# team12_assignment2_report

106062137 徐郁閎、106033233 周聖諺、p123786579 王麒銘

## Implement

1. vanillabench.properties
    a. 加入 `READ_WRITE_TX_RATE` 與在 `As2BenchConstants` 載入值後，我們更能在 As2BenchmarkRte.java 取值。
    b. 加入 `TOTAL_UPDATE_COUNT`
2. As2UpdateItemPriceTxnProc.java / UpdateItemTxnJdbcJob.java
   實作 Assignment 要求的 SQL 任務，先依據 i_id 選取對應的 i_name、i_price 然後進行更新，然後判斷 MAX，再 UPDATE 至正確的數值。
3. As2UpdateItemPriceParamGen.java / As2UpdateItemPriceProcParamHelper.java
    a. As2UpdateItemPriceProcParamHelper.java
       新增 `getUpdateCount()`、`getUpdateCount()`、`getMinPrice()` 等 method，方便 JDBC 及 procedure 取值。
    b. As2UpdateItemPriceParamGen.java
       使用 `RandomValueGenerator()`，隨機挑選要更新的 id 與 price 增加的幅度。
4. As2BenchTransactionType.java
   新增 `UPDATE_ITEM` 的 Enum Type
5. As2BenchmarkRte.java
    a. Constructor
       新增 `updateExecutor` 與 `readExecutor`
    b. getNextTxType()
       用 Math.random() 依據機率 `READ_WRITE_TX_RATE` 隨機選擇 Transaction Type
    c. getTxExeutor()
       依據傳入的 `type`，回傳對應的 `updateExecutor` / `readExecutor`
6. As2BenchJdbcExecutor.java / As2BenchStoredProcFactory.java
   新增 `UPDATE_ITEM` 的 case，對應到 UpdateItemPrice 的 Task。

## StatisticMgr

| time (sec) | throughput (txs) | avg_latency (ms) | min (ms) | max (ms) | 25th_lat (ms) | median_lat (ms) | 75th_lat (ms) |
|---|---|---|---|---|---|---|---|
| 5 | 16557 | 308 | 0 | 1867 | 96 | 318 | 454 |
| 10 | 16437 | 311 | 0 | 2035 | 96 | 323 | 456 |
| 15 | 16655 | 305 | 0 | 1844 | 96 | 312 | 451 |
| 20 | 16370 | 313 | 0 | 1798 | 97 | 323 | 458 |
| 25 | 16399 | 311 | 0 | 2365 | 97 | 326 | 456 |
| 30 | 16579 | 308 | 0 | 1938 | 97 | 321 | 452 |
| 35 | 16258 | 314 | 0 | 2363 | 98 | 326 | 462 |
| 40 | 15939 | 323 | 0 | 1911 | 103 | 321 | 474 |
| 45 | 19180 | 263 | 0 | 1499 | 151 | 245 | 354 |
| 50 | 17595 | 294 | 0 | 1708 | 123 | 280 | 416 |
| 55 | 18335 | 276 | 0 | 1279 | 156 | 258 | 371 |
| 60 | 17470 | 292 | 0 | 1763 | 136 | 276 | 407 |

# Experiements

## Environment

- CPU: Intel® Core™ i5-1035G4 CPU @ 1.10GHz
- RAM: 32GB RAM
- Disk: 1 TB SSD
- Operating Systems: Ubuntu 20.04

