

Evolution of Development Priorities in Key-value Stores Serving Large-scale Applications: The RocksDB Experience

FAST 2021

Siying Dong (Speaker), Andrew Kryczka, Yanqin Jin
Software Engineer, Facebook Inc

Michael Stumm
University of Toronto

FACEBOOK Infrastructure

Agenda

Design Goals of RocksDB

Evolution of Resource Optimization Targets

Lessons on Integrity Checking

Design Goals of RocksDB

RocksDB

Persistent Key-Value Store Library



RocksDB

RocksDB Design Goals

For Modern Hardware

SSDs, Multicore CPUs and serve high
throughput

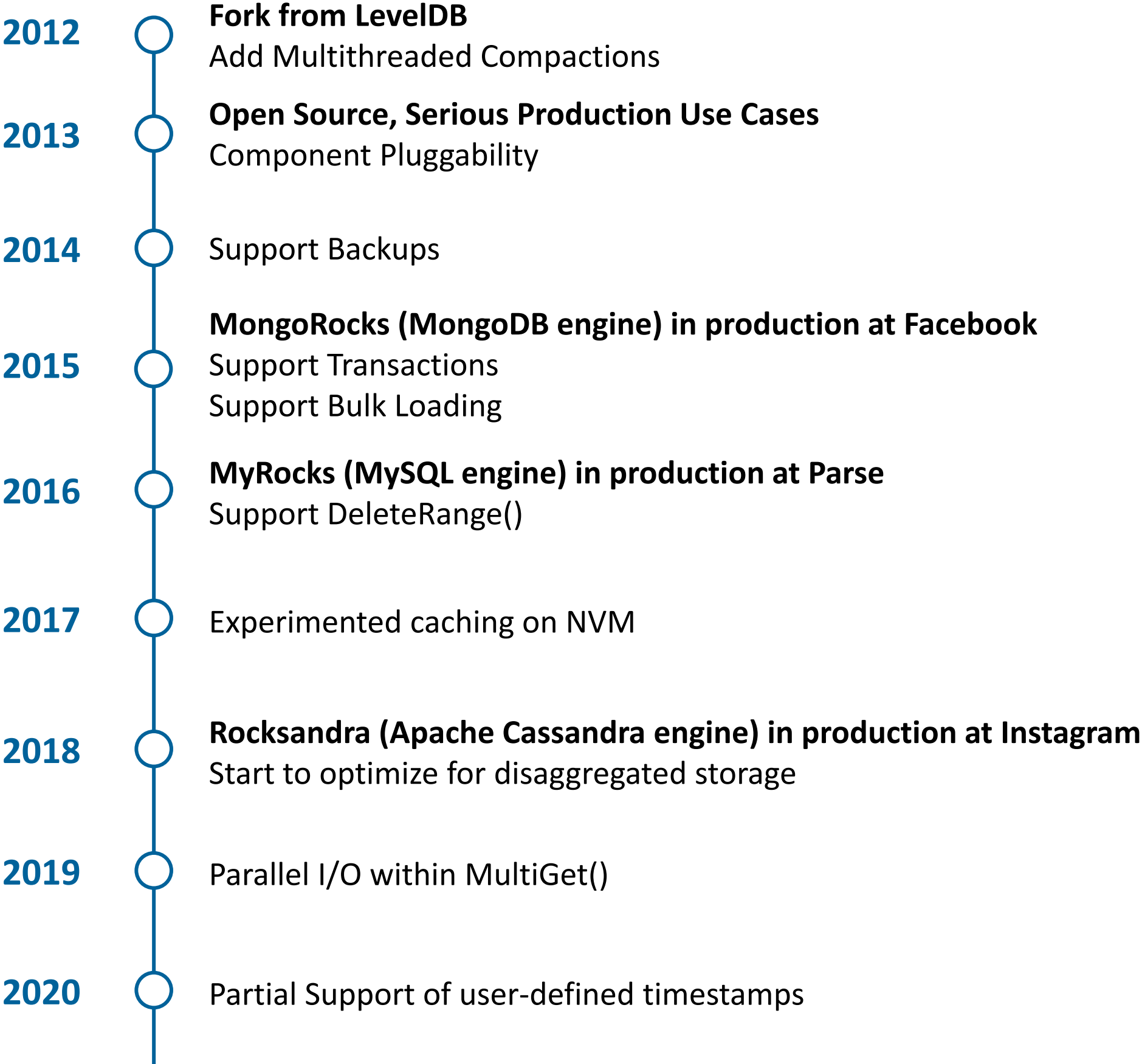
Versatile

Serve wide range of applications

Single Node

Doesn't handle replication, load
balancing, etc

RocksDB Timeline



Evolution of Resource Optimization Targets



First Optimization Goal: Write Amplification

Focusing on Write Amplification in Engine Layer

Often
Hundreds

Write amplification by
storage engine

Close to 1

Write amplification by file
systems

Between 1
and 3

SSD's internal write
amplification

Focusing on Write Amplification in Engine Layer

Often
Hundreds

Write amplification by
storage engine

Close to 1

Write amplification by file
systems

Between 1
and 3

SSD's internal write
amplification

A photograph of a large pile of cut logs, likely from birch or a similar light-colored tree. The logs are stacked horizontally, with their circular cross-sections visible. The wood is a warm, reddish-brown color, and the bark is a lighter, silvery-grey. The background is slightly blurred, showing more logs and some greenery in the distance. Overlaid on the center of the image is the text "Data Structure: Log-Structured Merge-Tree" in a white, sans-serif font.

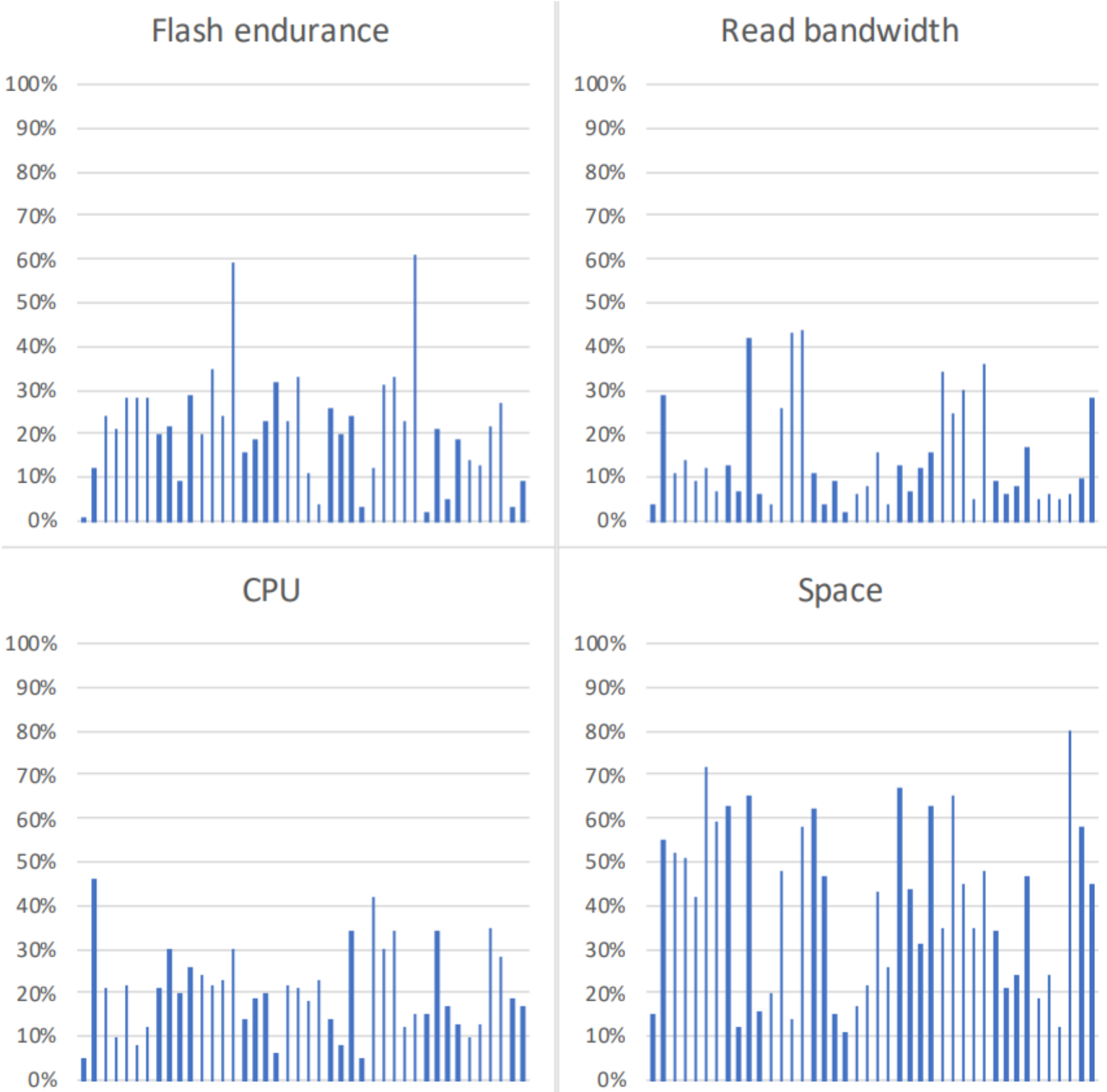
Data Structure: Log-Structured Merge-Tree

A background image of a student in a classroom, wearing a grey hoodie over a white shirt, sitting at a wooden desk and writing in a spiral notebook with a pen. Other students are blurred in the background.

Lesson

Space efficiency is the bottleneck for most applications using SSDs.

Resource Utilization for ZippyDB and MyRocks Use Cases



Good News

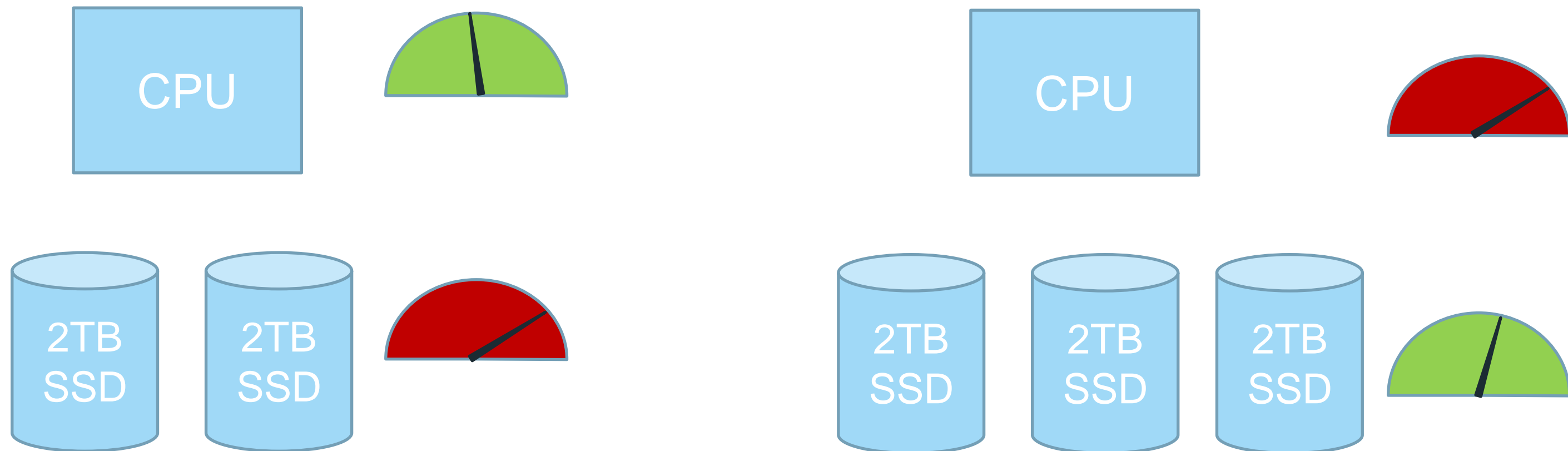
LSM-Tree is good for
space efficiency too
(with some optimizations)

