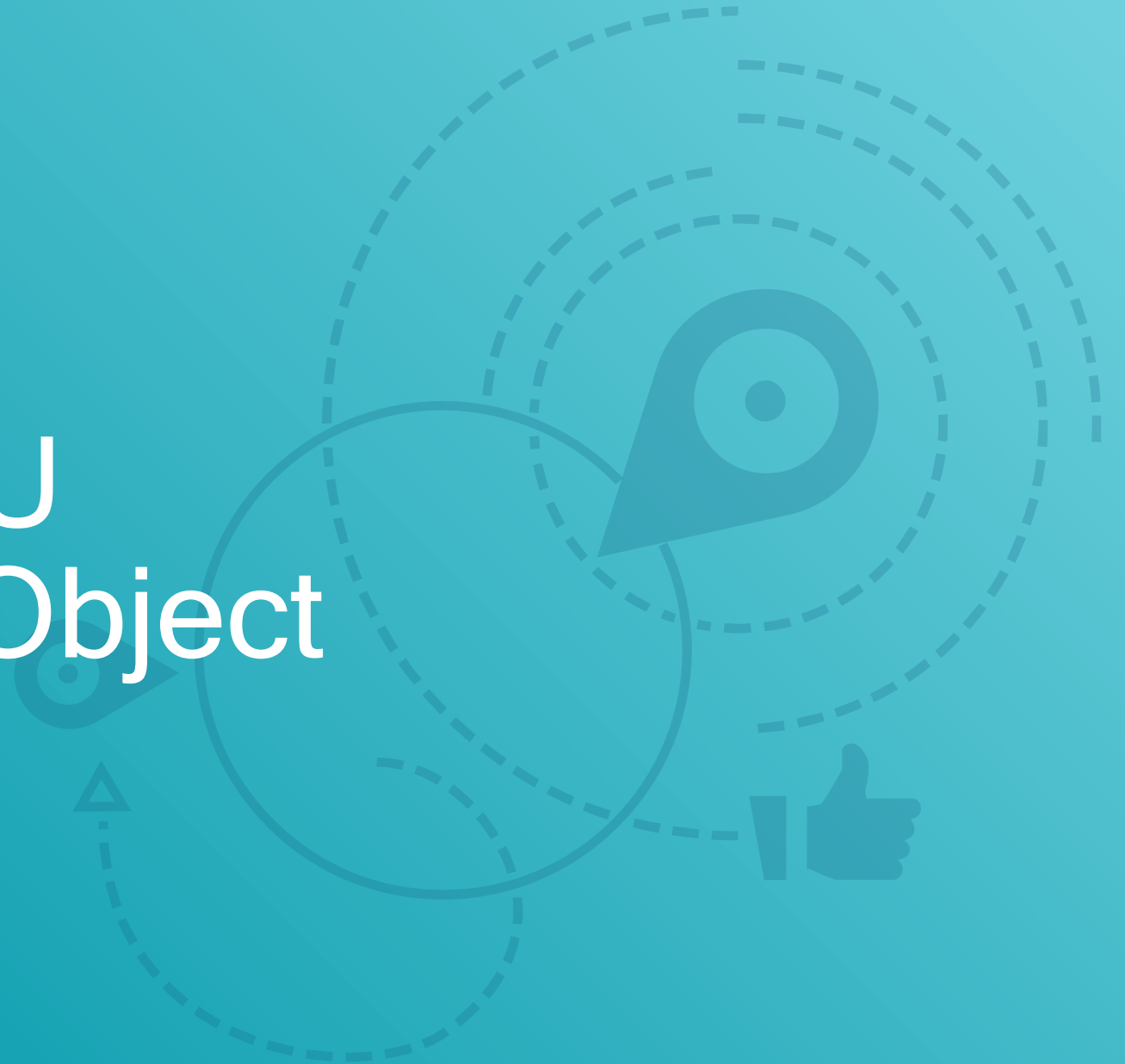




2016 Seoul DevU Pipeline Cache Object

Bill Licea-Kane
Engineer, Senior Staff
Qualcomm Technologies, Inc.
2016-10-21

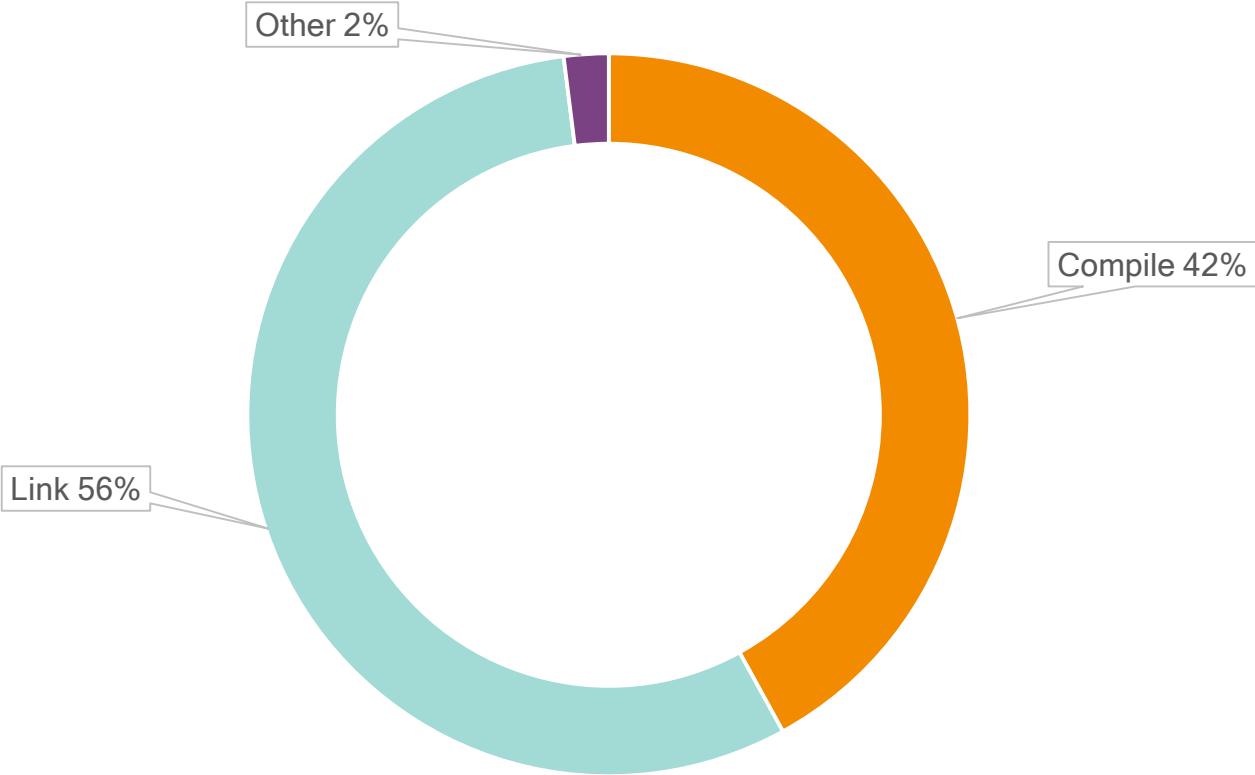


