

# Solid State Drives: A New Problem


Dwight A. Haworth, University of Nebraska at Omaha, USA

## ABSTRACT

*This paper explores the characteristics of solid state drives (SSDs) with regard to the probability that an update to the data will require a page be relocated. Storage characteristics of SSDs are described and the problem of drive wear is explained. Probabilities are developed for increases and decreases in value at the byte level. The byte-level probabilities are then extended to the SSD storage page. The implications of those probabilities for drive wear are examined and extended to file design considerations. In addition, the wear implications of current practices are pointed out where appropriate. In the end, the issue of frequency of data update becomes a new consideration that must be taken into account in all phases of system design.*

**Keywords:** Solid State Drives; File Structures; Drive Wear

## INTRODUCTION

 Solid State Drives (SSD) present a whole new challenge for the student of file structures.

Response time, long the Holy Grail of File Structures, is now minimized. Truly impressive access times and transfer rates have been reported in the trade press; Mearian (2009, p. 3) reports that an SSD provided 244 MB/sec read rate versus a HDD that provided a 98MB/sec read rate. Similar improvements in write rate are also reported. The improved performance arises from the fact that there is no seek time and no rotational delay.

On the downside, reliability statements are often vague or given in units that make comparison to hard disk drives impossible. As examples, SanDisk Corporation (2010) uses Long Term Endurance that is expressed in terabytes written over the entire life of the device; Seagate Technology (2010) uses annual failure rates; and Intel (2009) uses the more familiar MTBF. The reluctance of the manufacturers to use units of measure that would allow the reliability of an SSD to be compared to that of a hard disk drive (HDD) almost certainly arises in part from the lack of historical data.

The difficulty in developing comparison figures also comes from the fact that an SSD "wears out" in a way that is very different from an HDD. In an SSD, the wear factor is simply a function of the number of "flashes" that the storage blocks on the drive may suffer. After a certain number of flashes, the insulation around a transistor begins to deteriorate and the transistor will no longer hold its charge, making the retrieval of data unreliable (Ruth, 2008, p. 30).

To understand how NAND flash memory wears out, some additional information is needed. NAND flash memory is organized into blocks and pages. Blocks are the larger unit; pages are smaller units. According to the Wikipedia (2010), pages may be 512K, 2048K, or 4096K. Today we find that a page is often 4KBytes (Intel Corporation, 2009; Ruth, 2008, p. 28), which is consistent with the new standard hard disk sector size adopted by the disk manufacturers in 2007 (Chicoine, et al., 2007, p. 1). Ruth (2008, p. 28) gives the Samsung block size as 64 pages per block; other architectures are discussed in the Wikipedia (2010).

Blocks are initialized to all bits on. All reads and writes are in page-sized units. To write a page, the bits that must be zero to create the appropriate value are set to zero. Once a bit has been set to zero, it cannot be set back to one unless the entire block is reset to all-bits-on, "flashed." This means that pages cannot be rewritten in the same fashion that a sector on a magnetic disk may be rewritten.

If a page has been written with unused space (represented by 1 bits), that space may be used. The only restriction is that the original data cannot be changed in any way that will require a zero-bit to be set to 1. Should any change to the data require a zero-bit be changed to a 1, the entire page must be relocated to a fresh page, and the old page will be marked for recycling. When all of the pages of a block are marked for recycling, the block is reinitialized, "flushed," to all bits on and some small fraction of that block's life is then gone.

The wear caused by frequent updates may substantially shorten the life of the device. A colleague who attempted to use a USB "flash" drive in the same fashion that we have come to use floppy disks, for intensive editing and updating of presentations and other classroom materials, was unpleasantly surprised when the drive failed in four months. For the enterprise, the premature failure of an SSD could escalate the cost of ownership to an unacceptable level.

