RocksDB

http://rocksdb.org/

Embedded Key-Value Store for Flash and Faster Storage

Siying Dong

Database Engineering@Facebook

Overview

- RocksDB and its architecture
- Example use case in facebook.
- Why is RocksDB flash-friendly?
- How to run benchmark

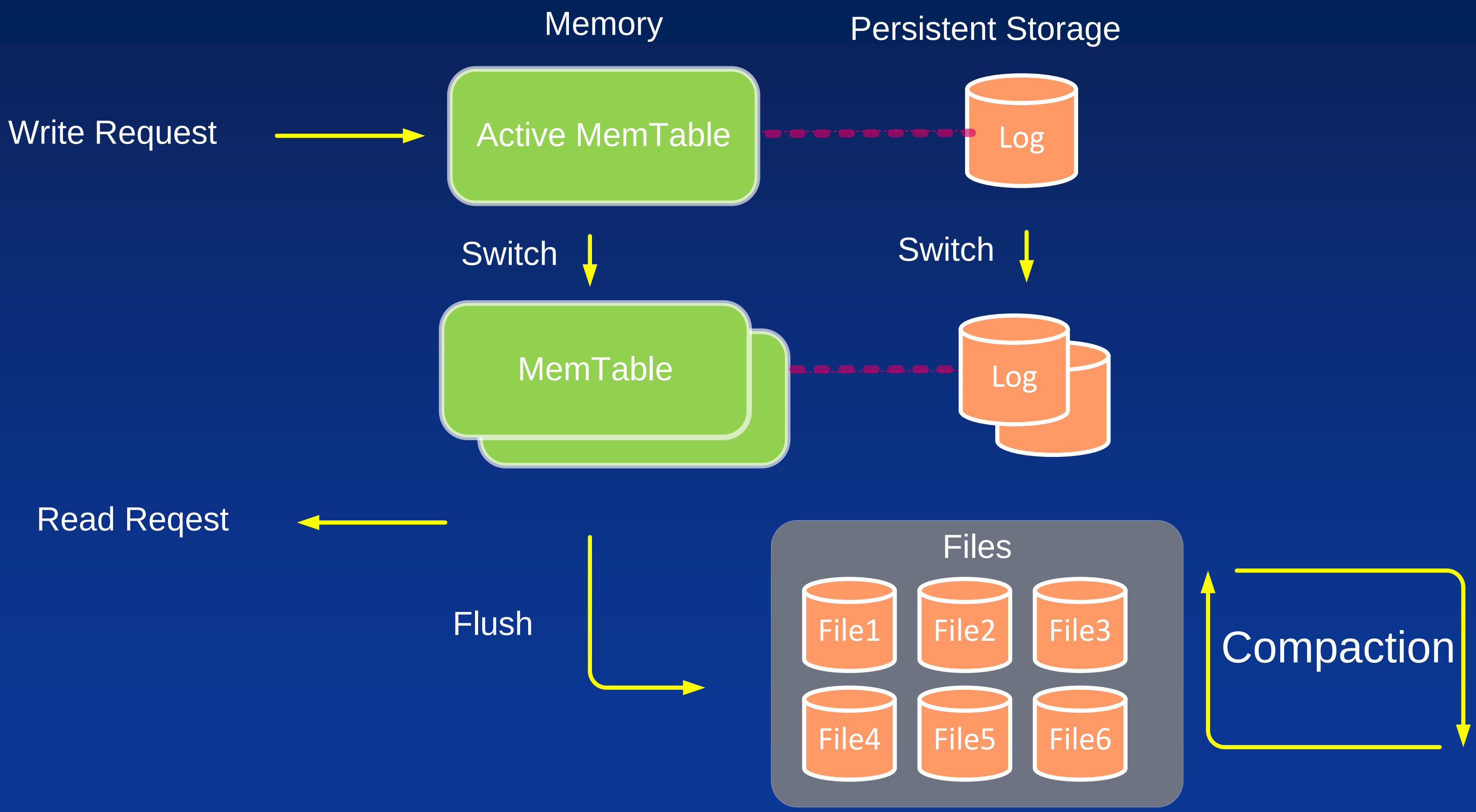
What is RocksDB

- Key-Value persistent store
- Embedded
- Optimized for fast storage
- Optimized for server workloads
- Open-Source, builds on LevelDB code base, written in C++

