

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

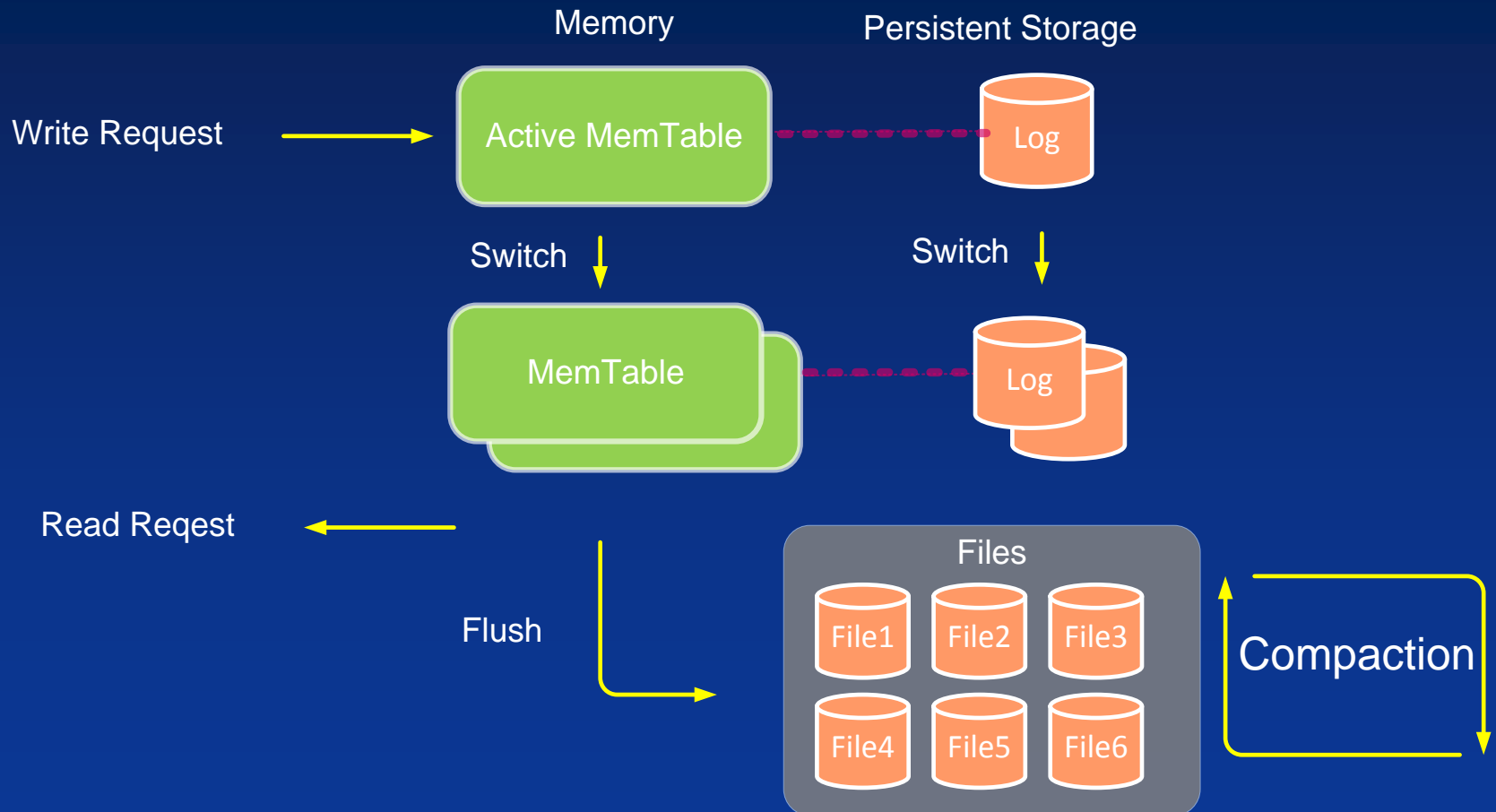
What is RocksDB

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

