nvme

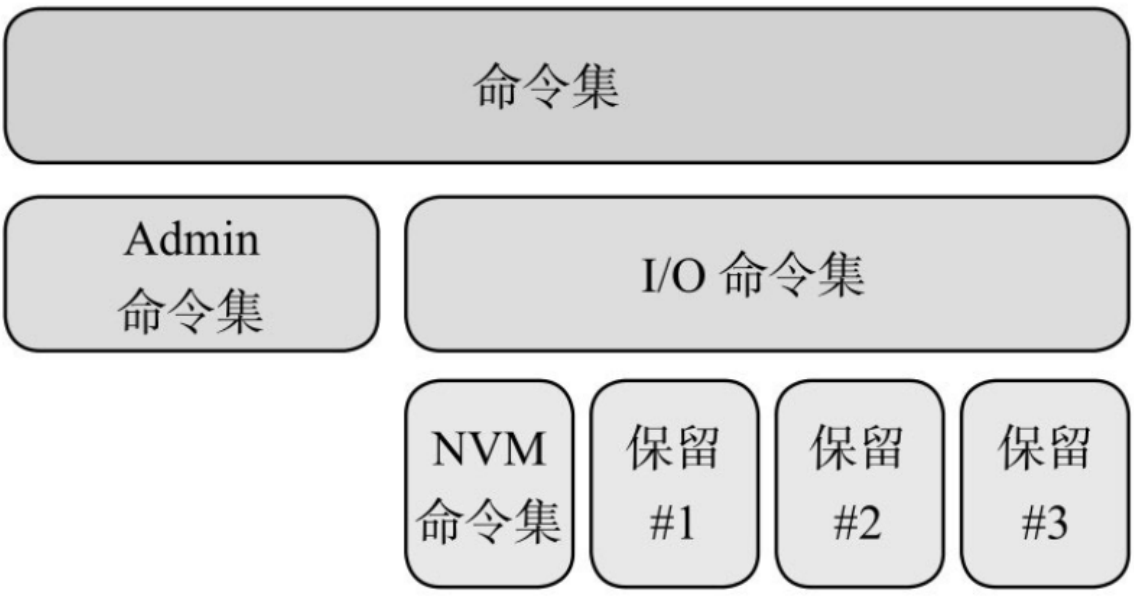
nvme协议

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

NVMe作为命令层和应用层协议，理论上可以适配在任何接口协议上。但NVMe协议的原配是PCIe。

nvme命令

NVMe制定了主机与SSD之间通信的命令，以及命令如何执行的。NVMe有两种命令，一种叫**Admin**命令，用以主机管理和控制SSD；另外一种就是**I/O命令**，用以主机和SSD之间数据的传输：



admin命令

NVMe支持的Admin命令如下图：

命令	必须还是可选	种类
Create I/O Submission Queue	必须	Queue 管理
Delete I/O Submission Queue	必须	
Create I/O Completion Queue	必须	
Delete I/O Completion Queue	必须	
ldentify	必须	配置
Get Features	必须	
Set Features	必须	
Get Log Page	必须	汇报状态信息
Asynchronous Event Request	必须	
Abort	必须	中止命令
Firmware lmage Download	可选	固件更新 / 管理
Firmware Activate	可选	
I/O Command Set Specific Commands	可选	I/O 命令集特有
Vendor Specific Commands	可选	商家特有

