Designing Storage Systems with Flash

Umesh Maheshwari

CTO, Nimble Storage

June 2012

The Five Buzzwords in Storage

nimblestorage

1. I keep my email in the ………………………….. cloud
2. Most of my servers are ……………………….. virtualized
3. My backup storage does it, and now
   I want my primary storage to do …………….. dedupe
4. My SAN is not big enough for my ………….. big data
5. My storage has the IOPS because it has…. flash

Storage Vendors Building with Flash

nimblestorage

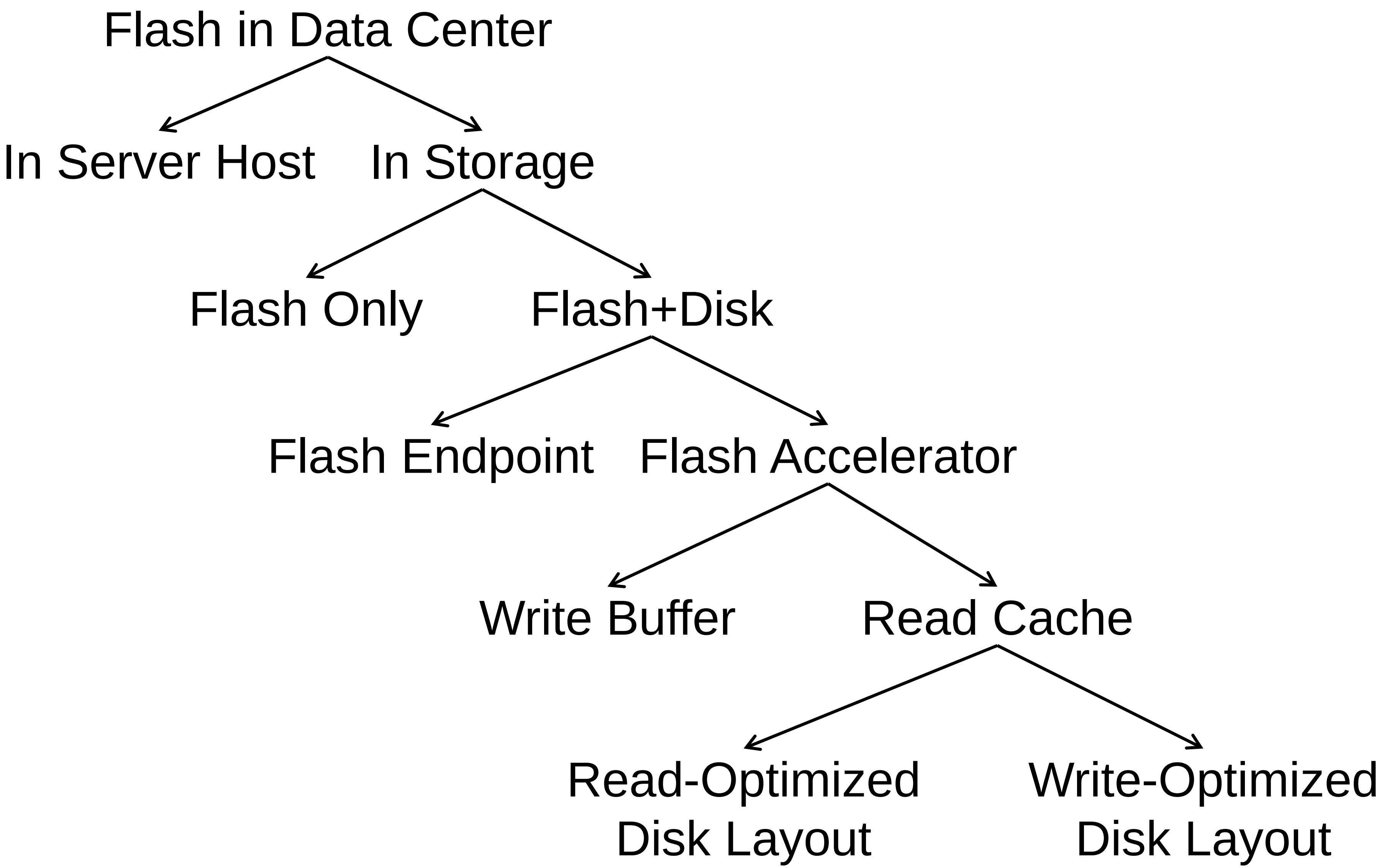
Public companies

- Dell
- EMC
- FusionIO
- HP
- IBM
- NetApp

Private companies

- NexGen
- Nimble Storage
- Nimbus Data
- Nutanix
- Pure Storage
- Solid Fire
- Tegile
- TinTri
- Violin
- WhipTail

