Solid State Storage and Databases: Where and How?

Thomas Heinis
t.heinis@imperial.ac.uk
Scale Lab - scale.doc.ic.ac.uk

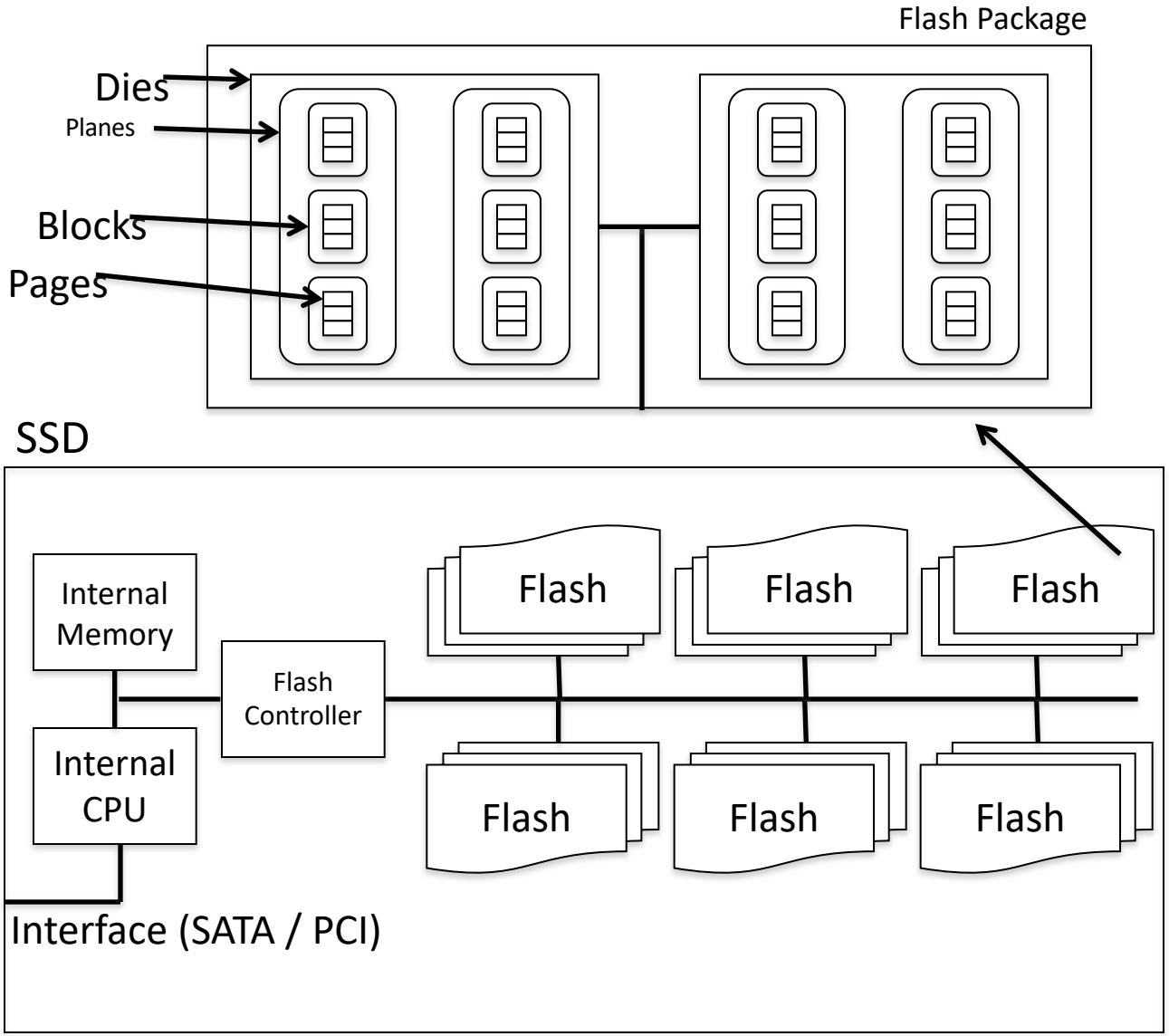
SCALE LAB

Imperial College London

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

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

Flash disks vs HDD

HDD

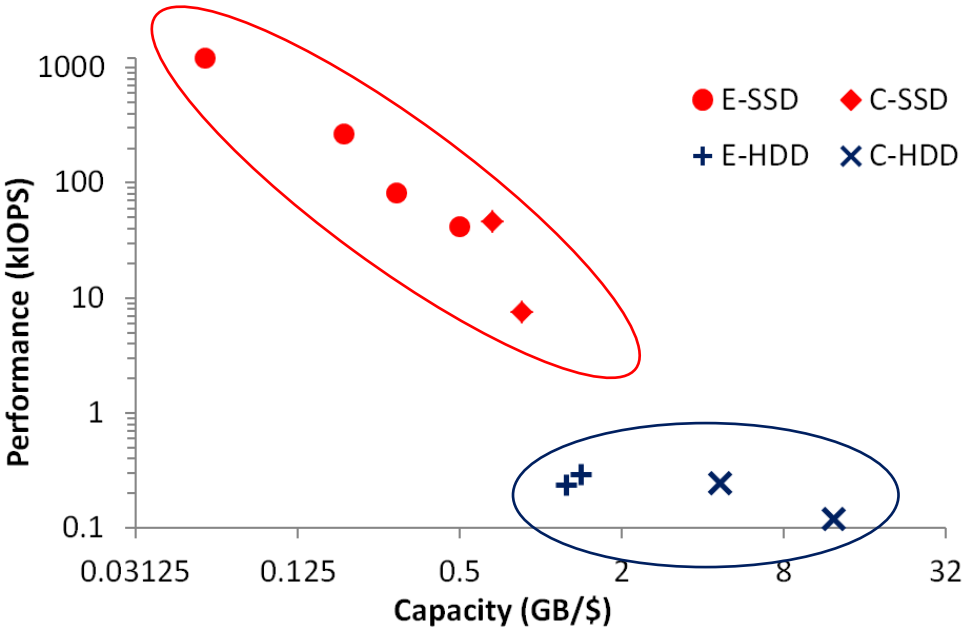
