Analysis of Garbage Collector Algorithms in Non-Volatile Memory Devices

Ananth Mahadevan[†]⊕, Mukundan Sridharan⊕, Kenneth W Parker⊕, Rajiv Ramnath[†]
email: {mahadeva}, {Ramnath}@cse.ohio-state.edu
email: {Mukundan.Sridharan}, {Kenneth.Parker}@Samraksh.com
⊕ The Samraksh Company
[†] The Ohio State University

December 30, 2012

1 Abstract

The performance results for five different Garbage Collection (GC) algorithms for Non-Volatile Memory Devices for three access patterns are presented in this report. The access patterns include a long-tailed distribution as well as Uniform distribution. The results indicate that Round-Robin style GC algorithms perform much better in all cases than Generational algorithms. This is counter-intuitive to the existing norms. Invocation of the GC in Flash devices is determined by the fullness of the device. Even at low fullness levels, Generational GCs have very low efficiency. In this paper, we compare the efficiency and the time taken for the individual GCs at fullness levels ranging from 2% to 98%. Existing research looks into using Flash as storage for data and RAM as cache. We analyze the performance of a Flash when it is used in place of a RAM.

2 Introduction

Flash memory is an important non volatile storage media due to its low power consumption and small size. It is increasingly being used as a storage device in embedded devices. Embedded devices are constrained by power and low memory capacity. Hence there is a need to have a very efficient primary storage system that allows fast read and write operations and thereby less power consumption. Flash devices generally have fast read accesses but have very slow write accesses.

Read time(ms)	Write time (ms)	Erase time (ms)
4	5	6

There are 2 major Flash devices available today - NAND and NOR. An important component that has an overhead and affects the performance of the storage system is the Garbage Collector (GC) algorithm. This report quantifies the performance of different GCs against different statistical traffic patterns such as Uniform, Pareto and Bi-Modal. Based on prior experiments, we have observed that the traffic pattern for the data can occur in short bursts or can arrive at regular intervals. This is the reason why we chose to select the afore-mentioned traffic patterns. A simulator for the Flash file system as well as the GC algorithms were coded in Matlab. We compare the performance of five different GC algorithms against three traffic patterns. Our experiments indicate that Round-Robin style algorithms have better efficiency than Generational algorithms.

In Solid State Devices such as NAND and NOR based Flash, new data is always written out-of-place. In a Log-Structured File System, this reduces the amount of free space and a Garbage Collector algorithm is invoked which defragments the device by moving all valid data together and erasing the invalid data. This is a critical factor in the performance and life-time of a Flash device. In this paper, we present our work in analyzing five different GC algorithms and compare their performances against various parameters. We also present the theoretical model behind our simulation and explain the reason behind the results we have obtained. A major contribution of our work is to measure the performance of Flash devices against traffic patterns that are generally observed in practice. The major goals of our work are:

- To find out if Flash can work as a good primary storage system.
- To create performance benchmarks and understand which Garbage Collection algorithm is better.
- To create statistical models to test the GC algorithms.

We simulated an application which is both equally read and write dominant. We also tested an application which has only writes.

The report is organized as follows: section 3 gives details on related work being conducted currently. Section 4 mentions about the current state-of-the-art in Flash algorithms and section 5 provides details on the Mathematical models behind our simulations. We conclude by outlining our results in section 6, details about our simulator in section 7 and finally talk about future work in this area.

2.1 Methodology

The GC and the applications will be implemented in Matlab 2011b. The tests will be run on the Ohio Super Computer center's cluster called Oakley. The fullness level will range from 2

Implementation details:

The application generates random records and sizes using the rand() function in Matlab. For Pareto distribution, each record is assigned a weight which decides the usage of a record. After the records are generated, based on a coin toss, it is decided whether the next operation will be a read or a write. Every time the GC is accessed, details such as amount of bytes moved, blocks erased, are captured. These details are then used to plot the required graphs.

