The LDBC Financial Benchmark

19th TUC @ London

Shipeng Qi

(with contributions from members of the FinBench Task Force)

About me

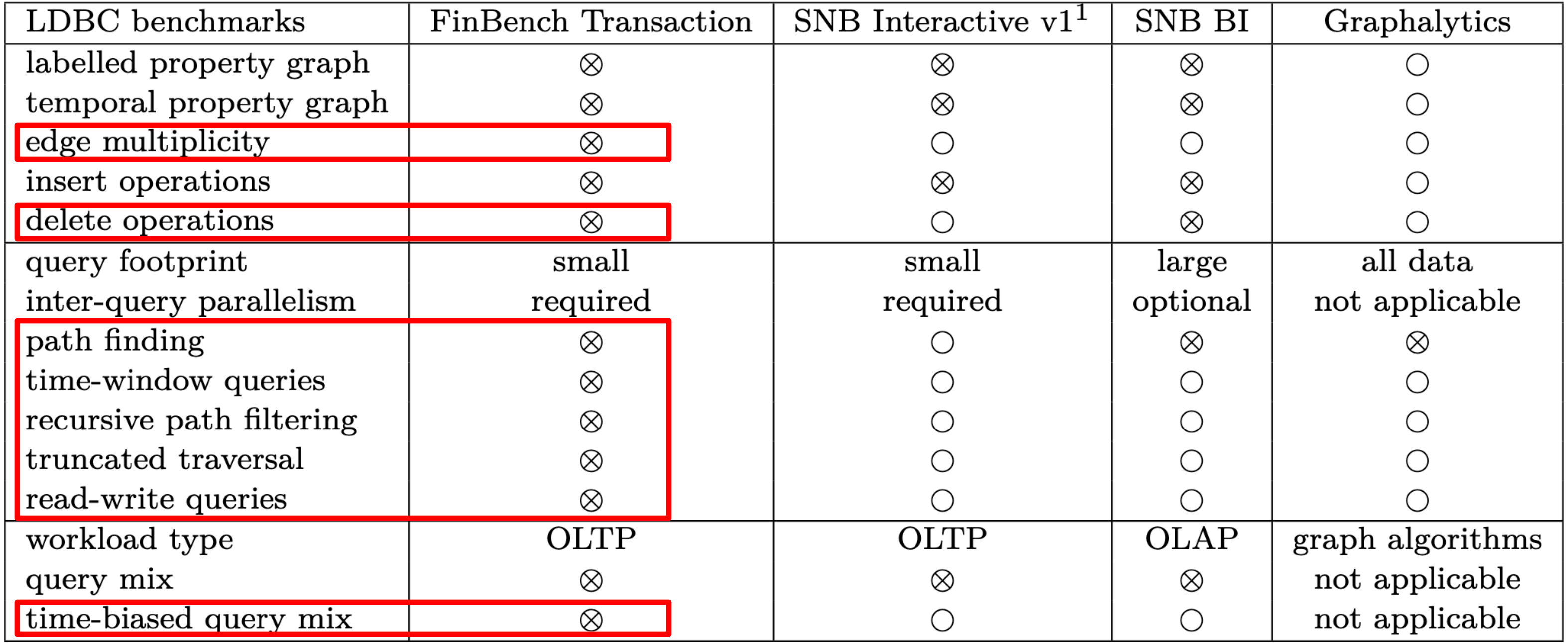
- Former Chief Product & Development Manager @ Startup
- LDBC FinBench Leader， LDBC Board Director
- Technical Expert in TuGraph team @ Ant Group
- TuGraph R&D and Open Source Develop Relationship Advocate

Motivation

Graph databases play a **pivotal role** in the FinTech industry. However, existing ones **fail to capture the unique characteristics** of financial datasets and workloads, making them **inadequate** for evaluating graph databases in financial scenarios.

