2010 Fifth IEEE International Conference on Networking, Architecture, and Storage

A Fine-grained Data Reconstruction Algorithm for Solid-state Disks

Peng Wang, Diqing Hu, Changsheng Xie, Jianzong Wang
Wuhan National Laboratory for Optoelectronics
Key Laboratory of Data Storage System, Ministry of Education
School of Computer Science and Technology
Huazhong University of Science and Technology, Hubei Wuhan P. R. China
wang.peng@smail.hust.edu.cn

Xiao Qin
Department of Computer Science and Software Engineering
Samuel Ginn College of Engineering
Auburn University
http://www.eng.auburn.edu/~xqin
xqin@auburn.edu

Abstract

Solid-state disks (SSDs) with high I/O performance are increasingly becoming popular. To extend the life time of flash memory, one can apply wear-leveling strategies to manage data blocks. However, wear-leveling strategies certainly inevitably degrade write performance. In addition to low write performance, wear-leveling strategies make one block unwritable when one bit of this block is invalid. Although data reconstruction techniques have been widely employed in disk arrays, the reconstruction techniques has not been studied in the context of solid-state disks. In this paper, we present a new fine-grained data-reconstruction algorithm for solid-state disks. The algorithm aims to provide a simple yet efficient wear-leveling strategy that improves both I/O performance and reliability of solid-state disks. Simulation experiments show that all data blocks have very similar in terms of erasure times. The number of extra erasures incurred by our algorithm is very marginal.

Keywords: *data reconstruction; area wear-leveling strategies, solid-state disc; disdegrading mechanism, high reliability.*

1. Introduction

Flash memory - a common storage medium – can be used as a non-volatile, shock-proof, energy-saving storage device. Typically, a flash memory is composed of a number of flash memory blocks, each of which is divided into a number of physical page. One block is the smallest unit of erase operations; one page is a unit for reading and writing operations. Prior to rewriting a physical page, one must have the entire block erased. One major drawback of flash memory is that the number of erase times is as low as anywhere from 100,000 to 1,000,000 [1][2]. If there is a single block

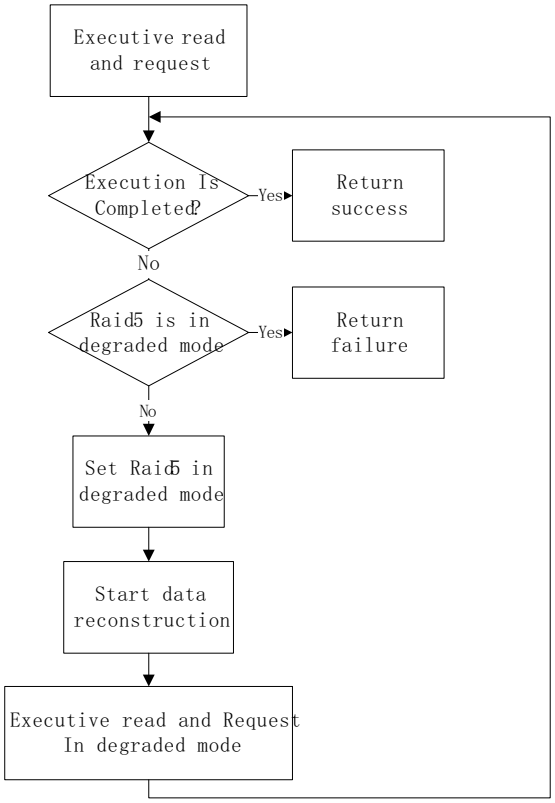

Figure 1. Data reconstruction for RAID-5.

978-0-7695-4134-1/10 $26.00 © 2010 IEEE
DOI 10.1109/NAS.2010.62

reaching its erase time limit, data stored in this block becomes unreliable.

Redundant array of independent disks or RAIDs have low cost, high I/O performance, and high reliability. As such, RAID has been widely applied to support data-intensive computing applications. In this study, we focus on data reconstruction processes in disk arrays of level 5 or RAID-5. A data reconstruction process is triggered if one disk fails. During the process of data reconstruction, the disk array is transitioned into a degraded mode. Fig. 1 depicts the data reconstruction process of RAID-5 [3].

RAID-5 makes no distinction among failure accesses when a disk array is in the degraded mode. In this paper, we pay attention to a way of integrating data reconstruction mechanism into solid-state disks by designing a fine-grained data reconstruction algorithm. Experimental results show that the life time of a physical block can be significantly extended (see, for example, [4]). Our simulation results demonstrate that our new data reconstruction algorithm is conducive to improve the reliability of flash-based solid-state disks.

