Fault Tolerant Algorithm for NVM to Reuse the Error Blocks

Li Zhu, Jinyu Zhan, Sihao Chen,
Yiming Zhang, Junhuan Yang, Wei Jiang
School of Information and Software Engineering
University of Electronic Science and Technology of China
Chengdu, China
Email: homedout@163.com; zhanjy@uestc.edu.cn;
csh950525@gmail.com; zhangluforever@hotmail.com;
yjhherny@163.com;weijiang@uestc.edu.cn

Lin Li
School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, China
Email: lilin@uestc.edu.cn

***Abstract*—Abstract—Non-Volatile Memory (NVM) has many advantages such as storage density, non-volatility, and lower energy consumption. However, they are constrained by limited write endurance and lack of an efficient mechanism to recovery the error data, which will affect the lifetime of NVM. In this paper, we propose a fault tolerant algorithm for NVM, that inspired by the redundant technology of RAID(Redundant Arrays of Independent Disks) to recovery the error data and reuse the error blocks. Memory architecture models and the algorithms of read, write and recovery are presented. Our experimental results show that more than 50% of error data can be recovered efficiently if the error rate is under 30%, and the recovery rate is nearly more than 80% if the error rate is under 10%, while the time consumption rate of our method is only within 5%.**

***Index Terms*—Keywords—NVM; redundant memory; RAID; recovery; fault tolerant**

I. Introduction

Recently, non-volatile storage(NVM) is widely used as a kind of computer memory that can retrieve stored information even after having been power cycled. The storage density of NVM is much better than many other types of memories, and the scale of it is very small so that it can be high-efficiency. However, once there exists some broken data or other faults[1], it will be discarded as unusable part by the manager, so it is a waste of storage space, and caused by its limit of write endurance, the lifetime of NVM will be reduced[2]. For example, the NAND flash memory in modern solid- state drives(SSDs) exits errors frequently in some positions of data blocks, the common practice is to permanently retire and discard the broken blocks, so the data needs to be written in new blocks or it will be lost, and the positions exist broken blocks are unreliable. So it is a challenge to recovery the broken blocks by putting them back to use and preserving the dependability and performance of the storage device.

The method used to recovery the data in NVM has been researched before. Replicated Memory (DRM)[3] is the first technique to rejuvenate pages that have faults and put them back to use for data storage. DRM picks pairs of faulty pages that do not have faults in the same bit position and stores replicated data in both pages. This redundant storage helps to recovery the original data through reading the non-faulty byte from at least one of the paired pages. Another proposal is Error Correcting Pointers[4], handles errors by encoding the locations of failed cells in a table and by assigning new cells to replace them. In another research[5], the author proposed rPRAM (redundancy PRAM), that explores extending PRAM device lifetime based on advanced redundancy techniques, assuming that the PCM-based main memory starts without any faults and does wear-leveling[6]to uniformly distribute the writes. When the first bit fault occurs, they temporarily decommission the PCM page k and place it in a separate pool of faulty PCM pages that are waiting to be matched with other compatible pages. Then finding a compatible group of g faulty PCM pages, and storing the corresponding parity to recovery the faulty data block. And some researches about bad block management, in NAND flash memory, repeated program/erase cycles damage the nitride layer of a cell, causing charges to be trapped in the dielectric. Accordingly, NAND flash as well as PCM cells exhibit a stuck-at fault model[7], [8]. In another research, s data dependent sparing is proposed, a new physical block sparing scheme that delays the retirement of a faulty block when the memory exhibits the stuck-at fault model.

In this paper, we propose a redundant method that inspired by RAID technology. Our mechanism is specifically based on RAID5 (block level striping with dedicated parity, where there is a dedicated parity block for every group). Accordingly, a specific and appropriate model should be established in NVM, so that the data allocation in NVM can be RAID-liked and the NVM will have the ability of redundancy. In our method, recovery operation will be done between data blocks and parity blocks when some errors exit in NVM, so the broken blocks will be reuse to a certain extent and the mechanism will be redundant. In our method, over 50% of the error blocks can be recovered efficiently when the error rate is under 30%, and over 80% of the error blocks can be recovered when the error rate is under 10%, while the time consumption is very low in these cases.