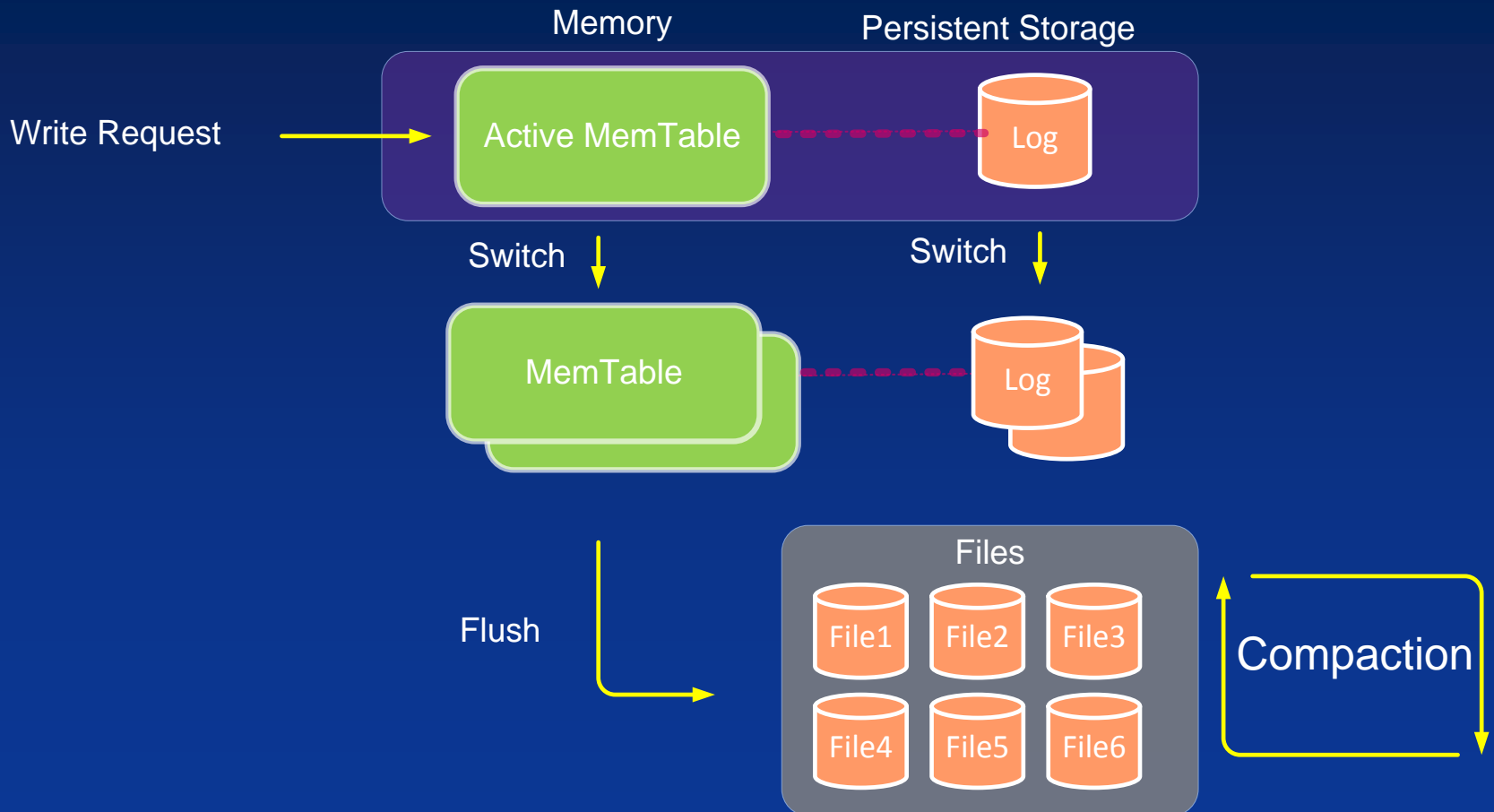
RocksDB API

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

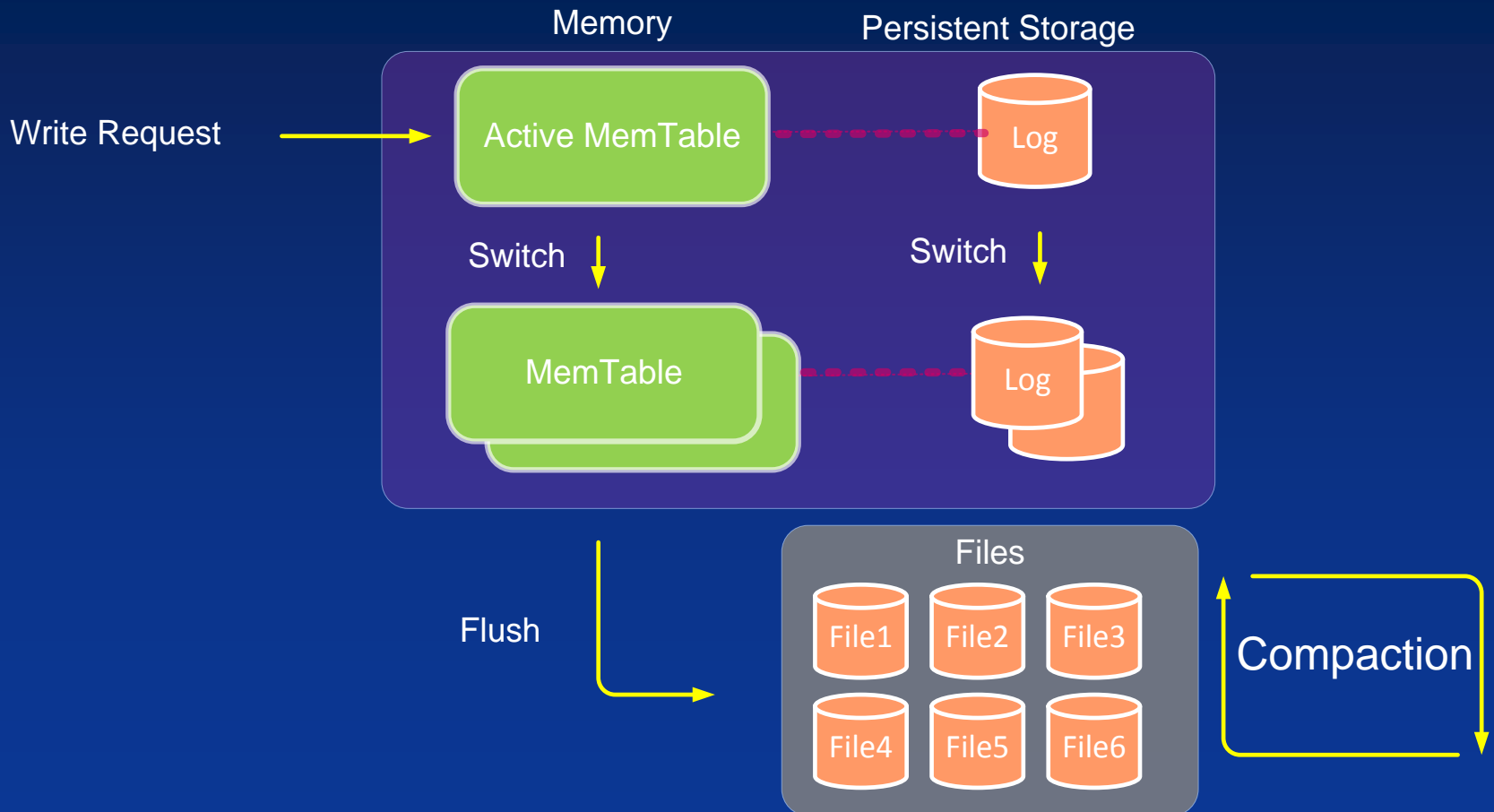