✓ Large – inexpensive capacity

x Inefficient random reads

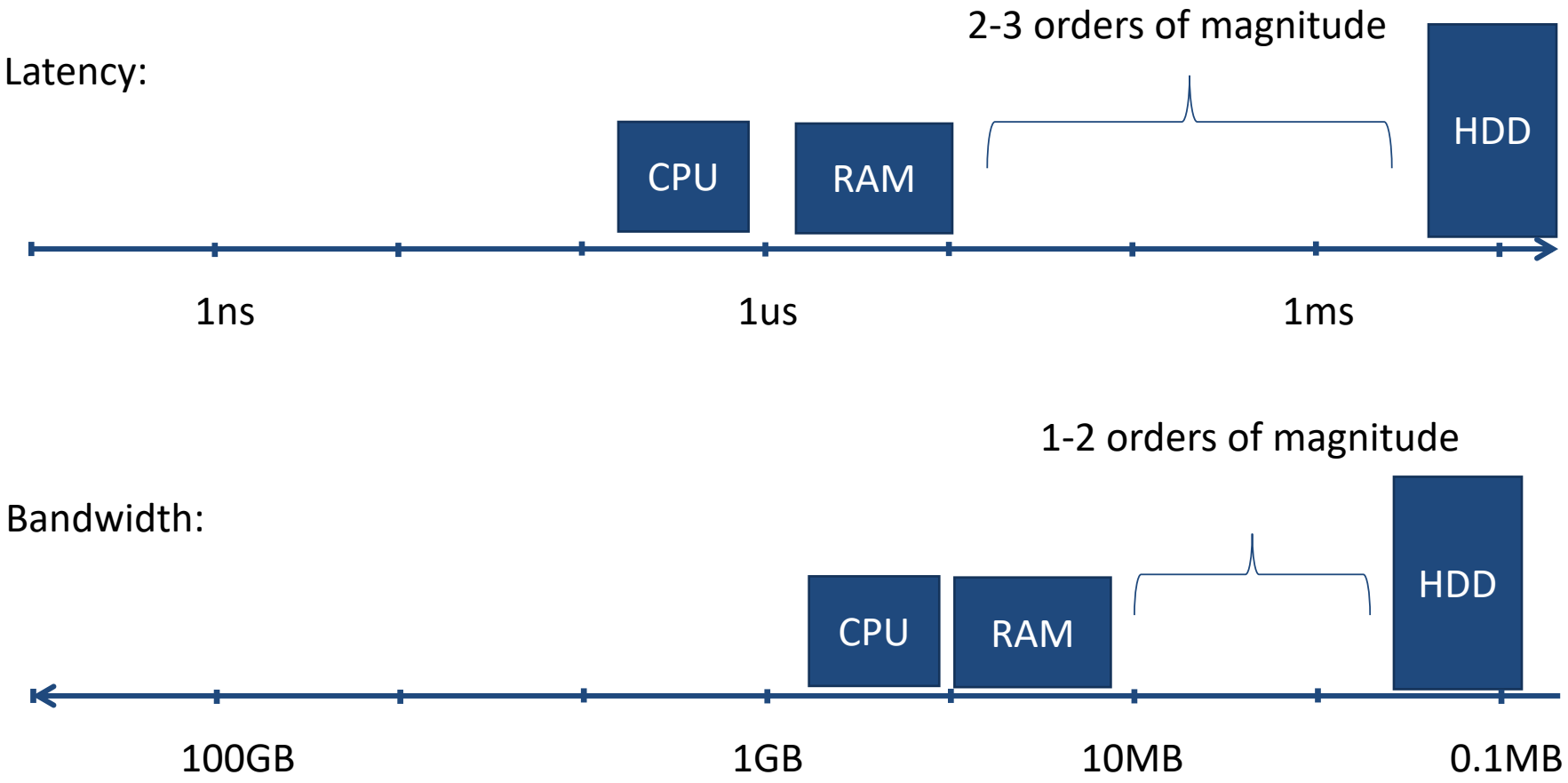
Flash disks

x Small – expensive capacity

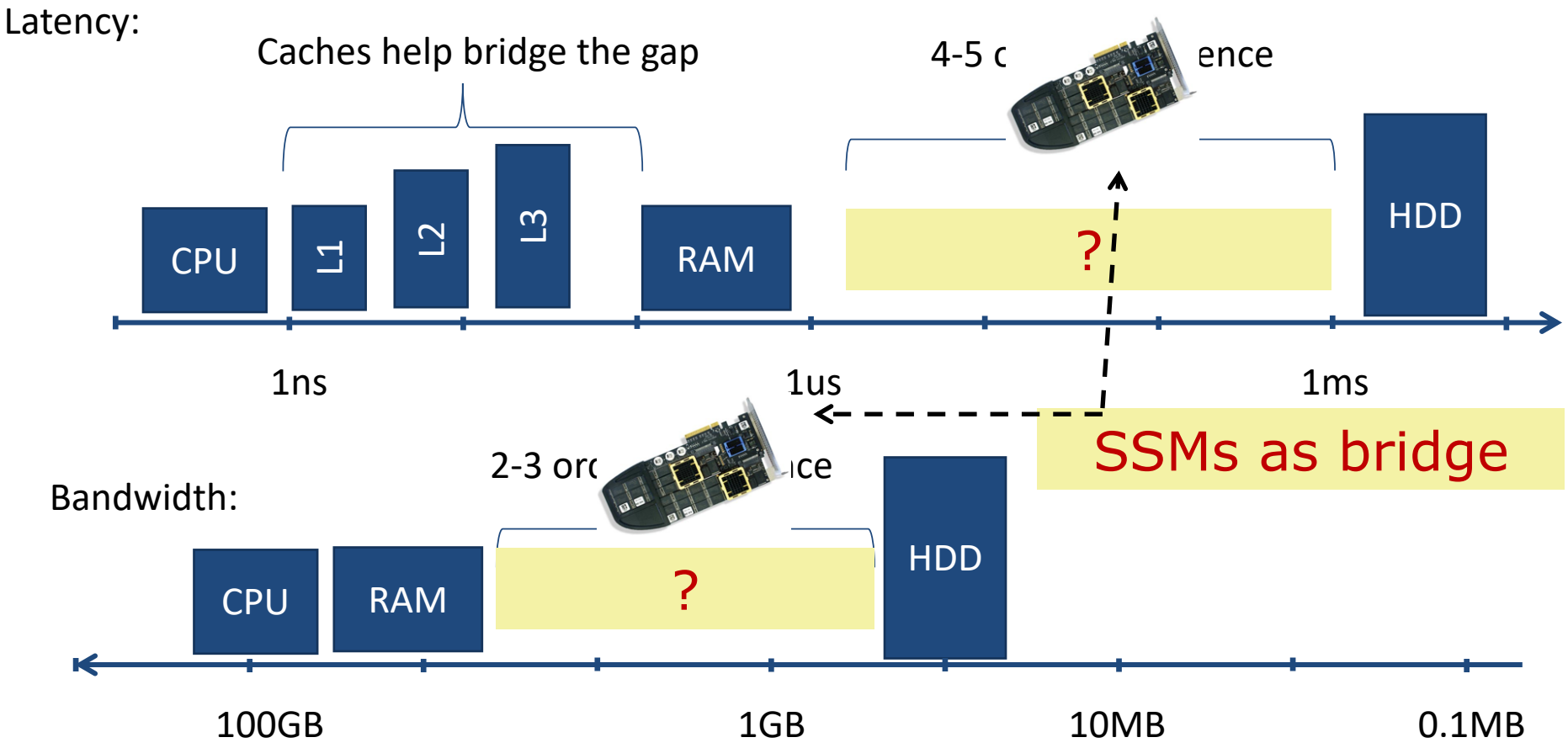
✓ Very efficient random reads



Storage Hierarchy -- 80's



Storage Hierarchy -- Now



"…Disk is Tape, Flash is Disk, RAM Locality is King"

SSM today

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

