Basic on storage and FileSystem

- Stefano Cozzini
- CNR-IOM and eXact lab srl

Agenda

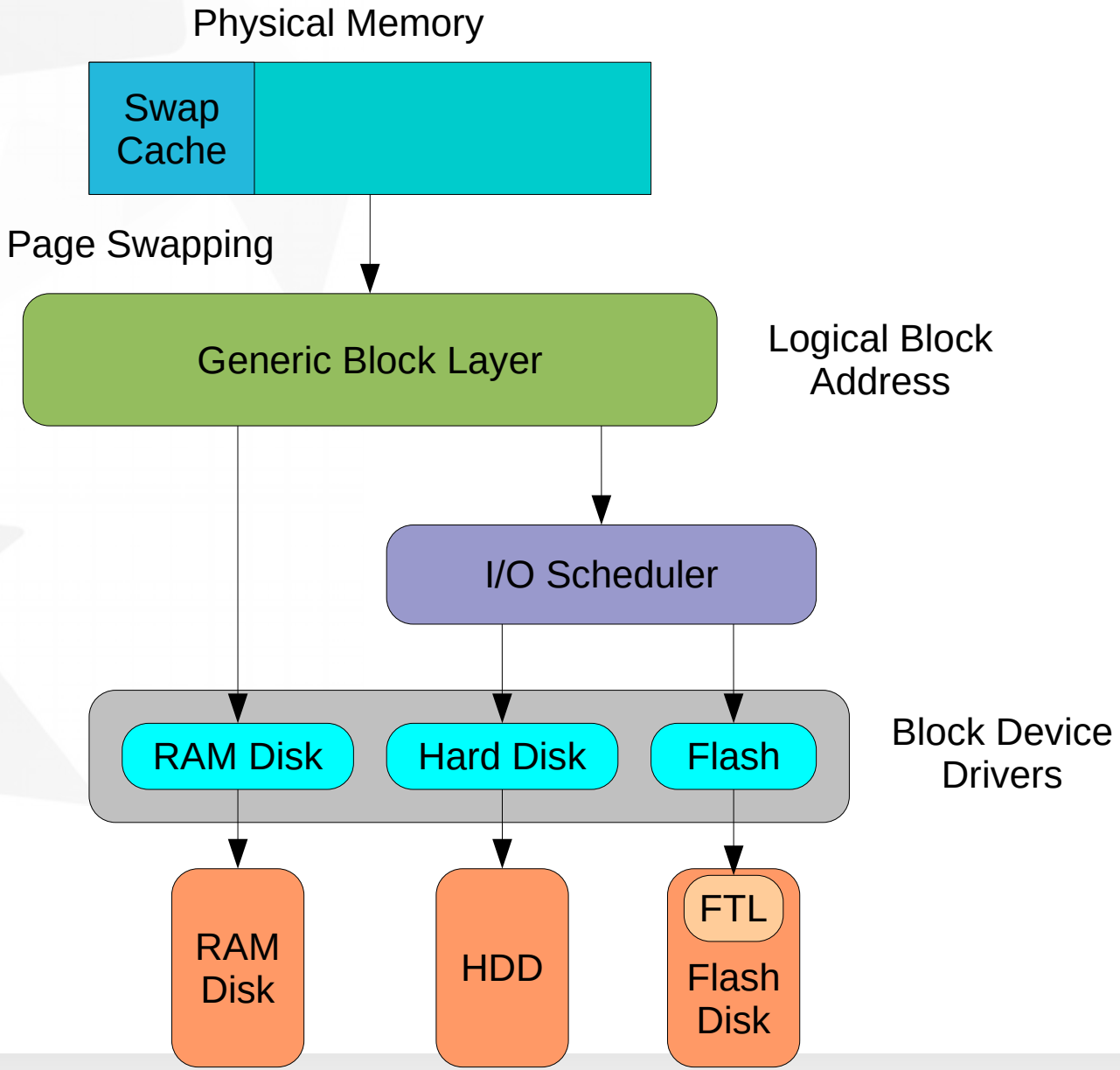
- Basic concepts on storage
- Basic concept on File Systems

Storage 101

Storage Hierarchy

- Same as with the memory hierarchy of Register -> Cache (L1->L2->L3) -> RAM storage follows a hierarchy with multiple levels:
  - RAM disk, I/O buffers or file system cache
  - Local disk (flash based, spinning disk) (SATA, SAS, RAID, SSD, JBOD, ...)
  - Local network attached device or file system server (NAS, SAN, NFS, CIFS, PFS,Lustre, GPFS, …)
  - Tape based archival system (often with disk cache)
  - External, distributed file systems (Cloud storage)

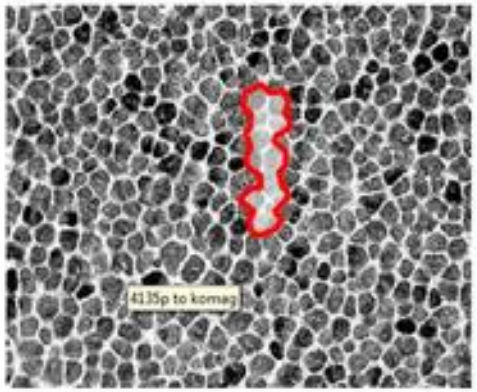
Storage Hierarchy (2)



Key metrics

- Bandwidth: volume of data read/written in a second
  → throughput metric
- IOPs: number of IO request processed by second
  → Is it a latency or a throughput metric ?
- Order of magnitudes
  - Intel v2/v3 CPU-DRAM: 80/100 GB/s
  - IB link: 5-10 GB/s
  - Hard Drive: ~100- 400 MB/s

Building storage system from bits

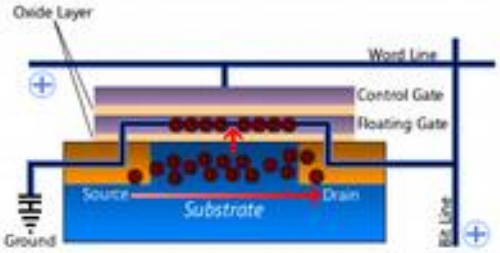


Magnetic or Solid State storage bits

Storage Devices

Software to aggregate many devices for performance

Software to handle device failures (erasure codes on blocks, across devices)



