11/14 Storage : SSD & Filesys Basics

Common memory leak from Lab 2.

```
exec() {
  struct vspace new;
  // initialize & set up new vspace;
1 vspaceinit(&p->vspace);
2 vspacecopy(&p->vspace, &new);
3 vspaceinstall(p);
4 vspacefree(&new);
}
```

sysinfo

→ what happens to your current page table & data?

this makes a deep copy of the new address vspace, is that what we want?

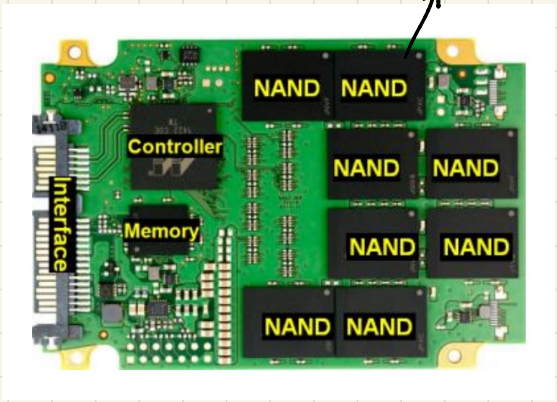
```
struct vregion {
  enum vr_direction dir;  // direction of growth
  uint64_t va_base;       // base of the region
  uint64_t size;          // size of region in bytes
  struct vpi_page *pages;  // pointer to array of page_infos
};

struct vspace {
  struct vregion regions[NREGIONS]; // the regions for a process' virtual space
  pml4e_t* pgtbl;                   // process' page table
};
```

→ vspace just stores pointers to things

swap from the old vspace to new one

```
exec() {
    struct vspace new;
    // initialize & set up new vspace;
    struct vspace old = p->vspace;   // saves old vspace struct, has
                                     // references to old page table &
                                     // bookkeeping data
    p->vspace = new;
    vspaceinstall(p); // switch to new
    vspacefree(&old); // safe to free old vspace.
}
```

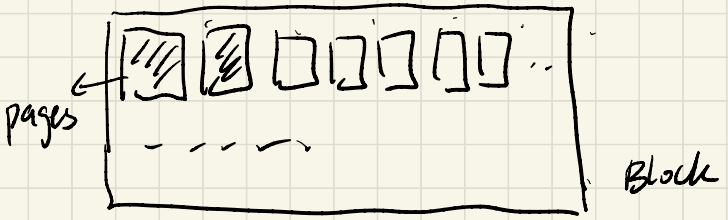
Solid State Drive



disk: sectors (512 byte)

SSD: page (2k-4k) read/write unit

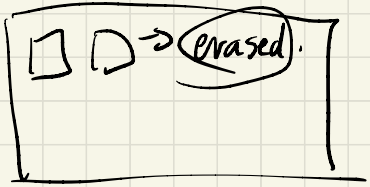
Erasure Block (1MB-8MB), erase unit



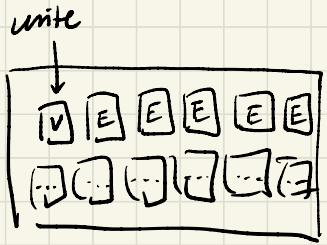
Erase: erase all pages in the block (set all bits to 1) single level cell

write: write 0s (1010010)

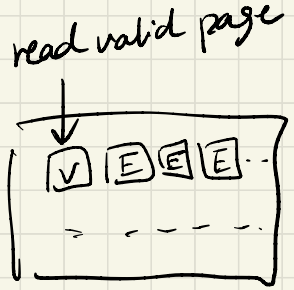

Initially (Invalid)



Erase (block) (Erased)



Write (page) (page written = Valid, other pages = Erased)



read (page)

Performance

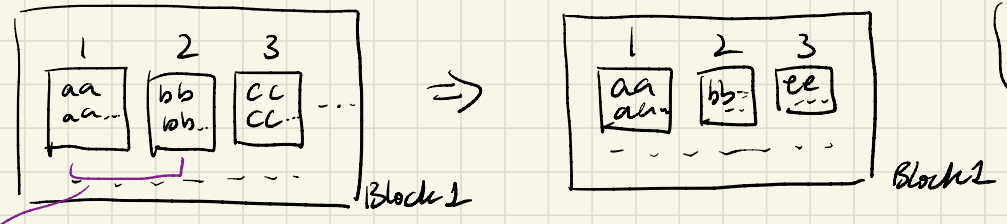
→ read (page level) : good (~10us) for segmental and random read (4KB)