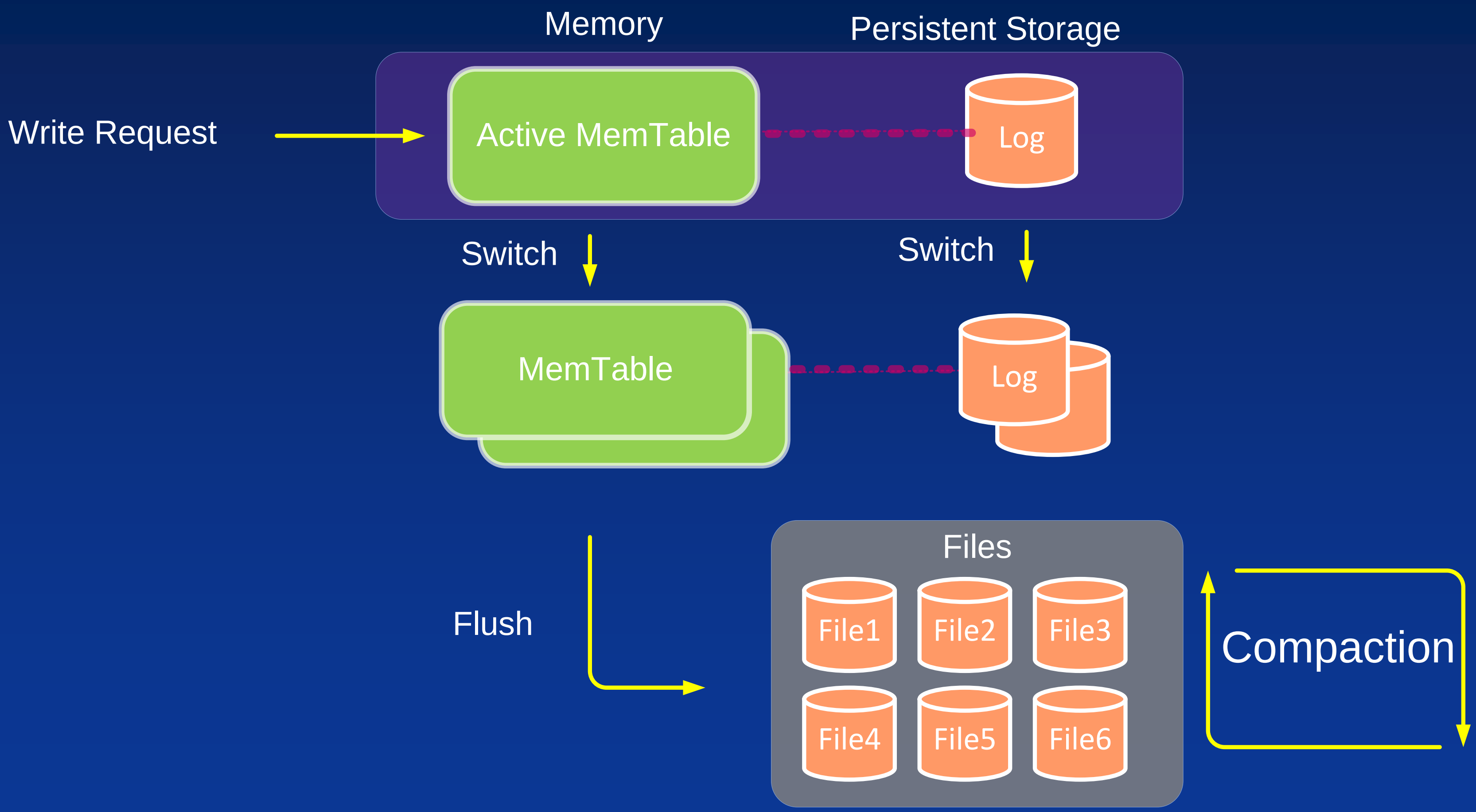
RocksDB API

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

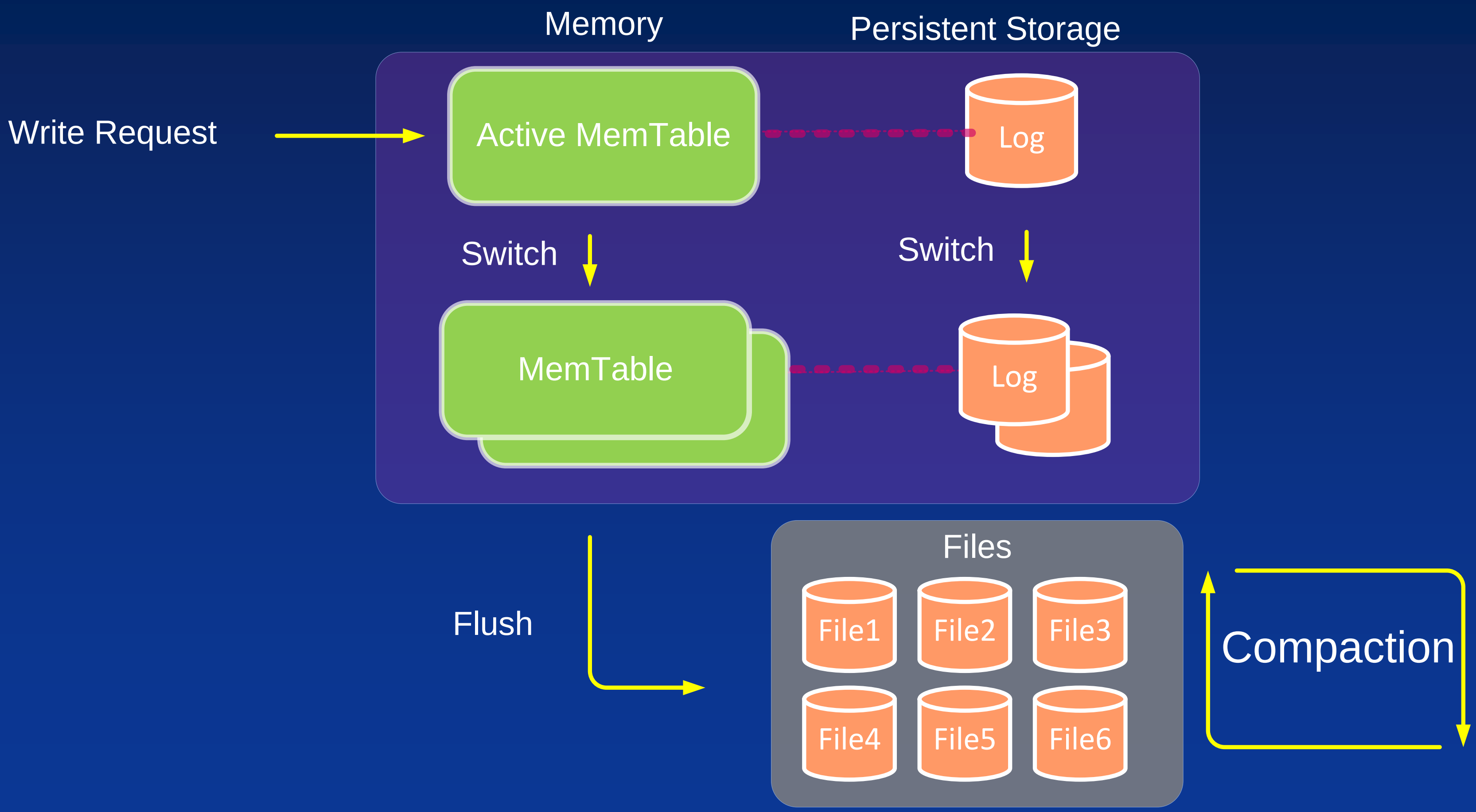
RocksDB Architecture



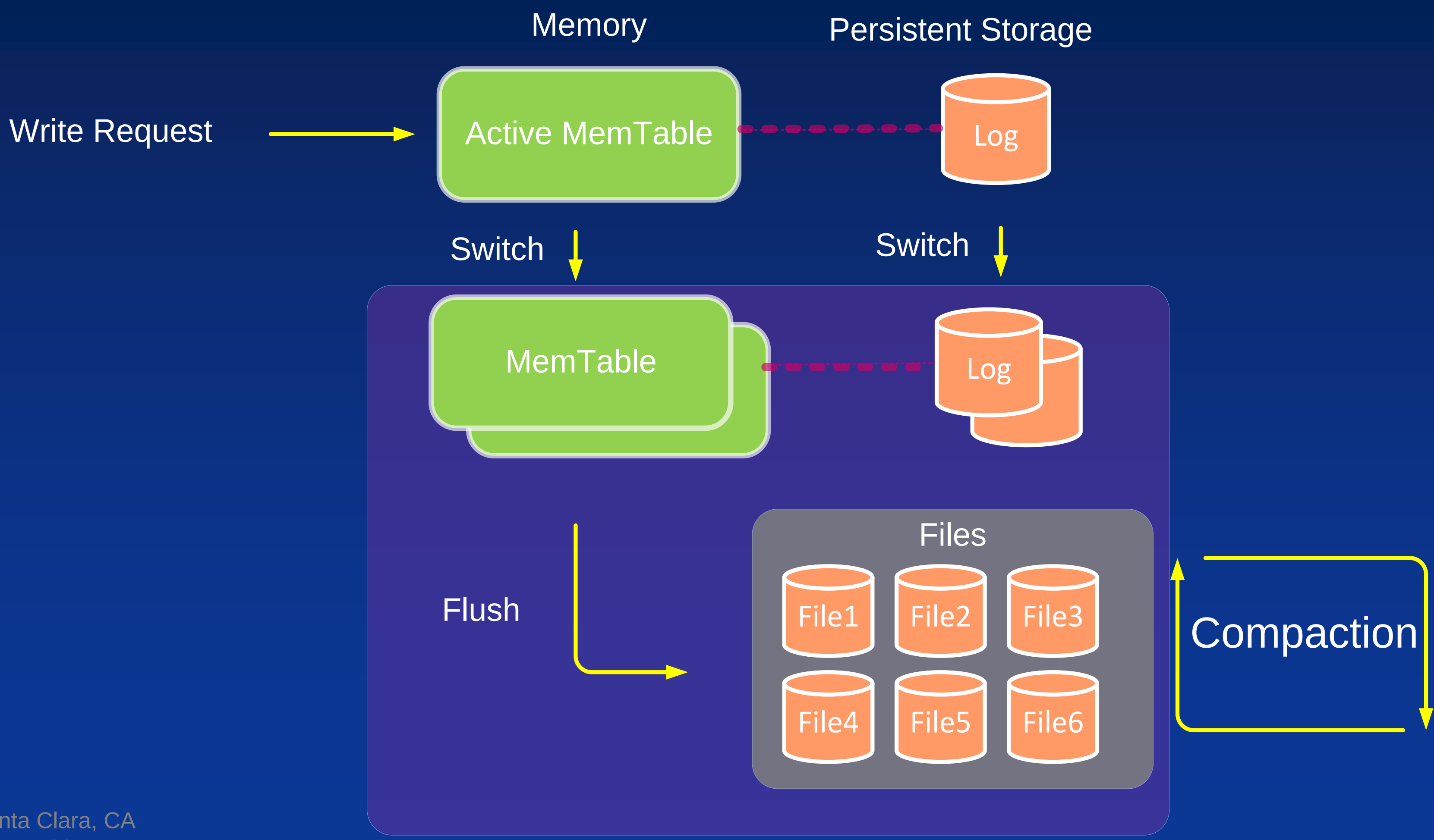
Write Path (1)


