Georgia Tech

Lecture 11: Write-Behind Logging

CREATING THE NEXT®

Administrivia

- Start preparing for mid-term exam

Today's Agenda

Recap

Disk-oriented vs In-Memory DBMSs

Persistent Memory DBMSs

Storage Engine Architectures

Recap

Larger-than-Memory Databases

- Allow an in-memory DBMS to store/access data on disk **without** bringing back all the slow parts of a disk-oriented DBMS.
  - Minimize the changes that we make to the DBMS that are required to deal with disk-resident data.
  - It is better to have only the **buffer manager** deal with moving data around
  - Rest of the DBMS can assume that data is in DRAM.
- Need to be aware of hardware access methods
  - In-memory Access = **Tuple**-Oriented.
  - Disk Access = **Block**-Oriented.

Design Decisions

- **Run-time Operation**
  - Cold Data Identification: When the DBMS runs out of DRAM space, what data should we evict?
- **Eviction Policies**
  - Timing: When to evict data?
  - Evicted Tuple Metadata: During eviction, what meta-data should we keep in DRAM to track disk-resident data and avoid false negatives?
- **Data Retrieval Policies**
  - Granularity: When we need data, how much should we bring in?
  - Merging: Where to put the retrieved data?

Reference

Persistent Memory DBMSs

Importance of Hardware

- People have been thinking about using hardware to accelerate DBMSs for decades.
- 1980s: Database Machines
- 2000s: FPGAs + Appliances
- 2010s: FPGAs + GPUs
- 2020s: PM + FPGAs + GPUs + CSAs + More!
- Reference

Persistent Memory

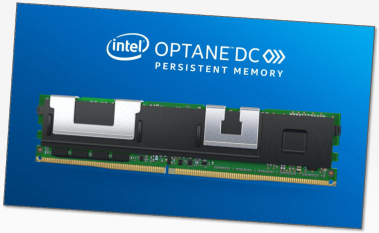
- Emerging storage technology that provide low latency read/writes like DRAM, but with persistent writes and large capacities like SSDs.
  - *a.k.a.*, Non-Volatile Memory, Storage-class Memory
- First-generation devices were block-addressable
- Second-generation devices are byte-addressable

Persistent Memory

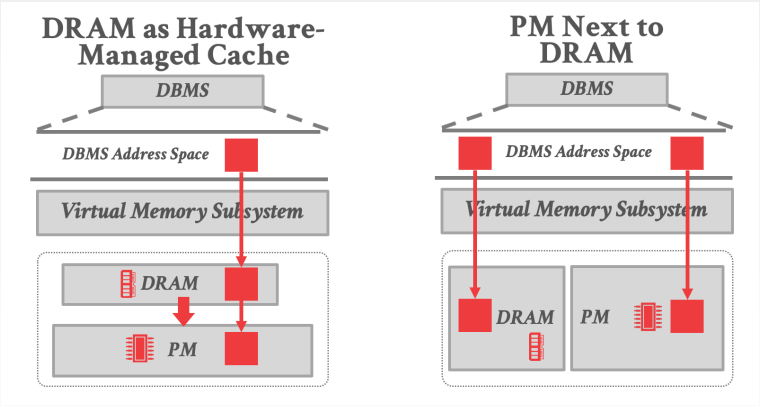
- Block-addressable Optane SSD
  - NVM Express works with PCI Express to transfer data to and from Optane SSDs
  - NVMe enables rapid storage in SSDs and is an improvement over older HDD-related interfaces (*e.g.*, Serial Attached SCSI (**SAS**) and Serial ATA (**SATA**))
- Byte-addressable Optane DIMMs
  - New assembly instructions and hardware support

Why This is for Real

- Industry has agreed to standard technologies and form factors (JDEC).
- Linux and Microsoft added support for PM in their kernels (DAX).
- Intel added new instructions for flushing cache lines to PM (CLFLUSH, CLWB).



PM Configurations



Reference

PM for Database Systems

- Block-addressable PM is not that interesting.
- Byte-addressable PM will be a game changer but will require some work to use correctly.
  - In-memory DBMSs will be better positioned to use byte-addressable PM.
  - Disk-oriented DBMSs will initially treat PM as just a faster SSD.

