Fine Grained Locking in the Validation Layers

Jeremy Gebben, LunarG, Inc

Understanding Fine Grained Locking

Speaker: Jeremy Gebben

Senior Graphics Software Engineer with 25 years of experience working on drivers for GPUs, high speed networking devices, and custom embedded hardware.

LUNARG

Introduction

- Fine grained locking is an optimization to improve validation of multi-threaded applications
- We will cover:
  - The effect of the optimization
  - How to enable it
  - What the optimization does
  - Current status and next steps

Lock contention w/ no validation

Sync Object / Function / Call Stack	Wait Time by Utilization ▼ I. P. C I.. O.	Wait Count	Object Type
Unknown 0xfc97a8c2	298.208s	1,706,581	Unknown
Multiple Objects	122.150s	6,620	Constant
Sleep	91.176s	9,203	Constant
IO completion	30.011s	7	Constant
Unknown 0x996ff434	29.995s	150	Unknown
Unknown 0x71f9d894	29.979s	2,843	Unknown
Unknown 0xbe9ba786	29.971s	11,010	Unknown
Unknown 0x0273f329	29.944s	13,986	Unknown
Unknown 0x44cd24fd	29.933s	7,503	Unknown
Unknown 0x58089030	29.894s	11,011	Unknown
Unknown 0x718d931f	29.119s	6	Unknown
Unknown 0xd4d28206	28.648s	11,010	Unknown
Unknown 0x1be38999	28.453s	13,665	Unknown
Unknown 0x0434dd6a	27.663s	6	Unknown
Unknown 0x4fccf9e0	18.499s	12,161	Unknown
Unknown 0x36a6cd69	12.223s	12,839	Unknown
Critical Section 0x92403f7	0.041s	511	Critical ...
Stream 0x7556b6e3	0.039s	7	Stream

- 3ms/frame
- 16 threads using Vulkan
- This is the worst case game for validation performance

LUNARG

Lock contention with validation enabled

Sync Object / Function / Call Stack	Wait Time by Utilization ▼ I. P. C I.. O	Wait Count	Object Type
▶ Unknown 0xfc97a8c2	281.026s	24,004	Unknown
▼ Read/Write Lock 0xd749cd04	182.794s	695,215	Read/...
▶ CoreChecks::WriteLock	155.941s	387,301	Read/...
▶ CoreChecks::ReadLock	26.852s	307,915	Read/...
▶ Multiple Objects	120.159s	6,602	Constant
▶ Sleep	90.662s	8,833	Constant
▶ Unknown 0x0434dd6a	59.697s	399	Unknown
▶ IO completion	30.009s	13	Constant
▶ Unknown 0xbe9ba786	30.007s	313	Unknown
▶ Unknown 0xd4d28206	30.007s	29	Unknown
▶ Unknown 0x996ff434	30.002s	150	Unknown
▶ Unknown 0x0273f329	29.994s	3,768	Unknown
▶ Unknown 0x71f9d894	29.984s	2,806	Unknown
▶ Unknown 0x58089030	29.969s	91	Unknown
▶ Unknown 0x44cd24fd	29.948s	7,500	Unknown
▶ Unknown 0x718d931f	29.226s	4	Unknown
▶ Unknown 0x1be38999	29.072s	3,768	Unknown

- 393ms/frame
- New R/W Lock waits from validation

LUNARG

Lock contention with fine grained locking

Sync Object / Function / Call Stack	Wait Time by Utilization ▼ I.. P.. O I... O..	Wait Count	Object Type
▶ Unknown 0xfc97a8c2	419.688s	37,327	Unkn...
▶ Multiple Objects	131.892s	6,603	Const...
▶ Sleep	90.843s	8,863	Const...
▶ Unknown 0x0434dd6a	55.973s	410	Unkn...
▶ IO completion	30.014s	17	Const...
▶ Unknown 0xbe9ba786	30.012s	337	Unkn...
▶ Unknown 0xd4d28206	30.009s	67	Unkn...
▶ Unknown 0x996ff434	30.004s	150	Unkn...
▶ Unknown 0x0273f329	29.997s	3,782	Unkn...
▶ Unknown 0x71f9d894	29.977s	2,833	Unkn...
▶ Unknown 0x58089030	29.963s	157	Unkn...
▶ Unknown 0x44cd24fd	29.911s	7,504	Unkn...
▶ Unknown 0x718d931f	29.072s	5	Unkn...
▶ Unknown 0x1be38999	28.986s	3,782	Unkn...
▶ Unknown 0x4fccf9e0	21.085s	1,207	Unkn...
▼ Read/Write Lock 0x1da	14.198s	594,963	Read/...
▶ BASE_NODE::AddPa	14.197s	594,935	Read/...
▶ BASE_NODE::Remo	0.002s	28	Read/...