RocksDB Architecture



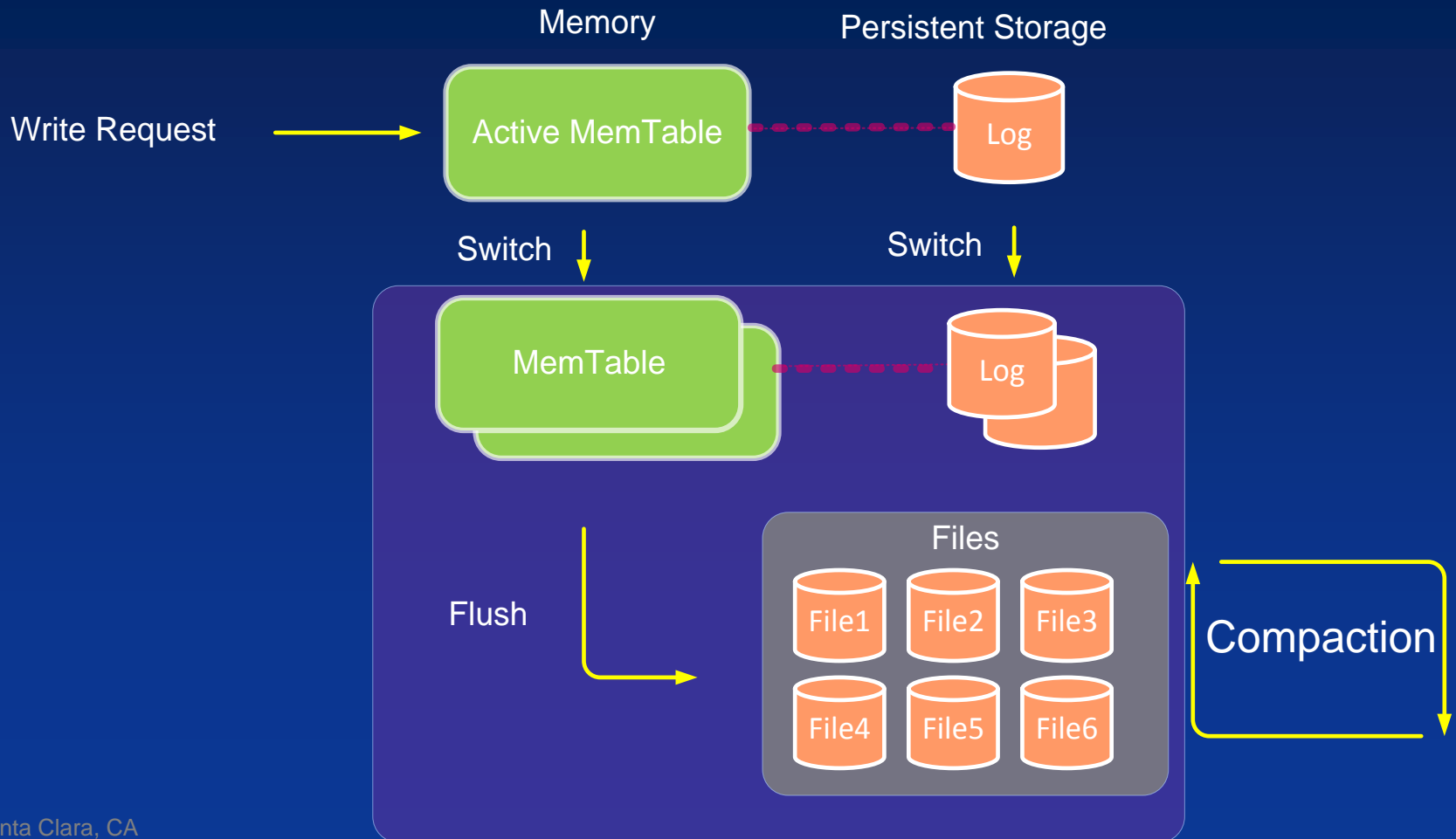
Write Path (1)



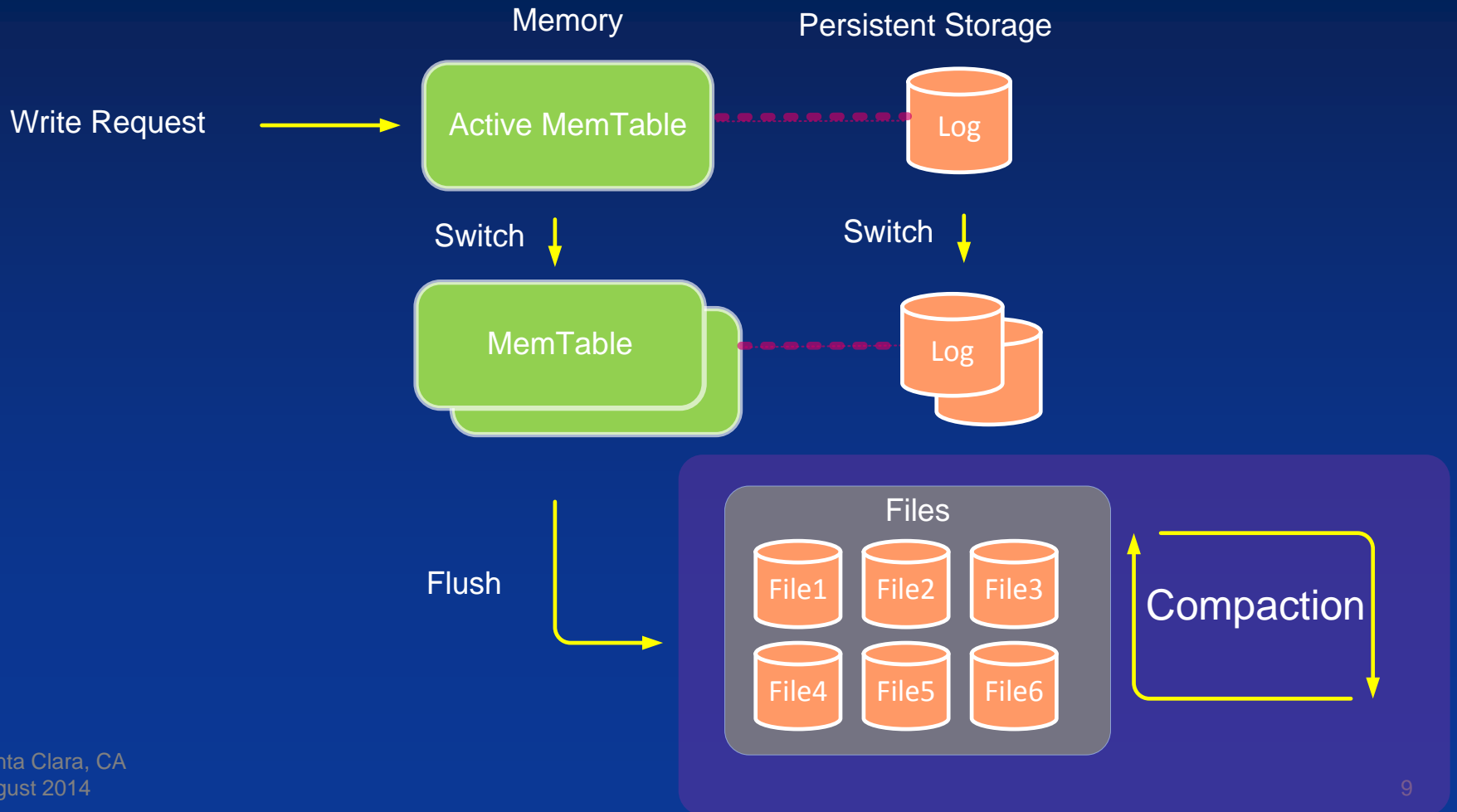
Write Path (2)



