Where Did My Disk I/O Go?

ntdebug 23 Nov 2011 12:19 PM **1**

Hello, Mr. Ninja back again. I recently discovered that although my team often tracks I/O from the file system through to the disk controller, we have never publicly documented the steps required to do this. This seems like a great opportunity for a blog because most of the structures are known, and they are even included in the public symbols.

When debugging a system that is hung, slow, or otherwise unresponsive you will likely encounter an IRP that has been sent from ntfs to the disk. Running !irp against such a request will show that the request has gone to disk.sys, but that is not really where the story ends.

Below is one such example of ntfs waiting with an IRP that appears to be stuck in disk.sys. You can determine what driver last handled the IRP by looking for the > character, this points to the current io stack location.

```
2: kd> !thread fffffa804f151040 e
THREAD fffffa804f151040  Cid 0004.33f8  Teb: 0000000000000000 Win32Thread: 0000000000000000 WAIT: (Executive) KernelMode Non-Alertable
    fffff8803836e730  NotificationEvent
IRP List:
    fffffa804f379440: (0006,0310) Flags: 00060043  Mdl: fffffa803c854090
Not impersonating
DeviceMap                 fffff8a000008720
Owning Process            fffffa8030cdeb30       Image:         System
Attached Process          N/A            Image:         N/A
Wait Start TickCount      34797397       Ticks: 1118 (0:00:00:17.440)
Context Switch Count      5893
UserTime                  00:00:00.000
KernelTime                00:00:00.296
Win32 Start Address nt!ExpWorkerThread (0xfffff80002ae2ef0)
Stack Init fffff88038370db0 Current fffff8803836e0d0
Base fffff88038371000 Limit fffff8803836b000 Call 0
Priority 16 BasePriority 13 UnusualBoost 0 ForegroundBoost 0 IoPriority 2 PagePriority 5
Child-SP          RetAddr           Call Site
fffff880`3836e110 fffff800`02addf32 nt!KiSwapContext+0x7a
fffff880`3836e250 fffff800`02ae074f nt!KiCommitThreadWait+0x1d2
fffff880`3836e2e0 fffff880`0164b3ff nt!KeWaitForSingleObject+0x19f
fffff880`3836e380 fffff880`01654224 Ntfs!NtfsNonCachedIo+0x23f
fffff880`3836e550 fffff880`0164f507 Ntfs!NtfsNonCachedUsaWrite+0x64
fffff880`3836e5e0 fffff880`016501a3 Ntfs!NtfsCommonWrite+0x2ca4
fffff880`3836e790 fffff800`02abebff Ntfs!NtfsFsdWrite+0x1c3
fffff880`3836ea10 fffff800`02b1cc00 nt!IoSynchronousPageWrite+0x24f
fffff880`3836ea90 fffff800`02b1b2d8 nt!MiFlushSectionInternal+0xb30
fffff880`3836ecc0 fffff800`02b1a83c nt!MmFlushSection+0x1f4
fffff880`3836ed80 fffff880`01653bb7 nt!CcFlushCache+0x7bc
fffff880`3836ee80 fffff880`01700037 Ntfs!LfsFlushLfcb+0x647
fffff880`3836f000 fffff880`017025b0 Ntfs!LfsFlushToLsnPriv+0x143
fffff880`3836f090 fffff880`0172445f Ntfs!LfsWriteLfsRestart+0xf0
fffff880`3836f0d0 fffff880`017242d0 Ntfs!LfsCloseLogFile+0x17f
fffff880`3836f190 fffff880`01715810 Ntfs!NtfsStopLogFile+0x70
fffff880`3836f1d0 fffff880`0172bfdb Ntfs!NtfsPerformDismountOnVcb+0x184
2: kd> !irp fffffa804f379440
Irp is active with 8 stacks 5 is current (= 0xfffffa804f379630)
 Mdl=fffffa803c854090: No System Buffer: Thread fffffa804f151040:  Irp stack trace.
     cmd  flg cl Device   File     Completion-Context
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                     Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                     Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000

                     Args: 00000000 00000000 00000000 00000000
 [  0, 0]   0  0 00000000 00000000 00000000-00000000
```

```
                        Args: 00000000 00000000 00000000 00000000
>[  4,34]  1c e0 fffffa8032052060 00000000 fffff880011bb010-fffffa803a604c90 Success Error Cancel
               \Driver\Disk     volmgr!VmpReadWriteCompletionRoutine
                        Args: 00001000 00000000 b5f8a000 00000000
 [  4, 0]   c e0 fffffa803a604b40 00000000 fffff88001cb5150-fffffa803a1ec180 Success Error Cancel
               \Driver\volmgr   volsnap!VspRefCountCompletionRoutine
                        Args: 00001000 00000000 b5e8a000 00000000
 [  4, 0]   c e1 fffffa803a1ec030 00000000 fffff8800164c344-fffff8803836e728 Success Error Cancel pending
               \Driver\volsnap  Ntfs!NtfsMasterIrpSyncCompletionRoutine
                        Args: 00001000 00000000 b5e8a000 00000000
 [  4, 0]   0  0 fffffa803d1bf030 fffffa803b268540 00000000-00000000
               \FileSystem\Ntfs
                        Args: 00001000 00000000 01d0c000 00000000
```

