A kind of algorithm that extend life expectancy by 3 folds :
Renice Non-Balance Wear Leveling Algorithm

The first portion of this article introduces the background of wear leveling and some commonly used algorithm while the second portion covers a kind of wear leveling that can prolong life by three folds : Non-balance Algorithm. Experts may skip directly to the later portion.

In order to prolong the life cycle of NAND flash by avoiding some blocks being frequently erased and becoming bad blocks rapidly while the existence of other less erased blocks will cause an unbalance Program/Erase situation that shorten the overall SSD product life as a drawback. There are various algorithm towards wear leveling and garbage collection raised by NAND flash practitioner as strategy to avoid these potential problems.

Background of Wear Leveling Algorithm

The wear leveling algorithm is based on the basic characteristics of flash memory :

1. Do not support local update (outplace-update that is data in original location cannot be overwrite or change directly, this data will need to be relocated to a new available page while the original location labeled as an invalid page and being erased before this original location can be written again);

2. Take the basic unit of page for write and block for erase. To erase block, the valid data of selected page will need to be relocated. When large number of block are chosen for erase, the data relocation of pages in the block for subsequent reuse will determine the erase counts of the different related blocks;

3. Each block has a limited number of times for erase where frequently erased block will soon become a "bad block". Therefore, the balance on times of erase for each block can allow a longer life of the flash memory.

FTL usually view page as three different types: valid page, invalid page and free page, if there is a matching physical page to a logical address, the corresponding page data is valid or known as a valid page. Conversely, it will be an invalid page and become a free page after garbage collection operation where the invalid page being erased. From this point, data can be written to the free page.

For example, when logical address A (assume corresponding physical address 1) data needs to be revised, it is not possible to modify address A data directly but to read this data to cache for modification and write the modified data to new physical address 6 with mapping to logical address A. TRIM will then mark physical address 1 to be invalid. When free pages get lesser, garbage collection will be activated to erase invalid pages for free pages. At this point, questions arising as :

As the basic unit for erase is block, the valid pages in block to be erased need to be relocated first before the block erase. So should block erase be done on block with least valid pages or significantly more valid

pages but less erase counts? Should erase counts of the valid block with cold data be considered? Or is there other condition to select some block for erase? Since garbage collection is directly linked to block erase count, the approach for a balance block erase instead of regular erase of hot data blocks or rare erase of other blocks contributed the background for wear leveling algorithm.

Garbage Collection Strategy

Wear leveling refers to implementation of garbage collection process as the target block selection, regulation or mechanism to ensure the erase counts of each block being balance.

A simple strategy can shorten the execution time of block recovery but result in uneven wear of each block while a complex strategy may balance in calculation but lower system efficiency. As such, a good recovery implementation will need a good balance between system efficiency and count of wear.

For the various garbage collection strategies, there are basic Greedy algorithm (select block with least valid page to recover) to the gradual revolutionized Cost-Benefit algorithm (consider block erase frequency with reference of cold and hot data), the formula as follows:

$$\frac{1-\mu}{2\mu} \times age$$

Then the CAT recovery algorithm, based on the Cost-Benefit that consider the number of block erase factors, the need for block recovery and the choice of smallest value block to recover. The formula as follows:

$$\frac{\mu}{1-\mu} \times \frac{1}{age} \times NumberOfCleaning$$

And CICL algorithm is the most optimized, needing recovery block effective page number and block erase counts as the two factors average to decide. 1 to decide the factors weights, 0 if the average number of block erase are the same whereby valid page number of recovery block can be considered. Otherwise, you will need to consider these two factors. The formula as follows:

$$\text{Clean Index} = (1-l)\cdot\mu_i + l\cdot\frac{\varepsilon_i}{\varepsilon_{\max}+1}\ ,\ l = \begin{cases} \dfrac{2}{1+e^{k_\varepsilon/\Delta_\varepsilon}}, & \text{if } \Delta_\varepsilon \neq 0 \\ 0, & \text{if } \Delta_\varepsilon = 0 \end{cases}$$

(ε : Block Erase Count μ : Block Valid Page Δ_ε : $\varepsilon_{max}-\varepsilon_{min}$)

Based on the basic idea of above garbage collection strategy, various Controller manufacturers have

evolved a variety of different garbage collection algorithms, as well as hot and cold data analysis method.

