# Assignment 5 Report

106062137 徐郁閎、106033233 周聖諺、p123786579 王麒銘

## **Implementation**

* Planners
  + BasicQueryPlanner
  + HeuristicQueryPlanner
  + SelingerLikeQueryPlanner
  + TablePlanner
  + BasicUpdatePlanner
  + IndexUpdatePlanner

以上 Planner 自 vanilladb.properties 讀取 org.vanilladb.core.query.planner.USE\_2V2PL 的布林值，並在 createPlan 一開始依據 is2V2PL 該值與否呼叫 Transaction 的 use2V2PL() 將 Transaction 中的 is2V2PL 設為 true。  
  
另外，在新增 TablePlan 時，也將 USE\_2V2PL 的布林值傳入。

* TablePlan、TableScan、TableInfo、RecordFile

新增可傳入 is2V2PL 布林值的 Constructor，好繼續將is2V2PL 布林值傳到 RecordPage。

* RecordPage

修改 getVal() 與 setVal()，在判斷 is2V2PL 布林值為 true 後，改呼叫 Transaction 的 addGetVal()、addSetVal()，讀寫 Transaction 內 workspace 的 value，而非直接呼叫原本 currentBuff 的 getVal() 與 setVal()。

* Transaction

新增 originalVals、modifiedBuffers、correspondDoLogs 等 HashMap，以 RecordId 為 key 分別記錄對應的初始值有修改過的 buffer 與是否要新增 Log。另外新增 workspace 的 HashMap，紀錄 RecordId 對應到的數值。

因此，當 addGetVal() 被呼叫時，若 workspace 有其對應的數值則直接回傳，沒有才至 currentBuff 取值、放入workspace 中。當 addSetVal() 被呼叫時，先自currentBuff 取值、紀錄至 originalVals 中，同時將改動新增至 modifiedBuffers、workspace、correspondDoLogs 中。

當 commit() 時，呼叫 certifyLog()，對於每個 modifiedBuffers，比對 originalVals 與 workspace、檢查哪些 RecordId 有修改，並將修改的值加回原本的 Buffer、同時新增 Log（呼叫 RecoveryMgr 的 certifyLogSetVal()）。

* ConcurrencyMgr、LockTable

在 ConcurrencyMgr 新增 certifyRecord()，呼叫 LockTable 的 cLock()。

在 LockTable 中新增 cLocker，並修改其餘每個 function 的對於 C\_LOCK 的判斷，以符合 2V2PL 的 Compatibility Table。在 transaction 呼叫 commit 的時候，就會將需要

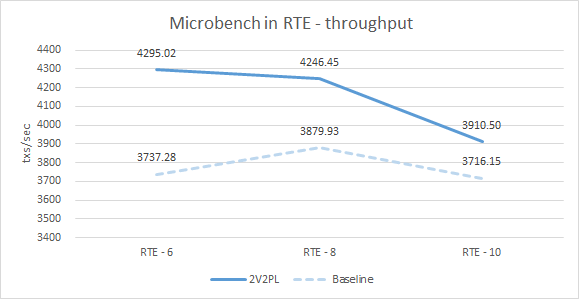
## **Environment**

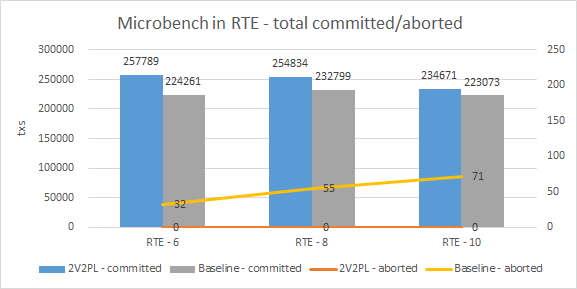
* CPU：Intel® Core™ i5-8250U Processor (6M Cache, up to 3.40 GHz)
* Operating Systems：Windows 10 Home
* RAM：8GB
* Disk：SSD 256 GB