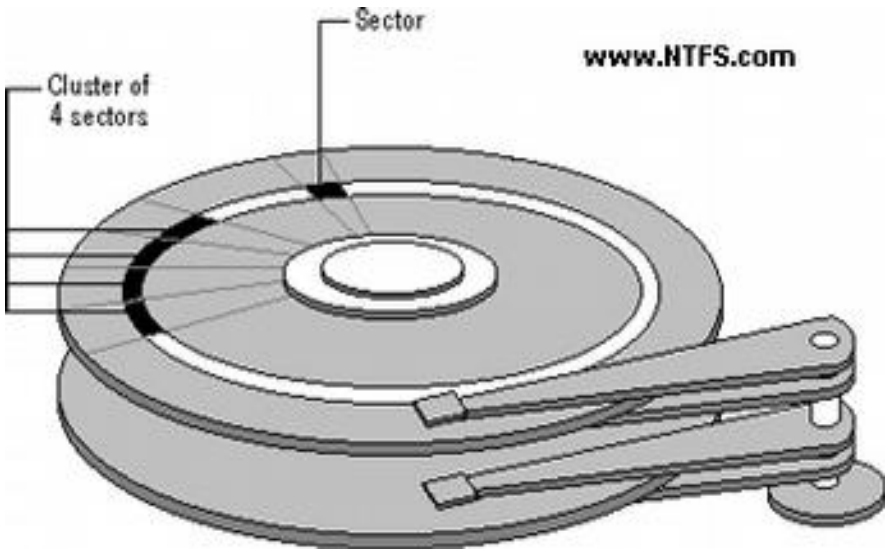
Software to handle software failures (server failover, write-ahead logging)

RAM Disk / Solid State Drive

- Unix-like OS environments very frequently create (small) temporary files in /tmp, etc.
  - faster access and less wear with RAM disk
- Linux provides “dynamic RAM disk” (*tmpfs*)
  - only existing files consume RAM
  - automatically cleared on reboot (-> volatile)
- Solid state drive is a **non**-volatile RAM disk
  - uses same interface as (spinning) hard drive
    - Battery buffered DRAM (fast, no wear, expensive)
    - Flash based (varied speed, wears out, varied cost)

HDD

- Rotating mechanical device
  - 7200, 1000, 15000 rpm.
- Head on the right track
  - (seek time) 4 ms
- Head on the right sector
  - (latency) 2ms
- Capacity: 4-12 TB
- Bandwidth: Read / Write ~ 150/250 MB/s



At constant rotating speed, where should I put my data to get max b

Current HDD technology

We can find several magnetic hard disk technologies today:

- Serial Advanced Technology Attachment (SATA)
- Serial Attached SCSI (SAS)
- Advanced Technology Attachment ([P]ATA/[E]IDE) (obsoleted by SATA)
- Small Computer System Interface (SCSI) (obsoleted by SAS)

Solid-State Drive (SSD)

pros:

- lower access time and latency
- no moving parts (silent, less susceptible to physical shock, low power consumption and heat production)
- available over SATA, SAS, PCIe, FC buses

cons:

- expensive, low capacity; usage limited to special purposes only (hardly used for big data-servers)
- limited write-cycle durability (depending on technology and ... price)
  - SLC NAND flash ~ 100K erases per cell
  - MLC NAND flash ~ 5K-30K erases per cell
  - TLC NAND flash ~ 300-500 erases per cell

SSD technology

- SLC – Single Level Cell
  - One threshold, one bit
  - 10^5 to 10^6 write cycles per page
- MLC – Multi Level Cell
  - Multiple thresholds, multiple bits (2 bits)
  - 10^4 write cycles per page
  - Denser and cheaper, but slower and less reliable
- TLC – Triple Level Cell
  - Cheapest, slowest writes
  - 500 write cycles per page!

SSD vs HDD

Latency ÷ 40: 0.1 ms vs 4 ms

Bandwidth x3: 450 MB/s vs 150 MB/s

Capacity ÷ 8/6 : 2 TB X SSD vs 12 TB (2018)

Price /Byte x10: $0.4 / GB vs $0.05 / GB

Some recent comparison

- UltraStar DC HC620 with SAS 12GB/s interface
  - Sustained transfer rate: 255 MBps read and write
- Samsung 970 Evo with PCIe 3 interface
  - Read speed 3,500 MBps
  - Write speed 2,500 MBps

