Memory & I/O

Yi Kuo @ HPC-I

國立清華大學叢集電腦競賽團隊
Student Cluster Competition Team of NTHU

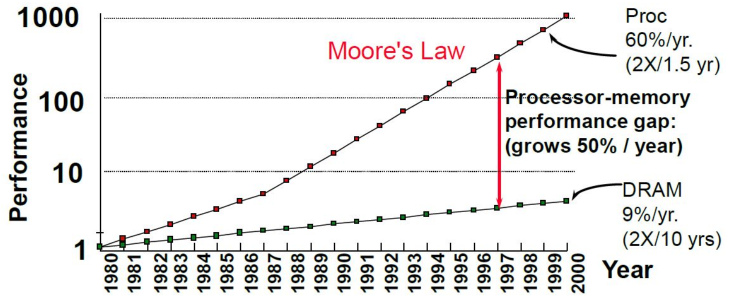
Outline

Memory

Memory Hierarchy

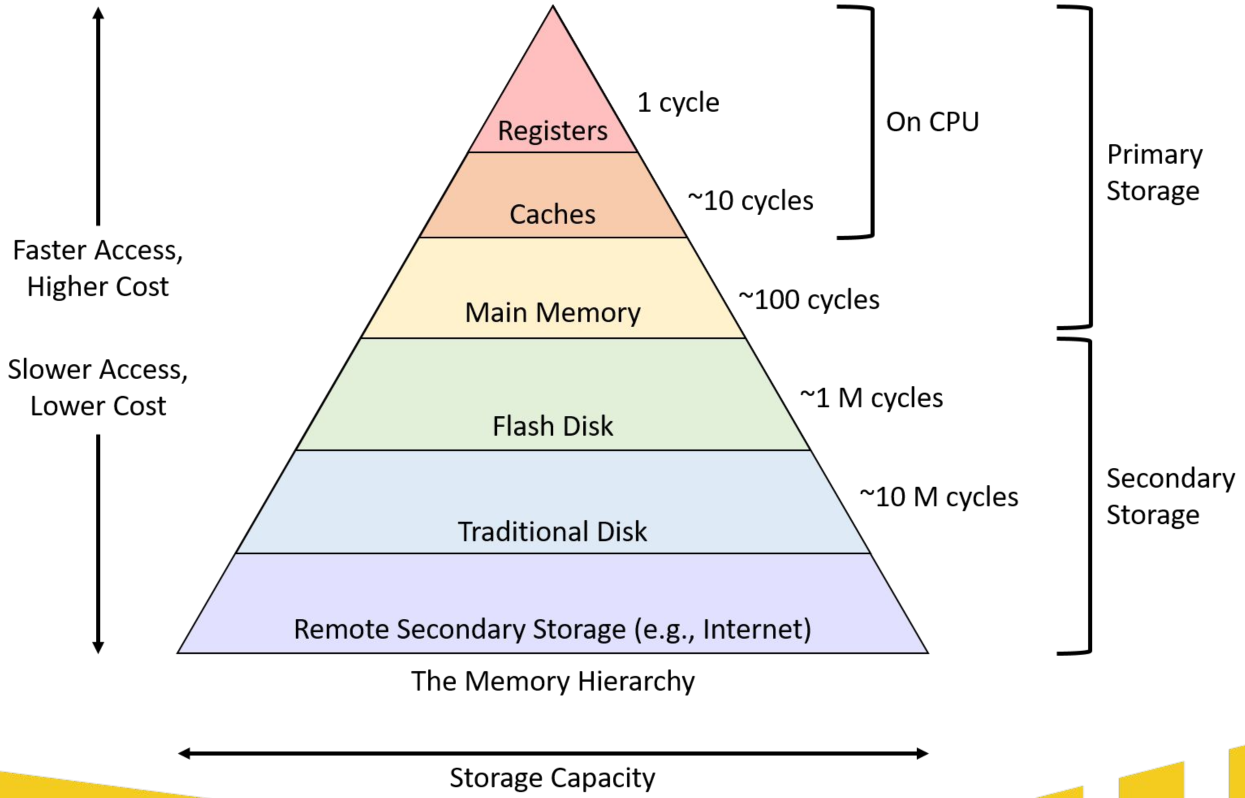
- CPUs could only access registers & load from main memory (DRAM)

- Main Memory is slow (compared to CPU)
  - Every memory access instructions would stall the CPU
  - We need cache!
    - Smaller, Faster, More expensive



- Main Memory is small & expensive (compared to disk)
  - We need to store data on secondary storage!
    - Bigger, Slower, Cheaper

Memory Hierarchy



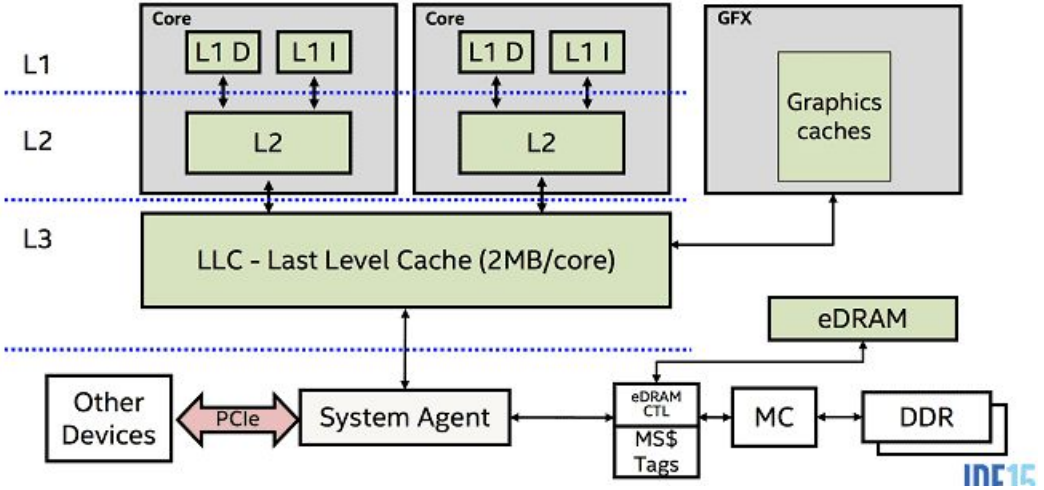
The Memory Hierarchy

CPU Caches

- Register: 1 cycle

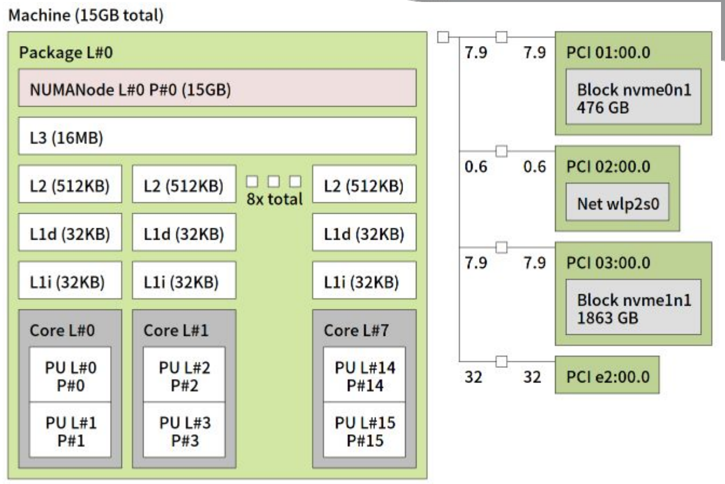
- L1 Cache: 4 cycles
  - L1i: Instruction Cache
  - L1d: Data Cache
- L2 Cache: 12 cycles
- L3 Cache (LLC): ~21 cycles
  - Shared among multiple cores

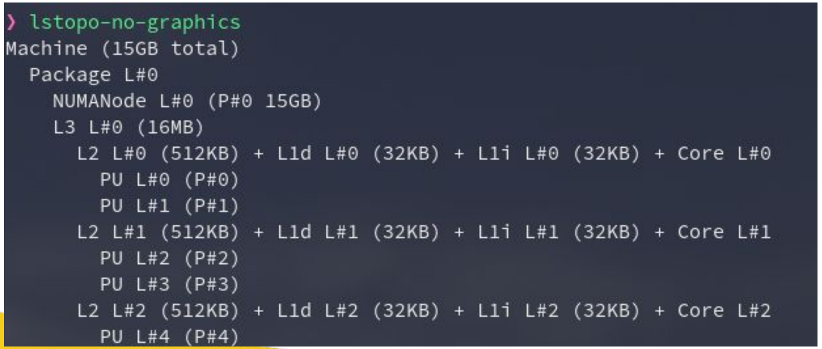
- RAM: ~117 cycles
- Disk: 10000+ cycles



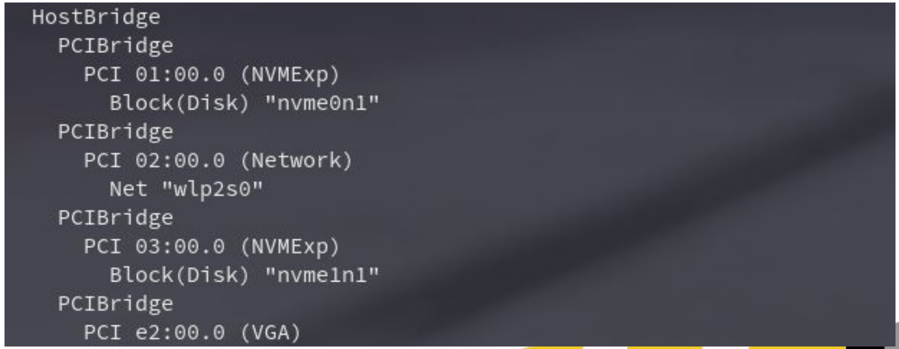
lstopo

- GUI mode
  - Requires X forwarding (ssh -Y)
- Console mode
  - Without $DISPLAY variable, or **`lstopo-no-graphics`**





```
❯ lstopo-no-graphics
Machine (15GB total)
  Package L#0
    NUMANode L#0 (P#0 15GB)
    L3 L#0 (16MB)
      L2 L#0 (512KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
        PU L#0 (P#0)
        PU L#1 (P#1)
      L2 L#1 (512KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1
        PU L#2 (P#2)
        PU L#3 (P#3)
      L2 L#2 (512KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2
        PU L#4 (P#4)
```



```
  HostBridge
    PCIBridge
      PCI 01:00.0 (NVMExp)
        Block(Disk) "nvme0n1"
    PCIBridge
      PCI 02:00.0 (Network)
        Net "wlp2s0"
    PCIBridge
      PCI 03:00.0 (NVMExp)
        Block(Disk) "nvme1n1"
    PCIBridge
      PCI e2:00.0 (VGA)
```

