



# High Performance Mixed Graph-Based Concurrency Control

**Jack Waudby** (Supervised by Paul Ezhilchelvan)

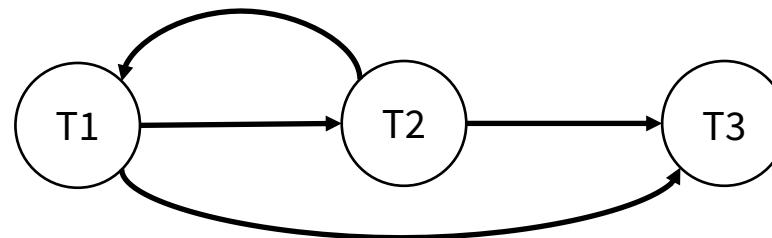
5 Sept 2022

# Concurrency Control

- Performance critical component of modern many-core databases
- Responsible for ensuring the effects of concurrently executing transactions are isolated from each other → serializability
- Concurrency control approaches;
  - Lock-based, e.g., two-phase locking
  - Timestamp-based, e.g., timestamp ordering
  - Validation-based, e.g., optimistic concurrency control
  - **Graph-based a.k.a serialization graph testing (SGT)**
- SGT directly uses the conflict graph theorem
  - Accepting all conflict serializable schedules → no unnecessary aborts

# Conflict Graph Theorem

- Practically databases provide conflict serializability
- Schedule;
  - $s = w1[x] r2[x] r2[y] w1[y] w2[z] w3[z] r3[x] c1 c3 c2$
- Conflict graph;



- Schedule is conflict serializable iff its corresponding conflict graph is acyclic

# Serialization Graph Testing

- Scheduler maintains a conflict graph
- For each operation in a transaction;
  - Determine conflicts
  - Insert corresponding edges into conflict graph (if do not already exist)
  - Execute cycle check
  - If no cycle then continue, else abort transaction
- At commit time delay until no incoming edges
- Historically deemed too costly
  - “*No False Negatives: Accepting All Useful Schedules in a Fast Serializable Many-Core System*”, Dominik Durner & Thomas Neumann, ICDE, 2019
  - Implementation using a concurrent graph data structure
  - Performance comparable with contemporary (SILO, TicToc) and classical protocols (2PL, OCC)

# Motivation

- SGT executes all transactions with Serializable isolation
- Performance of serializable transaction processing can be unsatisfactory for some applications
- Mixed databases allow transactions to run at different, weaker, isolation levels, e.g., Snapshot Isolation, Read Committed
- In practice weak isolation is widely used
  - 2017 DBA survey found the majority of transactions execute at Read Committed
- **Can SGT be extended to support transactions running at different isolation levels?**

# Work Plan

- Investigate the prevalence of mixed databases
- Protocol development;
  - What constitutes a valid execution in a mixed database?
- Initial implementation;
  - Can Durner & Neumann's graph data structure be adapted?

Phase 1

# Mixing in the Wild

- Surveyed isolation levels supported by 24 databases
- 7 isolation levels represented
- **18 databases supported multiple isolation levels**

Database System	Isolation Level						
	RU	RC	CS	SI	CR	RR	S
Actian Ingres 11.0	✓	✓	✓	✗	✗	✓	✗*
Clustrix 5.2	✗	✓ <sup>e</sup>	✗	✗	✗	✓ <sup>*c</sup>	✓
CockroachDB 20.1.5	✗	✗	✗	✗	✗	✗	✗*
Google Spanner	✗	✗	✗	✗	✗	✗	✗*
Greenplum 6.8	✓ <sup>b</sup>	✓*	✗	✗	✗	✓	✗
Dgraph 20.07	✗	✗	✗	✓*	✗	✗	✗
FaunaDB 2.12	✗	✗	✗	✓	✗	✗	✗*
Hyper	✗	✗	✗	✗	✗	✗	✓
IBM Db2 for z/OS 12.0	✓	✓ <sup>a</sup>	✓*	✗	✗	✓	✗
MySQL 8.0	✓	✓	✗	✗	✗	✓ <sup>*</sup>	✓
MemGraph 1.0	✗	✗	✗	✓*	✗	✗	✗
MemSQL 7.1	✗	✓ <sup>e</sup>	✗	✗	✗	✗	✗
MS SQL Server 2019	✓	✓*	✗	✓	✗	✓	✓
Neo4j 4.1	✗	✓*	✗	✗	✗	✗	✓
NuoDB 4.1	✗	✓	✗	✗	✓*	✗	✗
Oracle 11g 11.2	✗	✓*	✗	✓	✗	✗	✗
Oracle BerkeleyDB	✓	✓	✓	✓	✗	✗	✓
Oracle BerkeleyDB JE	✓	✓	✗	✗	✗	✓*	✓
Postgres 12.4	✓ <sup>b</sup>	✓*	✗	✗	✗	✓ <sup>c</sup>	✓
SAP HANA	✗	✓*	✗	✓	✗	✗	✗
SQLite 3.33	✓	✗	✗	✗	✗	✗	✓*
TiDB 4.0	✗	✗	✗	✓*	✓	✗	✗
VoltDB 10.0	✗	✗	✗	✗	✗	✗	✓*
YugaByteDB 2.2.2	✗	✗	✗	✓*	✗	✗	✓

