Week 12

Advanced FS: Log-Structured Designs (Part 2)

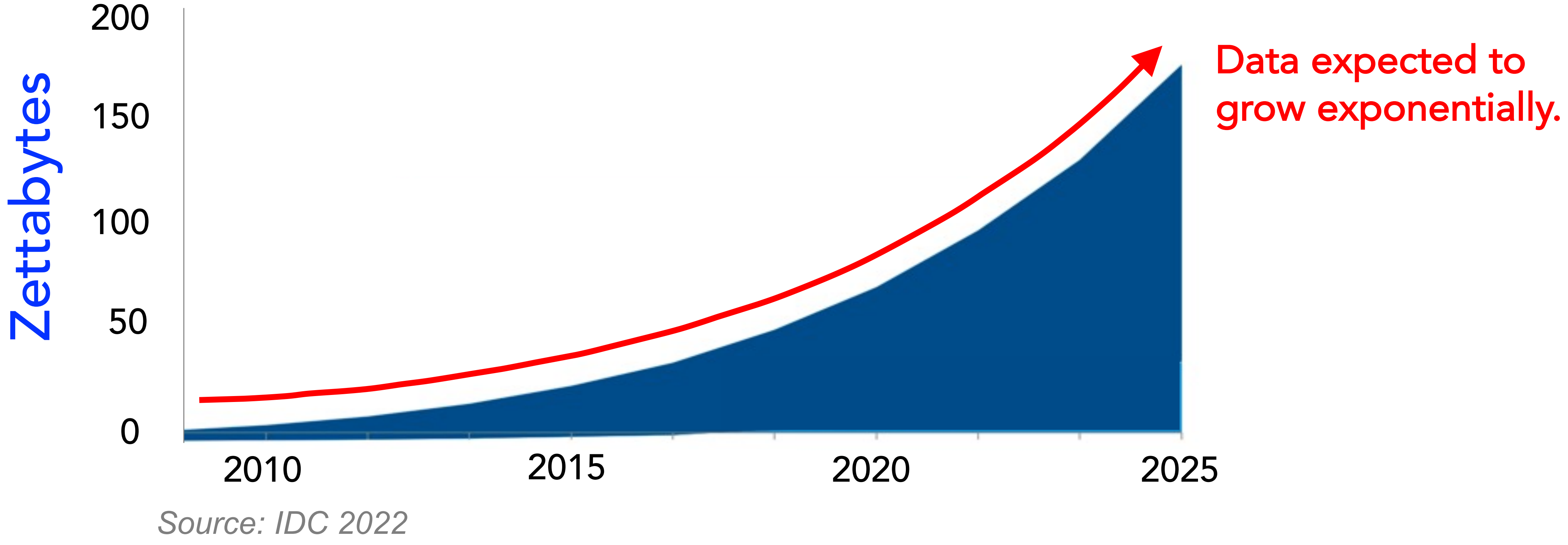
Oana Balmau

March 23, 2023

Log-structured key-value stores

We Produce Huge Amounts of Data

Data needs and will need to be served from persistent storage.



Past Workloads


Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

In other projects

Article Talk Read Edit View history Search Wikipedia

Sydney

From Wikipedia, the free encyclopedia

Coordinates: 33°51′54″S 151°12′34″E

This article is about the Australian metropolis. For the local government area, see City of Sydney. For other uses, see Sydney (disambiguation).

Sydney (/ˈsɪdni/ listen) *SID-nee*)[7] is the state capital of New South Wales and the most populous city in Australia and Oceania.[8] Located on Australia's east coast, the metropolis surrounds Port Jackson and extends about 70 km (43.5 mi) on its periphery towards the Blue Mountains to the west, Hawkesbury to the north, the Royal National Park to the south and Macarthur to the south-west.[9] Sydney is made up of 658 suburbs, 40 local government areas and 15 contiguous regions. Residents of the city are known as "Sydneysiders".[10] As of June 2017, Sydney's estimated metropolitan population was 5,230,330[11] and is home to approximately 65% of the state's population.[12]

Indigenous Australians have inhabited the Sydney area for at least 30,000 years, and thousands of engravings remain throughout the region, making it one of the richest in Australia in terms of Aboriginal archaeological sites. During his first Pacific voyage in 1770, Lieutenant James Cook and his crew became the first Europeans to chart the eastern coast of Australia, making landfall at Botany Bay and inspiring British interest in the area. In 1788, the *First Fleet* of convicts, led by Arthur Phillip, founded Sydney as a British penal colony, the first European settlement in Australia. Phillip named the city *Sydney* in recognition of Thomas Townshend, 1st Viscount Sydney.[13] Penal transportation to New South Wales ended soon after Sydney was

Sydney
New South Wales

Port Jackson with its Sydney Opera House and Sydney Harbour Bridge landmarks

Example: Wikipedia

Past Workloads


Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

In other projects

Sydney

From Wikipedia, the free encyclopedia

Coordinates: 33°51′54″S 151°12′34″E

This article is about the Australian metropolis. For the local government area, see City of Sydney. For other uses, see Sydney (disambiguation).

Sydney (/ˈsɪdni/ listen) *SID-nee*)[7] is the state capital of New South Wales and the most populous city in Australia and Oceania.[8] Located on Australia's east coast, the metropolis surrounds Port Jackson and extends about 70 km (43.5 mi) on its periphery towards the Blue Mountains to the west, Hawkesbury to the north, the Royal National Park to the south and Macarthur to the south-west.[9] Sydney is made up of 658 suburbs, 40 local government areas and 15 contiguous regions. Residents of the city are known as "Sydneysiders".[10] As of June 2017, Sydney's estimated metropolitan population was 5,230,330[11] and is home to approximately 65% of the state's population.[12]

Indigenous Australians have inhabited the Sydney area for at least 30,000 years, and thousands of engravings remain throughout the region, making it one of the richest in Australia in terms of Aboriginal archaeological sites. During his first Pacific voyage in 1770, Lieutenant James Cook and his crew became the first Europeans to chart the eastern coast of Australia, making landfall at Botany Bay and inspiring British interest in the area. In 1788, the *First Fleet* of convicts, led by Arthur Phillip, founded Sydney as a British penal colony, the first European settlement in Australia. Phillip named the city *Sydney* in recognition of Thomas Townshend, 1st Viscount Sydney.[13] Penal transportation to New South Wales ended soon after Sydney was

Sydney
New South Wales

Port Jackson with its Sydney Opera House and Sydney Harbour Bridge landmarks

Example: Wikipedia

Read mostly.

read:write ratio ~ 1000:1

Content rarely updated.

Static pages.

Future Workloads

Example: Internet of Things

Future Workloads

Example: Internet of Things

Mixed workload.

read:write ratio ~ 1:1

Reads: *read large volumes of sensor data for control and analytics.*

Writes*: new sensor data continuously recorded.*

Key-Value Stores (KVs)

Crucial component in systems with large amounts of diverse data.

Similar to a dictionary.

Key-value pairs
- multiple **sizes.**
- multiple **data types.**

Simple API: `Put()`, `Get()`, `Scan()`.

Key-Value Stores (KVs): the Key to Big, Unstructured Data

Crucial component in systems with large amounts of diverse, unstructured data.

RocksDB
redis
Google Cloud Bigtable
LEVELDB
mongoDB
cassandra
MEMCACHED
amazon DynamoDB

KV Stores

	Single machine	Distributed
Memory + Disk		
In-memory		

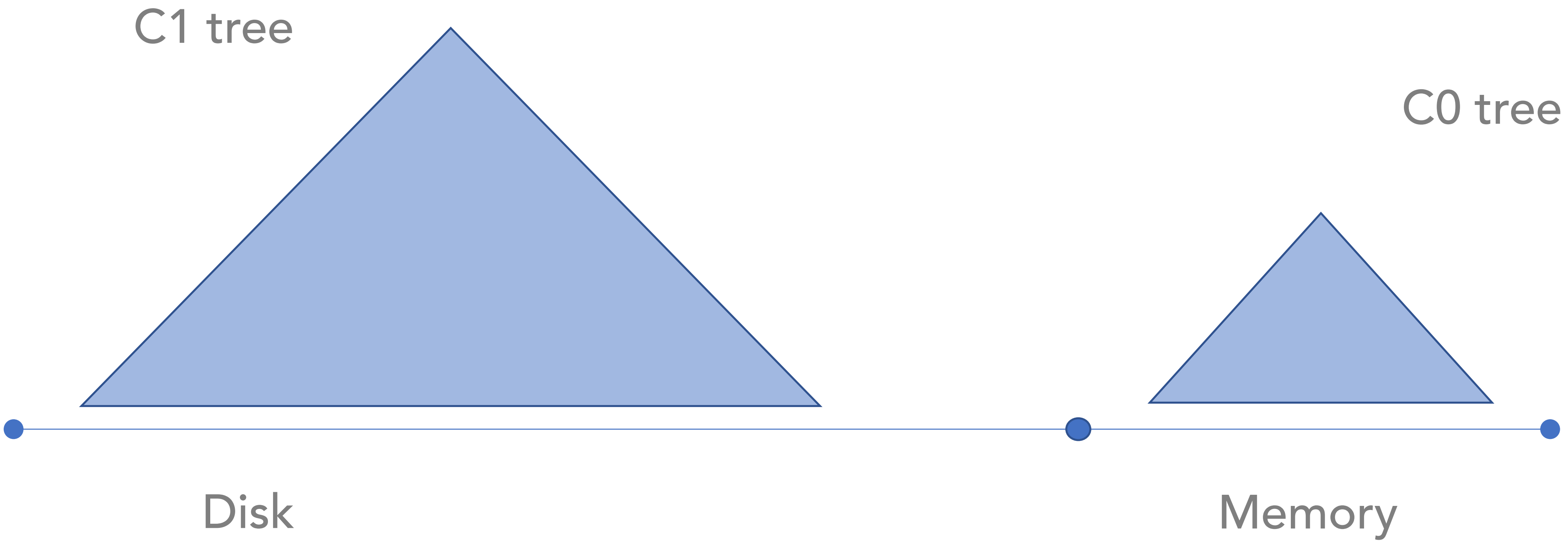
KV Stores

	Single machine	Distributed
Memory + Disk	RocksDB LEVELDB	cassandra mongoDB Google Cloud Bigtable
In-memory	MEMCACHED	redis amazon DynamoDB

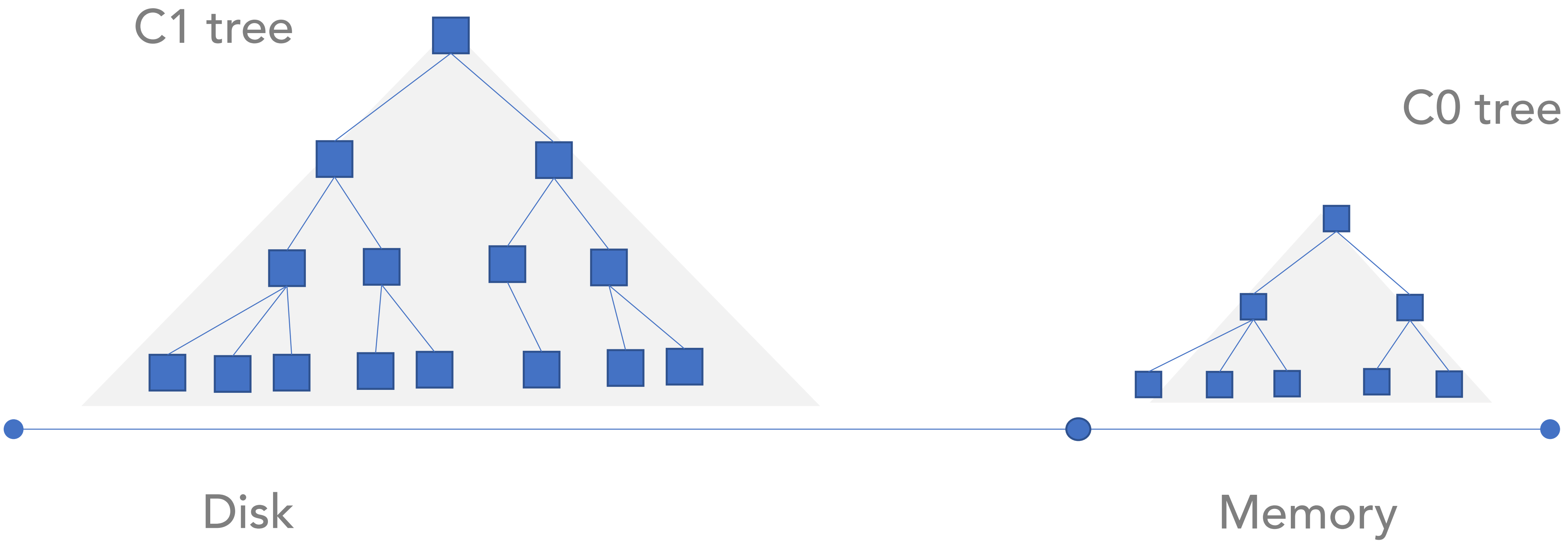
Log-structured KV Stores

	Single machine	Distributed
Memory + Disk	RocksDB LEVELDB	cassandra Google Cloud Bigtable mongoDB
In-memory	MEMCACHED	redis amazon DynamoDB

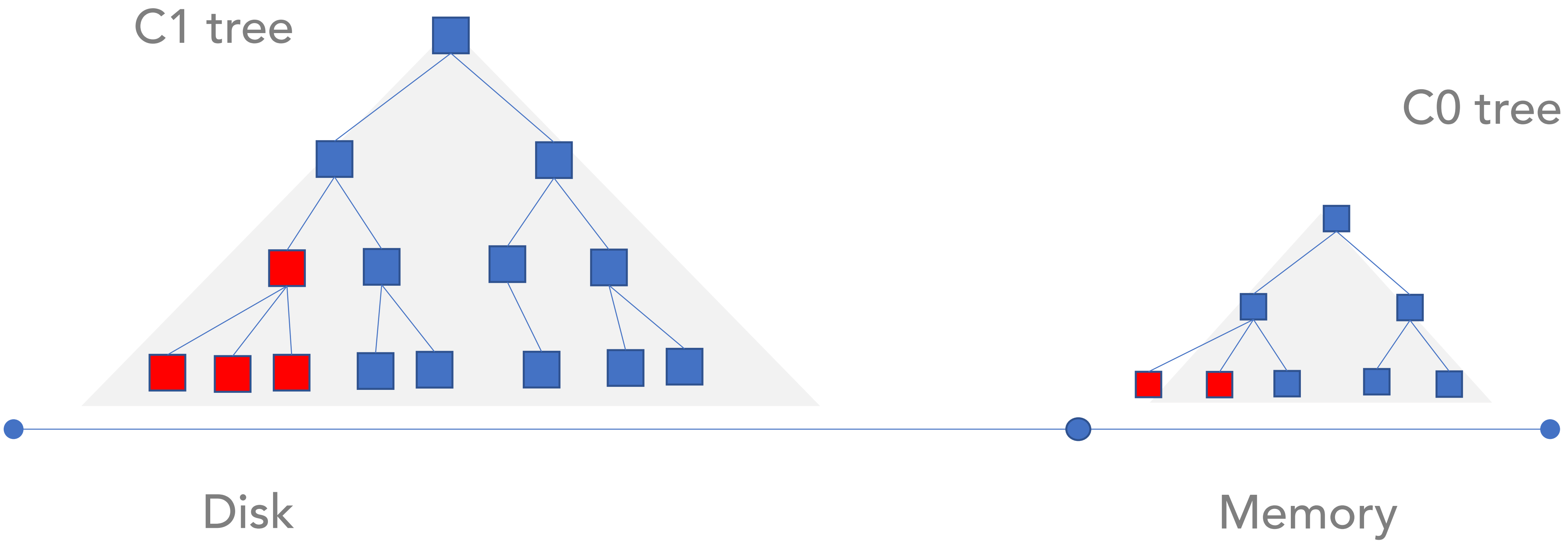
Log-structured merge tree (O'Neil et al,, 1996)



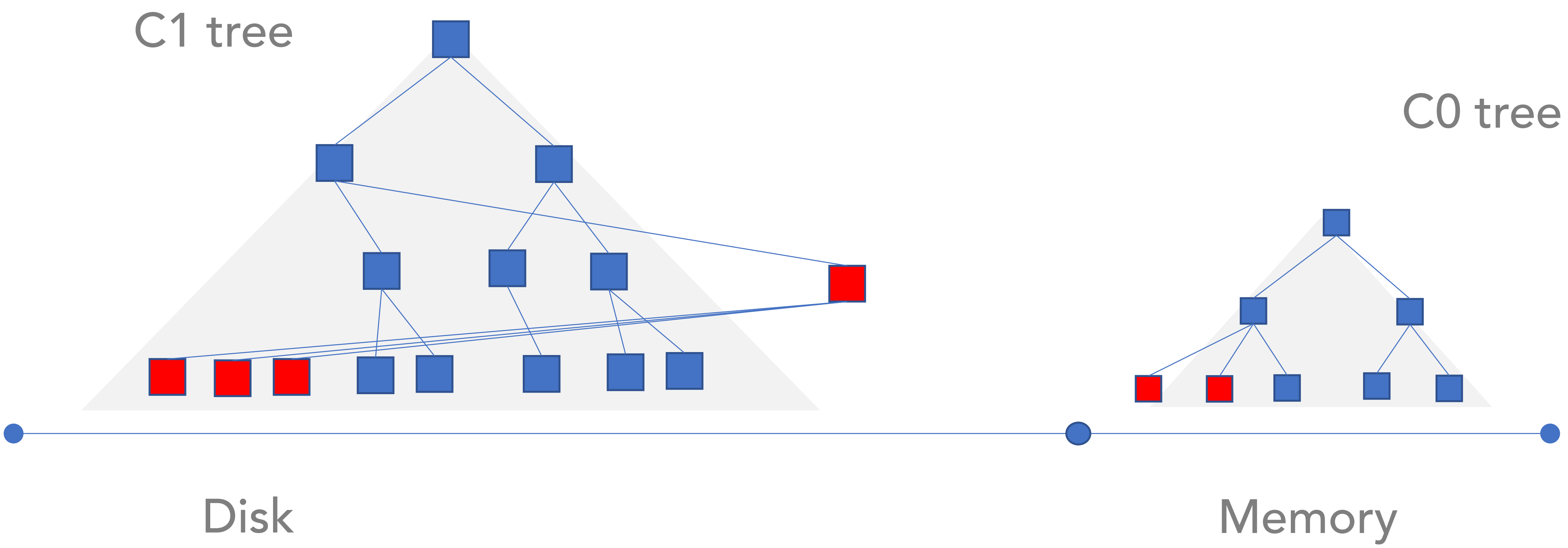
Log-structured merge tree (O'Neil et al,, 1996)



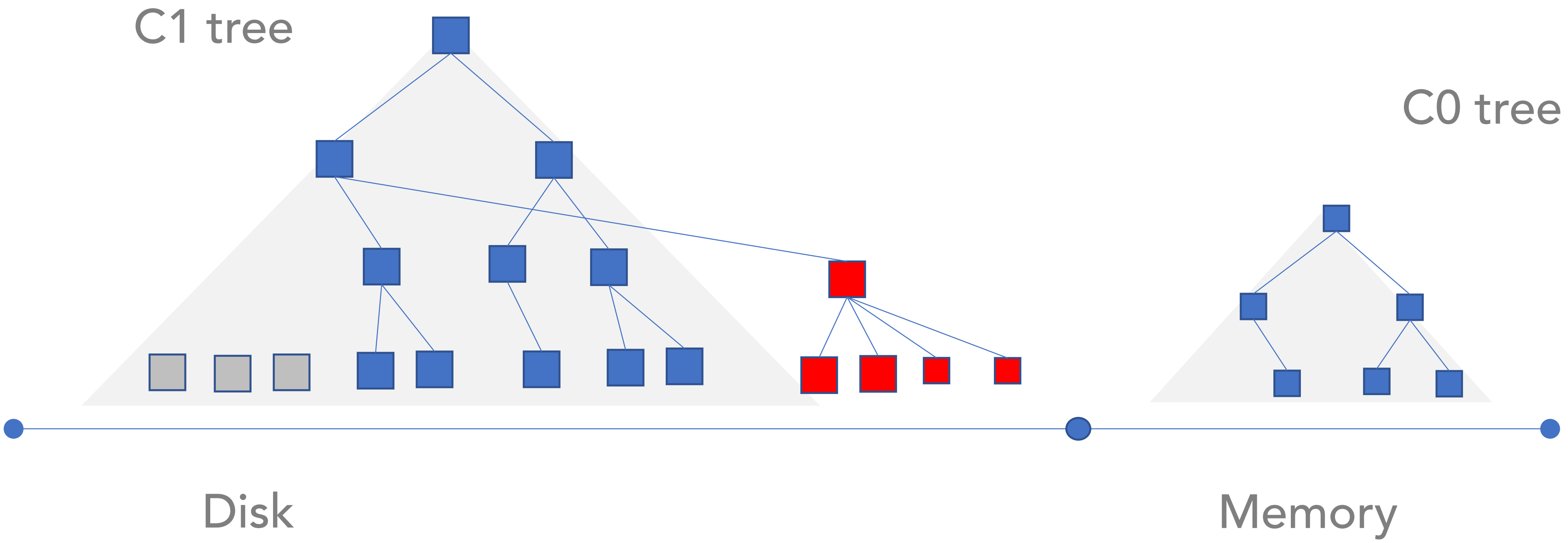
Log-structured merge tree (O’Neil et al,, 1996)



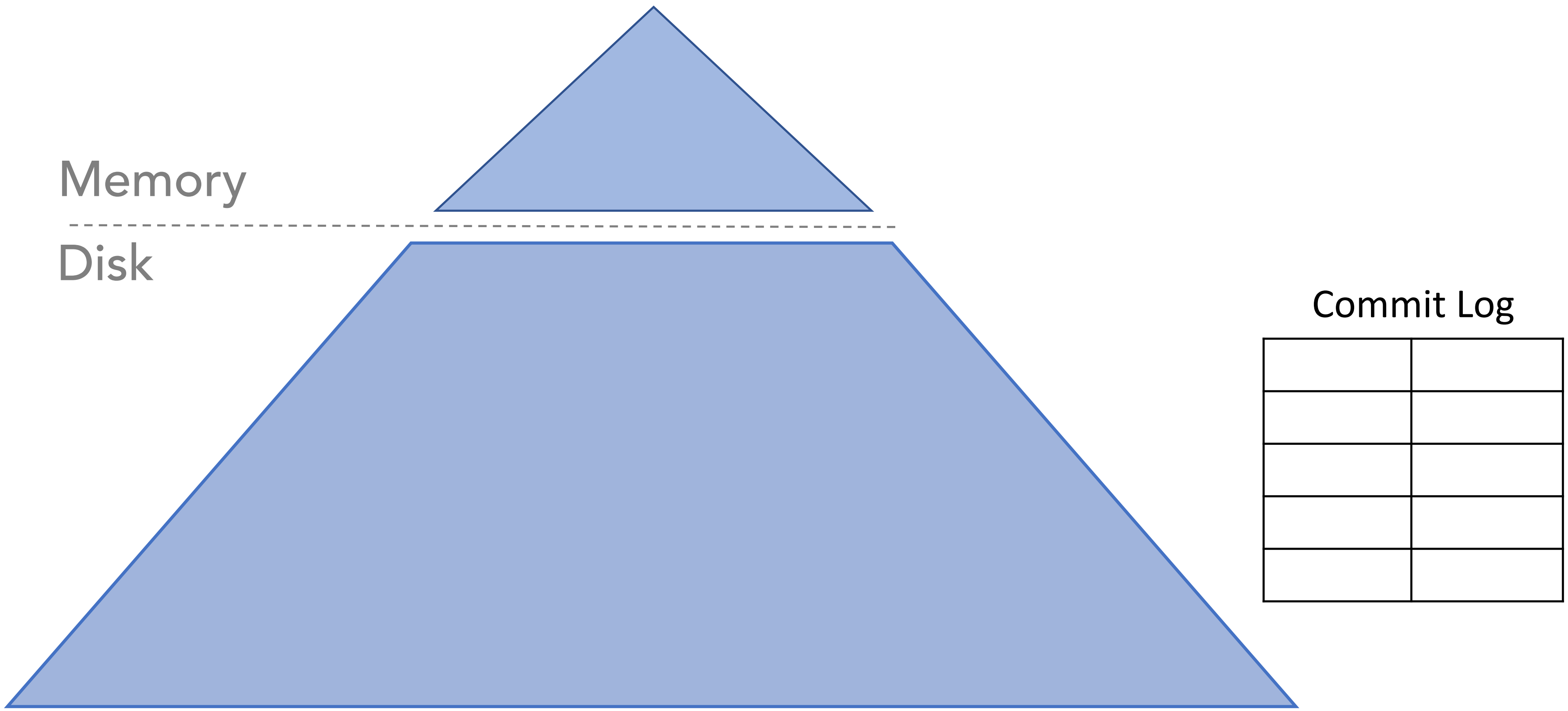
Log-structured merge tree (O'Neil et al,, 1996)



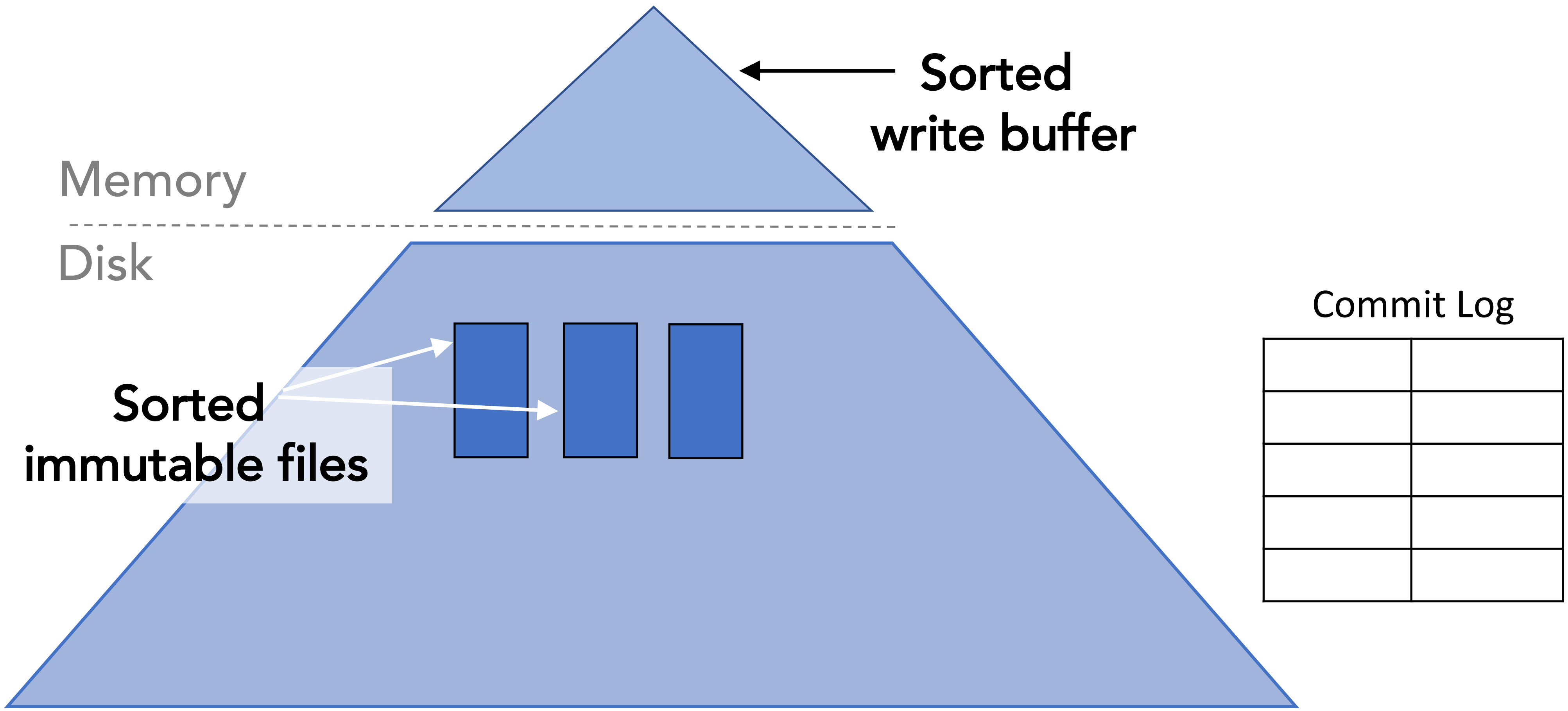
Log-structured merge tree (O'Neil et al,, 1996)