Cache Line & Locality

- The size of each data RAM entry is referred to “cache line size” or “block size”
- Common cache line size is 64 Bytes
  - Each time CPU loads 64 Bytes from main memory to cache
    - 16x consecutive 32-bit integers are loaded into the cache together

- Spatial Locality
  - Data that is located near each other in memory have a higher probability to be accessed together.
    - Arrays, Structures

- Temporal Locality
  - Data that is accessed repeatedly in a short duration of time is more likely to be accessed again in the near future.
    - Loops, Variables

Lab (as your Homework)

- Matrix Sum
  - https://gist.githubusercontent.com/YiPrograms/312b82fc3047ea121c697932b6b289f1/raw/mat_sum.cpp
  - `g++ mat_sum.cpp -o mat_sum`

- What's wrong with this program?
  - `perf stat -d ./mat_sum`

- Improve it according to cache locality

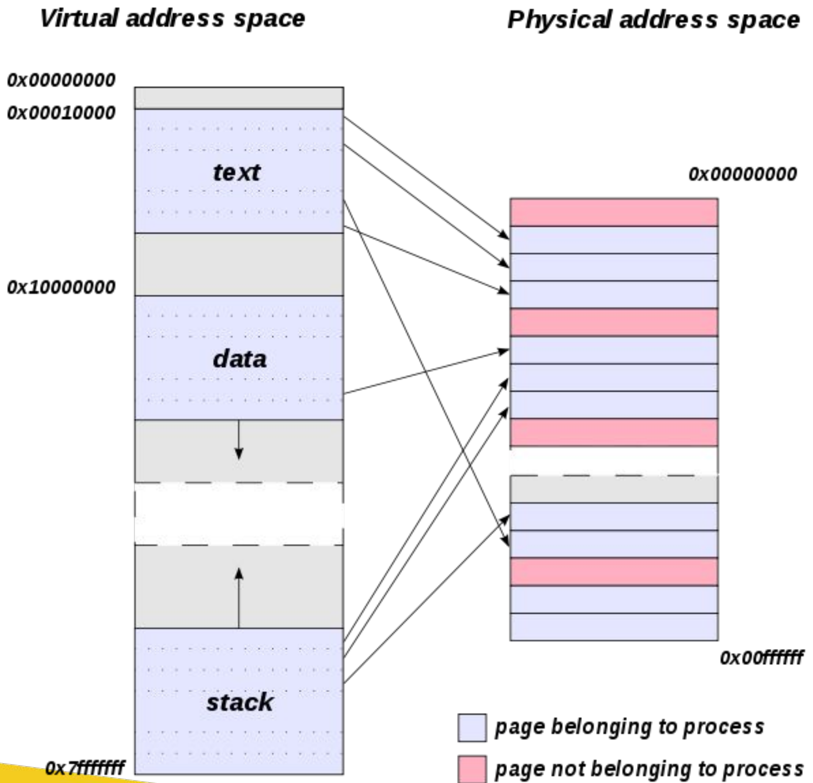
- Submission: https://docs.google.com/document/d/1S2Se8gpPpxFviomCSNQWC0ppGf8CKR6NME3sunRjFE0/edit?usp=sharing

Virtual Memory

- The main memory is a big array
- Each process would want to have contiguous memory
  - -> Fragmentation
- A process should not be able to access the memory of other processes

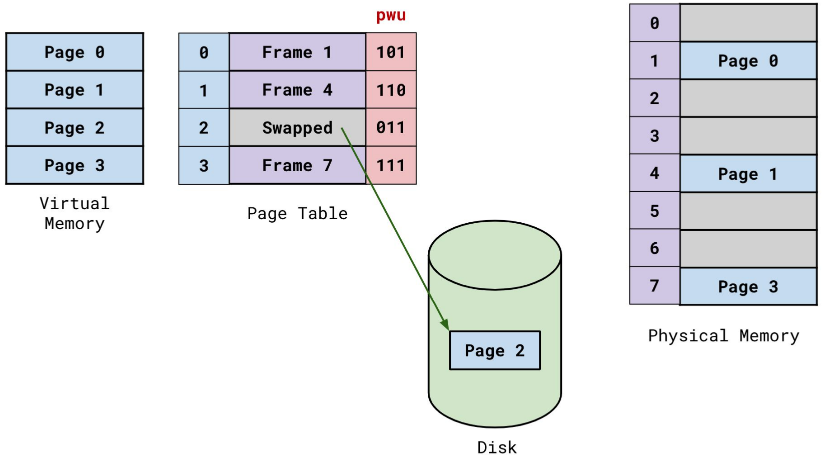
- Virtual Memory
  - Each process has its own memory address space (Virtual Address, or VA)
    - The instructions in executables use virtual addresses
  - OS stores a **Page Table** to map from virtual address to physical address
    - Every process has a page table
    - The page size is 4KB on most computers
    - **MMU hardware** reads the page table to do the translation
      - Page table in memory -> Slow -> **TLB** Cache

Virtual Memory



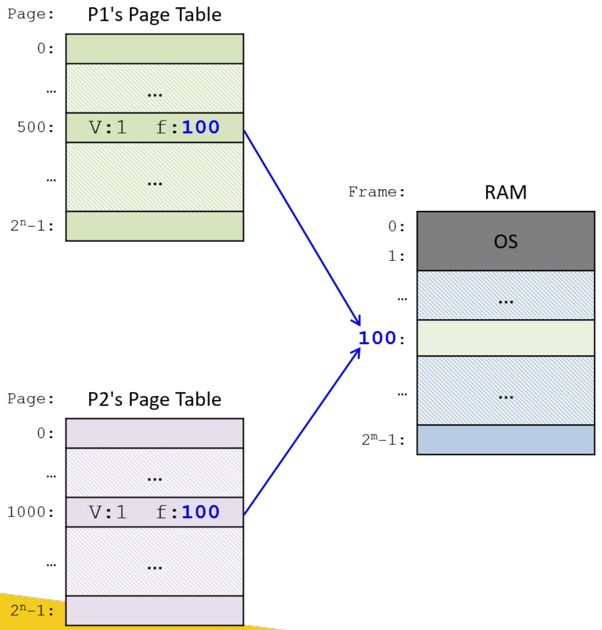
Swapping

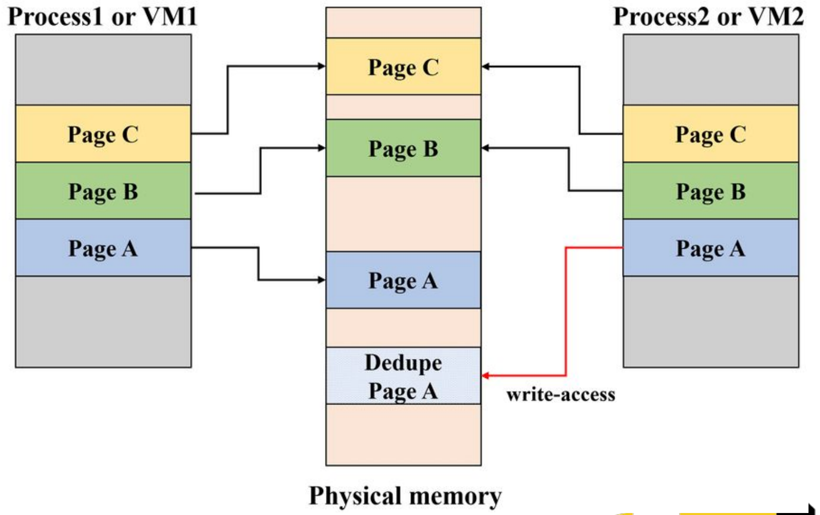
- Pages could be moved to secondary storage (e.g. Disks) to save main memory
- When the MMU failed to do the translation (the page is not in the memory, illegal access, etc.), **Page Fault** happens, and the control is handed to the OS
    - If the page is moved to secondary storage
        - OS moves the page back to main memory
        - Continue the program, and MMU would try the translation again
    - If it's an illegal access (e.g. page non-existent, writing read-only pages)
        - OS terminates the program with SIGSEGV (Segmentation Fault)



Shared Pages

- A physical page could be shared among multiple processes (page tables)
  - e.g. Shared Library, Copy on Write