## Experiment 1: The Effect of Number of RTE in SP

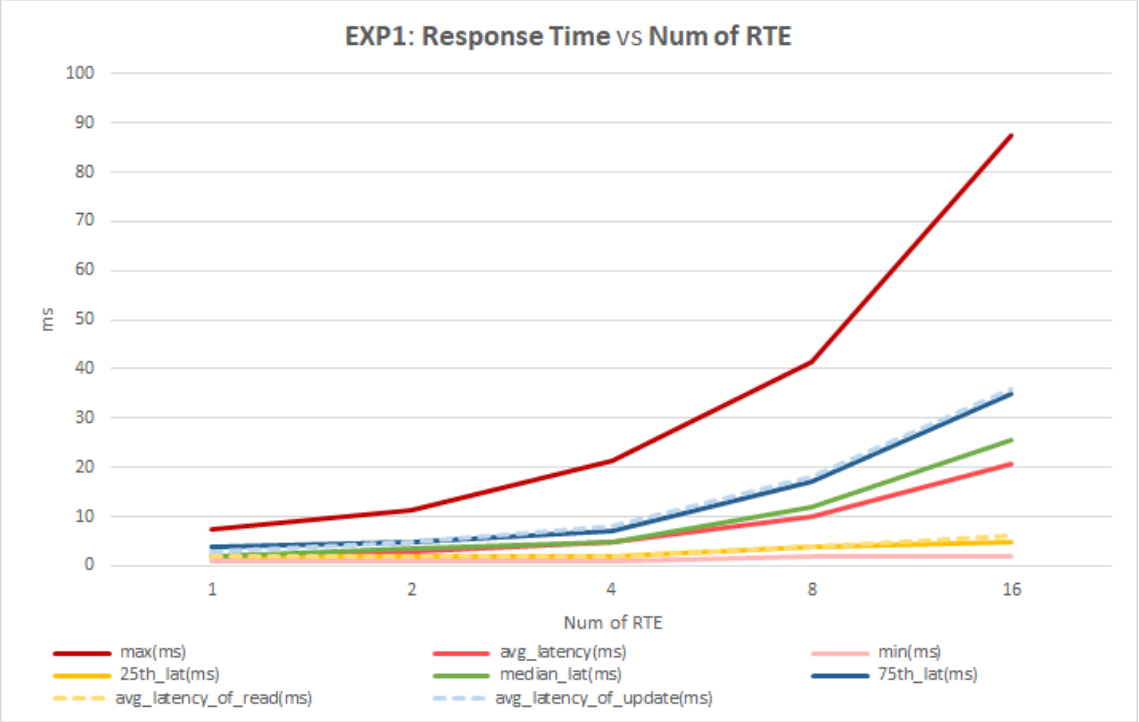


Figure 1.

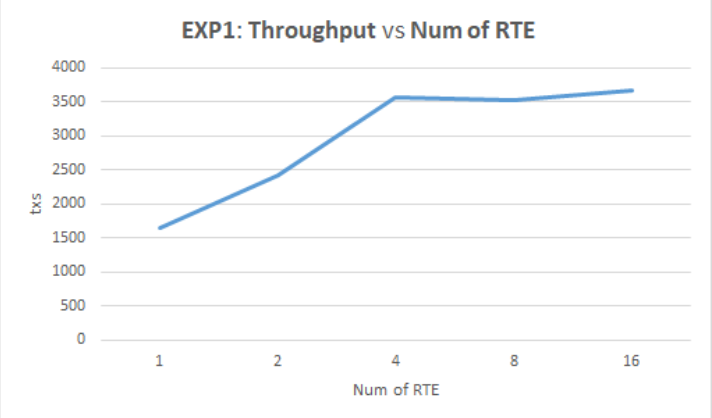


Figure 2.

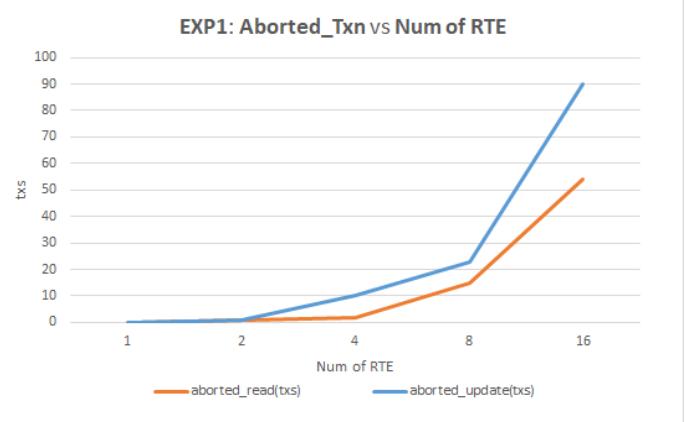


Figure 3.

觀察:

a. **Throughput** 在 **RTE = 4** 的時候到達極限
b. 在 **Figure 1.** 中，可以觀察到 **75 th latency** 緊貼著 **average latency of update**，而**25 th latency** 緊貼著 **Average Latency of Read**；且 **75 th latency**，隨著 **RTE** 增加而大幅增加，但 **25 th latency** 則僅有小幅增加。由此可知，**Update query**在 **DB** 滿載的時候 **Latency** 大幅增加，但**Read Query** 則沒什麼影響。