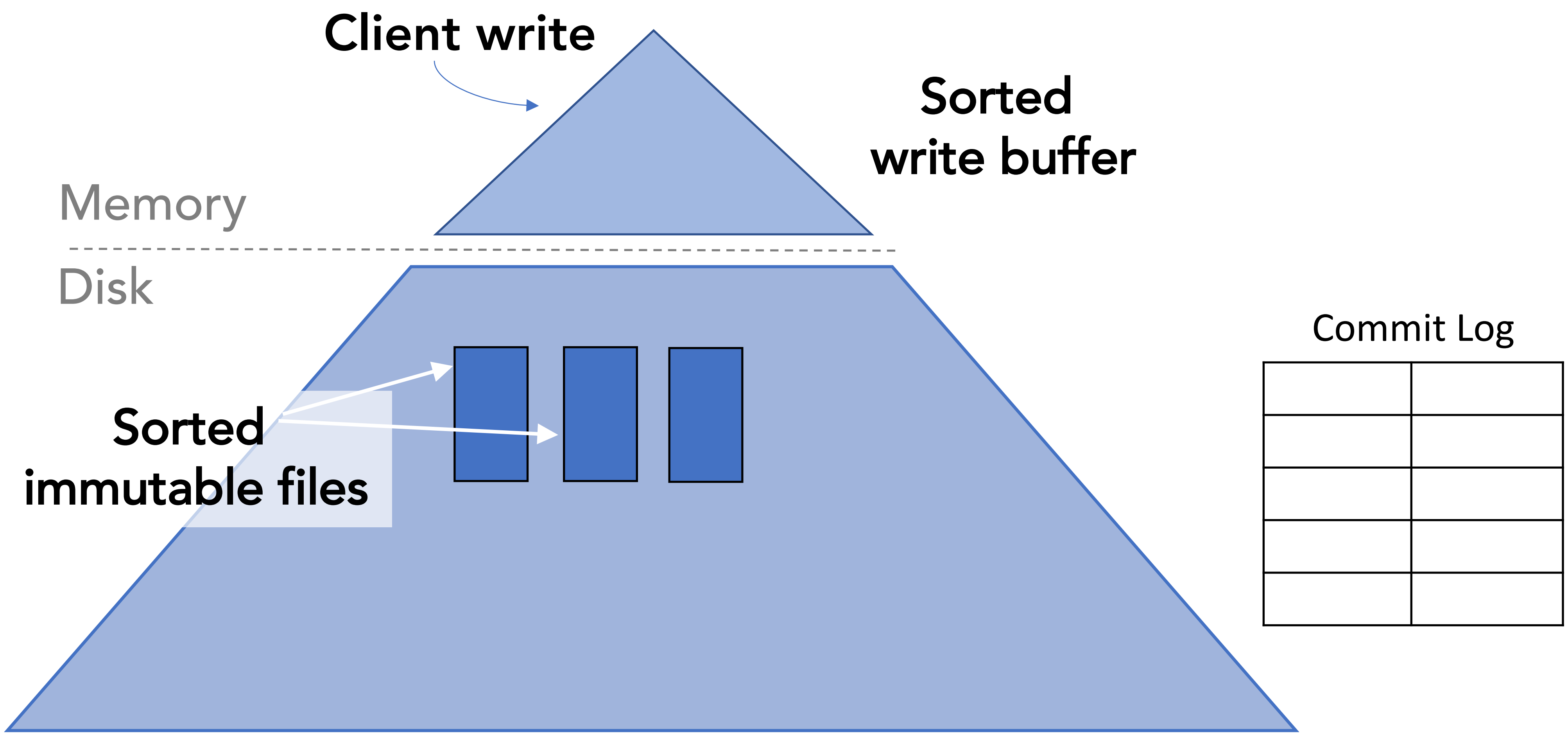
LSM KVs Internals



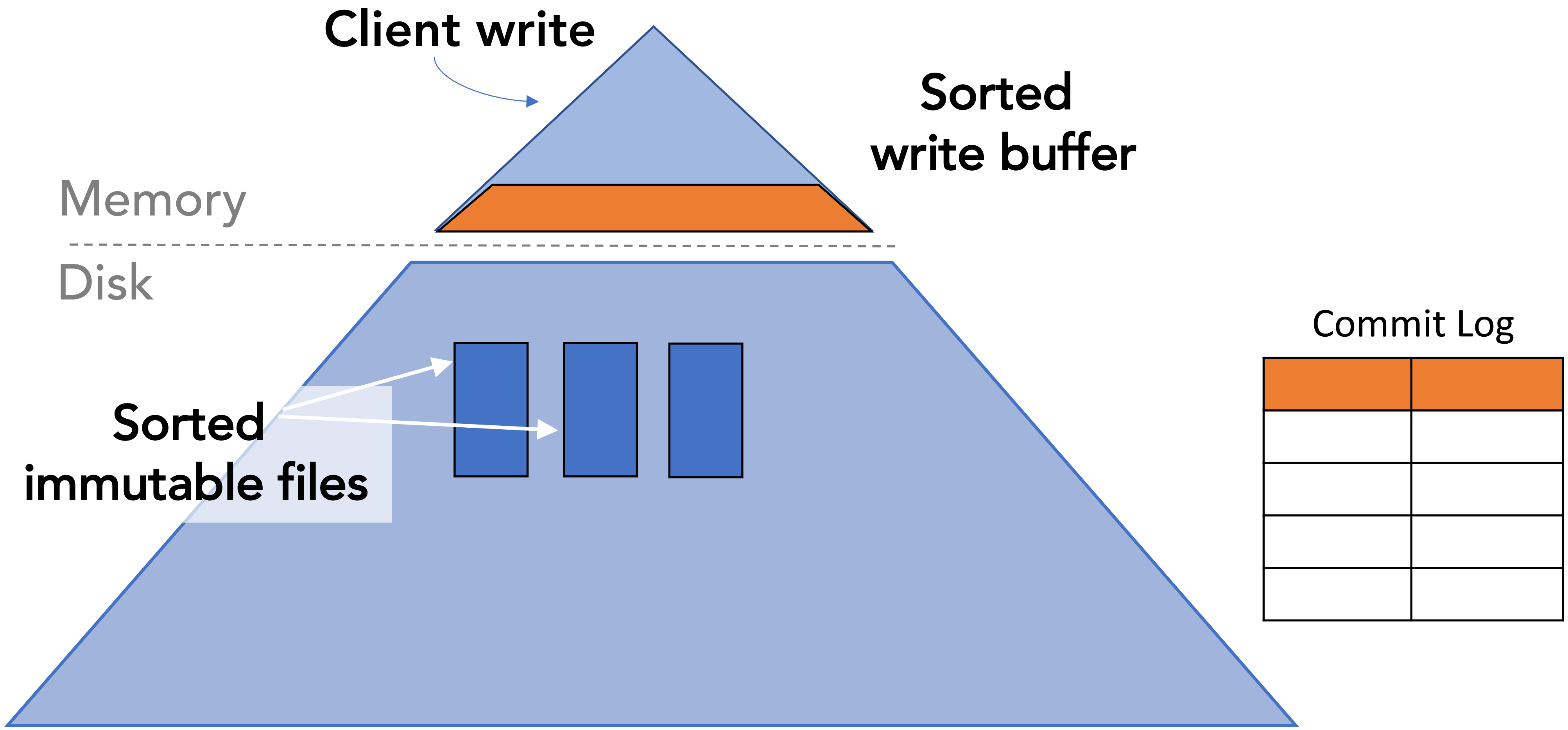
LSM KVs Internals



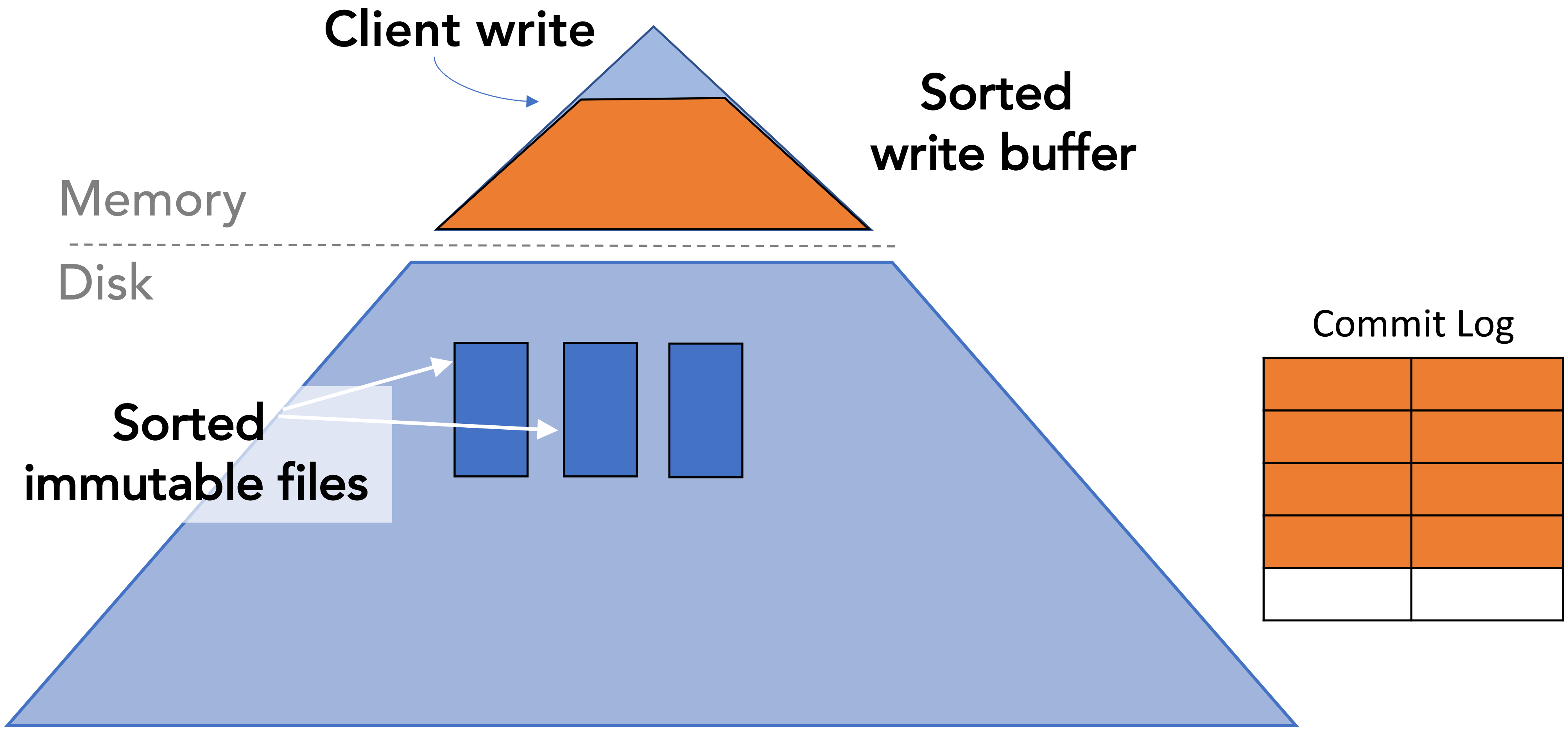
LSM KVs Internals



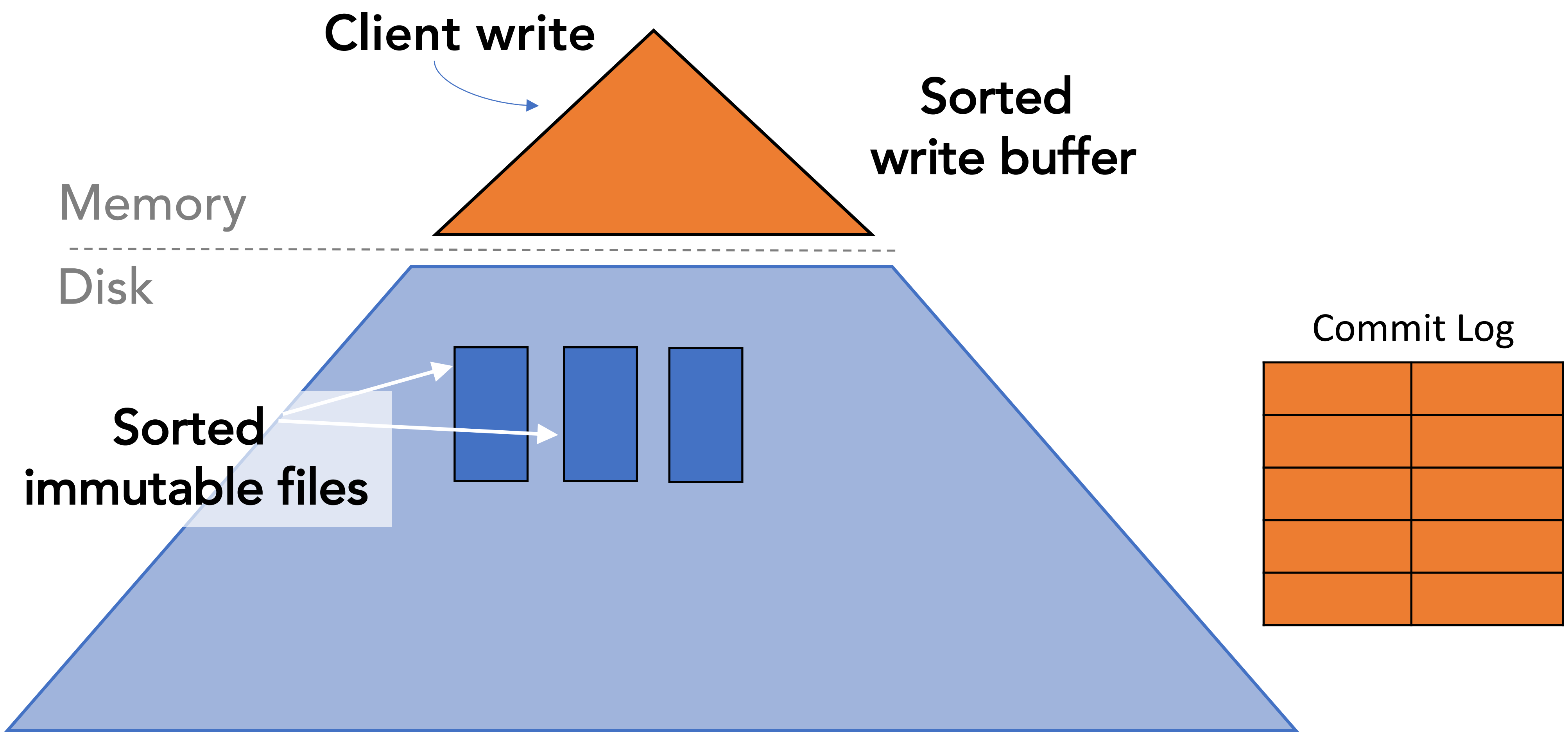
LSM KVs Internals



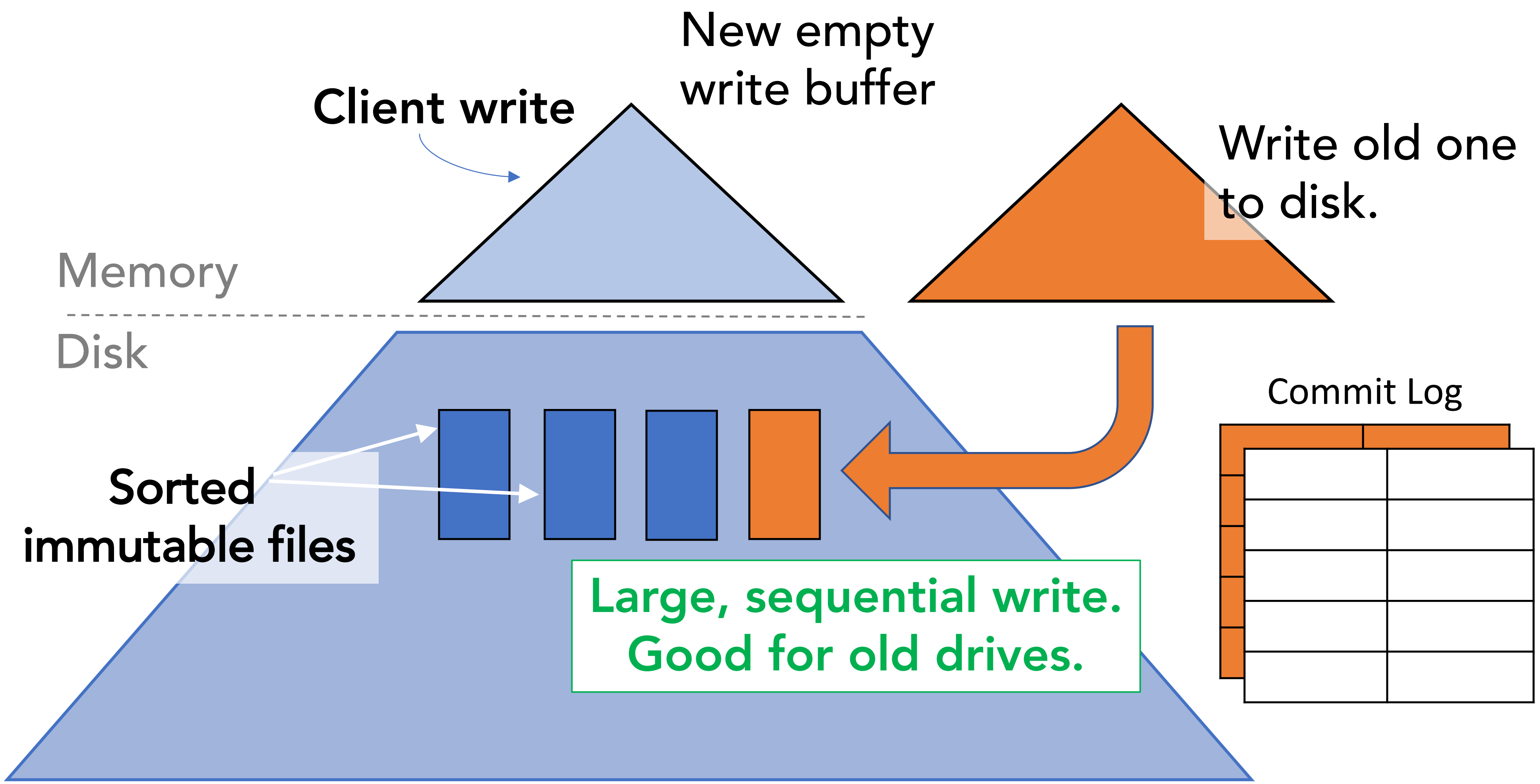
LSM KVs Internals



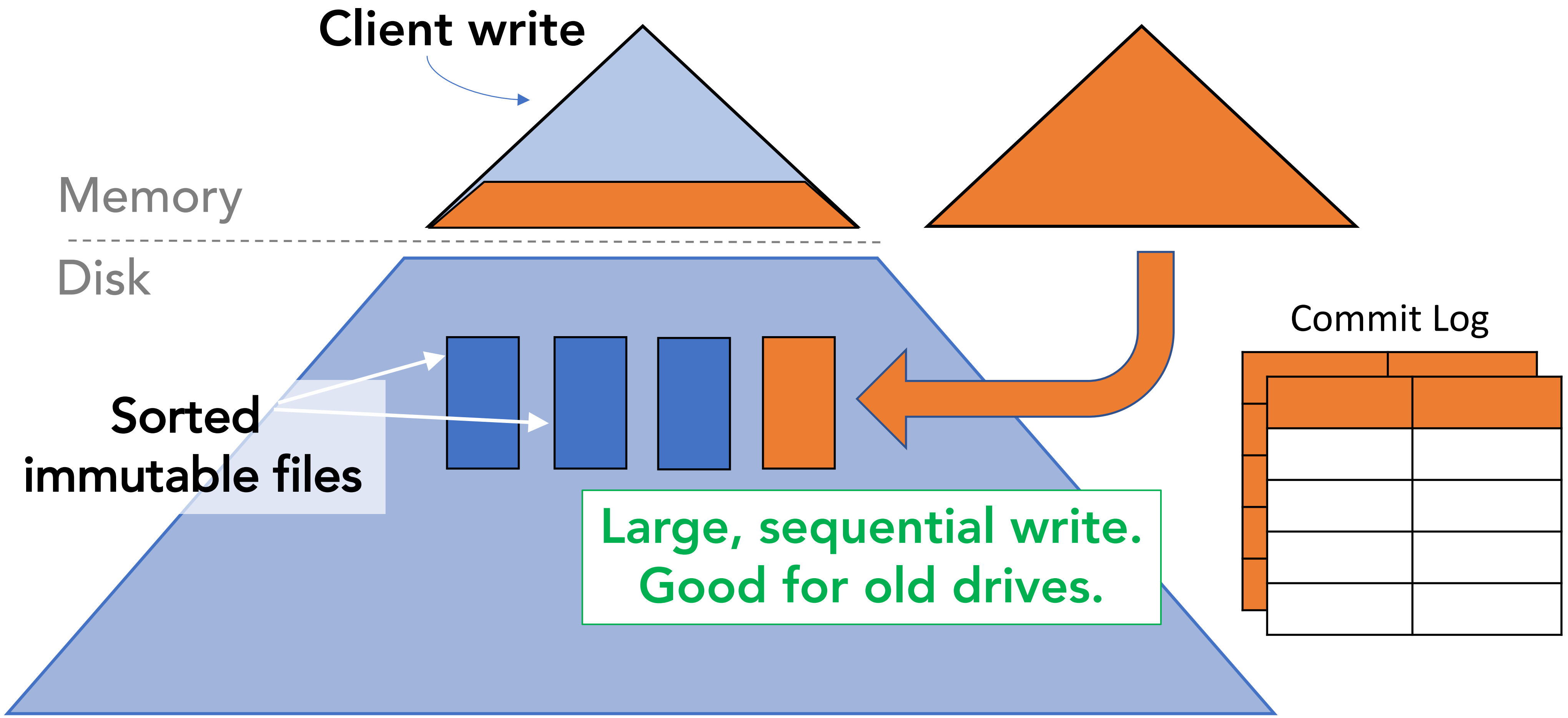
LSM KVs Internals



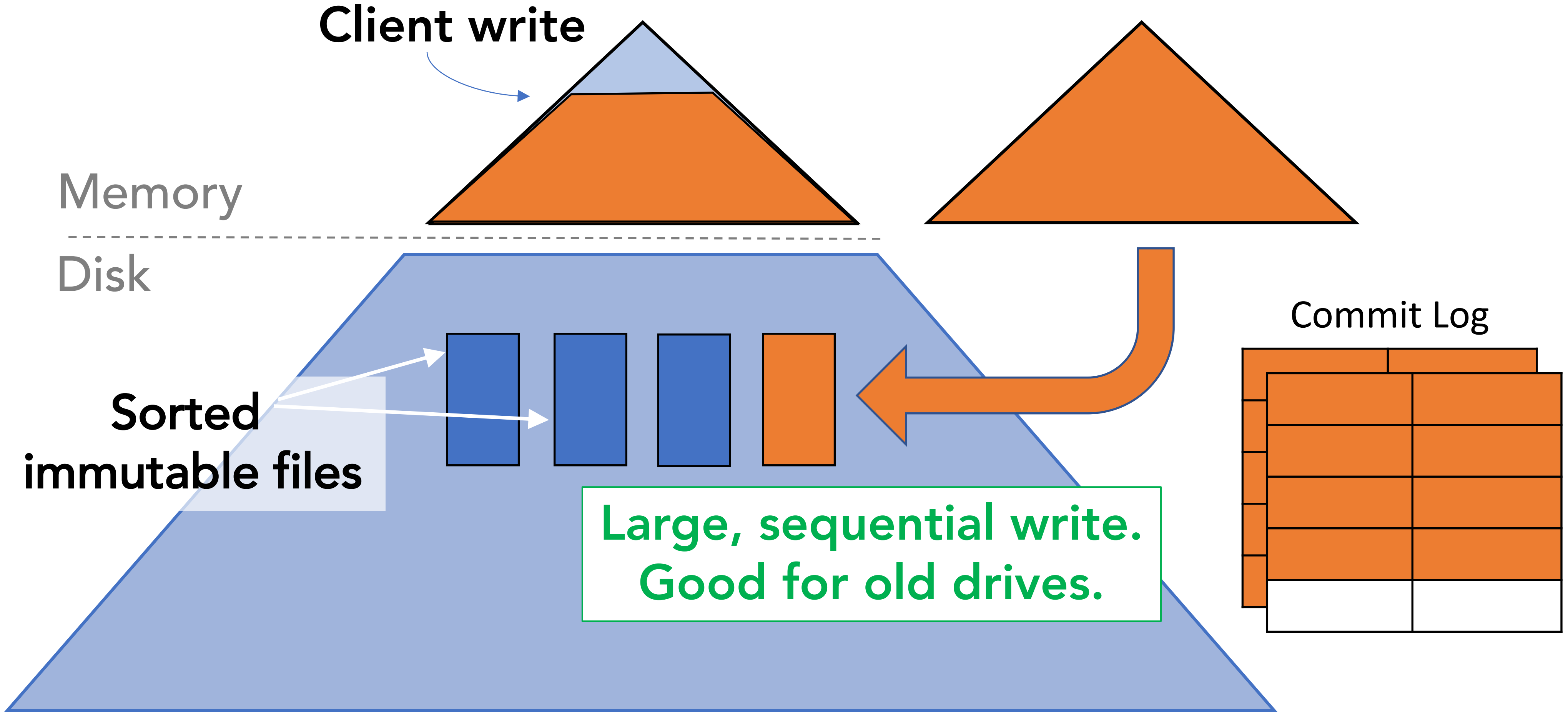
LSM KVs Internals



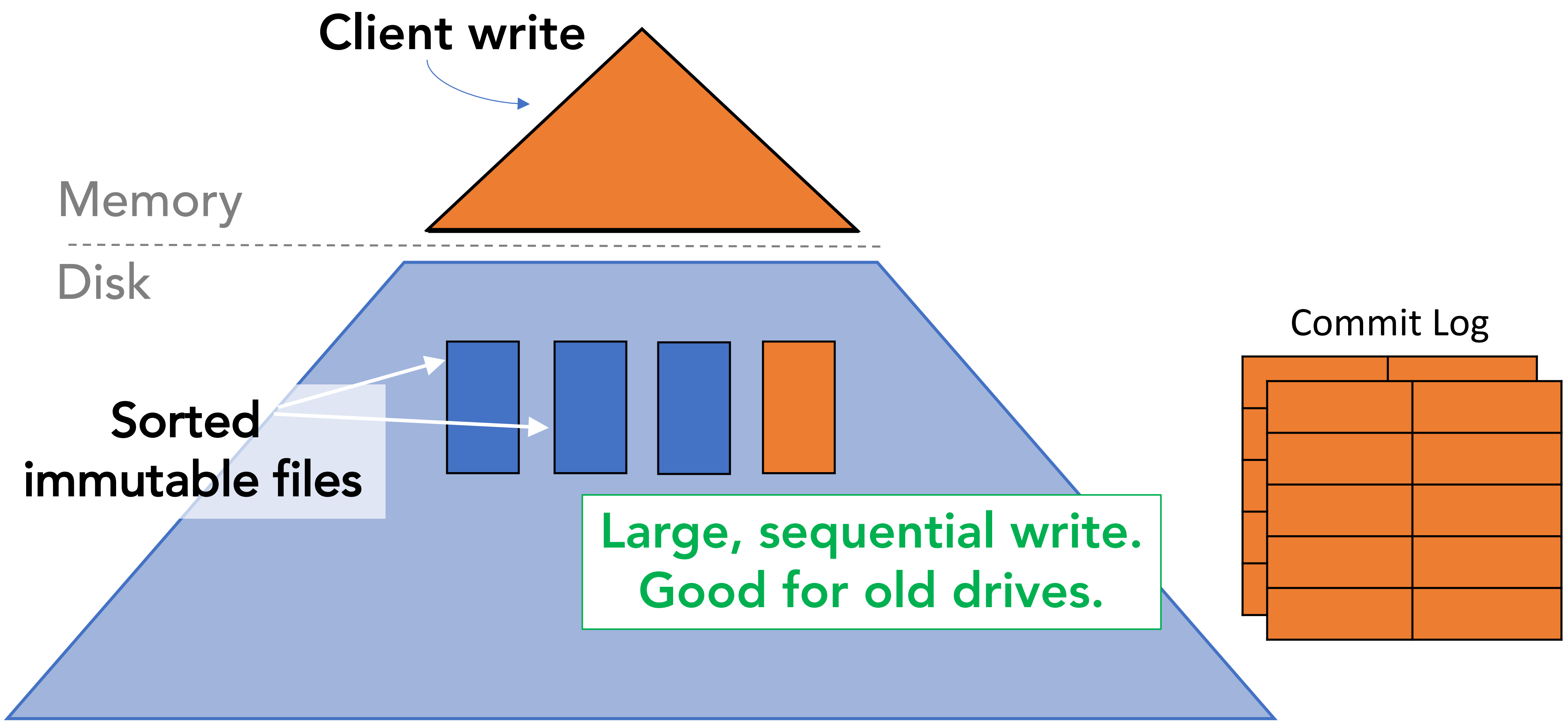
LSM KVs Internals



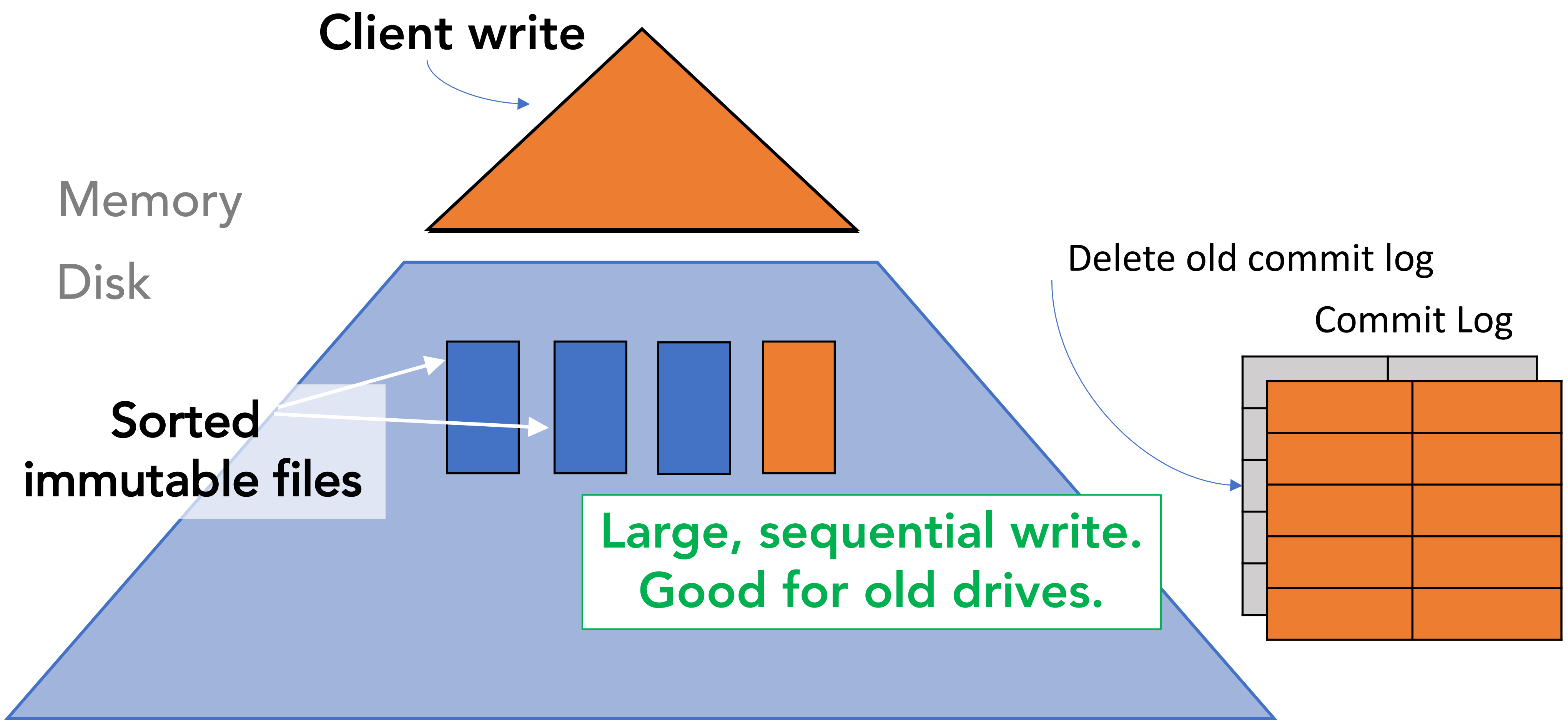
LSM KVs Internals



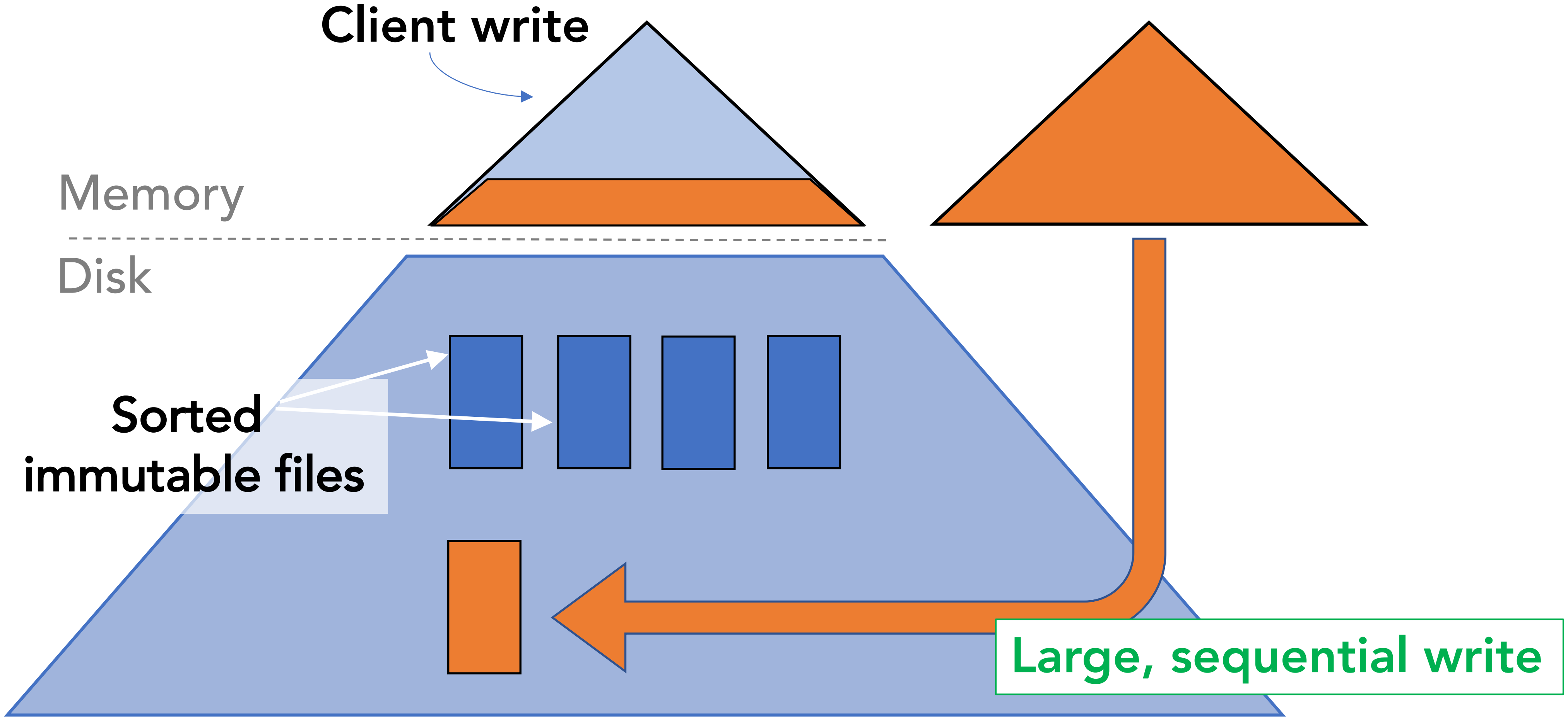
LSM KVs Internals



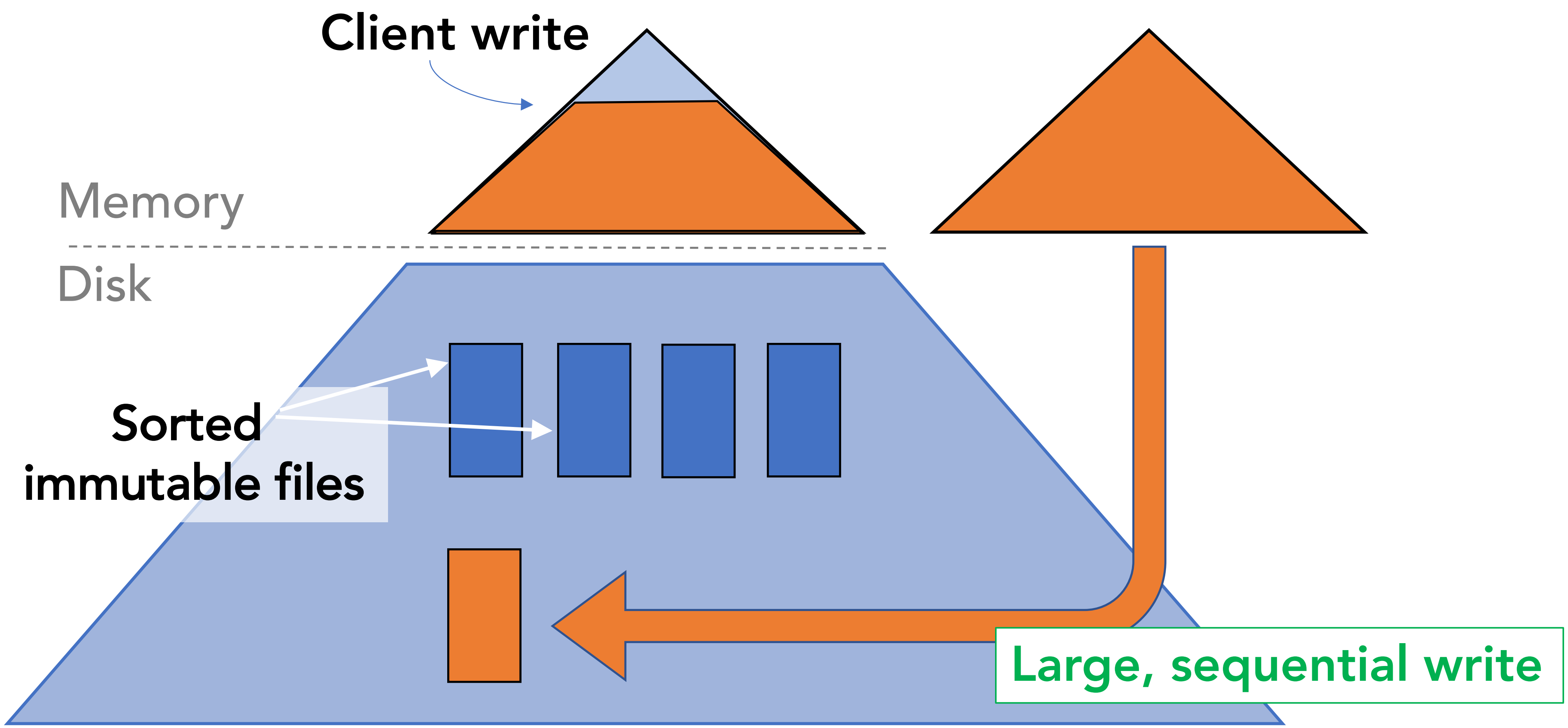
LSM KVs Internals



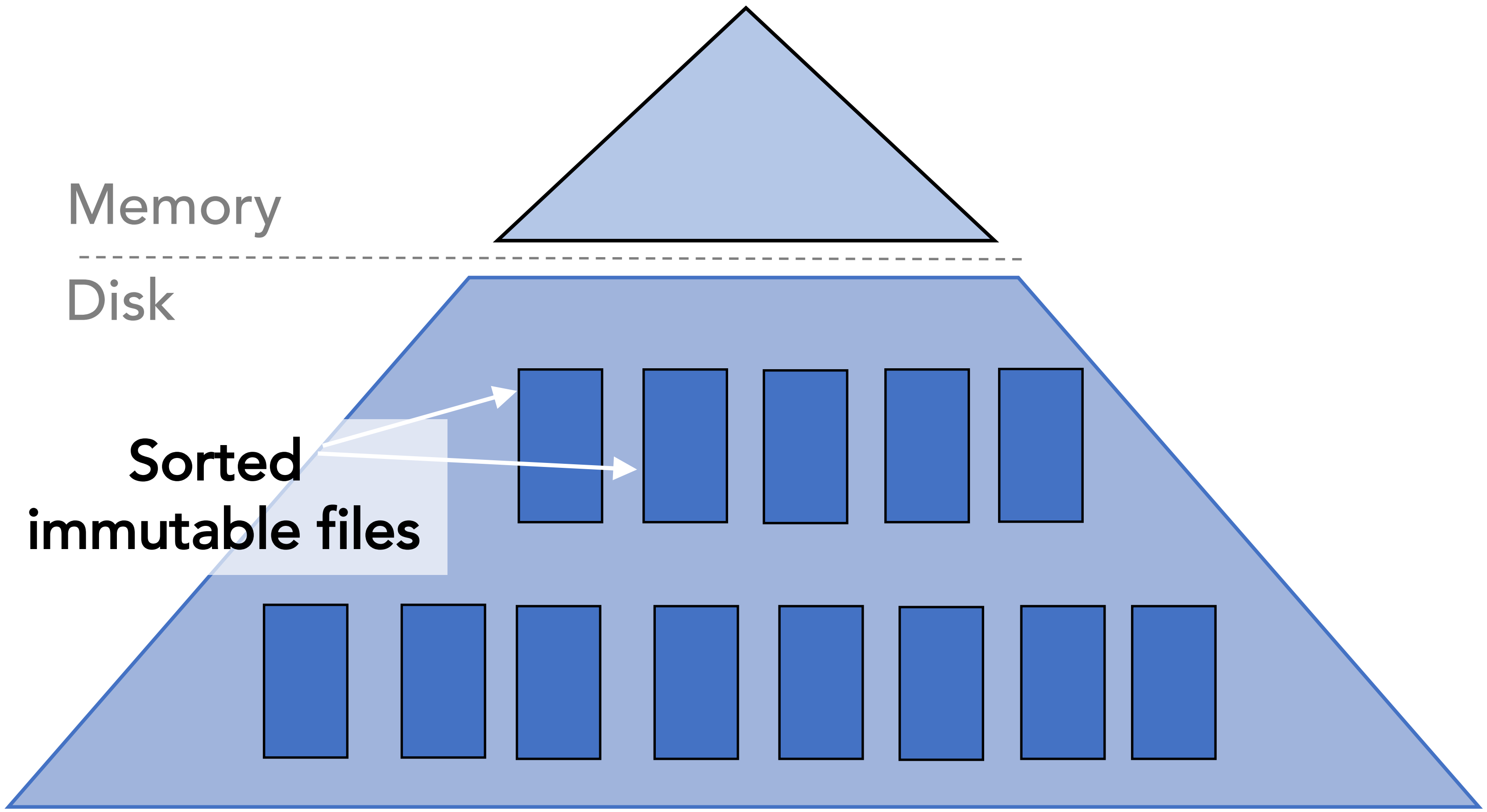
LSM KVs Internals



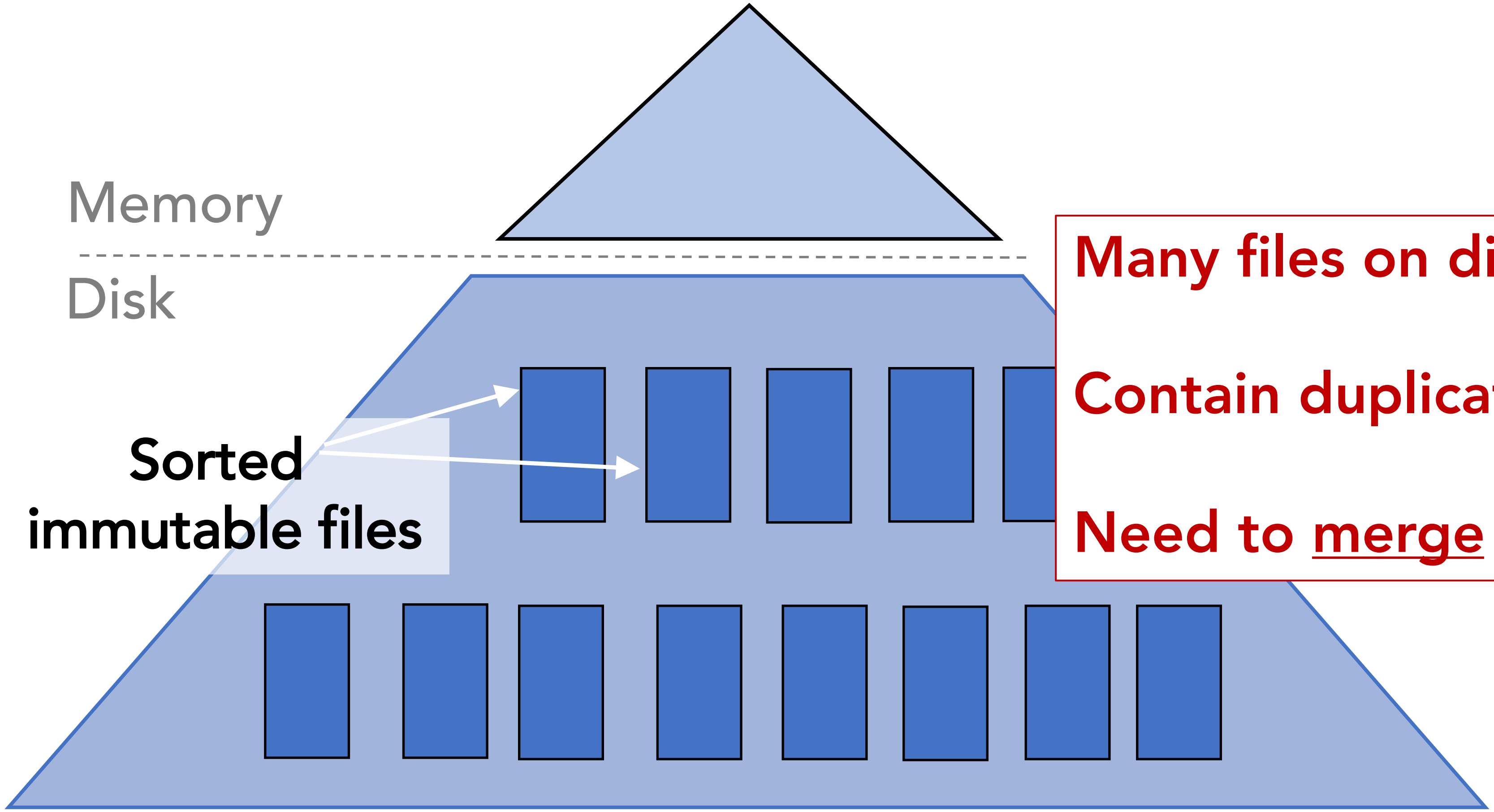
LSM KVs Internals



LSM KVs Internals



LSM KVs Internals

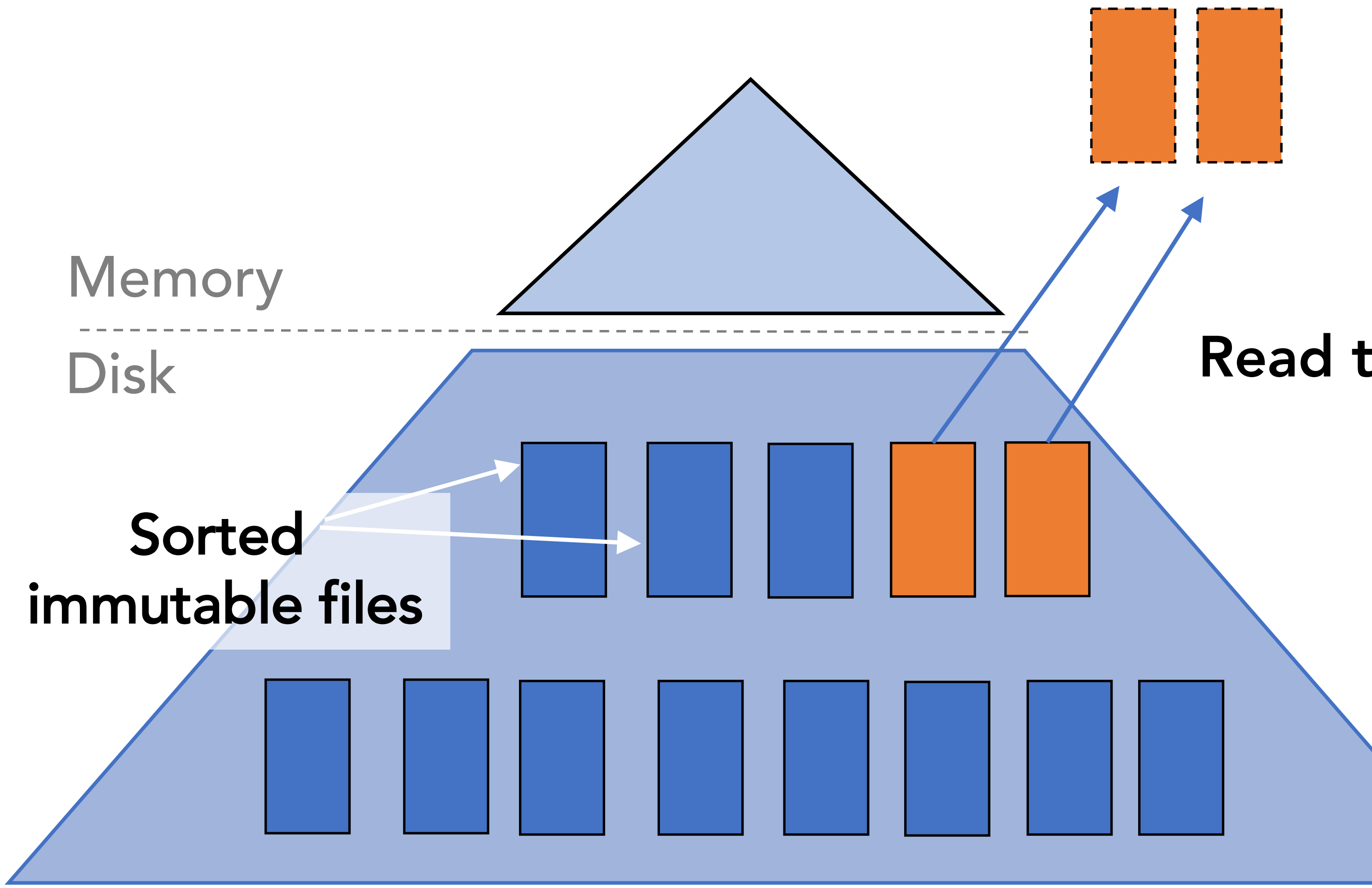


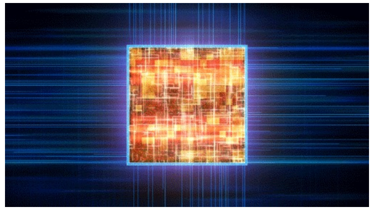
Many files on disk.

Contain duplicate data.

Need to <u>merge</u> files.

