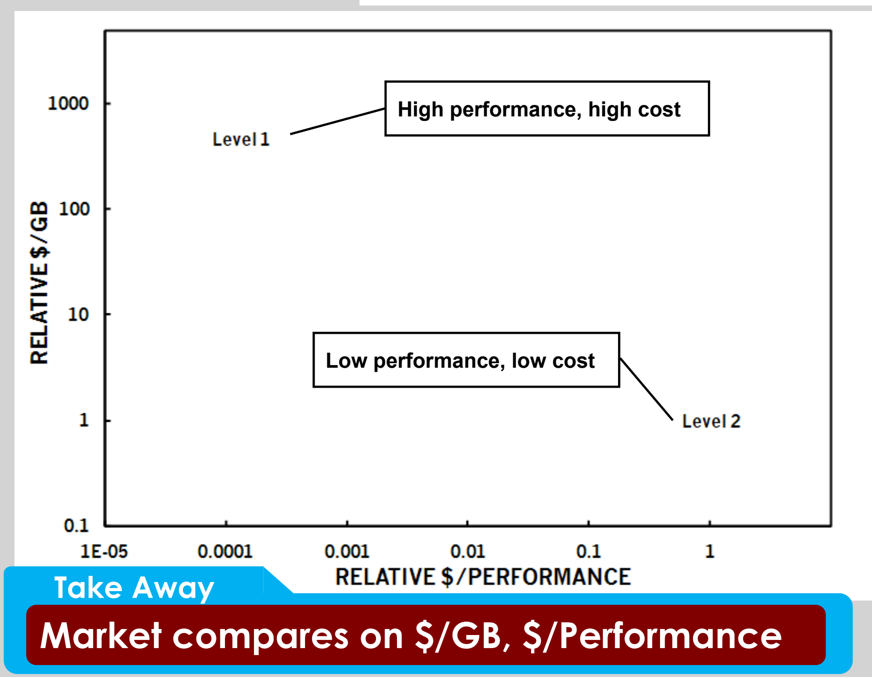
Flash Storage

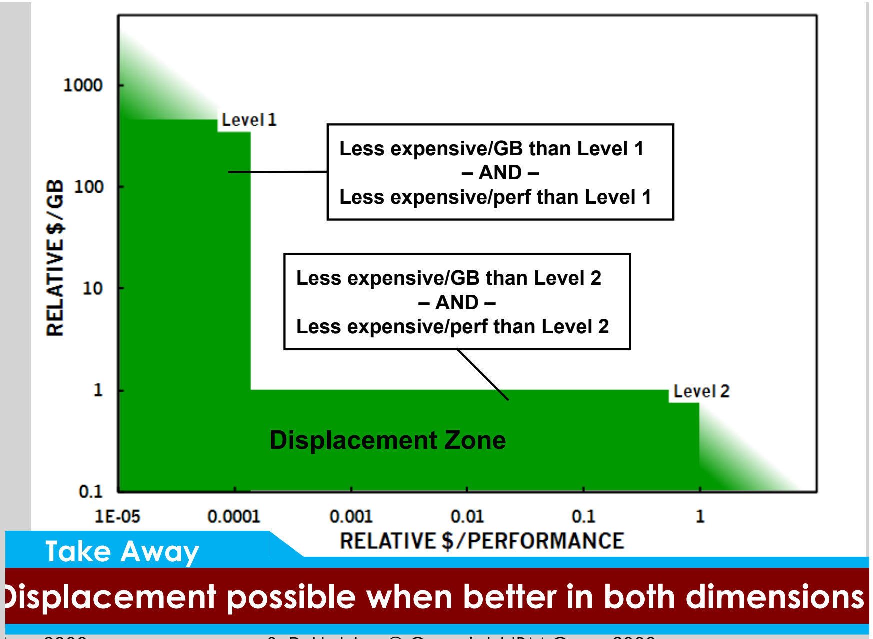
1. From reviews:
    a. SE-Util vs SE-Merge
    b. Why does write back perform better while write through can use silent eviction?
    c. Why different data vs log blocks?
2. Flash device basics
    a. All flash:
        i. Stores charge persistently that can later be sensed
        ii. Can only write changing a 1 to a zero; changing a zero to a 1 requires a coarse-grained and slow erase operation
    b. NOR flash:
        i. Allows random access (can be accessed like ROM)
        ii. Fast – 75ns, close to DRAM speeds for read, slow writes, very slow for erase (1 second)
        iii. Used for storing system code – can execute in place without copying to DRAM first, saving memory
        iv. Lower density than NAND
    c. NAND flash
        i. Page access only as a device; cannot be accessed as memory
        ii. 25 us reads, 1.5 ms erase, 100us writes
    d. All have endurance issues
        i. After erasing too many times, stops working.
3. Flash packaging
    a. Embedded flash
        i. No controller; raw flash accessible to CPU
        ii. All management (garbage colletion, striping of data) must be done by OS
        iii. Used in embedded devices
    b. Removable flash – CompactFlash, SD cards
        i. Simple controller in the package
        ii. Depending on generation:
            1. Simple mapping: writes do a read/modify/write in place
            2. Limited wear leveling
        iii. Used for consumer devices with sequential workloads
    c. SSDs – managed flash
        i. Controller implements translation layer

