國立台灣科技大學電子工程系

１１2學年度第 2 學期實務專題

期中進度報告書

透過平行化設計增進
固態硬碟之效能

組　　別：　1122A2

組　　員：　姓名：　林家慶　學號：　B11002026
姓名：　洪宥丞　學號：　B11002030

指導老師：　吳晉賢

中華民國　　年　　月　　日

題　　目：透過平行化設計增進固態硬態之效能

組員姓名及學號：B11002026林家慶、B11002030洪宥丞

組　　別：1122A2

指 導 老 師：吳晉賢

一、摘　要（進度報告）：

在SSDsim上以不同的工作負載(Workload)模擬成不同的應用程式，每個工作負載將產生數個工作項目(Request)。我們將所有的工作項目列出後，每經過一段時間掃描，形成一資料集(Dataset)。透過蒐集該資料集的資訊後，便能為該資料集設計專屬的通道分配模式(Channel Assignment Pattern)，使得資料集內的工作項目能更有效地被分配在特定記憶體位址中，以簡化SSD後續的處理，進而降低總延遲時間，達到效率提升。

目前已幾乎詳讀完SSDsim的源碼，更認識了架構後也發現了不少問題，我們統整出問題後，擬定解決方式，並開始著手改寫程式碼。

二、簡　介：

由於手機、電腦等科技逐漸發達，在使用上人們也越來越追求高處理速度、高效能、便利性等。影響電腦處理速度的原因，除了有優良的處理器之外，記憶體的工作順暢度也扮演著很重要的角色！假設我們在同時使用Google瀏覽器查資料，一邊也需要開著Word打報告，甚至同時間再開其他的應用程式，將會產生來自各種不同軟件的待處理工作項目(Requests)。而這些工作將經過序列排成一隊後，經過SSD進行處理，來管理電腦上各種資料的使用空間。

為此，我們打算設計一種演算法，實現並應用平行化處理的概念，以提升SSD的處理效率，進而優化使用者的體驗。

我們期望將request彙整成dataset，從中得取feature，根據feature便可以擬定通道分配策略，使request更有效地被分配在指定的通道上，以提升SSD的處理效率。

目前總共花了四個月左右才總算弄懂SSDsim大致的架構跟工作原理。期中發現了很多與我們目標之間的衝突點，如：SSDsim架構不夠完善、處理資料的格式不配對等等，所以才延遲了進度，想要理解地更透徹再來重新擬定研究方法。

三、研究方法

由於發現SSDsim的處理資料的機制跟我們想像的有所差別，所以我們將開始改寫模擬器，試圖使模擬器更適合我們進行研究。

需要改寫的部分：

1. 原始模擬器中，一次只能從trace file中提取一個request進行處理。這將導致我們無法先掃描資料集，再透過資料集的特徵搭配演算法產生功能性地通道分配策略。

2. 原始模擬器中，每個request經過節點的搬移來模擬處理進度，但總處理時間process time竟然是使用假定的？這導致幾乎每個request可能都有很相近的處理時間，將使得輸出資料不正確。

3. 原始模擬器中，並未模擬SSD的完整處理過程，如：將資料分配到channel、chip、page、die、plane…等等。模擬器存在兩個模擬部分，處理資料(使用節點)以及硬體架構(並未使用節點)，這導致request並未真的被分配到哪條channel，甚至哪塊block被執行，間接地影響我們無法實行通道分配策略，以及觀測結果。

以上三點是目前經過四個月反覆地閱讀源碼以及嘗試寫入自己的程式碼所得出的結論，接下來也將針對這三個部分進行改寫、整合。

四、討論：

關於先前提到的問題(1.)，我們有想到解決辦法。具體上是想先在模擬器輸入了一條request後並進入處理函數之前，先在迴圈外啟動計時器並設一個buffer將request儲存起來，然後跳過處理函數，使迴圈重新啟動並輸入第二條request再存入buffer，直到計時完成，便傳入flag使計時歸零且停止buffer存入資料，並進入特徵函數。

特徵函數內我們將把buffer內的資料依照讀寫比例及演算法計算出適合該request的通道為何，並一併紀錄在該request的封包裡。

接著離開特徵函數，此時dataset裡面的每個request都被改寫(多了通道的資訊)，而因為源碼中的分配函數以及處理函數一次只能處理一個request。所以接下來有兩種做法及可能涉及到的問題：

1. 離開特徵函數後，將buffer裡的新request依序放入分配函數以及處理函數，符合源碼的設計。但有機率因為先前等待request被存入buffer的時間，加上處理每條request的時間，可能延遲了預期的latency，稍微跟我們預期結果有差距。

2. 離開特徵函數後，一次傳入整個dataset給分配函數以及處理函數，但由於源碼中的函數一次只能處理單一request，所以可能需要大量改寫兩個函數的程式碼，使其可以完整操作，也可能會更貼近我們預期的結果。

3. 我們需要想辦法使處理函數確實應用到channel以及後續的硬體介面，所以處理函數勢必需要進行大量的改寫，以進行接軌，並提供更健全、真實的輸出數據。

五、時間進度表：

預定進度甘梯圖（Gantt Chart）：

月 次 工作項目	第1月	第2月	第3月	第4月	第5月	第6月	第7月	第8月	第9月	第10月	第11月	第12月	備註
Linux環境建置													
Ssdsim 程式碼解讀													
增加平行化處理(Feature Collector、演算法開發													
靜態與動態的資料分配													
優化結果與最後修改													
準備專題成果發表													
預定進度累計百分比													

六、參考資料：

SSDKeeper: Self-Adapting Channel Allocation to Improve the Performance of SSD Devices
https://ieeexplore.ieee.org/document/9139770

Parallelism-Aware Channel Partition for Read/Write Interference Mitigation in Solid-State Drives
https://www.mdpi.com/2079-9292/11/23/4048