A Reconfigurable FTL Architecture for NAND Flash-Based Applications

CHANIK PARK, WONMOON CHEON, JEONGUK KANG, KANGHO ROH, and WONHEE CHO
Samsung Electronics
and
JIN-SOO KIM
Korea Advanced Institute of Science and Technology

ACM Transactions on Embedded Computing Systems, Vol. 7, No. 4, Article 38, Publication date: July 2008.

Yen-Ting Liu

Outline

- Introduction
- Mapping Schemes
- Flexible Group Mapping
- Model and Analysis
- Experimental Results
- Conclusion

Introduction(1/2)

- FTL
  - Translate logical address to a physical address in flash memory
  - Efficient algorithms of an FTL have a significant impact on performance as well as lifetime

Introduction(2/2)

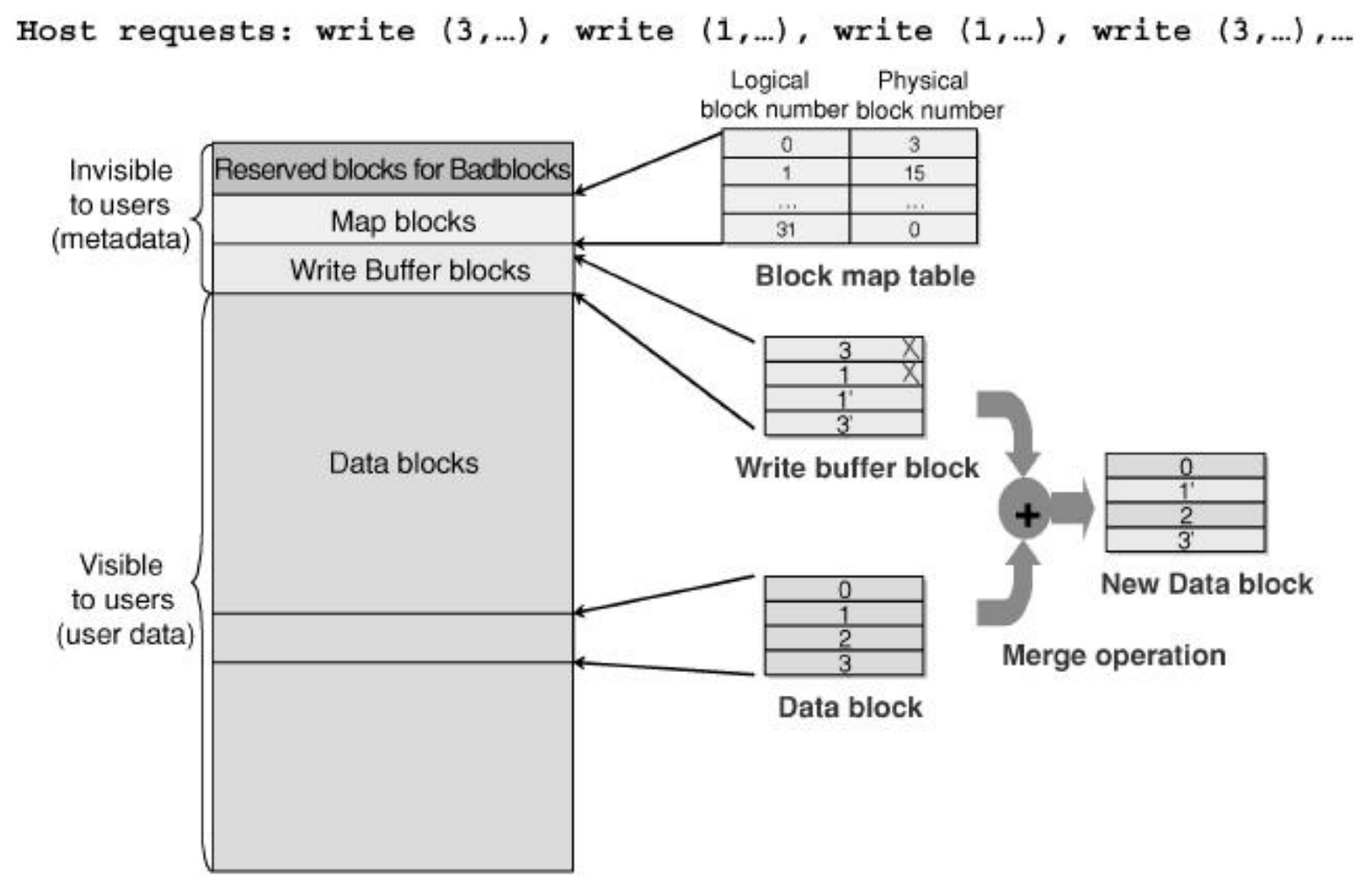


Fig. 4. Logical view of the FTL of NAND flash memory.

Mapping Schemes (1/6)

- Page mapping
  - A map table entry consists of an LPN (logical page number) and PPN (physical page number)
  - Requires a very large amount of memory space for the map table

Mapping Schemes (2/6)

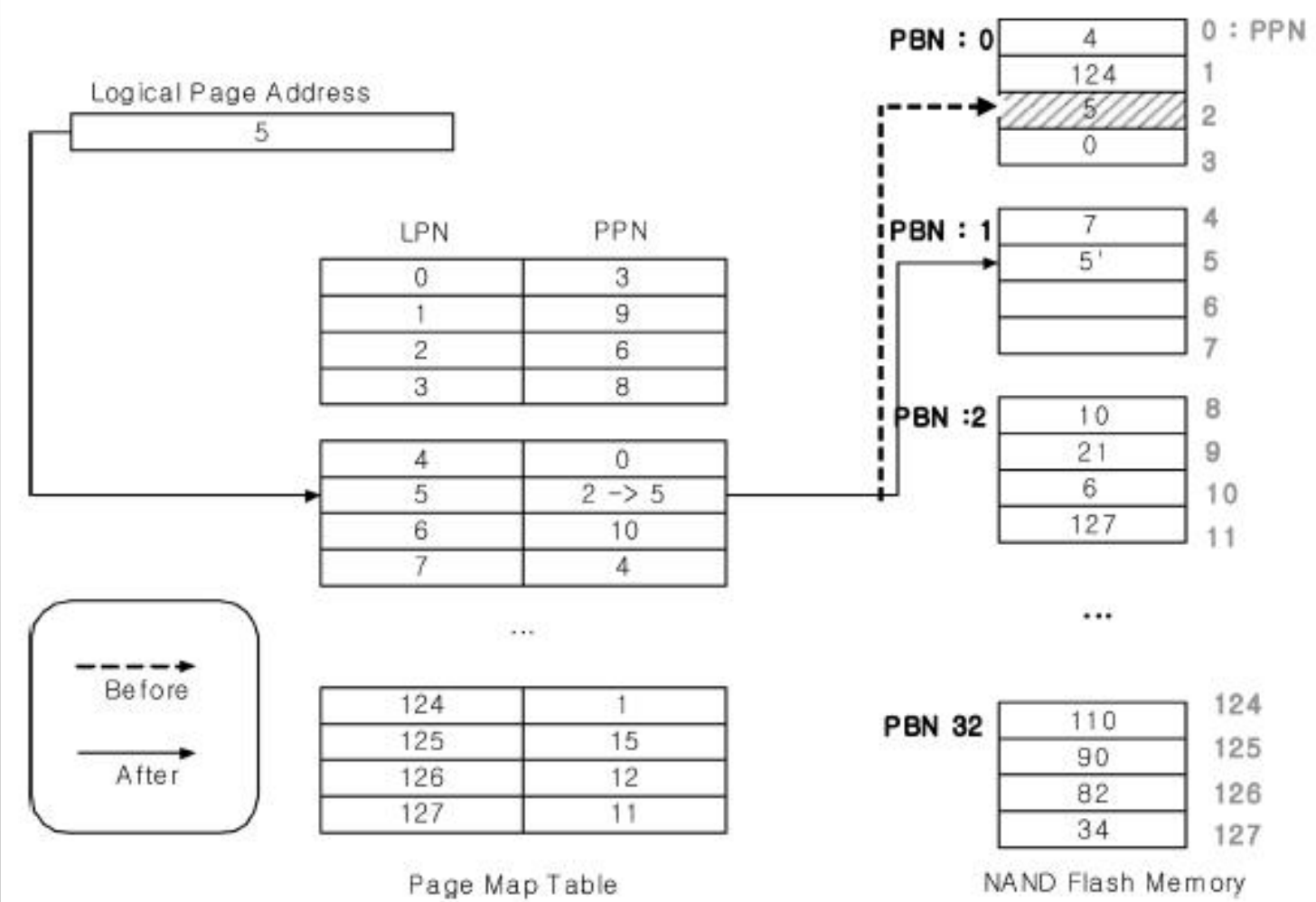


Fig. 5. Page-mapping scheme.