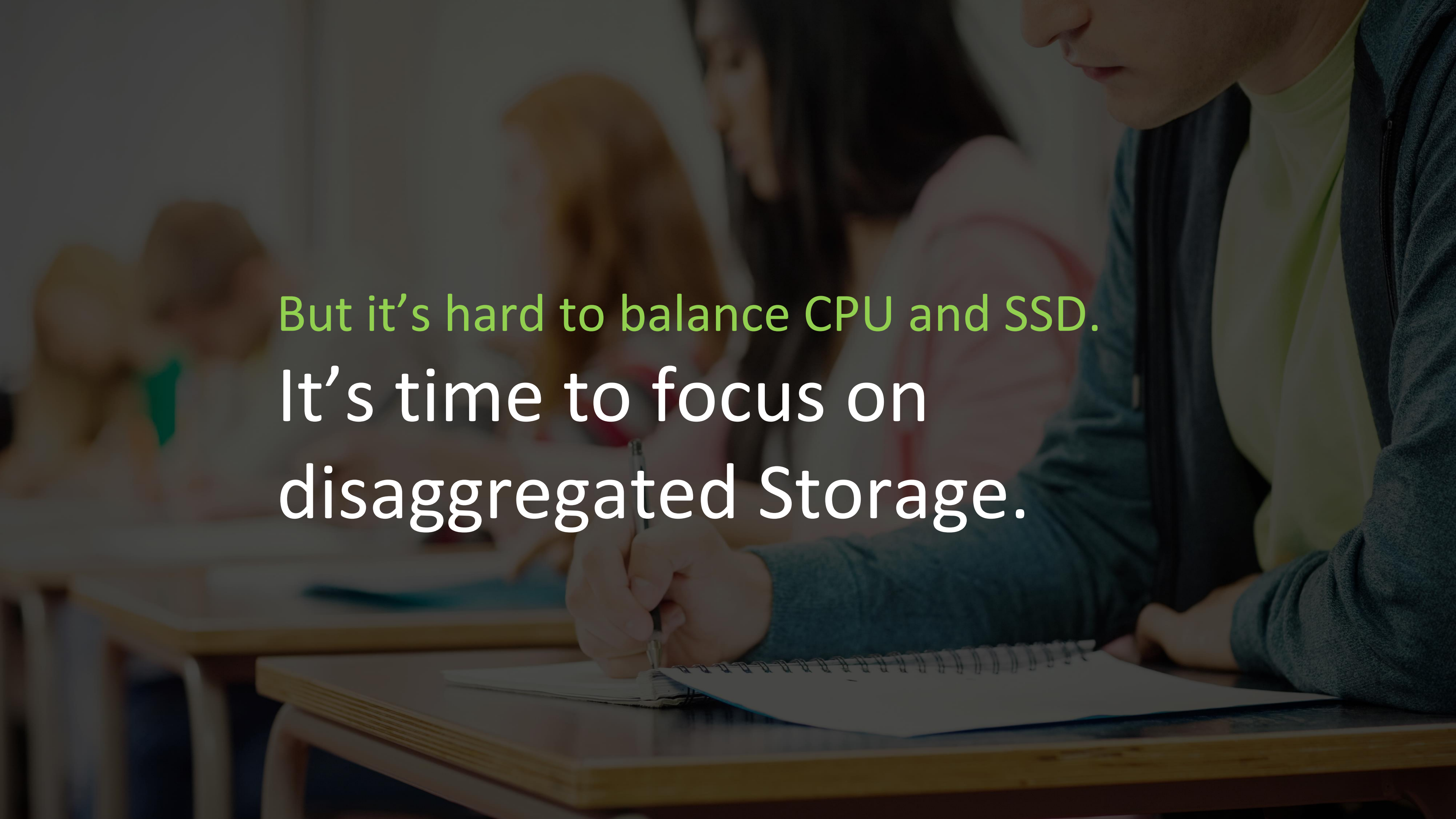
A background image of a classroom with students at desks. In the foreground, a student is writing in a spiral notebook. The image is dimmed to serve as a background for text.

Lesson

Reducing CPU Overhead is
becoming more important.

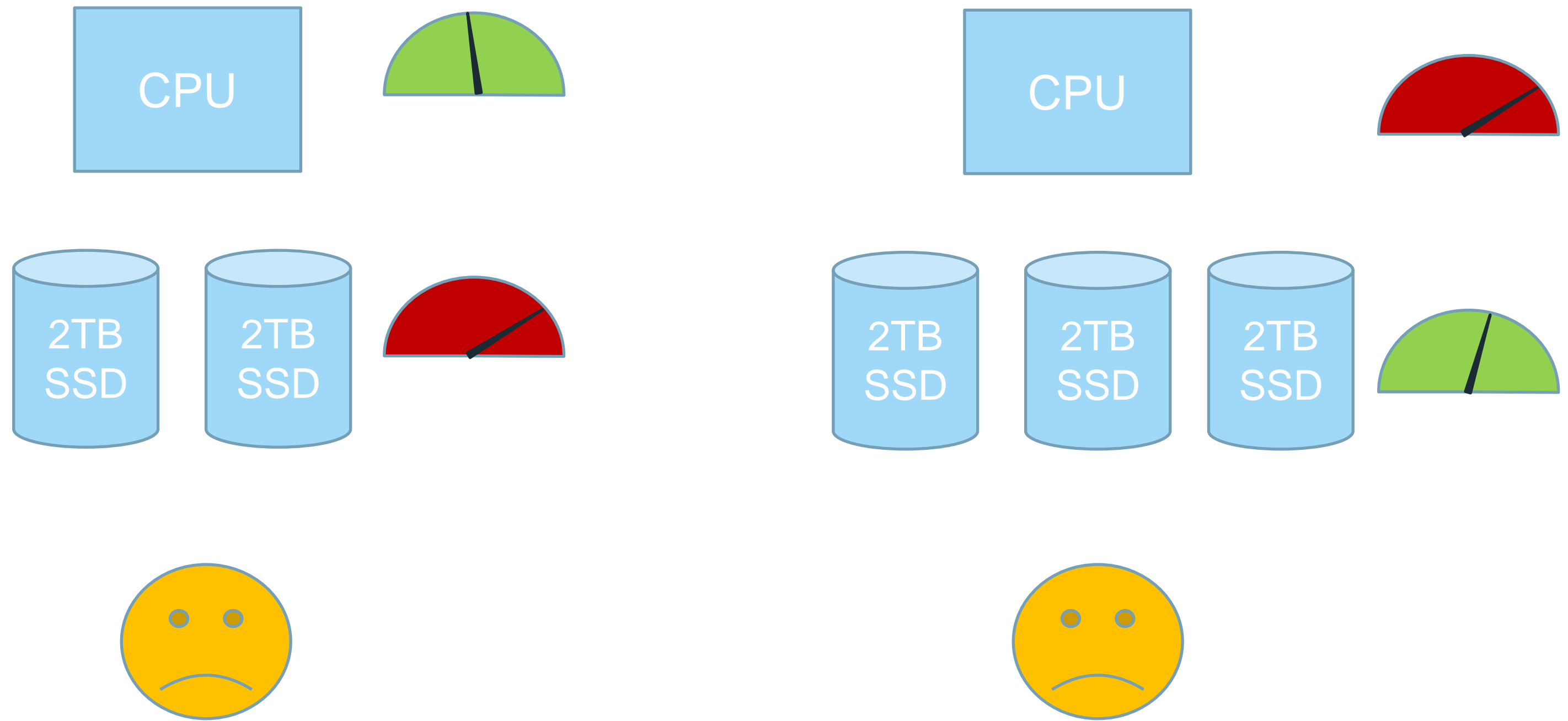
CPU/Space Balance can often be shifted through hardware configuration



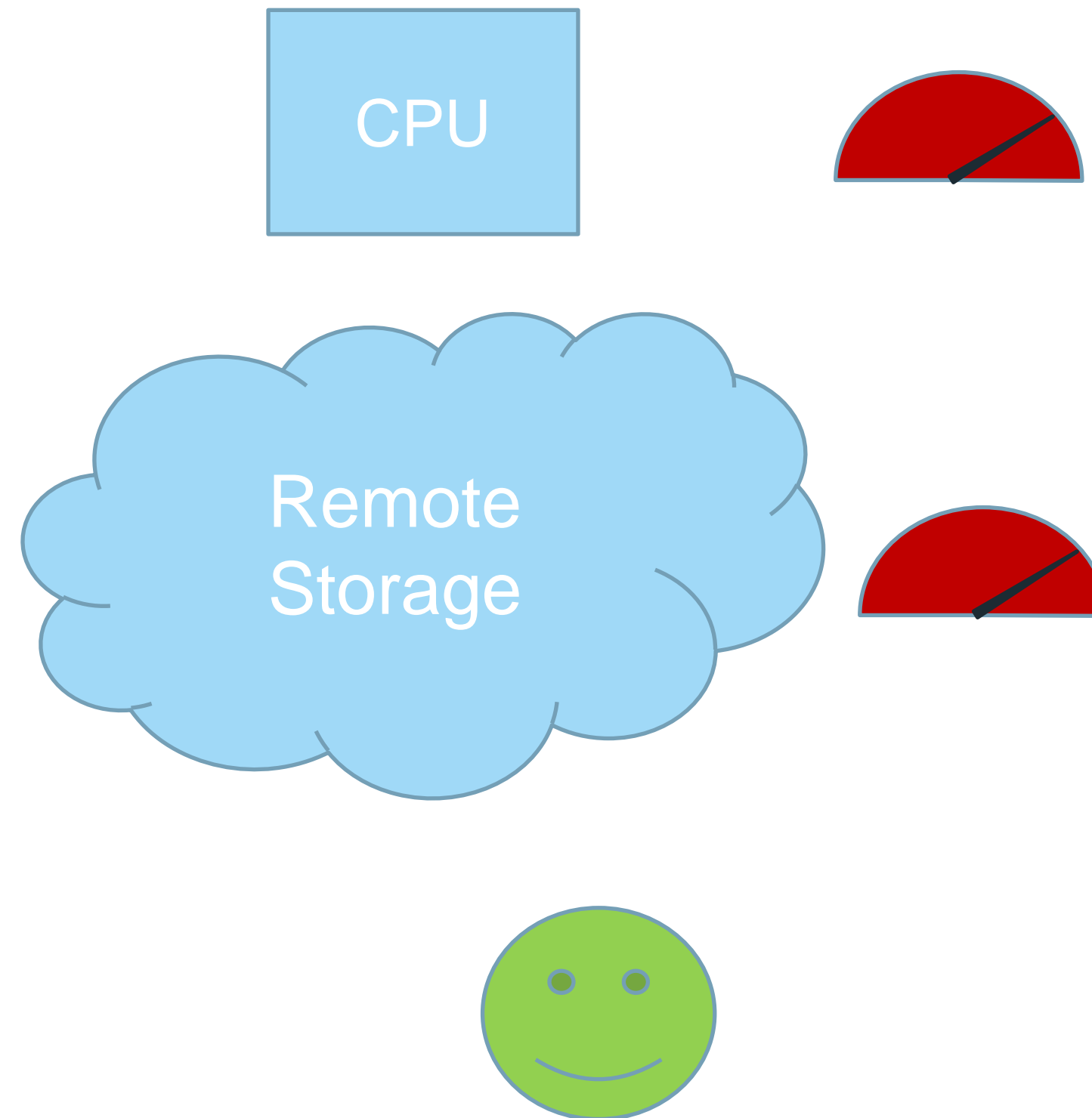
A background image of a student in a classroom, wearing a grey hoodie over a white shirt, sitting at a wooden desk and writing in a spiral notebook with a pen. Other students are blurred in the background.

But it's hard to balance CPU and SSD.
It's time to focus on
disaggregated Storage.