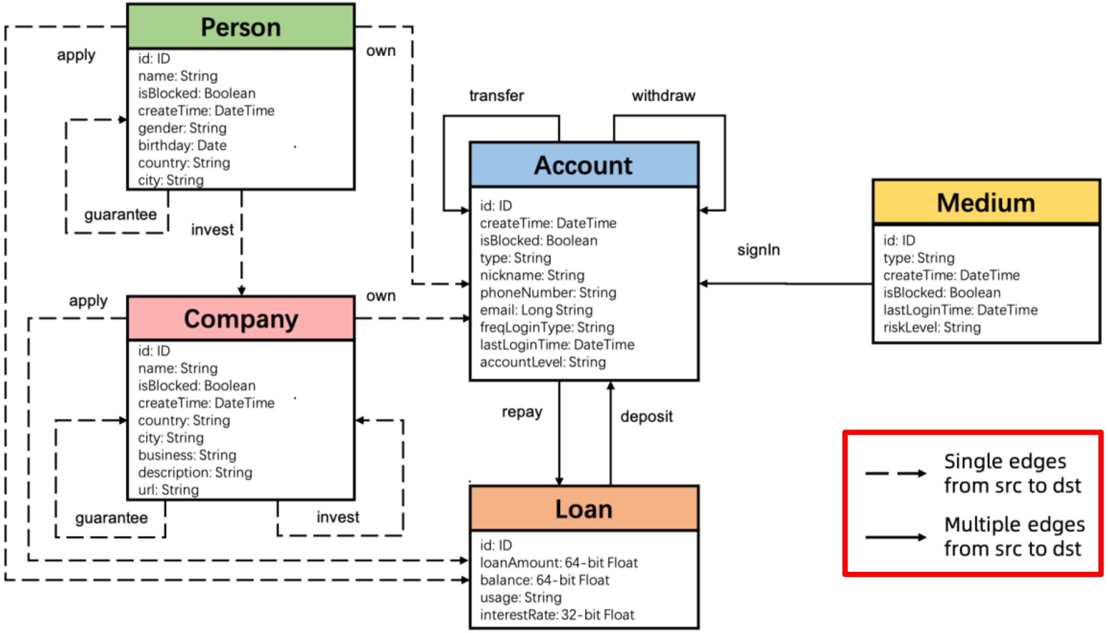
LDBC Financial Benchmark (FinBench) is a novel benchmark that adopts a **choke point-driven design methodology**, emphasizing performance bottlenecks, and incorporates distinct features.

New Features in FinBench



LDBC benchmarks	FinBench Transaction	SNB Interactive v1[1]	SNB BI	Graphalytics
labelled property graph	⊗	⊗	⊗	○
temporal property graph	⊗	⊗	⊗	○
edge multiplicity	⊗	○	○	○
insert operations	⊗	⊗	⊗	○
delete operations	⊗	○	⊗	○
query footprint	small	small	large	all data
inter-query parallelism	required	required	optional	not applicable
path finding	⊗	○	⊗	⊗
time-window queries	⊗	○	○	○
recursive path filtering	⊗	○	○	○
truncated traversal	⊗	○	○	○
read-write queries	⊗	○	○	○
workload type	OLTP	OLTP	OLAP	graph algorithms
query mix	⊗	⊗	⊗	not applicable
time-biased query mix	⊗	○	○	not applicable

Dataset Schema



Person
id: ID
name: String
isBlocked: Boolean
createTime: DateTime
gender: String
birthday: Date
country: String
city: String

Account
id: ID
createTime: DateTime
isBlocked: Boolean
type: String
nickname: String
phoneNumber: String
email: Long String
freqLoginType: String
lastLoginTime: DateTime
accountLevel: String

