

Overview of Concurrency in L-Store: 2VCC - Two-version Concurrency Control

Mohammad Sadoghi

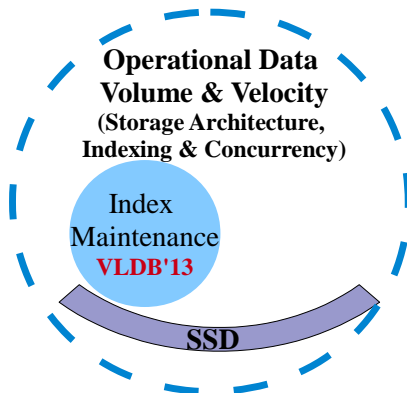
Exploratory Systems Lab
University of California, Davis

ECS165a



- 1 Data Velocity: Index Maintenance
- 2 Data Volume: MVCC Concurrency
- 3 Decentralized & Democratic Data Platform
- 4 References

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

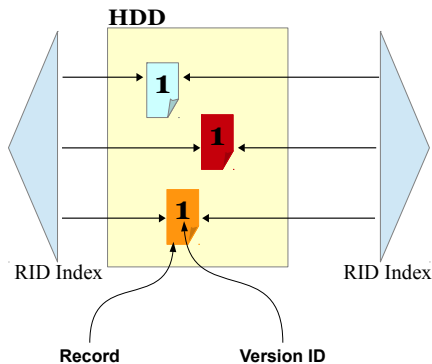
Observed Trends

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

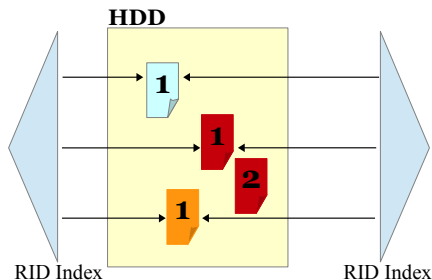
**Decouple Logical and Physical Locations of Records to
Reduce Index Maintenance**

Traditional Multi-version Indexing: Updating Records



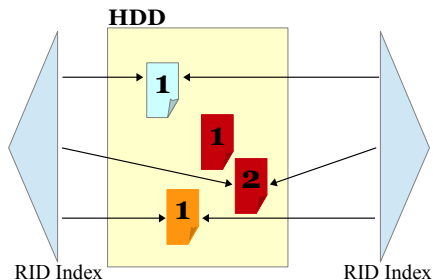
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records



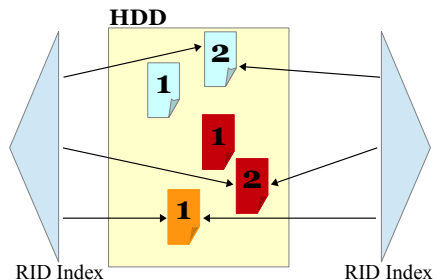
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records



