L-Store: Lineage-based Storage Architectures

ECS165A: Winter 2021

Slides are adopted from Sadoghi, et al.

*L-Store: A Real-time OLTP and OLAP System, EDBT’18*
Data Management at Macroscale: The Four V’s of Big Data

John Doe

Anthem BlueCross BlueShield

Walmart

FICO Score

MasterCard

VISA

IRS
Data Management at Macroscale: The Four V’s of Big Data

Motivations
L-Store
Evaluation
Conclusions
Data Management at Microscale: Volume & Velocity

OLTP
(Write-optimized)

Data Velocity

Sales

OLTP
(Write-optimized)
Data Management at Microscale: Volume & Velocity

OLAP (Read-optimized)  
Extract-Transform-Load (ETL)  
OLTP (Write-optimized)

Data Velocity

Data is Stale

Data Volume

Sales

Walmart

Mohammad Sadoghi  
ECS165A  
2 / 16
Data Management at Microscale: Volume & Velocity

OLAP (Read-optimized)
OLTP (Write-optimized)
Data Volume
Data Velocity
Sales
Reports

Walmart

OLTP
(Write-optimized)

Extract-Transform-Load (ETL)
One Size Does not Fit All As of 2012

Big Data Landscape

Copyright © 2012 Dave Feinleib
dave@rccrave.com
blogs.forbes.com/davefeinleib
One Size Does not Fit All As of 2017
Data Management at Microscale: Volume & Velocity

OLAP+OLTP (Read & Write-optimized)

Walmart

Reports

Sales

OLAP+OLTP

SAP HANA

ORACLE

Microsoft SQL Server

IBM BLU Acceleration

Fast answers, simply delivered.
Write-optimized (i.e., uncompressed & row-based) vs. read-optimized (i.e., compressed & column-based) layouts
Unifying OLTP and OLAP: Velocity & Volume Dimensions

Observed Trends

In operational databases, there is a pressing need to close the gap between the write-optimized layout for OLTP (i.e., row-wise) and the read-optimized layout for OLAP (i.e., column-wise).
Unifying OLTP and OLAP: Velocity & Volume Dimensions

Observed Trends

In operational databases, there is a pressing need to close the gap between the write-optimized layout for OLTP (i.e., row-wise) and the read-optimized layout for OLAP (i.e., column-wise).

Introducing a *lineage-based storage architecture*, a contention-free update mechanism over a native columnar storage in order to
Unifying OLTP and OLAP: Velocity & Volume Dimensions

Observed Trends

In operational databases, there is a pressing need to close the gap between the write-optimized layout for OLTP (i.e., row-wise) and the read-optimized layout for OLAP (i.e., column-wise).

Introducing a *lineage-based storage architecture*, a contention-free update mechanism over a native columnar storage in order to lazily and independently stage stable data from a write-optimized layout (i.e., OLTP) into a read-optimized layout (i.e., OLAP)
Lineage-based Storage Architecture (LSA): Intuition

Physical Update Independence: De-coupling data & its updates (reconstruction via in-page lineage tracking and lineage mapping)
Lineage-based Storage Architecture (LSA): Intuition

- **Base Pages** (Read-only)
- **Tail Pages** (Append-only)
- **Index**

**Lineage Mapping**
- Indirection layer, stable LID-to-RID mapping

**Monotonically Increasing Lineage**
- Updates are assigned RIDs in an increasing order

**Physical Update Independence: De-coupling data & its updates**
- Reconstruction via in-page lineage tracking and lineage mapping

Mohammad Sadoghi

ECS165A
**Lineage-based Storage Architecture (LSA): Intuition**

- **Monotonically Increasing Lineage**: Updates are assigned RIDs in an increasing order.
- **Lineage Mapping**: Indirection layer, stable LID-to-RID mapping.
- **Base Pages**: Read-only, stable anchored RIDs.
- **Tail Pages**: Append-only, physical update independence.
- **Latest Version**: Monotonically increasing RIDs.
- **Append-only Updates**: Physical update independence.
- **Index**: Points to stable LIDs (i.e., anchored RID).
- **Lazy Update Consolidation**: Snapshot reconstruction via lineage mapping & in-page tracking.
- **In-page Lineage Tracking**: Data Block RIDs remain unchanged (stable reference, anchored RIDs).
- **Consolidated Data**: Read-only.
- **Physical Update Independence**: Decoupling data & its updates (reconstruction via in-page lineage tracking and lineage mapping).

