

11/15/23

SSD

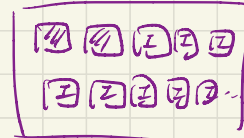
- > page (4K): read & write (program), takes tens of microseconds
- > block (1-8MB): erase an entire block, takes several milliseconds
- > wear leveling:
 - pages can only be erased and written so many times
 - spread writes around so pages wear out around the same time
 - need a way to remap pages => introduce level of abstraction

Logical Block Address (Block, page)

↓ Flash Translation Layer (implemented by SSD controller)

Physical Block Address (Block, page)

updates to LBA can be done on different pages, just update the mapping each time! old pages are now invalid & free to be recycled during Garbage Collection



Block

☑ - programmed valid page

I - invalid page

Performance

total time = access latency + transfer time + (erasure time)

read 10 pages (4KB), read latency (10us per page), transfer rate is 500 MiB/s

assume
serial request

$$\underbrace{(10 + 7.8)}_{1 \text{ request}} \times 10 = 178 \text{ us} = 0.178 \text{ ms.}$$

7.8 us per 4096 bytes


$$\text{IOPS. } \frac{10 \text{ ops}}{0.178 \text{ ms}} \times 1000 \text{ (ms} \rightarrow \text{s)} = 56 \text{ KIOPS}$$

980 PRO PCIe® 4.0 NVMe™ SSD 1TB

Total \$109.99 ~~\$159.99~~*

[BENEFITS](#) [SPECS](#) [REVIEWS](#) [SUPPORT](#) [RELATED](#)

[Chat about Black Friday Deals](#) [♥](#)

 Save an additional 5% with 3 pack!



Sequential Reads
up to **7,000 MB/s**

Sequential Writes
up to **5,100 MB/s**

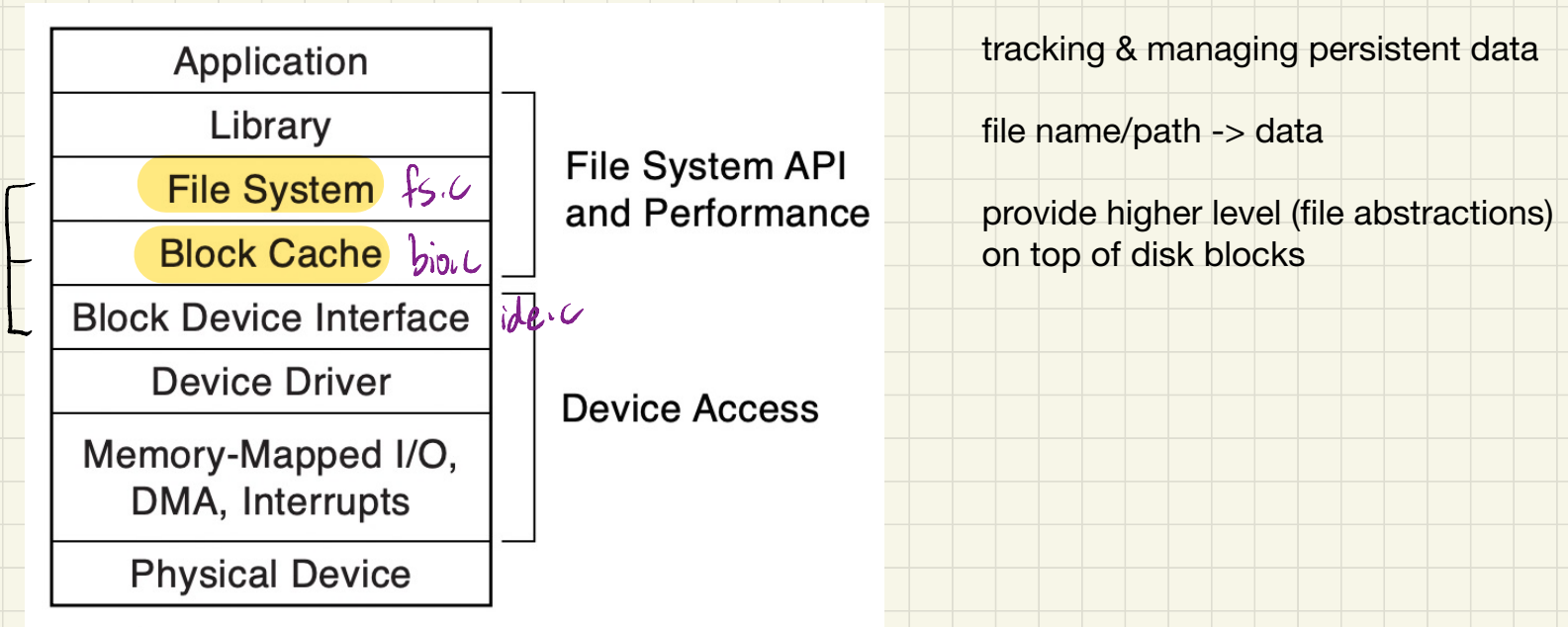


Random Reads
up to **1,000K IOPS**

Random Writes
up to **1,000K IOPS**

File System

software stack



Block Cache: cache disk blocks in memory for faster access, writes to a block might be cached, content of cached block may differ from the actual on block on disk

Filesystem Abstraction

-> operations: read, write, create, delete, link...

-> file abstraction

-> container of data

-> data: file content

-> metadata: size, permission, owner, access/modify time, *where data is located on disk*

-> directory abstraction

-> a way to group and organize files

-> how to implement the directory abstraction?

-> can we implement directory as file?

-> metadata (same fields as file)

-> data? files within the directory

* metadata for file also known as inode, file header...

Data Layout

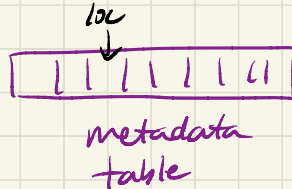
Format directory data as an array of directory entries (dirent, dentry)

"."	loc
".."	loc
"a.txt"	loc
"b.txt"	loc
"c.txt"	loc

name metadata location

→ where the metadata is on disk

loc: could be just disk block #,
or could be index into a metadata table



implement directory as file lets us
reuse existing abstraction,
and makes it easy to support
nested directories

Path: **"/a/b/c/d/data.txt"**

first "/"
refers to
root directory

"/" separates directory

absolute path: Starts w/ "/" root dir

relative path: "a/b/c", starts at current
working directory, chdir system
call to change wwd (cd)

read = "home / tom / foo.txt"

① read in "/"
metadata for root dir

File 2

bin	737
usr	924
home	158

② from root's metadata,
read root's data block

→ File 158

"/home"

③ from root's
data, locate &
read in home's
metadata

mike	682
ada	818
tom	830

④ from home's metadata,
read in home's data block

→ File 830

"/home/tom"

⑤ from home's
data, locate & read
in tom's metadata

music	320
work	219
foo.txt	871

⑥ from tom's metadata
read in tom's data

→ File 871

"/home/tom/foo.txt"

⑦ from tom's data,
locate & read in
foo.txt's metadata

The quick
brown fox
jumped
over the
lazy dog.

⑧ from foo.txt's
metadata, read in
foo.txt's data!