2. Fine-Grained Data Reconstruction

NAND flash reads and writes in units of pages (256 or 512 Bytes per page); data in flash memory is erased in units of block (4KB, 8KB, or 16KB per block). Since bit-error rates of NAND flash chips are relatively high, ECC checks on memory are highly recommended. As a result, it is desirable for flash-based solid-state disks to reserves redundant blocks to save ECC check bits, the size of which is approximately 1% of that of the data bits. After a data error occurs in a block, the failed block will be marked followed by replacing the redundant block with a new block. Note that replacements are handled by disk controllers. It is common that small NAND flash chips contain 32 pages, each of which consists 4224 bits (i.e., 512 + 16 Bytes = 4,224 bits per page). Unlike small chips, large flash chip contains 64 pages of 16,896 bits (i.e., 2048 + 64 Bytes = 16,896 bits per page). When one bit of 16,896 bits is inaccessible, the entire block will be marked accordingly. The downside of doing this is the low usage of storage space.

To address the above low usage problem of flash-based solid-state disks, we design the following data reconstruction strategy to efficiently use disk space when errors occur. Let us consider a disk request that attempt to read or write a data block in a solid-state disk. The following steps are carried out to process the request.

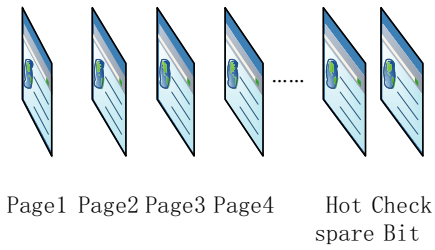

Figure 2. Structure of a block.

The fine-grained data reconstruction strategy:

a) In each block, we reserve one page as a hot spare page and one page as a check bit page. Figure 2 shows the structure of each block with a hot spare page and a check bit page.
b) Process a read or write access request.
c) Predict whether the access operation is successful. If the operation fails, go to step d); otherwise return the correct data or write the data.
d) Predict whether this block should be reconstructed. If data must be reconstructed, go to step f);
e) Predict whether the block should be mark as a bad block. If the block is a bad block, go to step g);
f) Copy the valid data of the failed page to the hot spare page; then reconstruct failure data using the valid pages as well as the check bit page. Next, reconstructed data are copied to the hot spare page. Note that reconstructing and copying are the two main steps of generating data for a failed block. In this step of data reconstruction, we make use of a virtual page in main memory or RAM to improve the performance of our data reconstruction mechanism. Using the access position, the reconstruction algorithm predicts whether the failure data is to be changed by this write operation. If the failure data will be modified, the write operation modifies data to the hot spare page. Otherwise, the reconstructed data should be written to the page. If a page is empty, the algorithm simply writes 1 to the page. This step is outlined in Figure 3 below.
g) If the number of total failure data bits is beyond the data reconstruction capability, the block will be marked bad block. Accordingly, the wear-leveling strategy chooses a new block to replace this bad block by copying all data to a new block and modifying the pointer to the new block.

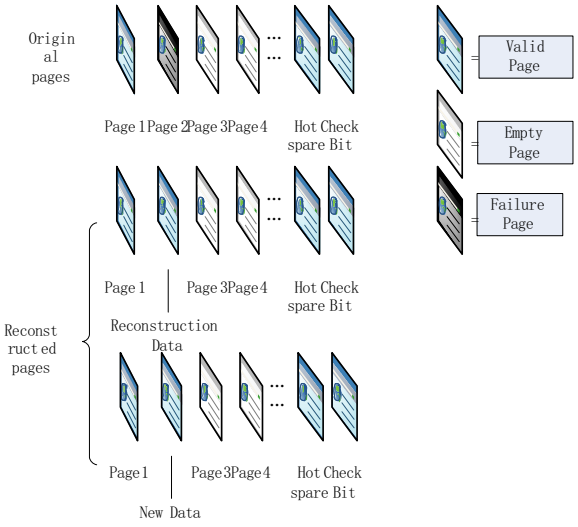

Figure 3. Reconstruction process (see step f in the data reconstruction algorithm)

In step c) where the algorithm predicts whether operations will succeed, we set a threshold to definite maximum number of failed bits in one page. The threshold value should be carefully decided upon the computational performance of solid-state disks. For example, if the computing capability is very high, the threshold value should be great, otherwise the threshold value should be small [5].

Figure 4 outlines the read and write processing procedure for the data reconstruction algorithm, which can alleviate the problem of low read or write performance due to failed blocks. Our scheme improves read/write performance because our algorithm only needs to reconstruct failed bits. Thus, other non-failed data can be directly read or write while failed bits are reconstructed. RAID-5 uses only XOR operation - an easy way to be implemented by FPGA or system-on-chip (SOC) to reconstruct data. Our approach improves the reliability of solid-state disks at the cost of marginal performance degradation, because a block is marked as a bad block if only two or more pages have failed bits. If one page is reserved in the first place, the data reconstruction algorithm can rely on the reserved page to replace a failed page. We argue that the fault tolerance of flash-based solid-state disk is expected to be improved by our data reconstruction algorithm, since only blocks with three or more bad pages are treated as inaccessible. Thus, applying our algorithm to solid-state disks can reduce the number of bad blocks. The details on the data construction procedure can be found in Figure 4.

