Team24
Final Report

106062216 馮謙
106062116 黃晨
106034061 曾靖渝

Target: Throughput

Contents

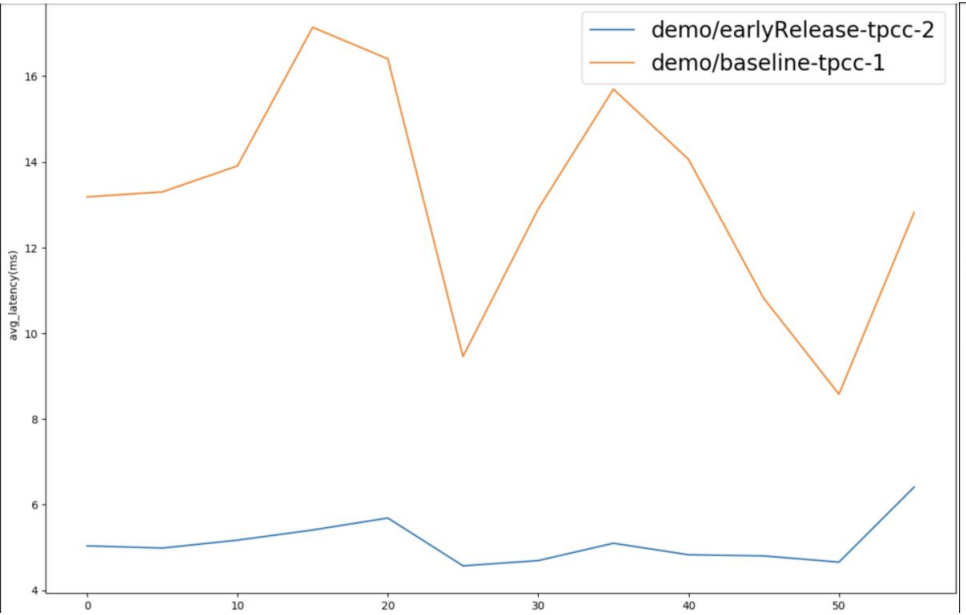
Experiments Method

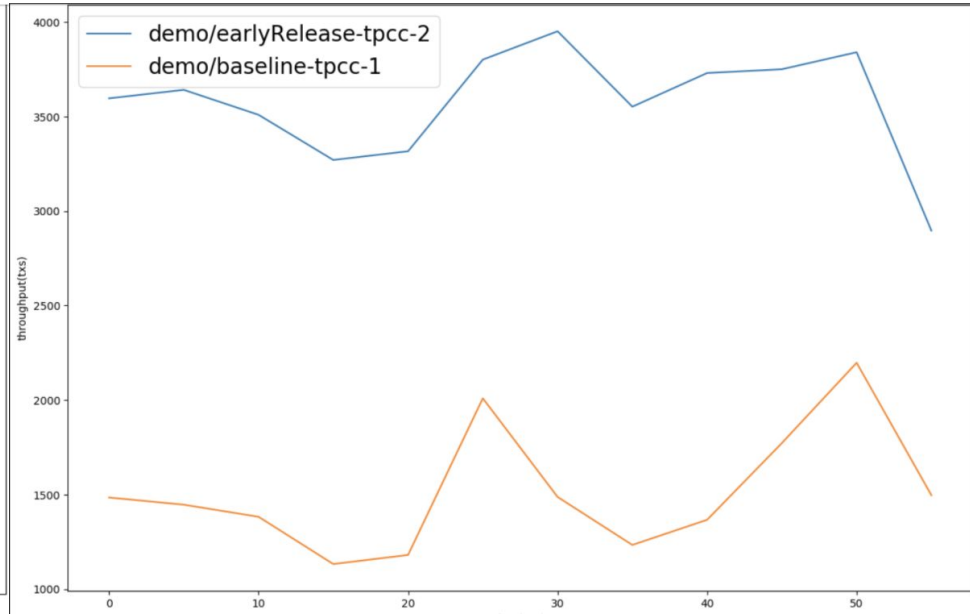
1. Test each case 3 times.
2. Compare the variance(Eliminate outlier)
3. Choose the median(Throughput)
4. Deviation 5%