- Flash

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

- PCM

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

Look similar to DBMS
Neither is a HDD drop-in replacement

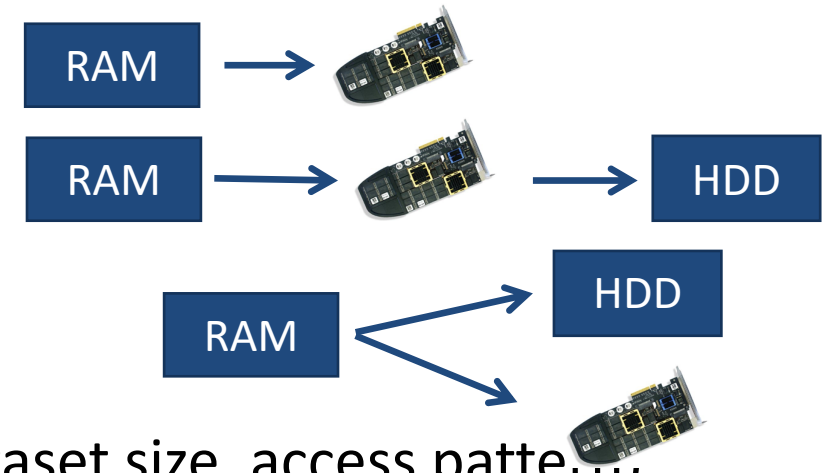
Storage and Data Management

- 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

Need to revisit DBMS design

What to do with flash?

