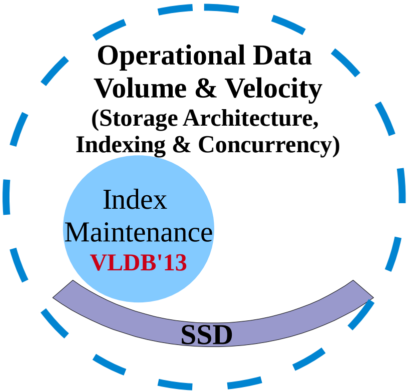
Concurrency Protocols in L-Store

Mohammad Sadoghi

Exploratory Systems Lab
University of California, Davis

ECS165a - Winter 2020

Extending Storage Hierarchy with Indirection Layer



Reducing Index maintenance: Velocity Dimension

Observed Trends

In the absence of in-place updates in operational multi-version databases, the cost of index maintenance becomes a major obstacle to cope with data velocity.

Reducing Index maintenance: Velocity Dimension

Observed Trends

In the absence of in-place updates in operational multi-version databases, the cost of index maintenance becomes a major obstacle to cope with data velocity.

Extending storage hierarchy (using fast non-volatile memory) with *an extra level of indirection* in order to

Reducing Index maintenance: Velocity Dimension

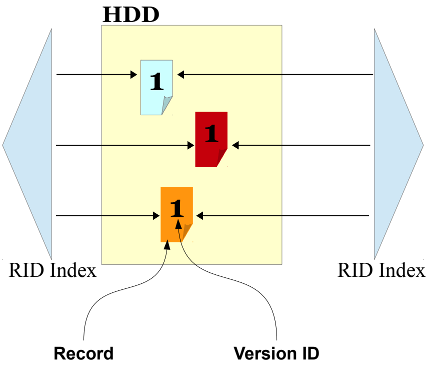
Observed Trends

In the absence of in-place updates in operational multi-version databases, the cost of index maintenance becomes a major obstacle to cope with data velocity.

Extending storage hierarchy (using fast non-volatile memory) with *an extra level of indirection* in order to

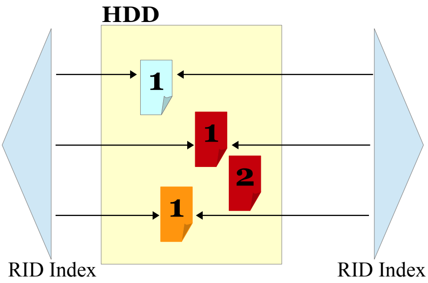
Decouple Logical and Physical Locations of Records to Reduce Index Maintenance

Traditional Multi-version Indexing: Updating Records



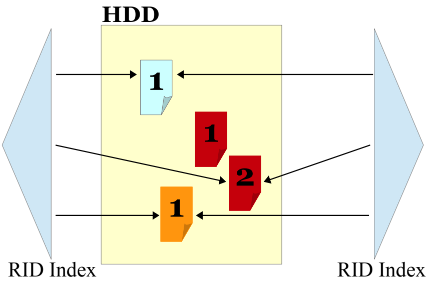
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records



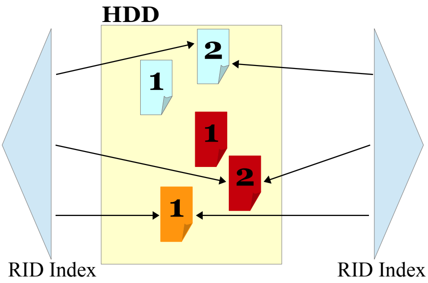
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records



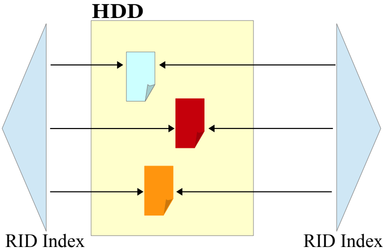
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records

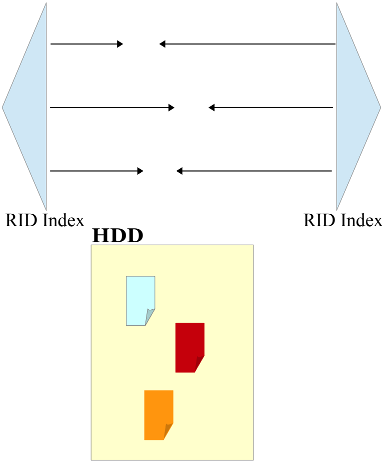


Updating random leaf pages

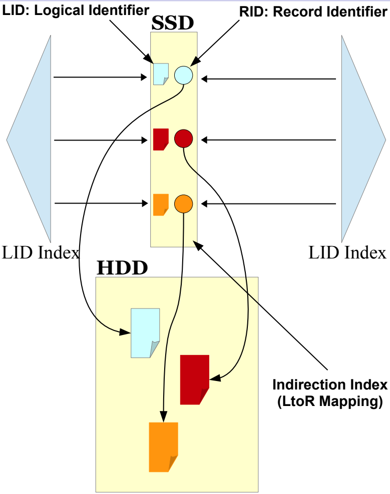
Indirection Indexing: Updating Records



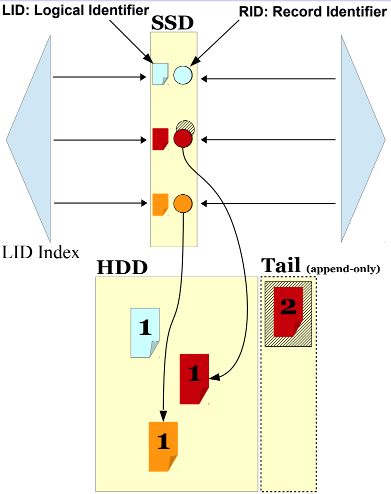
Indirection Indexing: Updating Records



Indirection Indexing: Updating Records

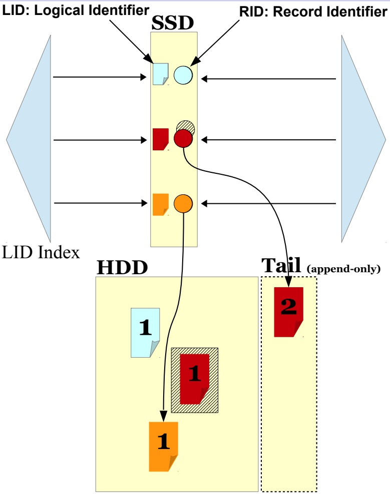


Indirection Indexing: Updating Records



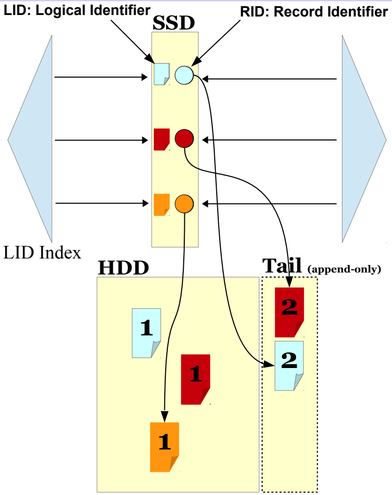
Eliminating random leaf-page updates

Indirection Indexing: Updating Records



Eliminating random leaf-page updates

Indirection Indexing: Updating Records



Eliminating random leaf-page updates

Analytical & Experimental Evaluations

Indirection Time Complexity Analysis

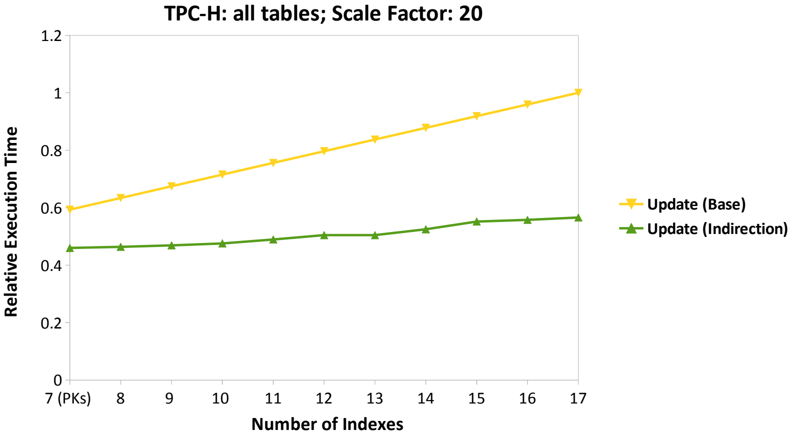
	Legend
K	Number of indexes
LB	LIDBlock size
M	Number of matching records

Method	Type	Imm. SSD	Def. SSD	Imm. HDD	Def. HDD
Base	Deletion	0	0	$2+K$	$\leq 1+K$
	Single-attr. update	**0**	**0**	$\mathbf{3+K}$	$\leq \mathbf{2+K}$
	Insertion	0	0	$1+K$	$\leq 1+K$
	Search Uniq.	0	0	2	0
	Search Mult.	0	0	$1+M$	0
Indirection	Deletion	2	0	2	≤ 3
	Single-attr. update	**2**	**0**	**4**	$\leq \mathbf{3}$
	Insertion	$2+2K$	$2K/LB$	1	$\leq 1+2K/LB$
	Search Uniq.	2	0	2	0
	Search Mult.	$1+M$	0	$1+M$	0

Experimental Setting

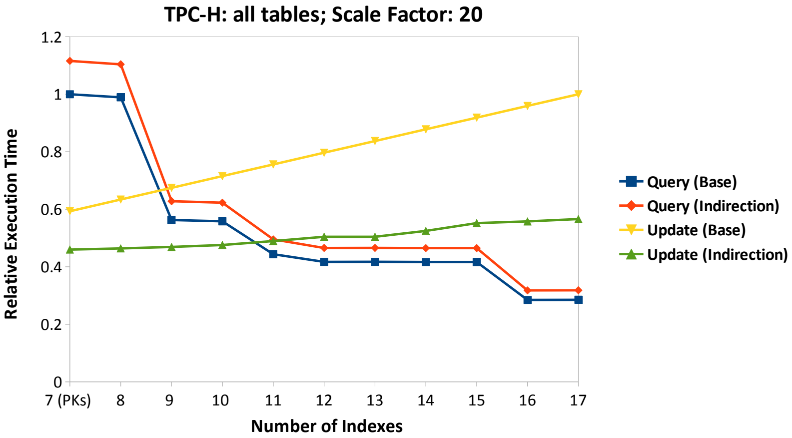
- Hardware:
  - (2 × 8-core) Intel(R) Xeon(R) CPU E7-4820 @ 2.00GHz, 32GB, 2 × HDD, SSD Fusion-io
- Software:
  - Database: IBM DB2 9.7
  - Prototyped in a commercial proprietary database
  - Prototyped in Apache Spark by UC Berkeley
  - LIBGist v.1.0: Generalized Search Tree C++ Library by UC Berkeley (**5K LOC**) (Predecessor of Generalized Search Tree (GiST) access method for PostgreSQL)
  - **LIBGist^{mv} Prototype:** Multi-version Generalized Search Tree C++ Library over LIBGist supporting Indirection/LIDBlock/DeltaBlock (**3K LOC**)
- Data:
  - TPC-H benchmark
  - Microsoft Hekaton micro benchmark

Indirection: Effect of Indexes in Operational Data Stores



Substantially improving the update time ...

