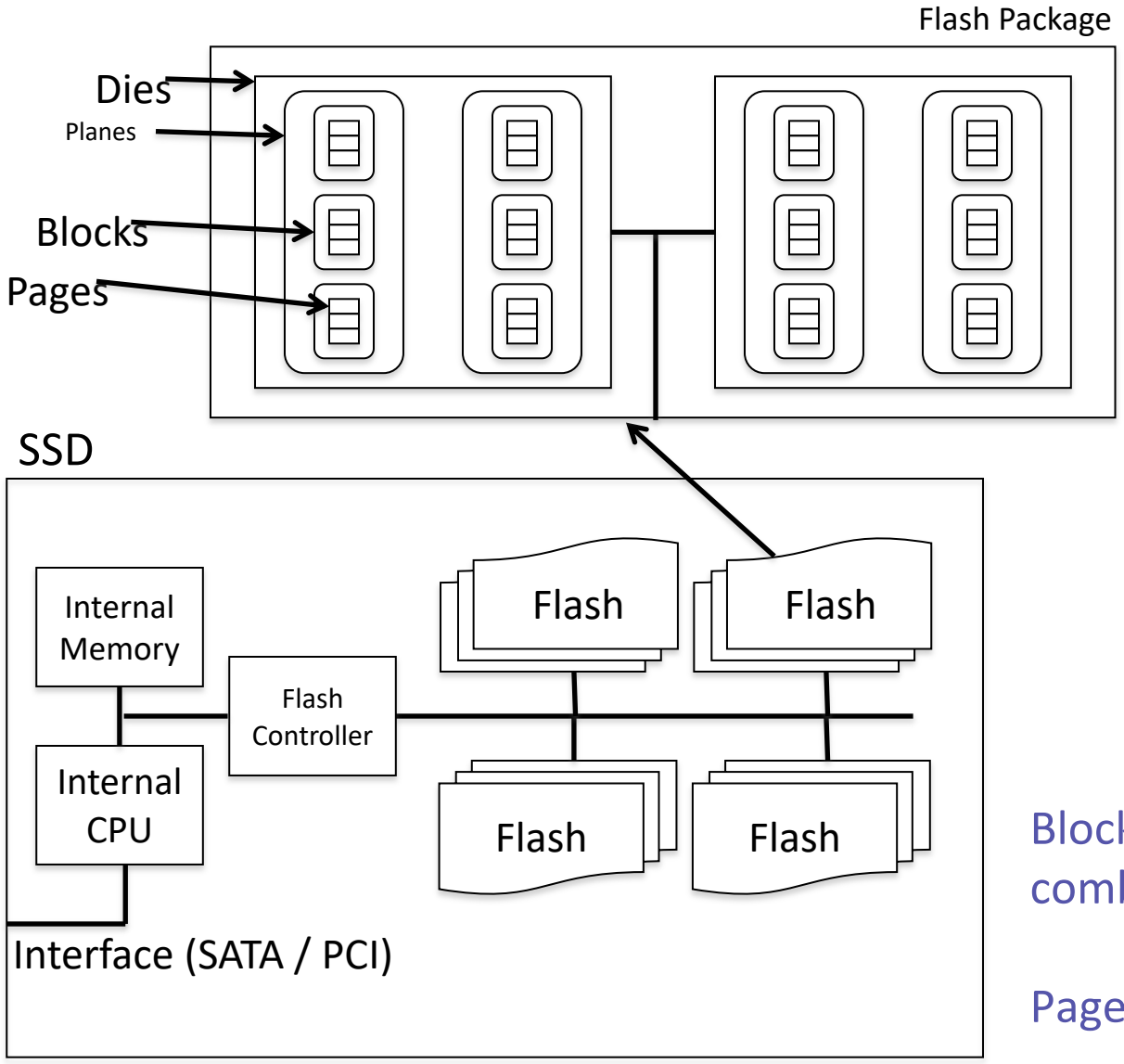
Solid State Storage and Databases: Where and How?

Flash disks

- Secondary storage *or* caching layer.
- Main advantage over disks: *random reads* equally fast as *sequential* reads.
- BUT: Slow random writes.
- Data organized in *pages* (similarly to disks) and pages organized in *flash blocks.*
- *Like RAM*, time to retrieve a disk page is not related to location on flash disk.

The Internals of Flash Disks



- Interconnected flash chips
- No mechanical limitations
- Maintain the block API – compatible with disks layout
- Internal parallelism in read/write
- Complex software driver

Blocks are organized as a combination of pages

Pages -> Blocks -> Planes -> Dies

Accessing a Flash Page

- Access time depends on
  - Device organization (internal parallelism)
  - Software efficiency (driver)
  - Bandwidth of flash packages

- Flash Translation Layer (FTL)
  - Complex device driver (firmware)
  - Tunes performance and device lifetime

Random Write: Copy content of a page from one place to another page, then append to it and write it to another page., and delete the old page. This causes random write speed is bit slower
Serial Write: Keep content in the main memory until a new flash page is available.
Deletion: Only able to delete a block than a page because the voltage is so high that cannot be isolated for a single page deletion

