CSE 451: Operating Systems

Lab Section: Week 6

Today

- Project 3 retrospective
- I/O scheduling: disks, flash

(I have no idea what's on tomorrow's quiz ☹)

Project 4

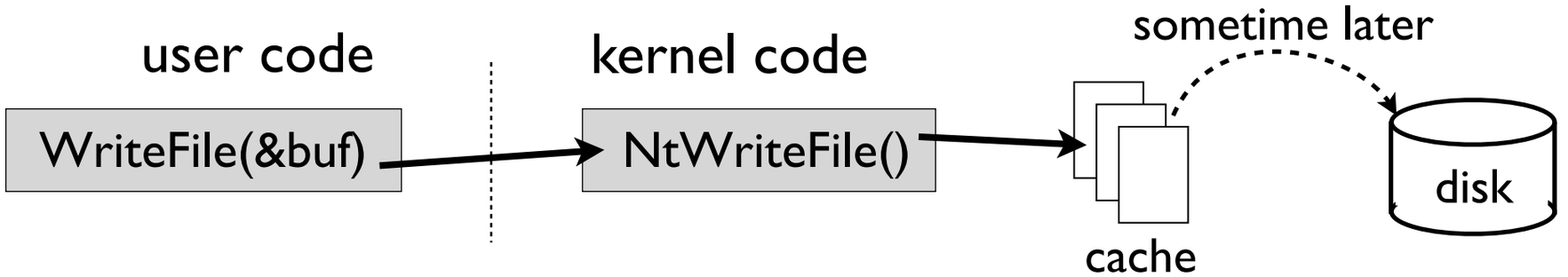
- Posted!
- Due Wednesday, March 9 at 11:59pm
  - in three weeks
- Goal: modify the FAT filesystem
  - kernel hacking! ☺
  - more next week

Project 3

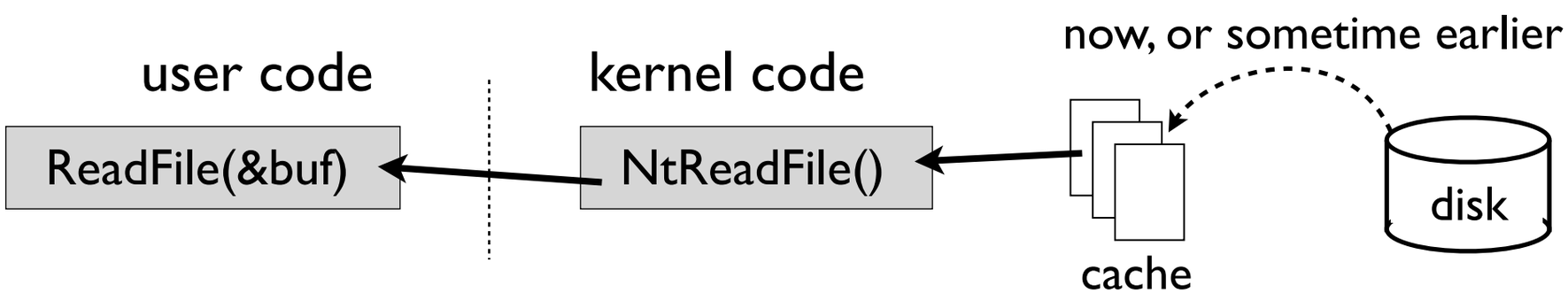
- What did your performance analysis look like?