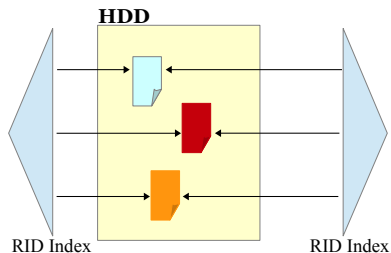
Updating random leaf pages

Traditional Multi-version Indexing: Updating Records

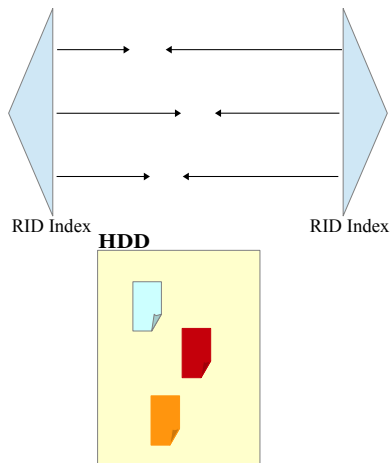


Updating random leaf pages

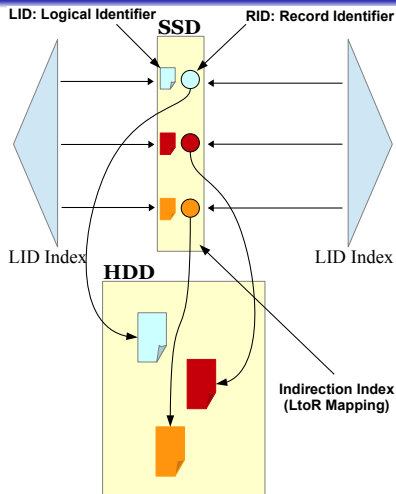
Indirection Indexing: Updating Records



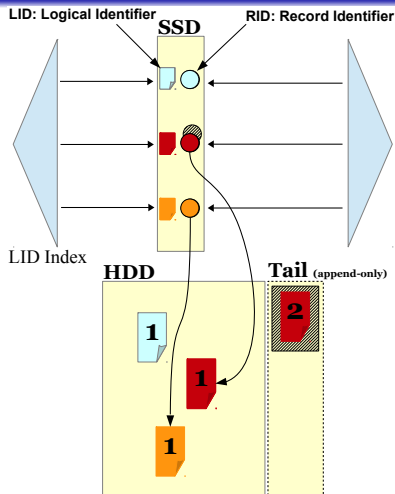
Indirection Indexing: Updating Records



Indirection Indexing: Updating Records

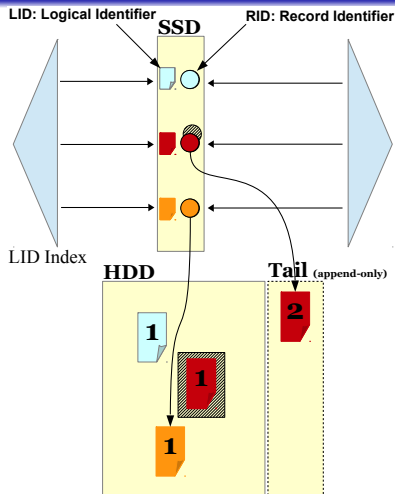


Indirection Indexing: Updating Records



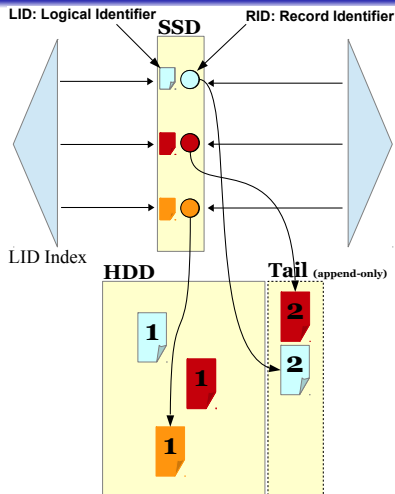
Eliminating random leaf-page updates

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	$3 + K$	$\leq 2 + K$
	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	≤ 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

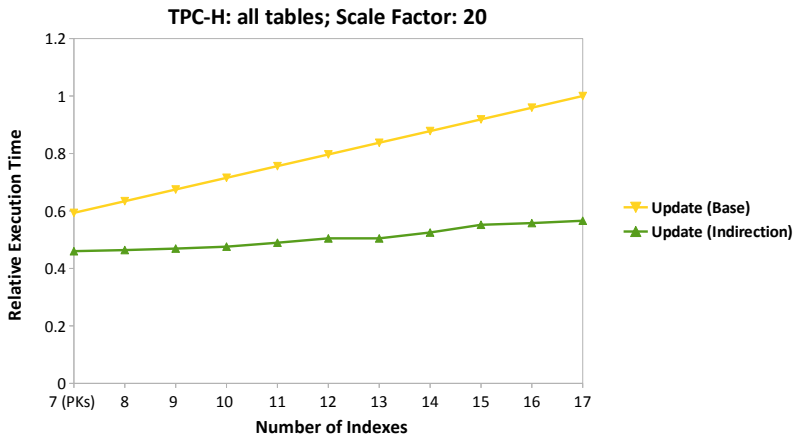
■ Software:

- 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

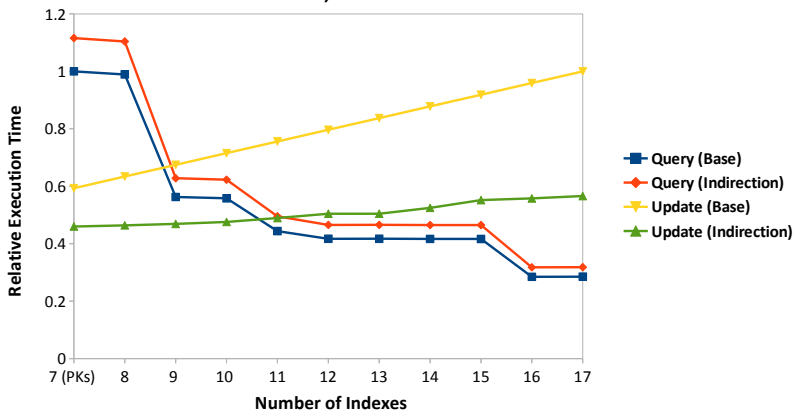
Indirection: Effect of Indexes in Operational Data Stores



Substantially improving the update time ...

Indirection: Effect of Indexes in Operational Data Stores

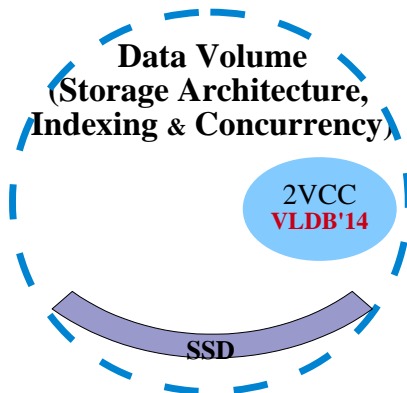
TPC-H: all tables; Scale Factor: 20



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

- 1 Data Velocity: Index Maintenance
- 2 Data Volume: MVCC Concurrency
- 3 Decentralized & Democratic Data Platform
- 4 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

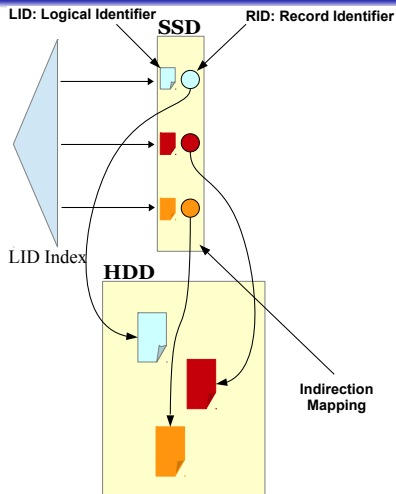
Observed Trends

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

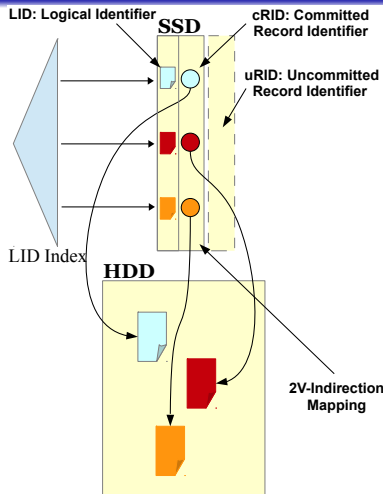
Decouple Readers/Writers to Reduce Contention
(Pessimistic and Optimistic Concurrency Control Coexistence)

2V-Indirection Indexing: Updating Records



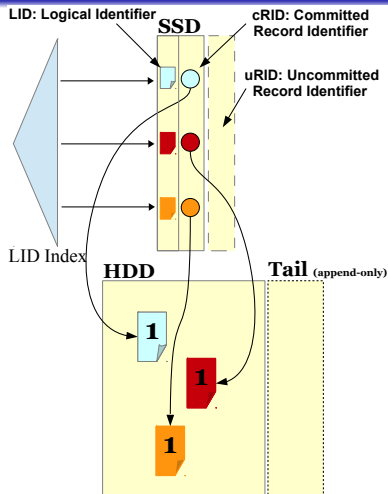
Recap: Indirection technique for reducing index maintenance

