【ISSTA 24】DBStorm: Generating Various Effective Workloads for Testing Isolation Levels

2024.09.24

中國人民大學
RENMIN UNIVERSITY OF CHINA

目录
CONTENTS

1 Background

2 DBStorm Framework

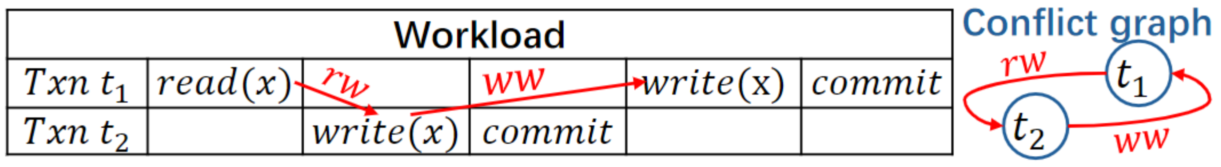
3 Experiments

Part 1

Background

Background

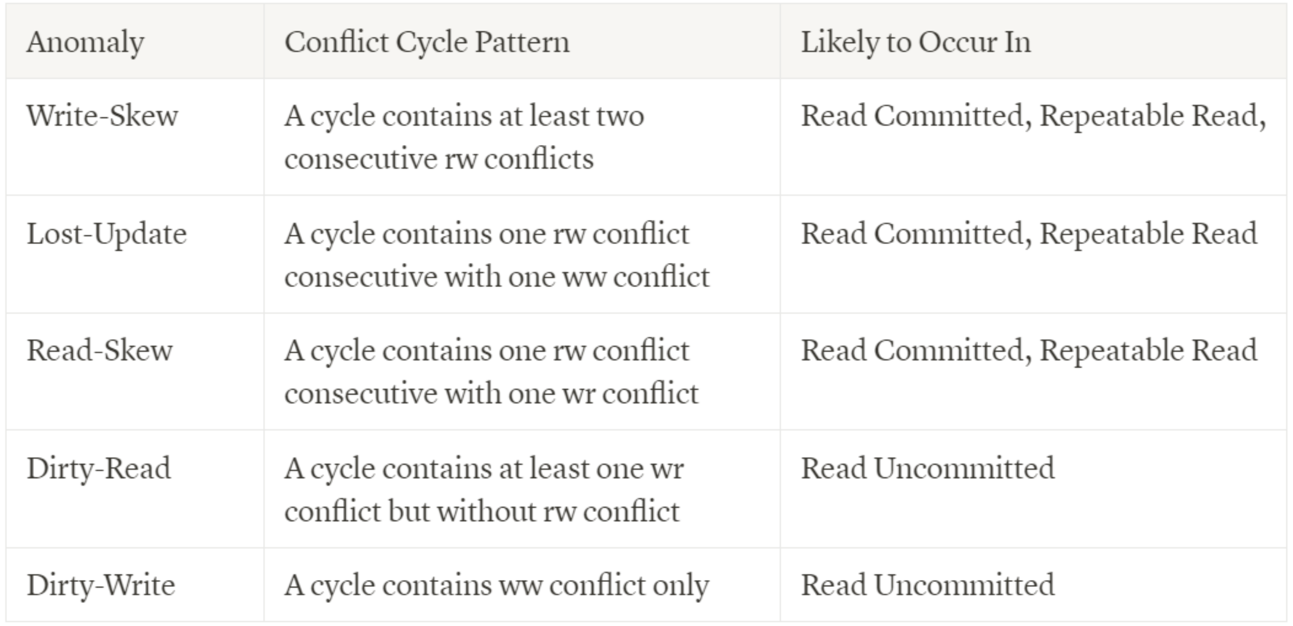
- 冲突图(Conflict Graph)来表示数据库事务之间的并发冲突



Workload					
$Txn\ t_1$	$read(x)$			$write(x)$	$commit$
$Txn\ t_2$		$write(x)$	$commit$		