Disk Caching

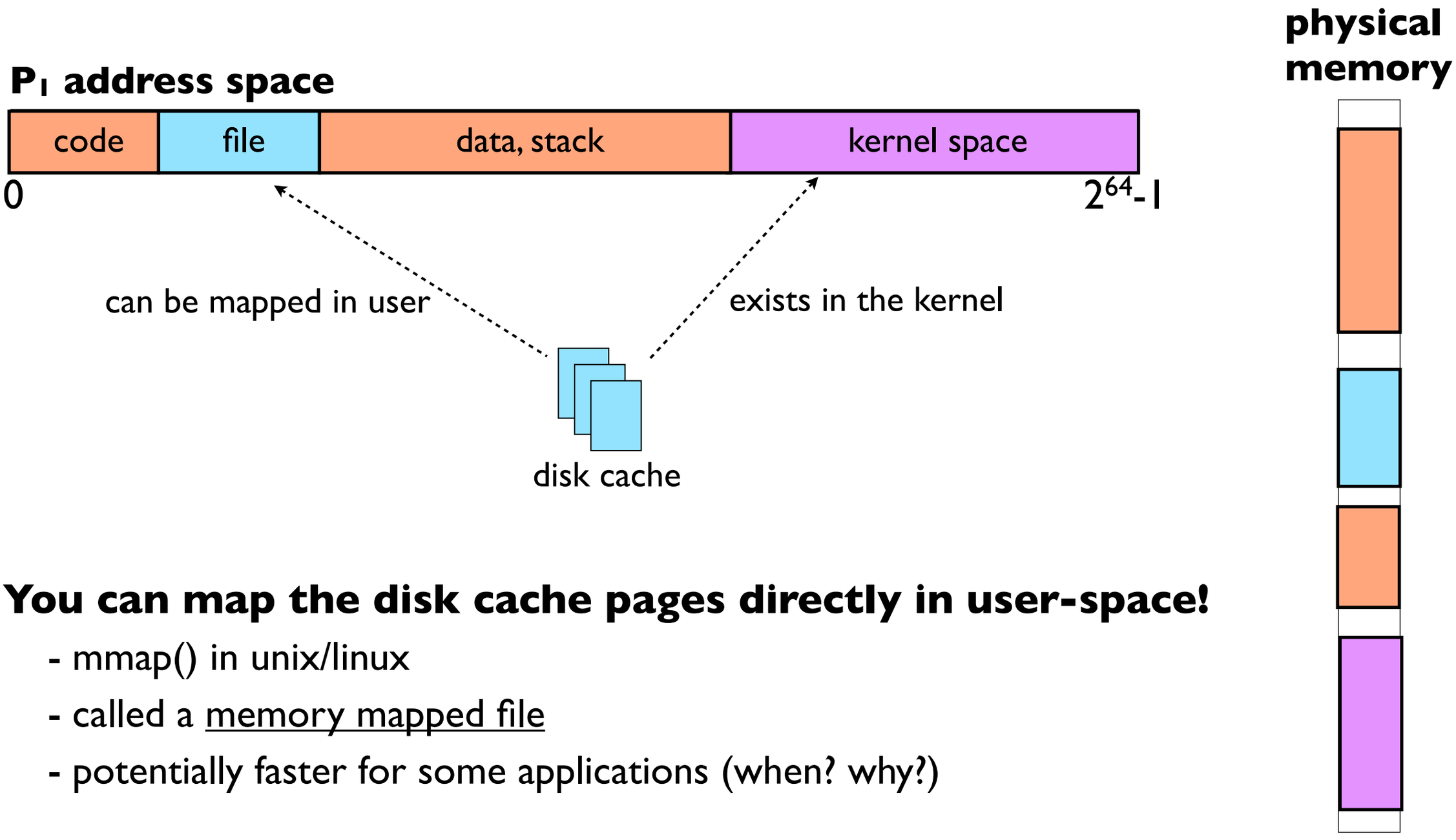
- This is what makes disk scheduling optimizations possible
- Write caching