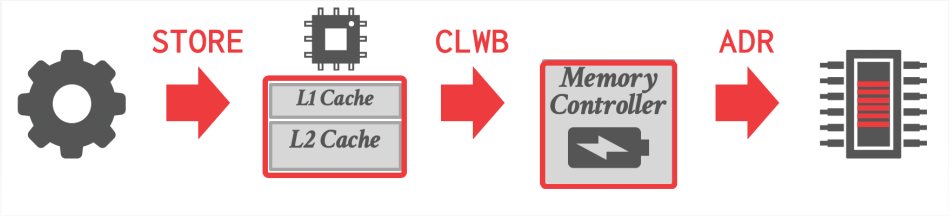
Storage & Recovery Methods

- Understand how a DBMS will behave on a system that only has byte-addressable PM.
- Develop PM-optimized implementations of standard DBMS architectures.
- Based on the N-Store prototype DBMS.
- Reference

Synchronization

- Existing programming models assume that any write to memory is non-volatile.
  - CPU decides when to move data from caches to DRAM.
- The DBMS needs a way to ensure that data is flushed from caches to PM.

Synchronization

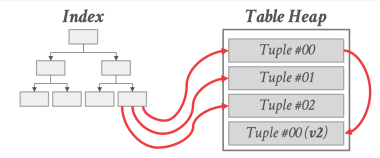


Synchronization

- Cache-line Flush (CLFLUSH)
  - This instruction allows the DBMS to flush a cache-line out to memory.
  - If that cache line contains modified data at any level of the cache hierarchy, that data is written back to memory.
- Cache-line Write Back (CLWB)
  - Writes back the cache line (if modified) to memory
  - The cache line may be retained in the cache hierarchy in non-modified state
  - Improves performance by reducing cache misses
  - CLWB instruction is ordered only by store-fencing (SFENCE) operation.
- Asynchronous DRAM Refresh (ADR)
  - In case of a power loss, there is sufficient reserve power to flush the stores pending in the memory controller back to Optane DIMM.
  - Stores are posted to the Write Pending Queue (WPQ) in the memory controller
- Reference

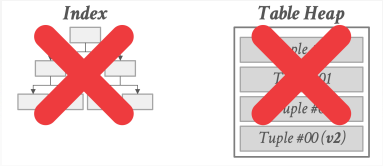
Naming

- If the DBMS process restarts, we need to make sure that all the pointers for in-memory data point to the same data.



Naming

- If the DBMS process restarts, we need to make sure that all the pointers for in-memory data point to the same data.



PM-Aware Memory Allocator

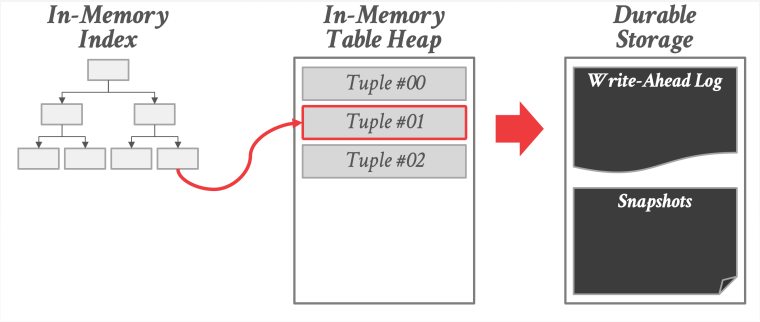
- **Feature 1: Synchronization**
  - The allocator writes back CPU cache lines to PM using the CLFLUSH instruction.
  - It then issues a SFENCE instruction to wait for the data to become durable on PM.
- **Feature 2: Naming**
  - The allocator ensures that virtual memory addresses assigned to a memory-mapped region never change even after the OS or DBMS restarts.

Storage Engine Architectures

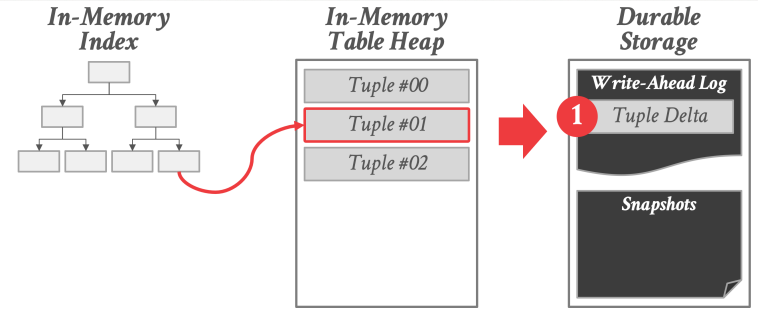
Storage Engine Architectures

- **Choice 1: In-place Updates**
  - Table heap with a write-ahead log + snapshots.
  - Example: VoltDB
- **Choice 2: Copy-on-Write**
  - Create a shadow copy of the table when updated.
  - No write-ahead log.
  - Example: LMDB
- **Choice 3: Log-structured**
  - All writes are appended to log. No table heap.
  - Example: RocksDB

