

The Story of RocksDB

Embedded Key-Value Store for Flash and RAM






Dhruba Borthakur & Haobo Xu
Database Engineering@Facebook









Dhruba Borthakur
Edit Profile








FAVORITES

-  News Feed
-  Messages 18
-  Events
-  Photos
-  Browse

PAGES

-  Rocksdb
-  Pages Feed 20+
-  Like Pages 20+
-  Create Ad

GROUPS

-  RocksDB Internal 1
-  Rocksdb Users Gr... 3
-  Database Engineering
-  Multifeed on rock... 1
-  Rocksdb for clicks... 10
-  Data @ Scale Conf... 1
-  Everstore 3
-  Data Infrastructur... 5
-  SI on mcrocksdb 2
-  Core Data Internal 4

 Update Status  Add Photos/Video

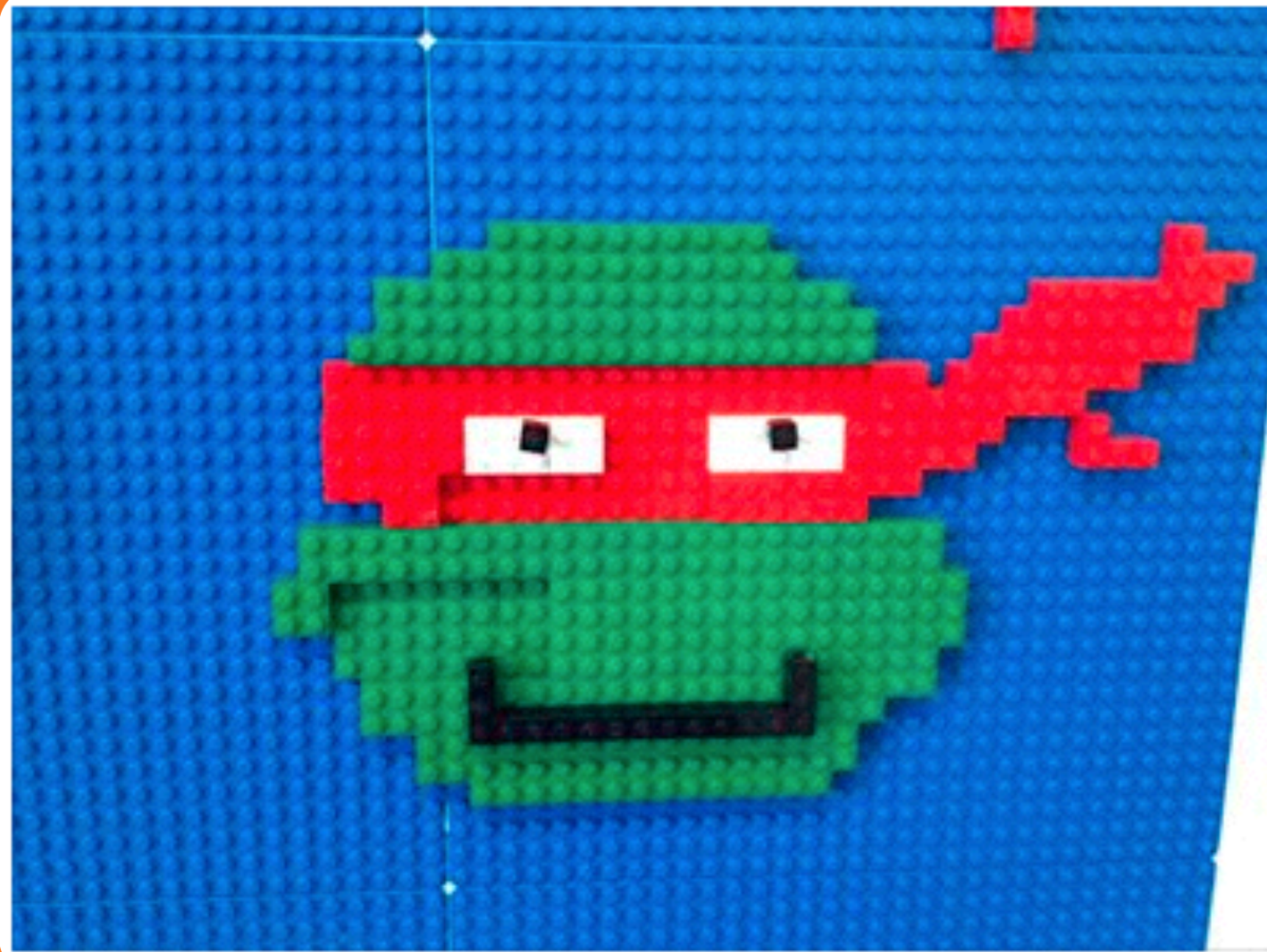
What's on your mind?

SORT ▾





Kai Liu

Ninja turtle completed! — with Wenjing Yu at Facebook HQ.



Like · Comment · Share · Yesterday at 3:46pm in Menlo Park · 🌐

 Rongrong Zhong, Dheeraj Kumar Singh, Volodymyr Krestianynkov and 8 others like this.

 Michel Tu It looks like you both are working hard 😊

 **Weiyan Wang's** birthday is today

Sponsored 

Create



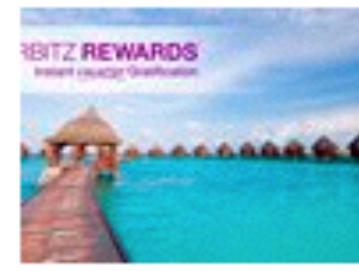
Sameet Agarwal and Manish Modi like Mercury Trip.

MERCURYTRIP
1.408.432.8747
1.510.573.5053
Open all Days

Mercury Trip
 Like

Join Orbitz Rewards

earn immediately. redeem insta...



It's on us! Get a pro code for 15% off hotels when you join the N Orbitz Rewards.

Save at Lowe's with Amex



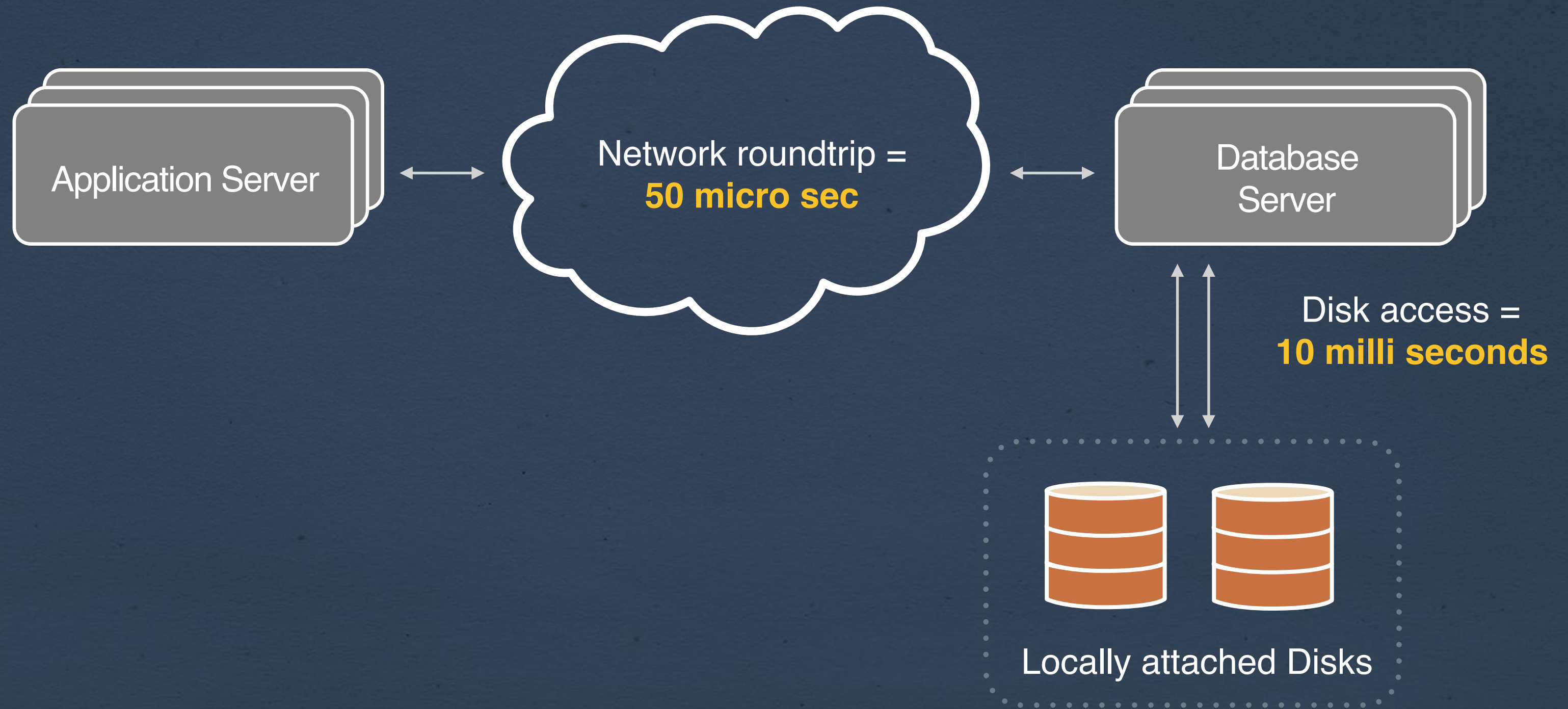
Home improvement on your to do list? Spend \$50 and get \$10 back 1x at Lowe's. See how.

