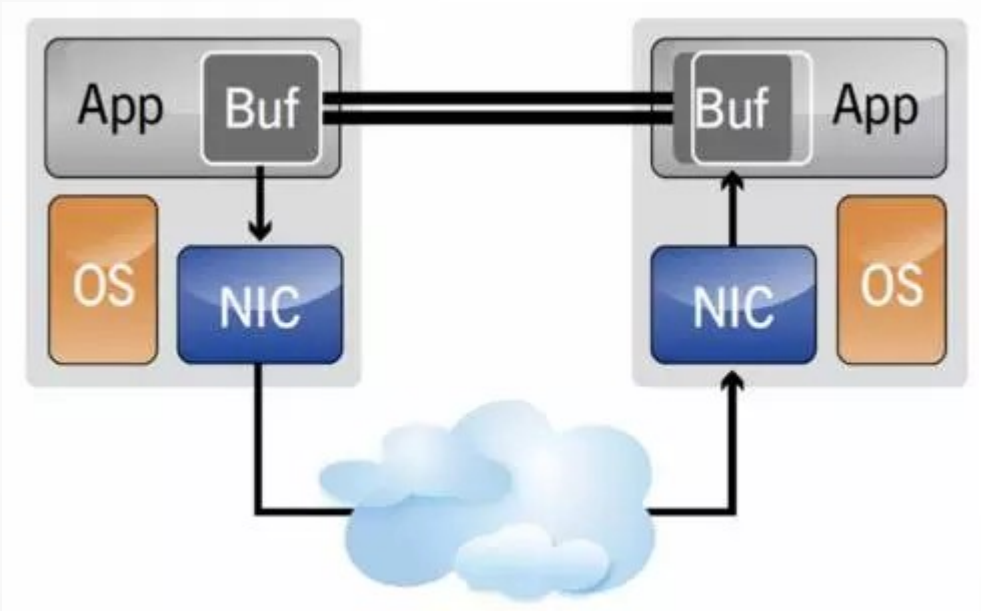
RDMA & NVMf

RDMA 是个什么鬼？相信大部分不关心高性能网络的童鞋都不太了解。但是 NVMe over Fabrics 的出现让搞存储的不得不抽出时间来看看这个东西，这篇文章就来介绍下我所了解的 RDMA 。

RDMA （ Remote Direct Memory Access ）意为在远端直接访问主机的内存，而不需要主机参与。如下图，当主机和 Client 端都配备 RDMA NIC 的时候，数据通过 NIC 的 DMA 引擎直接在两端内存之间转移，而不需要经过 OS 的网络协议栈。这种技术对于局域网高带宽的存储系统非常有吸引力。

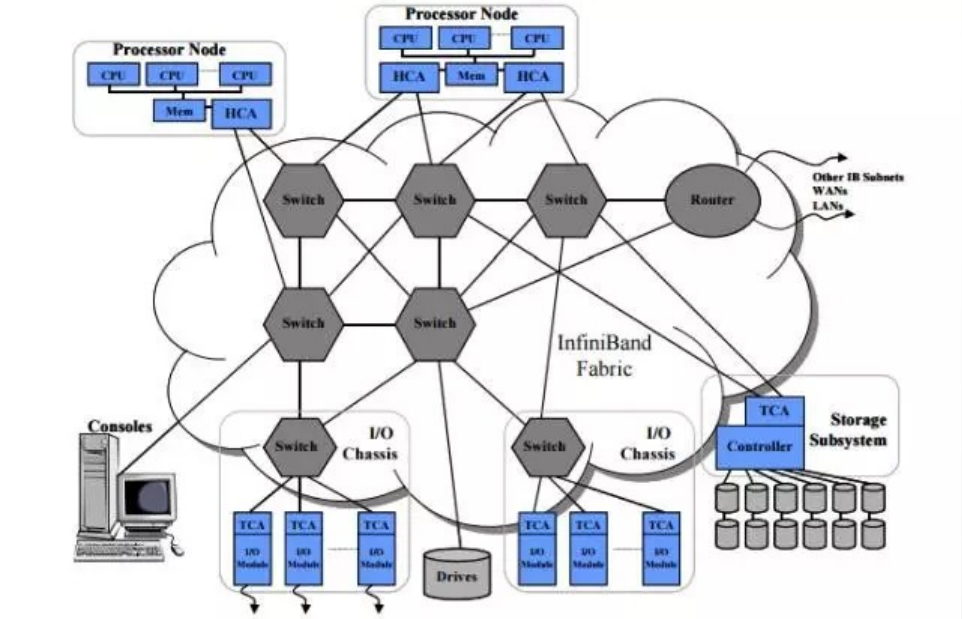
RDMA & NVMf



RDMA & NVMf

网络技术中，协议是必不可少的部分。 RDMA 环境下，传统的 TCP/IP 协议过于庞大，所以需要一种特殊的协议来发挥其优势，这就是 InfiniBand 协议（简称 IB ）。 InfiniBand 定义了一套完整的 IB 框架，这个框架中有我们在以太网中了解的大部分概念：交换机，路由器，子网络等。