“Our application requires many many many Pipeline State Objects. Creation time is a huge issue!”

Create Pipeline State Objects Create Time Percentage

Epic Games ProtoStar

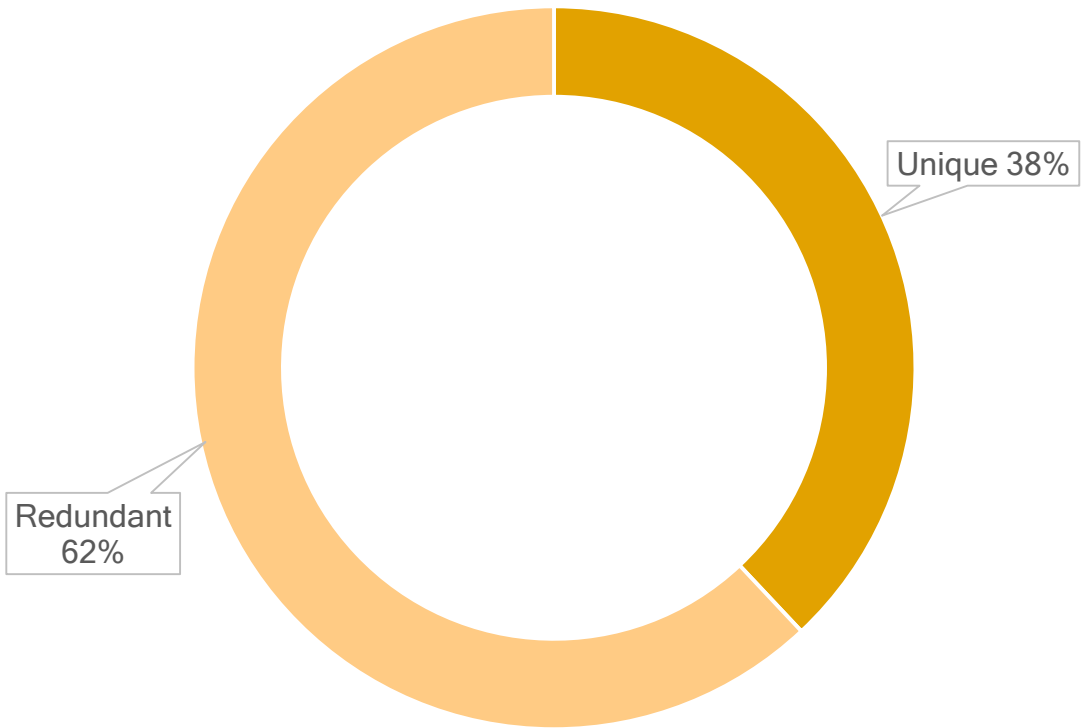
Pipeline State Objects Create Time Percentage