https://www.enterprisestorageforum.com/storage-hardware/ssd-vs-hdd-speed.html

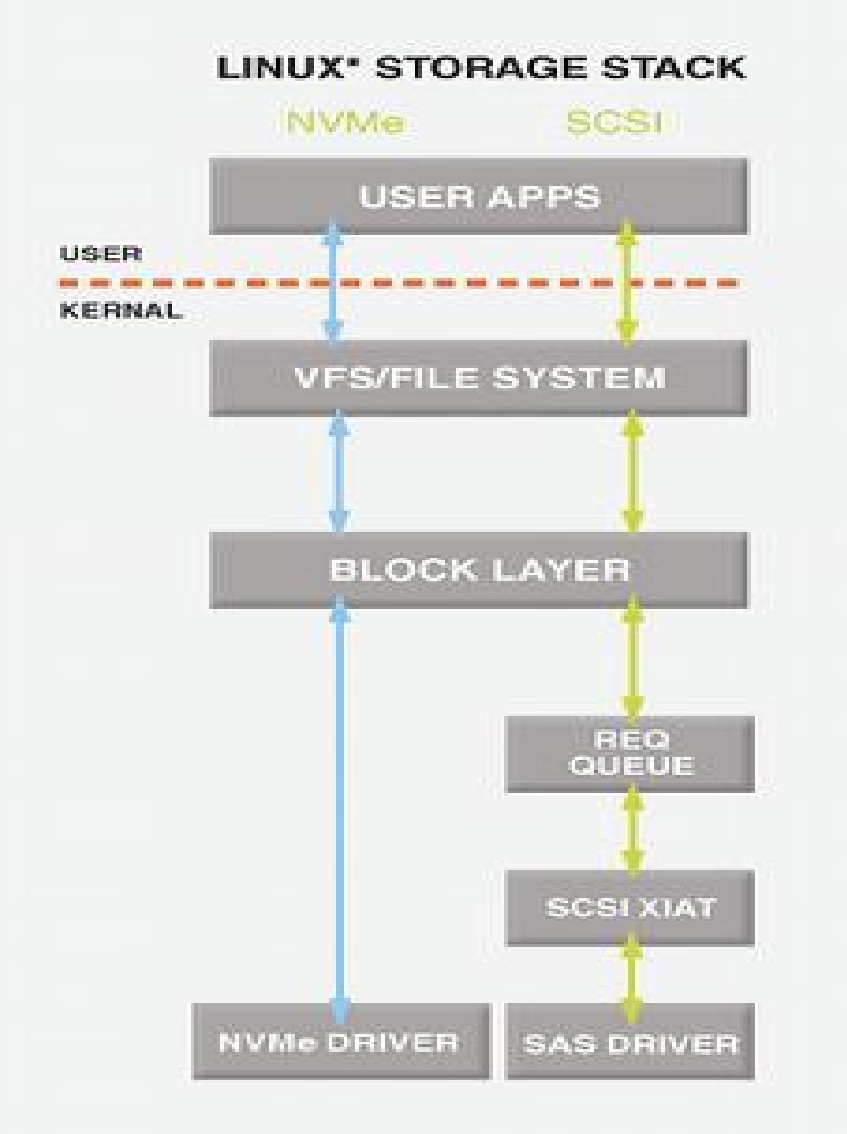
NVMe (Non-volatile Memory express)

SSD that can be connected on PCIe bus

SATA /SAS protocols designed for mechanical drive and are now the bottleneck

SAS SSD: 100 µsec - 450 MB/s

NVMe SSD: 20 µsec - 2500 MB/s



Memory/Disk prices

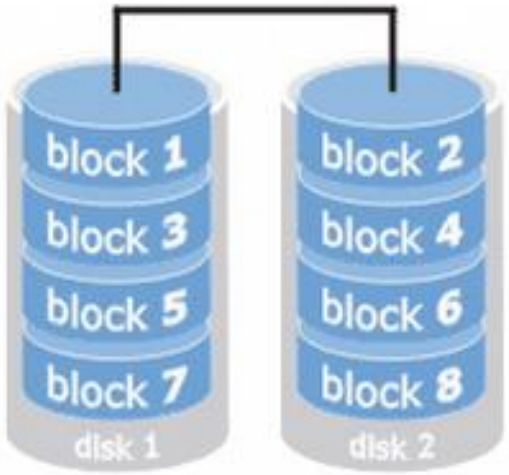
Description	$Price	1yr % chg	$/GB	
Samsung 16 GB 1600 MHz DDR3	76	-4%	$4.75	4.7
Samsung 32 GB 2133 MHz DDR4	306	+61%	$9.56	
Intel 3D Xpoint 32 GB SSD M.2 2280	80		$2.50	
Kingston 32 GB microSD U1/Class 10	13	-48%	$0.41	
SanDisk 32 GB microSD U3/Class 10	29	-41%	$1.10	
Samsung 1 TB SSD MZ-75E1T0	327	+6%	$0.33	0.33
Samsung 1 TB SSD MZ-7KE1T0BW	421	-2%	$0.42	
Seagate 2TB HDD ST2000DM006	67	-1%	$0.033	
Seagate 4TB HDD ST4000DM000	98	-16%	$0.025	0.025
Seagate 8TB 5900 RPM HDD ST8000AS0002	227	0%	$0.028	
HGST 8TB 7200 RPM HDD HUH728080ALE604	275	-42%	$0.034	
Seagate 10TB HDD ST100000VN004	340		$0.034	
HGST 10TB HDD HUH721010ALE604	350		$0.035	

Data from web search, August 22 2017

The Disk Bandwidth/Reliability Problem

Disks are slow: use lots of them in a parallel file system

However, disks are unreliable, and lot's of disks are even more unreliable



This simple two-disk system is twice as fast, but half as reliable, as a single-disk system

RAID

- RAID is a way to aggregate multiple physical devices into a larger virtual device
  - Redundant Array of Inexpensive Disks
  - Redundant Array of Independent Devices
- Invented by Patterson, Gibson, Katz, et al
  - http://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf
- Redundant data is computed and stored so the system can recover from disk failures
  - RAID was invented for bandwidth
  - RAID was successful because of its reliability

RAID reliability and performance..

Reliability or performance (or both) can be increased using different RAID “levels”.

Let us examine some of the most important:

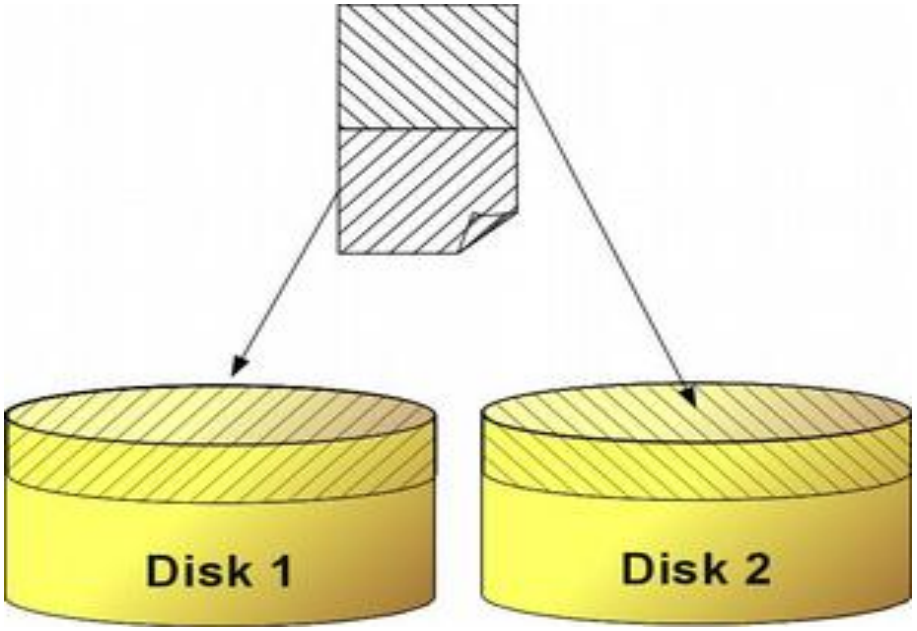
Definitions:

- S: Hard disk drive size.
- N: Number of hard disk drives in the array.
- P: Average performance of a single hard disk drive (MB/sec).

RAID 0 : striping

Performance = P * N

Capacity = N * S

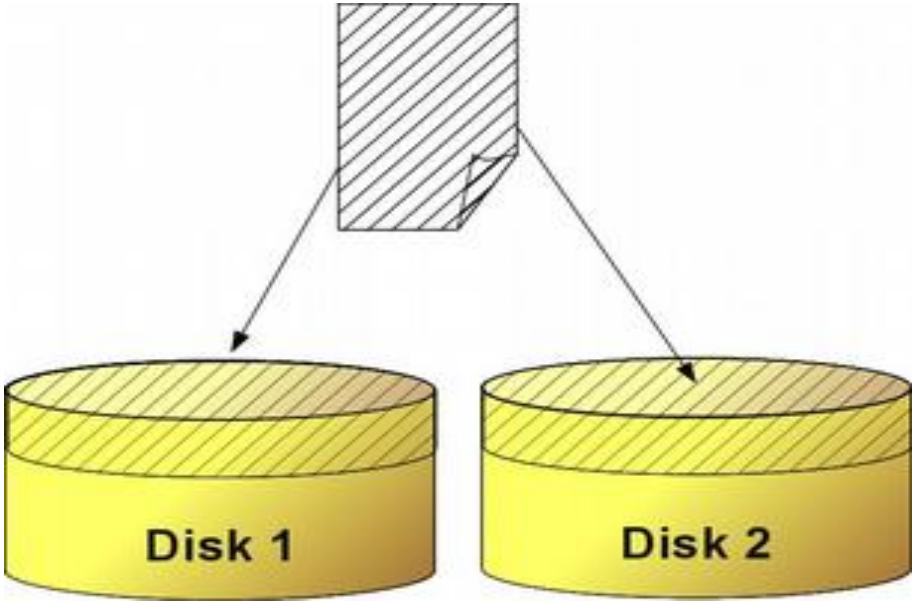


RAID 1 : redundancy

Write Perf. = P

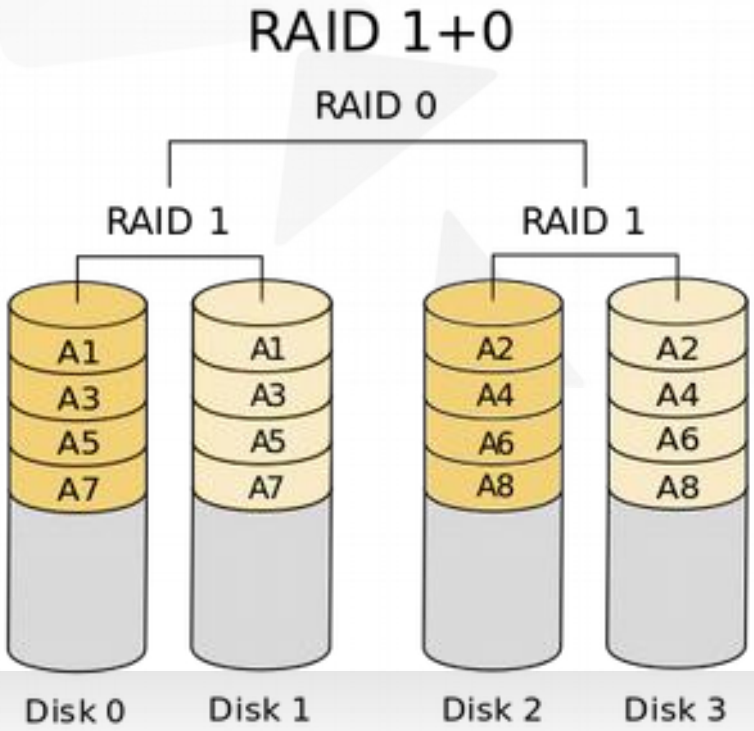
Read Perf. = P * N

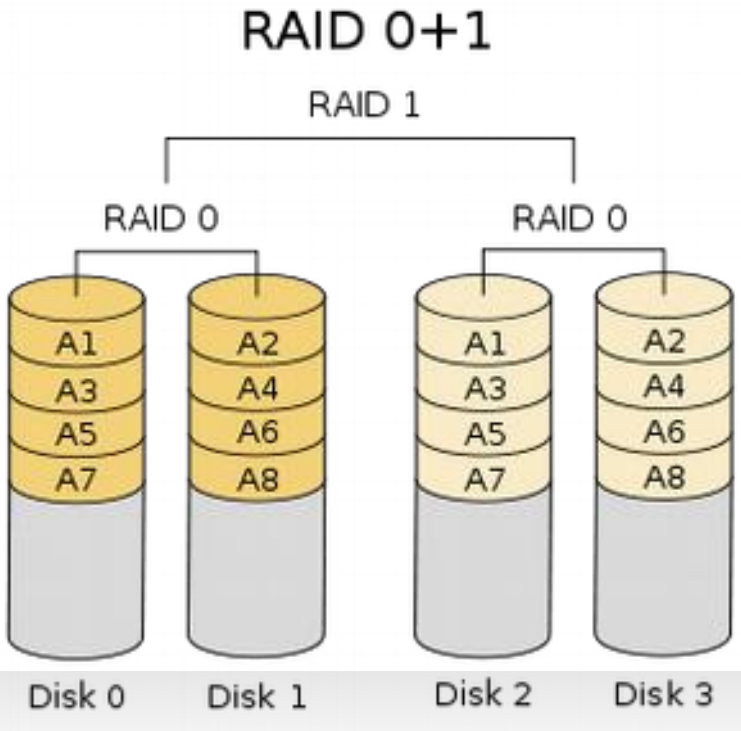
Capacity = S



RAID 10 : striping +redundancy (1+0 / 0+1)

