NVM/NVMe/NVDIMM/RDMA

Guoqing Jiang
gqjiang@suse.com

Contents

NVM

Non-volatile memory or non-volatile storage is a type of computer memory that can retrieve stored information even after having been power cycled. In contrast, volatile memory needs constant power in order to retain data.

There are three distinct storage types:

1. NVS: traditional Non-Volatile Storage in mechanical disks – Hard disk, Solid State Drive, Optical disk, floppy disk, etc.

2. NVM: storage in Non-Volatile Memory chips (Flash memory Storage) – EEPROM, SSD, NAND, etc.

3. NVMM: storage inside Non-Volatile Main Memory chips (Re-RAM) – NVDIMM-P, specifications to be released 2018.

NVMe

NVM Express (NVMe) or Non-Volatile Memory Host Controller Interface Specification (NVMHCIS) is an open logical device interface specification for accessing non-volatile storage media (mostly for SSD) attached via a PCI Express (PCIe) bus - Wiki

NVM Express (NVMe) is an interface that allows host software to communicate with a non-volatile memory subsystem. This interface is optimized for Enterprise and Client solid state drives, typically attached as a register level interface to the PCI Express interface

- NVMe 1.3b spec

NVMe

Historically, SSDs used buses such as SATA/SAS etc for interfacing with the rest of a computer system. However, SATA was designed primarily for interfacing with mechanical hard disk drives (HDDs), and it became increasingly inadequate for SSDs, which improved in speed over time. High-end SSDs had been made using the PCI Express bus before NVMe, but using non-standard specification interfaces. Similar as USB mass storage, standardizing the interface of SSDs, OS only need one driver to work with all SSDs adhering to the specification.

NVMe queue

NVMe uses queues mechanism to deliver message, and it has two types of queues:

- **Admin Queue for Admin Command Set**
  - One per NVMe controller with up to 4K elements per queue
  - Used to configure IO Queues and controller/feature management
- **I/O Queues for IO Command Sets (e.g., NVM command set)**
  - Up to 64K queues per NVMe controller with up to 64K elements per queue
  - Used to submit/complete IO commands

NVMe uses circular queues to pass messages (e.g., commands and command completion notifications.). The queues may be located anywhere in PCIe memory.

NVMe queue

Both admin queue and I/O queue have two type queues:

1. submission queues (SQ)

- Queues messages from host to controller
- Used to submit commands
- Identified by SQ ID

2. completion queues (CQ)

- Queues messages from controller to host
- Used to post command completions
- Identified by CQ ID
- May have an independent MSI-X interrupt per completion queue

For SMP, each core can have its own queue and no need for competing one command queue like sata, so this is the motivation for blk-mq?

Linux NVMe

NVMe target works like iSCSI target, it could be connect to host by RDMA or FC. NVMe host (actually client) could connect to a remote NVMe target (by FC or RDMA) or local storage through PCIe interface?

```
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme
host  Kconfig  Makefile  target
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme/target/
admin-cmd.c  configfs.c  core.c  discovery.c  fabrics-cmd.c  fc.c  fcloop.c
io-cmd.c  Kconfig  loop.c  Makefile  nvmet.h  rdma.c
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme/host/
core.c  fabrics.c  fabrics.h  fault_inject.c  fc.c  Kconfig  lightnvm.c  Makefile
multipath.c  nvme.h  pci.c  rdma.c  trace.c  trace.h
```

Linux NVMe

NVMe target works like iSCSI target, it could be connect to host by RDMA or FC. NVMe host (actually client) could connect to a remote NVMe target (by FC or RDMA) or local storage through PCIe interface?

```
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme
host  Kconfig  Makefile  target
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme/target/
admin-cmd.c  configfs.c  core.c  discovery.c  fabrics-cmd.c  fc.c  fcloop.c
io-cmd.c  Kconfig  loop.c  Makefile  nvmet.h  rdma.c
gqjiang@linux-8mgw:~/source-tree/linux> ls drivers/nvme/host/
core.c  fabrics.c  fabrics.h  fault_inject.c  fc.c  Kconfig  lightnvm.c  Makefile
multipath.c  nvme.h  pci.c  rdma.c  trace.c  trace.h
```

Linux NVMe

NVMf Target Configuration

NVMf Subsystem Port Configuration

NVMf Subsystem NQN and Namespace Configuration

```
$ tree /sys/kernel/config/nvmet
├── hosts
├── ports
│   └── 1
│       ├── addr_adrfam
│       ├── addr_traddr
│       ├── addr_treq
│       ├── addr_trsvcid
│       ├── addr_trtype
│       ├── referrals
│       └── subsystems
│           └── nqn.testiqn -> ../../../../nvmet/subsystems/nqn.testiqn
└── subsystems
    └── nqn.testiqn
        ├── allowed_hosts
        ├── attr_allow_any_host
        └── namespaces
            └── 1
                ├── device_nguid
                ├── device_path
                └── enable
```