## **EXP1: Microbench in RTE**

實驗參數：

* RTE = 6, 8, 10
* 其他為預設值





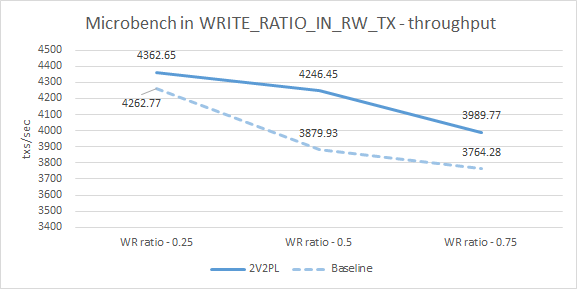
可以看到在 RTE = 8 時 baseline 的 throughput 達到尖峰，而 2V2PL 在 RTE = 6 時達到尖峰，因此選用 RTE = 8 作為實驗 2、3 (Microbench) 的基準。同時，2V2PL 明顯提升了 throughput 5% ~ 14%。同時可見，abort 的 transaction 也完全沒有。

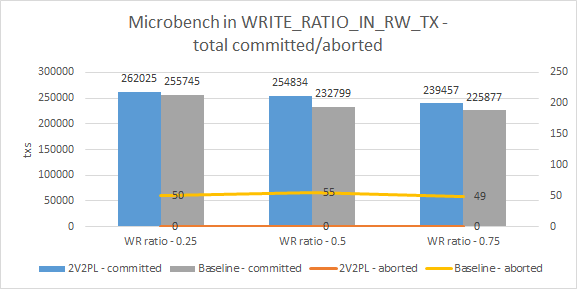
## 

## **EXP2: Microbench in WRITE\_RATIO\_IN\_RW\_TX**

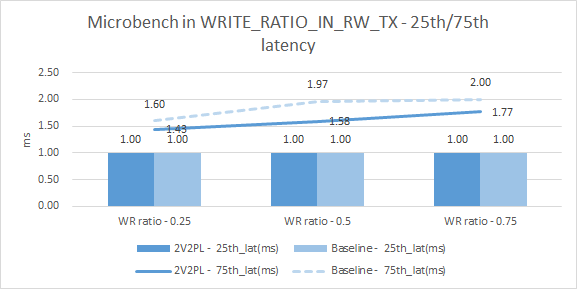
實驗參數：

* RTE = 8
* WRITE\_RATIO\_IN\_RW\_TX = 0.25, 0.5, 0.75
* 其他為預設值





從實驗數據可以看出，baseline 和 2V2PL 在 write 比例變高時，txns 皆下降，但 2V2PL 一直保持比 baseline 還高的 throughput，約高 2% ~ 9%。同時可見，abort 的 transaction 也完全消失。



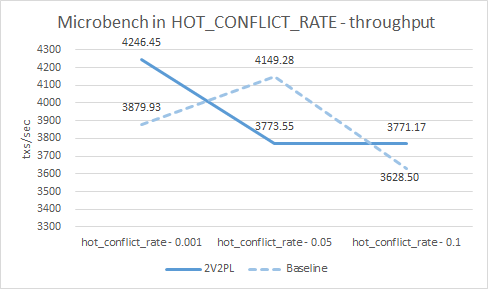
另一方面，從上圖可以看出，2V2PL 主要是優化最慢四分之一 (75 percentile) 的 transaction，相較 Baseline 的 latency 約縮短 10% ~ 20%。

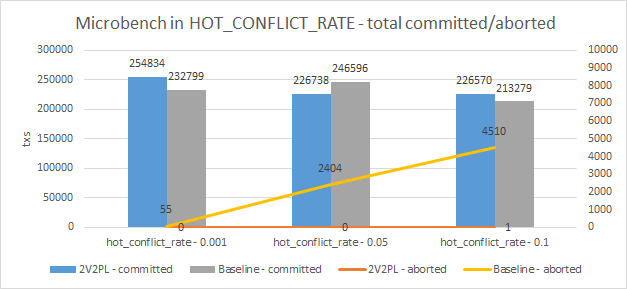
## 

## **EXP3: Microbench in HOT\_CONFLICT\_RATE**

實驗參數：

* RTE = 8
* HOT\_CONFLICT\_RATE = 0.001, 0.05, 0.1
* 其他為預設值





2V2PL 與 Baseline 在 HOT\_CONFLICT\_RATE 的 throughput 各有優劣。

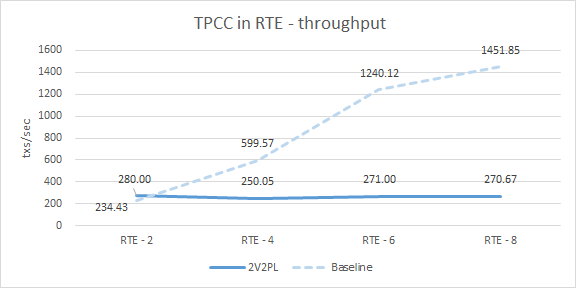
可是在 HOT\_CONFLICT\_RATE 越高的情況下，Baseline 的 aborted 數(**黃色線**) 量與 HOT\_CONFLICT\_RATE 成比例大幅上升，而 2V2PL 的 aborted 數 (**橘色線**) 幾乎不太會受到 HOT\_CONFLICT\_RATE 的影響。