Memory
Persistent Storage
Write Request
Active MemTable
Log
Switch
Switch
MemTable
Log
Flush
Files
File1
File2
File3
File4
File5
File6
Compaction

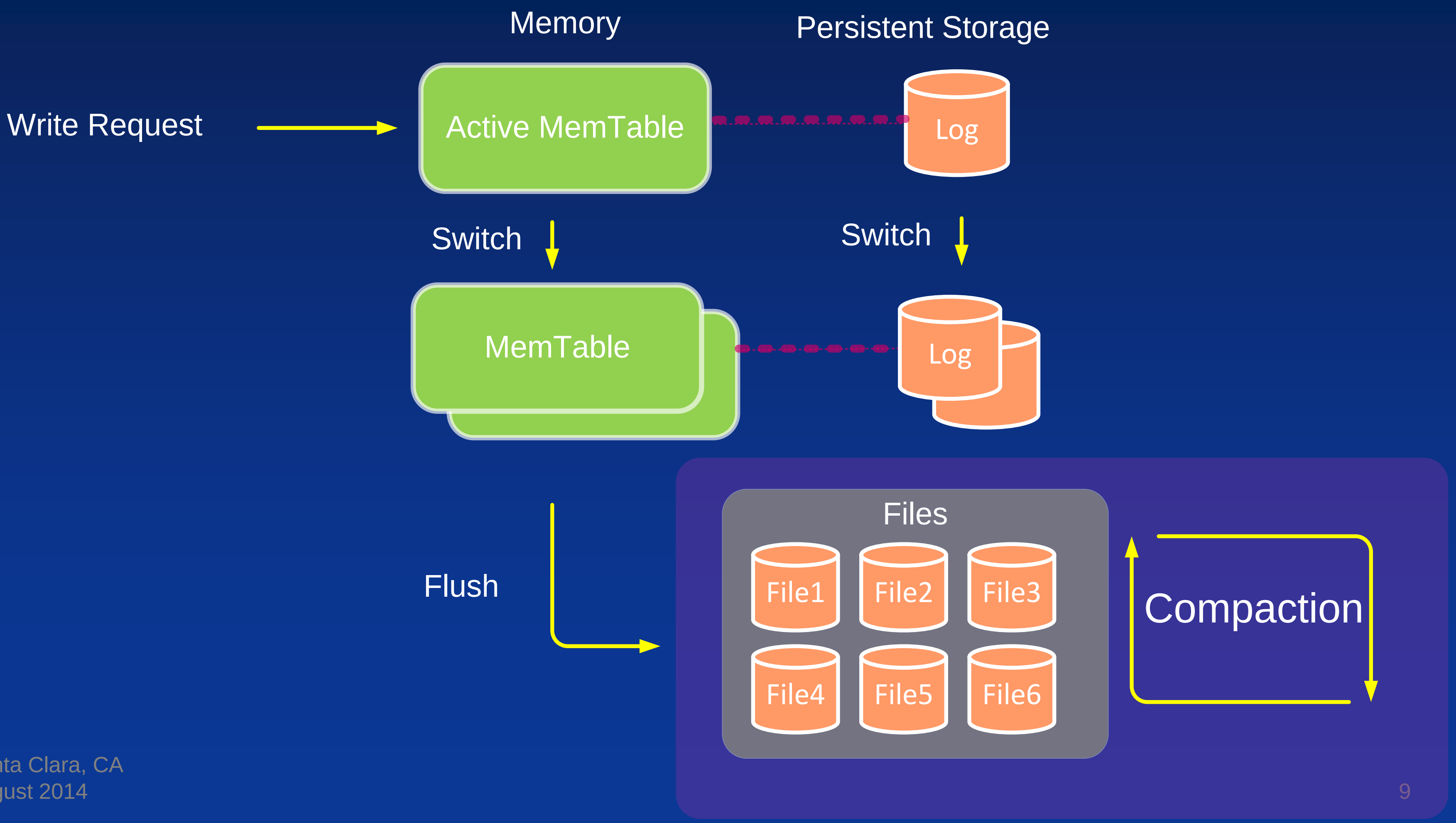
Write Path (2)


Memory
Persistent Storage
Write Request
Active MemTable
Log
Switch
Switch
MemTable
Log
Flush
Files
File1
File2
File3
File4
File5
File6
Compaction

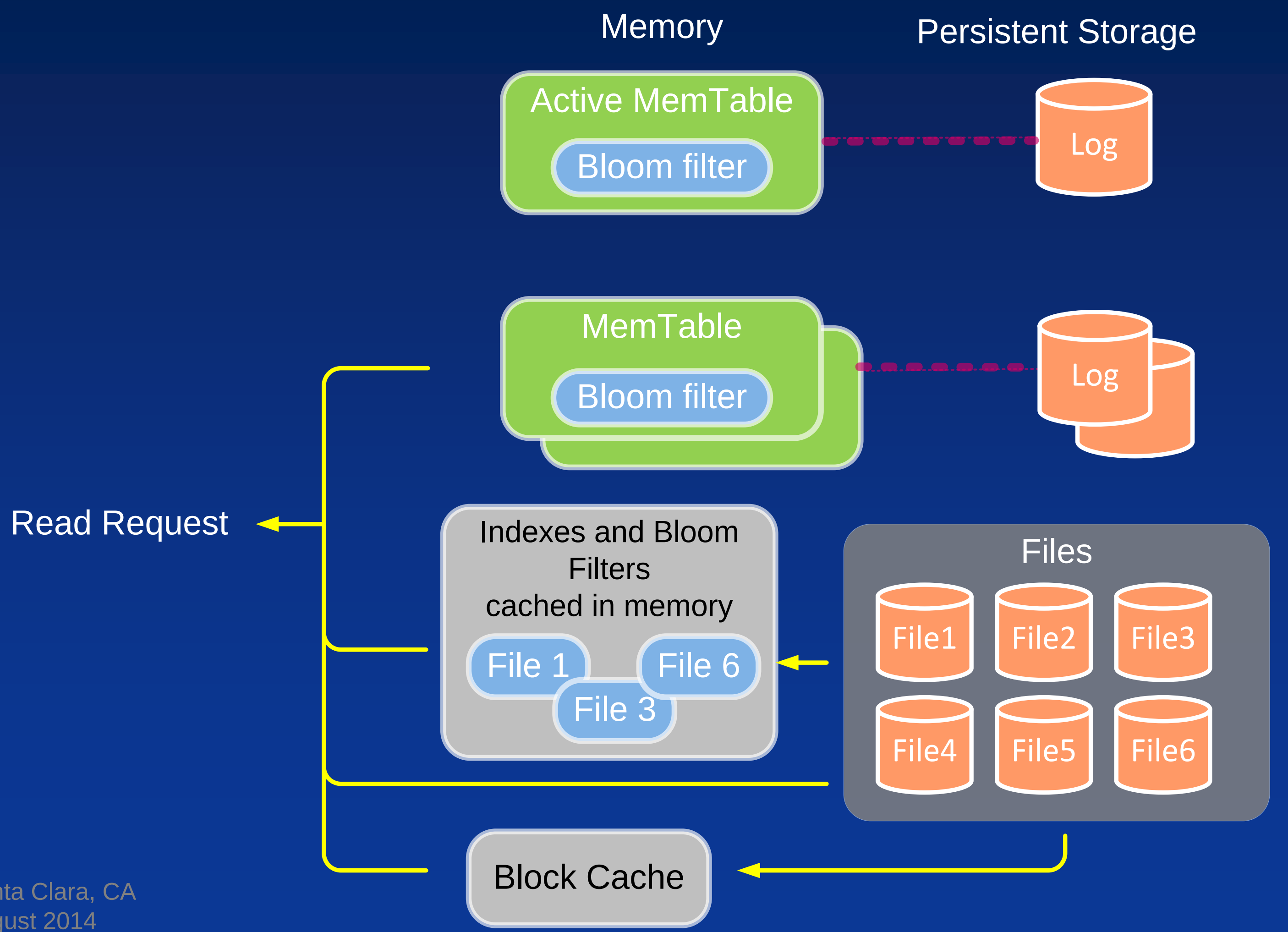
Write Path (3)