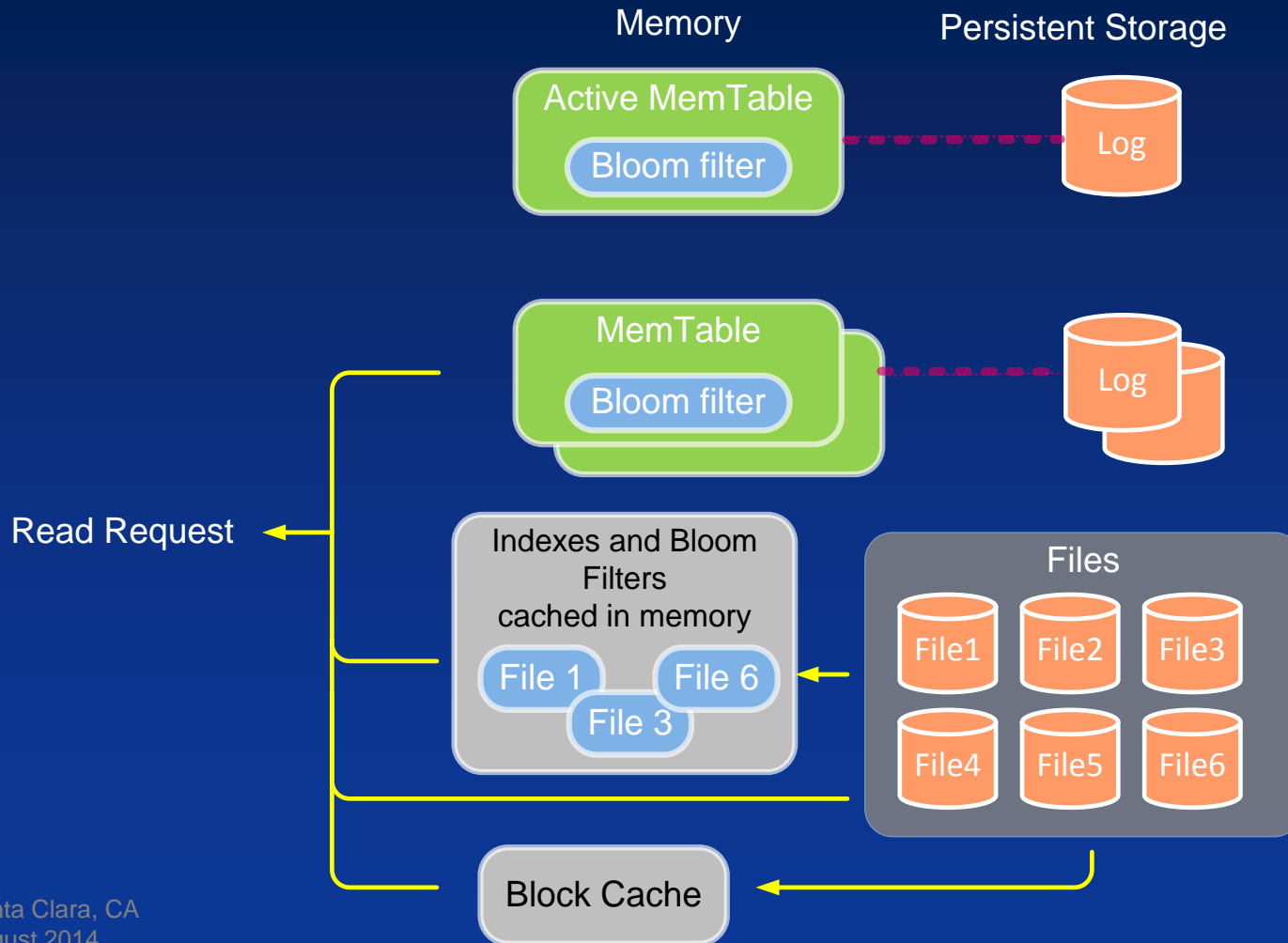
Write Path (3)



Write Path (4)