2V-Indirection Indexing: Updating Records



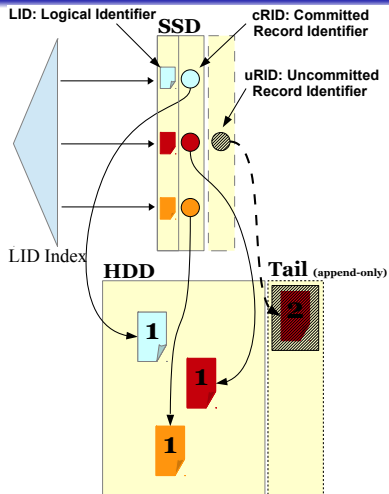
Extending the indirection to committed/uncommitted records

2V-Indirection Indexing: Updating Records



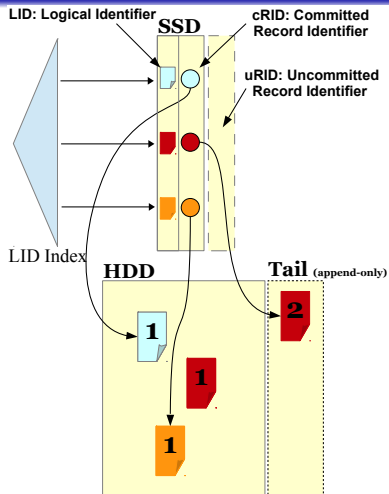
Extending the indirection to committed/uncommitted records

2V-Indirection Indexing: Updating Records



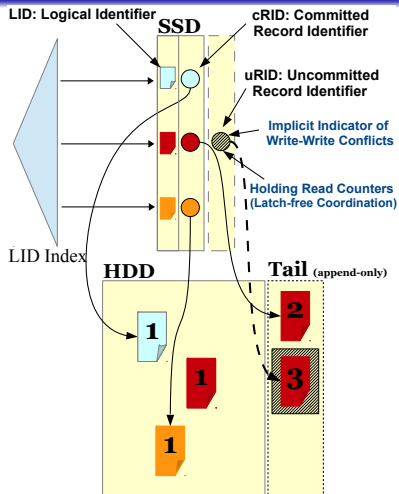
Decoupling readers/writers to eliminate contention

2V-Indirection Indexing: Updating Records



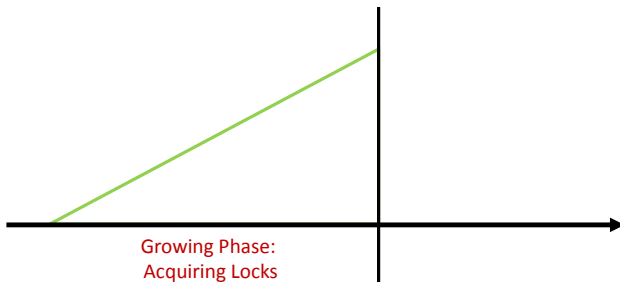
Decoupling readers/writers to eliminate contention

2V-Indirection Indexing: Updating Records



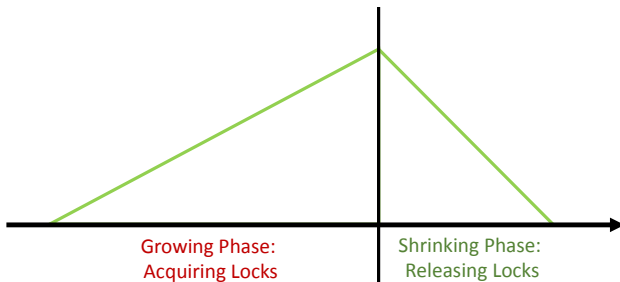
Decoupling readers/writers to eliminate contention

Overview of Two-version Concurrency Control Protocol



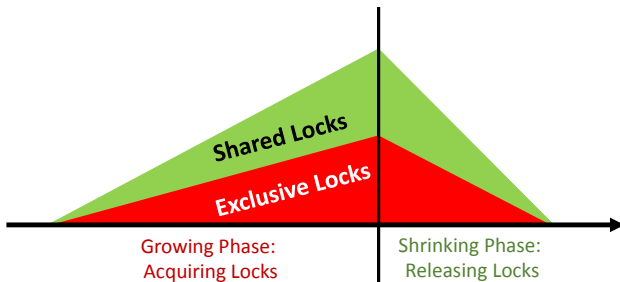
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