Memory
Persistent Storage
Write Request
Active MemTable
Log
Switch
Switch
MemTable
Log
Files
File1
File2
File3
File4
File5
File6
Flush
Compaction

Write Path (4)


Memory
Persistent Storage
Write Request
Active MemTable
Log
Switch
Switch
MemTable
Log
Flush
Files
File1
File2
File3
File4
File5
File6
Compaction

Read Path

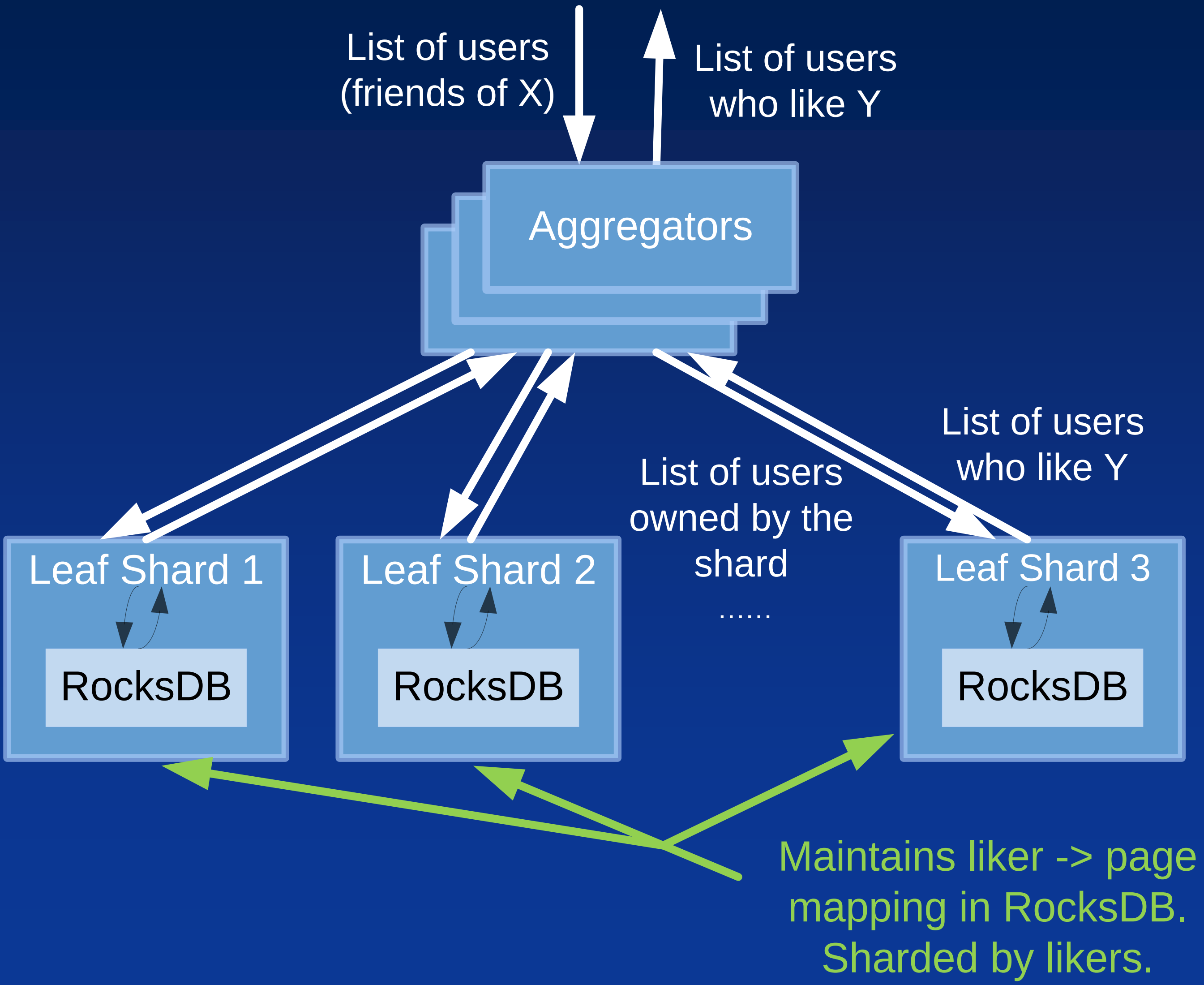


Memory
Persistent Storage
Active MemTable
Bloom filter
Log
MemTable
Bloom filter
Log
Read Request
Indexes and Bloom Filters cached in memory
File 1
File 6
File 3
Files
File1
File2
File3
File4
File5
File6
Block Cache

