TDR Introduction

Guangming Zhu

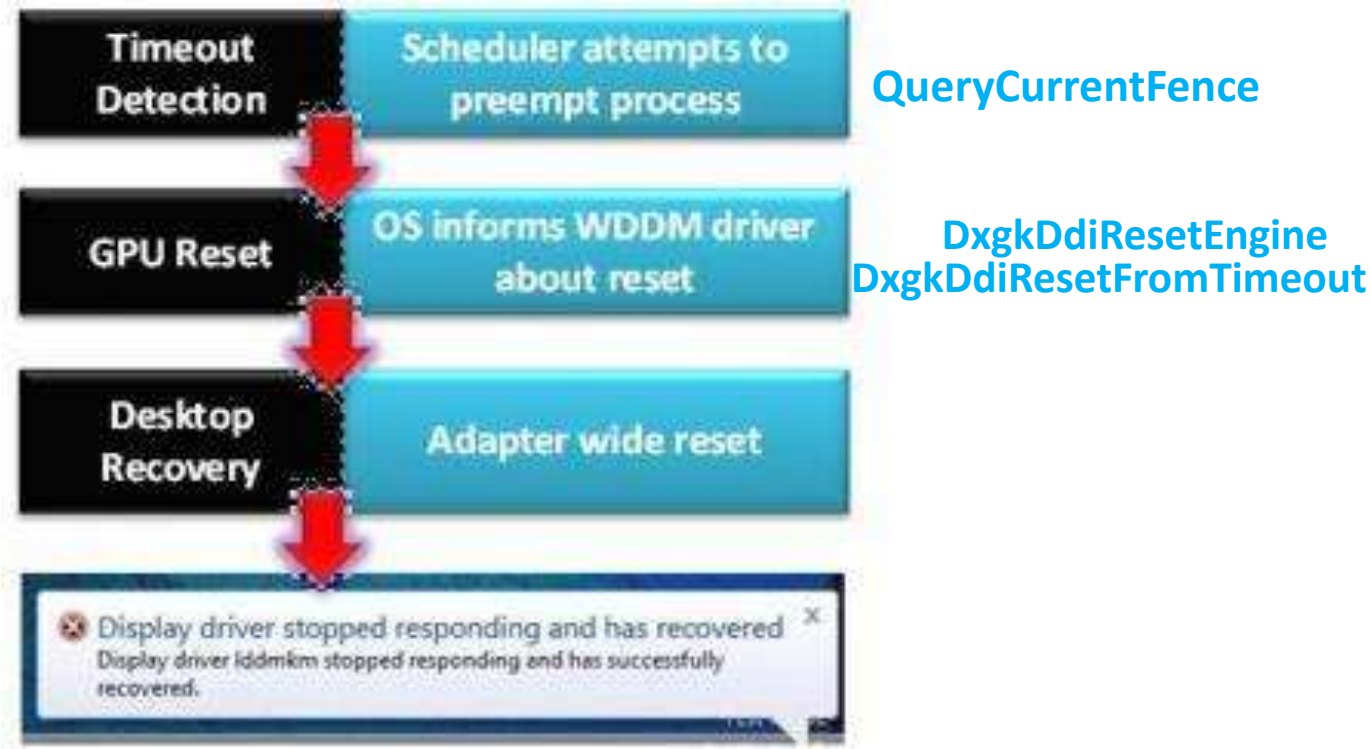
AGENDA

- *TDR (Timeout Detection Recovery)*
- TDR Triage
- TDR Debugging

TDR (Timeout Detection Recovery)

TDR (Timeout Detection Recovery)

- It appears to be completely "frozen“.



Timeout detection in WDDM

- If the GPU cannot compete or preempt the current task within the TDR timeout period, the OS diagnoses that the GPU is frozen.

- The default timeout period is 2 seconds.

- So hardware vendors should ensure that graphics operations (that is, direct memory access (DMA) buffer completion) take no more than 2 seconds in end-user scenarios as productivity and game play.

- OS will detect GPU whether hang by QueryCurrentFence.

- If last fence of submission still doesn’t return , OS identifies GPU hang.

Preparation for recovery

- When GPU hang was detected, OS will reset problem engine nodes by DxgkDdiResetEngine.

