101062124 戴宏穎

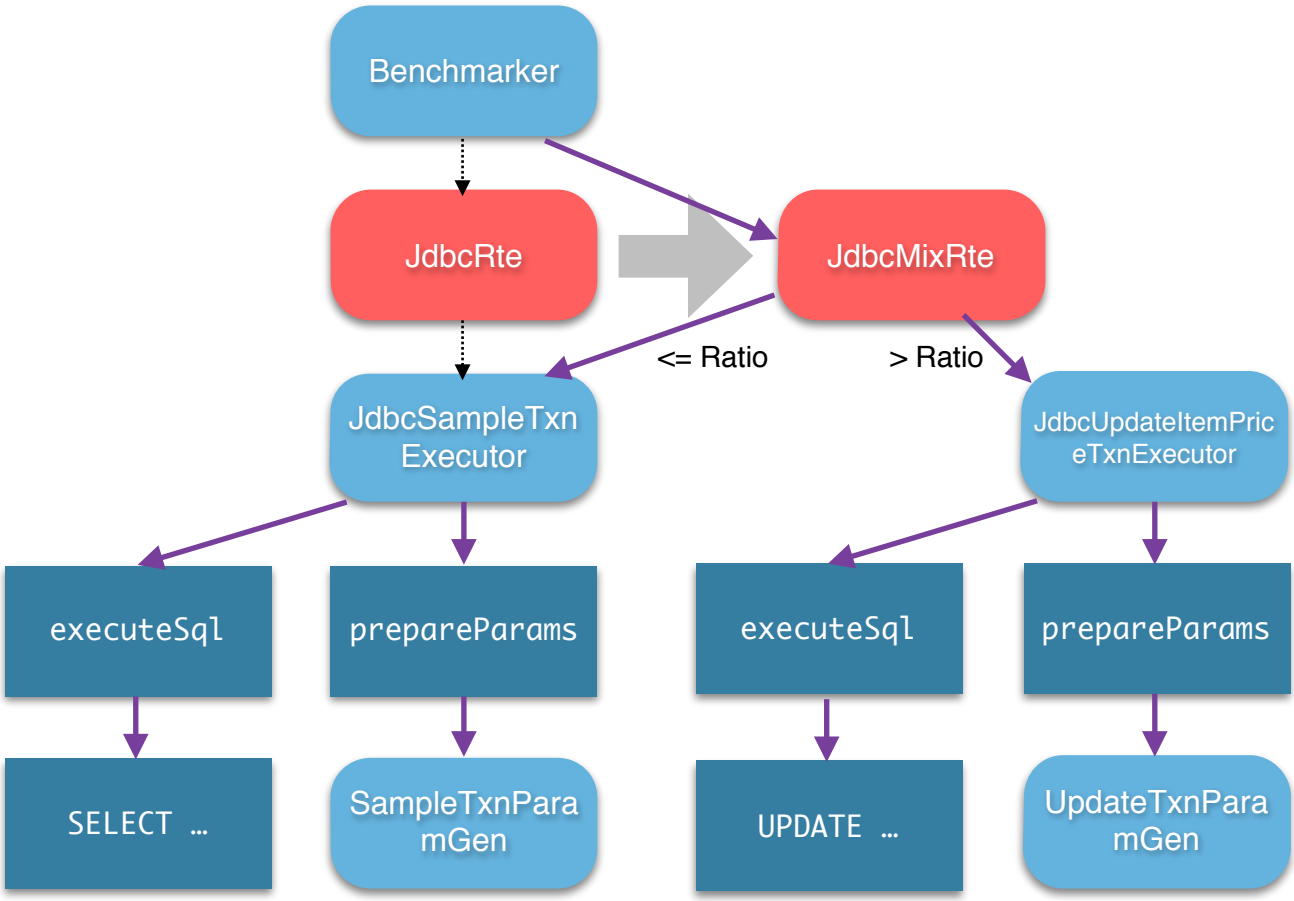
Cloud DB Assignment 1

最先，當我們開始跑 Benchmark 的時候，會從 Benchmarker 切入。
這時候就依照在 TPC-C Benchmark 裡面的參數 IS_JDBC 來決定使用哪一個 RTE，以下我將從使用 JDBC 與否的角度，分別分析兩種執行相關性，在之後會繼續探討要如何增加程式碼來完成這次作業的需求。

1. JDBC Mode
   A. JDBC 開啟後，在 RTE 的部分就會實體化 JdbcRte 這個 class ，進入執行時，還需要把執行 Transaction 的 JdbcSampleTxnExecutor 也實體化出來。在這個 TxnExecutor 裡面，有兩個方法需要完成：
      第一個是 prepareParams 用來產生等一下要 query 的參數，裡頭是透過實體化 SampleTxnParamGen ，透過他來隨機產生我們要 SELECT 的 item id。
      另一個則是我們實際上要做 SQL query 的方法，把語法與參數接好後，就直接送給 JdbcService.executeQuery 處理。

