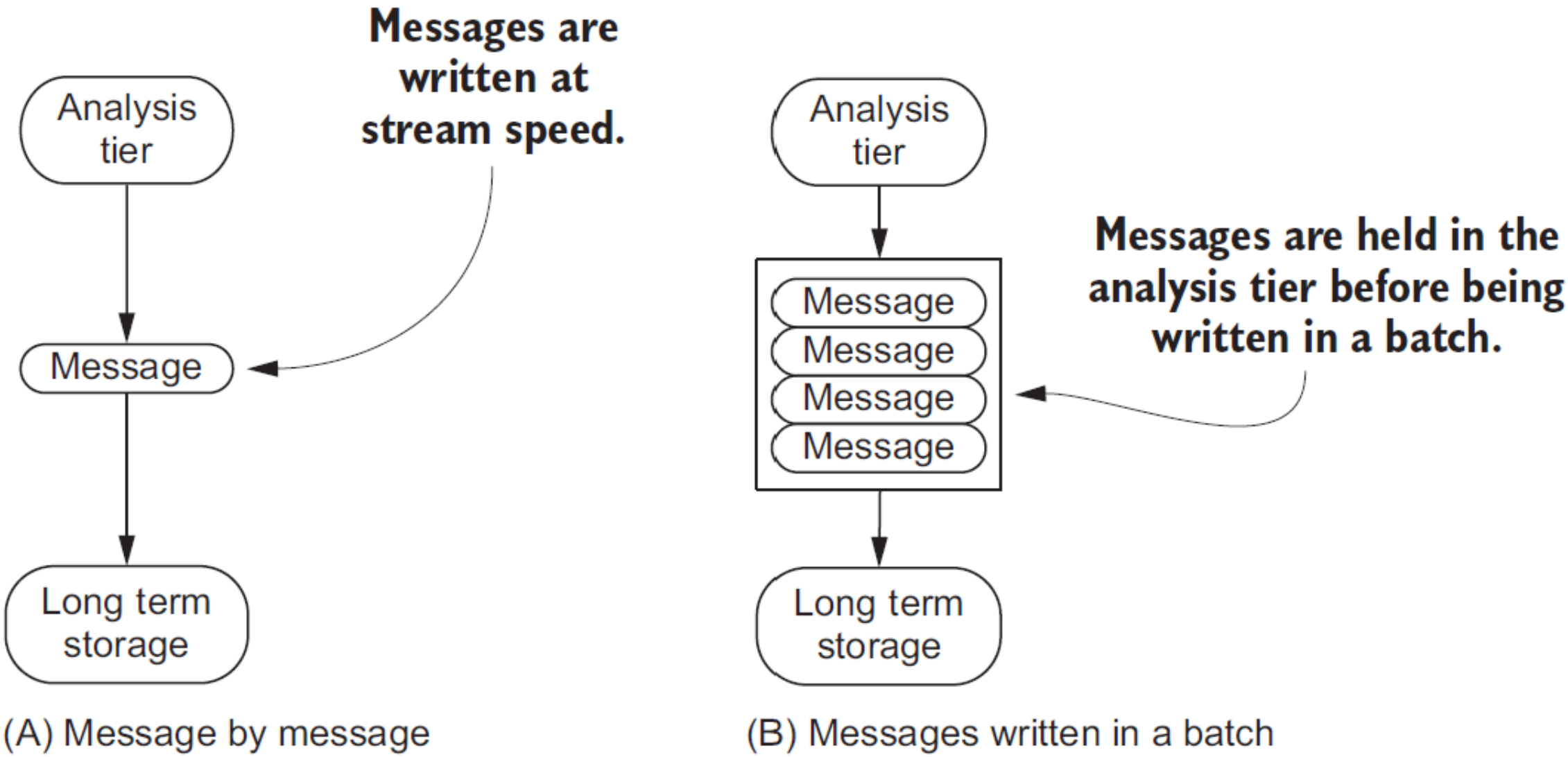
Day 8
Storing the data

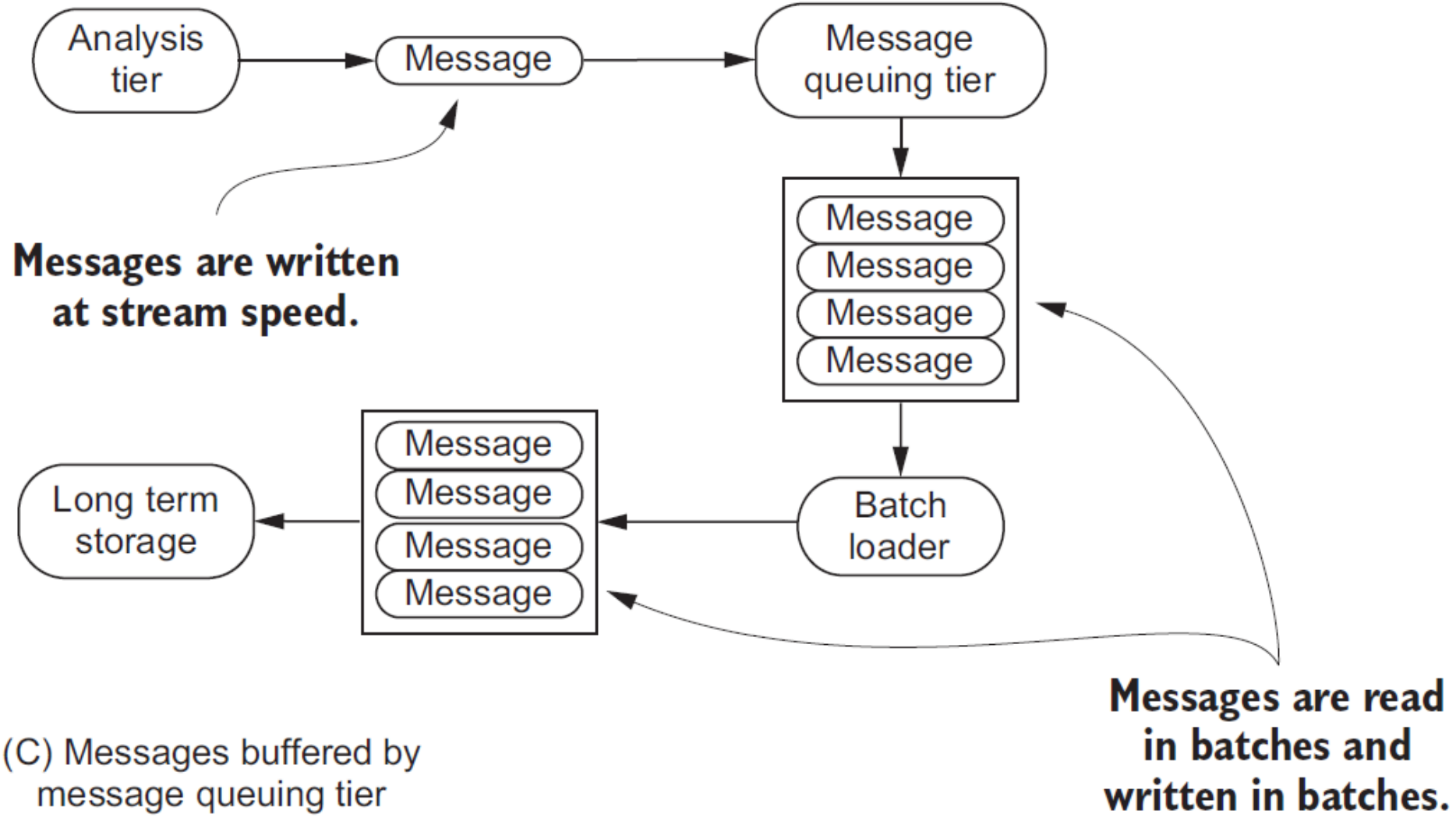
Lecturer: M.S. Le Minh Tan

What to do after analyzing data?

1. Discard the data
2. Push the data back into the streaming platform
3. Store the data for real-time usage
4. Store the data for batch/offline access



Direct writing



(C) Messages buffered by message queuing tier

Indirect writing

Batch

- Batch is a piece of data.
- Tools:
  - Linkedin’s Goblin project
  - Pinterest’s Secor

How about keeping in memory?

- Your machine’s memory is much faster than long-term storage.
- The space is not a problem anymore.
  - 32/64TB servers.
- We may also want to keep the data in long-term storage for backup.

In-memory types

- Embedded in-memory/flash-optimized
- Caching
- In-memory database/In-memory data grid

Embedded in-memory/flash-optimized

- All the system are packed into your software.
    - If the software is up, the database is up. If it is down, the database is down.