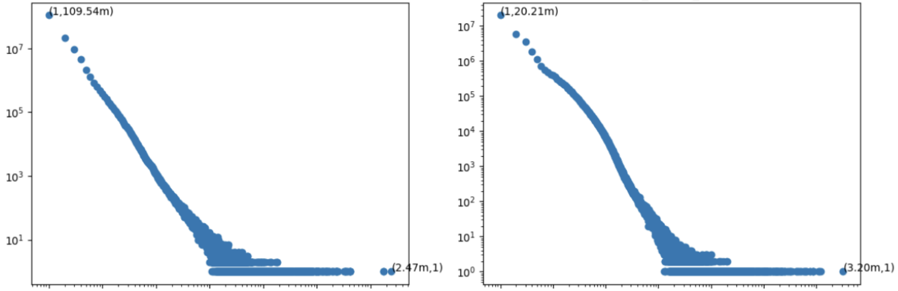
Medium
id: ID
type: String
createTime: DateTime
isBlocked: Boolean
lastLoginTime: DateTime
riskLevel: String

Company
id: ID
name: String
isBlocked: Boolean
createTime: DateTime
country: String
city: String
business: String
description: String
url: String

Loan
id: ID
loanAmount: 64-bit Float
balance: 64-bit Float
usage: String
interestRate: 32-bit Float

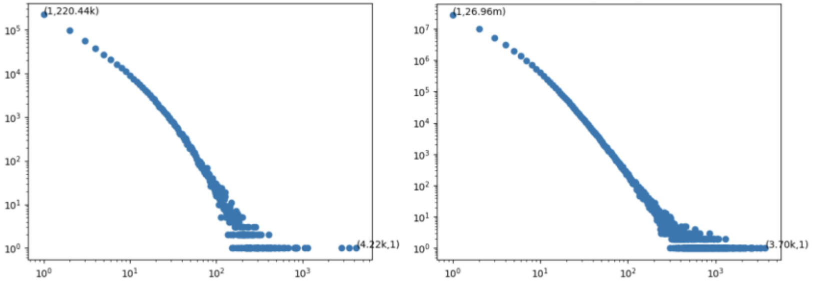
- Vertices are entities in financial systems, while edges are activities involving them
- Asymmetric dynamic temporal graph

Dataset Distribution



(a) Subgraph 1, in-degree **(b) Subgraph 1, out-degree**

Skewness in degree distribution following PowerLaw

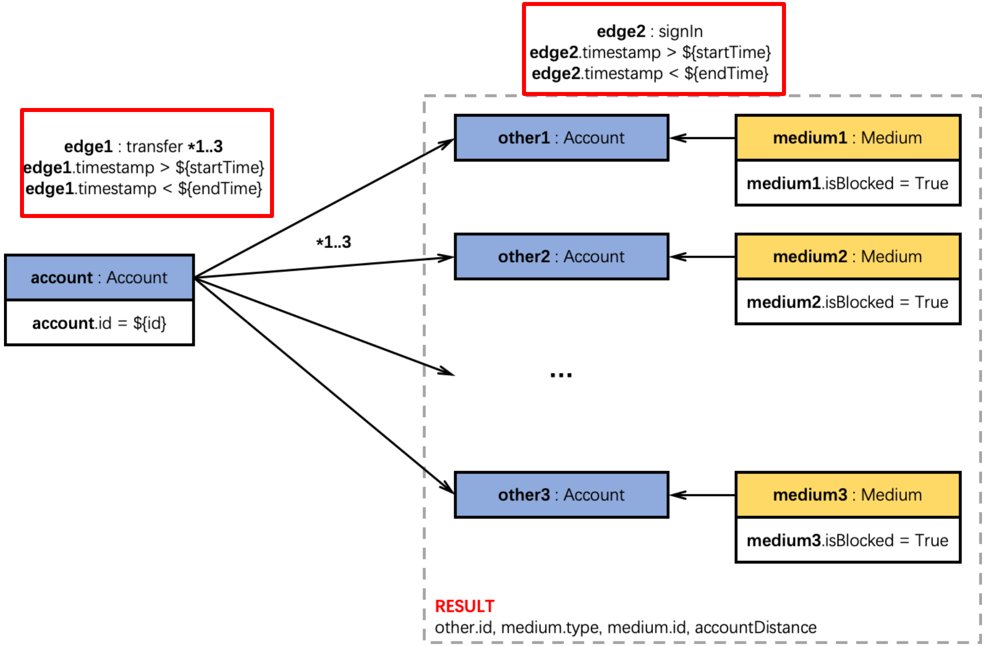


(a) In-edge multiplicity **(b) Out-edge multiplicity**

Skewness in multiplicity distribution following PowerLaw

Query: Time Window Filtering

- Fact: queries only look back in a limited time window
- Filtering: filter edges between ***startTime*** and ***endTime*** in traversal
- **Chokepoint**: temporal access locality and performance