IO命令

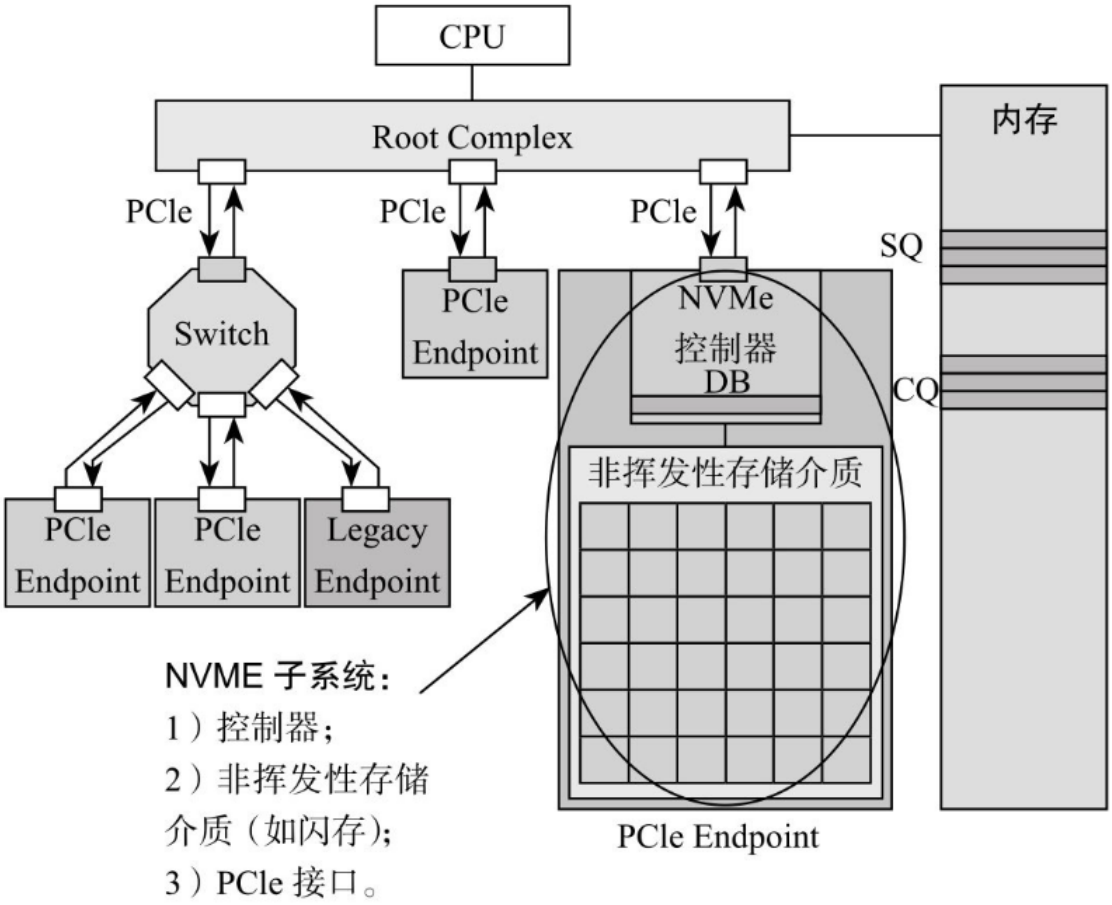
NVMe支持的I/O命令如图：

命令	必须还是可选	种类
Read	必须	必须的数据命令
Write	必须	
Flush	必须	
Write Uncorrectable	可选	可选的数据命令
Write Zeros	可选	
Compare	可选	
Dataset Management	可选	数据暗示
Reservation Acquire	可选	Reservation 命令
Reservation Register	可选	
Reservation Release	可选	
Reservation Report	可选	
Vendor Specific Commands	可选	商家特有

命令的执行

命令有了，那么，主机又是怎么把这些命令发送给SSD执行呢?

NVMe有三宝：**Submission Queue（SQ）**、**Completion Queue（CQ）**和**Doorbell Register（DB）**。**SQ和CQ位于主机的内存中**，**DB则位于SSD的控制器内部**，如下图：