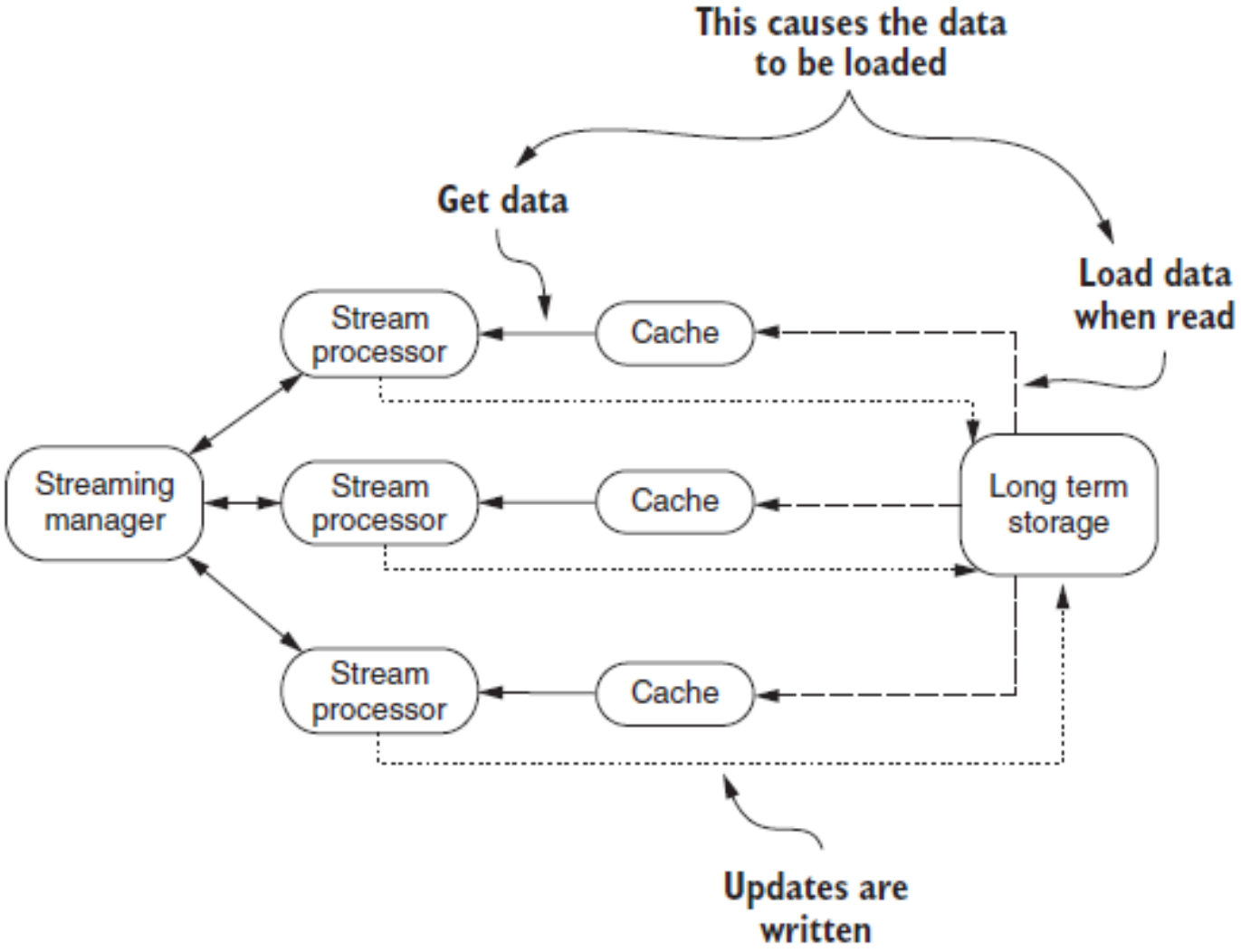
Technologies

- SQLite: Local storage
- RocksDB: Key-value fast storage
- LevelDB: Ordered key-value mapping
- LMDB: For read-heavy workloads
- Perset: Embeded database system for Java & .NET
- IndexedDB: Database stored inside browser

Caching system

- Store data in memory.
- No option to store outside of RAM.
- 5 stategies:
  - Read-through
  - Refresh-ahead
  - Write-through
  - Write-around
  - Write-back