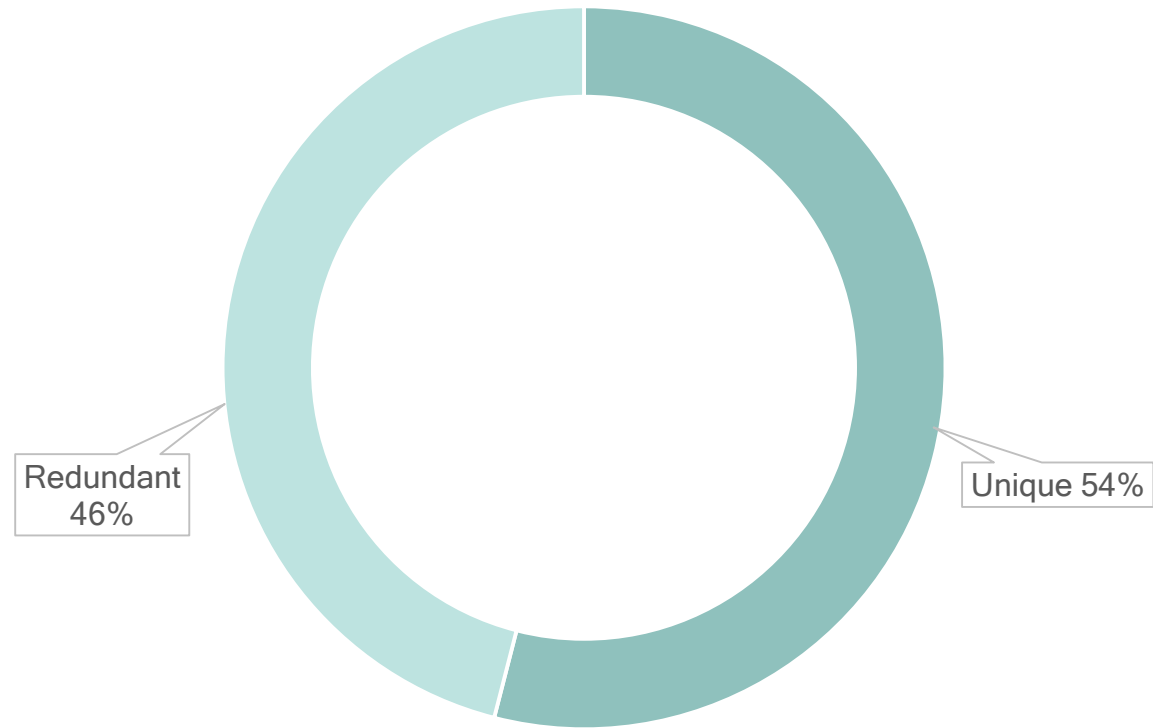
Compile and Link Redundancy

Epic Games ProtoStar

Compile



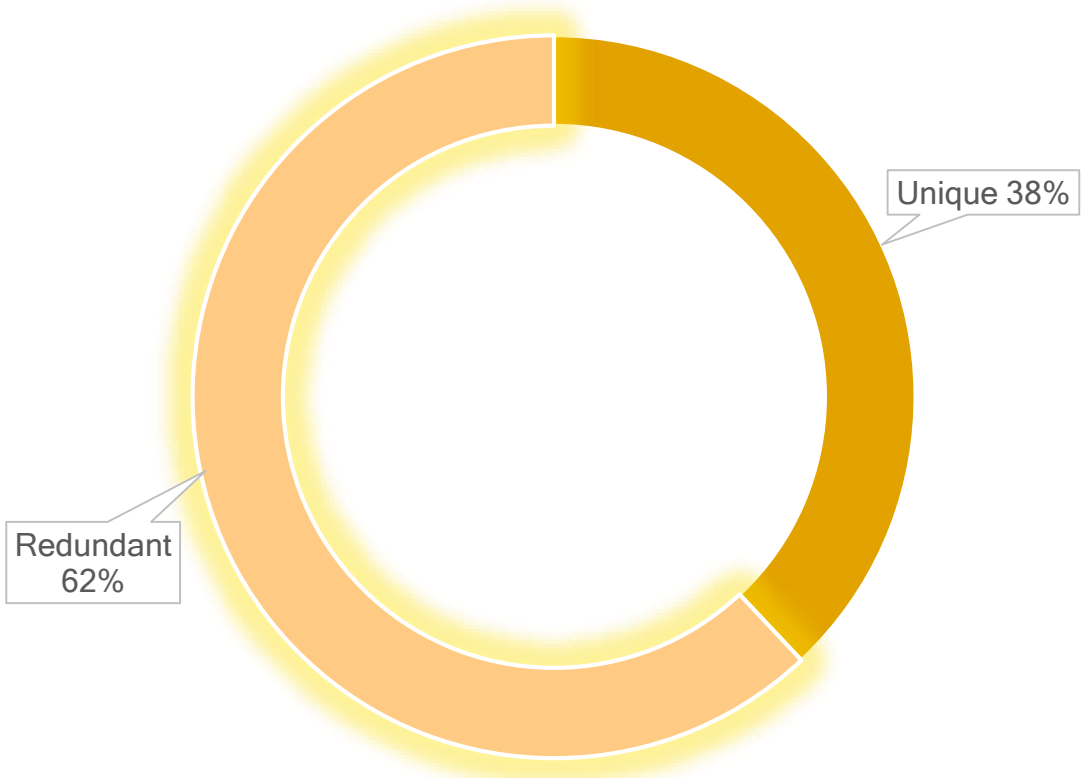