The rest of the paper is organized as follows. Section II introduces the architecture of the redundant model in NVM.

978-1-5090-3727-8/16 $31.00 © 2016 IEEE
DOI 10.1109/ICESS.2016.40

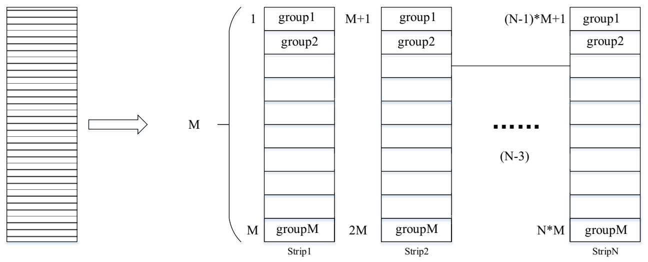

Fig. 1. The memory is divided into N strips

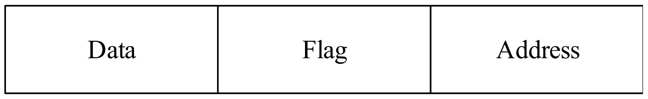

Fig. 2. Three segments in one block

Section III presents our recovery algorithm by illustrating the process of write, read and recovery. Section IV introduces our experiment methods and reports the experimental results. Finally, in Section V, we conclude this paper and talk about the future work.

II. MEMORY ARCHITECTURE

In this section, we present the description of the model and the concrete details of the three main processes, including write, read and recovery. Then, Algorithms of our method will be illustrated and examples will be shown to explain our method.

The method we proposed is based on RAID, according to the mechanism of RAID5, at least 3 disks are needed to use as storage devices, the parity blocks are distributed to the disks in a regular way, and the data blocks and the parity block of one group can be used to recovery the faulty blocks. In our method, we make the memory of NVM disk-like so that it will be redundant and be able to recovery the errors exist in NVM.

To establish a consistent and efficient model, a part of memory needs to be used, and the physical address of it should be continuous so that it can be managed conveniently.

In our method, as figure 1 shows, the memory is divided into N strips of the same size. Similarly, every strip is also divided into n blocks on average. At this time, the memory has been transformed into N strips with n blocks for each one. Shown as figure 2, the content of each block contains three segments, data segment, address segment and the flag segment. The data segment is used to store the data, while the address segment is used to store the pointer to the next block, and the flag segment is used to store the flag information which is set to 1 in all the parity blocks, whereas there is 0 in all the data blocks, so it can be used to differentiate between data blocks and parity blocks.

In order to reuse the faulty data in a managed way data and parity value are main information to recovery the faulty blocks in our method. All the parity value should be placed

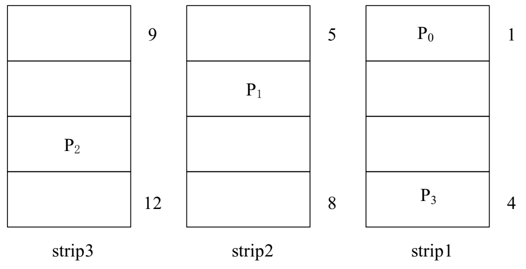

Fig. 3. Fig. 3.The location of parities

The reading of NVM Fault tolerant algorithm

```
begin
1 Write in NVM
2  count ← 0
3   for block are in BLOCKS do
4    if count ==0
5      if RecentNVM → misflag != 1
6       if RecentNVM → pariflag != 1
7     READ(value) from strip1, strip2 ... to strip n
8    end if
9   else excute recovery
10 end if
11     count ← 1
12      elif
13    if RecentNVM → misflag != 1
14   ifRecentNVM → pariflag != 1
15     READ(value) from stripn, strip(n-1) ... to strip1
16      end if
17    else excute recovery
18    end if
19       count ← 0
20    end if
21   end for
end
```

in several regular positions so that the loss of data will be reduced effectively in this case. Based on this, the positions of parity value are stipulated as follows. (we define N as the number of strips, n as the number of blocks for each strip, Pk as the location of parity block and B as size of one block).