3 Related Work

4 Modelling

5 Theoretical model - FIFE Uniform

Assumptions:
Total blocks in a flash device, B = 128
The size of a block, BlockSize = 8192
The device size is, deviceSize = B * BlockSize
The fullness level ranges between 1% and 98% and let N be the count of records per fullness level.
Let the count of dummy records per block be 1. dummyRec = 1
Average number of records per block,

$$R_b = \frac{BlockSize}{AvgRecSize} - dummyRec \tag{1}$$

Total number of accesses is, (Number of blocks * Records per block) - (Number of records)

$$NumOfAccess = ((B-1)*Rb) - (N-1) \tag{2}$$

NumOfAccess denotes the number of accesses after which a record in a certain block is accessed again. The equation can also be written as: $((B-1)*Rb) - (B-1)*E(ARB_c)$
where $(B-1)*E(ARB_c)$ is the total active records in the flash, which is N.

((B-1) * Rb) gives the total records (both active and inactive) in the entire flash and subtracting N removes the active records and leaves only the inactive. Therefore after going through the entire set of active and inactive records, we access a certain active record (hence the reason behind subtracting (N-1)). The above equation gives the best case scenario where N records are spread evenly across (B-1) blocks.

The expectation for a record to be hit is $\frac{1}{N}$, which is the pdf for Uniform distribution. Let the probability of missing a record be denoted by P_m.

$$Pm = \frac{(N-1)}{N} \tag{3}$$

Let the expectation of active records per block that is going to be cleaned be denoted by $E(ARB_c)$

Expectation is then, $E(ARB_c) = (P_m)^{NumOfAccess} * Rb$

The intuition behind the equation is that, a record is missed for NumOfAccess times before it is hit. Multiplying by Rb gives the relation for all records in a certain block.

The expectation of the average number of records per block can be of the exponential form $1 - e^{\lambda.x}$ where x is the fullness of flash (%). To find λ we can equate the exponential function to R_b when the flash is 100% full. The corresponding equation is:

$$1 - e^{\lambda.x} = R_b$$

$$\lambda = \frac{log(1 - R_b)}{100}$$

where 100 represents the fullness of the flash in percentage.

Expectation can therefore be given as:

$$E(ARB_c) = (1 - e^{\lambda.x})$$

where x is the fullness of flash (%)

For a Uniform Distribution, let the total accesses to the Flash be: 100,000. Out of this, let 1/2 be writes. Hence totalWrites = 50000

The total number of times Garbage Collector(GC) is called decides the total move cost. During one cycle, when active records are moved from one block (x) to another (y), next invocation of the GC is related to the amount of free space in block y. Hence the relation for number of times GC is invoked is:

$$NumberGCRuns = \frac{totalWrites}{(R_b - E(ARB_c))} \tag{4}$$

where $(R_b - E(ARB_c))$ gives the number of free slots where records can be stored.

The fullness of the Flash is given by:

$$FlashFullness = \frac{((N * AvgRecSize)}{deviceSize} \tag{5}$$

The expected move cost is:

$$E(MoveCost) = E(ARB_c) * AvgRecSize * NumberGCRuns \tag{6}$$

The useful write cost is cost involved in writing data to the Flash without any overhead of moving or erasing records:

$$UsefulWriteCost = totalWrites * AvgRecSize \tag{7}$$

The actual write cost is the cost of erasing and moving records in order to write a new record:

$$ActualWriteCost = UsefulWriteCost + ExpMoveCost \tag{8}$$

Efficiency of a Flash device is then:

$$Efficiency = \frac{UsefulWriteCost}{ActWriteCost} \tag{9}$$

6 Theoretical model - FIFE Pareto

Assumptions:
Total blocks in a flash device, B = 128
The size of a block, BlockSize = 8192
The device size is, deviceSize = B * BlockSize
The fullness level ranges between 1% and 98% and let N be the count of records per fullness level.
Let the count of dummy records per block be 1. dummyRec = 1
Average number of records per block,