Improvement of Early Release

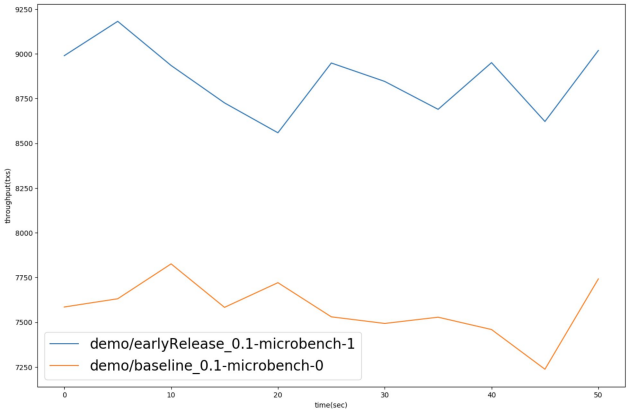
Early Release

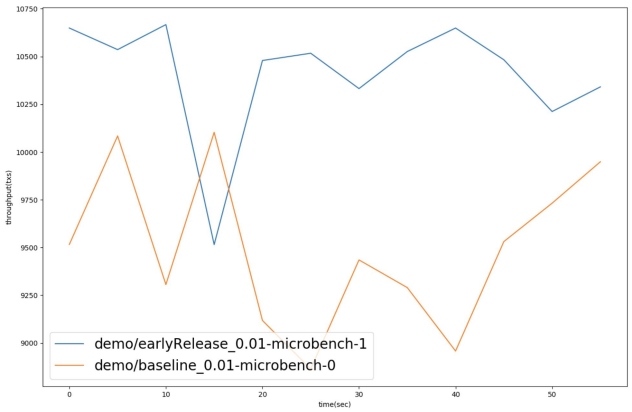
- Traditional: Release locks when transactions commit
  - 2PL -> S2PL (due to Cascading Rollback)
- Now: Calvin
  - No Abort -> Early Release

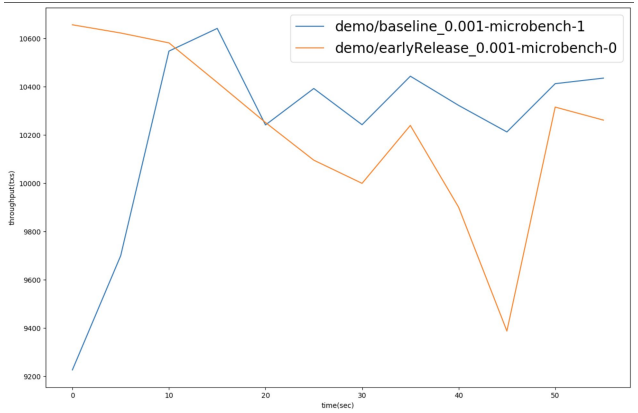




Early Release	Improvement(Commits)
Throughput	42857/18185=235.6%
Latency	6(ms)/12(ms) = 50%







Conflict Rate	EarlyRelease(commits)	Baseline(commits)	Improvement(commits)
0.1	106315	90144	106315/90144=117.9%
0.01	124907	113888	124907/113888=109.6%
0.001	122736	121826	122736/121826=107.4%

Improvement of Cache

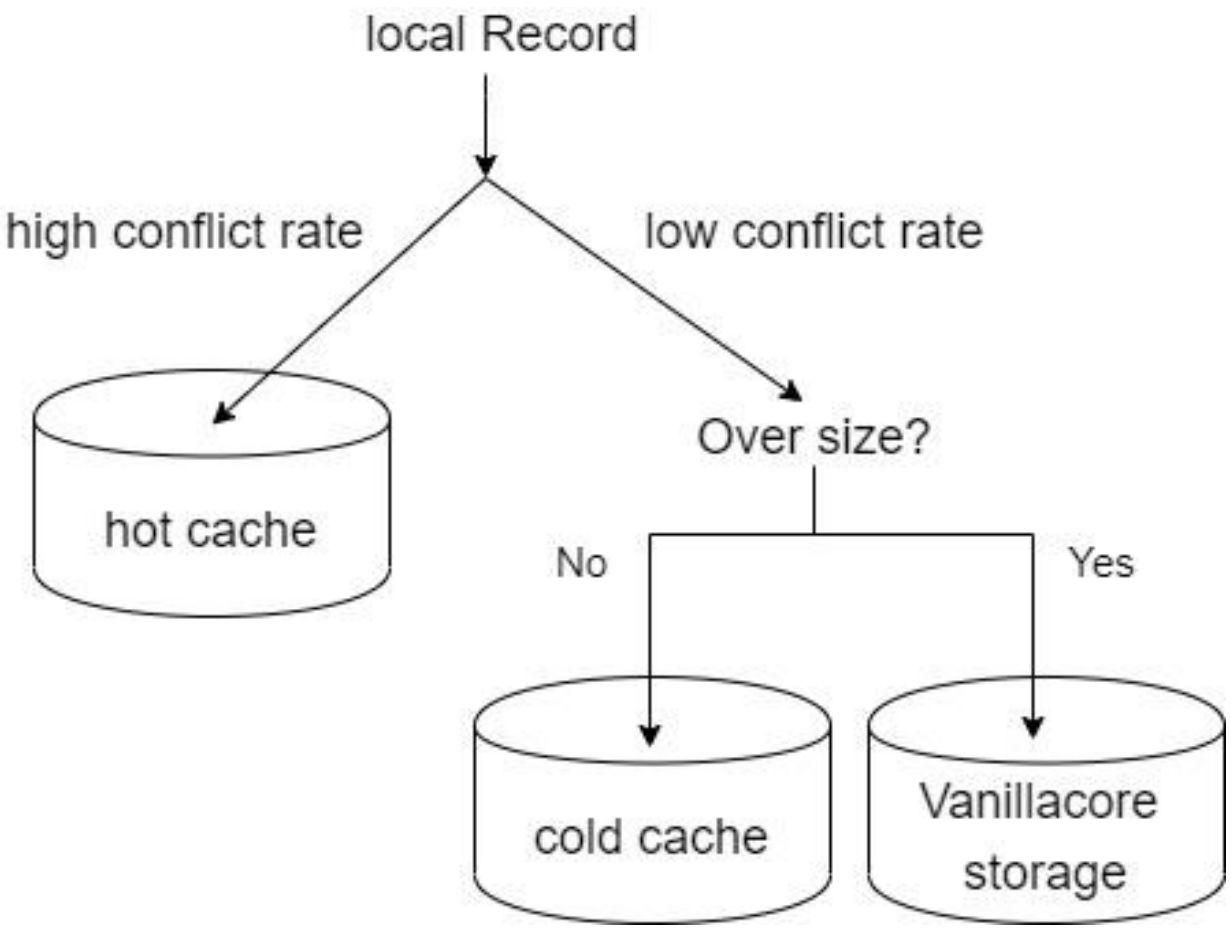
Locality?

