**Trim Support in NVMe Windows Driver**

**Introduction**

This patch adds support for the SCSI UNMAP command that Storport sends down

under Windows 8 when it carries out Trim requests. Upon receiving an UNMAP

command, the driver will in turn, translate the associated SCSI UNMAP

block descriptors into NVMe Dataset Mangement (DSM) range definitions and

send those to the controller along with a DSM command with Deallocate (AD)

attribute set per section 6.6 of NVMe spec.

Storport must initially query the driver to determine if the device

supports Trim and additionally, it must learn how that support is

implemented (i.e. it is done via WRITE SAME(16) or via UNMAP, etc.).

Associated parameters must also be ascertained: it determines

for example, in the case of UNMAP, how many UNMAP block descriptors

can be sent with a single command.

All of that information is retrieved by sending SCSI Inquiry

commands for the Logical Block Provisioning VPD page and the

Block Limits VPD page. There is one other VPD page,

the Block Device Characteristics page, which provides a couple of

other pieces of related information, such as whether the device

is non-rotating (SSD).

These VPD pages are now implemented in the driver and are populated

based on the *NVM Express: SCSI Translation Reference* document.

The work here was straightforward as the SCSI translation document fully lists all of the prescribed VPD page information. Moreover, the Windows 8 WDK,Storport.h file now has all of the associated #defines and structures for these VPD pages and for the UNMAP command itself. The Windows 7 WDK Storport.h only supported VPD pages up through VPD\_SCSI\_PORTS (0x88), whereas the Windows 8 WDK version of Storport.h includes: VPD\_BLOCK\_LIMITS (0xB0), VPD\_BLOCK\_DEVICE\_CHARACTERISTICS (0xB1), and VPD\_LOGICAL\_BLOCK\_PROVISIONING (0xB2).

**Windows 8 builds and Windows 7 Compatibility**

In order to retain compatibility with the Windows 7 WDK, and to only

advertise what is actually called upon by Storport under Windows 7,

the code for the new VPD pages and UNMAP is conditionally compiled

based on whether driver is being built for Windows 8 or beyond.

A #define WINDOWS\_8 was added to nvmestd.h and code that is conditionally

compiled is wrapped as follows:

 /\*UNMAP not supported prior to Win 8\*/

 #if \_WIN32\_WINNT >= WINDOWS\_8

 case SCSIOP\_UNMAP:

 returnStatus = SntiTranslateUnmap(pSrb);

 break;

 #endif

And the number VPD pages we previously supported is conditionally

compiled in nvmeSntiTypes.h

#if \_WIN32\_WINNT >= WINDOWS\_8

#define INQ\_NUM\_SUPPORTED\_VPD\_PAGES 6

#else

#define INQ\_NUM\_SUPPORTED\_VPD\_PAGES 3

#endif

For the Windows 7 WDK, the driver builds as before, with the same supported VPD pages

and SCSI commands supported. When built for Windows 8, support for the UNMAP command

and associated VPD pages is included.

UNMAP Feature Implementation Details

Changes made to Source Files

**nvmeSnti.c**

Five new functions were added:

VOID SntiTranslateBlockLimitsPage(

 PSCSI\_REQUEST\_BLOCK pSrb

);

VOID SntiTranslateBlockDeviceCharacteristicsPage(

 PSCSI\_REQUEST\_BLOCK pSrb

);

VOID SntiTranslateLogicalBlockProvisioningPage(

 PSCSI\_REQUEST\_BLOCK pSrb,

 PNVME\_LUN\_EXTENSION pLunExt

);

SNTI\_TRANSLATION\_STATUS SntiTranslateUnmap(

 PSCSI\_REQUEST\_BLOCK pSrb

SNTI\_STATUS SntiValidateUnmapLbaAndLength(

 PNVME\_LUN\_EXTENSION pLunExt,

 PNVME\_SRB\_EXTENSION pSrbExt,

 UINT64 lba,

 UINT32 length

);

**nvmestd.h**

A new field, dsmBuffer was added to the Srb Extension. This is used to house the range definitions passed with the DSM command as part of the UNMAP processing. A total of 4K

was needed for this buffer (size of range definition 16 bytes \*max ranges 256). While consuming this amount of space in the SRB extension is not an issue, we still wanted to be conservative and were actually able to get by with adding only 2K by combining this new 2K

area with the existing 2K prpList -- the prpList buffer is not needed for the UNMAP processing (it uses PRP1 only), so this works well. The addition of dsmBuffer is heavily commented and the buffer size calculation that achieves the 4K total, serves to saliently reinforce that there is a special association.

Also added to nvmestd.h were:

#define WINDOWS\_8 0x602

and

#define PAGE\_SIZE\_IN\_DWORDS PAGE\_SIZE\_IN\_4KB / 4

**nvmeSntiTypes.h**

For supplying information in the new VPD pages, several other items were

added to nvmeSntiTypes.h as follows:

For use in SntiTranslateSupportedVpdPages:

#define BYTE\_4 4

#define BYTE\_5 5

#define BYTE\_6 6

For populating VPD pages per NVMe SCSI Translation

Reference document:

#define INQ\_RESERVED 0

#define BLOCK\_LIMITS\_PAGE\_LENGTH 0x3C

#define BLOCK\_DEVICE\_CHAR\_PAGE\_LENGTH 0x3C

#define LOGICAL\_BLOCK\_PROVISIONING\_PAGE\_LENGTH 0x04

#define MAX\_UNMAP\_BLOCK\_DESCRIPTOR\_COUNT 256

/\* Rotation rate of 1 indicates non-rotating (SSD) \*/

#define MEDIUM\_ROTATIONAL\_RATE 0x0001

#define FORM\_FACTOR\_NOT\_REPORTED 0

#define NO\_THIN\_PROVISIONING\_THRESHHOLD 0

#define WR\_SAME\_16\_TO\_UNMAP\_NOT\_SUPPORTED 0

#define WR\_SAME\_10\_TO\_UNMAP\_NOT\_SUPPORTED 0

#define ANC\_NOT\_SUPPORTED 0

#define NO\_PROVISIONING\_GROUP\_DESCRIPTOR 0

/\*

 Set the following to 1 if your NVMe controller returns zeros when LBAs

 that have been previously UNMAPED (via DSM dealloc) are read

\*/

#define ZEROS\_RETURNED\_INDICATOR 1

**nvmestd.c**

In October of 2012, a patch had been put in NVMeFindAdapter to get around an

assert that was occurring on checked builds of Windows 8. It was subsequently

removed as it was determined there may be some performance hit.

Now that we are specifically testing Windows 8 features, it has been put back here as a

convenience to avoid getting the assert right out the chute. This can be taken out again if so desired.

 #if \_WIN32\_WINNT >= WINDOWS\_8

 /\* Ensure Storport allocates the DMA adapter object \*/

 StorPortGetUncachedExtension(pAE, pPCI, 1);

 #endif