Exclusive Customer Offer
att.com

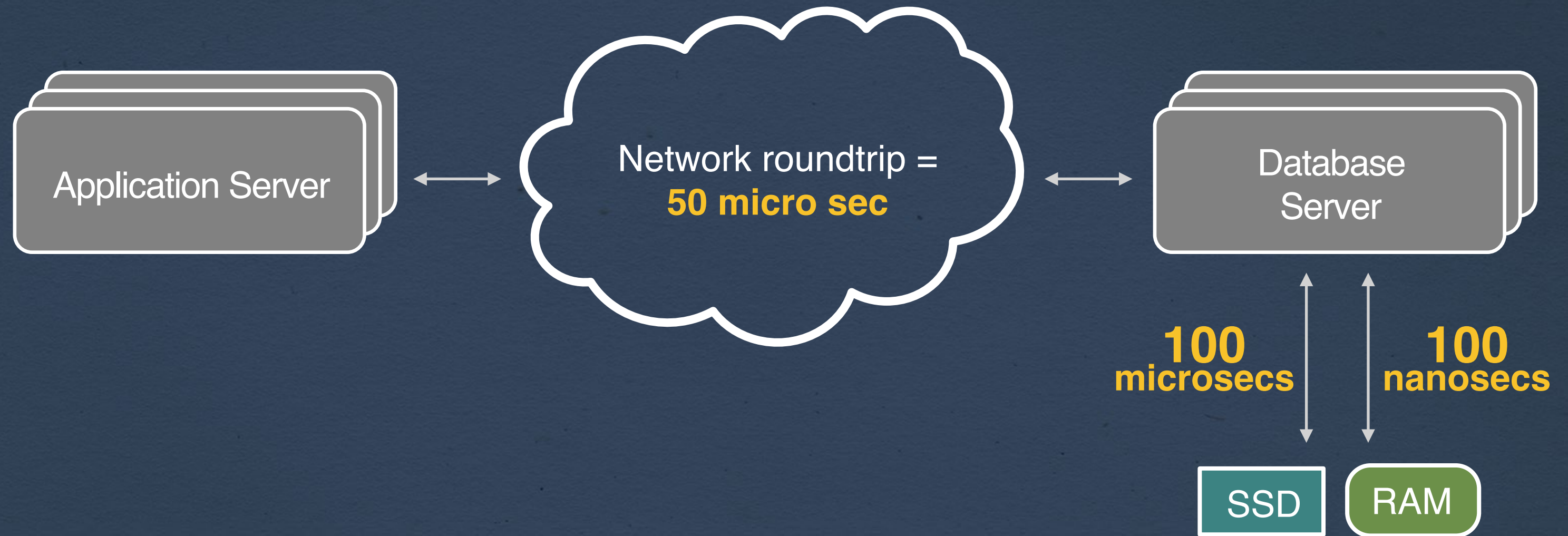


AT&T wireless customers save more! Get U-verse TV, Internet & Home Phone—\$74/mo for 1

