Team 16 Assignment 2 Phase 2 Report

1. 為了方便整理，我們將結果輸出到一個csv檔，方便我們整理資料。這樣之後需要做比較時，可以直接一行一行的比較，也可以用python之類的做精美的圖表。

```
private void outputCsvReport(String fileName) throws IOException {
        try (BufferedWriter writer = new BufferedWriter(new FileWriter(new File(outputDir, fileName + ".csv")))) {
                writer.write("time(sec), throughput(txs), avg_latency(ms), min(ms), max(ms), 25th_lat(ms), median_lat(ms), 75th_lat(ms)");
                writer.newLine();
                final Long unit = 5000000000L;
                List<Long> timeList = new ArrayList<Long>();
                Long cnt = 1L;
                Long startTime = recordStartTime;
                Long lastTime = recordStartTime;
                for(TxnResultSet resultSet : resultSets){
                        if(startTime + unit < resultSet.getTxnEndTime()){
                                Object[] objectArray = timeList.toArray();
                                int length = objectArray.length;
                                Long[]  timeArray = new Long[length];
                                Long min = 0L, max = 0L, avg = 0L;
                                for(int i = 0; i < length; i++) {
                                        timeArray[i] = (Long) objectArray[i];
                                        min = Math.min(min, timeArray[i]);
                                        max = Math.max(max, timeArray[i]);
                                        avg += timeArray[i];
                                }
                                avg /= length;
                                Arrays.sort(timeArray);
                                writer.write(String.format("%d,%d,%d,%d,%d,%d,%d,%d",
                                                cnt * 5,
                                                length, TimeUnit.NANOSECONDS.toMillis(avg), TimeUnit.NANOSECONDS.toMillis(min), TimeUnit.NANOSECONDS.toMillis(max),
                                                TimeUnit.NANOSECONDS.toMillis(timeArray[(int)((double)length * 0.25)]),
                                                TimeUnit.NANOSECONDS.toMillis(timeArray[(int)((double)length * 0.5)]),
                                                TimeUnit.NANOSECONDS.toMillis(timeArray[(int)((double)length * 0.75)])));
                                writer.newLine();
                                cnt += 1;
                                startTime = lastTime;
                                timeList.clear();
                        }
```

2. 原本throughput的計算是(總response time) / (txn 數)，但我們覺得會不準。因為可能指令之間會有空閒的時間，造成時間多算，或是會同時執行，導致少算，反正不論是哪種狀況都會造成誤差。因此我們將計算方式改成5秒內的txn總數當計算方式，認為此方法會增加throughput的準確性。
3. READ_WRITE_TX_RATE我們是在As2BenchmarkRte裡直接使用，但助教的code有先在As2BenchConstants裡先宣告。我們認為應該都是一樣的意思，沒有特別需要改的地方。但可能是助教想要模擬一個dbms會怎麼取參數，所以才這麼寫code。

```
READ_WRITE_TX_RATE = BenchProperties.getLoader().getPropertyAsDouble(
                        As2BenchConstants.class.getName() + ".READ_WRITE_TX_RATE", 0.00);
```

```
READ_WRITE_TX_RATE = BenchProperties.getLoader()
        .getPropertyAsDouble( propertyName: As2BenchmarkRte.class.getName() + ".READ_WRITE_TX_RATE",  defaultValue: 0.6);
```

4. 在ParamGen裡，助教會用一個UpdateItemPriceTxnParam把需要的id和number給包起來，看起來比較精美。相對的我們是把全部的東西就丟進list，好處就是比較方便，而且只要知道東西在哪，哪個是屬於哪個，基本上就不會有太大的障礙。但相對於助教的code，可能對其他人就麻煩一點，假如是多人在做同一個project，用助教的這種方法會比較合適

，不會大亂。

```
public class UpdateItemPriceTxnParam implements Serializable {
        public int itemId;
        public double raise;

        private static final long serialVersionUID=1l;

        public UpdateItemPriceTxnParam(int itemId, double raise) {
                this.itemId = itemId;
                this.raise = raise;
        }
}
```