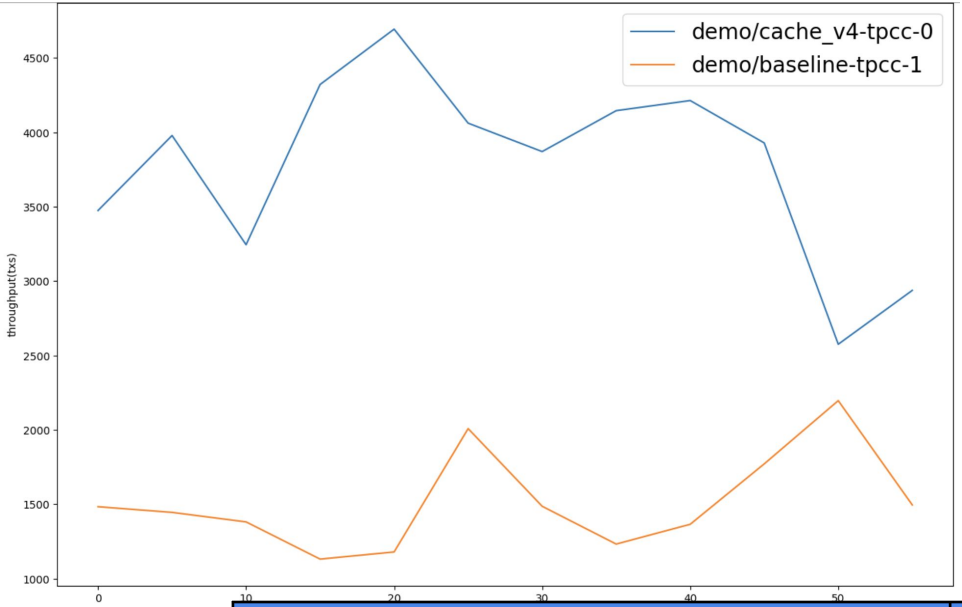
- Micro: conflict rate
- TPCC: not obvious

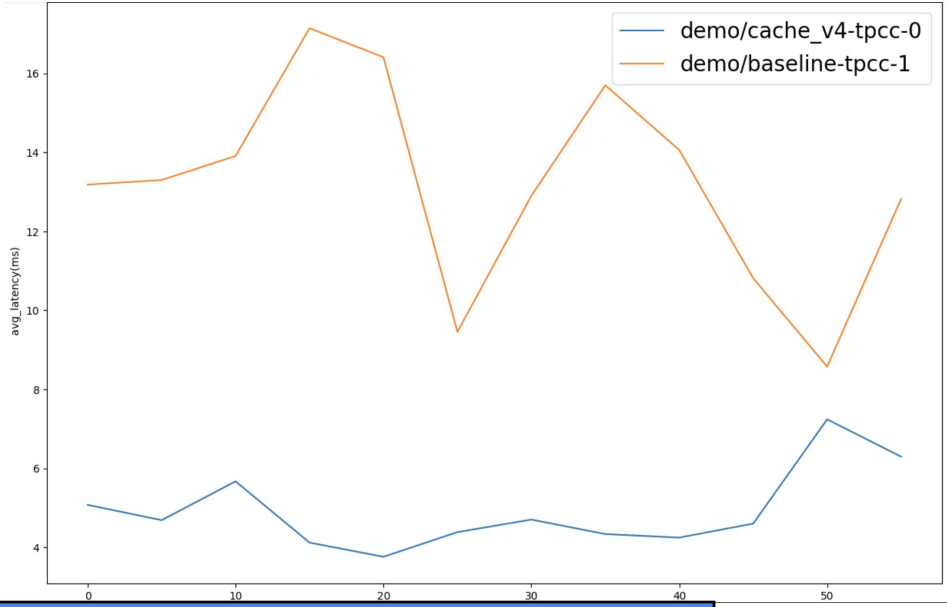
TPCC table size

TableName	Size
warehouse 🔥	W (property)
district 🔥	w*10
stock、item	100000
customer、history 、order	w*10*3000