\* Indicates the default setting.

<sup>a</sup> Referred to as *Read Stability*.

<sup>b</sup> Behaves like *Read Committed* due to MVCC implementation.

<sup>c</sup> Implemented as *Snapshot Isolation*.

<sup>d</sup> Requires manual lock management.

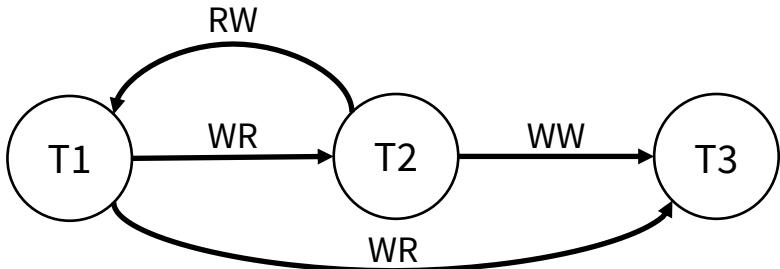
<sup>e</sup> Behaves like *Consistent Read*.

# Adya Model

Weak Consistency: A Generalized Theory and  
Optimistic Implementations for Distributed  
Transactions  
Atul Adya, PhD Thesis

Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions  
Atul Adya  
Submitted to the Department of Electrical Engineering and Computer Science  
Faculty Advisor: Prof. Albert C. Smith  
Degree of Philosophy  
MASSACHUSETTS INSTITUTE OF TECHNOLOGY  
© Massachusetts Institute of Technology 1998. All rights reserved.  
Author  
Department of Electrical Engineering and Computer Science  
Massachusetts Institute of Technology  
Cambridge, MA 02139  
Directed by  
Prof. Albert C. Smith  
Accepted by  
Albert C. Smith  
Chairman, Department Committee on Graduate Studies

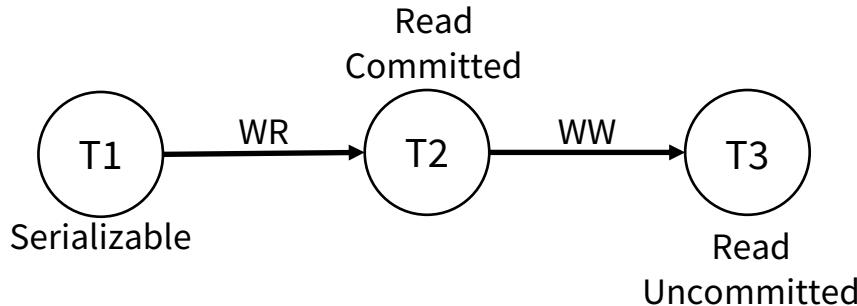
- Adya represents a schedule with a **direct serialization graph (DSG)**
- $s = w1[x_1] r2[x_1] r2[y_0] w1[y_1] w2[z_2] w3[z_3] r3[x_1] c1 c3 c2$



- **Anomalies** are defined by stating properties about the DSG
  - E.g., dirty write ( $G_0$ ) is a cycle containing only WW edges
- **Isolation levels** defined by the anomalies they prevent
  - E.g., Read Committed prevents  $G_0$  and  $G_1$  anomalies
  - Define 11 isolation levels

# Mixing-Correct Theorem

- In a mixed database schedule represented by a **mixed serialization graph (MSG)**
- MSG only includes relevant and obligatory conflicts;
  - All WW edges included
  - WR edges included if incoming to read committed/serializable transactions
  - RW edges are included if outgoing edges from serializable transactions.
- $s = w1[x1] r2[x1] r2[y0] w1[y1] w2[z2] w3[z3] r3[x1] c1 c3 c2$