Flash disks vs HDD

HDD

✓Large – inexpensive capacity

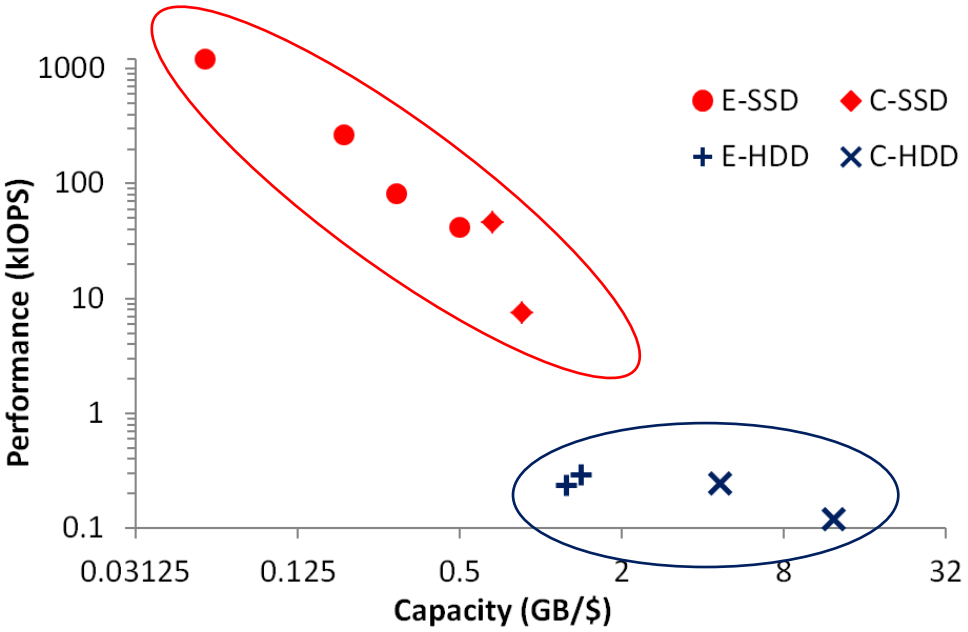
x Inefficient random reads

Flash disks

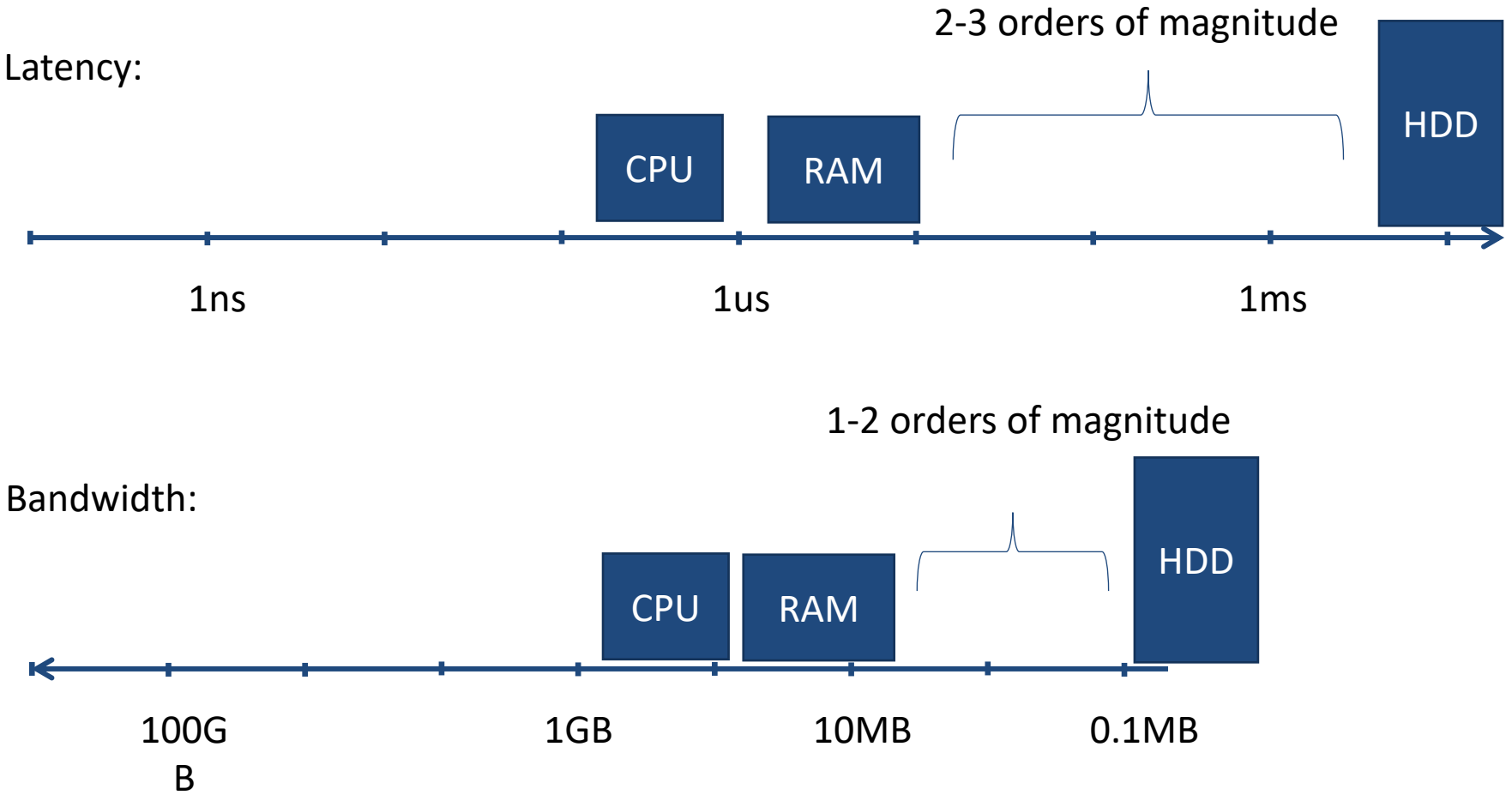
x Small – expensive capacity

✓Very efficient random reads

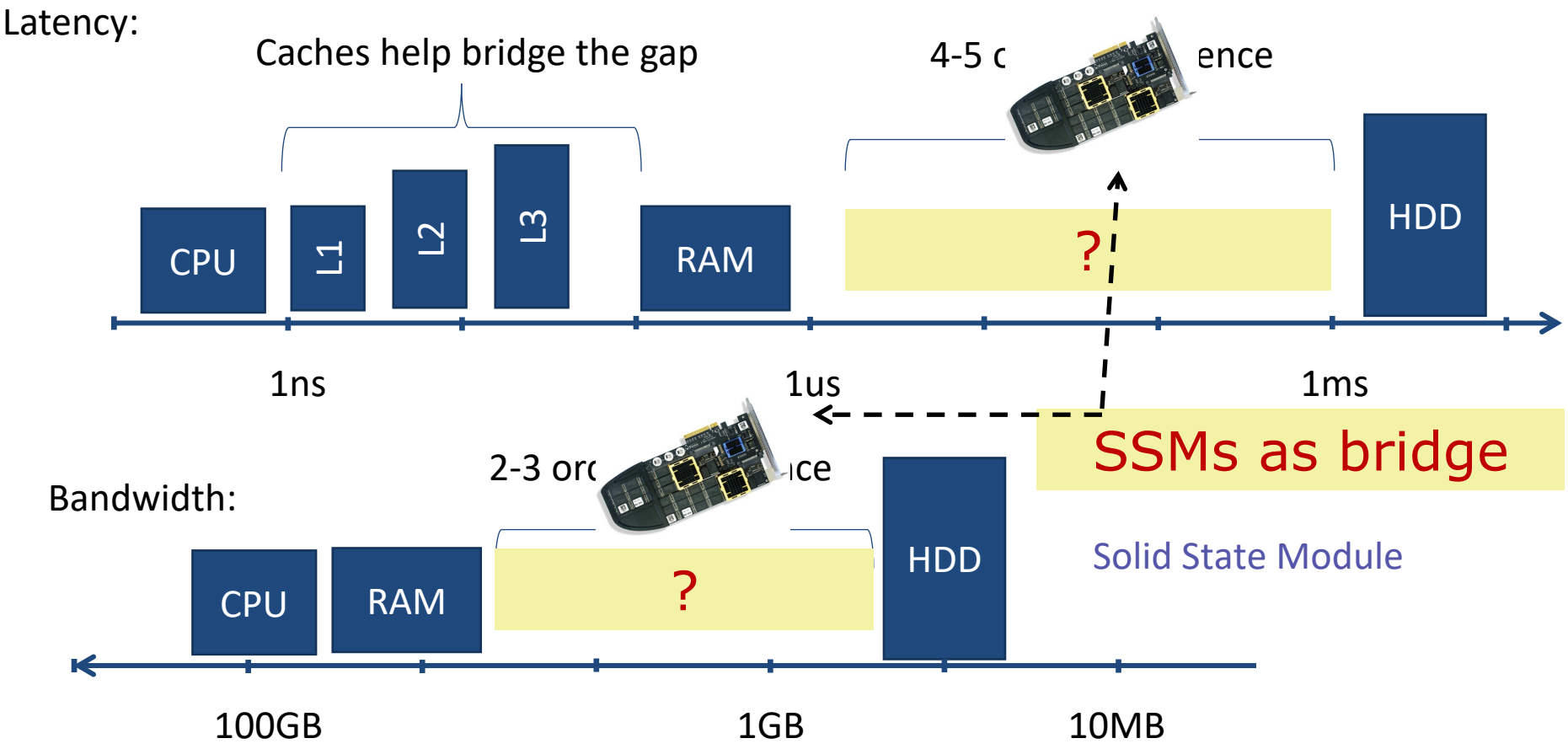
Enterprise v.s. Consumer



Storage Hierarchy -- 80's



Storage Hierarchy -- Now



SSM today

Phase-Changed Memory: Non-volatile RAM

- Only Flash & PCM pursued commercially
  - Flash most developed, PCM promising competitor

- Flash

Flash parameter	Status	Trend
Density	Not enough	↗☺ Denser => More Errors
Bulk erase size	Problematic	↗ ☹
