1. Write buffer：在SSD裡面放一塊RAM，來減少對SSD的寫入
   * 收集random write，裝滿write buffer後，在整批寫入SSD，減少寫入次數，改善效能
     + 利用write buffer特性，將頻繁寫入的資料留在write buffer，減少response time
     + 因為將頻繁寫入的資料留在write buffer，可直接對write buffer進行in place update，等到寫入動作完全結束後 or過一段時間後，再將新的資料整批（整個write buffer）寫入SSD->可減少SSD overwritten 的次數
       - 在寫回SSD時對資料進行排程 (FIFO…)->改善效能
   * Victim block：
     + 當write buffer滿了，而且write buffer內存在一些相較於新的write request而言沒那麼頻繁寫入的資料，可以將那些資料挑選為victim block，先寫入SSD，然後用新進的資料取代掉(in-place update)被選為victim block的資料
   * Hit ratio
     + 這裡指的不是放不放進write buffer，因為一定會放進去，而是指，放入write buffer的資料是否會被用到，以下舉一個，hit ratio=0%的例子來說明
       - 一開始，來了很多write request，所以程式將那些資料放入write buffer，之後又有新的request來，但不存在於write buffer中，於是程式挑選block踢掉，此時新的request剛好又是不存在於write buffer中的data page，於是程式又將另一個block踢掉
       - 然後因為預測極差的情況下，雖然一直踢，一直寫，但始終沒有write hit到任何當下在write buffer中的data page，最後，程式結束，hit ratio=0
2. Background information
   * Frequency
     + high：常常重複存取，代表資料有可能是random write->have high probability to be small write
       - 這類型資料會盡可能讓他留在write buffer，因為可藉由write buffer來減少response time
     + low：比較少重複存取，代表資料有可能是sequential write->have high probability to be large write.
       - Sequential write不一定需要放入write buffer改善效能，sequential write就算在HDD也有不錯的效能，SSD也是，大可直接寫入host端
3. Benefit function(x)=()\*()
   * Formula explanation
     + 以block為單位
     + write count：寫入該page次數總和
       - 有可能產生問題：許久以前常常寫入，最近很少寫入，導致write count很高，但實際上位於LRU端的狀況
       - 解決方式：
         * 雖然在訓練、預測時會因為上述狀況而有錯誤的判斷，但因為online有duration label可以避免這種情形
         * Online 時會將Duration label做demoting，很少被寫入的資料會漸漸的被踢掉，因此就算程式預測錯誤，也沒關係，eviction policy可以彌補這部分
     + ()：根據write count & block size，來算出單位page被寫入的頻率，但這會少考慮到一種情形：假設block 1有三個page，block 2有4個page，block 1, block 2每個page各自被寫入一次，這時候依照上述公式所算出的benefit值都會是1，沒將large block優先踢除
     + ()：上面沒能解決的問題，若再加上這個公式，就能夠成功被解決，因為多除以一個block size，就能夠讓large block assign smaller value，因為large block會除以更大的數字(block size)，由此將large block 給予更小的benefit value，來達成讓large block優先被剔除的目的
   * 考慮的點：
     + (A)踢掉small block，但裡面都是不常寫入的資料 VS (B)踢掉large block，但裡面有一些常被寫入的資料
       - A：須額外將沒更新的資料複製到新的block，增加GC額外的寫入動作，減少SSD壽命
       - B：將常寫入的資料踢掉，之後很可能會需要特地到SSD讀/寫，從SSD讀/寫，速度遠不如在write buffer 讀/寫，造成response time增加
       - 結論：就算增加response time，也比減少SSD壽命來的好，因此優先考量(B)
   * 額外須知：
     + 一定要將存取過的block放到MRU端，靠近MRU端前20% block絕對不挑選為victim block
   * 意義：將每個block的平均benefit來做評估，越小，越快被踢出write buffer(when write buffer is full)
   * Priority based