Linux NVMe

NVMf Target Configuration Example

- Port configuration
  - ports/1/addr_trtype:**rdma**
  - ports/1/addr_trsvcid:**1023**
  - ports/1/addr_traddr:**40.10.10.114**
  - ports/1/addr_treq:not specified
  - ports/1/addr_adrfam:ipv4

- Subsystem configuration
  - subsystems/**nqn.testiqn**/namespaces/1/enable:1
  - subsystems/nqn.testiqn/namespaces/1/device_nguid:00000000-0000-0000-0000-000000000000
  - subsystems/nqn.testiqn/namespaces/**1**/device_path:**/dev/nvme3n1**
  - subsystems/nqn.testiqn/attr_allow_any_host:1

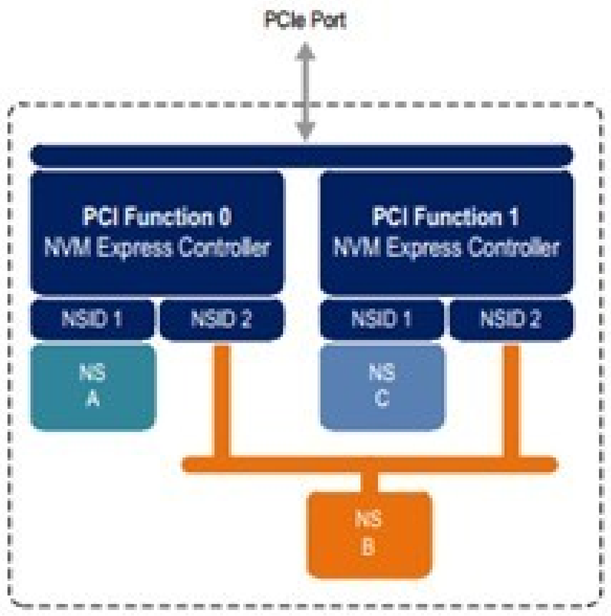
NVMe subsystem (target side)

Subsytem includes one or more controllers, one or more namespaces, one or more PCI Express ports, a non-volatile memory storage medium, and an interface between the controller(s) and non-volatile memory storage medium.

A Controller is associated with a single PCI Function.

A namespace is quantity of NVM physical memory wrapped as a logic block.

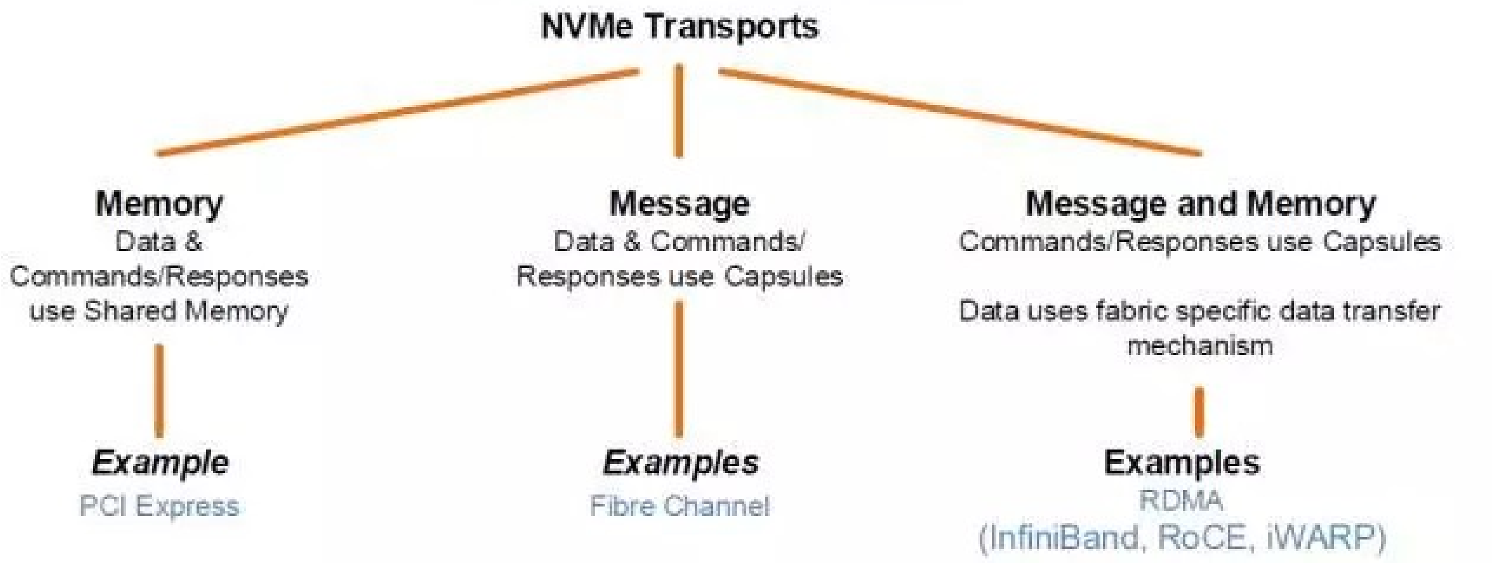
Port is exported to NVMe Client (Initiator) to create connection.



