Engineering and Physical Sciences Research Council

Pick ‘n’ Mix Isolation Levels: Mixed Serialization Graph Testing

J. Waudby[1], P. Ezhilchelvan[1], J. Webber[2]

5 Sept 2022

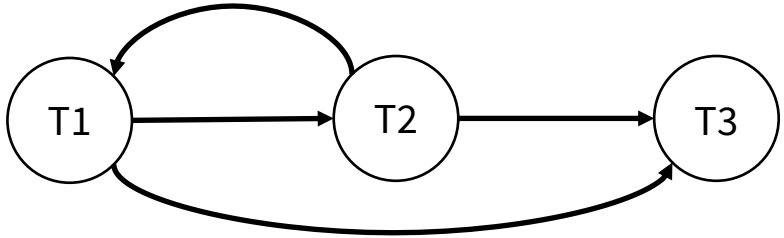
1. Newcastle University
2. Neo4j

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
- 3 conflicts: write-write, write-read, read-write
- 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?**
  - Investigate the prevalence of mixed databases
  - Protocol development; what constitutes a valid execution in a mixed database?
  - Implementation; can Durner & Neumann's graph data structure be adapted?

Mixing in the Wild

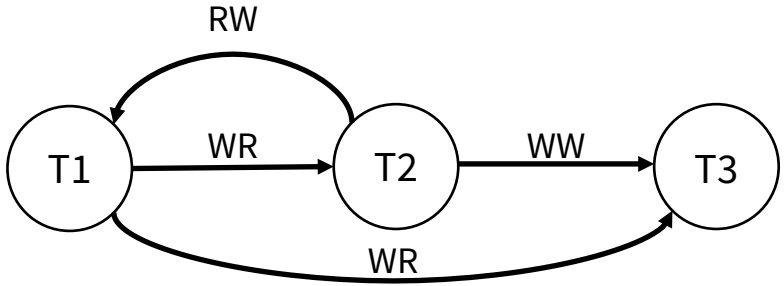
- Surveyed the isolation levels supported by 24 databases
- 7 isolation levels represented
- **18 databases supported multiple isolation levels**

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

* Indicates the default setting.
[a] Referred to as ***Read Stability***.
[b] Behaves like ***Read Committed*** due to MVCC implementation.
[c] Implemented as ***Snapshot Isolation***.
[d] Requires manual lock management.
[e] Behaves like ***Consistent Read***.

Graph-based Weak Isolation Formalism

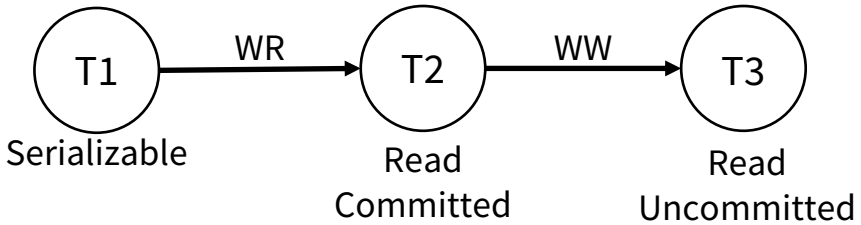
- *“Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions”*, Adya, PhD Thesis
- Represent schedule with a **direct serialization graph (DSG)**
  - s = w1[x1] r2[x1] r2[y0] w1[y1] w2[z2] w3[z3] r3[x1] c1 c3 c2



- **Anomalies** are defined by stating properties about the DSG
  - E.g., dirty write (G0) is a cycle containing only WW edges
- **Isolation levels** defined by the anomalies they prevent
  - E.g., Read Uncommitted prevents G0 anomalies

Mixing-Correct Theorem

- In a mixed database schedule represented by a **mixed serialization graph (MSG)**
- Annotate nodes with isolation level
- Include only 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 **(Change 1)**
- 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 (Change 2)**
  - 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 Read Uncommitted, Read Committed & Serializable
- **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