To learn more about what disk.sys is doing with this request we start by looking at the device extension. Disk.sys is a miniclass driver, it depends on classpnp.sys to do most of the work. The device extension will be a FUNCTIONAL_DEVICE_EXTENSION structure from classpnp.

```
2: kd> !devobj fffffa8032052060
Device object (fffffa8032052060) is for:
 DR36 \Driver\Disk DriverObject fffffa80319fa990
Current Irp 00000000 RefCount 0 Type 00000007 Flags 01002050
Vpb fffffa803204aba0 Dacl fffff9a100463450 DevExt fffffa80320521b0 DevObjExt fffffa8032052858 Dope fffffa803204ab30
ExtensionFlags (0x00000800)
                              Unknown flags 0x00000800
AttachedDevice (Upper) fffffa8032052b90 \Driver\partmgr
AttachedTo (Lower) fffffa8031dcc060 \Driver\mpio
Device queue is not busy.
2: kd> dt classpnp!_FUNCTIONAL_DEVICE_EXTENSION fffffa80320521b0
   +0x000 Version          : 3
   +0x008 DeviceObject     : 0xfffffa80`32052060 _DEVICE_OBJECT
   +0x000 CommonExtension  : _COMMON_DEVICE_EXTENSION
   +0x200 LowerPdo         : 0xfffffa80`31dcc060 _DEVICE_OBJECT
   +0x208 DeviceDescriptor : 0xfffffa80`320afeb0 _STORAGE_DEVICE_DESCRIPTOR
   +0x210 AdapterDescriptor : 0xfffffa80`32043910 _STORAGE_ADAPTER_DESCRIPTOR
   +0x218 DevicePowerState : 1 ( PowerDeviceD0 )
   +0x21c DMByteSkew       : 0
   +0x220 DMSkew           : 0
   +0x224 DMActive         : 0 ''
   +0x228 DiskGeometry     : _DISK_GEOMETRY
   +0x240 SenseData        : 0xfffffa80`320a65c0 _SENSE_DATA
   +0x248 TimeOutValue     : 0x3c
   +0x24c DeviceNumber     : 0x24
   +0x250 SrbFlags         : 0x200102
   +0x254 ErrorCount       : 0
   +0x258 LockCount        : 0n1
   +0x25c ProtectedLockCount : 0n0
   +0x260 InternalLockCount : 0n0
   +0x268 EjectSynchronizationEvent : _KEVENT
   +0x280 DeviceFlags      : 4
   +0x282 SectorShift      : 0x9 ''
   +0x283 CdbForceUnitAccess : 0 ''
   +0x288 MediaChangeDetectionInfo : (null)
   +0x290 Unused1          : (null)
   +0x298 Unused2          : (null)
   +0x2a0 KernelModeMcnContext : _FILE_OBJECT_EXTENSION
   +0x2b8 MediaChangeCount : 6
   +0x2c0 DeviceDirectory  : 0xffffffff`800003cc Void
   +0x2c8 ReleaseQueueSpinLock : 0
   +0x2d0 ReleaseQueueIrp  : (null)
   +0x2d8 ReleaseQueueSrb  : _SCSI_REQUEST_BLOCK
   +0x330 ReleaseQueueNeeded : 0 ''
   +0x331 ReleaseQueueInProgress : 0 ''
   +0x332 ReleaseQueueIrpFromPool : 0 ''
   +0x333 FailurePredicted : 0 ''
   +0x334 FailureReason    : 0
   +0x338 FailurePredictionInfo : (null)
   +0x340 PowerDownInProgress : 0 ''
   +0x344 EnumerationInterlock : 0
   +0x348 ChildLock        : _KEVENT
   +0x360 ChildLockOwner   : (null)
   +0x368 ChildLockAcquisitionCount : 0
   +0x36c ScanForSpecialFlags : 0
   +0x370 PowerRetryDpc    : _KDPC
   +0x3b0 PowerRetryTimer  : _KTIMER
   +0x3f0 PowerContext     : _CLASS_POWER_CONTEXT
   +0x478 PrivateFdoData   : 0xfffffa80`320bc010 _CLASS_PRIVATE_FDO_DATA
   +0x480 Reserved2        : 0
   +0x488 Reserved3        : 0
   +0x490 Reserved4        : 0
```