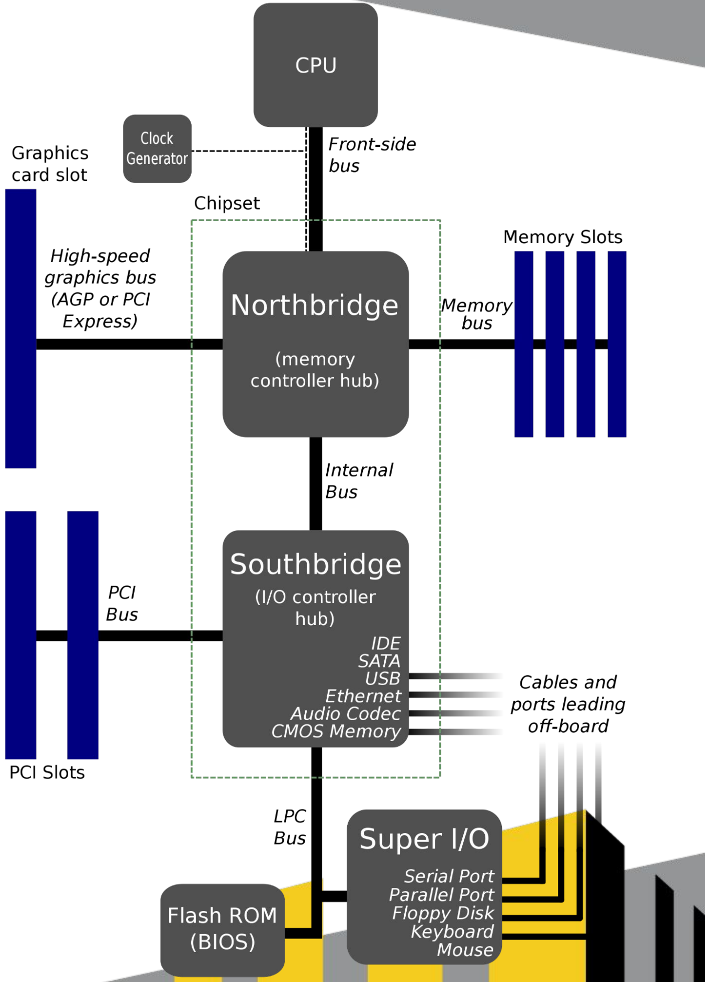




I/O

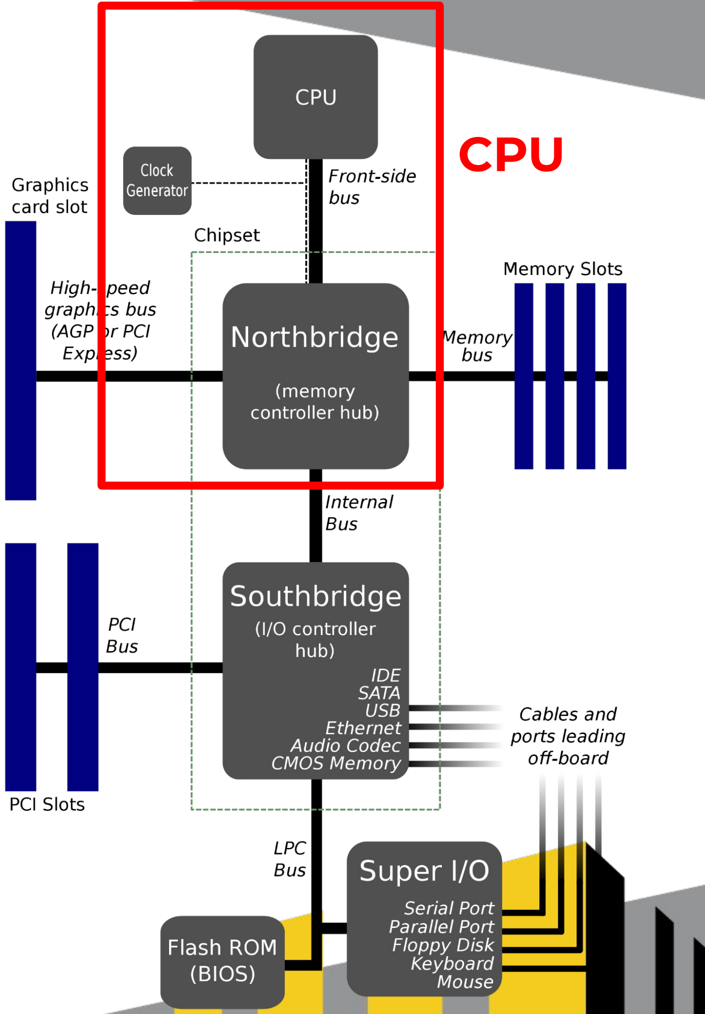
Chipset

- North Bridge
  - Memory controller
  - High speed I/O (e.g. PCIe)
- South Bridge (I/O Controller)
  - Low speed I/O
    - PCI
    - USB
    - SATA
    - etc.



Chipset

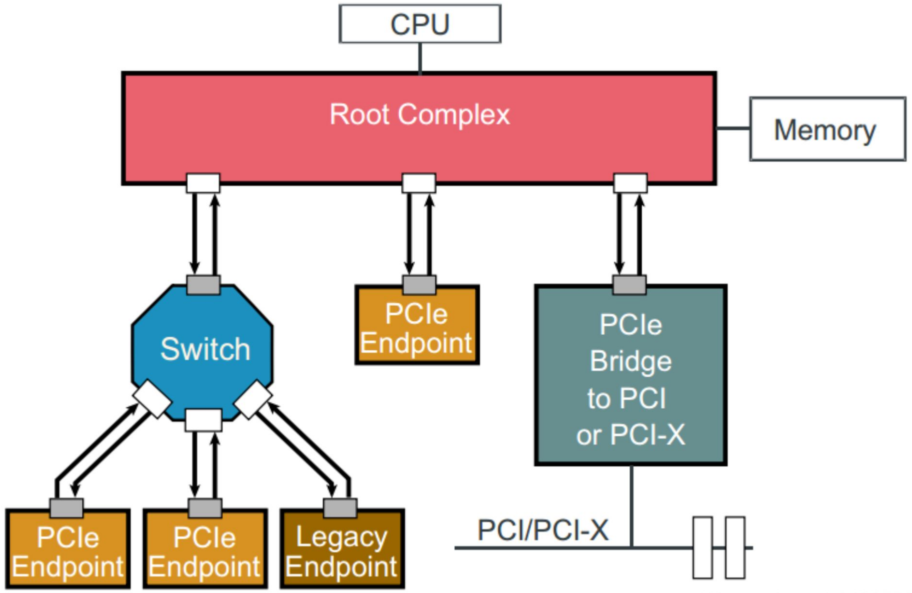
- North Bridge
  - Memory controller
  - High speed I/O (e.g. PCIe)
  - **Merged into CPU**
- South Bridge (I/O Controller)
  - Low speed I/O
    - PCI
    - USB
    - SATA
    - etc.



PCIe

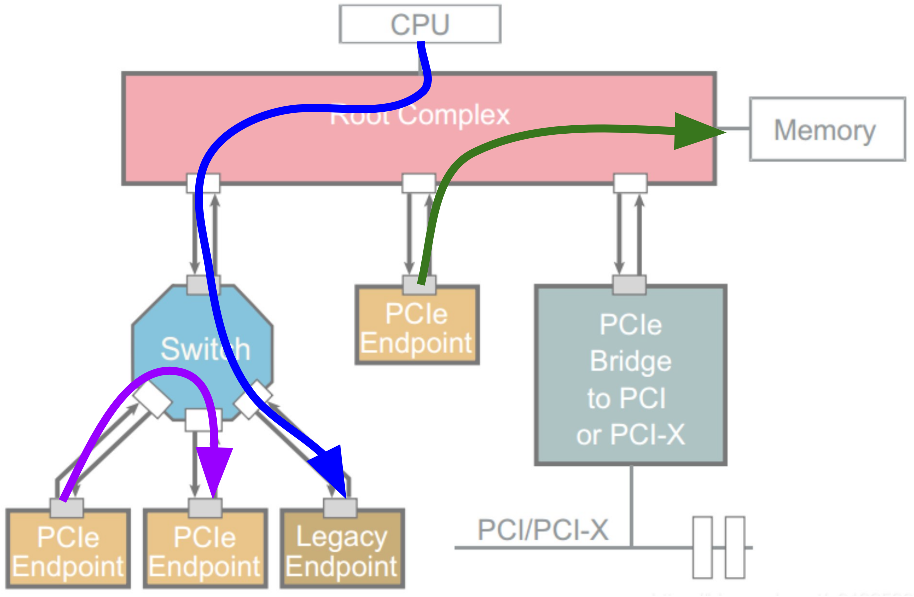
PCIe Topology

- Root Complex (RC)
  - The root of PCIe topology
  - Bridges between PCIe & CPU/other components
    - North Bridge
- PCIe Switch
  - Route PCIe packets for multiple PCIe devices
- Endpoint
  - PCIe devices (e.g. GPU, NIC)
  - Native PCIe: Memory Map only
  - Legacy PCIe: support IO requests



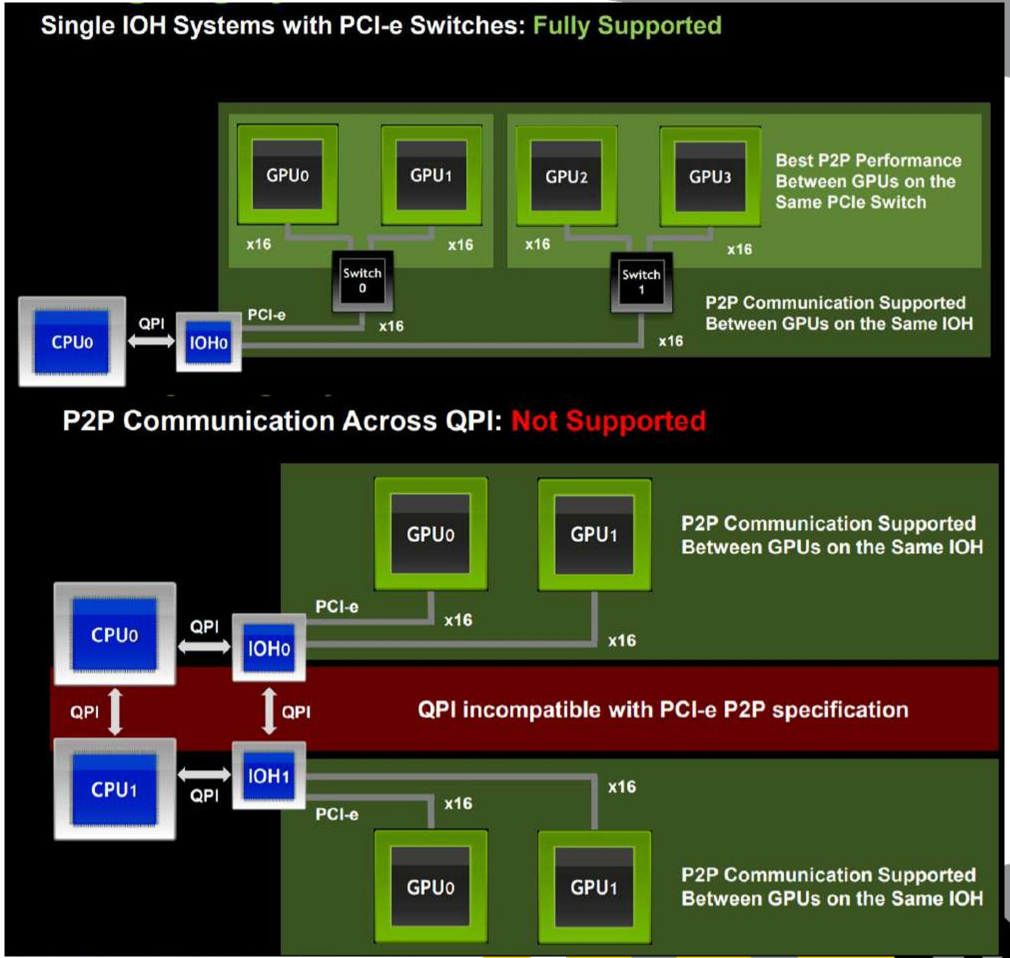
PCIe Transactions

- Programmed I/O
  - Initiated by the CPU to access PCIe devices
- DMA
  - Initiated by the endpoint
  - Directly access the memory without CPU's involvement
- P2P
  - Initiated from one endpoint, to directly access another endpoint