Figure 1: Example of the Conflict Graph

- 不同的异常对应特定的冲突环模式(Conflict Cycle Pattern)
- 现有的研究未能有效生成用于测试隔离级别的多样化负载，难以全面覆盖



Anomaly	Conflict Cycle Pattern	Likely to Occur In
Write-Skew	A cycle contains at least two consecutive rw conflicts	Read Committed, Repeatable Read,
Lost-Update	A cycle contains one rw conflict consecutive with one ww conflict	Read Committed, Repeatable Read
Read-Skew	A cycle contains one rw conflict consecutive with one wr conflict	Read Committed, Repeatable Read
Dirty-Read	A cycle contains at least one wr conflict but without rw conflict	Read Uncommitted
Dirty-Write	A cycle contains ww conflict only	Read Uncommitted

Challenges

C1: Black-box Testing : internal **I**solation **L**evel(**IL**) code is inaccessible

Generate a workload with a large number of conflict graphs.

C2: Effective Testing : **redundant** or **ineffective** workloads

A lightweight data state mirroring method: precise record access and efficiently generate dissimilar conflict graphs

C3: Anomaly-sensitive Testing : customize workloads to trigger **specific anomalies**

An implantation-based approach that orchestrates conflict record accesses

高效地生成精准且能够检测不同异常的workloads

中国人民大學
RENMIN UNIVERSITY OF CHINA

Part 2

DBStorm Framework

DBStorm Framework

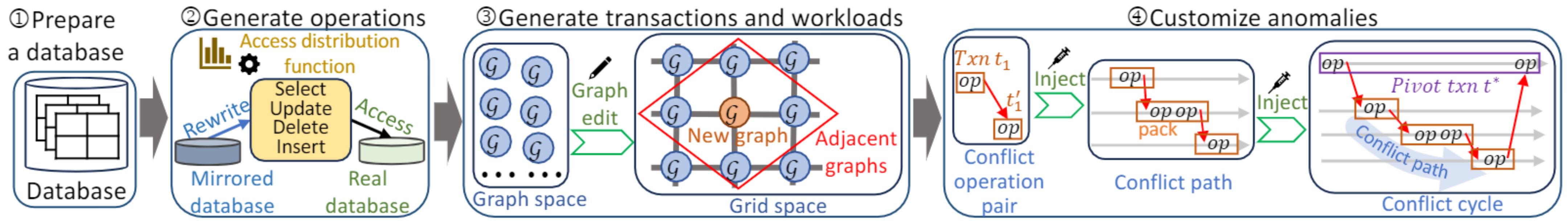


Figure 2: DBStorm Framework

① Preparing a DB instance: SQLancer/DT[2]

② Generating SQL Operations: Mirror the Data State and Instantiate Access Parameters

③ Grouping SQL Operations as Transactions： Conflict Graph Isomorphism Mapping

④ Customizing Non-serializable Anomalies： An implantation-based approach and a pivot based transaction scheduling policy.

Operation Generation

Challenge: ensure that each operation actually accesses its expected records

Mirroring the Data State

a distribution-guided method for mirroring and manipulating the fk/nk attributes.

R1-Insert: the data to be inserted is sampled from the data distribution function;

R2-Delete: the data to be deleted is randomly selected from the database;

R3-Update: performed by a deletion and then an insertion on the same record.

a partition-based method for mirroring and manipulating the pk attributes.