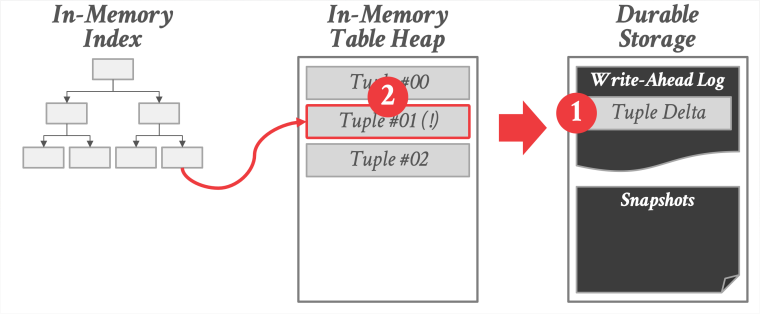
In-place Updates Engine



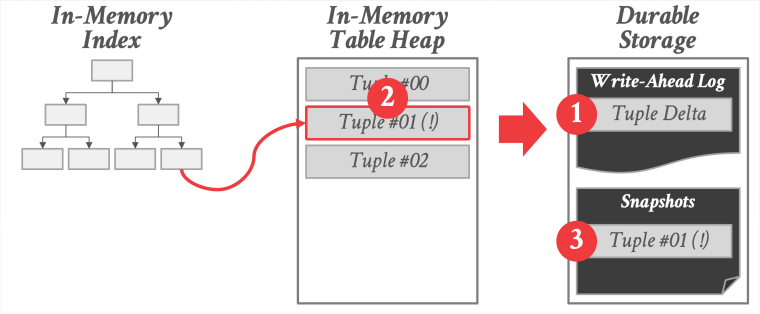
In-place Updates Engine



In-place Updates Engine



In-place Updates Engine



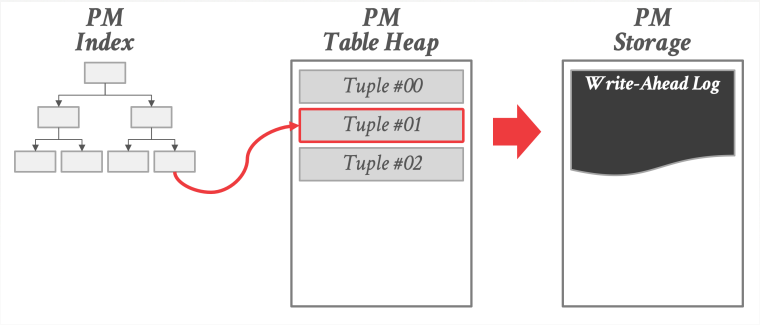
In-place Updates Engine

- Limitations
  - Duplicate Data
  - Recovery Latency

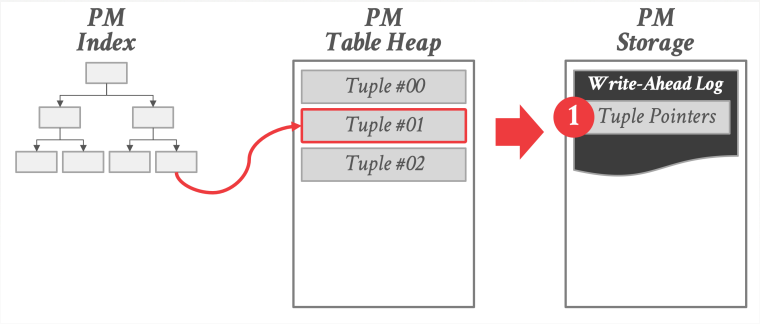
PM-Aware Architectures

- Leverage the allocator's **non-volatile pointers** to only record **what** changed rather than **how** it changed.
- The DBMS only must maintain a transient **UNDO log** for a txn until it commits.
  - Dirty cache lines from an uncommitted txn can be flushed by hardware to the memory controller.
  - **No REDO log** because we flush all the changes to PM at the time of commit.