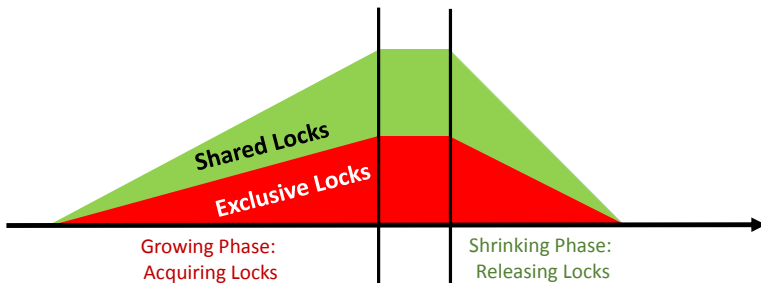
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



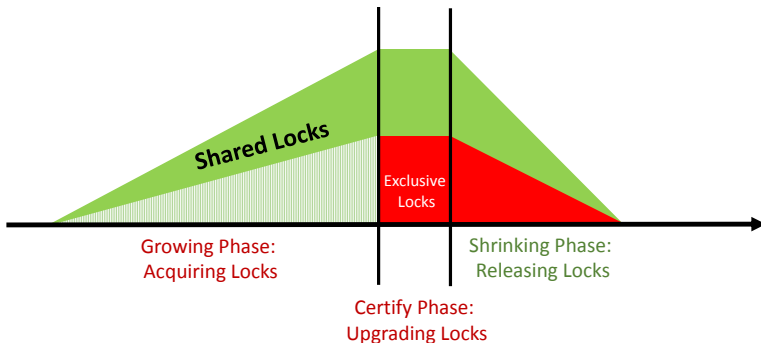
Two-phase locking (2PL) consisting of growing and shrinking phases

Overview of Two-version Concurrency Control Protocol



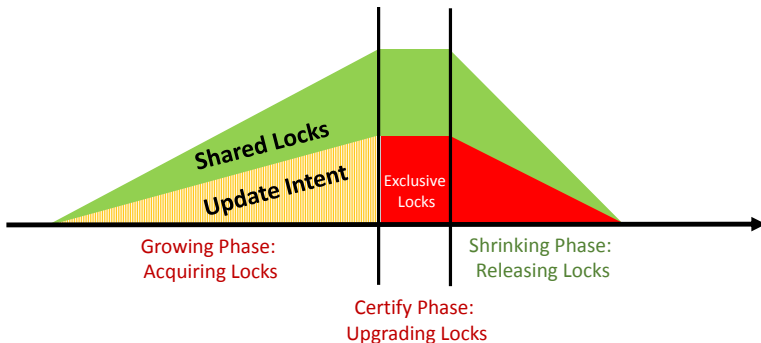
Extending 2PL with certify phase

Overview of Two-version Concurrency Control Protocol



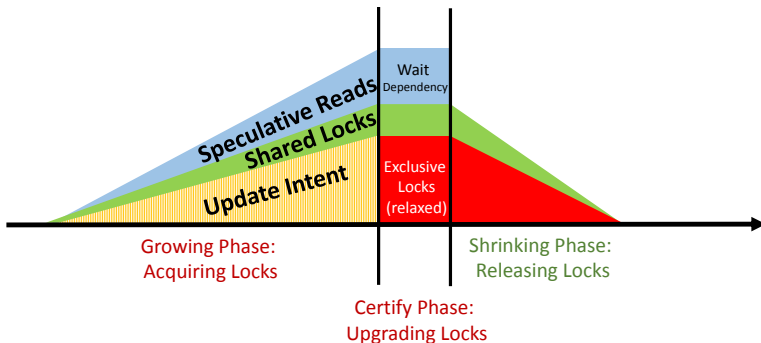
Exclusive locks held for shorter period (inherently optimistic)