edge2 : signIn
edge2.timestamp > ${startTime}
edge2.timestamp < ${endTime}
edge1 : transfer ***1..3**
edge1.timestamp > ${startTime}
edge1.timestamp < ${endTime}
***1..3**
account : Account
account.id = ${id}
other1 : Account
other2 : Account
...
other3 : Account
medium1 : Medium
medium1.isBlocked = True
medium2 : Medium
medium2.isBlocked = True
medium3 : Medium
medium3.isBlocked = True
RESULT
other.id, medium.type, medium.id, accountDistance

Blocked medium related accounts
[Ref: Transaction Complex Read 1]

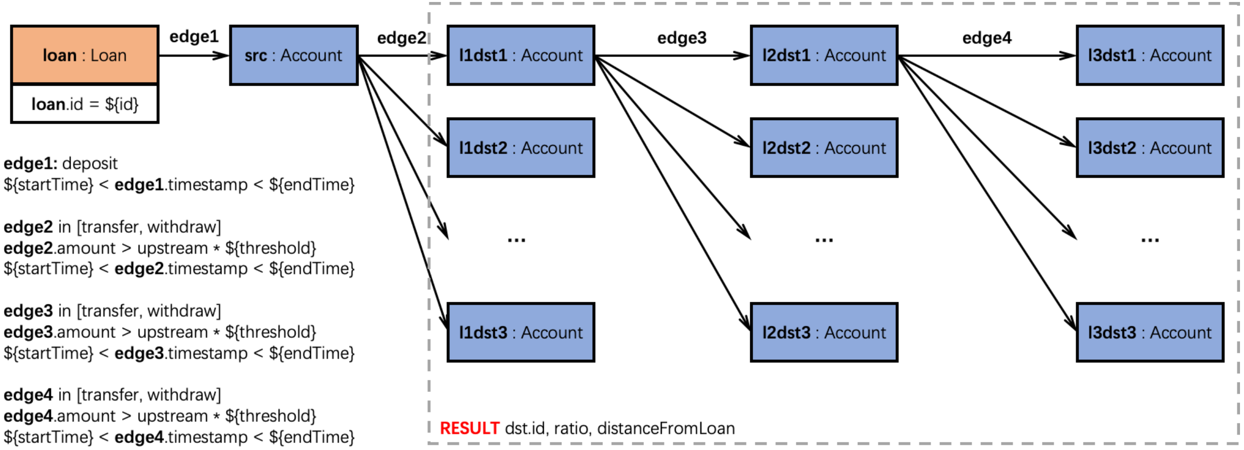
Query: Recursive Path Filtering

Assuming: A -[e1]-> B -[e2]-> ... -> X

- Timestamp order: e1 < … < ei
- Amount order: e1 > … > ei

Chokepoints:

- filter push-down in RPQs for monotonic orders
- language expression



loan : Loan

loan.id = ${id}

edge1

src : Account

edge2

l1dst1 : Account

l1dst2 : Account

...

l1dst3 : Account

edge3

l2dst1 : Account

l2dst2 : Account

...

l2dst3 : Account

edge4

l3dst1 : Account

l3dst2 : Account

...

l3dst3 : Account

edge1: deposit
${startTime} < **edge1**.timestamp < ${endTime}

edge2 in [transfer, withdraw]
edge2.amount > upstream * ${threshold}
${startTime} < **edge2**.timestamp < ${endTime}

edge3 in [transfer, withdraw]
edge3.amount > upstream * ${threshold}
${startTime} < **edge3**.timestamp < ${endTime}