$$P(k) = \begin{cases} P_{k-1} + (n+1) * B & (0 < k < N-1) \\ P_(k - N - 1) + (N-1) * B & (k \geq N-1) \end{cases}$$

According to the formula, we can easily find out the specific parity value that we need. Therefore, the data information is placed into the rest of the blocks. So the position of a group (several data and its parity value consist of a typical group) is always in some regular places in our model. For example, as figure 3 shows, we define 3 strips exit here, and each has 4 blocks, so the number of N is 3, the number of M is 4 at this time, and the position of the parity blocks should be converted as the formula.

The algorithm of distributing parity blocks

```
begin
1  While k < N-1 do
2   Pk = P(k-1) + (n + 1) * B
3     WRITE (RecentNVM → value ← parity) to Pk
4     end while
5       While k >= N-1 do
6         Pk = P(k-N-1) + (N - 1) * B
7      WRITE (RecentNVM → value ← parity) to Pk
8    end while
end
```

III. NVM FAULT TOLERANT ALGORITHM

In this section, we propose a specific algorithm that is used to adapt to our method. The three main operations include read, write and recovery. Then we will illustrate the algorithm by these three processes.

A. Read

The first data block can be easily found out in our method when data is need to be read out of NVM, in this case, data blocks in NVM will be accessed regularly through the linked list, however, parity blocks will not be read out because of their flag information is set to logical 1, so that all the information in parity blocks will be skipped during the read operation, eventually data information that needed by programs is completely read out of NVM.

B. Write

To begin with write operation, a widely used method named block level striping is utilized to divided data into several data blocks, so that data can be allocated into the data blocks in NVM. During the write operation, the change of the pointer begins from the first block of strip 1, and will convert to the first block of strip 2, then will point to the first block of the next strip, until it points to the first block of strip N, finally the group 1 has been traversed. At this time, the pointer will convert to the second block of small memory N and then back to the second block of strip 1 of group 2 and will turn to group3 and up until group M in the same way.so the groups and their blocks will be linked to a list through a regular way.

Data will be written into blocks during the write operation, information of the data blocks of the same group will be XOR so that the parity value of this group is obtained through the XOR operation, then the parity value is stored in the parity block of this group. As a result of it, a complete group with several data blocks and one parity block is formed during the write operation, similarly the rest of groups are formed.

In figure 4, there are 3 strips with 4 blocks for each one. Block A and block B will be allocated with data through the write operation, and then block C will store the parity value of group 1 after the XOR operation and then data will be put into block D and block F, block E is stored the parity value of group 2, similarly, block I and block H store the data, while block G stores the parity value of group 3.

The writing of NVM Fault tolerant algorithm

```
begin
1  Write in NVM
2   count ← 0
3    for block are in BLOCKS do
4     if count ==0
5        WRITE(value and parity) from strip1 ... to strip n
6         count ← 1
7     elif
8    WRITE(value and parity) from strip n to strip1
9  count ← 0
11     end if
12       end for
end
```