It's hard to balance CPU and SSD



RocksDB on Disaggregated Storage



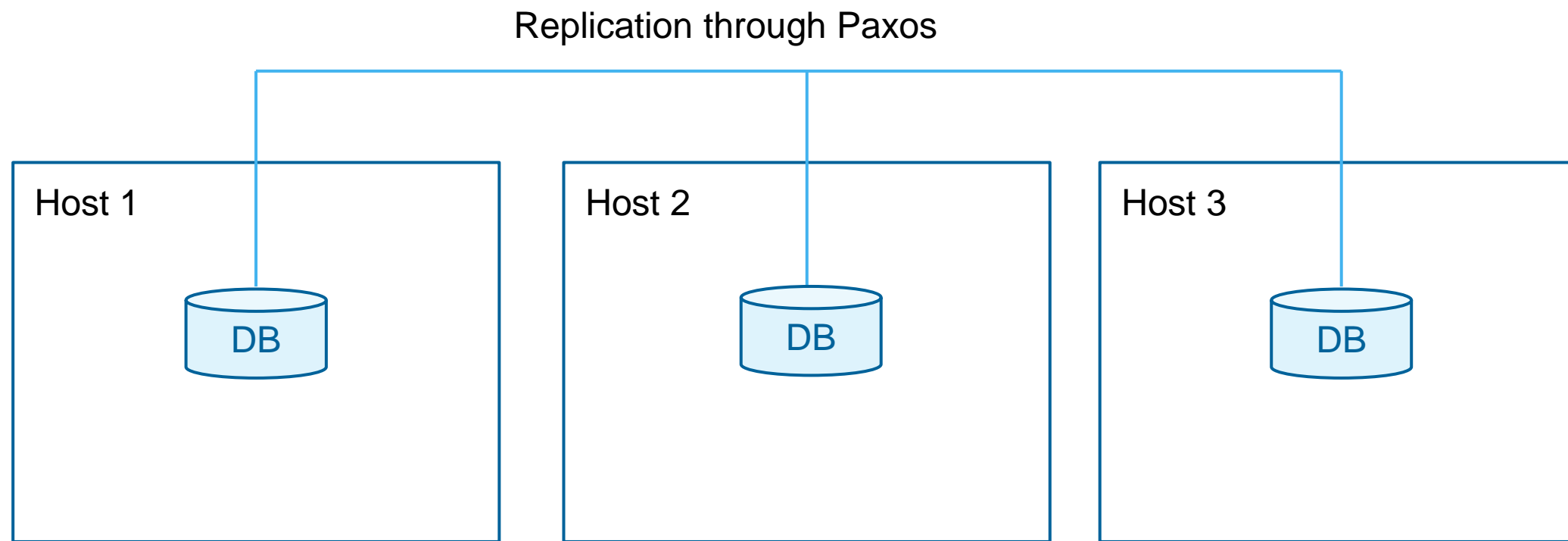
Recap Performance Optimization Targets

- Initial Optimization Targets are Write Amplification, and it is indeed important.
- Majority of use cases are bounded by SSD space.
- Reducing CPU overheads is becoming more important for efficiency
- Now working on disaggregated storage to achieve balanced CPU and SSD usage.