Access time	Good	↗ (slowly) 😐
Endurance	Bad	↘ ☹

- PCM

PCM parameter	Status	Trend
Density	Too low	↗☺
Access time	Very good	↗ 😐
Endurance	OK	?

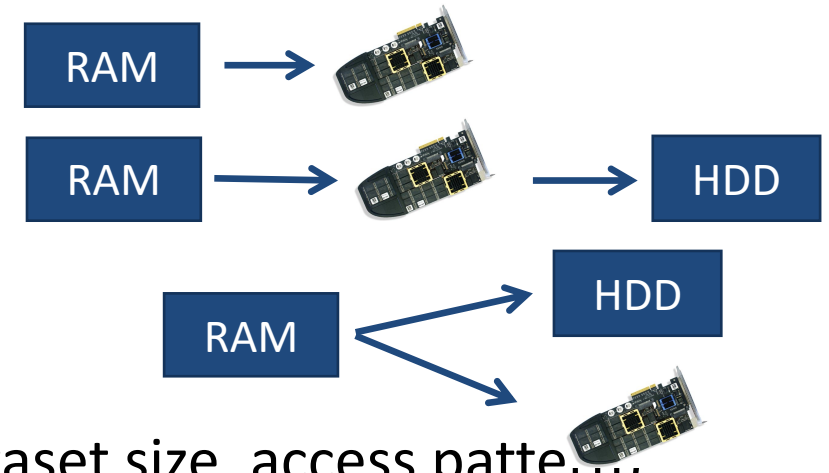
Somewhere below Memory but above HDD. It will not replace Flash, but as a subsitute

Storage and Data Management

- DBMS traditionally designed from ground up around a HDD model
- Some common HDD optimizations
  - Data structures:
    - B-trees, bitmap indexes, column organization, compression
  - Query plans (prefer sequential vs random access)
  - Buffer pool, buffering policies, Write-ahead logging
  - Column stores

What to do with flash?