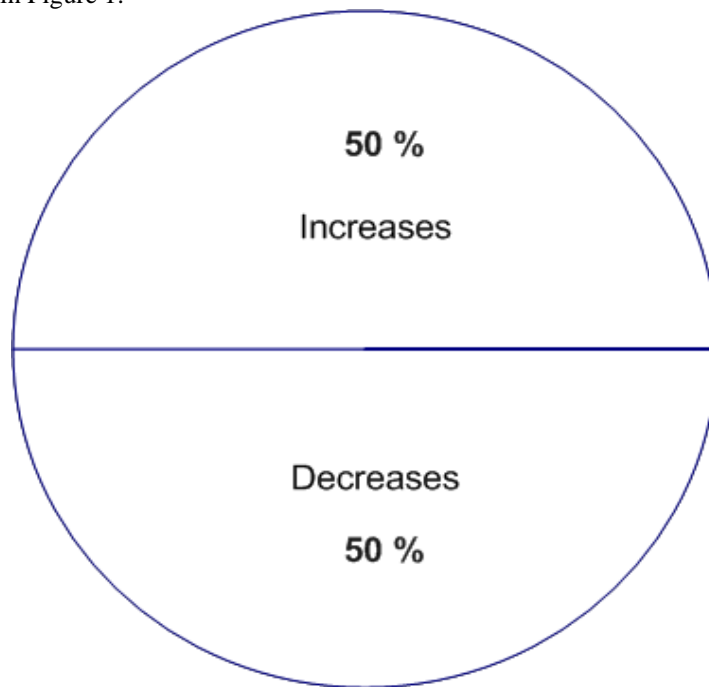
The foregoing provides incentive to investigate the pattern of changes that may arise in interactive updates. Many business systems consist of master records that are subject to interactive updates. Consider an inventory system to which updates to quantity-on-hand are posted. Most of each record is comprised of static fields, fields like uniform product code, product description, source, and so on. Only one field, quantity-on-hand, is changed frequently. It is increased when new shipments arrive and decreased when quantities are sold. It is useful to understand the proportion of these changes that require the page containing the record to be relocated. The purpose of this paper is to begin to understand these proportions.

**ANALYSIS**

This analysis will proceed by examining the changes in a single byte. All of the possible values of that byte will be considered, from 0 to 255. After the analysis of changes in a single byte, consideration will be given to the implications of multiple independent changes to a record. One of the assumptions of this analysis is that the page has no unused space; therefore, repositioning the updated record within the same page is not an option.

**INCREASES**

As a first step, a user selects a field for update. The update will result in an increase or decrease to that field, and without a priori knowledge of the user's intent, the probability of an increase or a decrease is the same, that is, 50 percent, as shown in Figure 1.