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



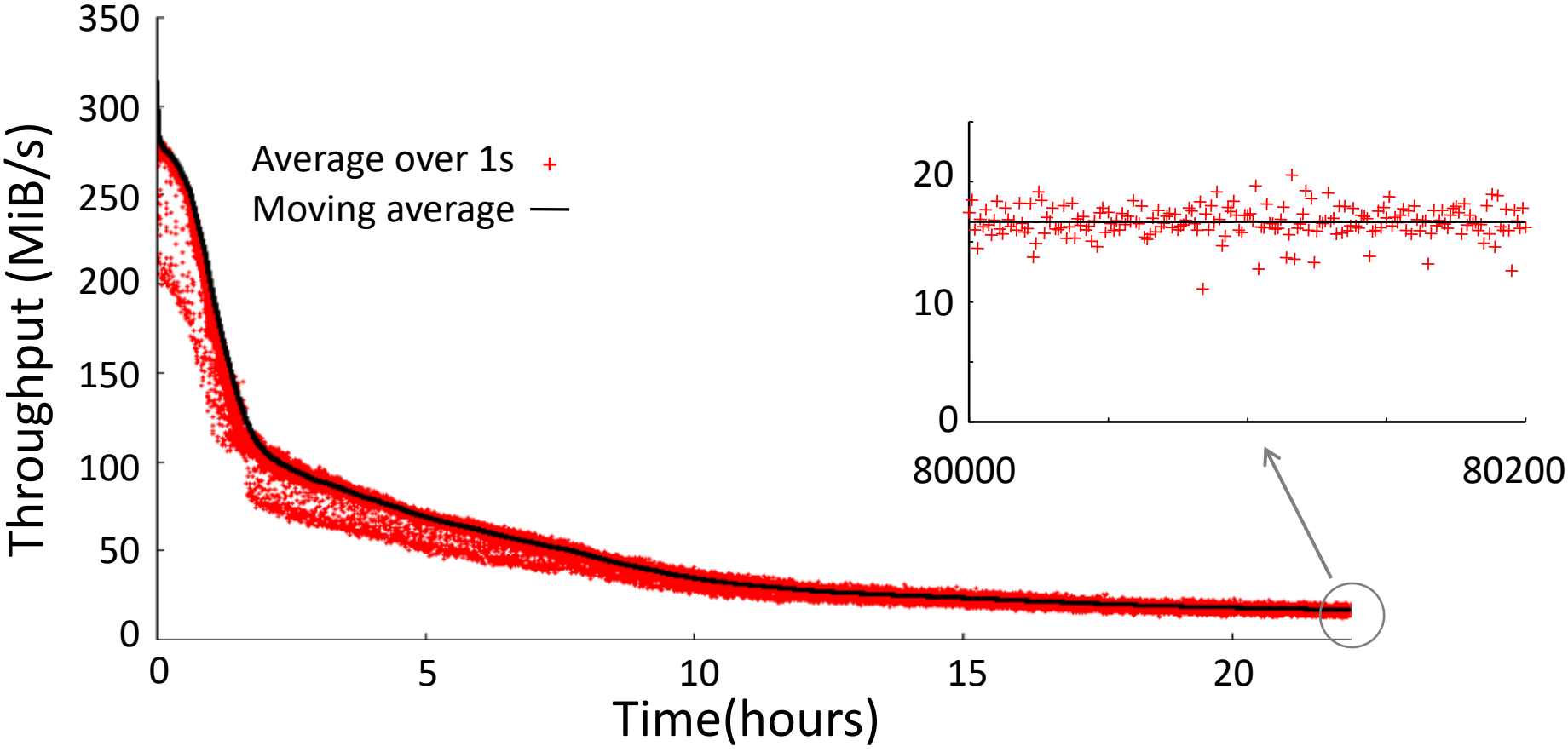
- No “correct” use
  - Depends on workload (dataset size, access patte
  - Future trends: e.g. flash density competitive with HDD

