A Skip-List-Based NOR-Flash File System: Exploiting the Use of Physical Pointers

學生：游仕宏

指導教授：張立平

ES^2 Lab @ NCTU

Outline

- Introduction
  - preface
  - Motivation
  - Skip list data structure
- Background and Related Work
- A Skip-List-Based File System over Nor flash
  - Fundamental Issue
  - A Skip List Implementation
  - Metadata objects and Data objects
  - Key-coding scheme
  - File and Directory operations
- Evaluation
- Conclusion
- Future work

Introduction

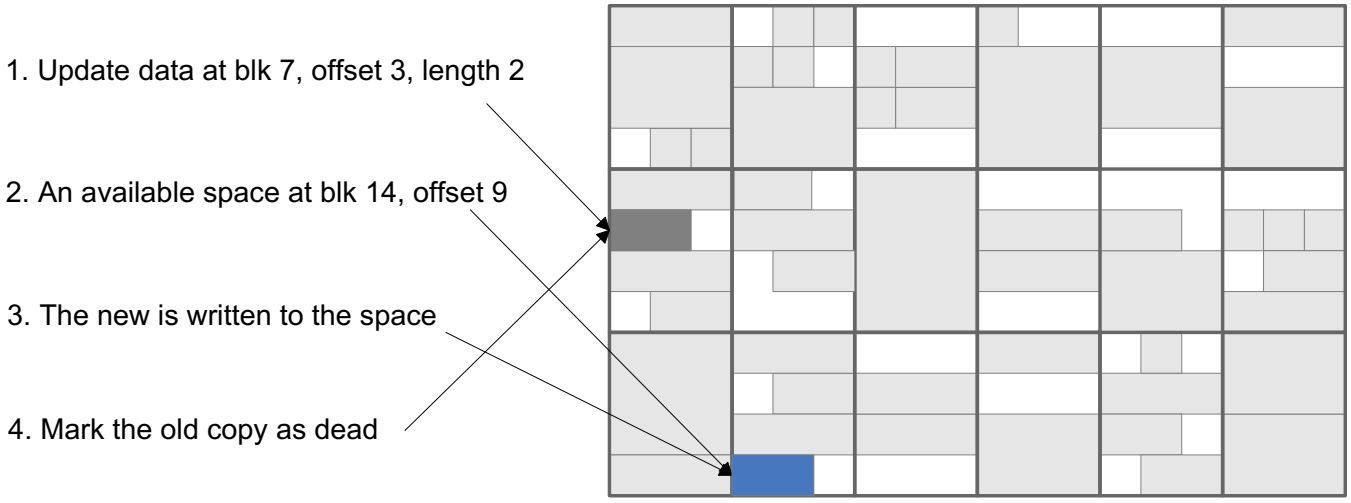
preface

- Shock-resistance、high density and power efficiency of Flash abstract embedded systems
- Flash substitute the traditional storage and be used widely on mobile/portable device
- Mobile/portable device is more powerful and Flash is too big that need to be managed efficiently

Introduction

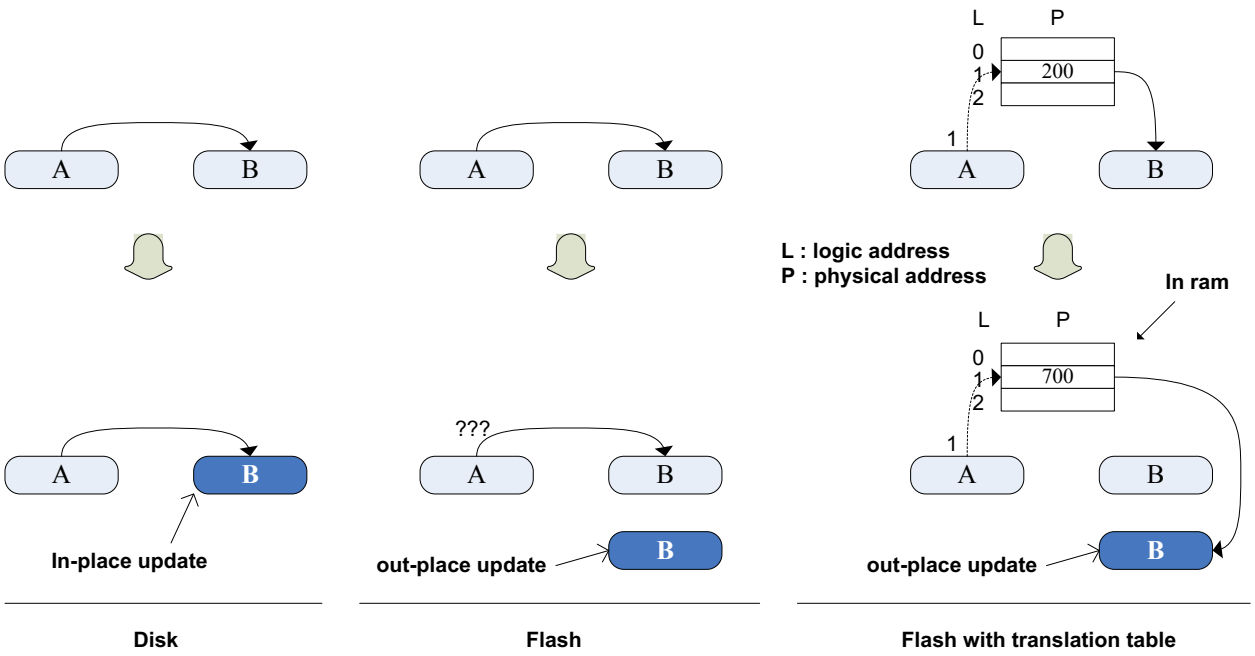
preface

- Physical property of Flash
  - Write-once, bulk-erasing, livespan
  - Performance, wear leveling → out-place update
- Out-place update



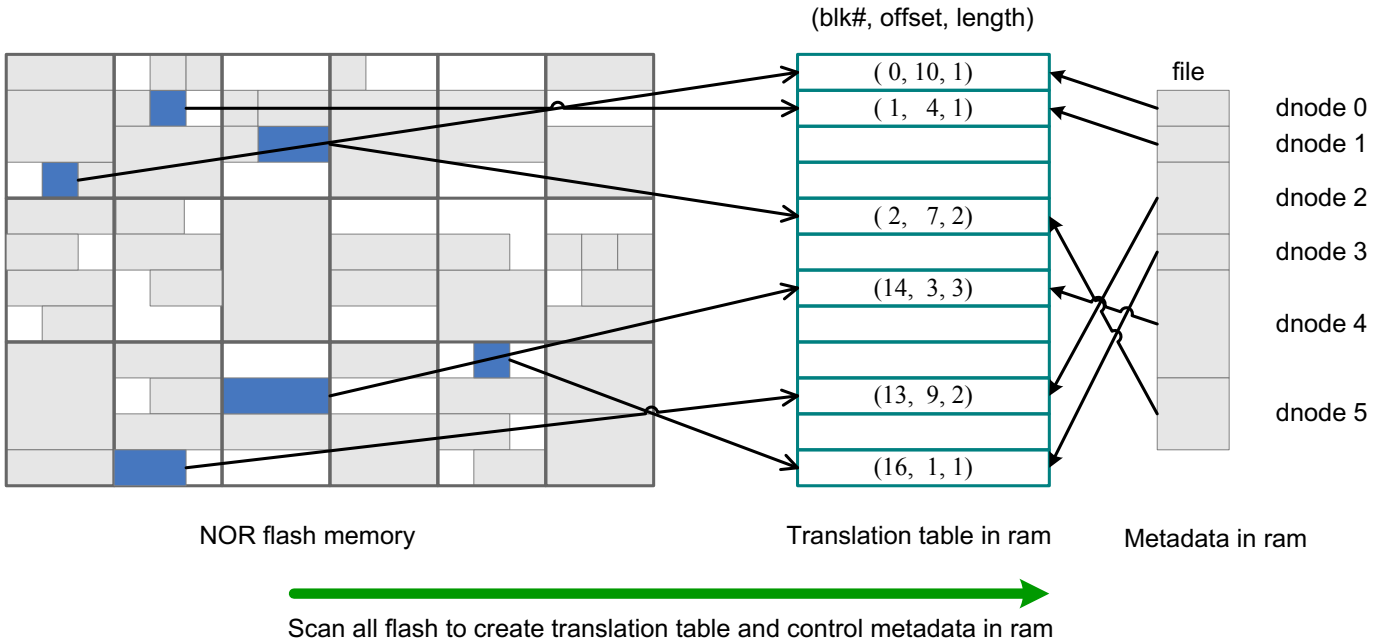
Introduction
preface

- The use of logic address decouples the updates to
  - data items
  - the pointers referring to the updated data items



Introduction
preface

- Logic address
  - Scanning on boot and long initialization time
  - Memory-space requirements → translation table, system control msg.



Introduction
Motivation

- Motivation
  - To investigate the use of physical pointers for data management over flash memory
- Goals
  - To eliminate RAM-resident translation table
  - To eliminate the initial scan procedure
  - To serve as an efficient data repository for applications and file systems

Introduction

Skip list data structure

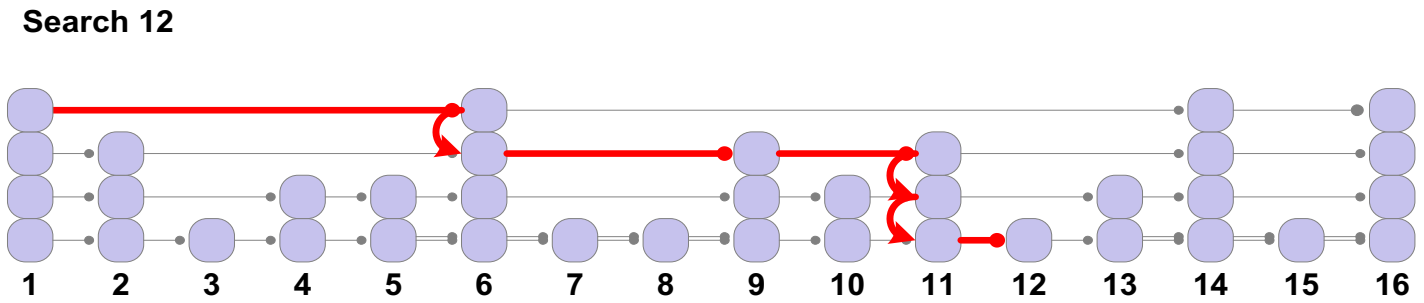
- Structured data
  - Filesystem
    - Ext2(inode), FAT(fat table)…
  - Index structure
    - AVL tree, B tree, hash…

All basically built on the pointers

Introduction

Skip list data structure

- Why skip list ? [1]
  - Keep technical problem of using pointer
    - No needing balance
    - Not tradeoff of space and time
  - Trade reads for writes by adjusting number of pointers
- Skip list
  - Singly linked list
  - Probability decide level of objects
  - Amortized O(logn)



[1]: William Pugh, “Skip lists: a probabilistic alternative to balanced trees,” Communications of the ACM archive, Vol 33, Issue 6, 1990

Introduction

Skip list data structure

- Comparison
  - Red Black Tree
    - Sequential insertions
      - Rotation for balance in RB-tree
    - Randomly insertions and deletions
      - Leaf node fill up the lack of deleted node
  - Skip list
    - All the same because random
  - Hash
    - No range query
    - Space cost

	Sequential order	Random order
Red Black Tree	360274	310968
Skip List (p=0.5)	193306	192360
Skip List (p=0.25)	168108	165408
Hash (linear-list buckets)	131072	82776

Sequential key permutation and random key permutation
for insert and delete 65536 of objects

Introduction

Background and Related Work

- Block device emulation
  - FTL, NFTL [2][3]
    - Translation table in ram for block mapping
- Native File system
  - JFFS1/2/3, YAFFS1/2 [4][5]
    - Take into consideration physical property of flash
    - Translation table in ram for data object
    - Control metadata in ram for system control
- Index structure
  - B-tree and R-tree over FTL [6][7]
    - Pack operations into reservation buffer to reduce influence of small writing
    - Node translation table for disorder info of operations on some node
  - MicroHash [8]
    - Physical pointer, good wear-leveling, good searching time
    - Specific application to sensor devices with temporal data

[2]: Intel Corporation, “Understanding the Flash Translation Layer(FTL) Specification.”