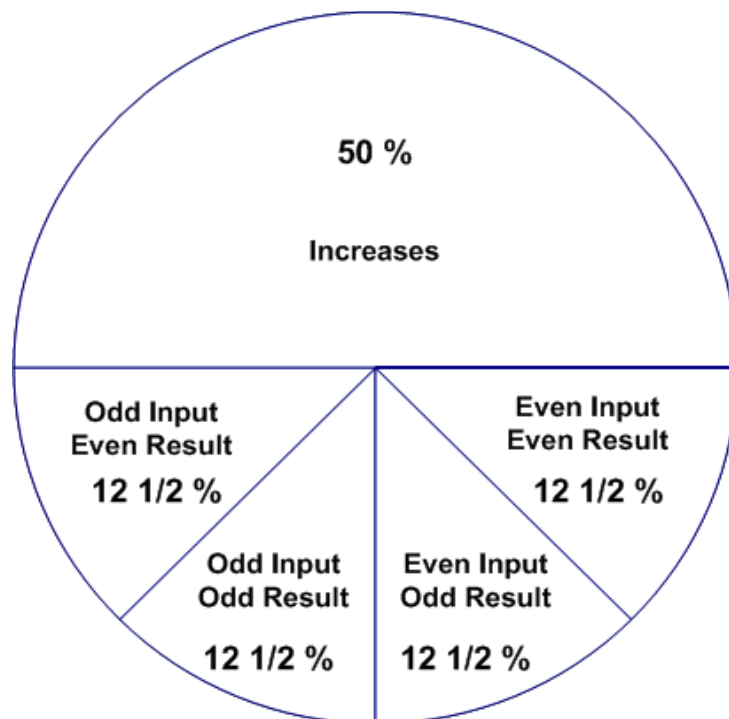


**Figure 1. Proportion of Increases and Decreases**

Any increase in the value of a byte will require that some bit that is zero be made a 1. This will require the page containing that byte be relocated. The conditional probability of a page being relocated given that the update is an increase is  $P(R|I) = 1.0$ .

**DECREASES**

The probability of a decrease is equally likely. Working in the lower half of the figure, it is useful to consider whether the input value is even or odd; this is a 50-50 probability. And in order to evaluate the bit changes, the results must be considered. Again, without a priori knowledge, whether the result is odd or even is a 50-50 probability. This gives four cases to consider: odd-input with even-result, odd-input with odd-result, even-input with odd-result, and even-input with even-result. When these proportions are applied to the decreases, each partition becomes 12.5 percent of the total updates. The situation is shown in Figure 2.



**Figure 2. Proportions of Update Events**

To understand the problem of decreases in value, it is useful to consider the specifics of each case shown in Figure 2. The simplest case is when the decrease from an odd number produces an odd result. In this situation, the low-order bit stays on. The changes occur in the seven higher-order bits. These seven bits represent even values from 2 through 254, 127 values in all. The problem is to take each of these possible starting values and analyze each possible lesser values to determine if one or more of the initial zero bits would be required to be turned on to represent that lesser value.

A selection of a few specific values demonstrates the process. Consider the value 254, all bits on; this value may be reduced to any of the 127 (zero included) lesser values without requiring a relocation. Now consider the value 128, only the high-order bit on. Any decrease, except to zero, will require that one or more of the lower-order bits be turned on and the page be relocated. This happens in 63 of the 64 possible decreases (98.44% of the cases). Table 1 shows a view of the values produced by a spreadsheet set up to aid in this evaluation. Row 1 shows the decimal value of the bits. Row 2 shows the bit setting for the initial value. The remaining rows give an exhaustive listing of all of the lesser values and whether a relocation of the page will be needed.

**Table 1. Values view to evaluate the value 128**

	A	B	C	D	E	F	G	H	I
1	128	64	32	16	8	4	2	Value	Relocate
2	1	0	0	0	0	0	0	128	
3	0	1	1	1	1	1	1	126	1
4	0	1	1	1	1	1	0	124	1
5	0	1	1	1	1	0	1	122	1
6	0	1	1	1	1	0	0	120	1
7	0	1	1	1	0	1	1	118	1
8	0	1	1	1	0	1	0	116	1
9	0	1	1	1	0	0	1	114	1

Note: Rows 10 through 54 omitted.

The analyst must exploit all of the capabilities of the modern spreadsheet to avoid the large number of chances for error that reside in the calculations that must be made. In Table 2 is a formula view of the column containing the values of the bit pattern. This field is a check to insure that all lesser valued bit combinations are evaluated.

**Table 2. Formula view of the values column**

	H
1	Value
2	=(A\$1*A2)+(B\$1*B2)+(C\$1*C2)+(D\$1*D2)+(E\$1*E2)+(F\$1*F2)+(G\$1*G2)
3	=(A\$1*A3)+(B\$1*B3)+(C\$1*C3)+(D\$1*D3)+(E\$1*E3)+(F\$1*F3)+(G\$1*G3)
4	=(A\$1*A4)+(B\$1*B4)+(C\$1*C4)+(D\$1*D4)+(E\$1*E4)+(F\$1*F4)+(G\$1*G4)
5	=(A\$1*A5)+(B\$1*B5)+(C\$1*C5)+(D\$1*D5)+(E\$1*E5)+(F\$1*F5)+(G\$1*G5)
6	=(A\$1*A6)+(B\$1*B6)+(C\$1*C6)+(D\$1*D6)+(E\$1*E6)+(F\$1*F6)+(G\$1*G6)
7	=(A\$1*A7)+(B\$1*B7)+(C\$1*C7)+(D\$1*D7)+(E\$1*E7)+(F\$1*F7)+(G\$1*G7)
8	=(A\$1*A8)+(B\$1*B8)+(C\$1*C8)+(D\$1*D8)+(E\$1*E8)+(F\$1*F8)+(G\$1*G8)
9	=(A\$1*A9)+(B\$1*B9)+(C\$1*C9)+(D\$1*D9)+(E\$1*E9)+(F\$1*F9)+(G\$1*G9)