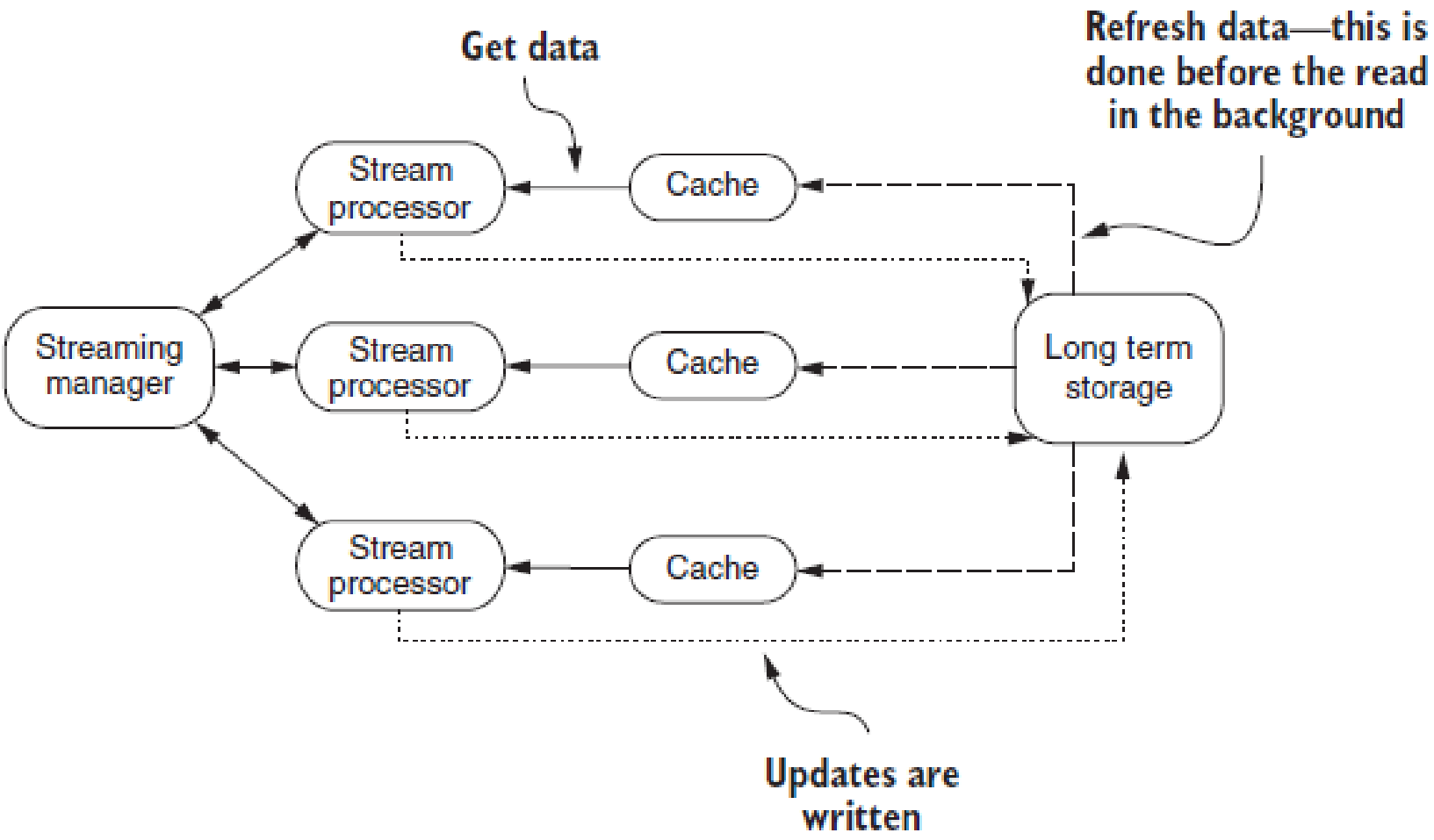
#1. Read-through



#2. Refresh-ahead

- Frequently refresh the cached accessed data.
- Unless the data is initially requested, it will be always from cache.
- May not be suitable for large caching.