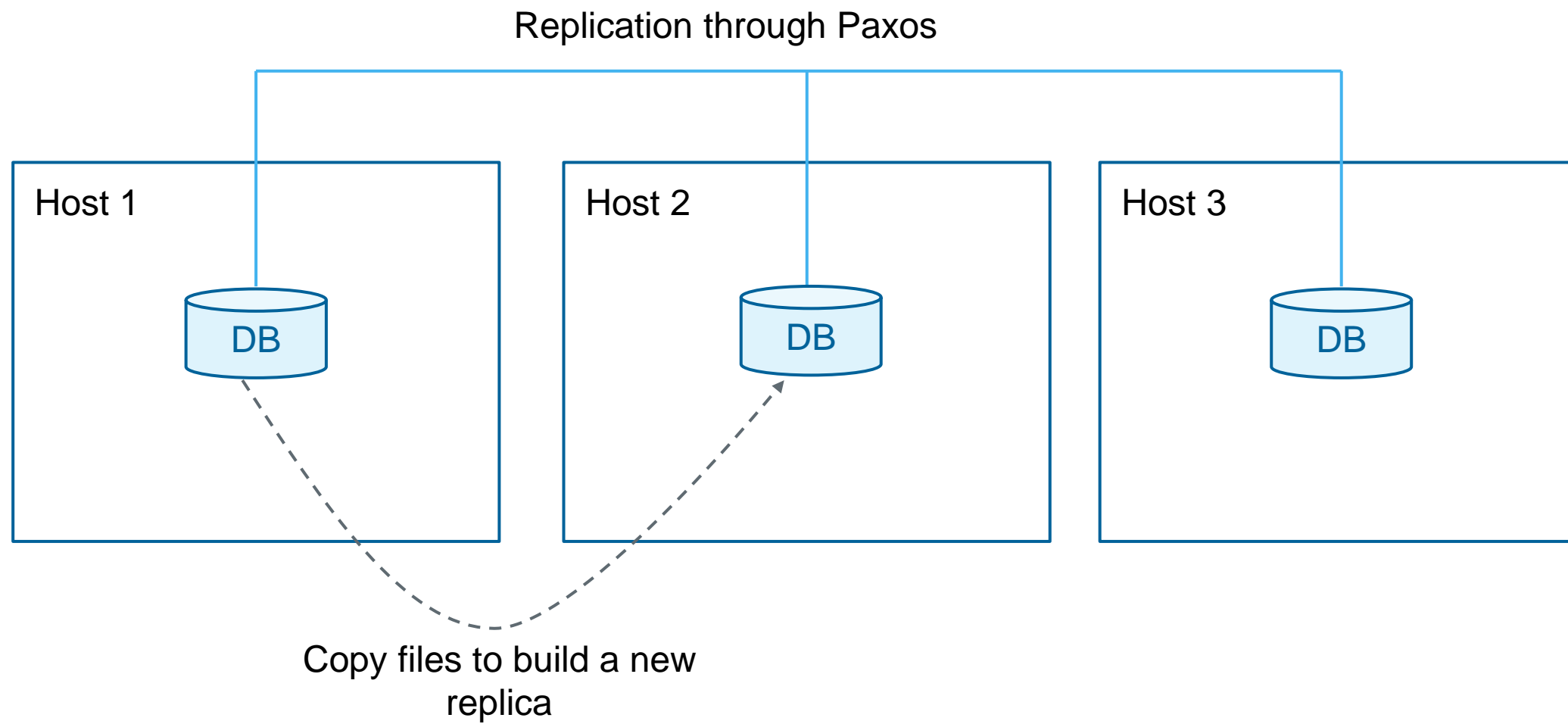


Lessons on Integrity Checking

Typical Way of Using RocksDB

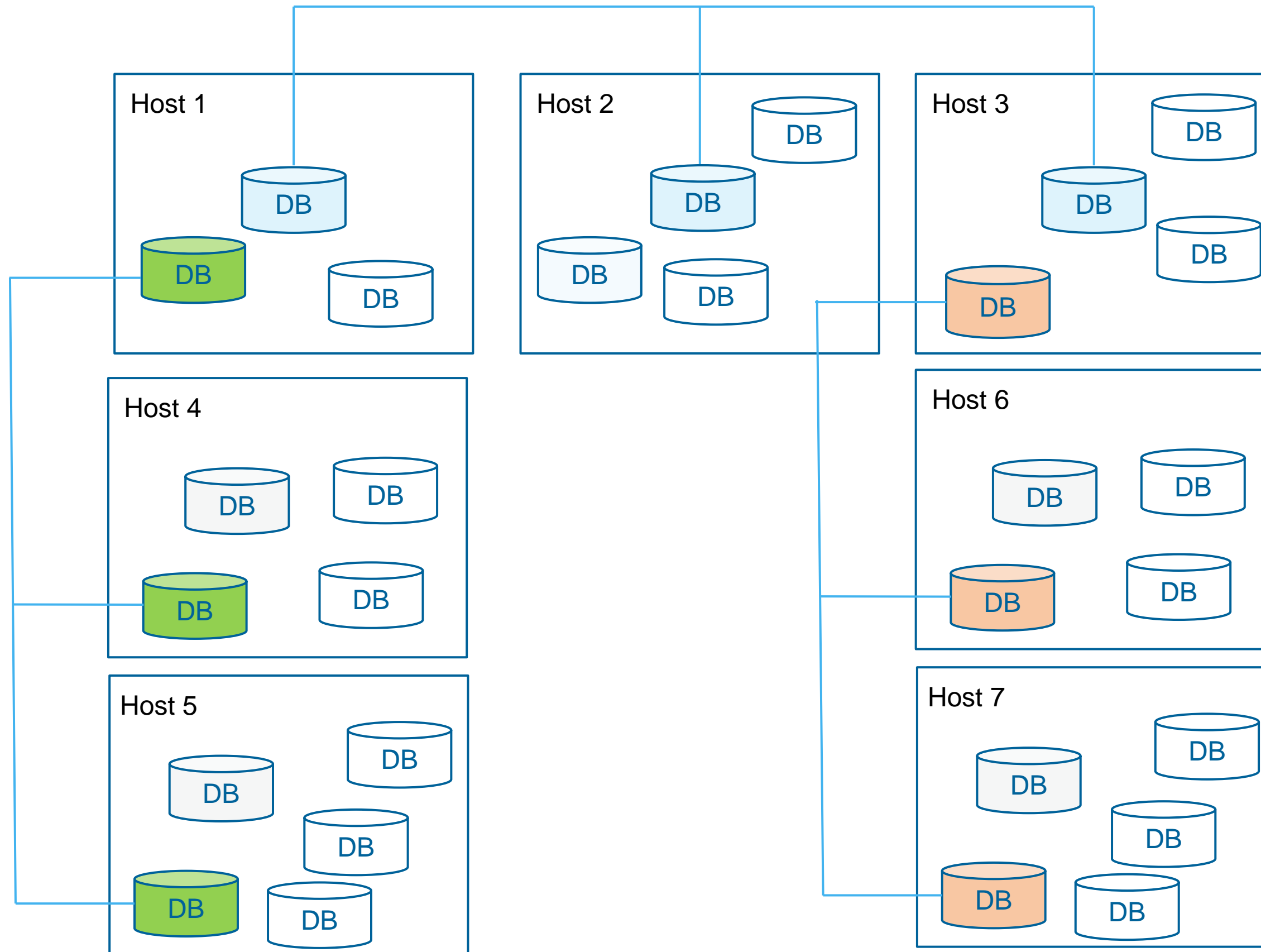


Typical Way of Using RocksDB

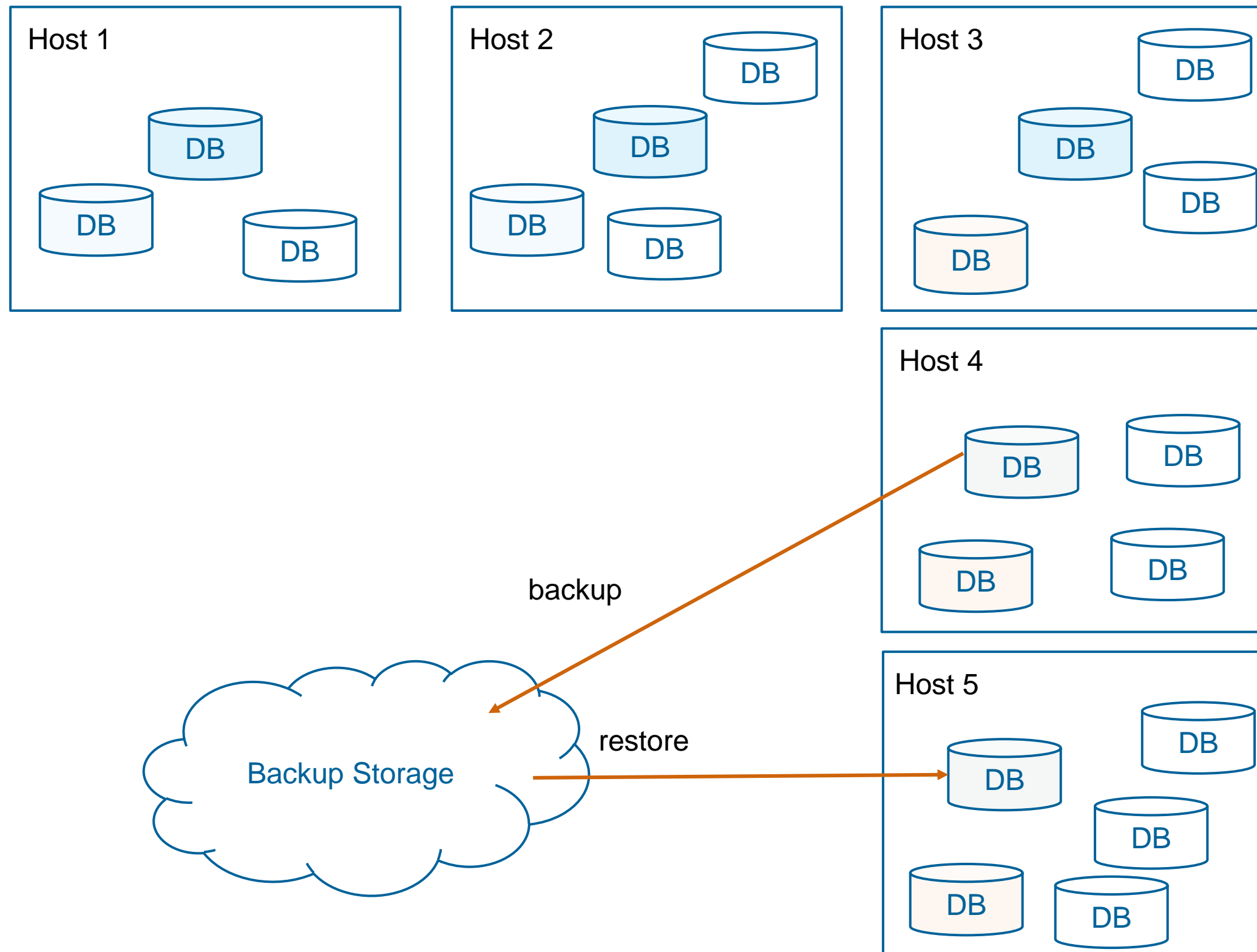


Typical Way of Using RocksDB

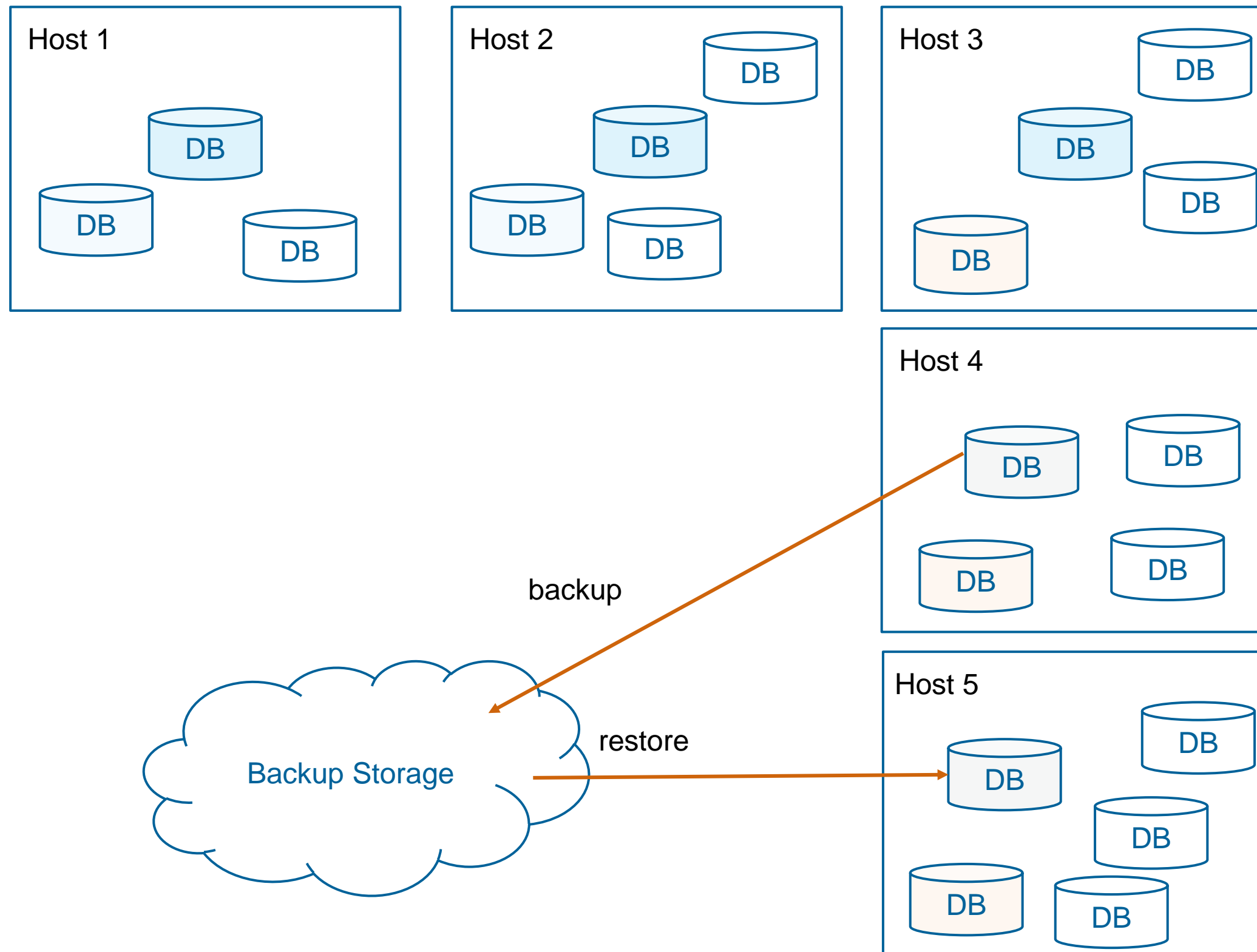
Replication through Paxos



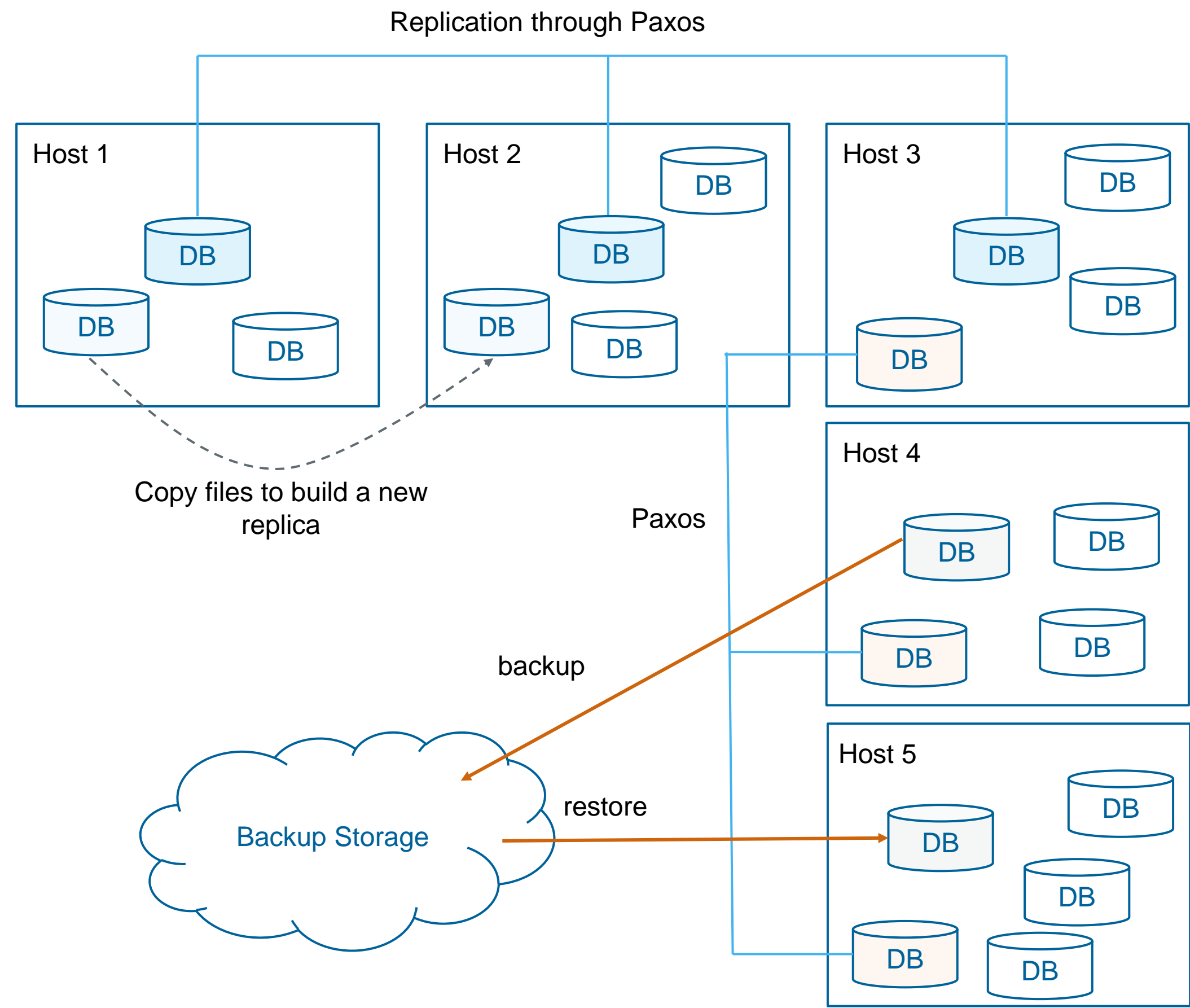
Typical Way of Using RocksDB



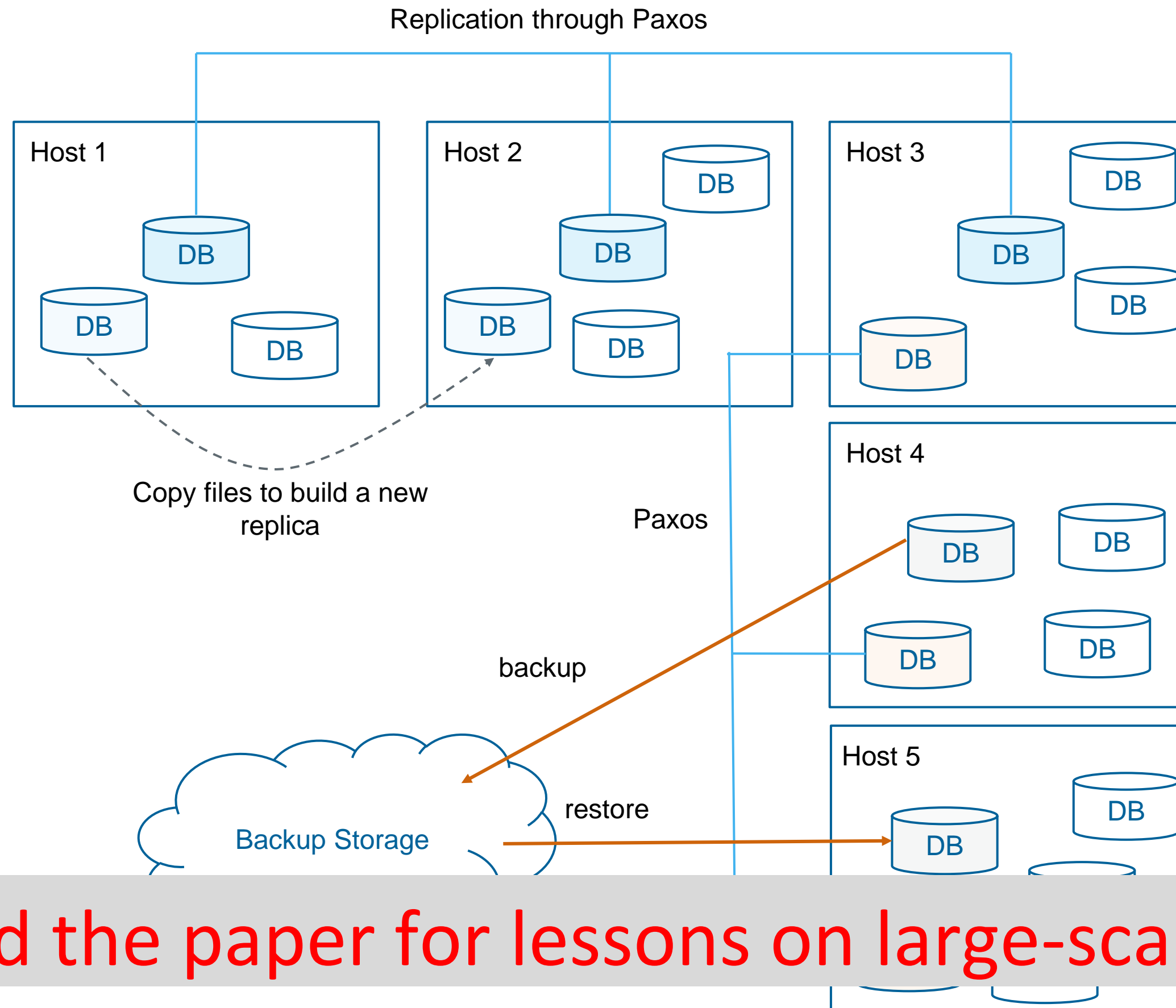
Typical Way of Using RocksDB



Typical Way of Using RocksDB

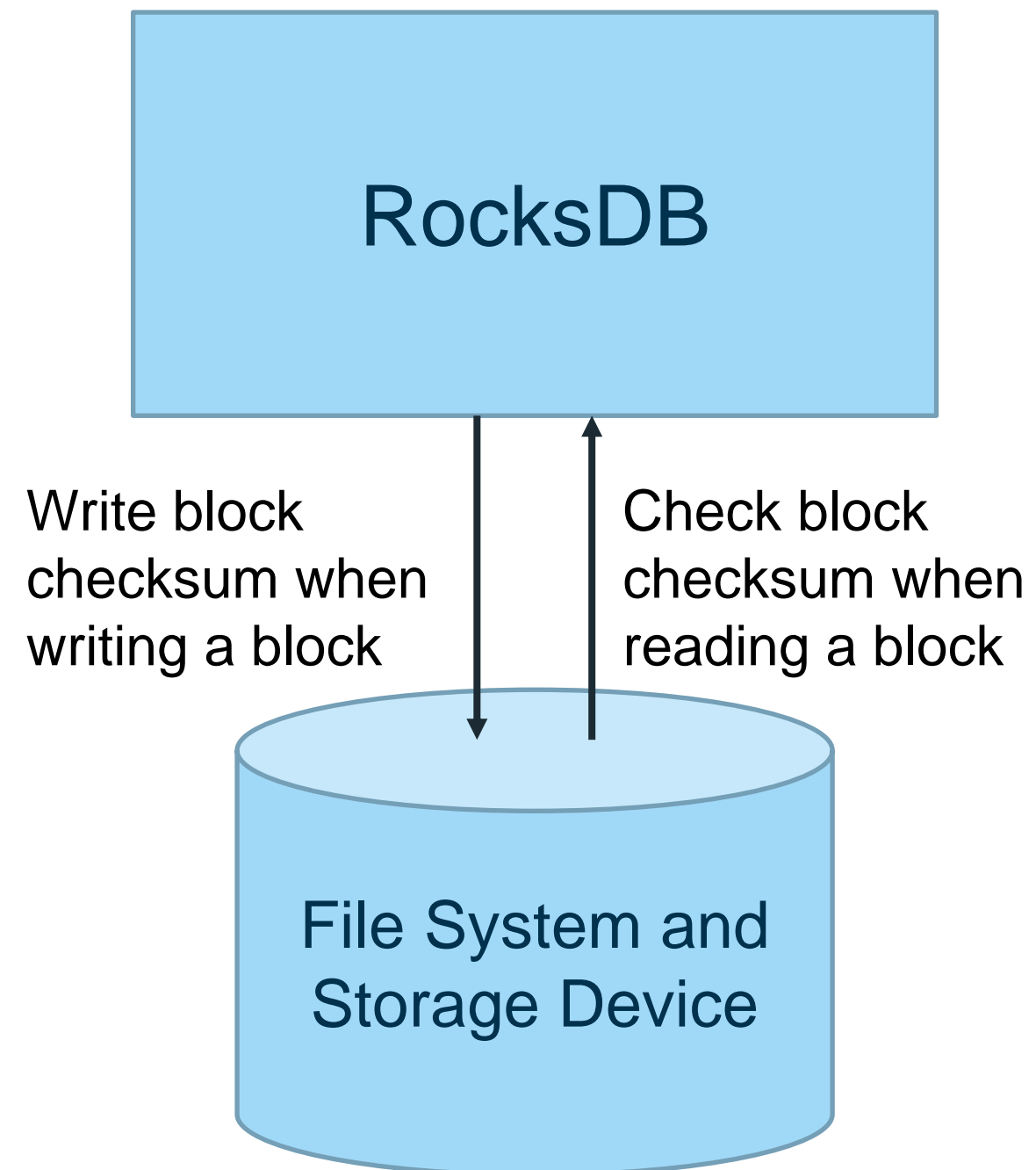