Note: Rows 10 through 54 and Columns A through G and I omitted.

The other critical aspect of the spreadsheet is the column that calculates whether the change in the bit pattern will require that a page be relocated. This, too, demands the accuracy of the spreadsheet formula. Table 3 shows the formula view of the relocate column. This formula was copied throughout, not only to the rows that are not shown but also to all of the 127 spreadsheets used to evaluate this situation and the spreadsheets used to evaluate the other situations. Of course, some editing was required for each row.

**Table 3. Formula view of the relocate column**

	I
1	Relocate
2	
3	=IF(B3=1,1,IF(C3=1,1,IF(D3=1,1,IF(E3=1,1,IF(F3=1,1,IF(G3=1,1,0))))))
4	=IF(B4=1,1,IF(C4=1,1,IF(D4=1,1,IF(E4=1,1,IF(F4=1,1,IF(G4=1,1,0))))))
5	=IF(B5=1,1,IF(C5=1,1,IF(D5=1,1,IF(E5=1,1,IF(F5=1,1,IF(G5=1,1,0))))))
6	=IF(B6=1,1,IF(C6=1,1,IF(D6=1,1,IF(E6=1,1,IF(F6=1,1,IF(G6=1,1,0))))))
7	=IF(B7=1,1,IF(C7=1,1,IF(D7=1,1,IF(E7=1,1,IF(F7=1,1,IF(G7=1,1,0))))))
8	=IF(B8=1,1,IF(C8=1,1,IF(D8=1,1,IF(E8=1,1,IF(F8=1,1,IF(G8=1,1,0))))))
9	=IF(B9=1,1,IF(C9=1,1,IF(D9=1,1,IF(E9=1,1,IF(F9=1,1,IF(G9=1,1,0))))))

Note: Rows 10 through 54 and Columns A through H omitted.

As a final example, consider the value 10. Two bits are on, the bit representing 8 and the bit representing 2. All other bits are off. If the value is decreased to 8, no relocation is needed; all that needs to be done is turn off the 2's bit. If the value is to be decreased to 6, the 8's bit must be turned off and the 4's bit must be turned on; the page must be either 2 or 0, this result may be achieved by only turning off bits; no relocation is required. In the case of a 10 as the initial value, pages will have to be relocated in 2 of the 5 decreases in value, or 40 percent of the cases. Table 4 shows a value view of the evaluation of the value 10.

**Table 4. Values view to evaluate the value 10**

	A	B	C	D	E	F	G	H	I
1	128	64	32	16	8	4	2	Value	Relocate
2	0	0	0	0	1	0	1	10	
3	0	0	0	0	1	0	0	8	0
4	0	0	0	0	0	1	1	6	1
5	0	0	0	0	0	1	0	4	1
6	0	0	0	0	0	0	1	2	0
7	0	0	0	0	0	0	0	0	0
8									
9								P(R DEE)	.40

Taking all possible combinations of odd inputs (127 odd numbers because the case of 0x01 cannot be decreased to an odd number) and calculating all possible reductions, there are bit changes that require a relocation in 74.65 percent of the cases. The conditional probability of a relocation, given a decrease event, an odd value input, and an odd result, is  $P(R|DOO) = .7465$ .