Flash Design Space



About Nimble Storage

nimblestorage

- Founded in early 2008:
  - Varun Mehta: NetApp, Panasas, DataDomain.
  - Umesh Maheshwari: MIT, Zambeel, DataDomain,
- Build storage arrays with good performance + capacity.
- Focused on mid-size enterprises.
- Mainstream apps: Exchange, SQL Server, VMFS.
- Started selling in mid 2010.
- The fastest growing storage array company!

Design Decision Point 1

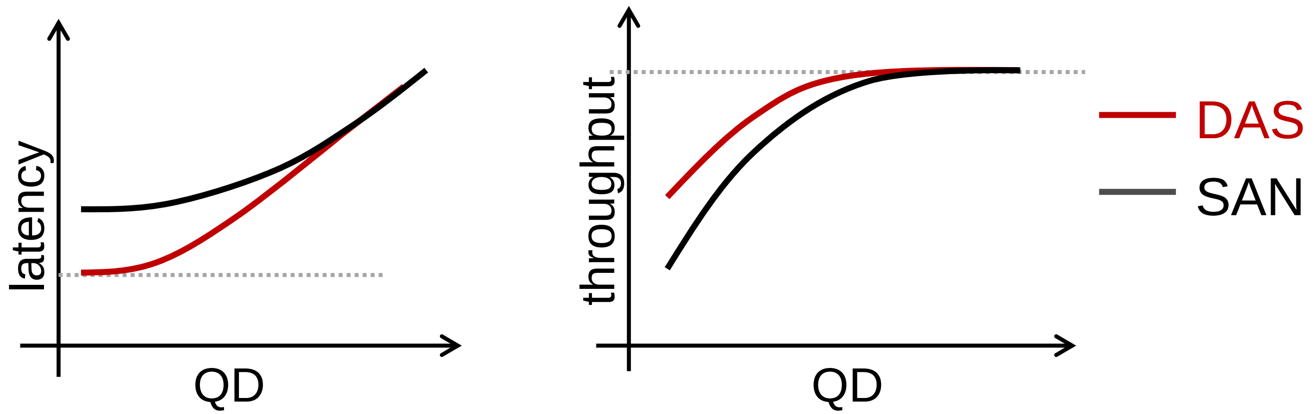
Flash in Data Center

In Server Host — In Storage

- In server host = direct-attached (DAS) = not shared
- In storage = SAN/NAS = shared

Reasons for Flash in Server Host

1. Lower latency
   – Flash read latency 20--200us.
   – Network roundtrip latency 10--100us.
   – Relevant apps: real-time trading.
   – But for most apps, network latency masked by queuing.
2. Scalable bandwidth
   – Don't need expensive network or scale-out storage.
   – Relevant applications: data analytics.
   – But for most apps, throughput not bound by network bandwidth.



Reasons for Flash in Storage

nimblestorage

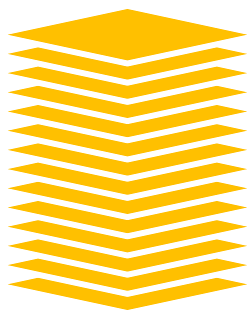
1. Supports multi-host applications
   – Distributed applications, e.g., file service.
   – VM migration from one host to another.
   – Can be done with DAS with host-to-host access
     - But complicated.
     - Lose proximity advantage of DAS.
2. High availability
   – Remains available when server host goes down.
   – Remains available even when storage controller goes down.
   – Can be done with DAS
     - But requires mirroring data.
3. Only flash in storage can accelerate storage block map.
   – Small amount of flash in storage has big impact.
   – Flash in host does not obviate flash in storage.