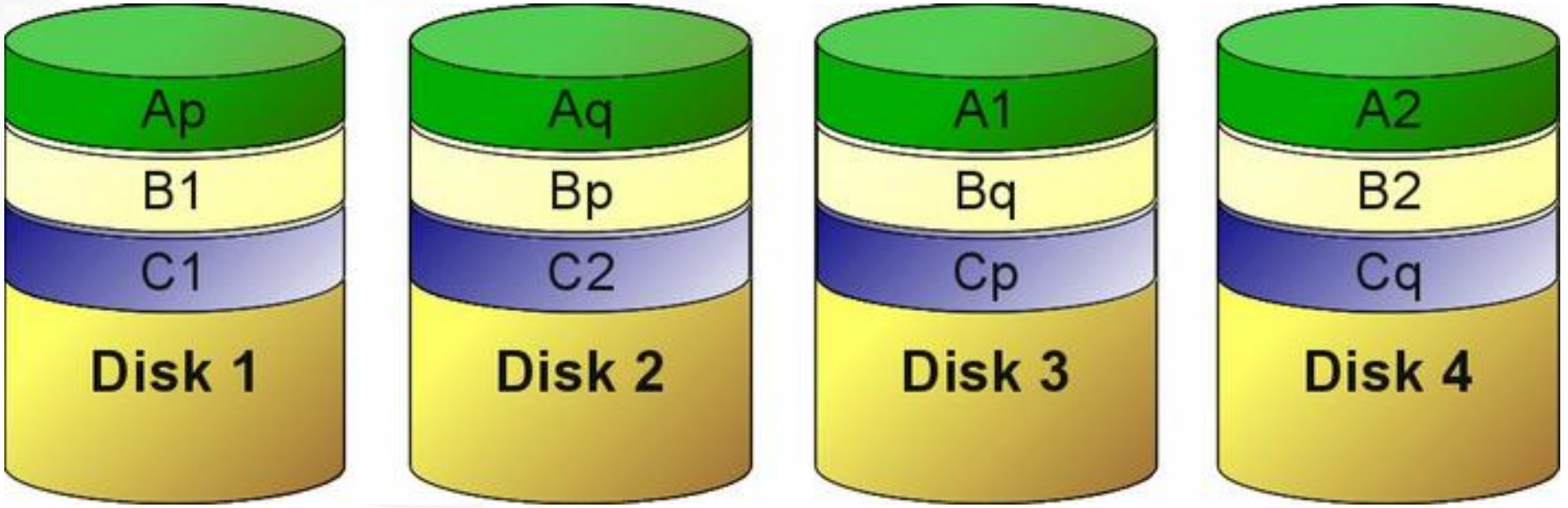
- Raid 1+0 / 10: mirrored sets in a striped set
  - the array can sustain multiple drive losses so long as no mirror loses all its drives
- Raid 0+1: striped sets in a mirrored set
  - if drives fail on both sides of the mirror the data are lost





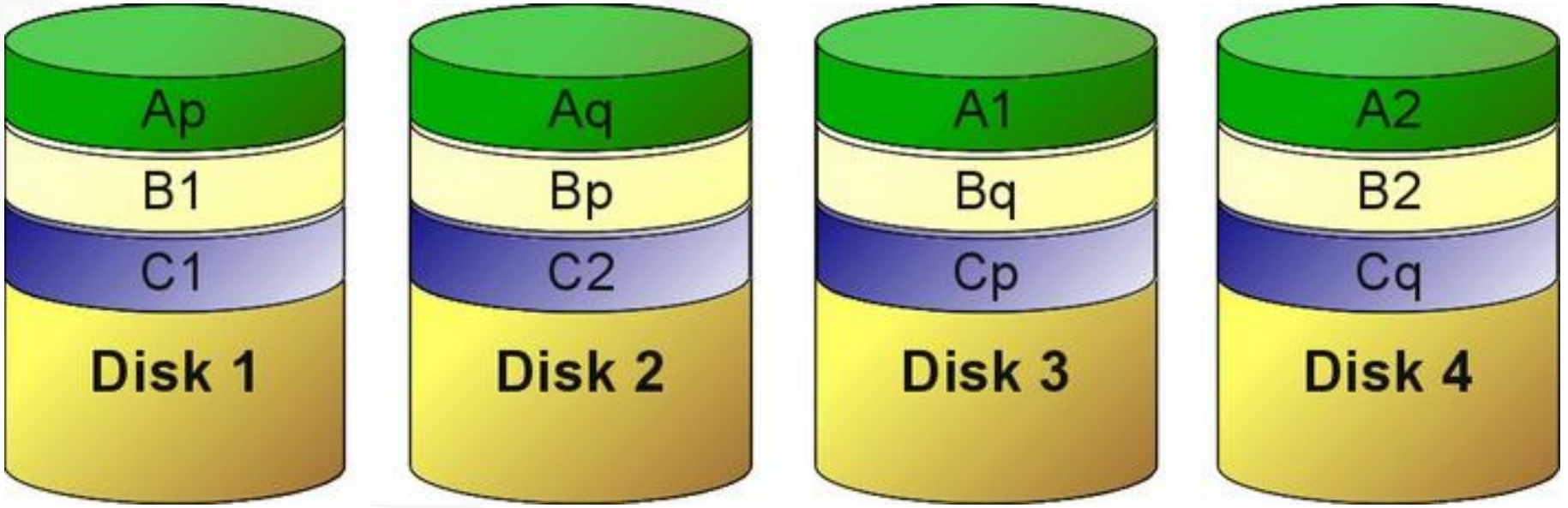
RAID 5

- One disk can fail
- Distributed parity code



RAID 6

- Two disks can fail
- Double distributed parity code



Notes on redundancy

Computing and updating parity negatively impact the performance. Upon drive failure, though, lost data can be reconstructed, and any subsequent read can be calculated from the distributed parity such that the drive failure is masked to the end user.