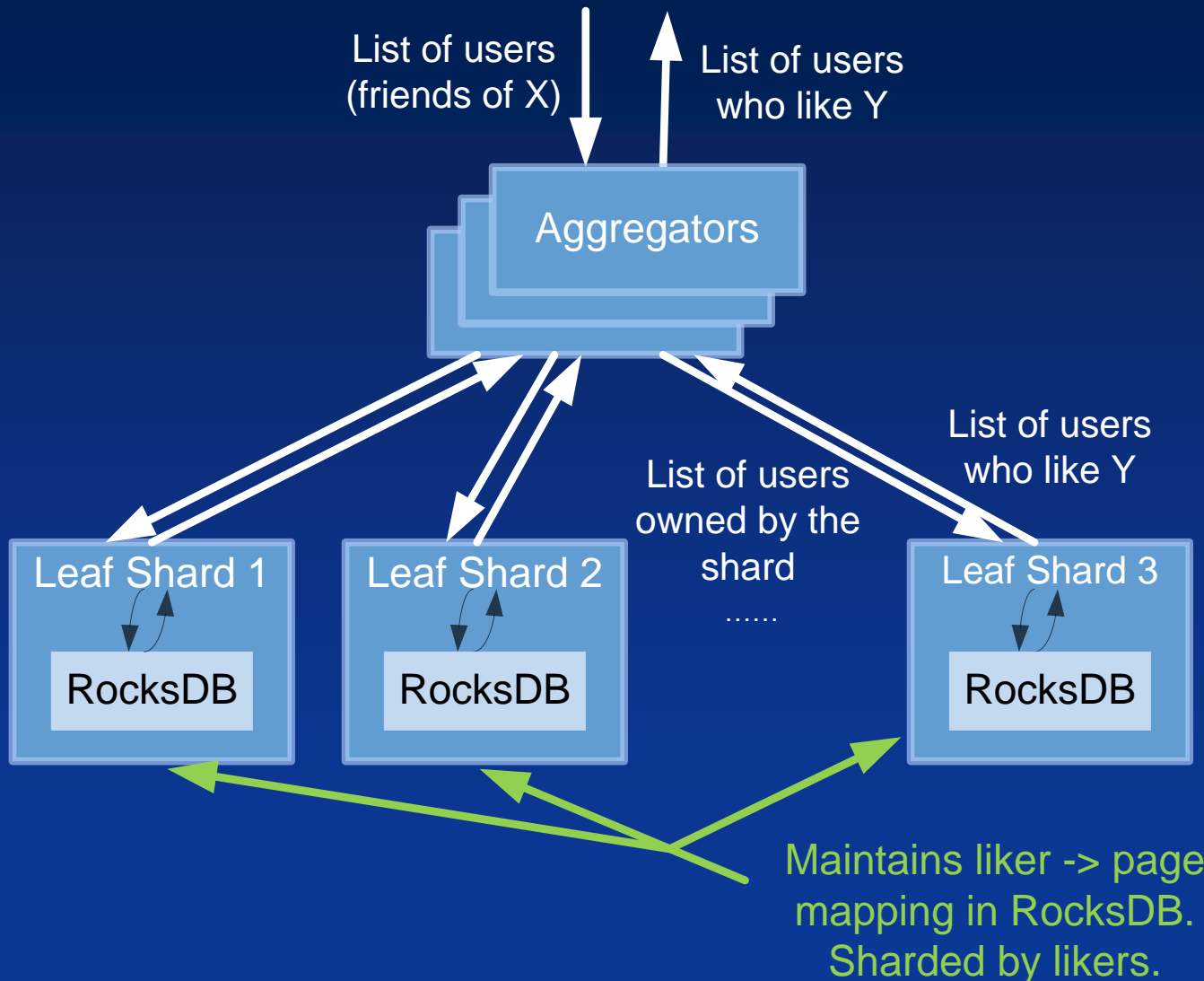
Read Path



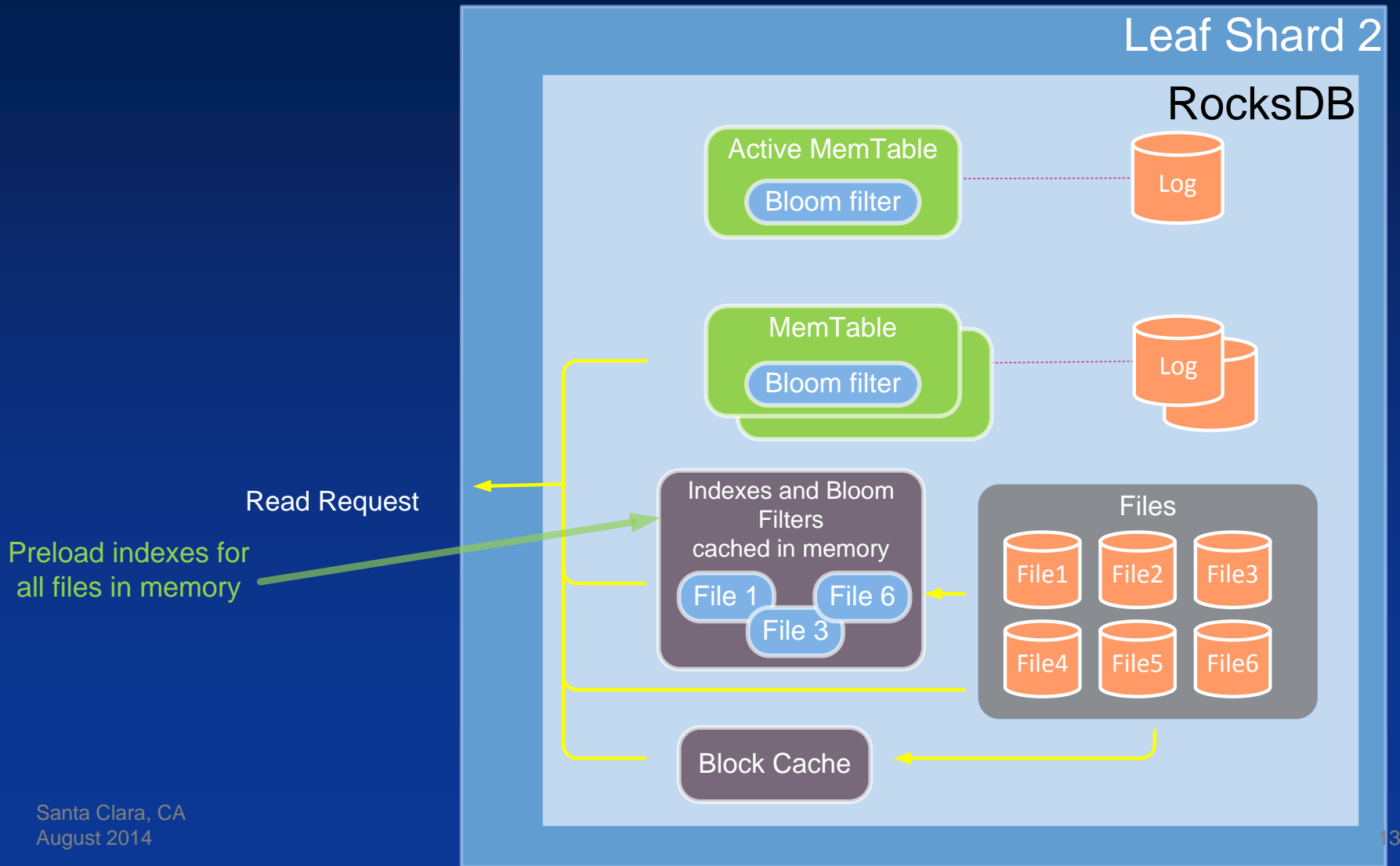
Example Use Case: Find all friends of user X who like Page Y

- 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

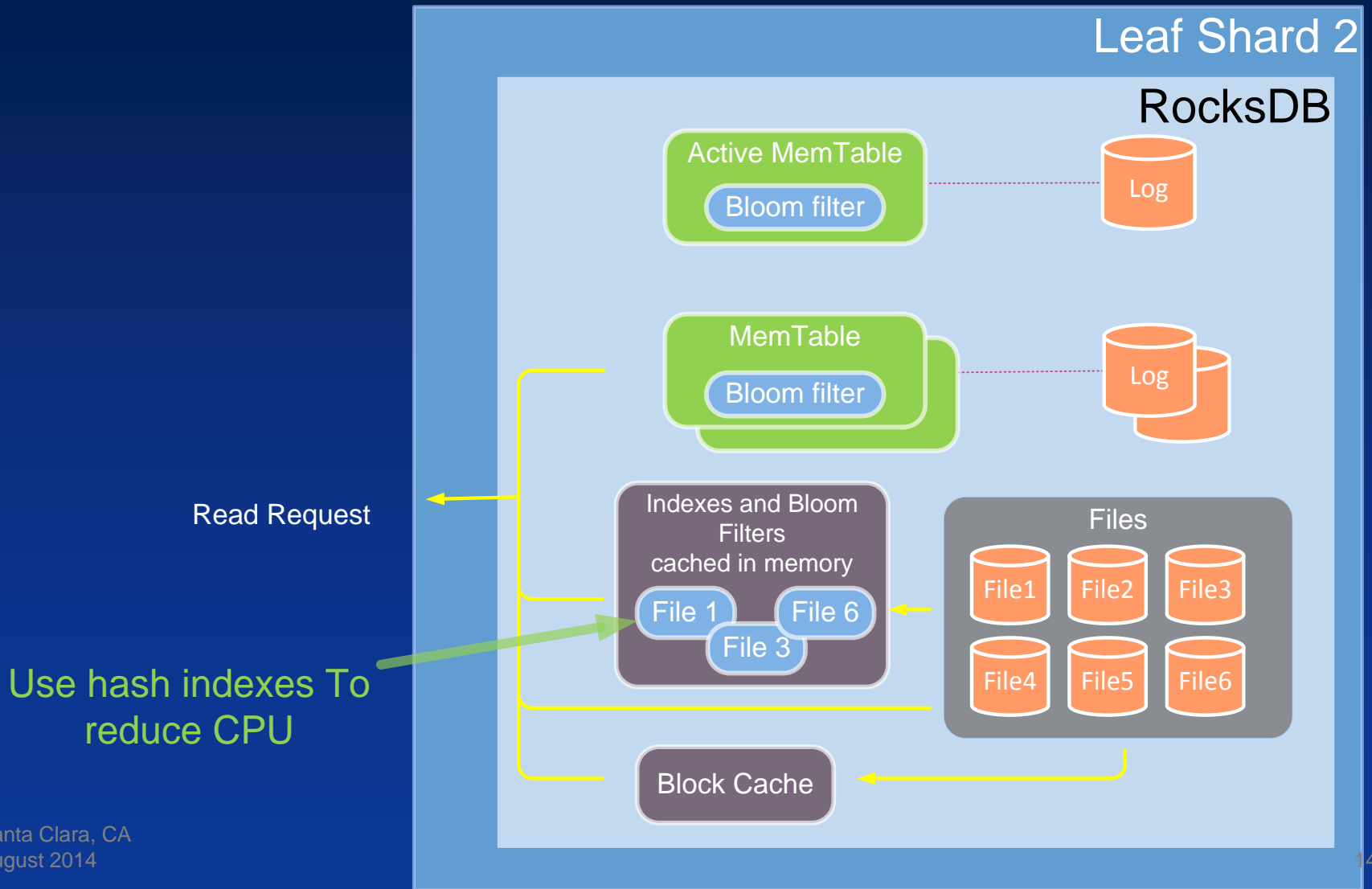
Example Use Case: Find all friends of user X who like page Y