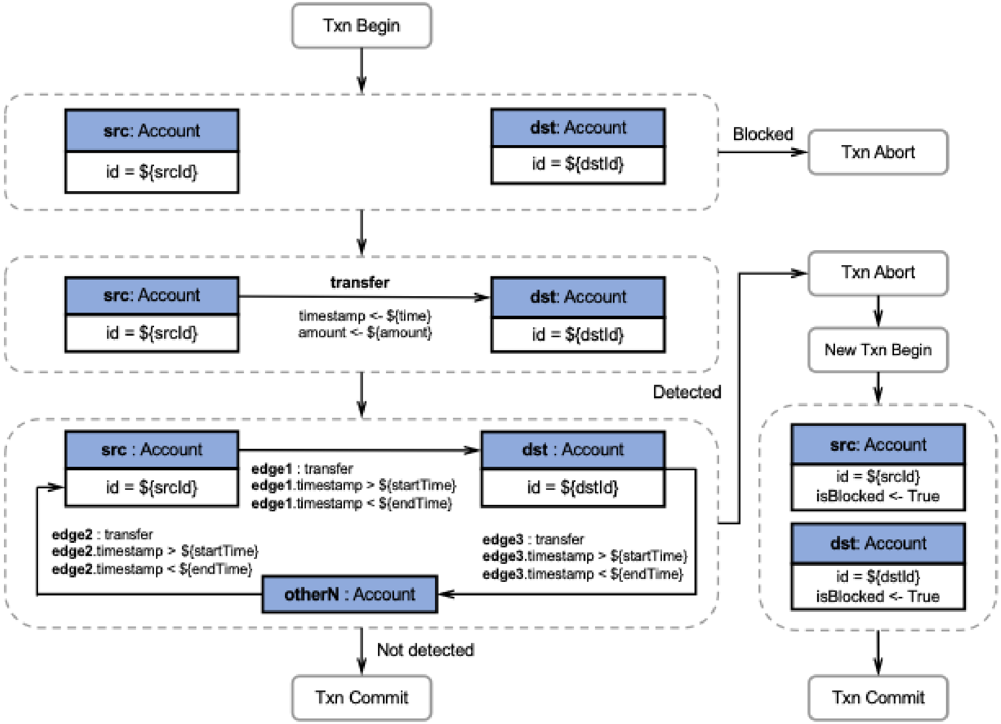
edge4 in [transfer, withdraw]
edge4.amount > upstream * ${threshold}
${startTime} < **edge4**.timestamp < ${endTime}

RESULT dst.id, ratio, distanceFromLoan

Transfer trace after loan applied
[Ref: Transaction Complex Read 8]

Query: Read-Write Query

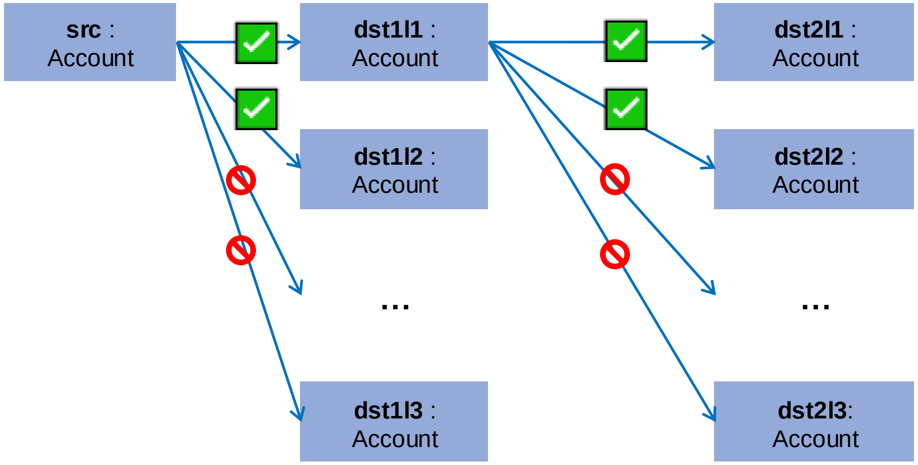
- Complex read query: it represents a risk control strategy
- RW query: Transaction-wrapped complex reads (risk control strategy)
- If the risk control strategy is not hit, transaction commits with write query. Otherwise, transaction aborts
- **Chokepoint**: write operation contention and conflicts