However, a single drive failure results in reduced performance of the entire array until the failed drive has been replaced and the associated data rebuilt.

The larger the drive, the longer the rebuild takes (up to several hours on busy systems or large disks/arrays).

Hot spare

Both hardware and software RAIDs with redundancy may support the use of a hot spare drive, a drive physically installed in the array which is inactive until an active drive fails, when the system automatically replaces the failed drive with the spare, rebuilding the array with the spare drive included. A hot spare can be shared by multiple RAID sets.

Subsequent additional failure(s) in the same RAID redundancy group before the array is fully rebuilt can cause data loss.

RAID 6 without a spare uses the same number of drives as RAID 5 with a hot spare and protects data against failure of up to two drives, but requires a more advanced RAID controller and may not perform as well.

RAID Parameters

Level	Description	Minimum # of drives	Space Efficiency	Fault Tolerance	Read Benefit	Write Benefit
RAID 0	Block-level striping without parity or mirroring.	2	1	0 (none)	nX	nX
RAID 1	Mirroring without parity or striping.	2	1/n	n-1 drives	nX	1X
RAID 5	Block-level striping with distributed parity.	3	1-1/n	1 drive	(n-1)X	(n-1)X
RAID 6	Block-level striping with double distributed parity.	4	1-2/n	2 drives	(n-2)X	(n-2)X
RAID 1+0/10	Striped set of mirrored sets.	4	*	needs 1 drive on each mirror set	*	*
RAID 0+1	Mirrored set of striped sets.	4	*	needs 1 working striped set	*	*

http://en.wikipedia.org/wiki/RAID

Basic on file systems

- Local file system structures as a building block
- Requirements and Challenges
- Metadata management

Filesystem

A file system is a set of methods and data structures used to organize, store, retrieve and manage information in a permanent storage medium, such as a hard disk.
Its main purpose is to represent and organize resources storage.

File System: elements

Namespace: is a way to assign names to the items stored and organize them hierarchically.

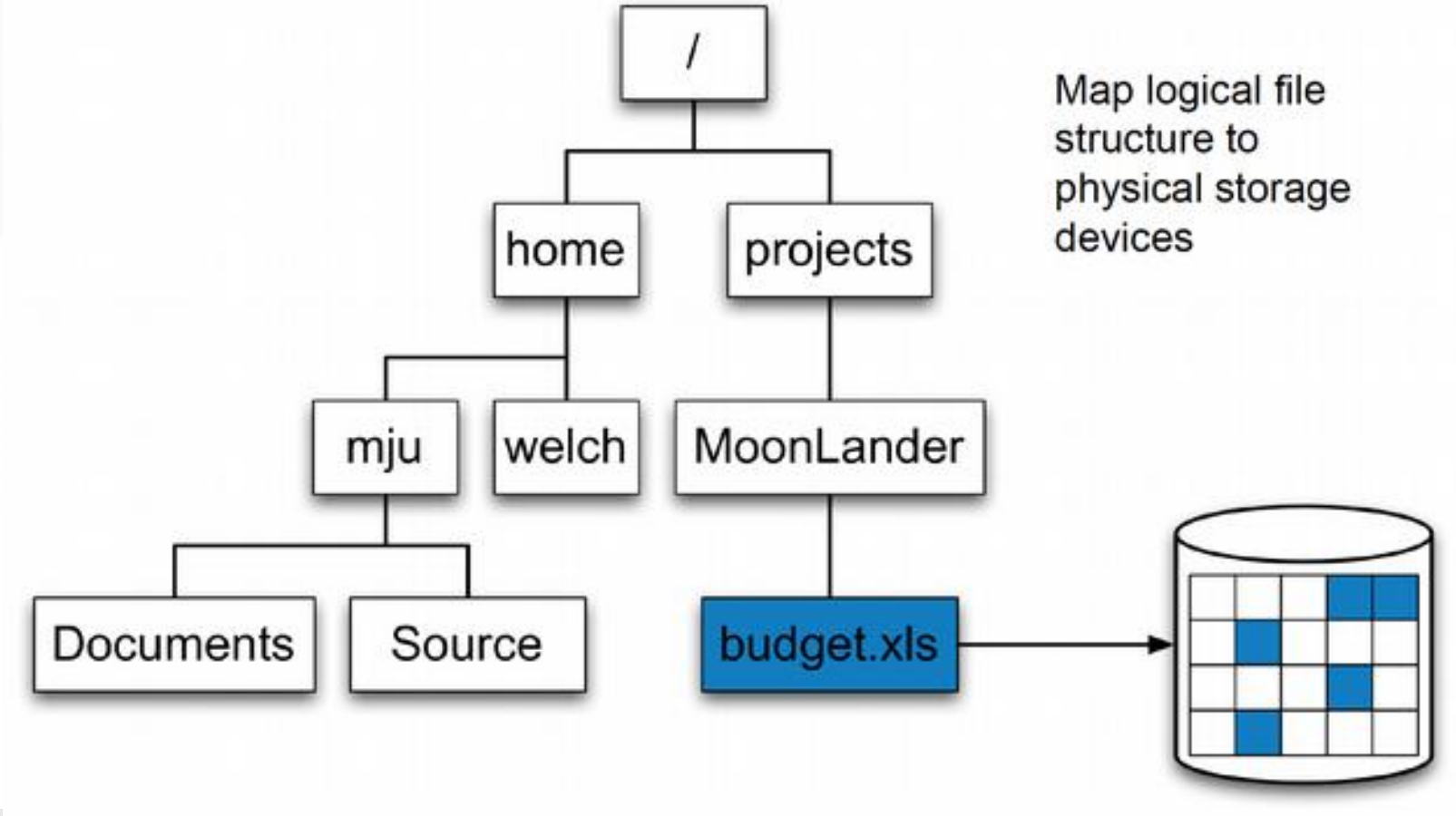
API: is a set of calls that allow the manipulation of stored items

Security Model: is a scheme to protect, hide and share data.

