

# Database Final Project Report

TEAM 6 107062318 李俊逸 107062202 陳敬和 107062237 張濬洋

## 1. Briefly explain what we exactly do

### 1.1 How to collect features

我們在server端建立一個FeatureMap class, 用來記錄實現所需的csv file, 內建一個Map featureMap, key是txNum, value則是一個FeatureCollection object, 這個object裡面除了原本的feature, 也有新增的feature。

至於csv檔案輸出, 我們在FeatureMap class寫了一個output\_featureMap\_csv的function, 以及新增一個storeProcedure, 讓client端可以在benchmark()結束前call那個function, 使server可以輸出csv file。

### 1.2 What features we collect and why

- i. txNum: tx的流水號。
- ii. latency: tx從被server收到, 到commit的時間。
- iii. startTime: startTime是server收到tx request, 所以我們在RemoteConnectionImpl.java的callStoredProc的第一行開始記錄。
- iv. readCount: size of readSet, 代表一個tx讀到的table數。
- v. writeCount: size of writeSet, 代表一個tx寫到的table數。
- vi. txnType: TPC-C有若干個txType, txType相同的話, 會執行相同的操作, 只是可能量體會有差別, 而量體可以由其他feature來capture, 如readCount, writeCount。以目前設定來說, txType有兩種, 分別是new\_order及payment。
- vii. concurrentlyExecutingTxNum: 如果同時有很多tx在執行的話, 代表系統比較繁忙, 我們預期這個值越高, latency越長。
- viii. readRecordSize: 由於readCount只有包含readSet, 而一個readSet代表有read一個table, readRecordSize則是紀錄access幾個record, 我們預期會有readCount一樣但readRecordSize不一樣的狀況, 紀錄得更仔細一些有助於model accuracy。
- ix. writeRecordSize: 理由與readRecordSize相同。
- x. numberOfQueuingTx: 紀錄tx在執行scheduleTransactionSerially()時, 拿到SERIAL\_CONTROL\_LOCK之後, 還有多少tx在等這個lock, 預期如果等待lock的人越多, 系統越繁忙, latency越大。

xi. memoryUsage: 紀錄在tx開始前, DB的memory使用狀況, 預期memory使用率越高, latency越高。

xii. cpuUsage: 紀錄在tx開始前, DB的cpu使用狀況, 預cpu使用率越高, latency越高。

### 1.3 Model structure & type

Baseline和improvement都用Random Forest, 參數設置除maxDepth設成30以外, 都與助教預設相同, 詳細如下:

*nTrees: 200, mtry: trainingSet.ncols() / 3, maxDepth: 30, maxNodes: 100, nodeSize: 5, subsample: 1.0*