   B. 為了能調整每次可以選擇是要做 Sample 還是 UpdatePrice 的動作，所以我把 JdbcRte 抽換成 JdbcMixRte。每次要做 Query 的時候都先產生一個 0~99 的亂數，如果比 Ratio（我們要測試的比率）來的小，就會測試 Sample （Read-Only）的指令；否則將會執行 Update （Read-Write）的指令。

   C. 為了與 Sample 有所區別，所以 Sample 有的 Class 我全部都產生一份對應的 UpdatePrice 系列 Class。而且 Update 需要有多達十筆資料的更新，產生這部分的參數就是透過改寫過的 UpdateTxnParamGen。

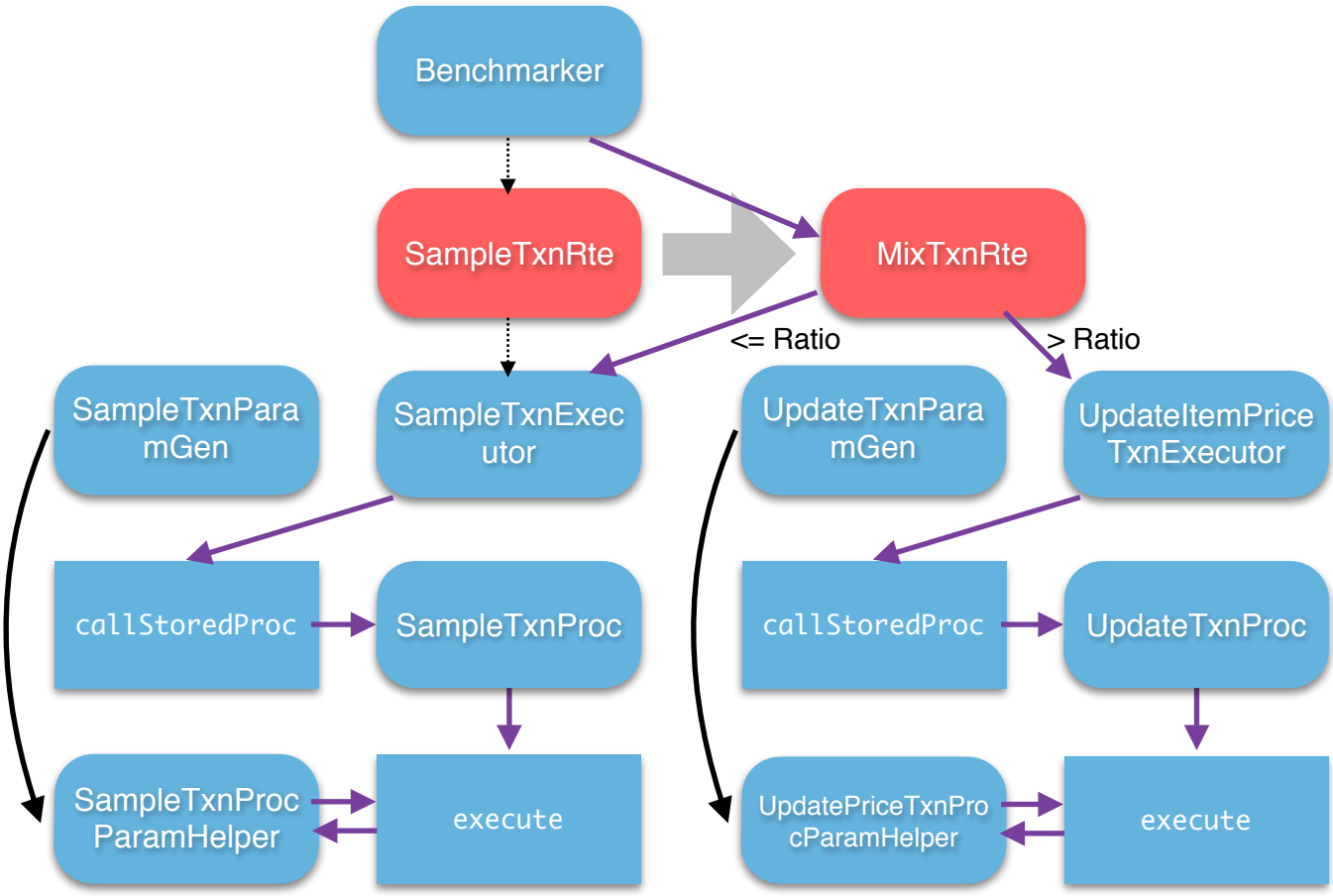


2. Stored procedures Mode
   A. 沒有開啟 JDBC Flag 的話，在 RTE 的部分就會實體化 SampleTxnRte，在 SampleTxnRte 裡面再把 SampleTxnExecutor 實體化出來，用來準備執行 Transaction，接下來的機制和 JDBC Mode 有比較大的差異。就是在這個 TxnExecutor 裡面，其實我們只要把不同的 TxnParamGen 傳進去就能在 callStoredProc 的階段透過查詢 Type 來呼叫對應的 TxnProc 。不過為了看起來比較一致，所以我在實作時還是建立出 UpdatePriceTxnExecutor 。