---

**Mohammad Sadoghi**

**ECS165A**

**7 / 16**
Overview of the lineage-based storage architecture
(base pages and tail pages are handled identically at the storage layer)
Records are range-partitioned and compressed into a set of ready-only base pages (accelerating analytical queries)
Recent updates for a range of records are clustered in their **tails pages** (transforming costly point updates into an amortized analytical-like query)
Recent updates for a range of records are clustered in their tails pages (transforming costly point updates into an amortized analytical-like query).
Recent updates are strictly appended, uncompressed in the pre-allocated space (eliminating the read/write contention)
L-Store: Detailed Design

Achieving (at most) 2-hop access to the latest version of any record (avoiding read performance deterioration for point queries)
L-Store: Detailed Design

- **Write Optimized**
  - (uncompressed, append-only updates)
- **Indirection Column**
  - (uncompressed, in-place update)
- **Indirection Column**
  - (back pointer to the previous version)
- **New Version**
- **Read Optimized**
  - (compressed, read-only pages)

Achieving (at most) 2-hop access to the latest version of any record (avoiding read performance deterioration for point queries)
L-Store: Detailed Design

- **Indirection Column (back pointer to the previous version)**
- **Backward Pointer**
- **Write Optimized (uncompressed, append-only updates)**
- **New Version**
- **Indirection Column (uncompressed, in-place update)**
- **Read Optimized (compressed, read-only pages)**

Achieving (at most) 2-hop access to the latest version of any record (avoiding read performance deterioration for point queries)
Motivations

L-Store

Evaluation

Conclusions

L-Store: Contention-free Merge

Contention-free merging of only stable data: read-only and committed data (no need to block on-going and new transactions)
L-Store: Contention-free Merge

Lazy independent merging of **base pages** with their corresponding **tail pages** (resembling a local left outer-join of the base and tail pages)
L-Store: Contention-free Merge

Asynchronous Lazy Merge (committed, consecutives updates)

Read Optimized (compressed, read-only pages)

Indirection Column (uncompressed, in-place update)

Write Optimized (uncompressed, append-only updates)

In-page, Independent Lineage Tracking

Indepedently tracking the lineage information within every page (no need to coordinate merges among different columns of the same records)
L-Store: Epoch-based Contention-free De-allocation

Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)
Contenion-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)
Contention-free page de-allocation using an epoch-based approach (no need to drain the ongoing transactions)
L-Store: Epoch-based Contention-free De-allocation

Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)
L-Store: Epoch-based Contention-free De-allocation

Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)
Experimental Analysis
Experimental Settings

- **Hardware:**
  - 2 × 6-core Intel(R) Xeon(R) CPU E5-2430 @ 2.20GHz, 64GB, 15 MB L3 cache

- **Workload:** Extended Microsoft Hekaton Benchmark
  - Comparison with *In-place Update + History* and *Delta + Blocking Merge*
  - Effect of varying contention levels
  - Effect of varying the read/write ratio of short update transactions
  - Effect of merge frequency on scan
  - Effect of varying the number of short update vs. long read-only transactions
  - Effect of varying L-Store data layouts (row vs. columnar)
  - Effect of varying the percentage of columns read in point queries
  - Comparison with log-structured storage architecture (*LevelDB*)
Achieving up to $40 \times$ as increasing the update contention
Effect of Merge Frequency on Scan Performance

Mixed OLTP + OLAP Workload; Low Contention
(1 Scan + 1 Merge Threads, Page Size = 32 KB)

Scan Execution Time (in seconds)
Number of Tail Records Processed per Merge

Merge process is essential in maintaining efficient scan performance.
Effect of Mixed Workloads: Update Performance

Mixed OLTP + OLAP Workload; Medium Contention
(Total of 17 Threads + 1 Merge Thread, Page Size = 32 KB)

Eliminating latching & locking results in a substantial performance improvement
Effect of Mixed Workloads: Read Performance

Mixed OLTP + OLAP Workload; Medium Contention
(Total of 17 Threads + 1 Merge Thread, Page Size = 32 KB)

Read Throughput (txn/s)

Number of Parallel Read-only Transactions

Coping with tens of update threads with a single merge thread
L-Store Key Contributions

- Unifying OLAP & OLTP by introducing lineage-based storage architecture (LSA)
- LSA is a native multi-version, columnar storage model that lazily & independently stages data from a write-optimized layout into a read-optimized one
- Contention-free merging of only stable data without blocking ongoing or incoming transactions
- Contention-free page de-allocation without draining ongoing transactions
- L-Store outperforms in-place update & delta approaches by factor of up to 8× on mixed OLTP/OLAP workloads and up to 40× on update-intensive workloads
Questions?

Thank you!

Exploratory Systems Lab (ExpoLab)
Website: https://expolab.org/