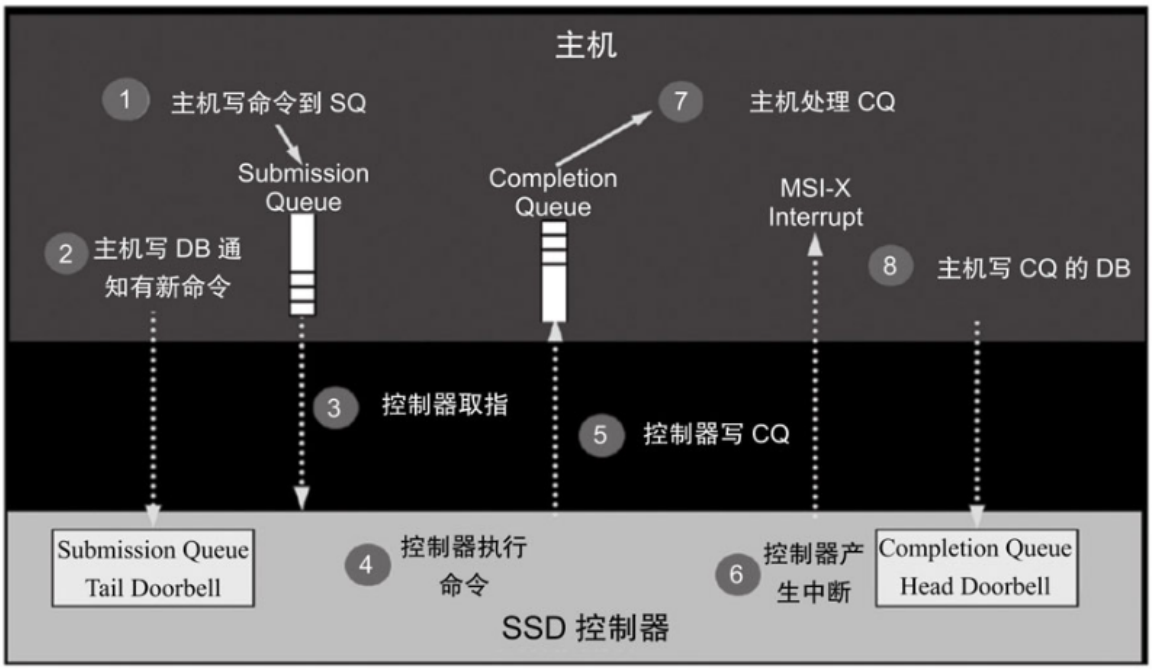


SQ位于主机内存中，主机要发送命令时，先把准备好的命令放在SQ中，然后通知SSD来取；

CQ也是位于主机内存中，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。

DB主机发送命令时，不是直接往SSD中发送命令，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？主机就是通过写SSD端的DB寄存器来告知SSD的。

nvme命令处理流程



NVMe处理命令需要八步。

第一步，主机写命令到SQ；

第二步，主机写SQ的DB，通知SSD取指；

第三步，SSD收到通知后，到SQ中取指；

第四步，SSD执行指令；

第五步，指令执行完成，SSD往CQ中写指令执行结果；

第六步，然后SSD发中断通知主机指令完成；

第七步，收到中断，主机处理CQ，查看指令完成状态；

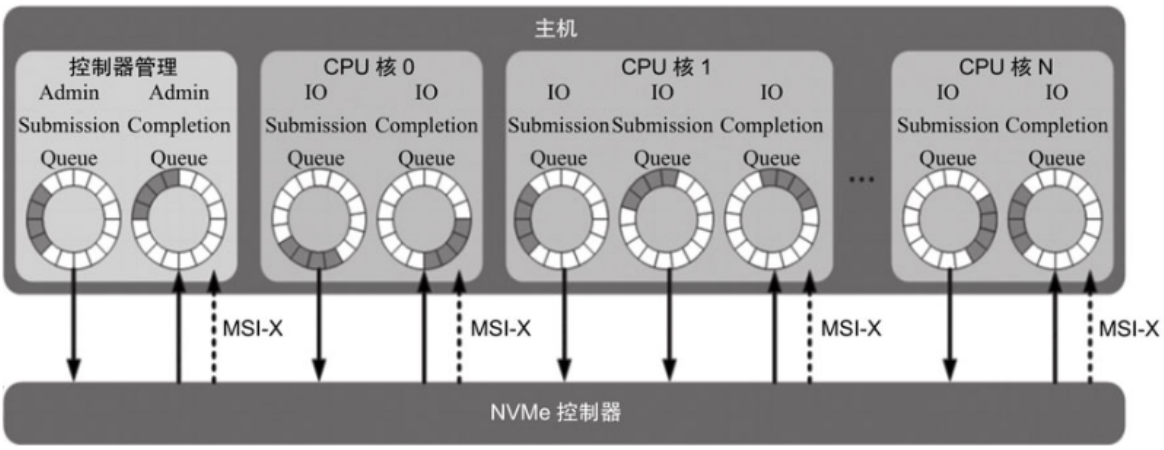
第八步，主机处理完CQ中的指令执行结果，通过DB回复SSD

SQ，CQ，DB

主机往SQ中写入命令，SSD往CQ中写入命令完成结果。**SQ与CQ的关系，可以是一对一的关系，也可以是多对一的关系**，但不管怎样，它们是成对的：有因就有果，有SQ就必然有CQ。

有两种SQ和CQ，**一种是Admin**，**另外一种是IO**，前者放Admin命令，用以主机管理控制SSD，后者放置IO命令，用以主机与SSD之间传输数据。Admin SQ/CQ和IO SQ/CQ各司其职。**IO SQ/CQ不是一生下来就有的，它们是通过Admin命令创建的。**