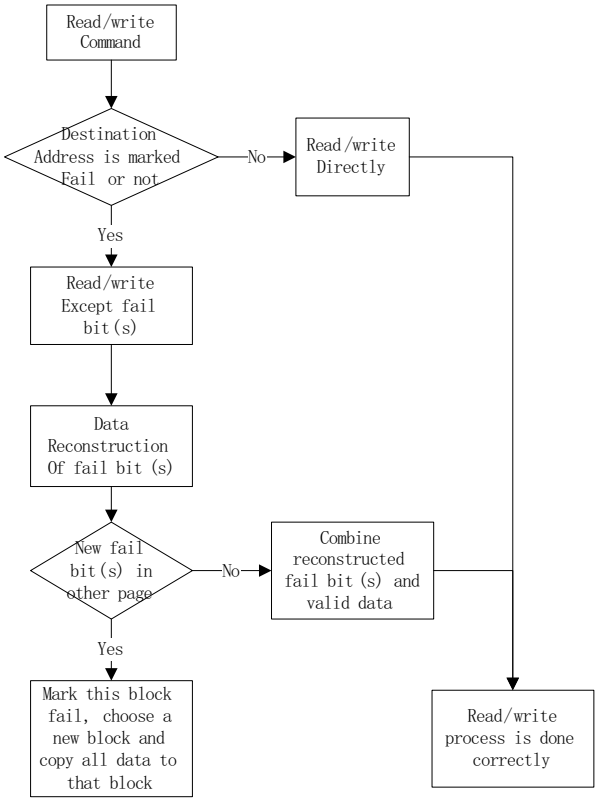

Figure 4. Read/Write process

Recall that in our fine-grained data reconstruction algorithm the basic storage unit is page. Without loss of generality, we assume that all pages in a block are identical in the sense that the pages have the same number of erasure times. To simplify the design of our reconstruction algorithm, we adopt and extend the conventional data reconstruction algorithm built for RAID-5.

$$\begin{cases} \text{Generate Check bit} & P = D0 \oplus D1 \oplus D \cdots \oplus Dn \\ \text{Reconstruct ed bit} & Di = D0 \oplus D1 \cdots \oplus Di-1 \oplus Di+1 \cdots \oplus Dn \oplus \text{Check num} \end{cases} \quad (1)$$

We use data check bits to detect and correct any error and; therefore, it is indispensable to maintain data consistency in our reconstruction algorithm. When data is written or updated in a page, the disk controller calculates check bits by performing a bit-wise XOR operation (see the upper expression in Equation 1). Similarly, using a bit-wise XOR operation leads to a perfect reconstruction of a failed bit.

Below we show the detailed implementation of the algorithm geared to the reconstruction of bad blocks.

```
static  int  NANDFlashReadPage(U32  block,U32
page,U8 *buffer){
       int i,j;
       unsigned int blockPage;
       U8 ecc0,ecc1,ecc2;
       U8 *bufPt=buffer;
       U8 se[16];
       page=page&0x1f;
       // each block has 32 pages
       blockPage = (block << 5) + page;
       initialize parameter
       for(i=0;i<10;i++);//wait
       NANDFlashWAITRB();//Wait tR
       for(i = 0; i < 512;i++){
        if (i == badbyte) {
          for (j = 0;j < page; j++)
          *bufPt++
XOR=NANDFlashReadPage(block,i,buffer)[i];
          for (j = page; j < 62; j++)
          // 64-1hot spare pages-1check bit
          // page
          *bufPt++ XOR=
         NANDFlashReadPage(block,i,buffer)[i];
        }
        // Read one page
        *bufPt++=NANDFlashRDDATA();
       }
       ecc0=rNFECC0;
       ecc1=rNFECC1;
       ecc2=rNFECC2;
       for(i=0;i<16;i++){
          //Read spare array
          se[i]=NANDFlashRDDATA();
       }
       NANDFlashnFCE_H();
       if(ecc0==se[0] && ecc1==se[1] &&
          ecc2==se[2]){
       Uart_Printf("[ECC
         OK:%x,%x,%x]\n",se[0],se[1],se[2]);
       return 1;
       }
       else {
       Uart_Printf("[ECC
ERROR(RD):read:%x,%x,%x, reg:%x,%x,%x]\n",
       se[0],se[1],se[2],ecc0,ecc1,ecc2);
       return 0;
       }
  }
```

In order to improve the error correction capability, one may use either the run-length limited code (RLL) or the low density parity check code (LDPC). RLL coding is a high efficiency coding; LDPC coding has good performance. LDPC coding not only is close to the theoretical Shannon limit, but also has flexible structure. Thus, the decoding complexity of LDPC is fairly low.