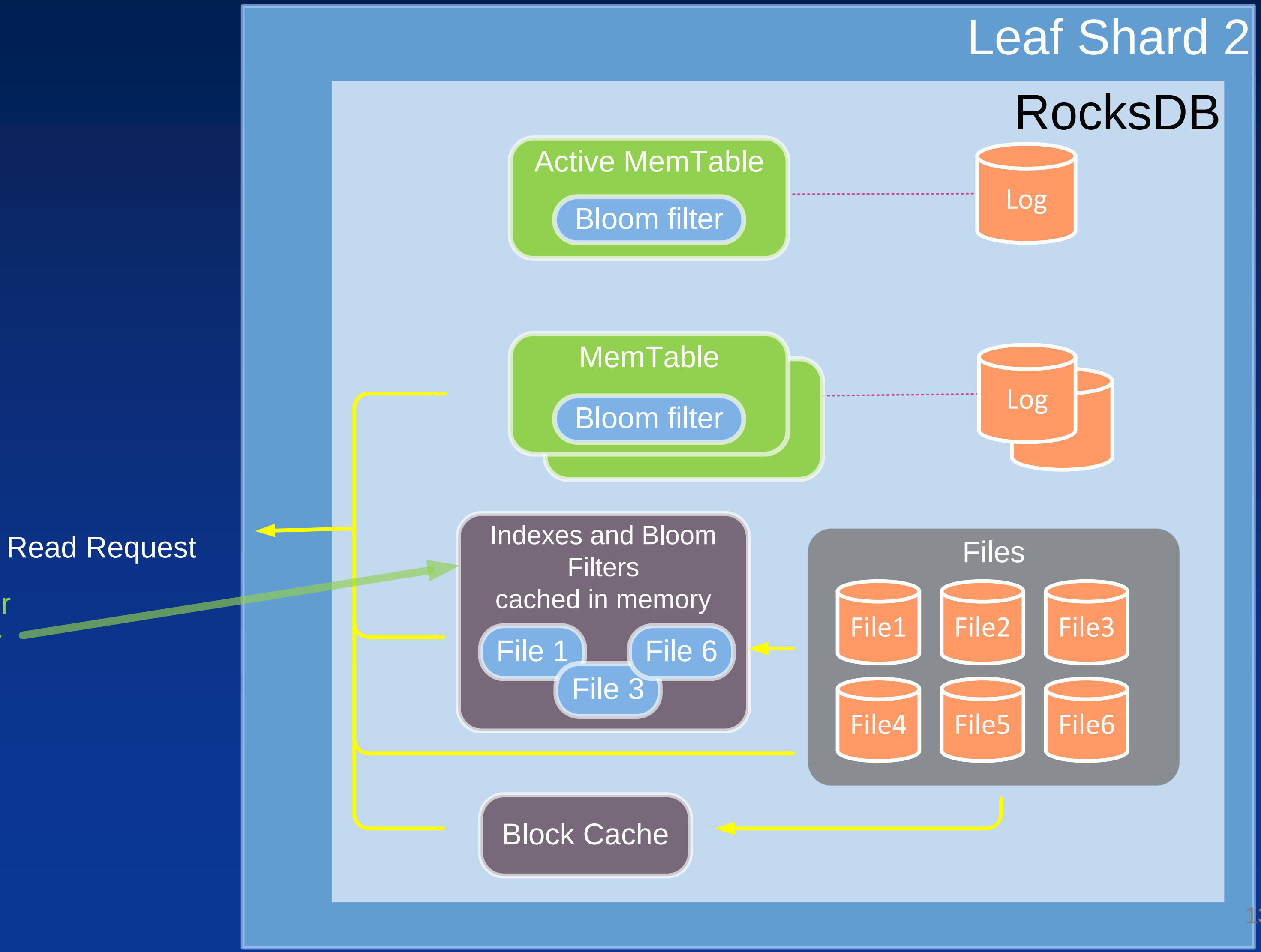
Example Use Case: Find all friends of user X who like Page Y

- Need to store liker-page mapping for fast look-up.
- Choice one: put the mapping in memory
  - Fast
  - Need to keep more replicas than needed by queries
- Choice two: put the mapping on flash
  - Slower, but still fast
  - One replica can handle fewer queries
  - Fewer hosts for one replica of data

Example Use Case: Find all friends of user X who like page Y

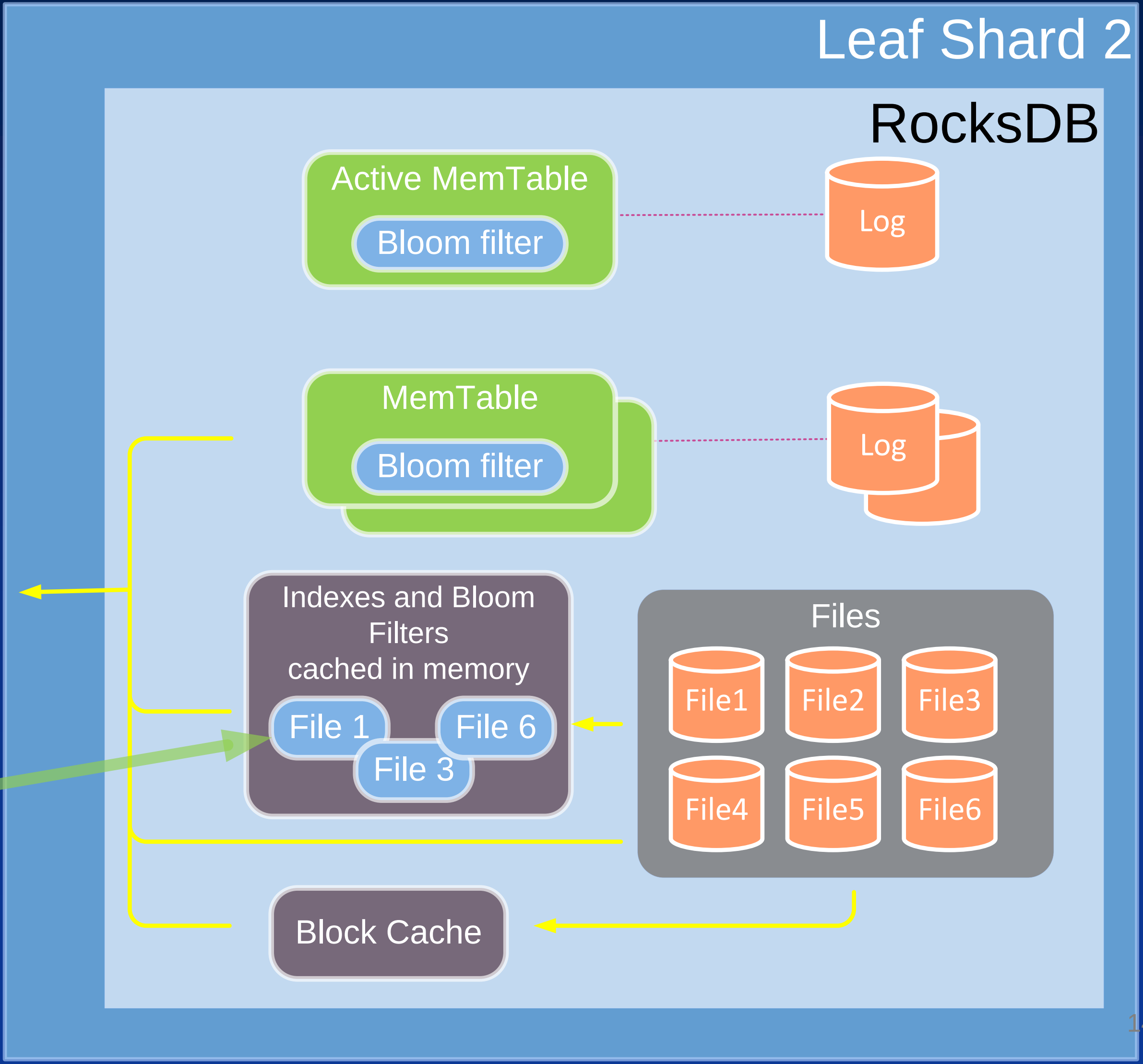

List of users (friends of X)
List of users who like Y
Aggregators
List of users owned by the shard
......
List of users who like Y
Leaf Shard 1
RocksDB
Leaf Shard 2
RocksDB
Leaf Shard 3
RocksDB
Maintains liker -> page mapping in RocksDB. Sharded by likers.