## Experiment 2: The Effect of TOTAL_READ&UPDATE_COUNT in SP

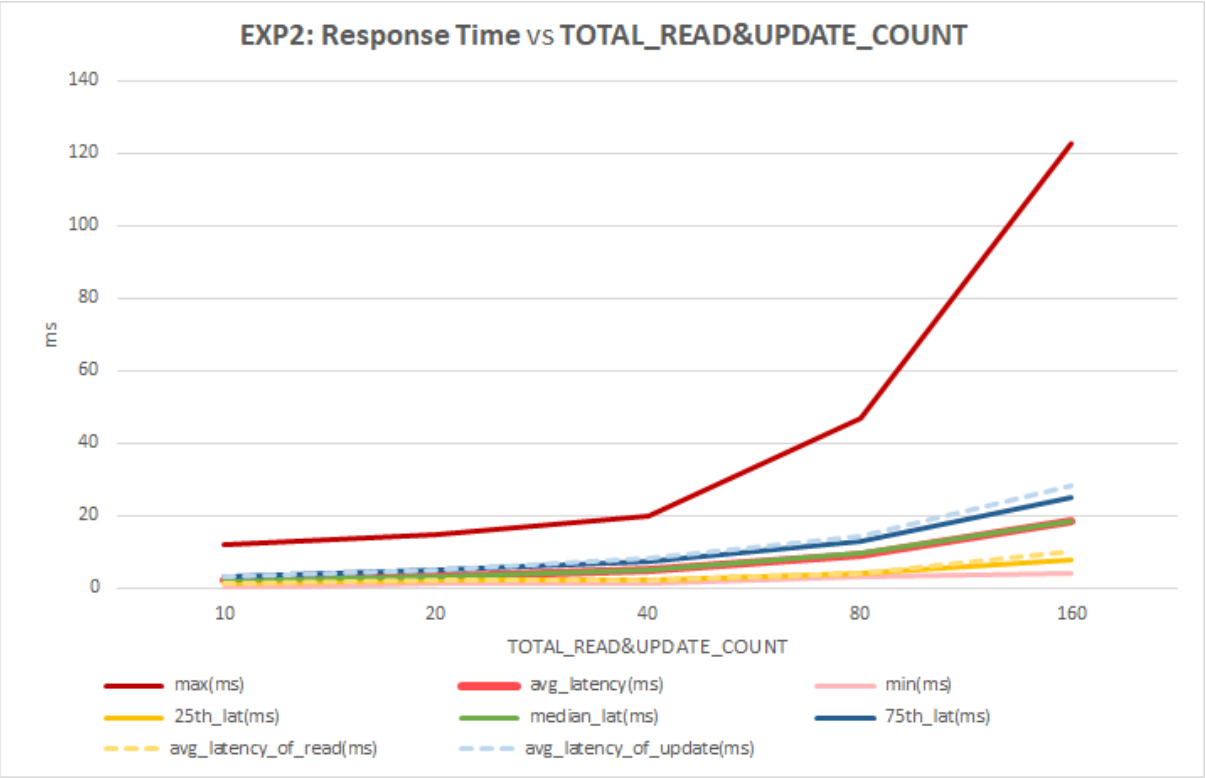


Figure 4.

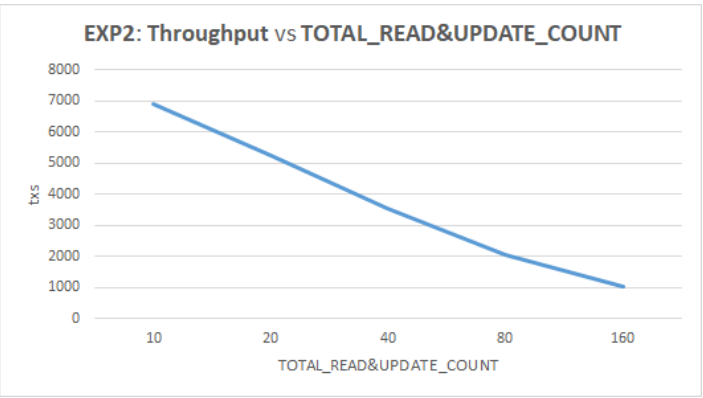


Figure 5.

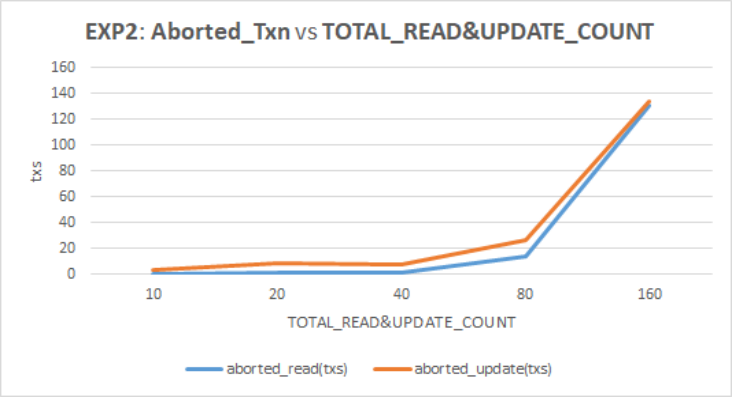


Figure 6.

觀察:

a. 在 **Figure 4.** 中，可以觀察到 **75 th latency** 緊貼著 **average latency of update**，而 **25 th latency** 緊貼著 **Average Latency of Read**；且 **75 th latency**，隨著 **TOTAL_READ & UPDATE_COUNT** 增加而大幅增加，但 **25 th latency** 則僅有小幅增加。由此可知，**Update Query** 在 **DB** 滿載的時候 **Latency** 大幅增加，但 **Read Query** 則沒什麼影響。