A Client-Server Architecture with disks

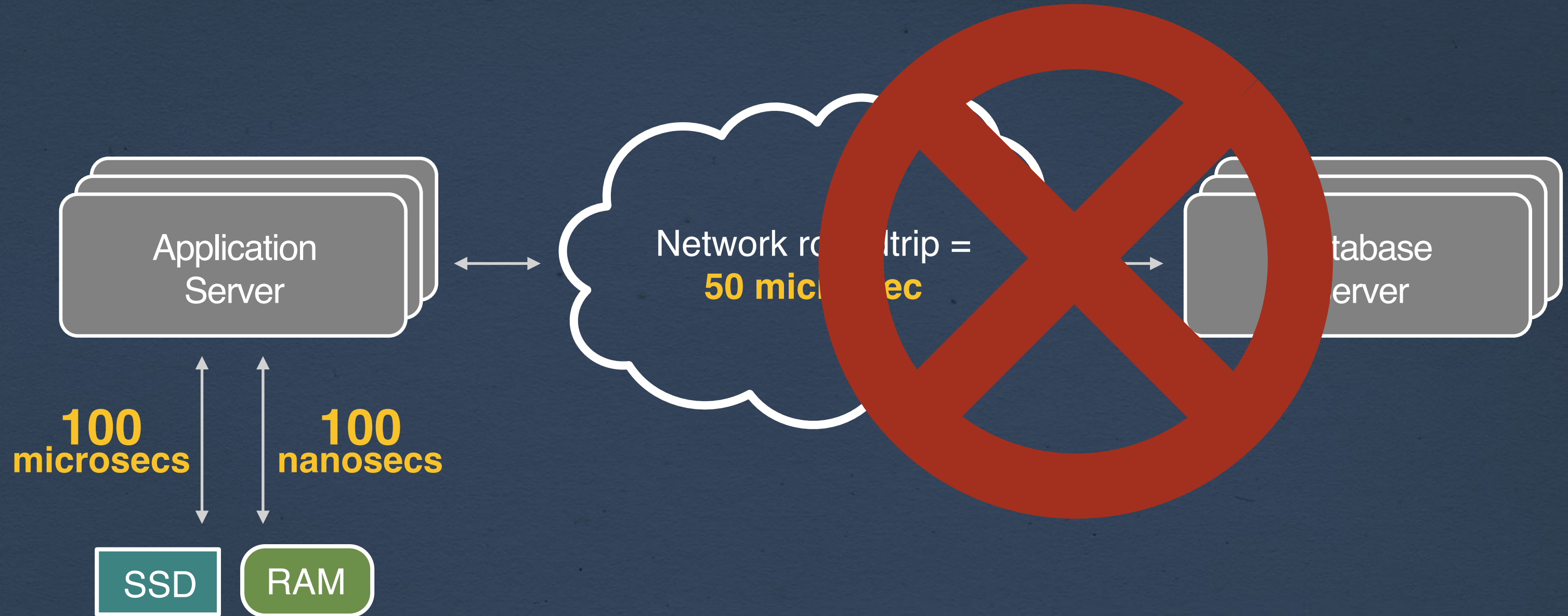


Client-Server Architecture with fast storage



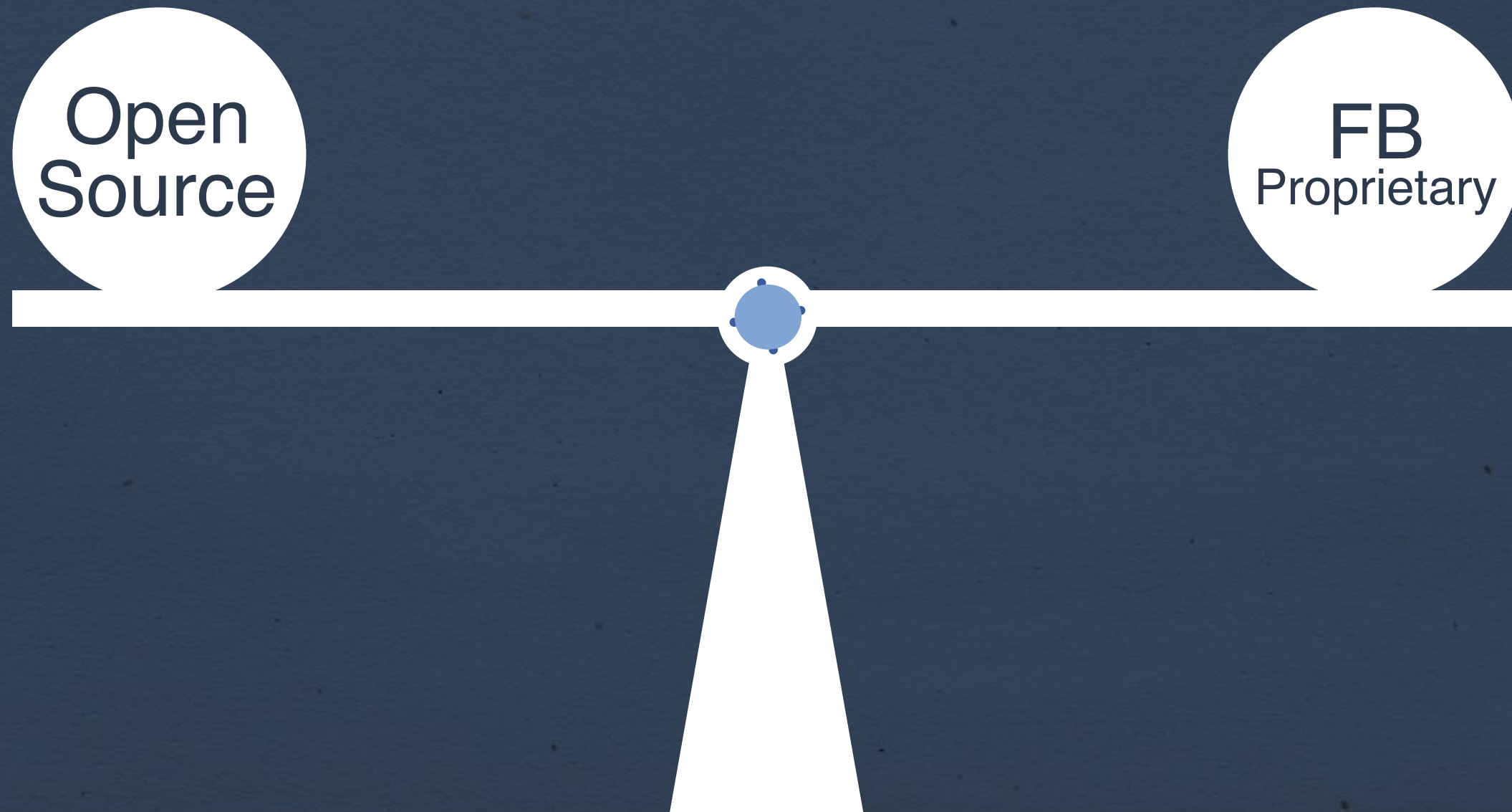
Latency dominated by network

Architecture of an Embedded Database



Storage attached directly to application servers

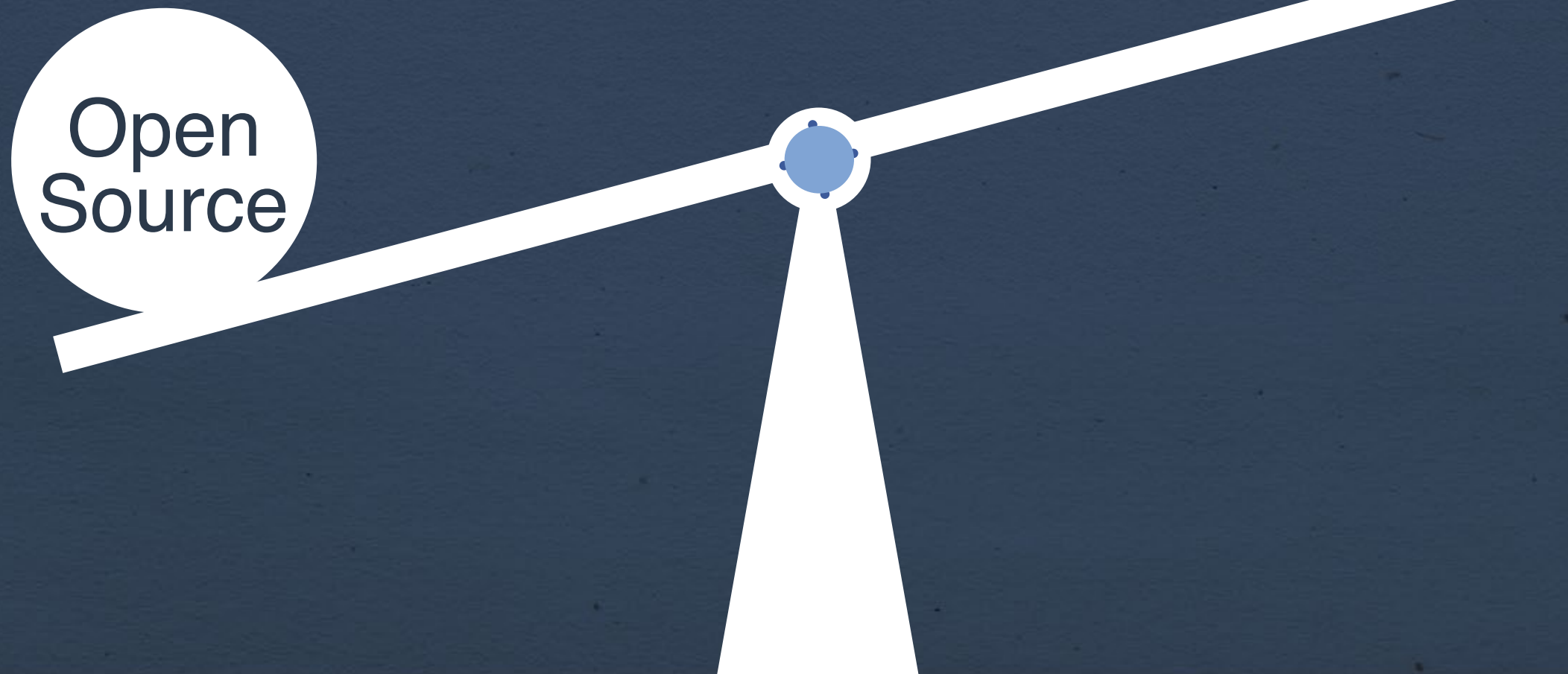
Any pre-existing embedded databases?



Any pre-existing embedded databases?

Key-value stores

1. Berkeley DB
2. SQLite
3. Kyoto TreeDB
4. LevelDB

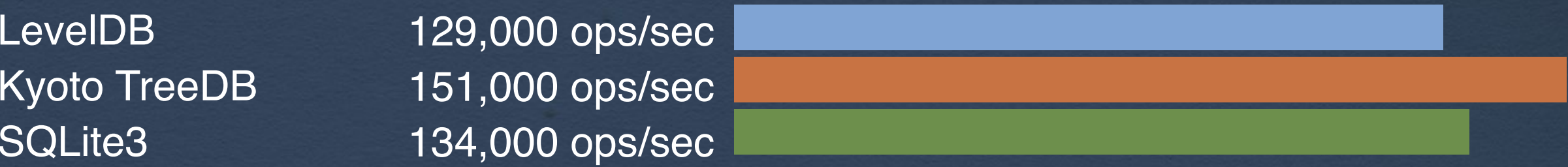


Any pre-existing embedded databases?



Comparison of open source databases

Random Reads



Random Writes



HBase and HDFS (2012)

Random Reads

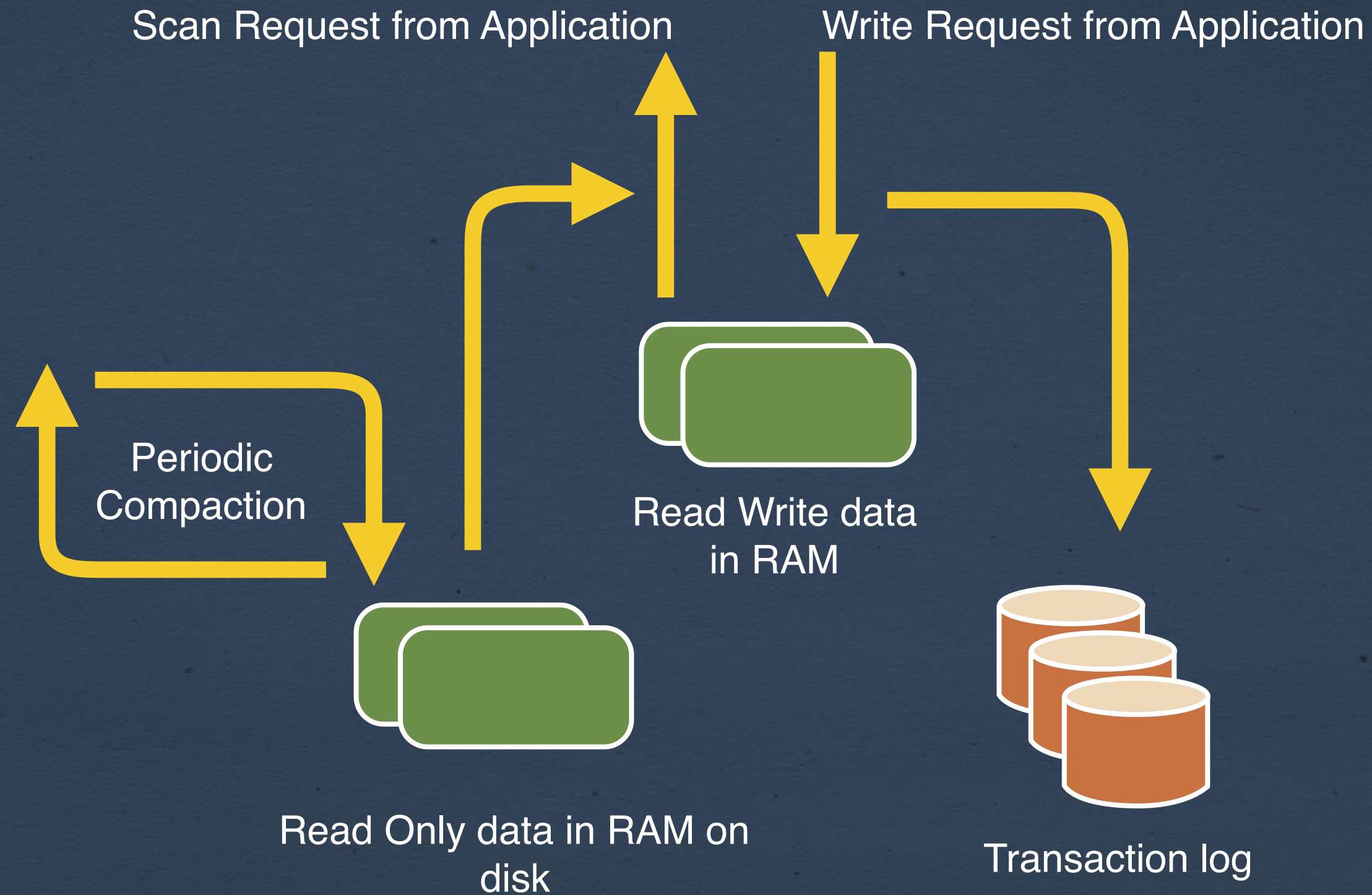
HDFS (1 node)	93,000 ops/sec
HBase (1 node)	35,000 ops/sec