NVMe over Fabric

Besides local connection method (PCIe) to NVM device, we can access remote NVM device through fabric, NVMe over Fabric includes: NVMe over FC (Fabric channel) and NVMe over RDMA.

Figure 1: Taxonomy of Transports


NVMe Transports
Memory
Data & Commands/Responses use Shared Memory
Message
Data & Commands/ Responses use Capsules
Message and Memory
Commands/Responses use Capsules
Data uses fabric specific data transfer mechanism
Example
PCI Express
Examples
Fibre Channel
Examples
RDMA
(InfiniBand, RoCE, iWARP)

RDMA

RDMA is a direct memory access from the memory of one computer into that of another without involving either one's operating system. This permits high-throughput, low-latency networking, which is especially useful in massively parallel computer clusters.

RDMA supports zero-copy networking by enabling the network adapter to transfer data directly to or from application memory, eliminating the need to copy data between application memory and the data buffers in the operating system.

RDMA

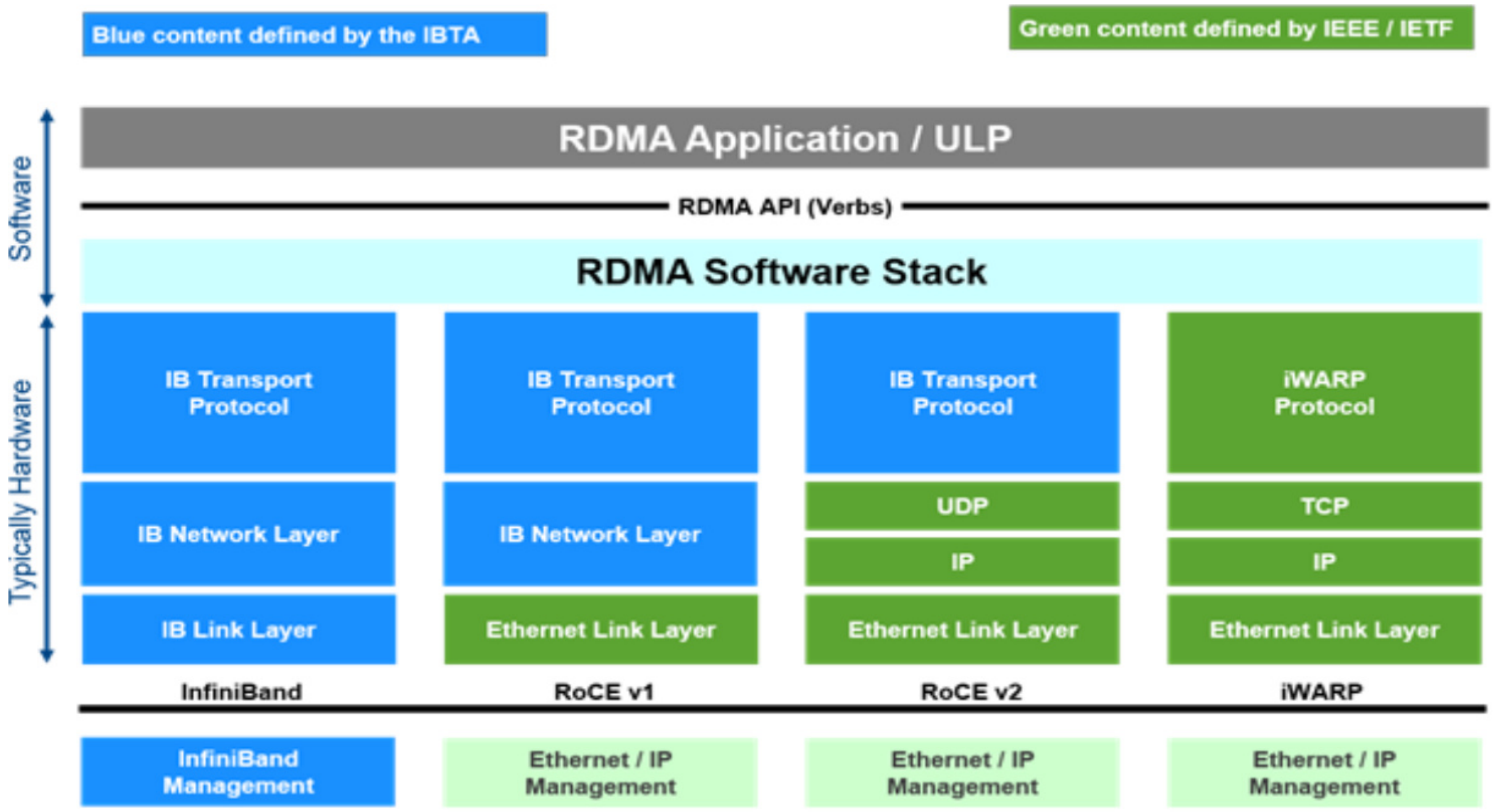
The major implementation of RDMA includes RDMA over Converged Ethernet (RoCE), InfiniBand and iWarp.

