Final Project: Calvin

106034061 曾靖渝
106062116 黃晨
106062216馮謙

A. Implementation

a. Cache

i. CalvinRecord

1. 繼承SpResultRecord 用來當作Cache 的最小儲存單位
2. 與RecKey最對應來區別query

ii. CacheMgr

1. 利用CalvinRecord 與 RecKey來區別query，別將其放入RemoteRecords的Map中，在StoreProcedure中被加入，在藉由Sender送至GroupServer。

b. Concurrancy

i. ConservativeConcurrancyMgr

1. 繼承ConcurrancyMgr，主要有三個 function，負責 register read、write locks，以及 request 所有 locks。

ii. ConservativeLockTable

1. 當 ConcurrancyMgr 要 register locks 的時候，lockTable 會把這個 txn 放到一個 requestList 當中，接著 txn 要 request locks的時候，必須是在 requestList 的第一個，如此可以確保 txn 的順序和拿到 locks 的順序是一樣的。

c. GroupCommunication

i. BatchSender

1. 將此Client 的request送至Server，並做好batch分割。

ii. GroupClient

1. 代表每個Client 的資料，接收RTE request，再將request傳到server.

iii. GroupServer

1. 代表每個Server(Node)，負責處理Client Request，並將Request message至下層的Scheduler處理。

iv. Pair

1. 將RecKey 與Calvin 做配對，作為配送資料的單位，將會在CalvinStoredProcedure時用收集資料並傳送到其他Server以及接收其他Server的資料，以達成RemoteRead.

v. Server2Server
    1. 將Record 即RecKey組成Pair，並打包成List，在Server之間傳送，以完成remoteRead。
vi. ServerResponse
    1. 回覆Server執行完的 query resultset給RTE;
vii. StoredProcInfo
    1. 資料發送的包裹，以Pars的形式將Client 送來的資料打包往下層送，方便Scheduler接收 CalvinStoredProcedure執行。

d. Server

i. Calvin
    1. 整個Calvin 上層的總控台，init 所有需要用到的class，如 Cache, Server, Scheduler, PartitionMgr.

e. Sql

i. RecKey
    1. 再進入底層的storage之前，用query的table name 以及 predicate當作record得unique ID，再將recKey存入 StoredProcedure並分為remote read, local read, local write，使得其可以在之後開planner去access每個底層record去做讀寫。

f. StoreProcedure

i. CalvinStoredProcedure
    1. 我們用 VanillaDB 的 taskMgr 產生一個 thread，來執行各種 StoreProcedure 的內容。
    2. 我們首先創建了一個 txn，接著為了分辨每個 sql 的 record 屬於哪個 partition，我們寫了一個 prepareRecKey()，將所有指令根據 metadata 分為 localRead、remoteRead 以及 localWrite.
    3. 接著，根據 calvin 的架構，我們依序執行 localRead、remoteRead、localWrite。最後，我們選定執行最後 Write 的 partition 作為負責回傳 resultSet 的 server。如果是 read only procedure，則選擇一個 node 代替。

ii. CalvinStoredProFactory
    1. 模仿StoredProcFactory的方式，用來生成 CalvinStoreProcedure.

g. Scheduler

i. CalvinScheduler
    1. 接收Server從Client 那邊收到的message，包裝成 StoreProcedureInfo傳給Scheduler
    2. Scheduler會根據processID建立CalvinStoreProcedure，並使用 CalvinStoreProcedure來形收到的query。