$$R_b = \frac{BlockSize}{AvgRecSize} - dummyRec \tag{10}$$

Total number of accesses is, (Number of blocks * Records per block) - (Number of records)

$$NumOfAccess = ((B-1) * Rb) - (N-1) \tag{11}$$

Minimum possible value of a variable X below which it will be accessed with increased frequency

$$X_m = 0.2$$

Positive parameter for Pareto distribution is:

$$\alpha = 1.3$$

Average number of records per block can also be represented as,

$$1 - e^{\lambda . x} = R_b$$

$$\lambda = \frac{log(1-R_b)}{100}$$

where 100 represents the fullness of the flash (%)

The calculated value of λ is used for different substitutions of x (flash fullness) to get the expectation of active records per block that is going to be cleaned.

$$E(ARB_c) = 1 - e^{\lambda . x}$$

The expectation for a record to be hit is the mean of a random variable for Pareto distribution, which is given by:

$$E(X) = \begin{cases} \frac{\alpha . X_m}{\alpha - 1} \; if \alpha > 1 \end{cases}$$

Therefore, expectation of a miss for Pareto distribution is:

$$P_m = 1 - \frac{\alpha . X_m}{\alpha - 1}$$

The expectation of active records in the block set to be cleaned can be assumed to be exponentially distributed. Expectation can be given as:

$$E(ARB_c) = (1 - e^{\lambda . x})$$

where x is the fullness of flash.

7 Theoretical model - 2-Gen Uniform

Assumptions:
Average size of a record = s
Size of Gen1 = F_1
Size of Gen2 = F_2

Total records that can be accomodated in Gen 1, $R_{gen1} = \frac{F_1}{s}$

Total records that can be accomodated in Gen 2, $R_{gen2} = \frac{F_2}{s}$

X_1 = Expectation of Active Records in Gen 1
X_2 = Expectation of Active Records in Gen 2

Probability of a miss, $P_m = \frac{(N-1)}{N}$

Records moved over to Gen 2 during GC is δ_2
Records left over in Gen 1 after GC is $\delta_1 = X_1 - \delta_2$

$$X_1 = \delta_1 + \delta_2 \tag{12}$$

$$X_1 = {P_m}^{A_1} * R_{gen1} \tag{13}$$

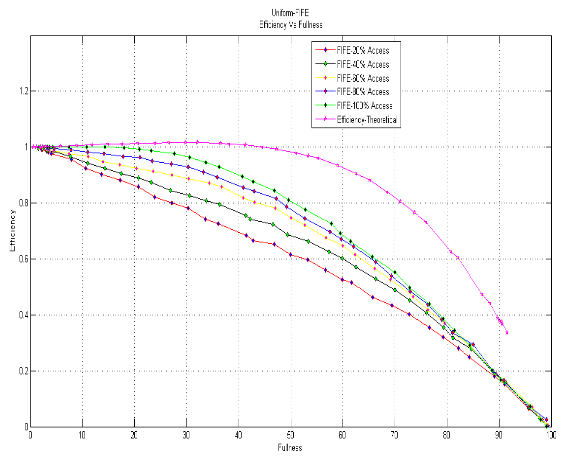
$$X_2 = {P_m}^{A_2} * R_{gen2} \tag{14}$$

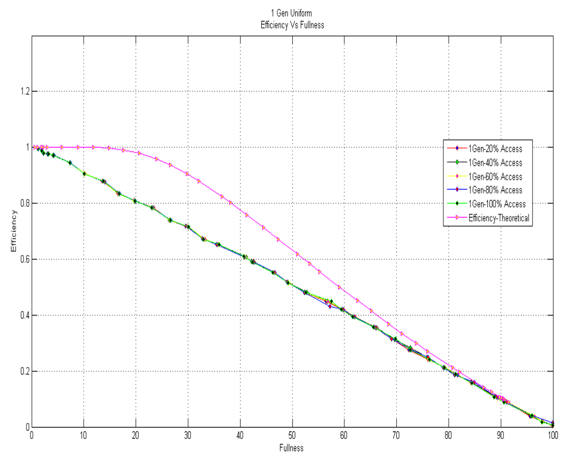
Count of accesses between cleans of Gen 1, $A_1 = \frac{F_1}{s} - \delta_1$
Count of accesses between cleans of Gen 2, $A_2 = \frac{F_2}{s} - \delta_2$