Cache


local Record
high conflict rate
low conflict rate
hot cache
Over size?
No
Yes
cold cache
Vanillacore storage





Cache	Improvement(Commits)
Throughput	45453/18185=249.9%
Latency	5(ms)/12(ms) = 41.6%

Implementation of Delayed Updates

Concepts

- In traditional DBMS, transactions are executed eagerly.
- However, there’re something that can be executed later.
- Ex: the updates that do not have data dependency with others.

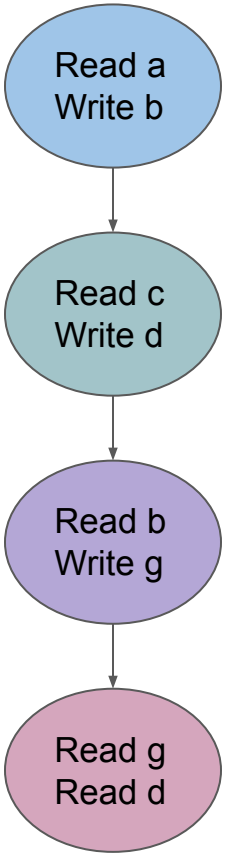
Reference: Lazy Evaluation of Transactions in Database Systems

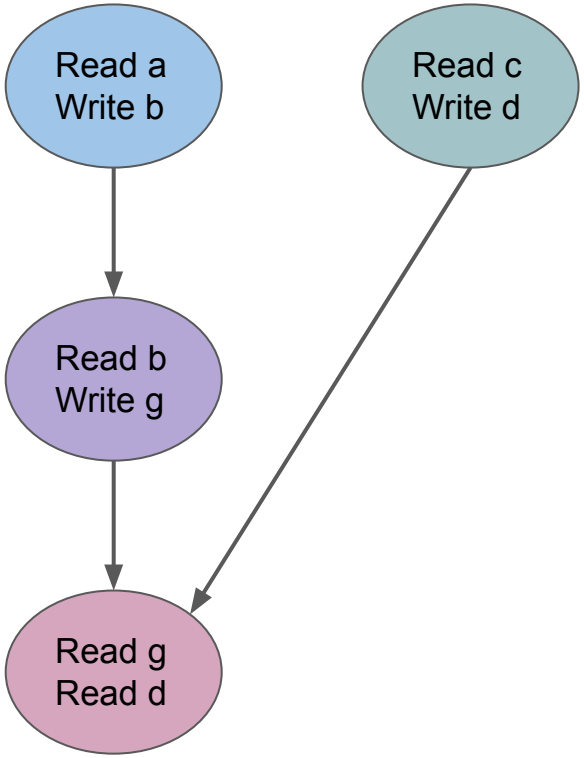
Trade-off

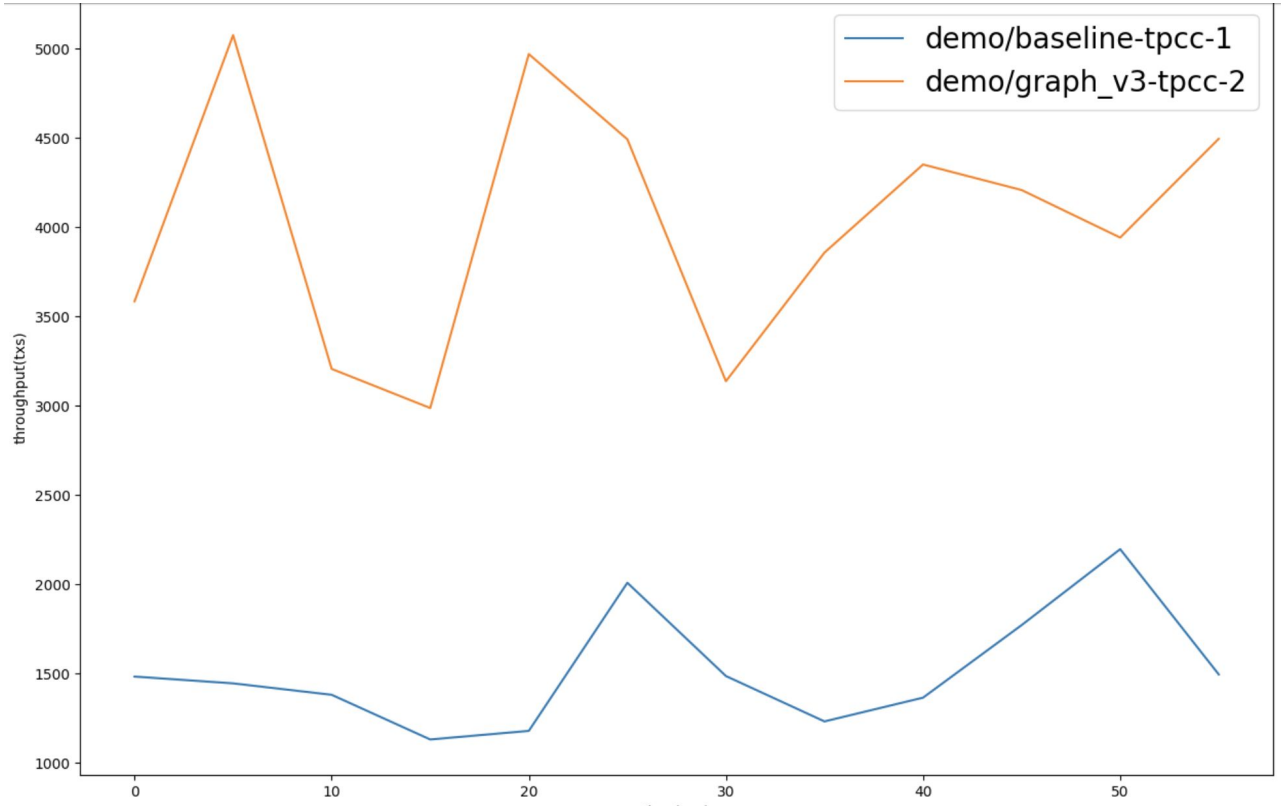
- Advantages
  - Reduce latency
  - Increase throughputs
  - Eliminate redundant updates
- Downsides
  - Higher read latencies
  - Overhead of determining the write set of a transaction

