**Final Project Phase2 Report**

**Team14 106030019 吳岱容 106070038 杜葳葳**

- Goal: Optimize **scalability** with the **TPC-C benchmark**
- Implementation
  - **Early lock release**
    - calvin實作conservative lock，原本是在tx commit的時候才放掉所有的lock，我們提前在每個record flush後就馬上放掉那個對應的lock

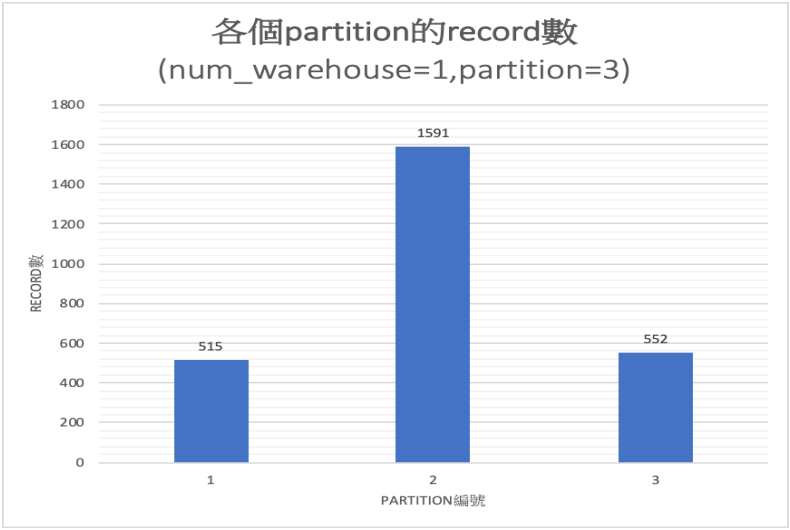
```
public void flush() {
    for (PrimaryKey key : dirtyKeys) {
        InMemoryRecord rec = cachedRecords.get(key);

        if (rec.isDeleted()) {
            VanillaCoreStorage.delete(key, tx);
            releaseLock(tx,key); // opt
        } else if (rec.isNewInserted()) {
            VanillaCoreStorage.insert(key, rec, tx);
            releaseLock(tx,key); // opt
        } else if (rec.isDirty()) {
            VanillaCoreStorage.update(key, rec, tx);
            releaseLock(tx,key); // opt
        }

    }
    dirtyKeys.clear();
}
```

  - **減少重複做hashcode的次數**
    - 原本有用到Hashcode 的地方幾乎都是每次需要時，就call一次hash function
    - 改成建構物件時先把hash紀錄起來，需要時直接回傳
  - **增加檢查的頻率**
    - 改為MILLISECONDS
  - **重新劃分partition**
    - 將較常相互讀取的table放在同個partition上，或是讓workload更均勻（有用一個counter計算各個partition的record數，如下圖，發現蠻不平均的）

- 但後來trace code後發現partition並非原先所想的單純用primary key的hash來進行劃分，而是已經有優化了，因此就沒有在做優化
- 另外，有嘗試修改劃分partition的方法，改為依照tpcc document上的說明，將相互存取較頻繁的tbl放在同個partition，但改動後在拿local和remote cache的資料那裡會出錯，因此後來沒有做改動




○ **改變server間傳record的protocol**

- 原本是以P2P互傳，每個server都要傳一次，改成以Total Ordering一次傳全部
- 不過經過實際測試過發現效能沒有變好反而變差，原因是total order其實是像多次P2P

```
private void pushRecordsToRemotes(Map<Integer, Set<PrimaryKey>> pushKeys, Map<PrimaryKey, InMemoryRecord> records) {
    RecordPackage pack = new RecordPackage(txNum);
    for (Map.Entry<Integer, Set<PrimaryKey>> entry : pushKeys.entrySet()) {
        //Integer targetNodeId = entry.getKey();
        Set<PrimaryKey> keys = entry.getValue();
        // Construct a record package

        for (PrimaryKey key : keys) {
            InMemoryRecord rec = records.get(key);
            if (rec == null)
                throw new RuntimeException("cannot find the record for " + key);
            pack.addRecord(rec);
        }
        // Push to the target
    }
    Calvin.connectionMgr().pushRemote(pack);
}
```

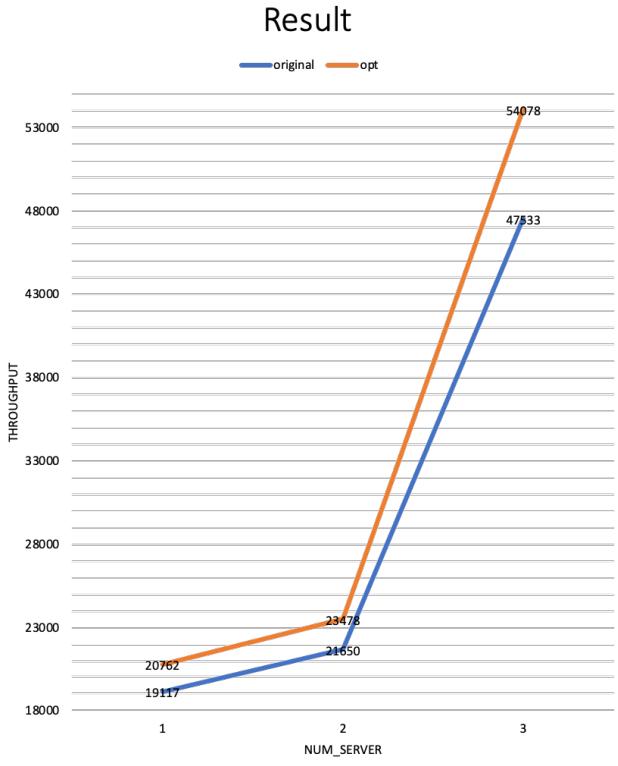
- Experiment
  - 實驗環境：
    - Client/Server:

      CPU: Intel® Core™ i7-8700 CPU @3.20GHz 3.19GHz

      RAM: 8.00GB

      SSD: 256GB
  - 實驗結果
    - TPC-C Benchmark, 1 Client
    - RTE = 20, NUM_WAREHOUSE = 2
    - 此為只有進行early lock release和hash的實驗結果

| #server | original | opt |
|---|---|---|
| 1 | 19117 | 20762 |
| 2 | 21650 | 23478 |
| 3 | 47533 | 54078 |




    - 比較上圖藍線（original）和橘線（opt）的斜率可以發現有小幅的增加

        - 各個優化皆有單獨測throughput和原始的比較，其中，early release lock的增加幅度最顯著

- 所遭遇的困難
    - 思考與嘗試了許多方法，但多數皆成效有限
    - 使用不同電腦跑實驗結果有些差異，造成過程中誤判成效
- 結論
    - 嘗試許多方法後發現，**當#server增加後，throughput增加，但不一定有scalability**
    - 增加scalability方式大概是要盡量想辦法讓transaction不要互相卡住
    - 改善lock的規則有可能可以提升scalability
    - 最後，這學期真的非常感謝老師與助教們，辛苦了！