[3]: Linux MTD project and M-System, “NAND Flash memory Translation Layer (NFTL).‘‘

[4]: Linux MTD Project, “Journaling Flash File System (JFFS), Journaling Flash File System 2 (JFFS2), and Journaling Flash File System 2 (JFFS3).‘‘

[5]: Aleph One Company, “Yet Another Flash Filing System, “http://www.yaffs.net/

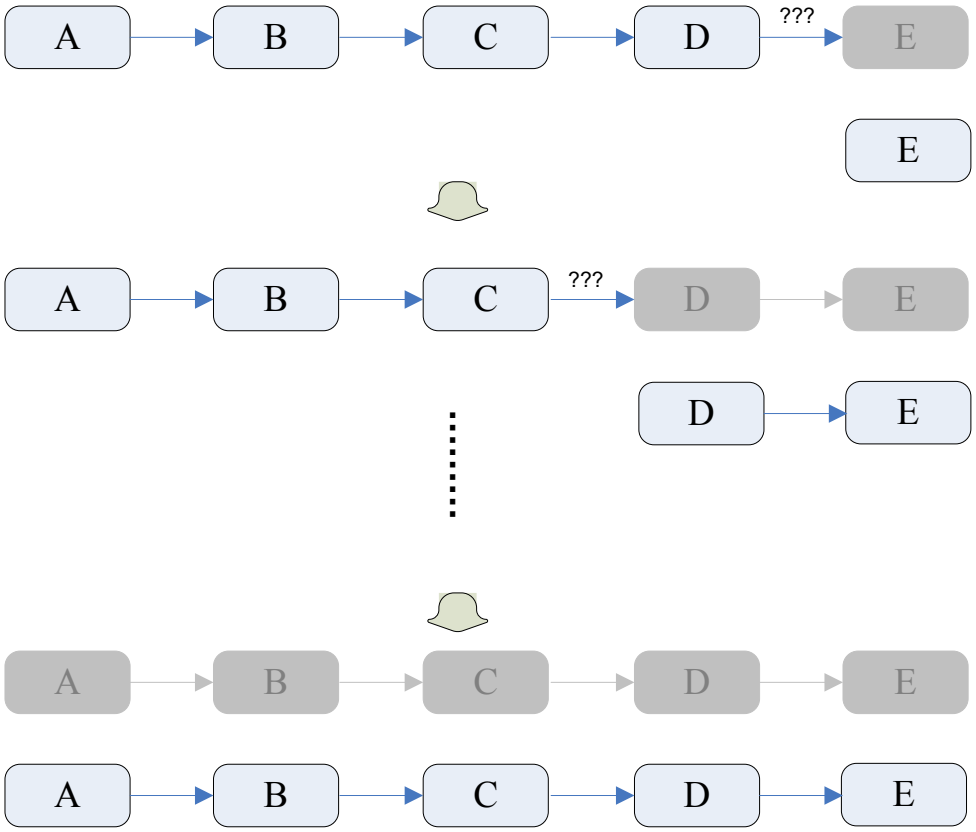
[6]: Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo, "An Efficient B-Tree Layer Implementation for Flash-Memory Storage Systems," Accepted and to appear in ACM Transactions on Embedded Computing Systems.

[7]: Chin-Hsien Wu, Li-Pin Chang, and Tei-Wei Kuo, "An Efficient R-Tree Implementation for Flash-Memory Storage Systems," the ACM GIS conference, 2003

[8]: Zeinalipour-Yazti, D., Lin, S., Kalogeraki, V., Gunopulos, D., and Najjar, W. 2005. Mi-crohash: An efficient index structure for flash-based sensor devices. In 4th USENIX Conference on File and Storage Technologies (FAST). 31 – 44.

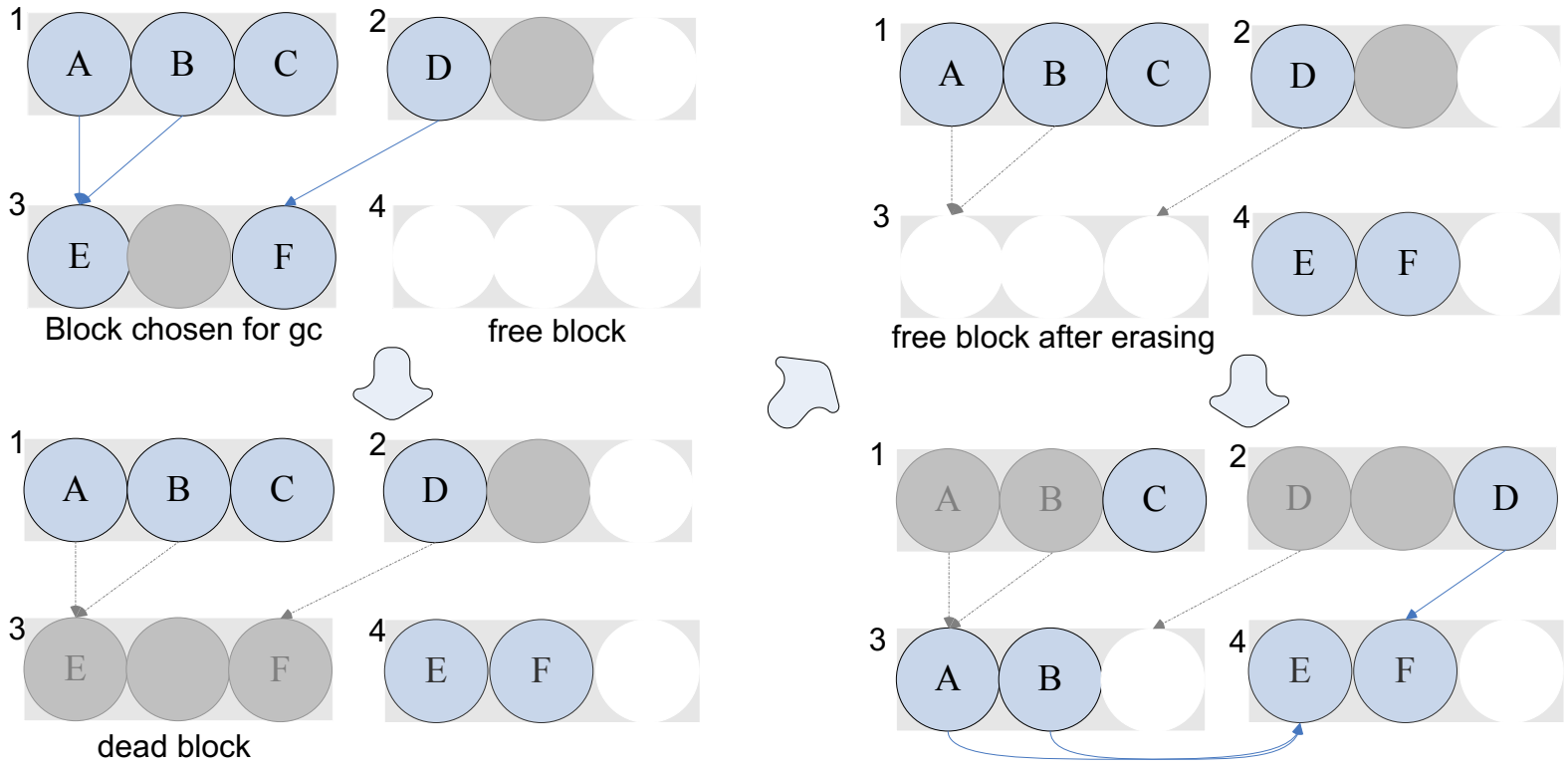
A Skip-List-Based File System over Nor flash:
Fundamental Issue

- Pointer-Update propagation



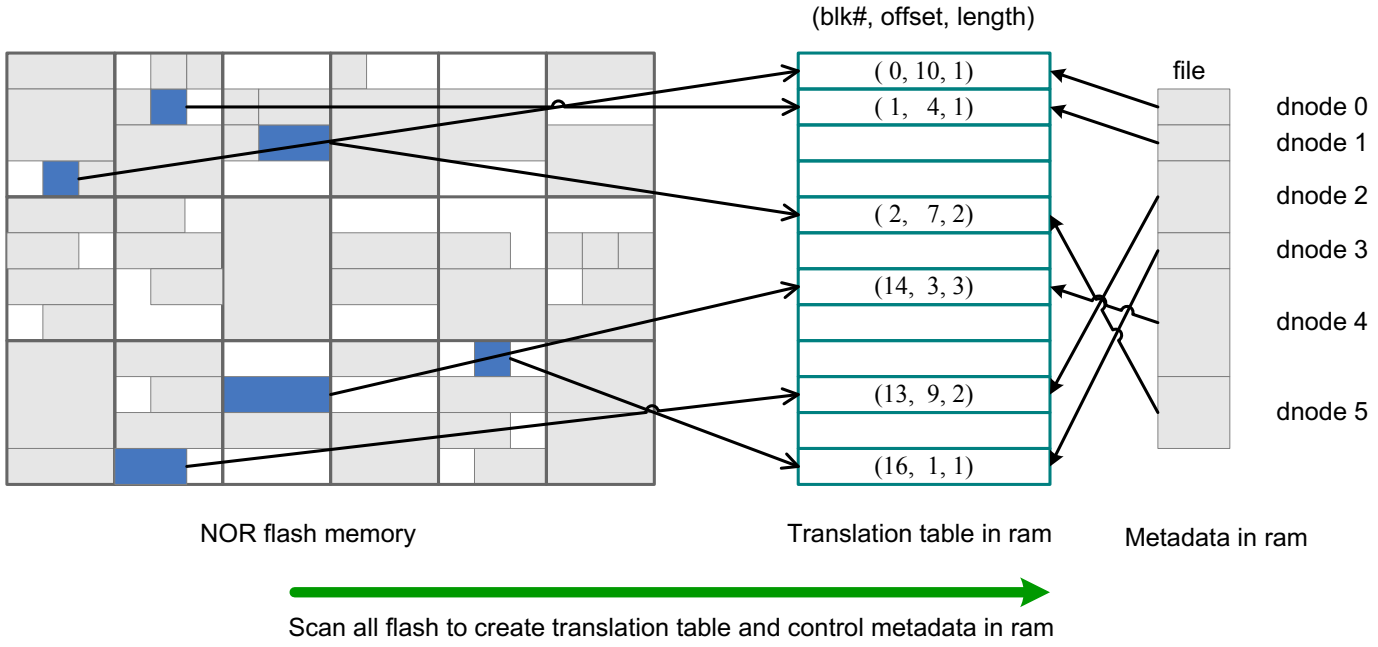
A Skip-List-Based File System over Nor flash:
Fundamental Issue

- Garbage collection Deadlock
  - Less free block remained after gc with physical pointer update