Overview of Two-version Concurrency Control Protocol



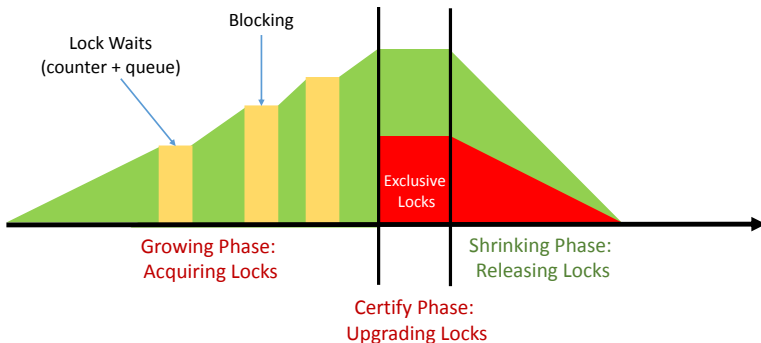
Exclusive locks held for shorter period (inherently optimistic)

Overview of Two-version Concurrency Control Protocol



Relaxed exclusive locks to allow speculative reads (increased optimism)

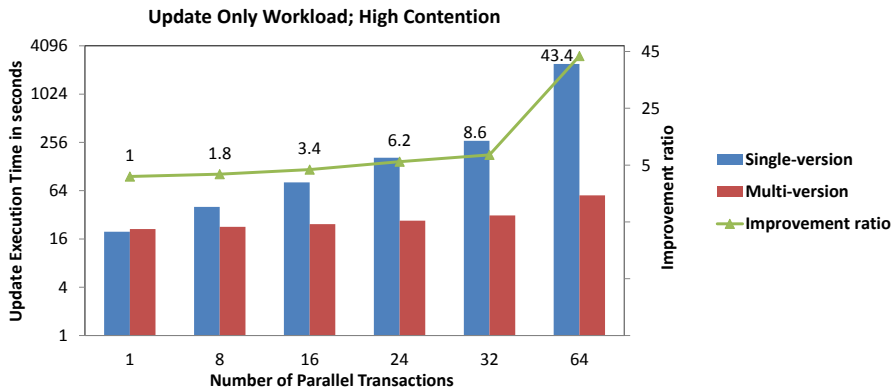
Overview of Two-version Concurrency Control Protocol



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

Experimental Analysis

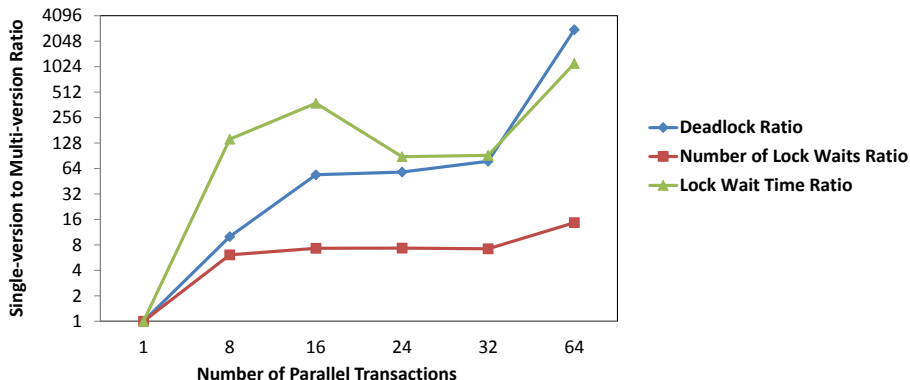
2VCC: Effect of Parallel Update Transactions



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

2VCC: Effect of Parallel Update Transactions

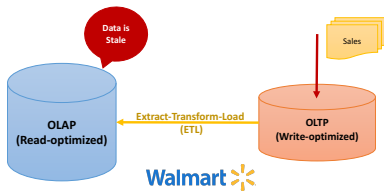
Lock Statistics Comparison; High Contention



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

- 1 Data Velocity: Index Maintenance
- 2 Data Volume: MVCC Concurrency
- 3 Decentralized & Democratic Data Platform**
- 4 References