Mapping Schemes (3/6)

- Block mapping
  - The logical page address is divided into a logical block number and a page offset
  - Every overwrite operation to the same logical page may incur a frequent block-level copy operation
  - It may show considerable performance degradation over random-access patterns

Mapping Schemes (4/6)

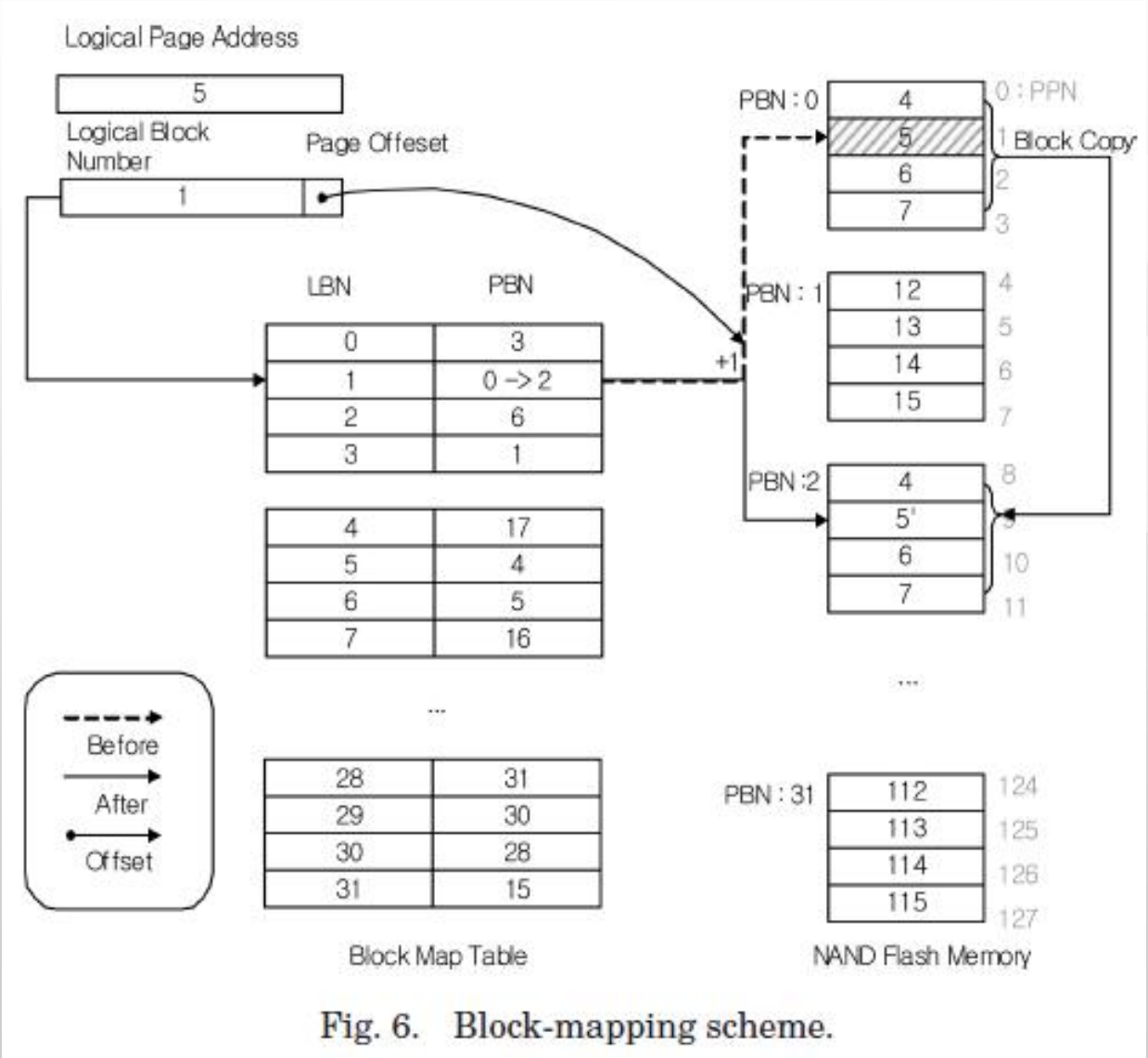


Fig. 6. Block-mapping scheme.

Mapping Schemes (5/6)

- Hybrid mapping scheme
  - A hybrid mapping know as the log block scheme
  - To maintain a small number of log blocks in flash memory to serve as write buffer blocks overwrite operation

Mapping Schemes (6/6)

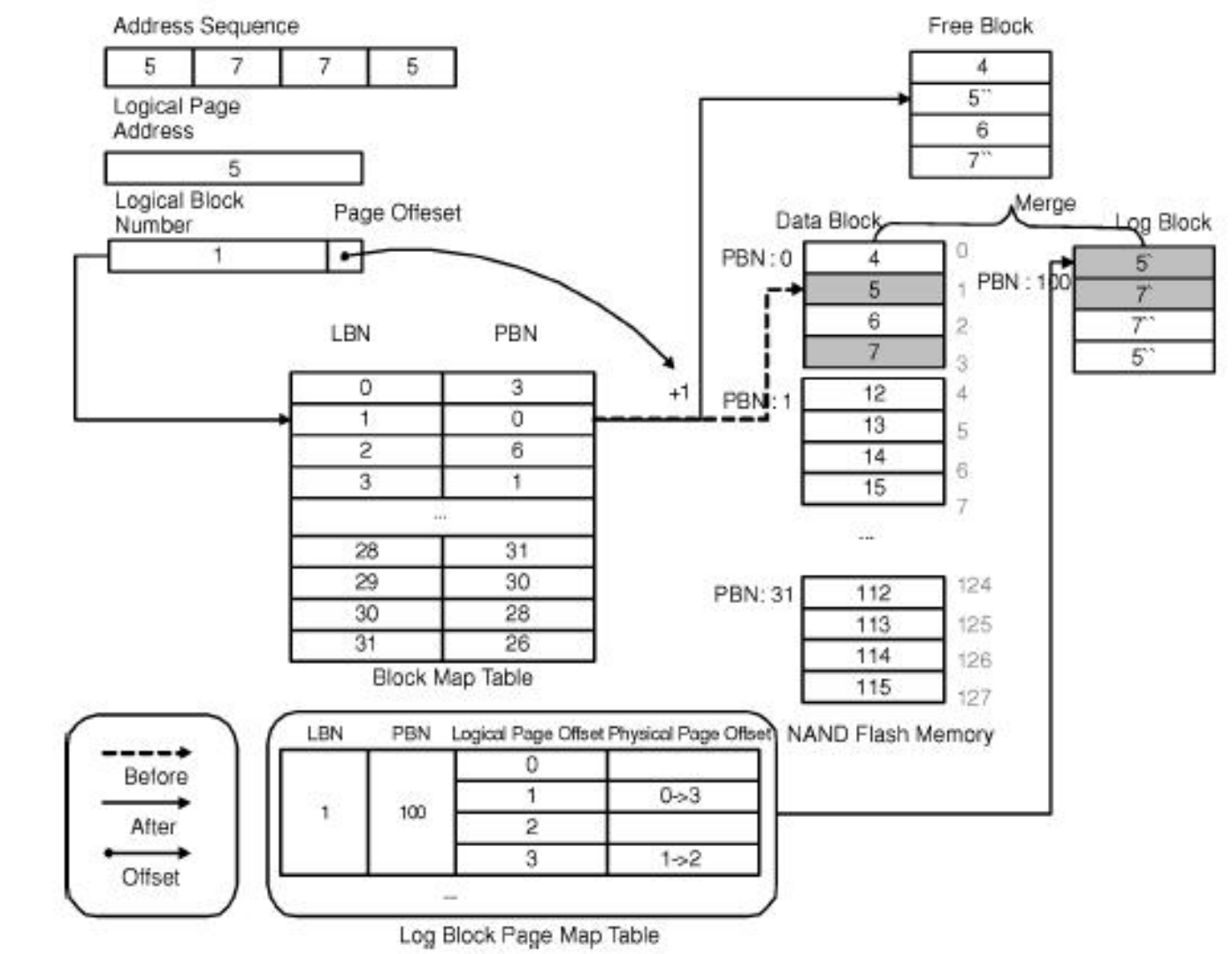


Fig. 7. Hybrid mapping scheme.