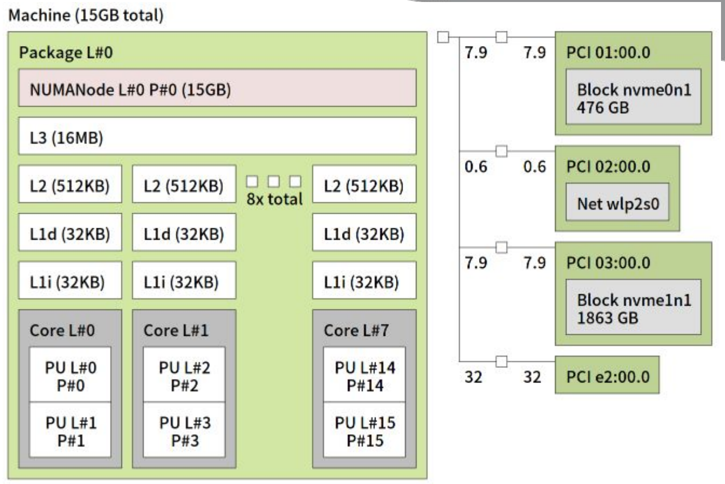
PCIe P2P

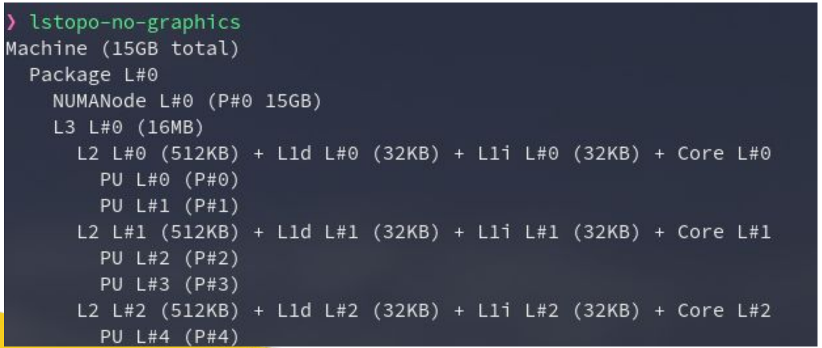
- Routing under PCIe Bridges provides the best P2P performance
- Routing via Root Complex would degrade performance (since RC is in the CPU)
- Routing between RCs are not supported by specification

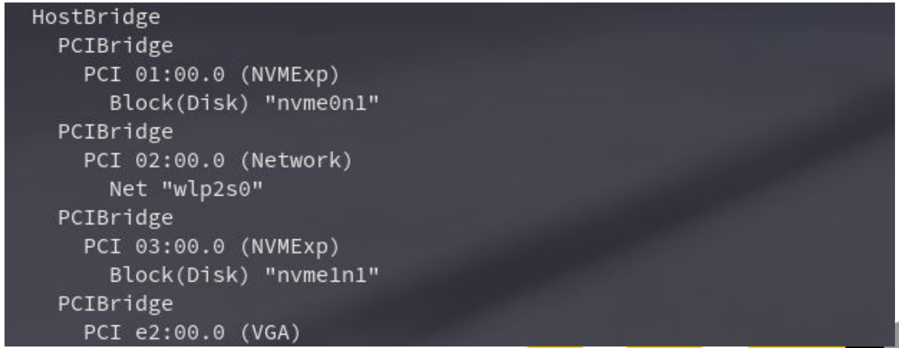

Single IOH Systems with PCI-e Switches: Fully Supported
GPU0
GPU1
GPU2
GPU3
Best P2P Performance Between GPUs on the Same PCIe Switch
x16
Switch 0
Switch 1
P2P Communication Supported Between GPUs on the Same IOH
CPU0
QPI
IOH0
PCI-e
P2P Communication Across QPI: Not Supported
P2P Communication Supported Between GPUs on the Same IOH
QPI incompatible with PCI-e P2P specification
CPU1
IOH1
P2P Communication Supported Between GPUs on the Same IOH

lstopo

- GUI mode
  - Requires X forwarding (ssh -Y)
- Console mode
  - Without $DISPLAY variable, or **`lstopo-no-graphics`**

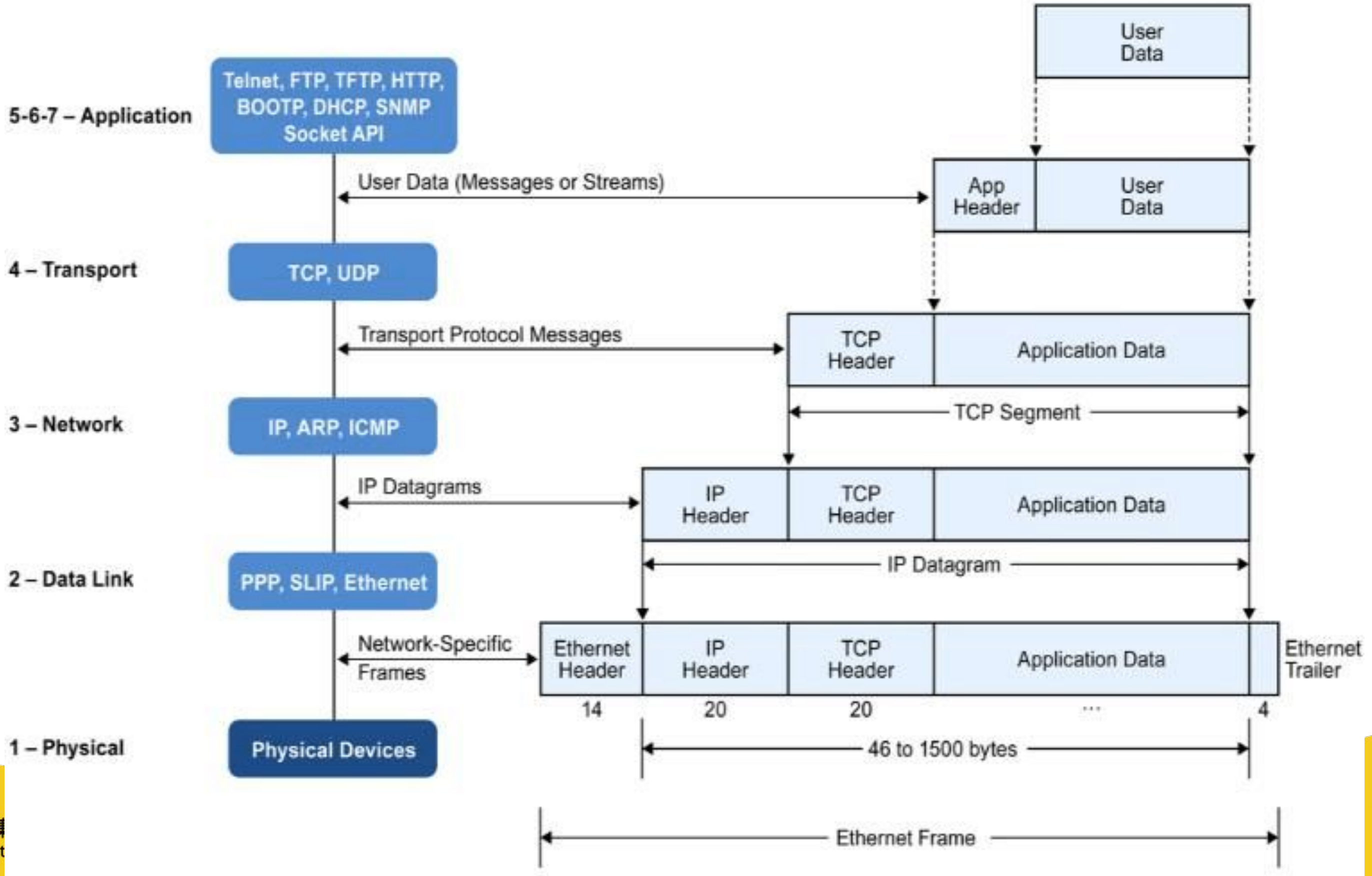



```
❯ lstopo-no-graphics
Machine (15GB total)
  Package L#0
    NUMANode L#0 (P#0 15GB)
    L3 L#0 (16MB)
      L2 L#0 (512KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
        PU L#0 (P#0)
        PU L#1 (P#1)
      L2 L#1 (512KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1
        PU L#2 (P#2)
        PU L#3 (P#3)
      L2 L#2 (512KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2
        PU L#4 (P#4)
```


```
  HostBridge
    PCIBridge
      PCI 01:00.0 (NVMExp)
        Block(Disk) "nvme0n1"
    PCIBridge
      PCI 02:00.0 (Network)
        Net "wlp2s0"
    PCIBridge
      PCI 03:00.0 (NVMExp)
        Block(Disk) "nvme1n1"
    PCIBridge
      PCI e2:00.0 (VGA)
```

Networking

Ethernet (IP Network)

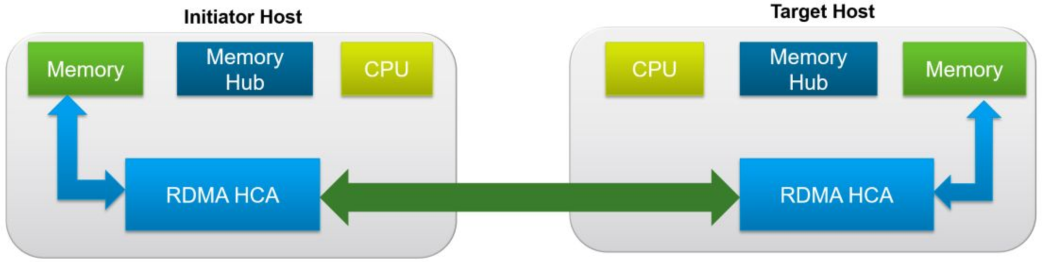


RDMA

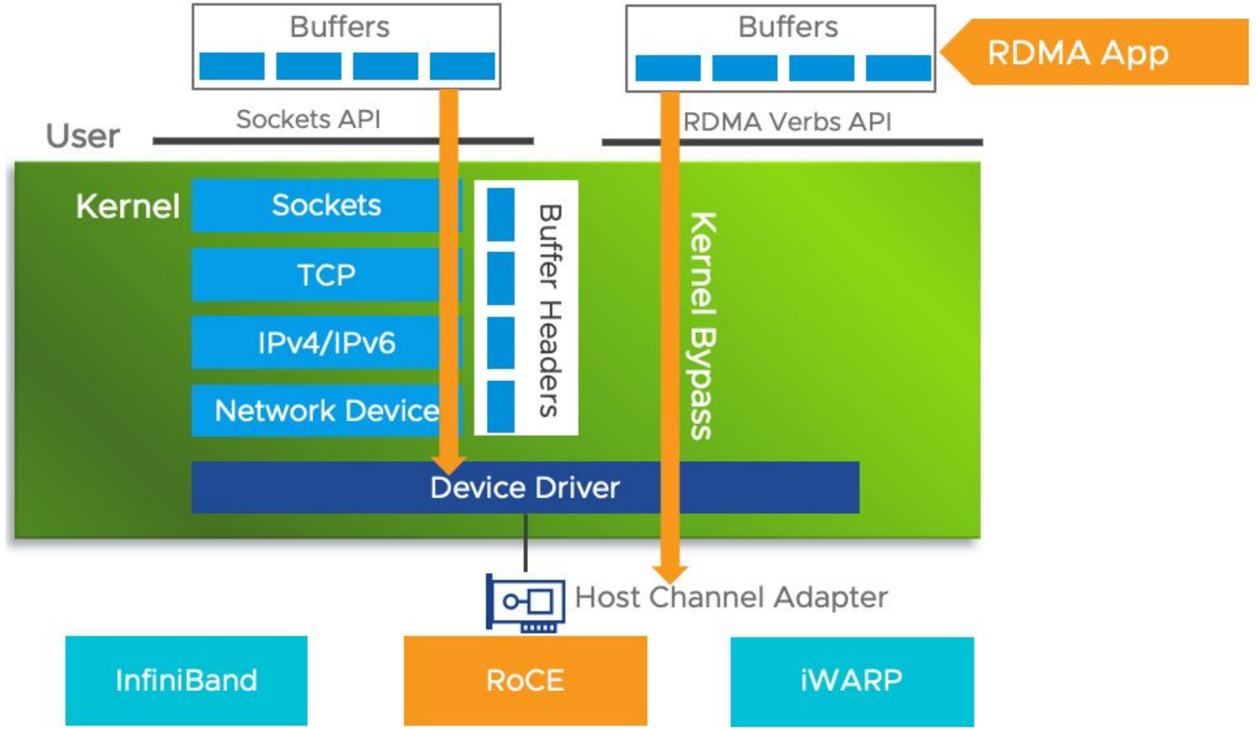
- **R**emote **D**irect **M**emory **A**ccess