Typical Way of Using RocksDB



Read the paper for lessons on large-scale systems

Minimum Integrity Check: Block Checksum

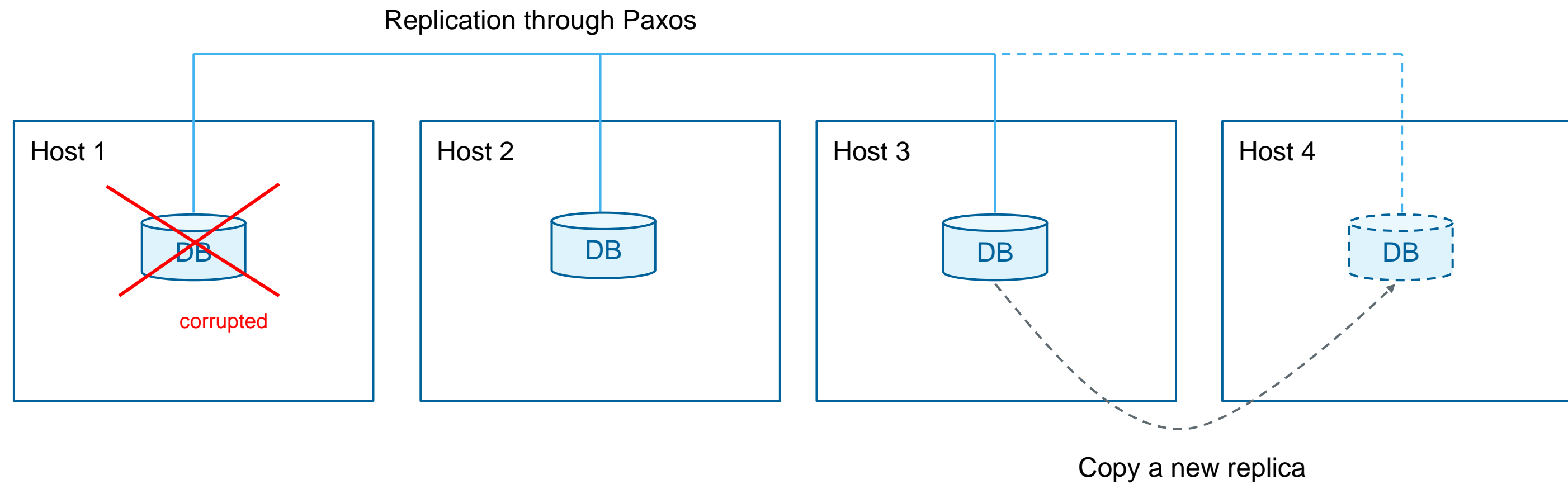


A background image of a student in a classroom, wearing a grey hoodie over a white shirt, sitting at a wooden desk and writing in a spiral notebook with a pen. Other students are blurred in the background.

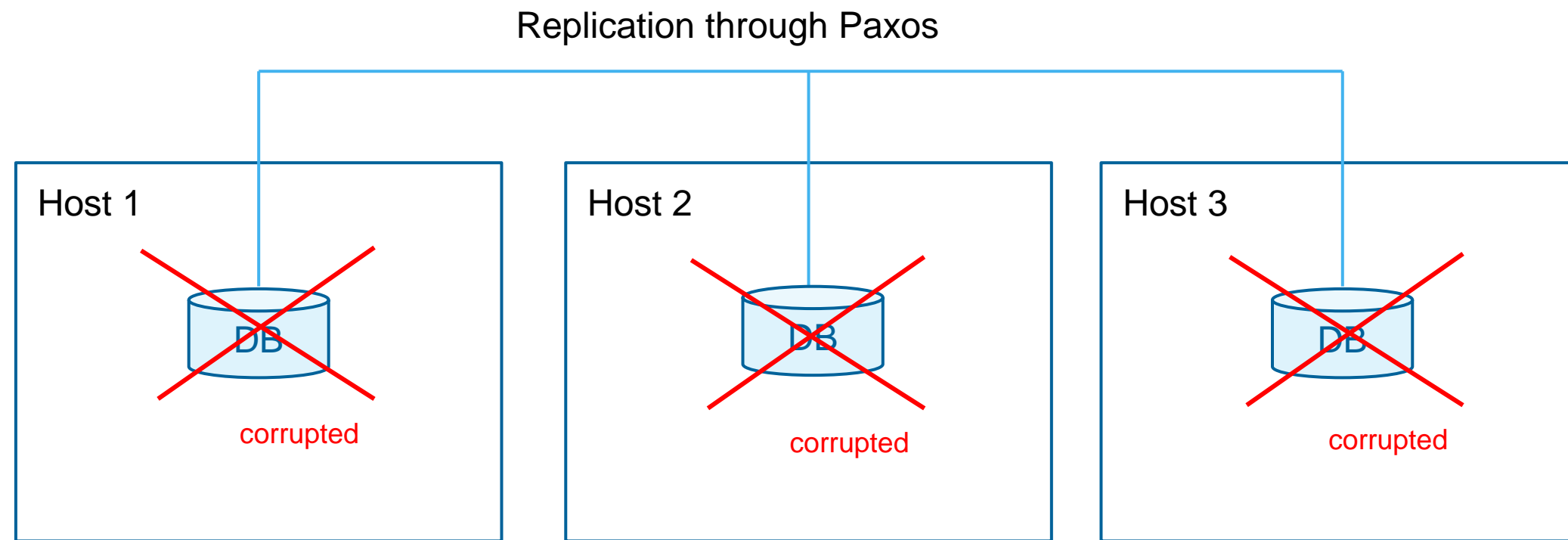
Lesson

It is beneficial to detect data corruptions earlier, rather than eventually.