Three example ways to use flash

1) Flash-only OLTP

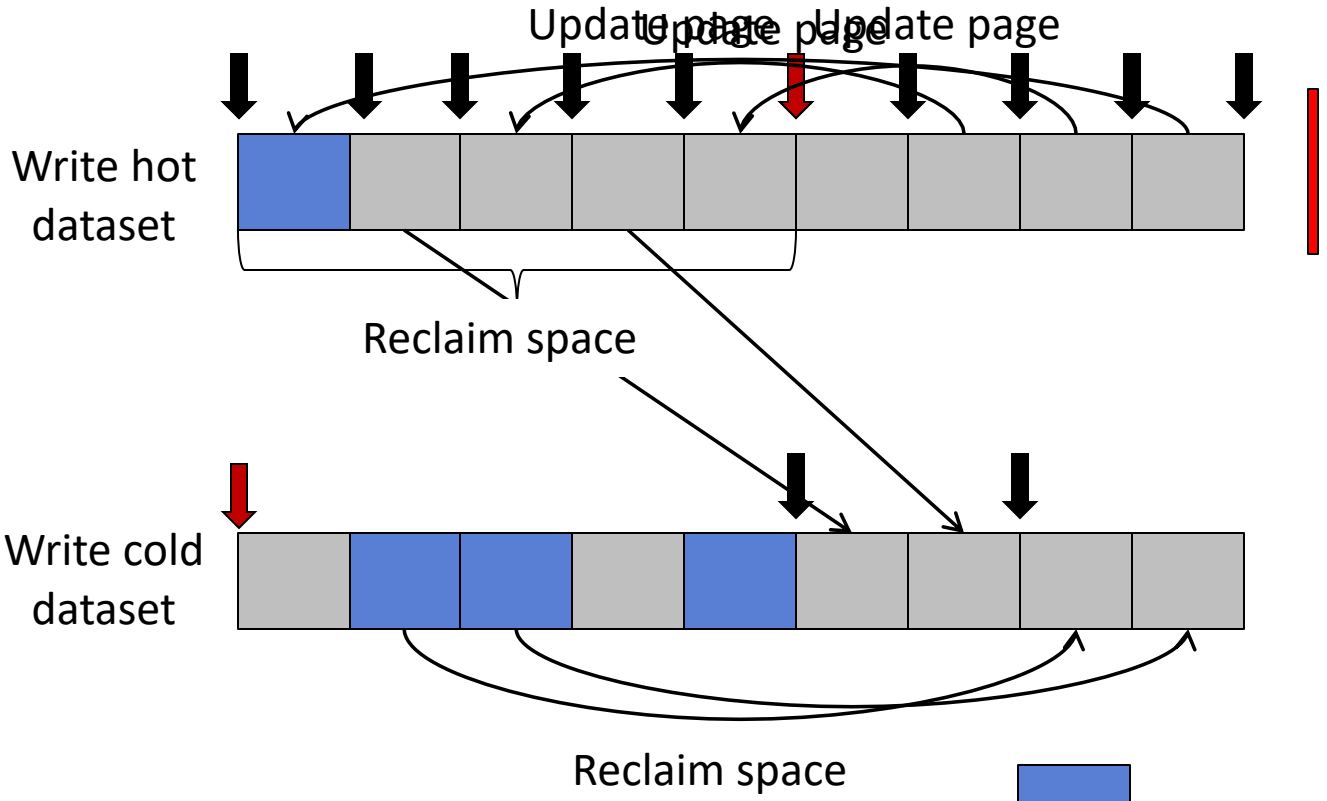
- OLTP I/O dominated by random reads/writes
- Random reads/writes much faster on flash
  - Also, smaller random-to-sequential gap
- Flash-resident workload
  - Usually a couple of flash devices can hold working set
- Should benefit from fast random access of flash

8KiB random writes – Fusion ioDrive



94% performance drop + unpredictability

Append/Pack



Write seq.

