Solid State Drive

→ persistent, block addressable, large capacity

NAND Flash packages connected by multiple channels, highly parallel architecture

Units: blocks, pages

a block contains hundreds of pages, page is the unit of read/write (4KB)

no moving parts
SSD Operations

- **Read a page (4KB)**: fast access  \(~ 10\)\(\mu\)s

- **Write a page (4KB)**: can only write to a clean page (program) program bits to Os to write data  \(~ 100\)\(\mu\)s

- **Erase a block (1-8MB)**: erase all pages within the block (all 1s) slow operation  \(~ 1-3\)\(\) ms

To do in-place update on SSD

- Want to reprogram this page
- Save valid programmed page(s) elsewhere (some clean pages)
- There's a valid bit associated w/ each page, easy way to mark garbage

```
+------------------+
| clean page       |
| programmed page  |
+------------------+
```

```
+------------------+
|                  |
| erase            |
|                  |
+------------------+
```

```
+------------------+
|                  |
| program w/ new data |
|                  |
+------------------+
```

```
+------------------+
|                  |
| restore valid pages w/ saved data |
|                  |
+------------------+
```

```
+------------------+
|                  |
| (metadata)       |
|                  |
+------------------+
```
SSD Reliability

- A page can only reliably endure 10-100k writes.
- Repeated writes to pages cause frequently modified pages to wear out faster (no longer retain data reliably).

- Wear-leveling: spread writes to different pages to wear out pages evenly → each write moves the data to a new page.

& Flash Translation Layer
- Translate logical block address
- Physical block address
- Garbage collection
  - Move sparsely valid pages into a new block, frees up a block for erase.
SSD Request Latency

\[
\text{total time} = \text{access latency} + \text{transfer time} + \text{(erasure time)}
\]

- latency of a read page request given 10 ms read latency & 500 MiB/s
- way less sensitive to access patterns
- much closer performance for sequential & random accesses

17.8 ms
7.8 ms

<table>
<thead>
<tr>
<th>980 PRO PCIe® 4.0 NVMe™ SSD 1TB</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total</strong> $109.99</td>
</tr>
</tbody>
</table>

- Save an additional 5% with 3 pack!
**Filesys Basics**

- **Application**
- **Library**
- **File System**
  - Block Cache
- **Block Device Interface**
- **Device Driver**
- **Memory-Mapped I/O, DMA, Interrupts**
- **Physical Device**

**File System API and Performance**

- User code (e.g. `xk test programs`)
- User libraries (e.g. `libc: stat() = open, fstat, close`)
- Create files/directories abstraction on top of disk blocks

**Device Access**

```c
struct {  
    struct spinlock lock;
    struct buf buf[NBUF];
};
struct buf {  
    int flags;
    uint dev;
    uint blkno;
    struct sleeplock lock;
    uint refcnt;
    struct buf *prev; // LRU cache list
    struct buf *next;
    struct buf *qnext; // disk queue
    uchar data[BSIZE];
};
#define B_VALID 0x2 // buffer has been read from disk
#define B_DIRTY 0x4 // buffer needs to be written to disk
```