Database Assignment 4 Report

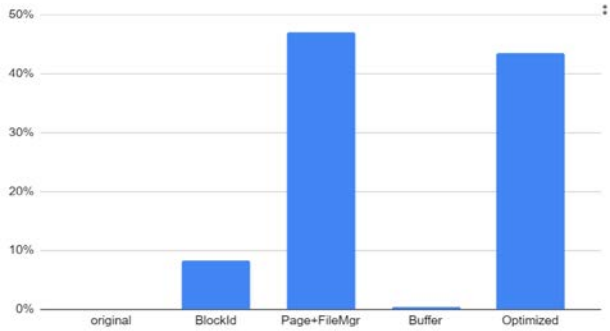
Team14 106070038 杜葳葳 106030019 吳岱容

- **Explanation**
  - **org.vanilladb.core.storage.file**
    - **BlockId:** 將 BlockId 物件先 hash 好，就不用每次都要重新 hash。
    - **FileMgr, Page:** 將 FileMgr 內的 method 的 syncronized 拿掉，轉由 Page 來做。Page 的部分，將原本用 syncronized 保護的 method 改為用 Reentrantlock 來防止多個 thread 讀寫資料出錯，同時縮小 critical section。
  - **org.vanilladb.core.storage.buffer**
    - 將 syncronized method 改為 ReentrantReadWriteLock，因爲存取 buffer 有 read 和 write 兩種方式，因此用 readlock 和 writelock 來控制，當為 read-read 時，不會鎖住（因為不會更改到 buffer 的資料），其他情況皆會鎖住。
- **Environment:** Intel Core i7-7700HQ CPU @ 2.8GHz, 8 GB RAM, 128 GB SSD
- **Each and Overall optimization**
  - 以下實驗參數設定

NUM_RTE	RW_TX_RATE	TOTAL_READ_COUNT	LOCAL_HOT_COUNT
20	0.1(improvement 處) 0.5(其他測試控制變因)	10	1
WRITE_RATIO_IN_RW_TX	HOT_CONFLICT_RATE	BUFFER_POOL_SIZE	
0.5	0.001	102400	

  - **Improvement**

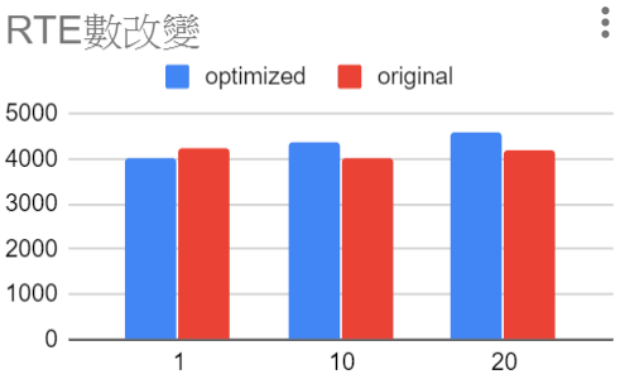
	Throughput	Avg_latency	**Speed Up**
Basic Version	15543	77	**0%**
BlockId	16831	71	**8.29%**
FileMgr+Page	22863	52	**47.10%**
Buffer	15612	77	**0.44%**
Optimized Version	22302	54	**43.49%**



由此看來，主要的優化成果可能來自 FileMgr 與 Page 的處理。

- **Optimize Parameter**
  - NUM_RTES

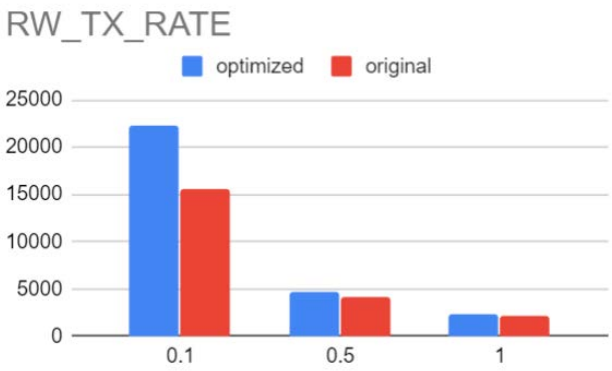
NUM_RTES	**1**	**10**	**20**
Basic Version	4226	4017	4182
Optimized Version	3999	4379	4608
Speed Up	**-5.37%**	**9.01%**	**10.19%**



NUM_RTES 越高，可同時執行的 thread 越多，因此 throughputs 越高，雖然 RTE=1 時優化後效能較差，但可以看到 RTE 增加有上升的趨勢，因為 RTE 數多、較需要去處理 thread 的問題。

- RW_TX_RATE (1.0 >= value >= 0.0)