h. Matadata

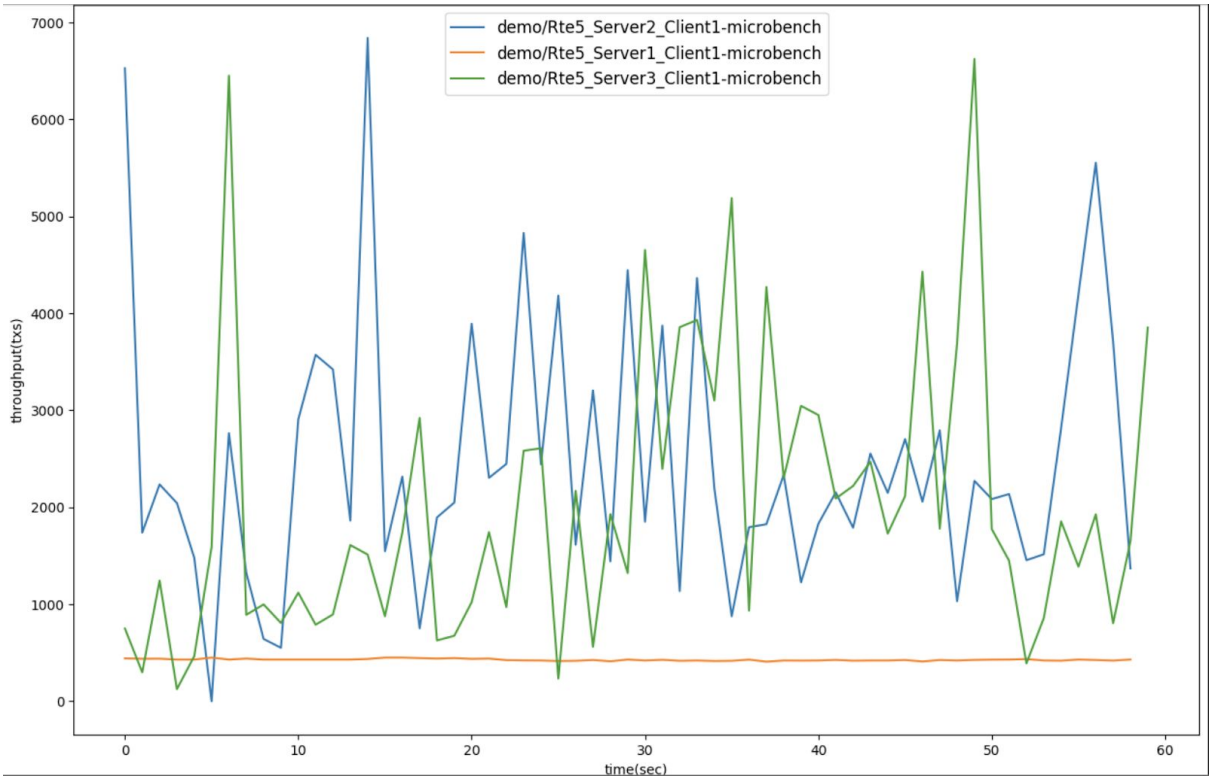
i. PartitionMgr

1. 用於區分各個 record 的 partition。我們原本是根據 recKey，也就是query 的 predicate 來判斷，但後來因為會錯誤，而改成用 tablename 去做 hash。