- **Schedule is mixing-correct iff its corresponding MSG is acyclic**

# Mixed Serialization Graph Testing

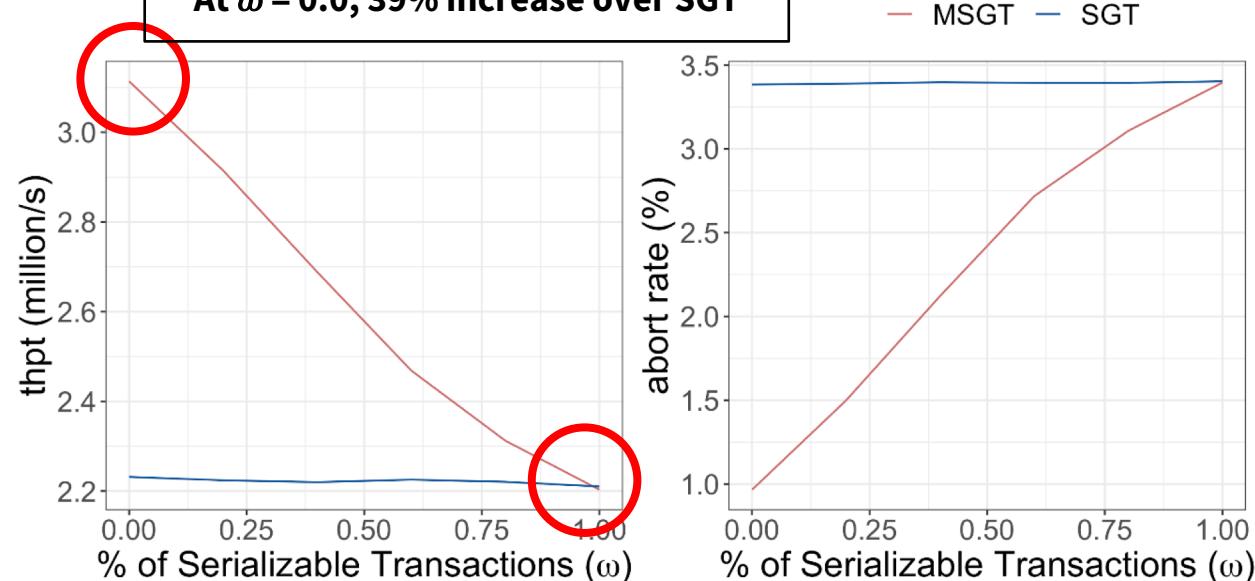
- Use concurrent graph data structure with nodes annotated with requested isolation level
- For each operation in a transaction;
  - Determine conflicts
  - Insert corresponding edges into conflict graph (if do not already exist) **as-per the mixing-correct theorem inclusion rules**
  - Execute cycle check
  - If no cycle then continue, else abort transaction
- At commit time delay until no incoming edges

# Evaluation

- **Implementation;** SGT and MSGT implemented in our prototype in-memory database. MSGT supports Serializable, Read Committed & Read Uncommitted
- **Hardware;** Azure Standard D48v3 instance with 48 virtualized CPU cores and 192GB of memory
- **Metrics;**
  - *Throughput:* number of transactions committed per second
  - *Abort rate:* rate at which transactions are being aborted
  - *Average latency:* the latency time of committed transactions (in ms) averaged across the measurement period
- **Benchmark;** YCSB with an additional parameter controlling proportion of serializable transactions

# Isolation Experiment

At  $\omega = 0.0$ , 39% increase over SGT



At  $\omega = 1.0$ , SGT outperforms MSGT

Parameters:

- $\theta = 0.8$  (medium contention)
- $U = 0.5$  (balanced mix)
- 10 ops/txn
- 40 cores

# Work Plan

- Investigate the prevalence of mixed databases
- Protocol development;
  - What constitutes a valid execution in a mixed database?
- Initial implementation;
  - Can Durner & Neumann's graph data structure be adapted?

Phase 1

- MSGT optimizations;
  - Relevant-DFS & Early commit
- Additional benchmarks, e.g., TPC-C, SmallBank, TATP

Phase 2

- Support additional isolation levels, e.g., snapshot isolation
- Compare with mixed-2PL
- Distributed MSGT

Phase 3

# Thanks for listening

Check out my podcast!



