Assignment 4 Phase 2 Report

110062229 翁語辰

110081014 程詩柔

110062171 陳彥成

buffer優化比較:

Buffer.java

助教:

- 多了兩個lock，分別是swapLock和 flushLock 可以做更細緻的 Lock striping。縮減了更多synchronized部分。

我們:

- 而我們只有在主要的幾個方法應用，使用contentLock。但是setVal和flush等就沒有做細緻的優化。

BufferMgr.java

助教:

- pinNew中多實現一個waitOnce的優化，unpin中也有類似的優化。可以保證不會出現livelock，保證每個都至少等一次。而且這樣的通知機制可以減少多餘的競爭。
- 在pin方法中用try-catch包起來，當出錯或中斷可以throws exception方便處理問題。

我們:

- 與助教大致相同，但flushAll、flushAllMyBuffers、available、unpinAll等，只有將synchronized寫到方法裡面，但其實可以直接拿掉。

BufferPoolMgr.java

助教:

- 用ReentrantLock 實作了 fileLocks, blockLocks矩陣的優化，提供了兩個 Lock strinping的方法。

我們:

- 在pin函數用一個ReentrantLock來防止race condition，但沒有控制lock的提前釋放的優化內容。pinNew只使用一個lock。所比多個lock的優勢差上許多，因為會大大的降低並行度。實際上不同的block和buffer可以被不同的thread所處理。循環選擇buffer雖然消耗一些時間，但確保每個thread被執行的公平性。

file優化比較:

FileMgr.java

助教:

- 由於synchronized會使一次只有一個thread能進入function中，所以適當移除可以提升運行效率，移除read,write,append,isNew的synchronized。

- getFileChannel()和delete()的synchronized中，利用hashCode()更快速的access檔案
- isFileEmpty中當發生cache miss或是file是空的會在找一次filesize

我們:

- 我們當時覺得write,append會去更動到原先檔案內容，所以必須做synchronized避免race condition，所以並沒有移除synchronized，反而是改為只針對function中的fileChannel.write()以及fileChannel.append()部分做synchronized。
- 此外我們認為getFileChannel只是要取得filename所對應的fileChannel所以可以synchronized移除，不過後來發現如果該file並未打開，會需要將其open並將channel append到map上所以這裡確實有synchronized的必要。

Page.java

助教:

- 移除read,write,append的synchronized

我們

- 作法與助教相同。

Block.java

助教:

- 利用hashCode() 及toString() 方法直接取出恆定的計算結果。減少call function次數

我們:

- 為了避免重複的計算，我們將toString()和hashCode()的計算部分在建構函數中調用一次，再存成實例的properties，而後便可以直接返回。不必多次計算。我們的修改方向與助教相同，但助教的方法比較簡潔快速。

總結:

助教相比我們做了更多的優化，例如使用更多lock，移除更多的synchronized，並改寫code讓function可以更快被調用。還有提供lock stripping、waiti one 等細節方法。

實際實驗效能比較:
我們:TOTAL - committed: 42034, aborted: 0, avg latency: 3 ms
助教:TOTAL - committed:49021, aborted: 1, avg latency: 2 ms
助教的優化方式總體而言比我們多了7000的committed。avg latency 也比我們少1ms, 可以驗證我們上述的討論。