如下图，系统中只有1对Admin SQ/CQ，它们是一一对应的关系；IOSQ/CQ却可以有很多，多达65535对（64K减去1对Admin SQ/CQ）



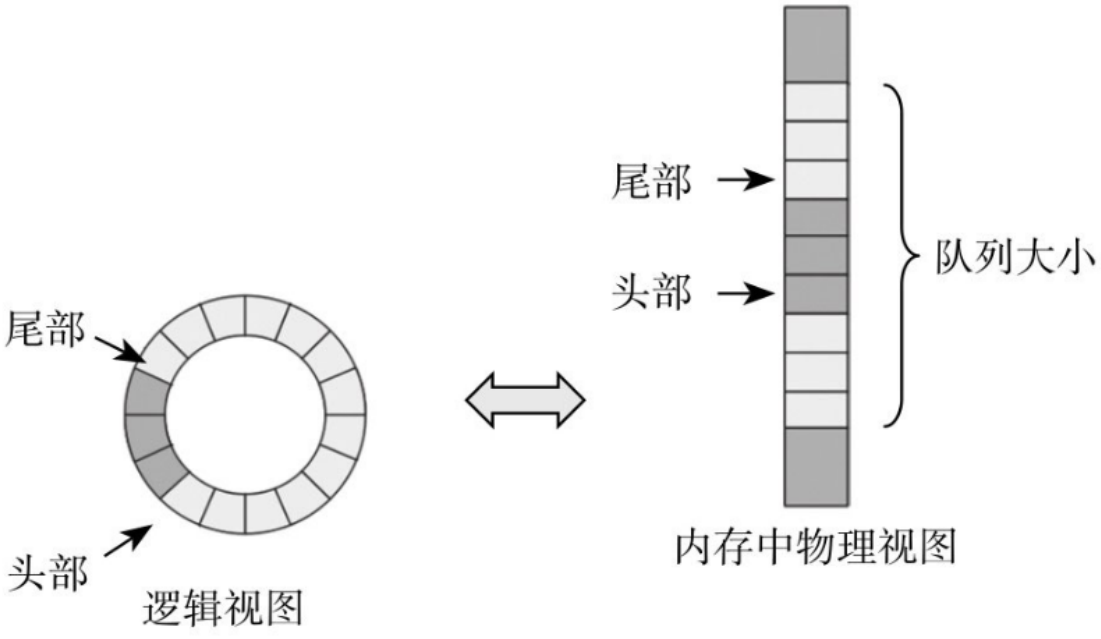
主机端每个CPU核（Core）可以有一个或者多个SQ，但只有一个CQ。给每个CPU核分配一对SQ/CQ好理解，为什么一个CPU核中还要多个SQ呢？一是性能需求，一个CPU核中有多线程，可以做到一个线程独享一个SQ；二是QoS需求。

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IO SQ/CQ，深度可以是2～65536（64K）。**队列深度也是可以配置的。**SQ/CQ的个数可以配置，每个SQ/CQ的深度也可以配置，因此NVMe的性能是可以通过配置队列个数和队列深度来灵活调节的。

SQ和CQ小结：

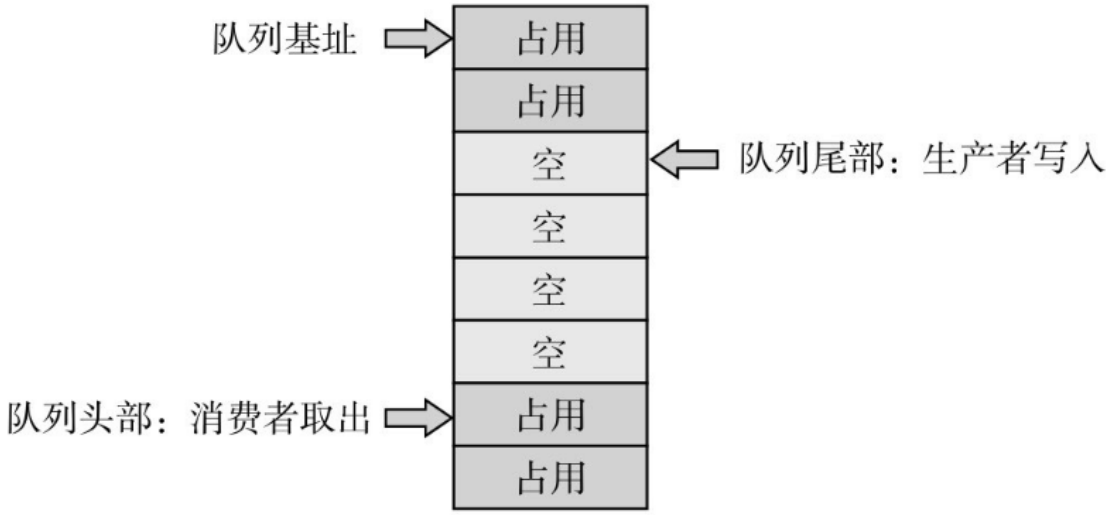
- SQ用以主机发命令，CQ用以SSD回命令完成状态；
- SQ/CQ可以在主机的内存中，也可以在SSD中，但一般在主机内存中；
- 两种类型的SQ/CQ:Admin和IO，前者发送Admin命令，后者发送IO命令；
- 系统中只能有一对Admin SQ/CQ，但可以有很多对IO SQ/CQ；
- IO SQ与CQ可以是一对一的关系，也可以是多对一的关系；
- IO SQ是可以赋予不同优先级的；
- IO SQ/CQ深度可达64K, Admin SQ/CQ深度可达4K；
- IO SQ/CQ的广度和深度都可以灵活配置；
- 每条命令大小是64B，每条命令完成状态是16B。