Static keys; *Dynamic keys* $\langle L, \lambda\rangle$, $L \in \{\mathit{free}, \mathit{read}, \mathit{write}\}$, $\lambda \in \{0,1\}$;

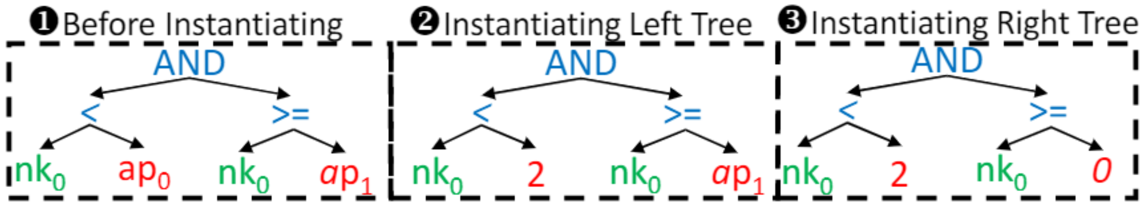
Instantiating Access Parameters：

Item-read/write with pk

pk dynamic & $\lambda = 0$, *insert;* $\lambda = 1$, *select update,delete; Static: update fk/nk*

Predicate-read with fk/nk

$pred \leftarrow$ *instantiate aps by sampling from the mirrored attribute*;



Transaction Generation

Challenge：生成足够多样化的冲突图， 图的相似度通过Graph Edit Distance衡量

C1：事务中的操作数量

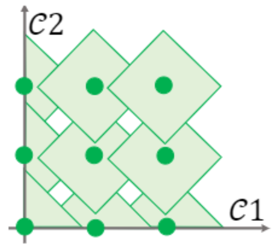
Randomly select a transaction from the shrinking group and merge its operation into a growing group transaction.

C2：事务读写操作比例

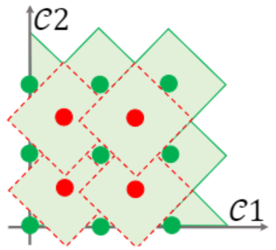
Randomly select a write operation and mutate it as a read operation.

history-independent & monotonically

Hybrid Editing Method：considers both C1's and C2's editing distances.



(a) Separate Editing Method



(b) Hybrid Editing Method

Anomaly Customization

An implantation-based approach

① 确定冲突环的模式 ϕ 和长度 L，相邻两个操作(*op*2*i* 和 *op*2*i*+1)需要访问相同的记录，并且至少有一个是写操作

② 启动一个长事务(pivot事务)并注入操作*op*0

③将队列中的操作植入其他事务

④完成冲突环的构造

{*op*0(R1),*op*1(W1), *op*2(R2), *op*3(W2), *op*4(R3), *op*5(W3)}

T1(pivot)	*T2*	*T3*
op0-R1		
	op1-W1	
	op2-R2	
		op3-W2
		op4-R3
op5-W3		

Anomaly Customization

An implantation-based approach

① 确定冲突环的模式 ϕ 和长度 L

② 启动一个长事务(pivot事务)并植入操作*op*0

③将队列中的操作植入其他事务

④完成冲突环的构造

{*op*0(R1),*op*1(W1), *op*2(R2), *op*3(W2), *op*4(R3), *op*5(W3)}

$$\begin{tikzcd}
\text{T1(pivot)} & \text{T2} & \text{T3} \\
\text{op0-R1} \arrow[dr] & & \\
& \text{op1-W1} & \\
& \text{op2-R2} \arrow[dr] & \\
& & \text{op3-W2} \\
& & \text{op4-R3} \arrow[dll] \\
\text{op5-W3} & &
\end{tikzcd}$$

Part 3

Experiments

IL-related Code Coverage

Baseline: [SQLsmith、Squirrel、SSQLancer、TQS] [TPC-C、SmallBank、YCSB]

[Jepsen、DT2 、Troc]