Design Decision Point 2

Flash in Storage

Flash Only

Flash+Disk Hybrid



Reasons for Flash-Only Storage

nimblestorage

1. Guaranteed high performance.
   – Lowers worst-case latency, not just expected latency.
   – Except possibly during garbage collection (GC).
   – Relevant apps: trading, medical?
2. Single is simpler.
   – Flash and disk have different performance/failure traits.
3. No mechanically moving parts.
   – Tolerates vibrations.
   – Relevant apps: military.

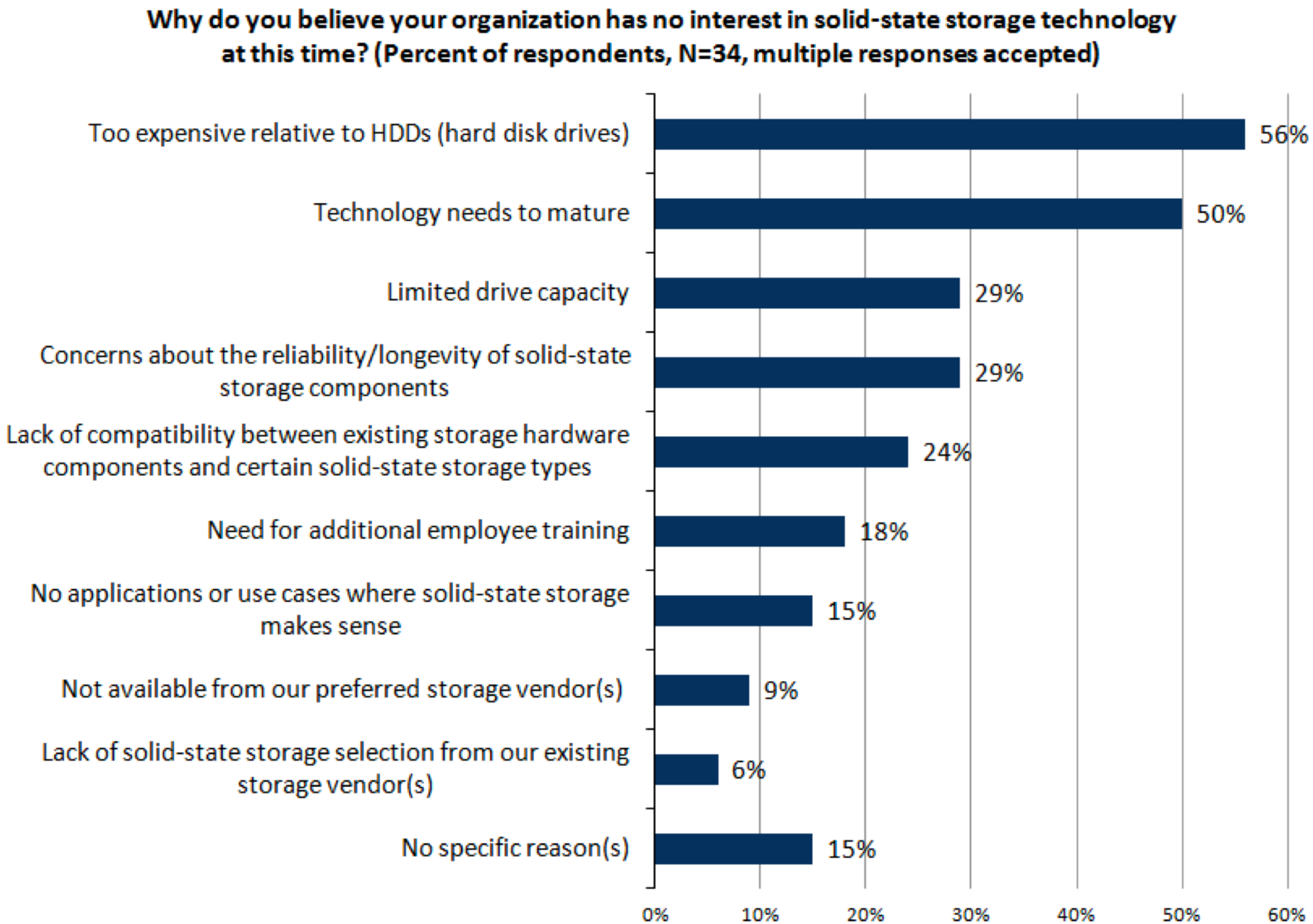
Reasons for Flash+Disk Hybrid

nimblestorage

1. Flash is expensive
   - Enterprise hi-density HDD: $0.10/GB.
   - Commodity MLC SSD: $1.50/GB (15x).
   - Enterprise MLC SSD: $3/GB (30x).
     - Sophisticated controller and firmware.
     - Overprovisioning to reduce write amp.
   - Not clear how fast flash:disk price ratio will fall.