Figure 1. InfiniBand System Fabric

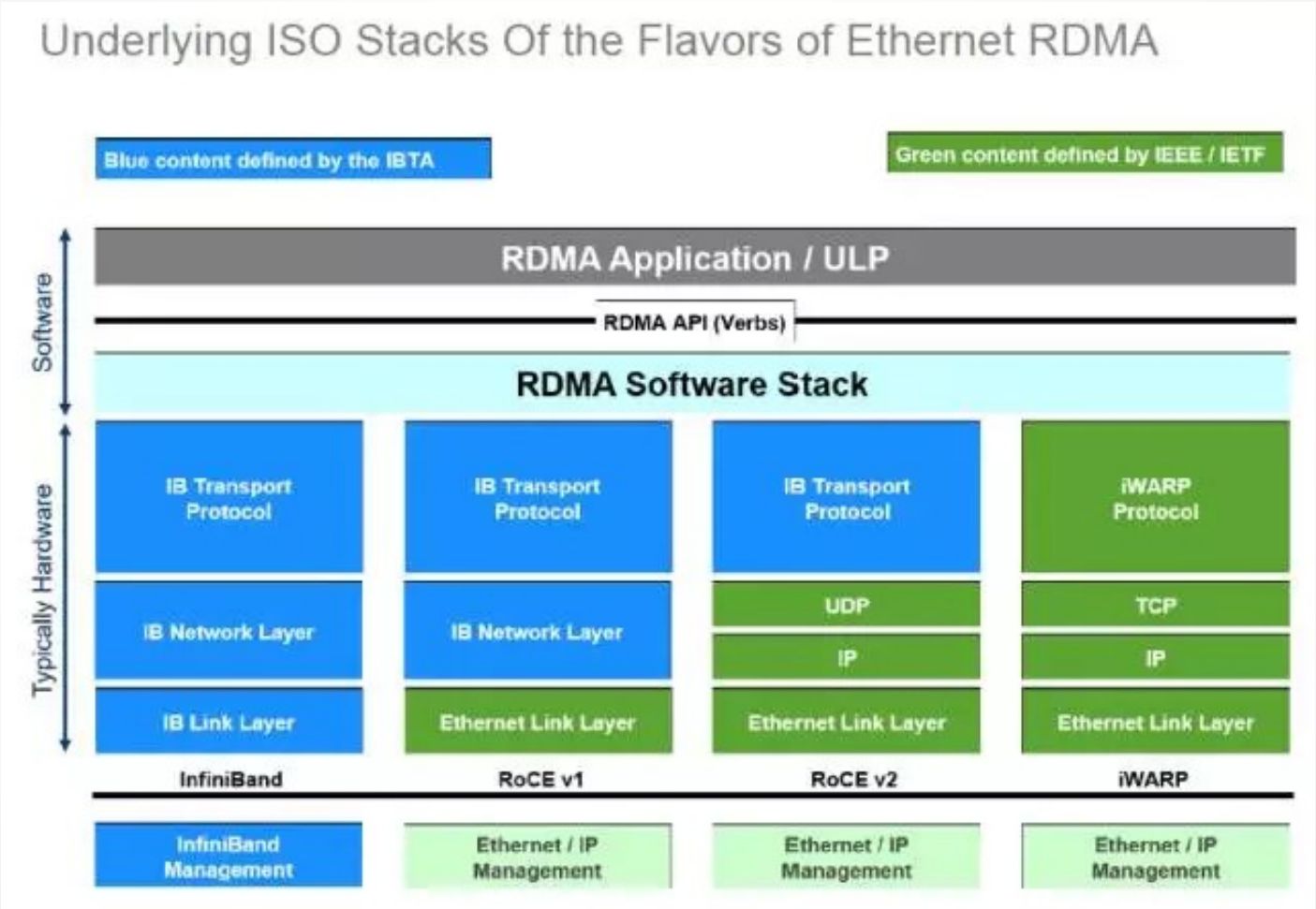


RDMA & NVMf

虽然 InfiniBand 看起来非常不错，但是组建一个 IB 网络，尤其当网络拓扑比较复杂的时候，对于习惯以太网的用户来说，在技术和成本方面花销太大。为了适应这方面的需求， IB 组织在 IB 协议基础上增加了适用于以太网的协议： ROCE 和 iWARP 。使用这两类协议就可以通过普通的以太网硬件组网。

