

Re: Cache coherency issues using AllocateCommonBuffer(..)

Source:

<http://www.tech-archive.net/Archive/Development/microsoft.public.development.device.drivers/2006-03/msg00696.html>

- *From:* "Eliyas Yakub [MSFT]" <elijasy@xxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Fri, 24 Mar 2006 07:43:06 -0800
-

The common buffer DMA APIs provided by NT on x86, x64, and ia64 platforms assume that DMA is coherent with the processor caches. Non-coherent DMA is something that MS folks have consistently tried to discourage for many years now. I will get the AllocateCommonBuffer doc fixed to talk about lack of support for non-coherent DMA.

You can workaround either by calling into Mm directly as you have already done or use MDLs rather than a common buffer, apply the appropriate noncached attribute to the pages locked into the MDLs, and then use the standard non-common buffer based DMA APIs.

Mapping the buffer noncached into user mode might fail because Mm detects the conflicting cached mapping generated by AllocateCommonBuffer and rejects the noncached mapping to avoid cache attribute aliasing. I have seen that failure with ZwMapViewOfSection.

-Eliyas

"Calvin Guan" <hguan@xxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:umpx98wTGHA.4772@xxxxxxxxxxxxxxxxxxxxxxxx

I can't tell why MmAllocateContiguousMemorySpecifyCache works not AllocateCommonBuffer.

Something I'd check:

1. Did you see the actual DMA write operation on the remote system?
2. What does the TLP attribute look like? Especially the NoSnoop bit?
3. How soon do you read from the host memory after you believe the DMA write is done? this could be write-posting issue. Do a read from the device register should flush the post-write data back to the host memory.

interesting issue. good luck,

--

Calvin Guan (Windows DDK MVP)

"smann" <ssm11b@xxxxxxxxxxxxx> wrote in message

Re: Cache coherency issues using AllocateCommonBuffer(..)

news:32565AE2-17A2-4EEF-BDD1-20FD87CFCAC6@xxxxxxxxxxxxxxxxxxxx

Hi,

I am seeing a cache coherency issue with memory allocated through AllocateCommonBuffer(..). This issue is occurring with a PCI Express driver.

The application I am running is in a dual host environment, where each host

(PC) can directly write to each other's main memory. This is achieved through a

special feature in the PCIe switch. Through various debugging techniques I

have verified that when (local) host correctly writes through the PCIe switch

to the physical memory of the other (remote) host. Once the data is written,

the remote host is signaled through the PCIe switch to read new data in its

buffer. I have also verified that the data read by the remote application from its shared memory buffer, is stale, i.e. cached data.

Through other debugging mechanisms I can invalidate the cache on the remote

host. When I do this the behavior of the remote host is corrected

and it reads the contents of the buffer not its cache.

The driver is using AllocateCommonBuffer from the DMA_OPERATIONS structure.

I tried other deprecated MmAllocateContiguousMemorySpecifyCache() function

and the cache coherency issue is resolved.

So I am baffled, I am almost certain I am allocating the buffer correctly w/

AllocateCommonBuffer(...). Here is a snapshot of what I am doing. In the

driver I allocate with the following call:

```
// Attempt to allocate buffer
pKernelVa =
pdx->pDmaAdapter->DmaOperations->AllocateCommonBuffer(
pdx->pDmaAdapter,
BufferSize,
&BufferLogicalAddress,
bCacheEnabled // Enable Caching for buffer?
);
```

Here the bCacheEnabled is set to FALSE. Then allocate the MDL with the following call:

Re: Cache coherency issues using AllocateCommonBuffer(..)

```
pBufferObj->pMdl =  
IoAllocateMdl(  
pKernelVa,  
BufferSize,  
FALSE, // Is this a secondary buffer?  
FALSE, // Charge quota?  
NULL // No IRP associated with MDL  
);
```

```
MmBuildMdlForNonPagedPool(  
pBufferObj->pMdl  
);
```

Next I am getting the User Virtual Address for the buffer through the following call:

```
pUserVa =  
MmMapLockedPagesSpecifyCache(  
pBufferObj->pMdl ,  
UserMode,  
CacheMode, // CacheMode == MmNonCached  
NULL,  
FALSE,  
NormalPagePriority  
);
```

I return the virtual address and the physical address to the user mode application. The Physical address of the memory remote host is passed to local host and is used for writing to the remote memory. After the remote memory is written, the remote host uses the virtual address to read data from the shared buffer.

And as I mentioned the data it reads is the data in its cache even though the buffer was allocated with the non-cachable attribute. When I ran multiple iterations of writes, I could see the data read by the remote host was from one of the previous writes.

If anyone has any idea on why the buffer being cached, I would appreciate the response?

—
smann

Re: Cache coherency issues using AllocateCommonBuffer(..)