- Hardware Offload
  - Network protocols for remote memory access and transfer are offloaded to the NIC
- Kernel Bypass
  - Without the involvement of Operating System, data can be sent and received directly between buffers of applications without being copied between network layers

- Protocols
  - Infiniband
  - RoCE
  - iWARP

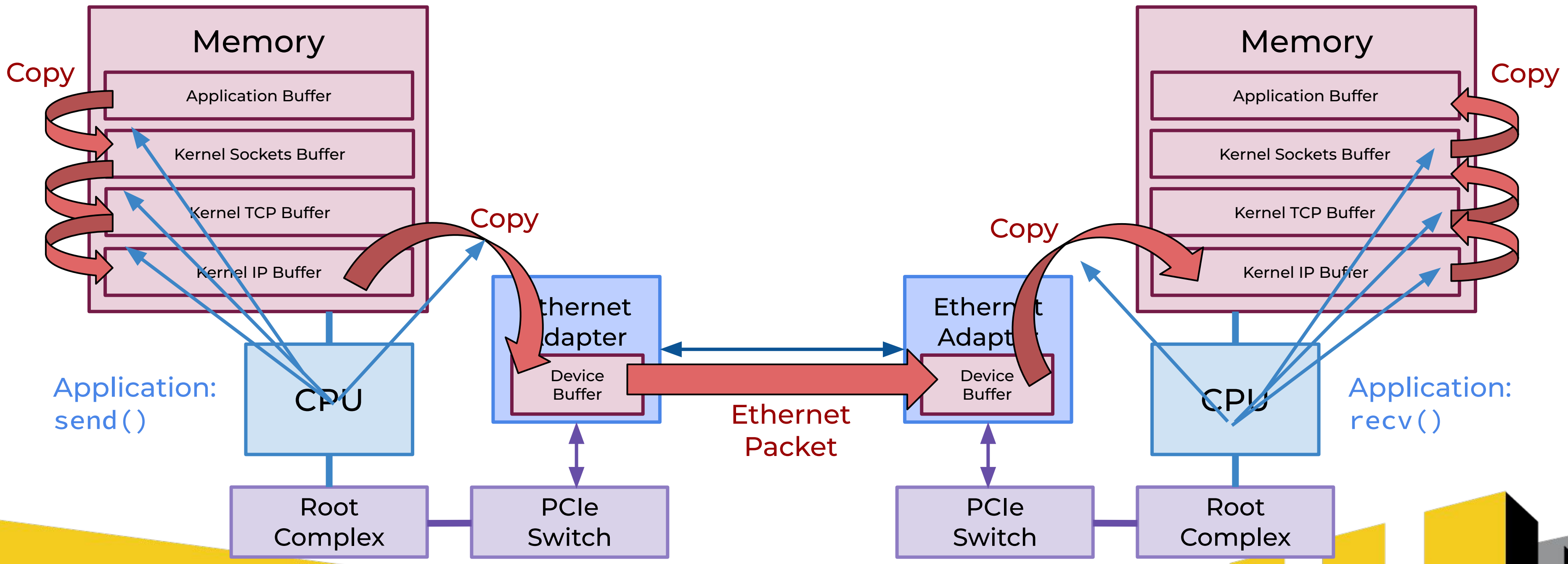


TCP vs RDMA


Buffers
Buffers
RDMA App
Sockets API
RDMA Verbs API
User
Kernel
Sockets
TCP
IPv4/IPv6
Network Device
Buffer Headers
Kernel Bypass
Device Driver
Host Channel Adapter
InfiniBand
RoCE
iWARP

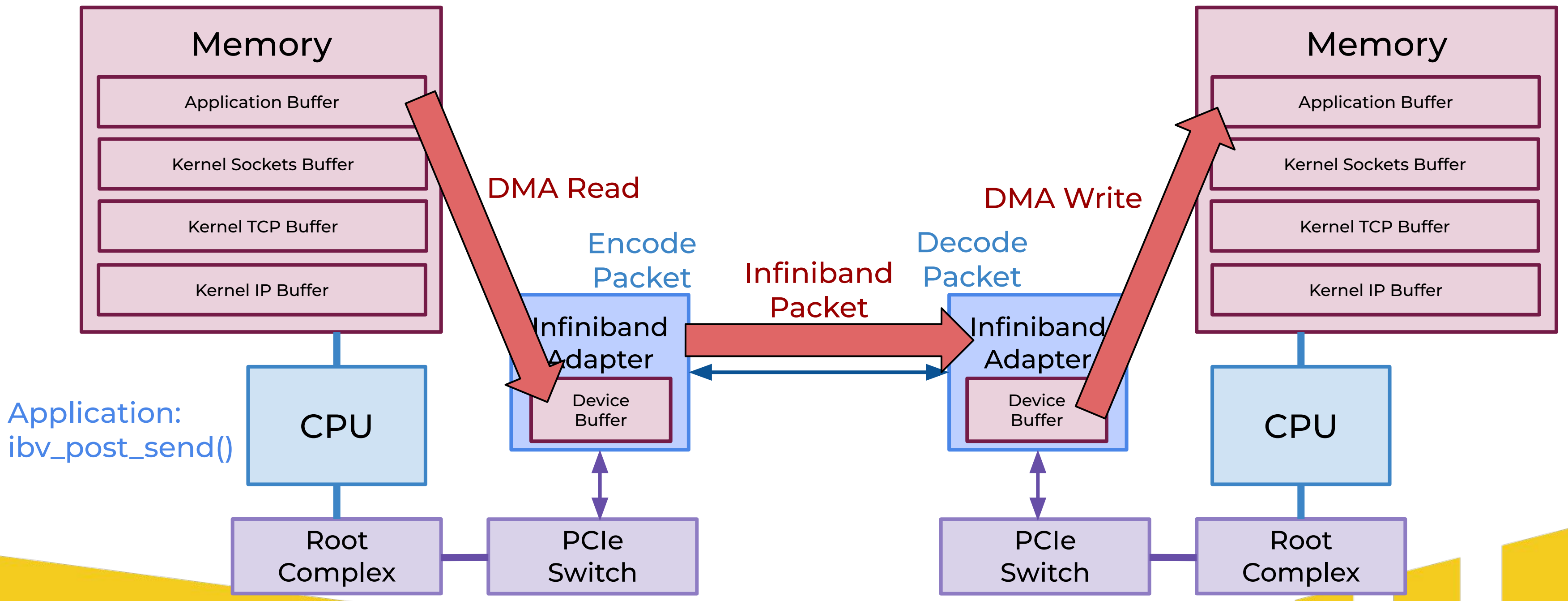
Transferring with Ethernet

- Left sends data to Right

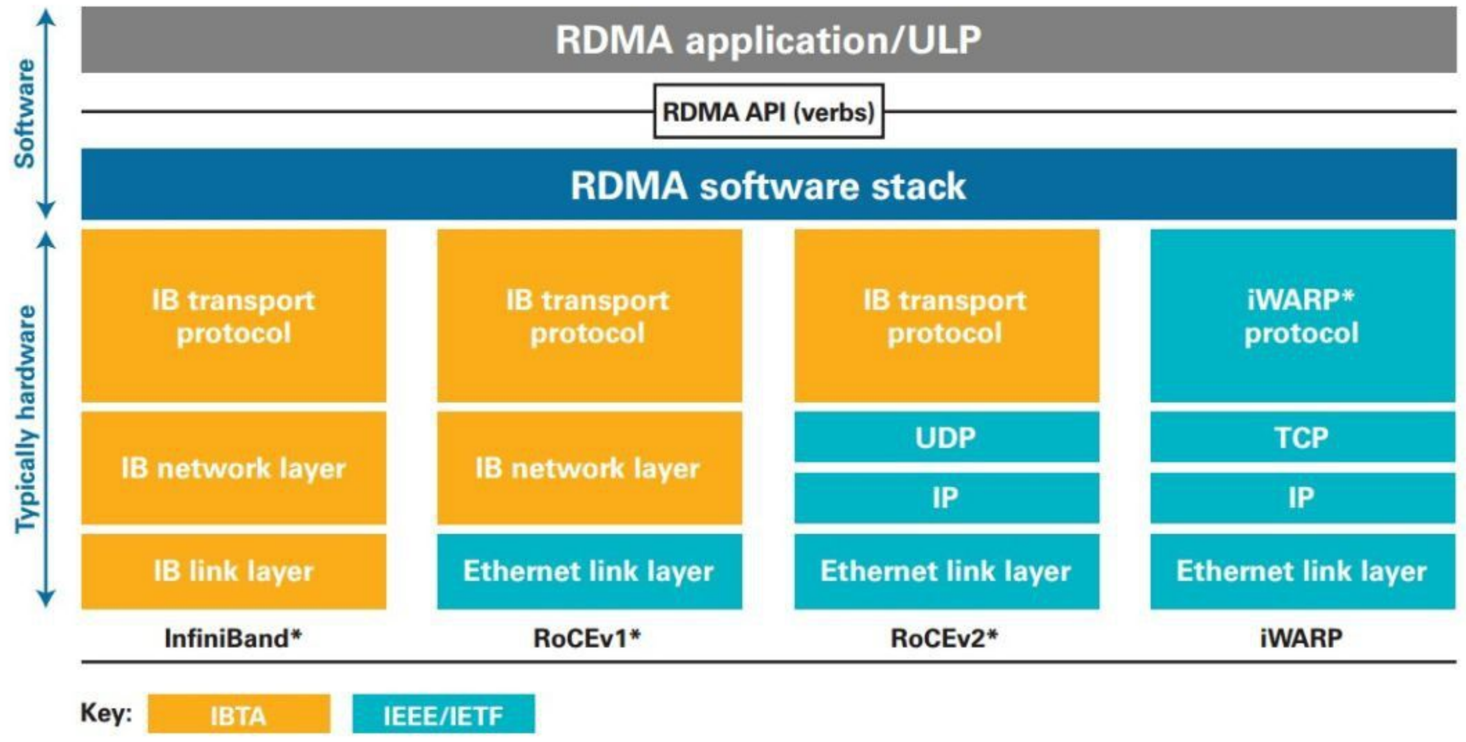


Transferring with RDMA

- Left sends data to Right



RDMA Protocols & Devices



Infiniband Data Rates

Data Rate	Throughput	
	1x	**4x (QSFP)**
SDR	2 Gb/s	**8 Gb/s**
DDR	4 Gb/s	**16 Gb/s**
QDR	8 Gb/s	**32 Gb/s**
FDR10	10 Gb/s	**40 Gb/s**
FDR	13.64 Gb/s	**54.54 Gb/s**
EDR	25 Gb/s	**100 Gb/s**
HDR	50 Gb/s	**200 Gb/s**
NDR	100 Gb/s	**400 Gb/s**
XDR	200 Gb/s	**800 Gb/s**
GDR	400 Gb/s	**1600 Gb/s**