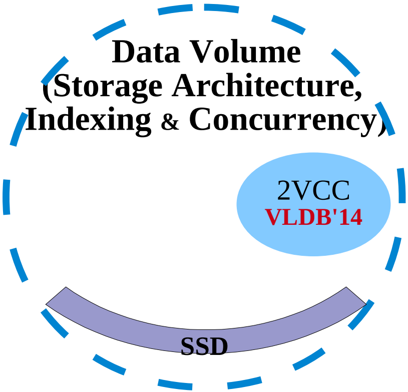
Indirection: Effect of Indexes in Operational Data Stores



... Consequently affording more indexes and significantly reducing the query time

1. Data Velocity: Index Maintenance
2. Data Volume: MVCC Concurrency
3. Data Volume: Coordination-free Concurrency
4. Combining Volume & Velocity: Lineage-based Storage Architecture
5. Decentralized & Democratic Data Platform
6. Conclusions
7. References

Introducing Multi-version Concurrency Control



Generalized Concurrency Control: Volume Dimension

Observed Trends

In operational multi-version databases, there is a tremendous opportunity to avoid clashes between readers (scanning a large volume of data) and writers (frequent updates).

Generalized Concurrency Control: Volume Dimension

Observed Trends

In operational multi-version databases, there is a tremendous opportunity to avoid clashes between readers (scanning a large volume of data) and writers (frequent updates).

Introducing a (latch-free) *two-version concurrency control (2VCC)* by extending indirection mapping (i.e., central coordination mechanism) and exploiting existing two-phase locking (2PL) in order to

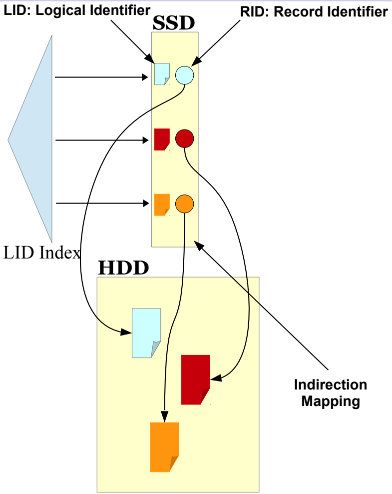
Generalized Concurrency Control: Volume Dimension

Observed Trends

In operational multi-version databases, there is a tremendous opportunity to avoid clashes between readers (scanning a large volume of data) and writers (frequent updates).

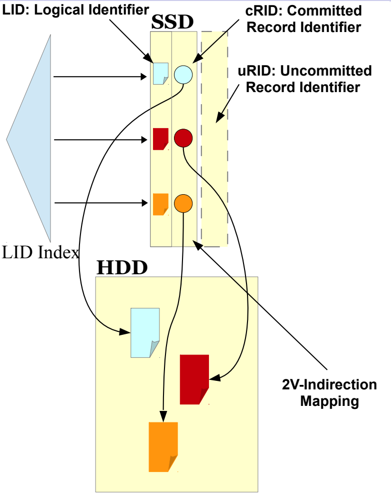
Introducing a (latch-free) *two-version concurrency control (2VCC)* by extending indirection mapping (i.e., central coordination mechanism) and exploiting existing two-phase locking (2PL) in order to
Decouple Readers/Writers to Reduce Contention
(Pessimistic and Optimistic Concurrency Control Coexistence)

2V-Indirection Indexing: Updating Records



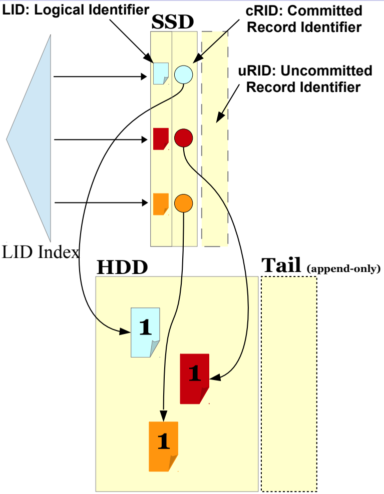
Recap: Indirection technique for reducing index maintenance

2V-Indirection Indexing: Updating Records



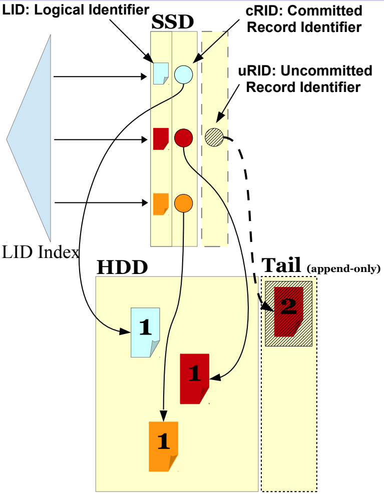
Extending the indirection to committed/uncommitted records

2V-Indirection Indexing: Updating Records



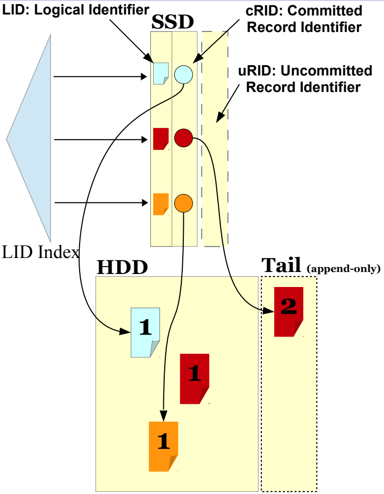
Extending the indirection to committed/uncommitted records

2V-Indirection Indexing: Updating Records



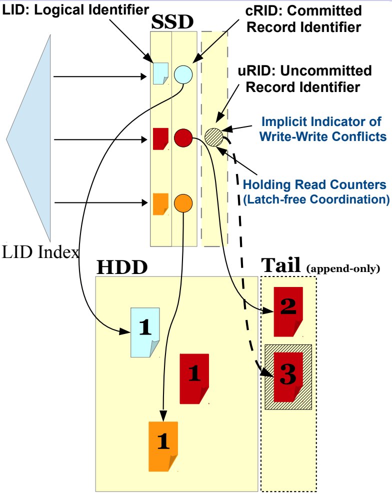
Decoupling readers/writers to eliminate contention

2V-Indirection Indexing: Updating Records



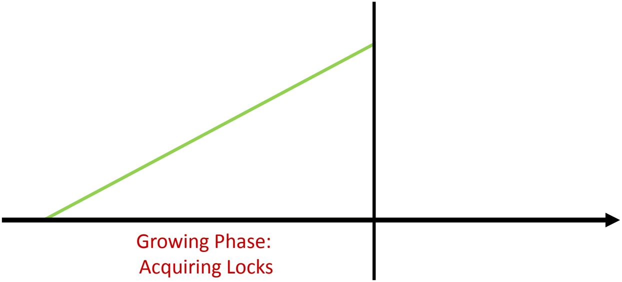
Decoupling readers/writers to eliminate contention

2V-Indirection Indexing: Updating Records



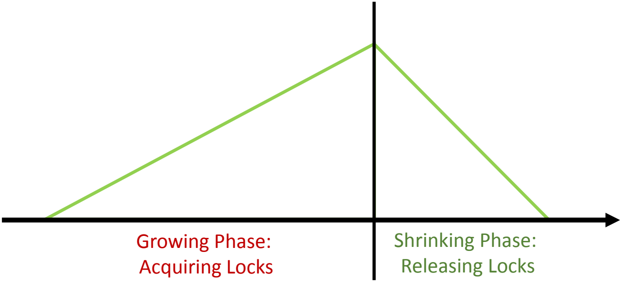
Decoupling readers/writers to eliminate contention

Overview of Two-version Concurrency Control Protocol



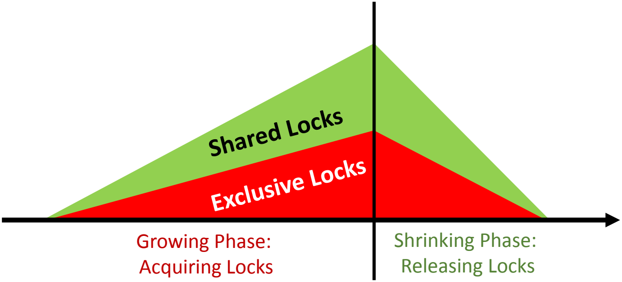
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



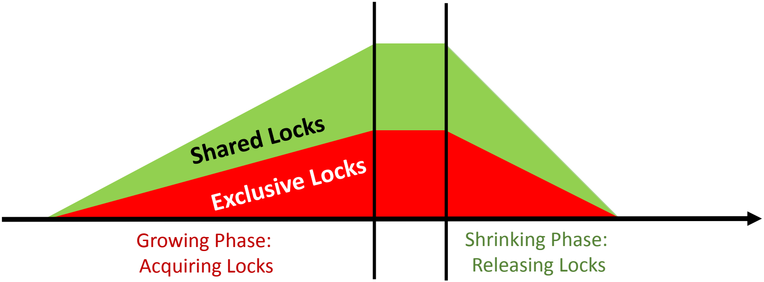
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



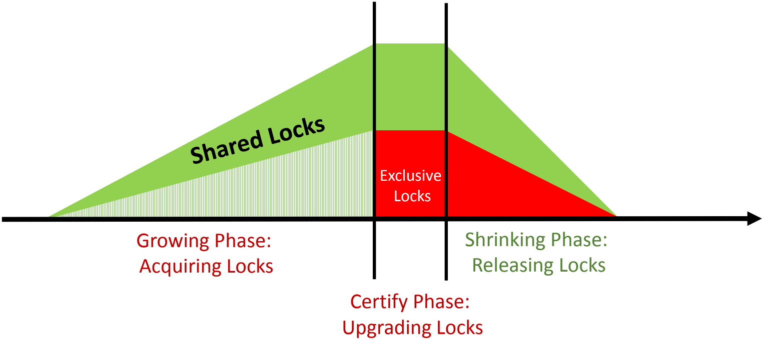
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