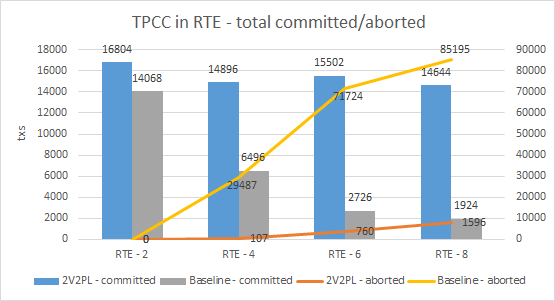
## 

## **EXP4: TPC-C in RTE**

實驗參數：

* RTE = 2, 4, 6, 8
* 其他為預設值





從上圖可看出，雖然 baseline 的 throughput 大幅勝過 2V2PL (上圖)，但絕大多數多出來的 throughput 都是 abort (下圖)，在 RTE = 8 時 baseline 的 throughput 約為 2V2PL 的 5.36 倍，然而，commit 成功的數量卻只有 2V2PL 的 13%，也就是說 **Baseline 的 transaction** **僅有 2.2% 成功 commit**。

由此可見， 2V2PL 在 RTE 增加的情況下，還是能很好的保持穩定的 throughput 以及成功的 commit，且在 RTE = 8 時 commit transaction 的數量約為 basline 的 5.36 倍。

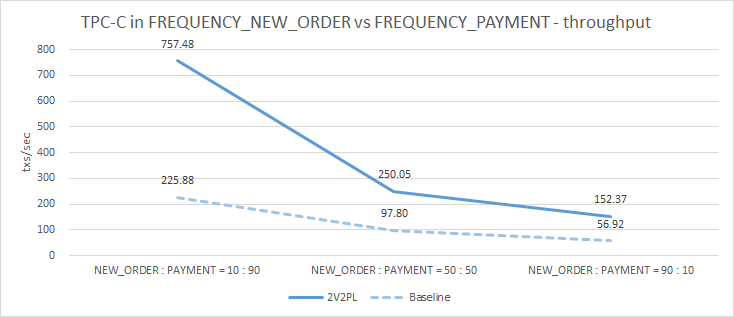
## **EXP5: TPC-C in FREQUENCY\_NEW\_ORDER vs FREQUENCY\_PAYMENT**

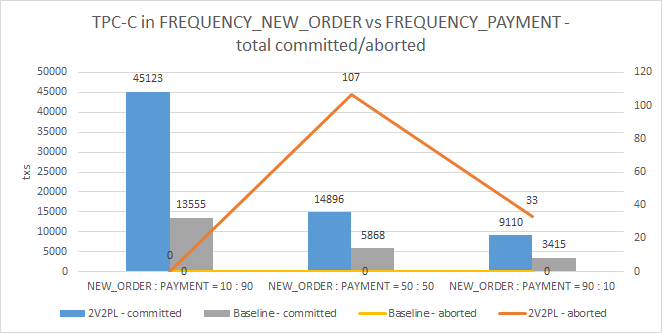
實驗參數：

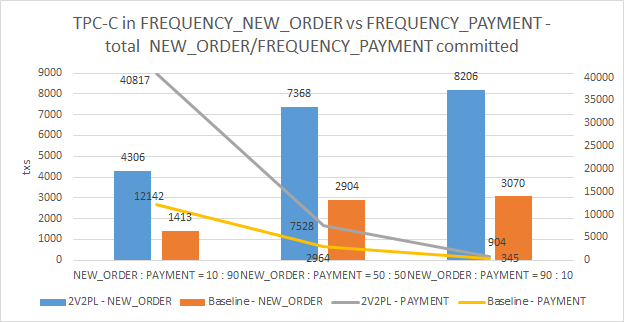
* RTE = 8
* (FREQUENCY\_NEW\_ORDER, FREQUENCY\_PAYMENT)

= (10, 90), (50, 50), (90, 10)

* 其他為預設值







從上圖可看出，無論 NEW\_ORDER 和 PAYMENT 的比例如何，2V2PL 的 throughput 和 committed transaction 都比 baseline 高三倍左右。

高下立判。