     - Increasing density ➔ poorer performance and reliability.
   - Data reduction helps, but
     - Compression works on both disk and flash (1.5—3x).
     - Dedupe easier on flash, but less dependable (good case: 2x).
2. Much of data is cold
   - Old email, stopped VMs, inactive test/dev databases.
   - Data protection based on snapshots + replication.

ESG: Reasons for Not Adopting Flash

Figure 2. Why Non-adopters of Solid-state Storage Have No Interest in the Technology



Source: Enterprise Strategy Group, 2011.

Reasons for Flash+Disk Hybrid

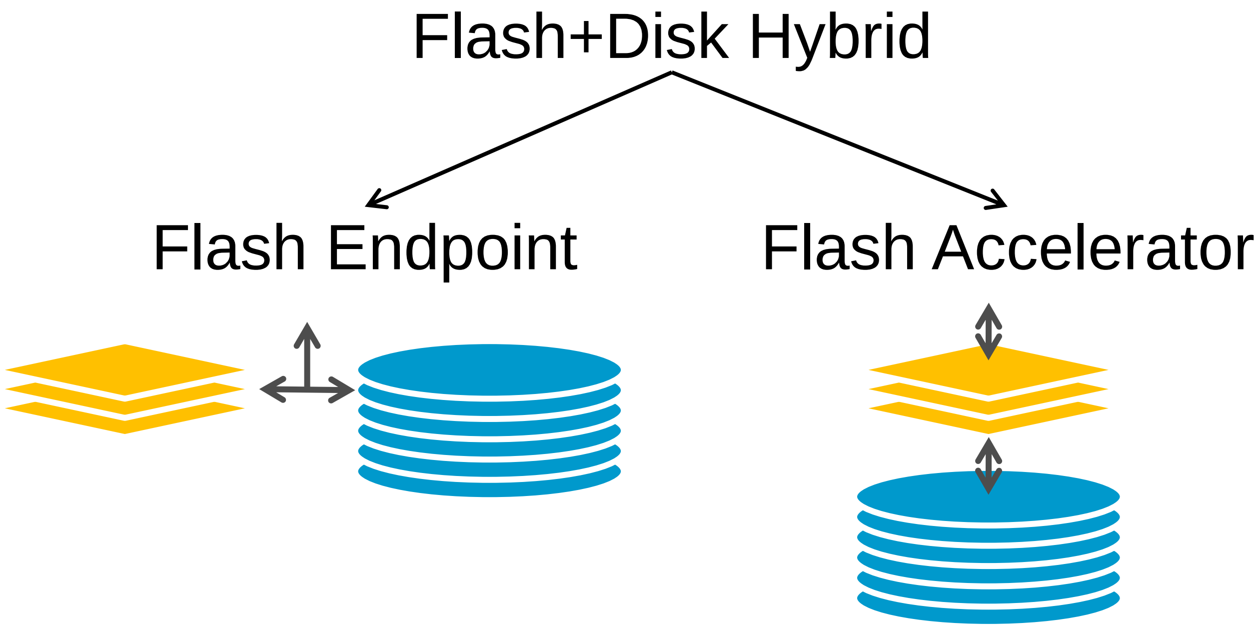
nimblestorage

3. Uncertain flash reliability.
   - Disk invented in 1950's.
   - Flash invented in 1980's, entered storage in 2000's.
   - Peculiarities: write-endurance, retention, read-disturb, etc.
   - Can be mitigated, but at further expense.