- If last fence of submission still doesn’t return , the GPU scheduler calls the display miniport driver’s DxgkDdiResetFromTimeout function to inform the driver that the OS detected a timeout.

- The driver must then reinitialize itself and reset the GPU. In addition, the driver must stop accessing memory and should not access hardware.

- The OS and the driver collect hardware and other state information that can be useful for post-recovery diagnosis.

Desktop Recovery

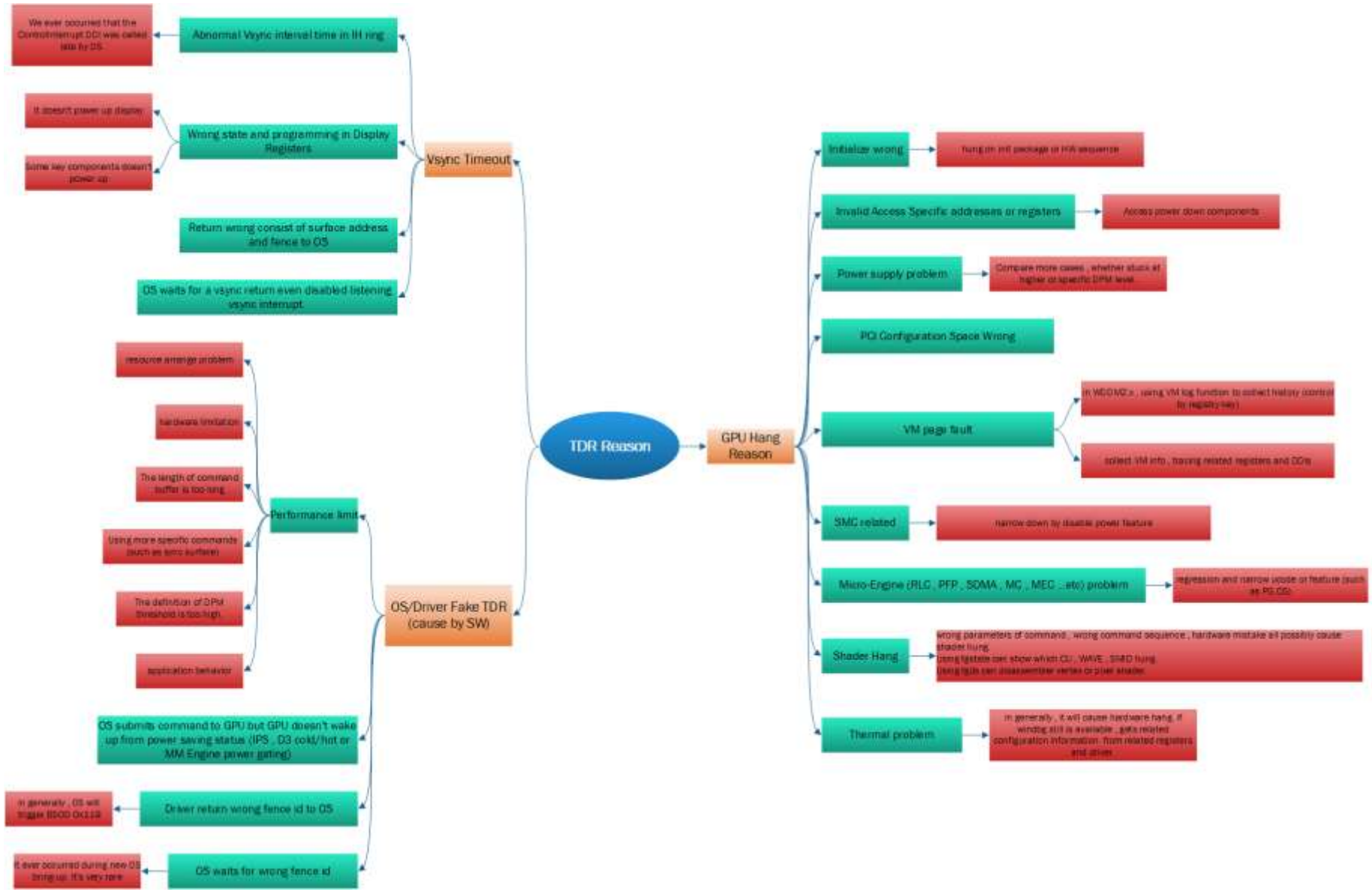
- The OS resets the appropriate state of the graphics stack.
- The video memory manager (part of Dxgkrnl.sys) purges all allocations from video memory.
- The display miniport driver resets the GPU hardware state.
- The graphics stack takes the final actions and restores the desktop to the responsive state.