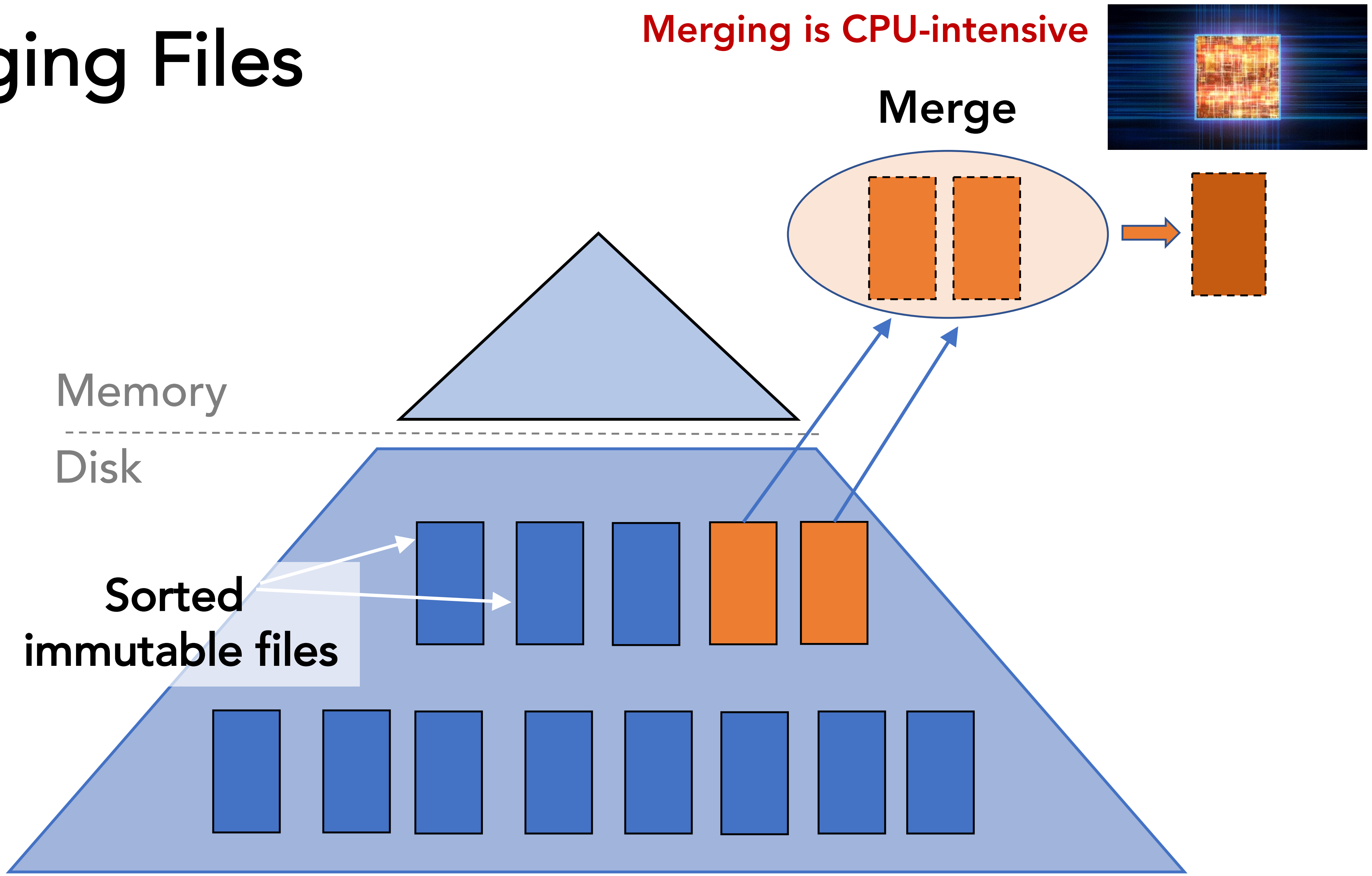
LSM KVs Internals: Merging Files





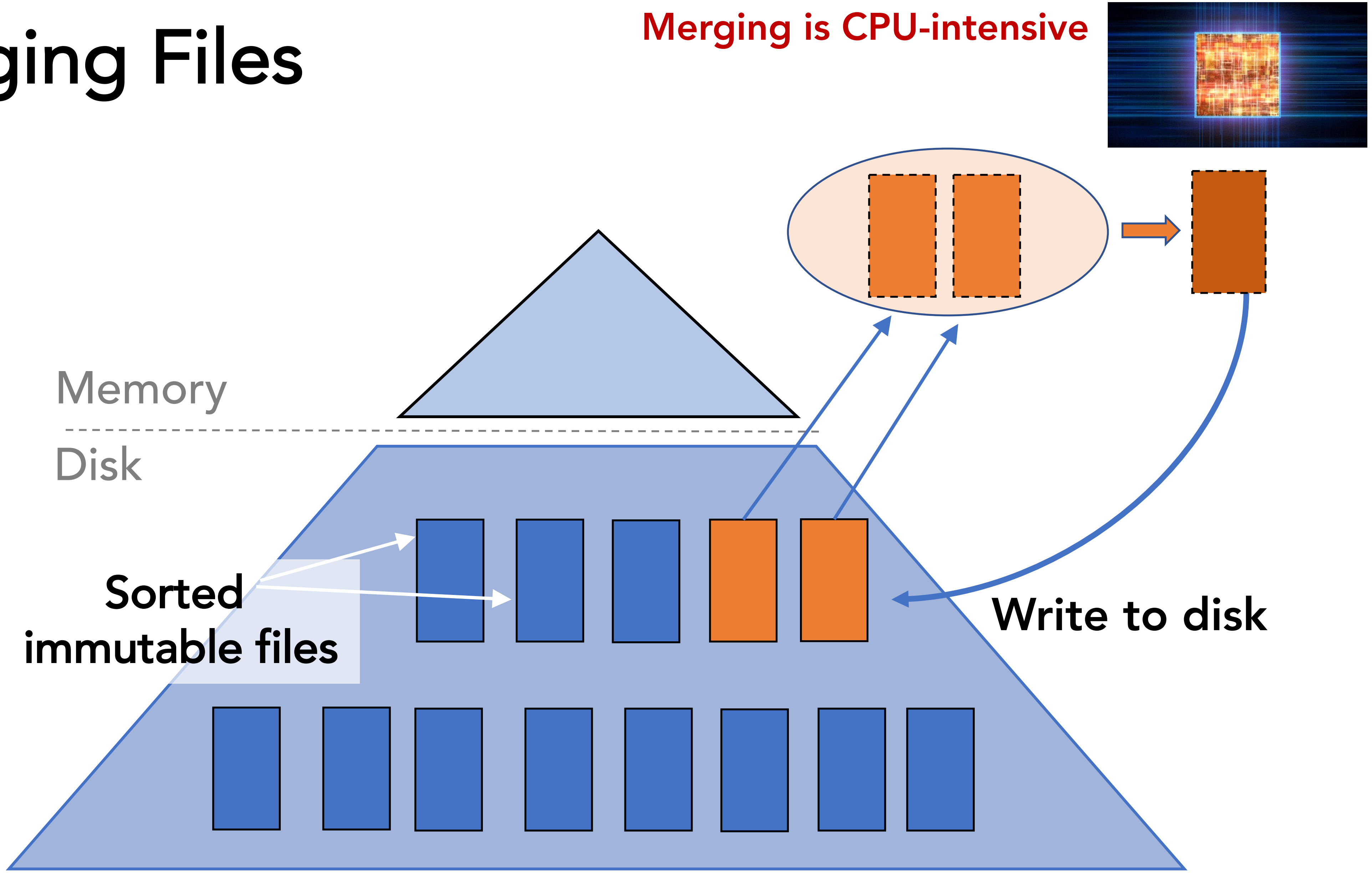
Merging is CPU-intensive

Merging Files



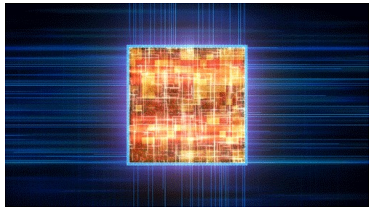
Merging Files

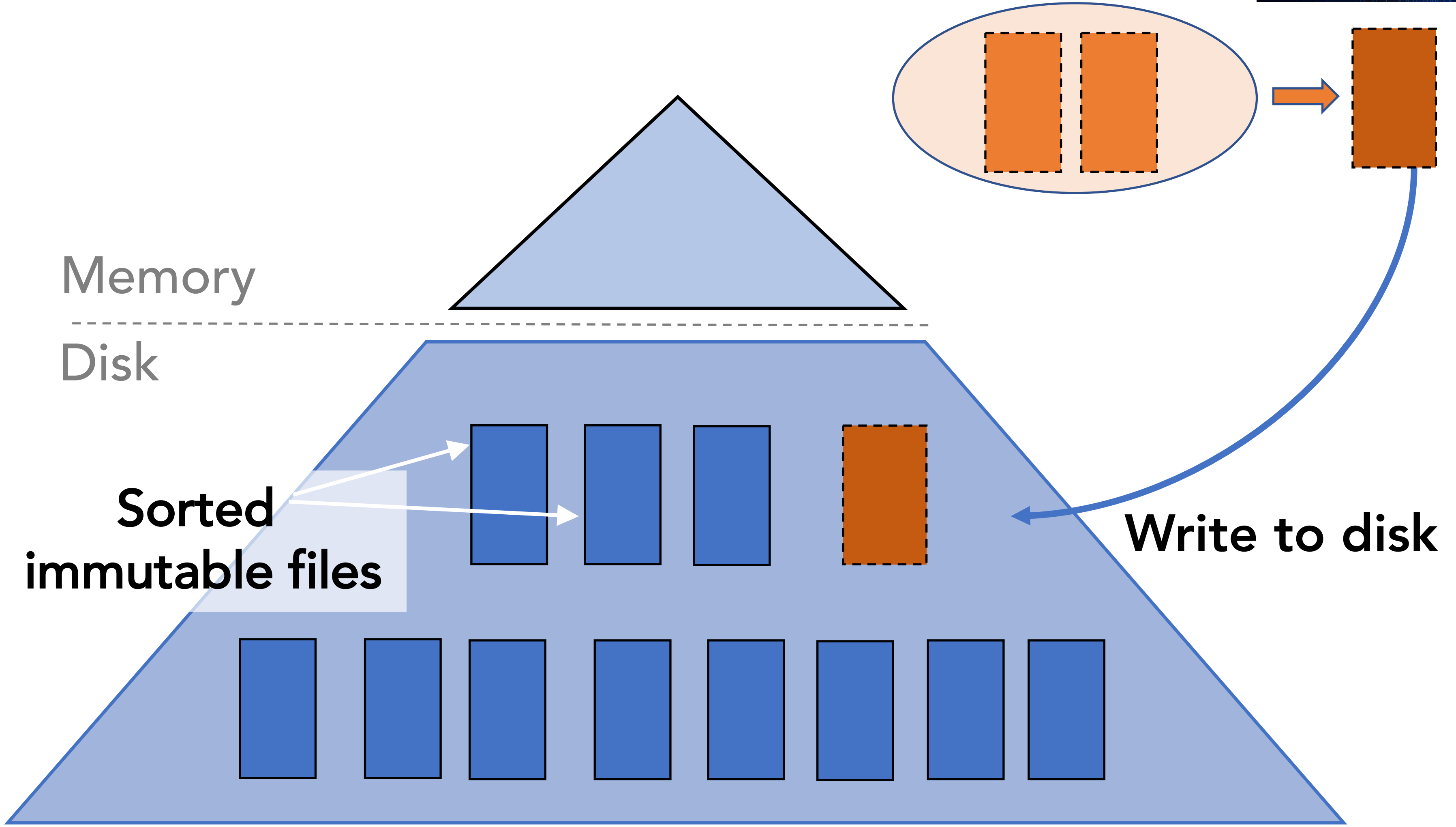
Merging is CPU-intensive



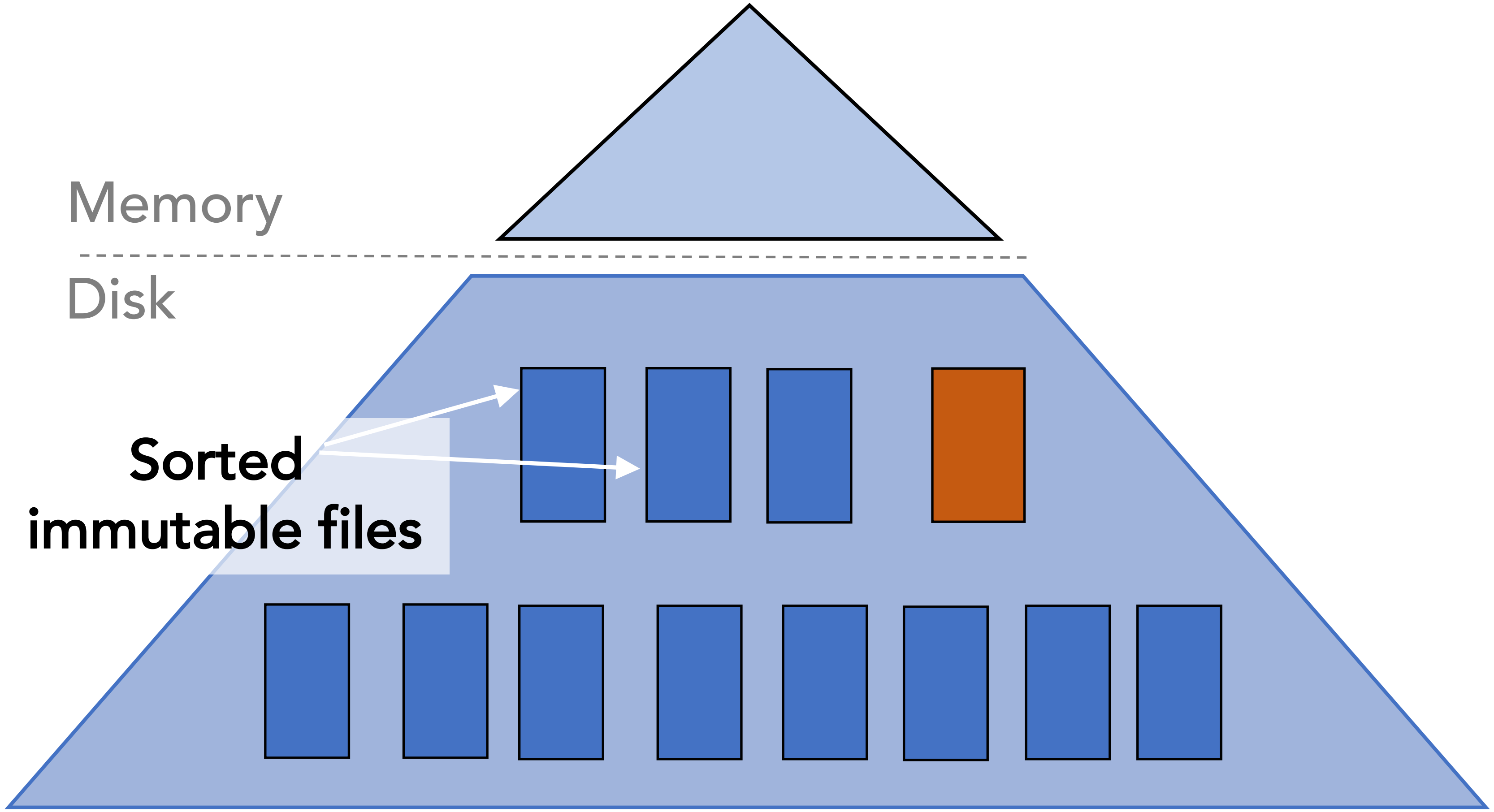
Merging Files

Merging is CPU-intensive

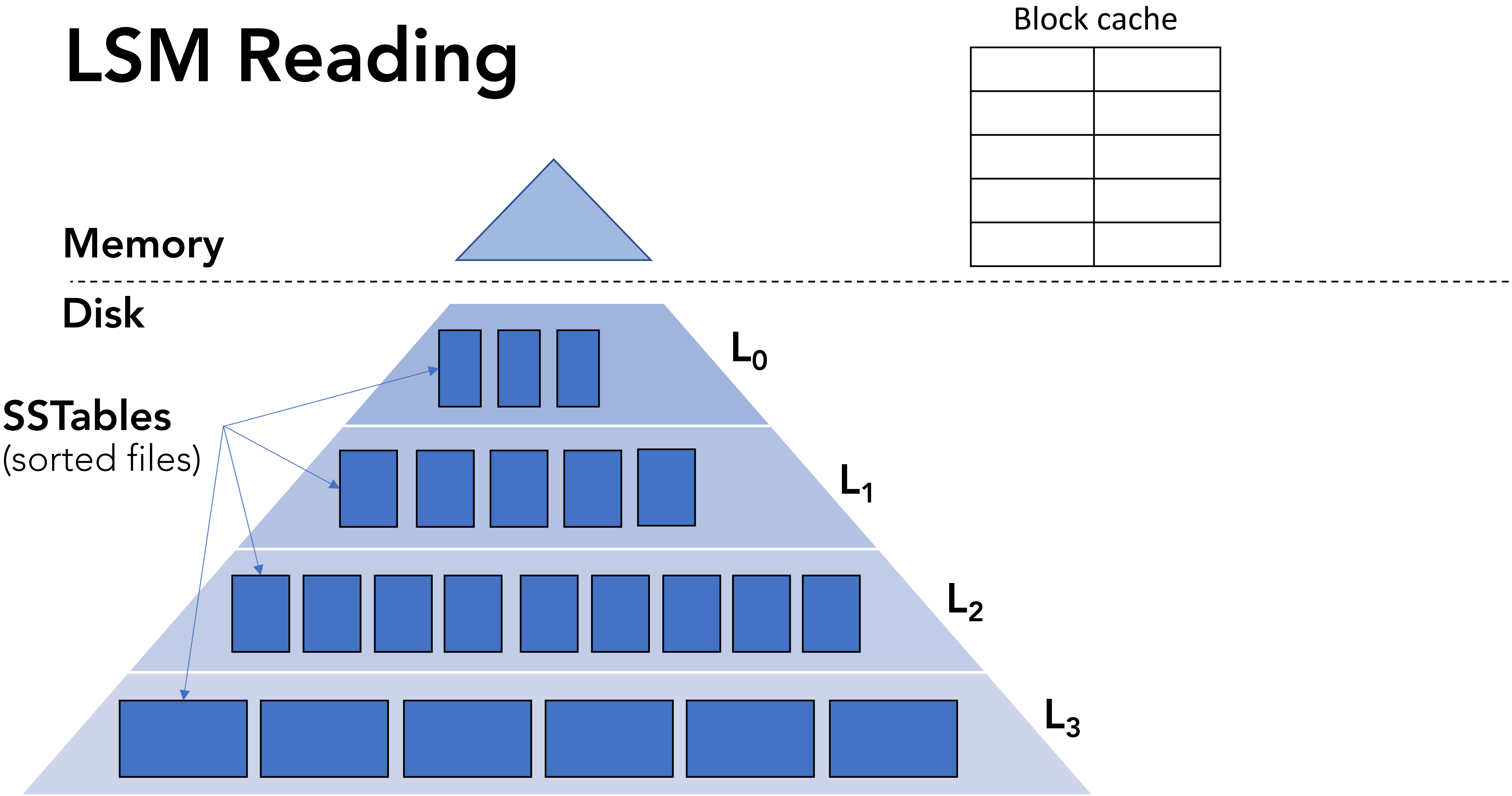




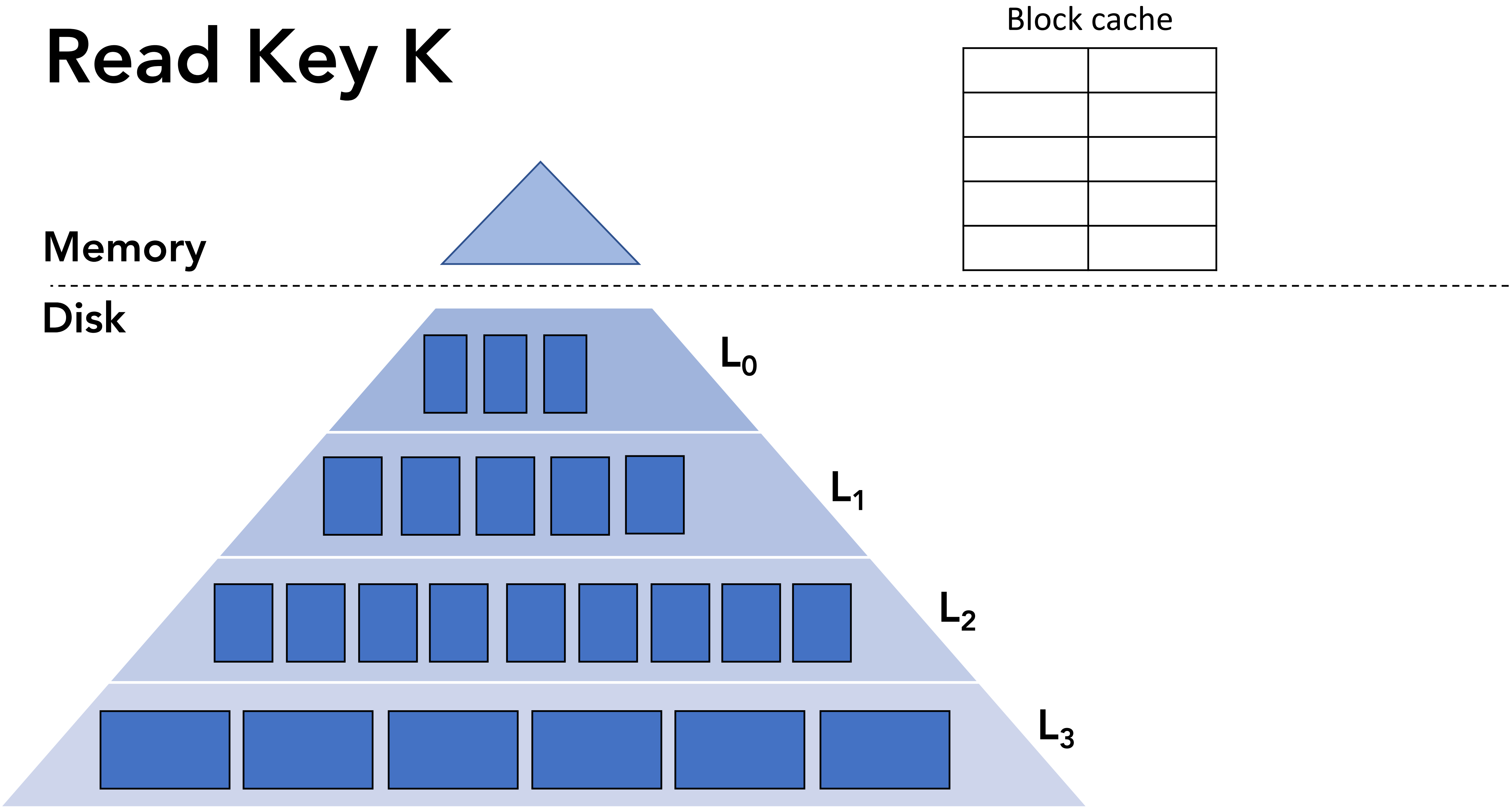
Merging Files



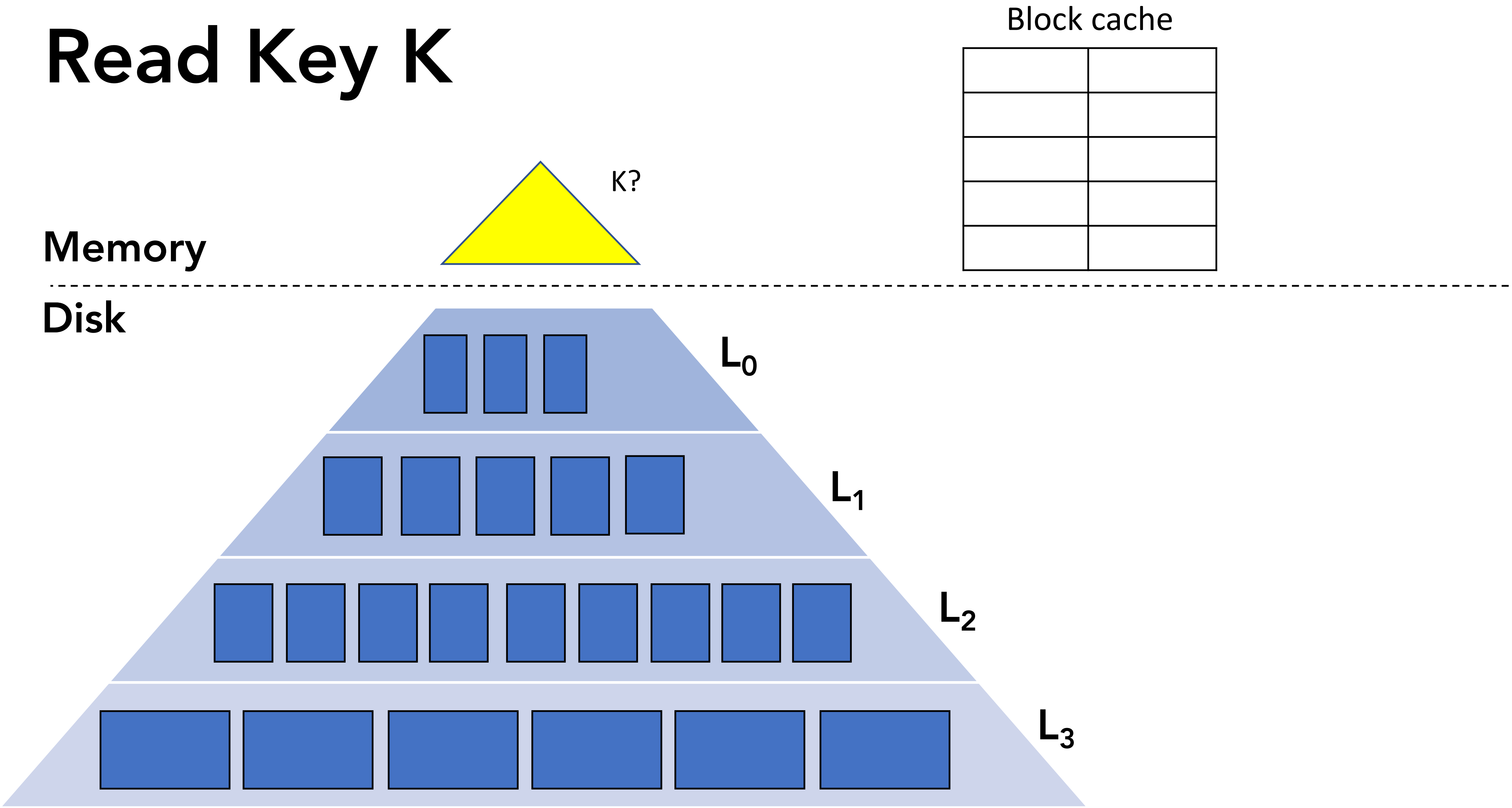
LSM Reading



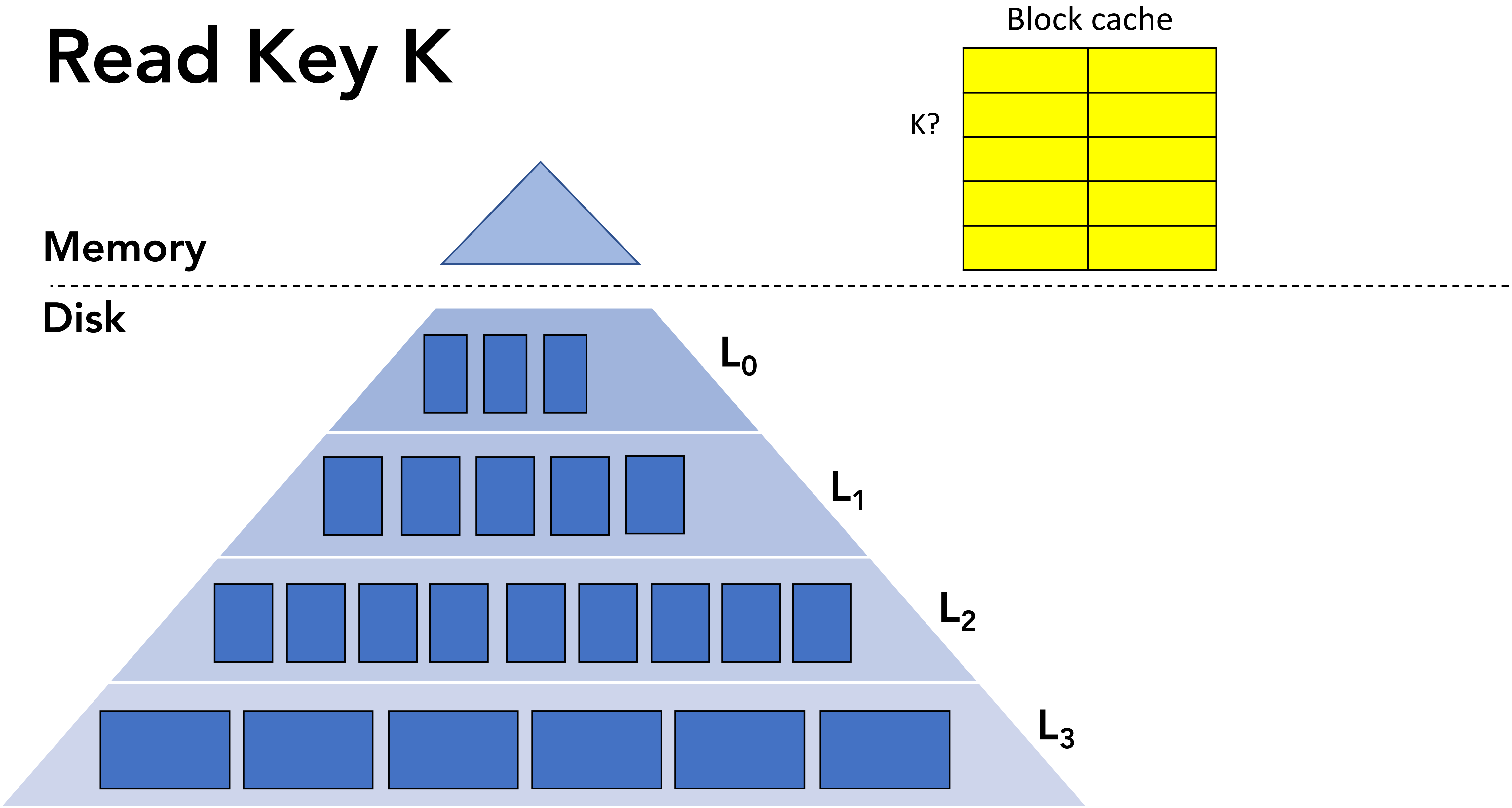
Read Key K



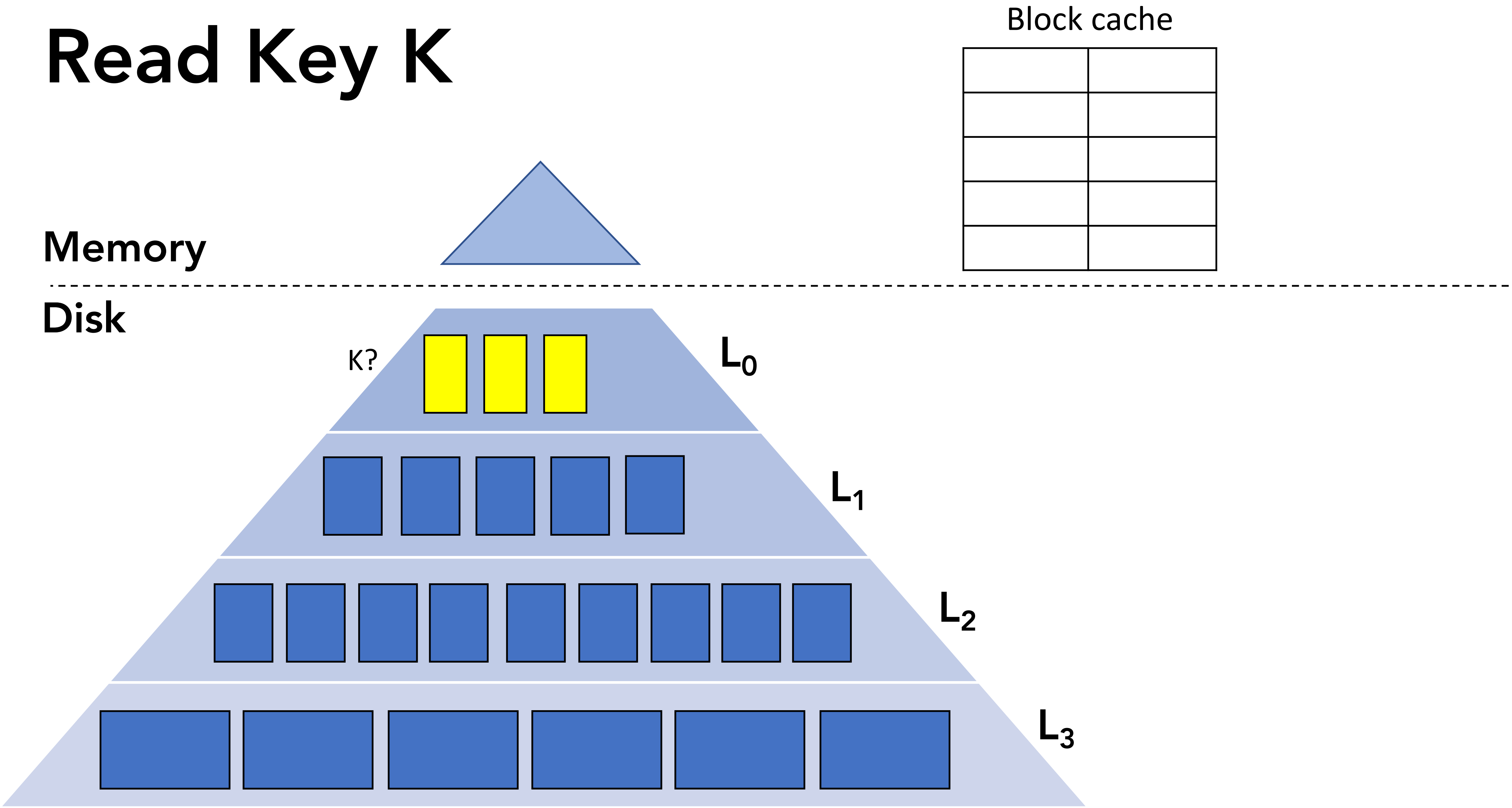
Read Key K



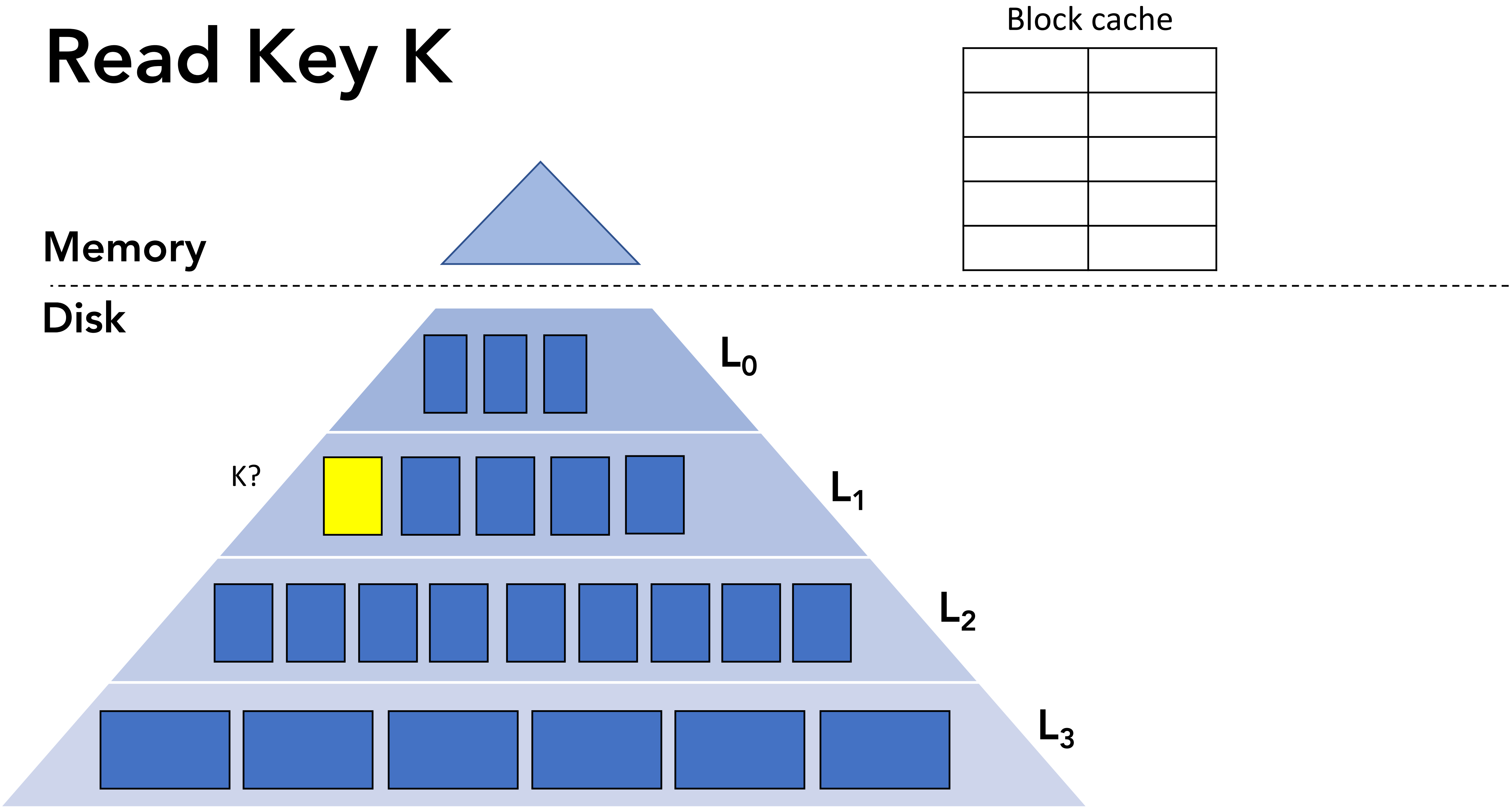
Read Key K



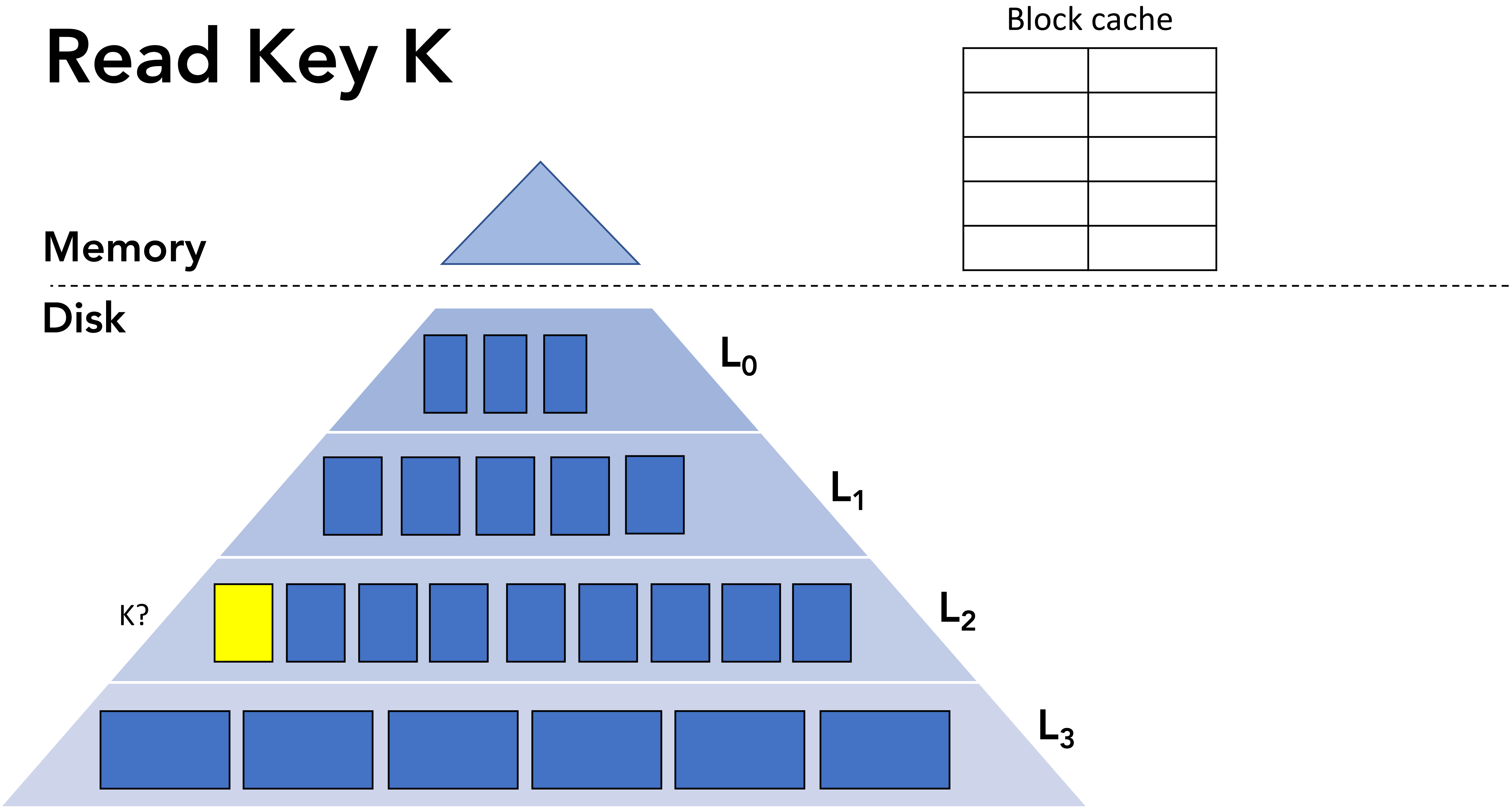
Read Key K



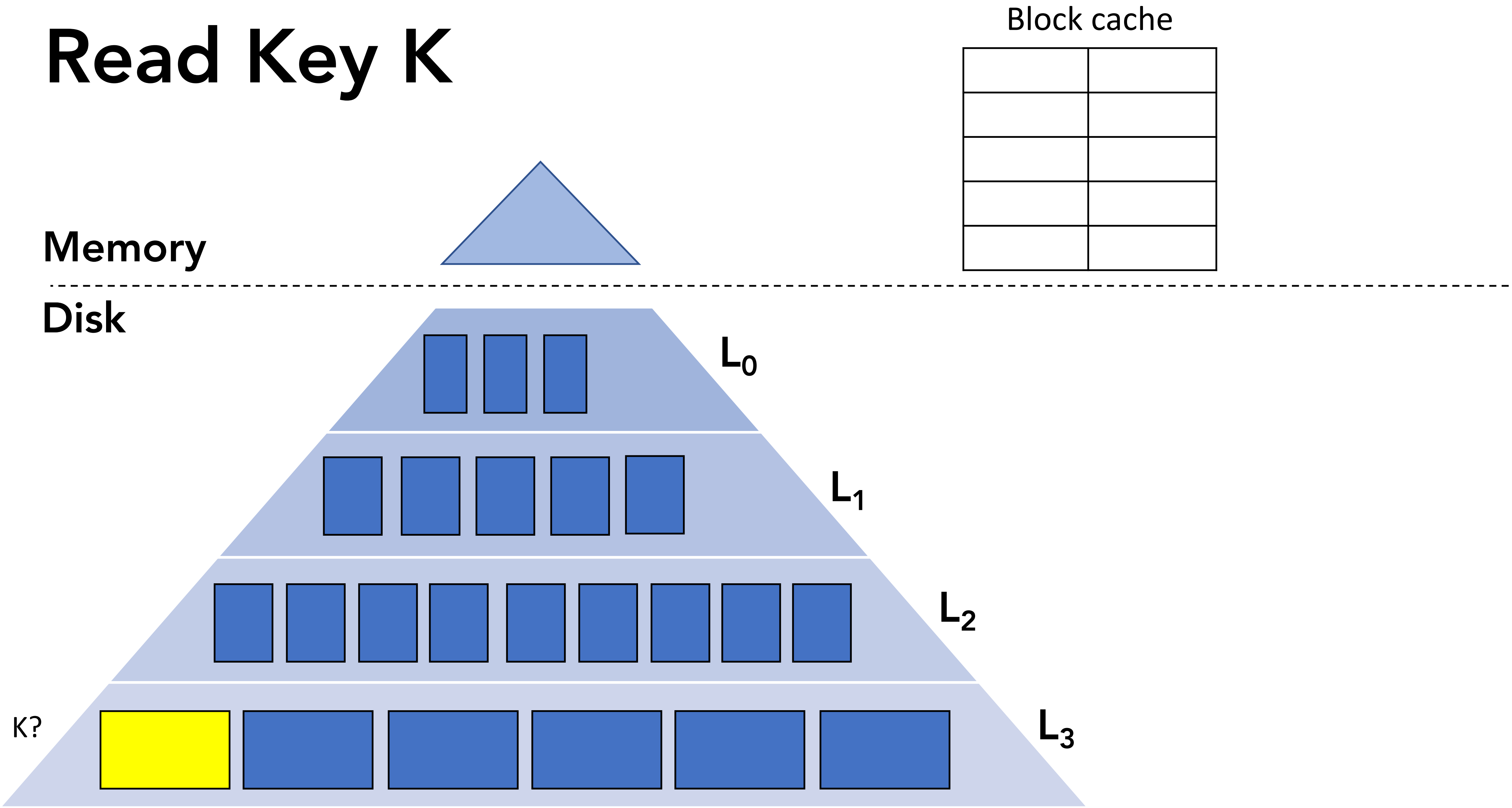
Read Key K



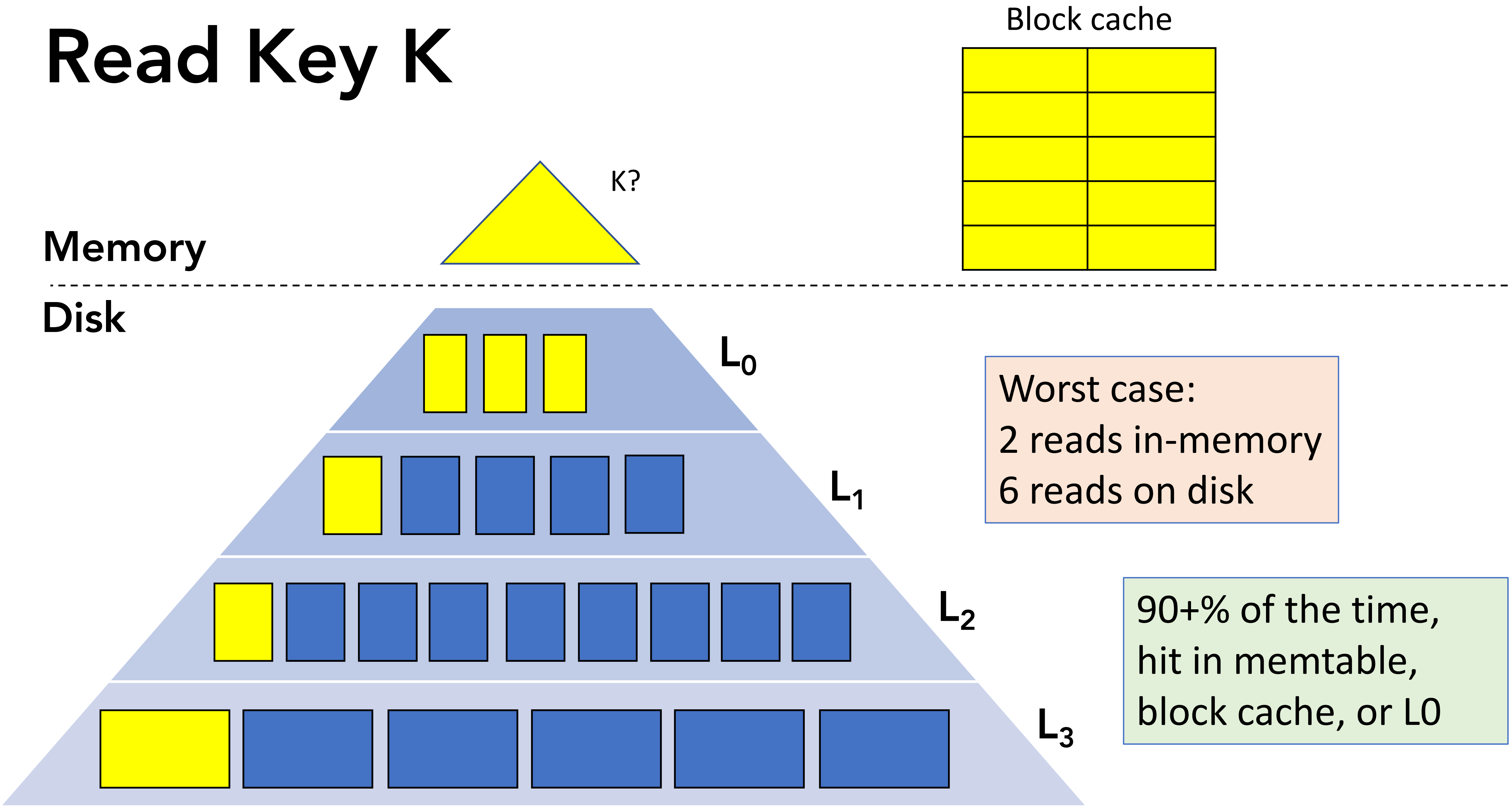
Read Key K



Read Key K



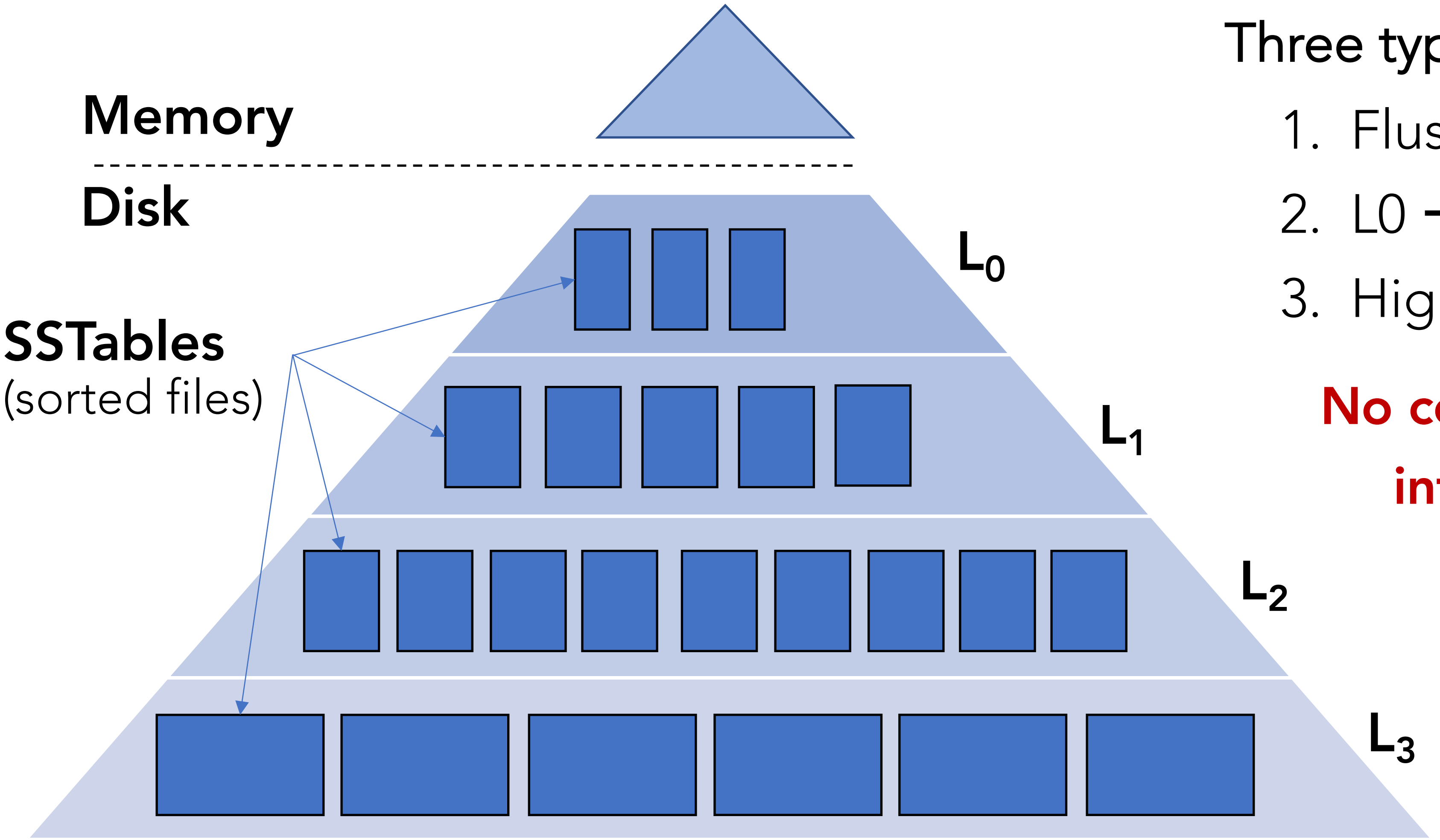
Read Key K



Worst case:
2 reads in-memory
6 reads on disk

90+% of the time, hit in memtable, block cache, or L0

LSM Internal Ops

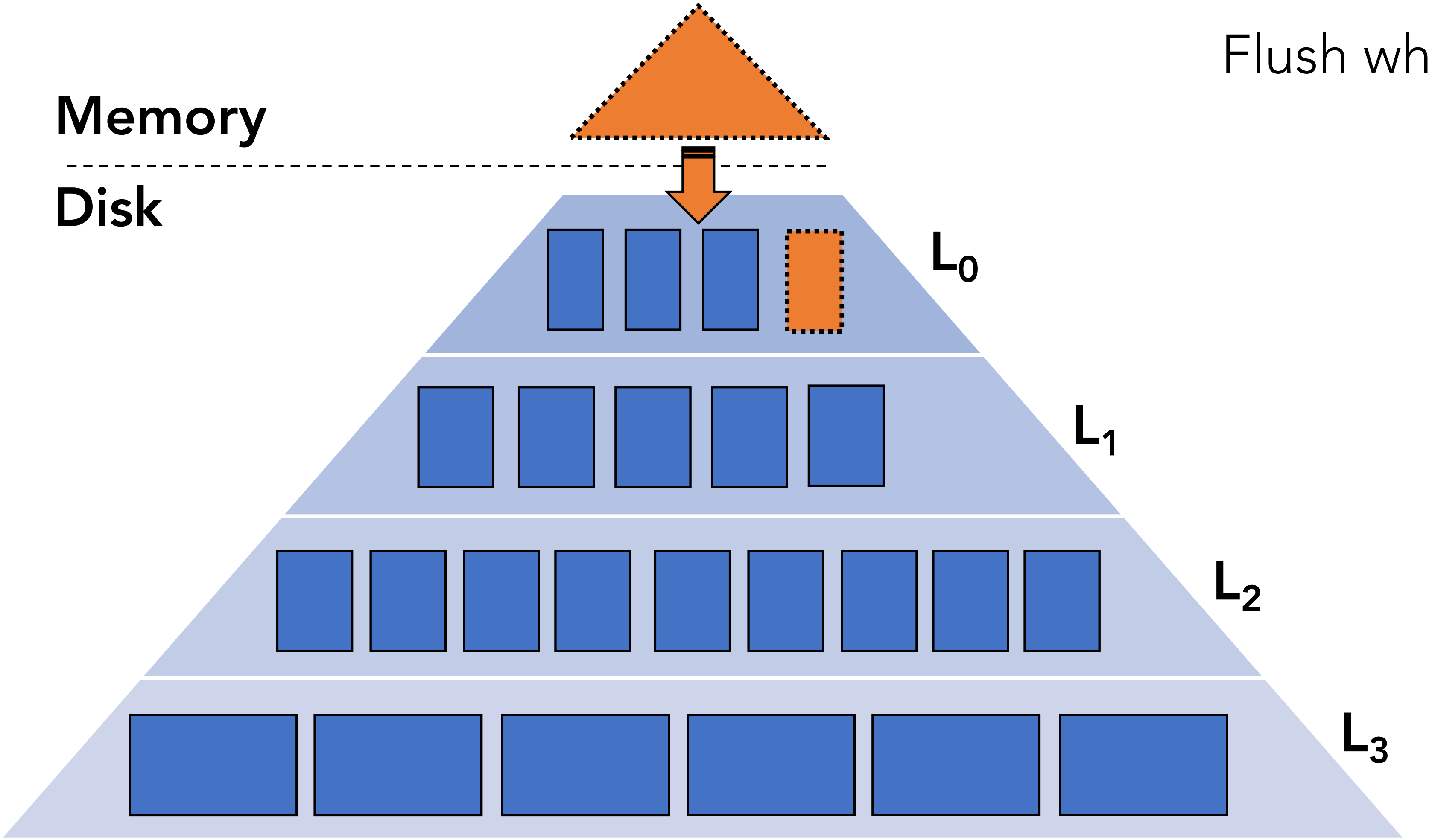


Three types of internal ops:

1. Flushing
2. L0 → L1 compaction
3. Higher level compactions

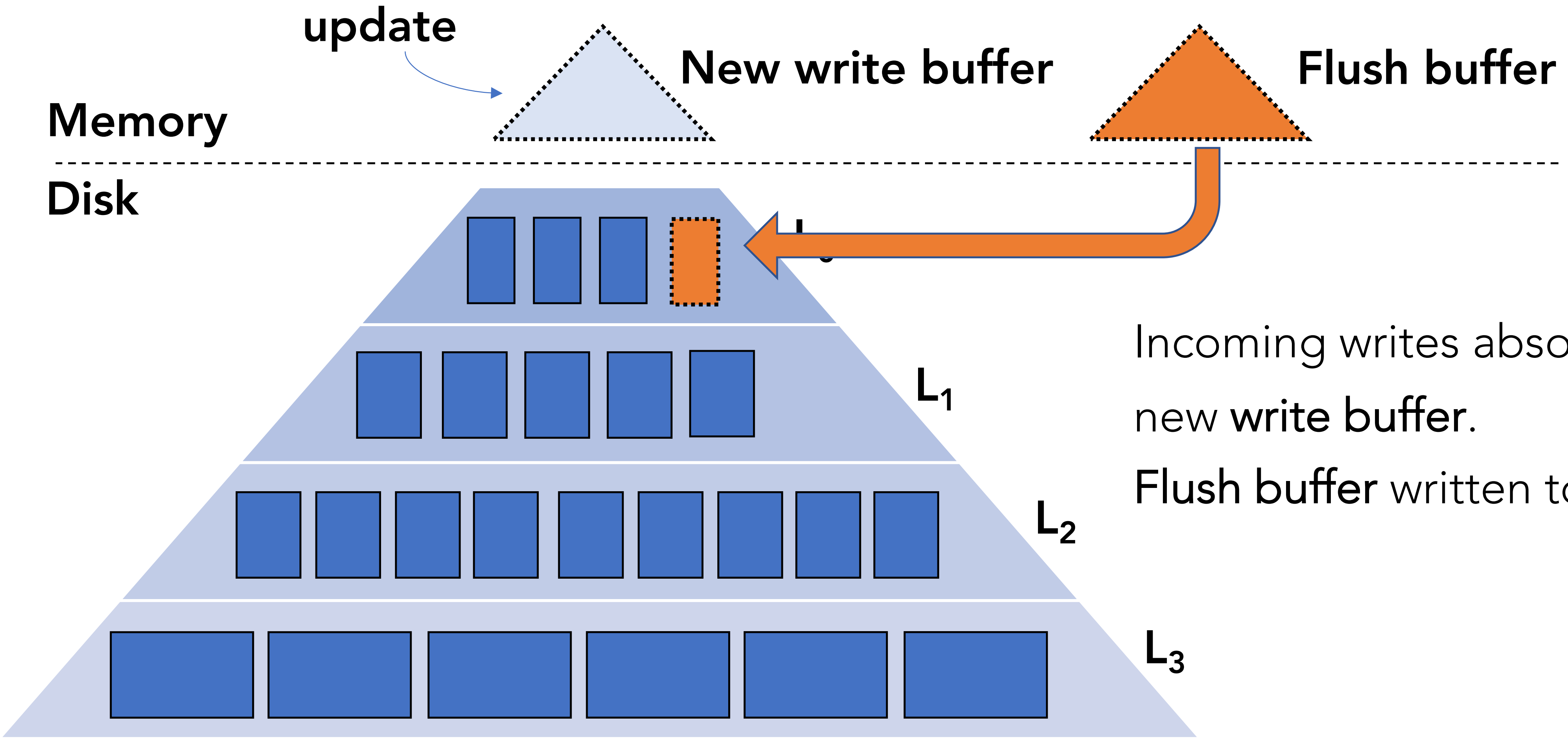
No coordination between internal operations.

LSM Internal Ops: Flushing



Flush when **Write buffer full**.

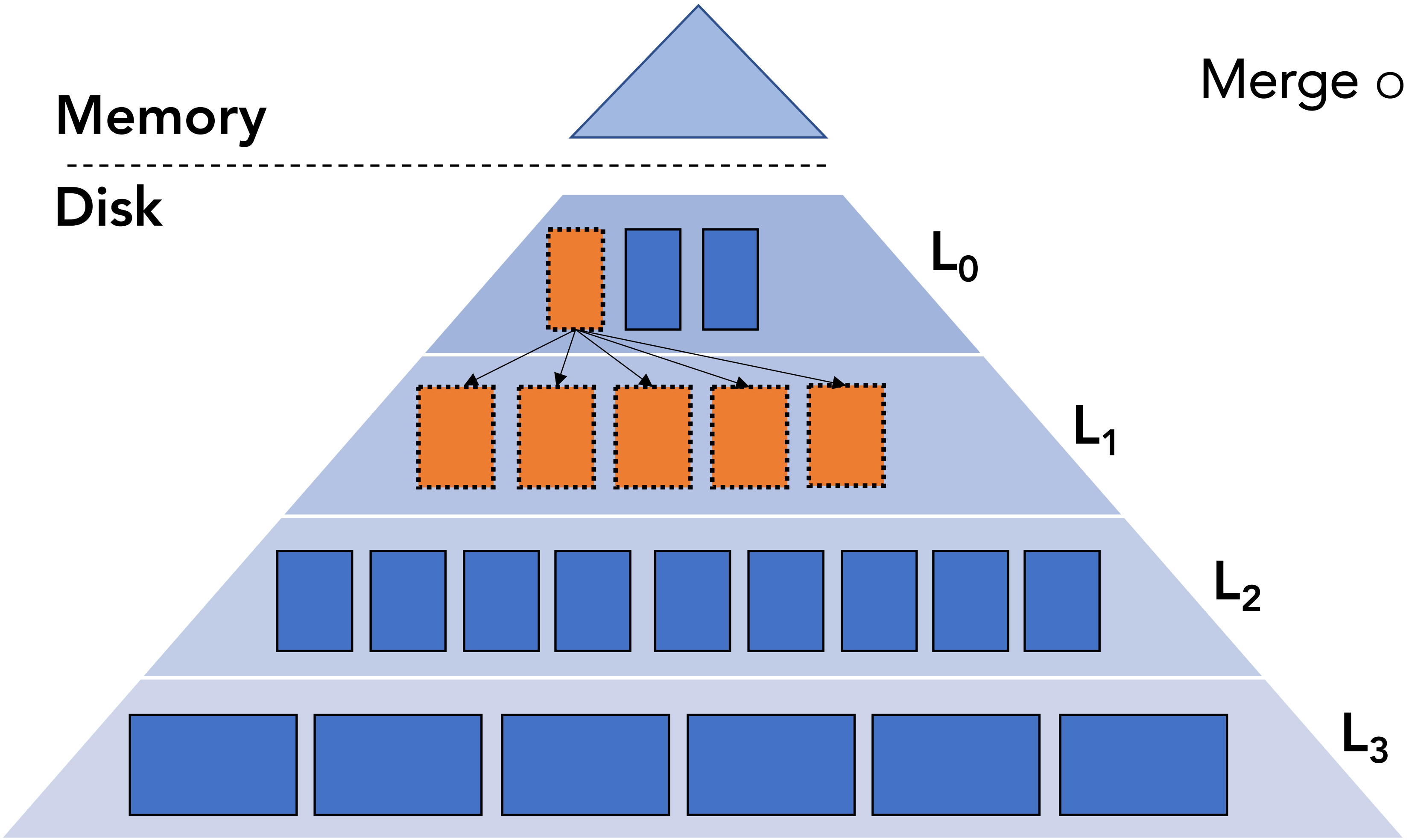
LSM Internal Ops: Flushing