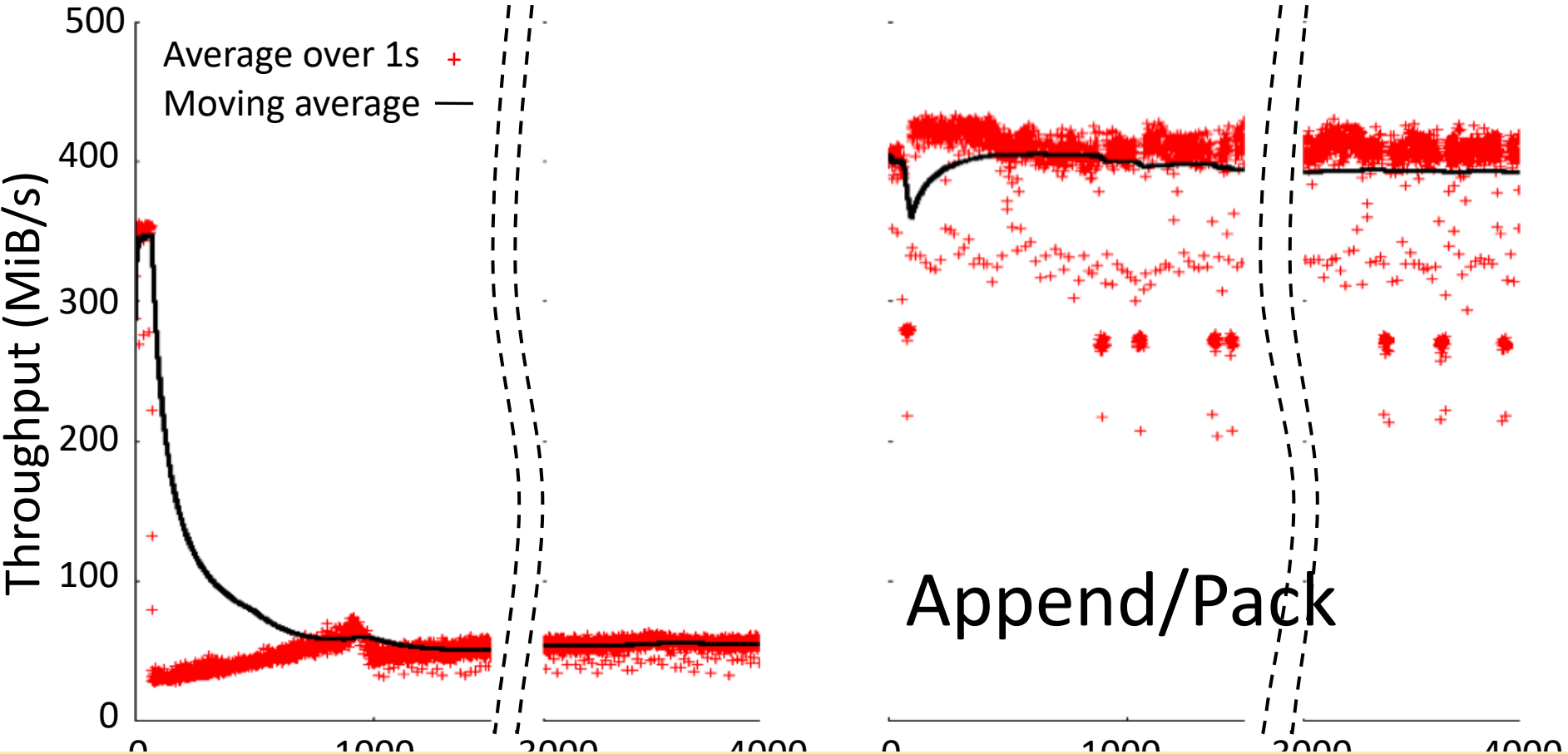
No in-place updates

Filter cold pages

Trade random writes with sequential ops

Append/Pack on Fusion 160GB PCIe

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



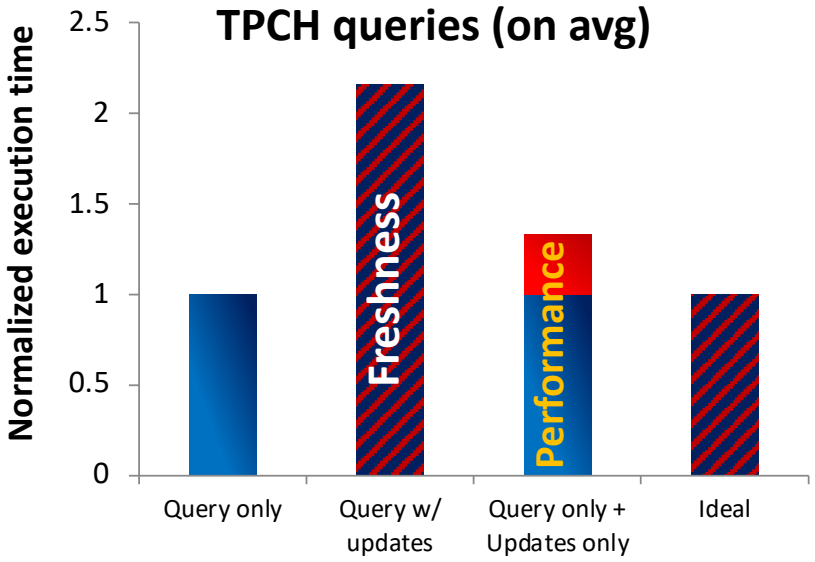
Log-structured writes good for flash-only OLTP

2) Flash-aided Business Intelligence (OLAP)

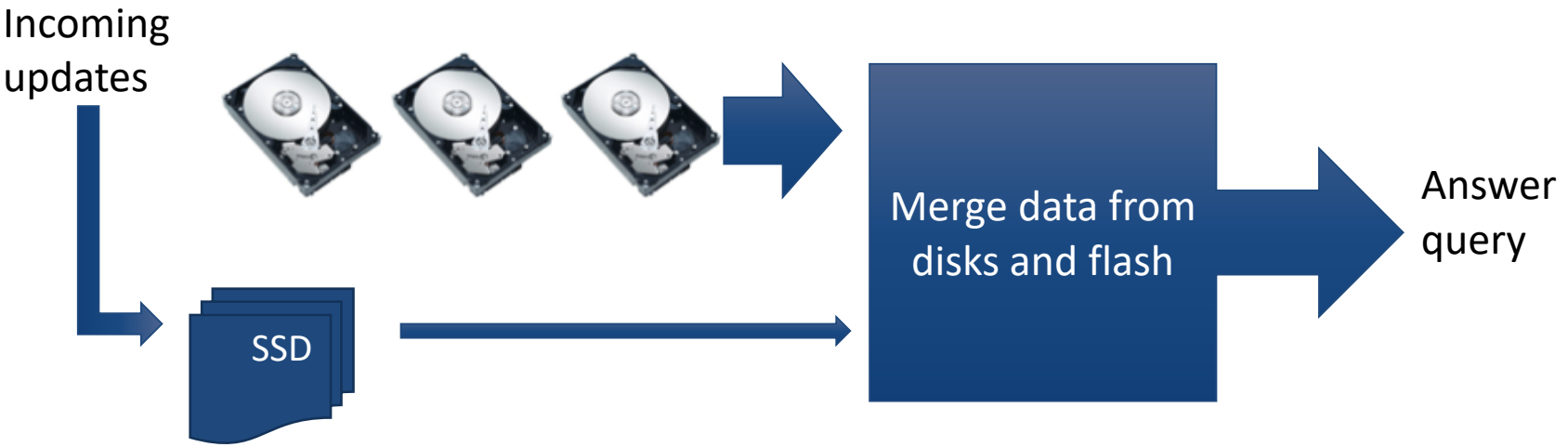
- Data warehouse workload
  - Read-only queries (scans)
  - Scattered updates
  - How to combine *efficiently*?