## Experiment 3: The Effect of READ_WRITE_TXN_RATE in SP

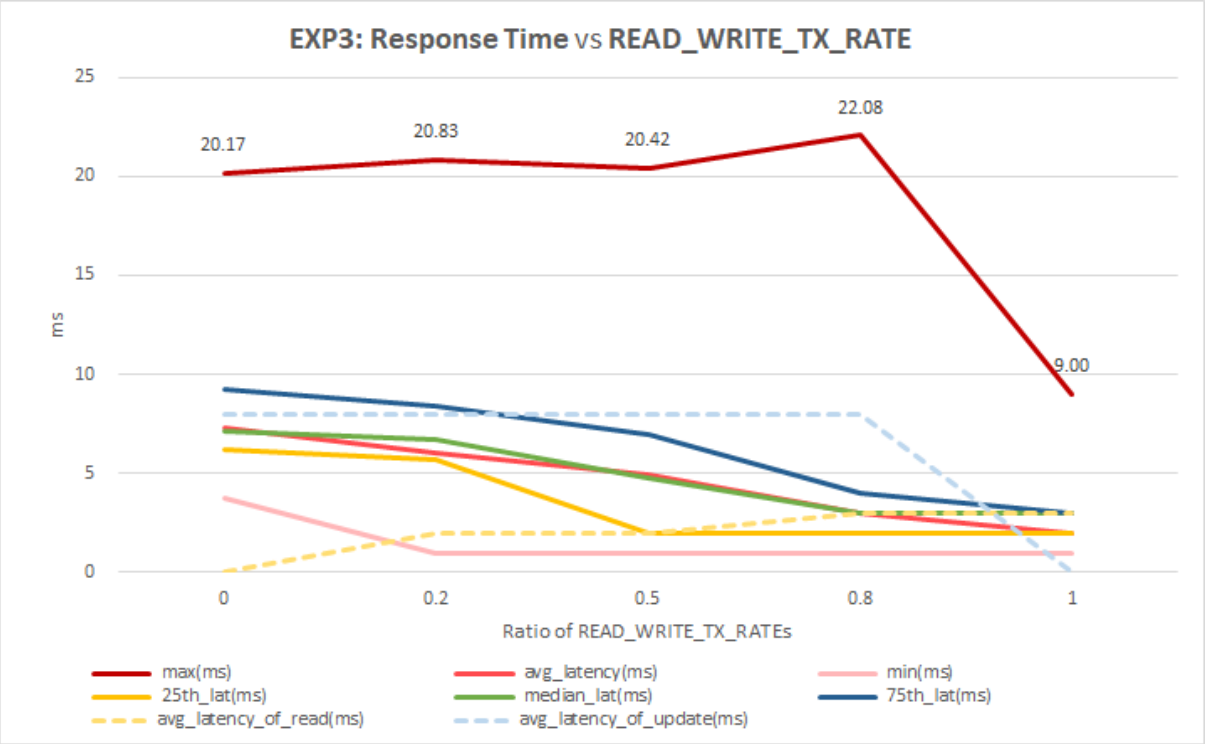


Figure 7.

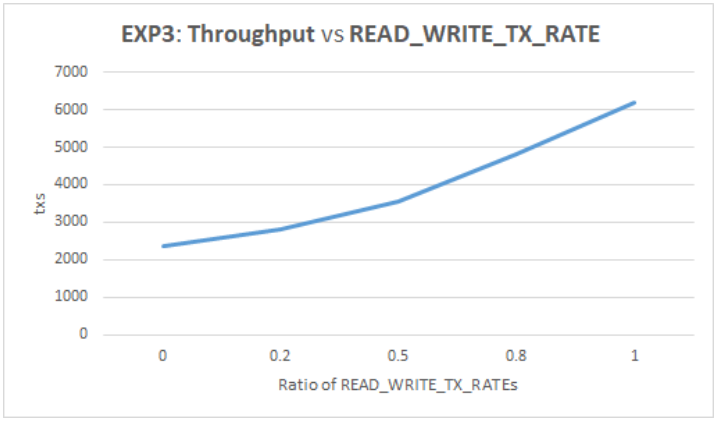


Figure 8.

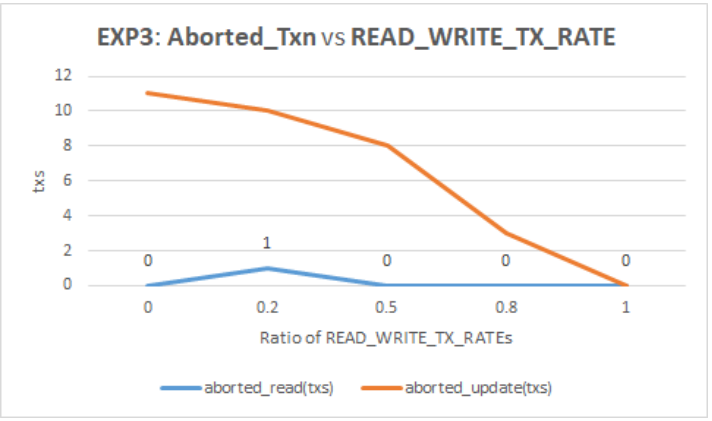


Figure 9.