3. A Simplified Wear-Leveling Strategy

A compelling feature of our fine-grained data reconstruction algorithm is that there is no pressing need of wear leveling, because in our strategy blocks identified as bad blocks are available for access. The fine-grained reconstruction algorithm can be used in conjunction with a simplified rather than a complicated wear-leveling strategy. Therefore, in this section we present a simple yet powerful wear-leveling strategy (see Step g in Section 2) that takes advantage of relative rank difference in a zone.

Most existing wear-leveling strategies make solid-state disks keep number of write for all blocks. If a solid-state disk has eight pieces of NAND flash chips each of which has 16K blocks, then every block has to use 17 bits to keep track of the number of writes. Therefore, the solid-state disk has to reserve 272 KBytes (ie., 17*16k*8 = 272 KBytes) to record a list of numbers of writes. Such a large list, of course, not only has a negative impact on the searching speed of the solid-state disk, but also wastes the storage capacity of the solid-state disk. Apart from the above drawbacks, the existing wear-leveling strategies compare relative numbers that count writes. Thus, the D-values of two numbers are more useful than the two numbers themselves.

In our simplified wear-leveling scheme, we define relative rank value Δ as the number of erasures between two blocks. We define the erasure number of the first block in a zone as the base value of this zone. Let us use an example to explain a way of setting relative rank values. If one block has a large number of erasures than the first block, then the relative rank value is set to 2 (i.e., $\Delta = 2$). If one block and the first block have the same number of erasures, then the relative rank value is set to 1 (i.e., $\Delta = 1$). Otherwise, the relative rank value is 0 (i.e., $\Delta = 0$). Besides relative rank values, the total erasure number of each zone is counted by our simplified wear-leveling strategy. Recall that our fine-grained data reconstruction algorithm triggers the wear-leveling strategy (see Step g in Section 2). We design a the following data reconstruction strategy that .

The data reconstruction strategy:

a) Save the erasure number of every block in a zone;
b) Randomly select a zone as a target zone;
c) If the total erasure numbers of the target zone is larger than the current block, the target zone is used more, go to step b);
d) If the corresponding block in the target zone $\Delta = 2$, go to step b);
e) Select the corresponding block in the target zone as the target block.

Although relative rank values only represent the theoretical approximations of true values, the mean error of relative rank values in the same zone are

usually very small. As such, we believe that the approximations are accurate in the majority of blocks.

Relative rank values can be used for dynamic wear leveling as long as an operating system scans each zone in a solid-state disk on a regular basis. When the total erasure number increases sharply, the simplified wear-leveling strategy chooses a less used zone to swap.

To demonstrate the effectiveness of our data reconstruction and wear-leveling strategies, we decided to leverage the simulated annealing algorithm. Previous studies show that molecularities satisfy the Boltzmann probability distribution at temperature *T*. Thus we have:

$$P(\bar{E} = E(r)) = \frac{1}{Z(T)} e^{-\frac{E(r)}{k_B T}}. \quad (2)$$

where *E(r)* represents state *r*'s energy, $k_B > 0$ denotes a Boltzmann constant, $\bar{E}$ is a random variable for a molecular, *Z(t)* represents a normalization factor of the probability distribution. If one chooses two parameters E_1 and E_2 (e.g., $E_1 < E_2$), the following Equations can be derived from the above Equation 2:

$$P(\bar{E} = E_1) - P(\bar{E} = E_2) = \frac{1}{Z(T)} e^{-\frac{E_1}{k_B T}} [1 - e^{-\frac{E_2 - E_1}{k_B T}}] \quad (3)$$

$$\because e^{-\frac{E_2 - E_1}{k_B T}} < 1, \forall T > 0 \quad (4)$$

$$\therefore P(\bar{E} = E_1) - P(\bar{E} = E_2) > 0, \forall T > 0 \quad (5)$$

$$\frac{\partial P(\bar{E} = E(r))}{\partial T} = \frac{e^{-\frac{E(r)}{k_B T}}}{Z(T) k_B T} [E(r) - \frac{\sum_{s \in D} E(s) e^{-\frac{E(s)}{k_B T}}}{Z(T)}] \quad (6)$$