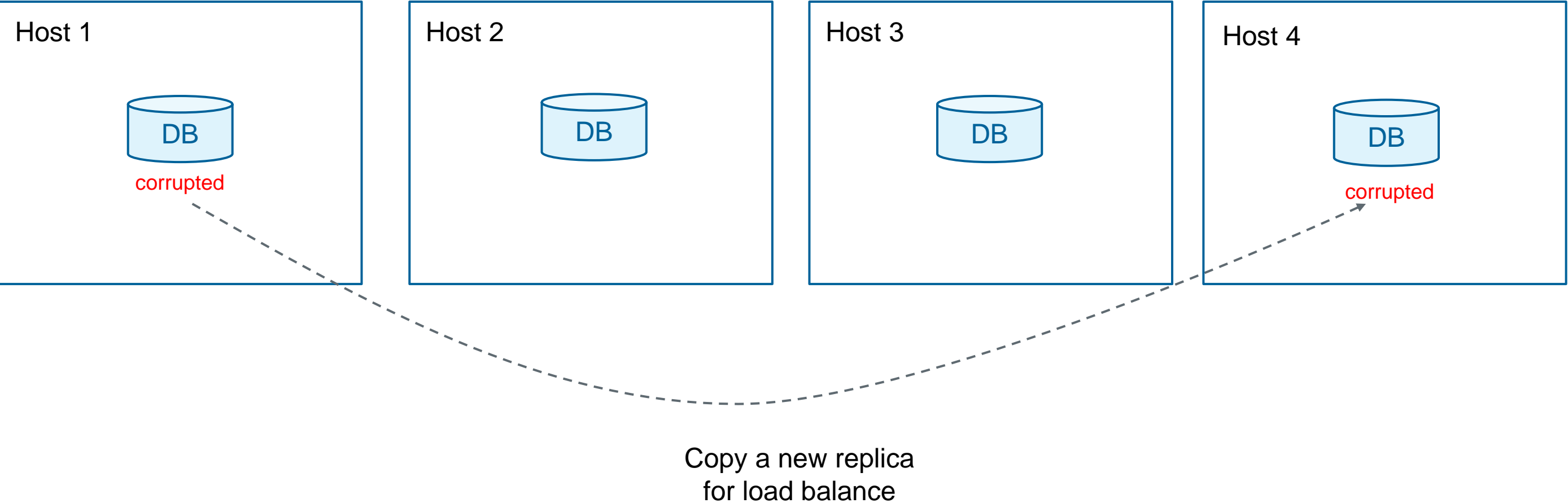
Detectable corruptions can be corrected by copying a healthy replica



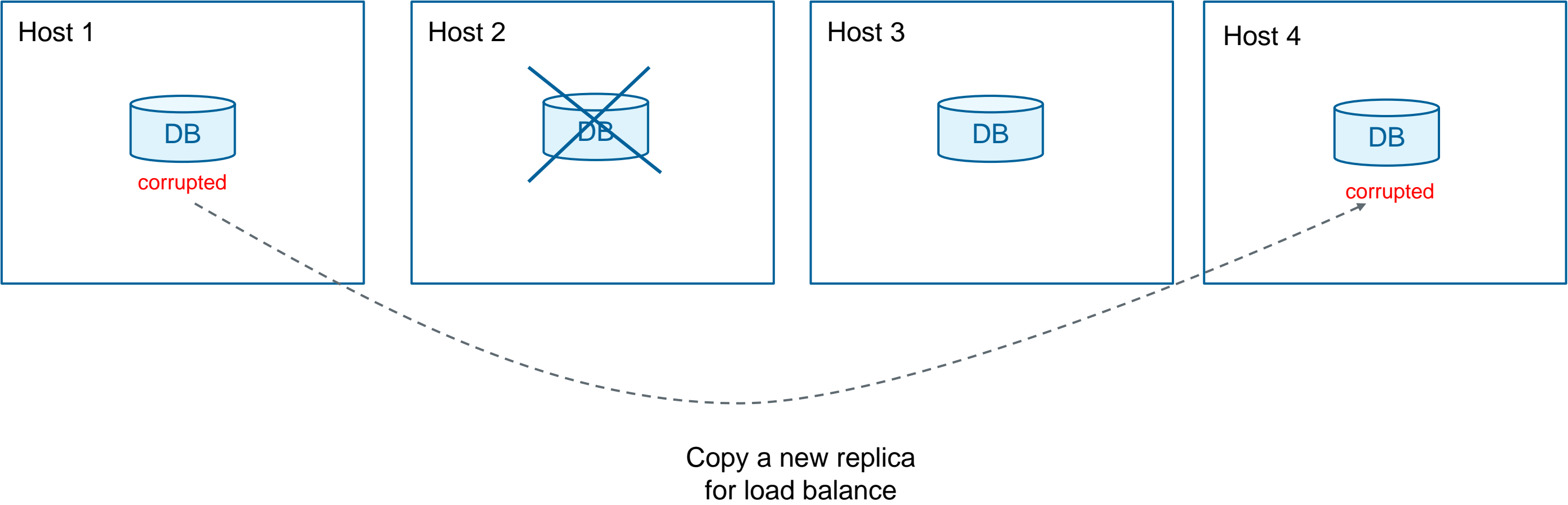
Delayed corruption detection risks data loss



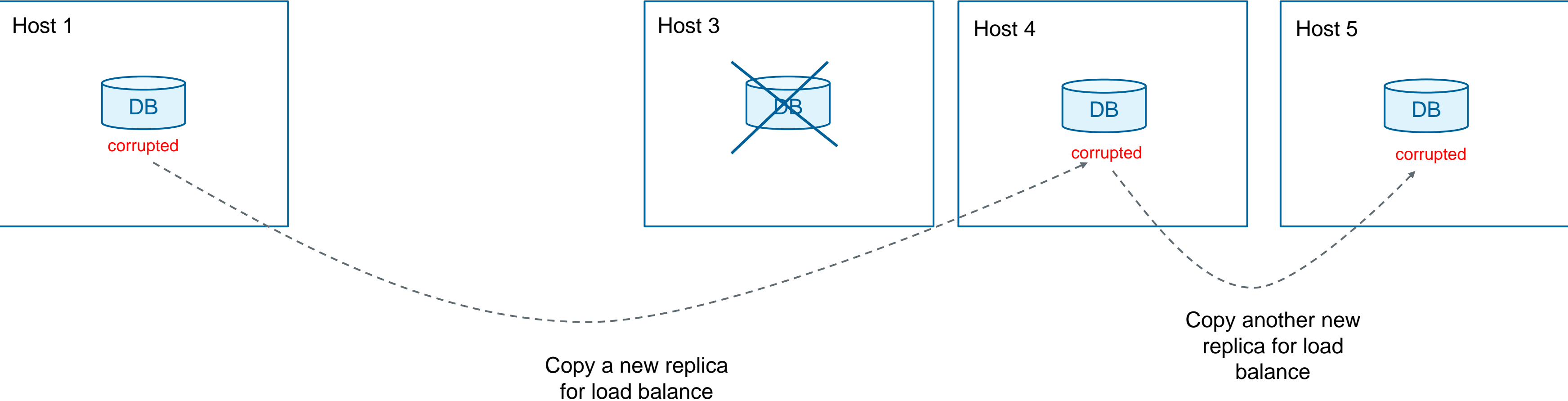
Unchecked Data Copying Risk Data Loss



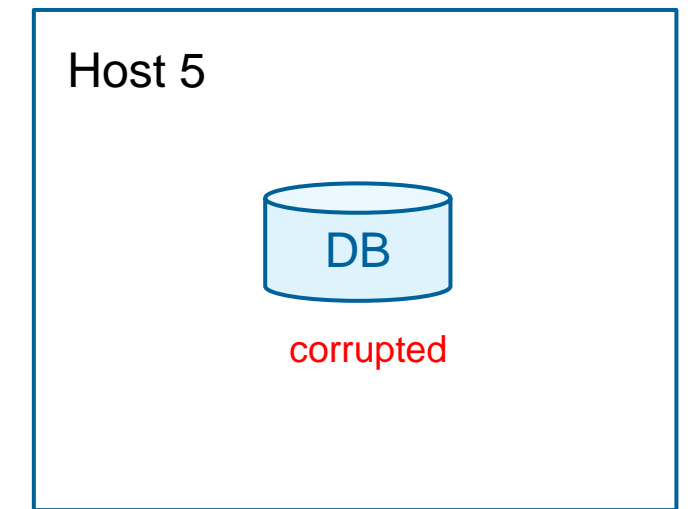
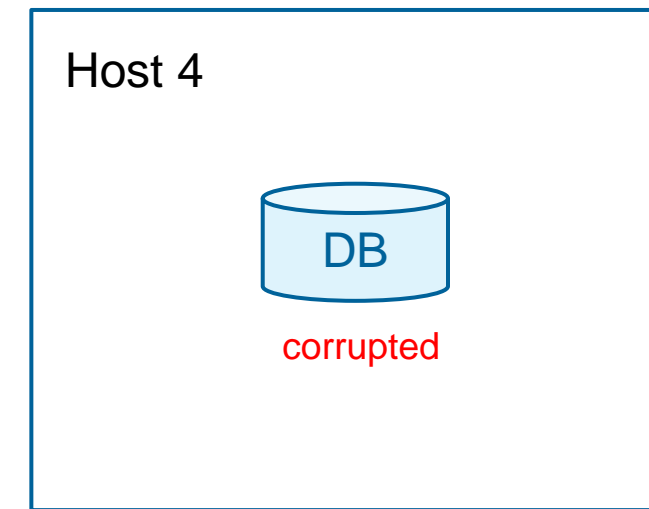
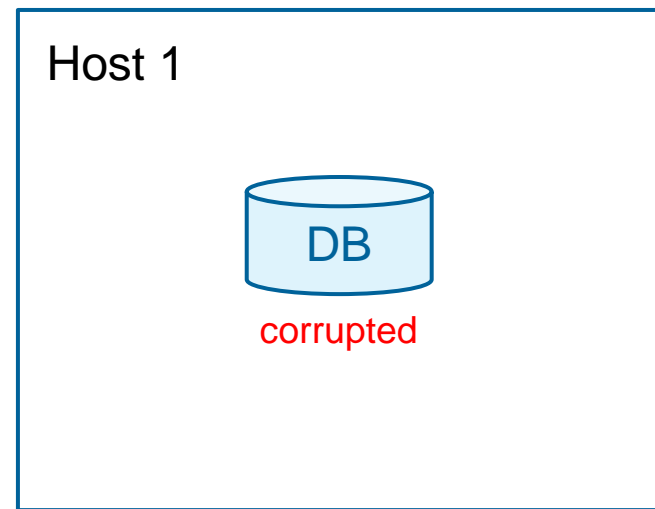
Unchecked Data Copying Risk Data Loss