Example Use Case: Find all friends of user X who like Page Y



Preload indexes for all files in memory

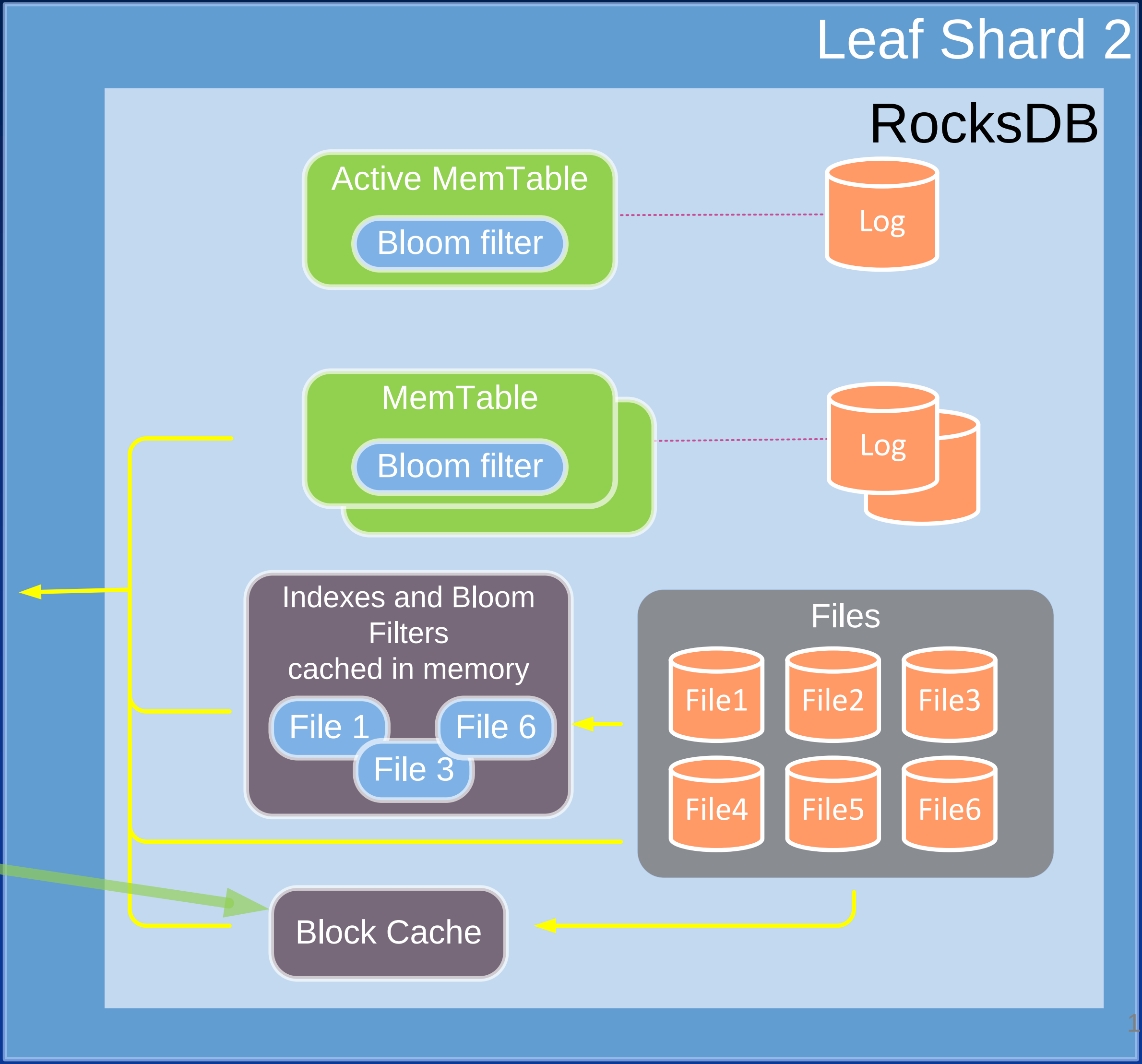
Example Use Case: Find all friends of user X who like Page Y



