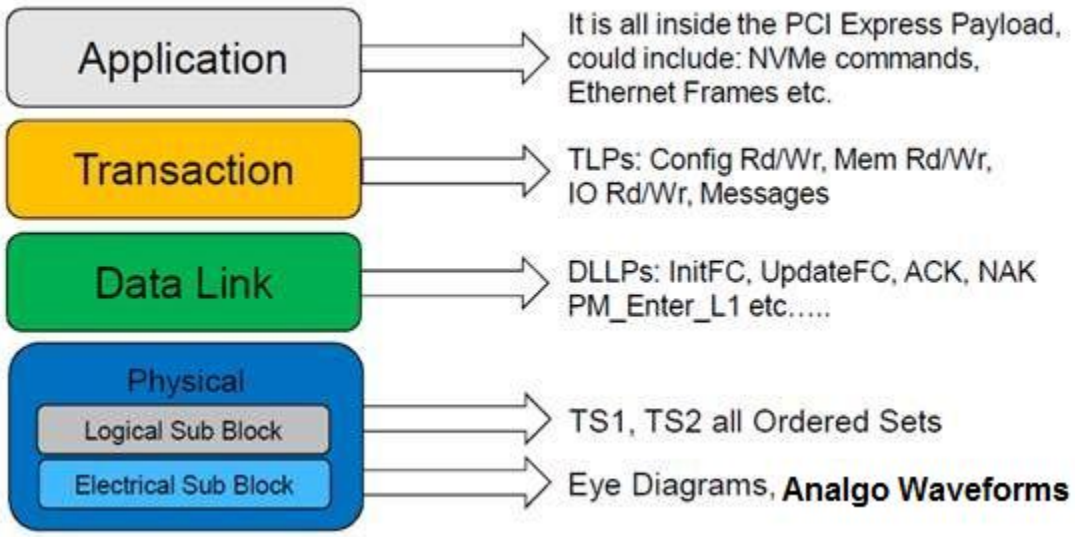
NVMe 原理简介

NVMe 是什么？

NVMe(Non-Volatile Memory express),是一种建立在 M.2 接口上的类似 AHCI 的一种协议,是专门为闪存类存储设计的协议。中文名 NVMe 协议 外文名 Non-Volatile Memory express。

NVMe 处于什么位置？

NVMe 是一种 Host 与 SSD 之间通讯的协议，它在协议栈中隶属高层



为什么需要 nvme?

- NVMe 是为 SSD 所生的。NVMe 出现之前，SSD 绝大多数走的是 AHCI 和 SATA 的协议，后者其实是为传统 HDD 服务的。与 HDD 相比，SSD 具有更低的延时和更高的性能，AHCI 已经不能跟上 SSD 性能发展的步伐了，已经成为制约 SSD 性能的瓶颈. SATA 现在最高带宽就是 600MB/s
- 可降低延迟超过 50%;
- NVMe PCIe SSD 可提供的 IOPs 十倍于高端企业级 SATA SSD;
- 自动功耗状态切换和动态能耗管理功能大大降低功耗;

命令如何执行？

NVMe 有两种命令，一种叫 Admin Command，用以 Host 管理和控制 SSD；另外一种就是 I/O Command，用以 Host 和 SSD 之间数据的传输

支持的 admin command

Figure 40: Opcodes for Admin Commands

Opcode (07) Generic Command	Opcode (06:02) Function	Opcode (01:00) Data Transfer	Opcode[2]	O/M[1]	Namespace Identifier Used[3]	Command
0b	000 00b	00b	00h	M	No	Delete I/O Submission Queue
0b	000 00b	01b	01h	M	No	Create I/O Submission Queue
0b	000 00b	10b	02h	M	Yes	Get Log Page
0b	000 01b	00b	04h	M	No	Delete I/O Completion Queue
0b	000 01b	01b	05h	M	No	Create I/O Completion Queue
0b	000 01b	10b	06h	M	Yes	Identify
0b	000 10b	00b	08h	M	No	Abort
0b	000 10b	01b	09h	M	Yes	Set Features
0b	000 10b	10b	0Ah	M	Yes	Get Features
0b	000 11b	00b	0Ch	M	No	Asynchronous Event Request
0b	000 11b	01b	0Dh	O	Yes	Namespace Management
0b	001 00b	00b	10h	O	No	Firmware Commit
0b	001 00b	01b	11h	O	No	Firmware Image Download
0b	001 01b	01b	15h	O	Yes	Namespace Attachment
	I/O Command Set Specific					
1b	na	Na	80h – BFh	O		I/O Command Set specific
	Vendor Specific					
1b	na	Na	C0h – FFh	O		Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. A subset of commands uses the Namespace Identifier field (CDW1.NSID).