LevelDB

- Code base only had 20k lines
- A neat library with clean interface, easy to embed
- A LSM implementation architecturally compatible with SSD

Log Structured Merge Architecture



LevelDB has low write rates

Facebook Application 1:

- Write rate 2 MB/sec only per machine
- Only one cpu was used for compaction

We developed multithreaded compaction



LevelDB has stalls

Facebook Application 2:

- P99 latencies were tens of seconds
- Single-threaded flush, conflict with compaction

We implemented thread aware compaction

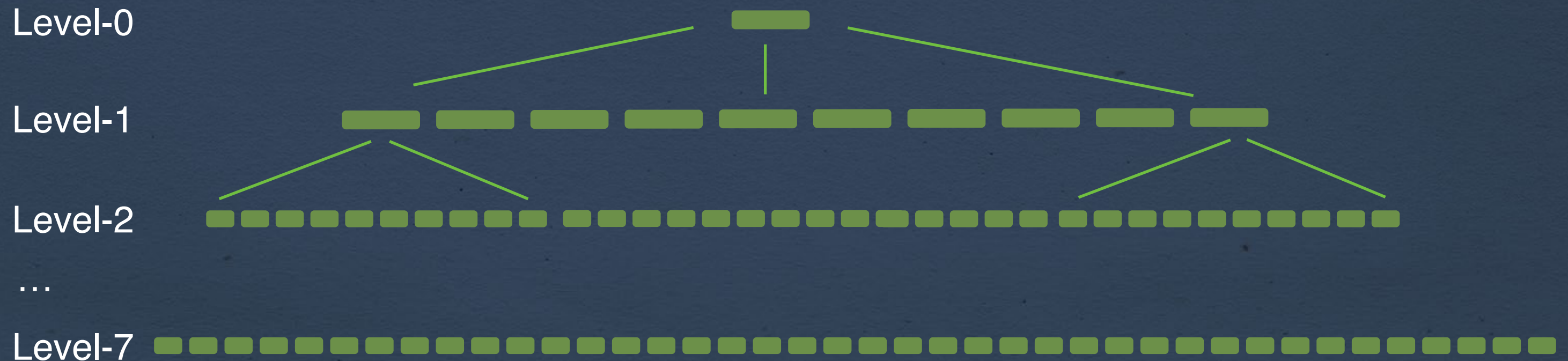
Dedicated thread(s)
to flush memtable

Pipelined memtables

P99 reduced to less
than a second

LevelDB has high write amplification

- Facebook Application 3:
 - Level Style Compaction
 - Worst write amplification of 70 very high



LevelDB Style Compaction

Our solution: lower write amplification

- Facebook Application 3:
 - We implemented Universal Style Compaction
 - Start from newest file, include next file in candidate set if
 - Candidate set size \geq size of next file

Write amplification reduced to **<10**

LevelDB: read modify write = 2X IOs

- Counter increments
 - Get value, value++, Put value
 - LevelDB uses 2X IOPS
- We implemented MergeRecord
 - Put “++” operation in MergeRecord
 - Background compaction merges all MergeRecords
 - Uses only 1X IOPS

LevelDB has a Rigid Design

- LevelDB Design
 - Cannot tune system, fixed file sizes
- We wanted a pluggable architecture
 - Pluggable memtable/sstable for RAM/Flash
 - Pluggable compaction filter (TimeToLive)
 - Pluggable compaction algorithm

The Changes we did to LevelDB

1

Inherited from LevelDB

- Log Structured Merge DB
- Gets/Puts/Scans of keys
- Forward and Reverse Iteration

2

RocksDB

- 10X higher write rate
- Fewer stalls
- 7x lower write amplification
- Blooms for range scans
- Ability to avoid read-modify-write
- Optimizations for flash or RAM
- And many more...

RocksDB is born!

- Key-Value persistent store
- Embedded
- Optimized for fast storage
- Server workloads



RocksDB

What is it not?

- Not distributed
- No failover
- Not highly-available, if machine dies you lose your data
- Focus is single node performance

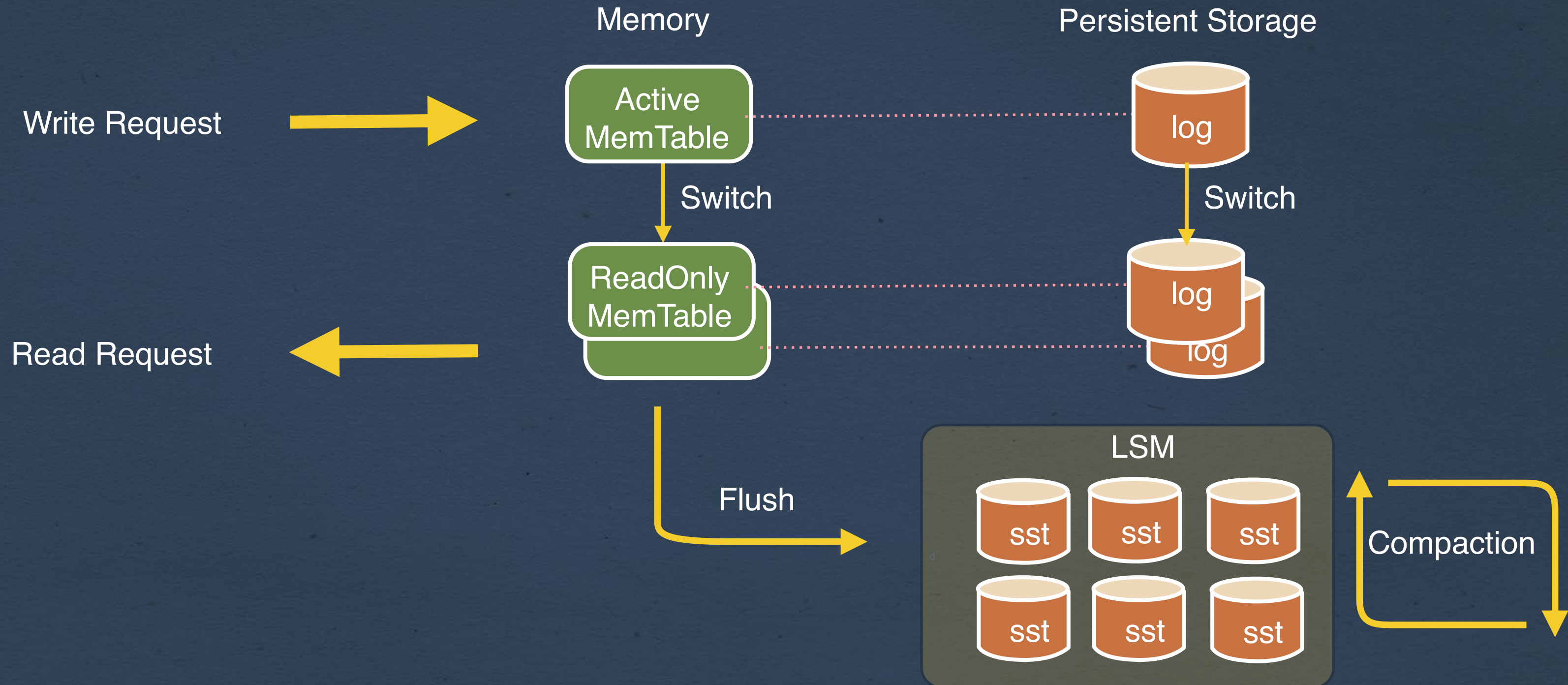


RocksDB API

- Keys and values are arbitrary byte arrays.
- Data are stored sorted by key.
- Update Operations: Put/Delete/Merge
- Queries: Get/Iterator