SQ/CQ中的“Q”指Queue，队列的意思，无论SQ还是CQ，都是队列，并且是环形队列。队列有几个要素，除了队列深度、队列内容，还有队列的头部（Head）和尾部（Tail）。如下图：



队伍头部的那个正在被服务或者等待被服务，一旦完成，就离开队伍。可见队列的头尾很重要，头决定谁会被马上服务，尾巴决定了新来的人站的位置。**DB就是用来记录了一个SQ或者CQ的头和尾。**每个SQ或者CQ，都有两个对应的DB:HeadDB和Tail DB。DB是在SSD端的寄存器，记录SQ和CQ的头和尾巴的位置。

下图是一个队列生产者消费者模型：

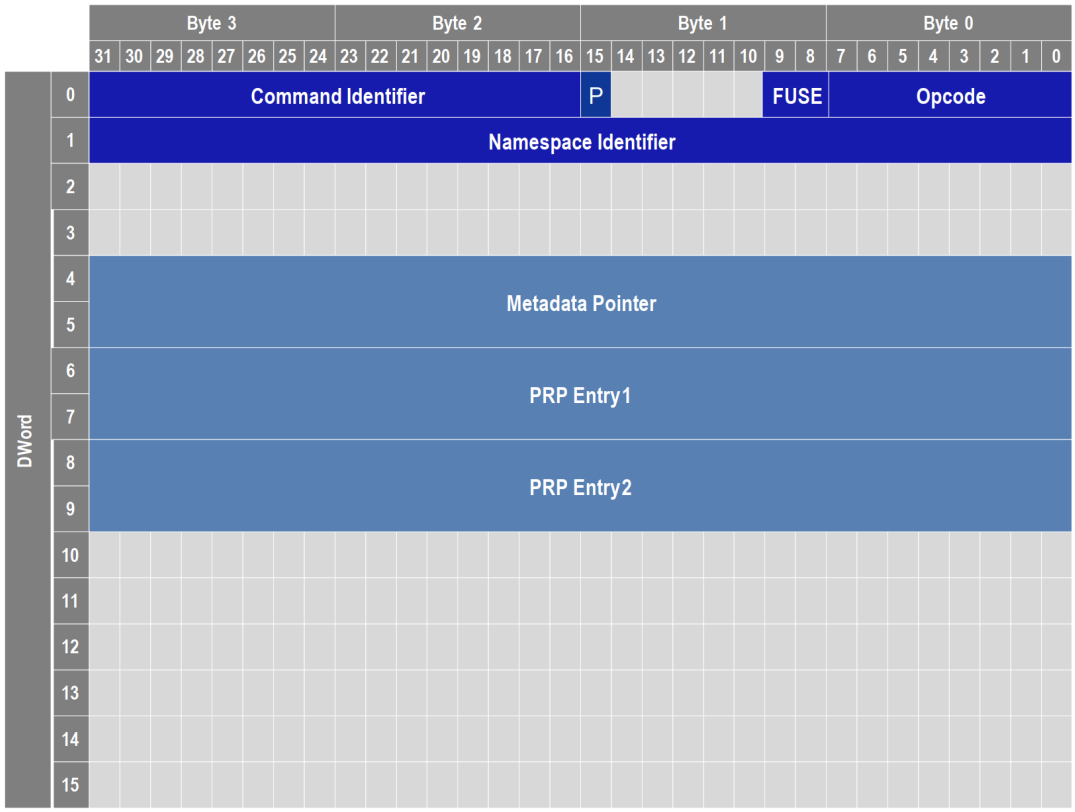


生产者往队列的尾部写入东西，消费者从队列的头部取出东西。**对一个SQ来说，它的生产者是主机**，因为它向SQ的尾部写入命令，消费者是SSD，因为它从SQ的头部取出指令执行；**对一个CQ来说，刚好相反，生产者是SSD，因为它向CQ的尾部写入命令完成信息**，消费者则是主机，它从CQ的头部取出命令完成信息。

DB在命令处理流程中起了什么作用呢?