In this study, we are particularly concerned with the lowest energy state. Let r_{min} be the lowest energy state, we can simplify the above Equations in the case where the lowest energy state is r_{min} . Thus, the following Equations can be derived from Equation 6 as:

$$\frac{\partial P(\bar{E} = E(r))}{\partial T} < 0 \quad (8)$$

$$\therefore P(\bar{E} = E_{r_{min}}) = \frac{1}{Z(T)} e^{-\frac{E(r_{min})}{k_B T}} = \frac{1}{|D_0| + R} \quad (9)$$

$$R = \sum_{s \in D, E(s) > E(r_{min})} e^{-\frac{E(r) - E(r_{min})}{k_B T}} \to 0, T \to 0 \quad (10)$$

To evaluate our data reconstruction and wear-leveling strategies, we simply consider a case where the temperature *T* approaches to zero. Using Equations 8 – 10, the Boltzmann probability distribution (see Equation 2) can be rewritten as below:

$$P(\bar{E} = E(r_{min})) \to \frac{1}{\|D_0\|}, T \to 0. \quad (11)$$

4. Reliability Analysis

Three typical metrics used to measure the reliability of a disk system include mean time to failure or MMTF, mean time between failures or MTBF, and mean time to data loss or MTTDL. Patterson *et al.*, developed a general framework for evaluating the mean time to failure of disk arrays (see [6] for the details of the evaluation framework). It is assumed that disk failures - following an exponential distribution - are independent of each other. Let *t* be time, *M* be the average life expectancy of disk systems. The reliability $R_{exp}(t)$ of disk systems can be expressed as a function of time t and average life expectancy *M* [5]. Thus, reliability $R_{exp}(t)$ is calculated as below.

$$R_{exp}(t) = e^{\frac{t}{M}} \quad (12)$$

If time *t* is much less than life expectancy *M*, then reliability $R_{exp}(t)$ can be approximated as Equation 13.

$$R_{exp}(t) = e^{\frac{t}{M}} \approx 1 - \frac{t}{M} \text{ (M= average life).} \quad (13)$$

Disk failure rate – the probability that a disk fails before time t or Prob(malfunction before time *t*) – is derived from reliability $R_{exp}(t)$ expressed by Equation 13. Thus, Prob(malfunction before time *t*) can be written as:

Prob(malfunction before time *t*) = 1 - $R_{exp}(t)$ $\approx$ *t/M.*

Let λ be the failure rate of a solid-state disk system, and μ be the repair rate of the disk system. Failure rate λ is inversely proportional to mean time to failure or $MTTF_{disk}$ (i.e., $\lambda = 1/MTTF_{disk}$). Similarly, repair rate μ is inversely proportional to mean time to repair or $MTTR_{disk}$ (i.e., $\mu = 1/MTTR_{disk}$).

The reliability of a single data striping group of the RAID-5 model can be expressed as a Markov chain shown in Figure 5, in which state 0 denotes normal conditions, state 1 represents that one page is breakdown, DL represents the data loss state, and N +

1 denotes the number of disks. The probability that the system transits from state 0 to 1 is the product of $N + 1$ and failure rate λ. The probability of transitions from state 1 to state 0 equals to repair rate μ.

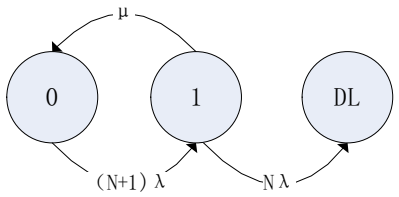

Figure 5. Markov chain of RAID-5.

Let W_i be the probability that the ith disk fails and ζ_i be the failure rate of the ith disk. Then the probability ζ that one page is down (see state 1 in Fig. 5) is the weighted sum of ζ_i ($1 \le i \le N$). Thus, we have

$$\zeta = \sum_{i=1}^{N}(W_i \times \zeta_i) = \sum_{i=1}^{N}\left(\frac{T_i}{T} \times \zeta_i\right) \quad (14)$$

$$R = \begin{pmatrix} R_{11} \\ R_{21} \\ \vdots \\ R_{n1} \end{pmatrix} = \begin{pmatrix} R_{111} & R_{121} \cdots & R_{1m1} \\ \vdots & & \vdots \\ R_{n11} & R_{n21} \cdots & R_{1m1} \end{pmatrix} \quad (15)$$

In the case where failure rate λ is far less than repair rate μ, the mean time to data loss value or MTTDL for RAID-5 can be expressed as Equation 16 (see [6][7] for details on mean time to data loss).

$$MTTDL = \frac{(2N+1)\lambda + u}{N(N+1)\lambda} \approx \frac{u}{N(N+1)\lambda} = \frac{MTTF^2}{N(N+1)MTTR} \quad (16)$$

5. Experimental Results

In this Section, we demonstrate that our new data reconstruction strategy along with the simplified wearing-leveling technique can extend the MTTF value of solid-state disks by anywhere from 30% to 50%.