Link



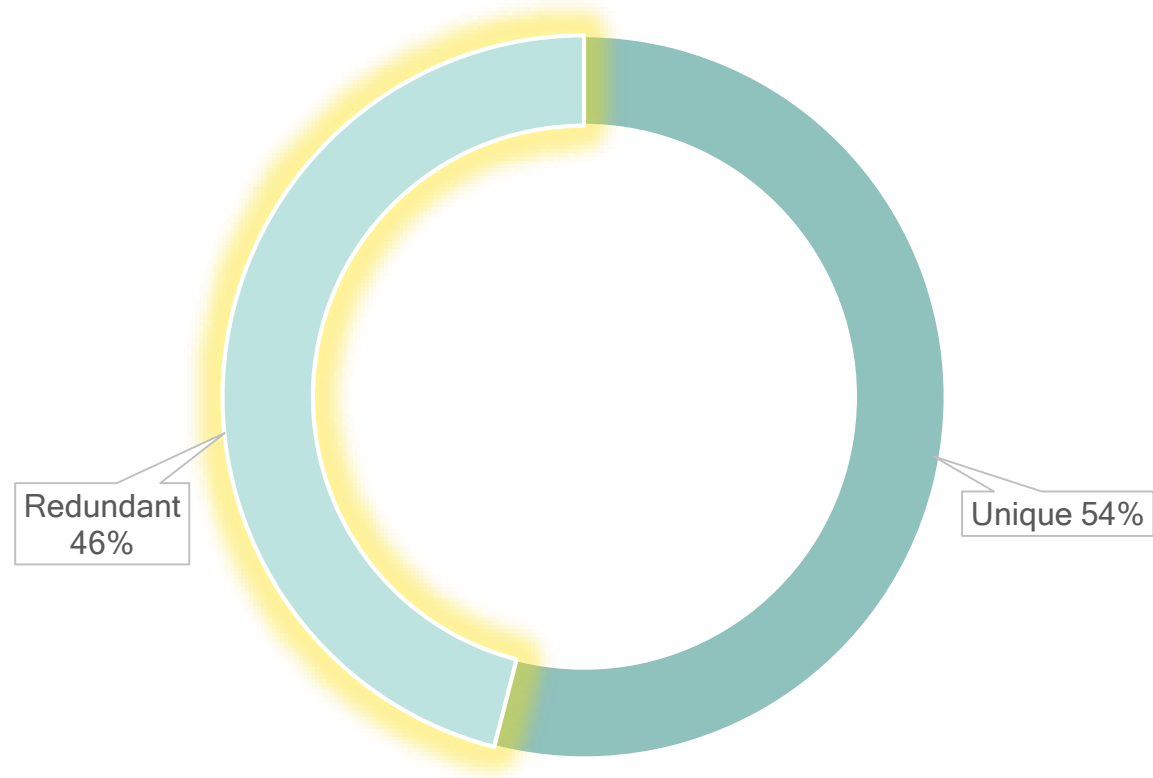
Compile and Link Redundancy

Epic Games ProtoStar

Compile



Link



Possible Solutions?