Implementation

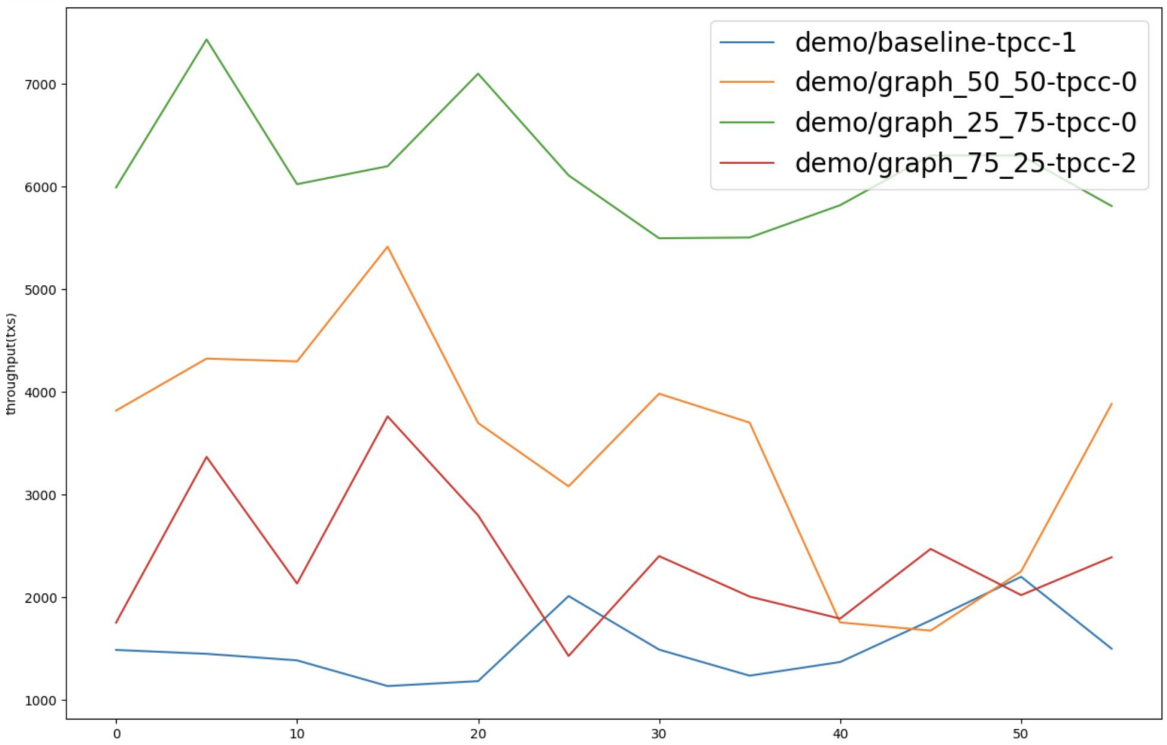
1. Build a directed graph
2. Execute the vertex whose indegree is zero
3. Remove data dependency
4. Back to 2.