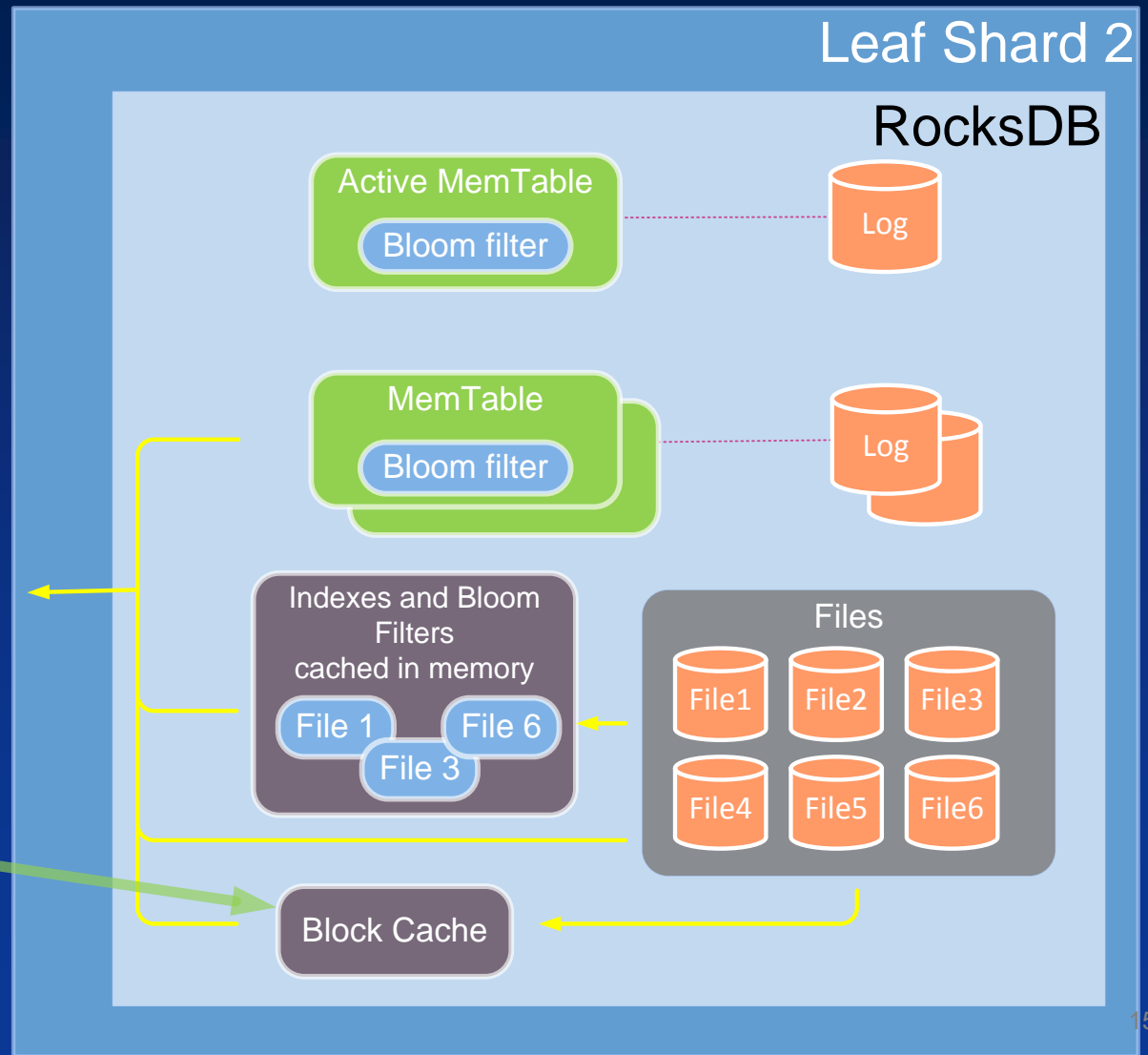
Example Use Case: Find all friends of user X who like Page Y