InfiniBand refers to two distinctly different concepts:

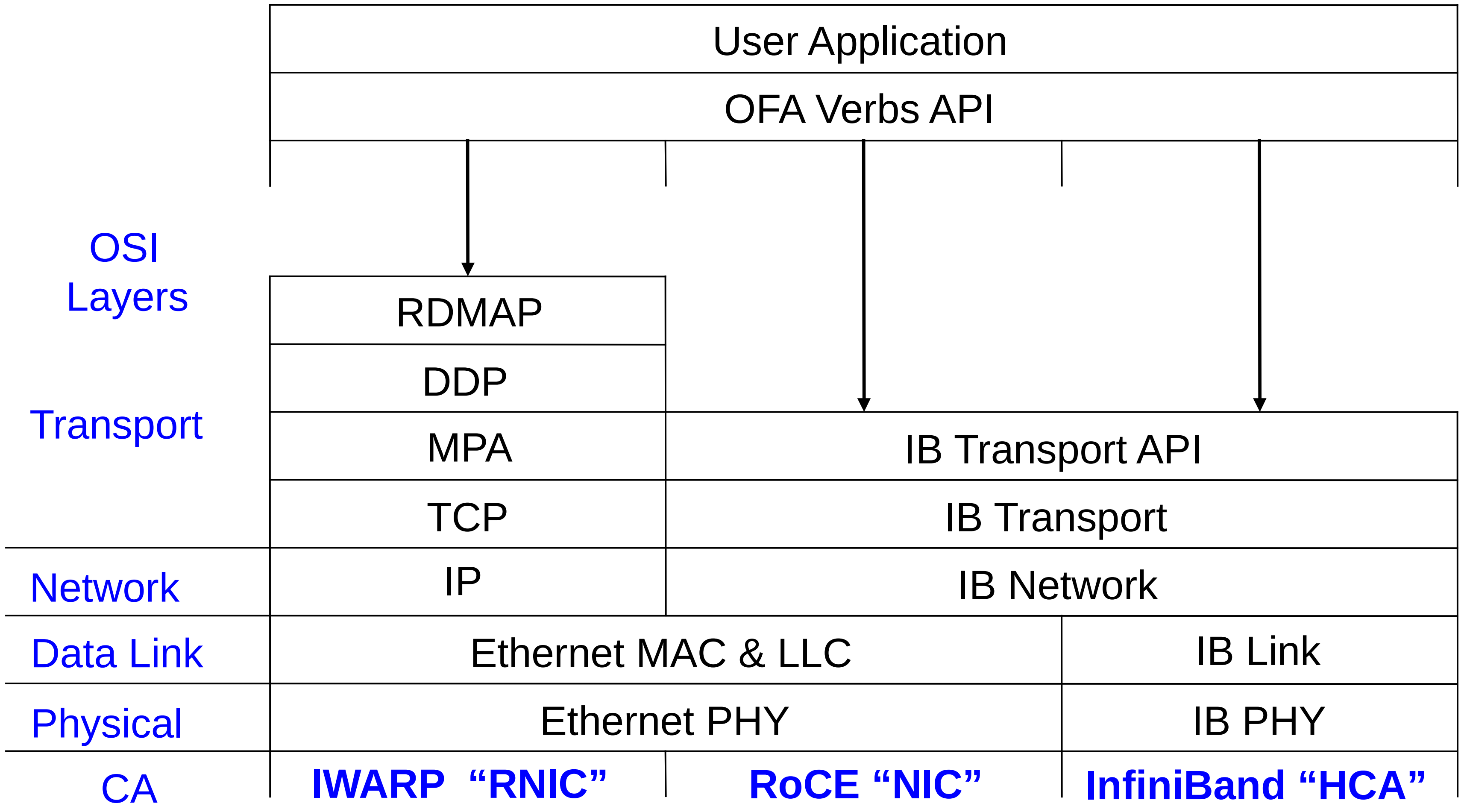
- A physical link-layer protocol for InfiniBand networks
- A higher-level programming API called the InfiniBand Verbs API

InfiniBand 需要依赖于专用的 InfiniBand 网络，因此可以提供非常好的服务质量，而 RoCE 和 iWarp 则可以基于以太网络，并使用专用的 RDMA NIC 和 Switch 来实现高服务质量。 RoCE 的两个版本中， v2 依赖于 UDP/IP 协议提供了 > 在局域网中灵活的路由和拥塞控制功能；iWarp 则是基于 TCP 协议提供了更加灵活的网络互联方式 .