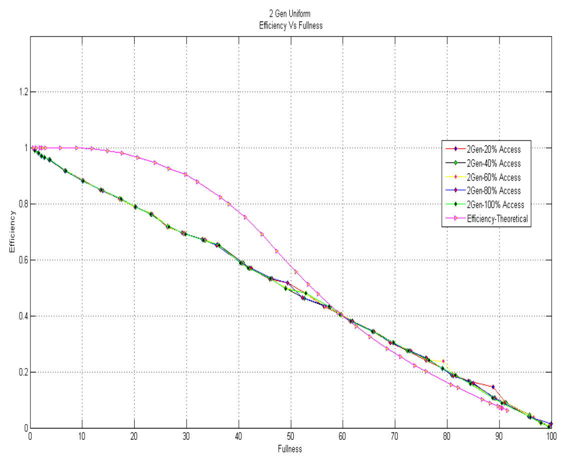
Count of accesses between cleans of Gen 2 is, $A_2 = A_1 * R_{21}$, where R_{21} is factor defined below
Ratio of cleans between Gen 1 and 2 is, $R_{21} = \frac{FreeSpaceRemaininginGen2}{X_1}$

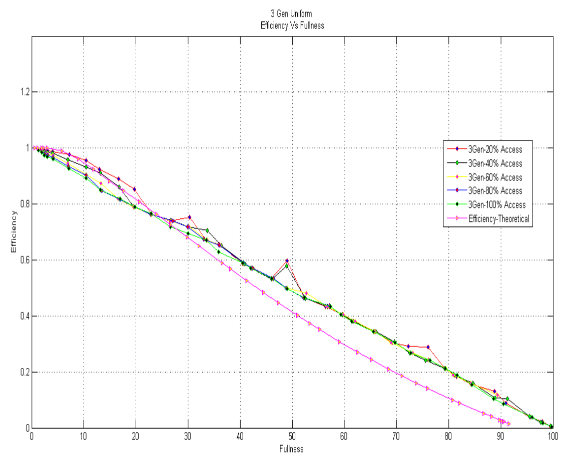
Free Space remaining in Gen 2, $FS_2 = R_{gen2} - X_2$
Therefore, $R_{21} = \frac{FS_2}{X_1}$

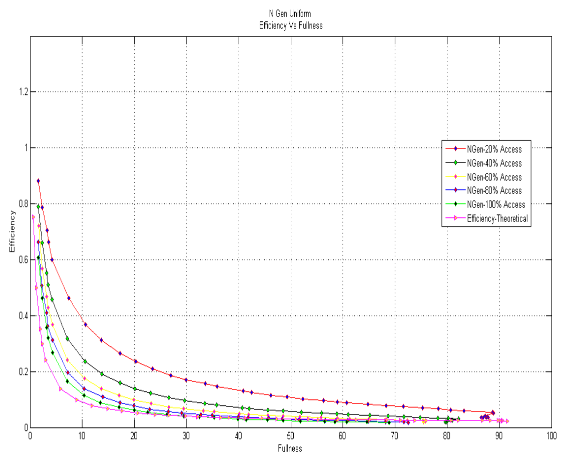
8 Results











9 Simulations

Summary

References

Bibliography

[1] Evgeny Budilovsky, Sivan Toledo, Aviad Zuck. Prototyping a High-Performance Low-Cost Solid-State Disk. *SYSTOR '11 Proceedings of the 4th Annual International Conference on Systems and Storage*