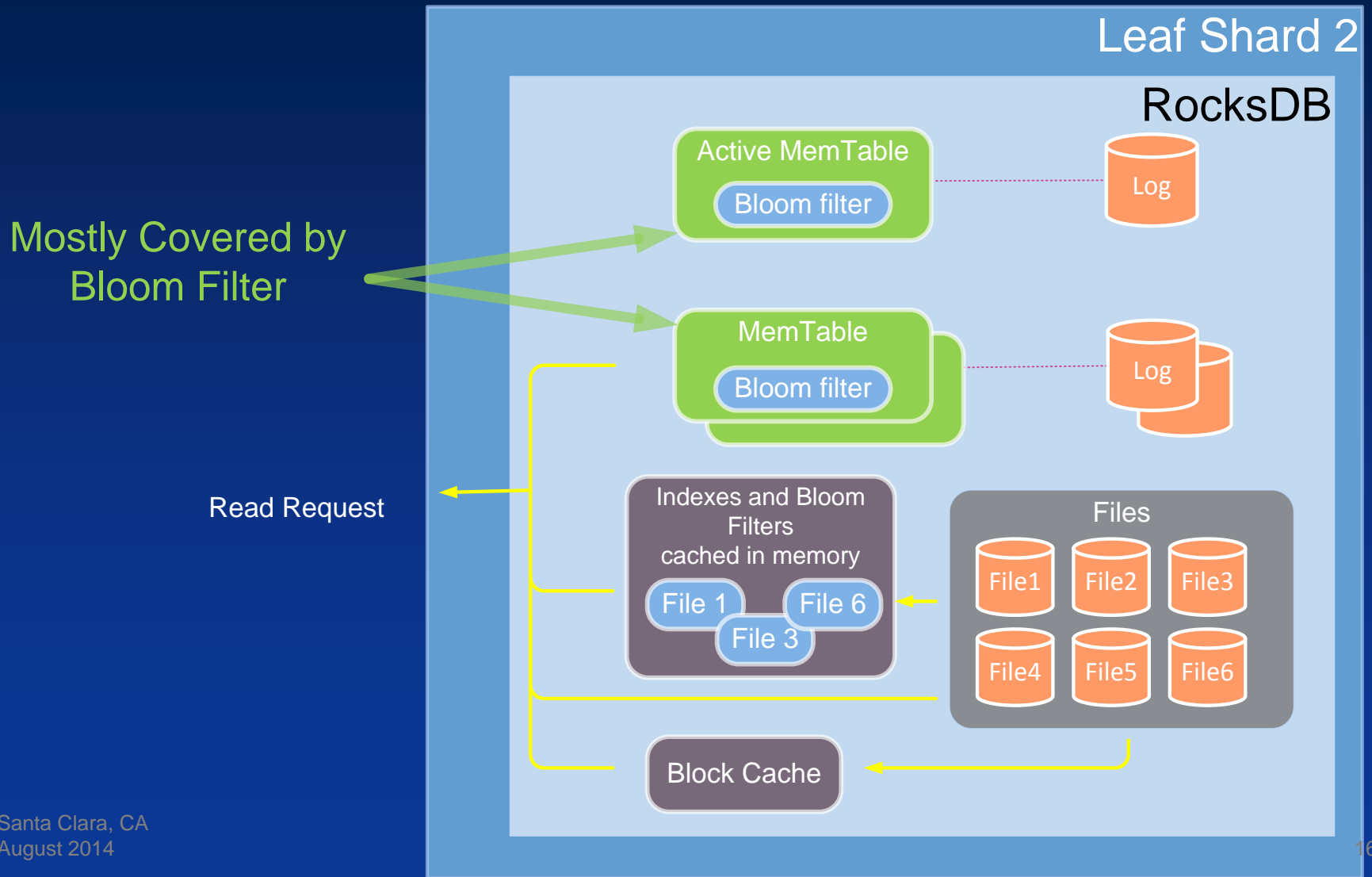
Example Use Case: Find all friends of user X who like Page Y



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



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



Why is RocksDB Friendly to Flash Devices?

Reason 1:

Tunable between device wear-out and read latency

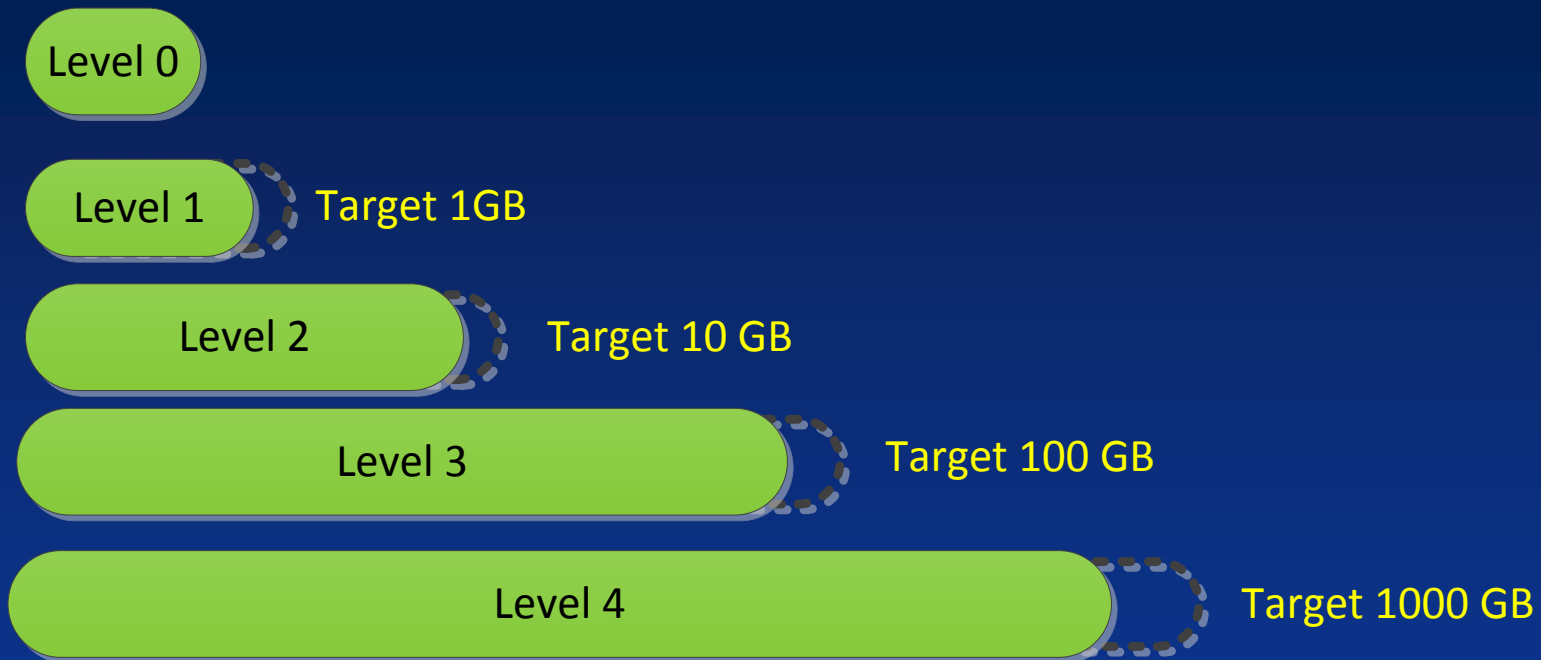
- Tunable compaction to trade-off
 - Read Amplification
 - Write Amplification
 - Space Amplification

