Database Assignment 4 Report 1

TEAM 6 107062318 李俊逸 107062202 陳敬和 107062237 張濬洋

Briefly explain what you exactly do for optimization.

Buffer

1. Lock Striping

Add lock striping on the bufferPoolMgr to improve the concurrency performance and also prevent the race condition problem happening.

File

1. Read-Write Lock

Add Read Write Lock on both Jaydio and Javanio's ioBuffer functions to improve the read and write concurrency. (Details can be found in the experiment results.)

2. Lock Striping

We try to use Lock Striping technique to optimize the file performance. First, we add a new fuction getSubFileIdByFileName() to divide locking into several buckets, then remove the "synchronized" in method which calls getFileChannel() function.

Next, we modify the getFileChannel() and delete() as below captures show to implement the lock striping technique.

```
private IoChannel getFileChannel(String fileName) throws IOException {
    synchronized(getSubFileIdByFileName(fileName))  {
        IoChannel fileChannel = openFiles.get(fileName);
```

```
public void delete(String fileName) {
    try {
        synchronized(getSubFileIdByFileName(fileName))  {
            // Close file, if it was opened
```

```
private Object getSubFileIdByFileName(String fileName) {
    return subFile[fileName.hashCode() % sizeOfSubFile];
}
```

>>Part of the modified methods

Experiments

1. Experiment environment

MacBook Air (M1, 2020), Apple M1 chip, 16 GB RAM, 256 GB SSD, macOS Big Sur Version 11.2

2. Buffer package optimization experiment results

2-1. BufferPoolMgr - Add Lock Striping (synchronized on file)

Settings

Dataset = TPCC

# Warehouse = 1
RTE = 2
BufferSize = 1024

Results

Origin Version	Optimized Version	Speed Up
37382	39222	5%

Explanation & Observation

Lock striping是一種分流的技術，使用者獲取資料的對象不是整個資料結構，而是一個資料結構中的一個bucket或stripe。使用上述技術將thread分群，讓各個群的thread搶各自的lock，既能避免race condition，也能用以提升效能 (striping)。

3. File package optimization experiment result

3-1. ioBuffer - Add Read Write Lock

Settings

Dataset = Micro Dataset

NUM_RTES=2
RW_TX_RATE=0.5
TOTAL_READ_COUNT=200
LOCAL_HOT_COUNT=1
WRITE_RATIO_IN_RW_TX=0.1
HOT_CONFLICT_RATE=0.001

Results

Origin Version	Optimized Version	Speed Up
34489	39872	16%

Explanation & Observation
為ioBuffer的function加上ReadWriteLock後，使得多個thread可以同時執行讀，或同時執行寫的動作，故throughput會提升。(沒加的時候，同時只能有一個thread read or write)

3-2. Lock Striping on FileMgr getFileChannel

Settings
Dataset = TPCC
# Warehouse = 1
RTE = 2
bufferSize = 1024, 100

Results

Buffer Size	Origin Version	Optimized Version	Speed Up
1024	39801	40611	2%
100	36537	38964	7%

Explanation & Observation
由於當VanillaCore有很大的bufferPool時，難以看出File的優化，故我們將size從1024調整為100進行實驗。如同我們在BufferPoolMgr中使用lock striping技術，將thread分群，讓各個群的thread搶各自的lock，既能避免race condition，也能用以提升效能。

4. Overall Optimization Results

Settings
Dataset = TPCC
# Warehouse = 1
RTE = 2
bufferSize = 100

Results

Buffer Size	Origin Version	Optimized Version	Speed Up
1024	35857	40571	13%
300	35428	37632	6%

Explanation & Observation
Overall Optimization在buffer size = 300時的提升6%，相較buffer size = 1024的環境下，效能13%，明顯降低。我們認為有以下三個原因:

1. fileMgr在buffer size減小時的效能增益不大。fileMgr的實驗數據中，在buffer size縮小10倍時，速度僅提升5%。

2. 在buffer size縮小時，beffer pool mgr的 lock striping效能提升有限（我們的lock striping分成30群）。

3. ioBuffer的效能提升雖然相對較高，但與overall測試的實驗環境不同；此外我們有測試iobuffer在overall的環境下的表現，效能並沒有顯著提升。

Discuss and conclude why your optimization works

我們在本次作業中主要運用兩項Concurrency優化的常用方法，分別是Lock Striping和Read Write Lock，在File, Buffer class的三處。我們的主要優化想法是來自於助教在課堂中的hint，他有提到Buffer, File都不應該在class的層級同步，而是應該以buffer, file的level來同步，縮小同步範圍，也就意味者減少不必要的等待，提升效能。

實驗數據當中，效能優化的程度並不明顯，我們需要不斷調整實驗環境，將環境盡量地調整成high concurrency時才能跑得比較好的環境。礙於時間因素，我們沒有實作與尋找更多其他能更明顯優化效能的技術，希望能在之後的lab學習。

另外，由於電腦效能不足，我們也不能開太高的#RTE，太多將造成多數的commit都會被捨棄。若電腦效能高一些，或許能將優化效能程度拉高。