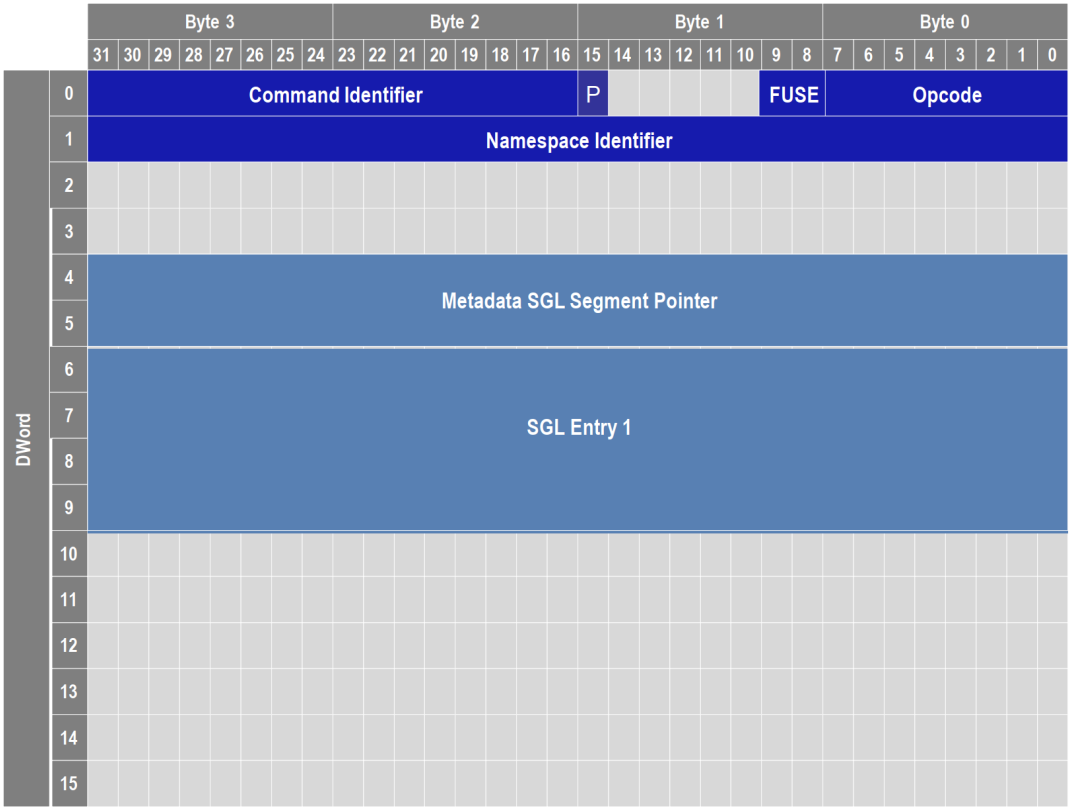
1. 对SQ来说，SSD是消费者，它直接和队列的头打交道，很清楚SQ的头在哪里，所以SQ head DB由SSD自己维护；但它不知道队伍有多长，尾巴在哪，后面还有多少命令等待执行，相反，主机知道，所以**SQ Tail DB由主机来更新。**SSD结合SQ的头和尾，就知道还有多少命令在SQ中等待执行了。
2. 对CQ来说，SSD是生产者，它很清楚CQ的尾巴在哪里，所以CQ Tail DB由自己更新，但是SSD不知道主机处理了多少条命令完成信息，需要主机告知，因此**CQ Head DB**由主机更新。SSD根据CQ的头和尾，就知道CQ还能不能，以及能接受多少命令完成信息。
3. DB还起到了通知作用：主机更新SQ Tail DB的同时，也是在告知SSD有新的命令需要处理；主机更新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

Submission Queue Element with PRPs(64B)



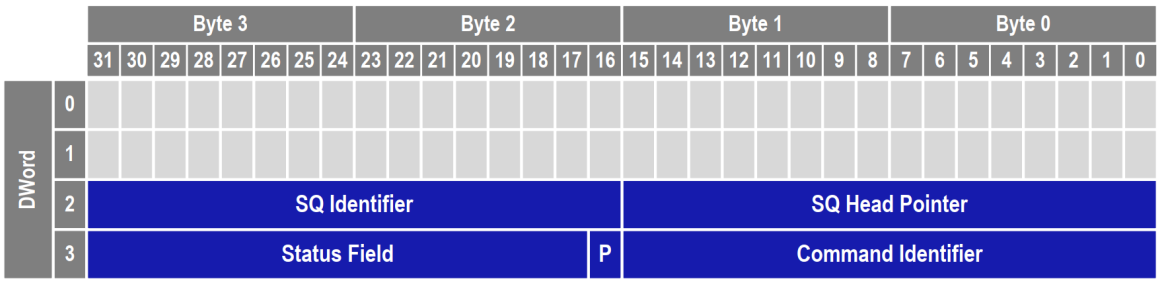
- Opcode – Command operation code
- Fused Operation (FUSE) – specifies if two commands should be executed as atomic unit (optional)
- PRP or SGL for Data Transfer = 0 specifies that PRP's are used; 1 specifies SGLs are used
- Command Identifier – Command ID within submission queue
- Namespace – Namespace on which command operates
- Metadata Pointer – Pointer to contiguous buffer containing metadata
- PRP Entry 1 – First PRP entry for the command or PRP list pointer depending on the command
- PRP Entry 2 – Second PRP entry for the command or PRP list pointer depending on the command