QSFP vs SFP

NVIDIA GPUs

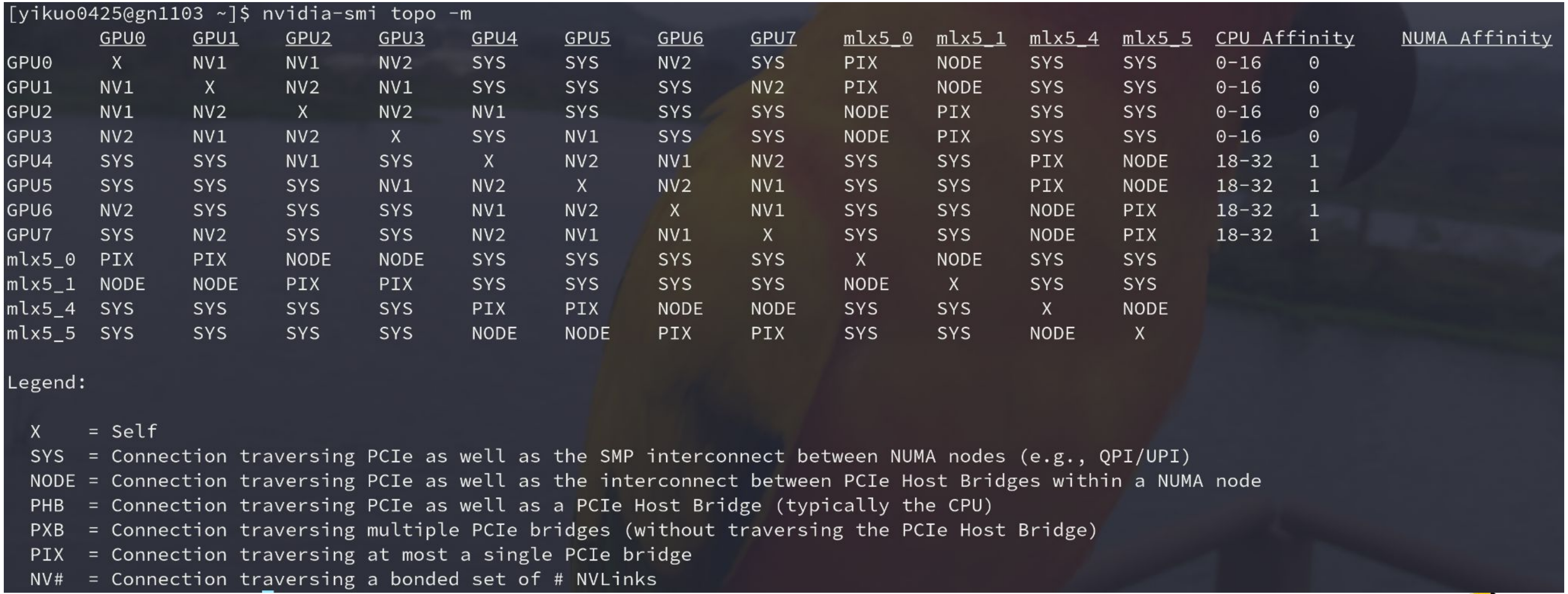
NVLINK

- NVLINK provides high-speed P2P communication between GPUs
- PCIe Version’s NVLINK could only connect pairs of nearby GPUs
- SXM Version’s NVLINK is built on the baseboard
- **`nvidia-smi nvlink -R -s`**

PCIe Version

SXM Version

nvidia-smi topo -m

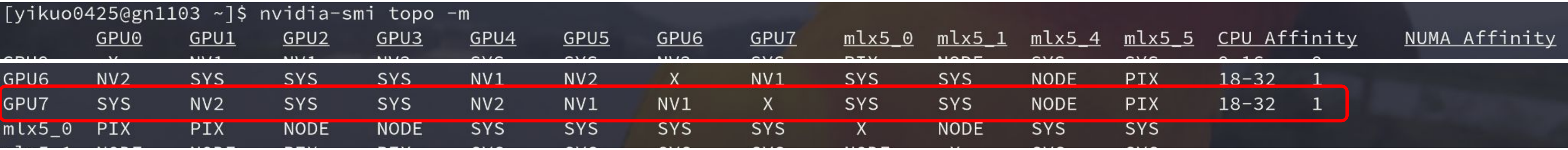


```
[yikuo0425@gn1103 ~]$ nvidia-smi topo -m
        GPU0    GPU1    GPU2    GPU3    GPU4    GPU5    GPU6    GPU7    mlx5_0  mlx5_1  mlx5_4  mlx5_5  CPU Affinity    NUMA Affinity
GPU0     X      NV1     NV1     NV2     SYS     SYS     NV2     SYS     PIX     NODE    SYS     SYS     0-16    0
GPU1    NV1      X      NV2     NV1     SYS     SYS     SYS     NV2     PIX     NODE    SYS     SYS     0-16    0
GPU2    NV1     NV2      X      NV2     NV1     SYS     SYS     SYS     NODE    PIX     SYS     SYS     0-16    0
GPU3    NV2     NV1     NV2      X      SYS     NV1     SYS     SYS     NODE    PIX     SYS     SYS     0-16    0
GPU4    SYS     SYS     NV1     SYS      X      NV2     NV1     NV2     SYS     SYS     PIX     NODE    18-32   1
GPU5    SYS     SYS     SYS     NV1     NV2      X      NV2     NV1     SYS     SYS     PIX     NODE    18-32   1
GPU6    NV2     SYS     SYS     SYS     NV1     NV2      X      NV1     SYS     SYS     NODE    PIX     18-32   1
GPU7    SYS     NV2     SYS     SYS     NV2     NV1     NV1      X      SYS     SYS     NODE    PIX     18-32   1
mlx5_0  PIX     PIX     NODE    NODE    SYS     SYS     SYS     SYS      X      NODE    SYS     SYS
mlx5_1  NODE    NODE    PIX     PIX     SYS     SYS     SYS     SYS     NODE     X      SYS     SYS
mlx5_4  SYS     SYS     SYS     SYS     PIX     PIX     NODE    NODE    SYS     SYS      X      NODE
mlx5_5  SYS     SYS     SYS     SYS     NODE    NODE    PIX     PIX     SYS     SYS     NODE     X

Legend:

  X    = Self
  SYS  = Connection traversing PCIe as well as the SMP interconnect between NUMA nodes (e.g., QPI/UPI)
  NODE = Connection traversing PCIe as well as the interconnect between PCIe Host Bridges within a NUMA node
  PHB  = Connection traversing PCIe as well as a PCIe Host Bridge (typically the CPU)
  PXB  = Connection traversing multiple PCIe bridges (without traversing the PCIe Host Bridge)
  PIX  = Connection traversing at most a single PCIe bridge
  NV#  = Connection traversing a bonded set of # NVLinks
```