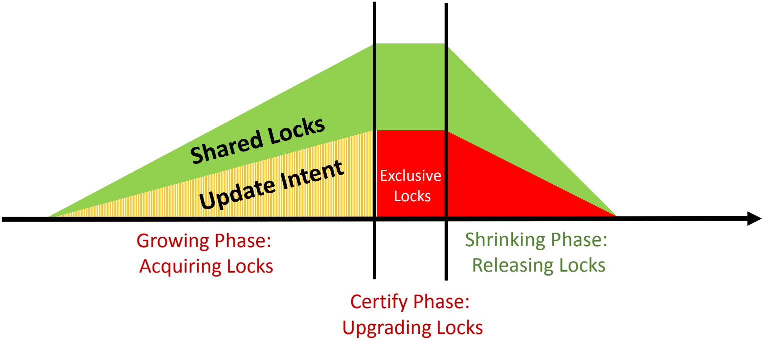
Extending 2PL with certify phase

Overview of Two-version Concurrency Control Protocol



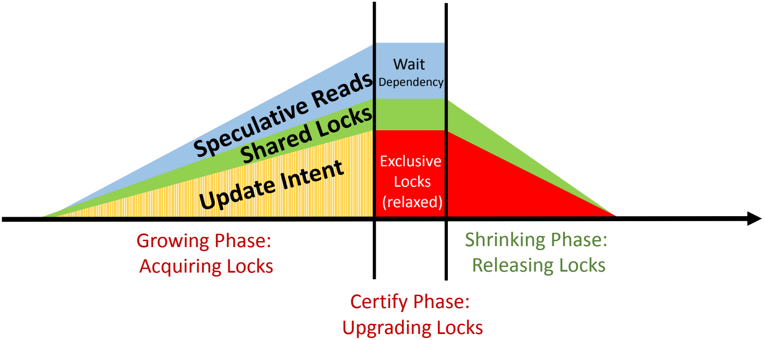
Exclusive locks held for shorter period (inherently optimistic)

Overview of Two-version Concurrency Control Protocol



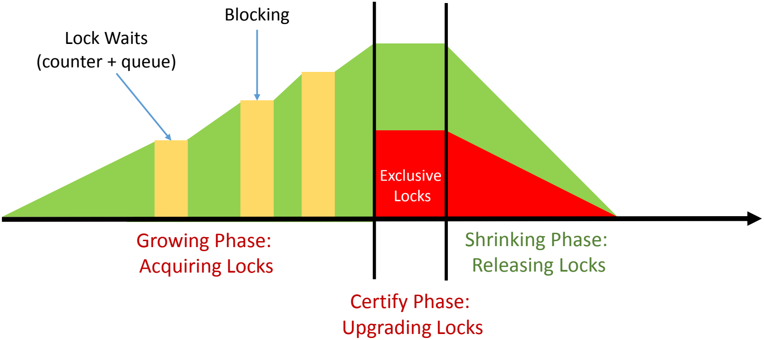
Exclusive locks held for shorter period (inherently optimistic)

Overview of Two-version Concurrency Control Protocol



Relaxed exclusive locks to allow speculative reads (increased optimism)

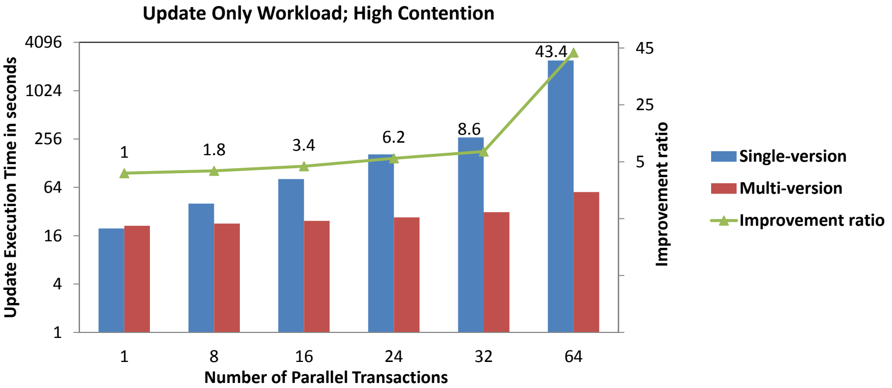
Overview of Two-version Concurrency Control Protocol



Trade-offs between blocking (i.e., locks) vs. non-blocking (i.e., read counters)

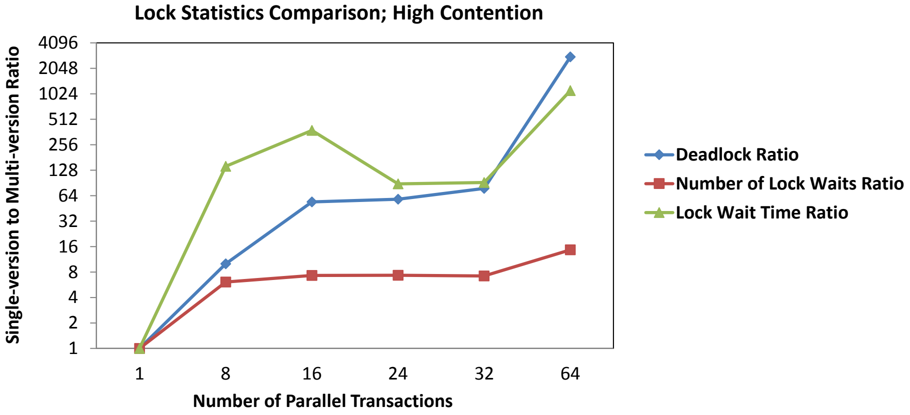
Experimental Analysis

2VCC: Effect of Parallel Update Transactions



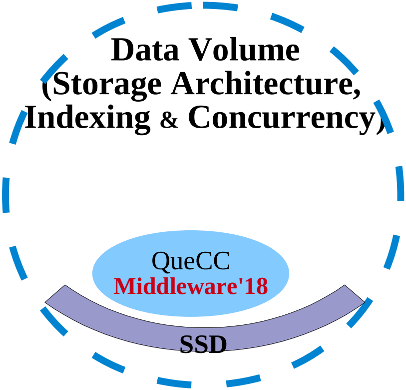
Substantial gain by reducing the read/write contention & using non-blocking operations

2VCC: Effect of Parallel Update Transactions



Substantial gain by reducing the read/write contention & using non-blocking operations

Introducing Coordination-free Concurrency Control



Confrontation-free Concurrency Control

Observed Trends

In operational databases, the use of pre-compiled stored procedures is predominant. There is a tremendous opportunity to exploit transaction prior knowledge to eliminate the need for coordination.

Confrontation-free Concurrency Control

Observed Trends

In operational databases, the use of pre-compiled stored procedures is predominant. There is a tremendous opportunity to exploit transaction prior knowledge to eliminate the need for coordination.

Is it possible to have concurrent execution over shared data (not limited to partitionable workloads) without having any concurrency controls?

Confrontation-free Concurrency Control

Observed Trends

In operational databases, the use of pre-compiled stored procedures is predominant. There is a tremendous opportunity to exploit transaction prior knowledge to eliminate the need for coordination.

Is it possible to have concurrent execution over shared data (not limited to partitionable workloads) without having any concurrency controls?

Introducing a *queue-oriented, control-free concurrency (QueCC)* based on two parallel & independent phases of priority-driven planning & execution.

Confrontation-free Concurrency Control

Observed Trends

In operational databases, the use of pre-compiled stored procedures is predominant. There is a tremendous opportunity to exploit transaction prior knowledge to eliminate the need for coordination.

Is it possible to have concurrent execution over shared data (not limited to partitionable workloads) without having any concurrency controls?

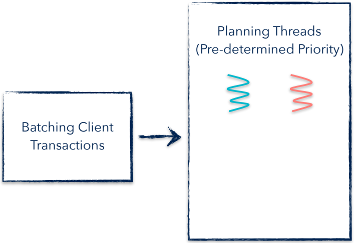
Introducing a *queue-oriented, control-free concurrency (QueCC)* based on two parallel & independent phases of priority-driven planning & execution.
Execution and Synchronization Decoupling

Queue-oriented, Control-free Concurrency (QueCC)

Batching Client Transactions

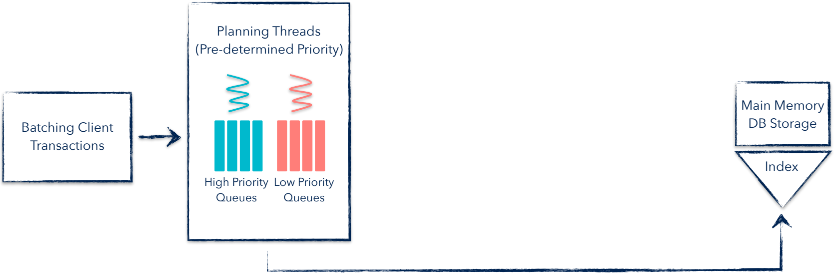
Execution & Synchronization Decoupling: Deterministic priority-based planning followed by queue-oriented execution

Queue-oriented, Control-free Concurrency (QueCC)



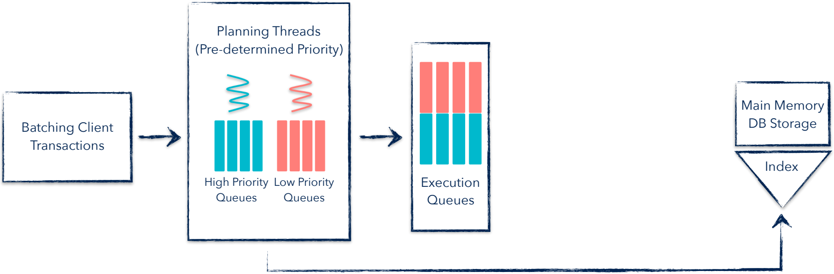
Execution & Synchronization Decoupling: Deterministic priority-based planning followed by queue-oriented execution

Queue-oriented, Control-free Concurrency (QueCC)



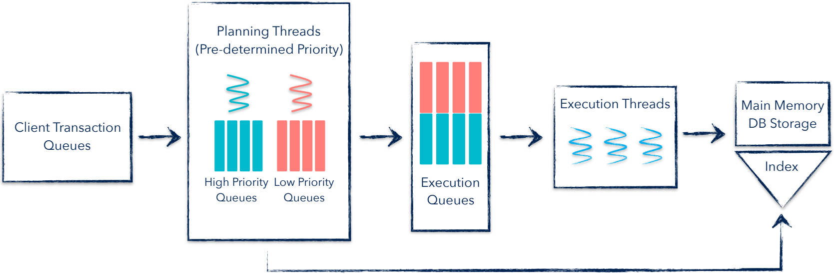
Execution & Synchronization Decoupling: Deterministic priority-based planning followed by queue-oriented execution

Queue-oriented, Control-free Concurrency (QueCC)



Execution & Synchronization Decoupling: Deterministic priority-based planning followed by queue-oriented execution