When the decrease from an odd number produces an even result, the low-order bit is turned off. This may be the only change, a decrease by one. In calculating the possible changes, the decreases by one are cases that do not require a relocation. As a result, the probability of a relocation drops slightly to 73.5 percent. The conditional probability of a relocation, given a decrease event, an odd value input, and an even result, is  $P(R|DOE) = .7350$ .

When the decrease is from an even number and produces an odd result, the low-order bit must be turned on in all cases. Therefore, the page must be relocated, and the probability is 100 percent. Changes in the higher-order bits do not alter the probabilities. The conditional probability of a relocation, given a decrease event, an even value input, and an odd result, is  $P(R|DEO) = 1.0$ .

Lastly, when the decrease is from an even number and produces an even result or a zero, the low-order bit is not changed. There are 127 even values, and as above, the bit changes require a relocation in 74.65 percent of the cases. The conditional probability of a relocation, given a decrease event, an even input, and an even result, is  $P(R|DEE) = .7465$ .

These conditional probabilities will be used to calculate the total probability of a relocation.

**TOTAL PROBABILITY OF RELOCATION**

The impact of these probabilities can only be understood in context. Returning to Figure 2 for the prior probability of each of the situations and multiplying the conditional probability by the prior probability of the situation gives joint probability of the situation relocation. The sum of the joint probabilities gives the probability of a relocation on any update. This information is summarized in Table 5.

**Table 5. Total Probability of Relocation.**

Situation	Prior Probabilities P(Situation)	Conditional Probabilities P(R Situation)	Joint Probabilities P(Situation ∩ R)
Increase	.5	1.00	.50
Decrease,E,E	.125	.7465	.0933
Decrease,E,O	.125	1.00	.125
Decrease,O,E	.125	.7350	.0919
Decrease,O,O	.125	.7465	.0933
Totals	1.00		P(R) = .9035

Summing the partial probabilities across all of the update conditions produces a total probability of relocation of 90.35 percent.

## **MULTIPLE TRANSACTIONS**

To consider the situation of multiple independent updates to the same record, the first step is to compute the probability of that record not being relocated after one update. This probability,  $P(NR)$ , is  $1 - P(R)$  or .0965. The probability of a record not being relocated after two independent updates is  $P(NR)$  times  $P(NR)$  or .0093. The probability of a relocation after two independent updates is .9907. Following the same process, the probability of a relocation after three independent updates is .9999. The chance of a relocation not being required is slightly less than one chance in ten thousand.

The previous analysis employed the probability for a single byte. The assumed context for that byte was one record. That is the logical context. The physical context is the page. Therefore, the probability developed above applies to the entire page. That is, a page that is updated once has a probability of requiring relocation of 90.35 percent; a page that is updated twice has a probability of requiring relocation of 99.07 percent; and a page that is updated thrice has a probability of requiring relocation of 99.99 percent.

## **DISCUSSION**

The argument that must be made here is that transactions that, by definition, cause increases in one or more fields in a record should be batched together and posted to the same page in some type of co-sequential process. This approach will minimize the need to relocate memory pages and thus minimize wear on the solid state drive. Because decreases result in a substantial proportion of relocations, the same argument should also be applied. The results here argue against any kind of real-time transaction processing, but that argument is not realistic in today's computing. More importantly, these results emphasize a need for more creative file organizations that minimize the wear on solid state drives.

One file organization that might offer a solution to the wear problem would be to place the static data on a solid state drive and the dynamic data on a traditional hard disk drive where the update process will not generate additional wear. This approach could also extend to indexed file organizations. Indexes that undergo frequent changes would be candidates for placement on magnetic media as would dynamic data while static data would be placed on solid state media. Of course, the determination of frequency of changes, which has not been much of a consideration up to this time, must become an important part of the analysis phase in the system design process.