Flexible Group Mapping (1/8)

- Configure the degree of sharing of log blocks among data blocks using the block-level spatial locality parameter, N
- N is the number of data blocks in a data block group

Flexible Group Mapping (2/8)

- A log block group is a set of log blocks related to a specific data block group
- The parameter K denotes the maximum number of log blocks that can be added to a log block group
- The K parameter explains the type of temporal locality within a block

Flexible Group Mapping (3/8)

```
Host requests: write (1,…),  write (1,…),  write (14,…), write (15,…),
               write (2,…),  write (2,…),  write (3,…),  write (3,…),
               write (24,…), write (25,…), write (26,…), write (27,…),
```

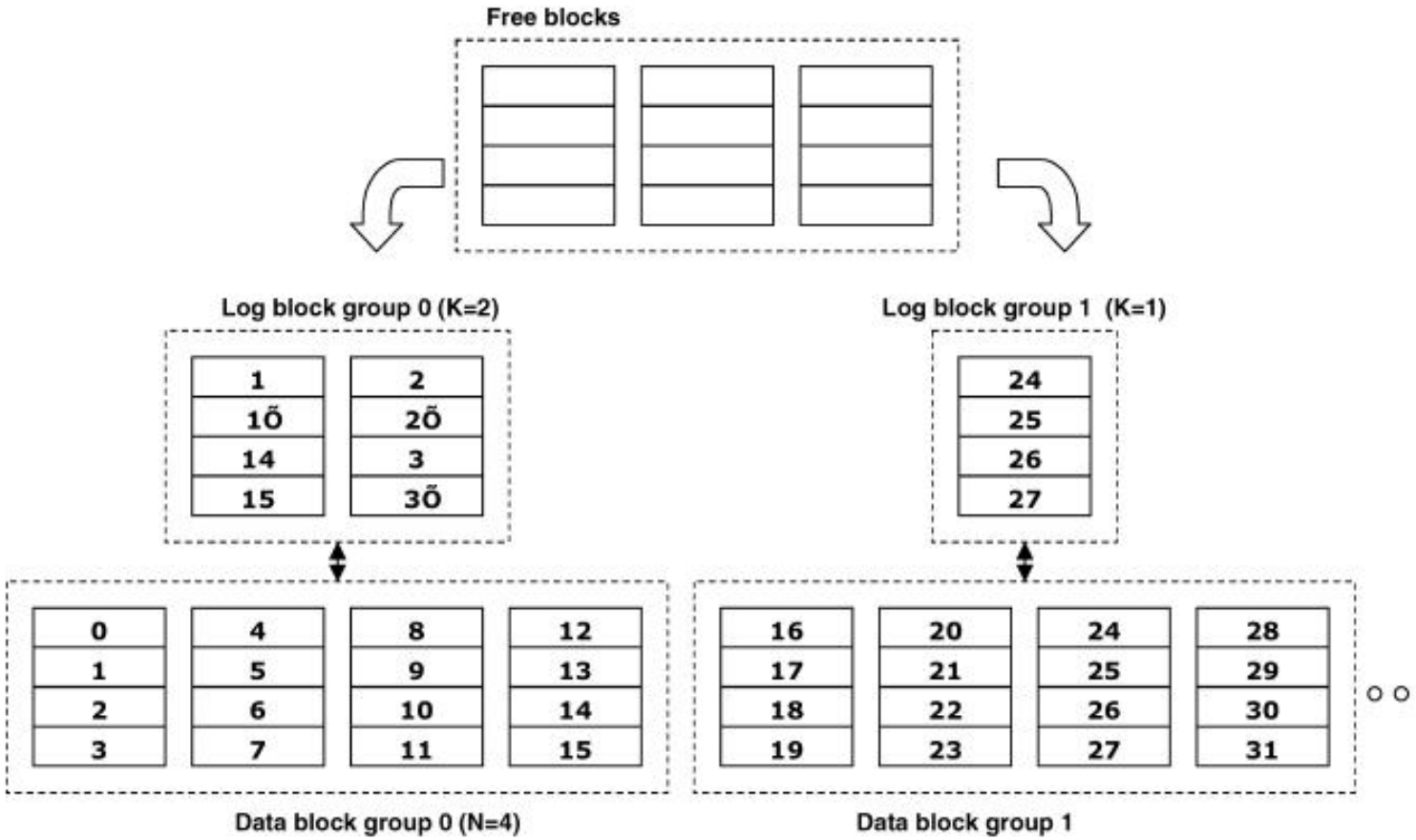


Fig. 8. Flexible group-mapping scheme.

Flexible Group Mapping (4/8)

- Write Scheme
  - Three mapping tebles
    - Data-block-mapping table (DBMT)
    - Log-block-mapping table (LBMT)
    - Log-page-mapping table (LPMT)

Flexible Group Mapping (5/8)