RocksDB Architecture

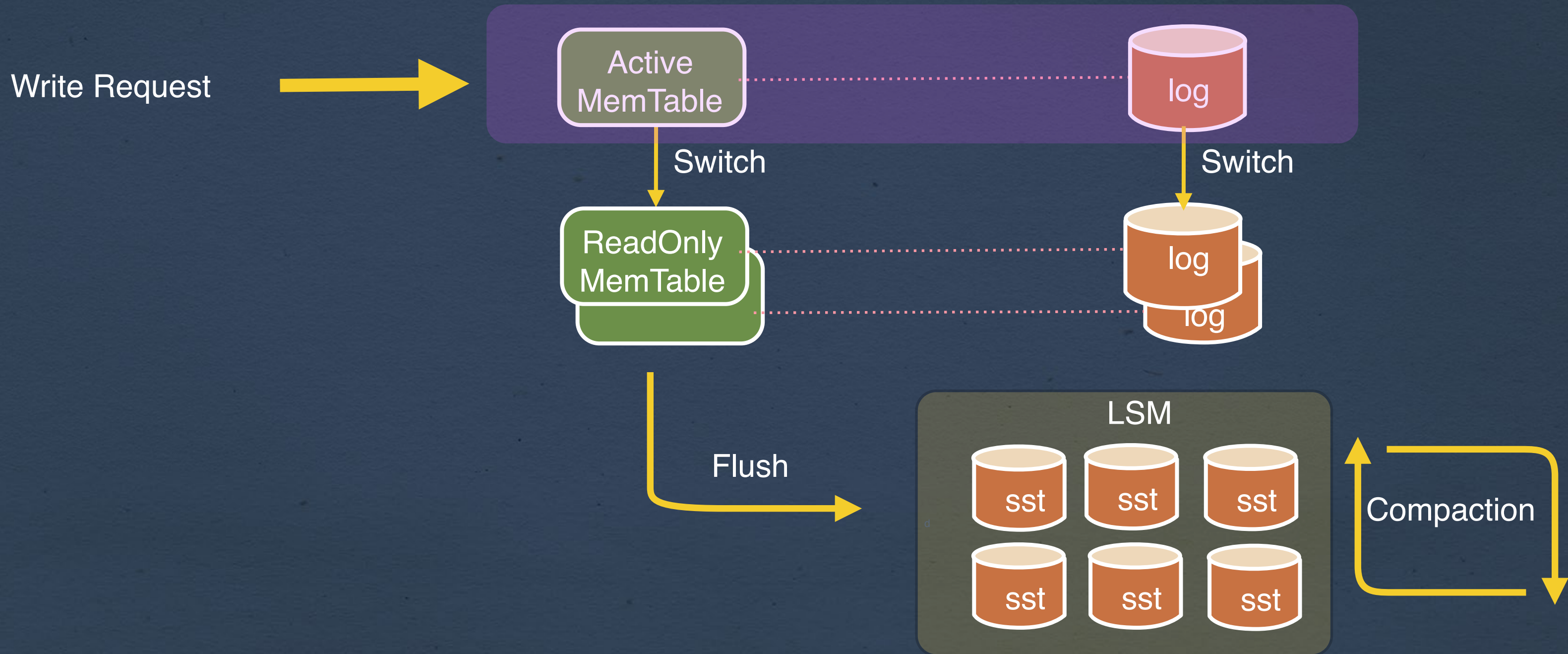


RocksDB -- Writes

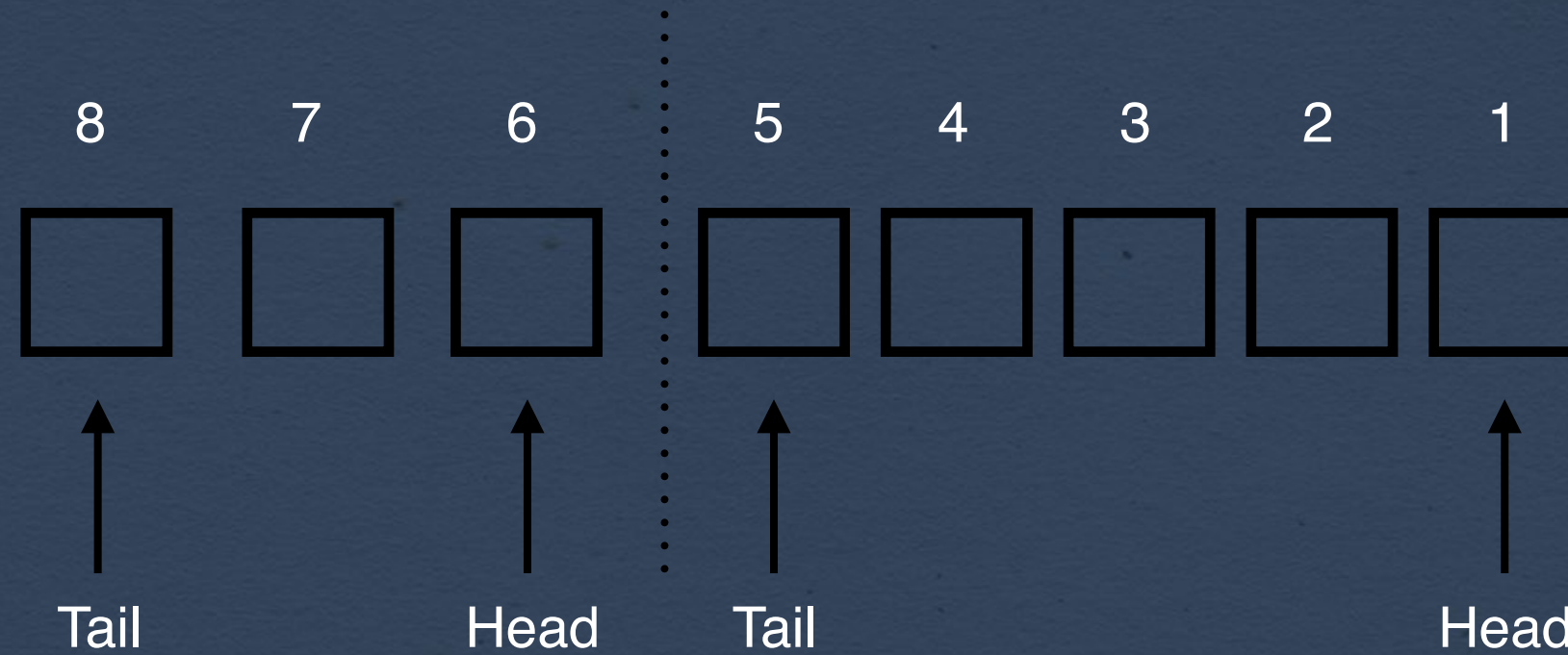
- New Puts are written to memory and optionally to transaction log
- Also can specify log write sync option for each individual write
- We say RocksDB is optimized for writes, what does this mean?