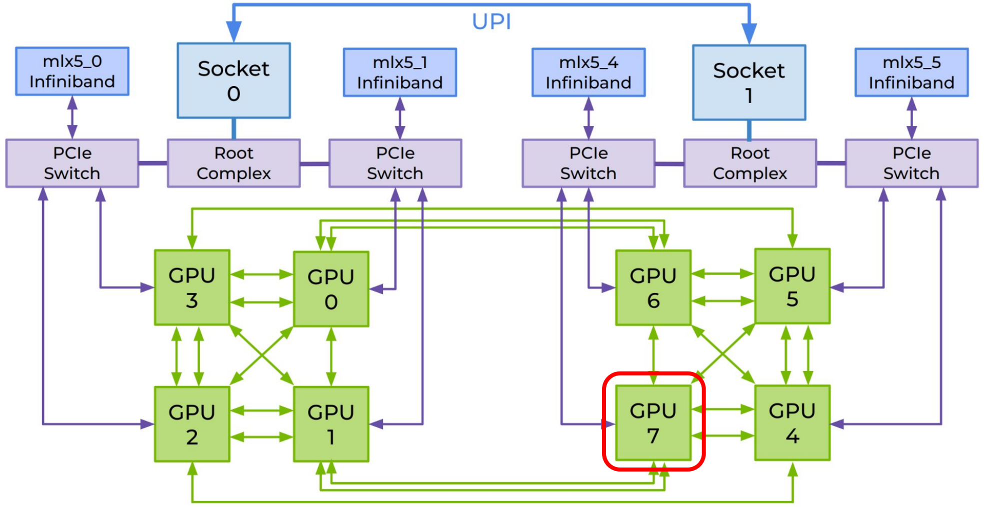
nvidia-smi topo -m



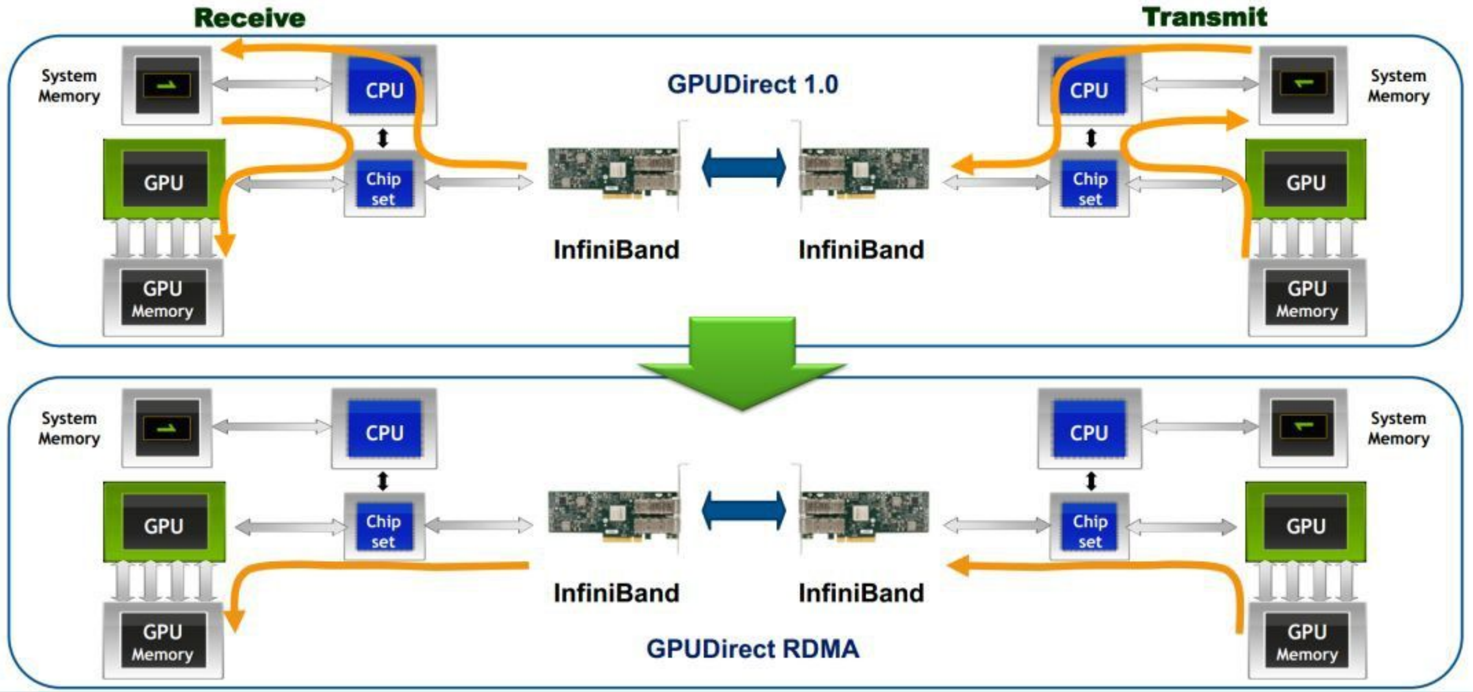
```
[yikuo0425@gn1103 ~]$ nvidia-smi  topo -m
        GPU0    GPU1    GPU2    GPU3    GPU4    GPU5    GPU6    GPU7    mlx5_0  mlx5_1  mlx5_4  mlx5_5  CPU Affinity    NUMA Affinity
GPU6    NV2     SYS     SYS     SYS     NV1     NV2      X      NV1     SYS     SYS     NODE    PIX     18-32   1
GPU7    SYS     NV2     SYS     SYS     NV2     NV1     NV1      X      SYS     SYS     NODE    PIX     18-32   1
mlx5_0  PIX     PIX     NODE    NODE    SYS     SYS     SYS     SYS      X      NODE    SYS     SYS
```

GPU7 is connected to the PCIe RC on NUMA node 1, and it would communicate to

- GPU 0, 2, 3
  - UPI over NUMA nodes
- GPU 5, 6
  - A pair of NVLINK
- GPU 1, 4
  - Two pairs of NVLINK
- mlx5_0 and mlx5_1 (IB NIC)
  - UPI over NUMA nodes
- mlx5_4
  - PCIe RC on the same NUMA node
- mlx5_5
  - PCIe Switch



GPUDirect RDMA

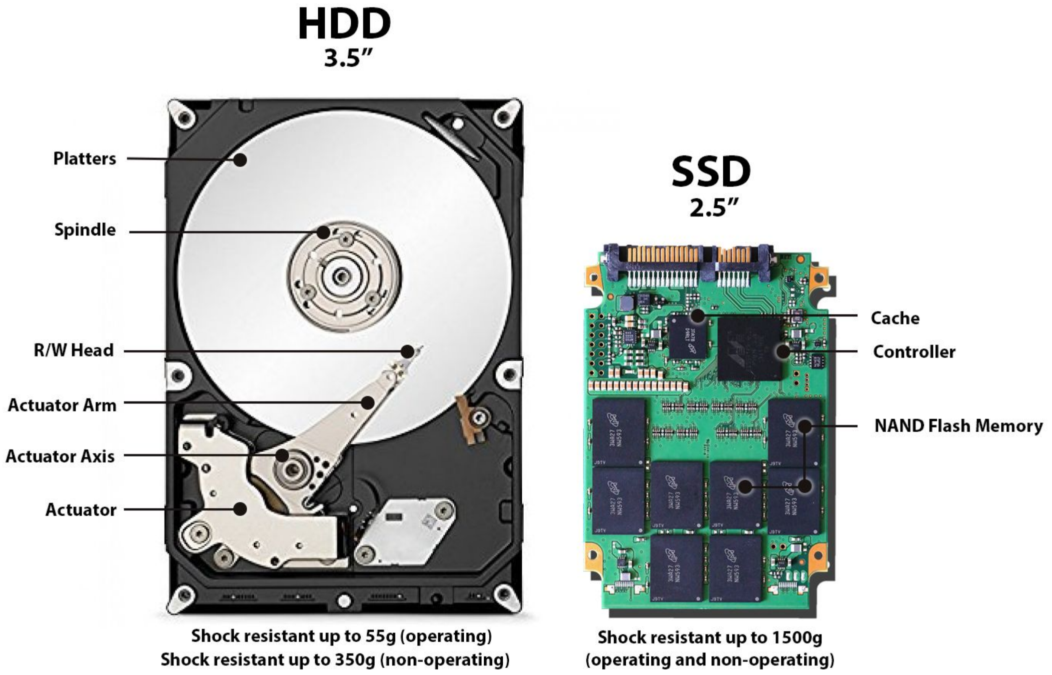


- Ensure that the HCA is on the same PCIe Switch as the GPU for best performance

Storage

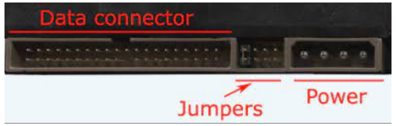
HDD & SSD

- HDD (Hard Disk Drive)
  - Mechanical Disks
  - Cheap
  - Slow
- SSD (Solid State Drive)
  - Flash Chips
  - Not so cheap
  - Fast

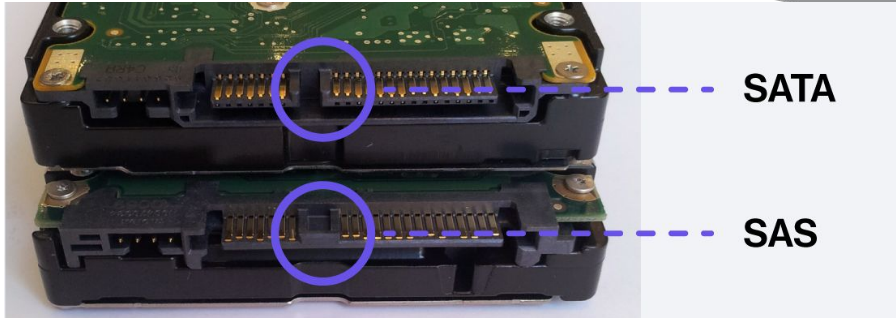

HDD
3.5″
Platters
Spindle
R/W Head
Actuator Arm
Actuator Axis
Actuator
Shock resistant up to 55g (operating)
Shock resistant up to 350g (non-operating)
SSD
2.5″
Cache
Controller
NAND Flash Memory
Shock resistant up to 1500g
(operating and non-operating)

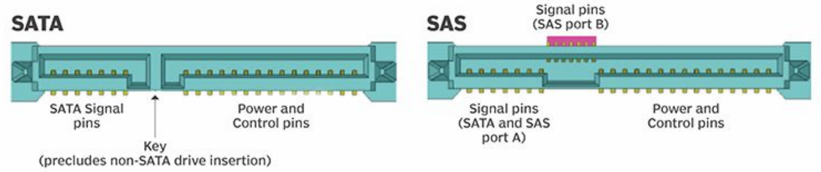
Protocols & Interfaces

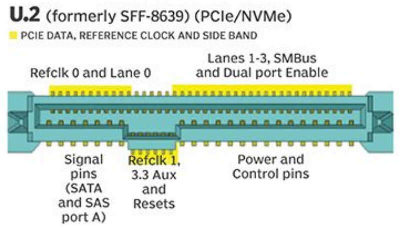
- IDE (PATA)



- SATA
- SAS
  - SAS drives are mostly used on servers, and has higher RPM.
  - SATA drives can be installed in SAS socket, but not vice versa.
- NVMe (PCIe)
  - PCIe
  - M.2 / NGFF (M Key)
  - U.2 / U.3
    - SATA & SAS drives can be installed in U.2 socket, but not vice versa.



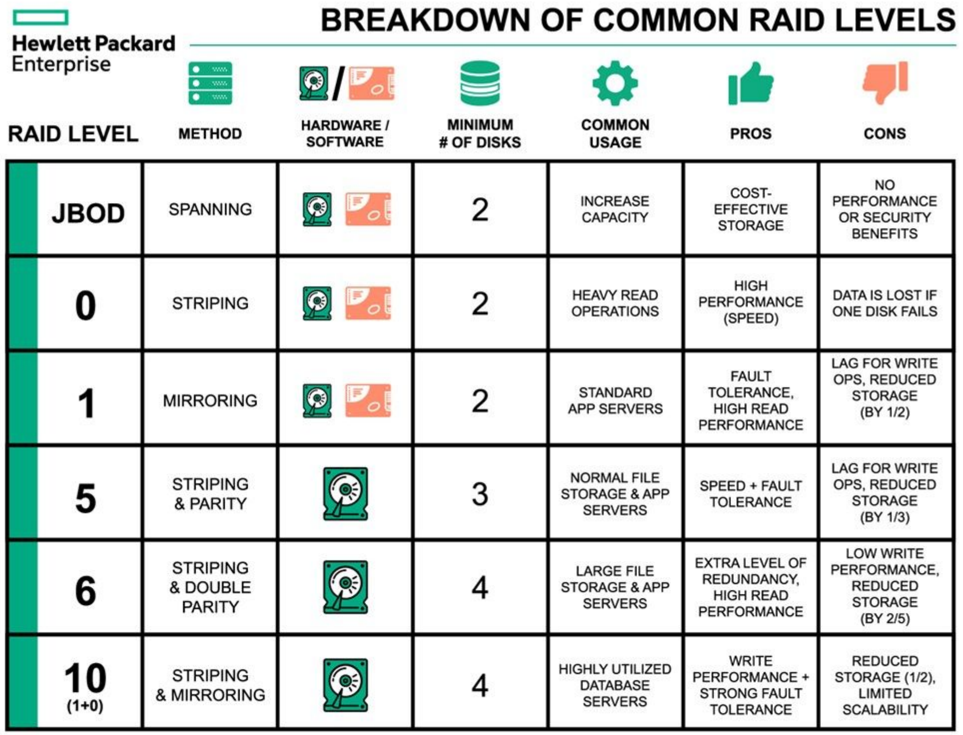



U.2 (formerly SFF-8639) (PCIe/NVMe)
PCIE DATA, REFERENCE CLOCK AND SIDE BAND
Refclk 0 and Lane 0
Lanes 1-3, SMBus and Dual port Enable
Signal pins (SATA and SAS port A)
Refclk 1, 3.3 Aux and Resets
Power and Control pins