Submission Queue Element with SGLs(64B)



- Opcode – Command operation code
- Fused Operation (FUSE) – specifies if two commands should be executed as atomic unit (optional)
- PRP or SGL for Data Transfer = 0 specifies that PRP's are used ; 1 specifies that SGLs are used
- Command Identifier – Command ID within submission queue
- Namespace – Namespace on which command operates
- Metadata SQL Segment Pointer – first SGL segment which describes the metadata to transfer
- SGL Entry 1 – the first SGL segment for the command

Completion Queue Element (16B)



- SQ Head Pointer – Submission queue head pointer associated with SQ Identifier
- SQ Identifier – Submission queue associated with completed command
- Command Identifier – Command ID within submission queue
- Phase Tag (P) – Indicates when new command is reached
- Status Field – Status associated with completed command

  A value of zero indicates successful command completion

DB小结：

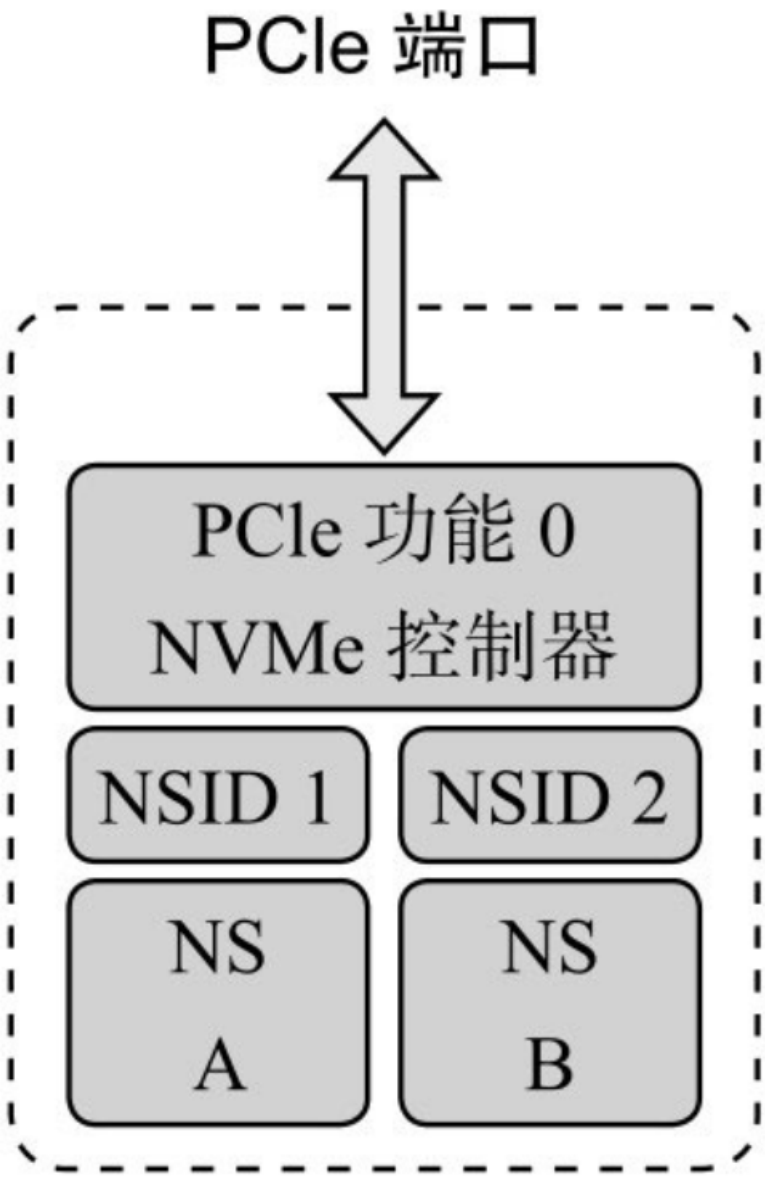
- DB在SSD控制器端，是寄存器；
- DB记录着SQ和CQ队列的头部和尾部；
- 每个SQ或者CQ有两个DB——Head DB和Tail DB；
- 主机只能写DB，不能读DB；
- 主机通过SSD往CQ中写入的命令完成状态获取其队列头部或者尾部。

namespace

一个NVMe SSD主要由SSD控制器、闪存空间和PCIe接口组成。如果把闪存空间划分成若干个**独立的逻辑空间**，每个空间逻辑块地址（LBA）范围是0到N-1（N是逻辑空间大小），这样划分出来的每一个逻辑空间我们就叫作NS。NVMe SSD可以是一个闪存空间对应多个逻辑空间。