- Reduce number of Pipeline State Objects

“We can’t do much of that.

Maybe if there was more dynamic state?

Maybe if pipeline stages didn’t have to be linked together?”

Possible Solutions?

- Group together similar Pipeline State Objects

“That helps some.

But we have many many many different shaders, so it is hard to group similar pipeline state objects together.”

Possible Solutions?

- Cache prior Pipeline State Objects

“That might work. Tell me more.”

Pipeline Cache

1

Creating a
Pipeline State
Object without a
Pipeline Cache

2

Creating a
Pipeline State
Object WITH a
Pipeline Cache

3

Storing and
Loading a
Pipeline Cache

Pipeline Cache

1

Creating a
Pipeline State
Object without a
Pipeline Cache

2

Creating a
Pipeline State
Object WITH a
Pipeline Cache

3

Storing and
Loading a
Pipeline Cache

Creating a Graphics Pipeline

Without Pipeline Cache

```
vkResult result;
```

```
VkGraphicsPipelineCreateInfo graphicsPipelineCreateInfo = {};
```

```
graphicsPipelineCreateInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;  
// (set up any additional create info...)
```

```
VkPipeline pipeline;
```

```
result = vkCreateGraphicsPipelines(  
    device, // VkDevice device  
    VK_NULL_HANDLE, // VkPipelineCache pipelineCache  
    1, // uint32_t createInfoCount  
    &graphicsPipelineCreateInfo, // const VkGraphicsPipelineCreateInfo* pCreateInfos  
    nullptr, // const VkAllocationCallbacks* pAllocator  
    &pipeline); // VkPipeline* pPipelines
```

Creating a Graphics Pipeline

Without Pipeline Cache

```
vkResult result;
```

```
VkGraphicsPipelineCreateInfo graphicsPipelineCreateInfo = {};
```

```
graphicsPipelineCreateInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;  
// (set up any additional create info...)
```

```
VkPipeline pipeline;
```

```
result = vkCreateGraphicsPipelines(  
    device, // VkDevice device  
    VK_NULL_HANDLE, // VkPipelineCache pipelineCache  
    1, // uint32_t createInfoCount  
    &graphicsPipelineCreateInfo, // const VkGraphicsPipelineCreateInfo* pCreateInfos  
    nullptr, // const VkAllocationCallbacks* pAllocator  
    &pipeline); // VkPipeline* pPipelines
```

Pipeline State Objects Create Time Percentage

Epic Games ProtoStar

Pipeline State Objects Create Time Percentage (normalized to “Without Cache”)



Pipeline Cache

1

Creating a
Pipeline State
Object without a
Pipeline Cache

2

Creating a
Pipeline State
Object WITH a
Pipeline Cache

3

Storing and
Loading a
Pipeline Cache

Creating a Pipeline Cache

```
vkResult result;

static VkPipelineCache pipelineCache;
VkPipelineCacheCreateInfo pipelineCacheCreateInfo = {};

pipelineCacheCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_CACHE_CREATE_INFO;
// (set up any additional create info...)

result = vkCreatePipelineCache(
    device, // VkDevice
    &pipelineCacheCreateInfo, // const VkPipelineCacheCreateInfo*
    nullptr, // const VkAllocationCallbacks*
    &pipelineCache); // VkPipelineCache*

// continued...
```

Creating a Pipeline Cache

```
vkResult result;

static VkPipelineCache pipelineCache;
VkPipelineCacheCreateInfo pipelineCacheCreateInfo = {};

pipelineCacheCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_CACHE_CREATE_INFO;
// (set up any additional create info...)

result = vkCreatePipelineCache(
    device, // VkDevice
    &pipelineCacheCreateInfo, // const VkPipelineCacheCreateInfo*
    nullptr, // const VkAllocationCallbacks*
    &pipelineCache); // VkPipelineCache*

// continued...
```

Creating a Graphics Pipeline

WITH Pipeline Cache

```
// ...continued
```

```
vkGraphicsPipelineCreateInfo graphicsPipelineCreateInfo = {};
```

```
graphicsPipelineCreateInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;  
// (set up any additional create info...)
```

```
vkPipeline pipeline;
```

```
result = vkCreateGraphicsPipelines(  
    device,                // VkDevice                device  
    pipelineCache,        // VkPipelineCache      pipelineCache  
    1,                    // uint32_t              createInfoCount  
    &graphicsPipelineCreateInfo, // const VkGraphicsPipelineCreateInfo* pCreateInfos  
    nullptr,              // const VkAllocationCallbacks* pAllocator  
    &pipeline);           // VkPipeline*          pPipelines
```