Incoming writes absorbed in new **write buffer**.

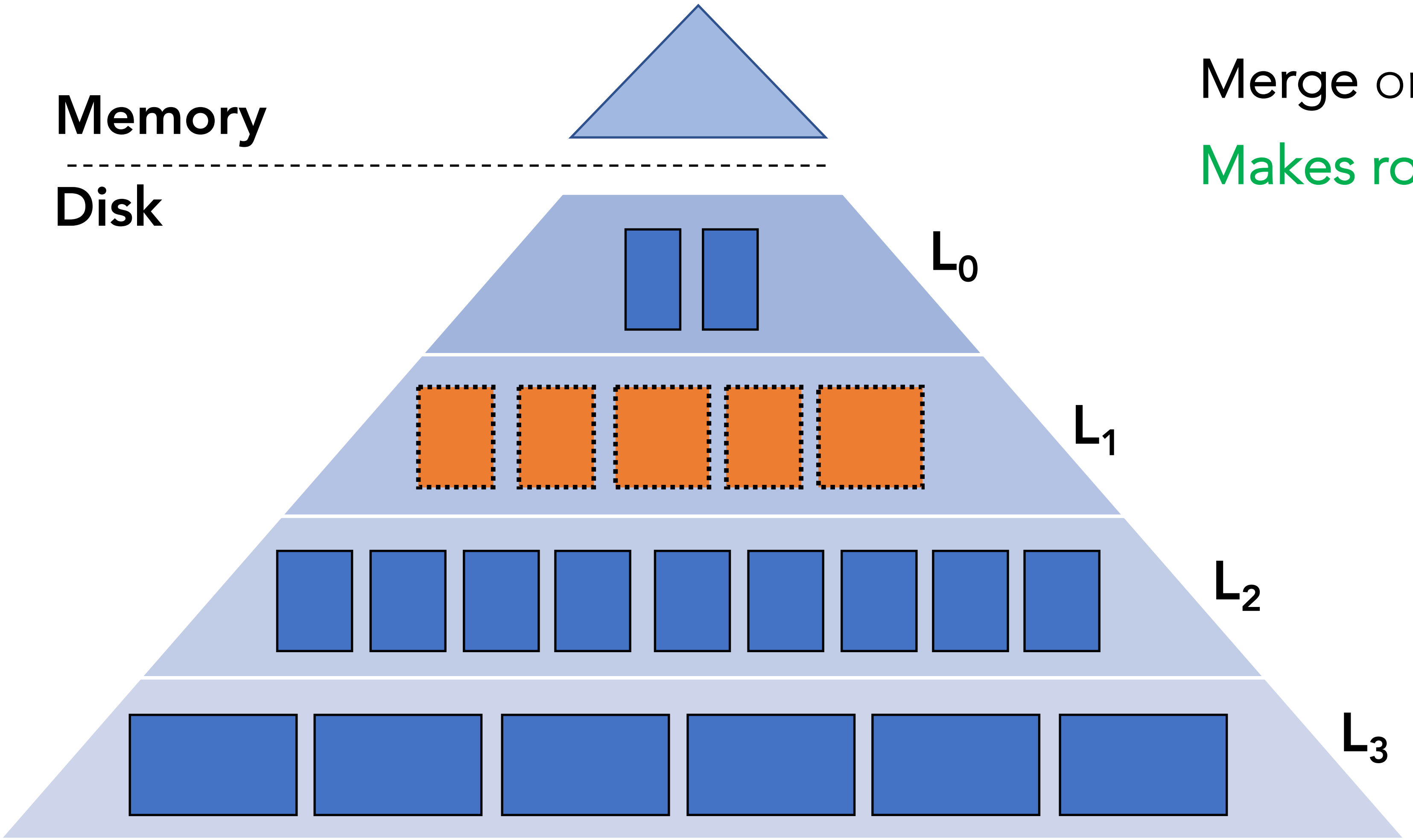
Flush buffer written to L0.

LSM Internal Ops: L0 → L1 compactions



Merge one L0 SSTable with L1.

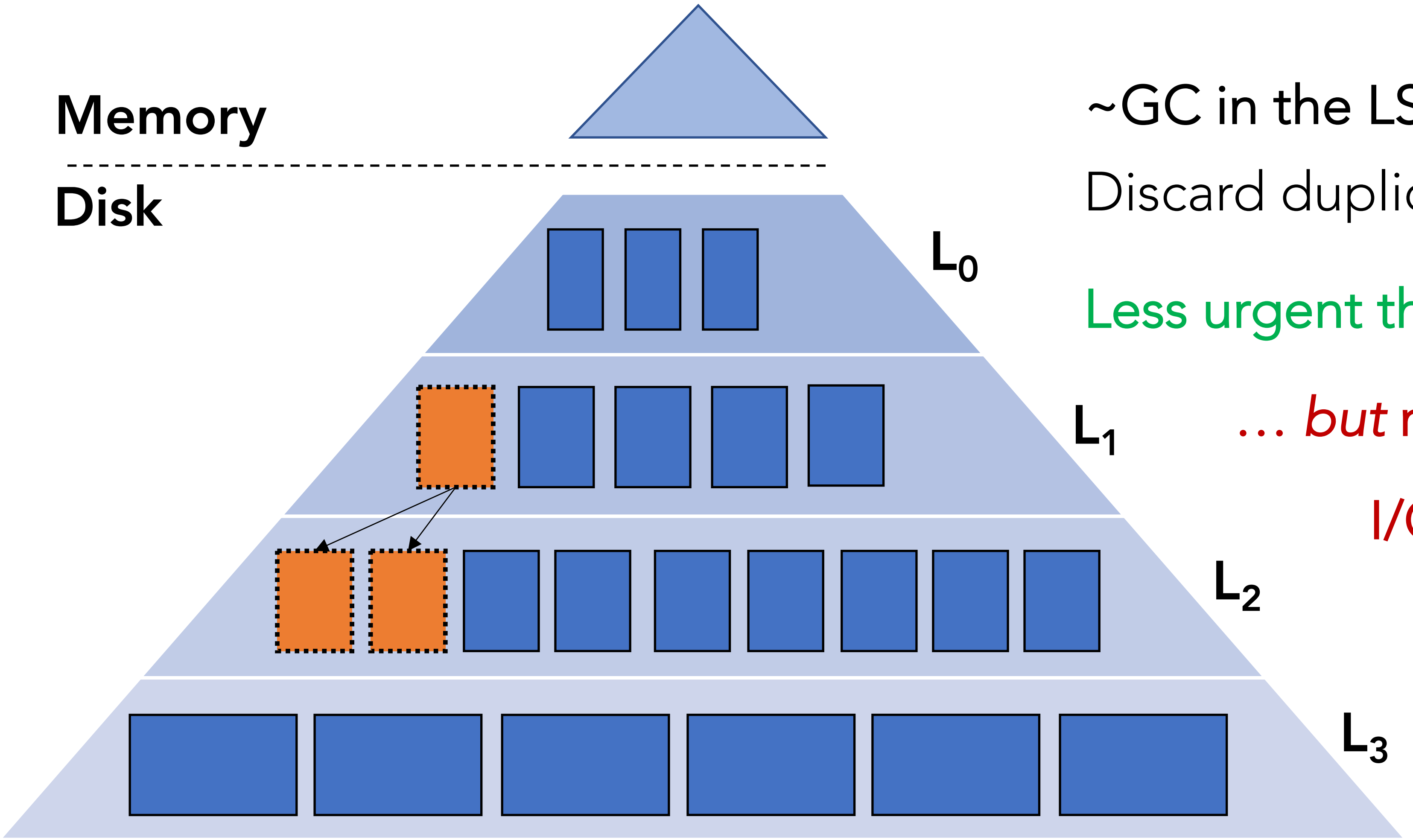
LSM Internal Ops: L0 → L1 compactions



Merge one L0 SSTable with L1.

Makes room on L0 for flushing.

LSM Internal Ops: Higher Level Compactions



~GC in the LSM tree.

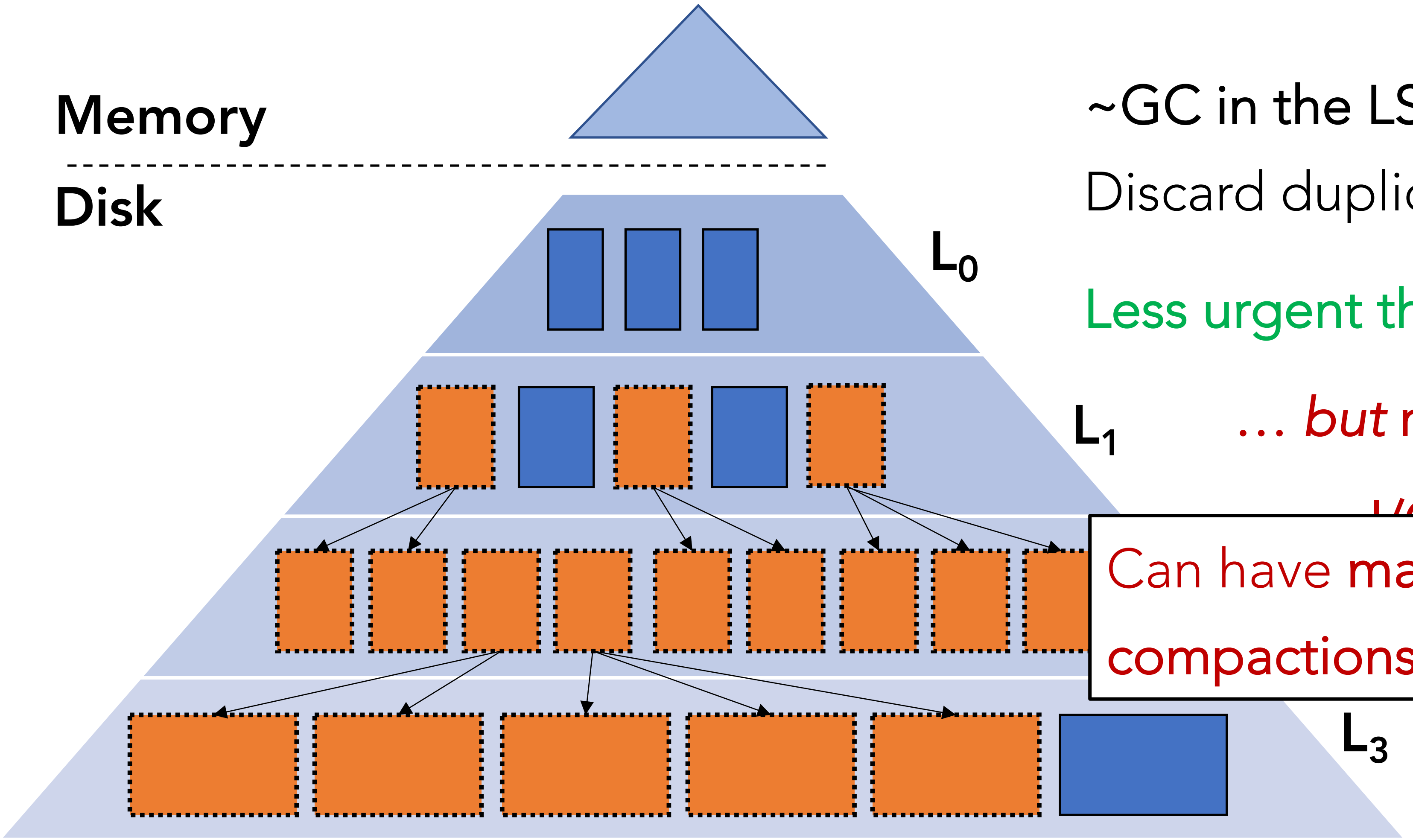
Discard duplicates & delete values.

Less urgent than L0→L1 compactions.

… *but* need to complete.

I/O bandwidth intensive.

LSM Internal Ops: Higher Level Compactions



~GC in the LSM tree.

Discard duplicates & delete values.

Less urgent than L0->L1 compactions.

… *but* **need to complete.**

Can have **many** higher level **compactions** running **in parallel**.

LSM KV Advantages

- Fast writes
  - Client-side: In-memory write, and cheap sequential append on disk
  - System-side: Cheap, sequential write when flushing to disk

- Reads mostly from memory
  - Reading optimization: using Bloom filters to check whether key is in a file on disk.

LSM KV Problems

- Write amplification.

- High latency because of compaction.

Write Amplification (WA)

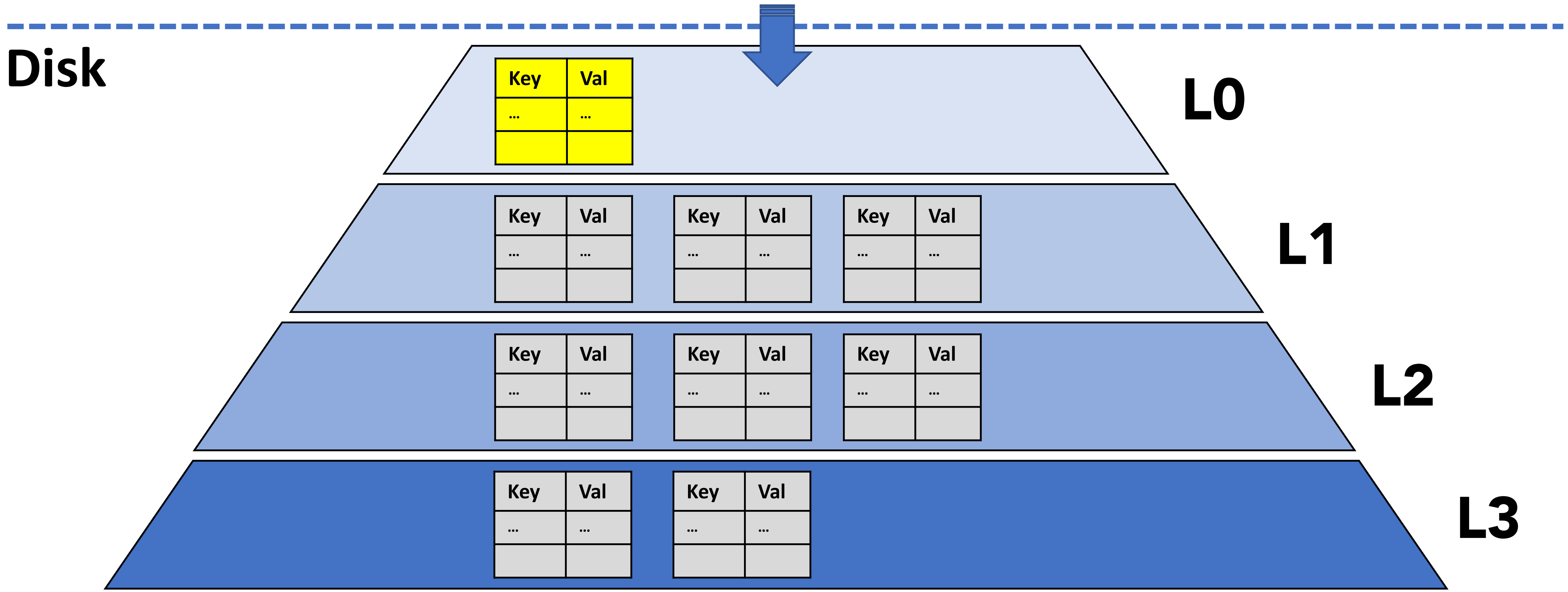
$$WA = \frac{\text{total data written to storage}}{\text{data written by app}}$$

Solutions

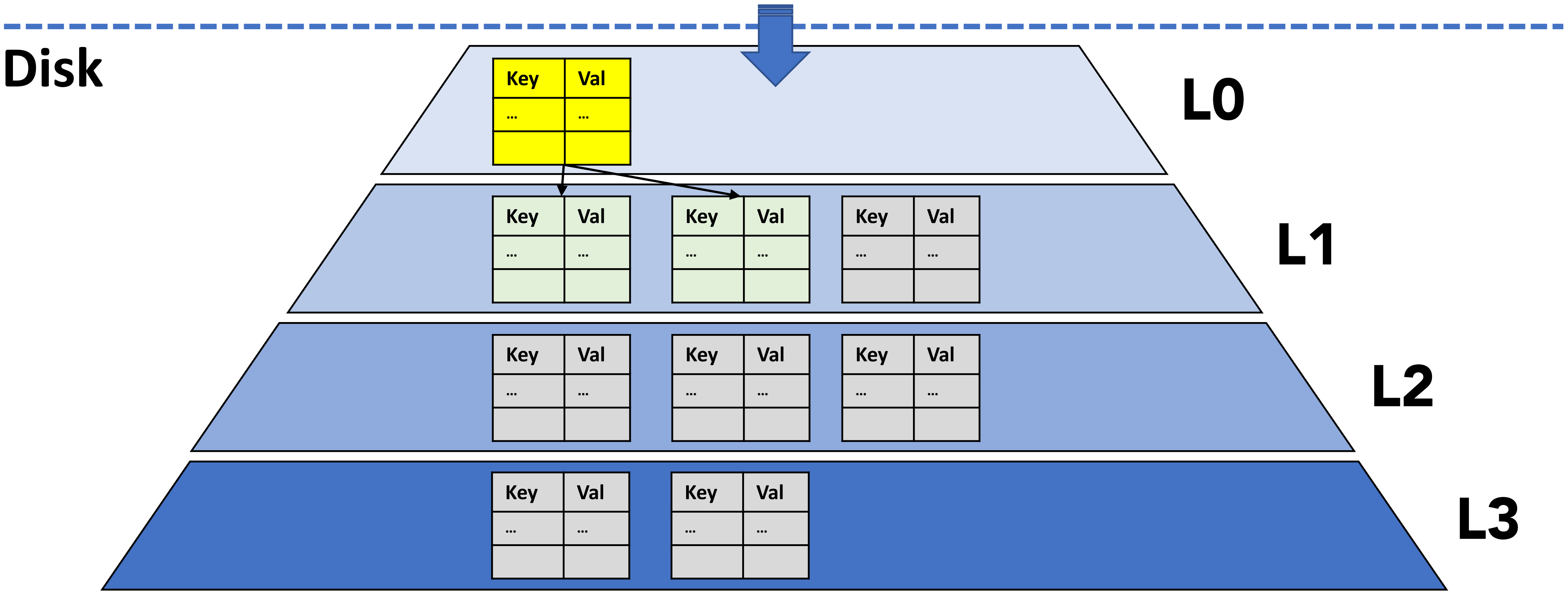
- Write amplification.
  - Cache hot keys