Delayed-Update	Improvement(Commits)
Throughput	48308/18185=265.6%
Latency	5(ms)/12(ms) = 41.6%



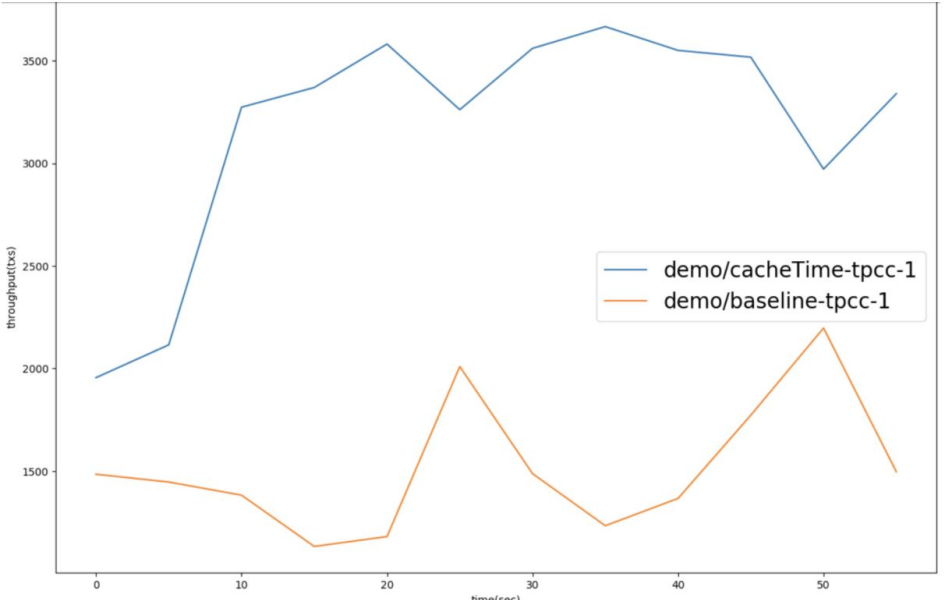
New Order : Payment	Throughput(Commits)	Latency(ms)
75:25	28301/18185 = 155.6%	8/12 = 66.6%
50:50	41887/18185 = 230.3%	6/12 = 50%
25:75	74090/18185 = 407.4%	3/12 = 25%

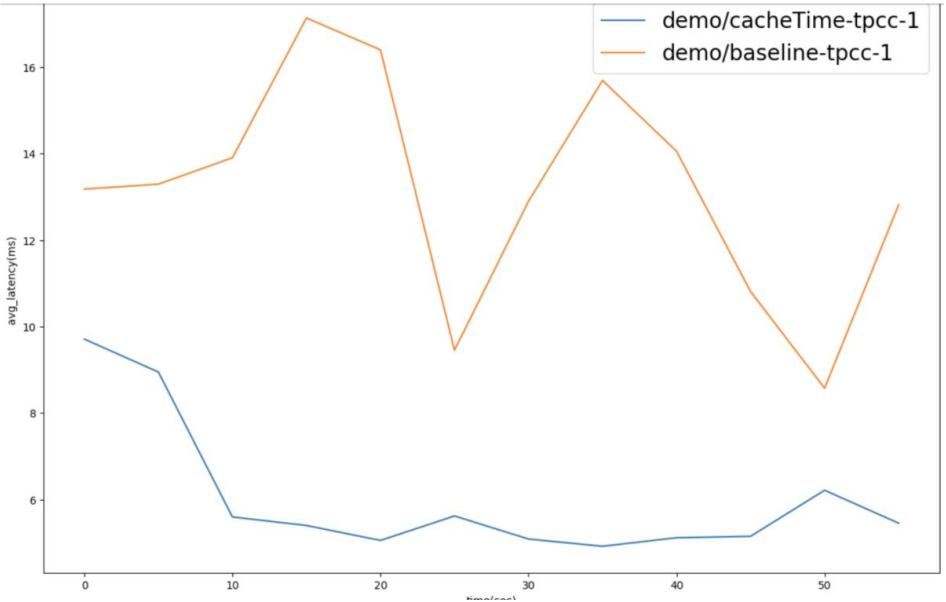
Improvement of Cache Time

Cache Time

- Follow the hint from TAs
- Frequency of data transmit

```
// Check every 1 second (fast enough?)
HoldPackage pack = newPacks.poll(1, TimeUnit.SECONDS);
if (pack != null && !handoverToTransaction(pack.txNum, pack.reco
    pending.add(pack);
}
```