国立清華大學叢集電腦競賽團隊
Student Cluster Competition Team of NTHU

RAID

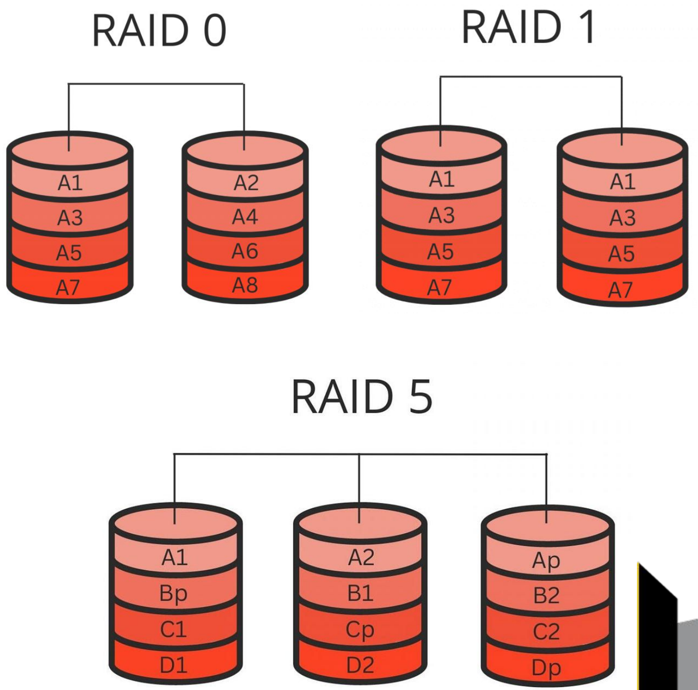
- Redundant Array of Independent Disks
  - Combining multiple disks to create a big storage, for speed & redundancy



Hewlett Packard Enterprise

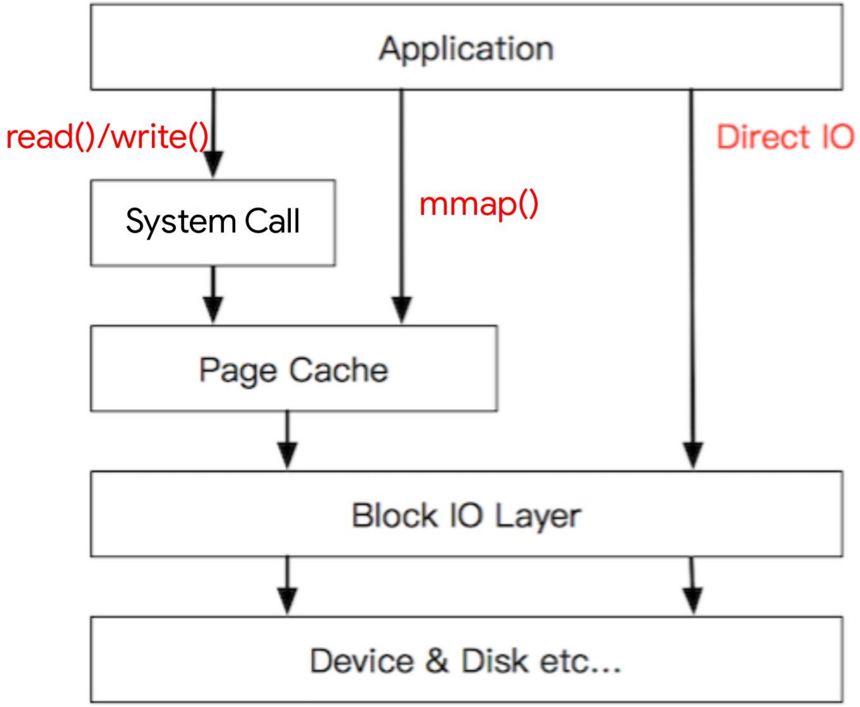
BREAKDOWN OF COMMON RAID LEVELS

RAID LEVEL	METHOD	HARDWARE / SOFTWARE	MINIMUM # OF DISKS	COMMON USAGE	PROS	CONS
JBOD	SPANNING	🖴 ▭	2	INCREASE CAPACITY	COST-EFFECTIVE STORAGE	NO PERFORMANCE OR SECURITY BENEFITS
0	STRIPING	🖴 ▭	2	HEAVY READ OPERATIONS	HIGH PERFORMANCE (SPEED)	DATA IS LOST IF ONE DISK FAILS
1	MIRRORING	🖴 ▭	2	STANDARD APP SERVERS	FAULT TOLERANCE, HIGH READ PERFORMANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/2)
5	STRIPING & PARITY	🖴	3	NORMAL FILE STORAGE & APP SERVERS	SPEED + FAULT TOLERANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/3)
6	STRIPING & DOUBLE PARITY	🖴	4	LARGE FILE STORAGE & APP SERVERS	EXTRA LEVEL OF REDUNDANCY, HIGH READ PERFORMANCE	LOW WRITE PERFORMANCE, REDUCED STORAGE (BY 2/5)
10 (1+0)	STRIPING & MIRRORING	🖴	4	HIGHLY UTILIZED DATABASE SERVERS	WRITE PERFORMANCE + STRONG FAULT TOLERANCE	REDUCED STORAGE (1/2), LIMITED SCALABILITY



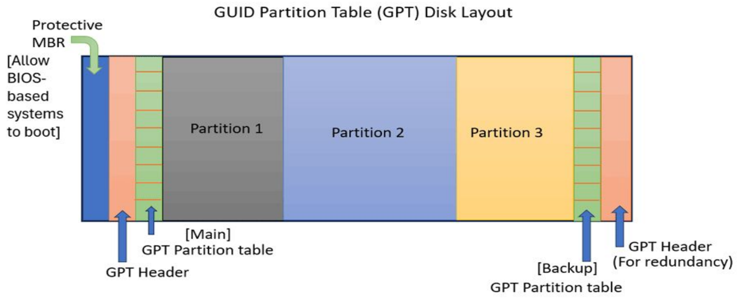
Linux File I/O

- Buffered I/O
  - Files are buffered with page caches
    - read() / write()
    - fread() / fwrite()
      - With buffer in user space
    - mmap()
      - Directly map page caches to user pages
      - Reduce the overhead of system calls
- Direct I/O (O_DIRECT)
  - Skip page caches
  - Read/Write directly from/to disk
    - Used in Database systems, which manage caches itself

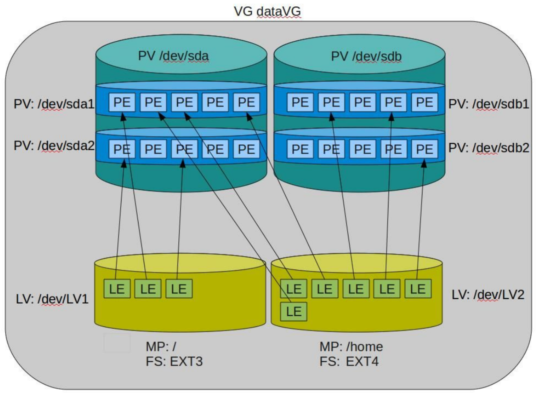


Partitions & LVM

- A disk is partitioned into multiple partitions
  - The partition table describes the partitions on the disk
    - MBR (Master Boot Record)
      - For legacy BIOS Systems
      - Max 2TB Disk
    - GPT (GUID Partition Table)
      - For UEFI Systems


GUID Partition Table (GPT) Disk Layout
Protective MBR
[Allow BIOS-based systems to boot]
Partition 1
Partition 2
Partition 3
[Main] GPT Partition table
GPT Header
[Backup] GPT Partition table
GPT Header (For redundancy)

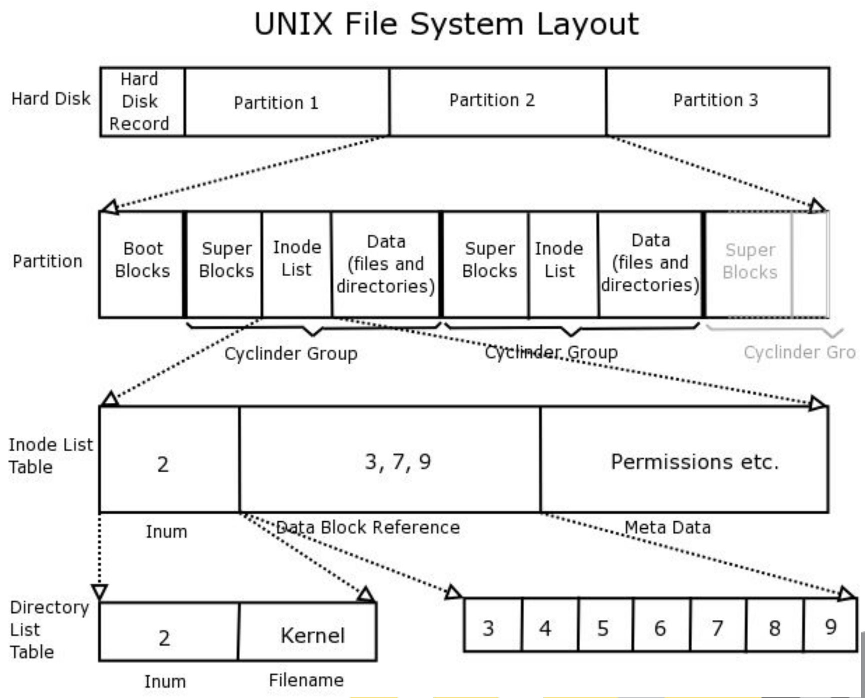
- LVM (Logical Volume Management)
  - Reducing fragmentation (Just like virtual memory)
  - Features
    - Dynamic volume resizing
    - Block level snapshot
    - RAID



File Systems

- Used to store the file & directory structures

- FAT (File Allocation Table)
- Journaling file system
  - e.g. ext3 / ext4, NTFS, HFS+
- ZFS & Btrfs
  - Support many features
    e.g. Snapshot, Compression, Encryption
- NFS (Network File System)
- tmpfs (RAM Disk)
  - `sudo mount -t tmpfs tmpfs /mnt`
- Distributed File System
  - e.g. BeeGFS, Lusture, GlusterFS


UNIX File System Layout
Hard Disk
Hard Disk Record
Partition 1
Partition 2
Partition 3
Partition
Boot Blocks
Super Blocks
Inode List
Data (files and directories)
Super Blocks
Inode List
Data (files and directories)
Super Blocks
Cyclinder Group
Cyclinder Group
Cyclinder Gro
Inode List Table
2
3, 7, 9
Permissions etc.
Inum
Data Block Reference
Meta Data
Directory List Table
2
Kernel
Inum
Filename
3
4
5
6
7
8
9