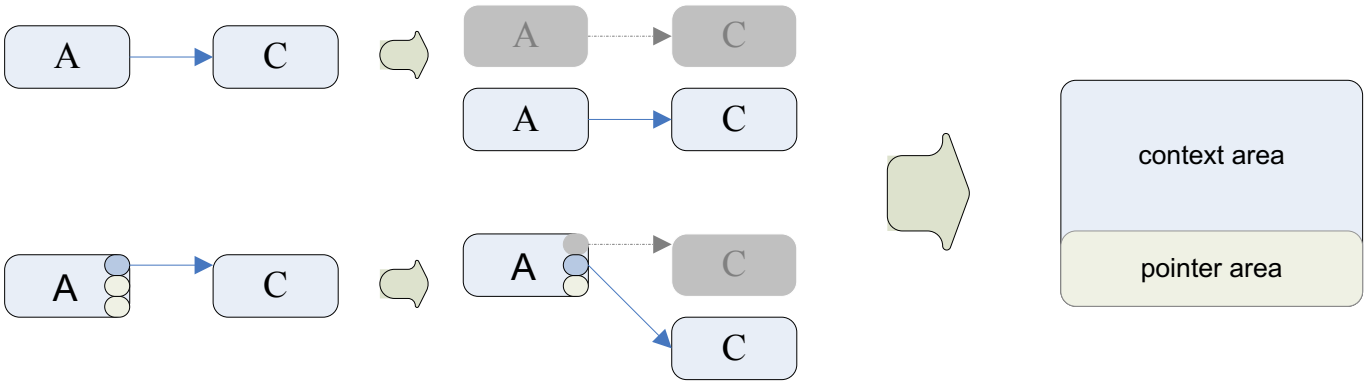
A Skip-List-Based File System over Nor flash:
Fundamental Issue

- Initialization scan
  - Anchored position:
    - Worn out some block quickly
  - Logical pointer
    - Scan on boot time



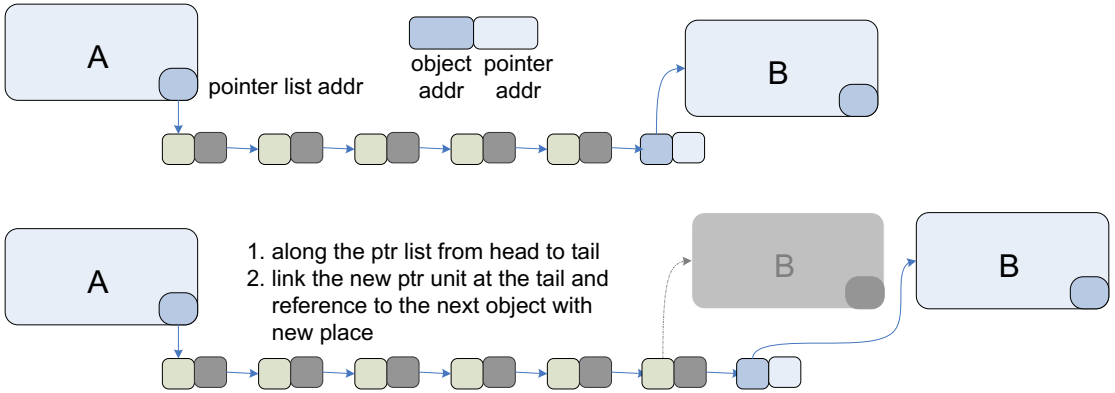
A Skip-List-Based File System over Nor flash: A Skip List Implementation

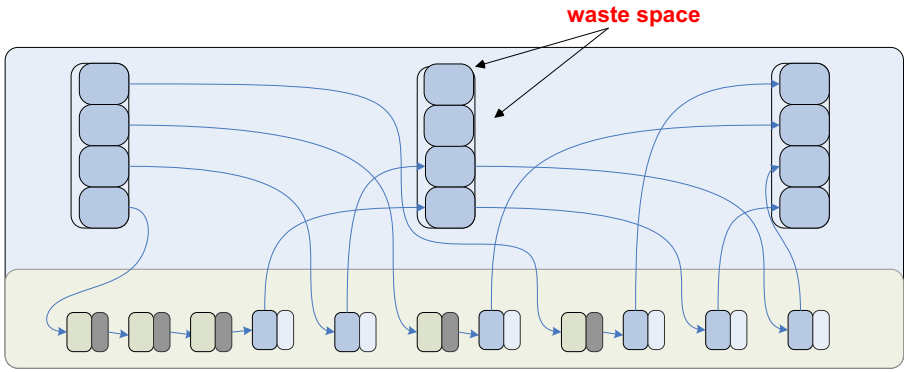
- Physical Pointers and Block Layout
  - Context area
    - store the object data with context
  - Pointer area
    - All object in the same block share the pointer area
    - Ex: delete an object



A Skip-List-Based File System over Nor flash:
A Skip List Implementation

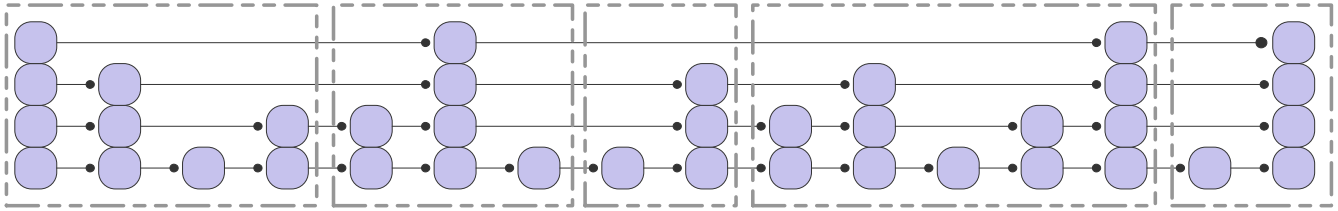
- Physical Pointers and Block Layout
  - Style of pointer update with new layout

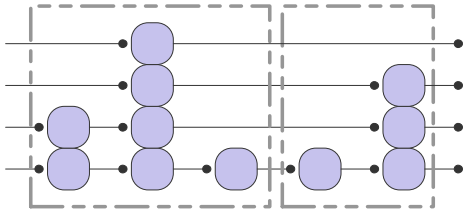




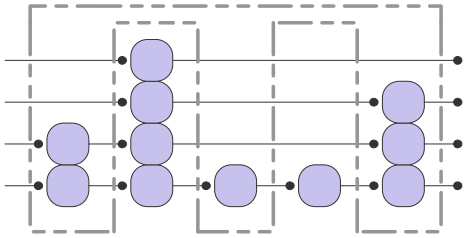
A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Garbage collection and Partition
  - Why partition?
    - Reduce mass in-pointers to number of max level of skip list

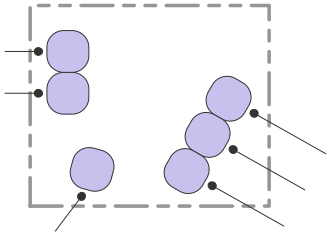




partition



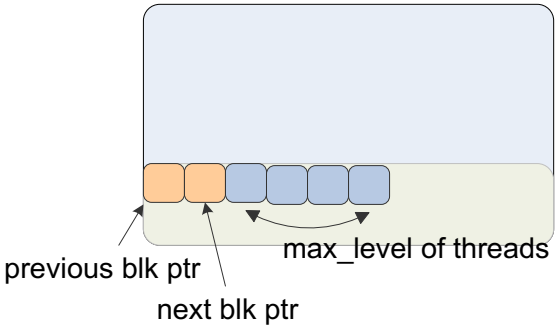
1. One segment in one block
2. sequential nodes and sequential bocks
3. Every object locate in only one block

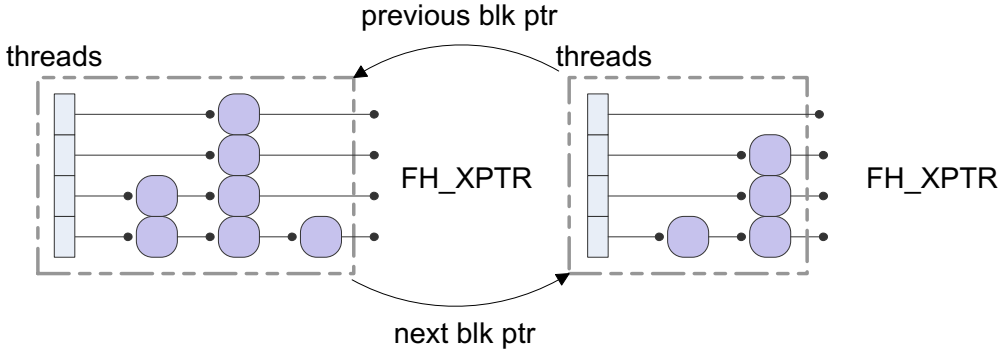


free laying

A Skip-List-Based File System over Nor flash:
A Skip List Implementation

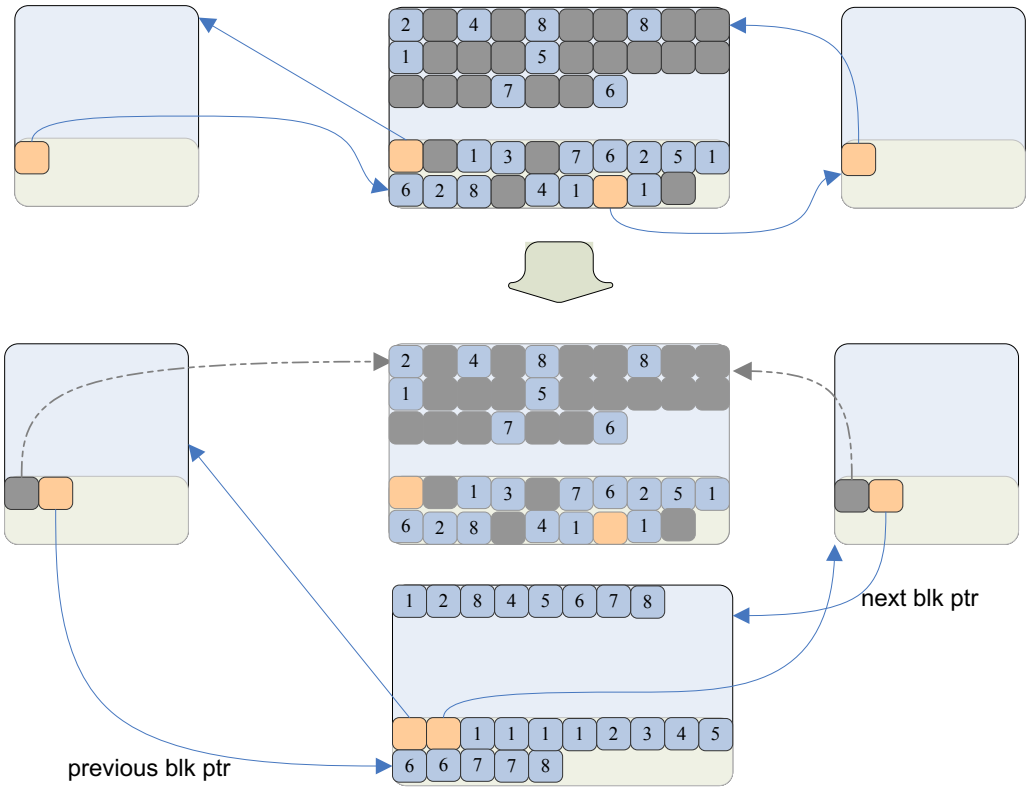
- Garbage collection and Partition
  - Threads
    - Carry on the level of the pointer list searching from previous block's objects
  - Previous and next block list
    - Reduce the number of pointer updates to two pointer
    - The two pointers also share the pointer area