To evaluate the reliability of a solid-state disk system equipped with our data reconstruction and wearing-leveling techniques, we calculated the mean time to failure values by applying both the model (see Section 4) and a disk simulator using C++ on a Linux platform.

In what follows, we first rely on the reliability model to quantify MTTF values. Using the model described in Section 4, Figure 5 shows the mean time to failure (MTTF) values of a traditional data construction strategy and our approach. The reliability model indicates that under random access workload conditions, both the traditional approach and our fine-grained data reconstruction algorithm are very sensitive to workload conditions. More specifically, regardless of data construction algorithms, increasing workload conditions reduces mean time to failure values. The results suggest that the reliability of solid-state disk systems is lowered by high workloads. Furthermore, Figure 5 shows that the MTTF values of the traditional strategy are smaller than those of our data construction algorithm, indicating that our approach significantly improves the reliability of solid-state disk systems.

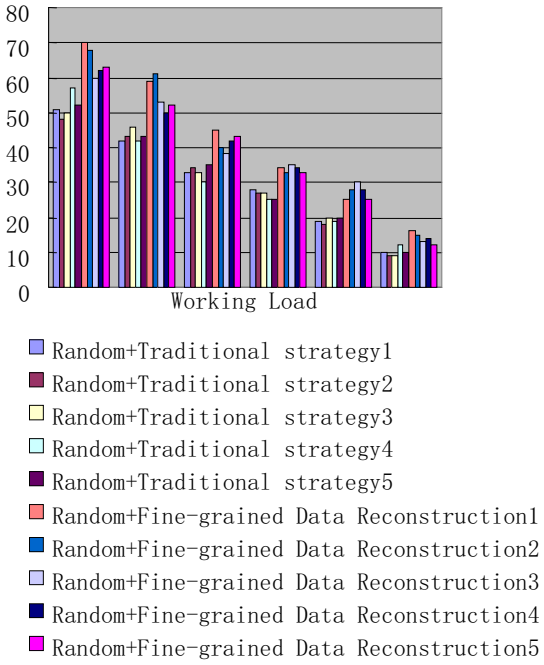

Figure 5. Mean time to failure values are derived from the reliability model presented in Section 4. Random I/O accesses are considered.

To valid the reliability model used to calculate the mean time to failure values plotted in Figure 5, we conducted extensive simulation studies to quantitatively evaluate the performance of our new techniques designed for solid-state disks under a wide variety of workload conditions. We augmented the DiskSim storage system simulator [8] with our data reconstruction mechanism. We used random I/O access patterns and the simulated annealing algorithm to evaluate the effectiveness of our proposed data reconstruction algorithm.

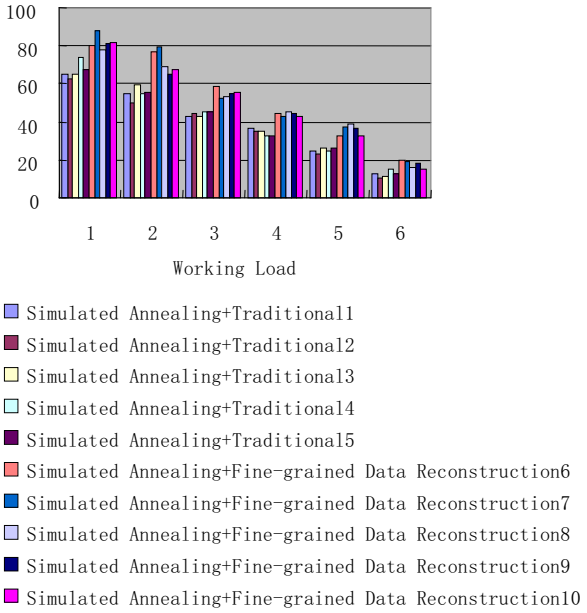

Figure 6. The simulated Annealing algorithm used to valid the reliability model described in Section 4.

In the simulated annealing algorithm, we consider the following simulated annealing parameters: initial temperature *t*, the temperature change rate *a*, the length of a Markov chain that is set to 1, and the maximum number of iterations *n*.

Figure 6 shows the simulation results for both the traditional and our data reconstruction algorithm. The simulation results plotted in Figure 6 are consistent with the modeling results depicted in Figure 5. Thus, the reliability model described in Section 4 is experimentally validated.