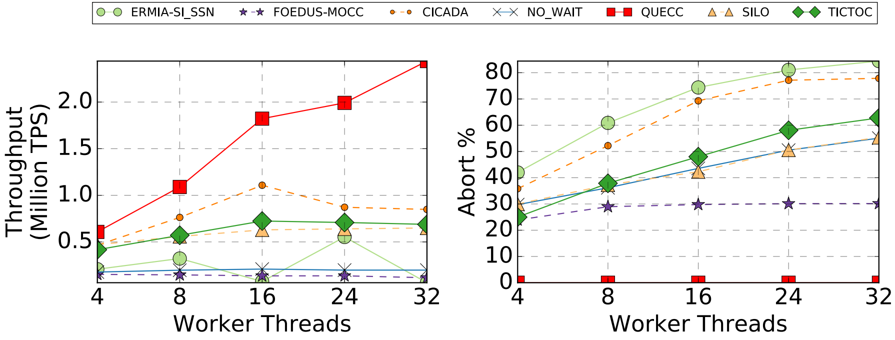
Queue-oriented, Control-free Concurrency (QueCC)



Execution & Synchronization Decoupling: Deterministic priority-based planning followed by queue-oriented execution

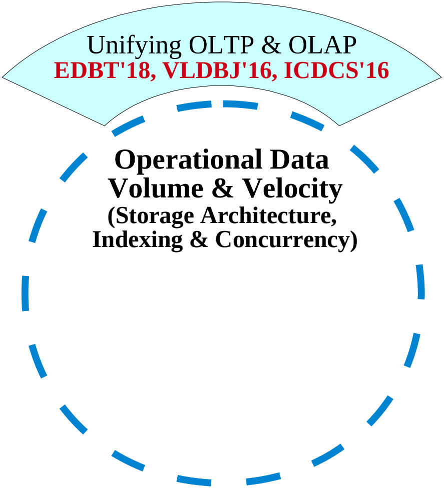
Experimental Analysis

QueCC: Effect of Parallel Update Transactions



Avoiding thread coordination & eliminating all execution-induced aborts

Unifying OLTP and OLAP



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

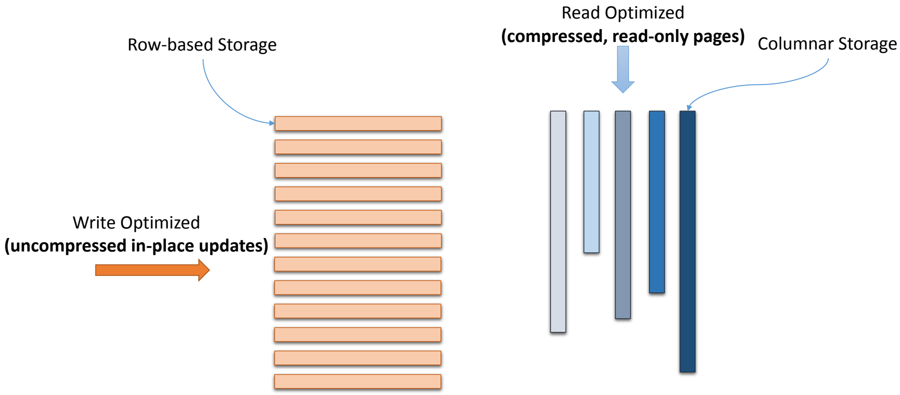
Observed Trends

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

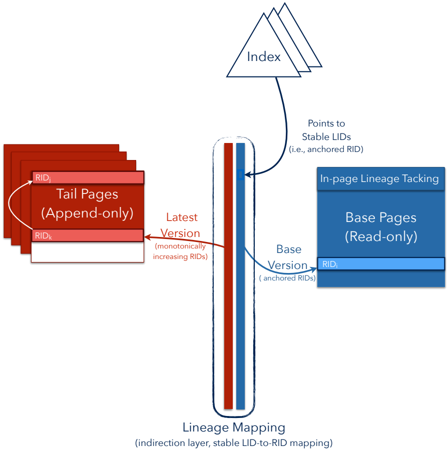
lazily and independently stage stable data from a write-optimized layout (i.e., OLTP) into a read-optimized layout (i.e., OLAP)

Storage Layout Conflict



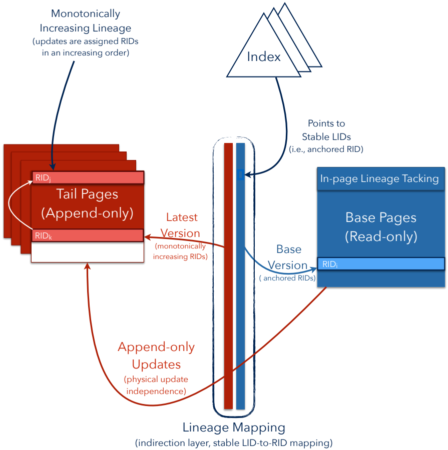
Write-optimized (i.e., uncompressed & row-based) vs. read-optimized (i.e., compressed & column-based) layouts

Lineage-based Storage Architecture (LSA): Intuition



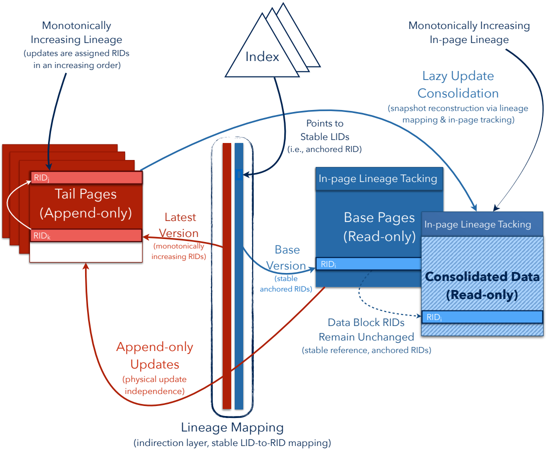
Physical Update Independence: De-coupling data & its updates
(reconstruction via in-page lineage tracking and lineage mapping)

Lineage-based Storage Architecture (LSA): Intuition



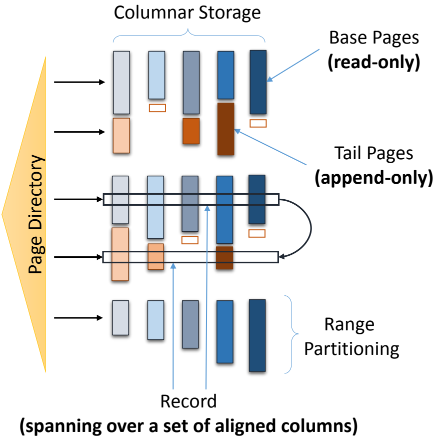
Physical Update Independence: De-coupling data & its updates
(reconstruction via in-page lineage tracking and lineage mapping)

Lineage-based Storage Architecture (LSA): Intuition



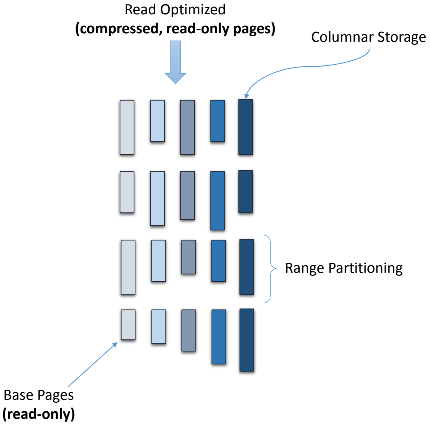
Physical Update Independence: De-coupling data & its updates (reconstruction via in-page lineage tracking and lineage mapping)

Lineage-based Storage Architecture (LSA): Overview



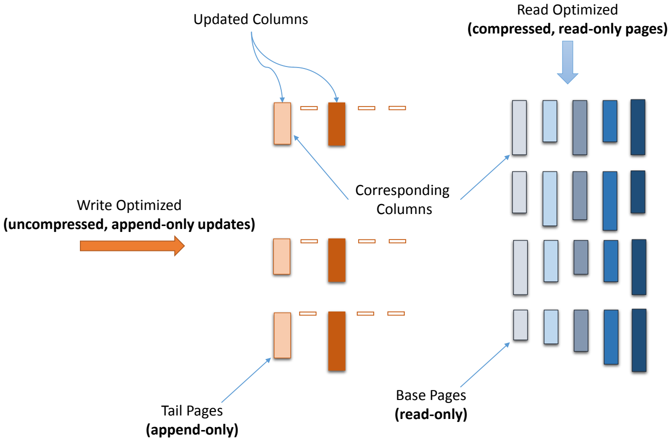
Overview of the lineage-based storage architecture
(**base pages** and **tail pages** are handled identically at the storage layer)

L-Store: Detailed Design



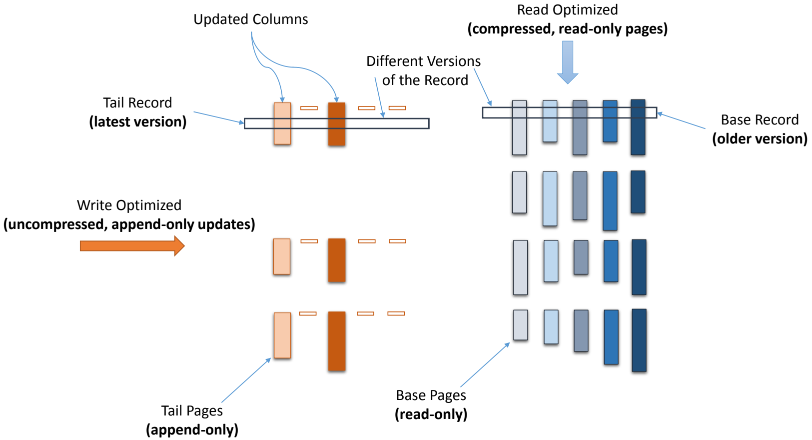
Records are range-partitioned and compressed into a set of ready-only **base pages** (accelerating analytical queries)

L-Store: Detailed Design



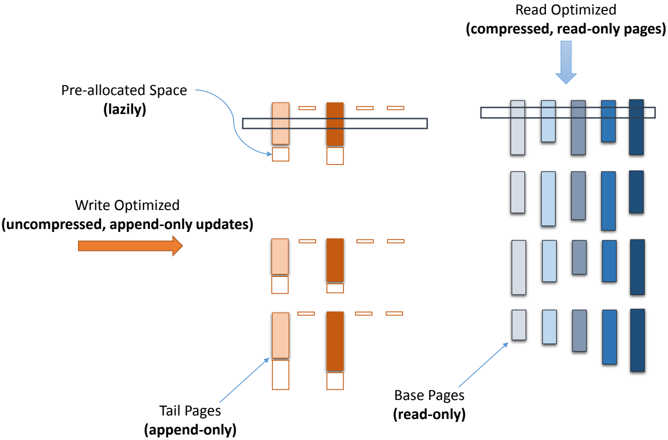
Recent updates for a range of records are clustered in their **tails pages** (transforming costly point updates into an amortized analytical-like query)