- High latency because of compaction.
  - Compaction scheduling

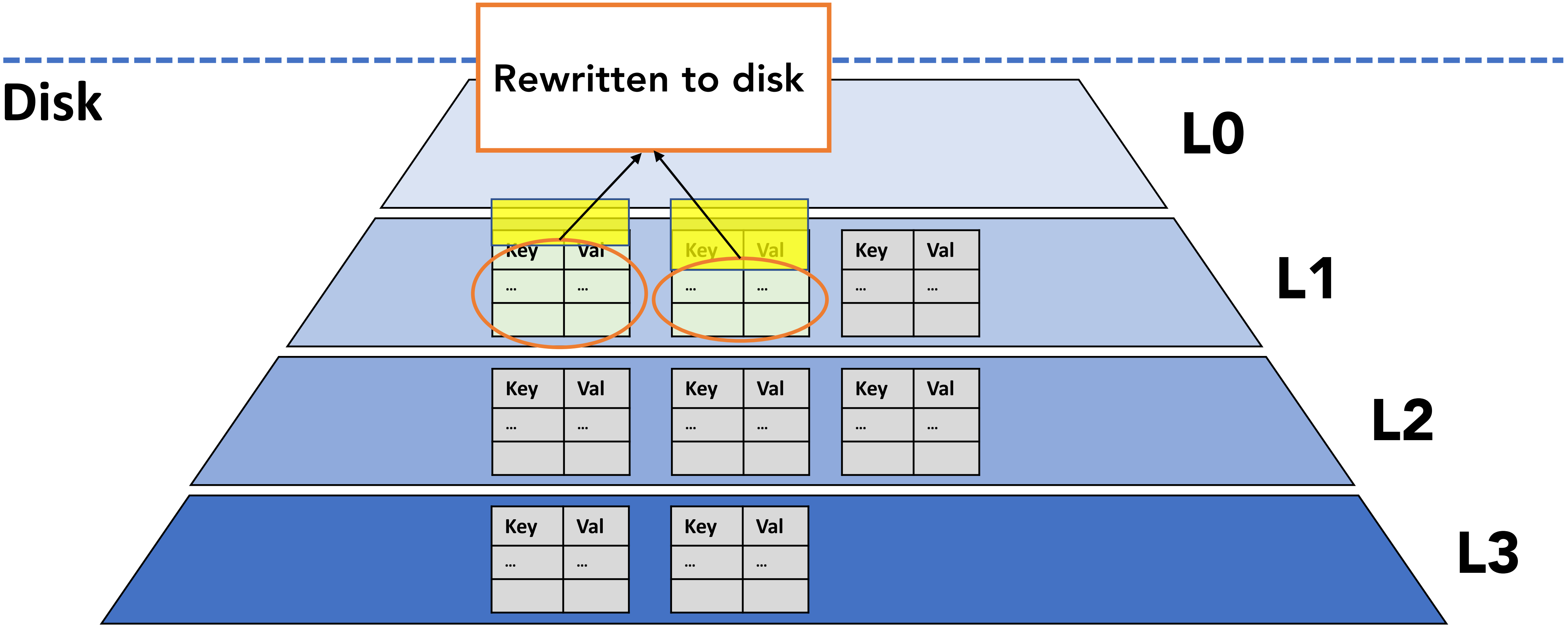
LSM Housekeeping: Compaction & WA



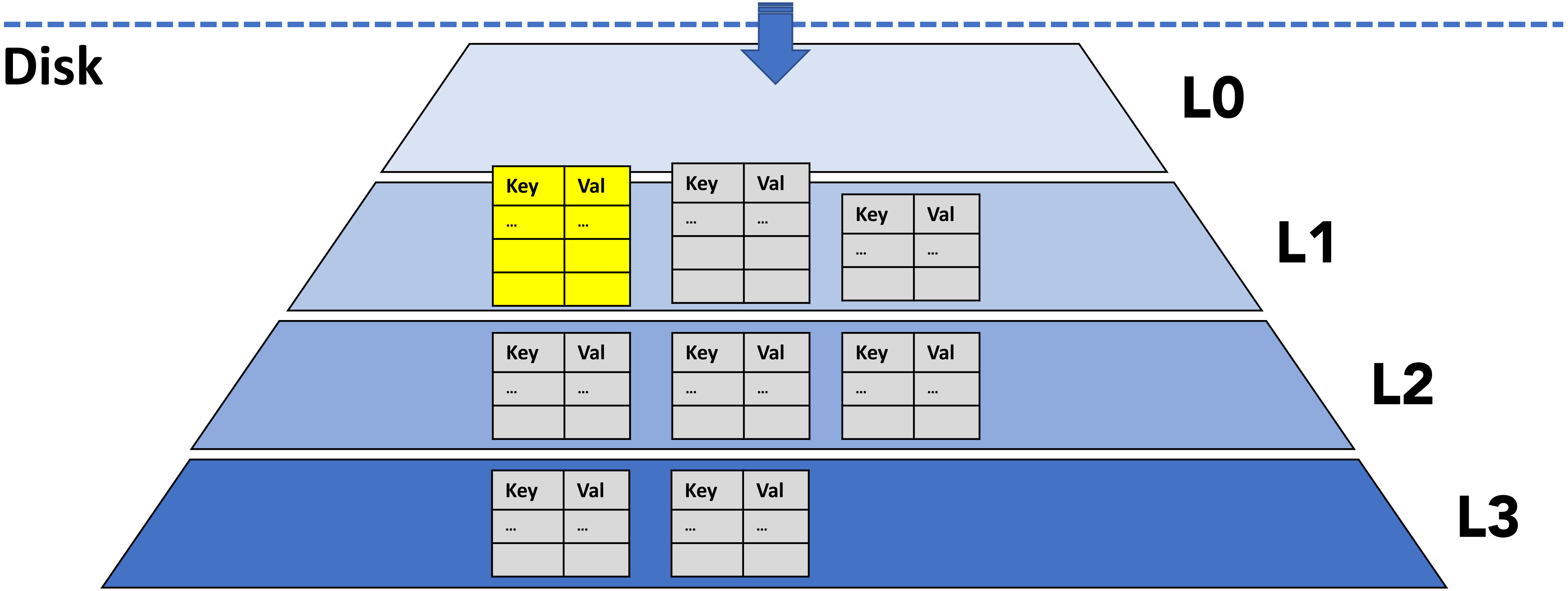
LSM Housekeeping: Compaction & WA



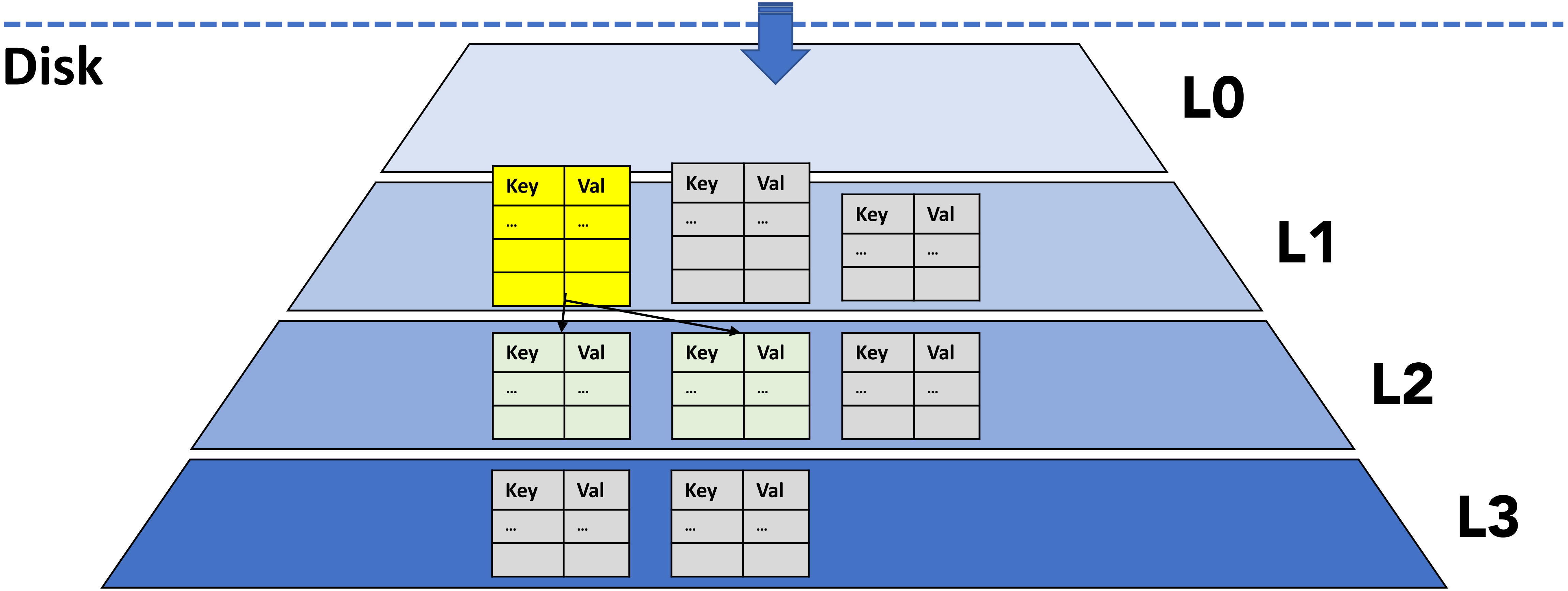
LSM Housekeeping: Compaction & WA



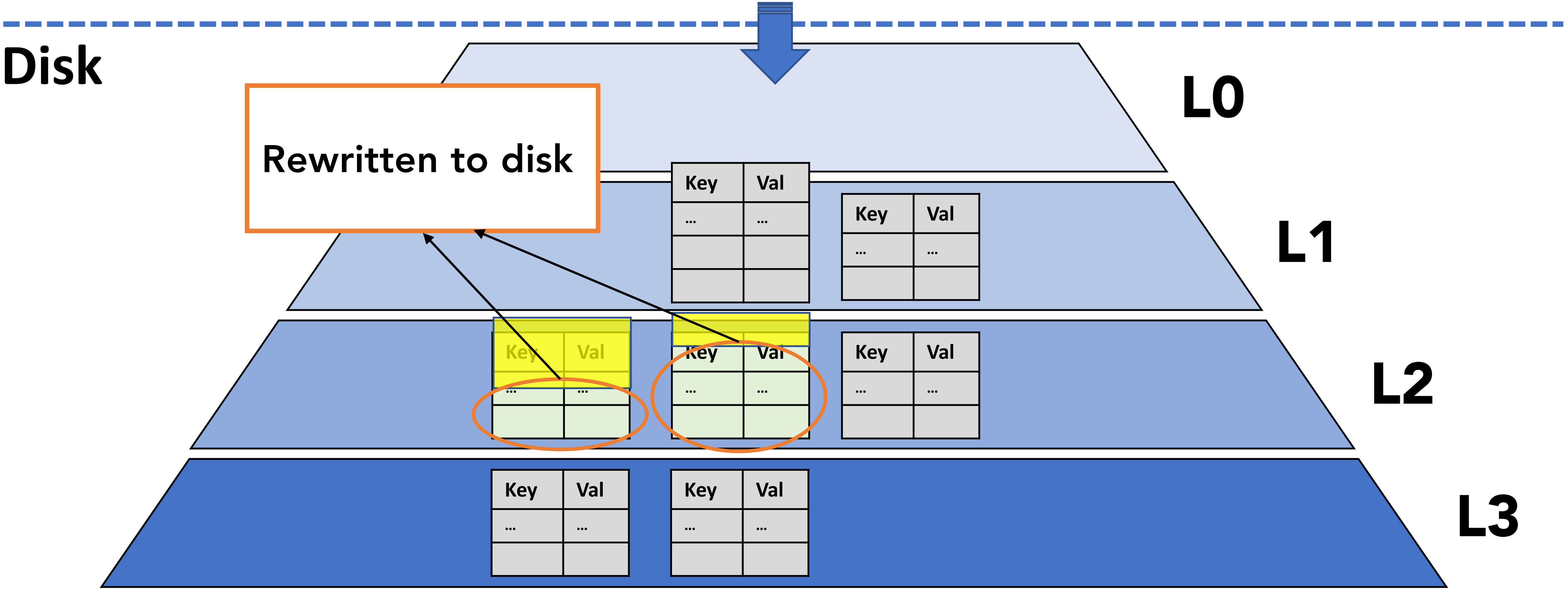
LSM Housekeeping: Compaction & WA



LSM Housekeeping: Compaction & WA

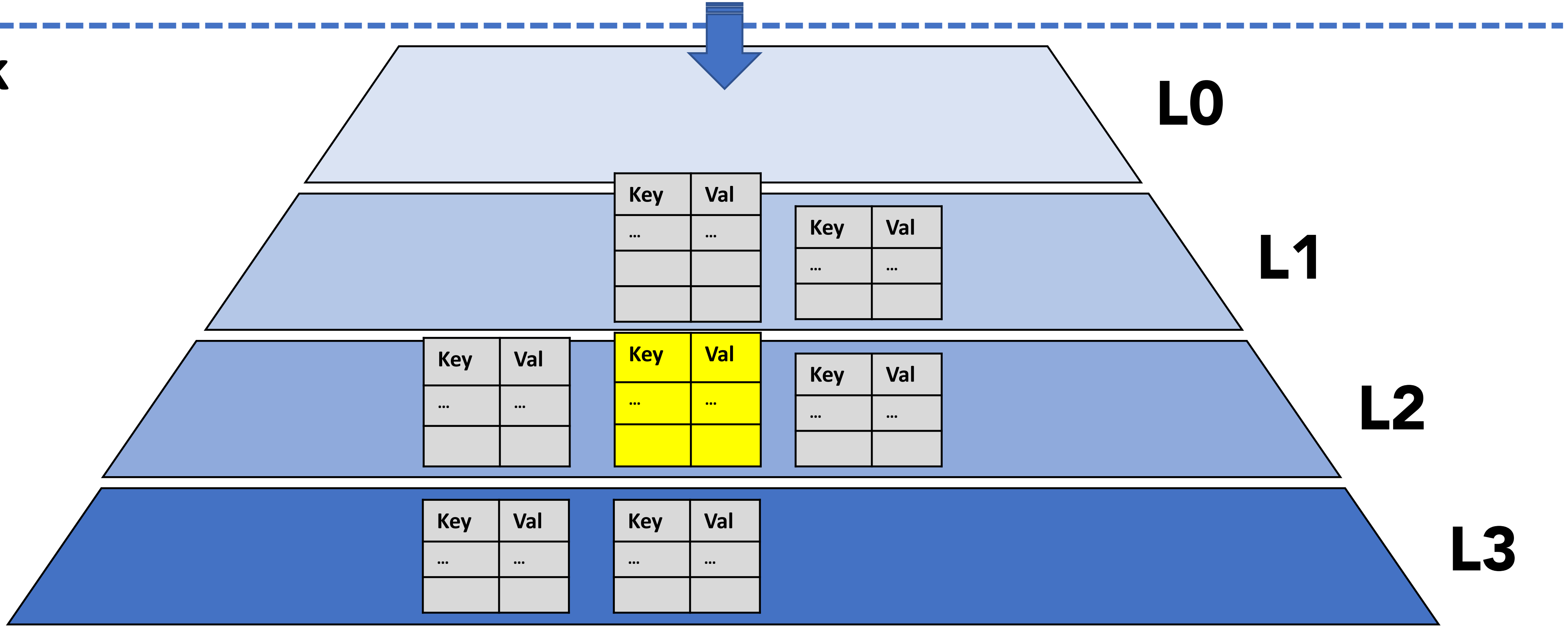


LSM Housekeeping: Compaction & WA

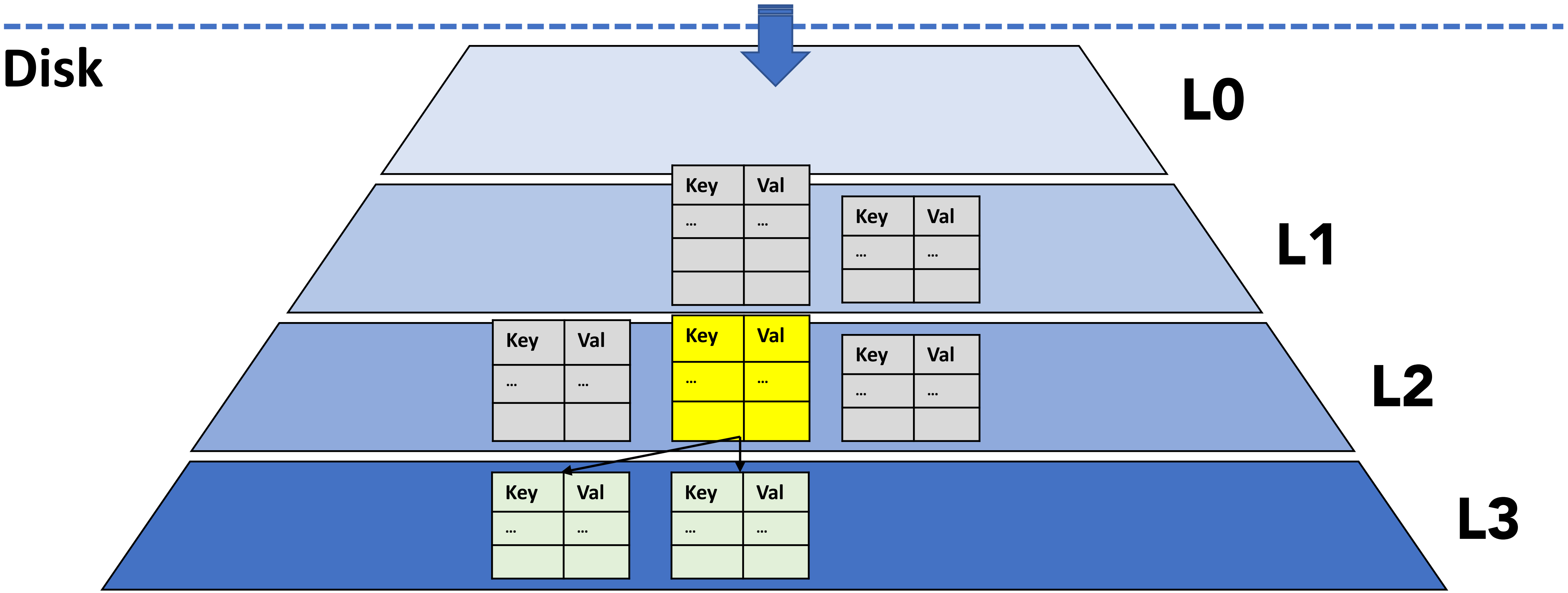


LSM Housekeeping: Compaction & WA

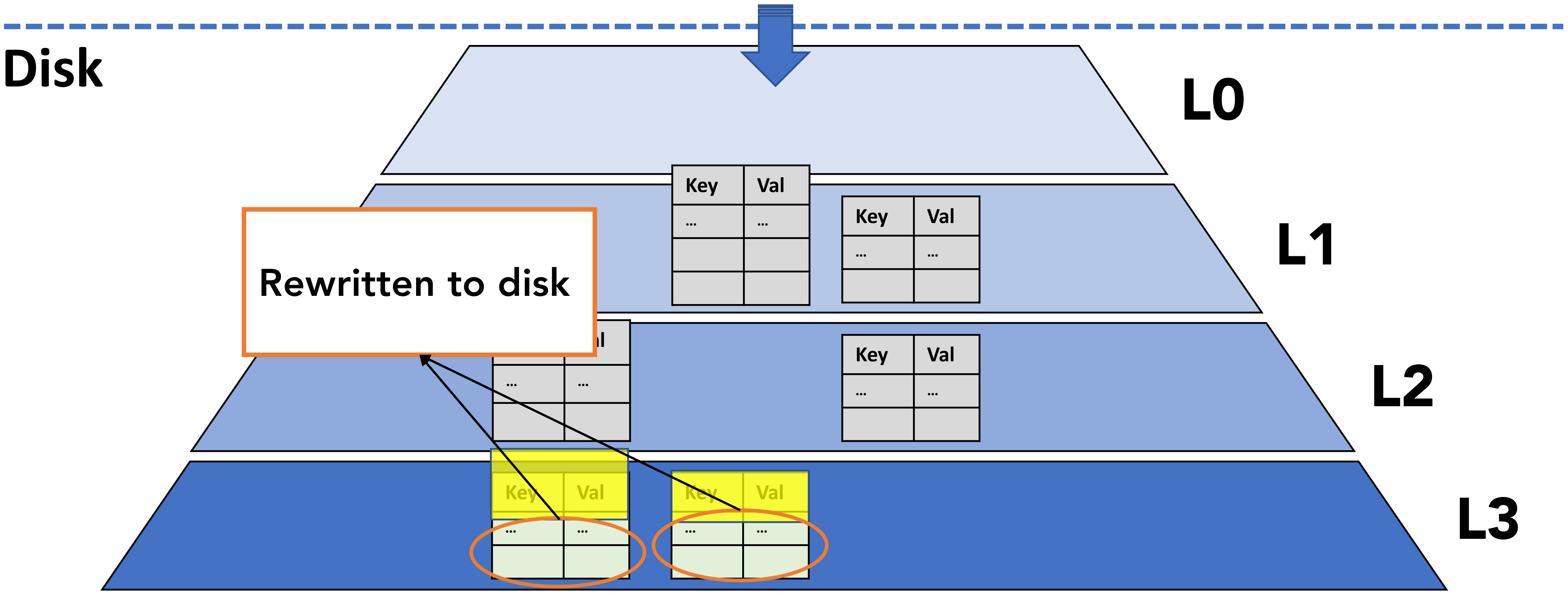
Disk



LSM Housekeeping: Compaction & WA

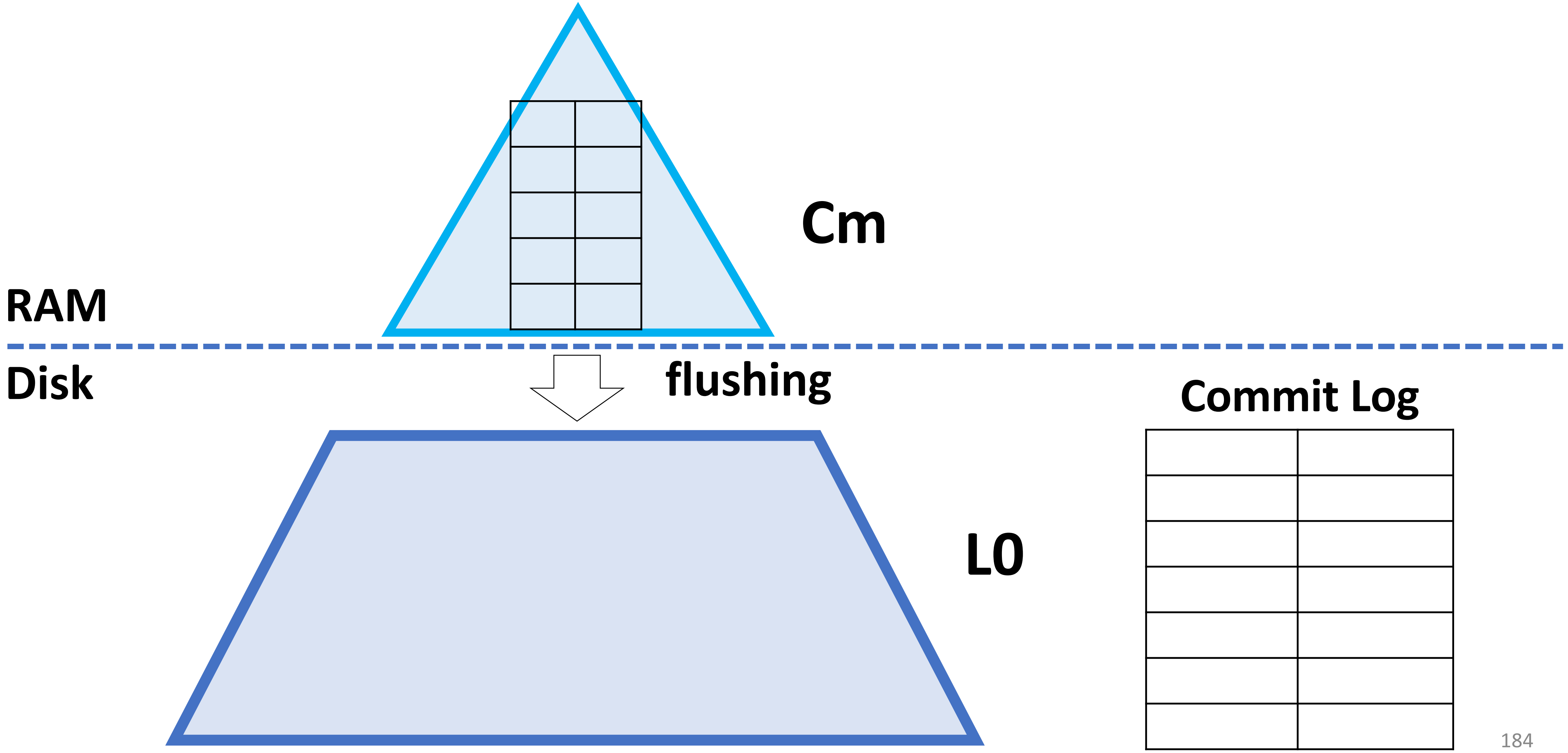


LSM Housekeeping: Compaction & WA

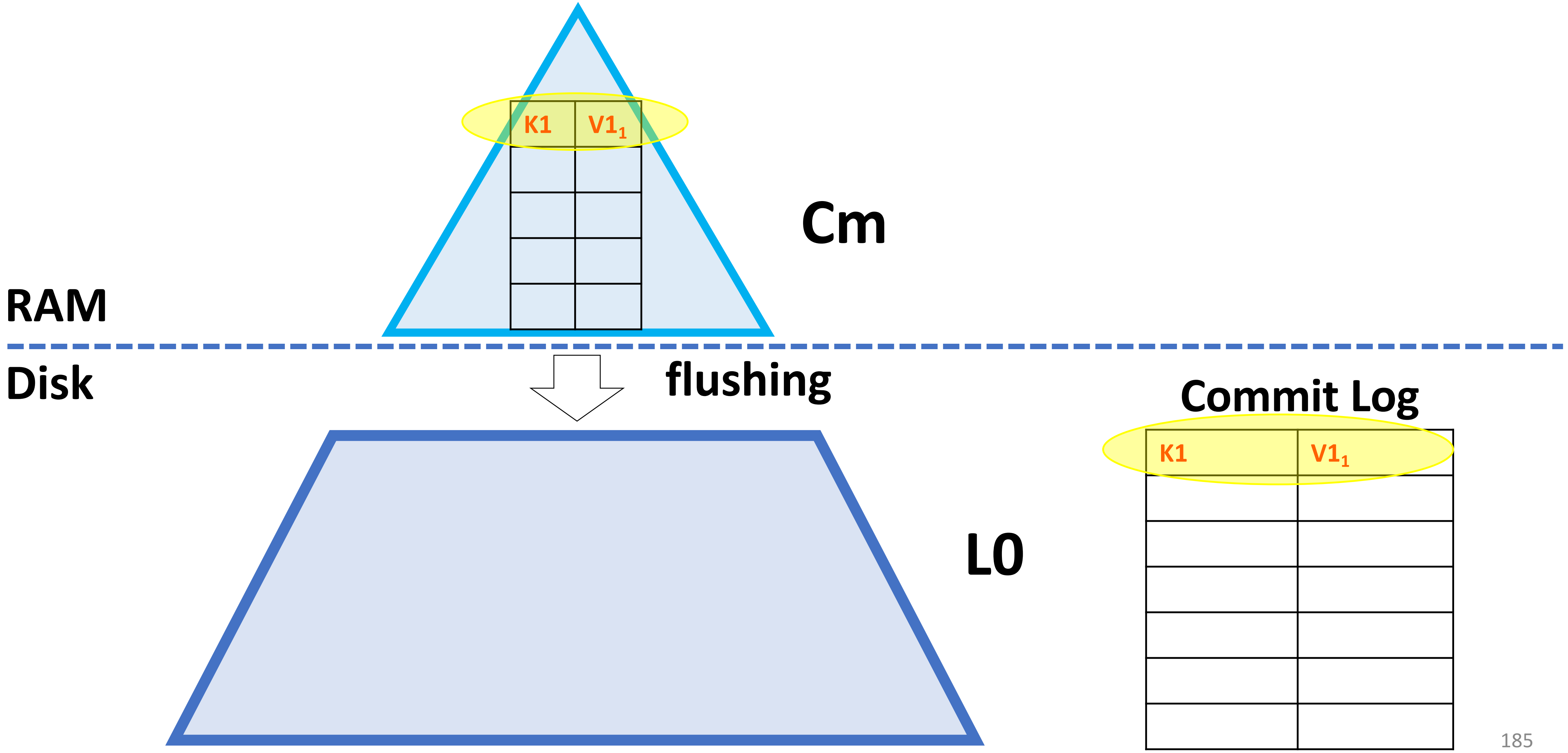


Approach 1: Cache hot keys

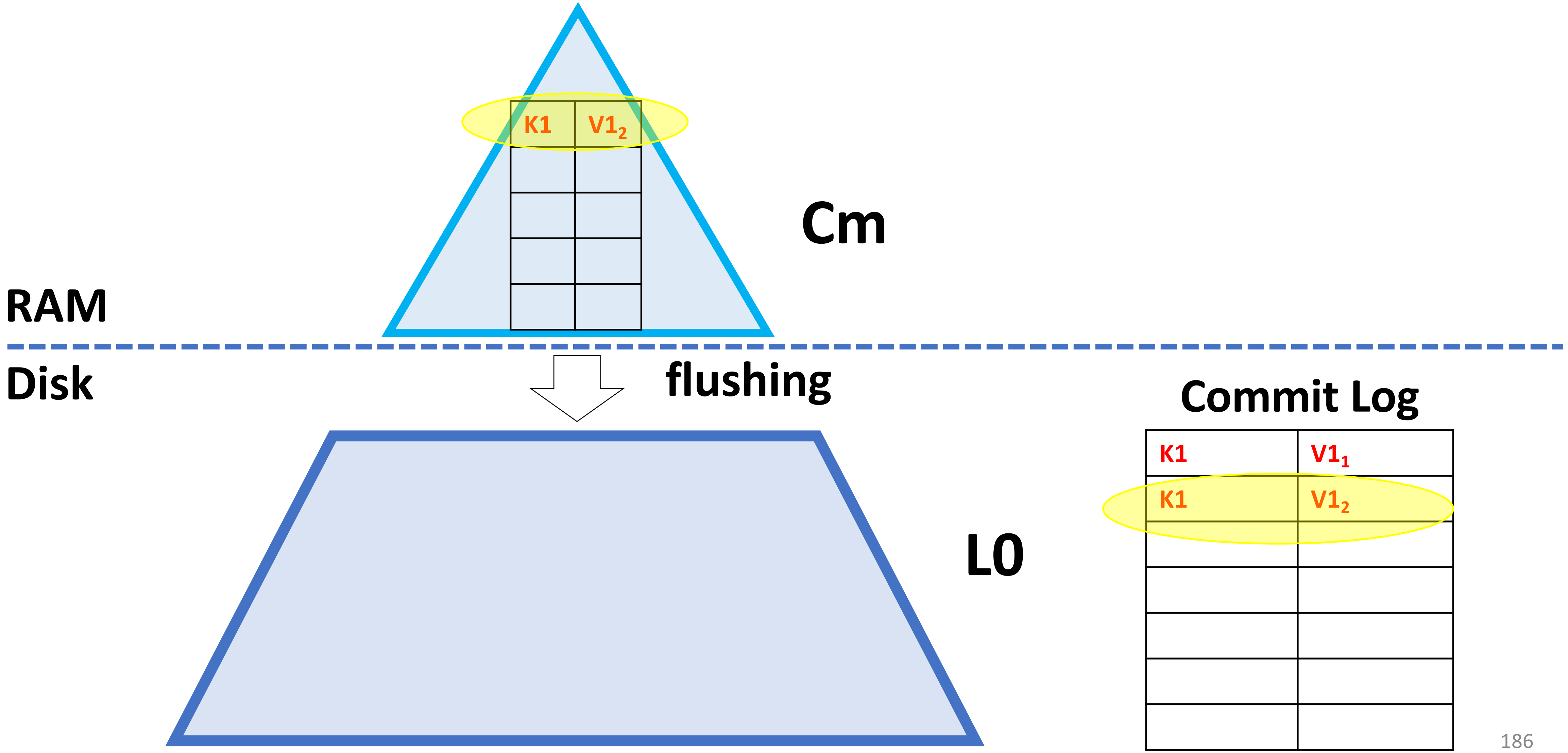
Problem: Flushing with Skewed Workloads



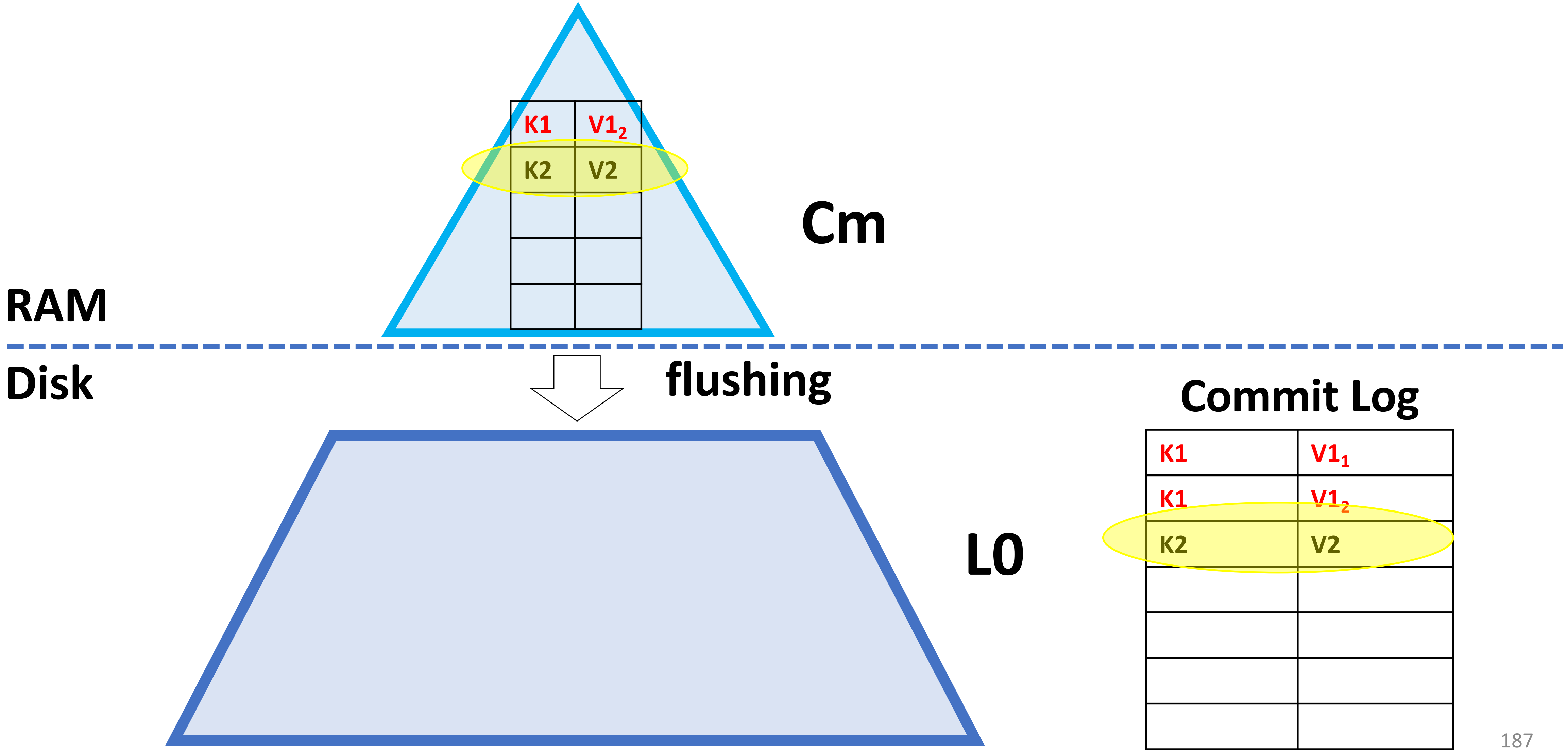
Problem: Flushing with Skewed Workloads



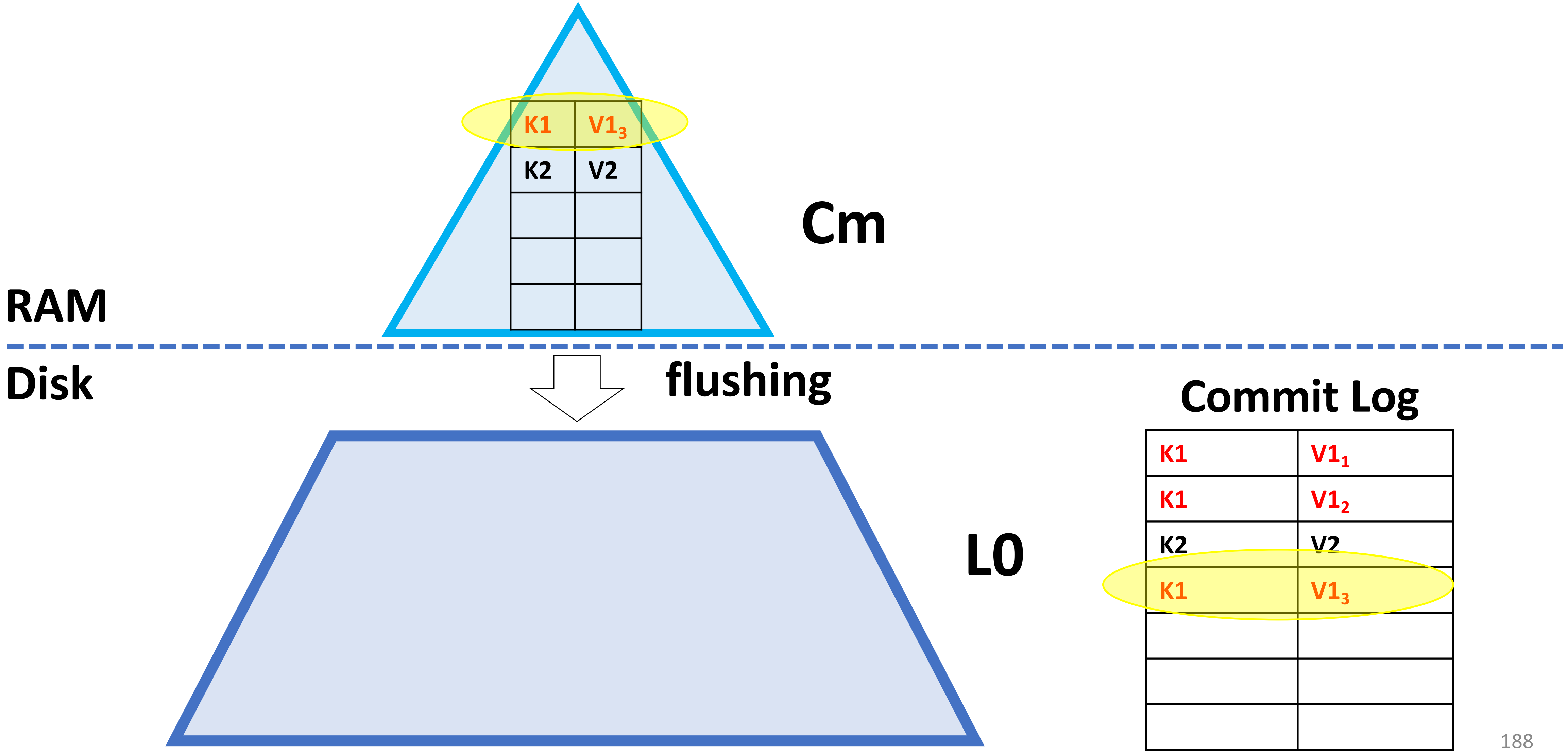
Problem: Flushing with Skewed Workloads



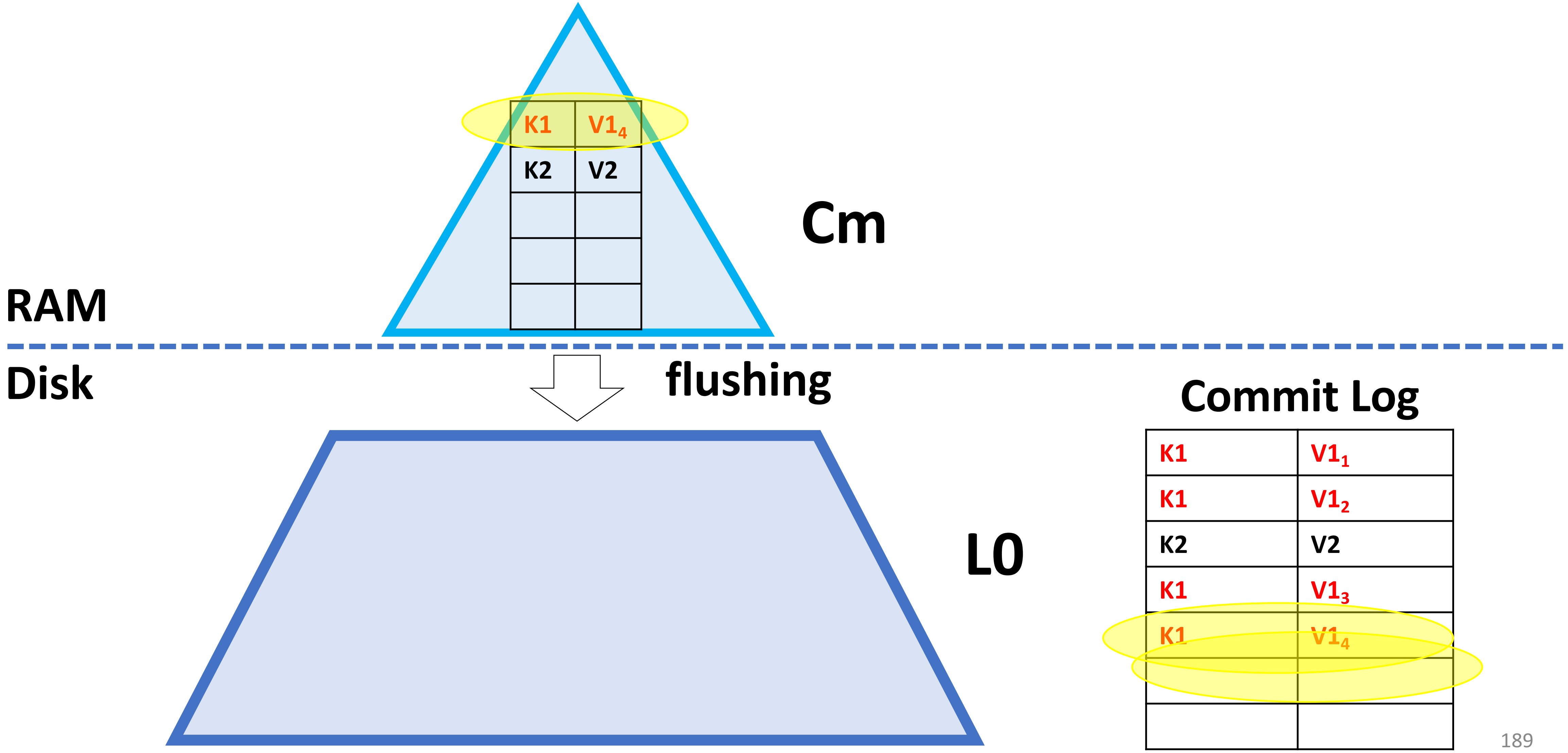
Problem: Flushing with Skewed Workloads



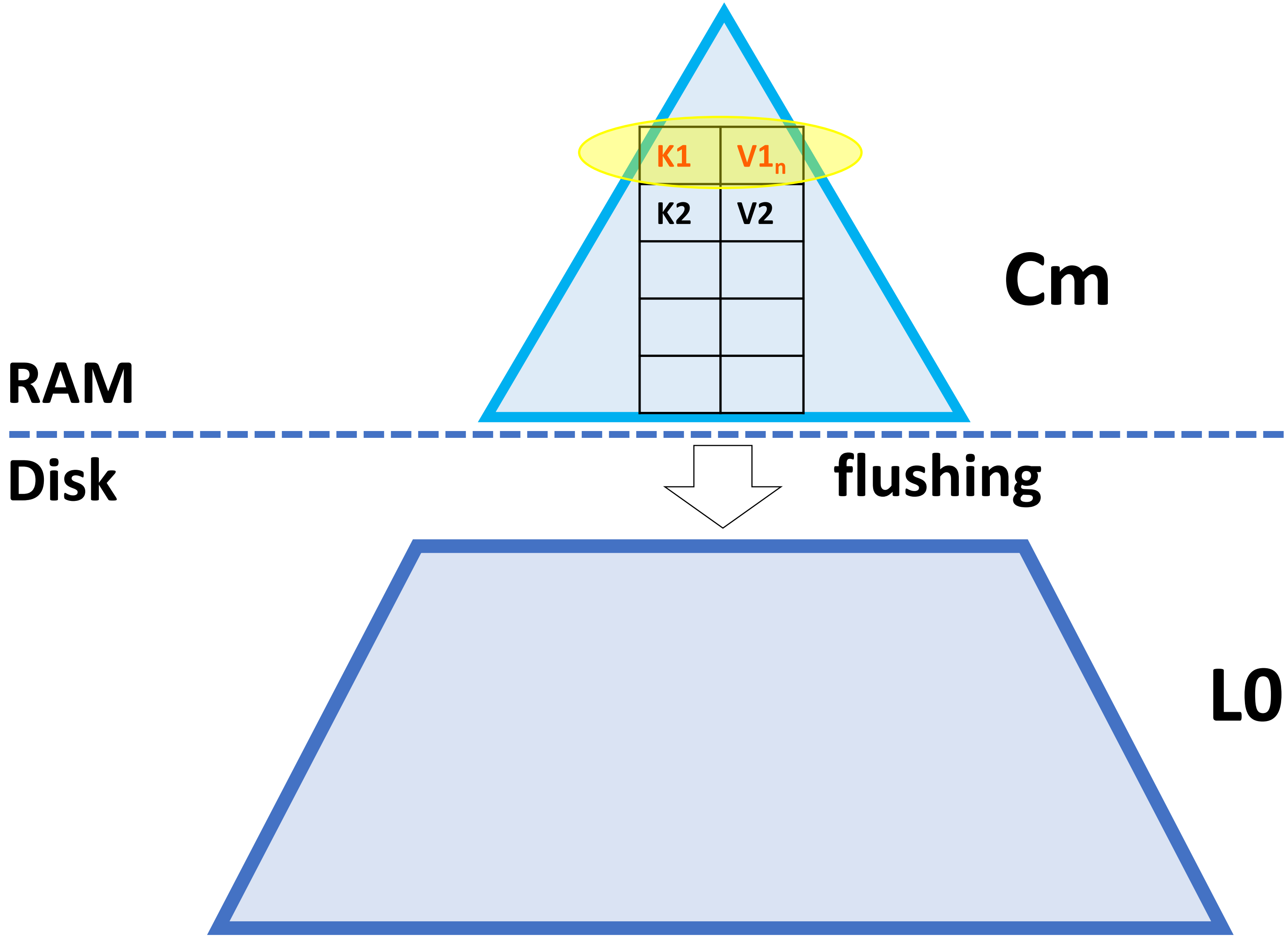
Problem: Flushing with Skewed Workloads



Problem: Flushing with Skewed Workloads



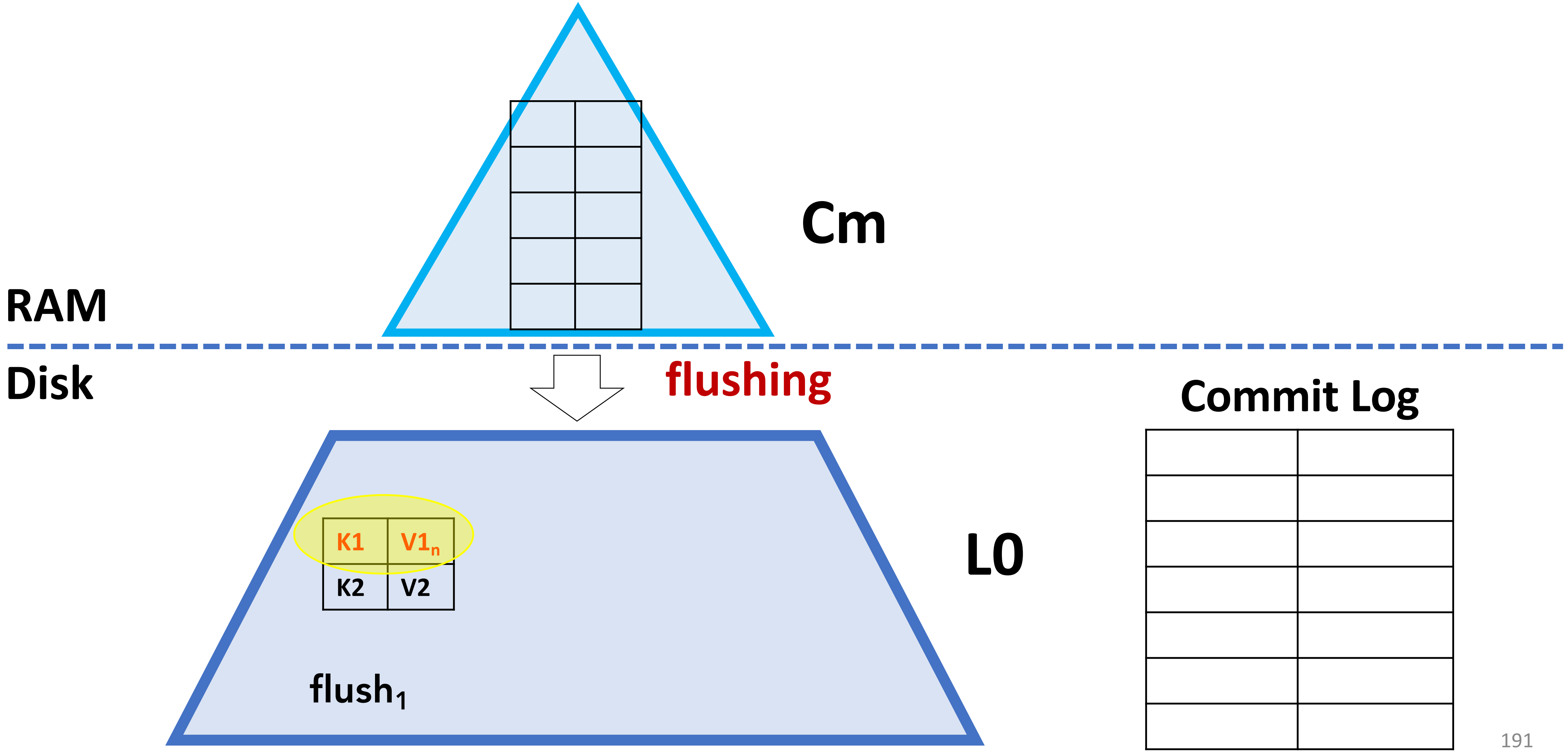
Problem: Flushing with Skewed Workloads



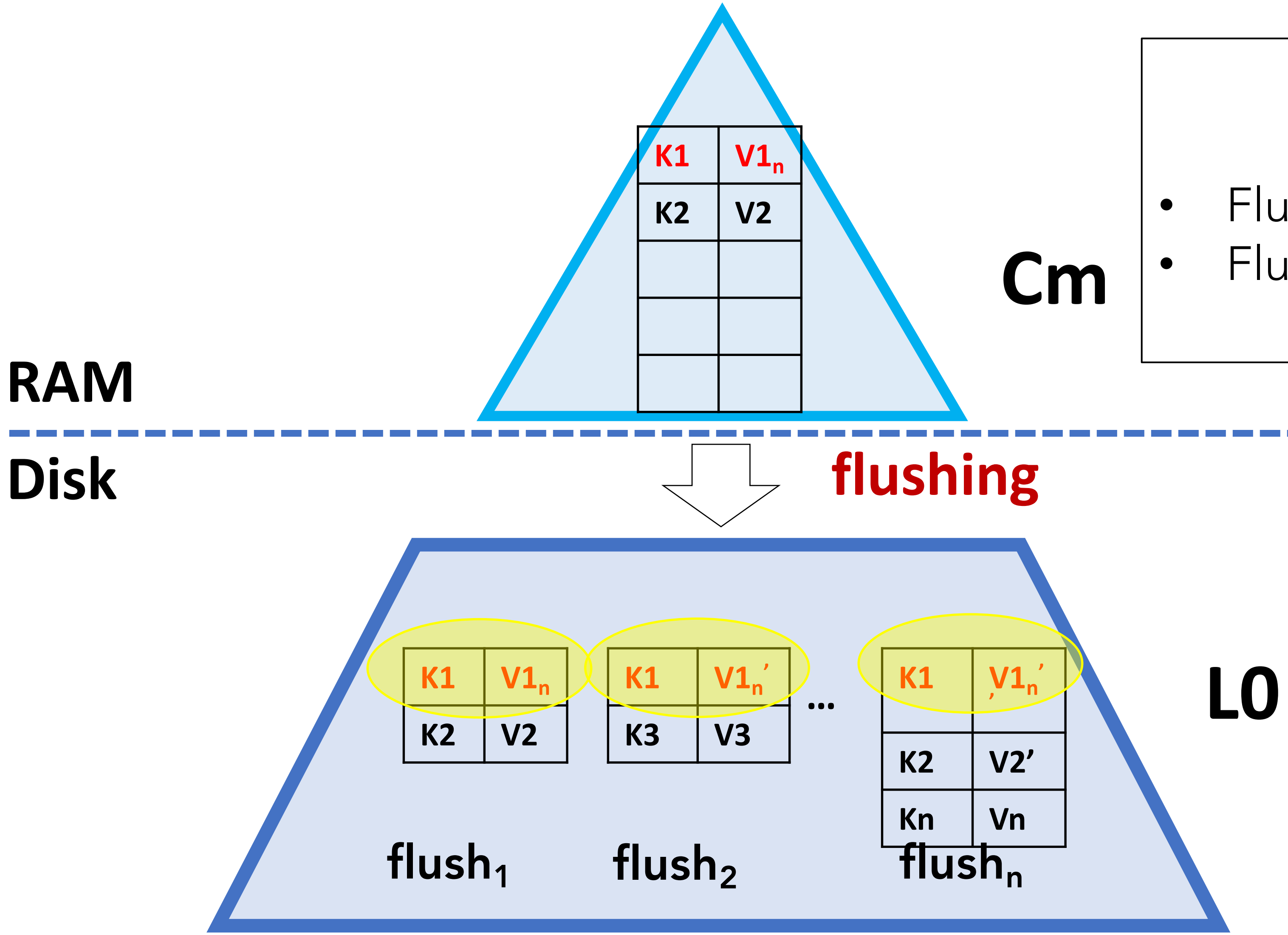
Commit Log

K1	$V1_1$
K1	$V1_2$
K2	V2
K1	$V1_3$
K1	$V1_4$
…	
K1	$V1_n$

Problem: Flushing with Skewed Workloads



Problem: Flushing with Skewed Workloads



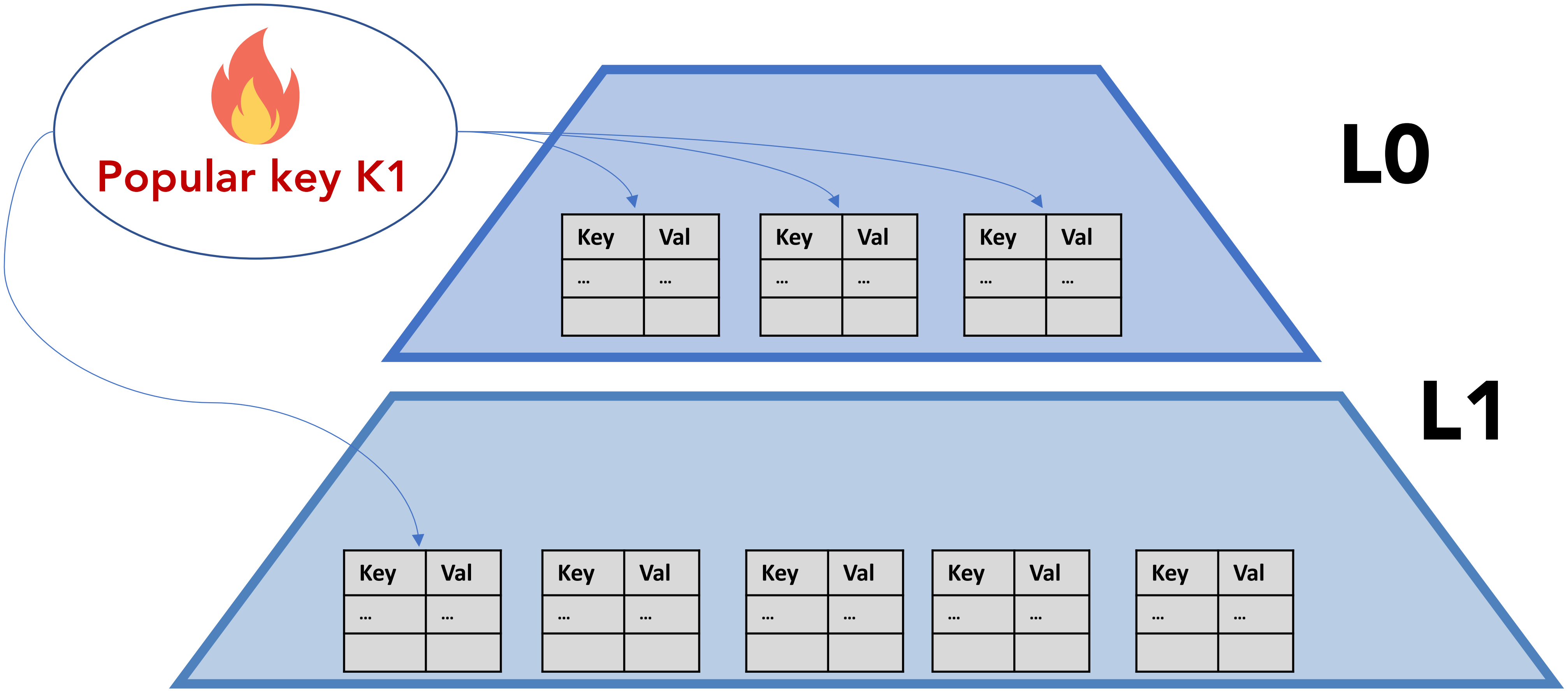
High data skew

- Flush because commit log is full
- Flush mostly empty mem comp

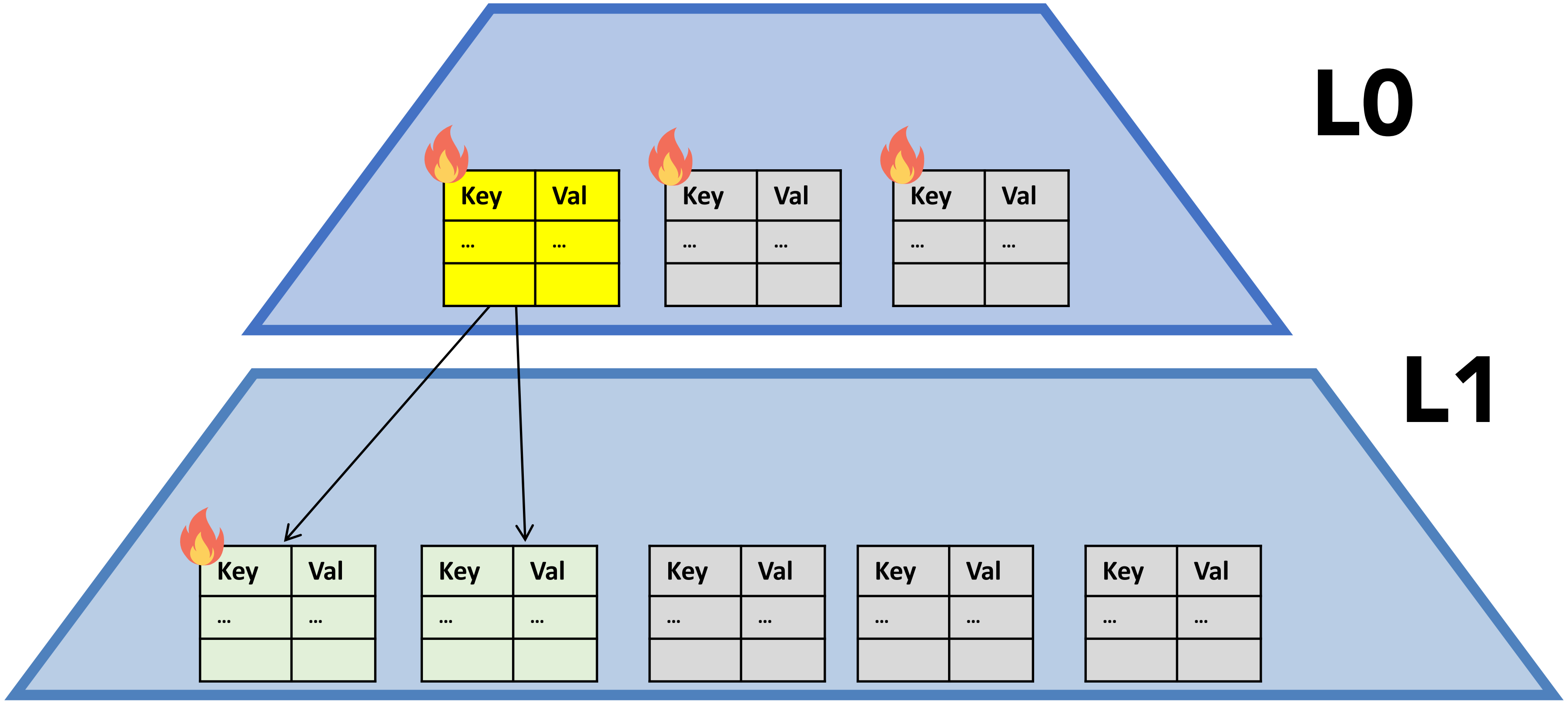
Commit Log

K1	$V1_1$