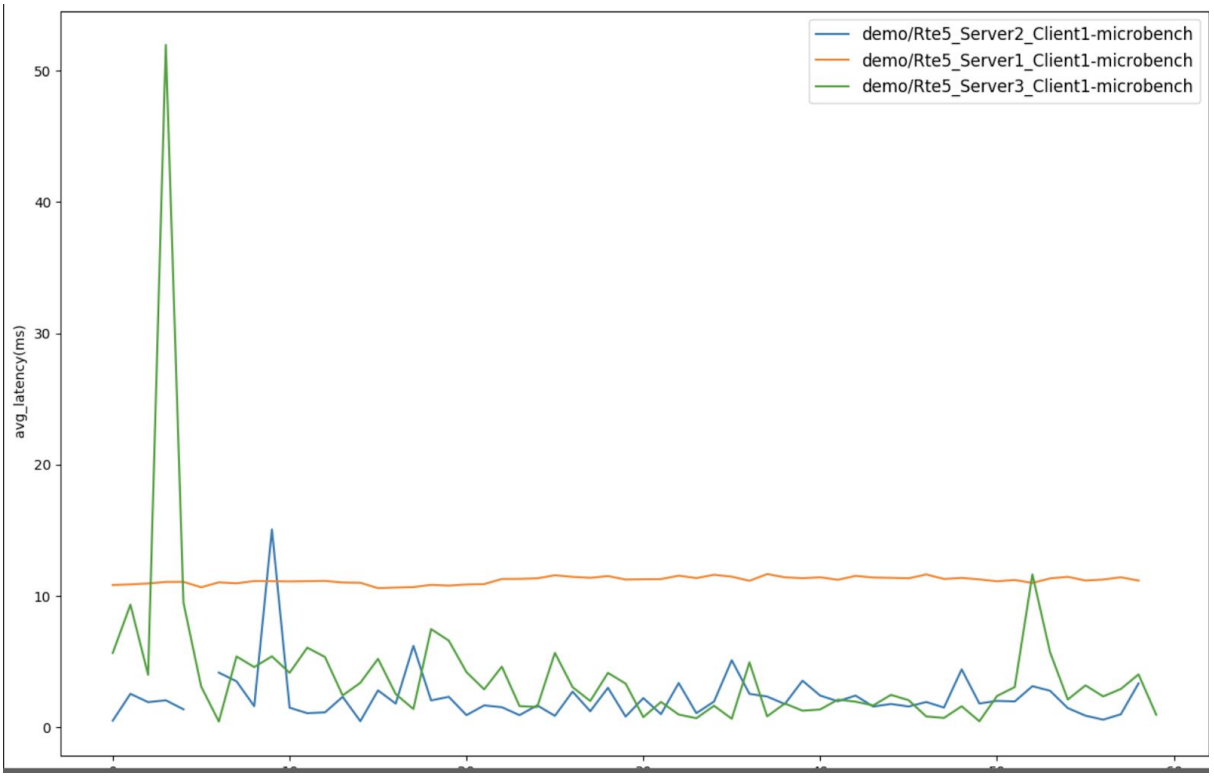
B. Experiments

a. Num of Servers(MicroBenchMark)

i. Throughput



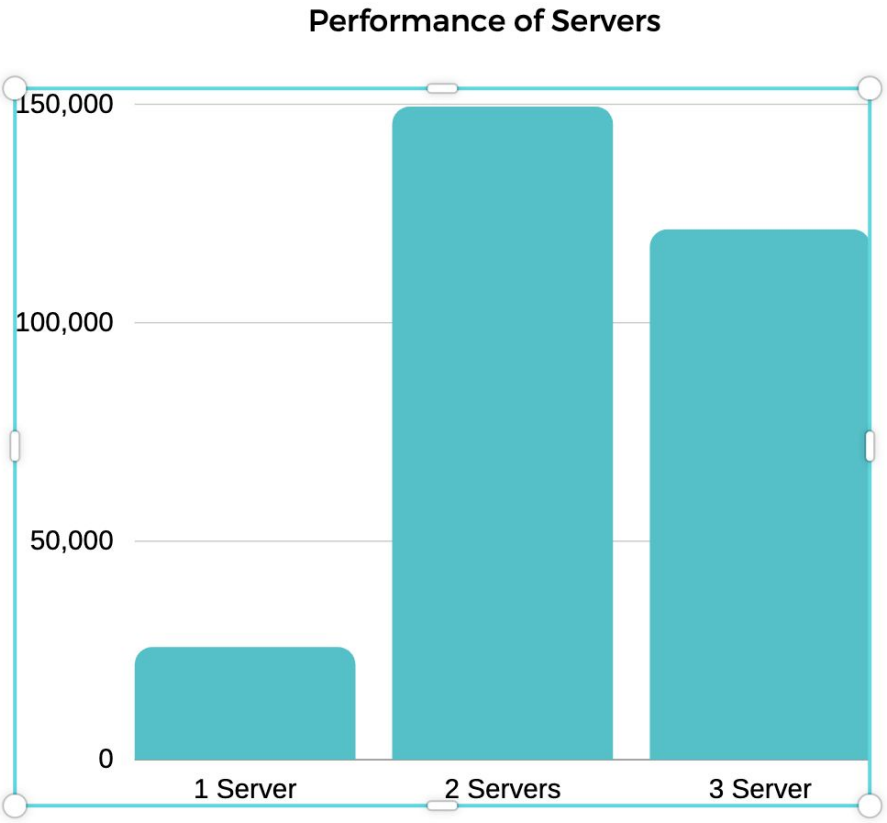
ii. Latency



iii. Analysis

1. 由上圖以及表格可以看出從一個server變成兩個server, performance 有明顯上升，推測是partition造成的近似平行query造成加速
2. 然而，在2個server到3個server後效能有些下降，且到了4個server，電腦直接當掉完全跑不起來，於是我們觀察cpu 以及ram 的使用，發現到3個server的時候已經使用過量，因此我們推測效能不足。

Num of Server	1	2	3
Throughput	25640	149326	121223
Latency	12(ms)	2(ms)	2(ms)


Performance of Servers
150,000
100,000
50,000
0
1 Server
2 Servers
3 Server