ii. Controller + software manages parallelism remapping, garbage collection

4. History of storage technologies
    a. Disks remain and will pretty much always be larger and slower
    b. Many candidate technologies- e.g. SD cards, MEMS
        i. All faster than disks, but smaller and more expensive?
    c. Why flash?
        i. Found a use disks couldn't work
            1. low power important; disks are ~5 watts
        ii. Capacity not as important
            1. Cell phones, early phones & mp3 players
            2. How much is a 8 GB hard disk compared to 8GB flash?
                a. answer: more
        iii. Market was big enough to sustain development and make money
        iv. Once Flash was mature, people started building SSDs for PCs/servers
    d. Nothing else has done this yet but disks
    e. Problem with adopting new storage technologies:
        i. Show memory/disk curve – speed vs capacity
        ii. For large capacity, disk always cheapest
        iii. For high performance, memory best
        iv. Hard to live in the middle

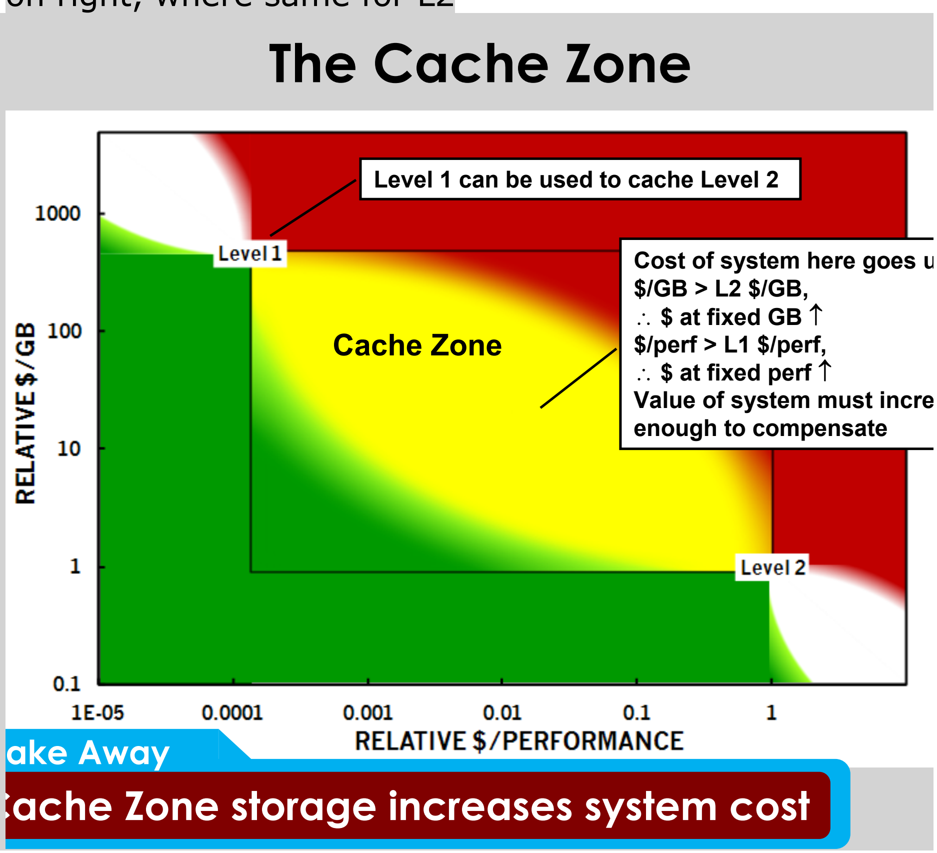


        v.
        vi.