- Read caching



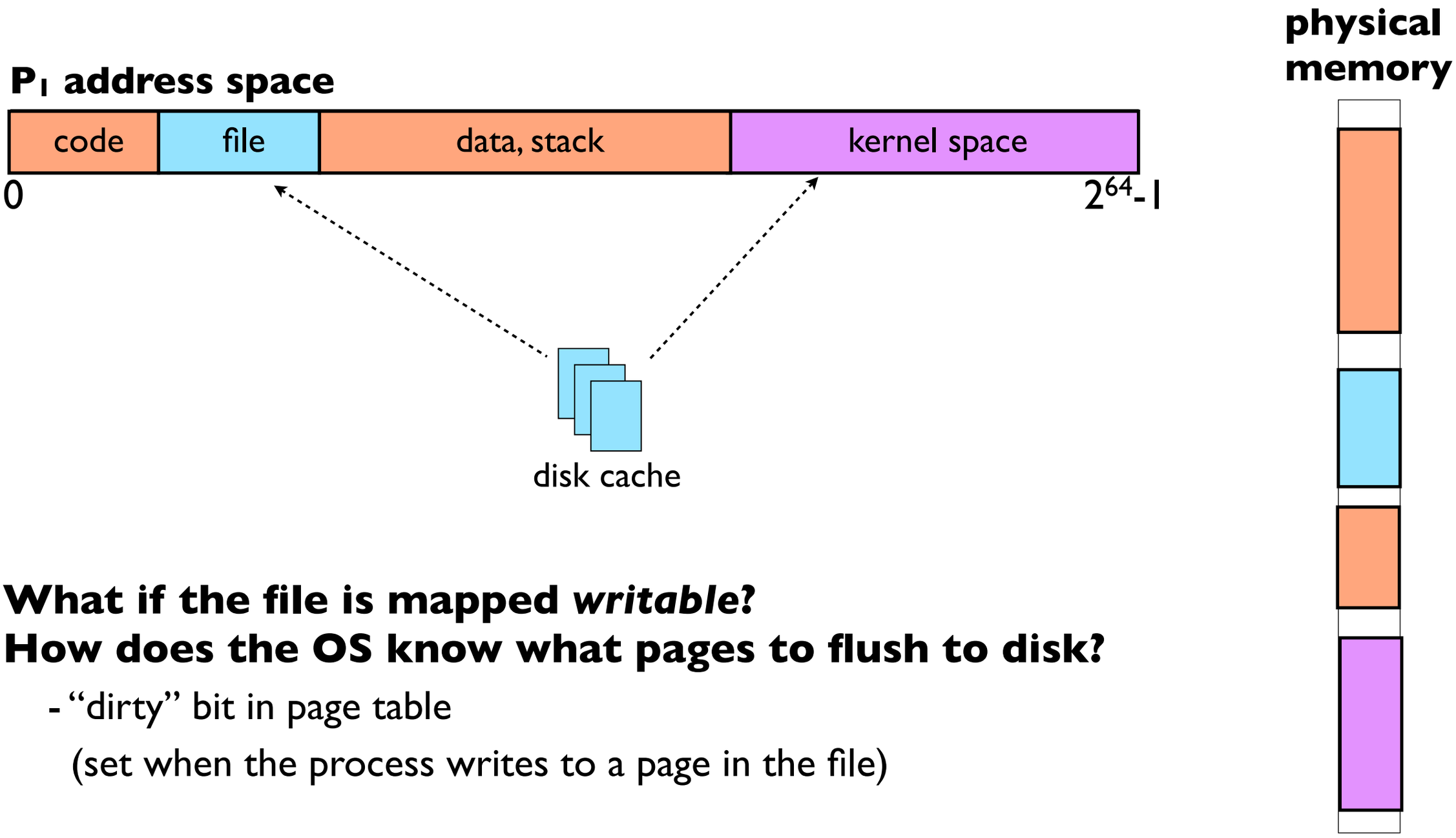
Memory Mapped I/O



You can map the disk cache pages directly in user-space!

- mmap() in unix/linux
- called a memory mapped file
- potentially faster for some applications (when? why?)