- Is flash the right solid-state replacement for disk?

Design Decision Point 3

nimblestorage



- Endpoint = resting point = tier = peer of disk
- Accelerator = cache/buffer for data on disk

Reasons for Flash as Endpoint

nimblestorage

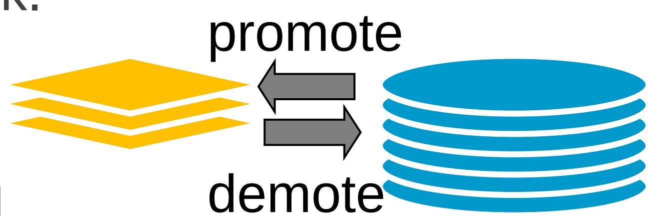
1. Simpler to use flash as peer of disk.
   – But automatic tiering is complex.
2. Avoids duplicate data on flash and disk.
   – But disk is 30x cheaper.

Reasons for Flash as Accelerator

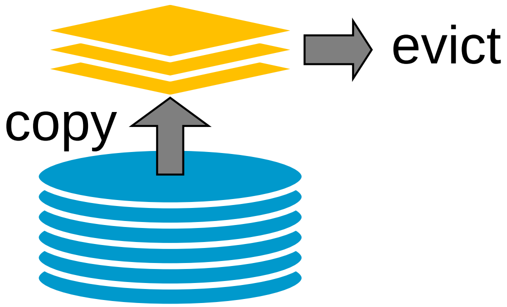
1. Migrating data between endpoints is slow.

   Causes:
   - Changing home location is expensive.
   - Promoting to flash requires demoting to disk.

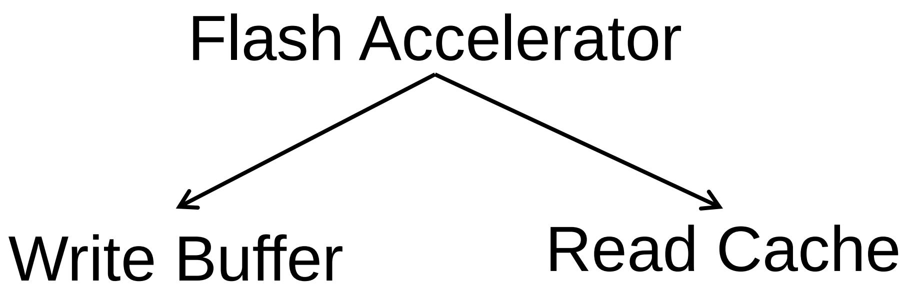
   Effects:
   - Bigger cost demands bigger benefit.
   - Takes hours/days to judge data as hot/cold.
   - Migrates data in large chunks.
   - Encourages “pinning” entire data set in flash.
   - Requires sizing flash tier conservatively.



2. Copying data for acceleration is fast.
   - Location in flash is temporary.
   - Copying data to flash does not require additional disk access.



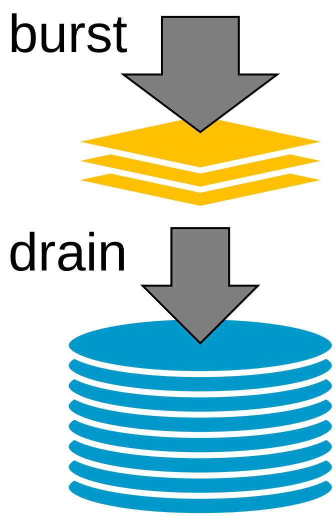
Design Decision Point 4



- Write buffer = write-back caching
- Read cache = write-through / write-invalidate

Reasons for Flash as Write Buffer

1. Low-latency writes.
   – Disk latency: 5ms, flash latency: 200—600us.
   – But NVRAM (DRAM + power protection) is faster: 50us➔50ns.
   – And NVRAM does not burn out.
2. Absorbs large burst of writes.
   – Larger buffer than NVRAM.
   – But sustainable throughput limited by drain to disk.
3. Greater opportunity to absorb overwrites.
4. Greater opportunity to re-sort writes.
   – Improves sequentiality of drained data.
   – But limited by consistent checkpoints on disk.