- Flash position in memory hierarchy
  - HDD replacement
  - Intermediate layer
  - Side by side with HDDs



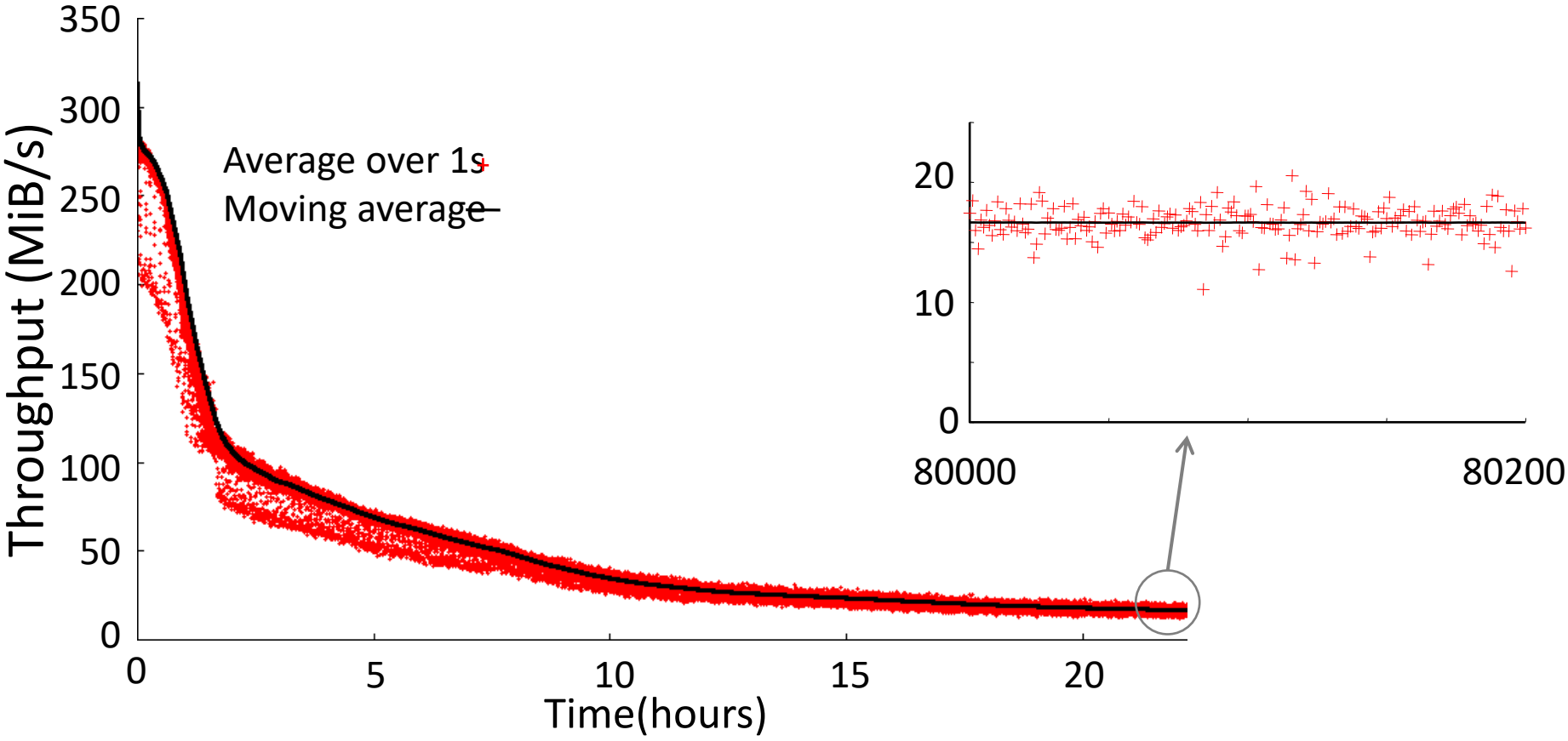
- No “correct” use
  - Depends on workload (dataset size, access patte
  - Future trends: e.g. flash density competitive with HDD

1) Flash-only OLTP

Online Transaction Processing

- OLTP I/O dominated by random reads/writes
- Random reads/writes much faster on flash
  - Also, smaller random-to-sequential gap
- Flash-resident workload
  - Usually a couple of flash devices can hold working set
- Should benefit from fast random access of flash