K2	V2
K1	$V1_2$
K1	$V1_3$
K1	$V1_4$
…	
K1	$V1_n$

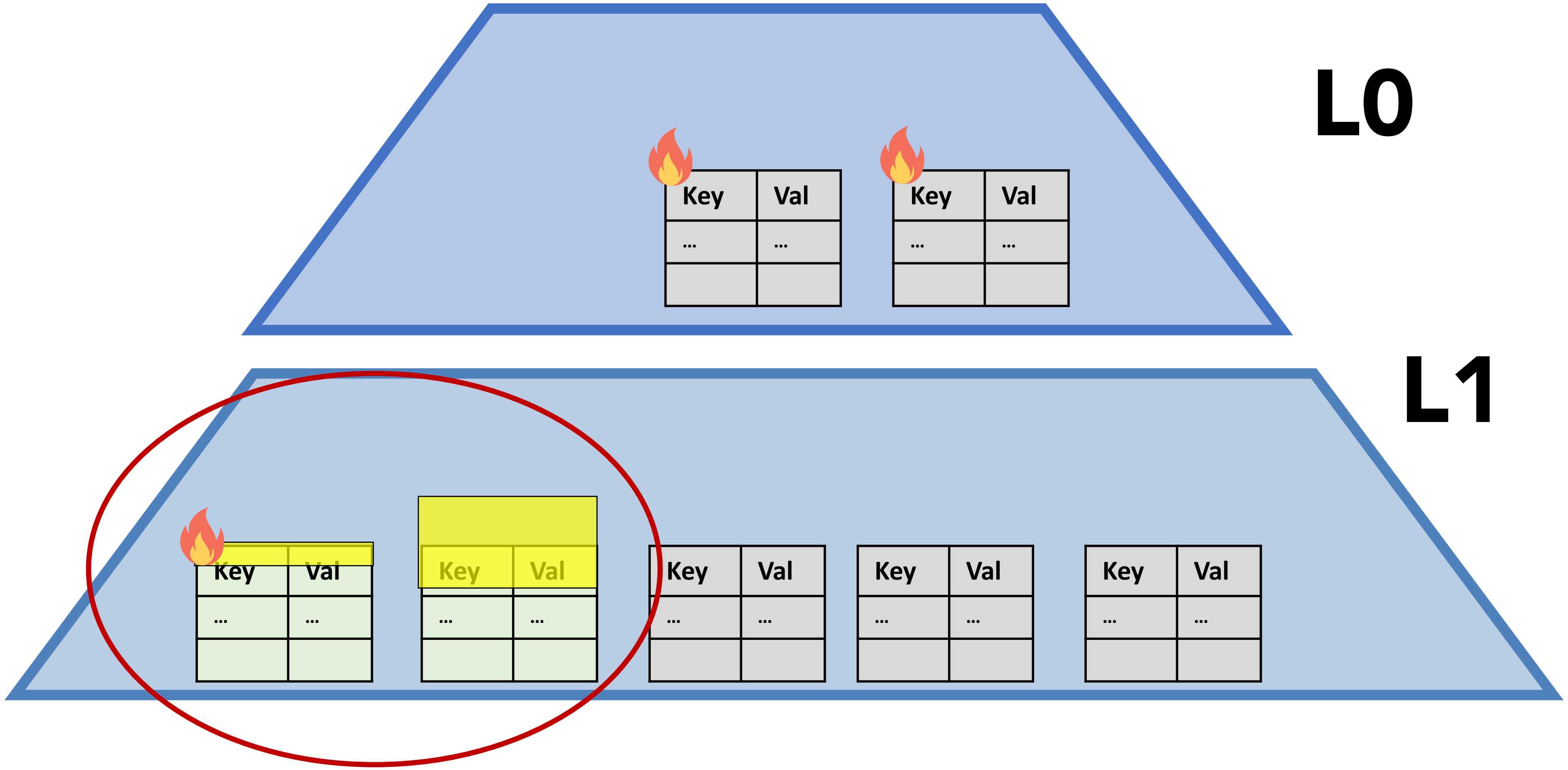
Problem: Compaction with Skewed Workloads



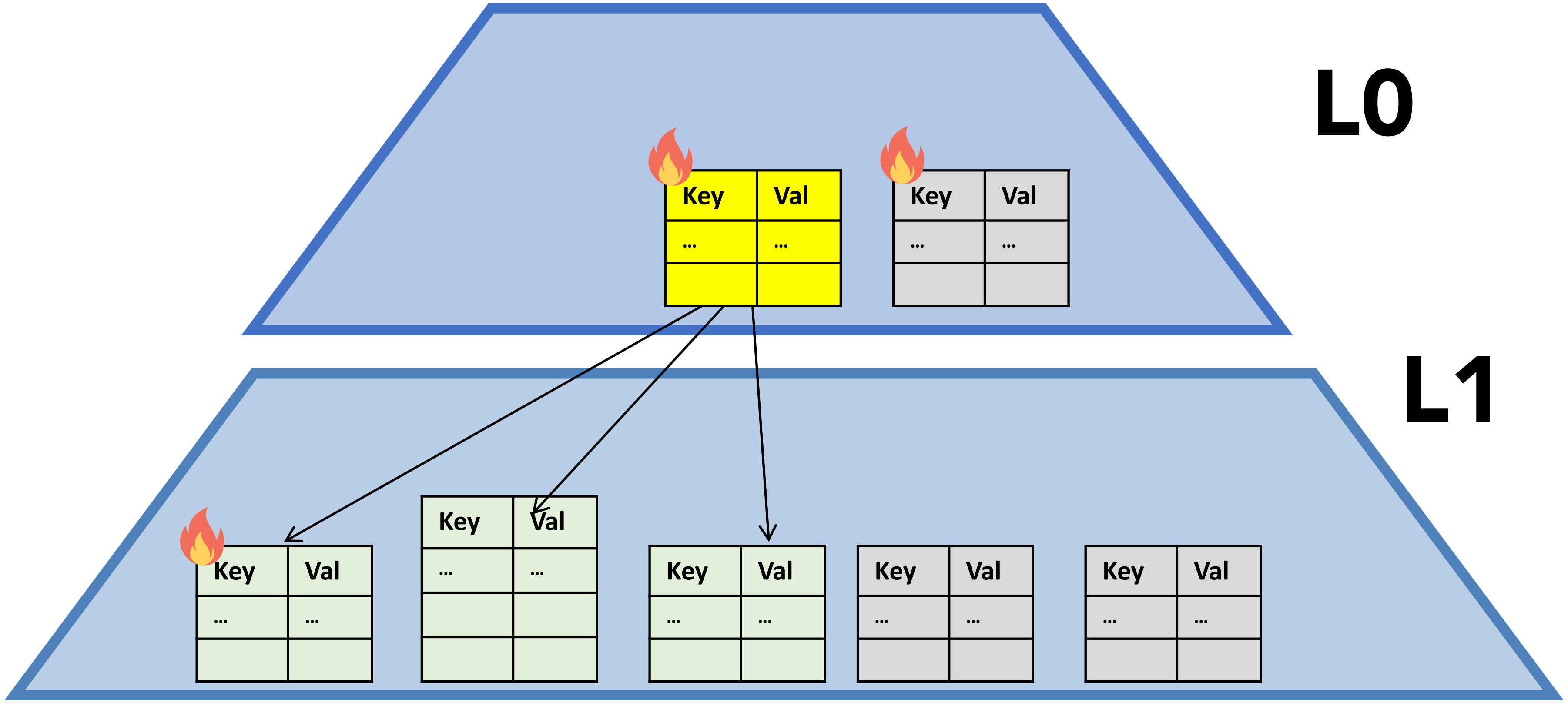
Problem: Compaction with Skewed Workloads



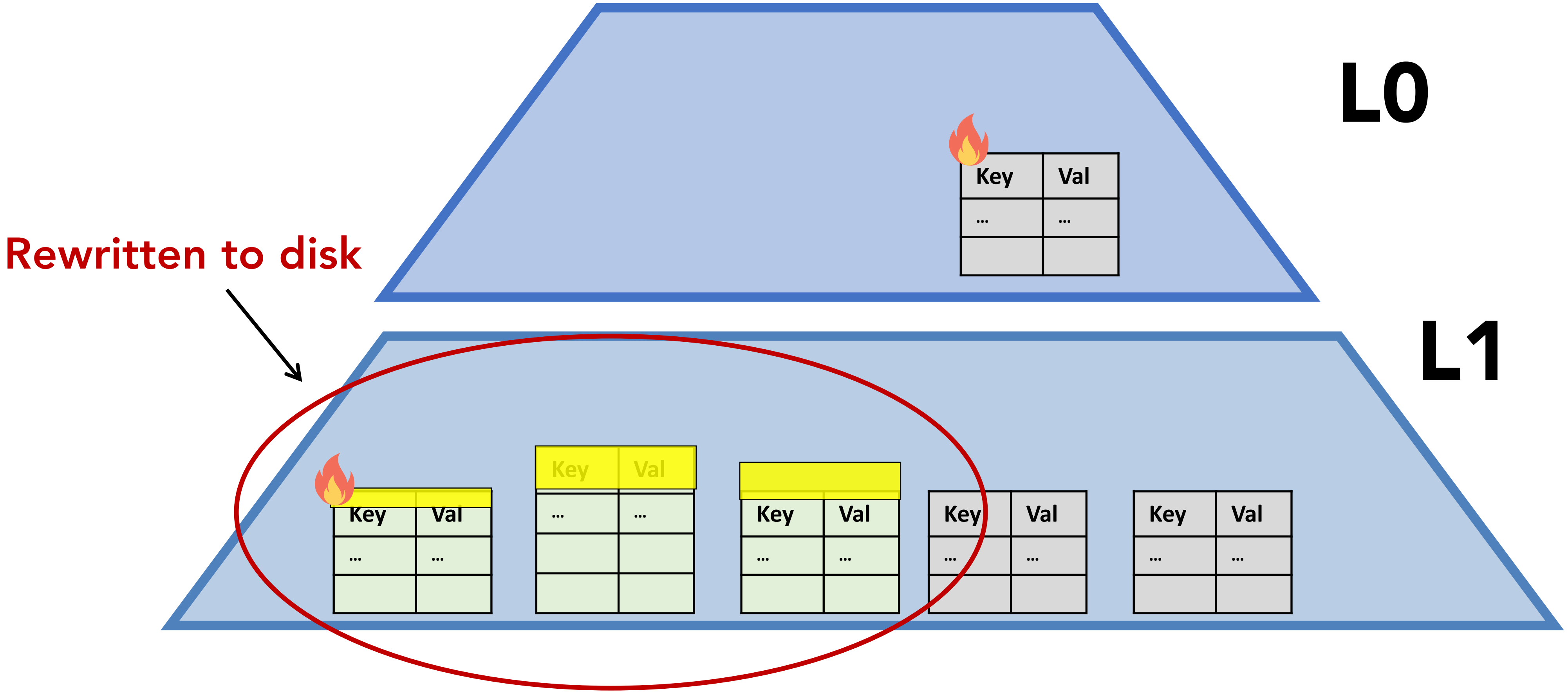
Problem: Compaction with Skewed Workloads



Problem: Compaction with Skewed Workloads

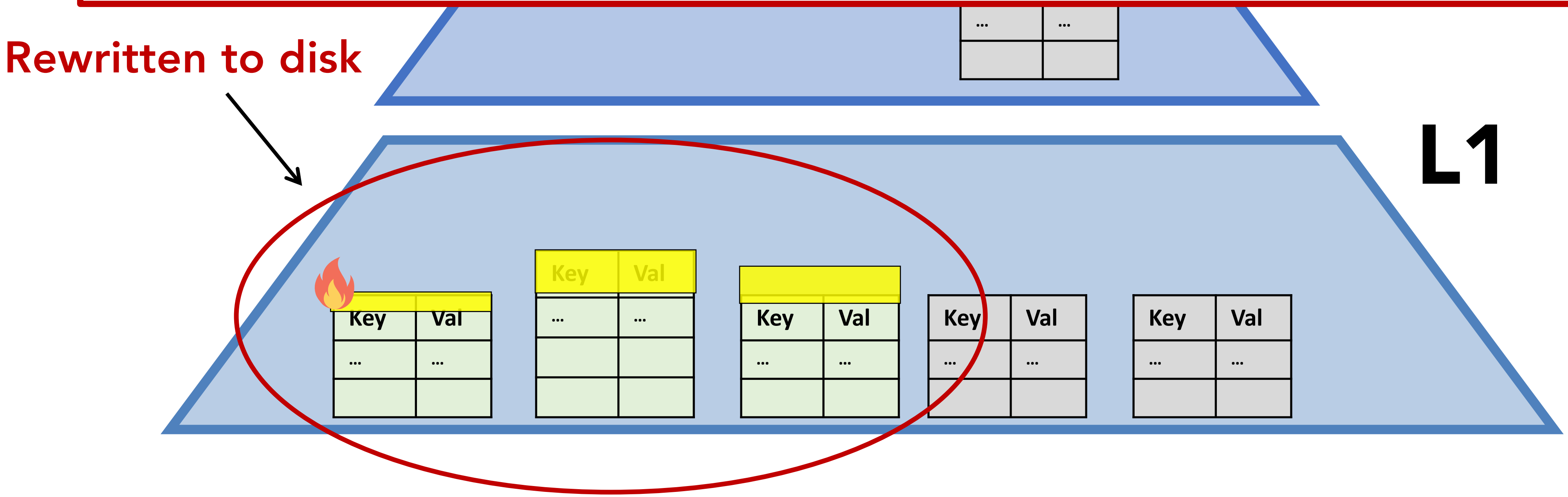


Problem: Compaction with Skewed Workloads

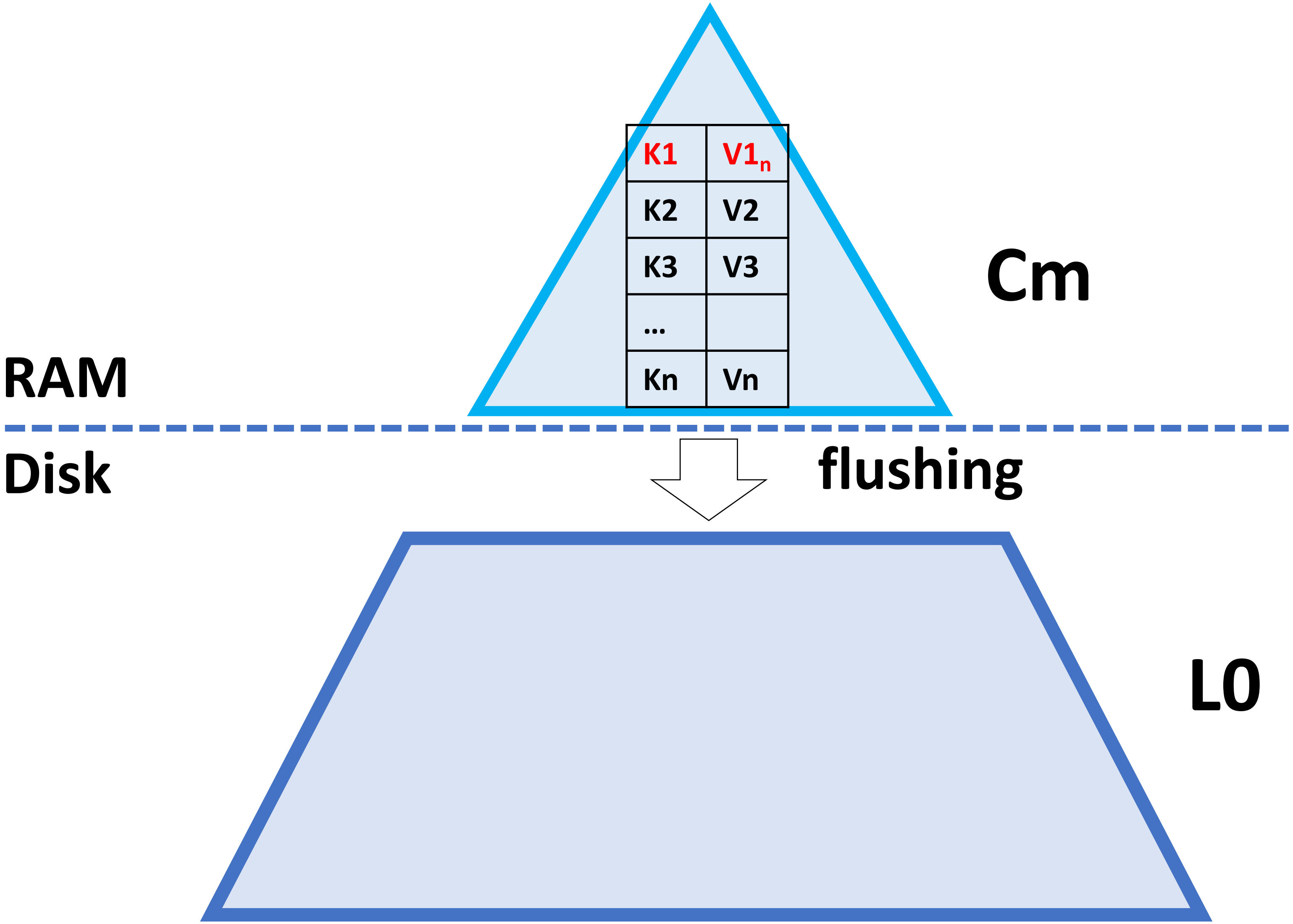


Problem: Compaction with Skewed Workloads

File on L1 **rewritten to disk twice** because of one key ☹



Solution: Hot-cold key separation



Commit Log

K1	$V1_1$
K2	V2
K1	$V1_2$
K1	$V1_3$
K1	$V1_4$
...	
K1	$V1_n$

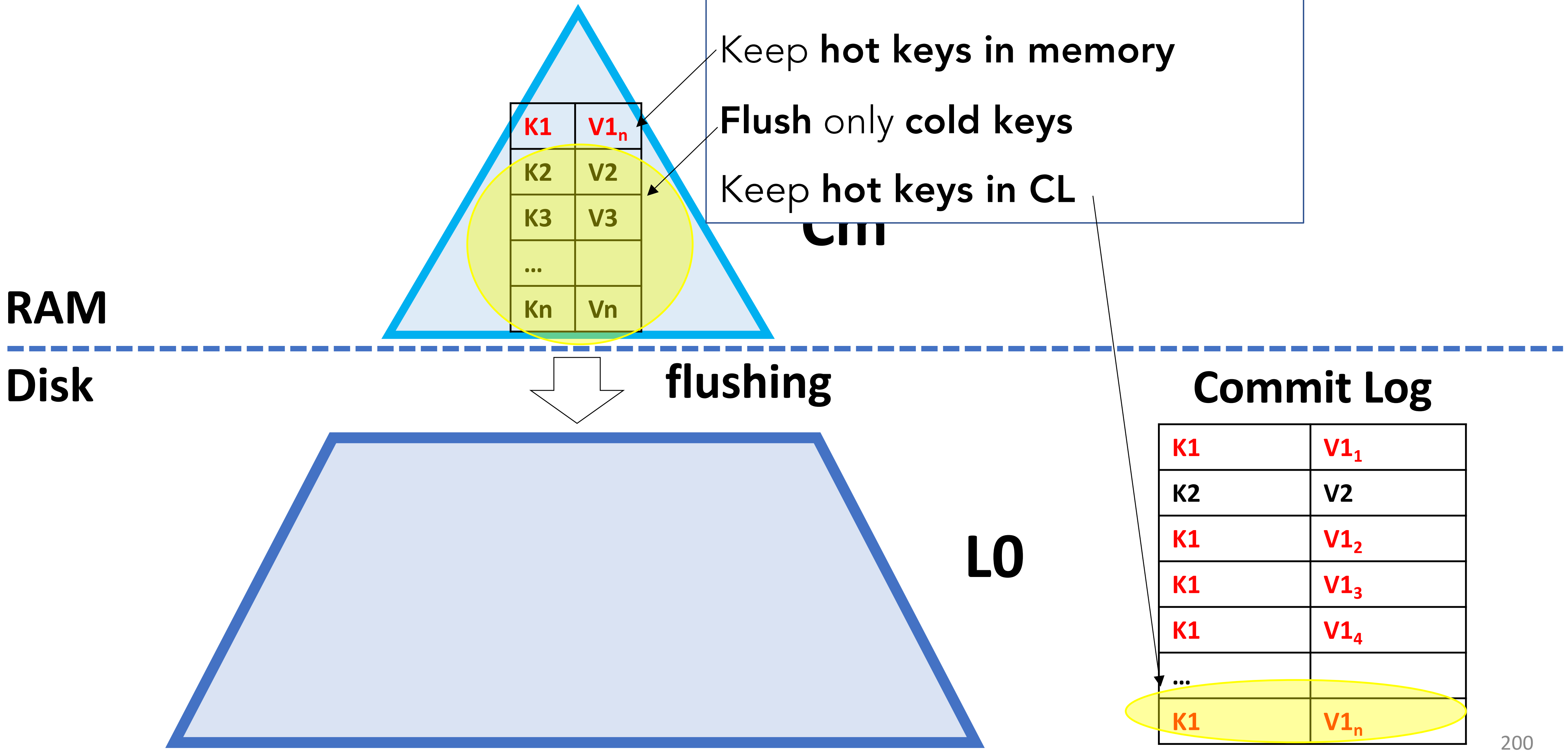
Solution: Hot-cold key separation

Idea:

Keep **hot keys in memory**

Flush only **cold keys**

Keep **hot keys in CL**



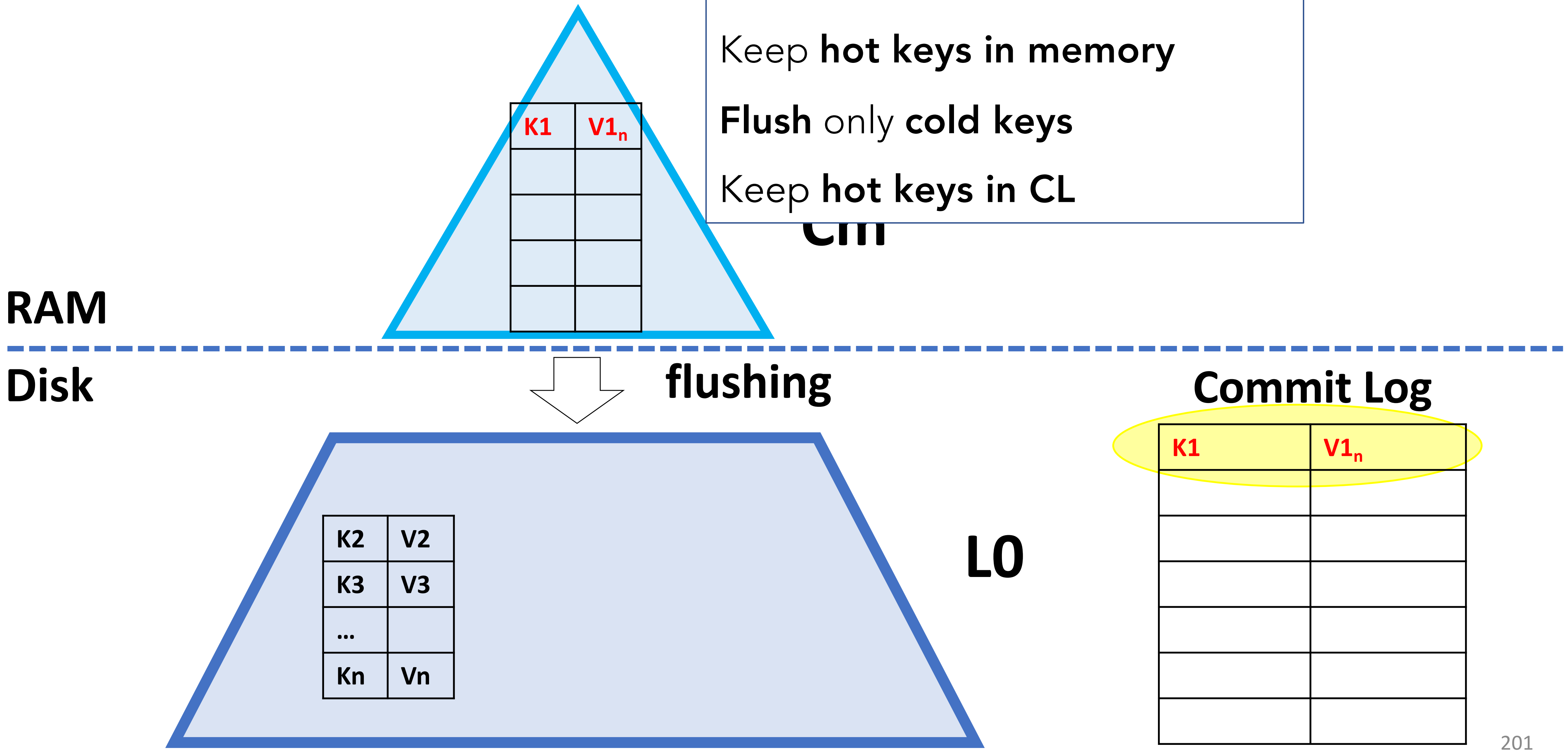
Solution: Hot-cold key separation

Idea:

Keep **hot keys in memory**

Flush only **cold keys**

Keep **hot keys in CL**



Hot-Cold Keys separation summary

✓ Good for skewed workloads.

✓ Reduce flushing WA: less data written from memory to disk.

✓ Reduce compaction WA: avoid repeatedly compacting hot keys.

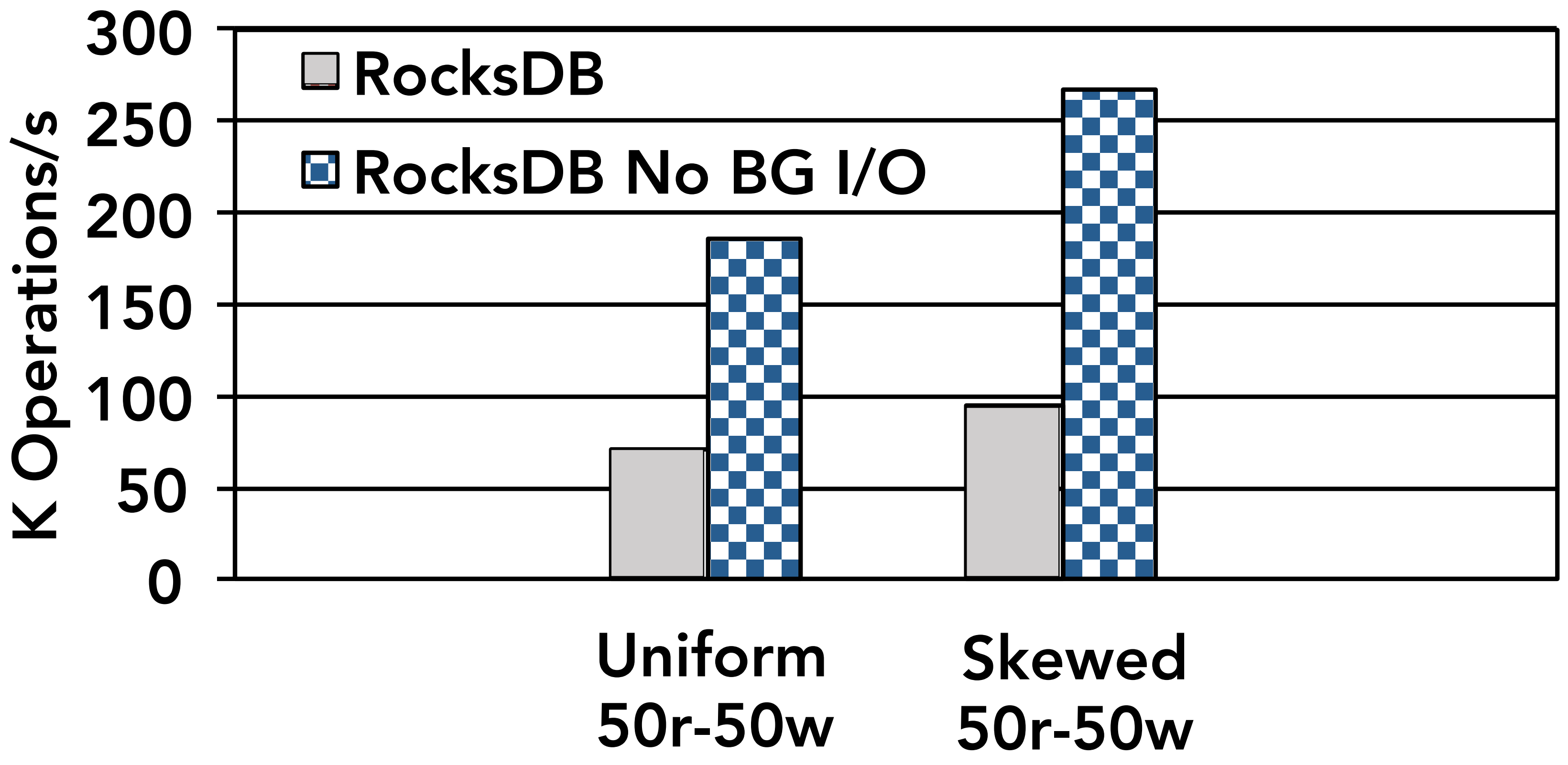
Solutions

- Write amplification.
  - Cache hot keys

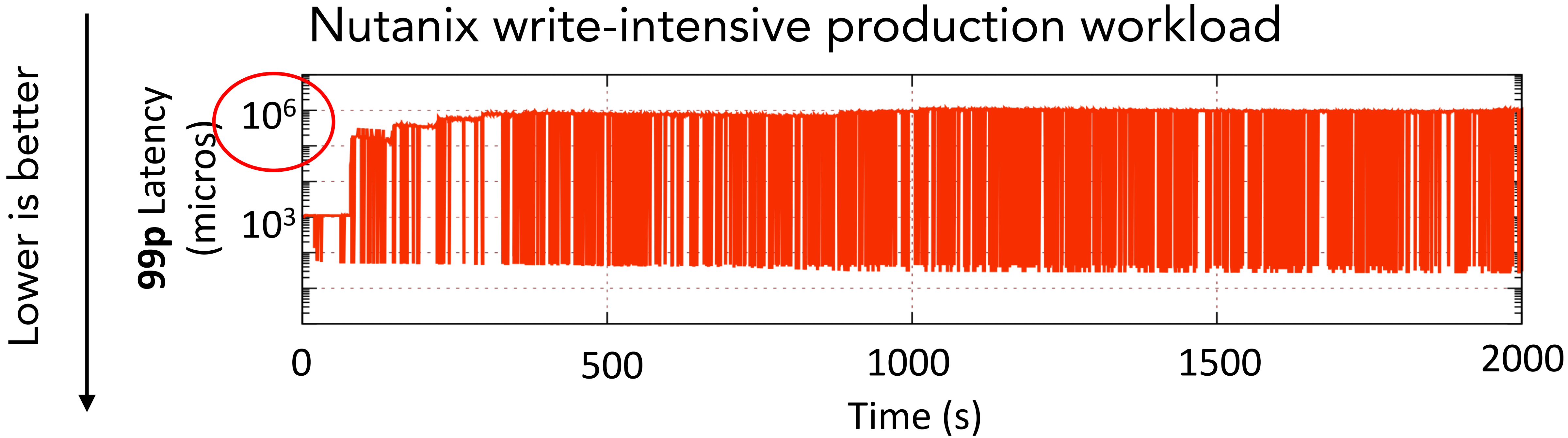
- **High latency because of compaction.**
  - Compaction scheduling

Background I/O Overhead

- **Long & slow bg. ops ⟶ slowdown of user ops.**



LSM KV Latency Spikes in RocksDB



Latency spikes of up to 1s in write dominated workloads!
Spikes are up to 3 orders of magnitude > median tail latency

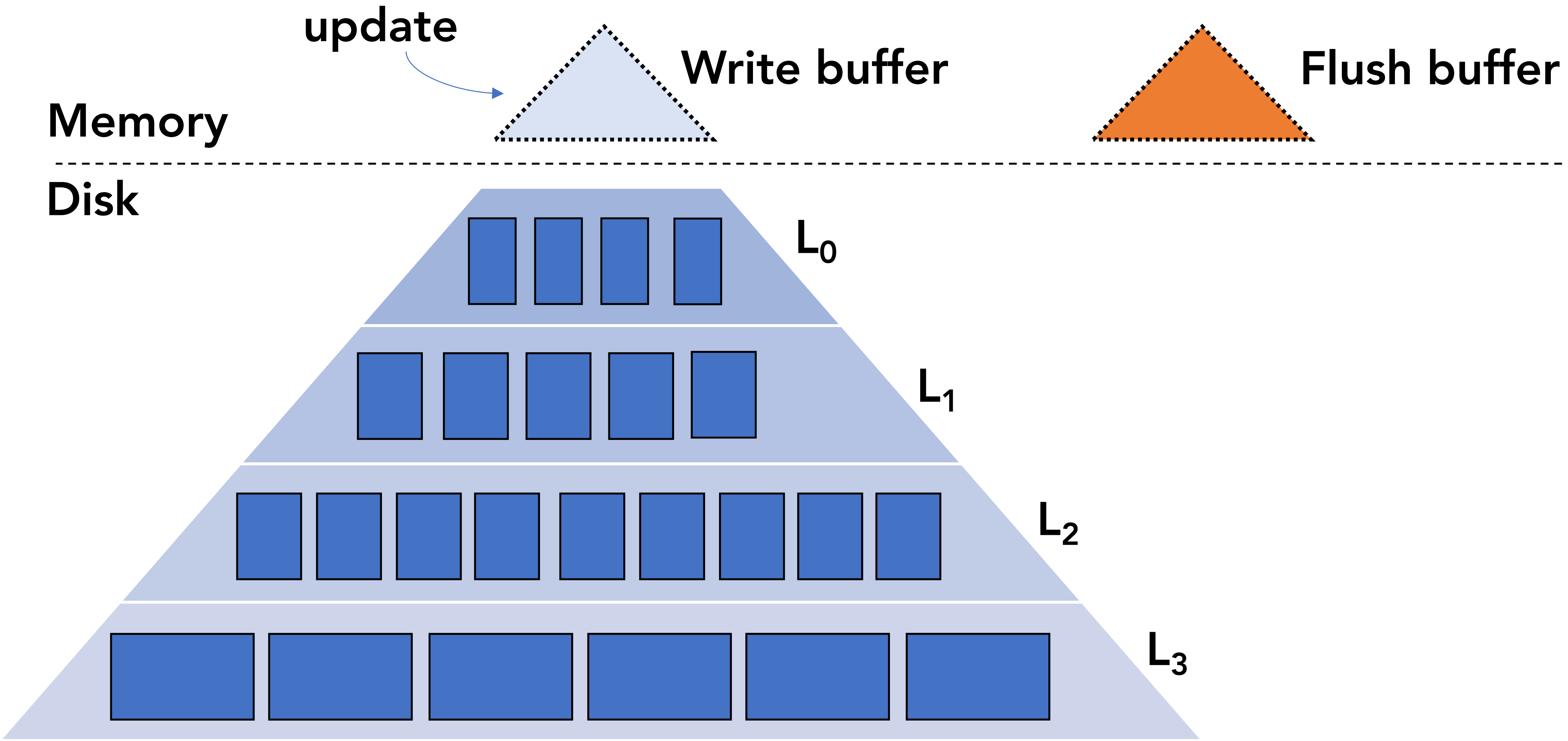
What Causes LSM Latency Spikes?