→ write (page level) / program : write to an erased page (~100us)

→ erase (block level) : takes a few ms

☆ can only program/erase so many times before the block becomes unusable → wear leveling - spread writes across blocks & pages

On erase, what happens to existing data in the block?



to overwrite page 3 we need to erase block 1

→ copy existing data to erased pages elsewhere, erase block and then write to block 1 pages

not the most efficient approach, had to move data around multiple times

Flash Translation Layer

→ logical to physical block # (page #, block not exposed to client)

↓ what we give to client (kernel)

Block 1 → ~~Block 1~~

↘ Block 2.

Performance Metrics

→ I/O Operations per second (IOPS)

→ I/O request: size (# of bytes to read write)
access patterns matter! (workload)

→ simple average IOPS

$$\frac{1}{\text{average request latency}}$$

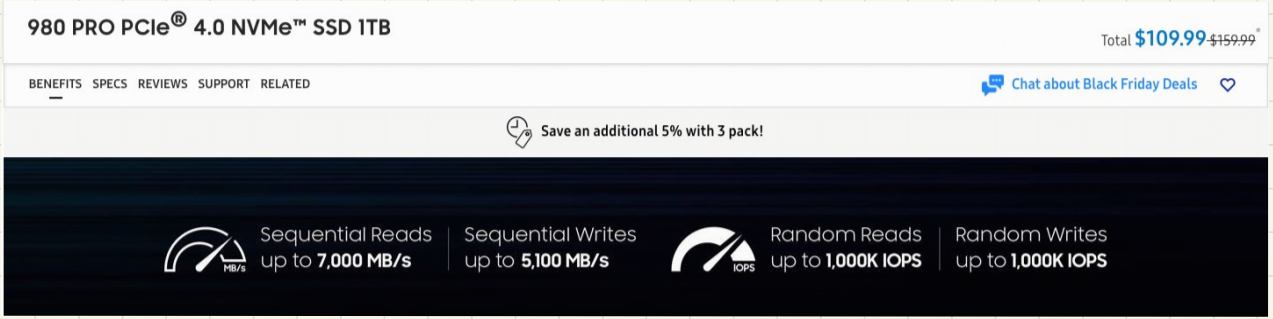
for disk: $\frac{1}{14\,ms} \approx 70$ IOPS

→ actual IOPS

$$\frac{\#\text{ of I/O requests}}{\text{total time to complete the requests}}$$

for 10 sequential reads on disk

$$\frac{10}{14\,ms} \approx 700 \text{ IOPS}$$

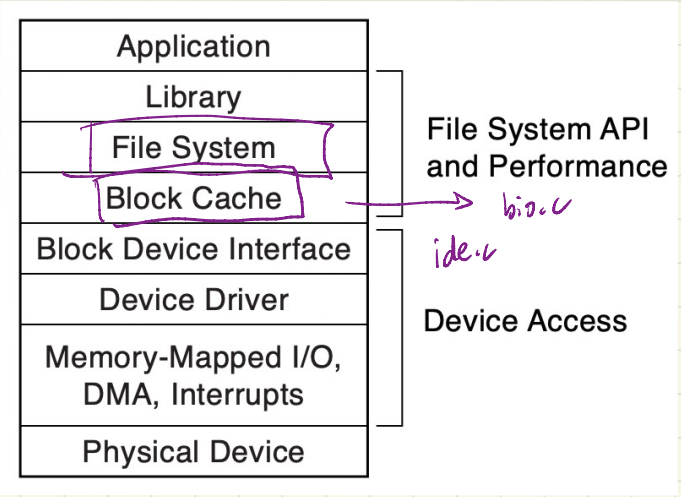


Filesystems

-> Abstraction for storage devices

-> names & data (files, directories)

-> manages storage devices & offers richer APIs



Data vs Metadata

↓ file data

↓ info about file (size, owner, permission ...)