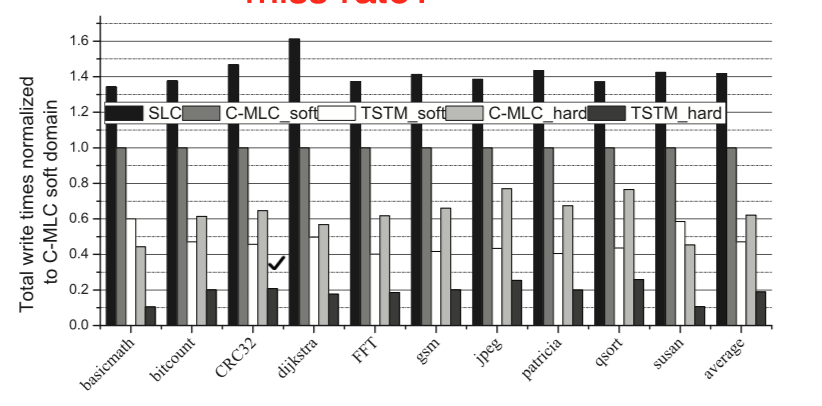
Lifetime of TSTM-MLC STT-RAM



Total write times計算方式：

計算cell 總寫入次數

SLC:

寫一個cell時算一次write times。Ex:512bit data write to 512cell->write times=512

CMLC-Soft:

加總HT\_cnt + ST\_cnt + TT\_cnt\*2

TSTM-Soft:

加總(HT\_cnt + ST\_cnt + TT\_cnt\*2)\*(2/3)

CMLC-Hard:

加總HT\_cnt + TT\_cnt

TSTM-Hard:

加總HT\_cnt + TT\_cnt\*(2/3)(\*註1)

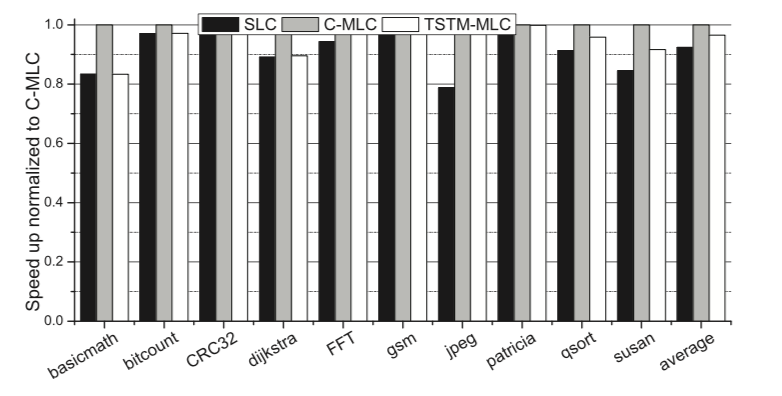
結果分析：

1. SLC的bit數本來就是CMLC以及TSTM的兩倍，所以一定會較高。
2. TSTM因為大幅減少了TT所以total write times的確也是在合理的範圍。(但是數據有小調整，調整原因參考註1.)

註：

1.會乘以(2/3)的理由是因為TSTM他是768 bit,但CMLC是512bit這樣比較起來會有點不公平(下圖是如果不乘以2/3的結果)，tstm及cmlc差異不明顯。

Performance



Latency 計算方式：

SLC :

(Write lat. + Read lat.)\*2(\*註1)

CMLC:

(Write TT lat. + Write Non-TT lat + Read lat.)

TSTM:

(Write TT lat. + Write Non-TT lat + Read lat. + encode lat.)\*(2/3) (\*註2)

結果分析：

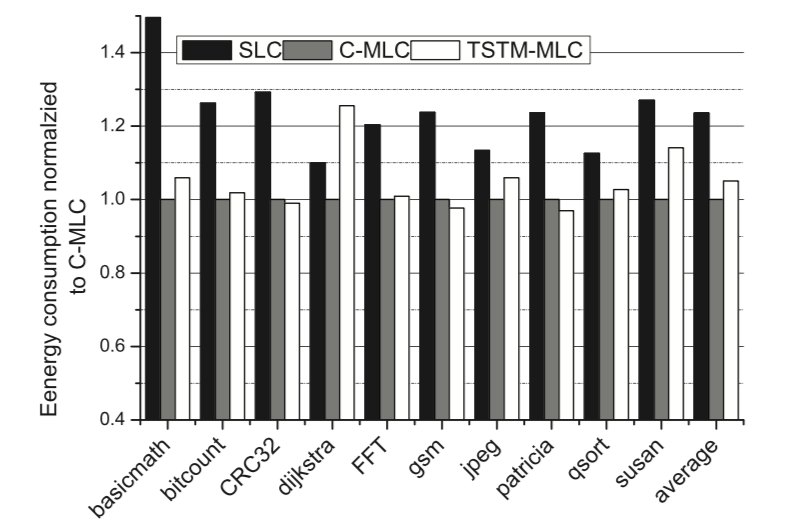
1. 這張圖表一直不太確定怎麼做的像paper一樣，理論上來說會用以cacheline算它的latency,可是不知道為什麼如果不做調整真的很難解釋結果。跟代傑比較後發現他是用cell為單位下去算,但我覺得這樣不太合理。

註:

1.把SLC(\*2)是預想如果做512bit的data, SLC lat會是2倍

2.把TSTM(\*2/3)是預想如果做512bit的data, TSTM lat會是2/3倍

Energy



energy計算方式：

SLC:

Write energy + read energy

CMLC:

(Write TT energy + Write Non-TT energy + Read energy)

TSTM:

(Write TT energy + Write Non-TT energy + (Read energy before encode) + Read energy)

結果分析:

1. 以TSTM來說，每次要做data編碼前都要先做read的動作,其實這個動作非常耗能。但是paper上TSTM都只高出CMLC一點點,這點讓我覺得很不合理。
2. Paper上有些trace TSTM的energy < CMLC energy,這部分我比較不確定為什麼，因為(Read energy before encode)這邊所造成overhead遠遠大於CMLC所以我目前無法做出這個結果。

總結：

暑期目標＆計畫:

TSTM方法實現，了解cache如何運作並且將其實作後加入paper方法嘗試是否能將其實現。

遇到的困難與解決方法：

起初讀paper的時候其實對cache的架構非常陌生，之前都只是課堂上聽過大概的架構，並沒有很仔細的去研究那些inclusive/non-inclusive cache或是write back/through機制等等。所以在實作cache的時候有遇到蠻多細節需要討論的部分,透過大家的討論多次的修正，ㄧ再修改流程圖及程式架構的部分，才完成了這次的實作練習。

心得感想：

暑假的東西對我來說得確難度不低,但好在有碩一同學們一起討論研究實作方法，真的給了我很大的幫助，但有點可惜的就是比較少時間大家聚在一起討論。過了這個暑假學到最多的東西，大概就是two-level cache的所有詳細機制,還有學到未來自己可能可以怎麼去設計實驗，以及可以何種圖表(Ex. Energy…)去呈現並說服別人提出的方法有讓問題得到改善。

建議：

若下屆學弟妹也會跟我們做類似的東西,建議可以讓他們不去實習，在實驗室一起做比較方便討論～