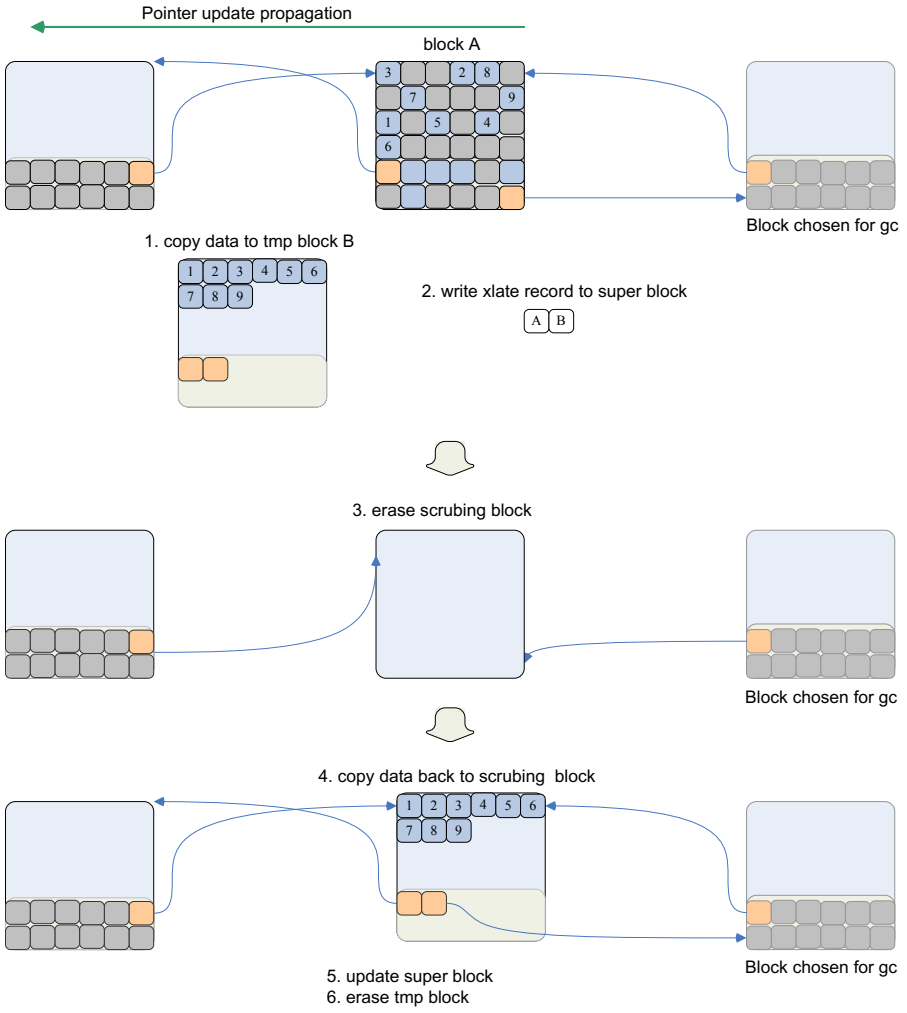
A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Garbage collection
  - One pointer update to previous block and next block



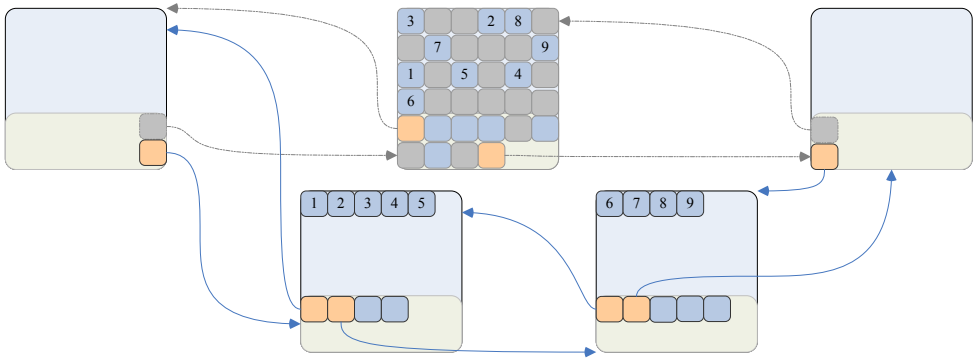
A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Pointer Scrubbing
  - Obstruct pointer propagation and garbage collection deadlock
  - Not frequently involved

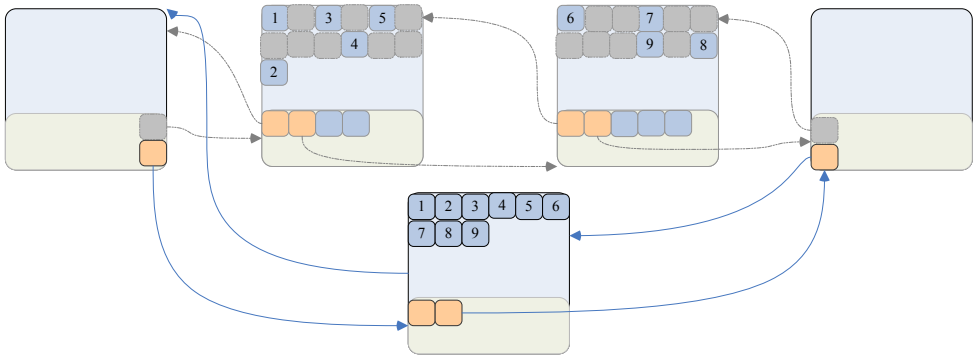


A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Split and Merge action of Partition
  - Objects only locate in one block → Full block and need space → split

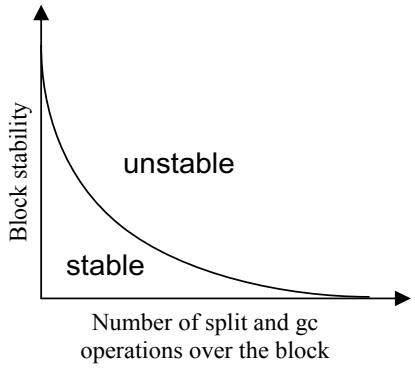


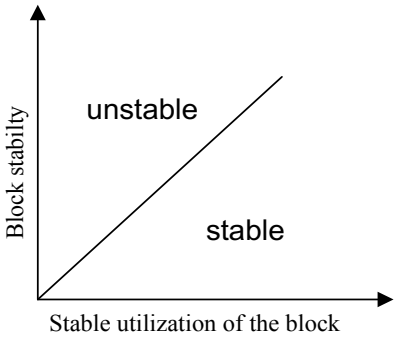
  - Need free block → merge



A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Strategy for Split and merge
  - Hot/cold data control the stability of block
  - Define:
    - Block stability
      - The upper bound of the utilization of block being stable
    - Stable block
      - The utilization of the block is less then block stability of the block
    - Two operations
      - decrement the block stability, when splitting or gc
      - increment the block stability, when meeting aging time





A Skip-List-Based File System over Nor flash:
A Skip List Implementation

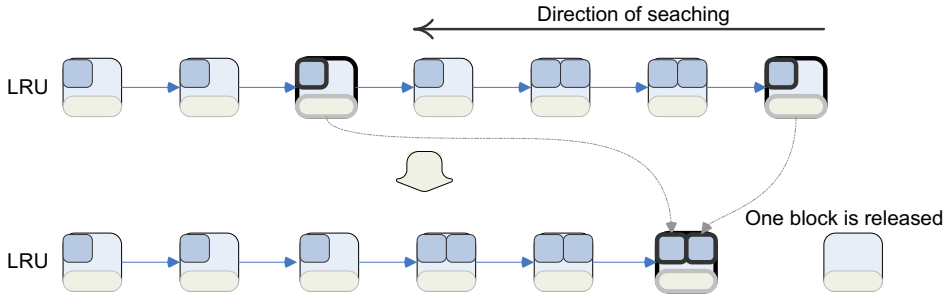
- Split and Merge action of Partition
  - Split policy
    - Idea
      - Split non-stable block
    - Reason
      - More non-stable block → hotter block → more hotter data in the block
    - Algo

```
If (the block is full)
        If (the block is non-stable)
            split the block
            if (split success)
                dec block stability
            else
                flash has no space
        else
            garbage collection on the block
```

A Skip-List-Based File System over Nor flash:
A Skip List Implementation

- Split and Merge action of Partition
  - Merge policy
    - Idea
      - Merge the least recently split and mergible blocks, and be sure the merged block is stable after merging the blocks
    - Reason
      - LRS: 1. merge cold blocks
        2. avoid merging the same block that just has been split
      - Stable after merging: avoid non-stable block after merging
    - Algo

```
if (no free block in system ||
            block stability == 1)
   merge action
   if (merge action success)
      get one free block
   else
      merge failed
```



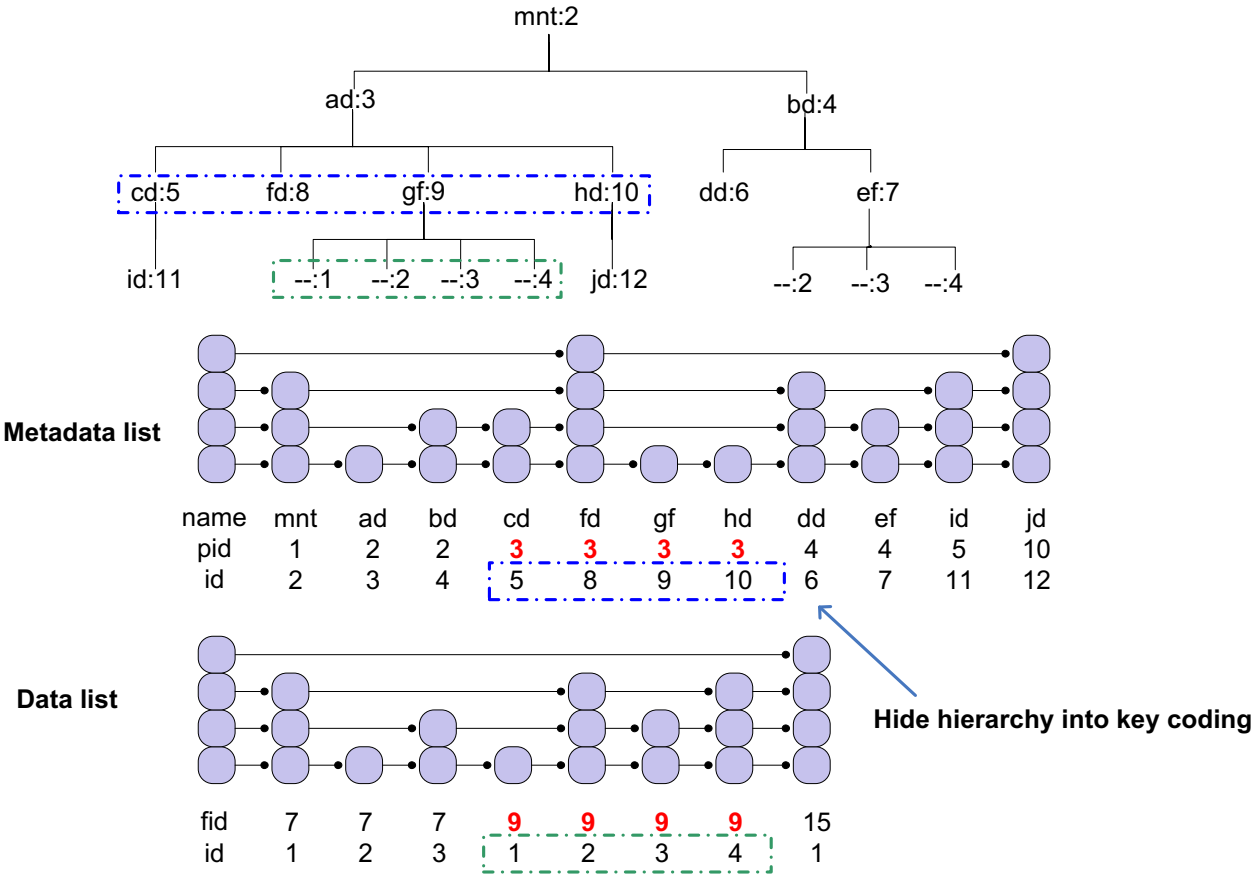
A Skip-List-Based File System over Nor flash:

Metadada objects and Data objects

- Metadata object
  - Maintain hierarchy of filesystem
    - Directory object
      - List head of metadata objects under the directory
    - File object
      - Represent the leaf object in the hierarchy of filesystem
- Data object
  - The instances of real context of File

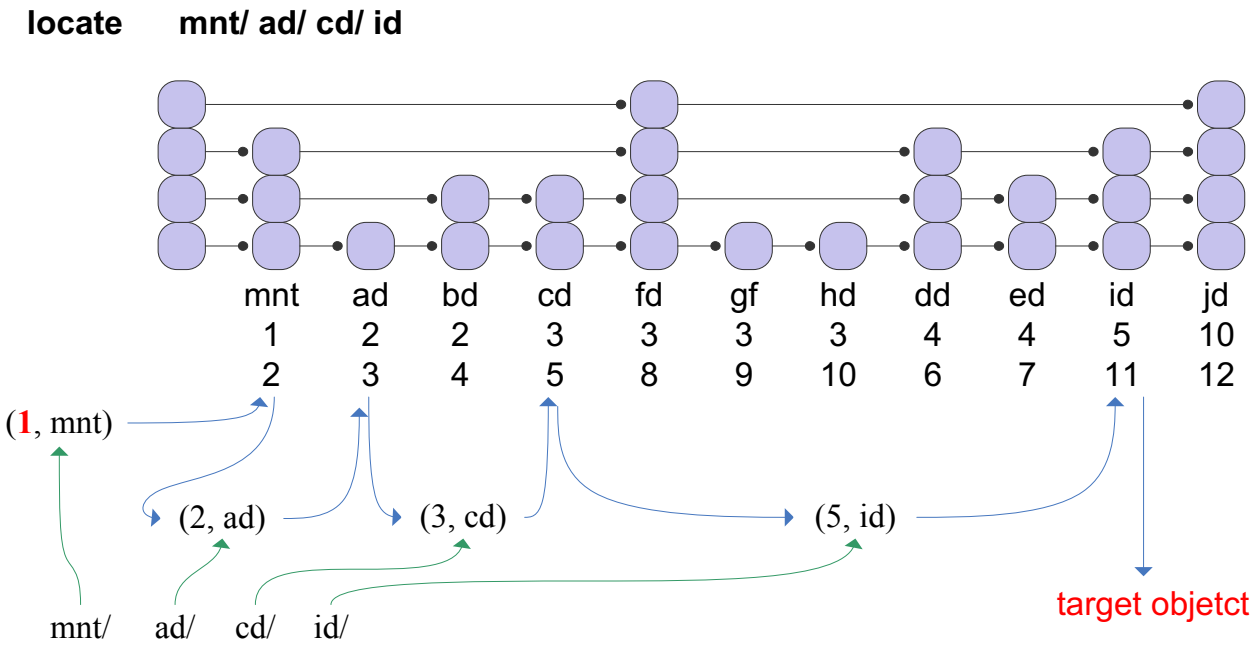
A Skip-List-Based File System over Nor flash:
Key-coding scheme

- Metadata object: (inode# of parent, name) → (pid, name)
- Data object: (inode# of file, chuck num) → (fid, cid)



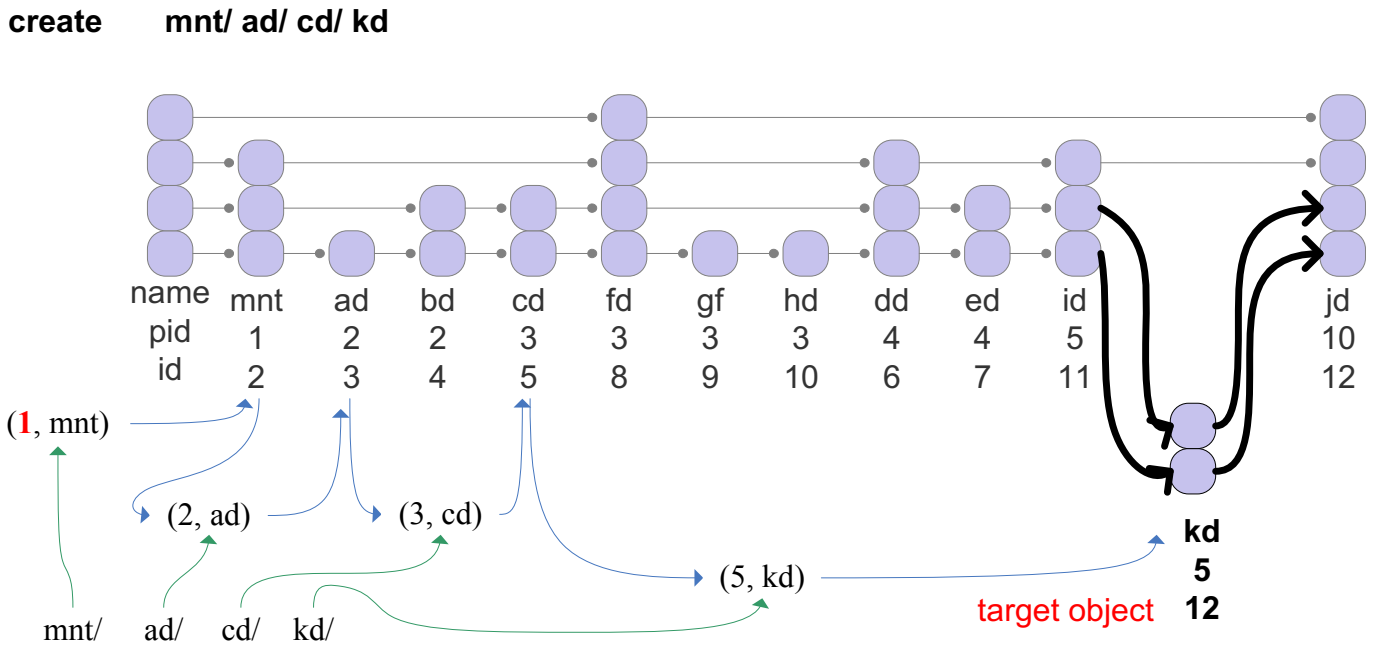
A Skip-List-Based File System over Nor flash:
File and Directory operations

- Locate an object
  - A series of searching of operating system



A Skip-List-Based File System over Nor flash:
File and Directory operations

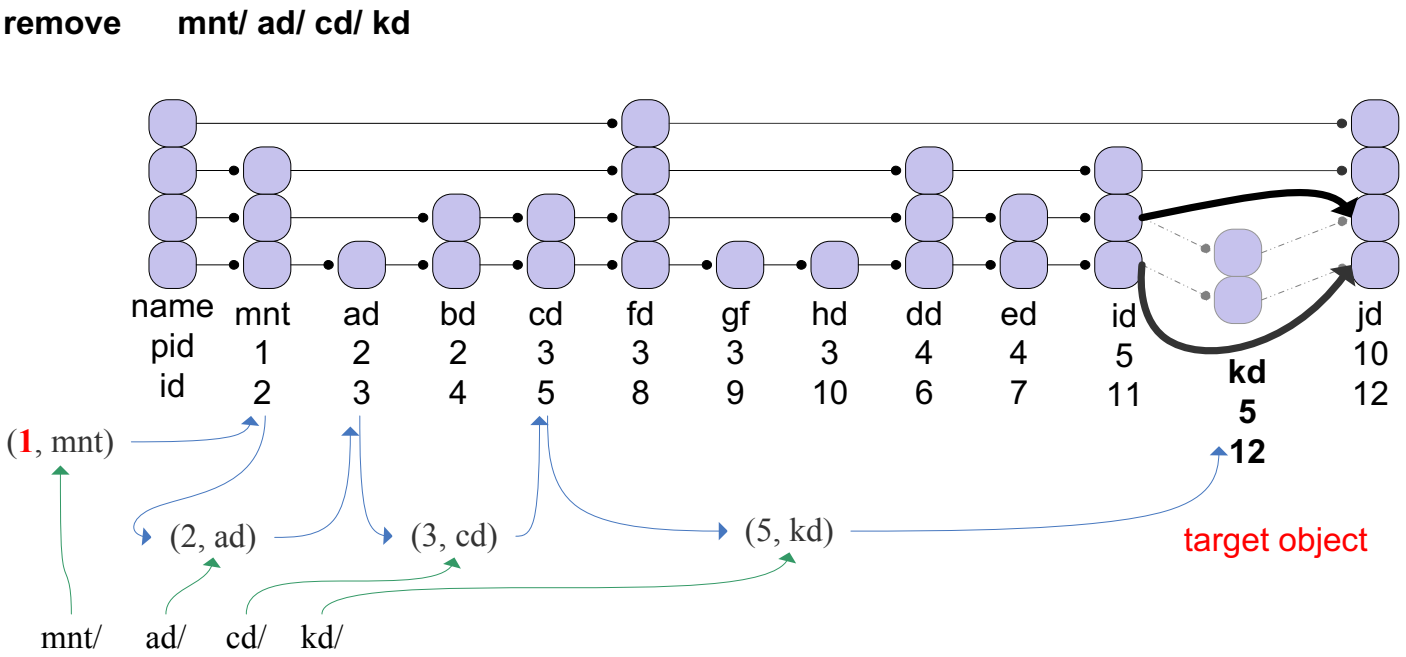
- Create a directory or a file
  - Insert an object



A Skip-List-Based File System over Nor flash:

File and Directory operations

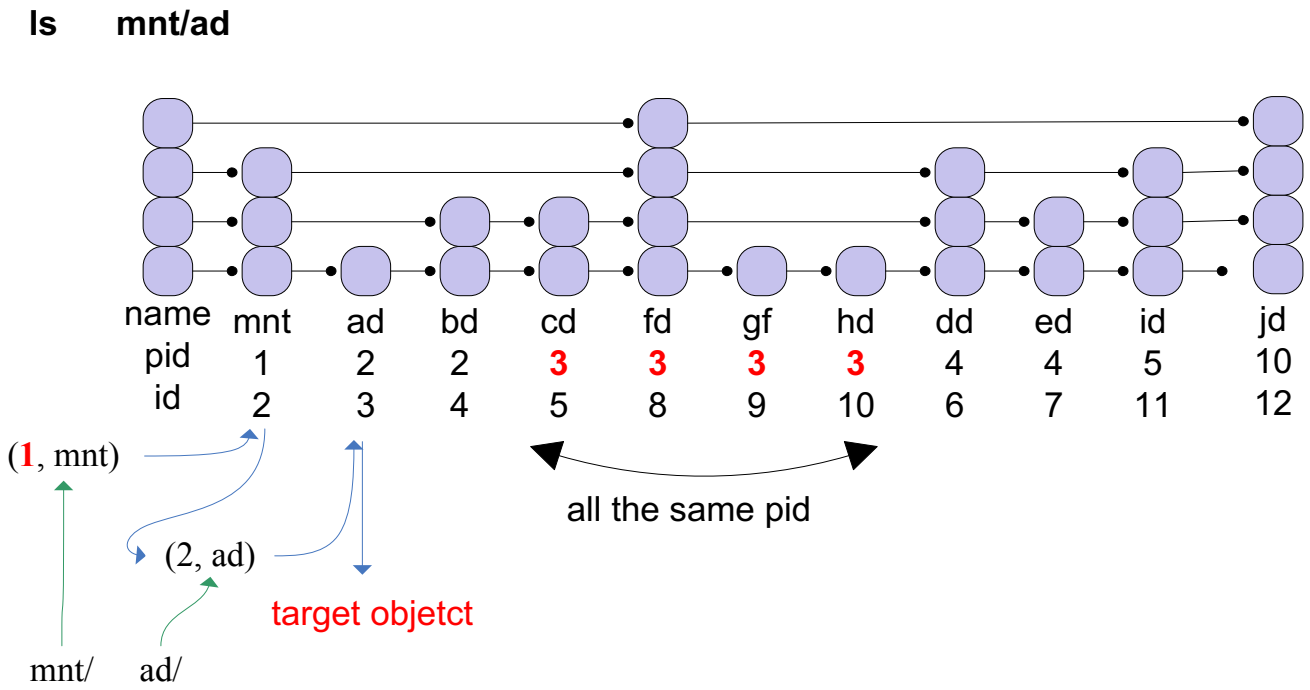
- Remove a directory or a file
  - Delete an object