VSYNC TDR & ENGINE TDR

- Vsync TDR:
  - Driver does not report present Id or address to OS, a Vsync TDR is issued indicating assumption that driver has failed to flip the frame buffer successfully.
- Engine TDR:
  - Driver does not report back fence in time. Attempt to recover through resetting the hanging engine/full ASIC. If not possible or recovery fails, engine TDR is issued.
- VSync TDR is related to display
  - If all engines are idle it means we have VSync TDR so ask DAL to debug.
  - Engine non idle get KMD and DAL involved.
- Engine TDR is related to graphics render.
  - If Graphics engines are busy (like CP) then need a scandump for further analyze.

TDR Triage

Kinds of Root Cause of TDR



Some Kinds of Root Cause of TDR

1. Fake TDR of Driver or OS

    It's not really hardware problem except performance limitation. Just software design has some wrong cause that.

2. VSync Timeout

    Any case let OS no receive Vsync notification.

3. GPU Hung

    Any block of GPU got a wrong cause hung.

1. Fake TDR of Driver or OS

- Performance limit.
  Issue is gone by enabled TDR or reduce command buffer size.

- Driver return wrong fence id to OS
  The fence of driver returned has wrong order with pervious fence.
  Checked by fence history.

- OS waits for wrong fence id
  The fence of OS waited has wrong order with pervious fence.
  Checked by fence history.

- OS submit render fence to GPU when GPU enter power saving state (IPS or D3 Cold)
  OS submit dedicated fence (non-sys paging) to GPU that entered power saving.
  Checked by m_UlpsConfig , !pci , device control class of pci filter driver
  OS Video scheduler !vidsch , !adapters