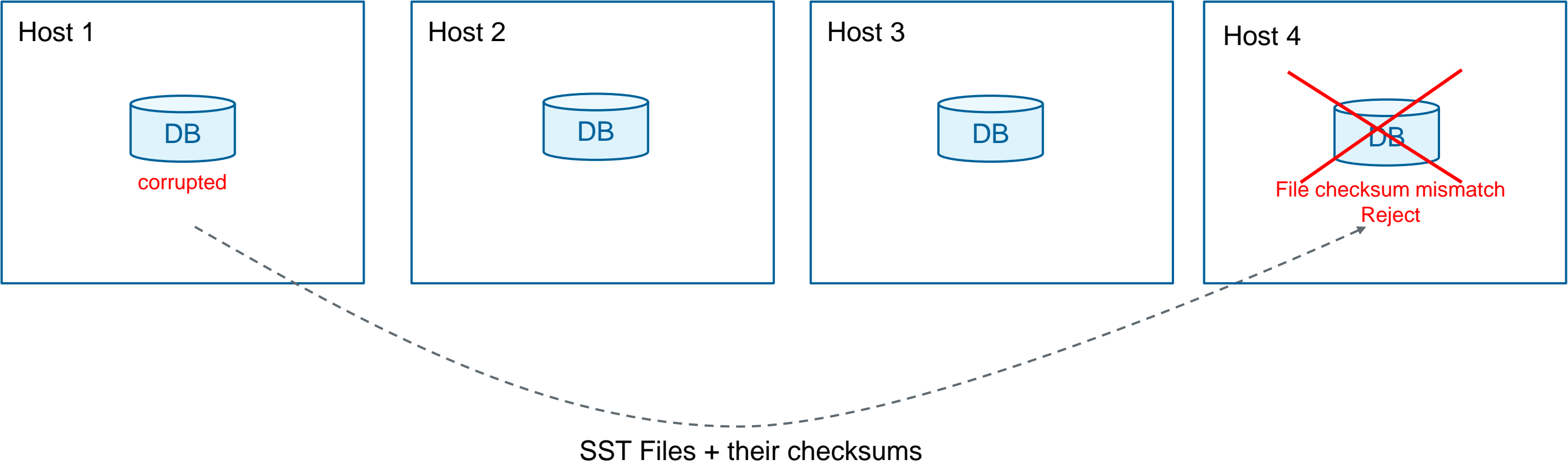
Unchecked Data Copying Risk Data Loss



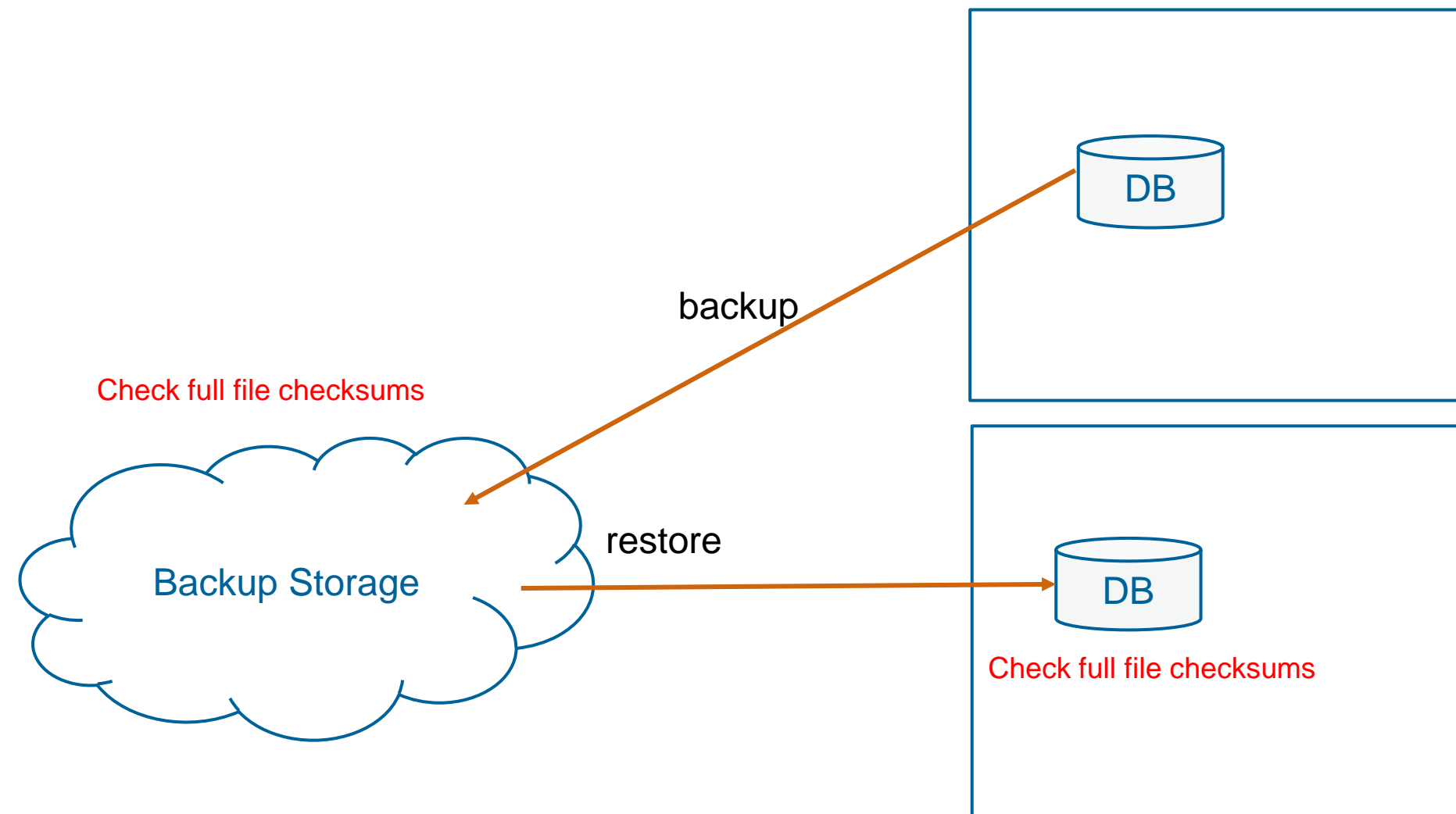
Unchecked Data Copying Risk Data Loss



Keep Full SST File Checksums



Keep Full SST File Checksums

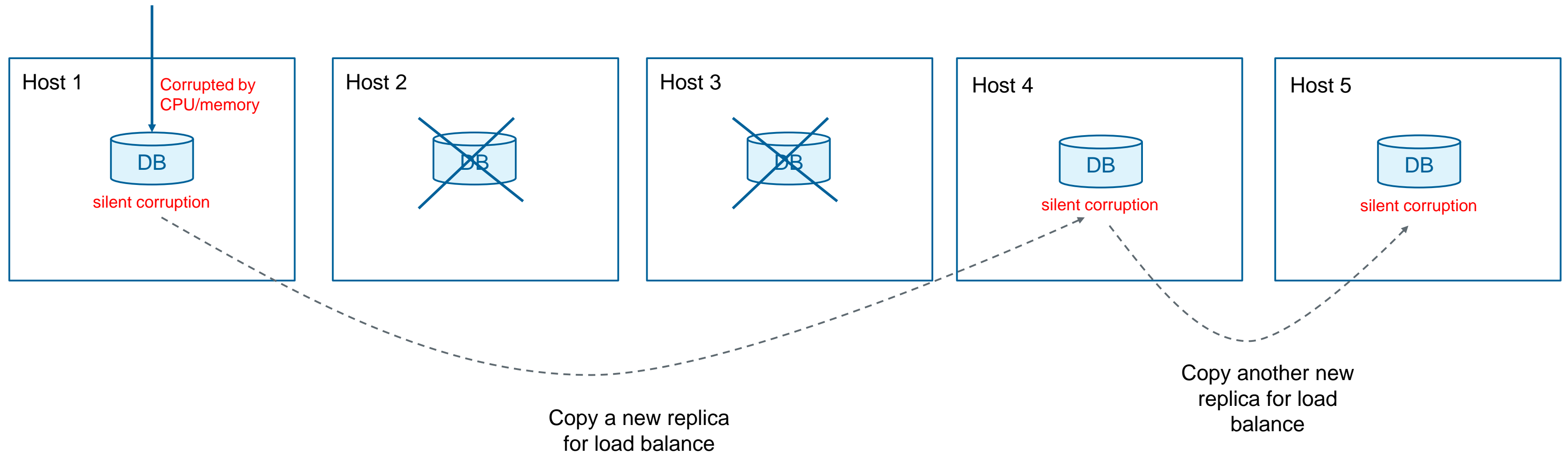


A background image of a student in a classroom, wearing a grey hoodie and writing in a spiral notebook. The image is dimmed to serve as a backdrop for the text.

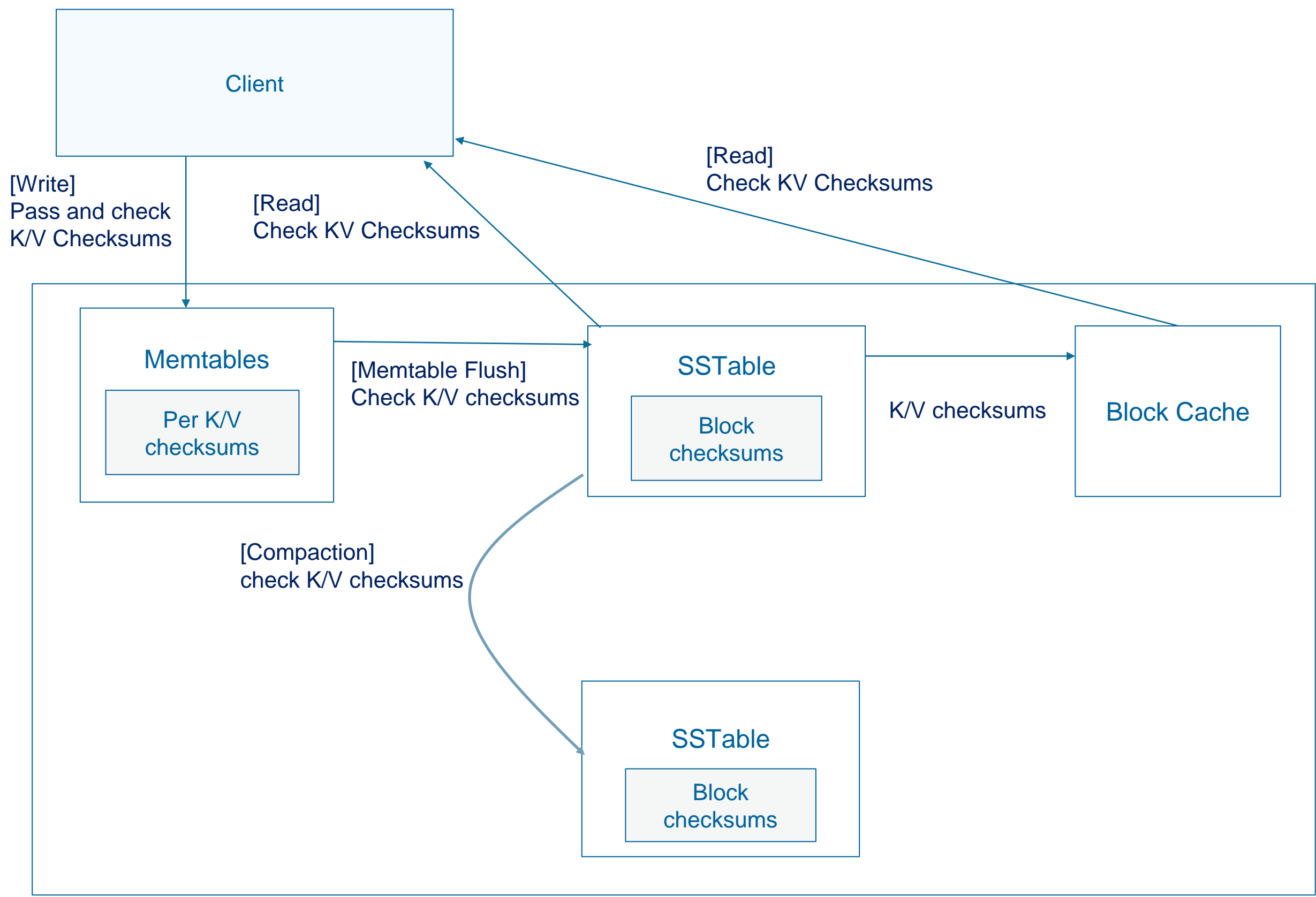
Lesson

CPU corruption does happen, though very rarely, and sometimes cannot be handled by data replication.