Transfer under transfer cycle detection strategy
[Ref: Transaction Read Write 1]

Query: Truncation

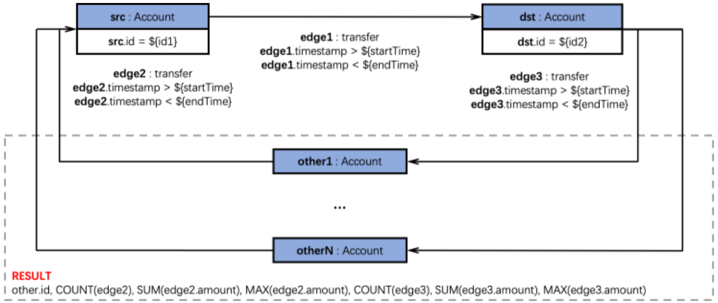
- In practice, system optimization cannot keep up with the increase of the workload complexity
- Truncate less-important edges to avoid complexity explosion, which is actually sampling.
- TruncationLimit and TruncationOrder is defined to ensure consistency of results.
- For example, keep only the top 100 edges in order of timestamp descending



Chokepoints:

- truncation push-down
- language expression

Query: Patterns in temporal graph



src : Account

src.id = ${id1}

dst : Account

dst.id = ${id2}

edge1 : transfer
edge1.timestamp > ${startTime}
edge1.timestamp < ${endTime}

edge2 : transfer
edge2.timestamp > ${startTime}
edge2.timestamp < ${endTime}

edge3 : transfer
edge3.timestamp > ${startTime}
edge3.timestamp < ${endTime}

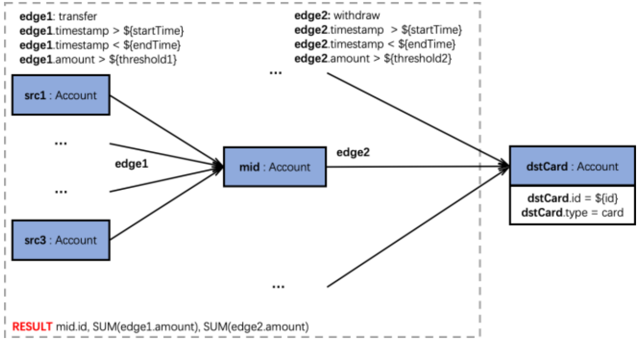
other1 : Account

...

otherN : Account

RESULT
other.id, COUNT(edge2), SUM(edge2.amount), MAX(edge2.amount), COUNT(edge3), SUM(edge3.amount), MAX(edge3.amount)

Cycle
[Ref: Transaction Complex Read 4]



edge1: transfer
edge1.timestamp > ${startTime}
edge1.timestamp < ${endTime}
edge1.amount > ${threshold1}

edge2: withdraw
edge2.timestamp > ${startTime}
edge2.timestamp < ${endTime}
edge2.amount > ${threshold2}

src1 : Account

...

edge1

...

src3 : Account

mid : Account

edge2

...

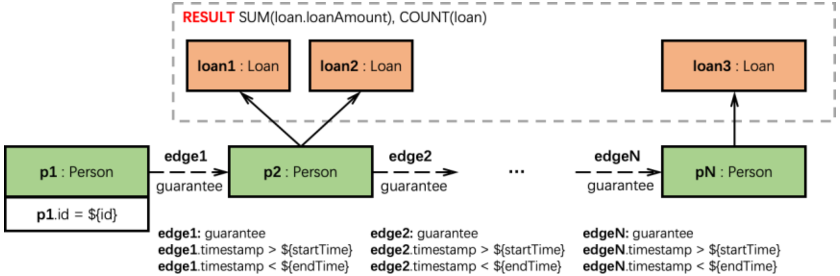
...