The information about requests is stored in the PrivateFdoData .

```
2: kd> dt 0xfffffa80`320bc010 _CLASS_PRIVATE_FDO_DATA
CLASSPNP!_CLASS_PRIVATE_FDO_DATA
   +0x000 SqmData          : 0x62a05
   +0x008 TrackingFlags    : 0
   +0x00c UpdateDiskPropertiesWorkItemActive : 0
   +0x010 LocalMinWorkingSetTransferPackets : 0x200
   +0x014 LocalMaxWorkingSetTransferPackets : 0x800
   +0x018 AllFdosListEntry : _LIST_ENTRY [ 0xfffffa80`320be028 - 0xfffffa80`320b8028 ]
   +0x028 Perf             : <unnamed-tag>
   +0x038 HackFlags        : 0
   +0x040 HotplugInfo      : _STORAGE_HOTPLUG_INFO
   +0x048 Retry            : <unnamed-tag>
```

```
   +0x0f0 TimerInitialized : 0 ''
   +0x0f1 LoggedTURFailureSinceLastIO : 0 ''
   +0x0f2 LoggedSYNCFailure : 0 ''
   +0x0f3 ReleaseQueueIrpAllocated : 0x1 ''
   +0x0f8 ReleaseQueueIrp  : 0xfffffa80`320bcc40 _IRP
   +0x100 AllTransferPacketsList : _LIST_ENTRY [ 0xfffffa80`320bbe60 - 0xfffffa80`4ed53d10 ]
   +0x110 FreeTransferPacketsList : _SLIST_HEADER
   +0x120 NumFreeTransferPackets : 0xff
   +0x124 NumTotalTransferPackets : 0x100
   +0x128 DbgPeakNumTransferPackets : 0x100
   +0x130 DeferredClientIrpList : _LIST_ENTRY [ 0xfffffa80`320bc140 - 0xfffffa80`320bc140 ]
   +0x140 HwMaxXferLen     : 0x80000
   +0x148 SrbTemplate      : _SCSI_REQUEST_BLOCK
   +0x1a0 SpinLock         : 0
   +0x1a8 LastKnownDriveCapacityData : _READ_CAPACITY_DATA_EX
   +0x1b4 IsCachedDriveCapDataValid : 0x1 ''
   +0x1b8 ErrorLogNextIndex : 6
   +0x1c0 ErrorLogs        : [16] _CLASS_ERROR_LOG_DATA
   +0x9c0 NumHighPriorityPagingIo : 0
   +0x9c4 MaxInterleavedNormalIo : 0
   +0x9c8 ThrottleStartTime : _LARGE_INTEGER 0x0
   +0x9d0 ThrottleStopTime : _LARGE_INTEGER 0x0
   +0x9d8 LongestThrottlePeriod : _LARGE_INTEGER 0x0
   +0x9e0 IdlePrioritySupported : 0x1 ''
   +0x9e8 IdleListLock     : 0
   +0x9f0 IdleIrpList      : _LIST_ENTRY [ 0xfffffa80`320bca00 - 0xfffffa80`320bca00 ]
   +0xa00 IdleTimer        : _KTIMER
   +0xa40 IdleDpc          : _KDPC
   +0xa80 IdleTimerInterval : 0x19
   +0xa82 StarvationCount  : 0x14
   +0xa84 IdleTimerTicks   : 0
   +0xa88 IdleTicks        : 0
   +0xa8c IdleIoCount      : 0
   +0xa90 IdleTimerStarted : 0n0
   +0xa98 LastIoTime       : _LARGE_INTEGER 0x1cc8bde`4f571cca
   +0xaa0 ActiveIoCount    : 0n1
   +0xaa4 ActiveIdleIoCount : 0n0
   +0xaa8 InterpretSenseInfo : (null)
   +0xab0 MaxPowerOperationRetryCount : 0
   +0xab8 PowerProcessIrp  : 0xfffffa80`320bd010 _IRP
   +0xac0 PerfCounterFrequency : _LARGE_INTEGER 0x23c3c4
```

The outstanding requests are stored in the AllTransferPacketsList. Classpnp uses a transfer packet to send the request to the lower level drivers. This allows the request to be split into smaller packets if necessary, and for the request to be retried if there is a failure.