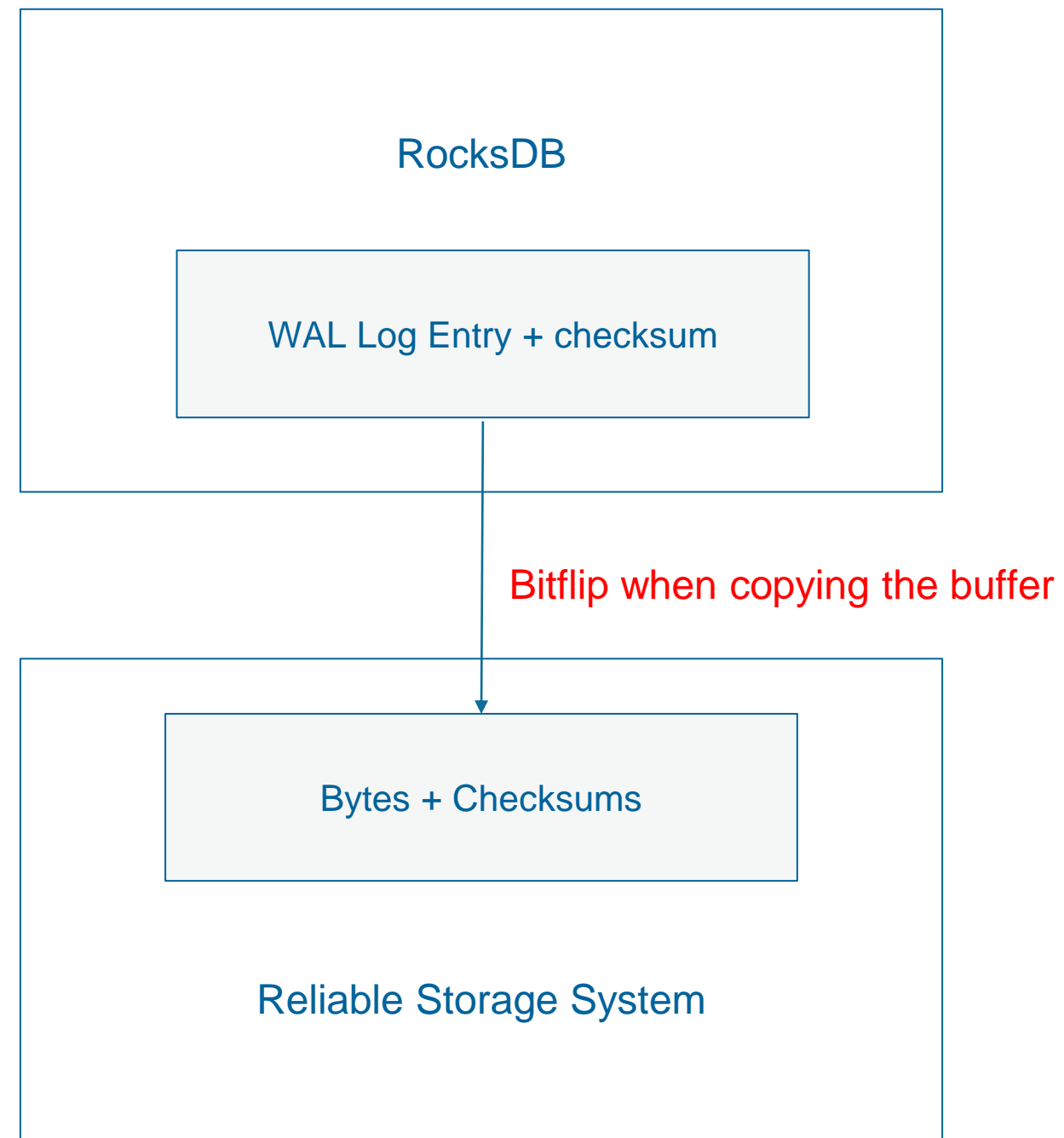
CPU corruption is silent and can cause data loss



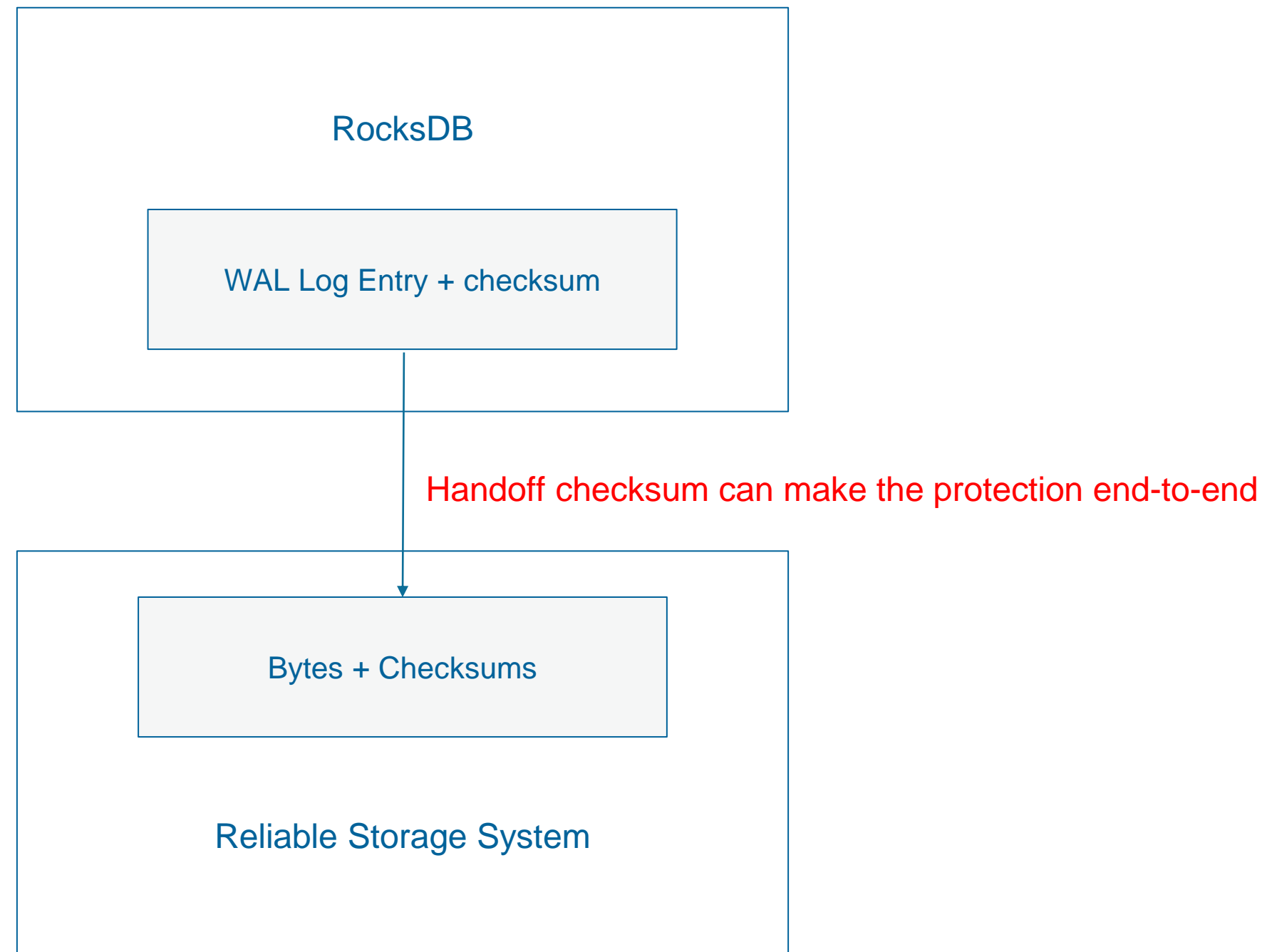
Use per K/V Checksum to Mitigate CPU corruption



CPU Corruption can cause handoff issues



CPU Corruption can cause handoff issues



Recap Lessons on Integrity Checking

- RocksDB users handle integrity errors by tossing back replica and building a new one, so delayed corruption detection risks data loss
- Full file checksums can help prevent corrupted copy from spreading
- CPU/memory corruption does happen, although very rarely
- Per KV checksum and handoff checksums could mitigate damage of CPU/memory corruption.



More in the paper

- More Production Data
- Thoughts in Newer Storage Technologies
- Lessons on serving large-scale systems
 - Resource management
 - WAL treatment
 - Rate-limit file deletions
 - Data format compatibility
 - Managing configurations
 - Replication and Backup Supports
- More lessons on failure handling
- Lessons on key-value interface

Evolution of Development Priorities in Key-value Stores Serving Large-scale Applications: The RocksDB Experience

Siying Dong[†], Andrew Kryczka[†], Yanqin Jin[†] and Michael Stumm[‡]

[†]Facebook Inc., 1 Hacker Way, Menlo Park, CA, U.S.A

[‡]University of Toronto, Toronto, Canada

Abstract

RocksDB is a key-value store targeting large-scale distributed systems and optimized for Solid State Drives (SSDs). This paper describes how our priorities in developing RocksDB have evolved over the last eight years. The evolution is the result both of hardware trends and of extensive experience running RocksDB at scale in production at a number of organizations. We describe how and why RocksDB's resource optimization target migrated from write amplification, to space amplification, to CPU utilization. Lessons from running large-scale applications taught us that resource allocation needs to be managed across different RocksDB instances, that data format needs to remain backward and forward compatible to allow incremental software rollout, and that appropriate support for database replication and backups are needed. Lessons from failure handling taught us that data corruption errors needed to be detected earlier and at every layer of the system.

1 Introduction

RocksDB [19, 54] is a high-performance persistent key-value storage engine created in 2012 by Facebook, based on Google's LevelDB code base [22]. It is optimized for the specific characteristics of Solid State Drives (SSDs), targets large-scale (distributed) applications, and is designed as a library component that is embedded in higher-level applications. As such, each RocksDB instance manages data on storage devices of just a single server node; it does not handle any inter-host operations, such as replication and load balancing, and it does not perform high-level operations, such as checkpoints — it leaves the implementation of these operations to the application, but provides appropriate support so they can do it effectively.

RocksDB and its various components are highly customizable, allowing the storage engine to be tailored to a wide spectrum of requirements and workloads; customizations can include the write-ahead log (WAL) treatment, the compression strategy, and the compaction strategy (a process that

removes dead data and optimizes LSM-trees as described in §2). RocksDB may be tuned for high write throughput or high read throughput, for space efficiency, or something in between. Due to its configurability, RocksDB is used by many applications, representing a wide range of use cases. At Facebook alone, RocksDB is used by over 30 different applications, in aggregate storing many hundreds of petabytes of production data. Besides being used as a storage engine for *databases* (e.g., MySQL [37], Rocksandra [6], CockroachDB [64], MongoDB [40], and TiDB [27]), RocksDB is also used for the following types of services with highly disparate characteristics (summarized in Table 1):

- **Stream processing:** RocksDB is used to store staging data in Apache Flink [12], Kafka Stream [31], Samza [43], and Facebook's Stylus [15].
- **Logging/queuing services:** RocksDB is used by Facebook's LogDevice [5] (that uses both SSDs and HDDs), Uber's Cherami [8], and Iron.io [29].
- **Index services:** RocksDB is used by Facebook's Dragon [59] and Rockset [58].
- **Caching on SSD:** In-memory caching services, such as Netflix's EVCache [7], Qihoo's Pika [51] and Redis [46], store data evicted from DRAM on SSDs using RocksDB.

A prior paper presented an analysis of several database applications using RocksDB [11]. Table 2 summarizes some of the key system metrics obtained from production workloads.

Having a storage engine that can support many different use cases offers the advantage that the same storage engine can be used across different applications. Indeed, having each application build its own storage subsystem is problematic, as doing so is challenging. Even simple applications need to protect against media corruption using checksums, guarantee data consistency after crashes, issue the right system calls in the correct order to guarantee durability of writes, and handle errors returned from the file system in a correct manner. A well-established common storage engine can deliver sophistication in all those domains.

Additional benefits are achieved from having a common storage engine when the client applications run within a com-

Thank you!

Questions?

A group of people in a modern office setting are gathered around a table, looking at a laptop screen. The scene is brightly lit with natural light. In the background, there are wooden lockers or storage units. The text "FACEBOOK Infrastructure" is overlaid in white, bold, sans-serif font across the middle of the image. The woman on the left has curly hair and is wearing a red plaid shirt. The man next to her is wearing a yellow beanie and a grey hoodie with white headphones around his neck. Another person is visible in the foreground, out of focus, wearing a blue shirt. The overall atmosphere is collaborative and professional.

FACEBOOK Infrastructure