Database Assignment 4 Report 2

TEAM 6 107062318 李俊逸 107062202 陳敬和 107062237 張濬洋

我們新增的優化有以下:

- FileMgr.java
  - Lock Striping

    作法跟解答類似，新建一個object list，讓function同步在這個list的element上，而不是同步在整個class。
- ioChannel
  - ReadWriteLock

    作法跟解答類似，新增ReadWriteLock，使得thread可以同時讀取，在read相對write的操作多時可以提升運行效率。
- BufferPoolMgr
  - LockStriping

    我們的LockStriping的lock是讓pin()及pinNew()同步於在fileName；解答則是分別建立fileLock和blockLock，並建立獲取lock的function，不同object也可以透過function取用相同的lock。此外，解答還有early lock release的優化。

    我們認為解答的lock寫法較佳，因為file跟block是不同object，取用各自的lock運行效能會比較好。

我們沒有實作，但解答有的優化有以下:

- Buffer.java
  - ContentLock，這跟ioBuffer的ReadWriteLock的機制相似，改變locking的機制減少不必要的synchronization overhead，提升運行效率。
  - delete synchronized keyword of block()
- BlockId.java
  - 把toString()和hashCode()改到constructor做一次就好，之後直接取用。
- FileMgr.java
  - isFileEmpty() caching機制，如果有access過file，直接查表即可知道file是否為空。
- ioBuffer
  - fileSize caching + 自己maintain fileSize，access file size就不用透過fileChannel。

其他:

- FileMgr.java

  在解答的第84行跟127行，兩邊都有重複新建object，應只需做一邊？
- Buffer.java

  Optimization中有swapLock的機制，目前我們無法理解為什麼解答的code可以正常運作，待助教時間或助教講解再詢問之。

與解答比較後的總體心得:

解答在storage部分優化得非常全面, trace code的過程中可以學習到很多high concurrency的優化機制, 也能在比較自己的implementation和解答差異的過程中, 深入研究不同的lock的適用場景及如何實作, 我們在本次trace code時更加瞭解如何對進行concurrency的優化方法。