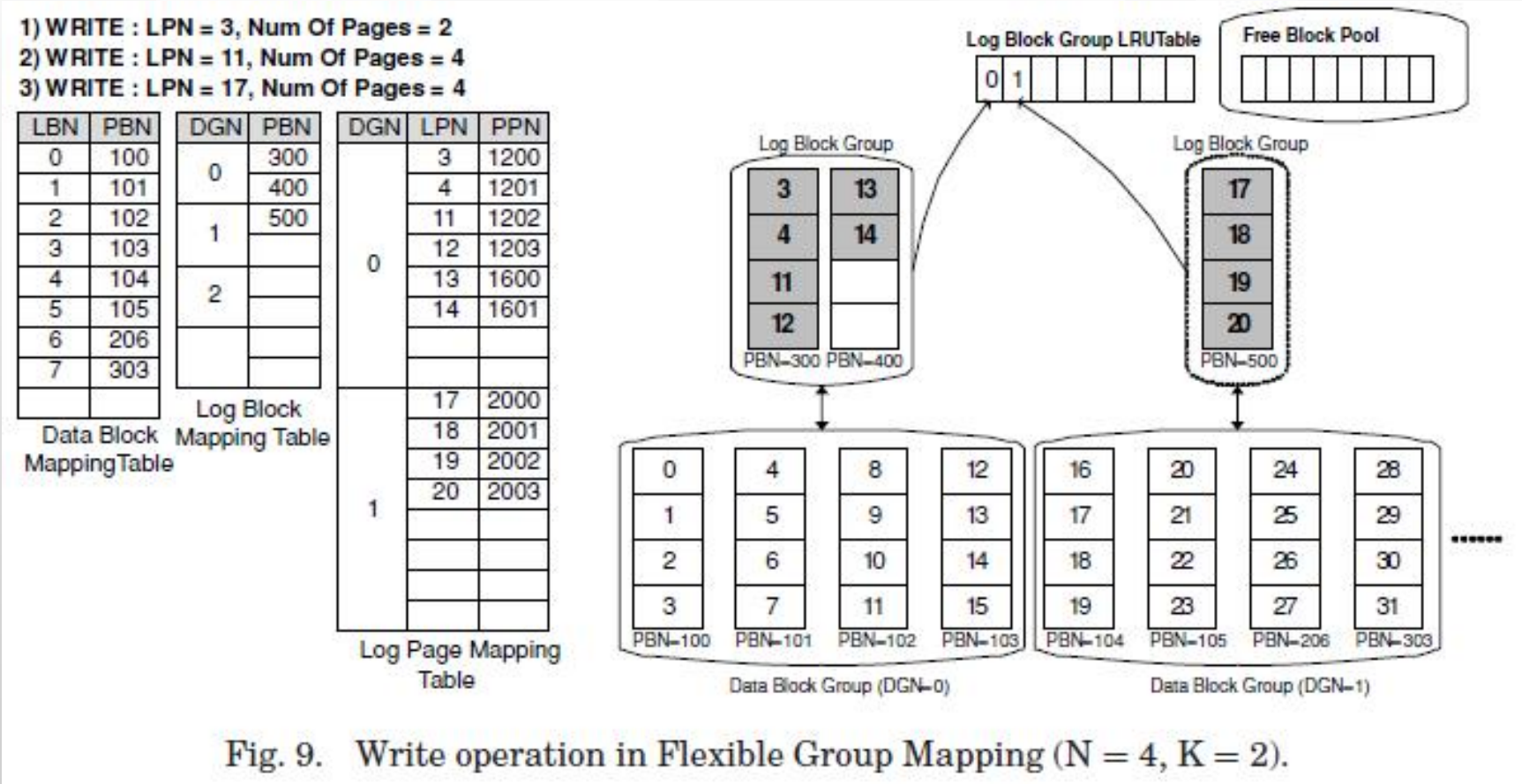


Fig. 9. Write operation in Flexible Group Mapping (N = 4, K = 2).

Flexible Group Mapping (6/8)

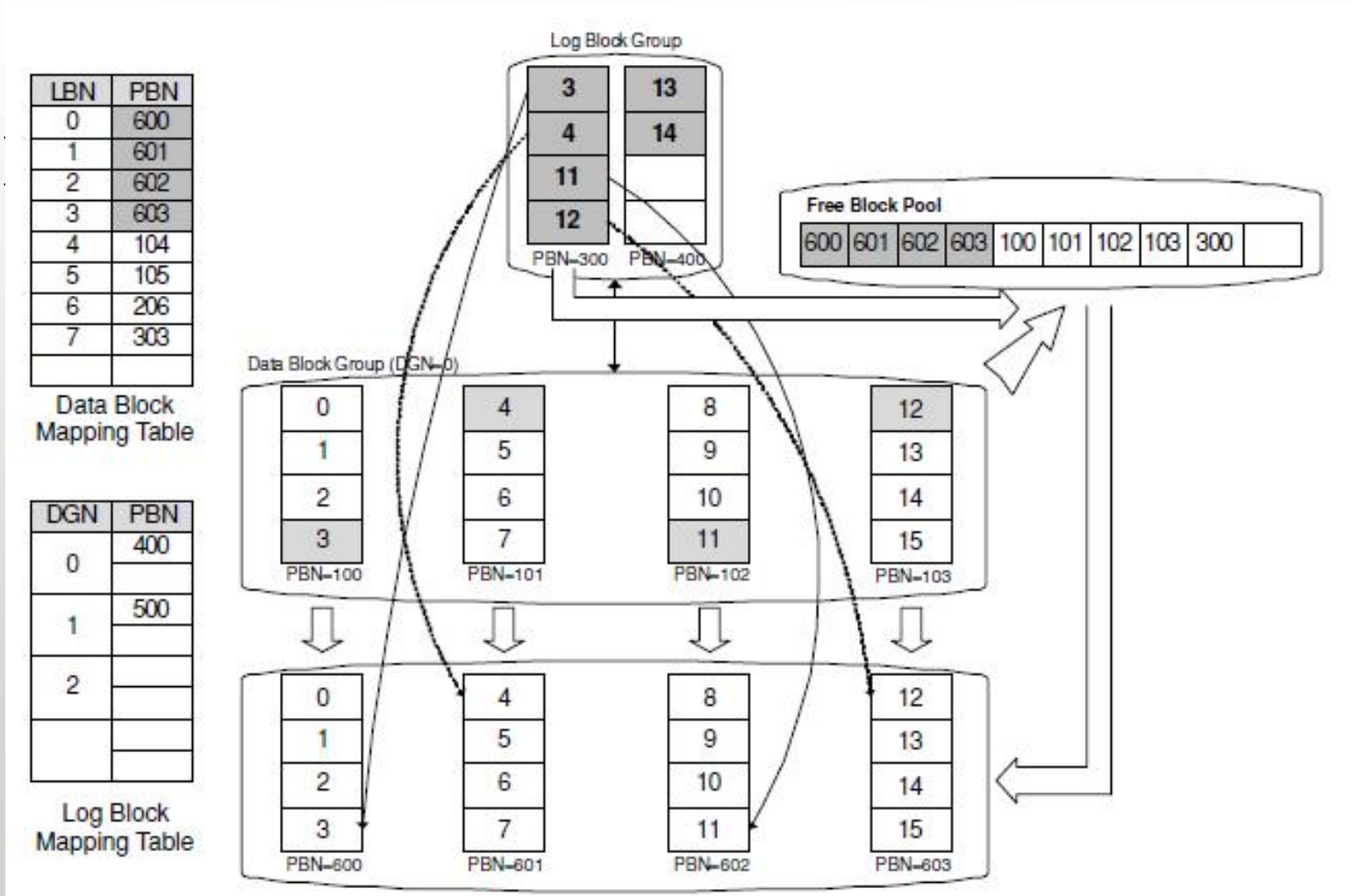
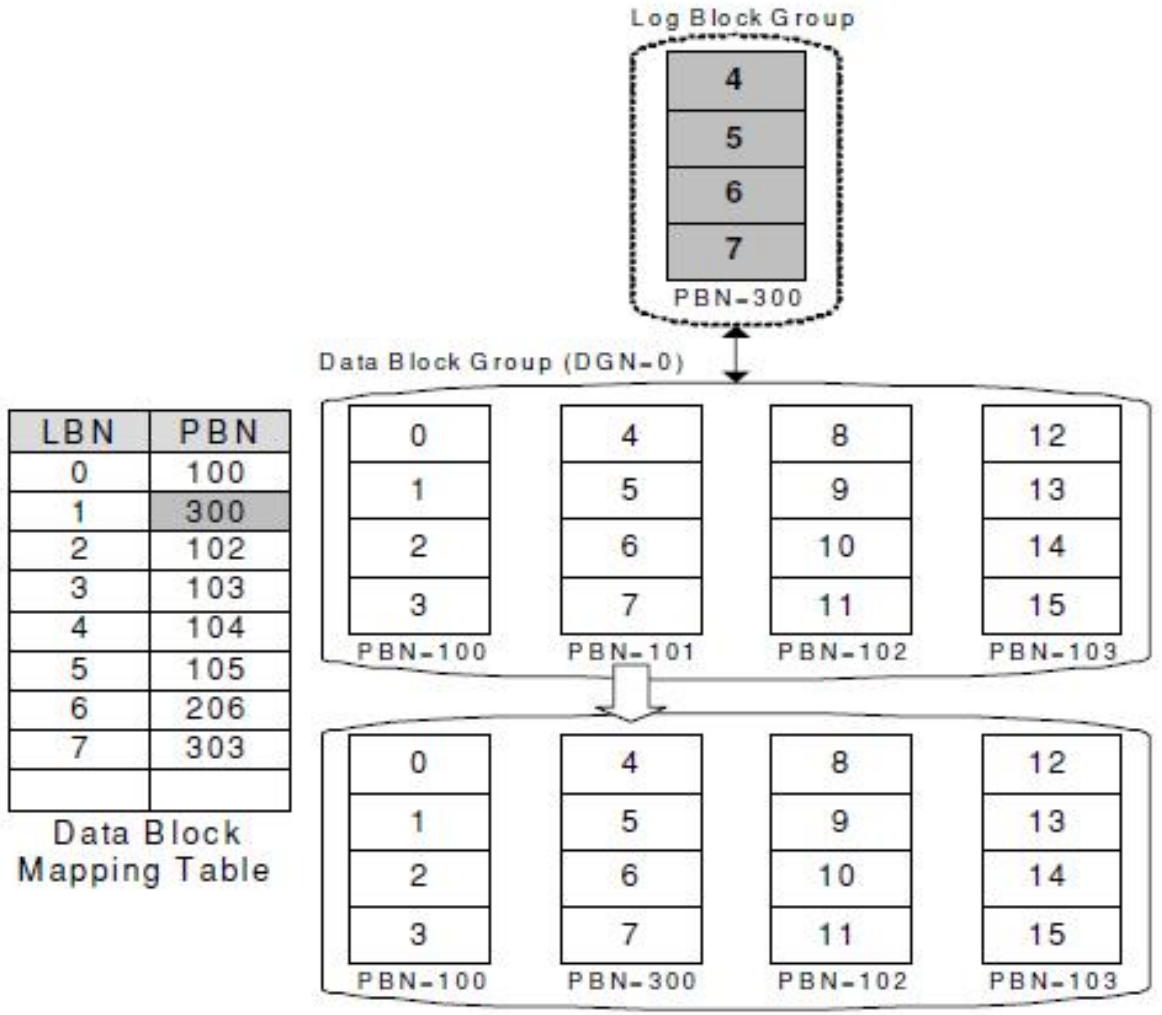