## 2. Experiments

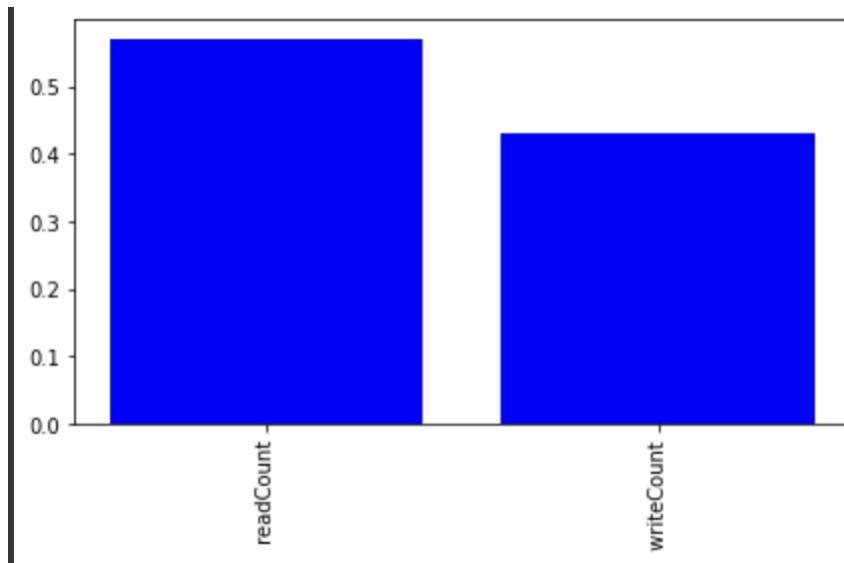
### 2.1 Experiment environment

Intel Core i5-8250 CPU @ 1.6GHz, 8 GB RAM, 1TB HDD, Windows 10

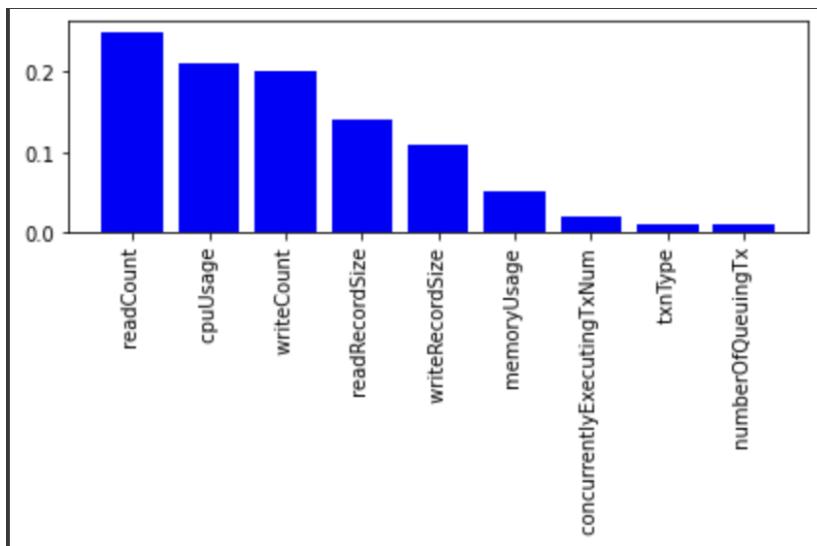
### 2.2 Based on the default workload

#### 2.2.1 Feature importance

##### i. Baseline



##### ii. Our data



## 2.2.2 Improvement of MAE and MRE score

### Baseline training result

```
===== Result Summary =====
Training MAE: 6429.44
Training MRE: 72.55%
```

### Our training result

```
===== Result Summary =====
Training MAE: 6075.00
Training MRE: 69.76%
```

### Baseline testing result

```
===== Result Summary =====
Testing MAE: 6535.30
Testing MRE: 76.90%
```

### Our testing result

```
===== Result Summary =====
Testing MAE: 6340.15
Testing MRE: 76.53%
```

## 2.3 The benchmarks and the chosen parameters

由於助教在5/30時在issue當中提到不能更動vanilladb.bench.properties, 故我們並沒有額外設定其他的workload, 都是TPC-C的default parameter。

## 2.4 Analyze and explain the result of the experiments

從2.2.2的結果來看, baseline和我們新增的data跑的結果來看, 在training的結果上進步幅度較多(Training MAE: 6429->6075, Training MRE: 72.55%->69.76%), 顯示model沒有overfit; 在testing的結果上有些許的improvement (Testing MAE: 6535->6340, Testing MRE: 76.9%->76.53%), 顯示我們新增的feature對模型預測只有微幅貢獻。我們挑選feature的根據會在第三、四點解釋。

## 3. Discuss and conclude why these features are *important*

除了原本的readCount, writeCount之外, 與前述feature相似的**readRecordSize**和**writeRecordSize**也有一定的重要性, 顯見I/O是latency主要影響因素。另外, 在tx開始執行前, 系統的資源使用狀態也是模型認為重要的feature。

另外, 我們認為其他feature在prediction時沒有起作用的原因是以下:

1. concurrentlyExecutingTxNum: cpuUsage, memoryUsage比起concurrentlyExecutingTxNum更能夠描述tx latency。
2. txnType: 因為只有payment和new\_order, 在只有兩種txnType的情況下, 起不了太大作用。
3. numberOfQueuingTx: 從numberOfQueuingTx看, 並不知道QueuingTxn的type, 所以效果不佳。因為如果相同QueuingTxnType的比較多, 我們預期latency較大。

我們嘗試新增很多feature, 但是對accuracy的影響不大, 或許是因為資料庫系統太過複雜, 我們還沒完全地掌握latency的影響因素, 未來我們會多加學習。

## 4. Discussion of other features we had considered but not *chosen*

### 4.1 Cardinality

Cardinality是指table之間的關聯性, 我們認為如果一個tx涉及到的table的cardinality越複雜, 則latency會越大。但我們trace code之後, 認為這個要抓這個feature的實作複雜, 難度過高, 故沒有implement。

### 4.2 Number of log, log interval

論文中有提到log也是可以使用的feature, 但是因為spec規定feature只能是sp.execute()之前拿得到的值, 所以log相關資訊在本次作業無法使用。

### 4.3 txPrepareTime

`txPrepareTime`是從server收到tx, 到tx拿到lock並且準備開始執行前的耗時。不過因為做實驗之前, 我們假定tx開始時間是tx被new的時間點。再加上conservative lock的機制是拿到lock才new tx, 故那時候認為這個feature沒用, 後來發現時來不及實作。