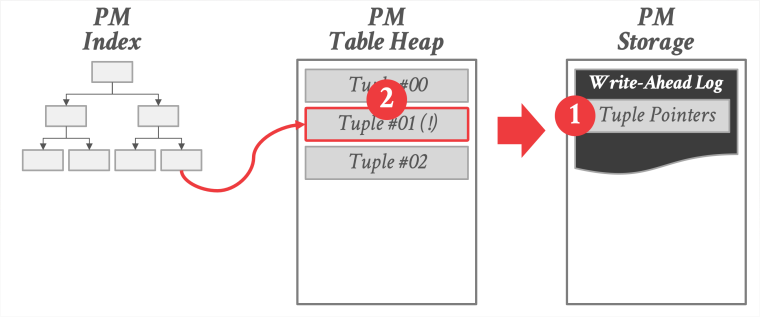
PM-Aware In-place Updates Engine



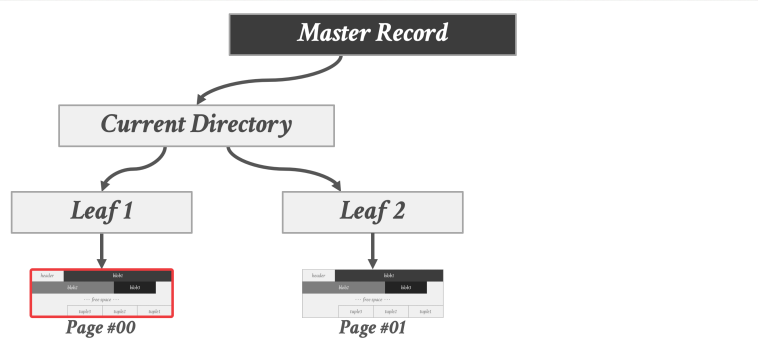
PM-Aware In-place Updates Engine



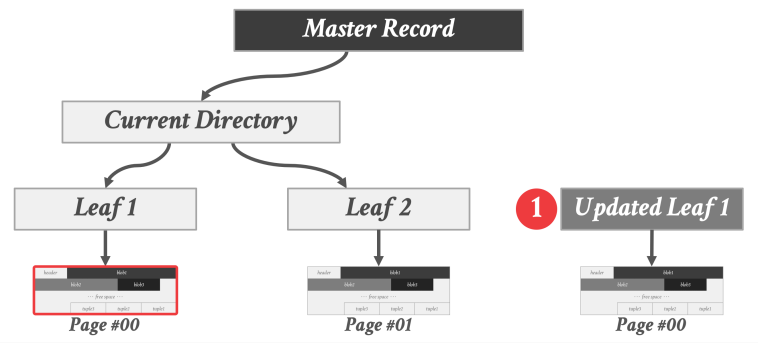
PM-Aware In-place Updates Engine



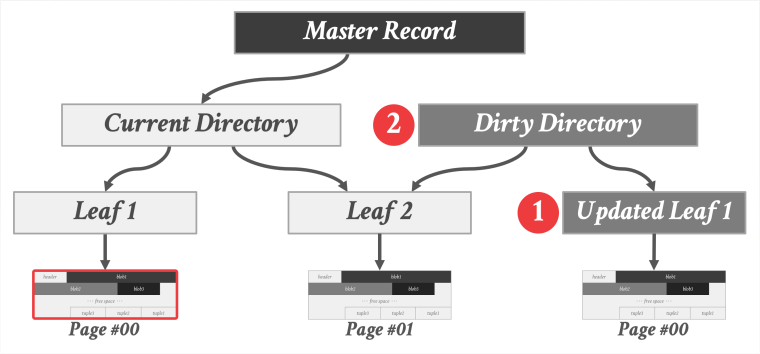
Copy-On-Write Engine



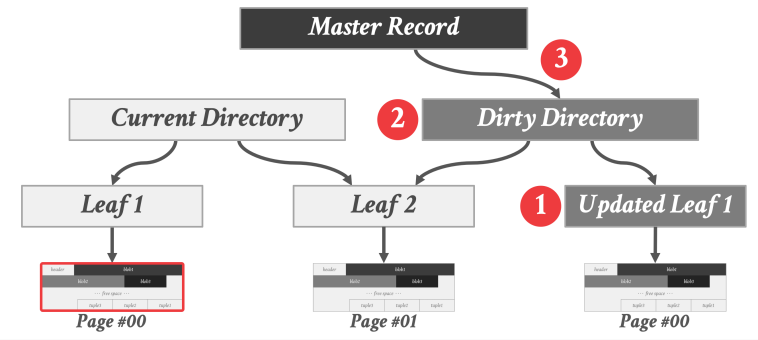
Copy-On-Write Engine



Copy-On-Write Engine



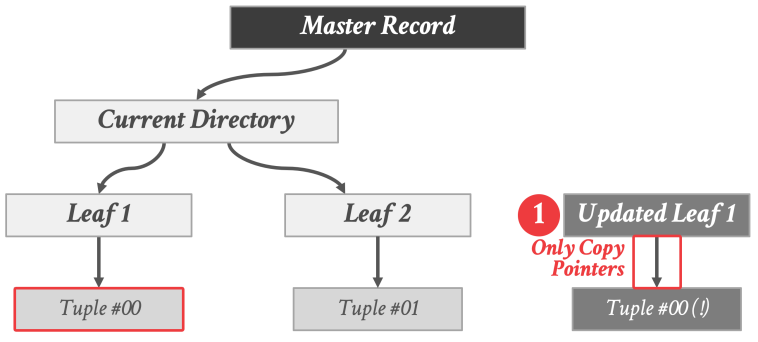
Copy-On-Write Engine