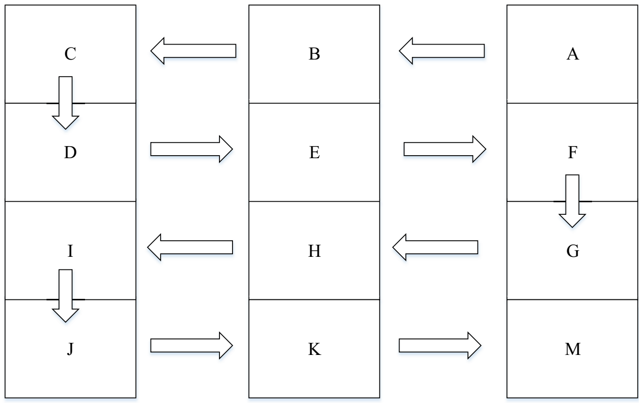

Fig. 4. The way of placing data information

C. Recovery

Data recovery is a practical function of our method, it can cope with the situation when broken blocks exit in NVM and effectively improve the utilization of the memory, so that the lifetime of NVM will be extended. The procedure is as follows, once some data blocks exhibit faults, instead of discarding them as unusable positions, the data blocks and the parity block of the group will be used to recovery the information Two cases exit here in this situation, in the first case, there are some faults in the data blocks, the value from the other data blocks and parity blocks will be XOR, so the broken data will be recovered. In the other case, there are some faults in the parity blocks, the value from the data blocks in their groups is utilize to be XOR and will gain a new parity value to renew the broken ones. For example, figure 4 shows how the broken blocks are recovered in our method, we distribute some data into data blocks at first, and the parity value of all the four groups are stored in parity blocks. Data block A and Parity block GH are broken as it shows, data block B and parity block AB are used to be XORso the data block A is recovered. The parity value will also be renewed by the operation of XOR between data block G and data block H. As a result, data will be recovered and the broken blocks will back to use.

IV. EXPERIMENTAL RESULTS

In this section, we give some experimental results to illustrate the recovery rates and time consumption of our algorithm.

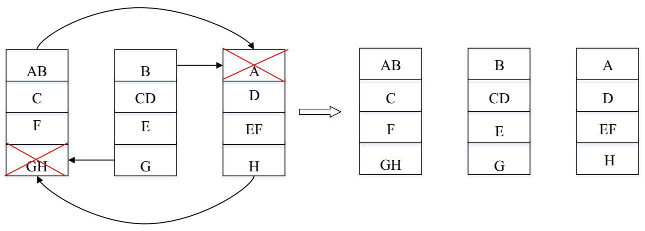

Fig. 5. recovery process

The recovery of NVM Fault tolerant algorithm

```
begin
1  Check mistakes in NVM
2  for block are in BLOCKS do
3     count ← 0
4     forstrip are in STRIPS do
5       if RecentNVM → misflag == 1
6         count ← count + 1
7     end if
8   if conut > 1
9  CANNOT recovery
10    else if
11      for strip are in STRIPS do
12    if RecentNVM → misflag == 1
13    $RecentNVM \to value \leftarrow 0$
14     forstrip are in STRIPS do
15      if$c_s trip! = strip$
16     $RecentNVM \to value = LastNVM[block][c_s trip] \to value;$
17    end if
18      end for
19   end for
end
```

In the experiments, we measure how many errors in the blocks can be recovered in an effective way and how many system overhead is acceptable to the system by using our methods.

In our experiment, we use a computer with Intel Core i5-4460 3.20GHz processor and the operating system we use is Ubuntu 14.04 LTS with a Linux kernel 3.13.0. To simulate the realistic environment, the main memory is divided into two parts, part 1 is managed by the operating system as the main memory, part 2 is used as the experimental environment to simulate the NVM. We give several storage blocks of 16KB as the units of data and parity blocks. As Fig. 6 (a) shows, model A is divided into 4 strips with 64 blocks of 16KB for each strip and there are 256 blocks as data blocks or parity blocks, then the storage capacity total is 4MB. Data is filled in the 3 data blocks and the parity value is filled in the parity block for each group. Each storage block is protected with an error correction code. Once there are some faults in the blocks, the value of misflag will change in our algorithm and the broken blocks will be marked. The position information of the blocks that exit errors are accessible and the broken blocks will be recovered only when the number of it within 2 blocks for each group during the read operation.

Since the recovery rate is related to the number of the blocks in each group and number of the groups in the models, 3 other models are given to measure the capacity and the consumption of recovery in our method. As Fig.6(b) shows, model B is 8 strips with 32 blocks for each strip. In Fig.6 (c), model C is 8 storage parts with 64 blocks for each part. In Fig.6 (d), model D is 4 strips with 128 blocks for each strip. The capacity of model A and model B is 4MB, while the capacity of model C and model D is 8MB.