A Skip-List-Based File System over Nor flash:

File and Directory operations

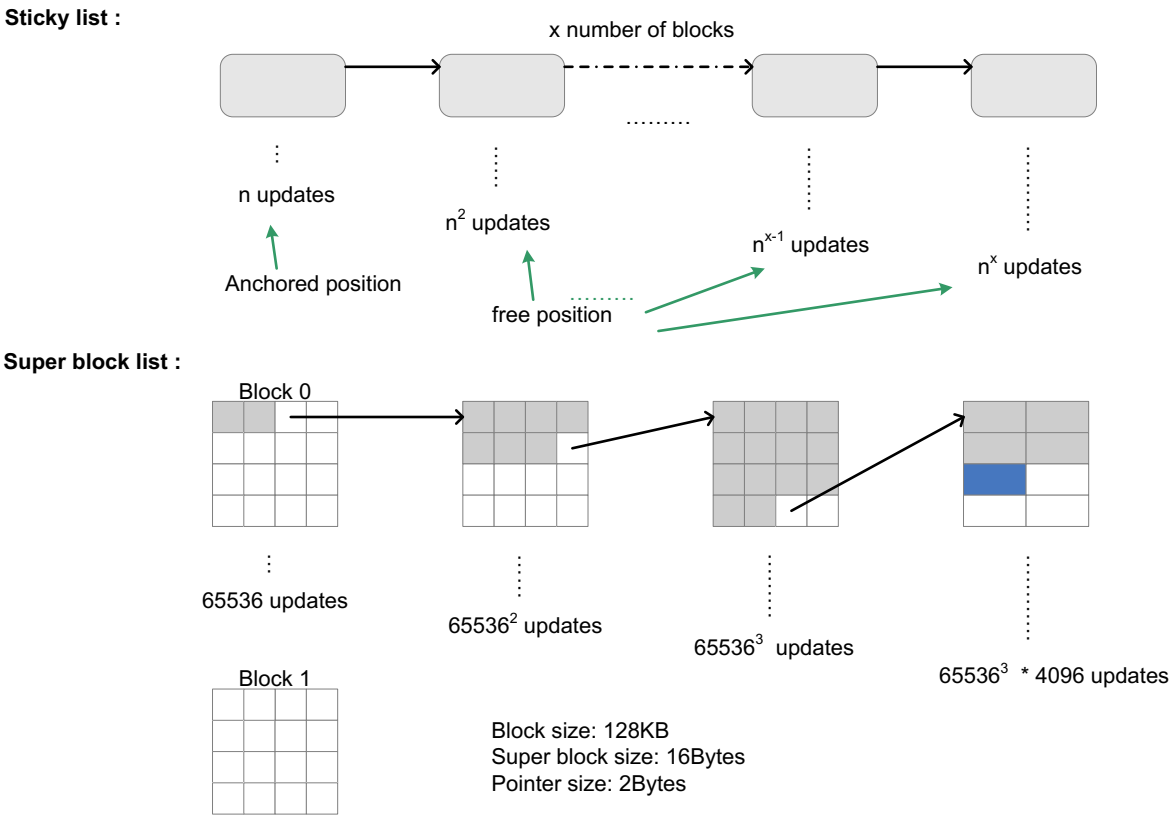
- Enumeration
  - the sub-list of metadata object enumerated



A Skip-List-Based File System over Nor flash:

File and Directory operations

- How super block list works
  - Fast initializing and not wearing out any spatial block



Evaluation

- System read overhead
  - Goal:
    - profile the reading overhead on the system
  - Workload:
    - sequential-key permutation of 20000 objects
  - Metrics:
    - (total read pointers – total read objects) / total read pointers

- Probability v.s. Max level
  - Goal:
    - to know the relationship of read/write operations with probability and max level
  - Workload:
    - (insert) sequential-key permutation of 20000 objects and (update) normal distribution of 10000 objects
      - Normal distribution: mean = 5000, stdev = 500
  - Metrics:
    - calculate and compare time of reading pointers , time of reading objects, and time of writing pointers

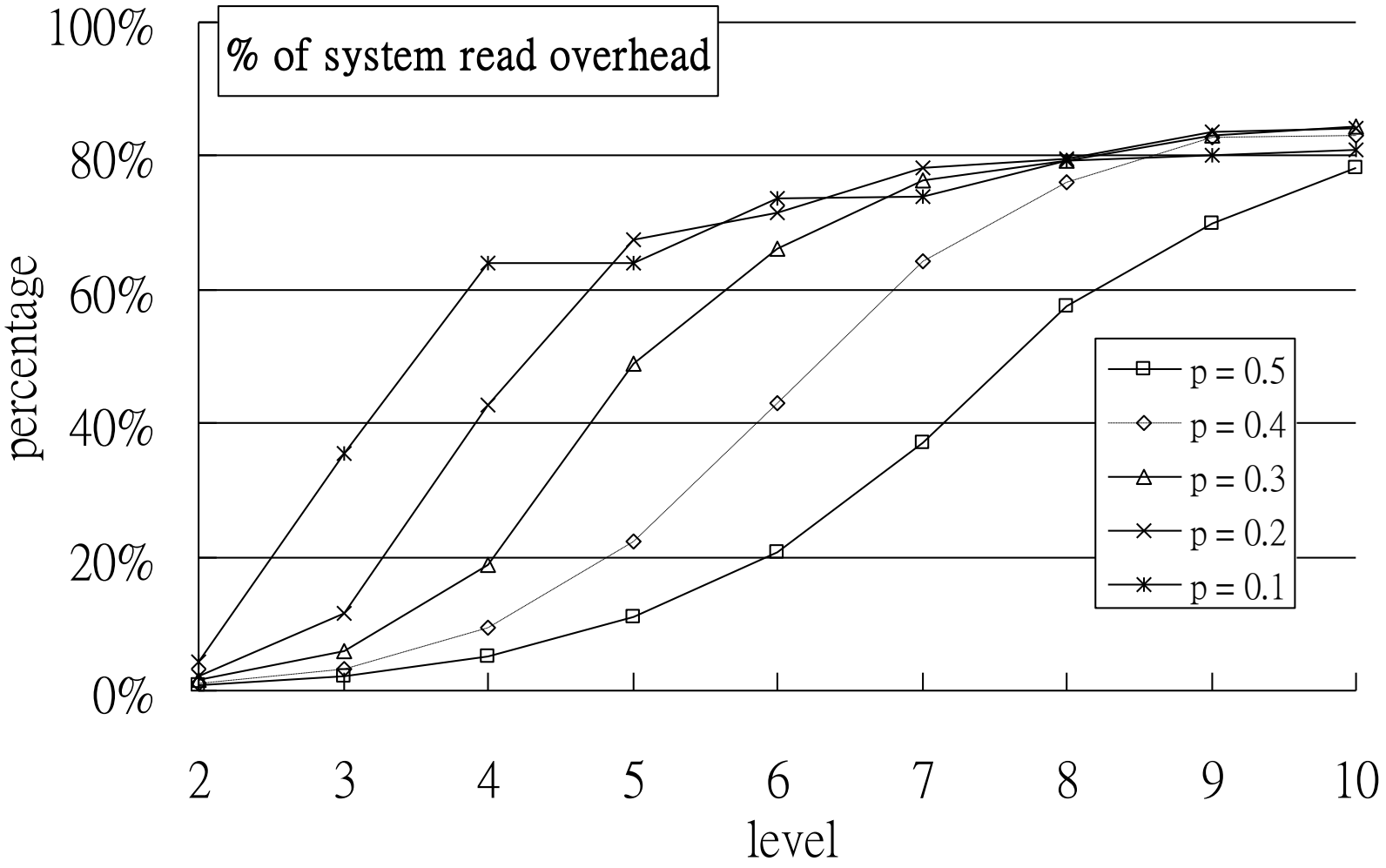
Evaluation

- The rate of pointers and objects
  - Goal:
    - tune the best rate of number of spare pointers and number of objects
  - Workload:
    - (insert) sequential-key permutation of 20000 objects and (update) 5 types of normal distributions of 10000 objects
      - Normal distribution: mean = 5000, stdev = 500, 400, 300, 200, 100
  - Metrics: compare the number of block erasing times

- The work of block stability
  - Goal:
    - test the relationship of block stability and utilization of the block
  - Workload:
    - (insert)sequential-key permutation of 20000 objects and (update) normal distribution of 10000 objects
      - Normal distribution: mean = 5000, stdev = 500
      - aging write time = 10000
  - Metrics: compare the block stability with number of block's objects