- Email: [j.waudby2@newcastle.ac.uk](mailto:j.waudby2@newcastle.ac.uk)
- Twitter: jwaudberry
- LinkedIn: jack-waudby

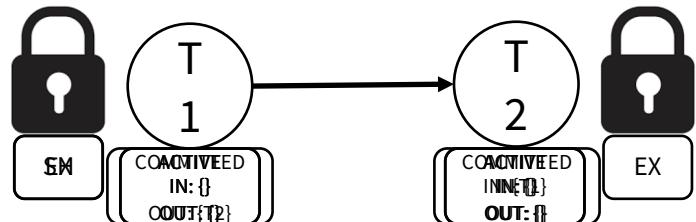


$\Omega$

# No False Negatives

- Efficient SGT implementation
- Concurrent graph data structure;
  - Nodes store a txn status (committed, active, aborted)
  - Nodes have 2 sets of pointers representing incoming and outgoing edges
- Node-level locking protocol;
  - If T1 detects a conflict with T2 and an edge does not exist
  - T1 acquires *shared lock* on T2 conflicting txn and inserts pointer
  - T1 executes cycle check using reduced depth-first search holding nodes with *shared locks*
  - T1 acquires an *exclusive lock* on its node and checks for incoming edges
  - If incoming edges, release *exclusive lock* and repeat. Else, release and commit
- Example;
  - $s = w1[x] \ r2[x] \ c2 \ c1$

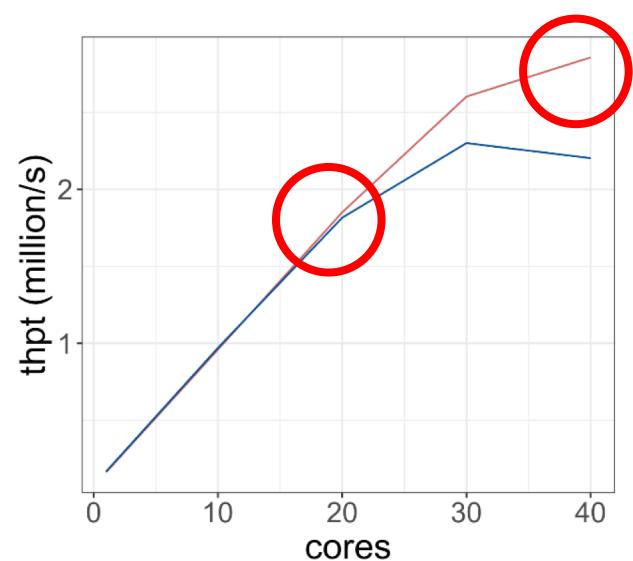
No False Negatives: Accepting All Useful Schedules  
in a Fast Serializable Many-Core System  
*Dominik Durner & Thomas Neumann*  
*ICDE, 2019*



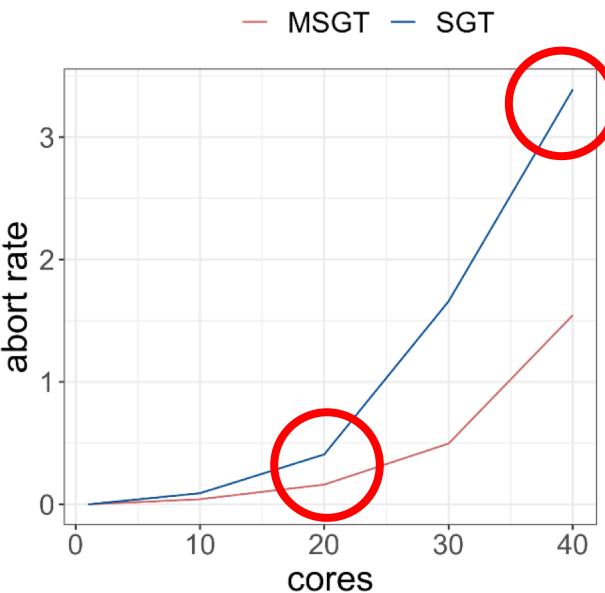
# Scalability Experiment

Parameters:

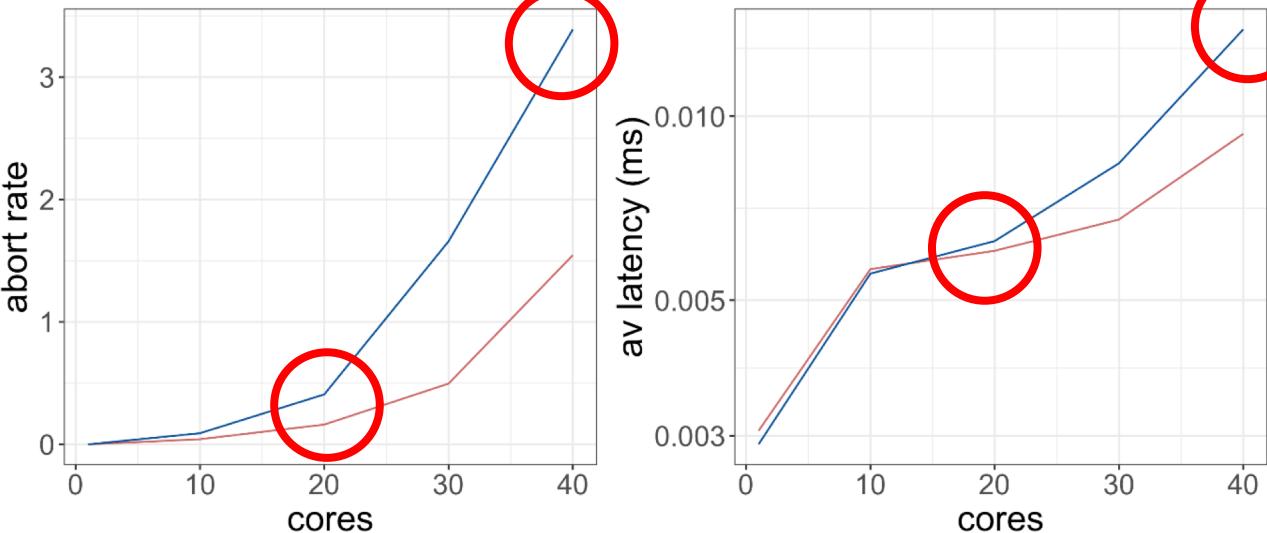
- $\omega = 0.2$  (low serializable)
- $U = 0.5$  (balanced mix)
- 10 ops/txn
- $\theta = 0.8$  (medium contention)



Up to 20 cores no difference



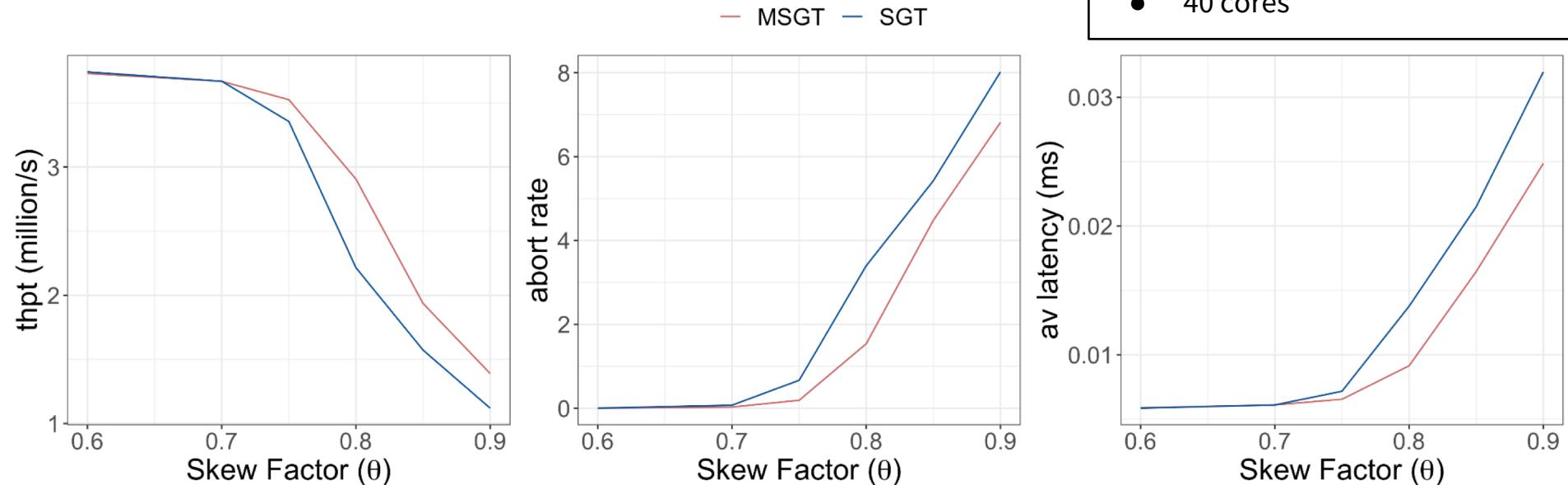
At 40 cores, throughput increases by 28%,  
abort rate halves, and latency drops



# Contention Experiment

Parameters:

- $\omega = 0.2$  (low serializable)
- $U = 0.5$  (balanced mix)
- 10 ops/txn
- 40 cores



# Update Rate Experiment

Parameters:

- $\omega = 0.2$  (low serializable)
- $\theta = 0.8$  (medium contention)
- 10 ops/txn
- 40 cores