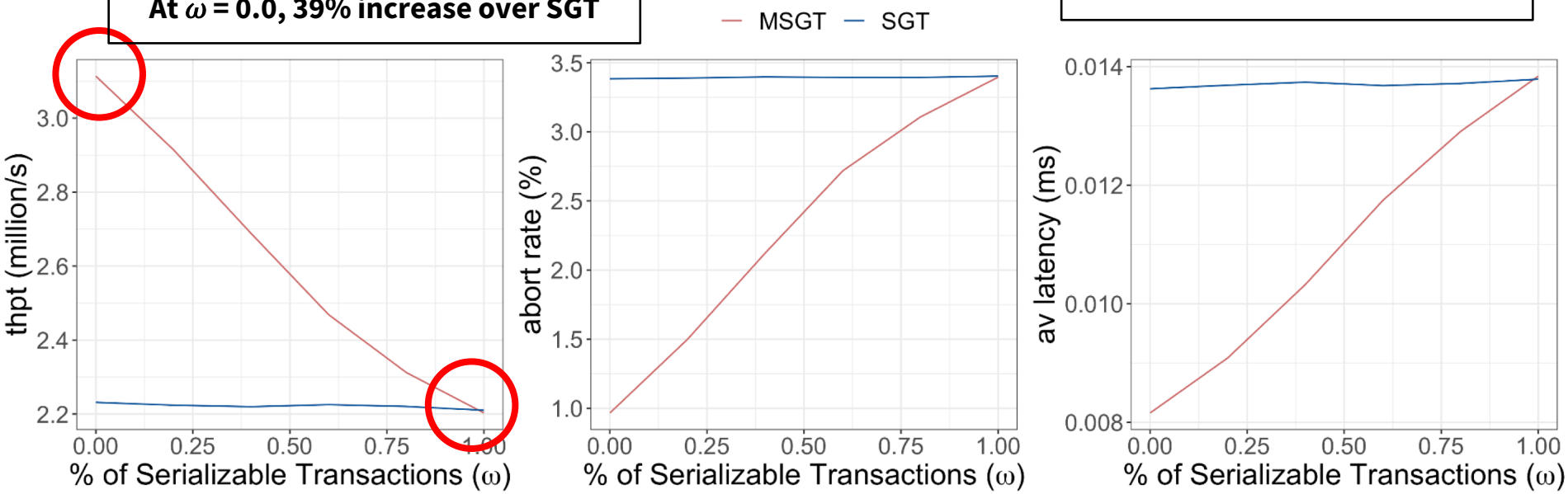
YCSB Benchmark

- Employed as a microbenchmark
- 4 workload factors;
  - Operations per transaction
  - Skew factor (θ) controls the contention level
  - Update rate (U)
  - Proportion of serializable transactions (ω)
- 4 experiments;
  - Isolation
  - Scalability
  - Contention
  - Update rate

Isolation Experiment

Parameters:
- θ = 0.8 (medium contention)
- U = 0.5 (balanced mix)
- 10 ops/txn
- 40 cores