Evaluation

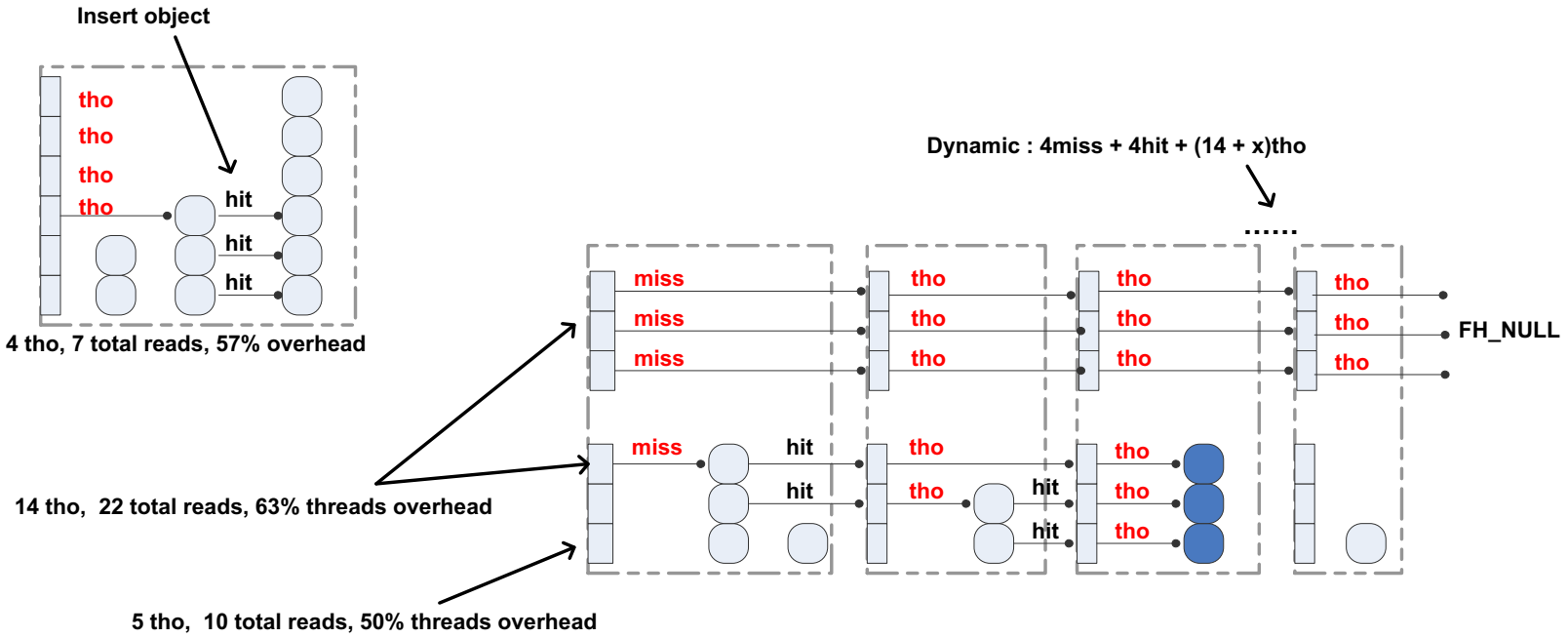
system read overhead



Evaluation

system read overhead

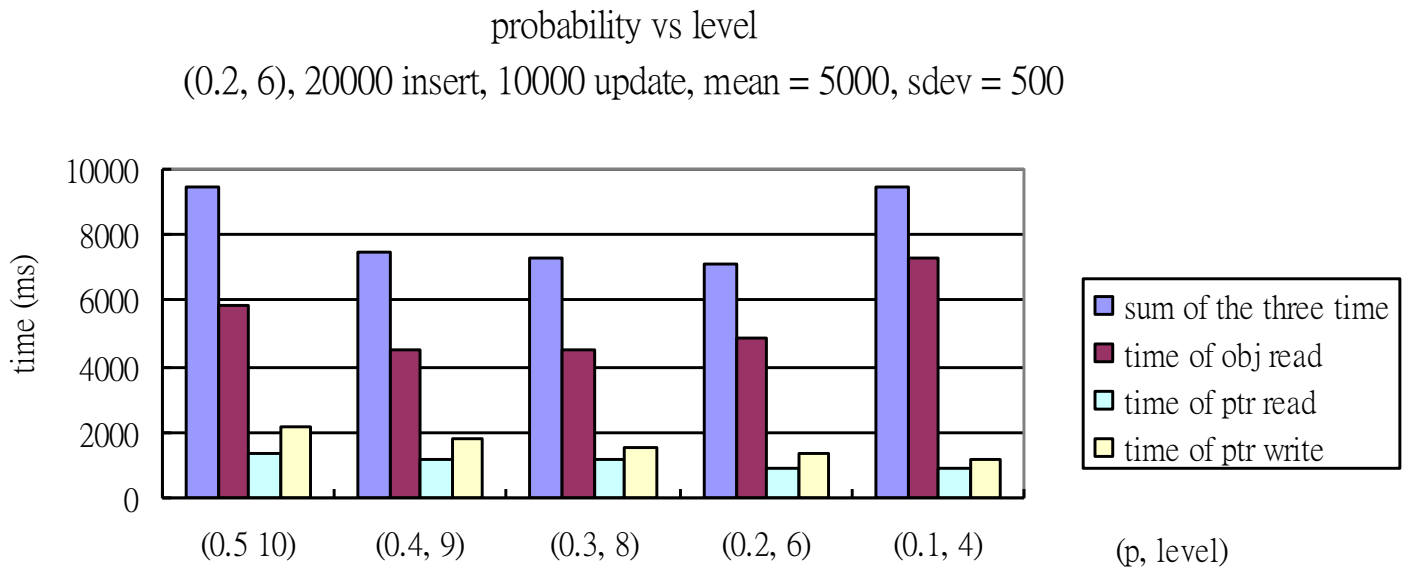
- Thread overhead
  - Too high thread overhead per block
  - higher level → higher threads overhead
  - Threads overhead dynamic the performance



Evaluation

Probability v.s. Max level

- (high prob, high level) → higher density of high level
- (low level, low level) → low density of dominating level
  - One object reading time = 80bytes x 80ns [9][10]
  - One pointer reading time = 4bytes x 80ns
  - One pointer writing time = 4bytes x 9us

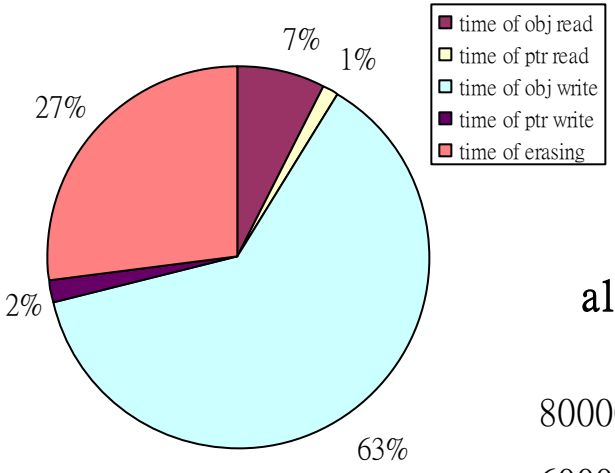


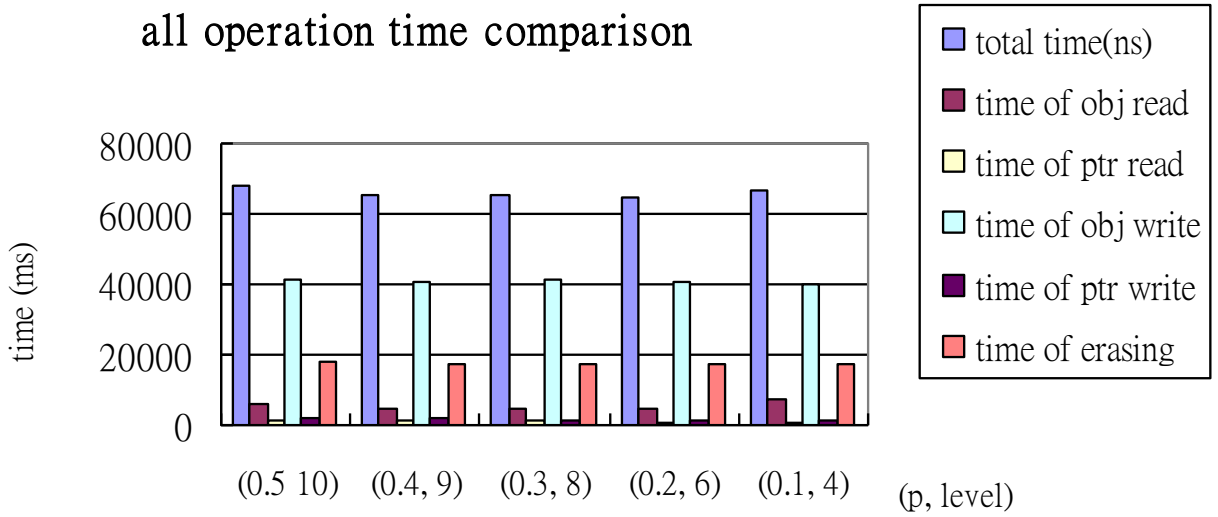
Cite[9]: Samsung Electronics Company, “K5A3x40YT(B)C 32M Bit (4Mx8/2Mx16) Dual Bank NOR Flash Memory datasheet”
Cite[10]: MXIC Electronics Company, “MX-29LV160CBTC-90G 16M-BIT [2Mx8/1Mx16] CMOS SINGLE VOLTAGE 3V ONLY FLASH MEMORY datasheet”