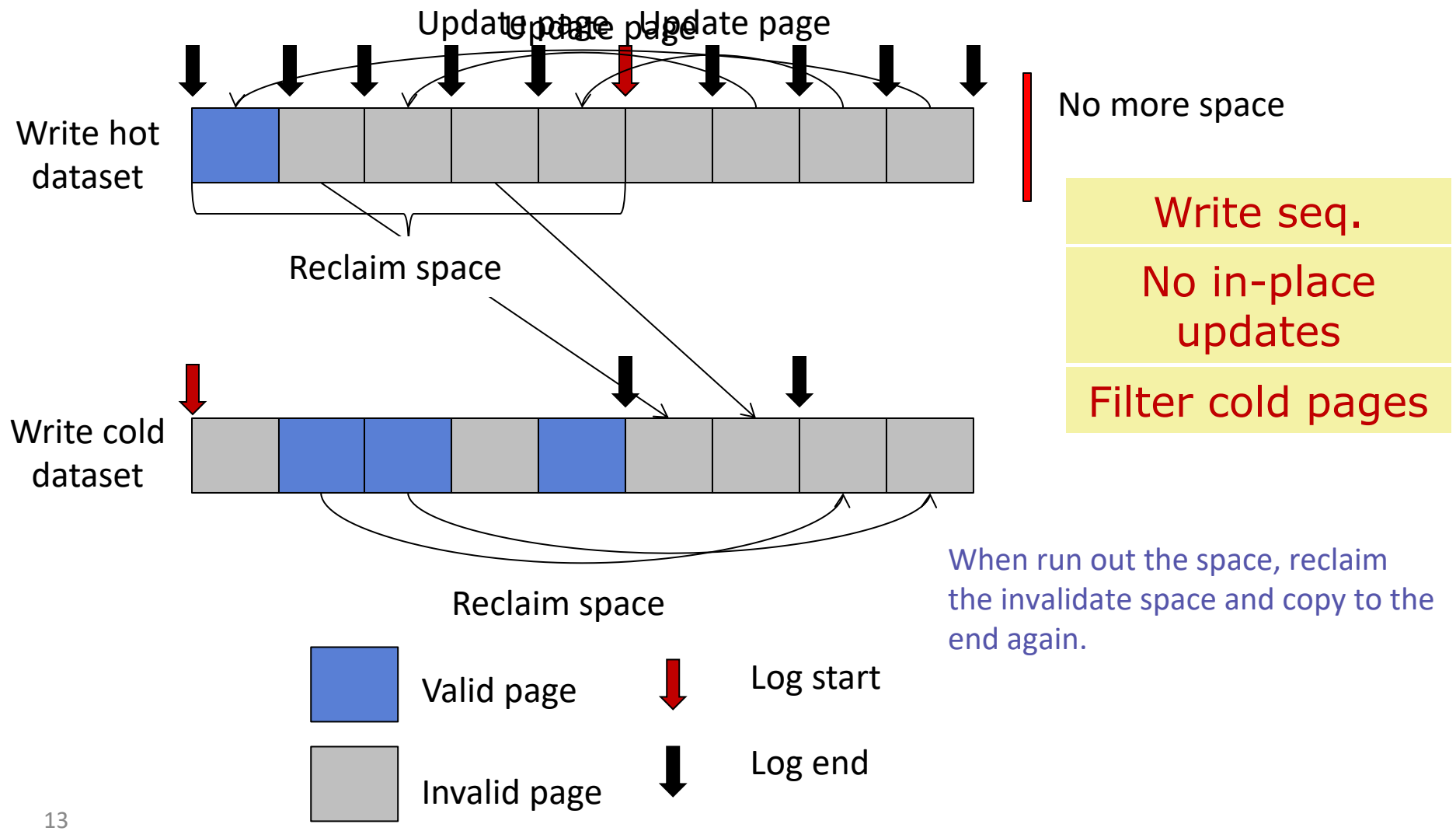
8KiB random writes – Fusion ioDrive



For Random Write -> Performance drops considerably, and not stable (i.e., High Variance)

Append/Pack

Write sequentially as much as we can.
If need in-place update, no Update to original place, but append to the end and invalidate the original page



Write seq.