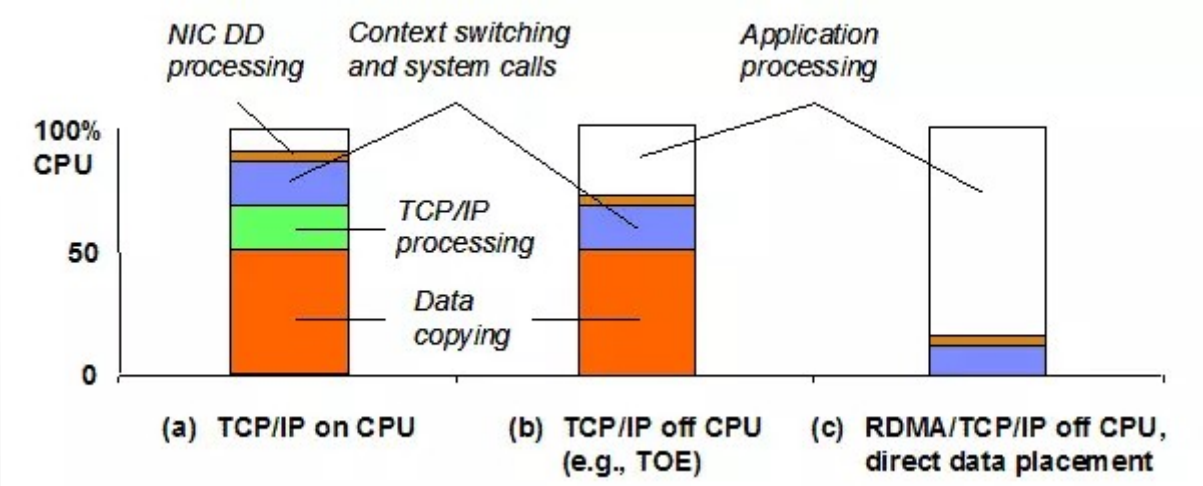
这些协议的关系可以看下图，其中 IB 性能最好， ROCE 则用得最多。无论是哪种技术，都必须保证 RDMA 的实现。

RDMA & NVMf


Underlying ISO Stacks Of the Flavors of Ethernet RDMA
Blue content defined by the IBTA
Green content defined by IEEE / IETF
Software
Typically Hardware
RDMA Application / ULP
RDMA API (Verbs)
RDMA Software Stack
IB Transport Protocol
IB Transport Protocol
IB Transport Protocol
iWARP Protocol
IB Network Layer
IB Network Layer
UDP
IP
TCP
IP
IB Link Layer
Ethernet Link Layer
Ethernet Link Layer
Ethernet Link Layer
InfiniBand
RoCE v1
RoCE v2
iWARP
InfiniBand Management
Ethernet / IP Management
Ethernet / IP Management
Ethernet / IP Management