Both reads and writes experience latency spikes.

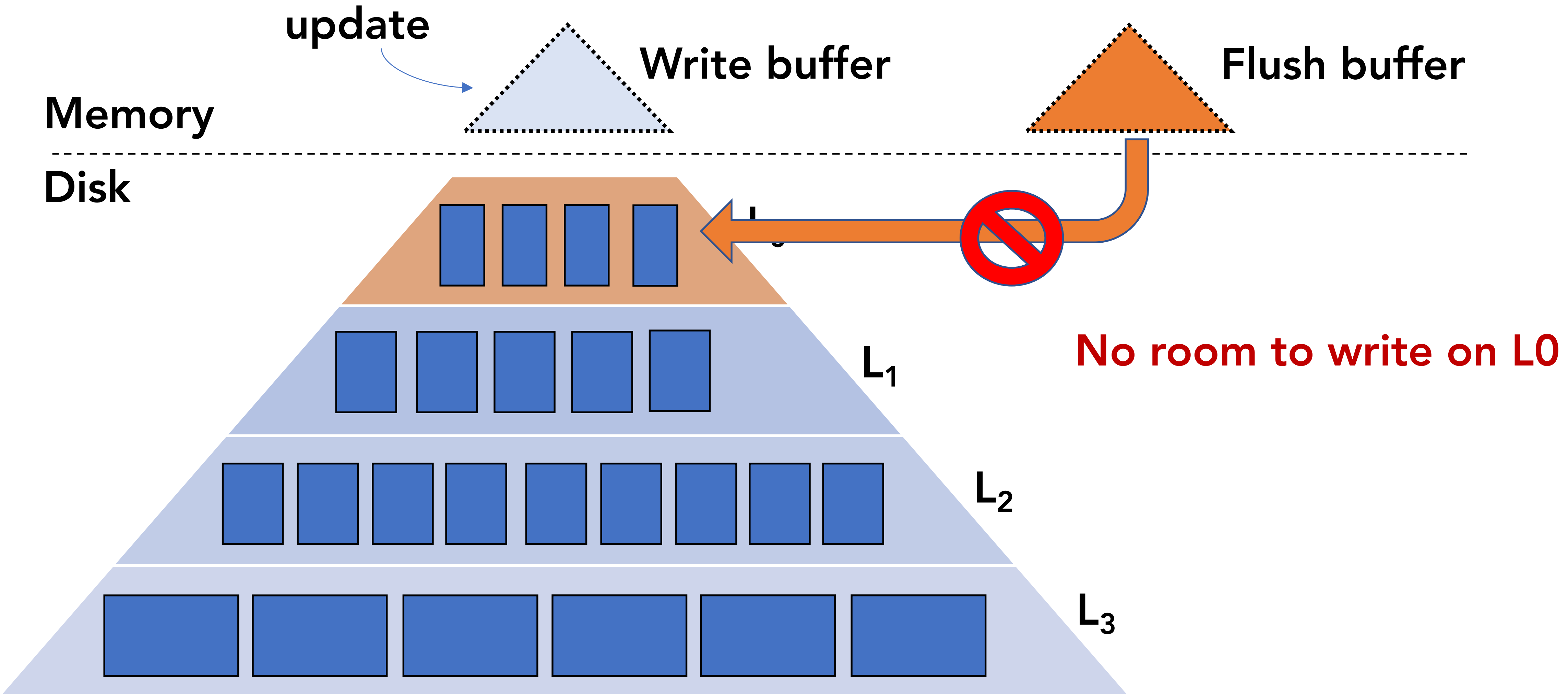
Focus on writes. Less intuitive.

Writes finish in memory. **Why do we have 1s latencies?**

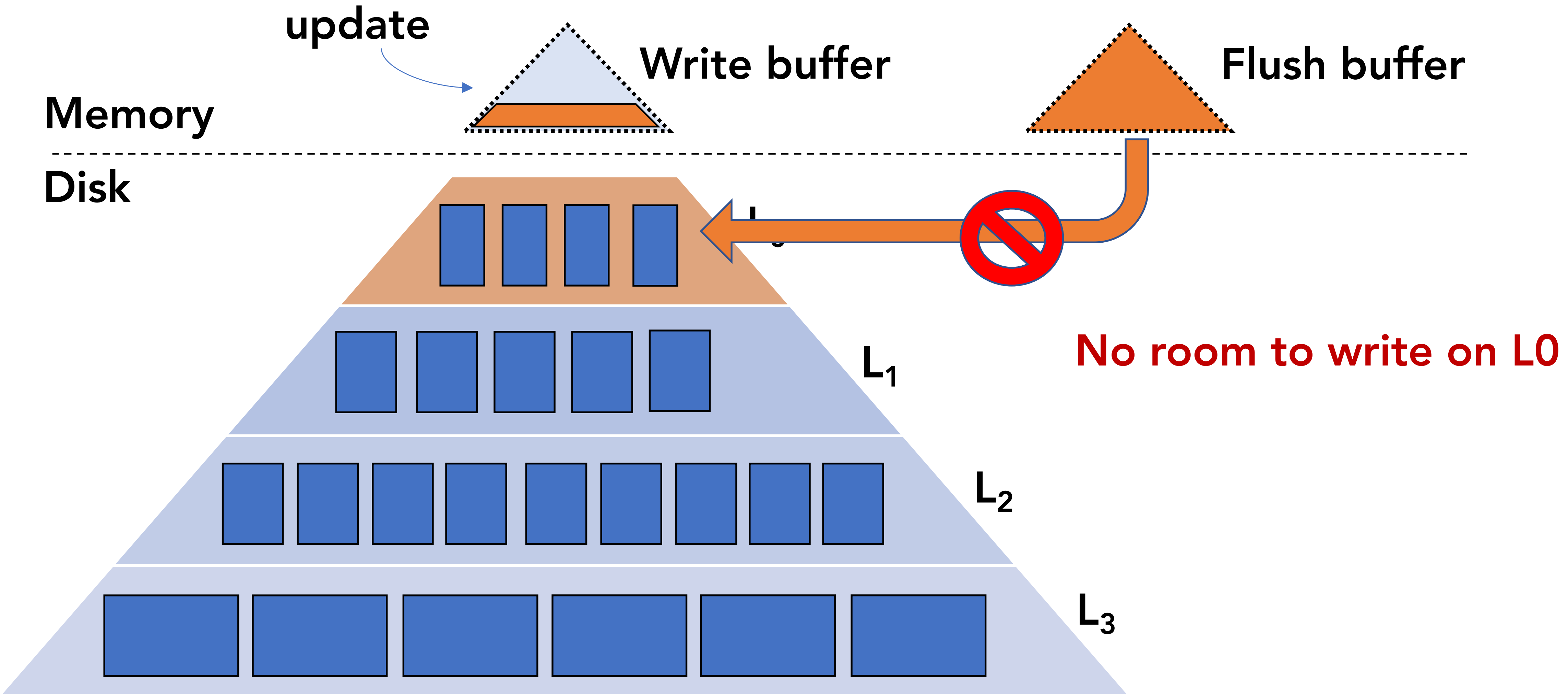
Cannot Flush



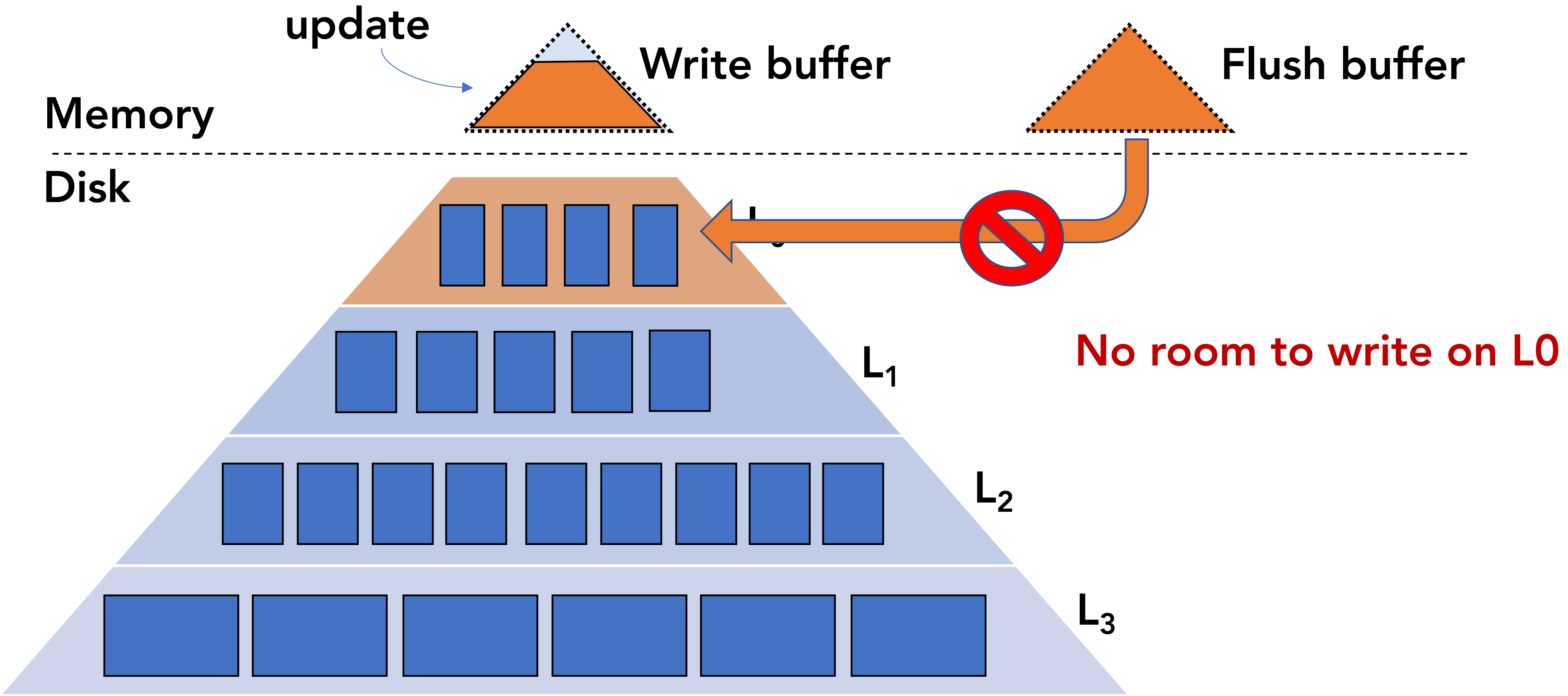
Cannot Flush



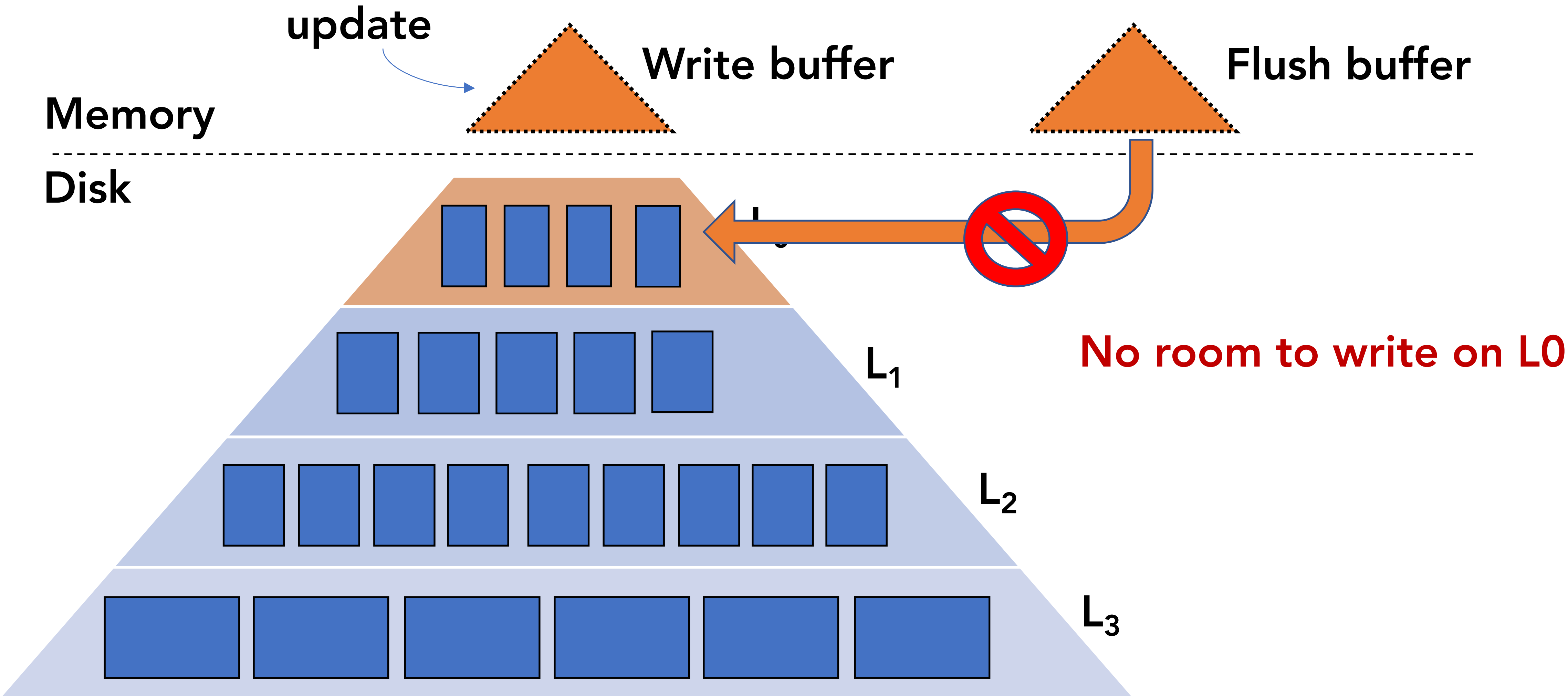
Cannot Flush



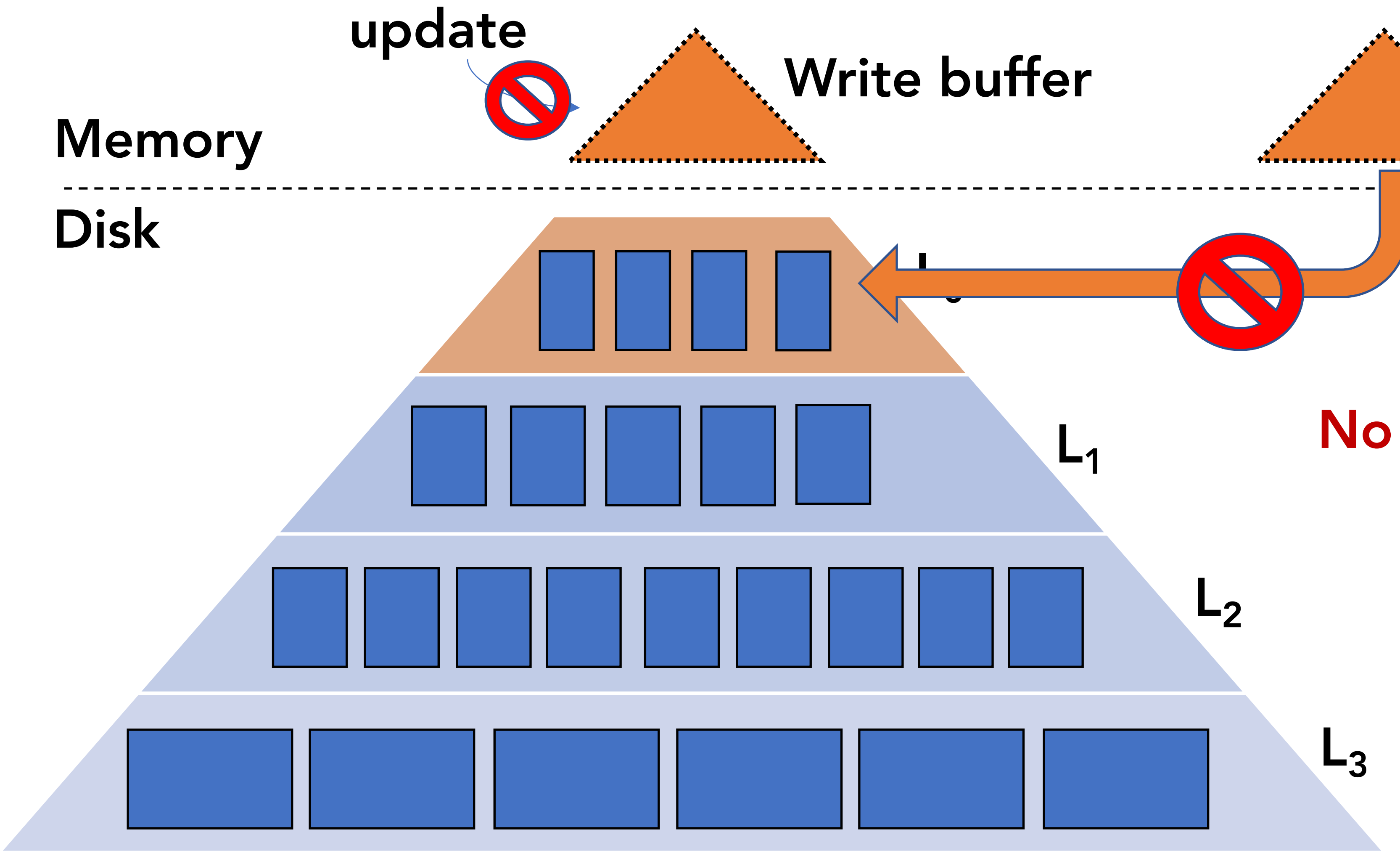
Cannot Flush



Cannot Flush



Cannot Flush



No room to write on L0

1. Writes Blocked Because L0 is Full.

No coordination between internal ops.

↓

Higher level compactions take over I/O.

↓

L0 → L1 compaction is too slow.

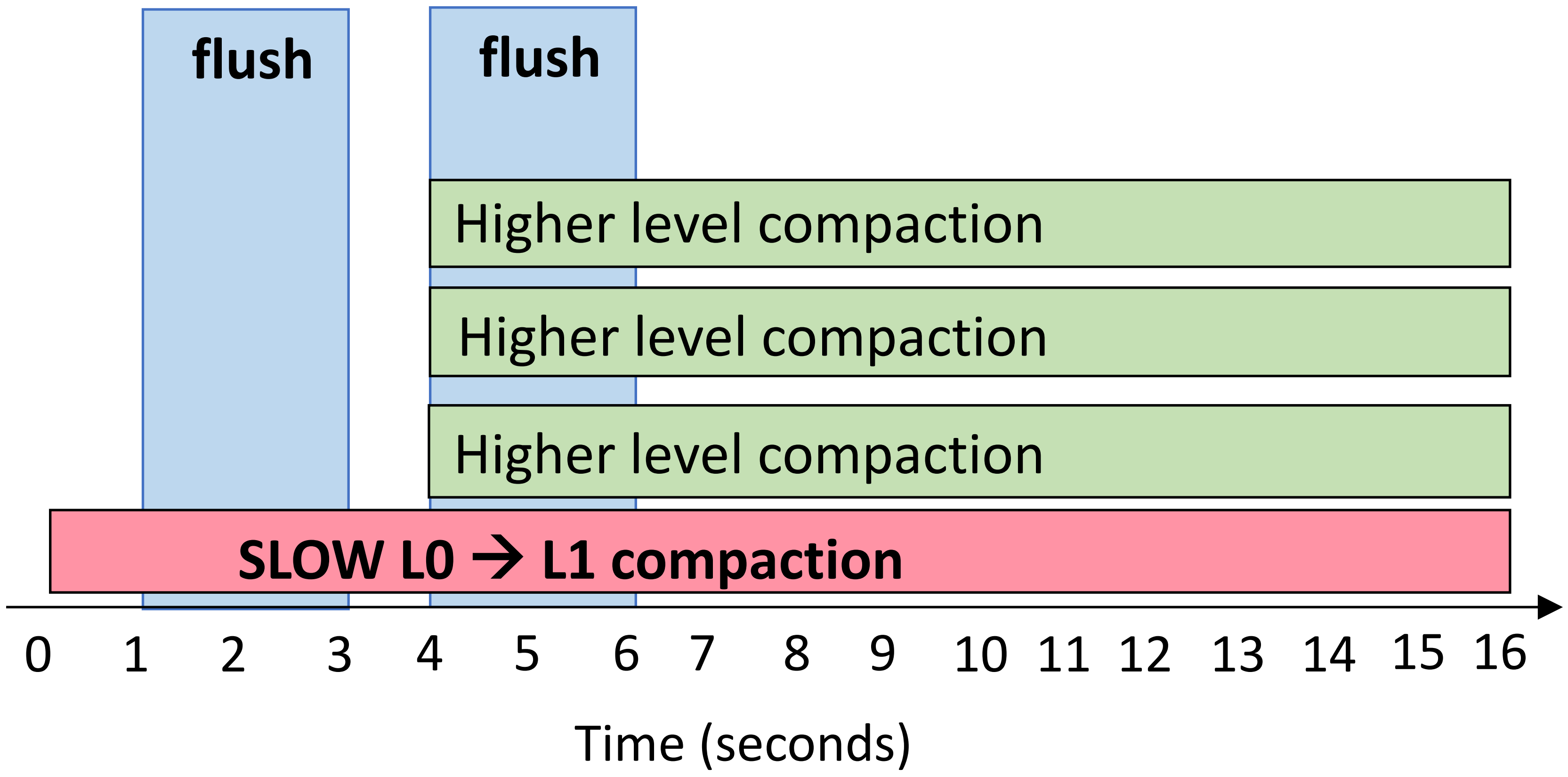
↓

Not enough space on L0.

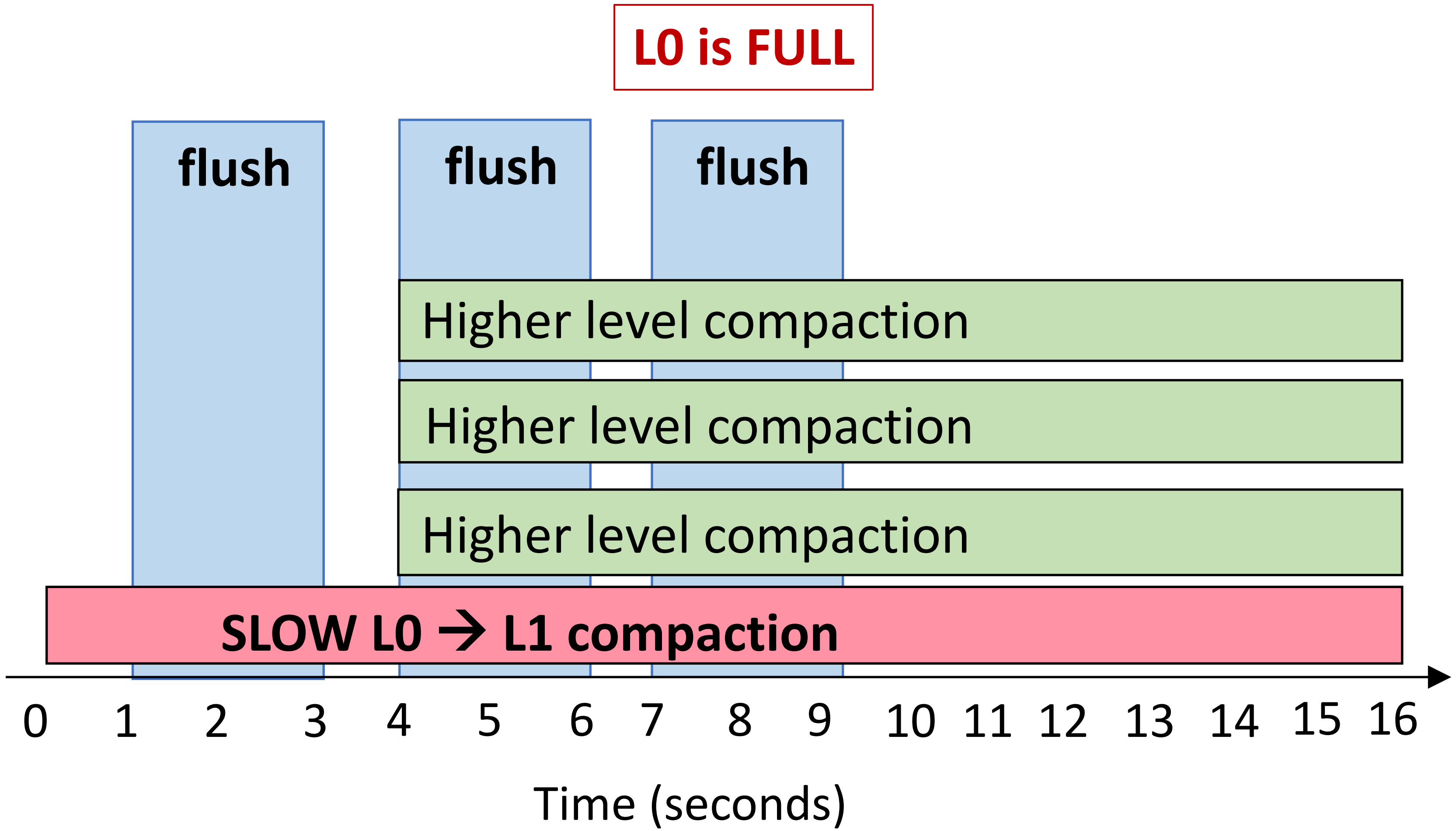
↓

Cannot flush memory component.

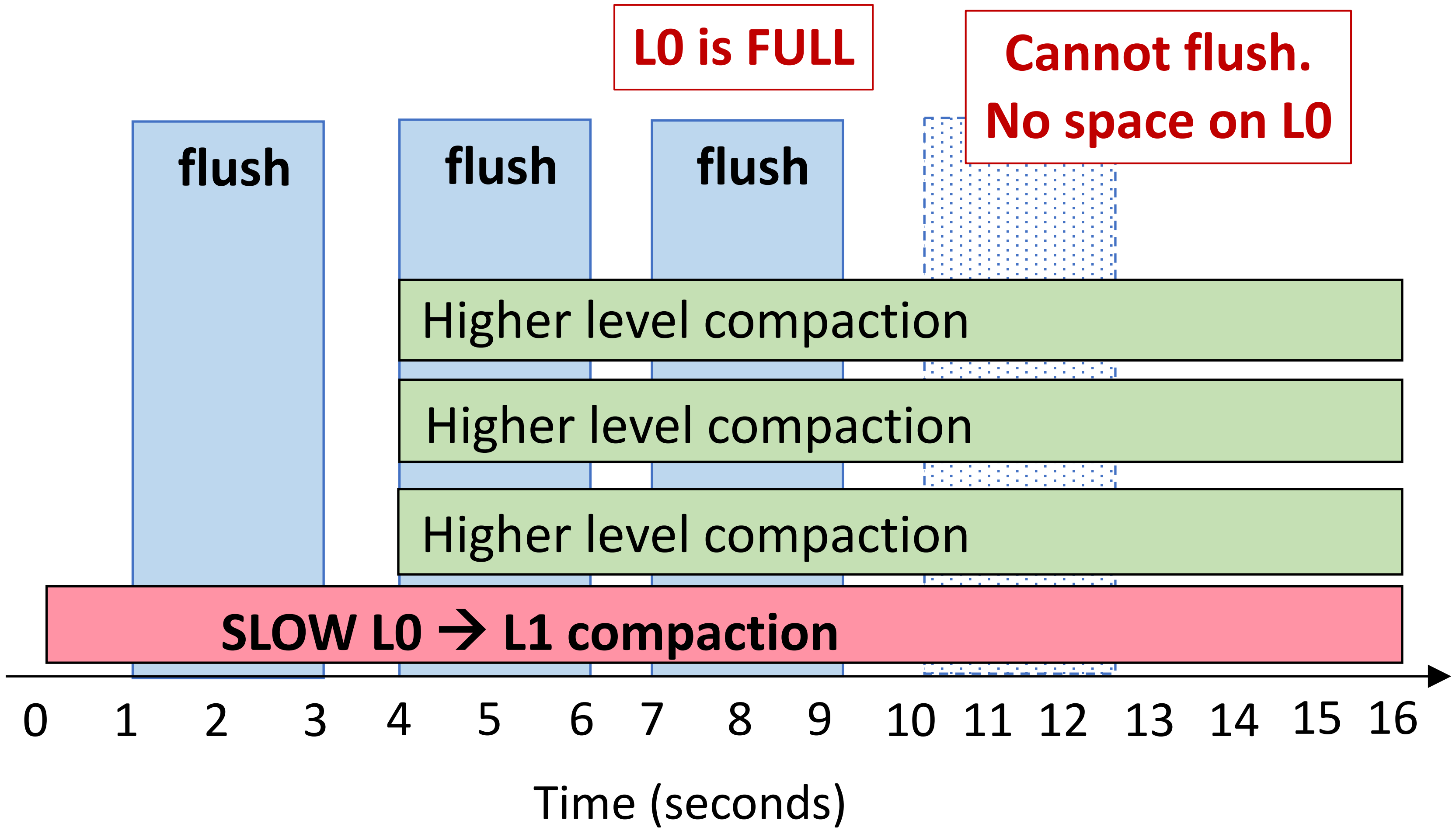
1. Writes Blocked Because L0 is Full.



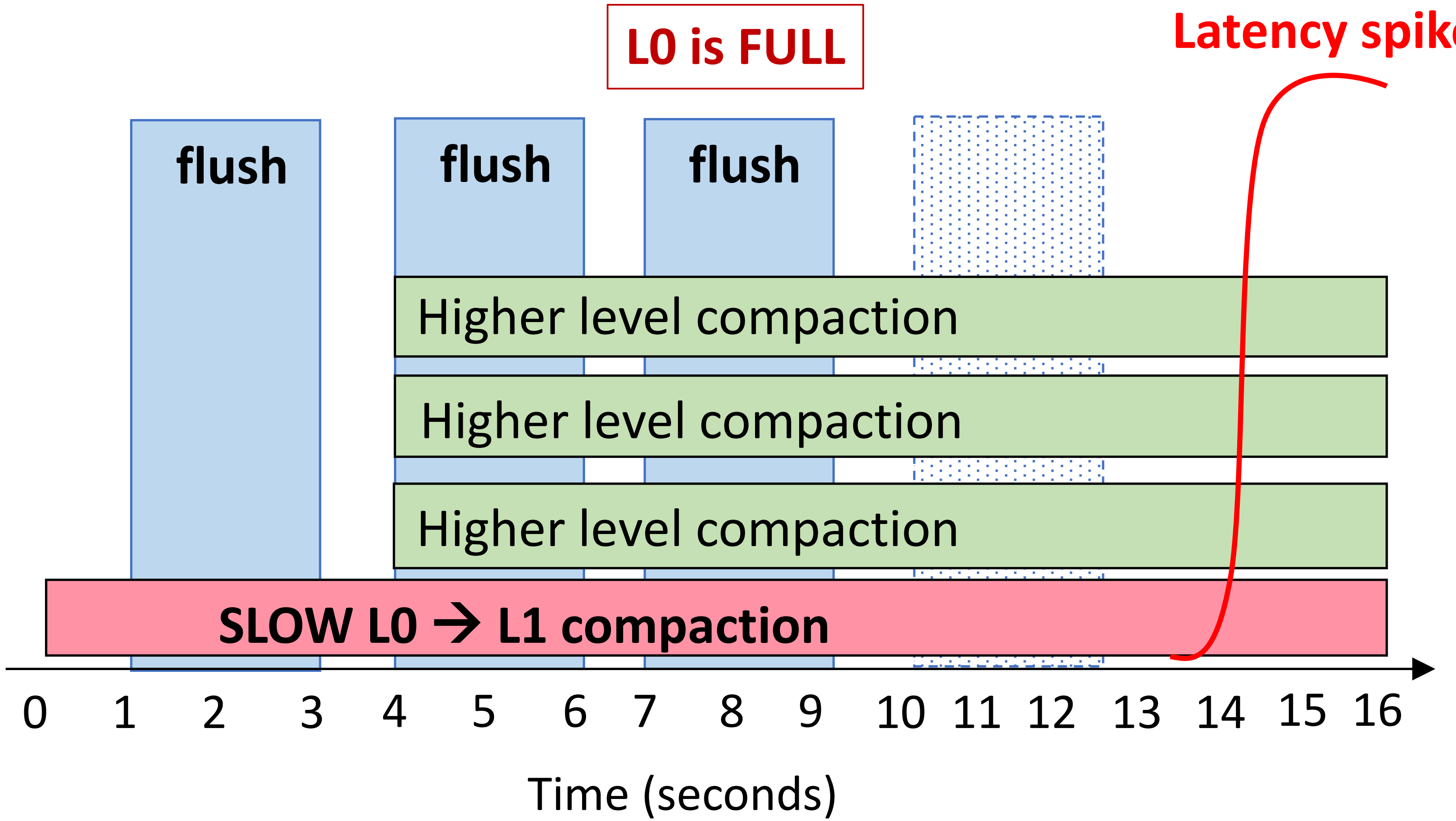
1. Writes Blocked Because L0 is Full.



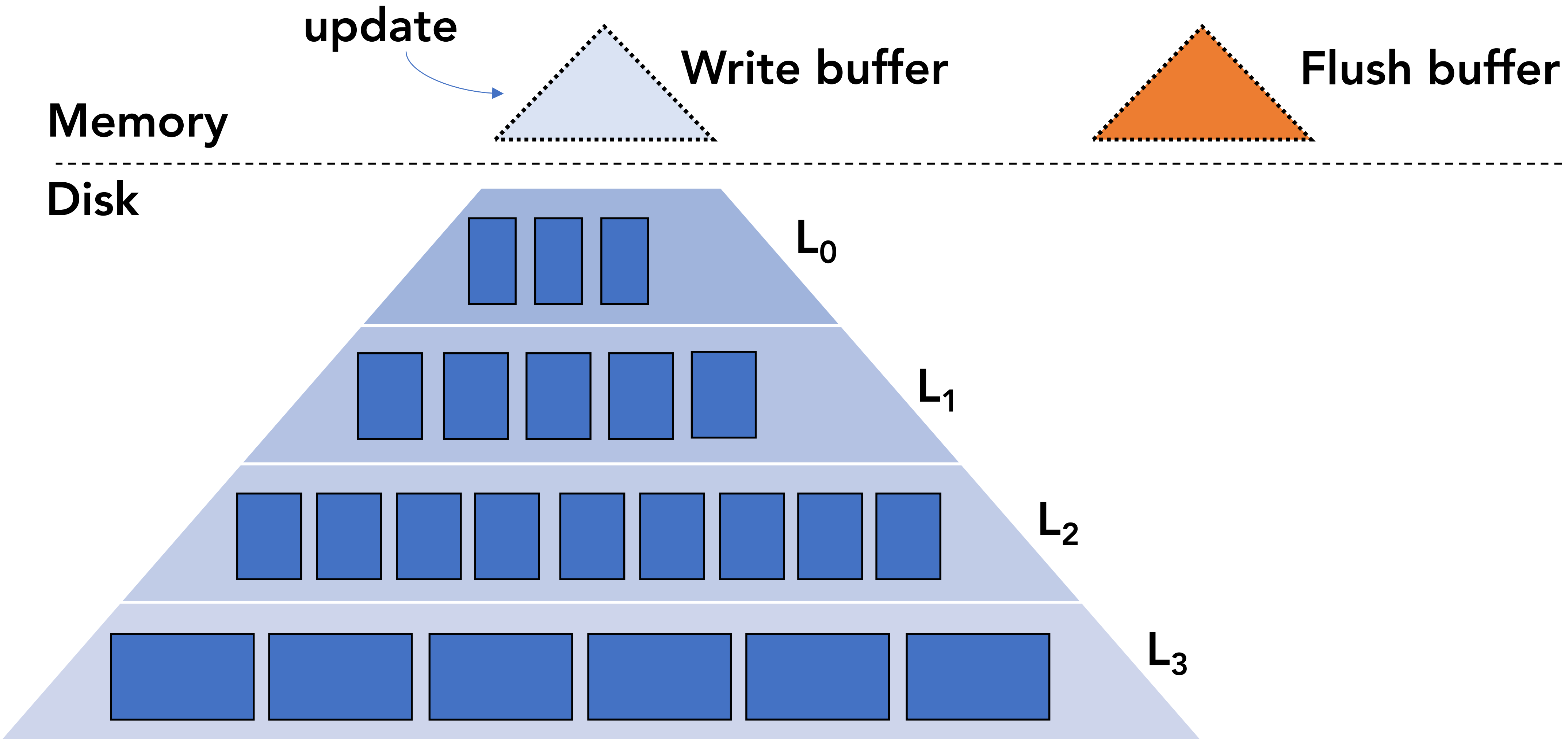
1. Writes Blocked Because L0 is Full.



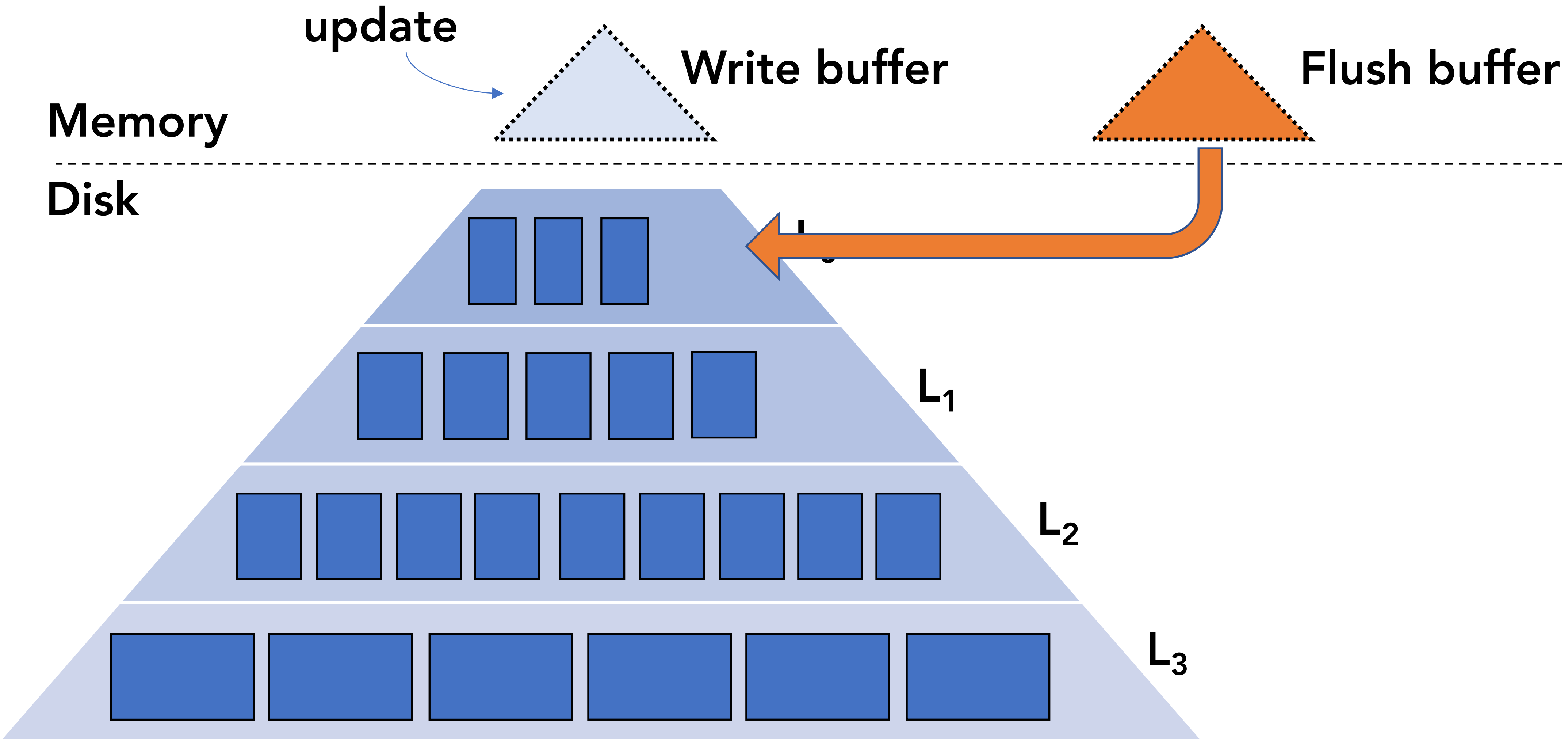
1. Writes Blocked Because L0 is Full.