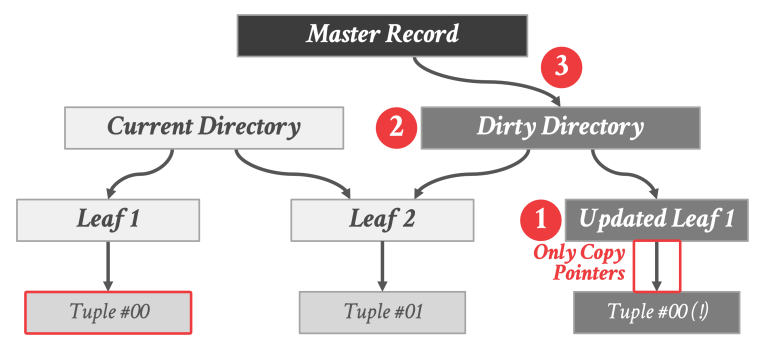
Copy-On-Write Engine

- Limitations
  - Expensive Copies

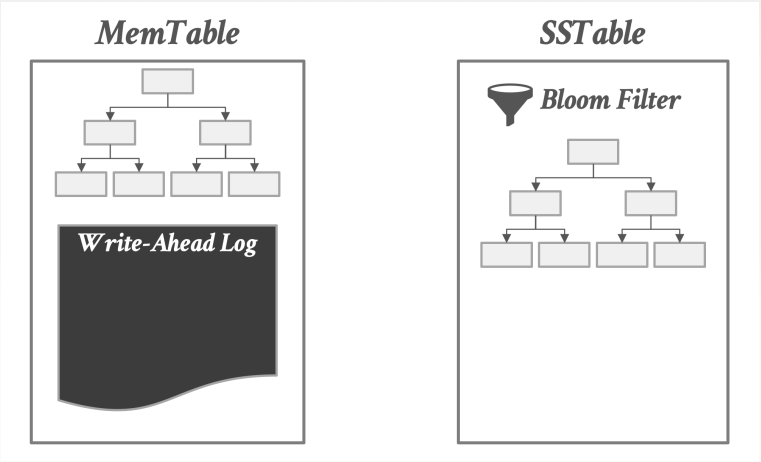
PM-Aware Copy-On-Write Engine



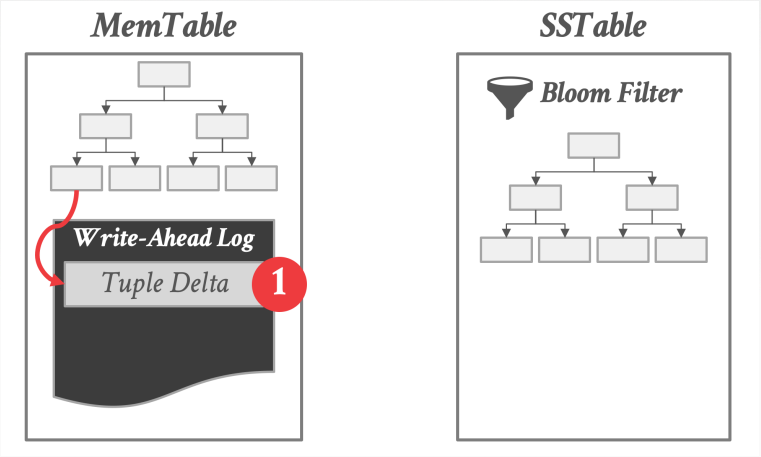
PM-Aware Copy-On-Write Engine



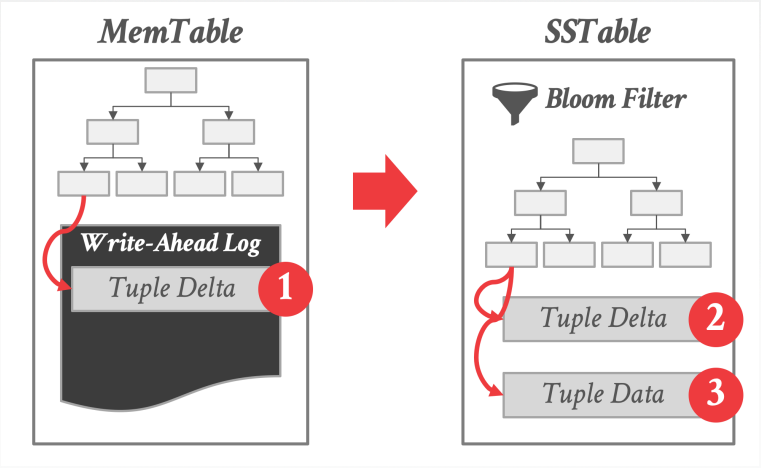
Log-Structured Engine



Log-Structured Engine



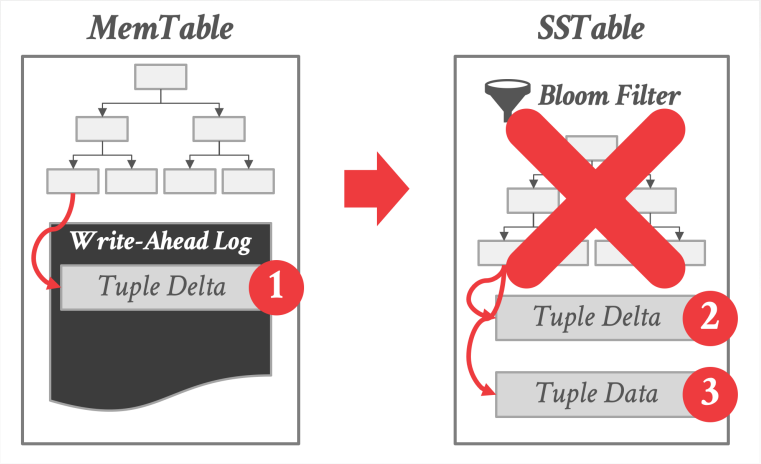
Log-Structured Engine



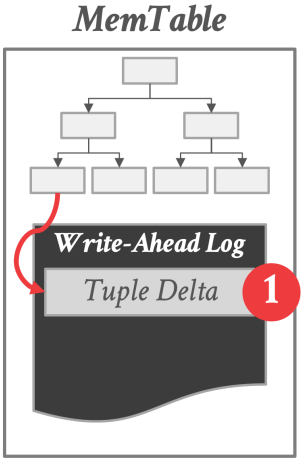
Log-Structured Engine

- Limitations
  - Duplicate Data
  - Compactions

PM-Aware Log-Structured Engine



PM-Aware Log-Structured Engine



PM Summary

- Optimization of **Storage Engine Architectures**
  - Leverage byte-addressability to **avoid unnecessary data duplication**.

Conclusion

- The design of a in-memory DBMS is significantly different than a disk-oriented system.
- The world has finally become comfortable with in-memory data storage and processing.