Implementation: is the code that couples the logical model to the storage medium.

Role of the filesytem: (1)

- Provide an unique namespace
- Store your data on the medium (disk/array of disks etc)



File Systems: Basic Concepts

Disk: A permanent storage medium of a certain size.

Block: The smallest unit writable by a disk or file system. Everything a file system does is composed of operations done on blocks.

Partition: A subset of all the blocks on a disk.

Volume: The term is used to refer to a disk or partition that has been initialized with a file system.

Superblock: The area of a volume where a file system stores its critical data.

File Systems: Basic Concepts (2/2)

Metadata: A general term referring to information that is about something but not directly part of it.

Journaling: A method of insuring the correctness of file system metadata even in the presence of power failures or unexpected reboots (atomic write).

Attribute: A name and value associated with the name. The value may have a defined type (string, integer, etc.).

File System for O.S.

- Linux
  - EXT family: ext4/ext3/ext2
  - XFS/ RaiserFS
  - …
- Mac
  - HFS+ (MacOs extended)
  - Apple File System (APFS)
- Windows
  - NTFS

Data and metadata

Metadata : Data to describe data attribute (and extended attribute)

- size, owner, creation date..etc..

Implementation of metadata data structures and access algorithms affects performance, as much as the physical devices and data layout

Meta-data are the bottleneck of scalability

- How many times do you type ls in a day?
  How many times to you write a file?
- ls is scanning all the files in the directory !

File system : data layout and inode

- Data structure pointed by the inode number, a unique identifier of a file in the file system
  - address of data block on the storage media
  - description of the file (POSIX)
    - Size of the file
    - Storage device ID
    - User ID of the file's owner.
    - Group ID of the file.
    - File type
    - File access right
    - Inode last modification time (ctime)
    - File content last modification time (mtime),
    - Last access time (atime).
    - Count of hard links pointed to the inode.
    - Pointers to the disk blocks that store the file's contents

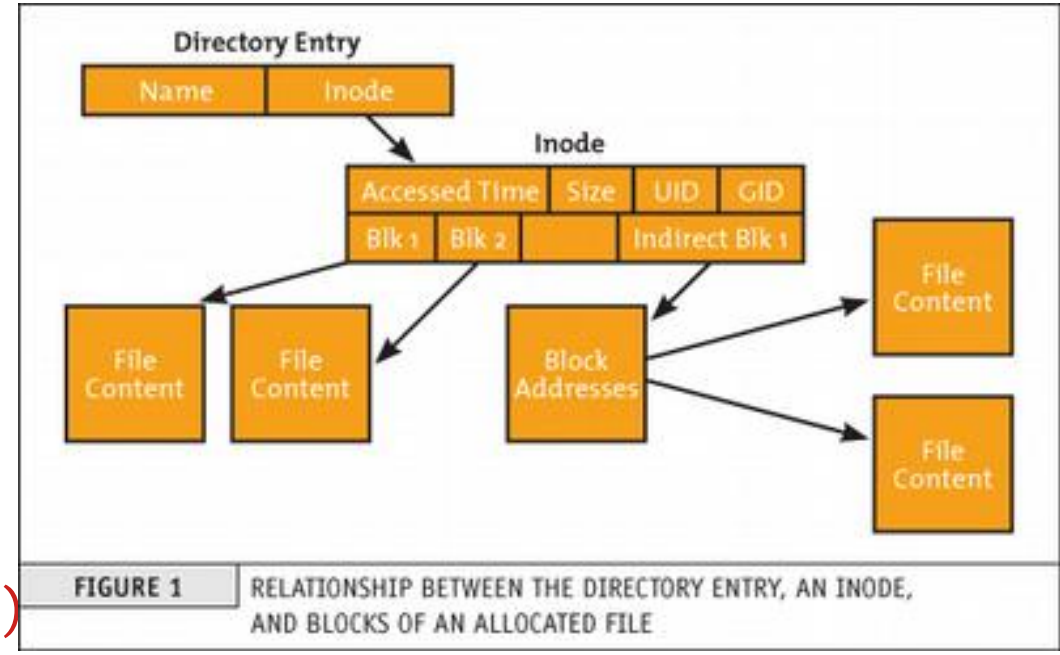


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

FS useful command to check inodes

- ls -i
- Df -i
- stat filename

```
[exact@login ~]$ stat hello.c
  File: ‘hello.c’
  Size: 219         Blocks: 8          IO Block: 4096   regular file
Device: fd07h/64775d   Inode: 1572988     Links: 1
Access: (0664/-rw-rw-r--)  Uid: (  669/   exact)   Gid: (  999/ exact)
Access: 2019-12-01 10:25:29.521223433 +0100
Modify: 2019-11-26 15:03:26.740781336 +0100
Change: 2019-11-26 15:03:26.740781336 +0100
 Birth: -
```

Posix interface

- API to access data and metadata (1988)
- POSIX interface is a useful, ubiquitous interface for building basic I/O tools.
- Standard I/O interface across many platforms.
  - open, read/write, close functions in C/C++/Fortran
- It allows buffered file I/O (streams) within (c/sdtio)

Buffered I/O

- Posix allows buffered I/O (libc/stdio)
- typically applications use this approach
- In typical workloads certain data is accessed repeatedly **beyond** an application lifetimes:
  - OS maintains buffer of recently used data
  - buffer competes with applications for RAM
  - OS can substitute swap disk for RAM
- Memory management unit (MMU) organizes address space in pages

Posix API (1)

Retrieve the list of all files in a directory: 1 call

`readdir`

Retrieve the attribute of the all files from this directory:

`fstat` 1 call per file !