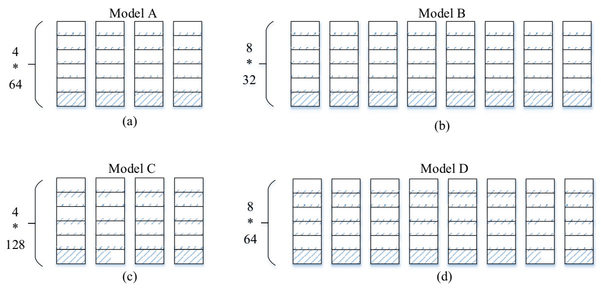

Fig. 6. Four models of storage space

A. *recovery Rates*

In order to evaluate the recovery rate, the broken blocks are distributed randomly. The experimental results are given for the different numbers of broken blocks several times to calculate the average value. Then we can give a result of recovery rate by these experiment data and record the recovery time and read time, so the capacity of recovery can be evaluated accurately. In our experiment, we measure their numerical values in these four cases to analyze the recovery rates and the time consumption.

Shown as Table I to Table IV, error number, error rate, recovery rate, read time, recovery time and the time consumption rate of the four models corresponding to the figure 6 are recorded. According to the experiments, we find that if there are more than 50 percent of the error blocks exit after the recovery operation, the ability of recovery is not efficient, so we define 50% as the threshold value of recovery rate, so it is significant to research the situations that the value of recovery rate is more than 50%.

As table I shows, the recovery rates decrease as the number of error blocks increase, and the recovery rate reaches the threshold value when the number of error blocks are 50, in which time the error rate is 19.5%. Shown as table II, the threshold value is achieved when the number of error blocks are 160, and the error rate is 31.3%. Table 3 shows that there are 30 error blocks and the error rate is 11.7% when the threshold value is achieved. It can be known from Table 4 that the recovery rate reached the threshold when the number of error blocks are 140 and the error rate is 27.3%.

Fig 7 shows the relationship of error number and recovery rate between model A and model B, while Figure 8 shows the same rate between model C and model D in figure 6. It represents the capacity of recovery between models of the same storage size. By comparison, the recovery rate of model A is almost more 16.4% than that of model B on average,

TABLE I
THE EXPERIMENT DATA OF MODEL A

Error Number	**Error Rate(%)**	**Recovery Rate(%)**	**Read Time(ms)**	**Recovery Time(ms)**	**Time Consumption Rate(%)**
10	3.9	92	46.60	0.20	0.43
20	7.8	81	48.70	0.81	1.66
30	11.7	73	50.68	1.91	3.78
40	15.6	64	49.48	2.80	5.6
50	19.5	52	48.75	5.0	10.35

TABLE II
THE EXPERIMENT DATA OF MODEL B

Error Number	**Error Rate(%)**	**Recovery Rate(%)**	**Read Time(ms)**	**Recovery Time(ms)**	**Time Consumption Rate(%)**
10	3.9	82	33.15	0.63	1.92
20	7.8	68	32.40	2.01	6.22
30	11.7	50	34.52	4.66	13.50
35	13.7	43	31.12	3.66	11.78
40	15.6	33	28.86	4.99	17.29

TABLE III
THE EXPERIMENT DATA OF MODEL C

Error Number	**Error Rate(%)**	**Recovery Rate(%)**	**Read Time(ms)**	**Recovery Time(ms)**	**Time Consumption Rate(%)**
50	7.8	21	65.34	1.15	1.77
100	9.8	78	61.65	3.37	5.47
120	19.5	64	60.29	3.51	5.83
140	23.4	58	59.34	4.60	7.75
160	31.3	51	54.62	5.01	9.17

recovery rate of model C is nearly more 8.6% than that of model D on average. we can find that the model with less number of strips will get a higher recovery rate than the other one of the same size. Comparing model A to model C, the error rate(the rate number of error blocks to the number of all the blocks) of model C is more 11.7% than that of model A when the recovery rate is near to 50%,and the error rate of model D is more 15.6% than that of model B when the recovery rate is near to 50%, the number of groups in model C is two times of groups in model A, and the number of groups in model D is two times of groups in model B, so the capacity can be improved by increasing the number of groups at a certain degree.