支持的 io command

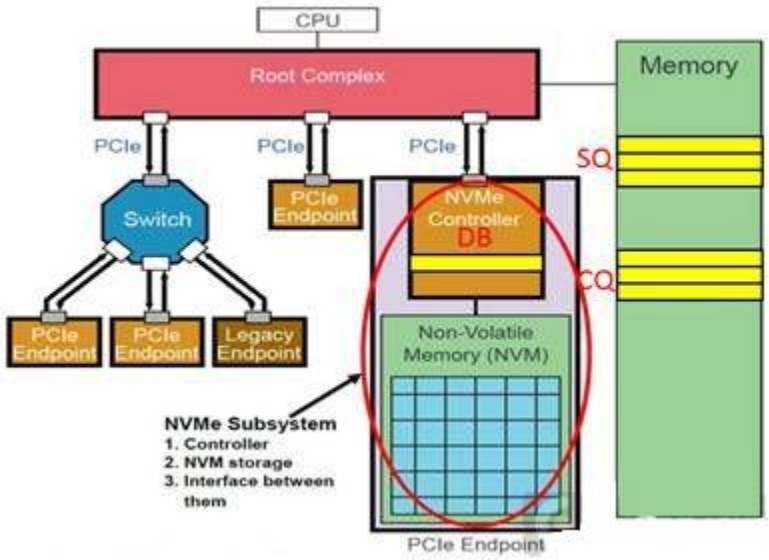
Figure 149: Opcodes for NVM Commands

Opcode (07) Standard Command	Opcode (06:02) Function	Opcode (01:00) Data Transfer	Opcode[2]	O/M[1]	Command[3]
0b	000 00b	00b	00h	M	Flush
0b	000 00b	01b	01h	M	Write
0b	000 00b	10b	02h	M	Read
0b	000 01b	00b	04h	O	Write Uncorrectable
0b	000 01b	01b	05h	O	Compare
0b	000 10b	00b	08h	O	Write Zeroes
0b	000 10b	01b	09h	O	Dataset Management
0b	000 11b	01b	0Dh	O[4]	Reservation Register
0b	000 11b	10b	0Eh	O[4]	Reservation Report
0b	001 00b	01b	11h	O[4]	Reservation Acquire
0b	001 01b	01b	15h	O[4]	Reservation Release
	Vendor Specific				
1b	na	na	80h – FFh	O	Vendor specific

NOTES:
1. O/M definition: O = Optional, M = Mandatory.
2. Opcodes not listed are reserved.
3. All NVM commands use the Namespace Identifier field (CDW1.NSID).
4. Mandatory if reservations are supported as indicated in the Identify Controller data structure.

下发命令流程

NVMe 内部：Submission Queue （SQ），Completion Queue（CQ）和 Doorbell Register （DB）。SQ 和 CQ 位于 Host 的内存中，DB 则位于 SSD 的控制器内部。