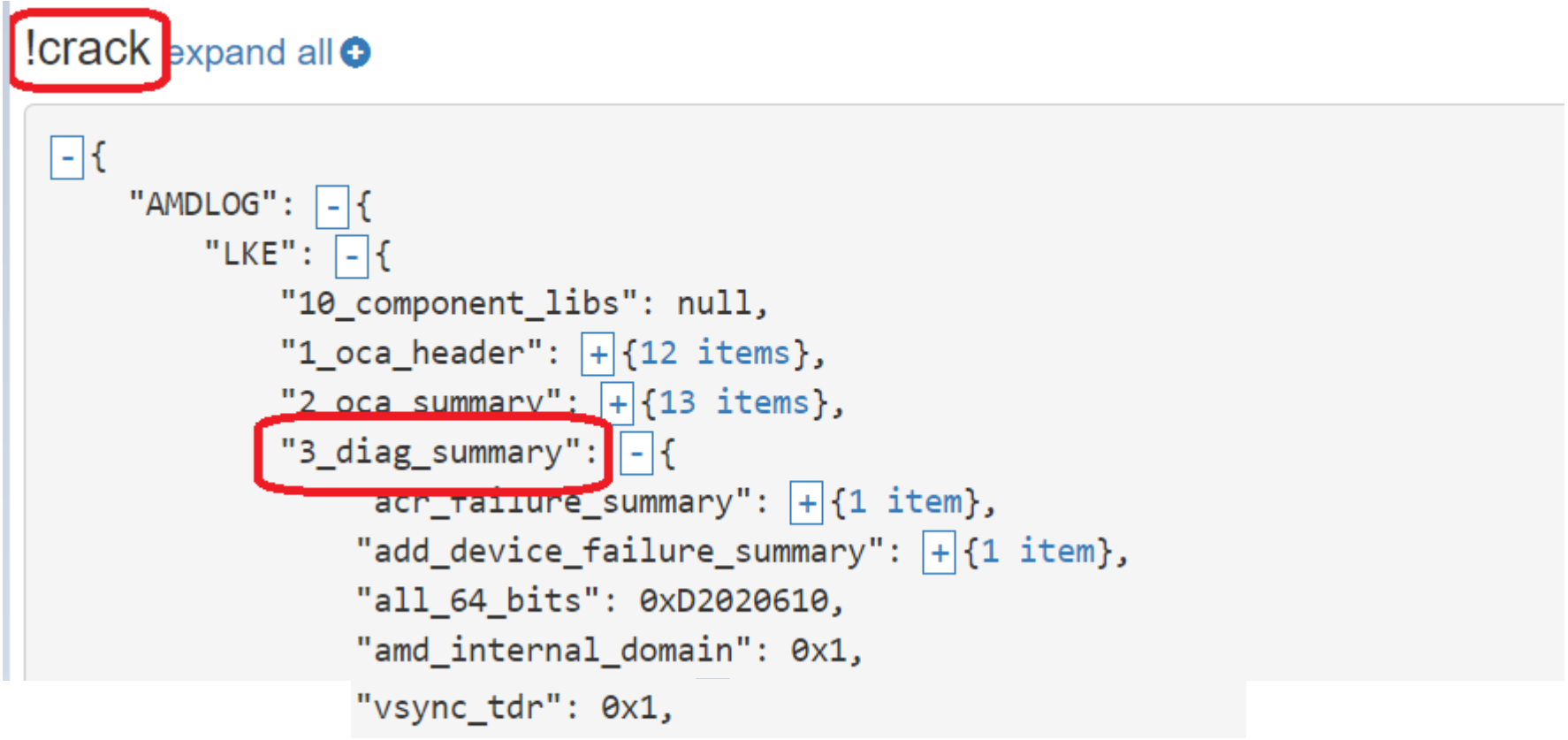
2. Vsync Timeout

- Display Registers.
  !glook crtc*
  !glook lvtma* //only for eDP
  !glook *hpd*

- Return wrong consist of surface address and fence to OS.
  Checked by VSync history.
  OS waits for a Vsync return even disabled listening VSync interrupt.
  Checked by IrqMgr virables and VSync history.
  Specific display has slow VSync in specific situation.
  Checked by VSync history and content of IH ring.

2. Vsync Timeout

- !crack -> LKE -> 3_diag_summary -> “vsync_tdr” = 0x1,
- Or “!analyze –v” -> timeoutReason



3. GPU Hung

- Initialization wrong.
  Hard to find out this kind of TDR reason.
  Usually discovered by regression test.

- Power supply problem
  In specific clock or voltage , issue is more easily duplicated.
  Issue only is duplicated on high clock or voltage.
  Do clock margin testing then measure power supply.

- VM page fault
  it's usually caused by GART's information wrong.
  GART is used for our GPU MMU for translate our GPU virtual address to CPU physical address then use DMA to access them.
  Because it's visible for CPU , so it can page out to secondary storage. That possibly causes GPU fetched wrong command and data.

3. GPU HUNG

Checked by map/unmap , resident / evict history and GART.
Related command in windbg , !glook vm* , !gdd , !mmstatus –details , !processvads , !gpuva , !paginhistory …
VM log is supported for WDDM 2.x. refer to VmConfig.cpp of KMD.

- Invalid Access Specific addresses or registers

  Accessing registers of block powered down or register offset is wrong.