RocksDB Write Path



RocksDB Write Group Commit



RocksDB Atomic Updates

```
#include "rocksdb/write_batch.h"

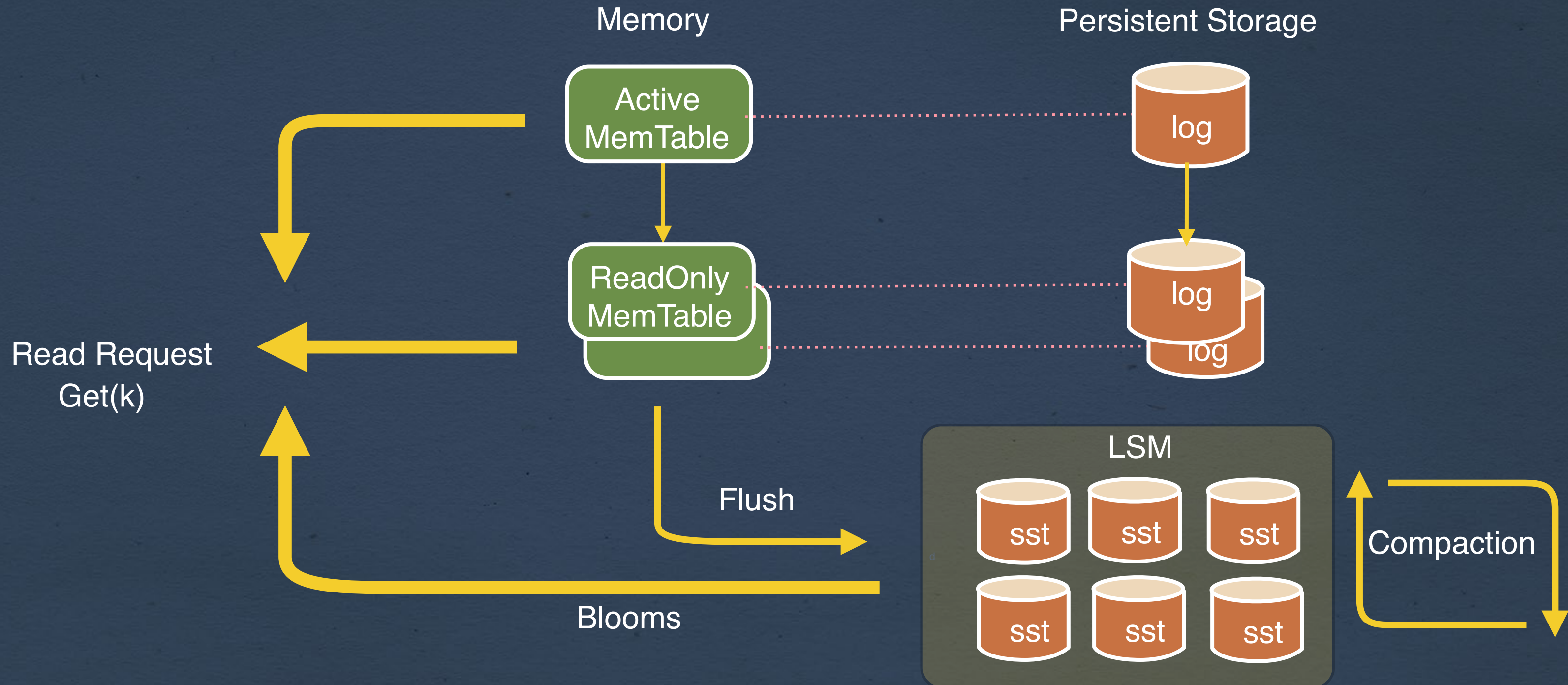
...
std::string value;
rocksdb::Status s = db->Get(rocksdb::ReadOptions(), key1, &value);
if (s.ok()) {
    rocksdb::WriteBatch batch;
    batch.Delete(key1);
    batch.Put(key2, value);
    s = db->Write(rocksdb::WriteOptions(), &batch);
}
```

RocksDB -- Reads

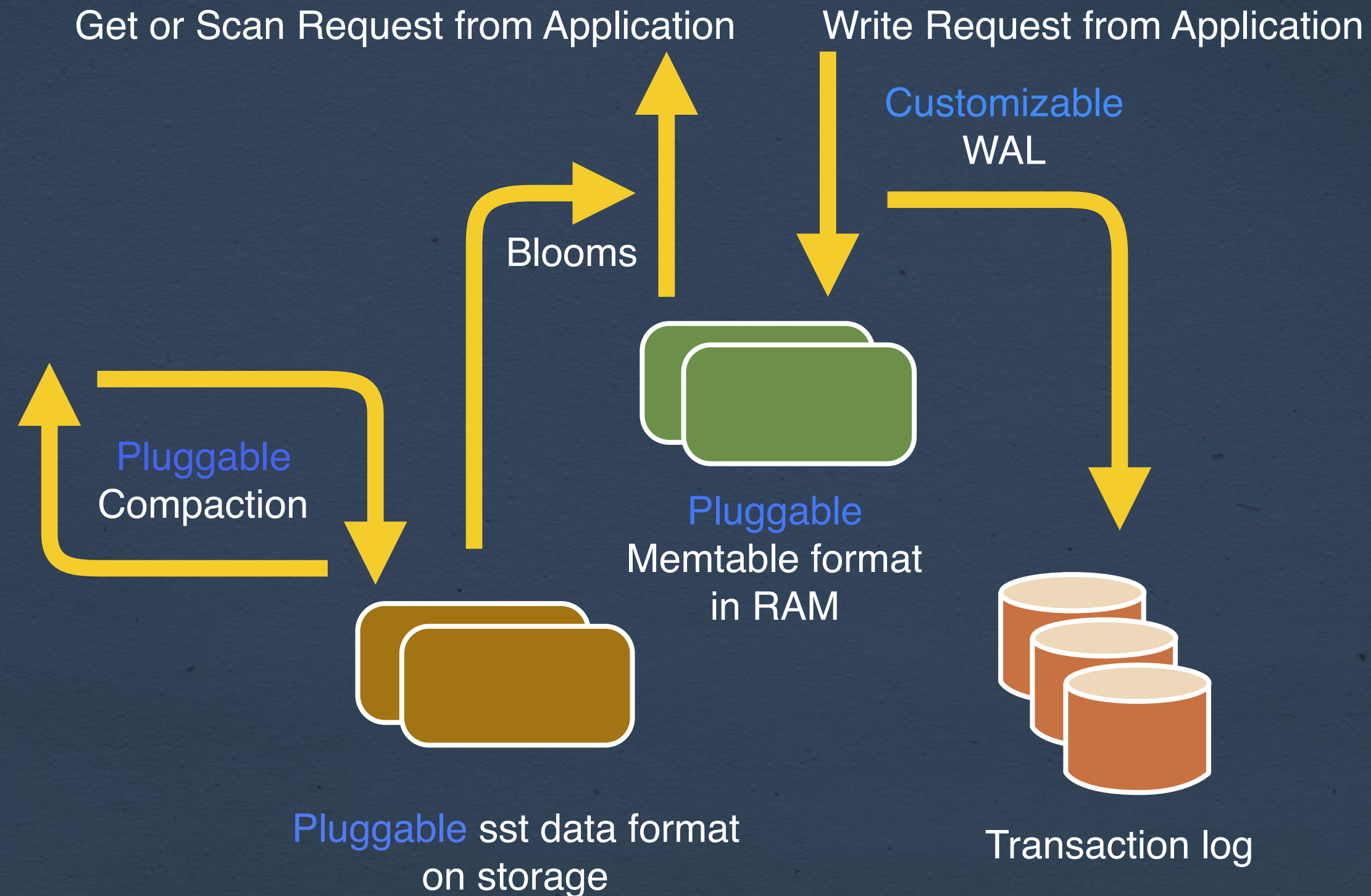
- Data could be in memory or on disk
- Consult multiple files to find the latest instance of the key
- Use bloom filters to reduce IO



RocksDB Read Path



RocksDB: Open & Pluggable



Example: Customizable WALogging

- In-house log-based Replication solution needs to know where a record came from in the multi-master scenario
- put that in payload directly?
- Solution:
 - A Put that speaks only to the log
 - Able to store arbitrary blob in the rocksdb Log stream for annotation

Example: Customizable WALogging

Replication Layer
In one write batch:

Write Request →

PutLogData("I came from Mars")

Put(k1,v1)