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
TCP
IP
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


User Application
OFA Verbs API
OSI Layers
Transport
Network
Data Link
Physical
CA
RDMAP
DDP
MPA
TCP
IP
IB Transport API
IB Transport
IB Network
Ethernet MAC & LLC
IB Link
Ethernet PHY
IB PHY
IWARP "RNIC"
RoCE "NIC"
InfiniBand "HCA"

RDMA RoCE

RDMA over Converged Ethernet (RoCE) is a network protocol that allows remote direct memory access (RDMA) over an Ethernet network.

- **Provide InfiniBand-like performance and efficiency to ubiquitous Ethernet infrastructure.**
- **Utilize the same transport and network layers from IB**
- **InfiniBand stack and swap the link layer for Ethernet.**
- **Implement IB verbs over Ethernet.**

RDMA RoCE

There are two RoCE versions, RoCE v1 and RoCE v2.

RoCE v1(Layer 2) 运作在 Ehternet Link Layer(Layer 2) 所以 Ethertype 0x8915 ，所以正常的 Frame 大小为 1500 bytes ，而 Jumbo Frame 则是 9000 bytes 。

RoCE v2(Layer 3) 运作在 UDP/IPv4 或 UDP/IPv6 之上 (Layer 3) ，采用 UDP Port 4791 进行传输。因为 RoCE v2 的封包是座落在 Layer 3 之上可进行路由，所以有时又会称为「 Routable RoCE 」或简称「 RRoCE

RDMA softRoCE

不同于 RoCE,softRoCE 适用于任何以太环境，无需依赖 NIC, switch, L2QoS 等支持。

softRoCE 的目标是在所有支持以太网的设备上都可以部署 RDMA 传输，其实现可分成两部分，对上通过 librxe 与 RDMA stack （ libibverbs ）耦合，对下通过 rxe.ko 与 linux stack layer3 耦合，用户通过某个 eth NIC 的 UDP 隧道为虚拟的 RDMA 设备传输 RoCE 数据。

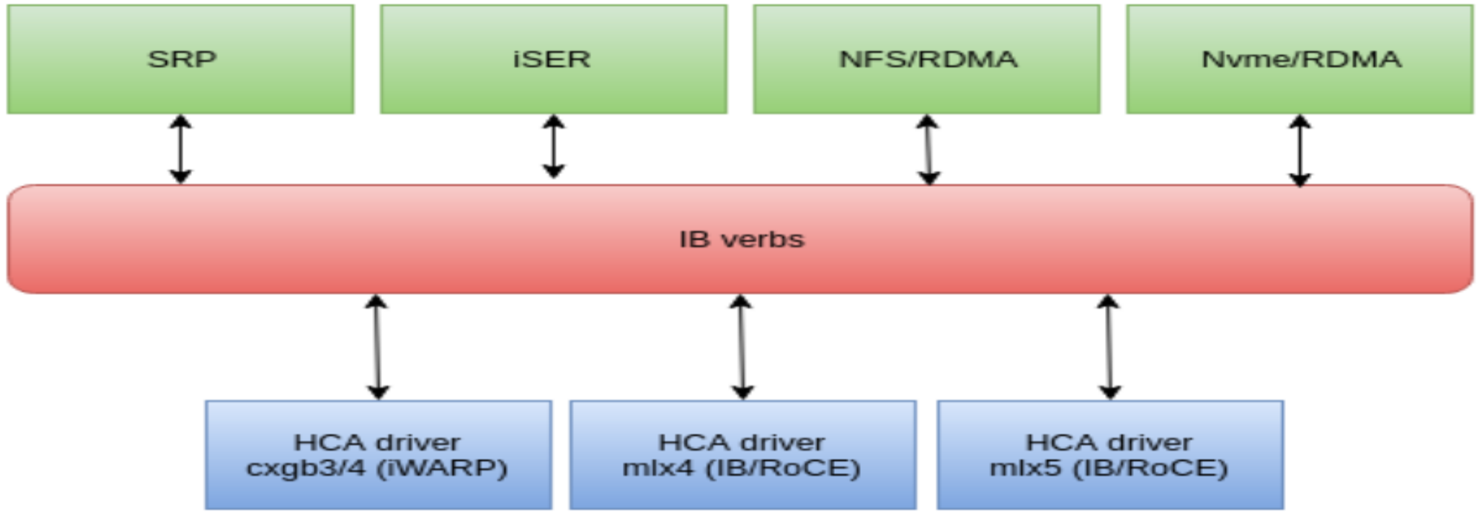
Linux RDMA

ULP

RDMA

Core layer

Device drivers


SRP
iSER
NFS/RDMA
Nvme/RDMA
IB verbs
HCA driver cxgb3/4 (iWARP)
HCA driver mlx4 (IB/RoCE)
HCA driver mlx5 (IB/RoCE)

The InfiniBand Verbs API (IB verbs) - an implementation of a remote direct memory access (RDMA) technology.

ULP – Upper layer protocol

HCA – host channel adapter, it is IB counterpart of NICs.