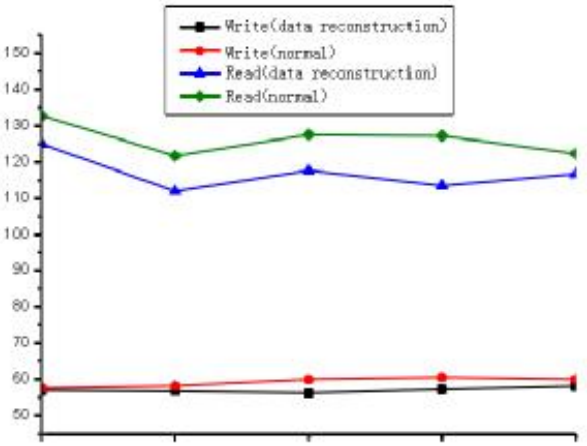

Figure 7. Read/write rate of a solid-state disk equipped with the fine-grained data reconstruction scheme.

After investigating the reliability of a solid-state disk equipped with our data reconstruction scheme, we focus on I/O performance of the solid-state disk. Figure 7 shows the read and write rates of the solid-state disk. The performance results illustrated in Figure 7 indicate that our fine-grained data reconstruction algorithm introduces approximately 5% to 10% overhead in terms of the read and write performance. These results confirm that our approach significantly improves the reliability of solid-state disks with marginal performance overhead.

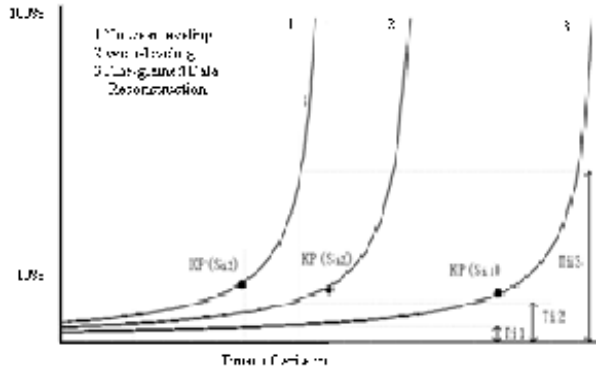

Figure 8 Impacts of the number of writes on the percentage of bad blocks.

A certain percentage of bad blocks can be used as a preliminary metric to evaluate the reliability of solid-state disks. Now we investigate impacts of the number of writes on the percentage of bad blocks. Figure 8 shows bad block percentages of our strategy and the two alternative approaches as functions of the time of writes. Figure 8 clearly demonstrates that compared a solid-state disk without employing wear-leveling, the wear-leveling technique improves the reliability of solid-state disks by significantly reducing the percentage of bad blocks. This observation is consistent with the results reported in [11][12]. More importantly, the results potted in Figure 8 show that the reliability of the solid-state disk can be further improved if our fine-grained data reconstruction algorithm is integrated with the wear-leveling technique.

High reliability, of course, comes at the cost of performance. Figure 9 shows the impact of I/O load measured in terms of I/O operations per second on response time. The results indicates that the fine-grained data reconstruction algorithm increased the average response time by approximately 10%. These results are consistent with those reported in Figure 7.

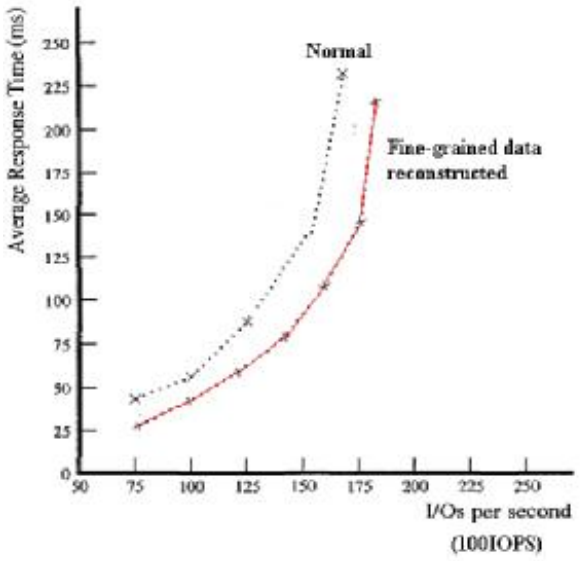

Figure 9. Impact of I/O load on average response time measured in millisecond.

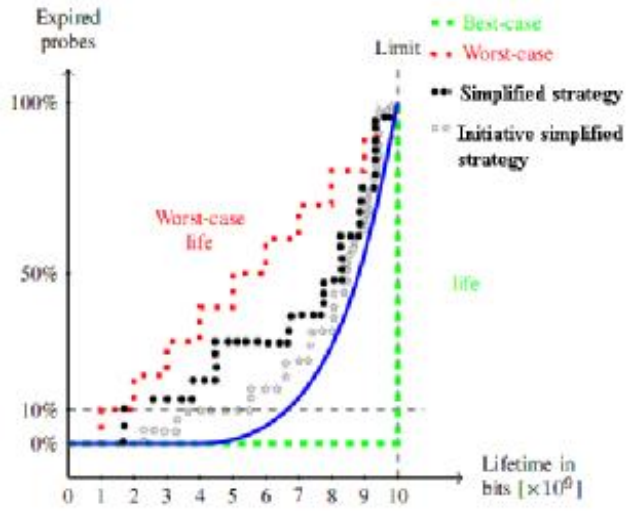

Figure 10. Expired probe

Figure 10 shows the correlations between lifetimes and expired probes in the context of solid-state disks. Details on expired probes can be found in [13]. We observe from Figure 10 that our simplified strategy described in Section 3 can substantially extend lifetimes of solid-state disks.