3. GPU HUNG

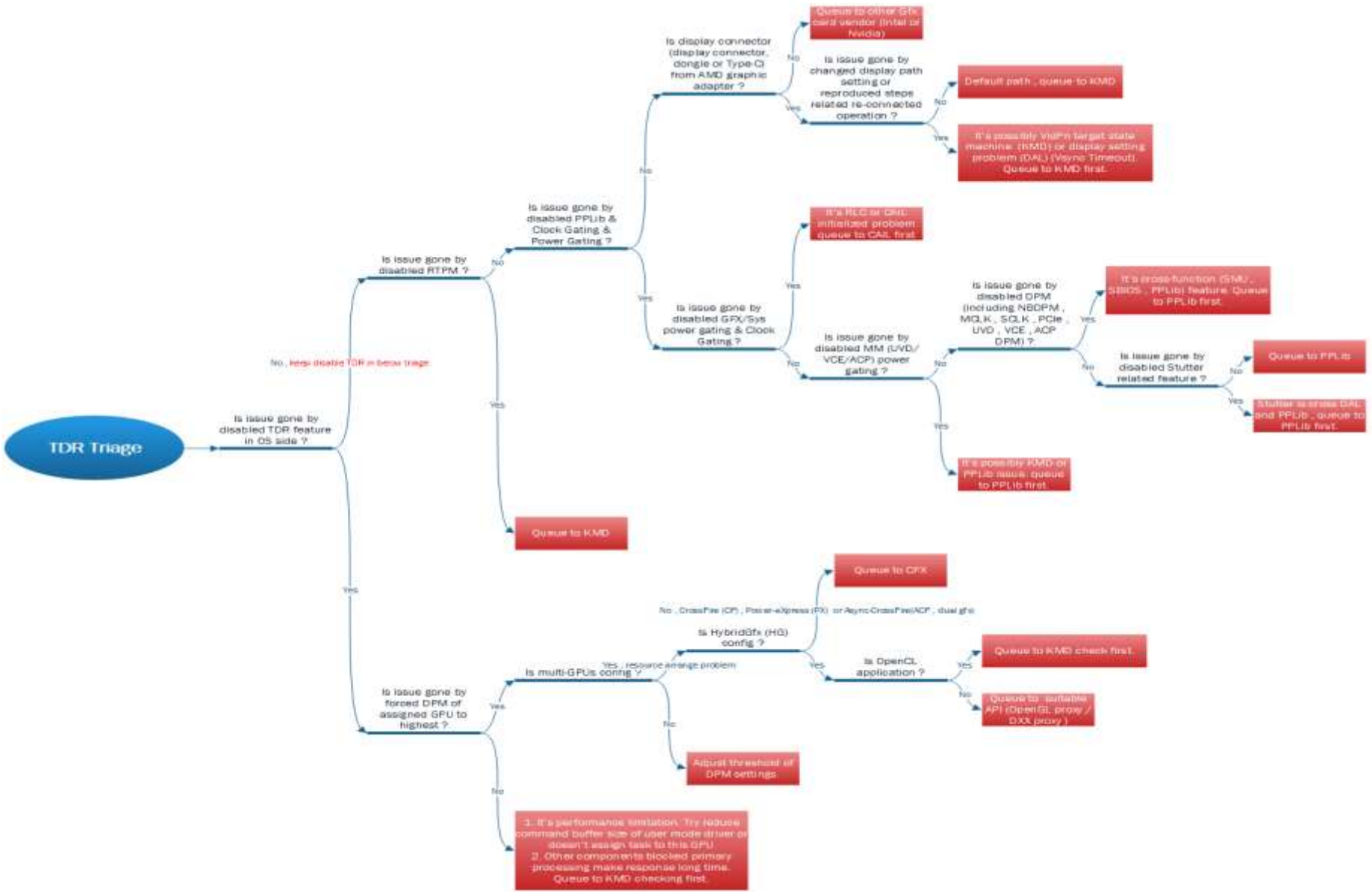
- Thermal problem
  The high temperature cause hardware unstable.
  Using physically method for prove that. It's hardly to isolate with power supplied reason.
  PPLib has several feature for avoided. You can try to adjust parameters or enabled related feature. (such as PowerTune , PowerControl , ThermalPolicy , VRHot , GPIO5 etc..)

- Shader Hang
  VM page fault usually causes this symptom.
  Rarely wrong command sequence from user mode driver. Application possibly use defer command.

Cause of Page Fault

- An error occurred when retrieving page.
- Page address could be:
  - Not mapped -> Queue to UMD which submitted fence
  - Mapped improperly
  - Address is wrong (usually 0x0)
- Could be the cause of TDR due to engine encountering page fault.