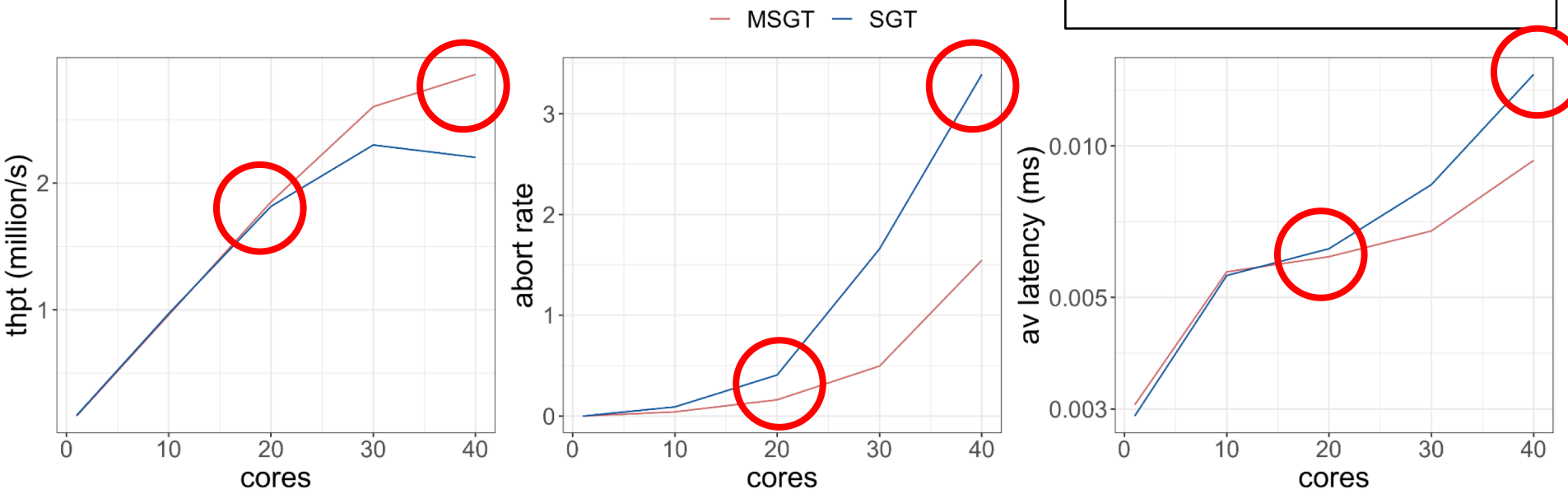
At ω = 0.0, 39% increase over SGT



At ω = 1.0, SGT outperforms MSGT

Scalability Experiment

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

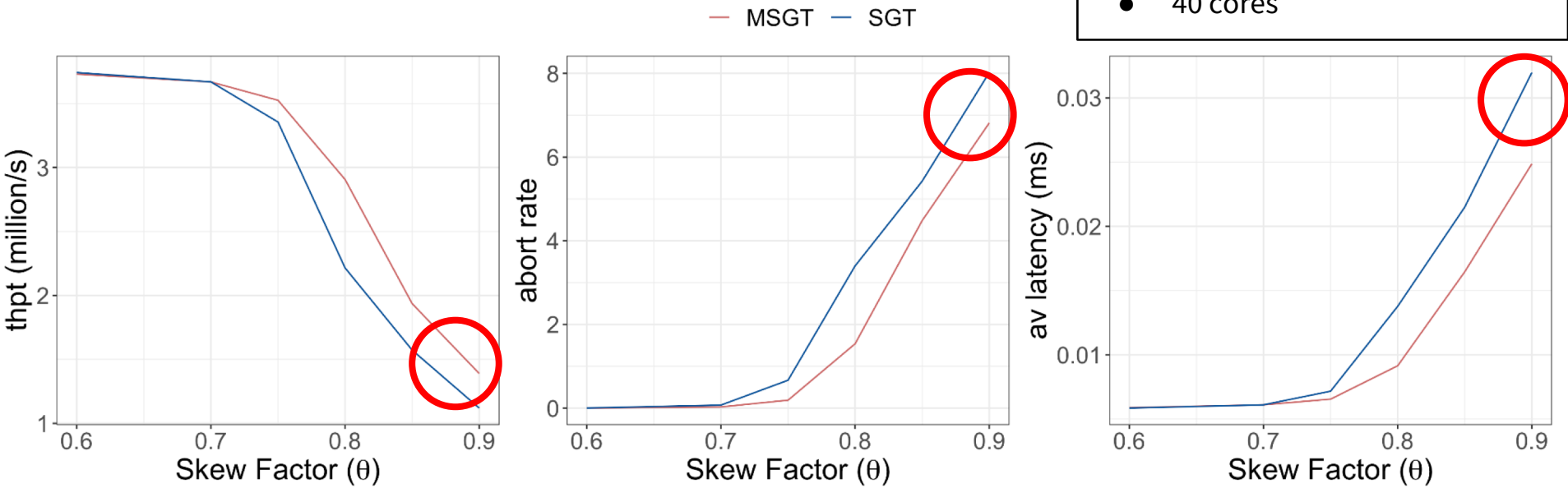


Up to 20 cores no difference

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

Contention Experiment

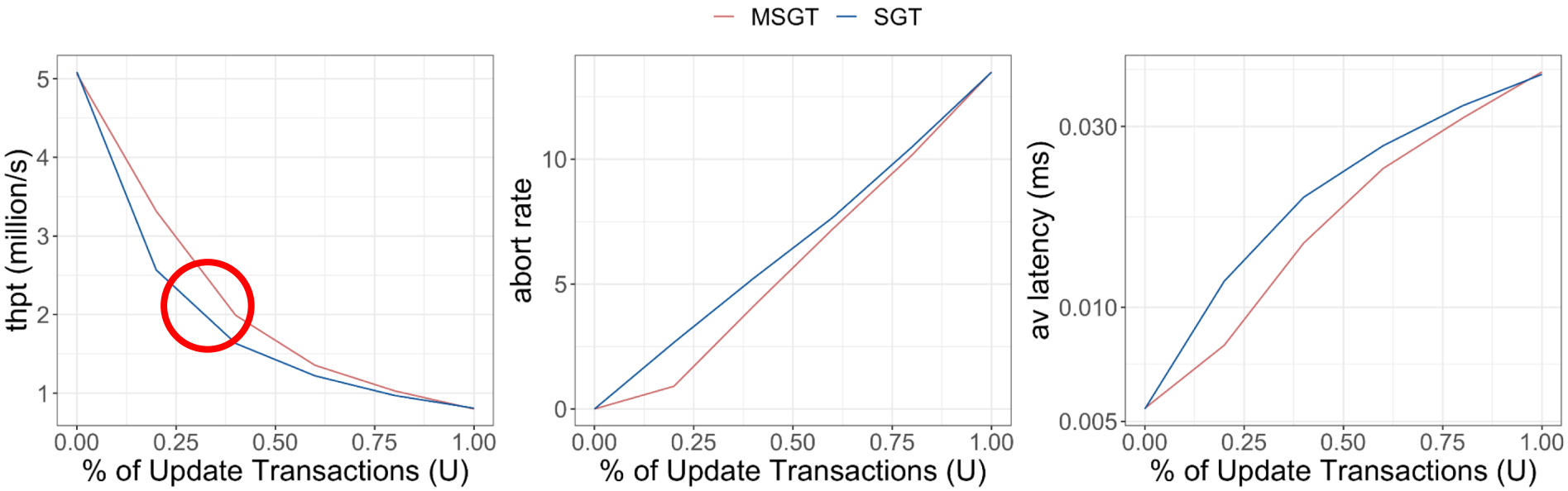
Parameters:
- $\omega = 0.2$ (low serializable)
- $U = 0.5$ (balanced mix)
- 10 ops/txn
- 40 cores



MSGT outperforms SGT under high contention

Update Rate Experiment

Parameters:
- ω = 0.2 (low serializable)
- θ = 0.8 (medium contention)
- 10 ops/txn
- 40 cores



MSGT outperforms SGT when the workload is balanced

TL;DR

- Serialization graph testing (SGT) accepts all conflict serializable schedules → no unnecessary aborts
- Historically deemed too computational expensive
- Durner & Neumann (2019) developed a concurrent graph data structure to implement SGT in a many-core database
- SGT executes all transaction with serializable isolation → practical databases often allow some transactions to execute with weaker isolation
