How the Streamlined Architecture of NVM Express Enables High Performance PCIe SSDs

Peter Onufryk
Director of Engineering
IDT
The Need for a Large Number of Parallel Commands

Need:
55 parallel 8KB reads
732 parallel 8KB writes
Scalable Queuing Interface

- Enables NUMA optimized drivers
  - One or more I/O submission queues, completion queue, and MSI-X interrupt per core
  - High performance and low latency command issue
  - No locking between cores

- Up to $2^{32}$ outstanding commands
  - Support for up to 64K I/O submission and completion queues
  - Each queue supports up to 64K outstanding commands
**Efficient Queuing Interface**

### Command Submission
1. Host writes command to submission queue
2. Host writes updated submission queue tail pointer to doorbell

### Command Processing
3. Controller fetches command
4. Controller processes command

### Command Completion
5. Controller writes completion to completion queue
6. Controller generates MSI-X interrupt
7. Host processes completion
8. Host writes updated completion queue head pointer to doorbell
NVMe Command Arbitration

Admin

Urgent

High Priority

Medium Priority

Low Priority

ASQ

SQ

SQ

SQ

SQ

SQ

SQ

SQ

SQ

RR

RR

RR

RR

RR

RR

WRR

Medium WRR Priority

High WRR Priority

Medium WRR Priority

Low WRR Priority

High Strict Priority

Medium Strict Priority

Low Strict Priority

Priority
Fixed Sized Commands & Completions

Submission Queue Entry (64B)

<table>
<thead>
<tr>
<th>Byte 3</th>
<th>Byte 2</th>
<th>Byte 1</th>
<th>Byte 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Identifier</td>
<td>Fuse</td>
<td>Opcode</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>D/Word</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3</td>
<td></td>
</tr>
<tr>
<td>4 5 6 7 8 9</td>
<td>Metadata Pointer</td>
</tr>
<tr>
<td>10 11 12 13 14 15</td>
<td>PRP Entry 1</td>
</tr>
<tr>
<td>16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31</td>
<td>PRP Entry 2</td>
</tr>
</tbody>
</table>

Completion Queue Entry (16B)

<table>
<thead>
<tr>
<th>Byte 3</th>
<th>Byte 2</th>
<th>Byte 1</th>
<th>Byte 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SQ Identifier</td>
<td>SQ Head Pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status Field</td>
<td>P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command Identifier</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Standard Fields Used By All Commands

Standard Fields Optionally Used By Commands
Benefit of Fixed Sized Commands

Submission Queues in PCIe Memory

Fixed Sized Commands Simplify Command Parsing, Arbitration, and Error Handling
# Simple Optimized Command Set

**Admin Commands**
- Create I/O Submission Queue
- Delete I/O Submission Queue
- Create I/O Completion Queue
- Delete I/O Completion Queue
- Get Log Page
- Identify
- Abort
- Set Features
- Get Features
- Asynchronous Event Request
- Firmware Activate (optional)
- Firmware Image Download (optional)

**NVM Admin Commands**
- Format NVM (optional)
- Security Send (optional)
- Security Receive (optional)

**NVM I/O Commands**
- Read
- Write
- Flush
- Write Uncorrectable (optional)
- Compare (optional)
- Dataset Management (optional)

---

### Summary

- **10 Required Admin Commands**
- **3 Required NVM I/O Commands**
NVM Creates New Challenges and Opportunities

- Logical to Physical Mapping
- Wear Leveling
- Storage Logical Block Address Range

Physical NAND Flash Pages

NVM Controller with Tiered Storage

- SLC NAND Flash
- MLC (2-bit) NAND Flash
- TLC NAND Flash
- Other NVM (MRAM, PCM ...)
- DRAM

Flash Translation Layer
NVMe Data Set Management Hints

Traditional Storage Command Set

NVMe Command Set with Data Set Management (DSM)
NVMe Data Set Management

Range Attributes

- Overall DSM Command
  - Deallocate
  - Integral write dataset
  - Integral read dataset

- Per DSM Range
  - Access size (in logical blocks)
  - Written in near future
  - Sequential read
  - Sequential write
  - Access latency (longer, typical, small)
  - Access frequency
    - Typical read and write
    - Infrequent read and write
    - Infrequent write, frequent read
    - Frequent write, infrequent read
    - Frequent read and write
Possible Sources of Out-Of-Order Data

- NAND or page $T_{Read}$ variation
- Target/LUN conflict
  - Operations associated with same command (e.g., multiple reads to NAND)
  - Different operation (e.g., previously issued program or erase)
- NAND error handling
  - ECC correction time variation, read-retry, …
- Flash channel conflict
Traditional Scatter Gather List (SGL)
I/O Operation and Host Memory

```
read(bufPtr, numBytes)
```

Process Virtual Memory

- C0
- C1
- C2
- C3
- C4
- C5
- C6
- C7
- C8

Host Physical Memory

- C0
- C1
- C2
- C3
- C4
- C5
- C6
- C7
- C8

Page Offset

bufPtr

numBytes
Summary

• Scalable and Efficient Queuing Interface
  ▪ Low overhead command issue and completion
  ▪ Parallel command execution
• Fixed Sized Commands
  ▪ Straightforward command fetch, parsing and arbitration
• Simple Command Set (3 required I/O commands)
  ▪ Fast command processing
• Data Set Management Hints
  ▪ Controller optimization of data placement
• Physical Region Pointers
  ▪ Simplified out-of-order data delivery