SRP – SCSI RDMA Protocol

iSER – iSCSI Extensions for RDMA

Linux RDMA

The source code for RDMA are in:

- drivers/infiniband/core/ - core layer
- drivers/infiniband/hw/ - hardware device drivers
- drivers/infiniband/sw/ - software drivers (e.g. Soft-RoCE)
- drivers/infiniband/ulp/ - ULP modules

The header files are located in include/rdma and include/uapi/rdma.
Also, userspace package libibverbs is needed.

PMEM

In computer science, persistent memory is any method or apparatus for efficiently storing data structures such that they can continue to be accessed using memory instructions or memory APIs even after the end of the process that created or last modified them.[1]

Often confused with non-volatile random-access memory (NVRAM), persistent memory is instead more closely linked to the concept of persistence in its emphasis on program state that exists outside the fault zone of the process that created it.

- Wiki Persistent memory

PMEM

For many years computer applications organized their data between two tiers: memory and storage. The emerging persistent memory technologies introduce a third tier.

Persistent memory (or pmem for short) is accessed like volatile memory, using processor load and store instructions, but it retains its contents across power loss like storage.

PMEM is a type of Non-Volatile Memory (NVM).

PMEM

Persistent Memory is byte addressable, which means individual memory addresses can be written with load/store instructions, whereas block addressability means using read/write instructions that access an entire block of data at the same time.

Persistent Memory is emerging as a technology that combines DRAM and NAND to accelerate I/O on the memory bus. Put simply, PM can be thought of as persistent storage in a DIMM slot. Hence we see the name NVDIMM or Non-Volatile DIMM being used.

NVDIMM

A non-volatile dual in-line memory module (NVDIMM) is a type of random-access memory for computers. Non-volatile memory is memory that retains its contents even when electrical power is removed, for example from an unexpected power loss, system crash, or normal shutdown. "Dual in-line" identifies the memory as using the DIMM package.

There are three types of NVDIMM implementations by JEDEC Standards org: NVDIMM-F, NVDIMM-D and NVDIMM-P.

The Linux source code is inside drivers/nvdimm.

NVDIMM-N

DIMM with flash storage and traditional DRAM on the same module. The computer accesses the traditional DRAM directly. In the event of a power failure, the module copies the data from the volatile traditional DRAM to the persistent flash storage, and copies it back when power is restored. It uses a small backup power source for this.

Currently the most popular and only product available on the market today, with capacities of 8, 16 and 32GB. An NVDIMM-N device can be written to as either a block or byte-addressable device.

NVDIMM-F

DIMM with flash storage. System users will need to pair the storage DIMM alongside a traditional DRAM DIMM.

This variation uses flash storage as the only storage medium. It provides block-level addressability to flash storage, with an onboard controller performing the translation task from system memory requests to the NAND storage.

NVDIMM-P

NVDIMM-P specifications will be released by JEDEC around 2018. It will allow storage inside computer main memory, using ReRAM technology, and a DDR5 interface. A NVDIMM-P has persistent DRAM, and can access external block-oriented (flash memory) drive for memory cache.

The third variation uses both DRAM and NAND, with significantly more NAND than DRAM. Although standards for NVDIMM-P haven’t been fully released, the implication is that the DRAM acts as a cache and either optimises writes to/from the NAND part or is used to improve latency.

NVDIMM

What purposes do the three products serve?

- NVDIMM-N delivers DRAM-level performance and simply adds persistence, but will be more expensive than DRAM because of the additional NAND and controller. So the extra cost here is for persistence.
- NVDIMM-F provides greater capacity and persistence at the expense of performance, but at lower pricing than DRAM.
- Bridging the gap is NVDIMM-P, which should be cheaper than DRAM (due to the ratio of NAND and DRAM onboard), offer persistence and near-DRAM performance.

NVDIMM

Disk-like NVDIMMs (Type F or P)

- Appear as disk drives to applications
- Accessed using disk stack

Memory-like NVDIMMs (Type N or P)

- Appear as memory to applications
- Applications store variables directly in RAM
- No IO or even DMA is required

Memory-like NVDIMMs are a type of persistent memory

3D XPoint

3D XPoint (pronounced three dee cross point) is a non-volatile memory (NVM) technology by Intel and Micron Technology; Initial prices are less than dynamic random-access memory (DRAM) but more than flash memory. There are SSD and DIMM memory modules based on 3D Xpoint technology.

3D XPoint (is a NVM technology) DIMMs will require server platform support because they are unlikely to operate as standard DDR4 DIMMs Intel and Micron extending the (DDR4) interface to support 3D XPoint DIMMs, so it is a proprietary approach.

DAX

The page cache is usually used to buffer reads and writes to files.
It is also used to provide the pages which are mapped into userspace by a call to mmap.

For block devices that are memory-like, the page cache pages would be unnecessary copies of the original storage. The DAX code removes the extra copy by performing reads and writes directly to the storage device. For file mappings, the storage device is mapped directly into userspace.