Creating a Graphics Pipeline

WITH Pipeline Cache

```
// ...continued
```

```
VkGraphicsPipelineCreateInfo graphicsPipelineCreateInfo = {};
```

```
graphicsPipelineCreateInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;  
// (set up any additional create info...)
```

```
VkPipeline pipeline;
```

```
result = vkCreateGraphicsPipelines(  
    device, // VkDevice device  
    pipelineCache, // VkPipelineCache pipelineCache  
    1, // uint32_t createInfoCount  
    &graphicsPipelineCreateInfo, // const VkGraphicsPipelineCreateInfo* pCreateInfos  
    nullptr, // const VkAllocationCallbacks* pAllocator  
    &pipeline); // VkPipeline* pPipelines
```

Creating a Graphics Pipeline

WITH Pipeline Cache

```
// ...continued
```

```
vkGraphicsPipelineCreateInfo graphicsPipelineCreateInfo = {};
```

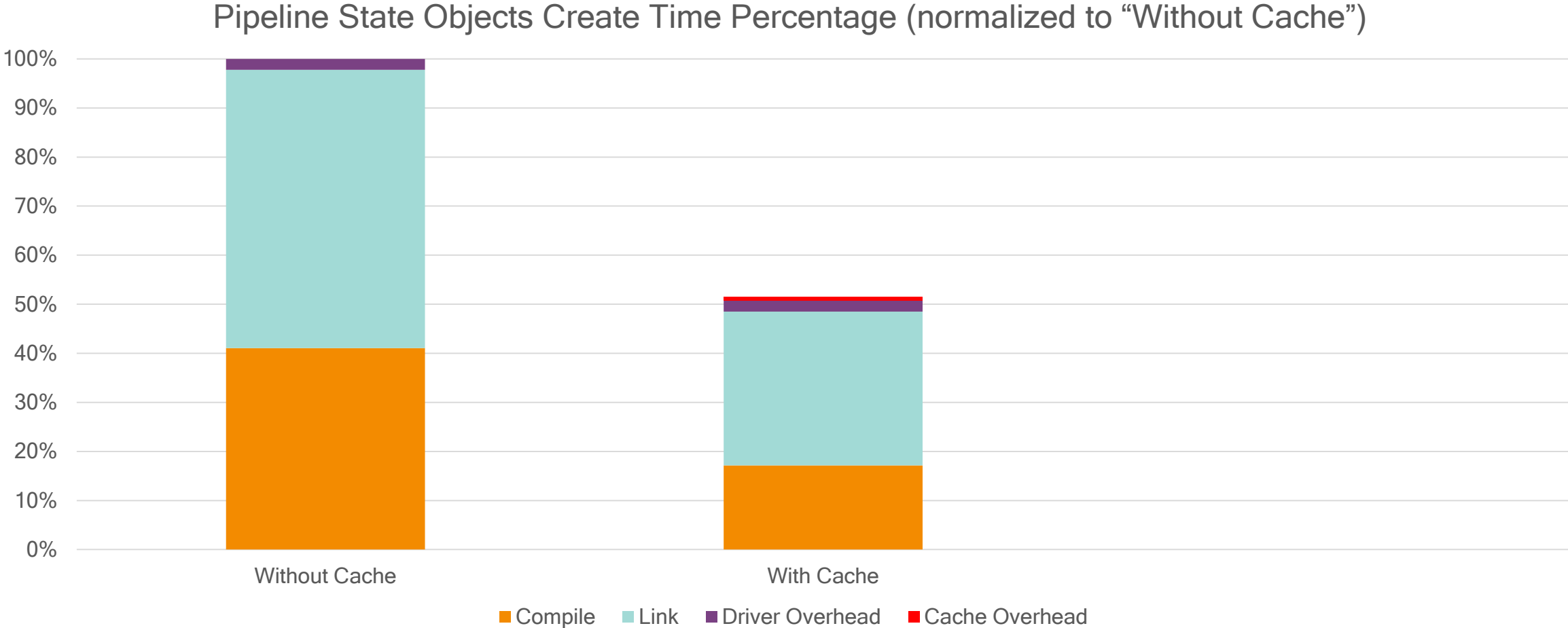
```
graphicsPipelineCreateInfo.sType = VK_STRUCTURE_TYPE_GRAPHICS_PIPELINE_CREATE_INFO;  
// (set up any additional create info...)
```

```
vkPipeline pipeline;
```

```
result = vkCreateGraphicsPipelines(  
    device,                // VkDevice                device  
    pipelineCache,        // VkPipelineCache    pipelineCache  
    1,                    // uint32_t            createInfoCount  
    &graphicsPipelineCreateInfo, // const VkGraphicsPipelineCreateInfo* pCreateInfos  
    nullptr,              // const VkAllocationCallbacks*    pAllocator  
    &pipeline);           // VkPipeline*        pPipelines
```