L-Store: Detailed Design



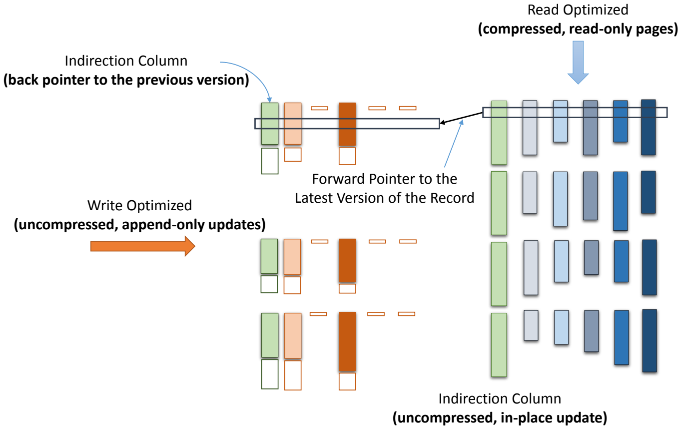
Recent updates for a range of records are clustered in their **tails pages** (transforming costly point updates into an amortized analytical-like query)

L-Store: Detailed Design



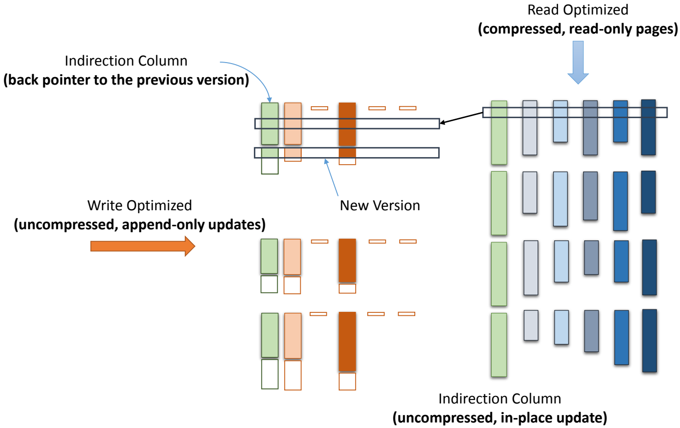
Recent updates are strictly appended, uncompressed in the pre-allocated space (eliminating the read/write contention)

L-Store: Detailed Design



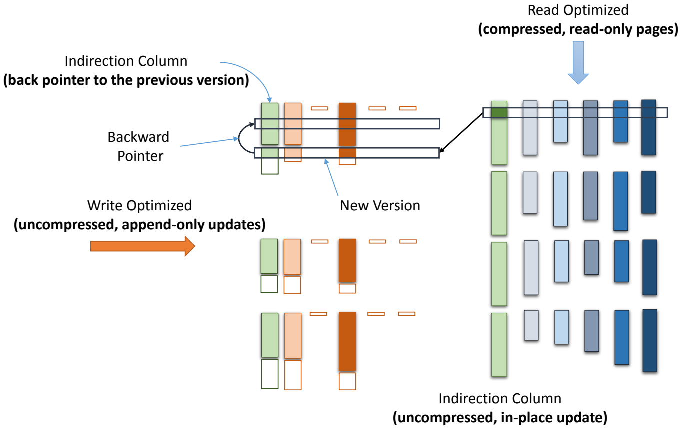
Achieving (at most) 2-hop access to the latest version of any record
(avoiding read performance deterioration for point queries)

L-Store: Detailed Design



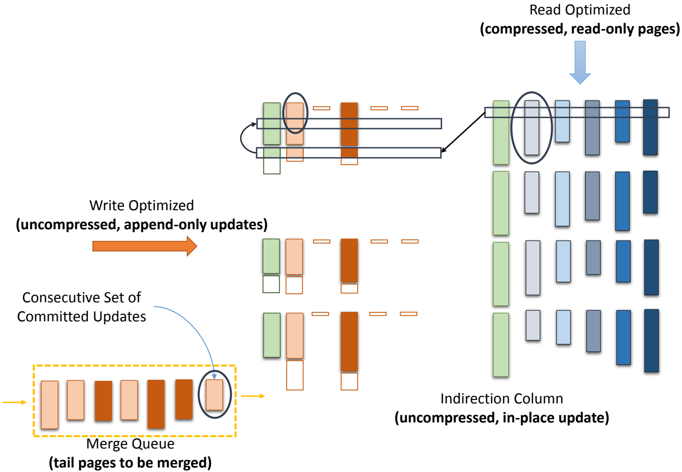
Achieving (at most) 2-hop access to the latest version of any record (avoiding read performance deterioration for point queries)

L-Store: Detailed Design



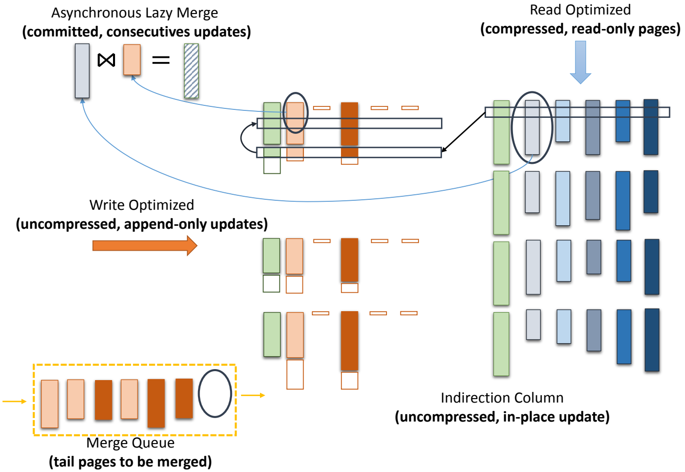
Achieving (at most) 2-hop access to the latest version of any record (avoiding read performance deterioration for point queries)

L-Store: Contention-free Merge



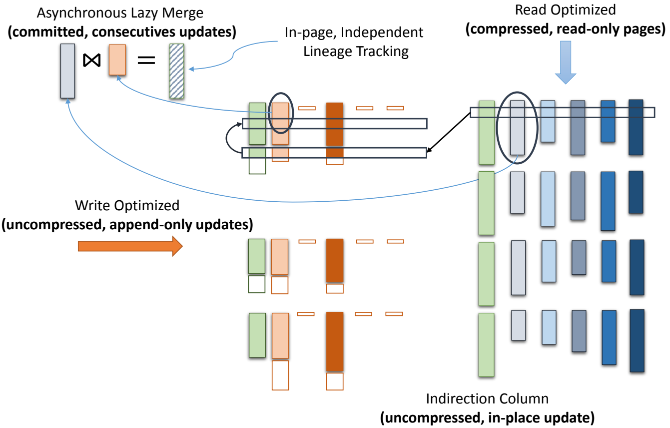
Contention-free merging of only stable data: read-only and committed data (no need to block on-going and new transactions)

L-Store: Contention-free Merge



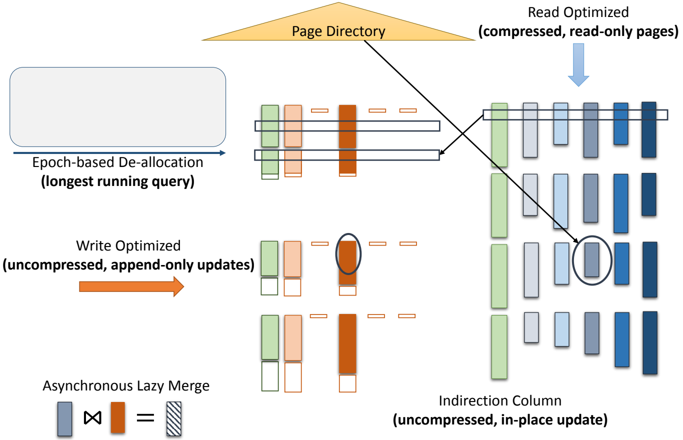
Lazy independent merging of **base pages** with their corresponding **tail pages** (resembling a local left outer-join of the base and tail pages)

L-Store: Contention-free Merge



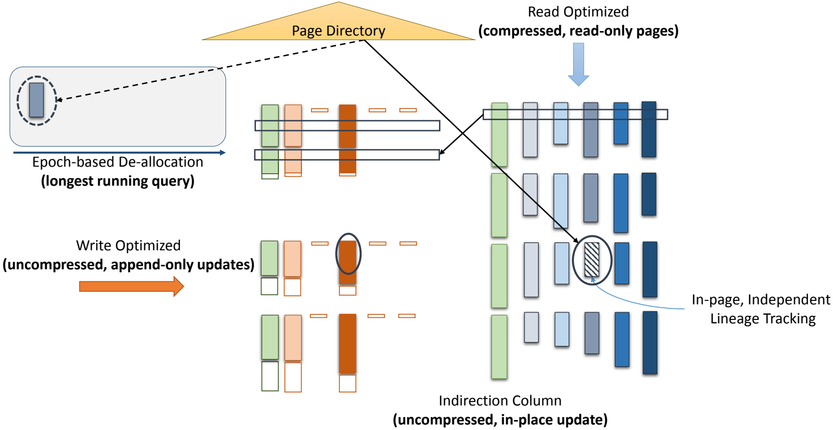
Independently tracking the lineage information within every page
(no need to coordinate merges among different columns of the same records)

L-Store: Epoch-based Contention-free De-allocation



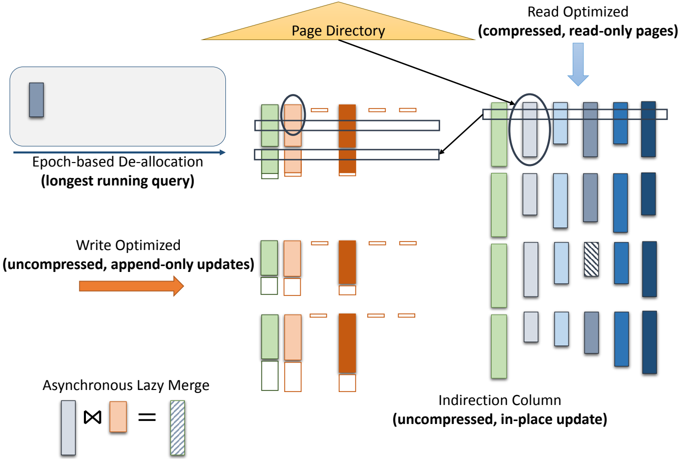
Contention-free page de-allocation using an epoch-based approach (no need to drain the ongoing transactions)