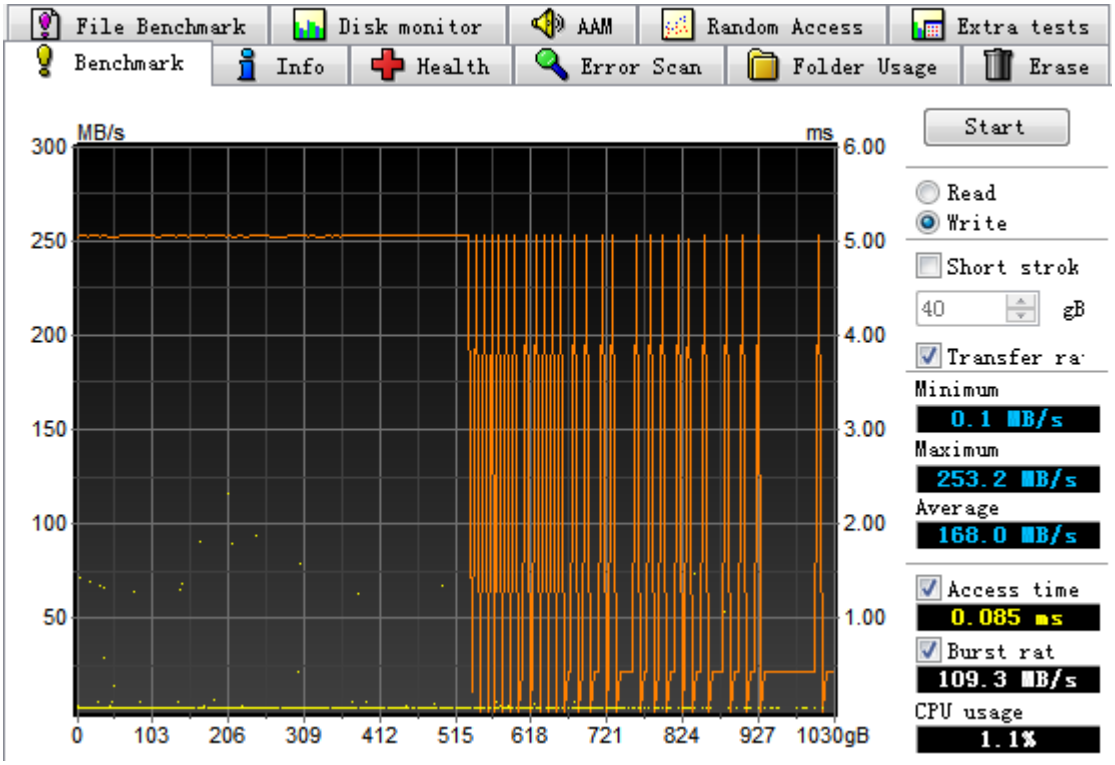
Passive and Active Recovery Strategy

Garbage collection has two important issues to consider: the timing of the recovery of the invalid block and the number of each recovery. Therefore, it is also related to two kinds of recovery strategies: passive recovery strategy and active recovery strategy.

Passive recovery strategy is when there is a write request, the system judges according to the present conditions whether to perform garbage collection. For this strategy, the system will usually refer a critical value set on available space and start garbage collection when the available space is less than the critical value. The amount of space to recover each time can be defined by the firmware engineer.

The disadvantage of this strategy is: during implementation of garbage collection, write requests will be delayed such that the higher number of blocks to recover, the longer the delay time and the write performance could fall rapidly. For ordinary end-users, they may even feel some system stoppage or down in operation. For user engaged in data logging, there will be high chance of data lost when garbage collection is being performed.