Reasons for Flash as Read Cache

nimblestorage

1. Allows control over flash wear.
   – Need to insert only cache-worthy data in flash.
   – Can throttle cache insertions based on remaining life.
2. Tolerates unreliability of flash gracefully.
   – Data in cache is subset of data on disk (all “clean”).
   – Can checksum, verify, and discard on failure.
   – Does not need high endurance flash.
3. Does not need parity or mirroring.
   – Dual parity costs around 20%.

Design Decision Point 5

Flash as Read Cache

Read-Optimized Disk Layout

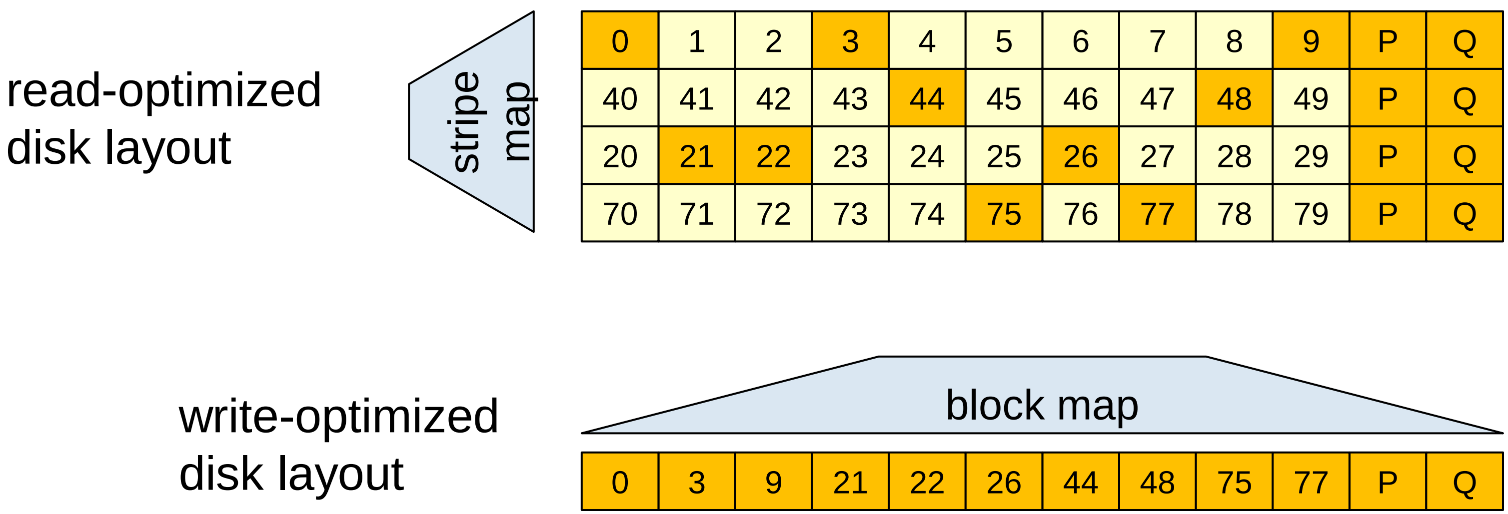
Write-Optimized Disk Layout

- Read-optimized = traditional, write in place
- Write-optimized = write coalescing