TDR Triage Steps



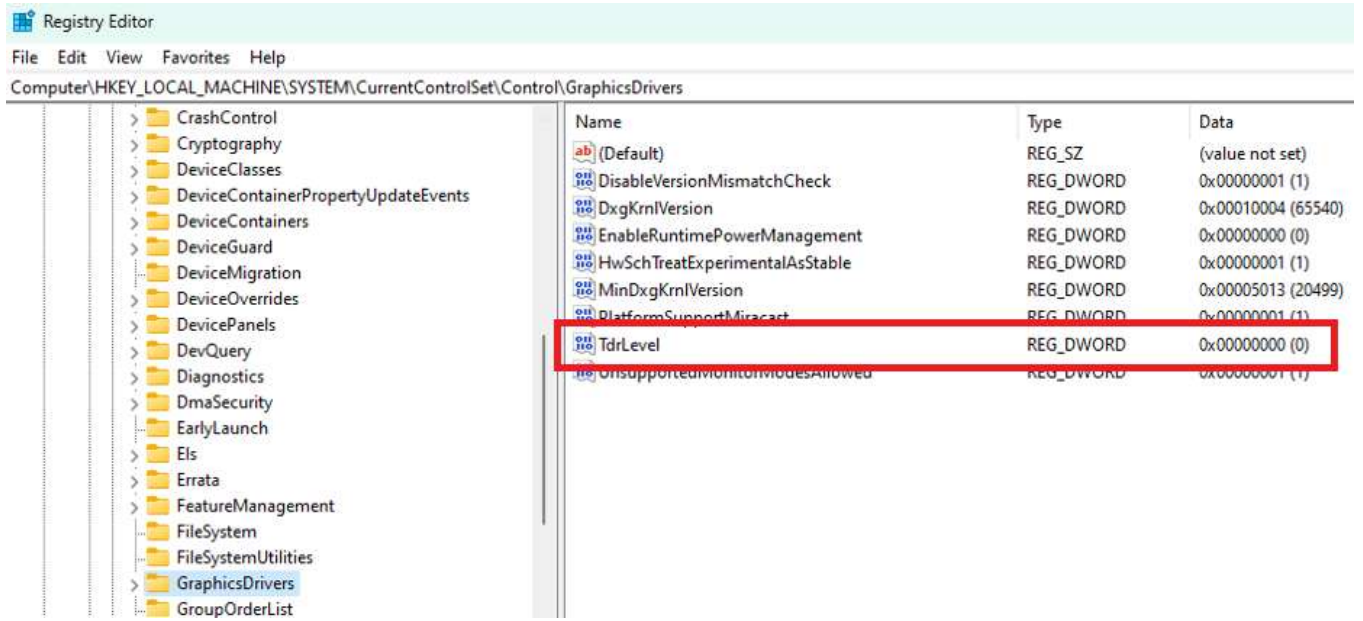
TDR Debugging

TDR Debug Steps

1. Before debugging, know why steps and operation causes this symptom. (Collection information)
   - Which Render API (OCL, OGL, DX9, DX11, DX12, etc) or Usage
   - Which Key Operations that reproduced steps.
     - S3/S4 enter or resume
     - Discrete GPU use any power saving feature (D3 cold)
     - re-connected display
     - Brightness adjust
2. Reduce range that investigating and debugging through adjusting re-procedure steps, registry keys and regression test. (Running narrow down and isolation)
3. Using Windbg connect to target device or open dump file. (Get more information)

Registry Key --- TdrLevel

- KeyPath : HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\GraphicsDrivers
- KeyValue : TdrLevel
- ValueType : REG_DWORD
- ValueData : 1



Value	Meaning
TdrLevelOff (0)	Detection disabled
TdrLevelBugcheck (1)	Bug check on detected timeout; for example, no recovery.
TdrLevelRecoverVGA (2)	Recover to VGA (not implemented).
TdrLevelRecover (3)	Recover on timeout (default value).

Registry Key --- TdrDebugMode

- KeyPath : HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\GraphicsDrivers
- KeyValue : TdrDebugMode
- ValueType : REG_DWORD
- ValueData : 0

Value	Meaning
TDR_DEBUG_MODE_OFF (0)	Break to kernel debugger before the recovery to allow investigation of the timeout.
TDR_DEBUG_MODE_IGNORE_TIMEOUT (1)	Ignore any timeout.
TDR_DEBUG_MODE_RECOVER_NO_PROMPT (2)	Recover without breaking into the debugger (default value).
TDR_DEBUG_MODE_RECOVER_UNCONDITIONAL (3)	Recover even if some recovery conditions aren't met (for example, recover on consecutive timeouts).