Pipeline State Objects Create Time Percentage

Epic Games ProtoStar



Pipeline Cache

1

Creating a
Pipeline State
Object without a
Pipeline Cache

2

Creating a
Pipeline State
Object WITH a
Pipeline Cache

3

Storing and
Loading a
Pipeline Cache

Get Pipeline Cache Data

```
vkResult result;

size_t pipelineCacheSize;
unsigned char pipelineCacheData[MAX_PIPELINE_CACHE_STORE];

result = vkGetPipelineCacheData(
    device, // VkDevice           device
    pipelineCache, // VkPipelineCache       pipelineCache
    &pipelineCacheSize, // size_t*              pDataSize
    nullptr); // void*                pData

pipelineCacheSize = min(pipelineCacheSize, MAX_PIPELINE_CACHE_STORE);

result = vkGetPipelineCacheData(
    device, // VkDevice           device
    pipelineCache, // VkPipelineCache       pipelineCache
    &pipelineCacheSize, // size_t*              pDataSize
    &pipelineCacheData[0]); // void*                pData

int storeResult = storeToStorage(pipelineCacheSize, &pipelineCacheData[0]);
```

Create Pipeline Cache

WITH initial data

```
vkResult result;

static VkPipelineCache pipelineCache;
VkPipelineCacheCreateInfo pipelineCacheCreateInfo = {};

size_t pipelineCacheSize;
unsigned char pipelineCacheData[MAX_PIPELINE_CACHE_STORE];

int loadResult = loadFromStorage(MAX_PIPELINE_CACHE_STORE, &pipelineCacheSize, &pipelineCacheData[0]);

pipelineCacheCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_CACHE_CREATE_INFO;
// (set up any additional create info...)
pipelineCacheCreateInfo.initialDataSize = pipelineCacheSize;
pipelineCacheCreateInfo.pInitialData = &pipelineCacheData[0];

result = vkCreatePipelineCache(
    device, // VkDevice
    &pipelineCacheCreateInfo, // const VkPipelineCacheCreateInfo*
    nullptr, // const VkAllocationCallbacks*
    &pipelineCache); // VkPipelineCache*

device, // device,
pCreateInfo, // pCreateInfo,
pAllocator, // pAllocator,
pPipelineCache); // pPipelineCache);
```

