slots
  - Only bind to used slots
  - Update descriptors each draw

Summary

- Source 2 Overview
- Porting to Vulkan
  - Shaders and Pipelines
  - Command Buffers
  - Memory Management
  - Descriptor Sets

Questions?

- dang@valvesoftware.com
- Khronos BOF:
  - Wed. August 12th, 5:30-7:30
  - JW Marriott LA Live in the Platinum Ballroom Salon F-I