B. Time consumption

The time consumption is the main factor of the system consumption in our method, we need to ensure that it will not cost too much time during the recovery operation. It is necessary to measure the time consumption of different cases with different number of error blocks in the cases, so we can know the range of time consumption and evaluate whether the consumption is acceptable for the programs.

As figure 9 and figure 10 show, the recovery time of model

TABLE IV
THE EXPERIMENT DATA OF MODEL D

Error Number	**Error Rate(%)**	**Recovery Rate(%)**	**Read Time(ms)**	**Recovery Time(ms)**	**Time Consumption Rate(%)**
50	9.8	77	50.11	1.99	3.99
70	13.7	71	47.90	3.04	6.35
100	19.5	64	46.59	4.66	10.01
120	23.4	56	46.29	5.48	11.84
140	27.3	50	46.37	6.26	13.49

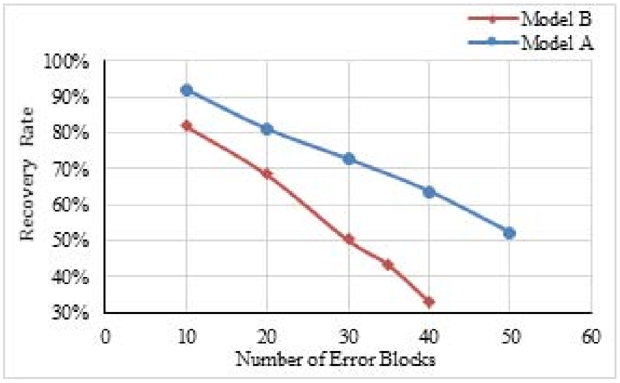

Fig. 7. The relationship between error number and recovery rate of model A and model B

B is more than that of model A, and the recovery time of model D is more than that of model C, we can find that the models that have less number of groups will product more recovery time, so the recovery time will be extended by enlarging the size of the memory. Accordingly, we can evaluate the affection of recovery time to the time cost in the read operation, thus the rate can be defined as follows.

Shown as Table I to Table IV, we can find that TCR(time consumption rate)will improve as error rate increases, but if the error rate is less than 10%, TCR will be less than 5% of the four models, in this situation, recovery operation will keep a low rate of time consumption. Comparing Table I to Table IV, we can find that it will more efficient for NVM if we divide the memory into less number of strips.

V. CONCLUSION AND FUTURE WORK

In this paper, we argue that it is inefficient for NVM to discard the error blocks as unusual parts. we propose a new method that inspired by RAID technology, which is redundant and be able to recovery error data. The result of experiment shows that model with less number of strips will get a higher recovery rate than other models of the same size, and the capacity of recovery can be improved by increasing the number of blocks in each strip in our method. Conclusively, more than 50% of error data can be recovered efficiently if the error rate is under 30%, and the recovery rate is nearly more than 80% if the error rate is under 10%, while the time consumption rate is less than 5% in this situation. In the future

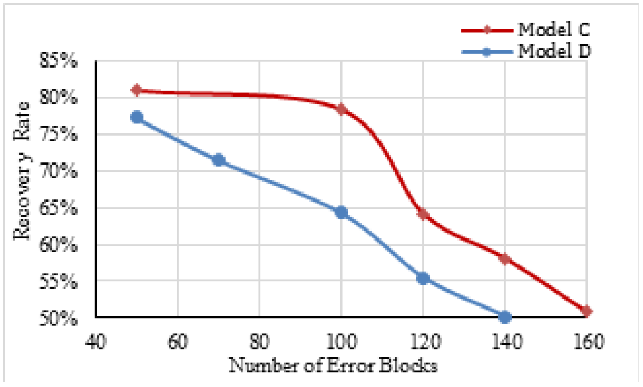

Fig. 8. The relationship between error number and recovery rate of model C and model D