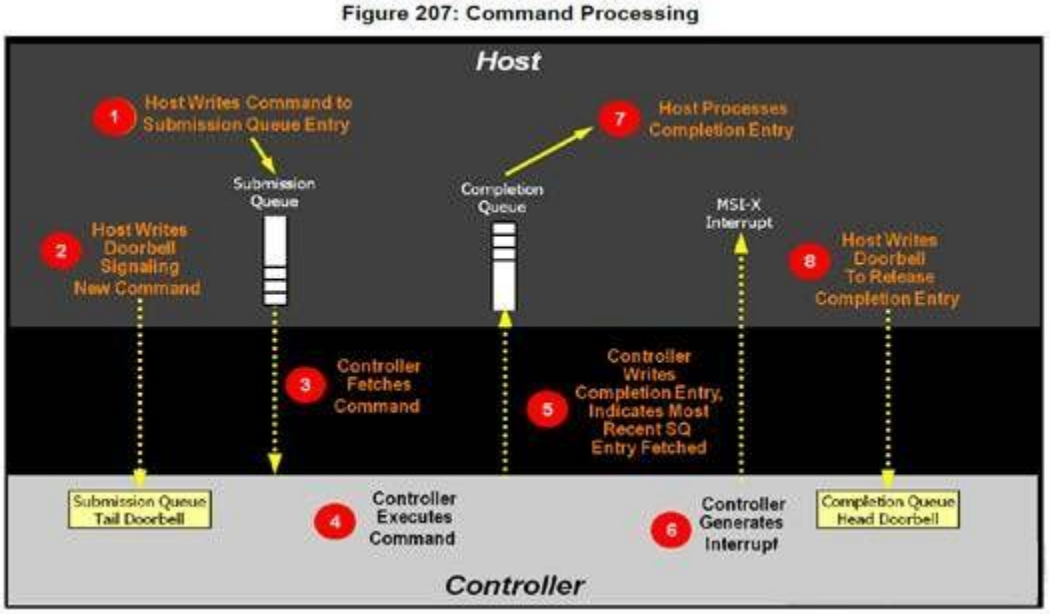


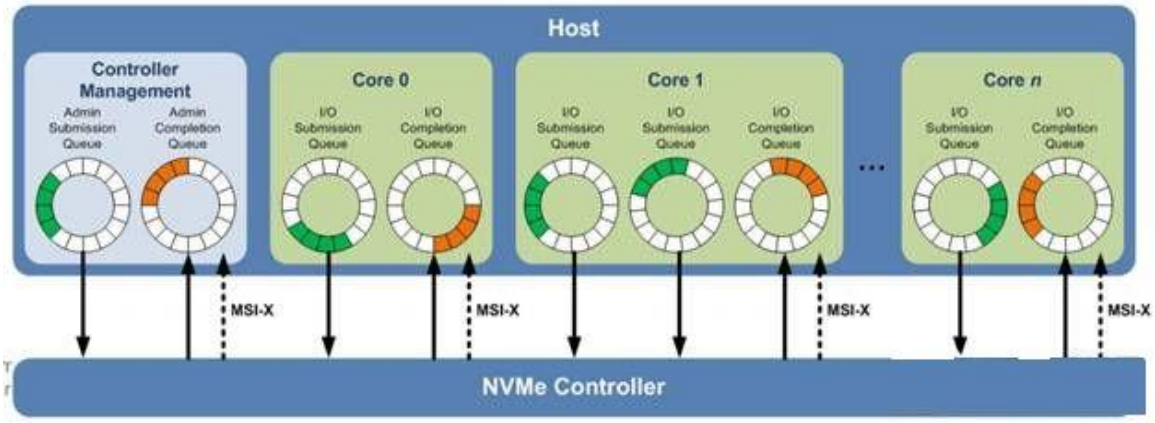
SSD 作为一个 PCIe Endpoint 通过 PCIe 连着 Root Complex （RC），然后 RC 连接着 CPU 和内存。RC 就是 CPU 的代言人

SQ 位于 Host 内存中，Host 要发送命令时，先把准备好的命令放在 SQ 中，然后通知 SSD 来取；CQ 也是位于 Host 内存中，一个命令执行完成，成功或失败，SSD 总会往 CQ 中写入命令完成状态。

Host 发送命令时，不是直接往 SSD 中发送命令的，而是把命令准备好放在自己的内存中，那怎么通知 SSD 来获取命令执行呢？Host 就是通过写 SSD 端的 DB 寄存器来告知 SSD 的：