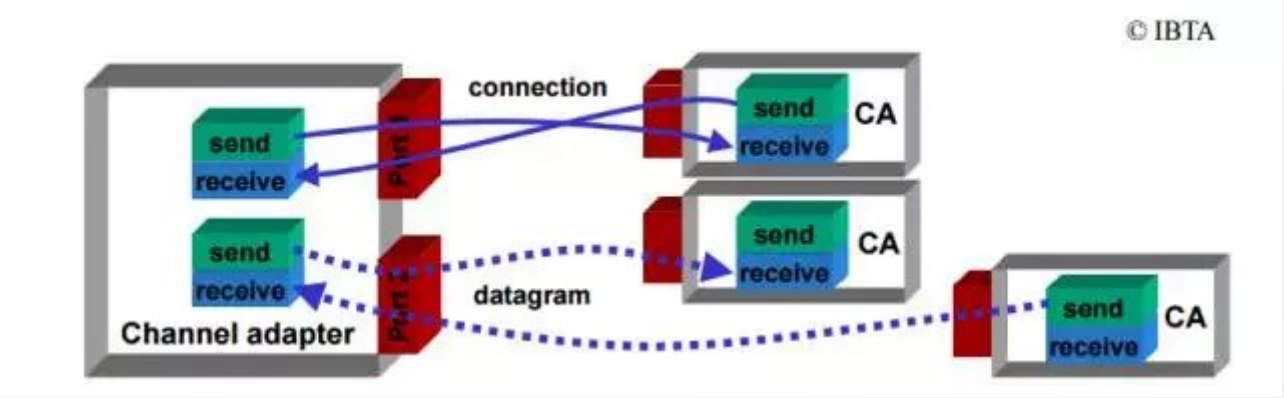
RDMA & NVMf

简单了解了 RDMA ，我们与传统的网卡对比来进一步说明。
普通的网卡都是基于 OS 的 TCP/IP 技术为上层提供网络服务。在 Linux 内核中，有一个著名的结构体 sk_buff ，这个结构体用来暂时存储传输的数据，它贯穿于内核网络协议栈和网卡驱动，用户的收发数据都要经过 sk_buff。可以推断，这种设计至少需要一次内存 copy ，再加上 TCP/IP 等的处理，整个下来就造成了不少的 overhead（引入的 Latency 和 CPU 处理时间）。



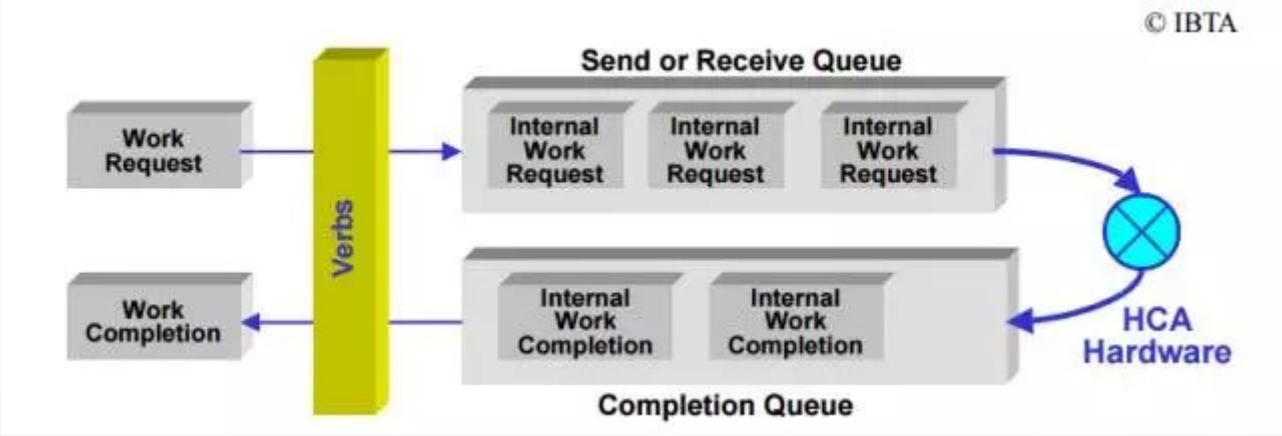
RDMA & NVMf

RDMA 在编程模型上跟 Socket 有几分相似之处，比如都会使用 Send 和 Receive 交换信息。在 RDMA 中，还有一个 Queue Pair （ QP ）的概念，每个 QP 由一个 Send Queue 和 Receive Queue 组成， Send Queue 用来发送数据， Receive Queue 则在另一端接收数据。在进行通信前， Server 和 Client 端都需要建立这样的 Queue Pair。 RDMA 支持多个 QP ，其数量限制由对应的网卡决定。



RDMA & NVMf

QP 中的传输使用 Work Request （ WR ）进行，而非数据流的形式。应用程序在 Work Request 中指定收发数据的地址（ RDMA 对数据存放的地址有要求，这些地址在使用前，必须注册到 IB 驱动中）。除此之外， QP 的 Send Queue 和 Receive Queue 还需要配备一个 Completion Queue ，这个 Completion Queue 用来保存 WR 处理结果（发送或者收到）， WR 信息可以从 Completion Queue 中的 Work Completion （ WC ）中获得。