觀察:

a. 基本上，隨著 **READ Query** 比例的增加，**Latency** 逐步下降，**Throughput** 逐漸上升，**Aborted Transactions** 逐漸下降。
b. 可以觀察到 **Max_Latency** 在 **READ Query** 占比降到 **0%** 的時候，會大幅下降，可推斷絕大部分 **Max_Latency** 是 **Update Query** 所導致。

## Experiment 4: The Effect of Number of RTE between SP and JDBC

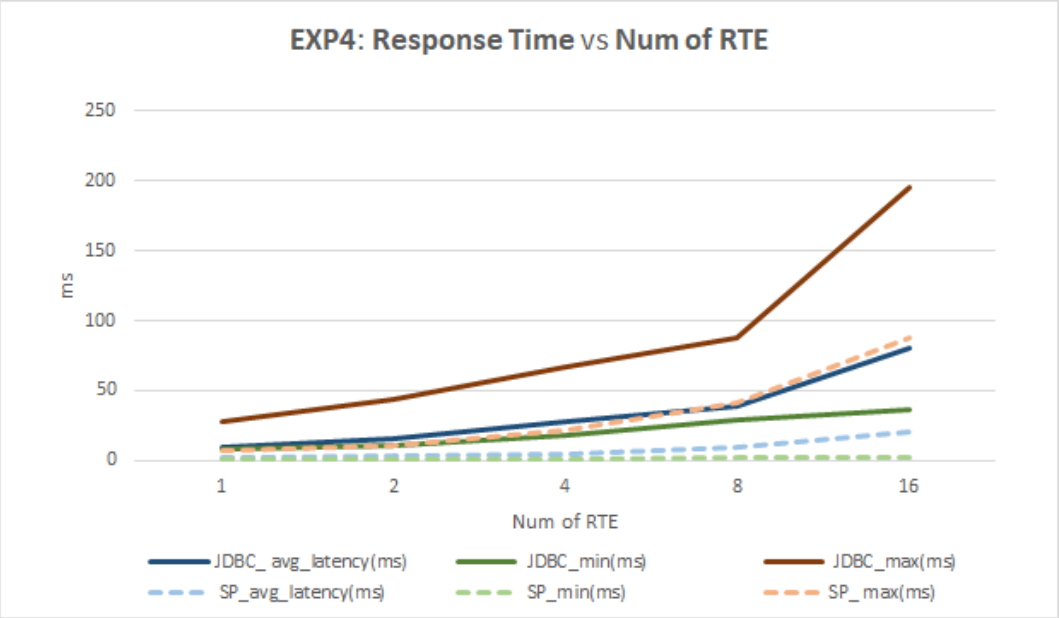


Figure 10.

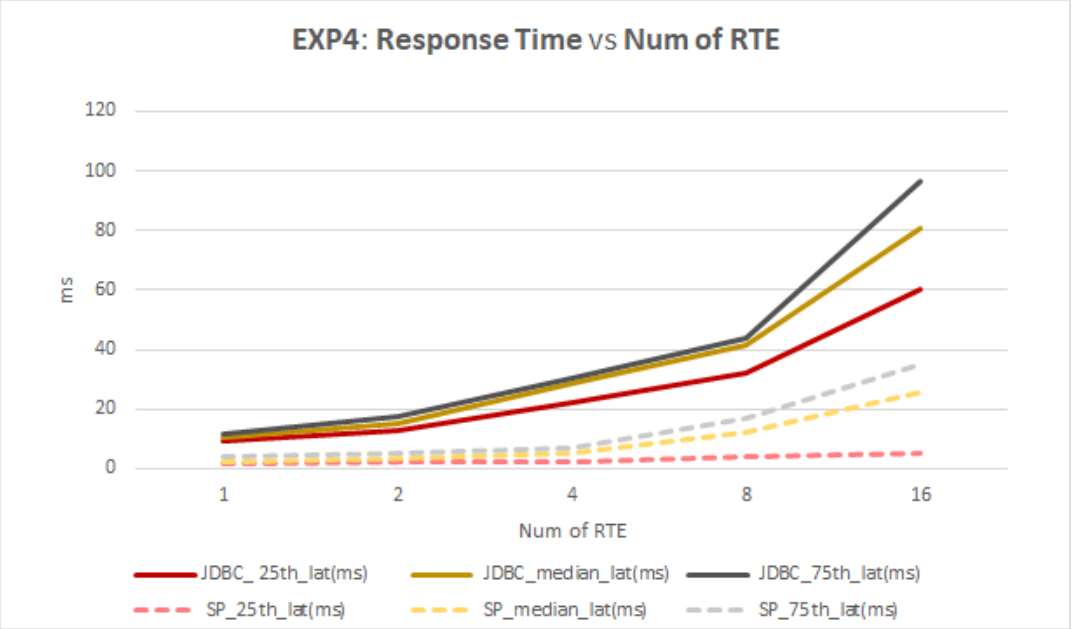


Figure 11.