Compaction 1: compact to one file



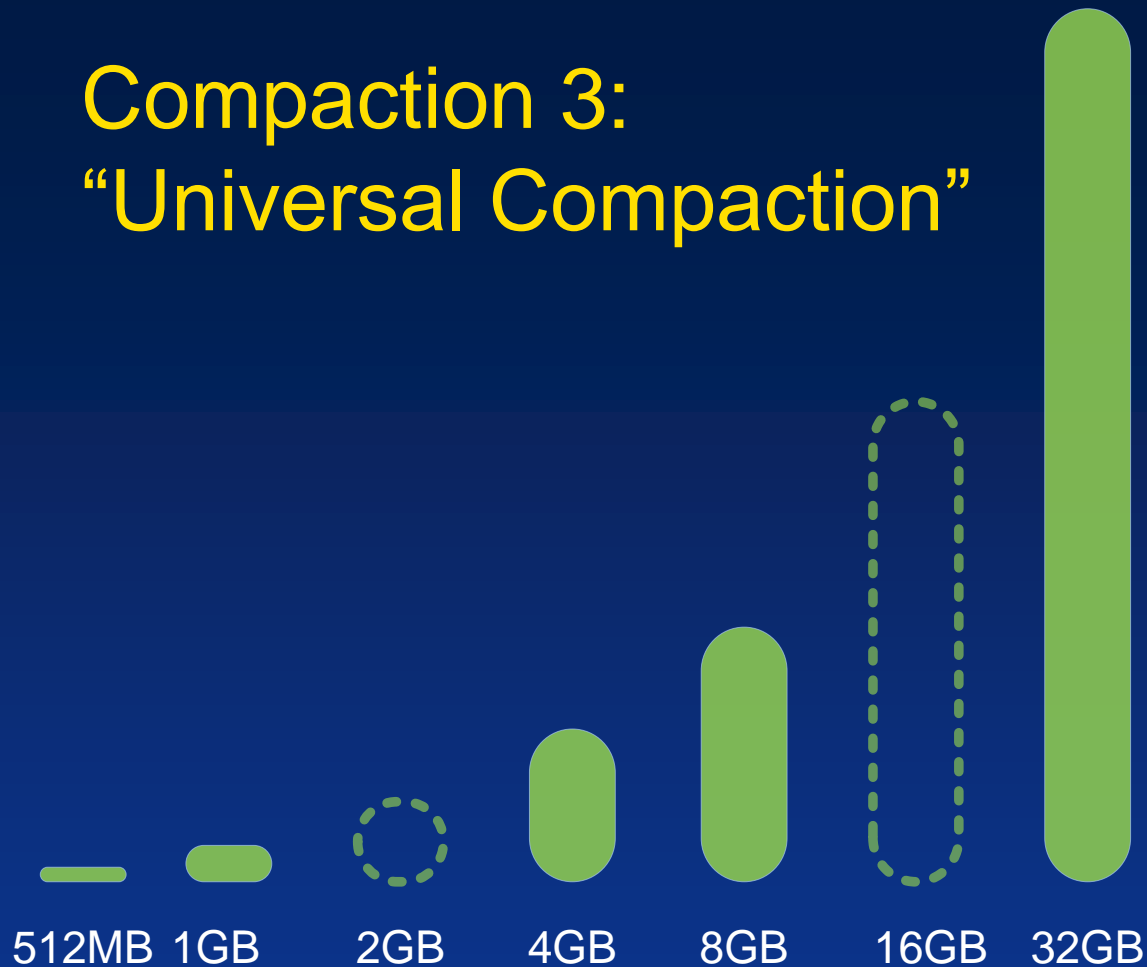
- Write Amplification = 1000
- Read Amplification = 2 or 1 using bloom
- Space Amplification = 1.001
- Need Double Space for compaction

Compaction 2: Leveled-Compaction



- Read Amplification: number of levels or 1 (using bloom)
- Write Amplification: $10 * \text{number of levels}$
- Space Amplification: 1.1

Compaction 3: “Universal Compaction”



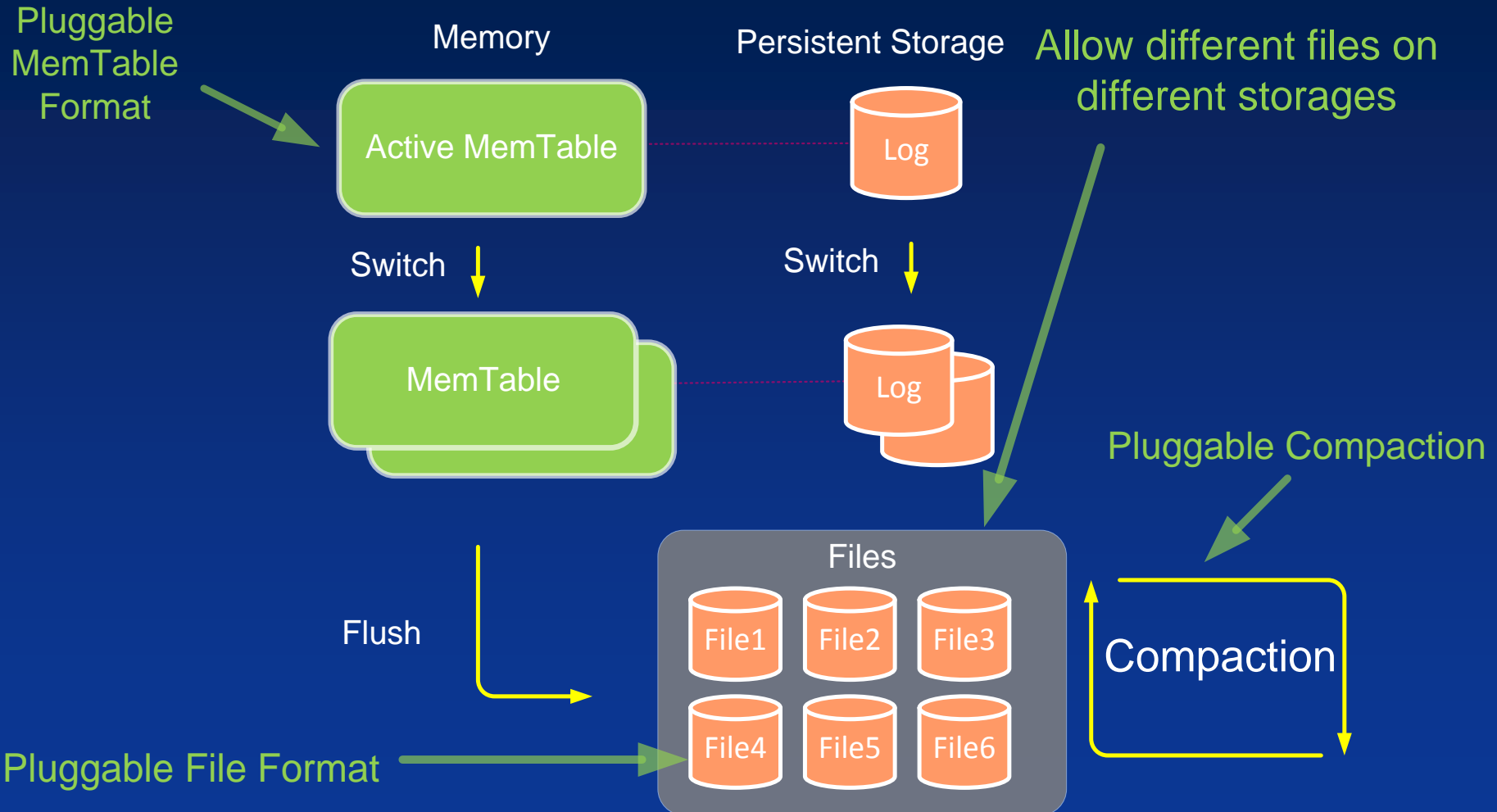
- Write Amplification \leq number of files
- Read Amplification: number of files or 1 (using bloom)
- Space Amplification: 2
- Need Double Space for compaction

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!