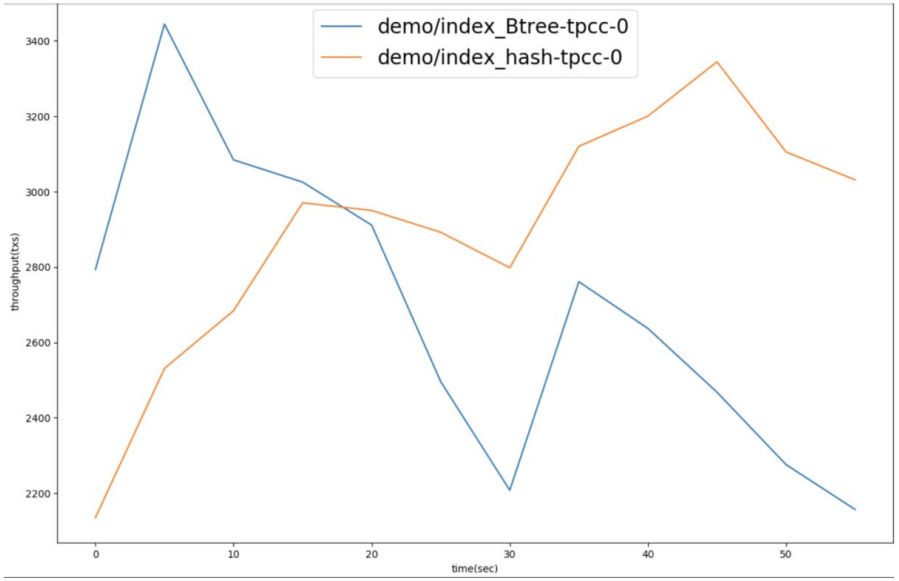


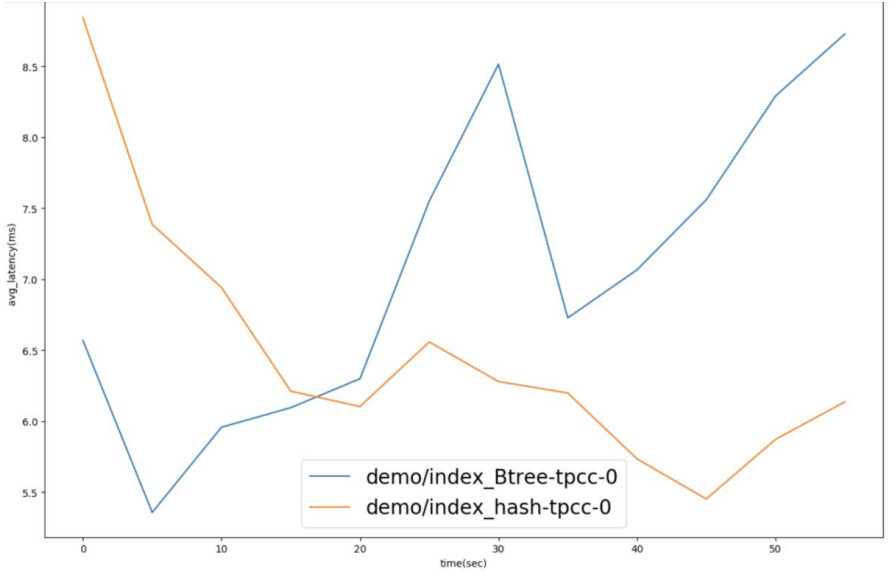
Cache Time	Improvement(Commits)
Throughput	38172/18185=209.9%
Latency	6(ms)/12(ms) = 50%

Improvement of hash bucket

Hash Index

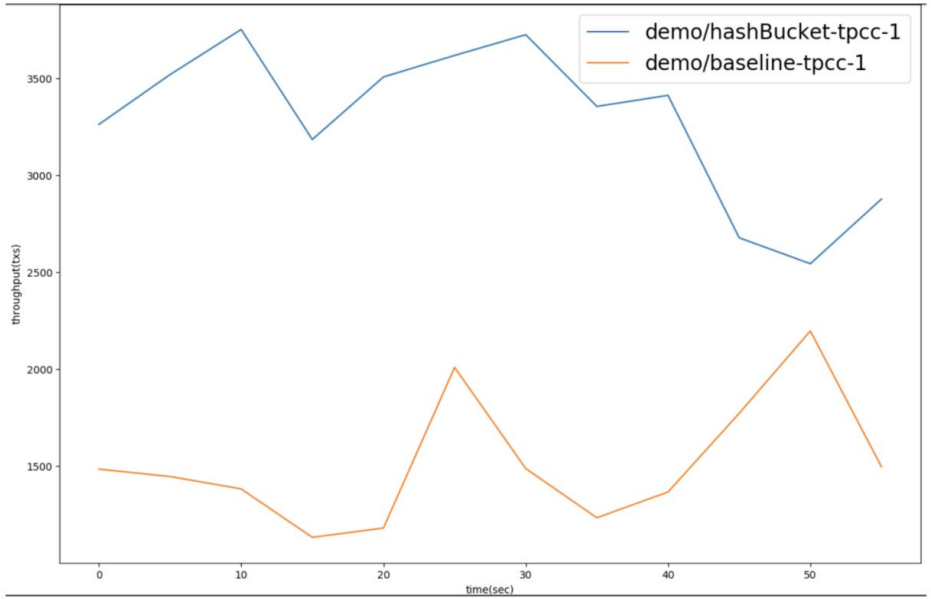
- Compare the performance fo Hash & Btree