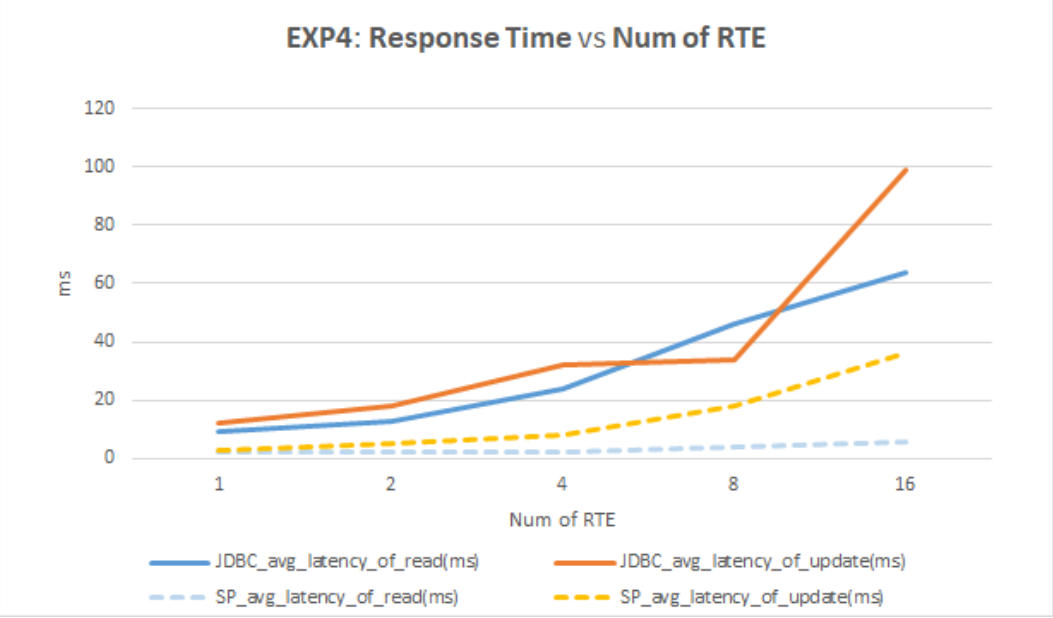


Figure 12.

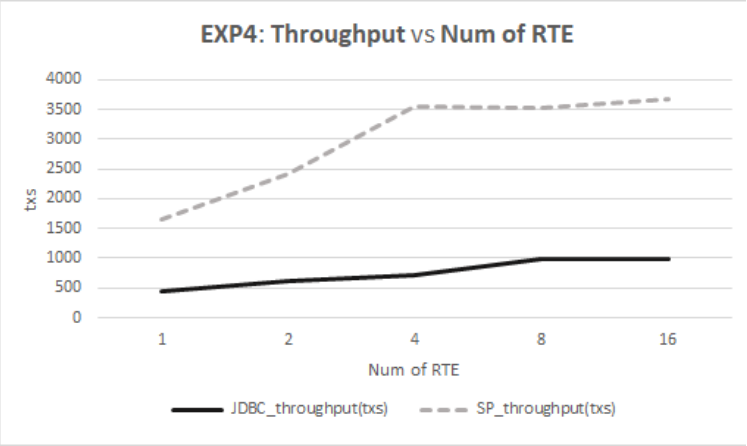


Figure 13.

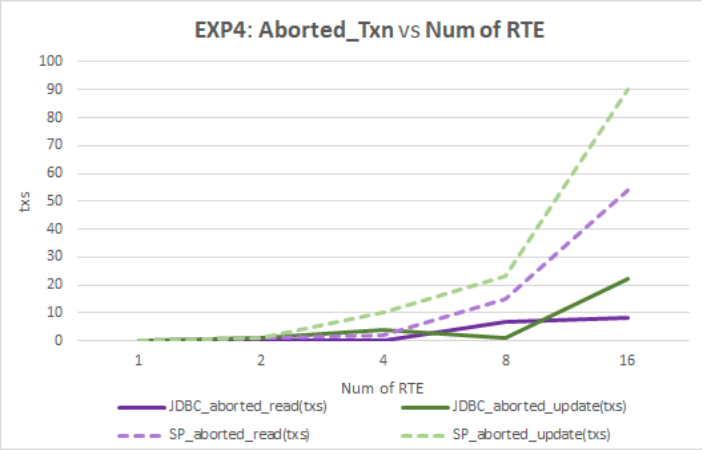


Figure 14.