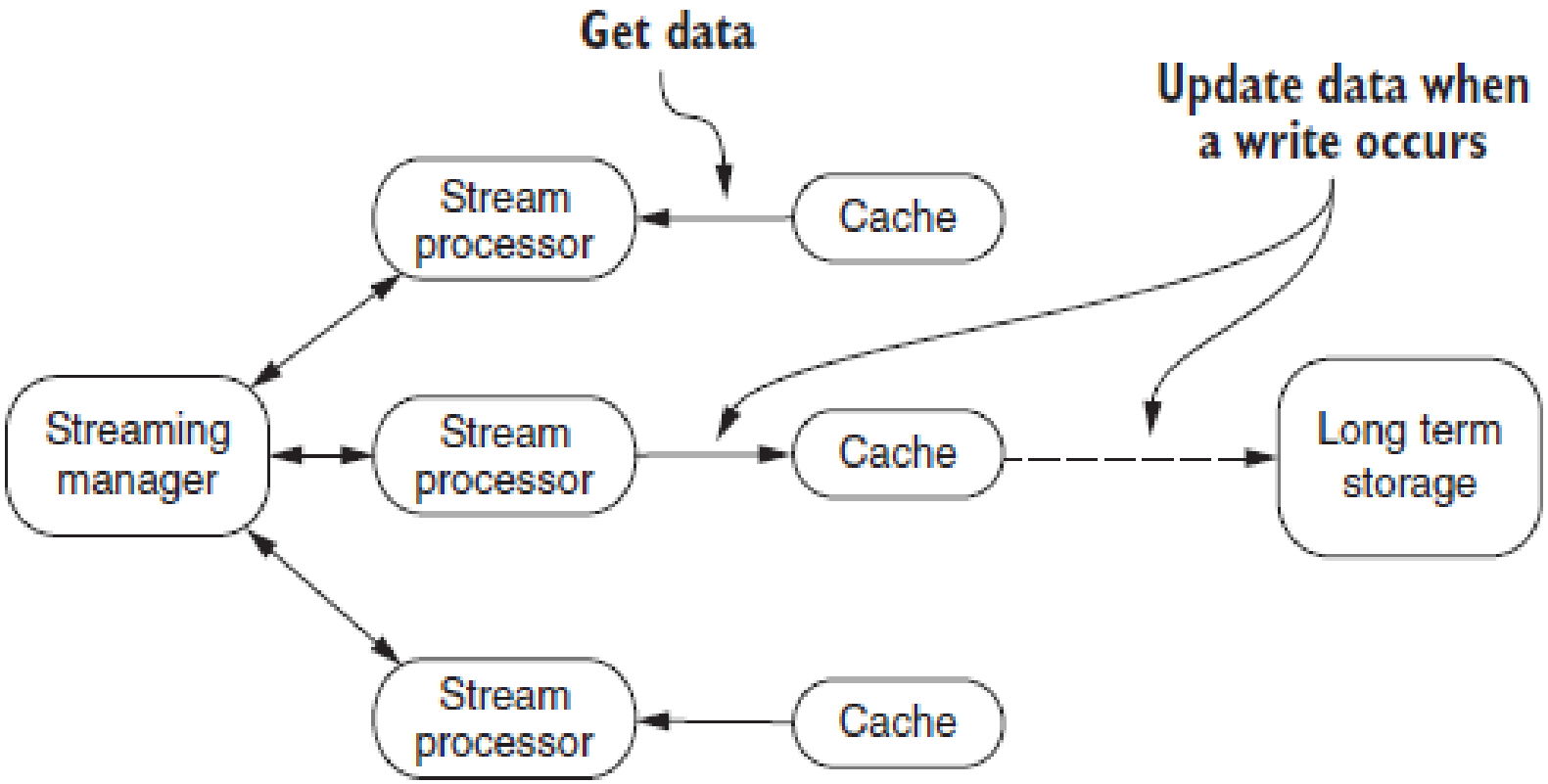
#2. Refresh-ahead



#3. Write-through

- The caching system will write the data.
- Cache is updated while writing data.
- However,
  - the writing latency increase.
  - the cache may not know if the data writing is successful or not.

#3. Write-through



#4. Write-around

- Updates the data to storage through a more complicated process.
- A update receiver at storage sends data to cache.
- The cache is passive.