DAX

The DAX code currently only supports files with a block size equal to your kernel's PAGE_SIZE specially optimized for RAM based storage, avoids an extra copy-between-kernel step.

Linux DAX is a replacement for the variation of XIP (execute in place).

See drivers/dax.

NVM programming model

Current programming model:

- Data records are created in volatile memory
- Memory operations
- Copied to HDD or SSD to make them persistent I/O operations

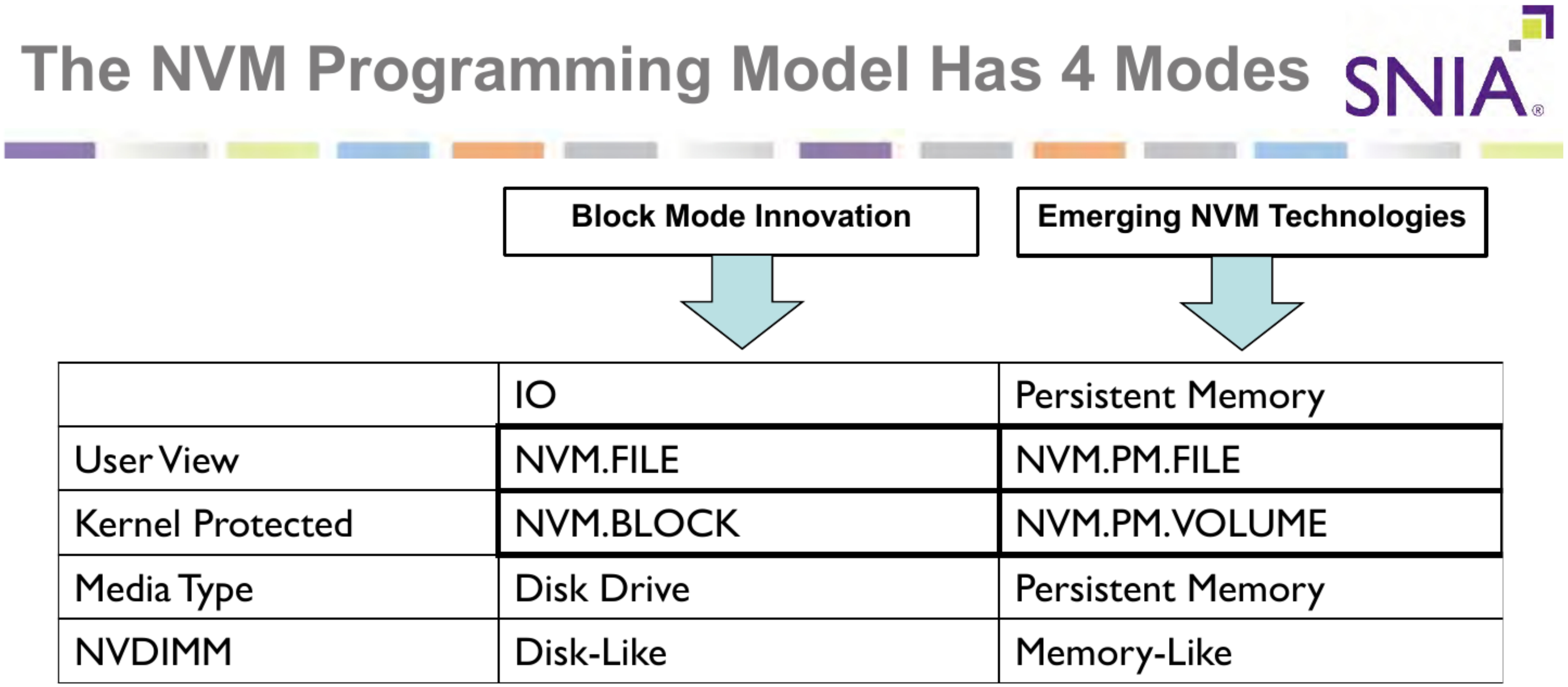
Opportunities provided by NVM devices:

- Software to skip the steps that copy data from memory to disks.
- Software can take advantage of the unique capabilities of both
- persistent memory and flash NVM

Need for a new programming model

- Application writes persistent data directly to NVM which can be treated just like RAM