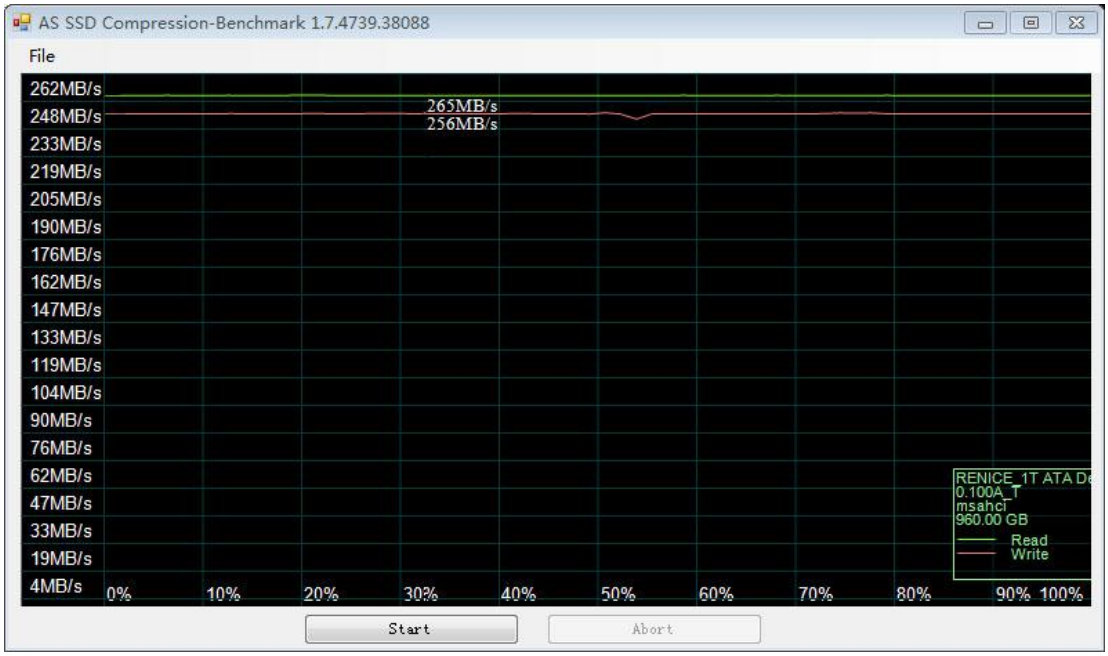
Below is the performance of passive garbage recovery mechanism:



Active recovery strategy is to use the system idle time to perform garbage collection using firmware setting as periodical task and regular check on remaining available blocks if there is a need to carry out garbage collection. Active recovery strategy is equivalent to the use of idle time in advance for garbage collection, to avoid compromise on performance as that from the passive strategy.

The sole reliance on system idle time to perform active garbage collection can lead to drawbacks as well assuming that there has been no systems idle time and garbage collection cannot be executed effectively. For example: urban road video surveillance where flash memory will be written constantly. It is under this kind of situation that both garbage collection strategies are essentially the same.

(the following content involves commercial secret and only general introduction, please do not consult any specific method for realization) Renice adopted active recovery strategy for optimization: PR-latency techniques. Meanwhile according to different application scenarios couple with automatically adjustable OP space, SSD can maintain the same performance even at 100% write/read loading. Using IOMeter and setting 2MB pattern under 100% write mode for continuous test, a straight line performance curve can still be observed.



Static and Dynamic Wear Leveling

Dynamic wear leveling is when a page of data needs to be changed, new data being written to lesser erased physical page. At the same time, the original page is marked as invalid page. The disadvantage of dynamic wear leveling being if recent written data needs updating shortly, corresponding block will be marked as invalid page. With such kind of frequent data renewal, it will give cold data blocks little chance for erase and overall likely to impact life expectancy of the flash memory. Dynamic wear leveling has been used commonly in early generation SSD controller although current ones utilize both dynamic and static combination.