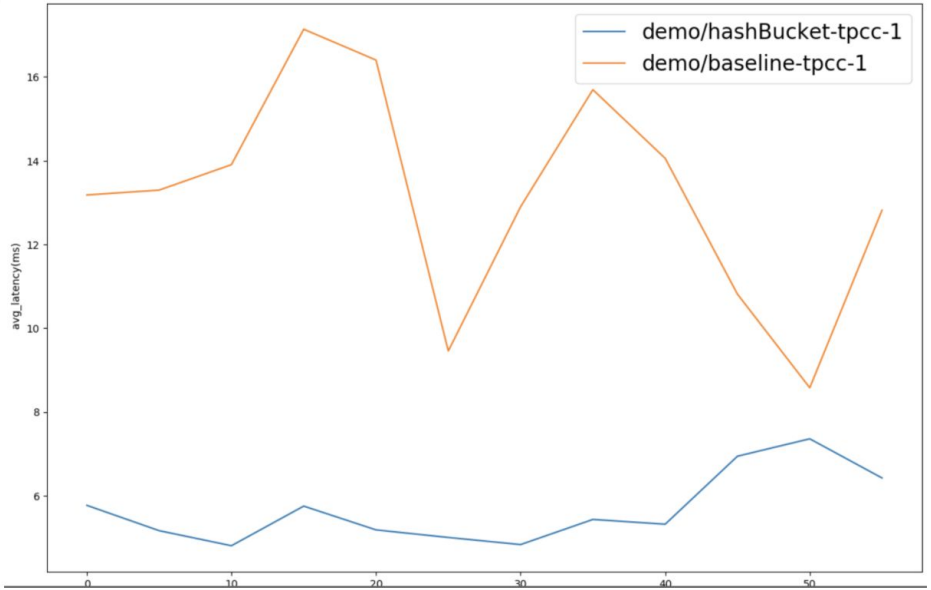




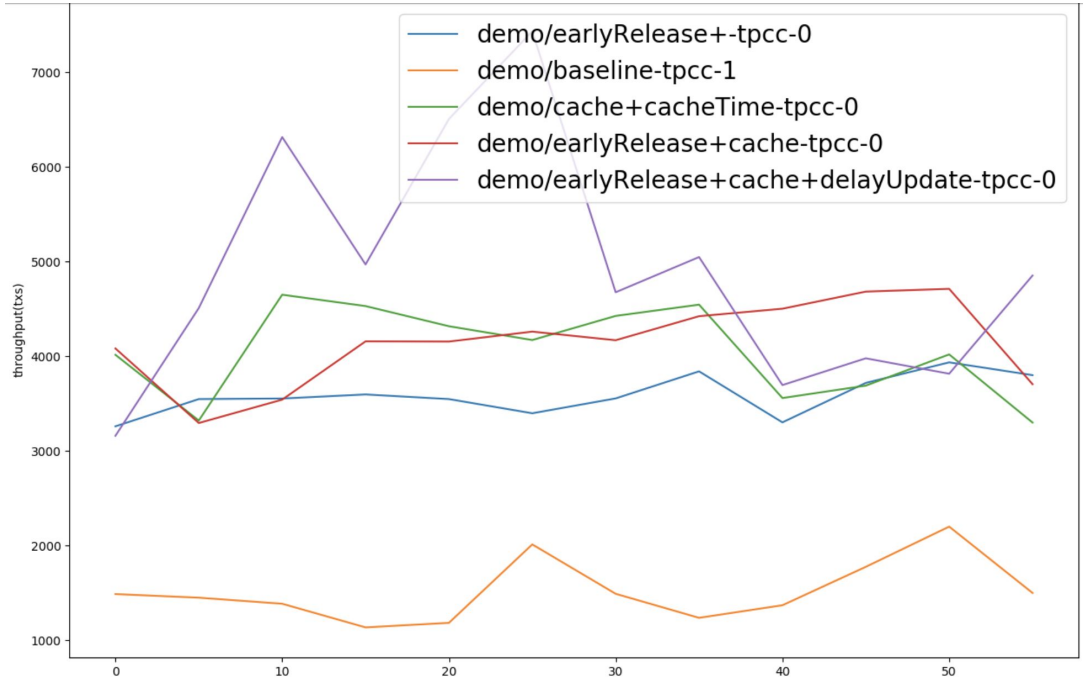
Hash Index

- Test Case(Query): Assign(=) Predicate
- Btree: Search
- Hash: If Bucket Number is large -> O(1)





Hash bucket	Improvement(Commits)
Throughput	39445/18185=216.9%
Latency	6(ms)/12(ms) = 50%



Version	Throughput(Commits)	Latency(ms)
Cache+CacheTime	48531/18185 = 266.8%	5/12 = 41.6%
EarlyRelease	43029/18185 = 236.6%	6/12 = 50%
EarlyRelease+Cache	49665/18185 = 273.1%	5/12 = 41.6%
EarlyRelease+Cache+Delay-Updated	58952/18185 = 324.1%	4/12 = 33.3%

Thanks