A higher code coverage implies a higher likelihood of detecting bugs

Access Ratio $\boldsymbol{\alpha}$ and Distribution Fitness $\boldsymbol{\beta}$

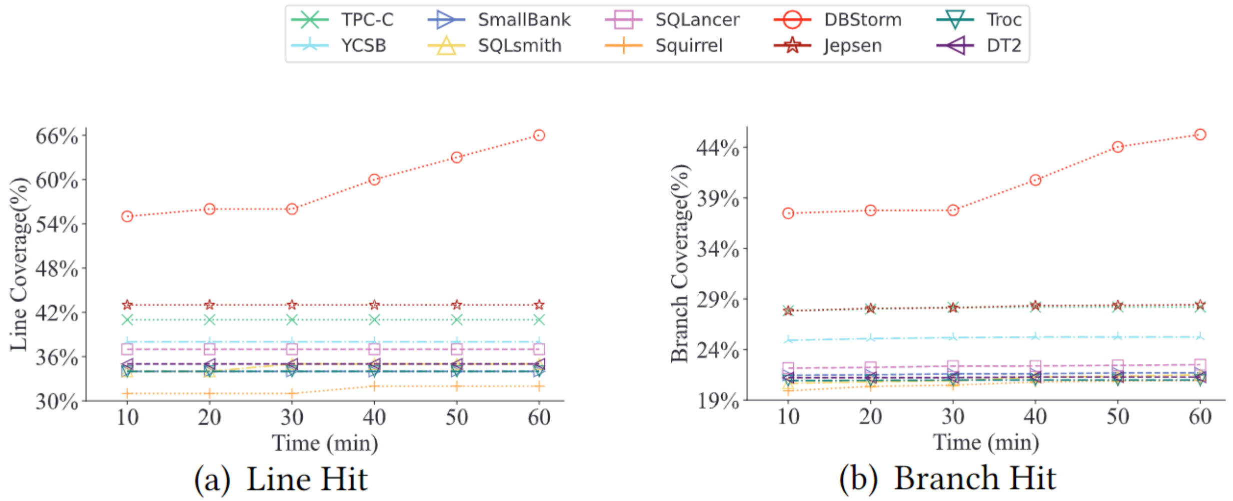


Figure 6: Code Coverage Comparison

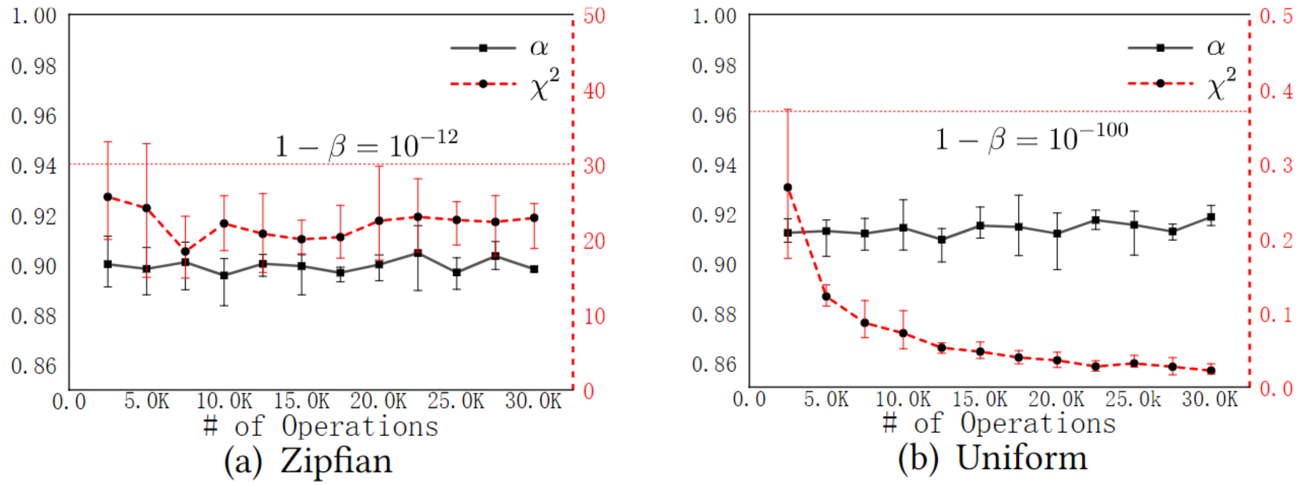


Figure 7: Access Ratio α and Distribution Fitness β

Effective Testing

Diversity of Conflict Graphs.