Read Request

Use hash indexes To reduce CPU

Example Use Case: Find all friends of user X who like Page Y



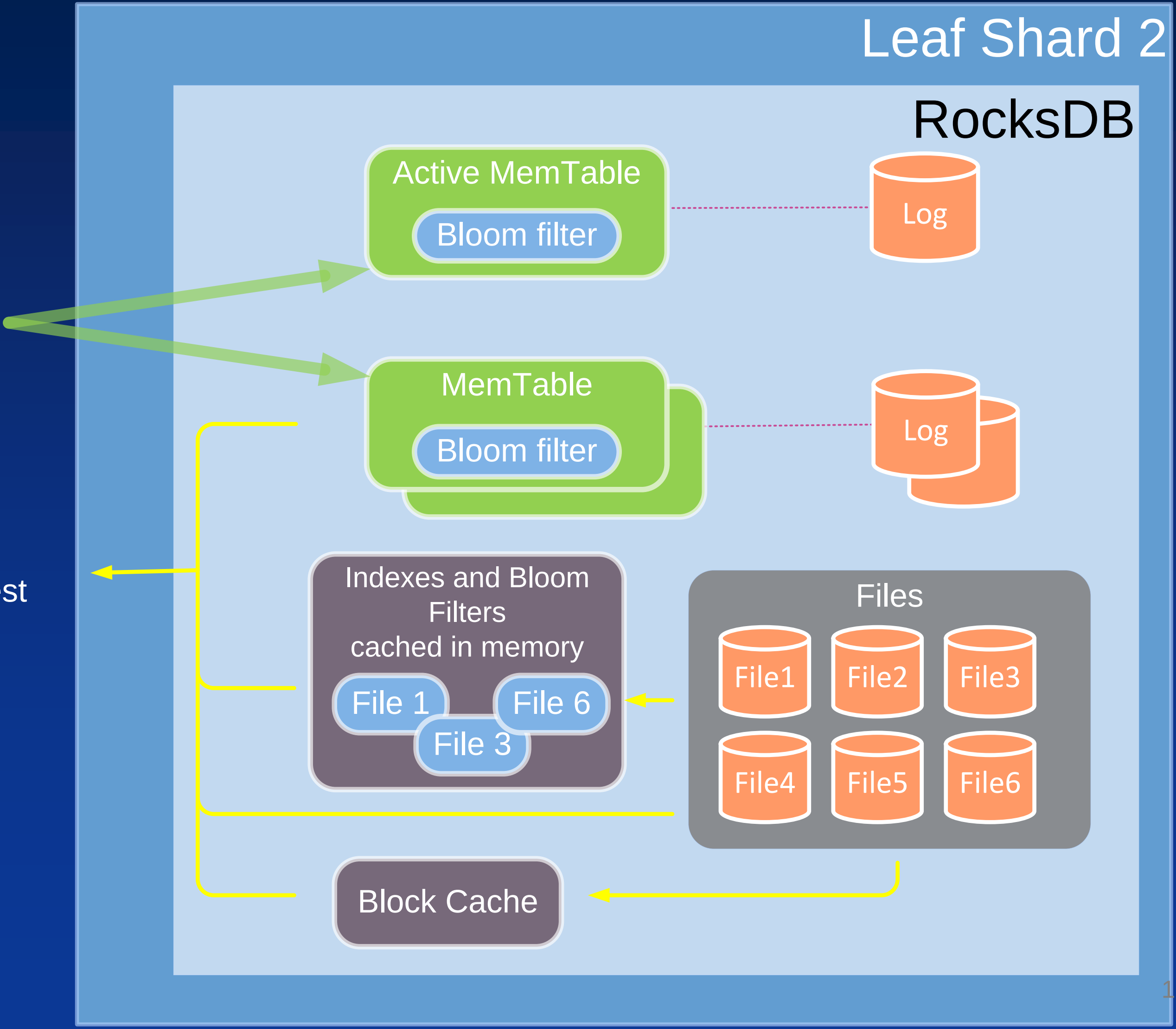
Read Request

Very small block cache to save memory

Example Use Case: Find all friends of user X who like Page Y

Mostly Covered by Bloom Filter

Read Request



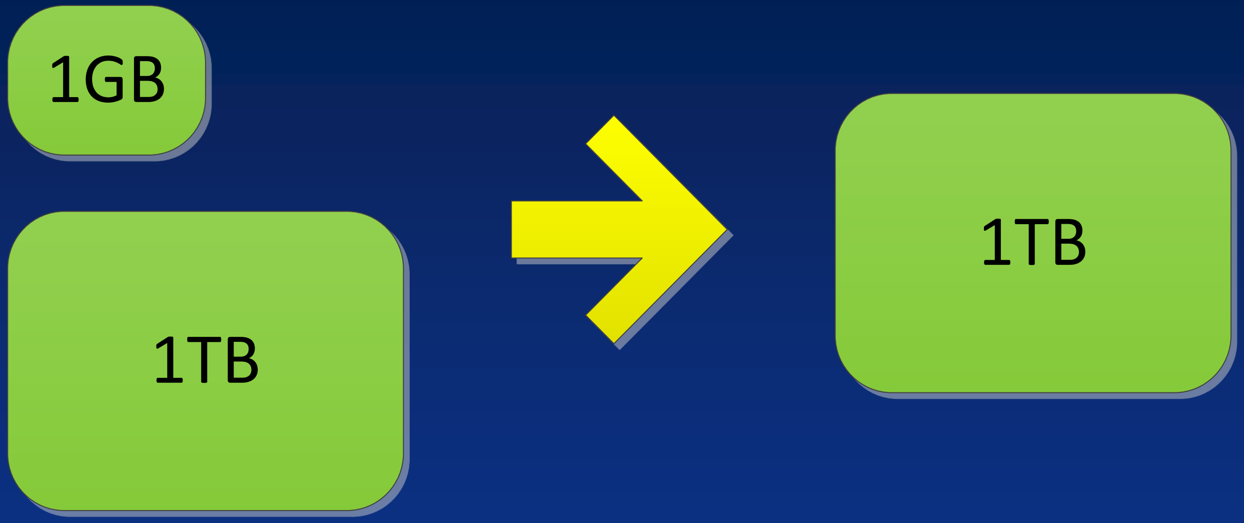
Why is RocksDB Friendly to Flash Devices?

Reason 1:

Tunable between device wear-out and read latency

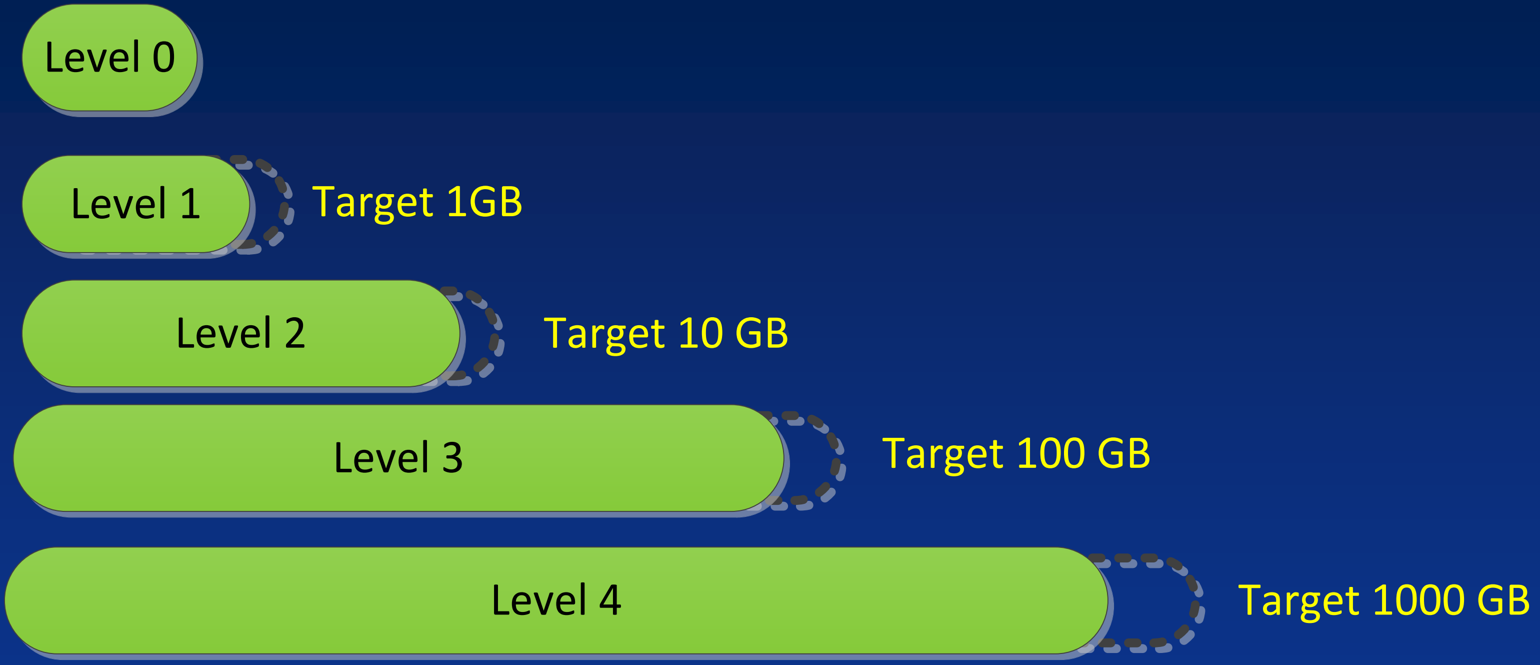
- Tunable compaction to trade-off
  - Read Amplification
  - Write Amplification
  - Space Amplification