Fig. 10. An example of a simple merge operation (N = 4, K = 2).

Flexible Group Mapping (7/8)

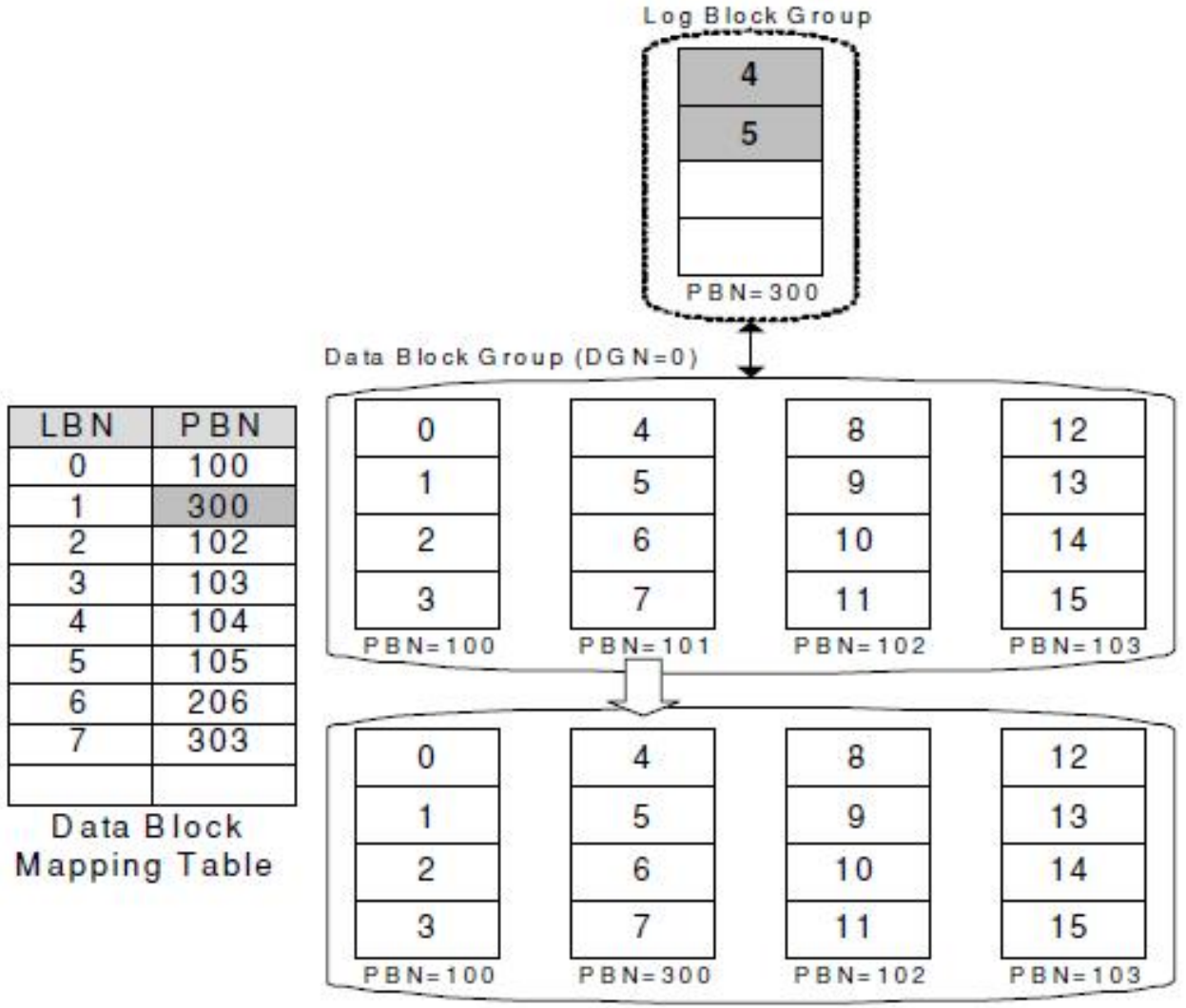


LBN	PBN
0	100
1	300
2	102
3	103
4	104
5	105
6	206
7	303

Data Block Mapping Table

Fig. 11. An example of a swap merge operation (N = 4, K = 2).

Flexible Group Mapping (8/8)



LBN	PBN
0	100
1	300
2	102
3	103
4	104
5	105
6	206
7	303

Data Block Mapping Table

Fig. 12. An example of a copy merge operation (N = 4, K = 2).

Model and Analysis (1/5)

- Workload Analysis
  - The entire group of write requests are divided into a series of nonoverlapping request windows w_j
  - The request density $RD_{i,j}$, is defined as the ratio of the number of requests accessed in the ith logical block ($C_{i,j}$) to the total number of requests in the jth window

$$N_j = \left\lfloor \min\left(\frac{1}{RD_{i,j}}\right) \right\rfloor \quad \text{for } i \in \text{all LBNs.}$$

Model and Analysis (2/5)

Requests (LPN): 0, 1, 2, 3, 4, 8, 4, 6, 8, 9, 13, 15, 13, 15, 10, 12, 3, 3, 10, 13

R0 R5 R10 R15

		W_1	W_2	W_3	W_4	W_5
LBN0	Page 0	R_0				
	Page 1	R_1				
	Page 2	R_2				
	Page 3	R_3				R_{16}, R_{17}
LBN1	Page 4		R_4, R_6			
	Page 5					
	Page 6		R_5, R_7			
	Page 7					
LBN2	Page 8			R_8		
	Page 9			R_9		
	Page 10				R_{14}	R_{18}
	Page 11					
LBN3	Page 12				R_{15}	
	Page 13			R_{10}	R_{12}	R_{19}
	Page 14					
	Page 15			R_{11}	R_{13}	

(a) An example request distribution table

	W_1	W_2	W_3	W_4	W_5
LBN0	1.00	0.00	0.00	0.00	0.50
LBN1	0.00	1.00	0.00	0.00	0.00
LBN2	0.00	0.00	0.50	0.25	0.25
LBN3	0.00	0.00	0.50	0.75	0.25
N_j	1	1	2	4	4

(b) Estimating N_j

	W_1	W_2	W_3	W_4	W_5	K_i
LBN0	0	0	0	0	1	1
LBN1	0	1	1	1	1	1
LBN2	0	0	0	0	1	1
LBN3	0	0	0	1	2	2

(c) Estimating K_i

Fig. 13. Estimating N_j and K_i from an example trace.