Recap: Data Management Challenges at Microscale



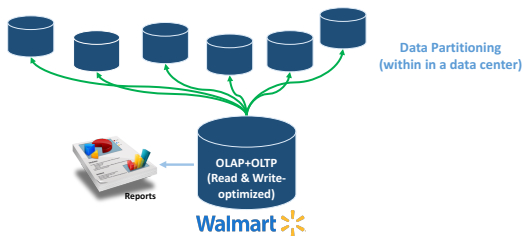
OLTP and OLAP data are isolated at microscale

Recap: Data Management Challenges at Microscale



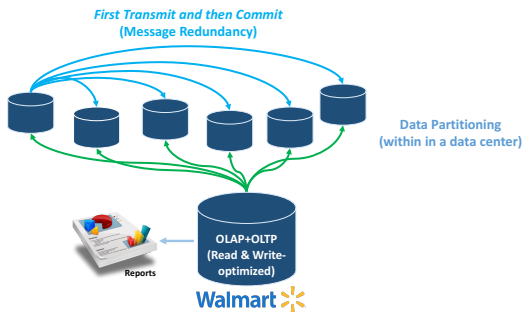
First step is to unify OLTP and OLAP

Platform Scaling: Data Partitioning



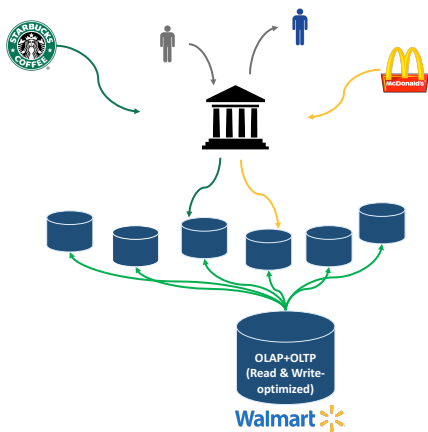
Moving towards distributed environment

Platform Scaling: Non-blocking Agreement Protocols



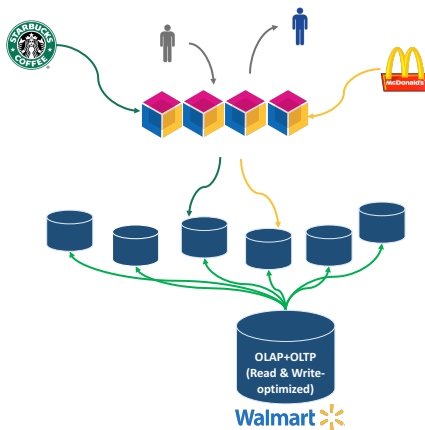
Message redundancy vs. latency trade-offs [EasyCommit, EDBT'18]

Central Control: Data Gate Keeper



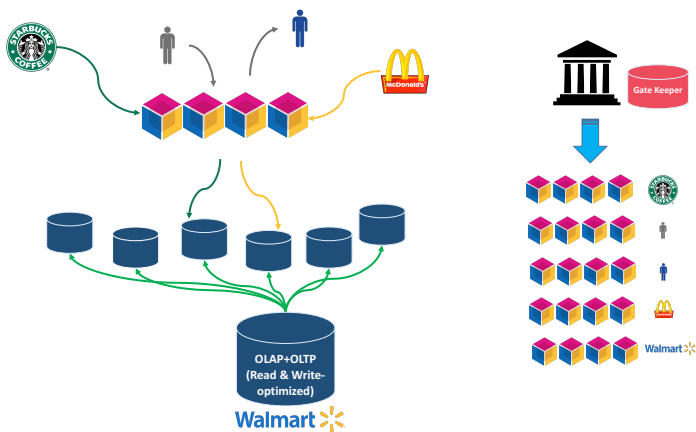
Conform to trusting the central authority and governance

Decentralized Control: Removing Data Barrier



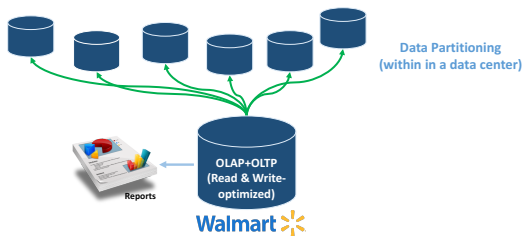
Seek trust in *decentralized* and *democratic* governance [PoE (EDBT'21), RCC (ICDE'21)]