流程步骤图





- 有两种 SQ 和 CQ，一种是 Admin，另外一种是 I/O，前者放 Admin 命令，用以 Host 管理控制 SSD，后者放置 I/O 命令，用以 Host 与 SSD 之间传输数据。
- 但系统中只有一对 Admin SQ/CQ，它们是一一对应的关系；I/O SQ/CQ 却可以很多，多达 65535（64K 减去一个 SQ/CQ）
- Host 端每个 Core 可以有一个或者多个 SQ，但只有一个 CQ。给每个 Core 分配一对 SQ/CQ 好理解，为什么一个 Core 中还要多个 SQ 呢？一是性能需求，一个 Core 中有多线程，可以做到一个线程独享一个 SQ；二是 QoS 需求，什么是 QoS？Quality of Service，服务质量。实际系统中用多少个 SQ，取决于系统配置和性能需求，可灵活设置 I/O SQ 个数
- NVMe 白皮书配置

Feature	Enterprise Recommended	Client Recommended
I/O Queues	16 to 128	2 to 8
Physically dis-contiguous queues	Design choice	No
Logical block size	4KB	4KB
Interrupt Support	MSI-X	MSI-X
Arbitration	WRR w/ Urgent or Round Robin	Round Robin
AER	Yes	Yes
Firmware Update	Required	Required
End-to-end data protection	Yes	No
SR-IOV support	Yes	No
Security Send and Receive	Yes	Yes

- 作为队列，每个 SQ 和 CQ 都有一定的深度：对 Admin SQ/CQ 来说，其深度可以是 2-4096（4K）；对 I/O SQ/CQ，深度可以是 2-65536(64K)。队列深度也是可以配置的。AHCI 只有一个命令队列，且队列深度是固定的 32
- 每条命令大小是 64 字节，每条命令完成状态是 16 字节；

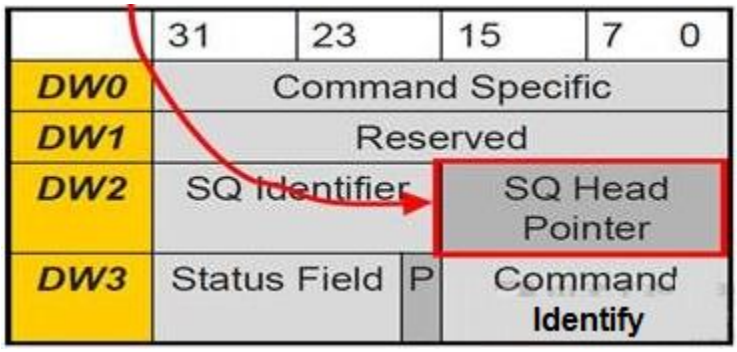
详解

DB 的另外一个作用，就是通知作用：Host 更新 SQ Tail DB 的同时，也是在告知 SSD 有新的命令需要处理；Host 更新 CQ Head DB 的同时，也是在告知 SSD，你返回的命令完成状态信息我已经处理，同时表示谢意。

这里有一个对 Host 不公平的地方，Host 对 DB 只能写，还仅限于写 SQ Tail DB 和 CQ Head DB，不能读取 DB

疑问？

1.SSD 在取指的时候，是偷偷进行的，Host 对此毫不知情。Host 发了取指通知后，它并不清楚 SSD 什么时候去取命令，取了多少命令。怎么破？



这是 SSD 往 CQ 中写入的命令完成状态信息（16 字节）。是的，SSD 往 CQ 中写入命令状态信息的同时，还把 SQ Head DB 的信息告知了 Host！！这样，Host 对 SQ 中 Head 和 Tail 的信息都有了，轻松玩转 SQ,一开始 CQ 中每条命令完成条目中的”P” bit 初始化为 0，SSD 在往 CQ 中写入命令完成条目时，会把”P”写成 1。记住一点，CQ 是在 Host 端的内存中，Host 可以检查 CQ 中的所有内容，当然包括”P”了。Host 记住上次的 Tail，然后往下一个一个检查”P”，就能得出新的 Tail 了。就是这样。

总结

DB 在 SSD Controller 端，是寄存器
DB 记录着 SQ 和 CQ 的 Head 和 Tail
每个 SQ 或者 CQ 有两个 DB: Head DB 和 Tail DB
Host 只能写 DB，不能读 DB
Host 通过 SSD 往 CQ 中写入的命令完成状态获取 Head 或者 Tail