Model and Analysis (3/5)

- LBN i has temporal locality in the request window W_j
  - If one or more page in LBN i are updated more than once during W_j
  - or if one or more page in LBN i written in the previous window w_{j-1} are written again in the current window W_j
- $K_{i,j}$ is considered as the number of occurrences that such conditions during the interval from W_0 to W_j
- $PAGE_{i,j}$ denotes a set of pages in LBN i written during request window W_j

Model and Analysis (4/5)

$$K_{i,0} = 0$$

$$K_{i,j} = K_{i,j-1} + d_{i,j} \quad for\ j > 0$$

$$where\ \ d_{i,j} = \begin{cases} 1\ \ if\ |PAGE_{i,j}| < C_{i,j} \vee \left(\sum_{k=0..j-1} PAGE_{i,k} \cap PAGE_{i,j} \right) \neq \Phi. \\ 0\ \ otherwise. \end{cases}$$

Model and Analysis (5/5)

Requests (LPN): 0, 1, 2, 3, 4, 8, 4, 6, 8, 9, 13, 15, 13, 15, 10, 12, 3, 3, 10, 13

R0 R5 R10 R15

		W_1	W_2	W_3	W_4	W_5
LBN0	Page 0	R_0				
	Page 1	R_1				
	Page 2	R_2				
	Page 3	R_3				R_{16}, R_{17}
LBN1	Page 4		R_4, R_6			
	Page 5					
	Page 6		R_5, R_7			
	Page 7					
LBN2	Page 8			R_8		
	Page 9			R_9		
	Page 10				R_{14}	R_{18}
	Page 11					
LBN3	Page 12				R_{15}	
	Page 13			R_{10}	R_{12}	R_{19}
	Page 14					
	Page 15			R_{11}	R_{13}	

(a) An example request distribution table

	W_1	W_2	W_3	W_4	W_5
LBN0	1.00	0.00	0.00	0.00	0.50
LBN1	0.00	1.00	0.00	0.00	0.00
LBN2	0.00	0.00	0.50	0.25	0.25
LBN3	0.00	0.00	0.50	0.75	0.25
N_j	1	1	2	4	4

(b) Estimating N_j

	W_1	W_2	W_3	W_4	W_5	K_i
LBN0	0	0	0	0	1	1
LBN1	0	1	1	1	1	1
LBN2	0	0	0	0	1	1
LBN3	0	0	0	1	2	2

(c) Estimating K_i

Fig. 13. Estimating N_j and K_i from an example trace.

Experimental Results (1/7)

- Trace analyzed
  - Intel Pentium-4 PC
  - 512MB RAM and 80GB hard disk
  - OS : Windows XP
  - File system : NTFS

Experimental Results (2/7)

Internet/MS Office

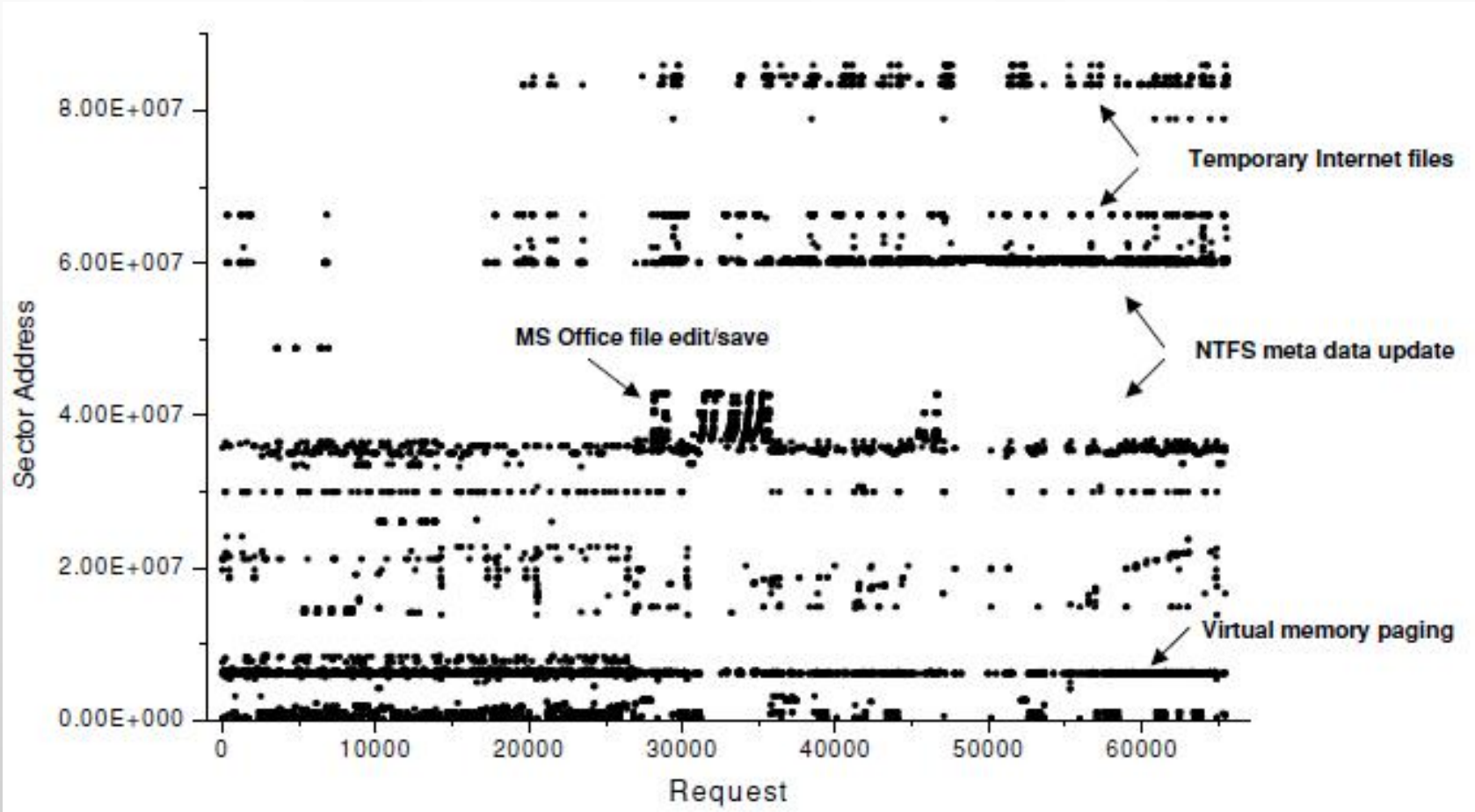


Fig. 14. Trace distribution from PC applications.

Experimental Results (3/7)