Dt command

```
*******************************************************************************
*                                                                             *
*                        Bugcheck Analysis                                    *
*                                                                             *
*******************************************************************************

VIDEO_TDR_FAILURE (116)
Attempt to reset the display driver and recover from timeout failed.
Arguments:
Arg1: ffff810e0bd11010, Optional pointer to internal TDR recovery context (TDR_RECOVERY_CONTEX
Arg2: fffff803`4a84b890, The pointer into responsible device driver module (e.g. owner tag).
Arg3: 0000000000000000, Optional error code (NTSTATUS) of the last failed operation.
Arg4: 0000000000000001, Optional internal context dependent data.

Debugging Details:
------------------
```

dt dxgkrnl!_TDR_RECOVERY_CONTEXT ffff810e0bd11010

```
2: kd> dt dxgkrnl!_TDR_RECOVERY_CONTEXT ffff810e0bd11010
   +0x000 Signature        : 0x52445476
   +0x008 pState           : 0xfffffe8f`09dff988 1 ( TdrStateTimeoutDetected )
   +0x010 TimeoutReason    : 6 ( TdrEngineTimeout )
   +0x018 Tick             : _ULARGE_INTEGER 0x0
   +0x020 pAdapter         : 0xffff810e`0b88a000 DXGADAPTER
   +0x028 referenceCookie  : 0xffffffff`ffffffff
   +0x030 pVidSchContext   : (null)
   +0x038 GPUTimeoutData   : _TDR_RECOVERY_GPU_DATA
   +0x050 CrtcTimeoutData  : _TDR_RECOVERY_CONTEXT::<unnamed-type-CrtcTimeoutData>
   +0x068 DbgOwnerTag      : 0xfffff803`4a84b890
   +0x070 PrivateDbgInfo   : _TDR_DEBUG_REPORT_PRIVATE_INFO
   +0xb18 pDbgReport       : 0xffff810e`0e045ae0 _WD_DEBUG_REPORT
   +0xb20 pDbgBuffer       : 0xffffaa03`0f000000 Void
   +0xb28 DbgBufferSize    : 0x2af8e
   +0xb30 pDumpBufferHelper : 0xffffaa03`0c6c2350 CTDR_DUMP_BUFFER
   +0xb38 pDbgInfoExtension : 0xffffaa03`0ebe47d0 _DXGKARG_COLLECTDBGINFO_EXT
   +0xb40 pDbgBufferUpdatePrivateInfo : 0xffffaa03`0f000140 Void
   +0xb48 ReferenceCount   : 0n1
   +0xb50 pScenarioContext : (null)
   +0xb58 pPowerThread     : (null)
   +0xb60 bReportSubmitted : 0 ''
```

Fence process

- OS calls Submit DDI(Display Driver Interface) with a Fence Id (Submitted) → Engine processed the Submission → Fence gets written to specific location in memory by the engine (Returned) → Submission is completed and an interrupt generated, the fence reported to the OS (Reported)
- Submitted: DxgkddiSubmitCommand/DxgkddiSubmitCommandVirtual
- Returned: DdiInterruptRoutine
- Reported: FenceManager::ReportCompletionFence， ProxyInterruptRoutine

```
[ff`7ddd2000] VM_CONTEXT3_PAGE_TABLE_BASE(GC) => 01`fd668001
[ff`7ddd2400] IB: 00`2e340000 # FENCE: 0000082d *?*
[ff`7ddd2800] FENCE: 00000009 *?*
10: kd> !glist /engine
All the numberS below are in HEX format
Ordinal  Scheduler Name(hub.vmid)                    *Submit* Reported Returned PreemptF
-------------------------------------------------------------------------------------
00       00        GRAPHICS(0.3)                        82d      82c      82c        8
01       1b        GRAPHICS_HIGHPRIORITY(0.d)             0        0        0        0
02       31        SPU_GPCOM                              0        0        0        0
03       1d        VIDEO_JPEG(1.b)                        0        0        0        0
04       07        DRMDMA1(0.c) *Paging*               2bce     2bce     2bce       2f
05       04        DRMDMA(0.b)                           7c       7c       7c        0
06       12        TIMER(0.3)                             0        0        0        0
07       13        PSP_UMI(1.8)                           0        0        0        0
33       34        KIQ                                    0        0        0        0
```

PresentID process (Vsync)

Submit fenceID to DAL

↓ handleInterruptVsync and NotifyInterruptCb

Flip

↓ Request to flip to a new plane with a presentID:
SetVidPnSourceAddressWithMultiPlaneOverlay3

DAL report fenceID to KMD

↓ reporting traces: ??amdkmdag!Dal3DbgExt -> vsync_int_trace

KMD report fenceID to OS

↓ ProxyDxgkcbNotifyInterrupt

OS log (!adapters -> scheduler)

Q & A

Thanks