Memory Mapped I/O



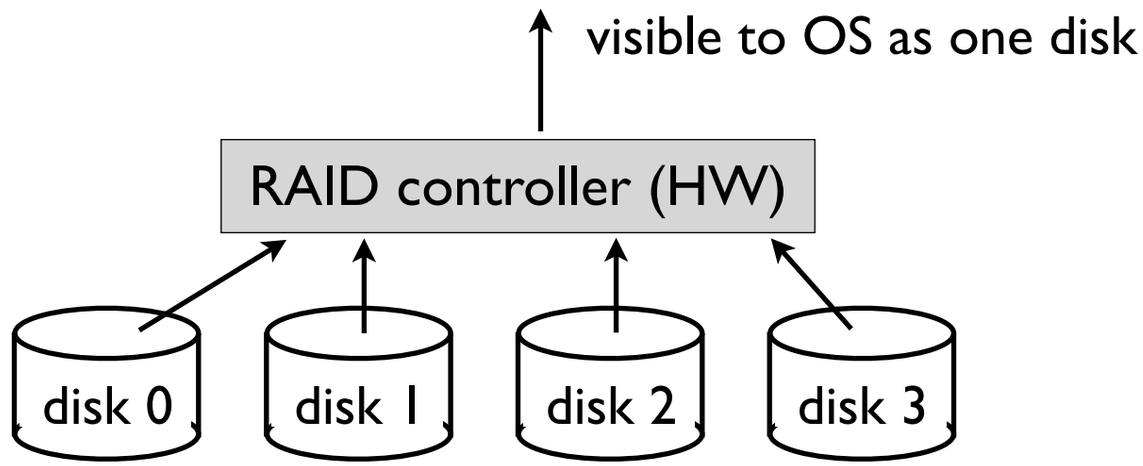
What if the file is mapped *writable*?
How does the OS know what pages to flush to disk?

- "dirty" bit in page table

  (set when the process writes to a page in the file)

RAID Level 0

- "Stripes" data across disks
  - makes one disk look like many



- Advantages of putting RAID controller in HW?
  - makes OS job easier
- Disadvantages?
  - kernel can't optimize!

Solid State Disks (flash)

- This is what's on a usb stick
- Similar to disks
  - data is stored in blocks
- Similar to memory
  - no spinning platters ... random access is fast!
  - all those disk scheduling algorithms are unnecessary
  - much faster than disks (for reads...)

- Solid state drives are based on NAND flash memory
  - no moving parts; performance characteristics driven by electronics and physics – more like RAM than spinning disk
  - relative technological newcomer, so costs are still quite high in comparison to hard drives, but dropping fast

SSD performance: reads

- Reads
  - unit of read is a *page*, typically 4KB large
  - today's SSD can typically handle 10,000 – 100,000 reads/s
    - 0.01 – 0.1 ms read latency (50-1000x better than disk seeks)
    - 40-400 MB/s read throughput (1-3x better than disk seq. thpt)

SSD performance: writes

- Writes
  - flash media must be *erased* before it can be written to
  - unit of erase is a block, typically 64-256 pages long
    - usually takes 1-2ms to erase a block
    - blocks can only be erased a certain number of times before they become unusable – typically 10,000 – 1,000,000 times
  - unit of write is a page
    - writing a page can be 2-10x slower than reading a page
- Writing to an SSD is complicated
  - random write to existing block: read block, erase block, write back modified block
    - leads to hard-drive like performance (300 random writes / s)
  - sequential writes to erased blocks: fast!
    - SSD-read like performance (100-200 MB/s)

SSDs: dealing with erases, writes

- Lots of higher-level strategies can help hide the warts of an SSD
  - many of these work by virtualizing pages and blocks on the drive (i.e., exposing logical pages, not physical pages, to the rest of the computer)
  - wear-leveling: when writing, try to spread erases out evenly across physical blocks of of the SSD
    - Intel promises 100GB/day x 5 years for its SSD drives
  - log-structured filesystems: convert random writes within a filesystem to log appends on the SSD (more later)
  - build drives out of arrays of SSDs, add lots of cache