In addition, the results argue for large buffer pools and large caches that will increase the likelihood that more than one change will be posted to a memory page before it is returned to the drive. Of course, the buffer-pool issue can be addressed when the system is configured. These configuration parameters should be addressed at the system design phase. Large caches are also a system configuration issue, but it requires a hardware solution. This also should be addressed at the system design phase. And in the case of using an SSD to replace an HDD, the cost of retrofitting servers and clients with additional cache memory and the cost of reconfiguring the buffer pools on all clients must be included in the total cost of the replacement.

Many of the schemes that have been invented to reduce seeking are no longer relevant.

Many practices that have been put into place to make retrieval time shorter will only accelerate the wear on a solid state drive. Consider the defrag utility. It exists to consolidate a file to the same track, or if large, then the same cylinder. This avoids seek time, the largest component of access time. With a solid state drive, it will result in parts of the file being relocated and rewritten (more wear for no gain). File systems should be configured to block attempts to defrag a solid state drive.

Consider the practice of some applications, particularly word processors, but all of the applications in Microsoft Office, to do an automatic file save at intervals. While well intentioned, frequent saves of a file that is being edited will most likely lead to pages being relocated and to added wear on the solid state device. Such a result can be avoided by transferring the file that is to be edited to conventional media until editing is completed.

## CONCLUSION

Drive wear results from relocating storage pages on the drive. During random updates, the total probability of relocation ( $P(R) = .9035$ ) is so great that attention must be given to methods to minimize relocations. File design can minimize some of the wear, but other approaches must be considered as well. Large caches and large buffer pools will help minimize wear, and some system-level practices must be reconsidered in light of the effect of those practices on drive wear.

In all, drive wear presents a whole new set of considerations for the system designer, and some of those considerations do not easily coordinate with practices of the past that focus on minimizing response time.

## AUTHOR INFORMATION

**Dwight A. Haworth** received his B.S. degree from the United States Air Force Academy, CO, in 1963. He retired from the United States Air Force in 1981. He received his Ph.D. in Management Information Systems from Texas Tech University, Lubbock, TX, in 1990. His research interests are information assurance and systems development and performance.

## REFERENCES

1. Chicoine, P., Hassner, M., Noblitt, M., Silvus, G., Weber, B., & Grochowshi, E. (2007). Hard disk drive long data sector white paper. Milpitas, CA: IDEMA. Retrieved June 10, 2010, from [http://www.idema.org/smartsite/modules/local/data\\_file/show\\_file.php?cmd=download&data\\_file\\_id=1779](http://www.idema.org/smartsite/modules/local/data_file/show_file.php?cmd=download&data_file_id=1779).
2. Intel Corporation (2009). Intel® X25-M and X18-M Mainstream SATA Solid-State Drives. Retrieved June 10, 2010, from <http://download.intel.com/design/flash/nand/mainstream/322208.pdf>.
3. Mearian, L. (2009, June 18). Review: Hard disk vs. solid-state drive -- is an SSD worth the money? Computerworld. Retrieved June 2, 2010, from [http://www.computerworld.com/s/article/9134468/Review\\_Hard\\_disk\\_vs.\\_solid\\_state\\_drive\\_is\\_an\\_SSD\\_worth\\_the\\_money](http://www.computerworld.com/s/article/9134468/Review_Hard_disk_vs._solid_state_drive_is_an_SSD_worth_the_money).
4. Ruth, G. (2008). *Solid-state disks: Coming to a system near you*. Midvale, UT: Burton Group.
5. SanDisk Corporation (2010). SanDisk® G3 Solid State Drive. Retrieved June 10, 2010, from <http://www.sandisk.com/products/computing-products/sandisk-g3-solid-state-drive>.
6. Seagate Technology (2010). Seagate® Pulsar™ SATA 3Gb/s 200GB Solid State Drive. Retrieved June 10, 2010, from <http://www.seagate.com/www/en-us/products/enterprise-ssd-hdd/pulsar/#tTabContentSpecifications>.
7. Wikipedia (2010, June 15). Flash memory. Retrieved June 15, 2010, from [http://en.wikipedia.org/wiki/Flash\\_memory](http://en.wikipedia.org/wiki/Flash_memory).

NOTES