- 191ms/frame
- Validation R/W waits have much smaller affect
- 150% improvement in framerate for many shipping games

LUNARG

How to enable

Vulkan Configurator:

Environment variable: `export VK_LAYER_FINE_GRAINED_LOCKING=1`

Vk_settings.txt: `khronos_validation.fine_grained_locking = true`

Currently only affects Core Validation

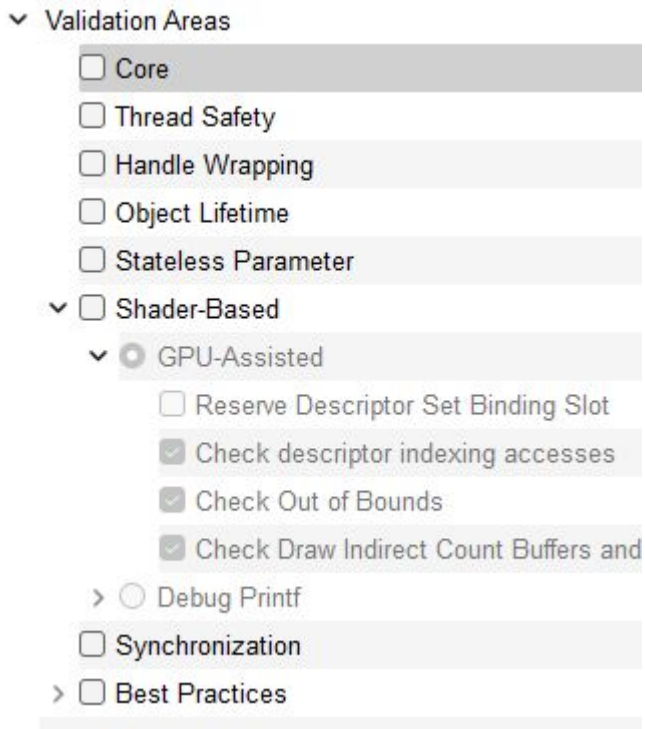
What happens in a Vulkan call

```
// Validate phase
// layer_data is the per-VkInstance or VkDevice data saved by the layer
// object_dispatch is a vector of the active ValidationObjects
for (auto intercept : layer_data->object_dispatch) {
    auto lock = intercept->ReadLock();
    skip |= intercept->PreCallValidateFoo(...)
    if (skip) return VK_ERROR_VALIDATION_FAILED_EXT;
}
// PreCallRecord phase
for (auto intercept : layer_data->object_dispatch) {
    auto lock = intercept->WriteLock();
    intercept->PreCallRecordFoo(...);
}
// call down to next layer / ICD
VkResult result = DispatchFoo(...);
// PostCallRecord phase
for (auto intercept : layer_data->object_dispatch) {
    auto lock = intercept->WriteLock();
    intercept->PostCallRecordFoo(...);
}
```

- LOTS of code in each Validate or Record method!
- `ReadLock()` and `WriteLock()` return a std::unique_lock on a `std::shared_mutex`
- Fine grained locking causes the `unique_lock` to use the `std::defer_lock` policy
- This disables the locking in this part of the code
- Other locks added to guard specific data

LUNARG

Validation Areas / Objects



- Every checkbox (except Handle Wrapping) enables a Validation Object in the layer
  - Enabling multiple is supported but is likely slow
- Thread Safety, Object Lifetime and Stateless
  - Have always used std::defer_lock
- Core
  - Enabled in SDK 1.3.204
- Best Practices, GPU-Assisted, Debug Printf, Synchronization
  - Will be enabled in a future SDK release

LUNARG

Why are there still locks at all?

- Validation cannot assume the application is correct
  - All handles validated by lookup in Device or Instance level maps (must be thread safe)
- State objects
  - Stored in thread safe maps
  - Store information for each Vulkan object, needed for validation checks
  - Reference counted with shared_ptr so that they can be used without holding the map lock
- Sometimes, the layer must update state objects when external synchronization is not required by Vulkan
- Example: VkImage layouts
  - Hardly any external synchronization requirements for VkImage
  - We track current layout for every subresource, which can be changed by many commands

Locking goals

- Thread Safety validation should be use used to make sure applications meet Vulkan external synchronization requirements
- Programs that run without errors from Thread Safety validation
  - MUST not crash
  - MUST produce the same set of errors when validation with or without fine grained locking enabled.
  - The order in which errors are output MAY change from run to run due to unpredictability of CPU scheduling of multiple threads.
- Programs that have errors from Thread Safety validation
  - SHOULD not cause crashes in the validation layer
  - MAY produce incorrect output

State object locking policies

- Immutable member data, set in constructor and never changed
  - Data members that are public and const require no locking
- Fully encapsulated and locked member data
  - Data members are private and accessors fully control locking
- Encapsulated with limited interactions with other state objects
  - As above but interactions between state objects requires care due to lock interactions
  - Example: VkQueue, VkSemaphore, and VkFence
- Public non-const data and user controlled locking (VkCommandBuffer)
  - Massive amounts of state, sometimes changes outside of external sync requirements
  - Not feasible to provide thread safe accessors without large performance impact
  - Caller is responsible for locking
  - Very fragile and hopefully will be improved

Next steps

- Implement Fine Grained locking in Best Practices, GPUAV, DebugPrintf, SyncVal
- Improve Command Buffer state object locking
- Add more tests and benchmarks to CI
- Gather feedback (via github issues). Please give this a try and let us know how it goes!
- Switch to default-on in a future SDK release
    - Will leave the ability to turn off for debugging

More information

Usage guide in Validation Layer documentation

Design document

Intel VTune profiler