每个NS都有一个名称与ID，ID是独一无二的，系统就是通过NS的ID来区分不同的NS。

如图下图所示，整个闪存空间划分成两个NS，名字分别是NS A和NS B，对应的NS ID分别是1和2。如果NS A大小是M（以逻辑块大小为单位），NS B大小是N，则它们的逻辑地址空间分别是0到M-1和0到N-1。主机读写SSD，都是要在命令中指定读写的是哪个NS中的逻辑块。原因很简单，如果不指定NS，对同一个LBA来说，假设就是LBA 0, SSD根本就不知道去读或者写哪里，因为有两个逻辑空间，每个逻辑空间都有LBA 0。



对每个NS来说，都有一个4KB大小的数据结构来描述它。该数据结构描述了该NS的大小，整个空间已经写了多少，每个LBA的大小，端到端数据保护相关设置，以及该NS是属于某个控制器还是几个控制器可以共享等。NS由主机创建和管理，**每个创建好的NS，从主机操作系统角度看来，就是一个独立的磁盘，用户可在每个NS做分区等操作。**

NVMe SSD 如何使用BAR空间

BAR空间是PCIe设备体现自己功能的地方。NVMe协议在BAR空间上定义了一系列的寄存器。有了这些寄存器，Host就可以与NVMe SSD的Controller进行交互了。如下图，可以看到有Controller 状态，设置等寄存器，管理命令（Admin Command）配置寄存器和NVMe队列寄存器（Doorbell用来通知Controller I/O Request操作信息）

3.1 Register Definition

The following table describes the register map for the controller.

Start	End	Symbol	Description
00h	07h	CAP	Controller Capabilities
08h	0Bh	VS	Version
0Ch	0Fh	INTMS	Interrupt Mask Set
10h	13h	INTMC	Interrupt Mask Clear
14h	17h	CC	Controller Configuration
18h	1Bh	Reserved	Reserved
1Ch	1Fh	CSTS	Controller Status
20h	23h	NSSR	NVM Subsystem Reset (Optional)
24h	27h	AQA	Admin Queue Attributes
28h	2Fh	ASQ	Admin Submission Queue Base Address
30h	37h	ACQ	Admin Completion Queue Base Address
38h	EFFh	Reserved	Reserved
F00h	FFFh	Reserved	Command Set Specific
1000h	1003h	SQ0TDBL	Submission Queue 0 Tail Doorbell (Admin)
1000h + (1 * (4 << CAP.DSTRD))	1003h + (1 * (4 << CAP.DSTRD))	CQ0HDBL	Completion Queue 0 Head Doorbell (Admin)
1000h + (2 * (4 << CAP.DSTRD))	1003h + (2 * (4 << CAP.DSTRD))	SQ1TDBL	Submission Queue 1 Tail Doorbell
1000h + (3 * (4 << CAP.DSTRD))	1003h + (3 * (4 << CAP.DSTRD))	CQ1HDBL	Completion Queue 1 Head Doorbell
1000h + (4 * (4 << CAP.DSTRD))	1003h + (4 * (4 << CAP.DSTRD))	SQ2TDBL	Submission Queue 2 Tail Doorbell
1000h + (5 * (4 << CAP.DSTRD))	1003h + (5 * (4 << CAP.DSTRD))	CQ2HDBL	Completion Queue 2 Head Doorbell
...	...	...	...
1000h+ (2y * (4 << CAP.DSTRD))	1003h + (2y * (4 << CAP.DSTRD))	SQyTDBL	Submission Queue y Tail Doorbell
1000h + ((2y + 1) * (4 << CAP.DSTRD))	1003h + ((2y + 1) * (4 << CAP.DSTRD))	CQyHDBL	Completion Queue y Head Doorbell
			Vendor Specific (Optional)

##

Submission Queue Element with PRPs(64B)

参考：

1. 深入浅出SSD
2. https://blog.csdn.net/Memblaze_2011/article/details/52766905
3. An NVM Express Tutorial