RDMA & NVMf

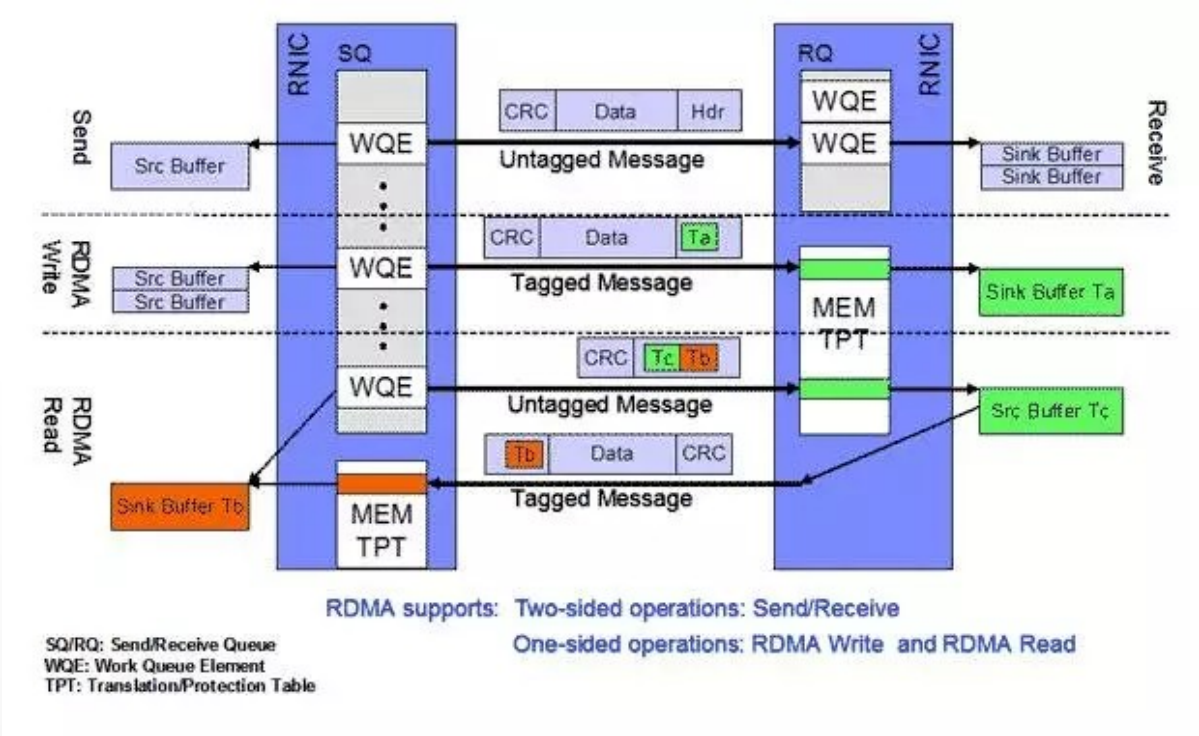
进一步讲，WR 还分为 Receive WR 和 Send WR ，Receive WR 用来指定另一端发过来的数据存放位置，Send WR 则是实际的数据发送请求。在主机中注册的内存类型（ ib_access_flags ）决定了远端 client 操作主机内存的方式。如果具有 Remote Access 权限，则可以直接在 Send WR 中指定待操作的地址（此为 RDMA Read/Write 操作），主机无需参与操作；否则 Send WR 对远端地址没有控制权，即发送的数据的存放地址不由 Send WR 决定（只能 Send/Recv 操作），主机需要处理请求。使用哪种操作方式可以在 ib_wr_opcode 中指定。

```
enum ib_wr_opcode {
        IB_WR_RDMA_WRITE,
        IB_WR_RDMA_WRITE_WITH_IMM,
        IB_WR_SEND,
        IB_WR_SEND_WITH_IMM,
        IB_WR_RDMA_READ,
        IB_WR_ATOMIC_CMP_AND_SWP,
        IB_WR_ATOMIC_FETCH_AND_ADD,
        IB_WR_LSO,
        IB_WR_SEND_WITH_INV,
        IB_WR_RDMA_READ_WITH_INV,
```

```
enum ib_access_flags {
        IB_ACCESS_LOCAL_WRITE   = 1,
        IB_ACCESS_REMOTE_WRITE  = (1<<1),
        IB_ACCESS_REMOTE_READ   = (1<<2),
        IB_ACCESS_REMOTE_ATOMIC = (1<<3),
        IB_ACCESS_MW_BIND       = (1<<4),
        IB_ZERO_BASED           = (1<<5),
        IB_ACCESS_ON_DEMAND     = (1<<6),
};
```