觀察:

a. **RTE** 數量增加, **Response Time** 會增加。
   i. 當 **RTE** 數量到 **16** 的時候, **Response Time** 會大幅增加, 推測是由於實驗電腦的 **CPU** 為 **4 core / 8 thread** 因此 **RTE** 到 **16** 的時候達到電腦的瓶頸。
   ii. **Figure 12.** 的橘線在 **RTE = 8** 的時候比藍線還低, 此應是實驗誤差。

## Experiment 5: The Effect of READ_WRITE_TXN_RATE between SP and JDBC

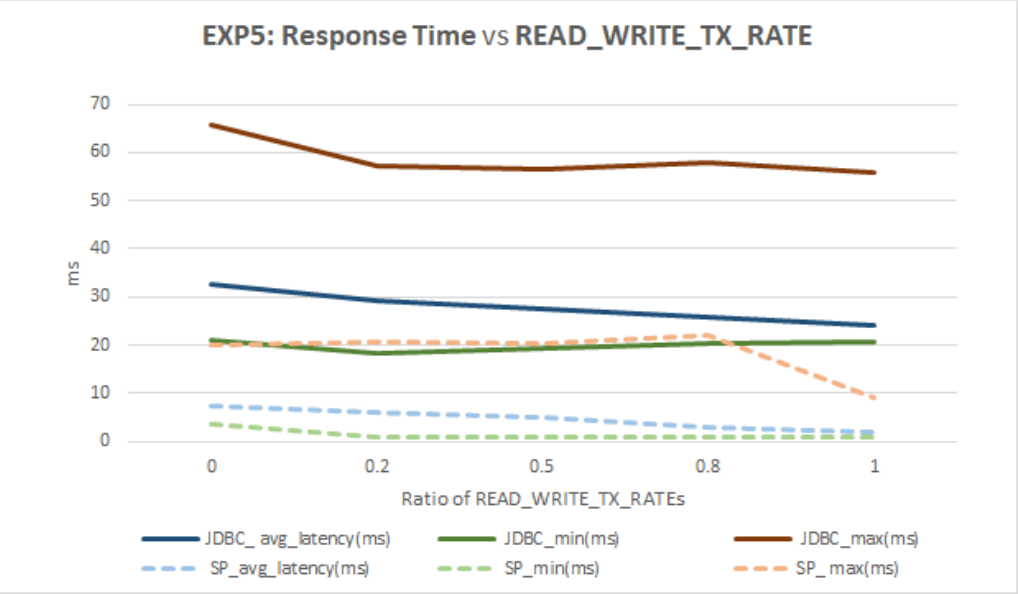


Figure 15.

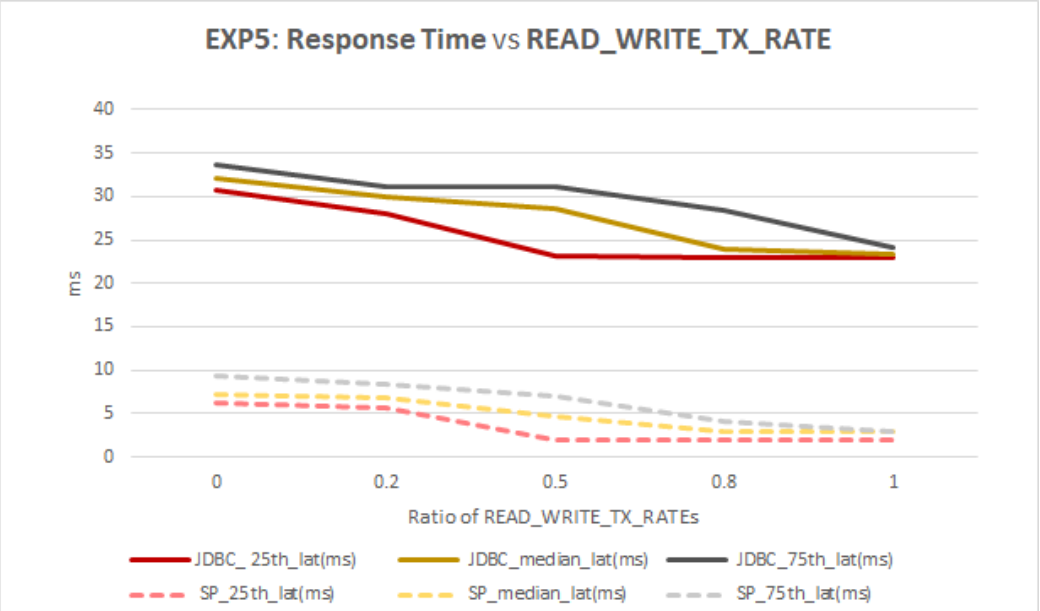


Figure 16.