Questions:

What happens if the content of a directory is modified by another process?

Scaling the Filesystem...

- Original POSIX environment was unshared, direct-attached storage
- RAID and Volume Managers aggregate devices safely
  - Scale performance within the same machine ！

Posix API (2)

Posix assumes atomicity and ubiquity

Changes are visible immediately to all clients

Problem for parallel accesses:

- POSIX requires a strict consistency to sequential order : lock

  (Create a directory is an atomic operation with immediate global view)
- No support for non-contiguous I/O !!

MPI-IO can be useful here..

Measure (raw) performance

dd: to measure storage system performance

```
$dd if=/dev/zero  of=/dev/null count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.000242478 s, 2.1 MB/s
$dd if=/dev/zero  of=~/big-write count=1M
1048576+0 records in
1048576+0 records out
536870912 bytes (537 MB) copied, 3.43889 s, 156 MB/s
```

Questions:

1- Why such a difference between the two runs?

2 Why copying unit of 512B ?

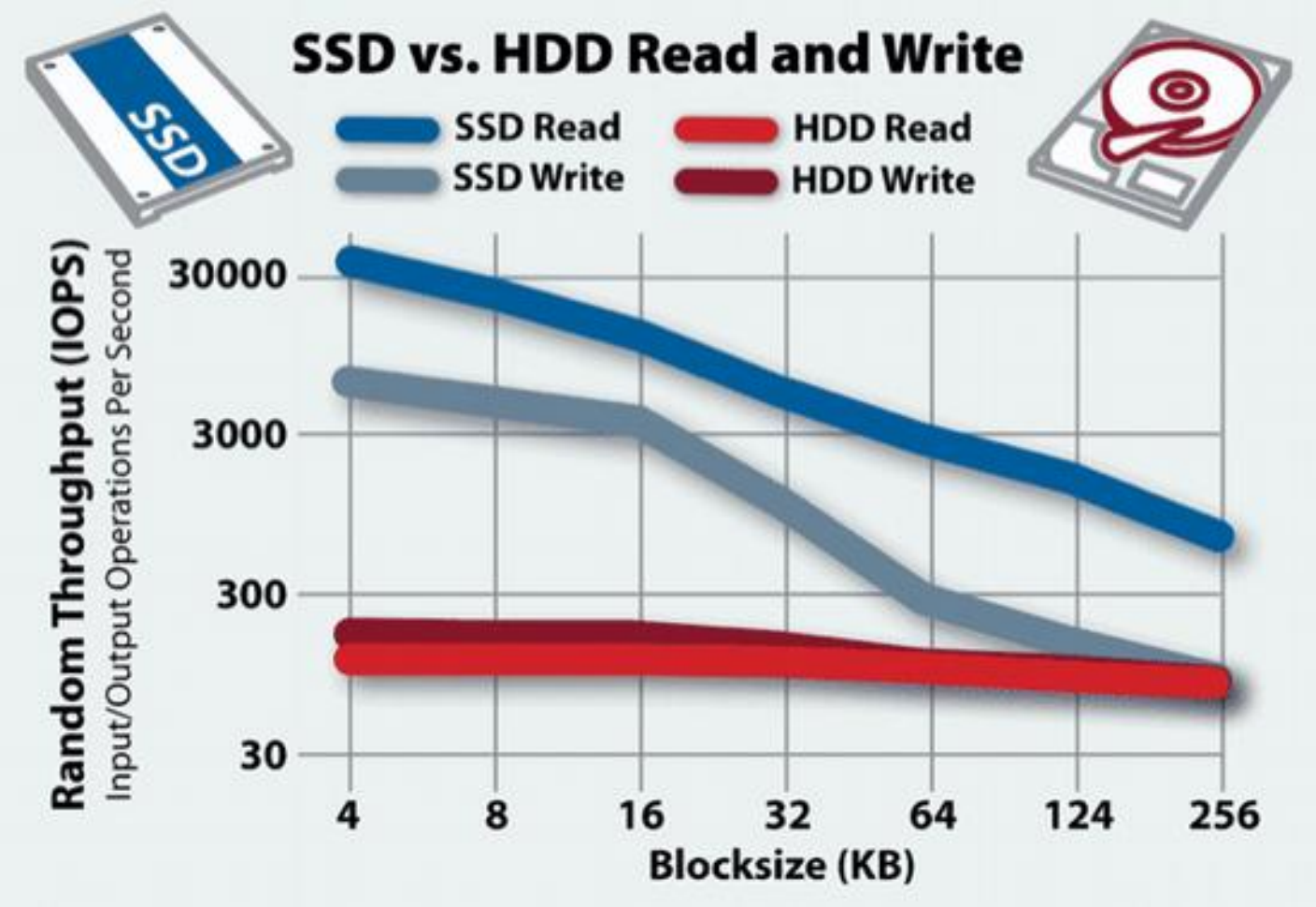
Blocksize on FS

- 512 byte is a typical block size of the disk:
  - IT cannot read less than 512 bytes, if you want to read less, read 512 bytes and discard the rest.
- File System blocksize can be different :

```
[exact@login ~]$ stat -f .
  File: "."
    ID: 9d0420af3cbc070e Namelen: 255     Type: ext2/ext3
Block size: 4096       Fundamental block size: 4096
Blocks: Total: 372561982  Free: 51012529   Available: 32646449
Inodes: Total: 94633984   Free: 90641935
```

Blocksize effect in the Random access

- The performance DISK is not a single number



Proposed exercises

- Identify your FileSystem and its properties
- Measure/Estimate the rough performance of your hard-drive
- Compare it with the ramfs on your linux box and on your cluster system

```
exact@login ~]$ df
Filesystem                   1K-blocks        Used  Available Use% Mounted
on
/dev/mapper/SysVG-Root        51474912    33126208   15710880  68% /
devtmpfs                      16358128           0   16358128   0% /dev
tmpfs                         16371480      501024   15870456   4% /dev/shm
```