TQS： Embedding-based

3× higher than TQS in graph generation throughput and remains linear scalable

2× higher than TQS in generating non-isomorphic graphs

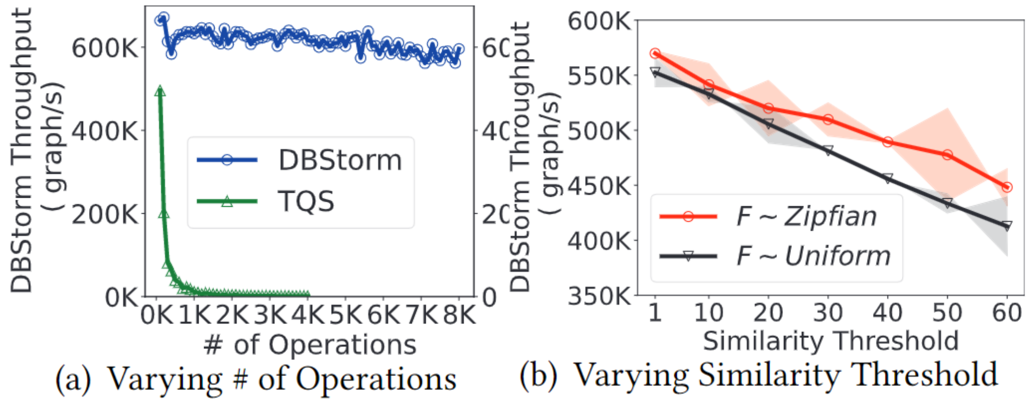


Figure 8: The Throughput of Diverse Graph Generation

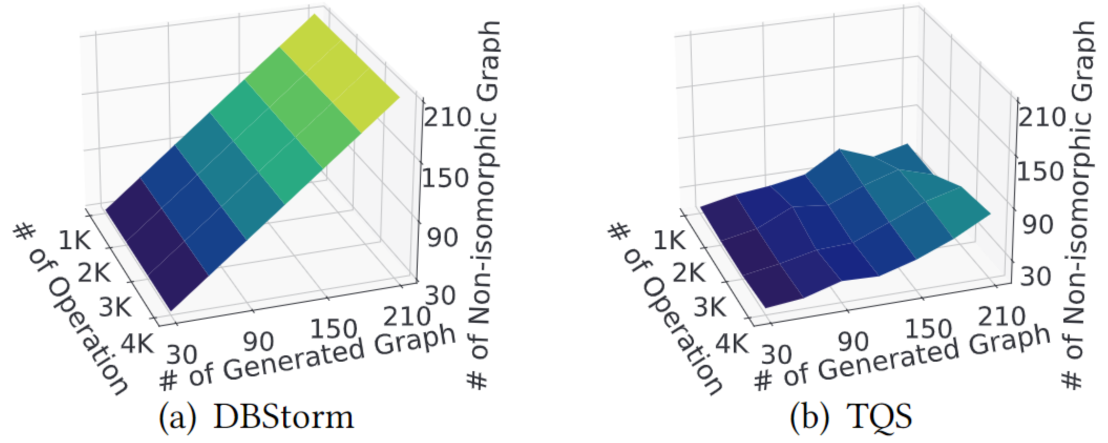


Figure 9: The Accuracy of Non-isomorphic Graph Generation

Anomaly-Sensitive Testing

Disable CC : DBStorm can form long conflict cycles and generate a wide variety of anomalies.