5. Conclusions

Recognizing that traditional coarse-grained data reconstruction approaches are inadequate for solid-state disk, we developed in this study a new find-grained data reconstruction. A compelling feature of our fine-grained data reconstruction algorithm is that there is no pressing need of wear leveling, because in our strategy blocks identified as bad blocks are available for access. We designed a simplified rather than a complicated wear-leveling strategy. The simplified wear-leveling scheme was integrated with the fine-grained data reconstruction algorithm. Experimental results show that our fine-grained data-reconstruction algorithm coupled with the a simple yet efficient wear-leveling strategy can improve both reliability and I/O performance of solid-state disks. For example, all data blocks have very similar in terms of erasure times. The number of extra erasures incurred by our algorithm is very marginal.

As a future research direction, we will integrate coding techniques other than ECC into our data reconstruction scheme. Future coding techniques to be addressed include RLL, Turbo or even LDPC coding. We firmly believe that these coding techniques can further improve reliability of solid-state disks.

Acknowledgment

The work reported in this paper was supported by the National Natural Science Foundation of China (Grant No. 60933002) and the Natural 863 Plan under the Grant No. 2009AA01A402. Xiao Qin's work was supported by the U.S. National Science Foundation under Grants CCF-0845257 (CAREER), CNS-0917137 (CSR), CNS-0757778 (CSR), CCF-0742187 (CPA), CNS-0831502 (CyberTrust), CNS-0855251 (CRI), OCI-0753305 (CI-TEAM), DUE-0837341 (CCLI), and DUE-0830831 (SFS).

References

[1] Esther Spanjer, .Flash management: How it extends the lifetime of solid-state Flash disks in VMEbus system, VMEbus System,2005.8

[2] Chang L P, Kuo T W. Adaptive striping architecture for flash memory storage systems of embedded systems [C]. In: Proceedings of the IEEE Real-time and Embedded Technology and Applications Symposium, IEEE Computer Society, 2002, 1872 196.

[3] D. Patterson, G. Gibson and R. Katz. A Case for Redundant arrays of Inexpensive Disks (RAID).In: Proc. Of the ACM SIGMOD'88.1988.109~116

[4] P.M. Chen，E.K. Lee，G.A. Gibson et al. RAID: High-Performance ， Reliable Secondary Storage.ACM Computing Surveys, 1994, Vol.26(2):145~185

[5] Gang Wang: Architecture and Data Layout of Network Based Disk Arrays, 2004

[6] G. A. Gibson and D. A. Patterson, "Designing disk arrays for high data reliability," Journal for Parallel and Distributed Computing, vol. 17, pp. 4-27, January 1993.

[7] B. Schroeder and G. A. Gibson, “A large-scale study of failures in high-performance computing systems,” Proc. 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2006), Philadelphia, PA, pp. 249–258 , June 2006.

[8] B. Schroeder and G. A. Gibson, “Disk failures in the real world: What does an MTTF of 1,000,000 hours mean to you?” Proc. 5th USENIX Conference on File and Storage Technologies (FAST '07), San Jose, CA, pp. 1–16, Feb. 2007.

[9] Nitin Agrawal, Vijayan Prabhakaran, Ted Wobber, John D. Davis, Mark Manasse, Rina Panigrahy: Design Tradeoffs for SSD Performance. Usenix Annual Technical Conference (USENIX '08), June 2008, Boston, MA.

[10] Nitin Agrawal, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. Generating Realistic Impressions for File-System Benchmarking. Proceedings of the 7th Conference on File and Storage Technologies (FAST '09), Feb 2009, San Francisco, CA.

[11] Chang Yuan-hao, Hsieh Jen-wei , Kuo Tei-wei. Endurance enhancement of flash-memory storage systems: an efficient static wear leveling design. In Proceedings of the 44th Annual Conference on Design Automation. ACM Press, 2007, 2122-217.

[12] Chang Li-pin. On efficient wearleveling for largescale flashmemory storage systems. Proceedings of the 2007 ACM Symposium on Applied Computing. ACM Press, 2007, 11262 1130.

[13] Rosenblum M, O usterhout K. The design and imp lementation of a log-structured file system. Computer Systems, 1992, 10(1) : 26-252.