Compaction 1: compact to one file



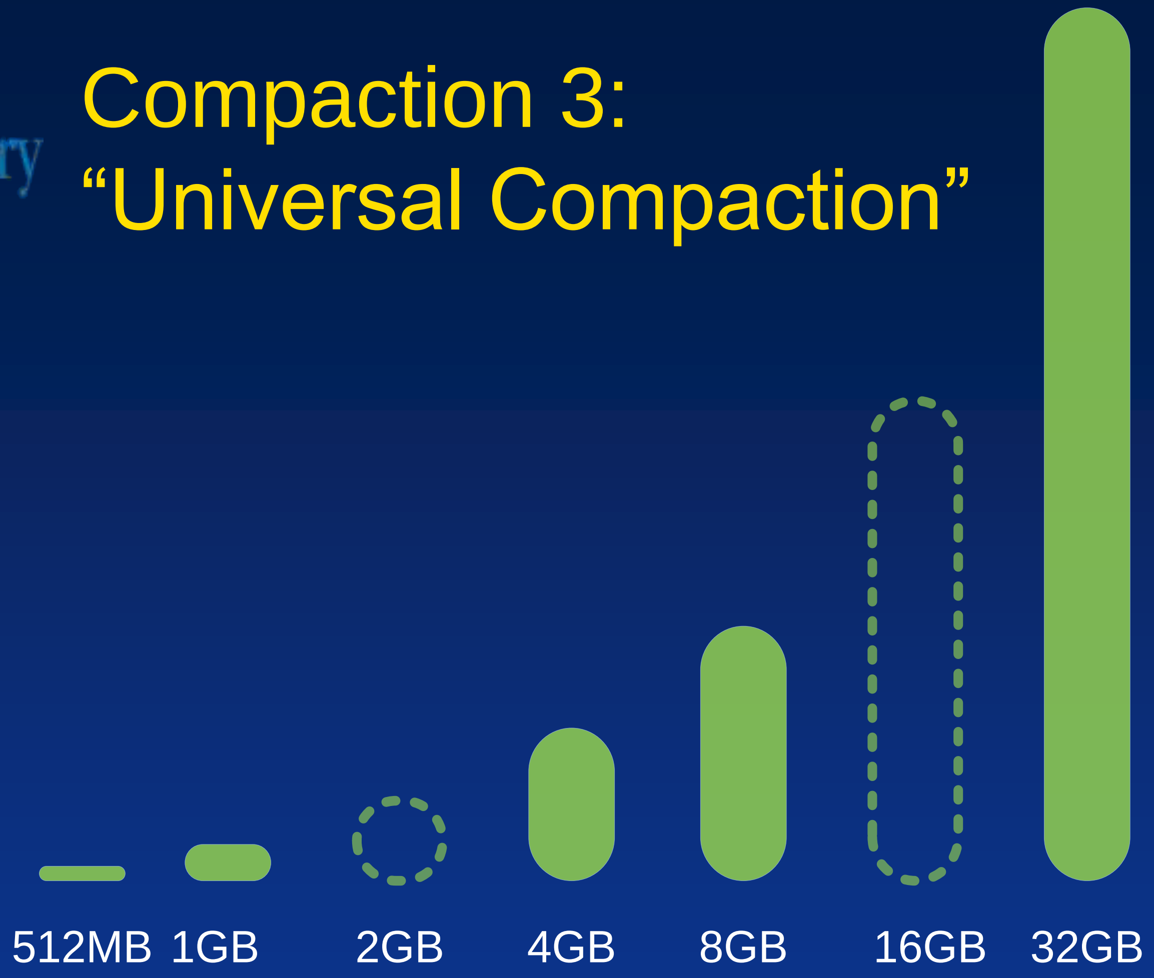
- Write Amplification = 1000
- Read Amplification = 2 or 1 using bloom
- Space Amplification = 1.001
- Need Double Space for compaction

Compaction 2: Leveled-Compaction



- Read Amplification: number of levels or 1 (using bloom)
- Write Amplification: 10 * number of levels
- Space Amplification: 1.1

Compaction 3: “Universal Compaction”



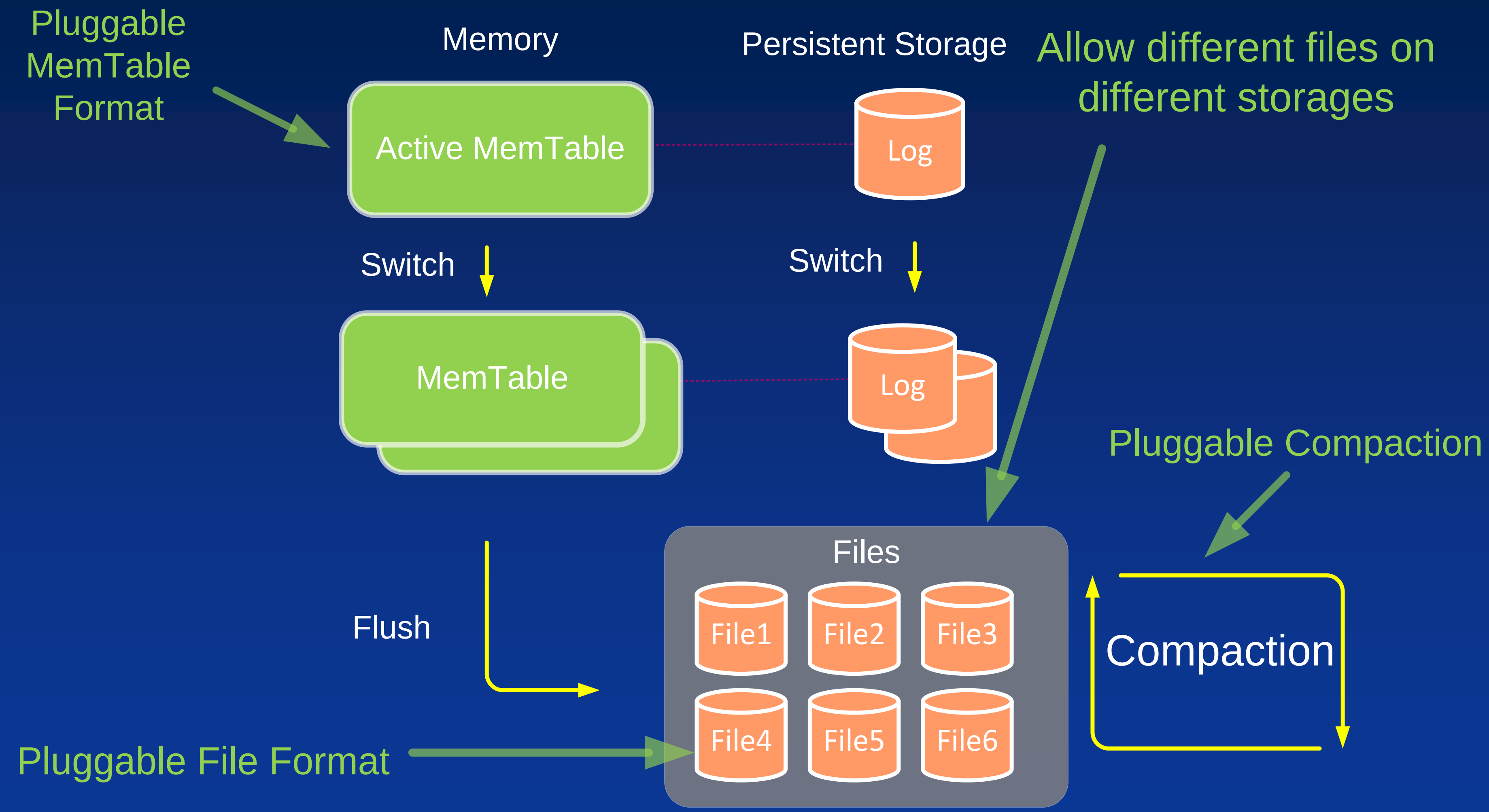
- Write Amplification <= number of files
- Read Amplification: number of files or 1 (using bloom)
- Space Amplification: 2
- Need Double Space for compaction

Comparing Compaction (1TB DB, 1GB flush size)

	Get() Read-Amp	Range Scan Read-Amp	Prefixed scan Read-Amp	Write-Amp	Space-Amp	Double Space Issue?
Compaction 1 (to one file)	1 (using bloom)	2	<= 2	1000	1.001	Yes
Compaction 2 (“Leveled”)	1 (using bloom)	5	<= 5 (using bloom)	40	1.1	No
Compaction 3 (“Universal”)	1 (using bloom)	11	<= 11 (using bloom)	<= 11	2	Yes

- Write-Amp: Write Amplification
- Read-Amp: Read Amplification
- Space-Amp: Space Amplification

Why is RocksDB Friendly to Flash Devices? Reason 2. Pluggable



Why is RocksDB Friendly to Flash Devices?

Reason 3: Optimized for fast storage

- Lock-free reads
- Optimize to reduce CPU usage

Benchmarking RocksDB

- Use db_bench
- Our benchmark setting and results:
- https://github.com/facebook/rocksdb/wiki/Performance-Benchmarks
- Find all information on http://rocksdb.org/
- Benchmark RocksDB on your devices!

Take-Away

- RocksDB and its architecture
- Example use case in facebook.
- RocksDB is flash-friendly
- Benchmark RocksDB on your devices!

Visit http://rocksdb.org/
for more information!

Thank you!