Enable CC : DBStorm can trigger more prevention actions than that of other tools

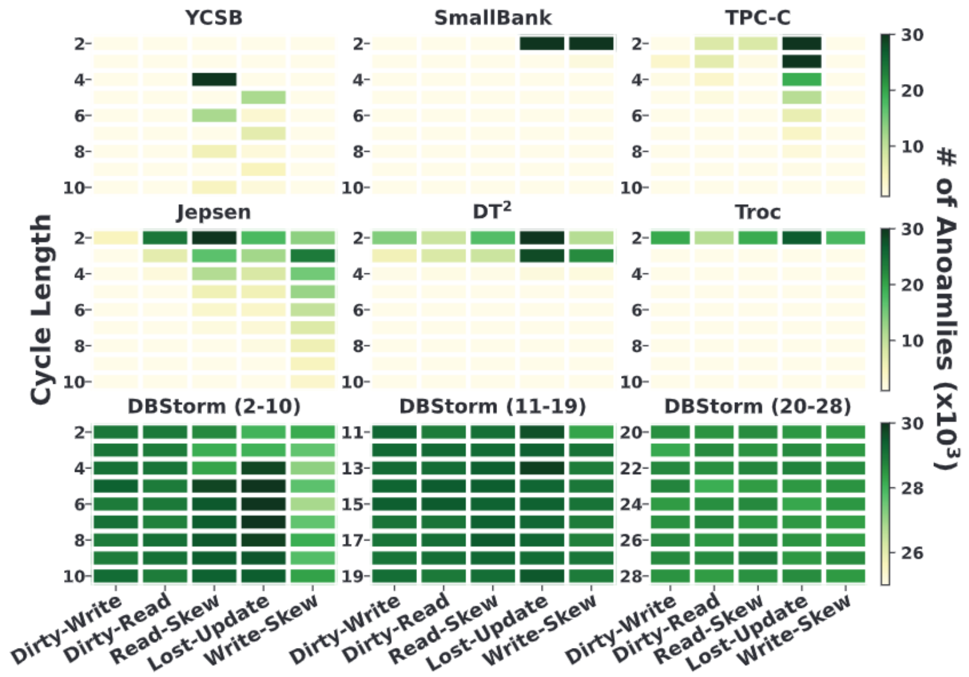


Figure 10: Anomaly Formation Comparison

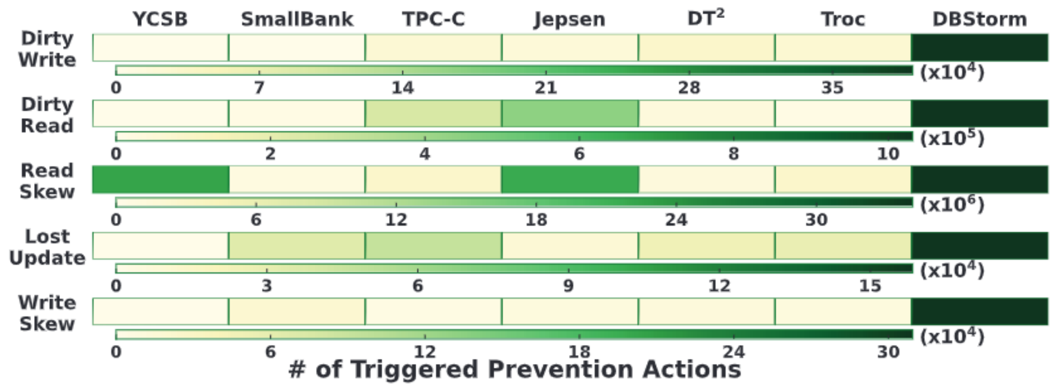


Figure 11: Anomaly Prevention Action Comparison

Table 2: Bug Comparison

Total/Confirmed/Fixed	DBStorm	TxCheck	Jepsen	DT^2
Isolation	20/12/6	7/6/0	22/21/13	4/2/1
Non-isolation	13/12/10	49/46/18	33/33/20	12/10/0
Total/Confirmed/Fixed	**Troc**	**SQLsmith**	**SQLancer**	**Squirrel**
Isolation	5/5/1	0/0/0	0/0/0	0/0/0
Non-isolation	7/7/0	248/236/183	449/445/415	63/63/52