#5. Write-back/write-behind

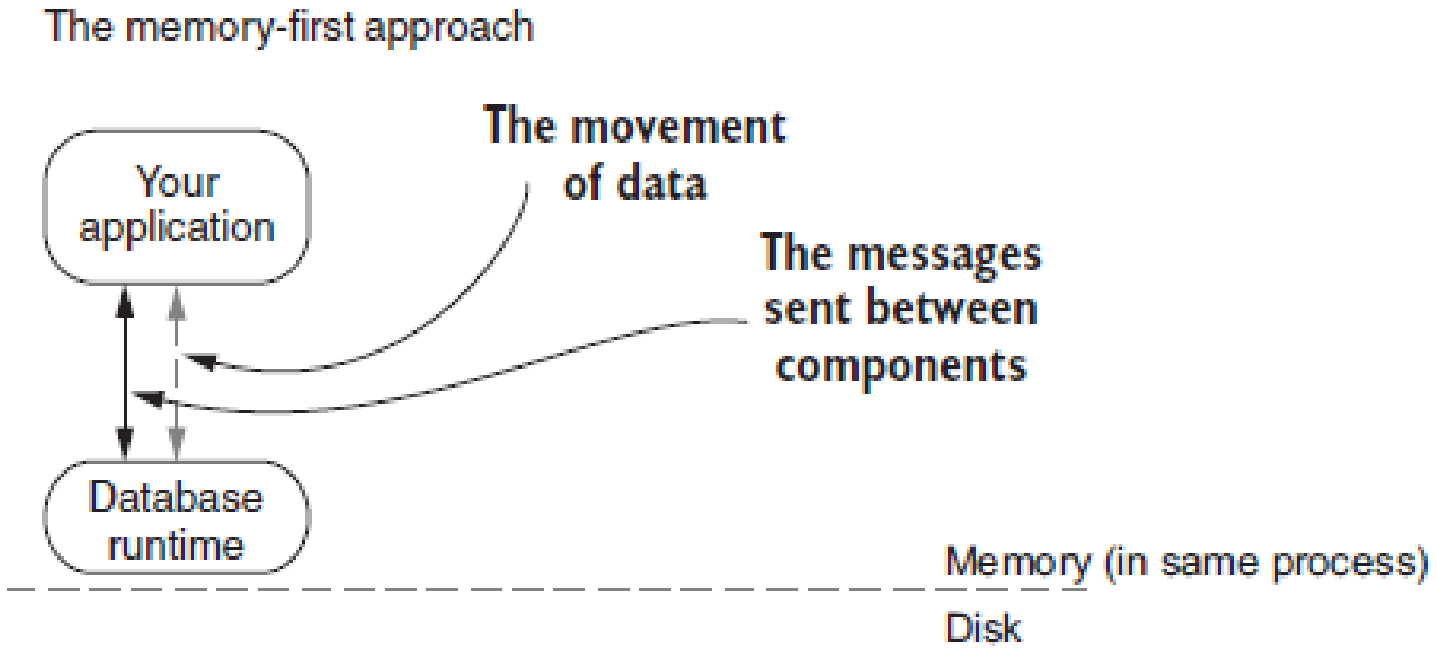
- An upgrade of write-through.
- While updating the cache, storage receives new data in the background.
  - Reduce writing latency.
  - Increase chances of data loss.

Technologies

- Memcached
- EHCache
- Hazelcast
- Redis

In-memory database (IMDB)

- Sometimes called
  - In-memory database management system
  - In-memory data grids (IMDG), but with minor difference



Technologies

- SingleStore (MemSQL)
- VoltDB
- Aerospike
- Apache Geode
- Couchbase
- Apache Ignite
- Hazelcast
- Inifispan

Exercises: Count-min sketch (CMS)

- Usage: Estimate the frequency of data with lowest memory.
- Example: 1, 2, 2, 3, 4, 4, 4
  - The frequency of 2 is 2.
  - The frequency of 4 is 3.

Simple approach

- Declare *counts*: a list of 0s.
- Iterate each data:
  - counts[data]++

Count-min approach

- We define d hash functions.
- Next, create a $d \times w$ matrix (hash table) A.
- When count data M, increase $A[i, d_i(M)]$ by 1.
- When getting result of N, get the $\min(A[i, d_i(N)])$.

We want a small hash table

- Choose low-bit hash functions d.
- To increase the accuracy, try using more hash functions.

Try these

- CRC-8
- Sum8
- Xor8
- https://crccalc.com/
- https://en.wikipedia.org/wiki/List_of_hash_functions