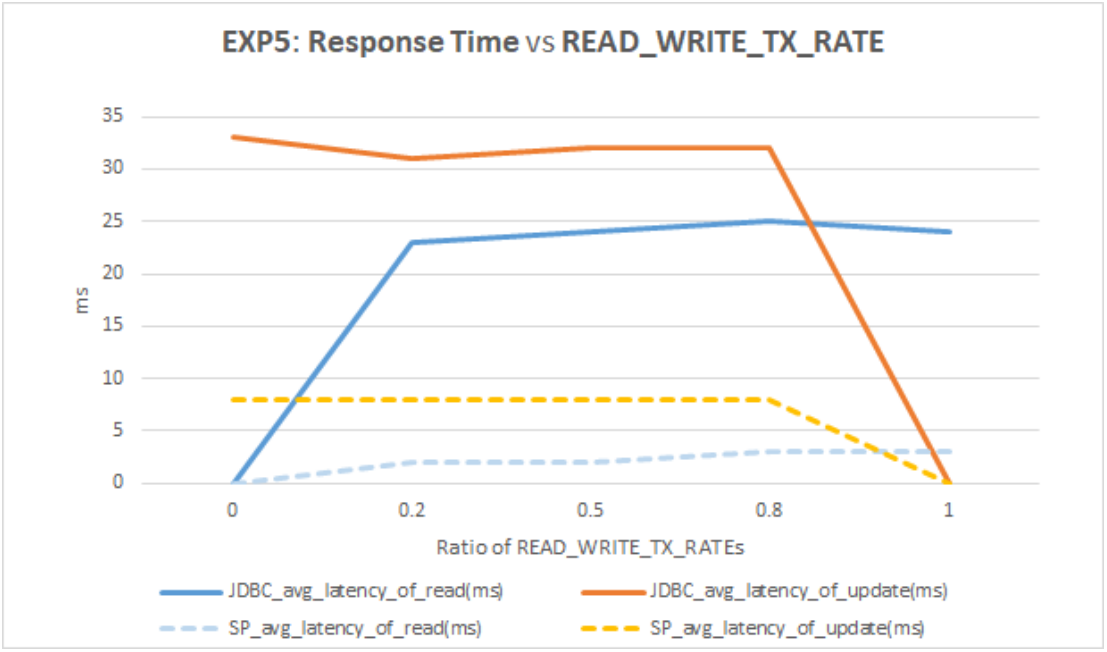


Figure 17.

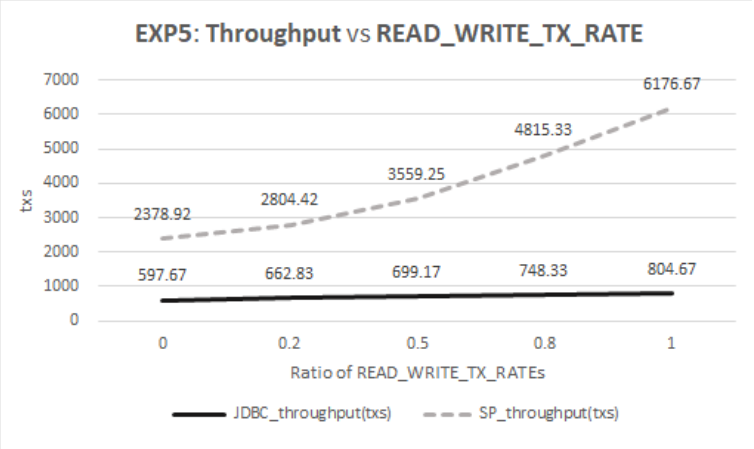


Figure 18.

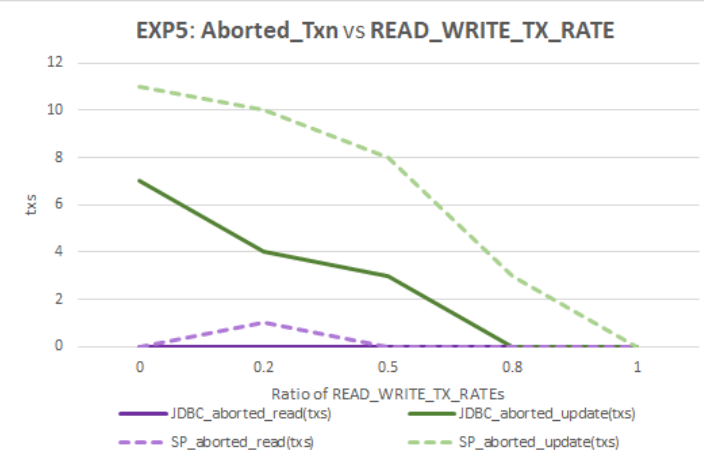


Figure 19.

觀察:

a. 基本上，不論 **read/write** 的比例，**SP** 的 **Response Time** 都比 **JDBC** 短，且如同 **EXP3**，當 **READ_WRITE_TXN_RATE** 越趨近於 **1**（**read** 的比例越高），**Response Time**越短。
b. 在 **Figure 17, 18** 中，可以看到由於 **JDBC read** 與 **update** 的 **Response Time** 都比 **SP** 高，因此，隨著 **READ_WRITE_TXN_RATE** 變大，**SP** 的 **Throughput** 會增長的比 **JDBC** 高許多。
c. **Figure 19** 呈現與 **Figure 14** 相同的趨勢，即由於 **SP** 的 **Txn** 比 **JDBC** 多，**Abort** 的 **Txn** 也比較多。