Reasons for Read-Optimized Disk Layout

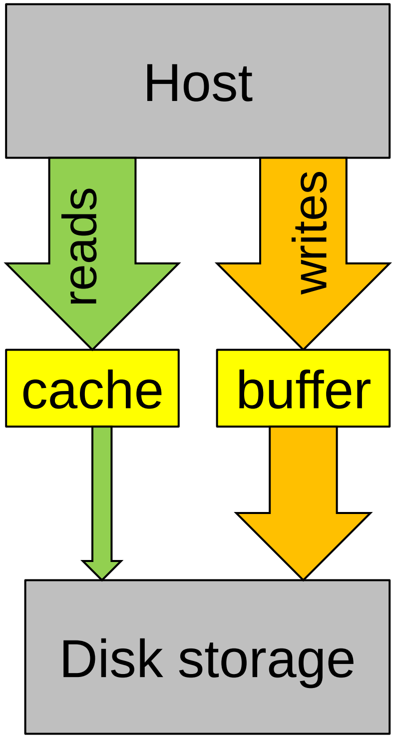
nimblestorage

1. Preserves sequential locality on disk.
   – Fast sequential reads.
2. Smaller map of logical addresses to physical locations.
   – Common unit of mapping = RAID stripe.
   – Write-optimized unit of mapping = block.



Reasons for Write-Optimized Disk Layout

1. I/O to disk increasingly dominated by writes.
   – Large caches in host/storage reduce reads greatly.
   – Rosenblum in 1991: *Increasing memory sizes will make the caches more and more effective at satisfying read requests. As a result, disk traffic will become dominated by writes.*
   – Buffers do not reduce writes much.
2. Coalesces random writes.
   – Leverages sequential throughput of disk.
3. Redirect-on-write snapshots.
   – No copy-on-write overhead.



Write-Optimized Disk Layouts

nimblestorage

- Write in holes
  - Writes into free space: gets fragmented over time.
  - Opportunistic coalescing based on hole size.
  - E.g., WAFL, ZFS.
- Write in large stripes
  - Aka log-structured.
  - GC sweeps holes into full stripes.
  - Guaranteed coalescing.
  - Supports variable block size: compression.
  - Preserves write-order locality for reads.
  - Prevalent within SSDs, not with disk.
  - GC requires fast and efficient block map---enabled by flash!