Create Pipeline Cache

WITH initial data

```
vkResult result;

static VkPipelineCache pipelineCache;
VkPipelineCacheCreateInfo pipelineCacheCreateInfo = {};

size_t pipelineCacheSize;
unsigned char pipelineCacheData[MAX_PIPELINE_CACHE_STORE];

int loadResult = loadFromStorage(MAX_PIPELINE_CACHE_STORE, &pipelineCacheSize, &pipelineCacheData[0]);

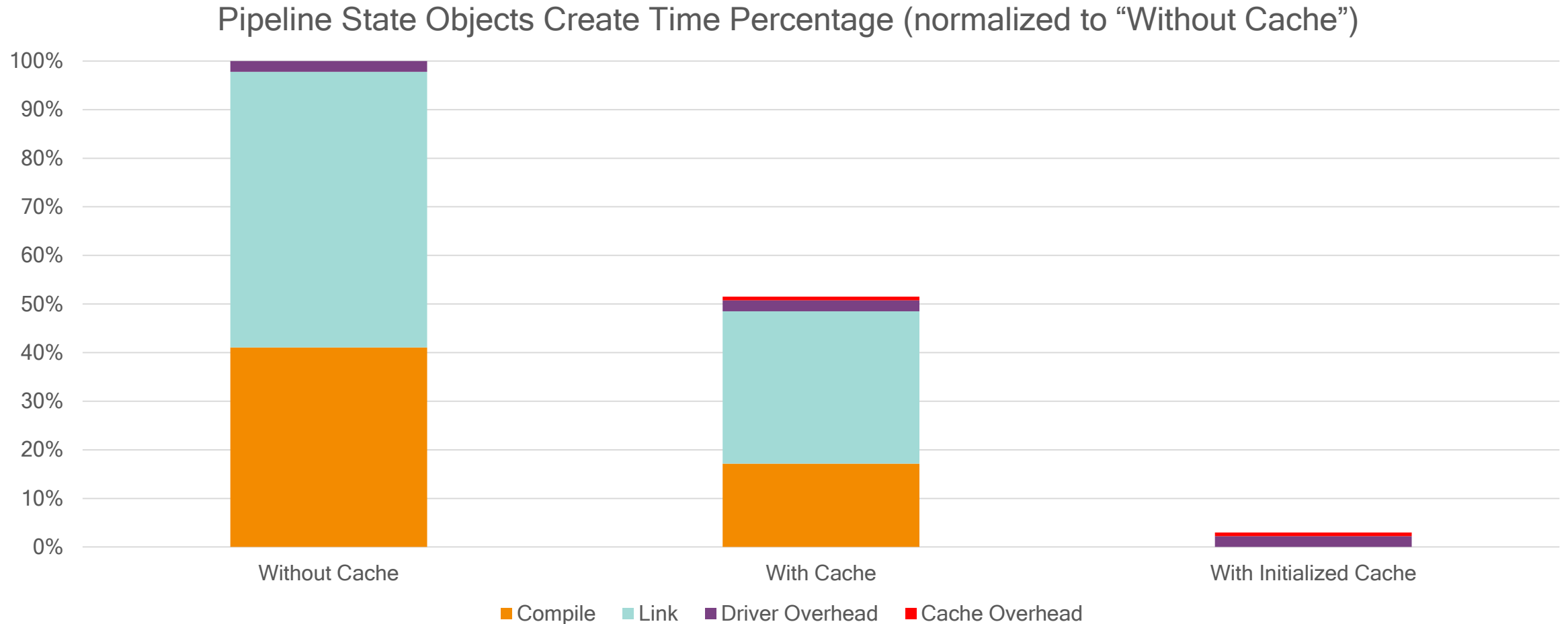
pipelineCacheCreateInfo.sType = VK_STRUCTURE_TYPE_PIPELINE_CACHE_CREATE_INFO;
// (set up any additional create info...)
pipelineCacheCreateInfo.initialDataSize = pipelineCacheSize;
pipelineCacheCreateInfo.pInitialData = &pipelineCacheData[0];

result = vkCreatePipelineCache(
    device, // VkDevice
    &pipelineCacheCreateInfo, // const VkPipelineCacheCreateInfo*
    nullptr, // const VkAllocationCallbacks*
    &pipelineCache); // VkPipelineCache*

device, // device,
pCreateInfo, // pCreateInfo,
pAllocator, // pAllocator,
pPipelineCache); // pPipelineCache);
```

Pipeline State Objects Create Time

Epic Games ProtoStar



Pipeline Cache Object is internally synchronized

Applications MAY access a pipeline cache from multiple threads without external synchronization

BUT - Host access to Pipeline Cache object MUST be externally synchronized

Pipeline Cache Header

Offset	Size	Description
0	4	length in bytes of the entire pipeline cache header written as a stream of bytes, with the least significant byte first
4	4	a <code>VkPipelineCacheHeaderVersion</code> value written as a stream of bytes, with the least significant byte first
8	4	a vendor ID equal to <code>VkPhysicalDeviceProperties::vendorID</code> written as a stream of bytes, with the least significant byte first
12	4	a device ID equal to <code>VkPhysicalDeviceProperties::deviceId</code> written as a stream of bytes, with the least significant byte first
16	<code>VK_UUID_SIZE</code>	a pipeline cache ID equal to <code>VkPhysicalDeviceProperties::pipelineCacheUUID</code>

Summary

- Using a pipeline cache is easy
- Using a pipeline cache reduces redundant compiles and links
- Saving and loading a pipeline cache may reduce compile and link to first application use
 - Local storage on device required
 - Cache may be invalidated by OS updates and/or Driver updates
- Storing a cache in the cloud is an option
 - Wi-Fi or Cellular Data required

Qualcomm[®] Snapdragon[™] SDK for Android

A product of Qualcomm Technologies, Inc.

www.qualcomm.com/software/snapdragon-sdk-android

Acknowledgements

Dan Archard
Engineer, Principal
ACG
Qualcomm Technologies, Inc.

Epic Games

Vulkan Working Group

Thank you

Follow us on:    

For more information, visit us at:

www.qualcomm.com & www.qualcomm.com/blog

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

©2016 Qualcomm Technologies, Inc. and/or its affiliated companies. All Rights Reserved.

Qualcomm and Snapdragon are trademarks of Qualcomm Incorporated, registered in the United States and other countries. Other products and brand names may be trademarks or registered trademarks of their respective owners.

References in this presentation to “Qualcomm” may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes Qualcomm’s licensing business, QTL, and the vast majority of its patent portfolio. Qualcomm Technologies, Inc., a wholly-owned subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of Qualcomm’s engineering, research and development functions, and substantially all of its product and services businesses, including its semiconductor business, QCT.