dstCard : Account

dstCard.id = ${id}
dstCard.type = card

RESULT mid.id, SUM(edge1.amount), SUM(edge2.amount)

Tree
[Ref: Transaction Complex Read 6]



RESULT SUM(loan.loanAmount), COUNT(loan)

loan1 : Loan

loan2 : Loan

loan3 : Loan

p1 : Person

p1.id = ${id}

edge1
guarantee

p2 : Person

edge2
guarantee

...

edgeN
guarantee

pN : Person

edge1: guarantee
edge1.timestamp > ${startTime}
edge1.timestamp < ${endTime}

edge2: guarantee
edge2.timestamp > ${startTime}
edge2.timestamp < ${endTime}

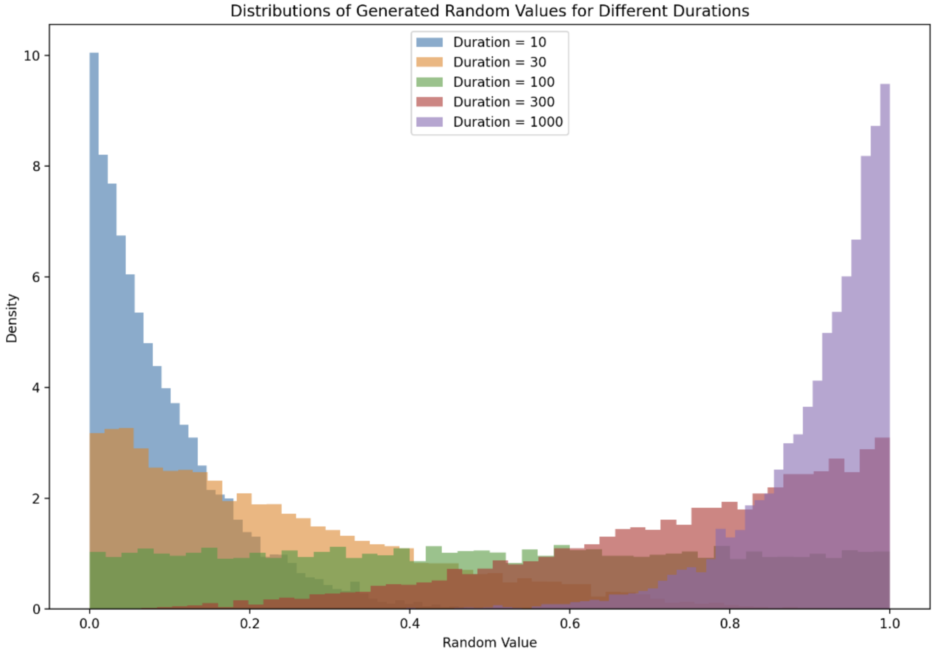
edgeN: guarantee
edgeN.timestamp > ${startTime}
edgeN.timestamp < ${endTime}

chain
[Ref: Transaction Complex Read 11]

Workload: Time-biased query mix

$$Prob(x) = 1 - U^{k \cdot t}$$

Longer temporal window in complex read queries results in more simple read queries following



The probability density distribution of the time-biased function

Future work

- Paper submitted. Working on the new version v0.2 release
- Ongoing Implementation: TigerGraph, gpStore(gStore team), Pyrrho
- Official audits planned in 2025
- New features planned: auto-benchmarking, htap metrics (freshness), new workload for AP

LDBC

Thanks!

ANT GROUP

创邻科技 CREATE LINK

NebulaGraph

海致星图

ULTIPA

intel

KATANA GRAPH

TigerGraph

pometry

Acknowledgement

Developers: Shipeng Qi, Bing Tong, Jiatao Hu, Bin Yang, Changyuan Wang

Collaboration: Tao Lv@CSTC, Prof. Lei Zou@PKU, Prof. Malcome Crowe, Prof. Qiang Yin@SJTU

Welcome collaboration on benchmark and research on chokepoints

Contact me at **shipeng.qi AT ldbcouncil.org**

WeChat QR Code

LDBC

The graph & RDF
benchmark reference