Democratic Control: Removing Trust Barrier



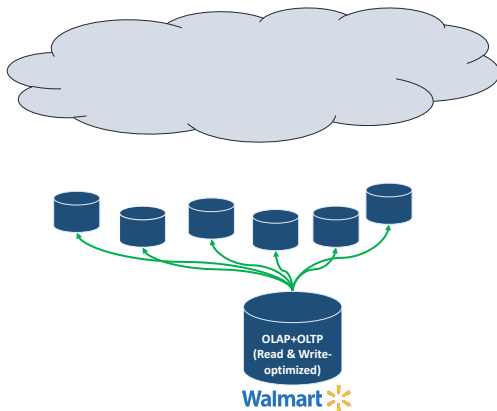
Seek trust in *decentralized* and *democratic* governance [PoE (EDBT'21), RCC (ICDE'21)]

Global-scale Reliable Platform over Unreliable Hardware



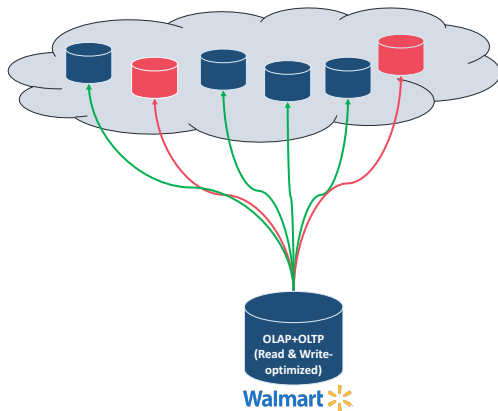
Self-managed infrastructure

Global-scale Reliable Platform over Unreliable Hardware



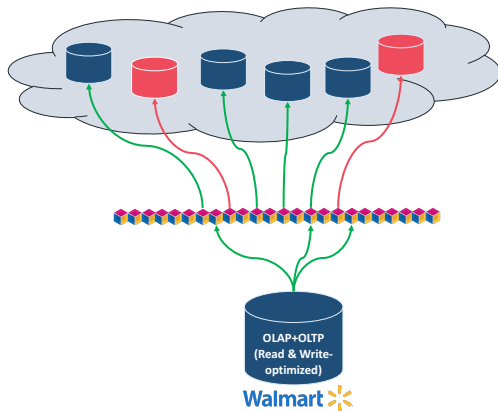
Cloud-managed infrastructure (trust the provider)

Global-scale Reliable Platform over Unreliable Hardware



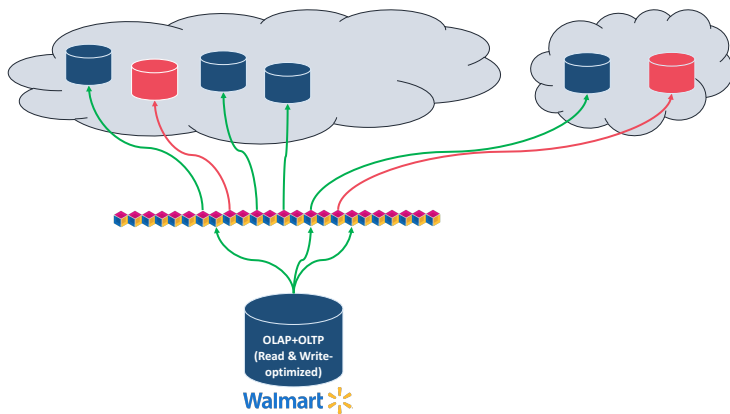
Cloud-managed infrastructure (trust the provider)

Global-scale Reliable Platform over Unreliable Hardware



Light-weight, fault-tolerant, trusted middleware [Blockplane, (ICDE'18)]

Global-scale Reliable Platform over Unreliable Hardware



Global Scale fault-tolerant protocols [GeoBFT (VLDB'20), Delayed Replication (ICDT'20)]

Questions?
Thank you!

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



[illegible]

- [illegible]