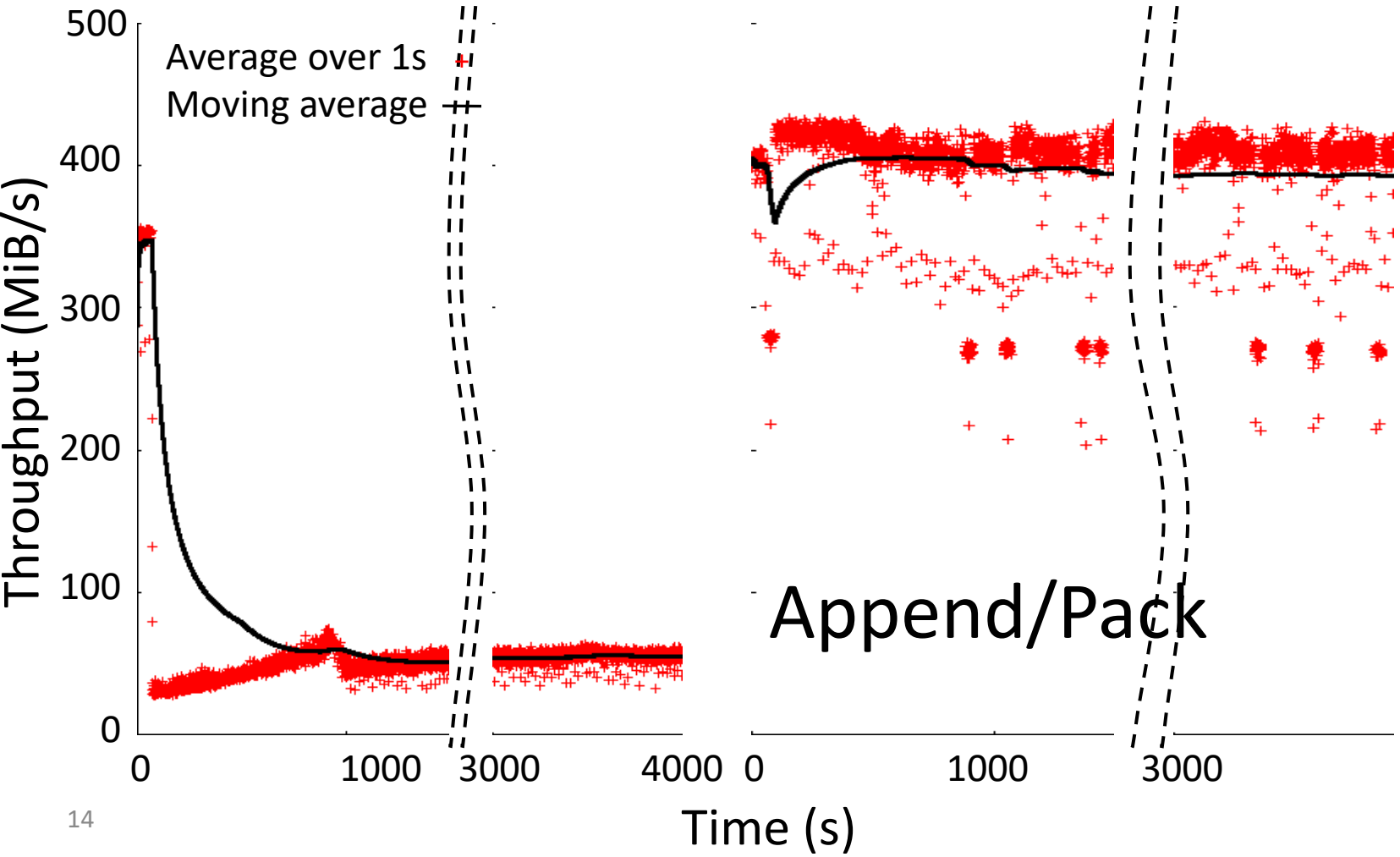
No in-place updates

Filter cold pages

When run out the space, reclaim the invalidate space and copy to the end again.

Append/Pack on Fusion 160GB PCIe

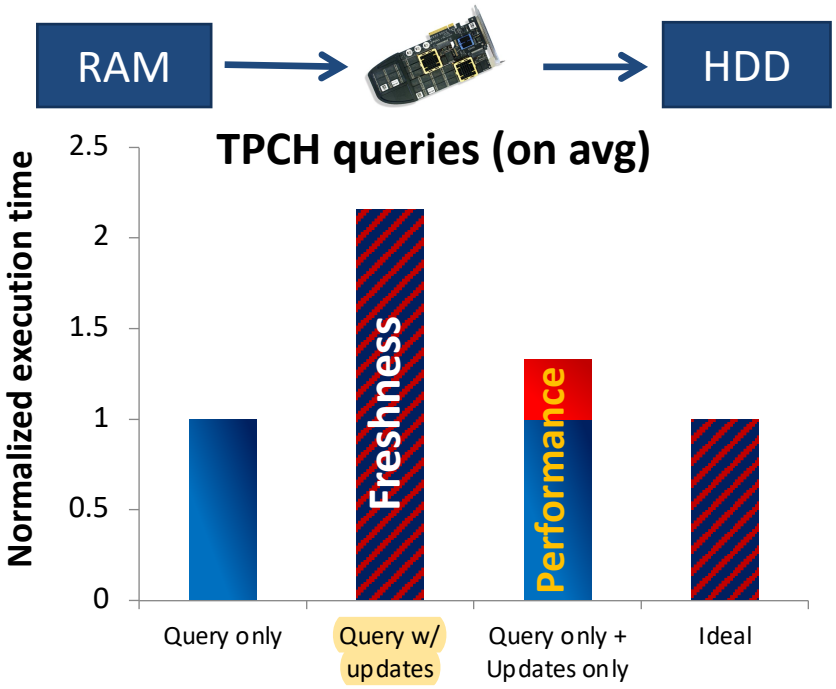
>16 threads, 50% Rand Write / 50% Rand Read, 8KiB I/Os



2) Flash-aided Business Intelligence (OLAP)

Online Analytical Processing

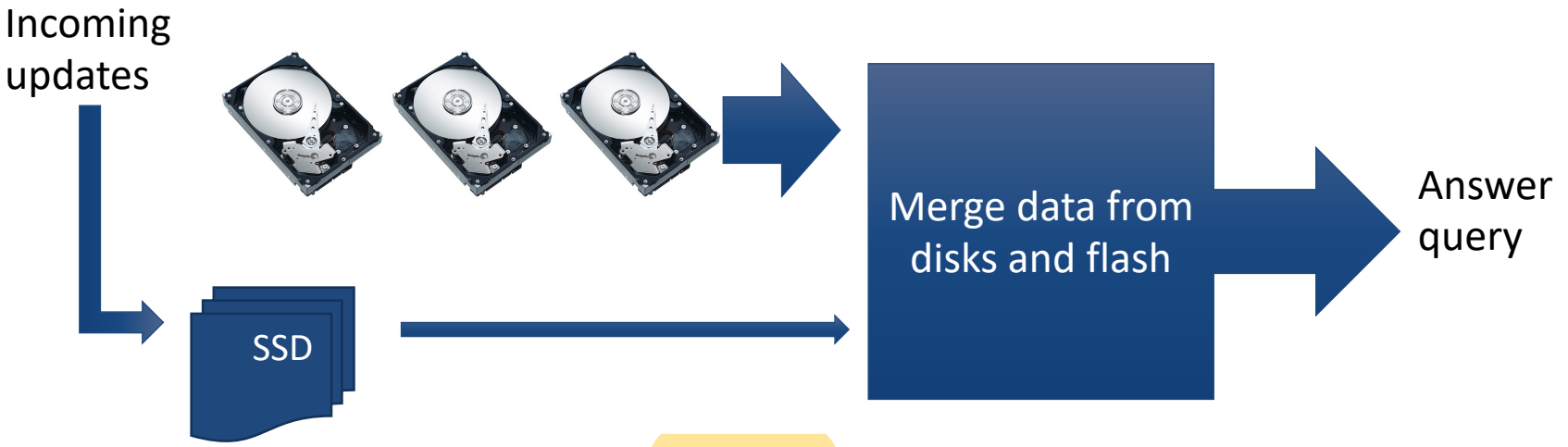
- Data warehouse workload
  - Read-only queries (scans)
  - Scattered updates
  - How to combine *efficiently*?