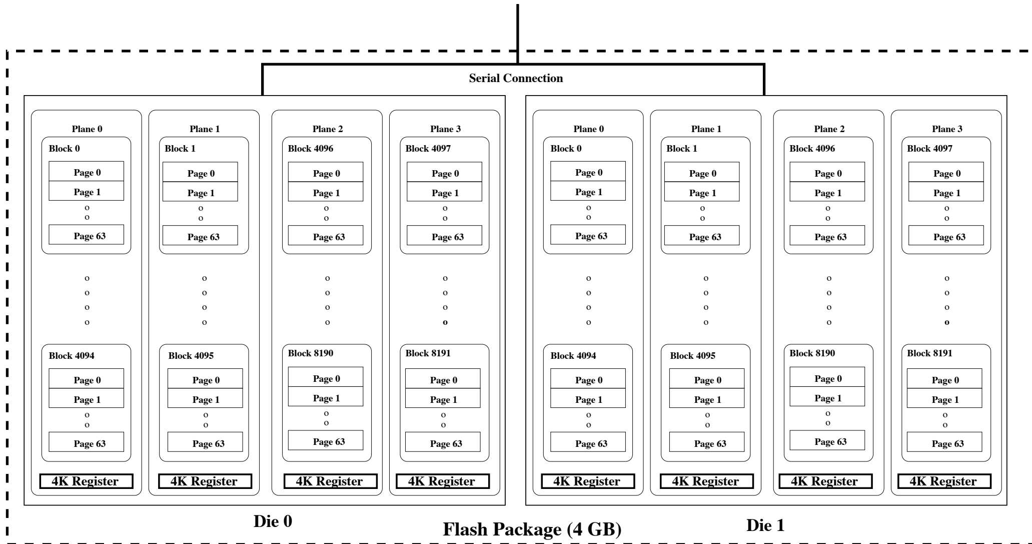
vii.

viii. Dead zone – at top; where more expensive than L1, and on right, where same for L2



ix.

5. SSD Organization
   a. Physical layout

i. 

ii. Multiple dimensions: dies, split into planes, with blocks containing pages

iii. Parallelism across planes, between some dies internall

iv. SLC vs MLC:
   1. SLC is one bit (2 voltage levels) per cell, fast
   2. MLC is 2 bits (4 voltage levels) per cell: higher capacity, lower speed, lower endurance (100,000 erases per SLC, 10K for MLC, 1 K for TLC)

v. Connections:
   1. SATA: limited by protocol, overhead of protocol, bandwidth of protocol
   2. PCIe: implement device driver, can do anything
      a. example: FusionIO
   3. NVMe: standardized protocol over PCIe; very fast

b. Flash Translation Layer
   i. Implement SATA interface from disks
      1. read block, write block
      2. Manage internal parallelism
         a. strip writes, reads across dies/planes
         b. Wear leveling to handle endurance
   ii. Internally implement on FLASH pages/blocks
      1. Challenge: doing writes
         a. read-modify-write of whole erase block (cheap but slow and unreliable)
         b. Log structuring (fast but complicated)
      2. With log structuring, need a map of where to find blocks
         a. Compare to LFS inode map, but need for every block not just file inodes

iii. Local map variations:
    1. Block-level only
        a. can move a whole erase block, but pages must be at the right place in the erase block
        b. Allows wear leveling, but still inefficient
        c. Uses little memory
    2. Page-level
        a. Complete flexibility (like virtual memory)
        b. Need a lot of memory to store, or need extra reads to get from Flash (16x more space)
    3. Hybrid:
        a. "Log area" for incoming writes (LFS active segments) that is page level
        b. "Data area" for post-garbage collection data that is block level (plus bitmap of missing pages)

iv. Wear leveling:
    1. Dynamic wear leveling: change where data is written so not written to same place
    2. Static wear leveling: relocate data that hasn't been written to make low-wear spaces available

v. Garbage collection/cleaning
    1. Essentially same tradeoffs as LFS
    2. Most SSDs use greedy policy, not age-based policy
    3. Overprovision (like reserving space in LFS) to do less frequent GC
        a. More blocks overwritten, so less data to be copied
        b. Allows more GC in parallel, so GC takes less time
    4. Sustained write performance is the performance of the GC

c. Persistence
    i. Both data and logical map must be persistent to find data
    ii. Cheap SSDs write logical map at shut down; on crash, may not have updated
        1. An extra metadata write per block write drops perf in half
    iii. Solution: flash includes "out-of-band" area; 128 bytes per page
        1. Store reverse-map here
        2. Can reconstruct mapping by scanning these
        3. Reduce scanning by tracking which pages could be dirty

d. SATA extensions
    i. One new command needed: “Discard” or “Trim”
        1. Tells device when blocks are no longer needed by FS
        2. Need not be copied as part of GC
        3. Makes GC more like LFS – device is not “always full”
        4. Often slow – some SSDs erase data when discarded.
            a. Much cheaper when larger
e. Performance take-aways
    i. Random reads are fine, but a bit slower than sequential due to less parallelism
    ii. Sequential writes tend to be faster, as don’t need to GC if overwrite complete block
    iii. Random writes devolve to speed of GC; if log-structure with page-based FTL can be as fast as sequential writes until GC kicks off

6. FlashTier – see slides