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

- Ideally, *zero* overhead

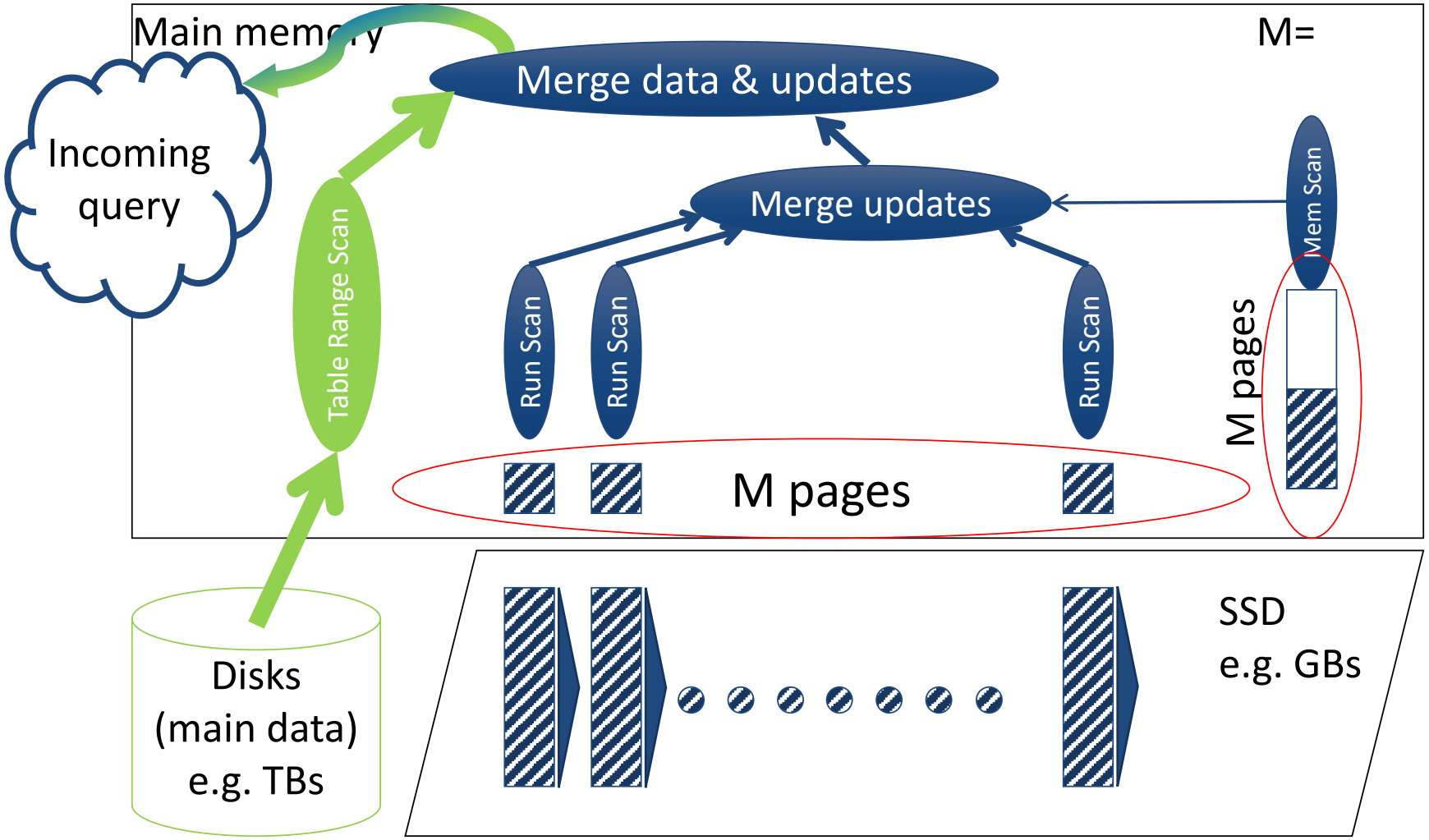


Flash as a (write) cache for analytics



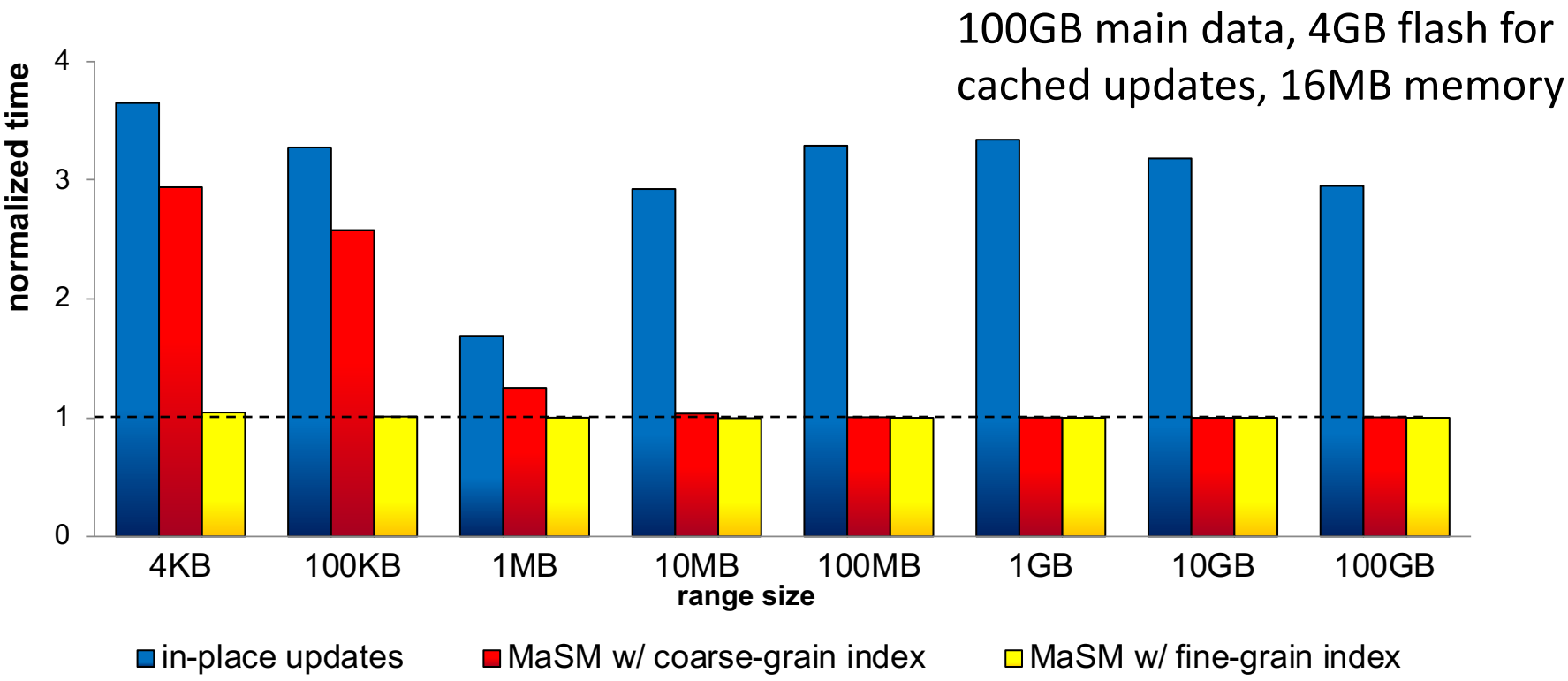
- Buffer updates on Flash instead of memory
  - Flash has *larger capacity* and *smaller price*
- **But:** Flash limitations
  - Access time: Avoid random writes
  - Endurance: Limit/control total # of writes

Materialized Sort-Merge (MaSM)



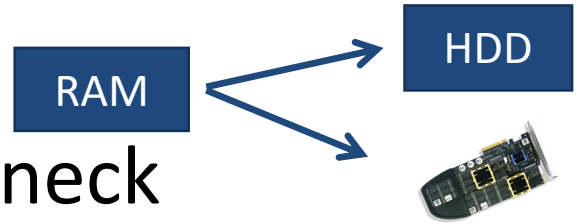
3-point merge with minimum overhead

Seagate Barracuda + Intel X25-E SSD