MP3 file download

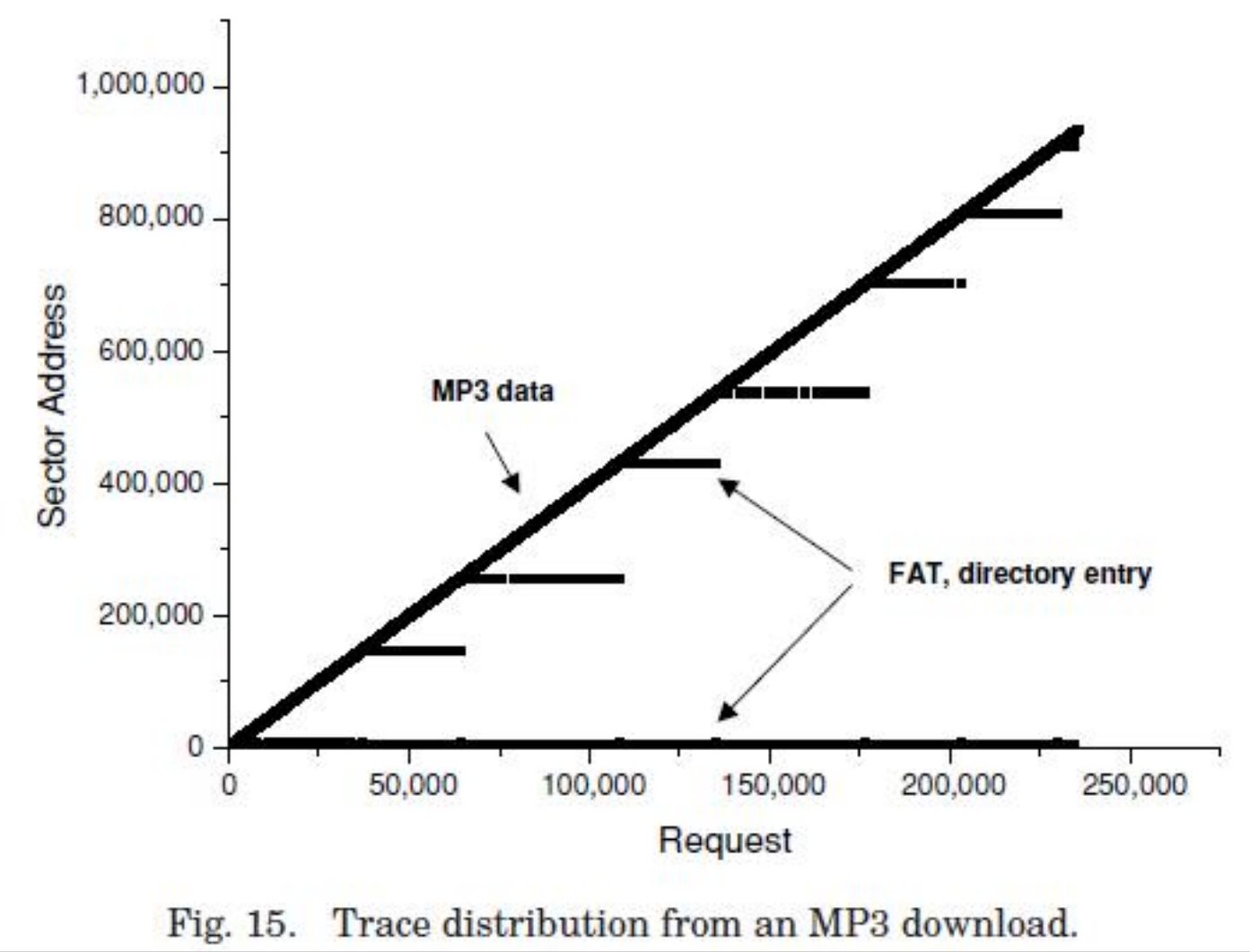


Fig. 15. Trace distribution from an MP3 download.

Experimental Results (4/7)

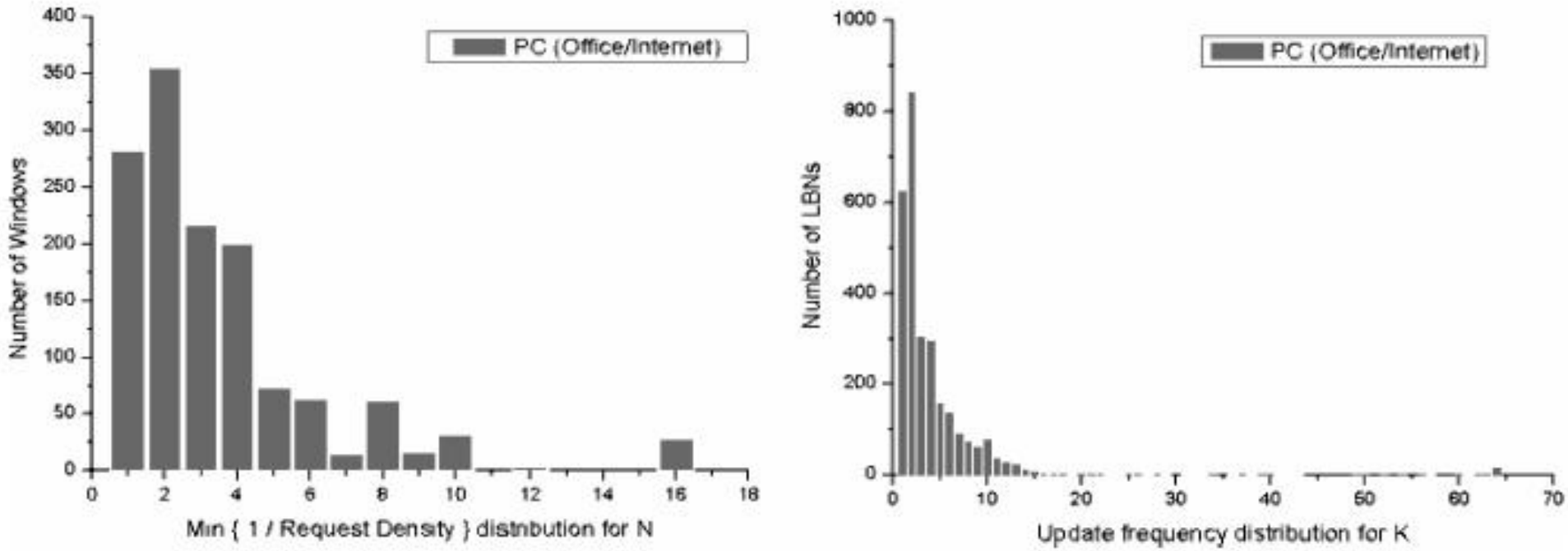


Fig. 16. Distribution of N's and K's for PC applications.

Experimental Results (5/7)

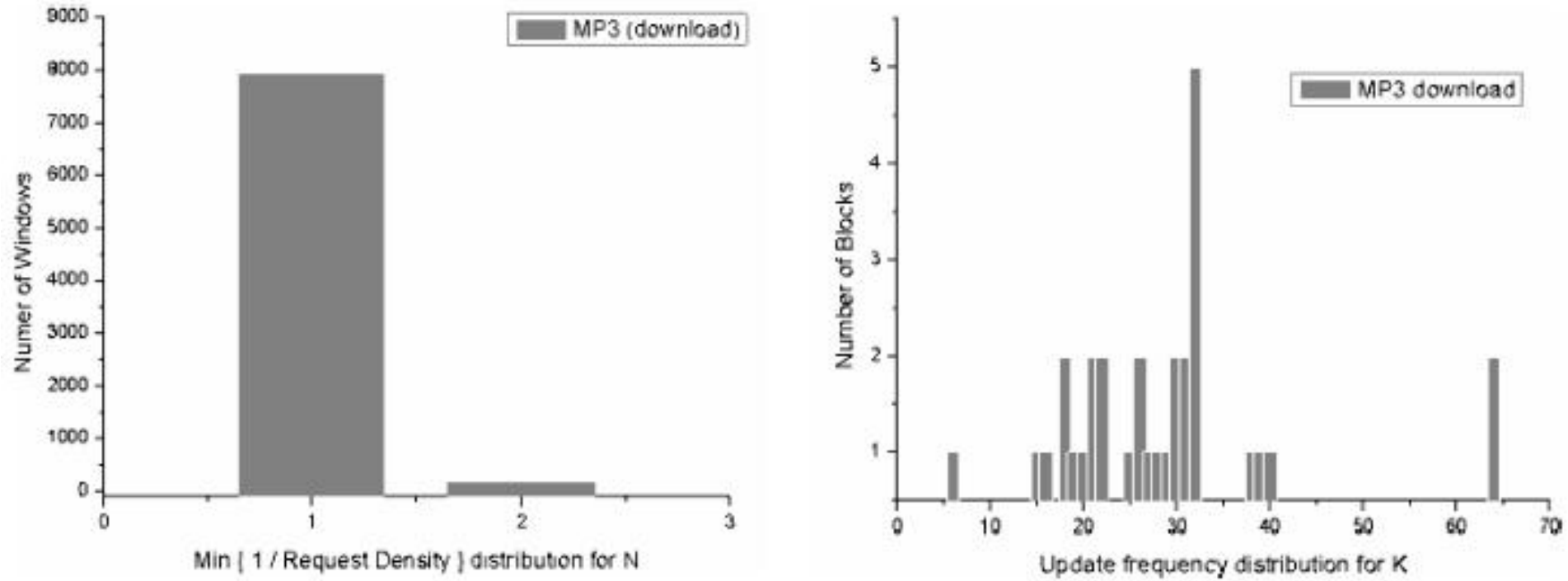


Fig. 17. Distribution of N and K values for the MP3 application.