- **Mixed serialization graph testing blends Durner & Neumann's data structure with Adya's weak isolation formalism to allow users to pick 'n' mix isolation levels on a per-transaction basis**

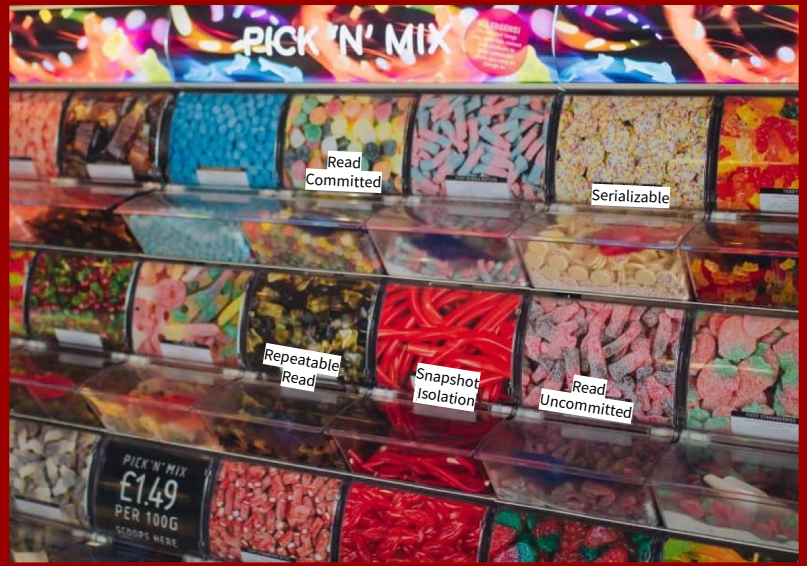
Conclusion & Future Work

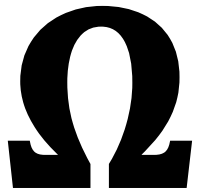
- Strengthens recent work refuting the assumption that graph-based concurrency control is impractical
- When workloads contain transactions running at weaker isolation levels MSGT is able to outperform SGT
- Future work;
  - Compare performance with other mixed concurrency control protocols
  - Support additional isolation levels, e.g., snapshot isolation
  - Extend the performance evaluation to include industry standard benchmarks such as TPCx-IoT and TPC-C

Thanks for listening

Check out my podcast!

- Email: j.waudby2@newcastle.ac.uk
- Twitter: jwaudberry
- LinkedIn: jack-waudby





No False Negatives

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

- 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