Active
MemTable

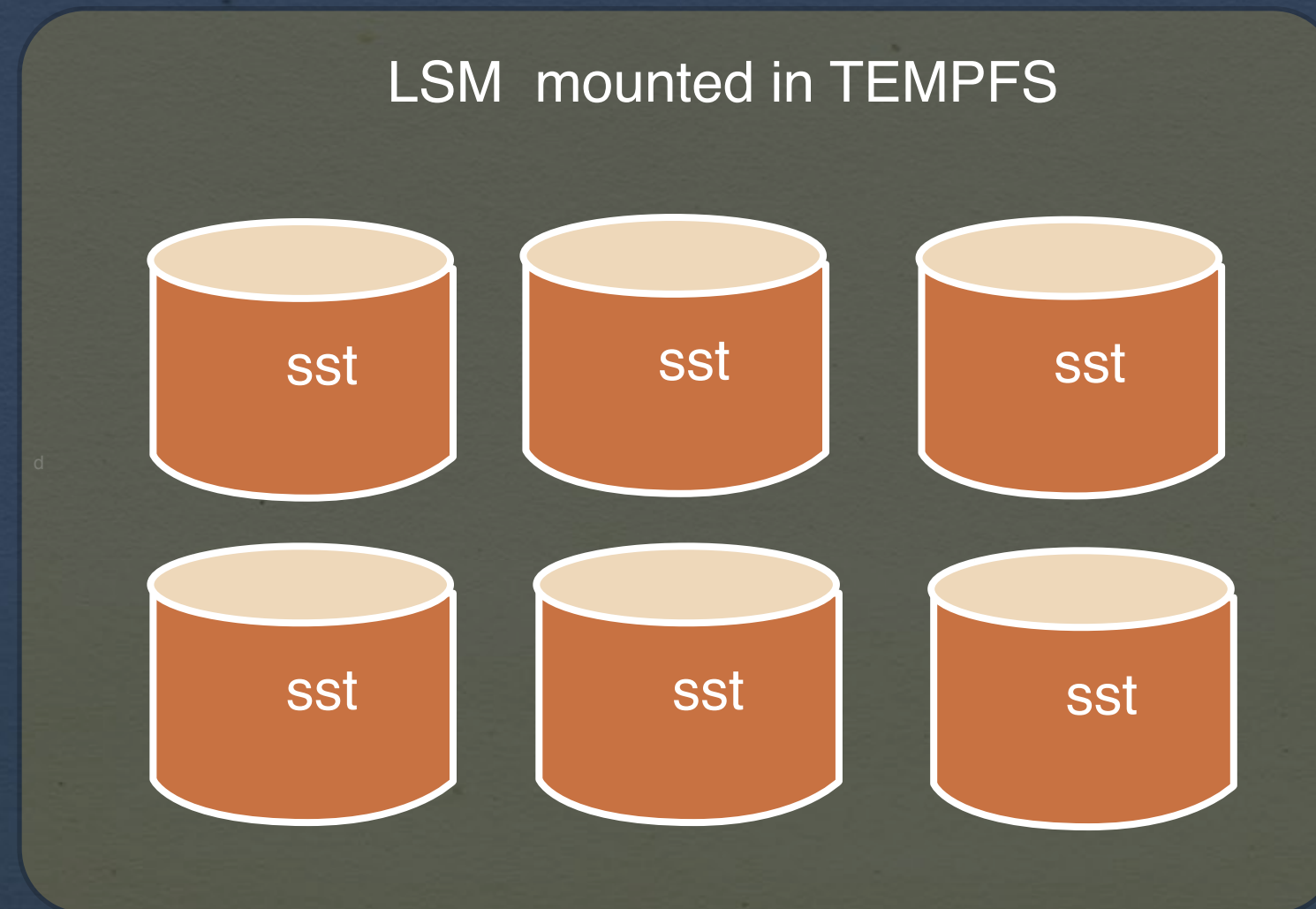


log



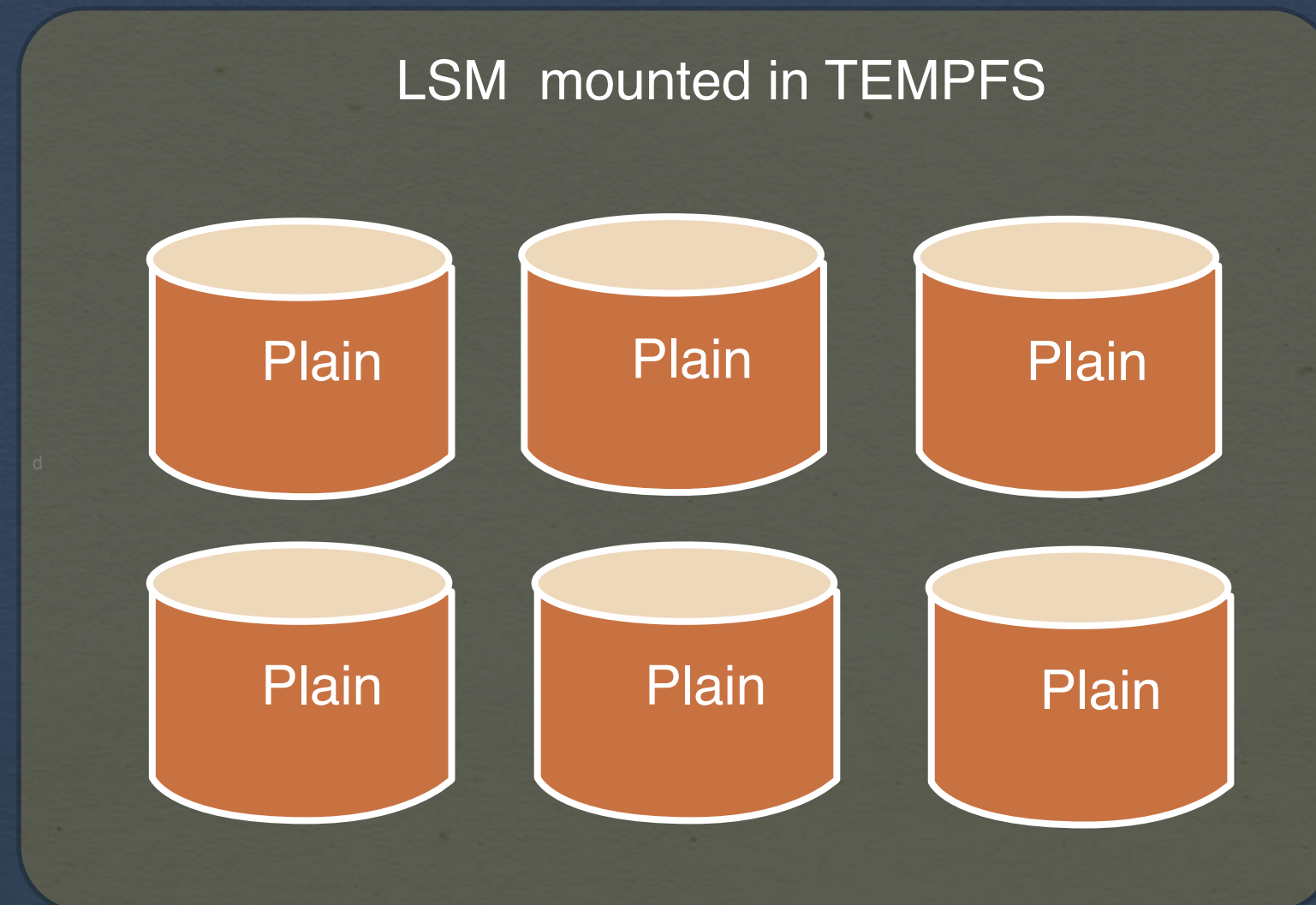
Example: Pluggable SST format

- One Facebook use case needs extremely fast response (micro second) but could tolerate some loss of durability



Example: Pluggable SST format

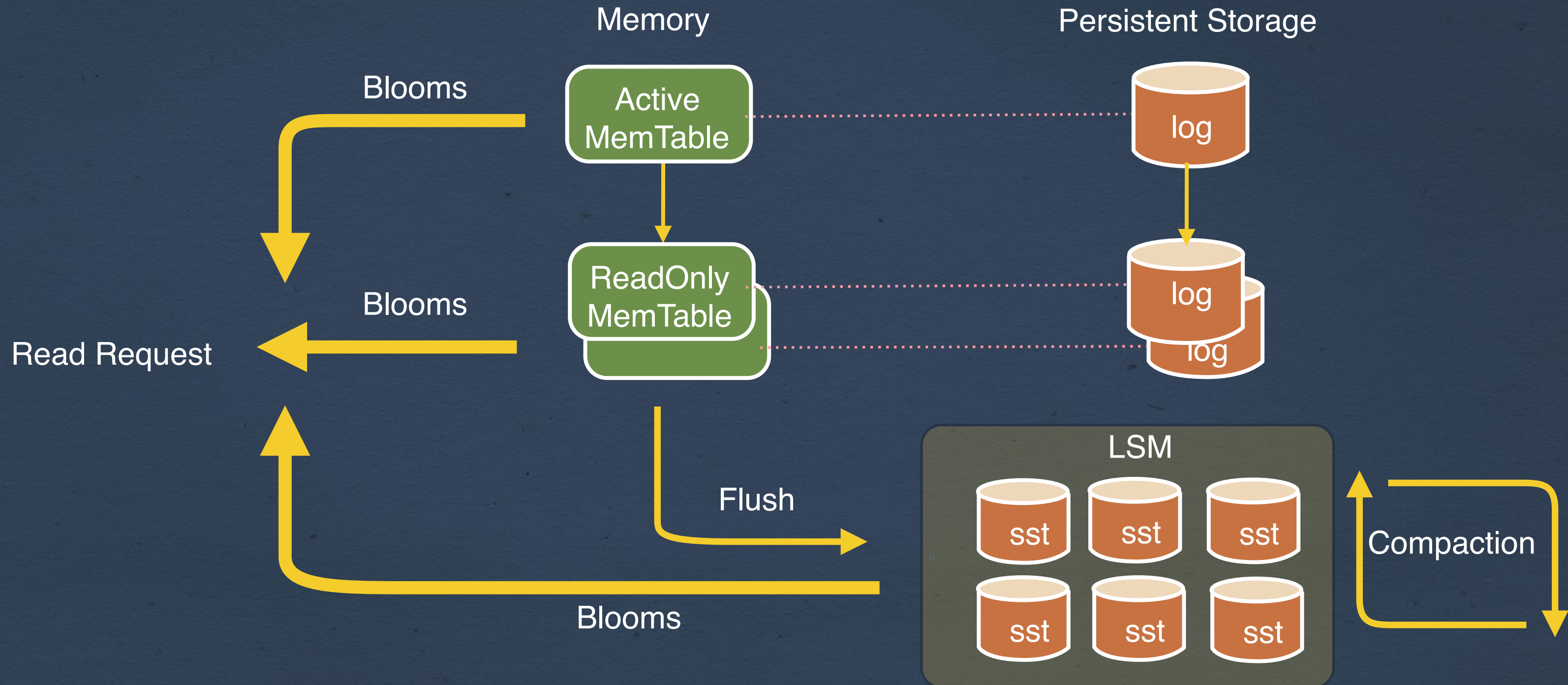
- Still not performant: existing sst format is block based



Example: Blooms for MemTable

- Same use case, after we optimized sst access, memtable lookup becomes a major cost in query
- Cause: Get needs to go through the memtable lookups that eventually return no data
- Solution: Just add a bloom filter to memtable!