   B. 與 JDBC Mode 類似，為了能選擇要做 Sample 還是 UpdatePrice 的動作，所以 SampleTxnRte 也被我抽換成 MixTxnRte。每次要做 Query 的時候都先產生一個 0~99 的亂數，如果比 Ratio（我們要測試的比率）來的小，就會測試 Sample （Read-Only）的指令；否則將會執行 Update （Read-Write）的指令。

   C. 如同我前面 2.A 所說的理由，在所有 Sample 有的 Class 全部還是產生一系列對應的 UpdatePrice。所需要的 TxnParamGen 參數產生器都是共用 JDBC Mode 的，不需要重新在撰寫一遍。

   D. 這邊其實我遇到比較大的疑惑是 trace 到 callStoredProc 後突然間就不知道跳去哪裡了，剛好被包在 jar 檔案裡面，沒辦法看到細節的部分，所以有了很大的困擾。不知道怎麼繼續下去，還好有助教與同學們的說明與幫助，讓我瞭解到要怎麼更正與溝通 TxnProc 的部分。

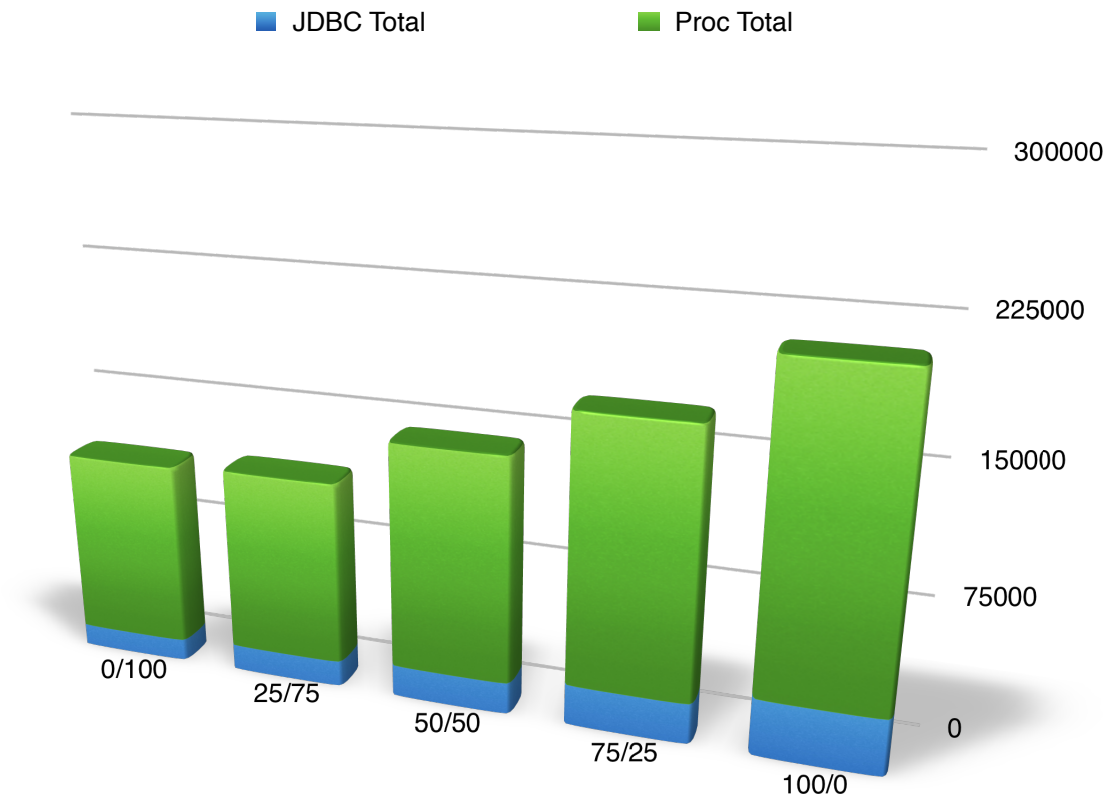


分析不同 Ratio 與 Mode 下的效能（每次測試時間皆為一分鐘）

本表每種皆測試取平均值（Stored Procedure 簡稱為 Proc）

Sample/ UpdatePrice	0/100	25/75	50/50	75/25	100/0
JDBC S/U	0/12256	3637/10956	8961/8929	17133/5716	31793/0
JDBC Total	12256	14593	17890	22849	31793
Proc S/U	0/105659	26265/79015	64327/64816	113474/37739	179565/0
Proc Total	105659	105280	129143	151213	179565

把上面表格製成圖表後我們會發現：



使用 Stored Procedure 的速度會遠高於使用 JDBC ，個人覺得是因為 JDBC 每一次都是開新的 SQL ，並不像 Stored Procedure 那樣可以先開好 SQL 等著填空。也有可能是因為使用 JDBC 是還要透過 Driver 等等協定才能完成一次的 Query ，效能會比 Stored Procedure 還要慢上許多。

另外，Read-Only 的次數越多，Read-Write 的次數越少，效能也隨之上升。我認為原因就在於只做 Read-Only 本身就比較輕量，相對於要做寫入動作的 Update （而且還是每次都做十次），所以消耗的效能就會十分顯著。