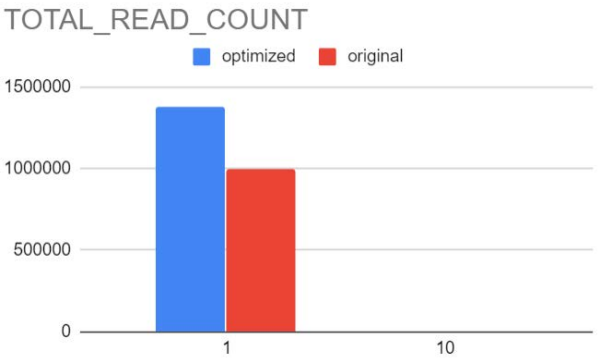
RW_TX	**0.1**	**0.5**	**1**
Basic Version	15543	4182	2166
Optimized Version	22302	4608	2387
Speed Up	**43.49%**	**10.19%**	**10.20%**



RW_TX_RATE 越高代表 Write 的 tx 越多，由 hw2 可知，read 比 write 來得快，因此 rate 越高，throughputs 越小，因為優化有使用 ReadWriteReentrantLock 做控制，因此當 read 數目較多時，效果看起來較明顯。

- TOTAL_READ_COUNT (value >= 1)

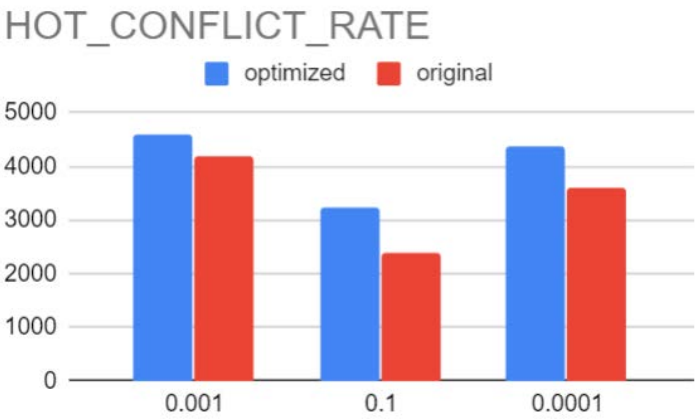
Total_Read	**1**	**10**
Basic Version	997206	4182
Optimized Version	1382419	4608
Speed Up	**38.63%**	**10.19%**



Record 數越多，throughputs 越小，優化後提升的比例也以 record 數小為佳。

- HOT_CONFLICT_RATE

HOT_CON	**0.1**	**0.001**	**0.0001**
Basic Version	2370	4182	3595
Optimized Version	3223	4608	4389
Speed Up	**36.00%**	**10.20%**	**22.90%**



附註：0.1 有很多 aborted

HOT_CONFLICT_RATE 較高的情況下，optimized 後的版本有明顯的成長，因為在 threaded 的方面有較好的解決方法。

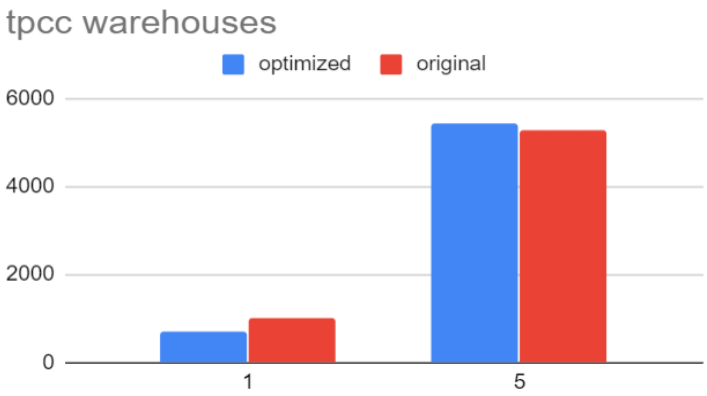
- BUFFER_POOL_SIZE

BUFFER_POOL	1024	102400
Basic Version	994	4182
Optimized Version	1029	4608
Speed Up	**3.52%**	**10.19%**

Buffer 的數量越多，throughput 越高，優化後的效能也會顯著提升。

- NUM_WAREHOUSES

NUM_WAREHOUSES	1	5
Basic Version	1042	5302
Optimized Version	717	5432
Speed Up	**-30.0%**	**2.45%**



使用 tpcc 時，優化版本還較原版差，但能看到當 warehouse 數目上升後，優化版逐漸追上，優化版本在資料量較多時可以有較好的處理。

- **Analyze and Explanation**
  - **優化方法**
    - 透過對 hashcode 做 caching，減少重複做 hash
    - 將 syncronized 用 lock 取代，同時縮小 critical section，減少其他 thread 等待的時間
    - Buffermgr 是每個 thread 獨立的，不需用 syncronized 包，然而 bufferpoolmgr 只有一個且是所有 thread 共用，需要用 lock 保護。因此，我們將不必要的 syncronized 拿掉，將 critical section 盡量縮小
  - **實驗**
    - 在各個實驗中，優化後的 throughputs 皆比原本的高，但優化的程度根據 benchmark 的特性和參數而有所不同