L-Store: Epoch-based Contention-free De-allocation



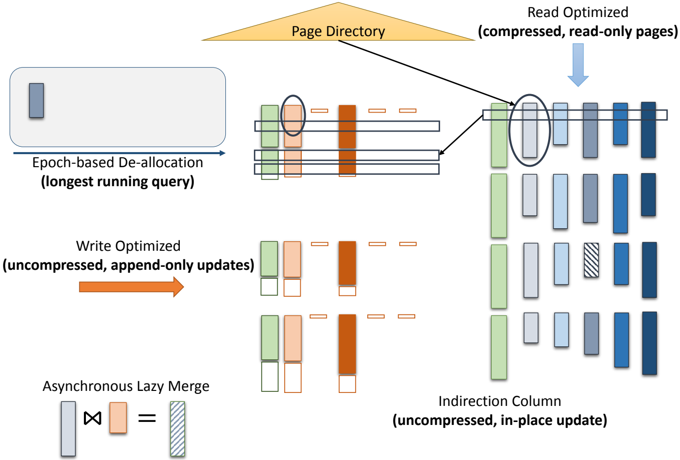
Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)

L-Store: Epoch-based Contention-free De-allocation



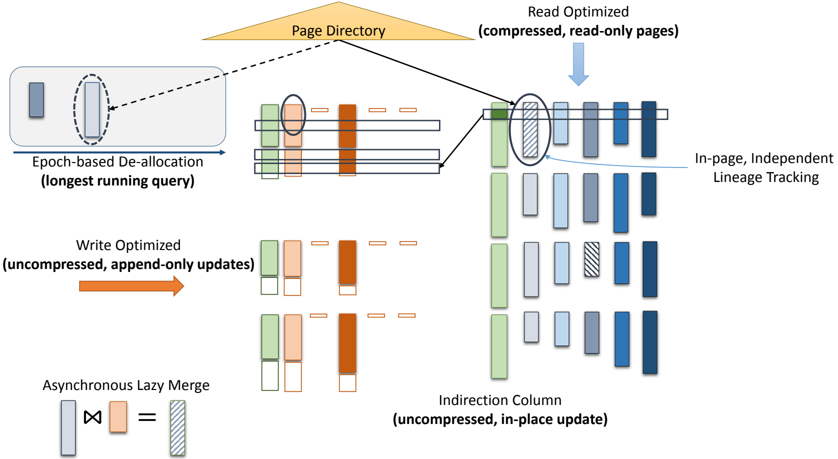
Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)

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



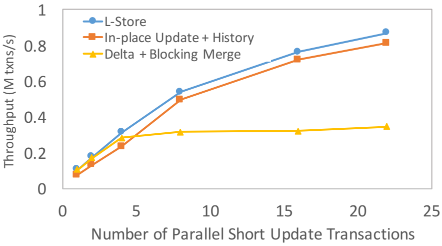
Contention-free page de-allocation using an epoch-based approach
(no need to drain the ongoing transactions)

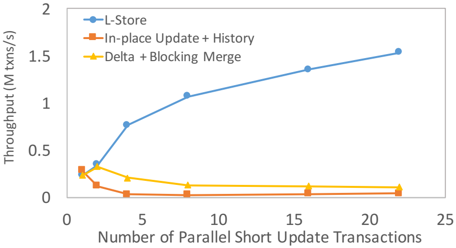
Experimental Analysis

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*)

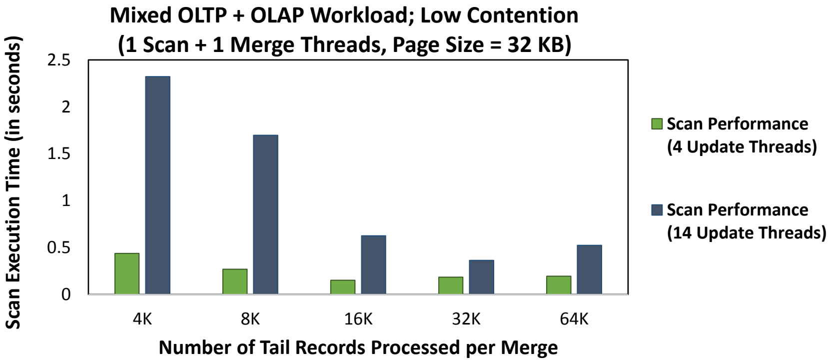
Effect of Varying Contention Levels





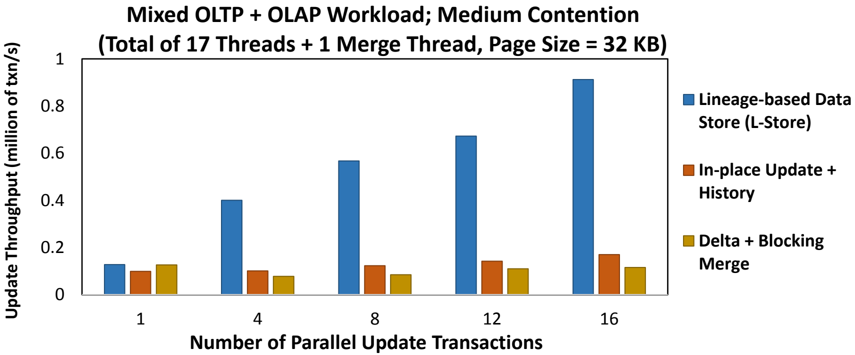
Achieving up to **40**× as increasing the update contention

Effect of Merge Frequency on Scan Performance



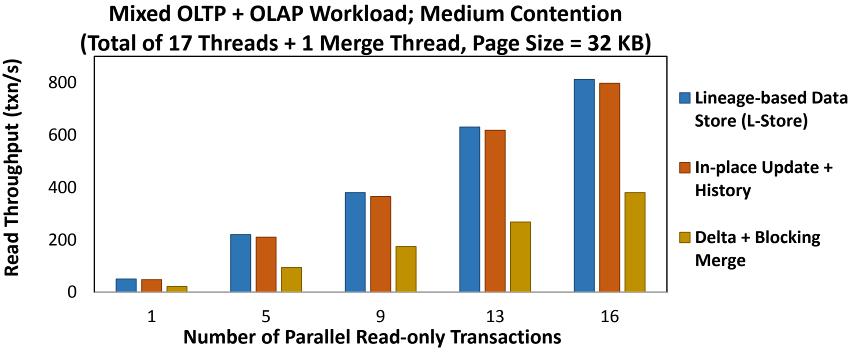
Merge process is essential in maintaining efficient scan performance

Effect of Mixed Workloads: Update Performance



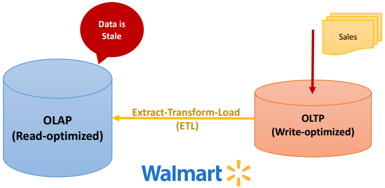
Eliminating latching & locking results in a substantial performance improvement

Effect of Mixed Workloads: Read Performance



Coping with tens of update threads with a single merge thread

Recap: Data Management Challenges at Microscale

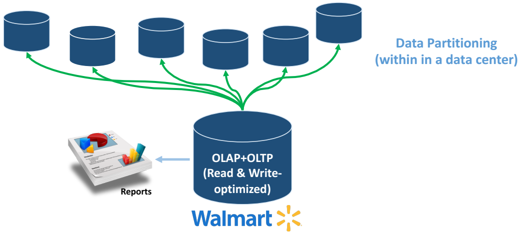


OLTP and OLAP data are isolated at microscale

Recap: Data Management Challenges at Microscale

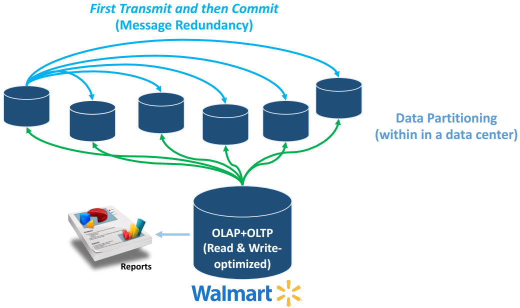
First step is to unify OLTP and OLAP

Platform Scaling: Data Partitioning



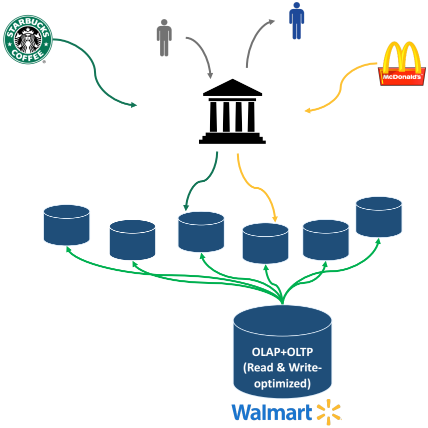
Moving towards distributed environment

Platform Scaling: Non-blocking Agreement Protocols



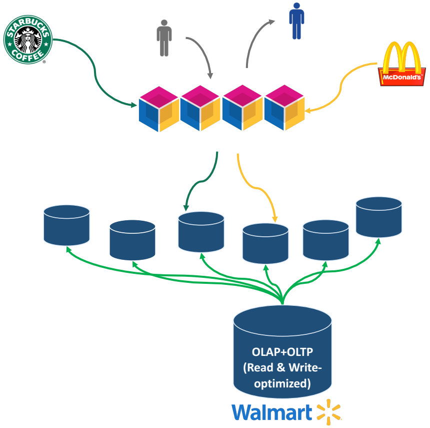
Message redundancy vs. latency trade-offs [EasyCommit, EDBT'18]

Central Control: Data Gate Keeper



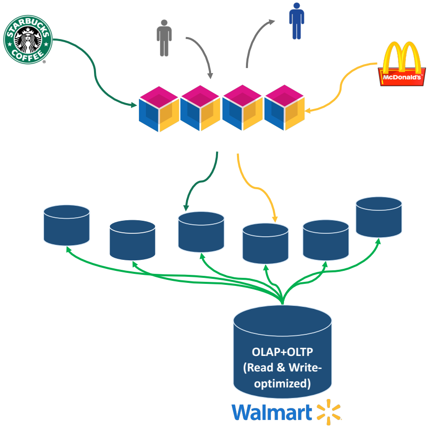
Conform to trusting the central authority and governance

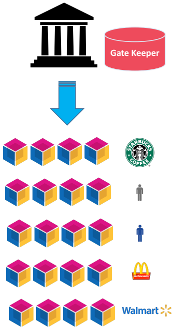
Decentralized Control: Removing Data Barrier



Seek trust in *decentralized* and *democratic* governance [PoE (under submission)]