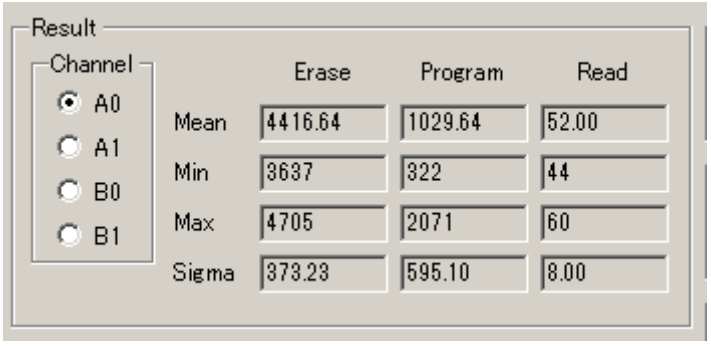
Static wear leveling is considering for blocks (cold data) with low chance of data update. For example: blocks with system or some read-only data where such cold data block is updated less frequently as those with hot data. Static wear leveling algorithm will include these cold data blocks for wear balancing and hereby increase overall life expectancy of the flash memory.

There are many static wear leveling algorithm and none can claim as a perfect one. They basically evolved from generations and dependent on different application demands that may lead to a different choice of algorithm.

A kind of algorithm that extend life expectancy by 3 folds : Renice Non-Balance Wear Leveling Algorithm

Current garbage collection algorithms surround on even distribution of block erase for design and they assume wear threshold of the blocks are identical basically. In fact, the wear value of each page and block are different. When every block of a flash memory has been erased evenly, some will become bad block first inevitably while others can last longer. This is dictated by the wafer design technology and current process cannot ensure each block to be identical. Furthermore, there bound to have imperfection in wafer and they are randomly distributed beyond control.

Below new NAND flash (model: JS29F16B08CCME2, Intel 16GB MLC) after 10 times Program/Erase. The results showed that each block's erase time is different whereby different operating time implies the original state of each block is different. One block with shorter operating time should be in better health condition in theory than another having longer time.



If the wear resistance of each block is not balanced and artificial algorithm assume block erase as identical, this will not extend life expectancy of the flash memory but instead obstruct their optimal life usage.

Renice Non-Balance algorithm idea as: An able man is always busy. Against the assumption of "each block have identical life" as basis of the balancing algorithm, each page and block is evaluated for their degree of wear resistance during operation. The higher wear resistance blocks are erased more times while the lower ones get protected instead. Only when every block is not a bad block, this will significantly reduce the possible block erase count. For example: 1TB of data to be written on 100 blocks and 1000 blocks. Assuming the 100 blocks need to be erased 10 times, then the 1000 blocks will only be erased 1 time.

Another key of Non-Balance algorithm is when the block UBER (Uncorrectable Bit Error Rate) reaches a certain threshold, MLC mode should be switched automatically to SLC mode.

The critical technical challenge of Non-Balance algorithm is to judge the true wear resistance of flash memory (example 3000 P/E cycle for page1, 5000 P/E cycle for Page3).

There are many methods to test the page real wear resistance through some key indicator or several in combination for judgment. It can also be done under high or low temperature conditions for accelerated test with key performance indicators of a certain type of NAND flash set in advance. The idea for specific practice as follows:

Through destructive test mode on a certain type of NAND flash, proceed thorough P/E test for the NAND flash to record their RBER (raw bit error rate), UBER and operation time. From these accurate data consolidated, a mathematical model can be established among them in order to define different bit error rates (RBER and raw bit error under low intensity ECC), operation time corresponding to page real wear resistance and high temperature accelerated test to refine this mathematical model.

Although the practical operational view may highly increase the complexity (testing is a very time consuming process and the need for constant refining of the mathematical model to enhance its accuracy), the actual effect will be far better than practice that do not consider the actual page wear resistance. Technical progress starts very often from details and patience.

SSD is the best practitioner of cask theory whereby SSD overall life expectancy relies on that weakest piece of flash. Therefore, the screening for consistent wear resistance flash will decide how long the SSD can last. Otherwise, no matter how good is the algorithm, it will eventually be defeated by the weakest piece of the flash.