RDMA & NVMf

这两种方式就是经常提到的 one-sided 和 Two-sided。 one-sided （ RDMA Read/Write ）相比于 Two-sided 的好处是释放了主机端的 CPU ，降低了传输的 Latency。从下图可以看出， one-sided 方式在主机端无需生成 WQE ，也就不需要处理 Work Completion。



RDMA & NVMf

最后，我们回到 NVMe over Fabrics ，以 client 的一个写请求的处理过程来展示 NVMf 如何利用 RDMA 技术。

1 ， NVMe Queue 与 Client 端 RDMA QP 一一对应，把 NVMe Submission Queue 中的 NVMe Command 存放到 RDMA QP 注册的内存地址中（有可能带上 I/O Payload ），然后通过 RDMA Send Queue 发送出去；

2 ，当 Target 的 NIC 收到 Work Request 后把 NVMe Command DMA 到 Target RDMA QP 注册的内存中，并在 Targe 的 QP 的 Completion Queue 中设置一个 Work Completion 。

3 ， Target 处理 Work Completion 时，把 NVMe Command 发送到后端 PCIe NVMe 驱动，如果本次传输没有带上 I/O Payload ，则使用 RDMA Read 获取；

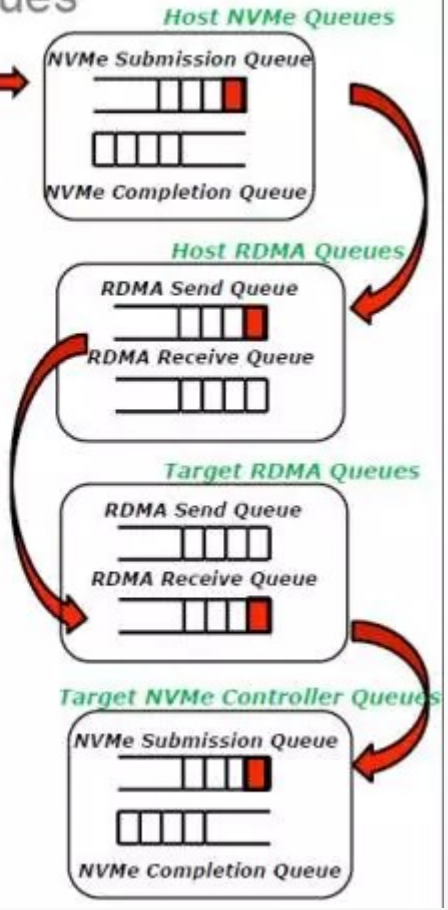
4 ， Target 收到 PCIe NVMe 驱动处理结果后通过 QP 的 Send Queue 把 NVMe Completion 结果返回给 client 。

RDMA & NVMf

Queues, Capsules, and More Queues

Example of Host Write To Remote Target

- NVMe Host Driver encapsulates the NVMe Submission Queue Entry (including data) into a fabric-neutral Command Capsule and passes it to the NVMe RDMA Transport
- Capsules are placed in Host RNIC RDMA Send Queue and become an RDMA_SEND payload
- Target RNIC at a Fabric Port receives Capsule in an RDMA Receive Queue
- RNIC places the Capsule SQE and data into target host memory
- RNIC signals the RDMA Receive Completion to the target's NVMe RDMA Transport
- Target processes NVMe Command and Data
- Target encapsulates the NVMe Completion Entry into a fabric-neutral Response Capsule and passes it to NVMe RDMA Transport



RDMA & NVMf

上面这个流程是当前 NVMf 的实现，可以看出，目前 NVMe Command 使用的是 two-sided 形式。一部分原因是 Target 端 CPU 需要处理 QP 的 Work Completion ：将收到的 NVMe 命令提交给 PCIe NVMe 驱动。如果把这一块能够 offload ，也许能够实现 NVMf 的 one-sided 传输，到时候性能会更强悍。

总结

这篇文章介绍了 NVMf 中最先实现的 Transport-RDMA 的部分技术细节。 RDMA 已经是一个成熟的技术，之前更多的是用在高性能计算中，而 NVMe 的出现，让它的使用范围迅速扩大。对于今天的存储从业者尤其是 NVMe 领域，了解 RDMA 技术对于跟进 NVMe 的发展趋势有比较大的帮助，希望这篇文章对您了解 RDMA 有一定的帮助，但是更多的 RDMA 技术细节还需要根据自己的需求去挖掘

RDMA & NVMf