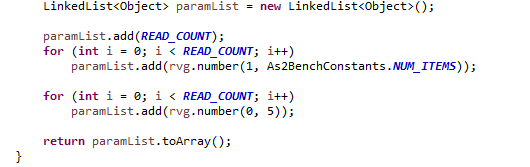
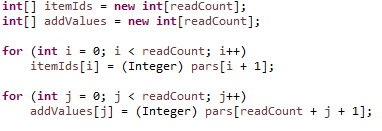
DB19 assignment 2 report

Implementation of ParamGen

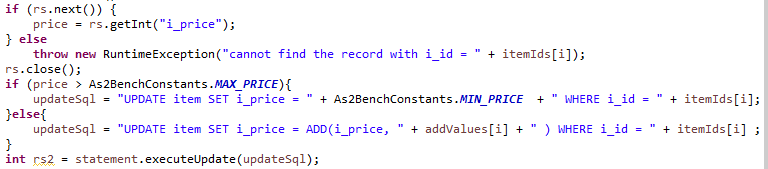


除了原本的隨機選取item外，另外再paramList後面多加了用了做UpdatePrice用的參數。

Implementation of UpdatePriceJdbcJob



在開始的時候多加了對Update參數的處理，存在addValues[]裡面。

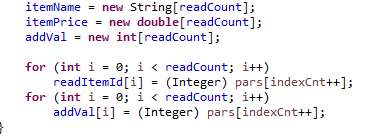


在執行完statement.executeQuery後，先抓出他的Price來判斷是否有超過As2BenchConstants.MAX\_PRICE，若有則設成MIN\_PRICE，否則加上先前產生的addValue，然後執行UPDATE的SQL指令。

Implementation of UpdatePriceProcParamHelper

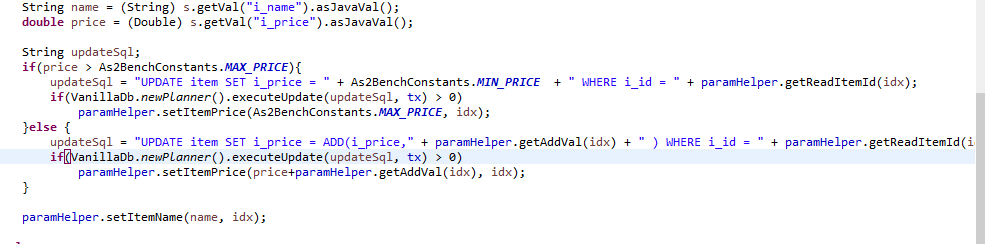


增加一個getAddVal()，來讓SP取的需要update的參數



在prepareParameters裡面處理pars。

Implementation of UpdatePriceProc



在取得Price後，先抓出他的Price來判斷是否有超過As2BenchConstants.MAX\_PRICE，若有則設成MIN\_PRICE，否則加上先前產生的addValue，然後執行UPDATE的SQL指令。

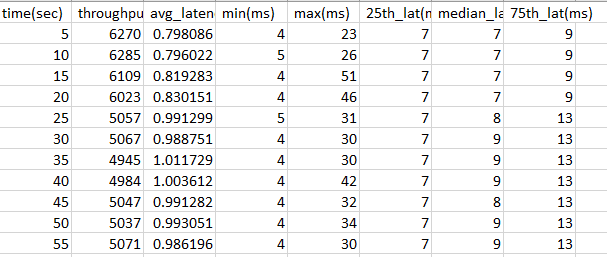
實驗環境: 如下圖所示，硬碟則為256GB SSD



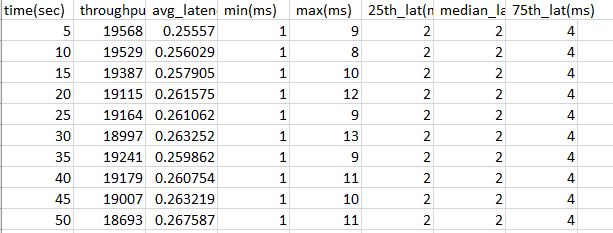
實驗結果:

註: 關於avg\_latency特別小的原因，是因為這裡的latency是用   
5000(微秒)/完成的transaction數量，又因為trasaction有做平行化處理的緣故，所以avg\_latency會較小，但右邊五個attribute是對每一個transaction做計算，因此出現min比avg\_latency大的現象。

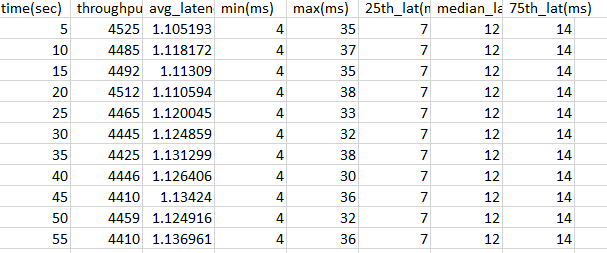
READ\_WRITE\_TX\_RATE 設為0.3，且連線方式為JDBC



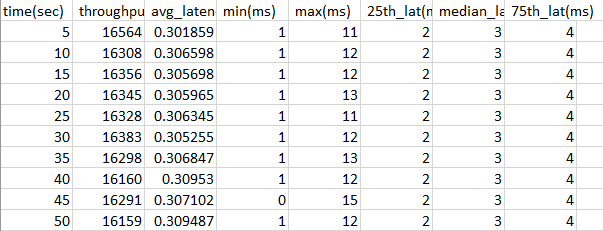
READ\_WRITE\_TX\_RATE 設為0.3，且連線方式為SP



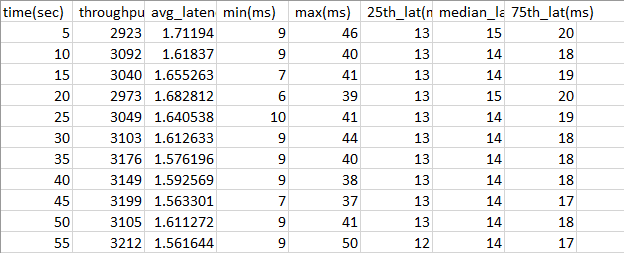
READ\_WRITE\_TX\_RATE 設為0.5，且連線方式為JDBC



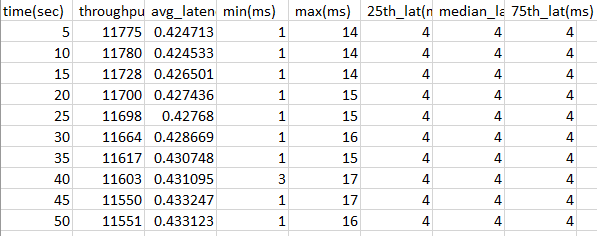
READ\_WRITE\_TX\_RATE 設為0.5，且連線方式為SP



READ\_WRITE\_TX\_RATE 設為1，且連線方式為JDBC



READ\_WRITE\_TX\_RATE 設為1，且連線方式為SP



實驗觀察:

以上面六次不同設定的實驗結果來說，我們觀察到在相同的READ\_WRITE\_TX\_RATE，也就是相同的updateTxn的比例下，

用JDBC溝通server跟client端，在每五秒內的統計中，不論是平均、最小、最大、中位數、四分位數等，所花的latency都較以SP方式溝通的結果來的高，即代表用SP的方式來server和client端，較JDBC來的快

而從同樣的連線方式，但是不同的READ\_WRITE\_TX\_RATE來比較的話，對JDBC來說，不論是平均、最小、最大、中位數、四分位數等，所花的latency都隨著updateTxn 比例的提高而上升，因為updateTxn不但要先做一次select 指令， 要更新table裡面的值也有可能會遇synchronize的問題，一定會花較久的時間。

相較於JDBC，雖然SP的各項latency也有隨著updateTxn 比例的提高而上升，但因為本來速度就非常快，因此變化的幅度沒有那麼大，甚至在READ\_WRITE\_TX\_RATE 分別為0.3、1時，min latency的值幾乎都沒有變化。