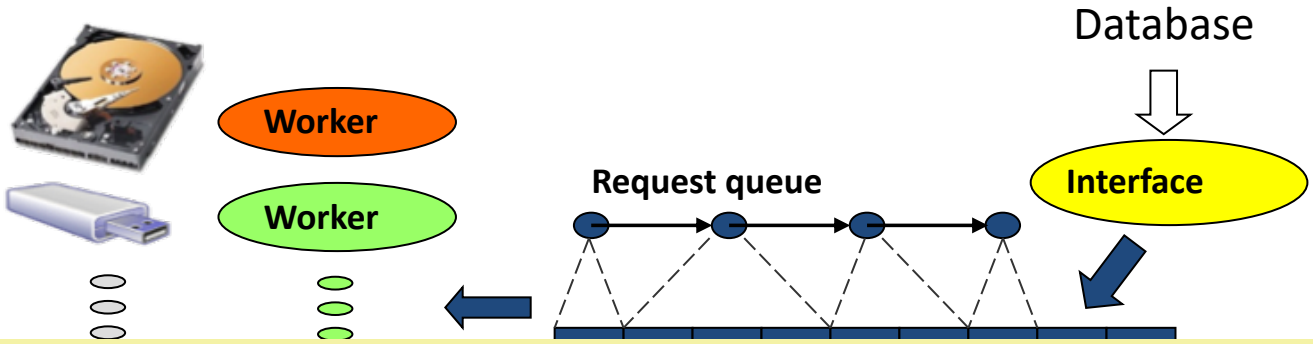


- negligible impact on 10MB or larger scans
- fine-grain index incurs 4% overhead for 4KB ranges (modeling point queries)

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



Faster recovery at lower price

SSD+DBMS: Where and how?

1. SSM as helper of a memory level (DBMS unchanged)
2. Adapt I/O pattern, “small” DBMS changes
3. Change storage mgmt, query optimization

Conclusions

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

  But SW needs to help in building!

- 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