Experimental Results (6/7)

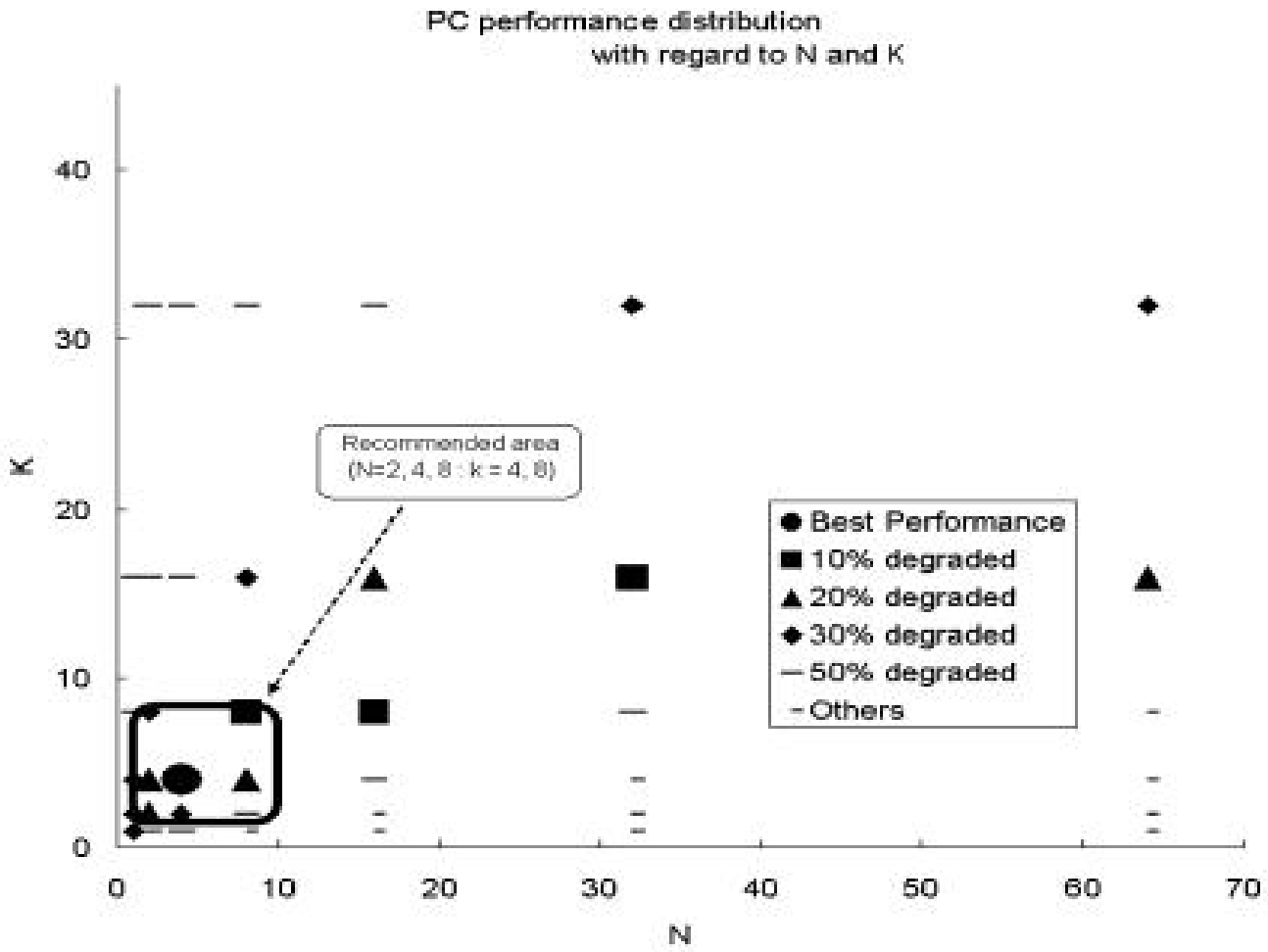


Fig. 18. PC application performance variation with the change of N and K.

Experimental Results (7/7)

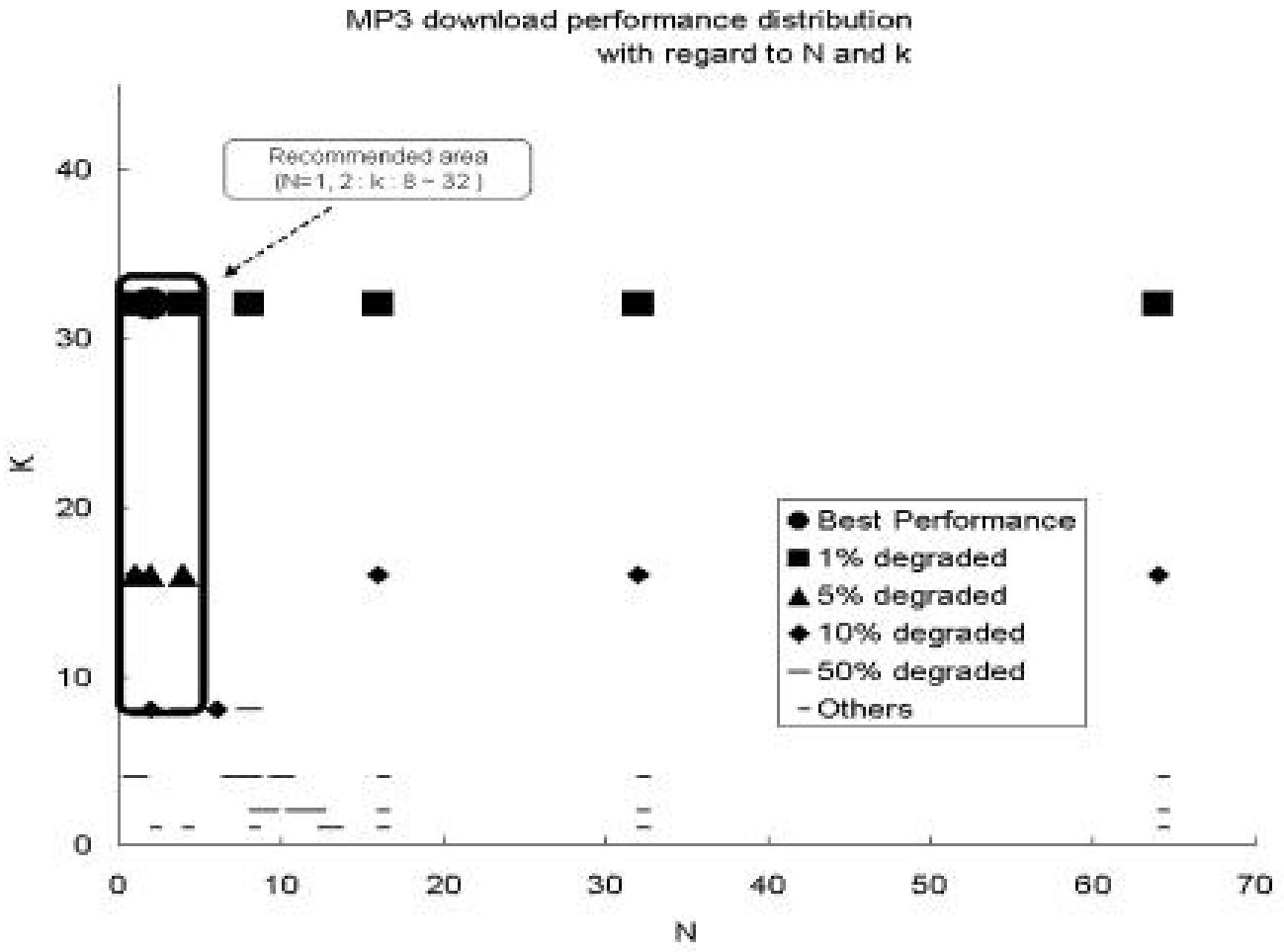


Fig. 19. MP3 download performance variation with the change of N and K.

Conclusions

- A reconfigurable FTL architecture to efficiently handle NAND flash applications
- Using two parameters, N and K to configure the FTL