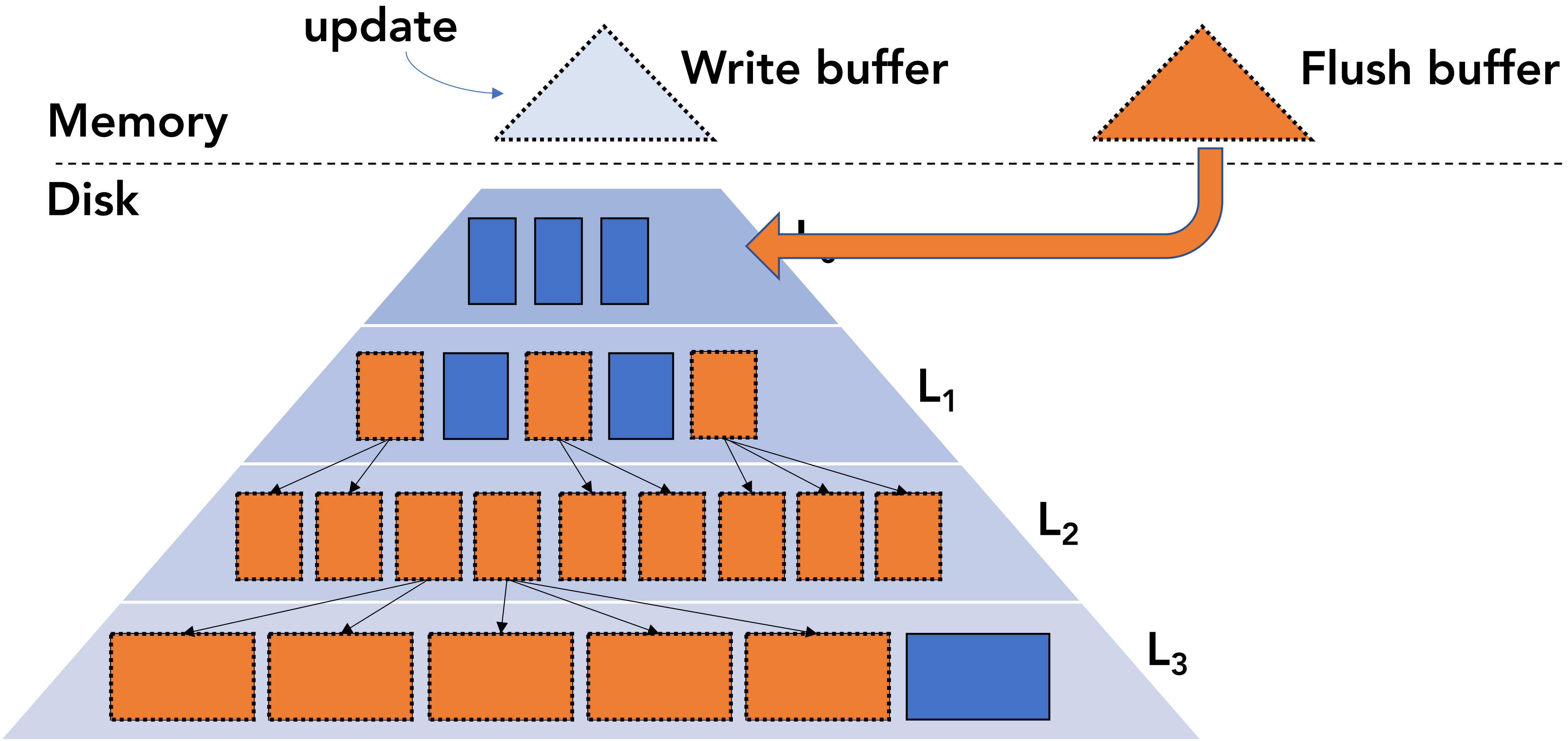
Flushing is Slow



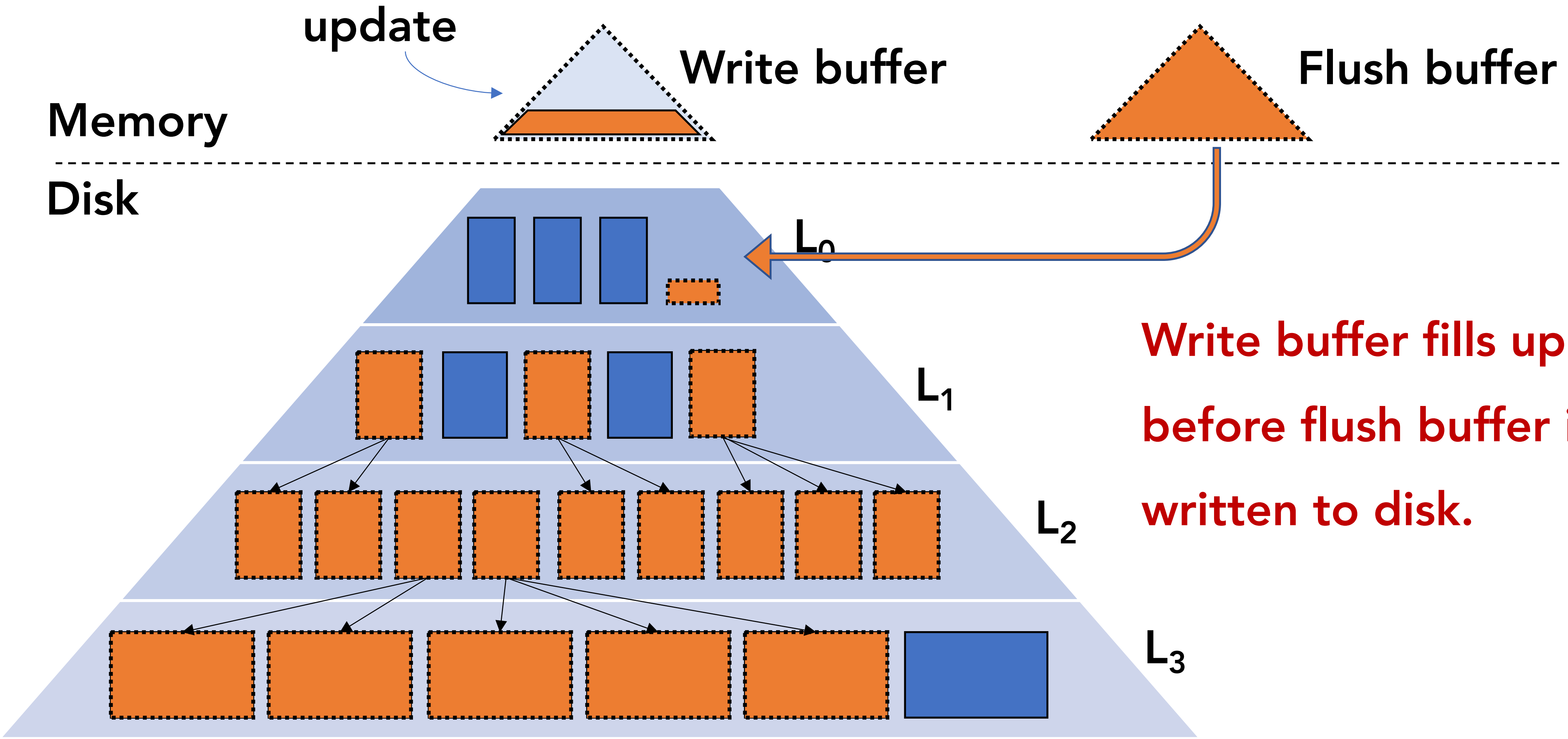
Flushing is Slow



Flushing is Slow

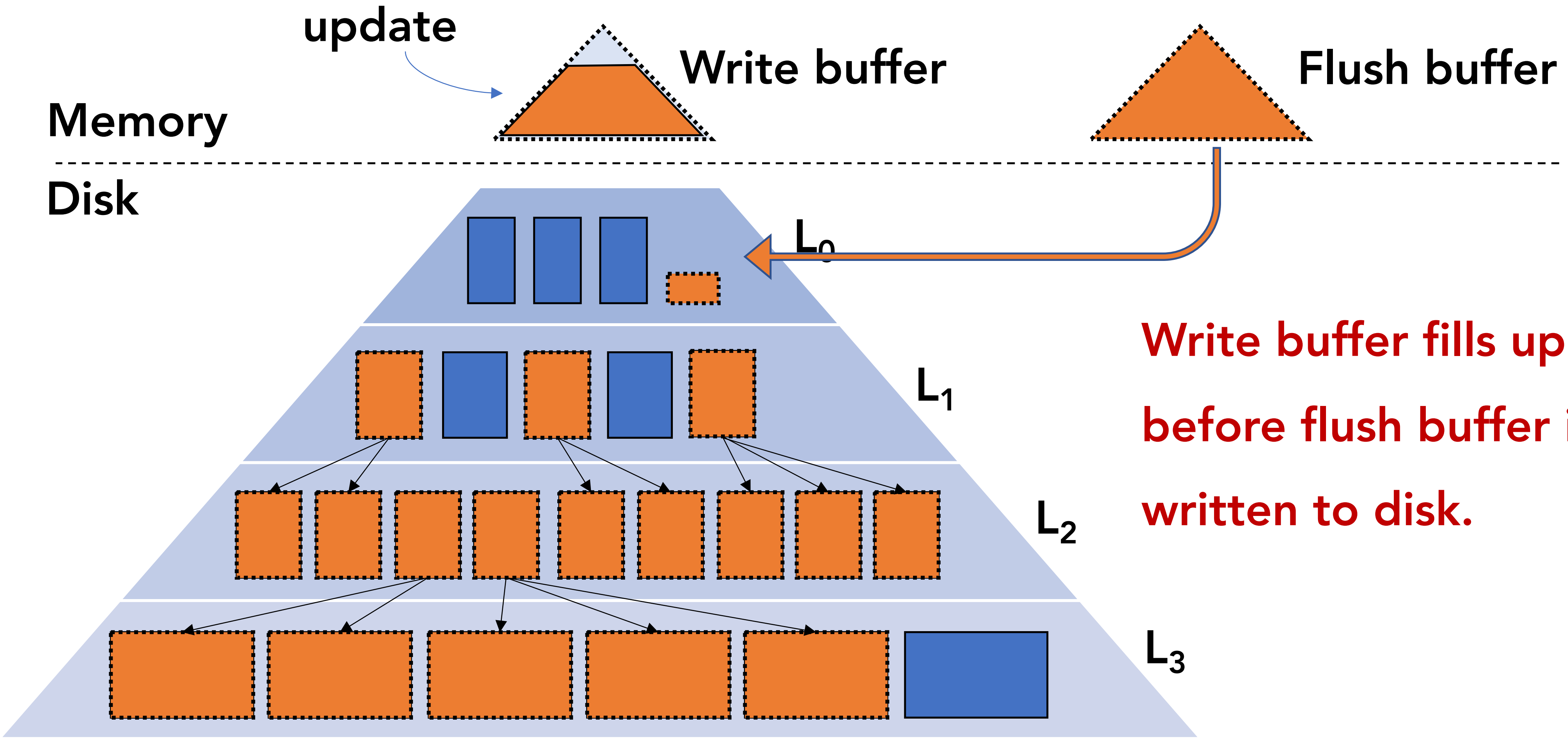


Flushing is Slow



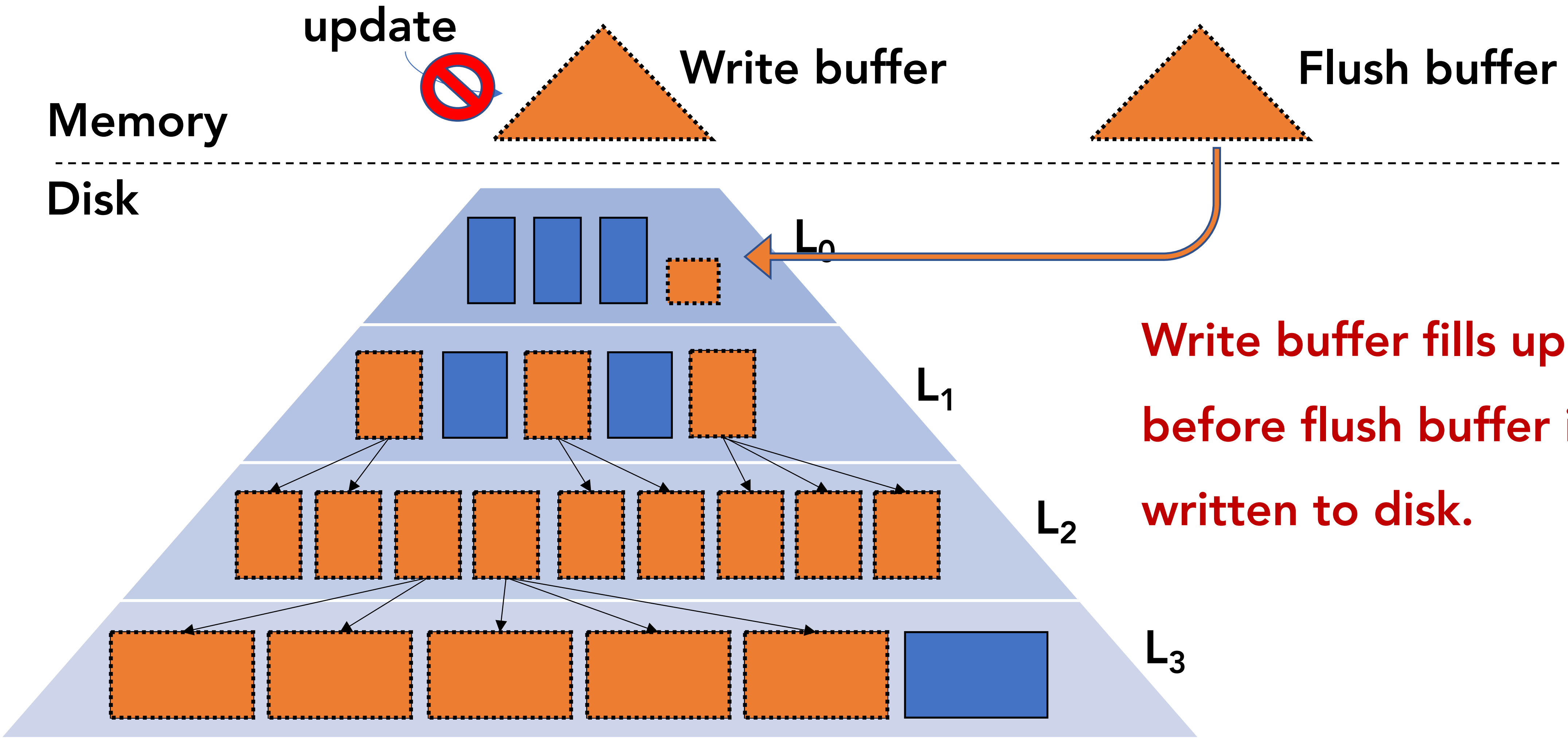
Write buffer fills up before flush buffer is written to disk.

Flushing is Slow



Write buffer fills up before flush buffer is written to disk.

Flushing is Slow



Write buffer fills up before flush buffer is written to disk.

2. Writes Blocked Because Flushing is Slow.

No coordination between internal ops.

↓

Higher level compactions take over I/O.

↓

Flushing does not have enough I/O.

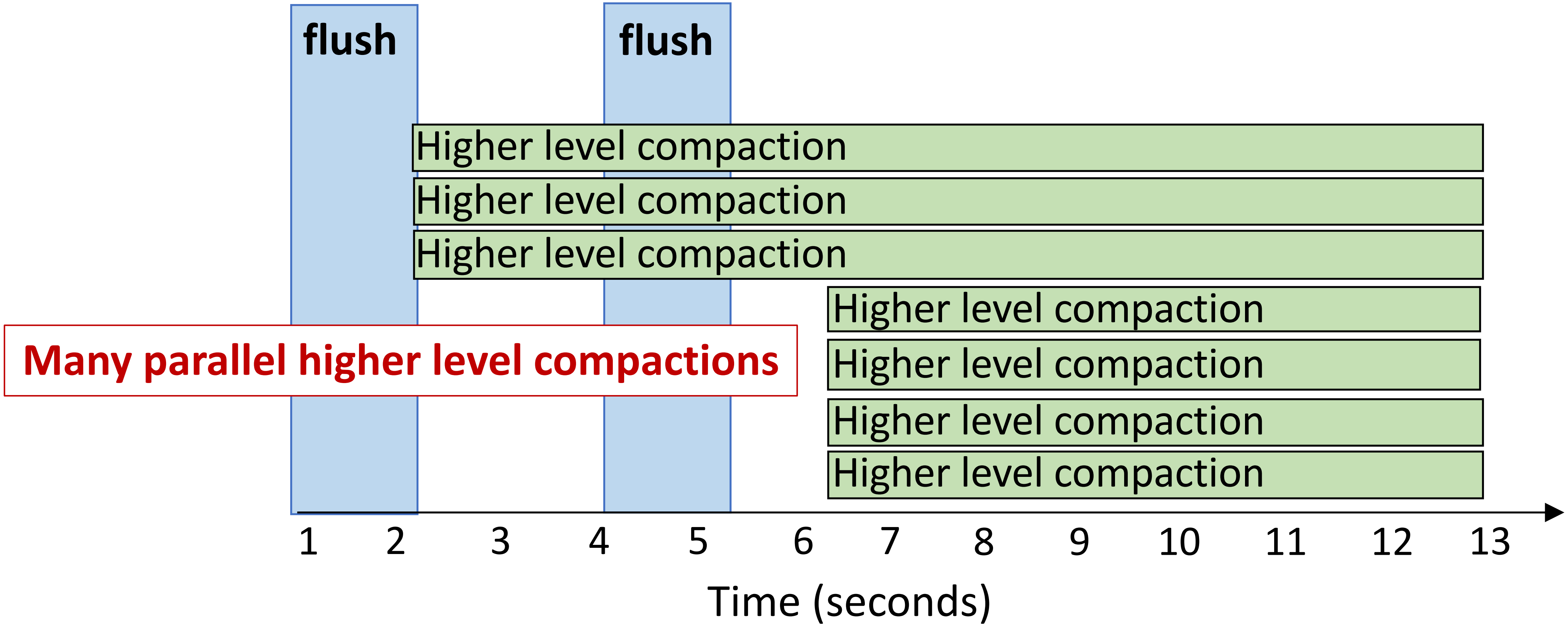
↓

Flushing is very slow.

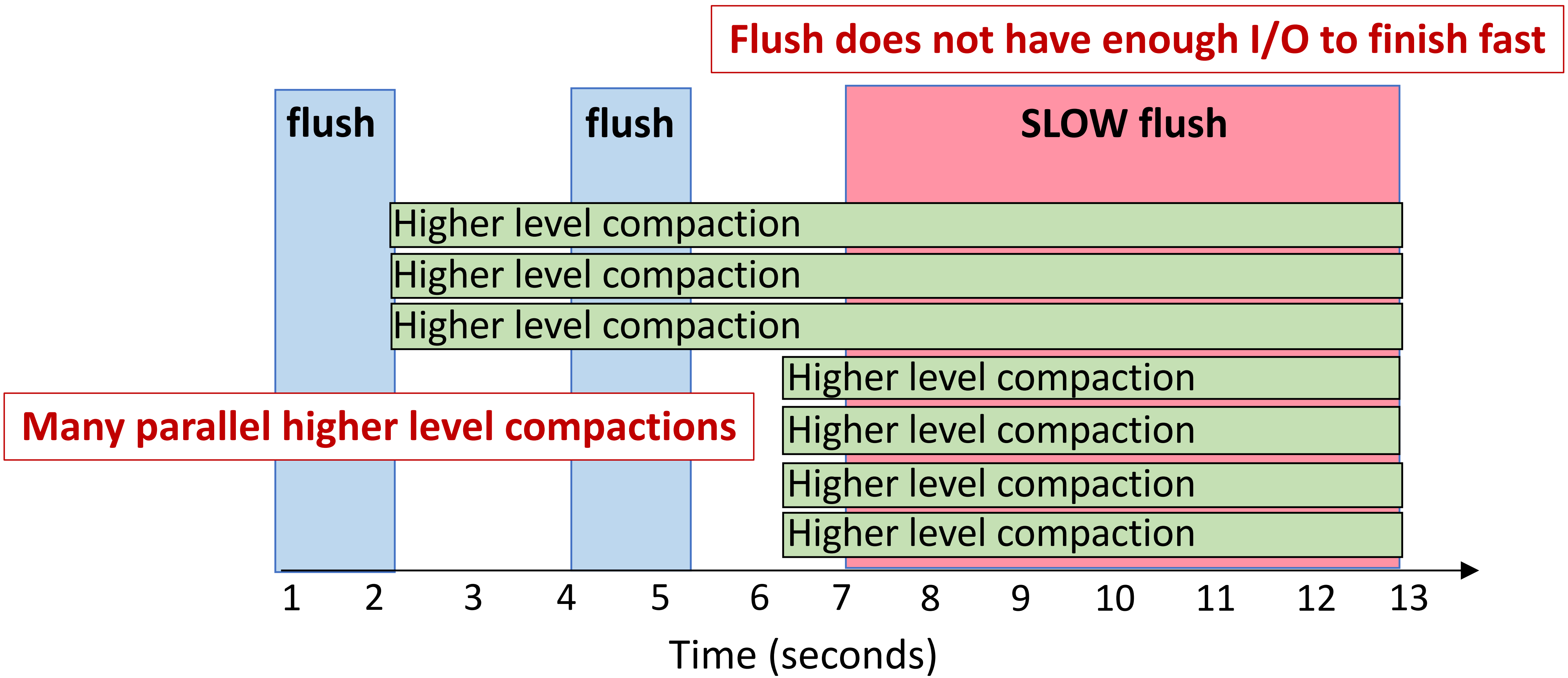
↓

Memory component becomes full.

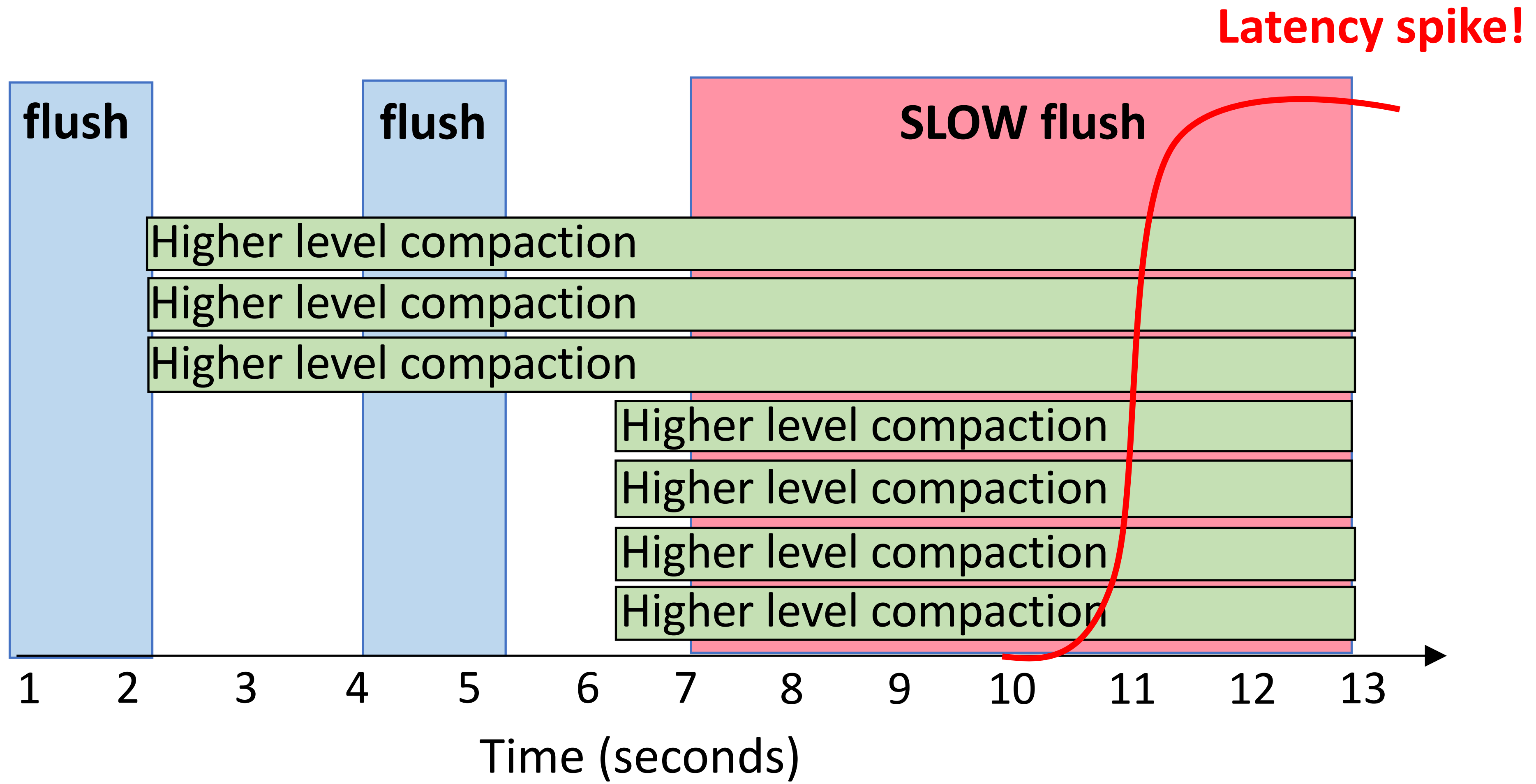
2. Writes Blocked Because Flushing is Slow.



2. Writes Blocked Because Flushing is Slow.



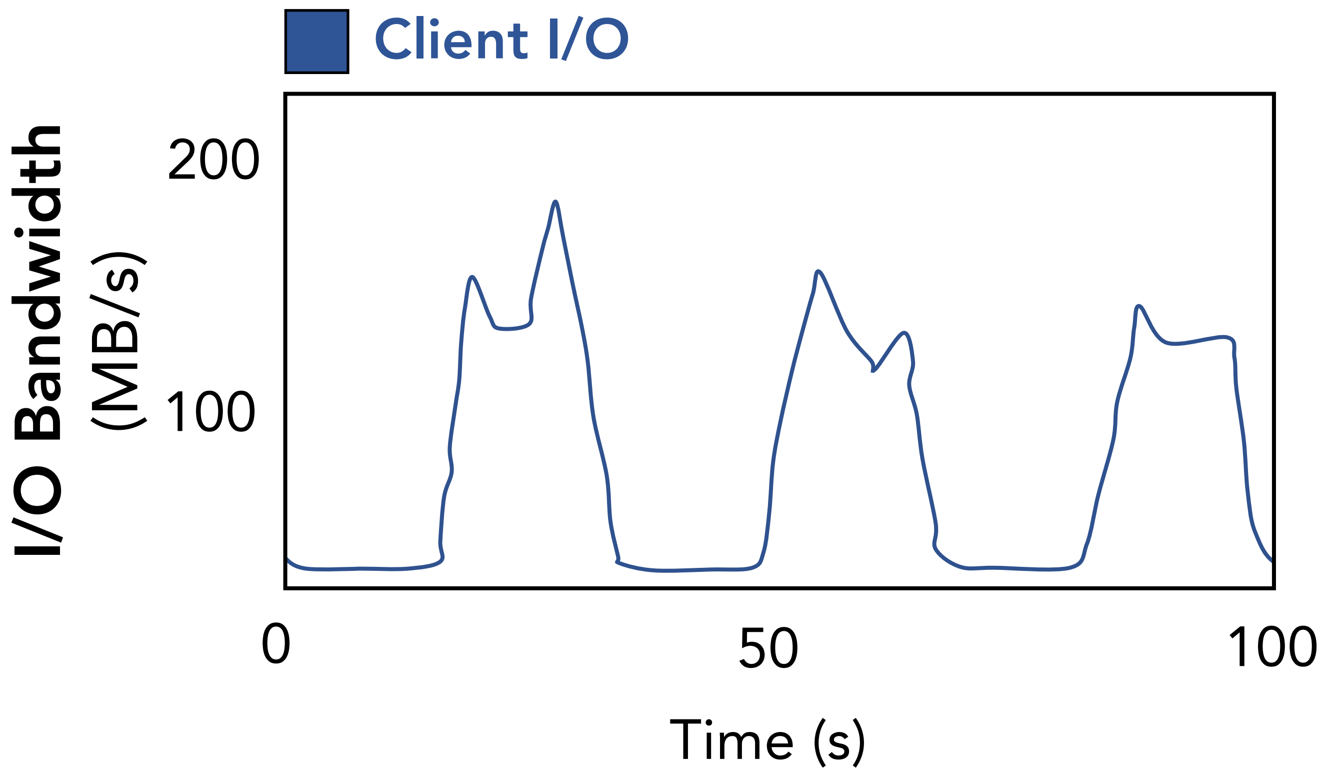
2. Writes Blocked Because Flushing is Slow.



Compaction scheduling

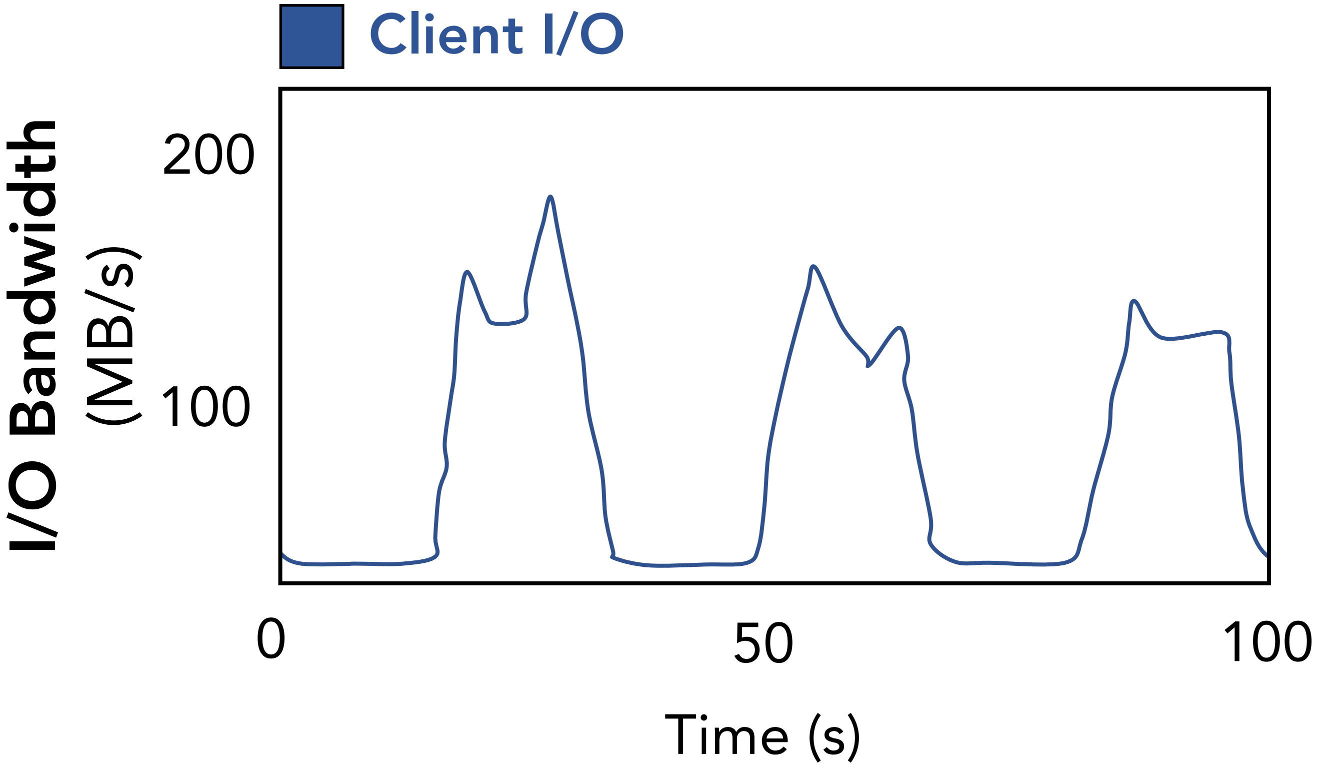
Opportunistically allocate I/O for compactions

Real Nutanix client load example



Opportunistically allocate I/O for compactions

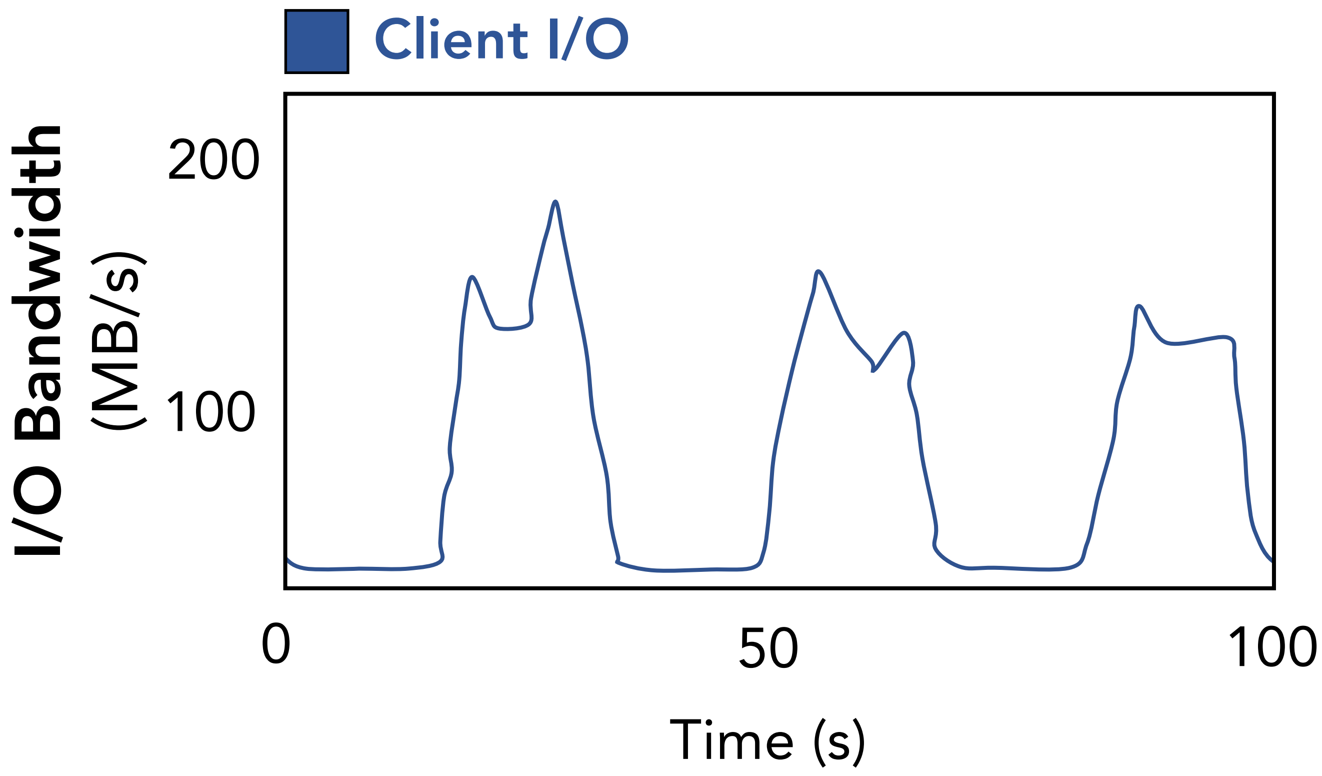
Real Nutanix client load example



Client workload is **not constant**.

Opportunistically allocate I/O for compactions

Real Nutanix client load example

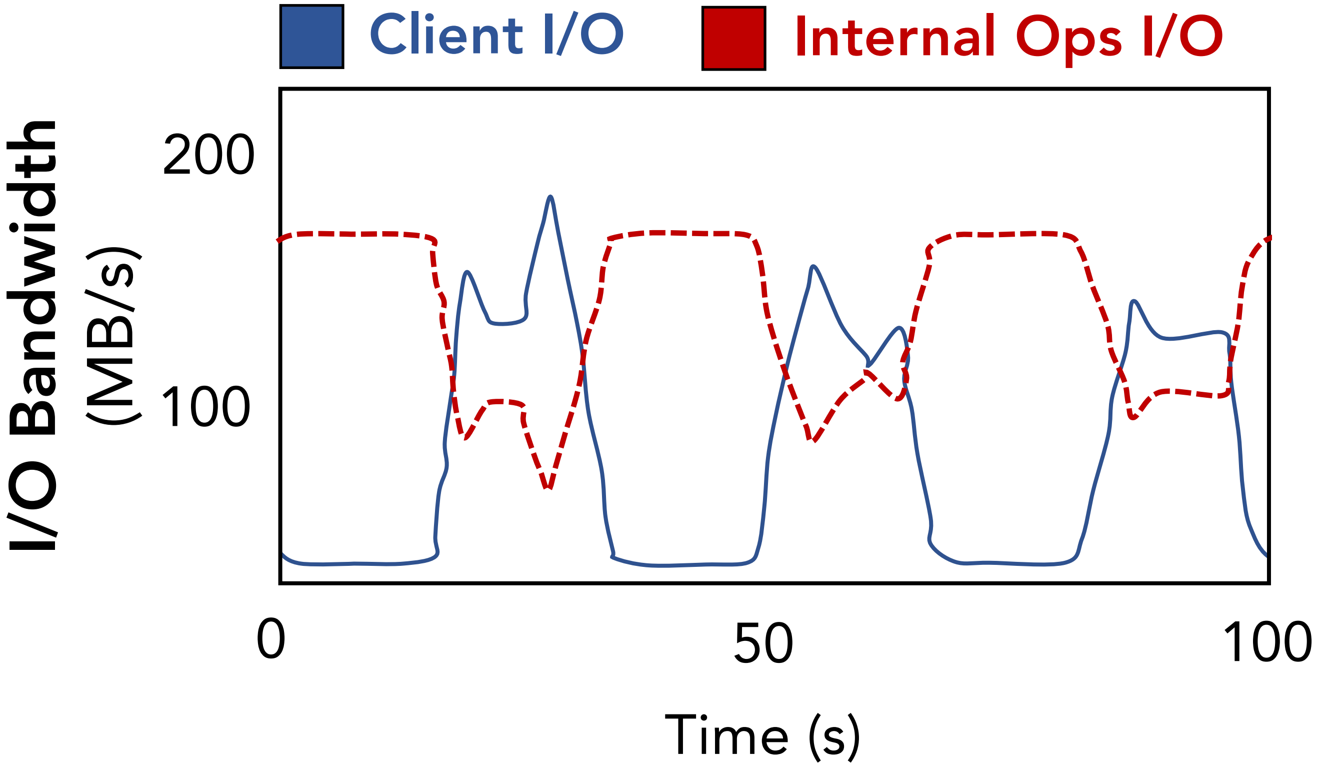


Client workload is **not constant.**

SILK **continuously monitors** client **I/O bandwidth** use.

Opportunistically allocate I/O for compactions

Real Nutanix client load example

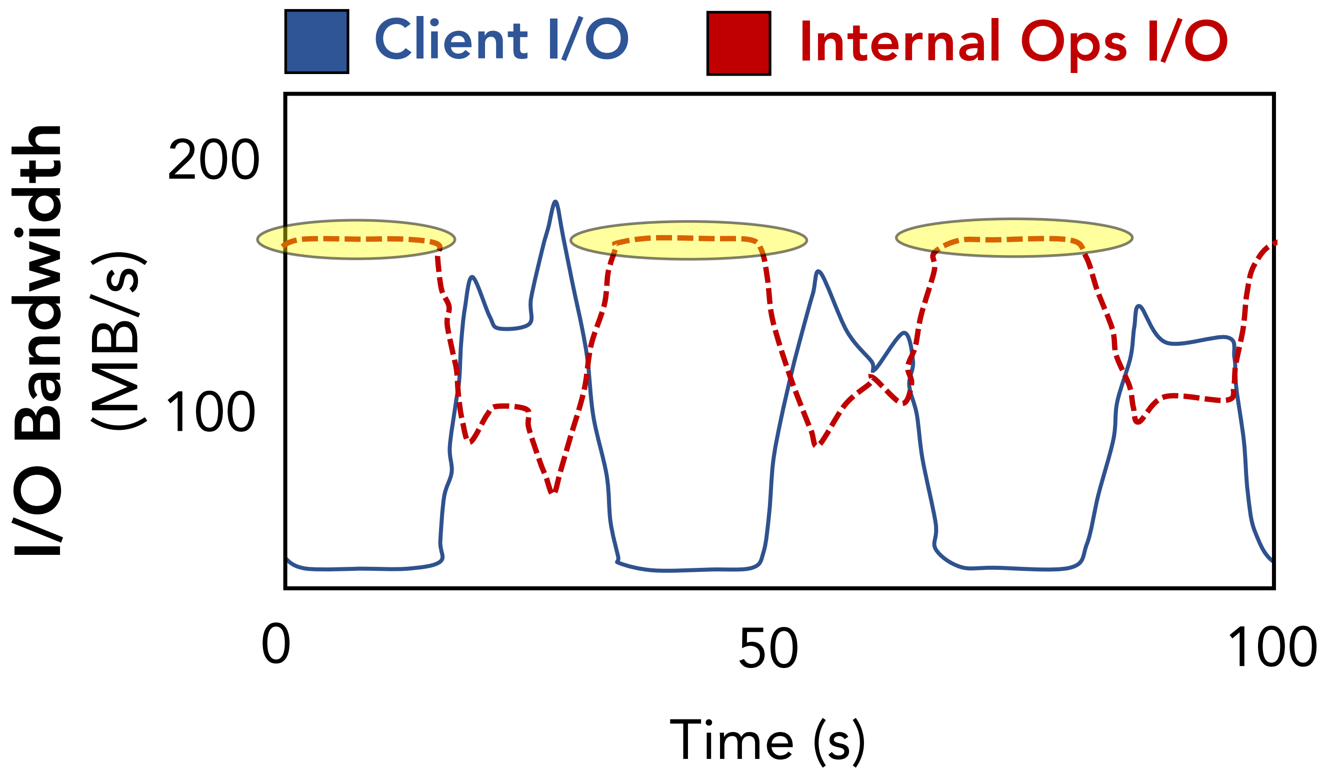


Allocate **less I/O to compactions** during **client peaks.**

Allocate **more I/O to compactions** during **client low load.**

Opportunistically allocate I/O for compactions

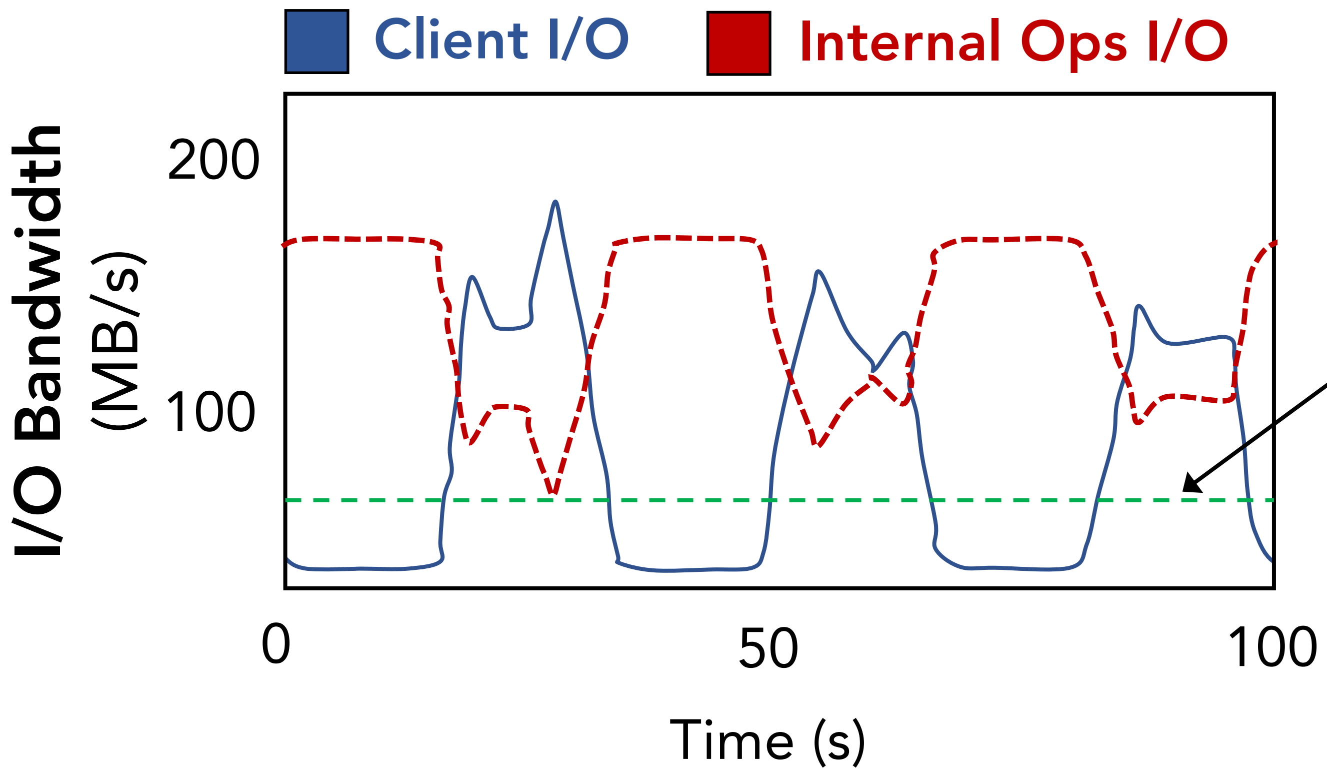
Real Nutanix client load example



More I/O to high level compactions during low load → **don't fall behind**.

Opportunistically allocate I/O for compactions

Real Nutanix client load example



More I/O to high level compactions during low load → **don't fall behind**.

Even in peak load, guarantee min I/O for flushing and L0 → L1 compaction.

Why does log-based design work for KV but not for FS?

- Fewer operations on the metadata in LSM KVs compared to LFS.
- Level-based design makes cleaning less disruptive than in LFS.
- Operations in KVs are simpler.
  - No need for transactions, snapshots.
  - More relaxed crash consistency.
  - More relaxed concurrency model (e.g., for multi-key operations).

Summary: LSM Key-Value Stores

- Absorbs writes in memory and writes sequentially to disk.
- Reads mostly from cache.
  - Reads from disk: bit more expensive but few
- Write amplification
- Compaction