NVM programming model



The NVM Programming Model Has 4 Modes

SNIA
Block Mode Innovation
Emerging NVM Technologies

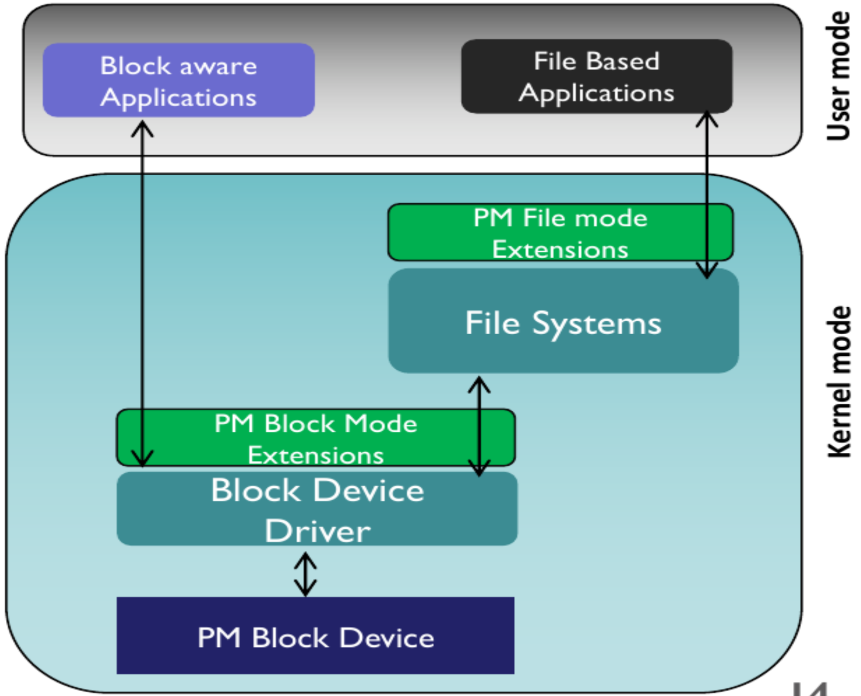
	IO	Persistent Memory
User View	NVM.FILE	NVM.PM.FILE
Kernel Protected	NVM.BLOCK	NVM.PM.VOLUME
Media Type	Disk Drive	Persistent Memory
NVDIMM	Disk-Like	Memory-Like

NVM programming model

File and Block Mode Extensions

SNIA®

- NVM.BLOCK Mode
  - Targeted for file systems and block-aware applications
  - Atomic writes
  - Length and alignment granularities
  - Thin provisioning management
- NVM.FILE Mode
  - Targeted for file based apps.
  - Discovery and use of atomic write features
  - Discovery of granularities

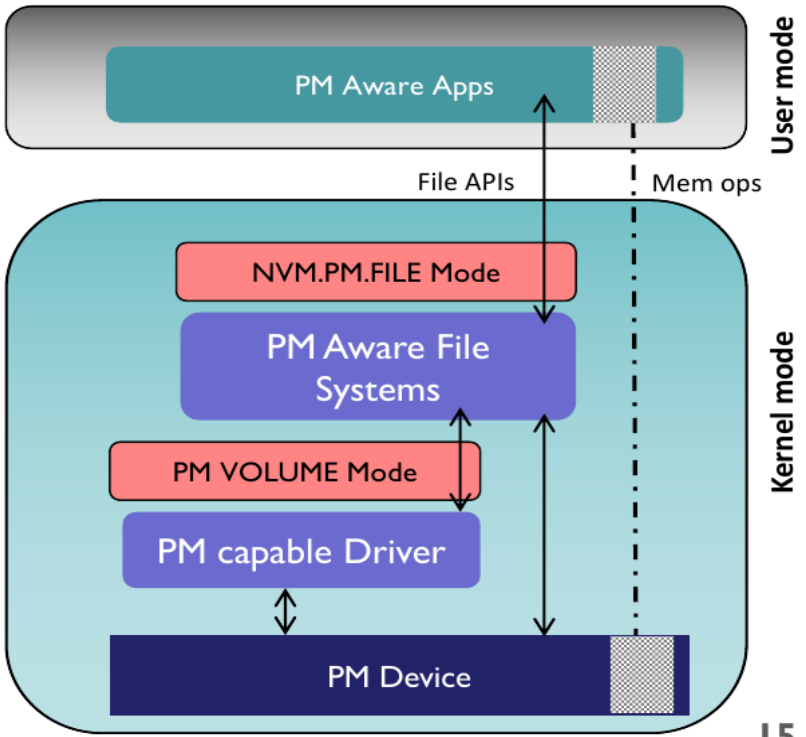


NVM programming model

Persistent Memory (PM) Modes

SNIA®

- NVM.PM.VOLUME Mode
  - Software abstraction for persistent memory hardware
  - Address ranges
  - Thin provisioning management
- NVM.PM.FILE Mode
  - Application behavior for accessing PM
  - Mapping PM files to application address space
  - Syncing PM files



Further readings

- https://en.wikipedia.org/wiki/NVM_Express
- https://en.wikipedia.org/wiki/RDMA_over_Converged_Ethernet
- http://linux-iscsi.org/wiki/InfiniBand
- https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/networking_guide/sec-configuring_soft-_roce
- http://www.cnblogs.com/rodenpark/p/6220519.html
- http://www.cnblogs.com/echo1937/p/7018266.html
- https://community.mellanox.com/docs/DOC-2504
- https://en.wikipedia.org/wiki/Persistent_memory
- https://en.wikipedia.org/wiki/NVDIMM
- https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf
- NVMe 1.3b spec

Question?

SUSE®

We adapt. You succeed.