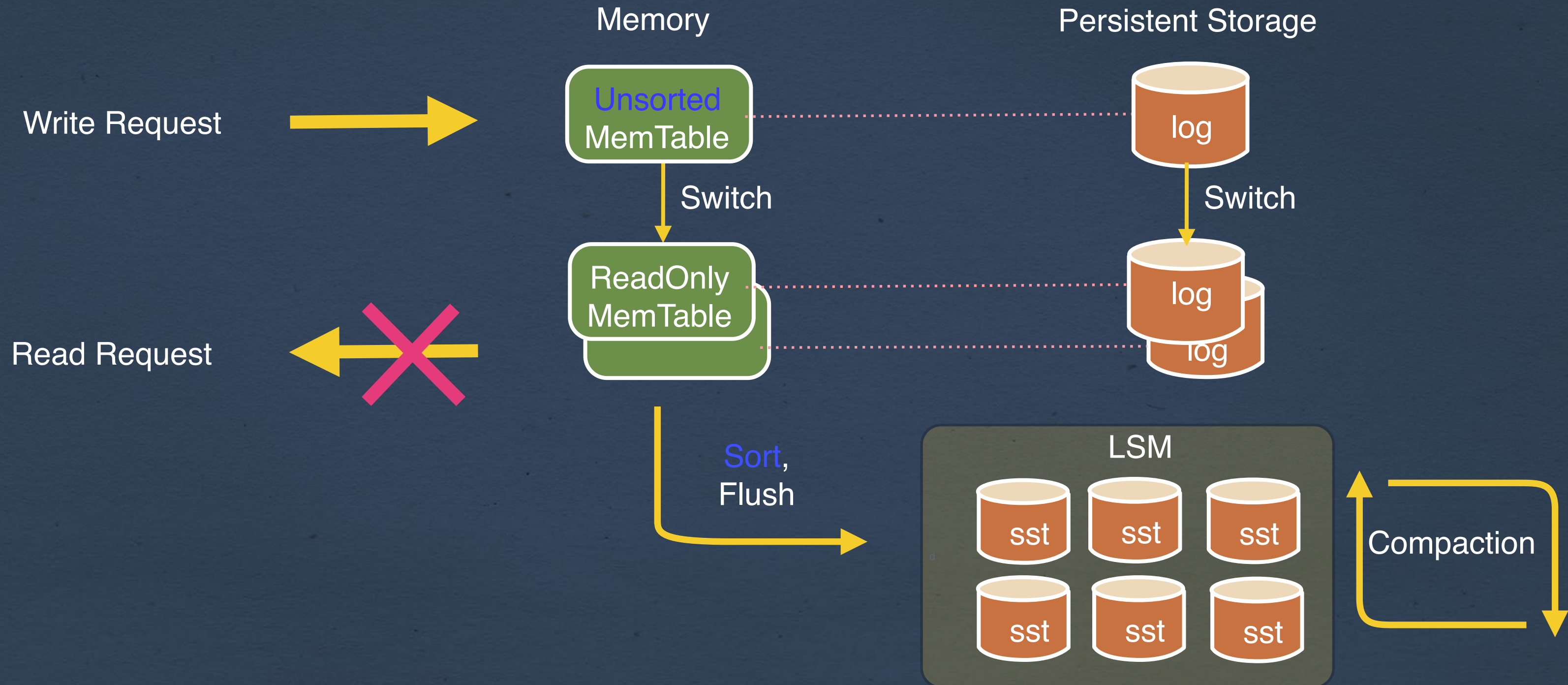
RocksDB Read Path



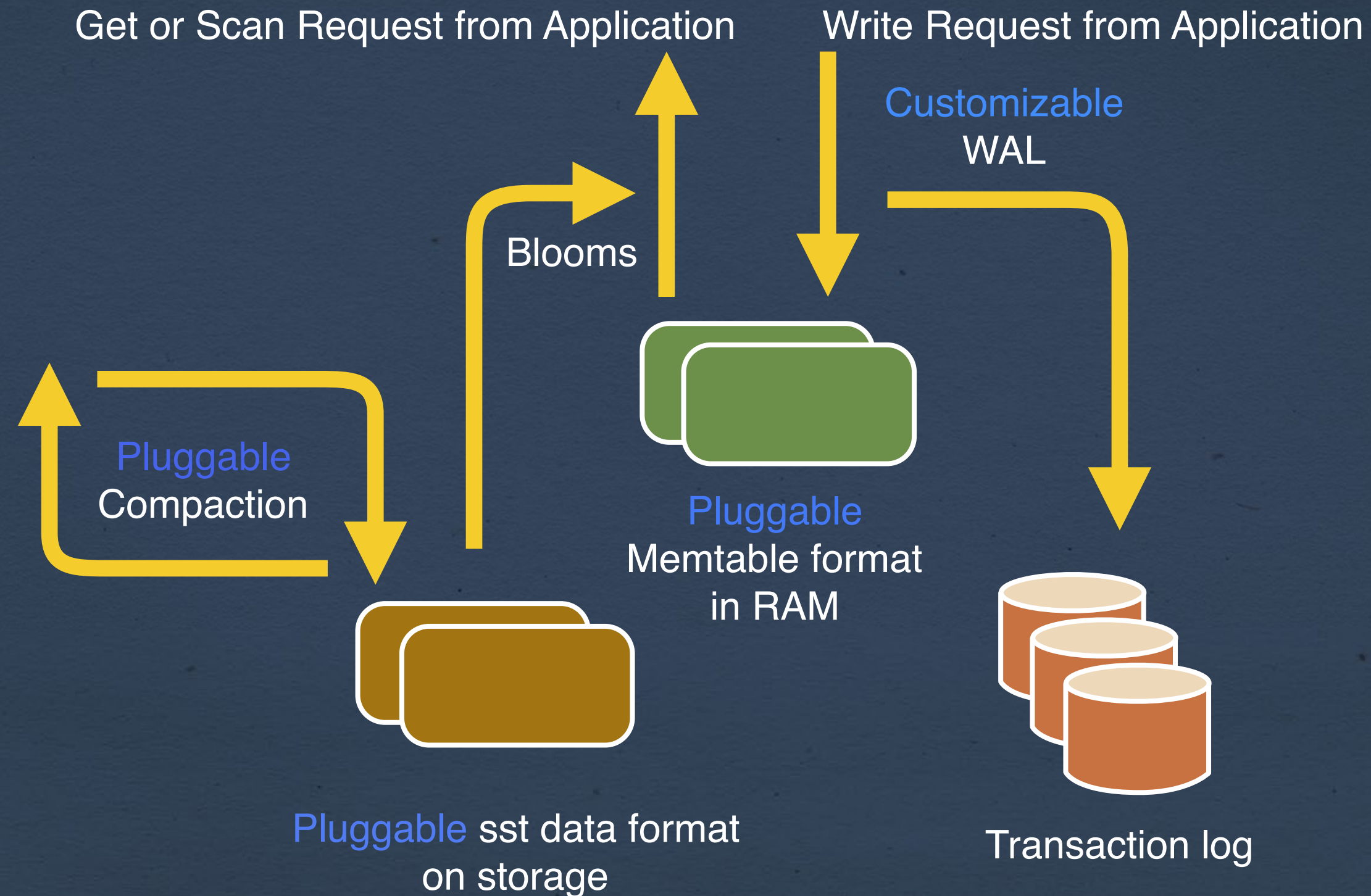
Example: Pluggable memtable format

- Another Facebook use case has a distinct load phase where no query is issued.
- Problem: Ingestion rate is low, with a lot of free CPU and IO bandwidth left
- Cause: Key comparison needed to insert an entry to an ordered memtable
- Solution: A new memtable representation that does not keep keys sorted

Example: Pluggable memtable format



RocksDB: Open & Pluggable



Notable New Features

- TTL
- Backup Support
- Column Family
- Optimistic Transaction
- Pessimistic Transaction (with row level locking)
- Even two phase commit is coming...

Come Hack with us

- RocksDB is Open Sourced
 - <http://rocksdb.org>



- Help us HACK RocksDB



rocksdb.org/meetup.html