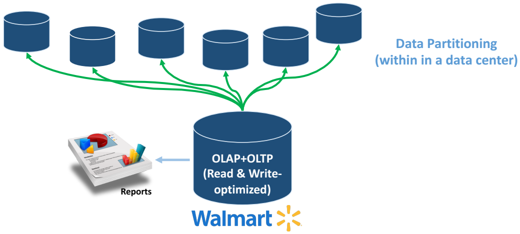
Democratic Control: Removing Trust Barrier





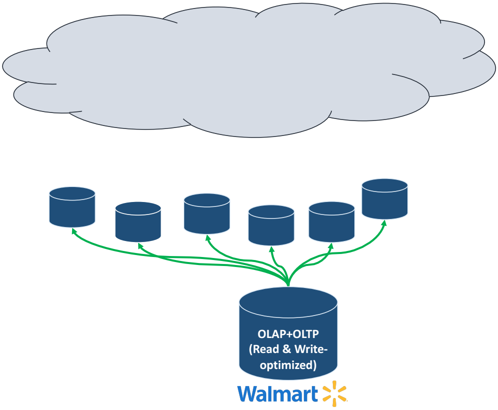
Seek trust in *decentralized* and *democratic* governance [PoE (under submission)]

Global-scale Reliable Platform over Unreliable Hardware



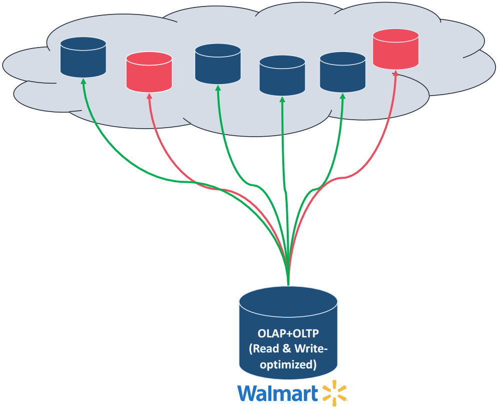
Self-managed infrastructure

Global-scale Reliable Platform over Unreliable Hardware



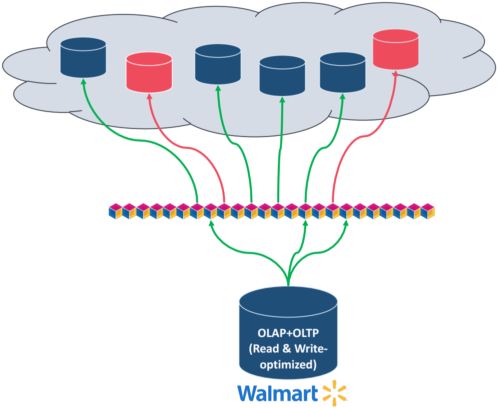
Cloud-managed infrastructure (trust the provider)

Global-scale Reliable Platform over Unreliable Hardware



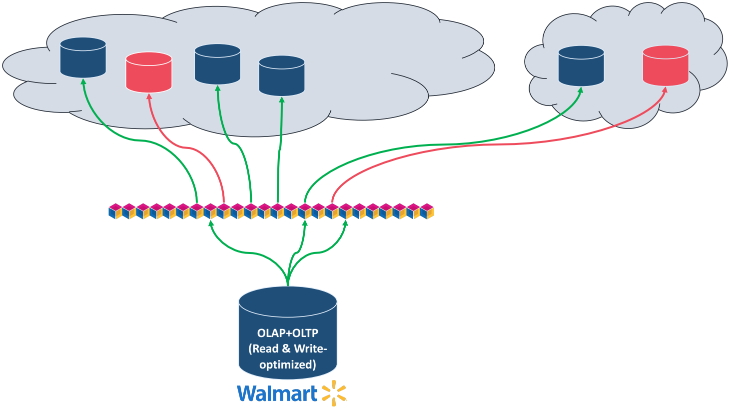
Cloud-managed infrastructure (trust the provider)

Global-scale Reliable Platform over Unreliable Hardware



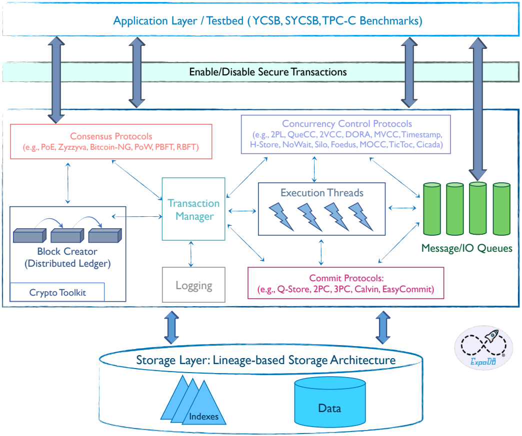
Light-weight, fault-tolerant, trusted middleware [Blockplane, (under submission)]

Global-scale Reliable Platform over Unreliable Hardware



Fault-tolerant protocols vs. consistency models [MultiBFT, GeoBFT (under submission)]

ExpoDB: Exploratory Data Platform Architecture



A decentralized & democratic platform to unify OLTP and OLAP

Contributions & Outlook

ExpoDB: Decentralized & Democratic Platform

- **Decentralized & Democratic Control**: PoE, MultiBFT, GeoBFT [under submission]
- **Reliability over Unreliable Hardware**: Blockplane [under submission]

Operational Data Stores: Velocity & Volume

- **Index Maintenance:** Indirection Technique **[VLDB'13, VLDBJ'16]**
- **Concurrency Control:** 2VCC Technique **[VLDB'14, Middleware'16],** EasyCommit **[EDBT'18],** QueCC **[Middleware'18]**
- **Hybrid Storage:** Enhancing Key-Value Store **[VLDB'12, ICDE'14]**
- **Real-time OLTP+OLAP:** Lineage-based Data Store (*L-Store*) **[EDBT-18,ICDCS'16, 30+ Patents]**

Stream Processing: Velocity

- **High-dimensional Indexing**: BE-Tree **[SIGMOD'11, TODS'13]**, Compressed Stream Processing **[ICDE'14]**
- **(Distributed) Top-*k* Indexing**: BE*-Tree **[ICDE'12, ICDCS'13, Middleware'17, ICDCS'17]**
- **Hardware Acceleration**: FPGAs **[VLDB'10, ICDE'12, VLDB'13, ICDE'15, SIGMOD Record'15, ICDE'16, USENIX ATC'16, ICDCS'17, ICDE'18]**
- **Novel Mappings**: XML/XPath **[EDBT'11]**, Distributed Workflow **[TDKE'15, SIGMOD'15, ICDE'16, Middleware'16]**

Questions?

Thank you!

Exploratory Systems Lab (ExpoLab)
Website: https://msadoghi.github.io/

ExpoLab
Creativity Unfolded

ExpoDB

Related Publications (Patents Omitted)

I. Abdelaziz, A. Fokoue, O. Hassanzadeh, P. Zhang, M. Sadoghi.
Large-scale structural and textual similarity-based mining of knowledge graph to predict drug-drug interactions.
In *J. Web Sem.* pages 104–117, 2017.

A. Fokoue, O. Hassanzadeh, **M. Sadoghi**, and P. Zhang.
Predicting drug-drug interactions through similarity-based link prediction over web data.
In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 175–178, 2016.

A. Fokoue, **M. Sadoghi**, O. Hassanzadeh, and P. Zhang.
Predicting drug-drug interactions through large-scale similarity-based link prediction.
In *The Semantic Web. Latest Advances and New Domains - 13th International Conference, ESWC 2016, Heraklion, Crete, Greece, May 29 - June 2, 2016, Proceedings*, pages 774–789, 2016.

Rajesh R. Bordawekar and **M. Sadoghi**.
Accelerating database workloads by software-hardware-system co-design.
In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1428–1431, 2016.

A. Chandel, O. Hassanzadeh, N. Koudas, **M. Sadoghi**, and D. Srivastava.
Benchmarking declarative approximate selection predicates.
In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD 2007, pages 353–364, New York, NY, USA, 2007. ACM.

A. Farroukh, **M. Sadoghi**, and H.-A. Jacobsen.
Towards vulnerability-based intrusion detection with event processing.
In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS 2011, pages 171–182, New York, New York, USA, 2011. ACM.

O. Hassanzadeh, **M. Sadoghi**, and R. J. Miller.
Accuracy of approximate string joins using grams.
In V. Ganti and F. Naumann, editors, *Proceedings of the Fifth International Workshop on Quality in Databases, QDB 2007 at VLDB 2007 conference, Vienna, Austria, September 23, 2007*, pages 11–18, 2007.

M. Hemmatpour, B. Montrucchio, M. Rebaudengo, and **M. Sadoghi**.
Kanzi: A distributed, in-memory key-value store.
In *Proceedings of the 17th Annual Middleware Conference, Trento, Italy, December 1216, 2016*, 2016.

M. Jergler, **M. Sadoghi**, and H.-A. Jacobsen.
D2WORM: A management infrastructure for distributed data-centric workflows.
In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 1427–1432, 2015.

M. Jergler, **M. Sadoghi**, and H.-A. Jacobsen.
Geo-distribution of flexible business processes over publish/subscribe paradigm.
In *Proceedings of the 17th Annual Middleware Conference, Trento, Italy, December 1216, 2016*, 2016.

P. Menon, T. Rabl, **M. Sadoghi**, and H. Jacobsen.
CaSSanDra: An SSD boosted key-value store.
In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 1162–1167, 2014.

P. Menon, T. Rabl, **M. Sadoghi**, and H. Jacobsen.
Optimizing key-value stores for hybrid storage architectures.
In *the 24th International Conference of Center for Advanced Studies on Collaborative Research, IBM CASCON 2014. Toronto, Canada, November 3-5, 2014*. 2014.

N. Mohammadreza, **M. Sadoghi**, and H.-A. Jacobsen.
The FQP vision: Flexible query processing on a reconfigurable computing fabric.
SIGMOD Record - Special Issue on Visionary Ideas in Data Management , 44(2), pages 5–10, 2015.

V. Muthusamy, Y. Yoon, **M. Sadoghi**, and H.-A. Jacobsen.
eQoSystem: supporting fluid distributed service-oriented workflows.
In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS 2011, pages 381–382, New York, New York, USA, 2011. ACM.

M. Najafi, **M. Sadoghi**, and H. Jacobsen.
Flexible query processor on FPGAs.
PVLDB 2013, 6(12):1310–1313, 2013.

M. Najafi, **M. Sadoghi**, and H. Jacobsen.
Configurable hardware-based streaming architecture using online programmable-blocks.
In *the 31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 819–830, 2015.

M. Najafi, **M. Sadoghi**, and H. Jacobsen.
The FQP vision: Flexible query processing on a reconfigurable computing fabric.
SIGMOD Record, 44(2):5–10, 2015.