Evaluation

The rate of pointers and objects

- Erase time and object write time dominate the performance

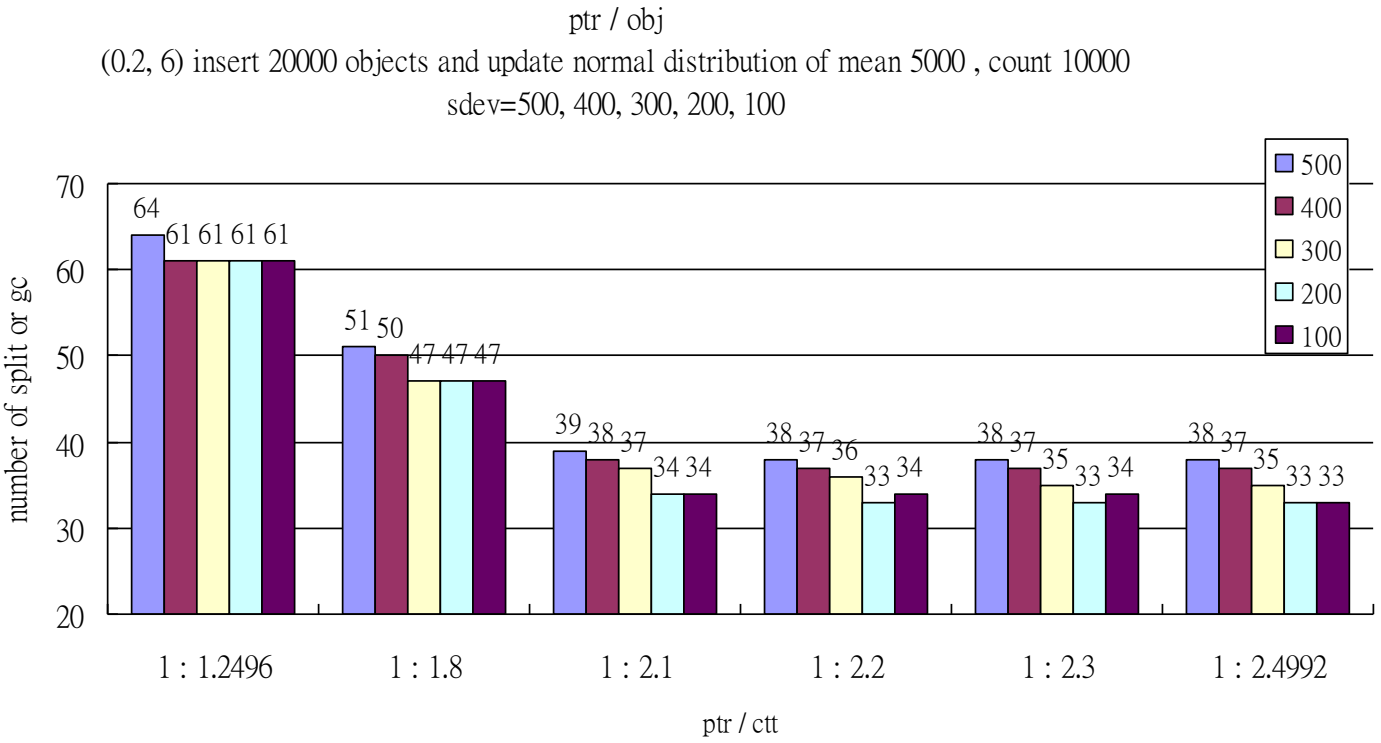




Evaluation

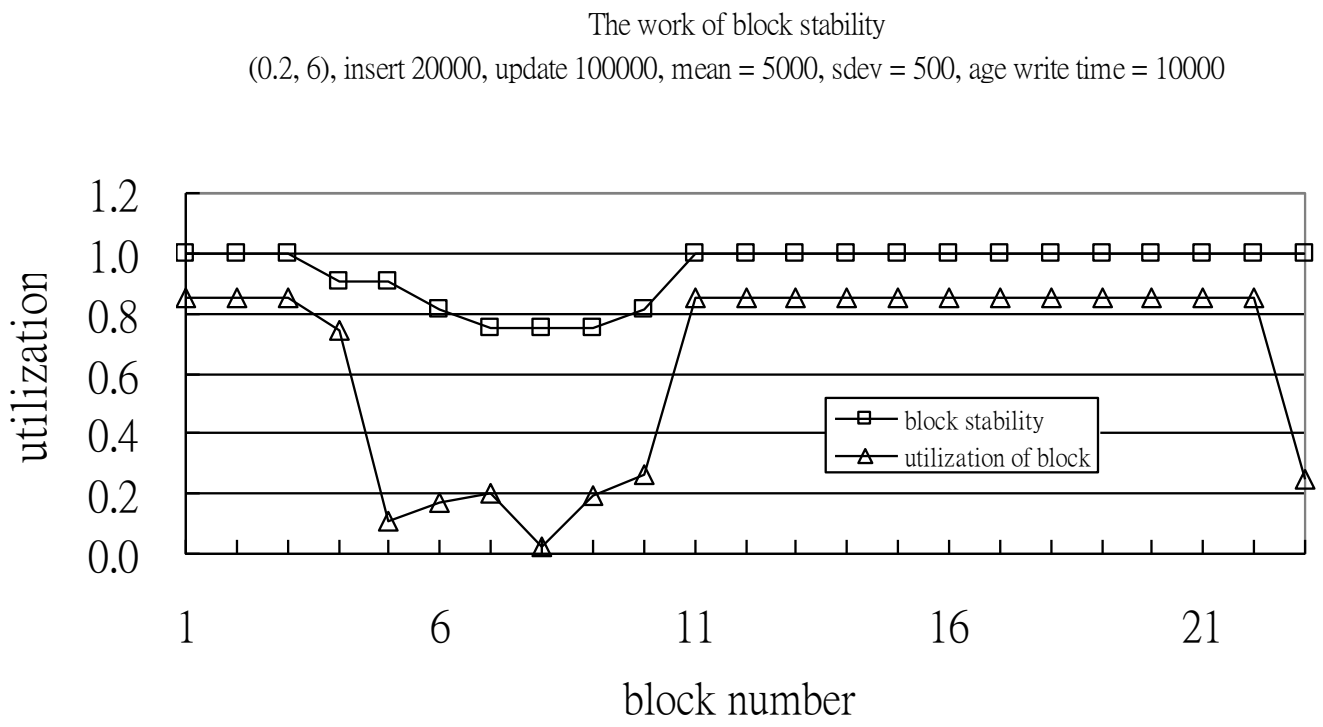
The rate of pointers and objects

- Low stdev→focus of update→more space→less gc+split→ less block erasing
- Max rate of #pointer over #object = $2*(1+\sum_{x=1}^{ML-1} P)$



Evaluation

The work of block stability

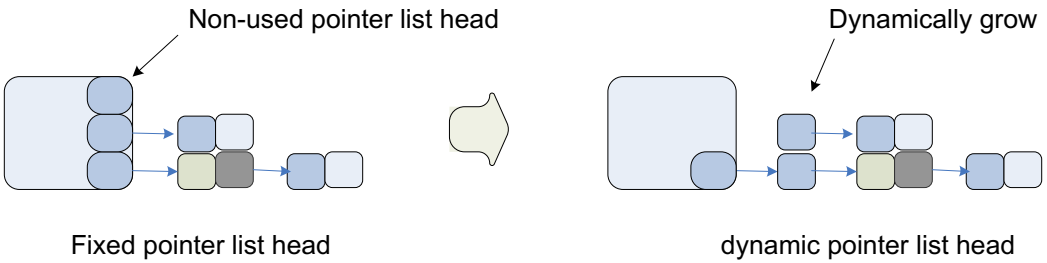


Conclusion

- Flash is more and more general equipment in embedded systems
- Traditional management of flash use translation layer to decouple updates of objects
  - Scan on boot time
  - Large ram cost
- Skip list on Nor-Flash by physical pointers
  - Issue and resolution
    - Pointer update propagation → ctt/ptr area and partition
    - Gc deadlock → pointer scrub
    - Initialization scan→ super block list (sticky list)
  - pros
    - Decouple the drawbacks of method of logical address
  - cons
    - Design of threads oppose effective access

Future work

- Design layout can be more smart
  - Dynamic max level of pointer list head in object



  - Evaluate threads with method of null object or letting native skip list through blocks with Max-level update overhead
  - Dynamic rate of #pointer over #objects

- Complete filesystem implementation
  - compare read/write-file operations with jffs2
  - compare skip list with RB-tree of jffs2

Reference

- [1] L. P. Chang, "On efficient wear leveling for large-scale Fash-memory storagesystems ," the ACM symposium on Applied computing, March, 2007.
- [2] G. M. Adelson-Velskii and E. M. Landis. "An algorithm for the organization of information," Soviet Math. Doclady 3, pages 1259{1263, 1962.
- [3] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Fash-memory based FileSystem," Proceedings of the USENIX Technical Conference, 1995.
- [4] C. Reummler and J, Wilkes, "UNIX disk access patterns," Proceedings of USENIX Technical Conference, 1993.
- [5] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J.A. Tauber, "Storage Alternatives for Mobile Computers," Proceedings of the USENIX Operating System Design and Implementation, 1994.
- [6] "K9F2808U0B 16Mb*8 NAND ◦ash-memory Data Sheet," Samsung Electronics Company.
- [7] M. Wu, and W. Zwaenepoel, "eNVy: A Non-Volatile, Main Memory Storage System," Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems, 1994.
- [8] D. Woodhouse, Red Hat, Inc. "JFFS: The Journalling Flash File System".
- [9] Aleph One Company, "Yet Another Flash Filing System".
- [10] Intel Corporation, "Understanding the Flash Translation Layer(FTL) Specification".
- [11] Compact Flash Association, "*CompactF lashTM* 1.4 Speci¯cation," 1998.
- [12] M-Systems, Flash-memory Translation Layer for NAND Fash (NFTL).
- [13] A. Inoue and D. Wong, "NAND Flash Applications Design Guide," http://www.semicon.toshiba.co.jp/eng/prd/memory/doc/pdf/nand applicationguide e.pdf, April, 2003.
- [14] "Wear Leveling in Single Level Cell NAND Flash Memories," STMicroelectronics Application Note (AN1822), 2006.
- [15] D. Woodhouse, \JFFS: The Journaling Flash File System,"
- [16] MC9S12UF32 System on a Chip Guide V01.04,"Motorola,inc.,2002
- [17] USB Thumb Drive reference design DRM061,"Freescale Semiconductor,September, 2004.

Reference

- [18] A. Kawaguchi, S. Nishioka, and H. Motoda, "A Fash-memory based FileSystem," Proceedings of the USENIX Technical Conference, 1995.
- [19] C. Reummler and J, Wilkes, "UNIX disk access patterns," Proceedings of USENIX Technical Conference, 1993.
- [20] W. Vogels, "File system usage in Windows NT 4.0," Proceedings Of the 17-th Acm Symposium On Operating Systems Principles, 1999
- [21] F. Douglis, R. Caceres, F. Kaashoek, K. Li, B. Marsh, and J.A. Tauber, Storage Alternatives for Mobile Computers," Proceedings of the USENIX Operating System Design and Implementation, 1994.
- [22] Samsung Electronics Company, "K9NBG08U5M 4Gb * 8 Bit NAND Flash Memory Data Sheet".
- [23] Samsung Electronics Company, "K9GAG08U0M 2G * 8 Bit NAND Flash Memory Data Sheet (Preliminary)".
- [24] H. J. Kim and S. G. Lee, "A New Flash-Memory Management for Flash Storage System," Proceedings of the Computer Software and Applications Conference, 1999. An Effective Flash Memory Manager for Reliable Flash Memory Space Management," IEICE Transactions on Information and System, Vol. E85-D, No. 6, 2002.
- [25] L. P. Chang, T. W. Kuo, "Real-time Garbage Collection for Flash-Memory Storage System in Embedded Systems," ACM Transaction on Embedded Computing Systems, Vol 3, No. 4, 2004.
- [26] L. P. Chang, and T. W. Kuo, An Adaptive Striping Architecture for Flash Memory Storage Systems of Embedded Systems," Proceedings of The 8th IEEE Real-Time and Embedded Technology and Applications Symposium, 2002.
- [27] L. P. Chang and T. W. Kuo, "An efficient management scheme for large- scale ◦ash-memory storage systems," Proceedings of the ACM Symposium on Applied Computing, 2004; Efficient Management for Large-Scale Flash- Memory Storage Systems with Resource Conservation", ACM Transactions on Storage, Vol. 1, Issue 4, 2005.
- [28] J. W. Hsieh, T. W. Kuo, and L. P. Chang "Efficient Identification of Hot Data for Flash Memory Storage Systems," ACM Transactions on Storage, Volume 2, Issue 1, 2006.
- [29] M. Wu, and W. Zwaenepoel, "eNVy: A Non-Volatile, Main Memory Storage System," Proceedings of the 6th International Conference on Architectural Support for Programming Languages and Operating Systems, 1994.
- [30] D. Woodhouse, \JFFS: The Journalling Flash File System," Proceedings of Ottawa Linux Symposium, 2001.
- [31] C. Manning and Wookey, "\YAFFS Specification," Aleph One Limited, http://www.aleph1.co.uk/node/37, Dec, 2001.
- [32] SSFDC Forum, "*SmartMediaTM* Speci¯cation", 1999.
- [33] M-Systems, "Flash-memory Translation Layer for NAND ◦ash (NFTL)"
- [34] M-Systems, "TrufFFSr Wear-Leveling Mechanism," Technical Note TNDOC-017.

Reference

- [35] R, G. Seidel and C. R. Aragon, "Randomized Search Trees," Algorithmica, 16:464-497 (1996).
- [36] A. Inoue and D.Wong, "NAND Flash Applications Design Guide," Toshiba America Electronic Components (TAEC) White Papers, April, 2003.
- [37] M. L. Chiang, Paul C. H. Lee, and R. C. Chang, "\Using Data Clustering To Improve Cleaning Performance For Flash Memory," Software – Practice and Experience, Vol. 29, No. 3, 1999.
- [38] SanDisk Corporation, "Sandisk Flash Memory Cards Wear Leveling, " http://www.sandisk.com/Assets/File/OEM/WhitePapersAndBrochures/RS- MMC/WPaperWearLevelv1.0.pdf, 2003.
- [39] T. Gleixner, F. Haverkamp, and A. Bityutskiy, "UBI - Unsorted Block Images," http://www.linux-mtd.infradead.org/doc/ubi.html, 2006.
- [40] E. Gal and S. Toledo, \Algorithms And Data Structures For Flash Memories," ACM Computing Surveys, Vol. 37, Issue 2, 2005.
- [41] Freescale Semiconductor, "RDHCS12UF32TD : USB Thumb Drive Reference Design"
- [42] "Wear Leveling in Single Level Cell NAND Flash Memories," STMicroelec-
- tronics Application Note (AN1822), 2006.
- [43] K. Kim and J. Choi, "Future Outlook of NAND Flash Technology for 40nm Node and Beyond," The 21-st IEEE Non-Volatile Semiconductor Memory Workshop, 2006.
- [44] J. Kim, J. M. Kim, S. H. Noh, S. L. Min, and Y. Cho, "A Space-Efficient Flash Translation Layer for CompactFash Systems," IEEE Transactions on Concumer Electronics, Vol. 48, No. 2, 2002.
- [45] P. Cappelletti, C. Golla, P. Olivo, and E. Zanoni, \Flash Memories" Kluwer Academic Publishers, 2001.
- [46] L. P. Chang, "On Efficient Wear-Leveling for Large-Scale Flash-Memory Storage Systems," Proceedings of the 22st ACM Symposium on Applied Computing, 2007.
- [47] Samsung Electronics Company, "NAND Flash-based Solid State Disk Data Sheet," http://www.samsung.com/ Products/ Semiconductor/ FlashSSD/ download/ Standard type.pdf.

Evaluation

- System overhead
  - One pointer writing overhead : $\sum_{x=1}^{ML-1} P^x$ per object, threads per block
  - Pointer reading overhead :