We can dump the AllTransferPacketsList with !list and then search for our irp, it will be in the OriginalIrp field of one of the transfer packets. Note that the output from dt will displayed with a `, while the output from !thread does not, so you will need to add a ` when searching through the !list output. Also, there may be multiple transfer packets with the same OriginalIrp.

```
2: kd> !list "-t classpnp!_TRANSFER_PACKET.AllPktsListEntry.Flink -e -x \"??@$extret; dt classpnp!_TRANSFER_PACKET @$extret\"
0xfffffa80`320bbe60"
…
??@$extret; dt classpnp!_TRANSFER_PACKET @$extret
unsigned int64 0xfffffa80`399ad5e0
   +0x000 AllPktsListEntry : _LIST_ENTRY [ 0xfffffa80`3bae2b40 - 0xfffffa80`3bc7cb30 ]
   +0x010 SlistEntry       : _SLIST_ENTRY
   +0x020 Irp              : 0xfffffa80`3bb71570 _IRP
   +0x028 Fdo              : 0xfffffa80`32052060 _DEVICE_OBJECT
   +0x030 OriginalIrp      : 0xfffffa80`4f379440 _IRP
   +0x038 CompleteOriginalIrpWhenLastPacketCompletes : 0x1 ''
   +0x03c NumRetries       : 8
   +0x040 RetryTimer       : _KTIMER
   +0x080 RetryTimerDPC    : _KDPC
   +0x0c0 RetryIn100nsUnits : 0n0
   +0x0c8 SyncEventPtr     : (null)
   +0x0d0 DriverUsesStartIO : 0 ''
   +0x0d1 InLowMemRetry    : 0 ''
   +0x0d8 LowMemRetry_remainingBufPtr : (null)
   +0x0e0 LowMemRetry_remainingBufLen : 0
   +0x0e8 LowMemRetry_nextChunkTargetLocation : _LARGE_INTEGER 0x0
   +0x0f0 BufPtrCopy       : 0xfffffa80`40d79000  "RCRD("
   +0x0f8 BufLenCopy       : 0x1000
   +0x100 TargetLocationCopy : _LARGE_INTEGER 0xb5f8a000
   +0x108 SrbErrorSenseData : _SENSE_DATA
   +0x120 Srb              : _SCSI_REQUEST_BLOCK
   +0x178 UsePartialMdl    : 0 ''
   +0x180 PartialMdl       : 0xfffffa80`3bfda010 _MDL
   +0x188 RetryHistory     : (null)
   +0x190 RequestStartTime : 0
…
```

Now we can view the irp that classpnp sent to the lower level drivers and determine what it is doing.

```
2: kd> !irp 0xfffffa80`3bb71570
Irp is active with 3 stacks 3 is current (= 0xfffffa80`3bb71688)
 Mdl=fffffa803c854090: No System Buffer: Thread 00000000:  Irp stack trace.  Pending has been returned
     cmd  flg cl Device   File     Completion-Context
 [  0, 0]   0  2 00000000 00000000 00000000-00000000

                     Args: 00000000 00000000 00000000 ffffffffc0000185
>[  f, 0]  1c  0 fffffa8031dcc060 00000000 fffff8800107d1a0-fffffa80413ec4c0
               \Driver\elxstor  mpio!MPIOPdoCompletion
```

```
                        Args: fffffa80399ad700 00000000 00000000 fffffa80413ec4c0
[  f, 0]  1c e1 fffffa8031dcc060 00000000 fffff88001d61a00-fffffa80399ad5e0 Success Error Cancel pending
              \Driver\mpio     CLASSPNP!TransferPktComplete
                        Args: fffffa80399ad700 00000000 00000000 fffffa80413ec4c0
```

We can see that the request has been sent to the disk driver. More specifically the request has been sent to the storport miniport driver elxstor. From this data we can usually assume that the request has been sent to the disk drive and we are waiting for the disk to respond. There may be conditions where the request is stuck in storport, or in the miniport, however those conditions are beyond the scope of this article.

As you can see, there are several drivers between the disk.sys mini class driver and the actual physical disk drive. It is often necessary to determine how far down the storage driver stack a request has been before you can determine where it is stuck.

Comments

GoodOne
23 Nov 2011 9:21 PM

Great! I face issues about IRP pending in Disk driver stack. This article will help me to search where it stuck.

Thanks again!