M. Najafi, **M. Sadoghi**, and H. Jacobsen.
Splitjoin: A scalable, low-latency stream join architecture with adjustable ordering precision.
In *2016 USENIX Annual Technical Conference, USENIX ATC 2016, Denver, CO, USA, June 22-24, 2016.*, pages 493–505, 2016.

T. Rabl, **M. Sadoghi**, S. Gómez-Villamor, V. Muntés-Mulero, H.-A. Jacobsen, and S. Mankovskii.
Solving big data challenges for enterprise application performance management.
Proceedings of the VLDB Endowment, PVLDB 2012, 5(12):1724–1735, Aug. 2012.

T. Rabl, **M. Sadoghi**, K. Zhang, and H. Jacobsen.
Poster: MADES - a multi-layered, adaptive, distributed event store.
In S. Chakravarthy, S. D. Urban, P. Pietzuch, and E. A. Rundensteiner, editors, *The 7th ACM International Conference on Distributed Event-Based Systems, DEBS 2013, Arlington, TX, USA - June 29 - July 03, 2013*, pages 343–344. ACM, 2013.

T. Rabl, K. Zhang, **M. Sadoghi**, N. K. Pandey, A. Nigam, C. Wang, and H.-A. Jacobsen.
Solving manufacturing equipment monitoring through efficient complex event processing: Debs grand challenge.
In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, DEBS 2012, pages 335–340, Berlin, Germany, 2012. ACM.

M. Sadoghi.
Towards an extensible efficient event processing kernel.
In *Proceedings of the SIGMOD/PODS 2012 PhD Symposium*, PhD 2012, pages 3–8, Scottsdale, Arizona, USA, 2012. ACM.

M. Sadoghi, I. Burcea, and H.-A. Jacobsen.
GPX-Matcher: a generic Boolean predicate-based XPath expression matcher.
In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT 2011, pages 45–56, Uppsala, Sweden, 2011. ACM.

M. Sadoghi, M. Canim, B. Bhattacharjee, F. Nagel, and K. A. Ross.
Reducing database locking contention through multi-version concurrency.
PVLDB 2014, 7(13):1331–1342, 2014.

M. Sadoghi and H. Jacobsen.
Adaptive parallel compressed event matching.
In *IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 - April 4, 2014*, pages 364–375, 2014.

M. Sadoghi and H.-A. Jacobsen.
BE-Tree: an index structure to efficiently match Boolean expressions over high-dimensional discrete space.
In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, SIGMOD 2011, pages 637–648, Athens, Greece, 2011. ACM.

M. Sadoghi and H.-A. Jacobsen.
Location-based matching in publish/subscribe revisited.
In *Proceedings of the Posters and Demo Track*, Middleware 2012, pages 9:1–9:2, Montreal, Quebec, Canada, 2012. ACM.

M. Sadoghi and H.-A. Jacobsen.
Relevance matters: Capitalizing on less (top-k matching in publish/subscribe).
In *IEEE 28th International Conference on Data Engineering*, ICDE 2012, pages 786–797, Arlington, Virginia, USA, 2012. IEEE Computer Society.

M. Sadoghi and H.-A. Jacobsen.
Analysis and optimization for Boolean expression indexing.
ACM Trans. Database Syst., TODS 2013, 38(2):8:1–8:47, July 2013.

M. Sadoghi, H.-A. Jacobsen, M. Labrecque, W. Shum, and H. Singh.
Efficient event processing through reconfigurable hardware for algorithmic trading.
Proceedings of the VLDB Endowment, PVLDB 2010, 3(2):1525–1528, 2010.

M. Sadoghi, R. Javed, N. Tarafdar, H. Singh, R. Palaniappan, and H.-A. Jacobsen.
Multi-query stream processing on fpgas.
In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE 2012, pages 1229–1232, Washington, DC, USA, 2012. IEEE Computer Society.

M. Sadoghi, M. Jergler, H.-A. Jacobsen, R. Hull, and R. Vaculin.
Safe distribution and parallel execution of data-centric workflows over the publish/subscribe abstraction.
IEEE Trans. on Knowl. and Data Eng., TKDE 2015, 2015.

M. Sadoghi, M. Jergler, H.-A. Jacobsen, R. Hull, and R. Vaculin.
Safe distribution and parallel execution of data-centric workflows over the publish/subscribe abstraction.
In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 1498–1499, 2016.

M. Sadoghi and N. Koudas.
SPIDER: Data Quality & Data Cleaning Project.
In the 15th International Conference of Center for Advanced Studies on Collaborative Research, IBM CASCON 2005. October 17-20, 2005. 2005.

M. Sadoghi, K. A. Ross, M. Canim, and B. Bhattacharjee.
Making updates disk-I/O friendly using SSDs.
PVLDB 2013, 6(11):997–1008, 2013.

M. Sadoghi, K. A. Ross, M. Canim, and B. Bhattacharjee.
Exploiting ssds in operational multiversion databases.
VLDB J., 25(5):651–672, 2016.

M. Sadoghi, S. Bhattacherjee, B. Bhattacharjee, and M. Canim.
L-Store: A real-time OLTP and OLAP system.
CoRR, abs/1601.04084, 2016.

M. Sadoghi, H. Singh, and H.-A. Jacobsen.
fpga-ToPSS: line-speed event processing on FPGAs.
In *Proceedings of the 5th ACM international conference on Distributed event-based system*, DEBS 2011, pages 373–374, New York, New York, USA, 2011. ACM.

M. Sadoghi, H. Singh, and H.-A. Jacobsen.
Towards highly parallel event processing through reconfigurable hardware.
In *Proceedings of the Seventh International Workshop on Data Management on New Hardware*, DaMoN 2011 at SIGMOD, pages 27–32, Athens, Greece, 2011. ACM.

M. Sadoghi.
ExpoDB: An Exploratory Data Science Platform.
In *Proceedings of the 8th Biennial Conference on Innovative Data Systems Research, CIDR 2017, Chaminade, CA, USA, January 8-11, 2017.*

M. Sadoghi, K. Srinivas, O. Hassanzadeh, Y-C. Chang, M. Canim, A. Fokoue, and Y A. Feldman.
Self-curating databases.
In *Proceedings of the 19th International Conference on Extending Database Technology, EDBT'16, Bordeaux, France, March 15-16, 2016.*, pages 467–472, 2016.

T. Nguyen, M. Rodriguez-Muro, O. Hassanzadeh, A. Massimiliano Gliozzo, **M. Sadoghi**.
Joint Learning of Local and Global Features for Entity Linking via Neural Networks.
In *Proceedings of the 26th International Conference on Computational Linguistics, COLING'16, Osaka, Japan, December 11-16, 2016*, pages 2310–2320, 2016.

K. Zhang, **M. Sadoghi**, V. Muthusamy, and H.-A. Jacobsen.
Distributed ranked data dissemination in social networks.
In *Proceedings of 33rd International Conference on Distributed Computing Systems*, ICDCS 2013, Washington, DC, USA, 2013. IEEE Computer Society.

K. Zhang, **M. Sadoghi**, and H.-A. Jacobsen.
DL-store: A distributed hybrid OLTP and OLAP data processing engine.
In *36th IEEE International Conference on Distributed Computing Systems, ICDCS 2016, Nara, Japan, June 27-30, 2016*, pages 769–770, 2016.

M. Najafi, K. Zhang, H.-A. Jacobsen, **M. Sadoghi**.
Hardware Acceleration Landscape for Distributed Real-time Analytics: Virtues and Limitations.
In *IEEE 37^{th} International Conference on Distributed Computing Systems (ICDCS 2017), Atlanta, GA, USA, 5-8 June, 2017*, pages 1938–1948, 2017.

K. Zhang, V. Muthusamy, **M. Sadoghi**, H.-A. Jacobsen.
Efficient Covering for Top-k Filtering in Content-Based Publish/Subscribe Systems.
In *IEEE 37^{th} International Conference on Distributed Computing Systems (ICDCS 2017), Atlanta, GA, USA, 5-8 June, 2017*, pages 2039–2044, 2017.

Ehab Abdelhamid, Mustafa Canim, Mohammad Sadoghi, Bishwaranjan Bhattacharjee, Yuan-Chi Chang, and Panos Kalnis.
Incremental frequent subgraph mining on large evolving graphs.
IEEE Trans. Knowl. Data Eng., 29(12):2710–2723, 2017.

Ehab Abdelhamid, Mustafa Canim, Mohammad Sadoghi, Bishwaranjan Bhattacharjee, Yuan-Chi Chang, and Panos Kalnis.
Incremental frequent subgraph mining on large evolving graphs.
2018.

Gonzalo I. Diaz, Achille Fokoue, and Mohammad Sadoghi.
Embeds: Scalable, ontology-aware graph embeddings.
In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 433–436, 2018.

Jana Giceva and Mohammad Sadoghi.
Hybrid OLTP and OLAP.
In Sherif Sakr and Albert Zomaya, editors, *Encyclopedia of Big Data Technologies*, pages 1–11. Springer International Publishing, Cham, 2018.

Suyash Gupta and Mohammad Sadoghi.
Blockchain Transaction Processing.
In Sherif Sakr and Albert Zomaya, editors, *Encyclopedia of Big Data Technologies*, pages 1–11. Springer International Publishing, Cham, 2018.

Suyash Gupta and Mohammad Sadoghi.
Easycommit: A non-blocking two-phase commit protocol.
In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 157–168, 2018.

Mohamed S. Hassan, Tatiana Kuznetsova, Hyun Chai Jeong, Walid G. Aref, and Mohammad Sadoghi.
Extending in-memory relational database engines with native graph support.
In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 25–36, 2018.

Mohamed S. Hassan, Tatiana Kuznetsova, Hyun Chai Jeong, Walid G. Aref, and Mohammad Sadoghi.
Grfusion: Graphs as first-class citizens in main-memory relational database systems.
In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 1789–1792, 2018.

Mohammedreza Najafi, Mohammad Sadoghi, and Hans-Arno Jacobsen.
A scalable circular pipeline design for multi-way stream joins in hardware.
In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, 2018.

Mohammad Sadoghi, Souvik Bhattacherjee, Bishwaranjan Bhattacharjee, and Mustafa Canim.
L-store: A real-time OLTP and OLAP system.
In *Proceedings of the 21th International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018.*, pages 540–551, 2018.

Kaiwen Zhang, Mohammad Sadoghi, Vinod Muthusamy, and Hans-Arno Jacobsen.
Efficient covering for top-k filtering in content-based publish/subscribe systems.
In *Proceedings of the 18th ACM/IFIP/USENIX Middleware Conference, Las Vegas, NV, USA, December 11 - 15, 2017*, pages 174–184, 2017.

T. Qadah and M. Sadoghi.
QueCC: A queue-oriented, control-free concurrency architecture.
In *Proceedings of the 19th ACM/IFIP/USENIX Middleware Conference, Rennes, France, December 10 - 14, 2018*, 2018.