- Traditionally two choices
  - **Freshness:** in-place updates
  - **Performance:** batch updates

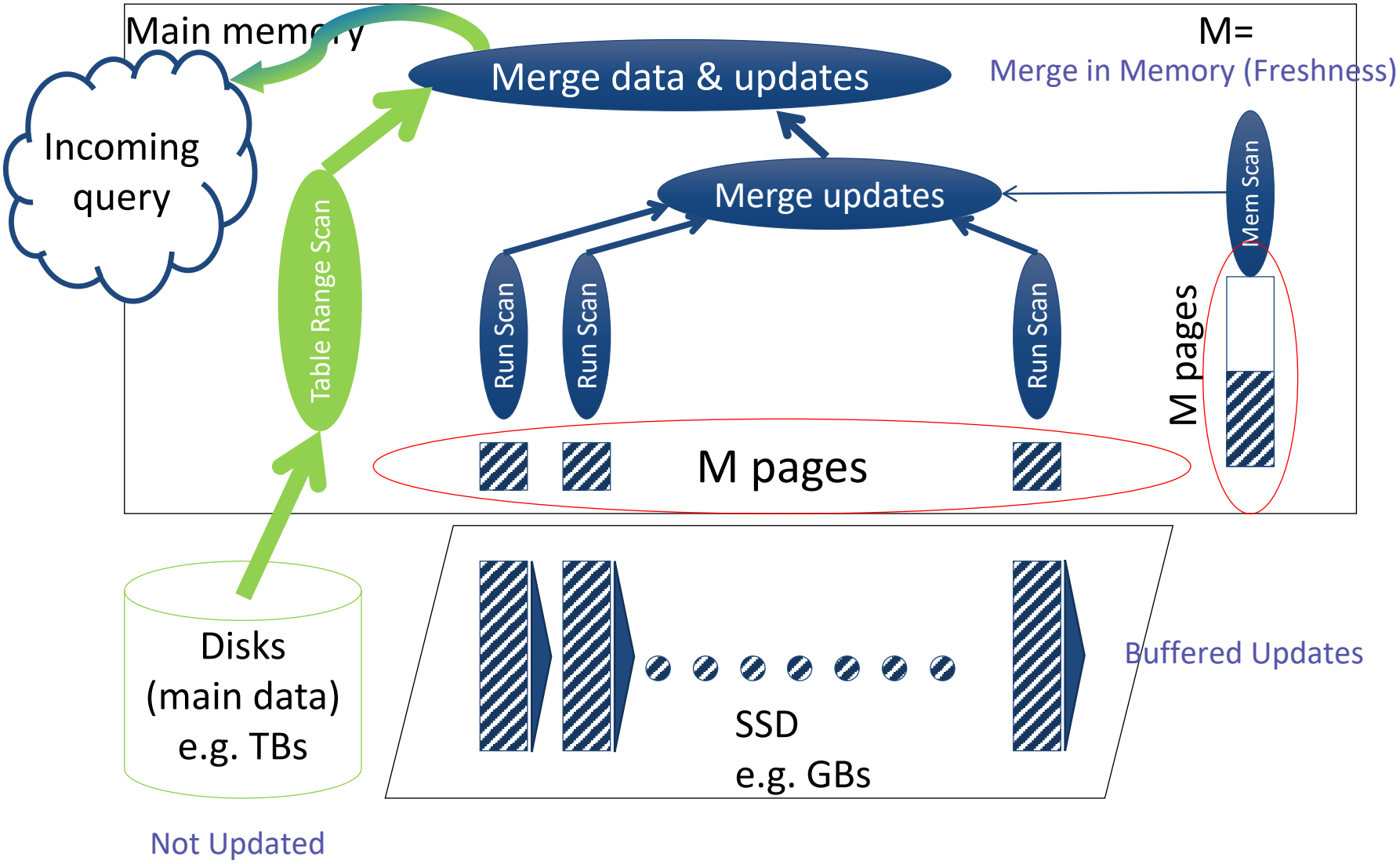
- Ideally, *zero* overhead

Flash as a (write) cache for analytics

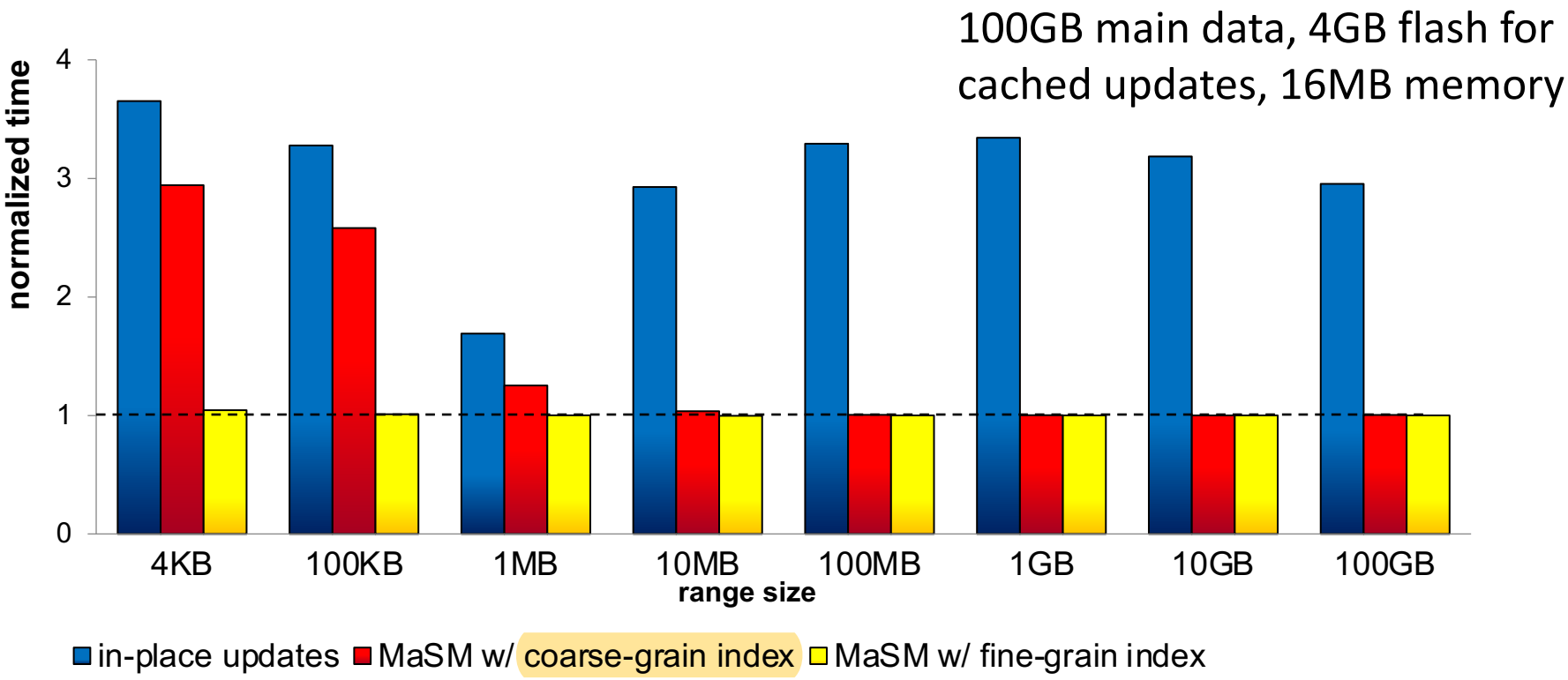


- Buffer updates on Flash instead of memory
  - Flash has *larger capacity* and *smaller price*
- **But:** Flash limitations
  - Access time: Avoid random writes
  - Endurance: Limit/control total # of writes

Materialized Sort-Merge (MaSM)

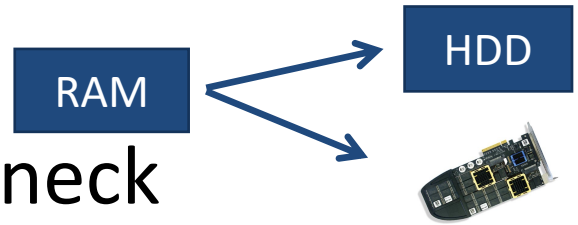


Seagate Barracuda + Intel X25-E SSD

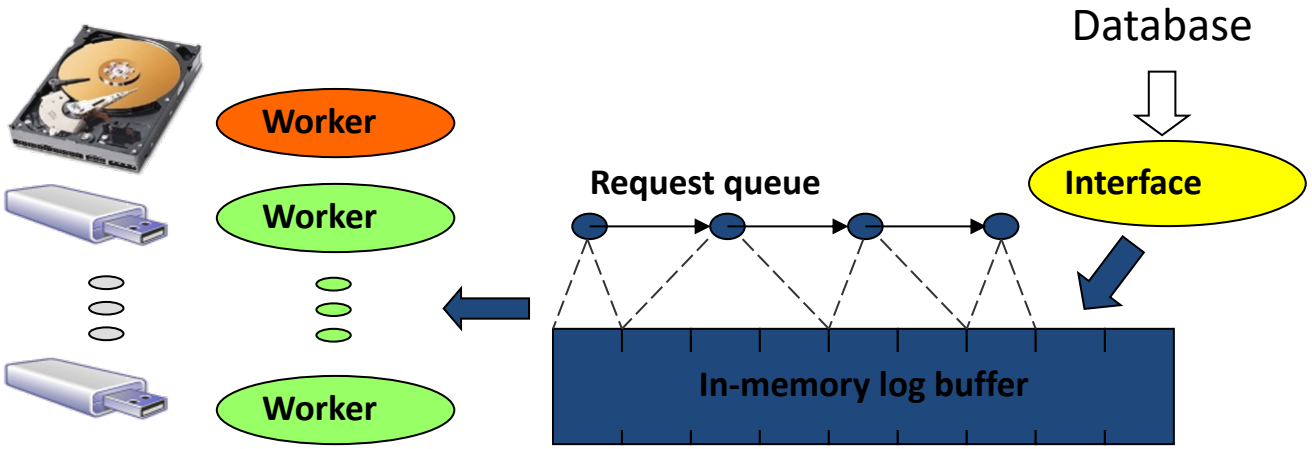


- negligible impact on 10MB or larger scans
- fine-grain index incurs 4% overhead for 4KB ranges (modeling point queries)

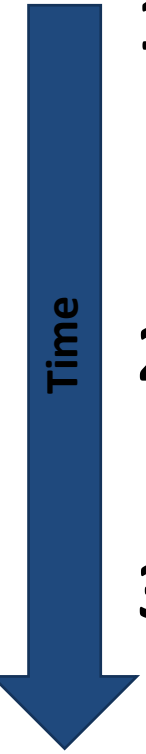
3) Logging on Flash+HDD



- Transactional logging: major bottleneck
  - Today, OLTP DBs fit into main memory
  - But still must flush redo log to stable media
- Log access pattern: small sequential writes
  - HDDs incur full rotational delays



SSD+DBMS: Where and how?



1. SSM as helper of a memory level (DBMS unchanged)

2. Adapt I/O pattern, "small" DBMS changes

   Adopt more Random Read in DBMS for SSD

3. Change storage mgmt, query optimization

   Query Optimization: Particularly for SSD strength

Conclusions

- SSM *can* help bridge the I/O gap

  But SW needs to help in building! Software Adopt SSD Features

- Many flash/SSM uses in data management
  - Stream processing, hash tables, graph DBs
- SSM a very rapidly evolving field
  - several possible commercially viable technologies
  - memristor variations