An Efficient, Balanced System

nimblestorage

Large flash cache

↙ ↘

Read IOPS

Optimized block map

↓

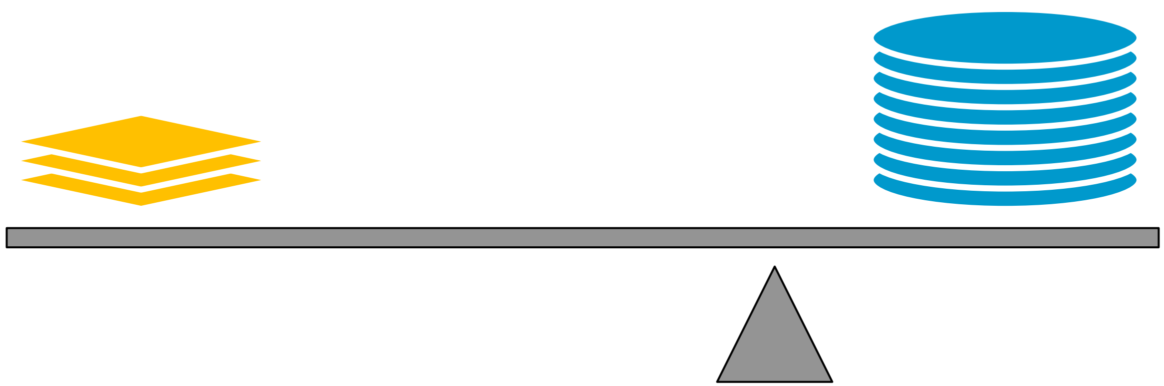
Garbage collection

↓

Write in large stripes on disk

↓

Write IOPS

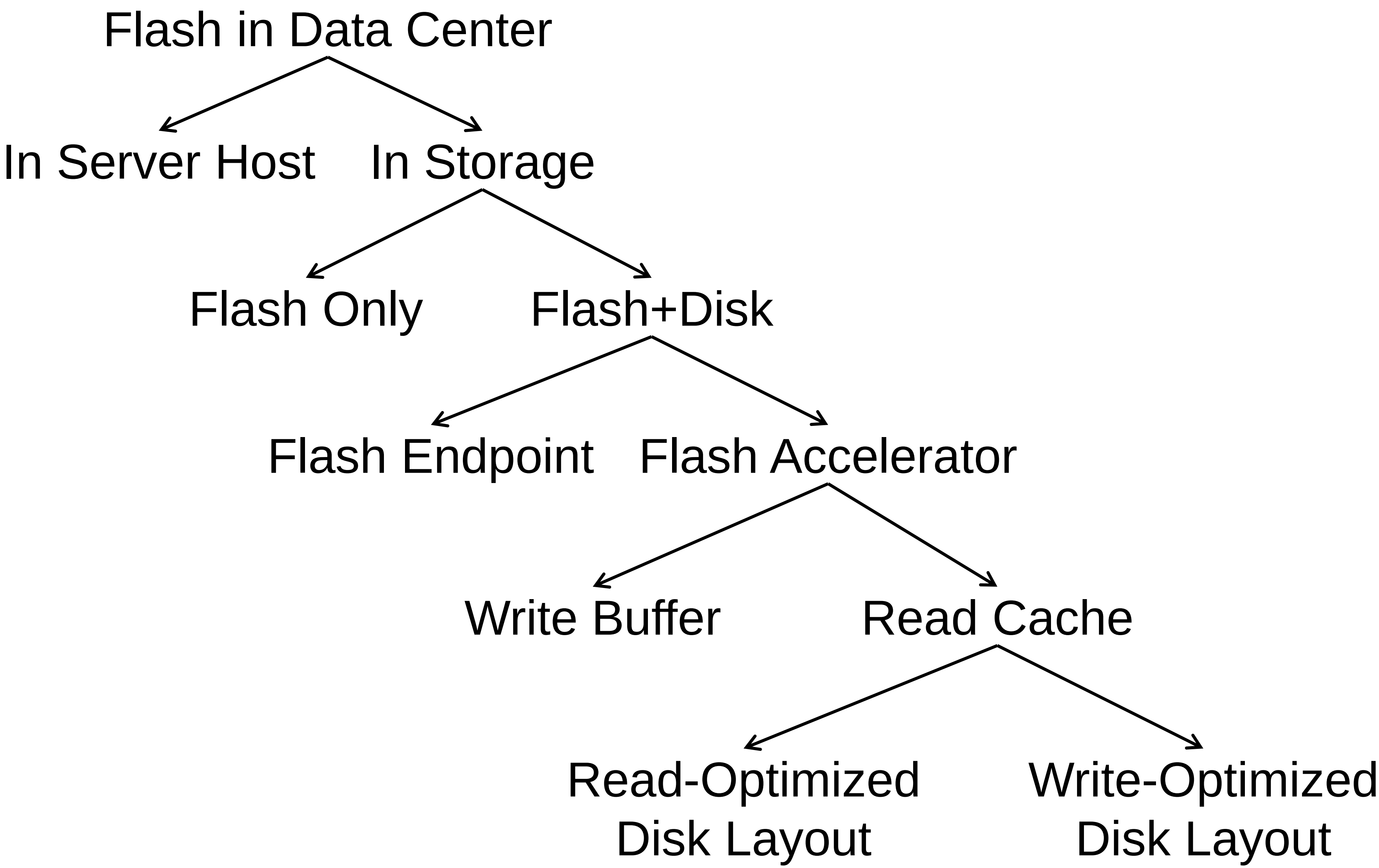


Wish List

nimblestorage

1. SSDs un-encumbered by random-write optimizations.
   - Manage bad-blocks, wear-leveling, read-disturb, but not GC.
     - Re-map at erase-unit level, not page level.
   - Storage SW can do GC at system level, more predictably.
   - Smaller map ➔ faster operation.
     - Will remove a layer of indirection from SSD.
     - Storage SW can subsume it into its own layer.
2. SSD API optimized for caching
   - Leaky block device?
3. More research on cache effectiveness and QoS
   - Lightweight algorithm for predicting miss rate vs. cache size.
   - Given multiple workloads, how much cache to give to each
     - To maximize efficiency.
     - To meet SLAs.

Questions?


Flash in Data Center
In Server Host
In Storage
Flash Only
Flash+Disk
Flash Endpoint
Flash Accelerator
Write Buffer
Read Cache
Read-Optimized Disk Layout
Write-Optimized Disk Layout