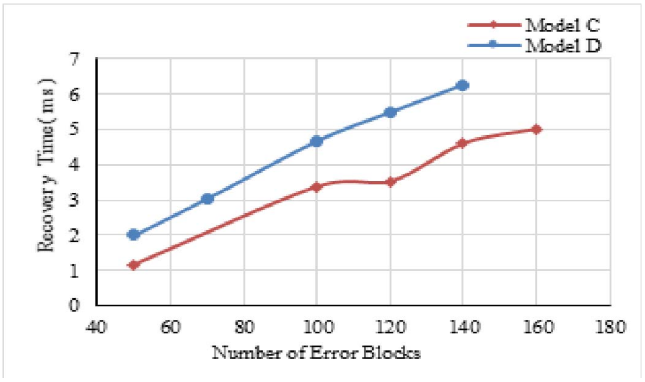

Fig. 9. The relationship between error number and recovery time of model D and model C

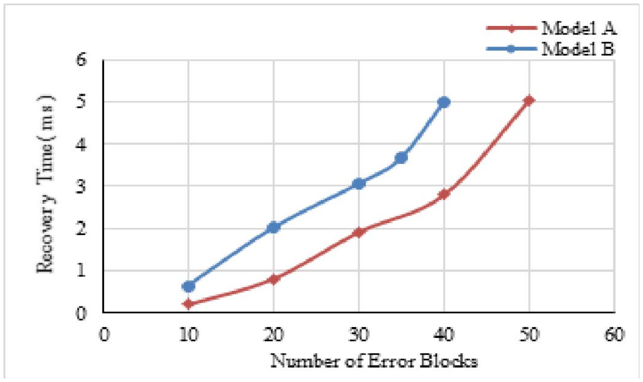

Fig. 10. The relationship between error number and recovery time of model B and model A

work, many different schemes will be researched to improve the efficiency of recovery with less write operation.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61003032, the Research Fund of National Key Laboratory of Computer Architecture under Grant No. CARCH201501, and the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing under Grant No. 2016A09.

References

[1] S. Lee, T. Kim, K. Kim, and J. Kim, "Lifetime management of flash-based ssds using recovery-aware dynamic throttling," in *Proceedings of the 10th USENIX Conference on File and Storage Technologies*, ser. FAST'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 26–26. [Online]. Available: http://dl.acm.org/citation.cfm?id=2208461.2208487

[2] N. Papandreou, T. Parnell, H. Pozidis, T. Mittelholzer, E. Eleftheriou, C. Camp, T. Griffin, G. Tressler, and A. Walls, "Enhancing the reliability of mlc NAND flash memory systems by read channel optimization," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 20, no. 4, pp. 62:1–62:24, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2699866

[3] E. Ipek, J. Condit, E. B. Nightingale, D. Burger, and T. Moscibroda, "Dynamically replicated memory: Building reliable systems from nanoscale resistive memories," *SIGPLAN Not.*, vol. 45, no. 3, pp. 3–14, Mar. 2010. [Online]. Available: http://doi.acm.org/10.1145/1735971.1736023

[4] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ecp, not ECC, for hard failures in resistive memories," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 141–152, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1815980

[5] J. Chen, G. Venkataramani, and H. H. Huang, "Exploring dynamic redundancy to resuscitate faulty PCM blocks," *J. Emerg. Technol. Comput. Syst.*, vol. 10, no. 4, pp. 31:1–31:23, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2602156

[6] M. K. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali, "Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 14–23. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669117

[7] L. M. Grupp, A. M. Caulfield, J. Coburn, S. Swanson, E. Yaakobi, P. H. Siegel, and J. K. Wolf, "Characterizing flash memory: Anomalies, observations, and applications," in *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO 42. New York, NY, USA: ACM, 2009, pp. 24–33. [Online]. Available: http://doi.acm.org/10.1145/1669112.1669118

[8] S. Schechter, G. H. Loh, K. Strauss, and D. Burger, "Use ecp, not ECC, for hard failures in resistive memories," *SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 141–152, Jun. 2010. [Online]. Available: http://doi.acm.org/10.1145/1816038.1815980