

On Efficient Wear-Leveling for Large-Scale Flash-Memory Storage Systems

Prof. Li-Pin Chang 張立平

National Chiao-Tung University, Taiwan

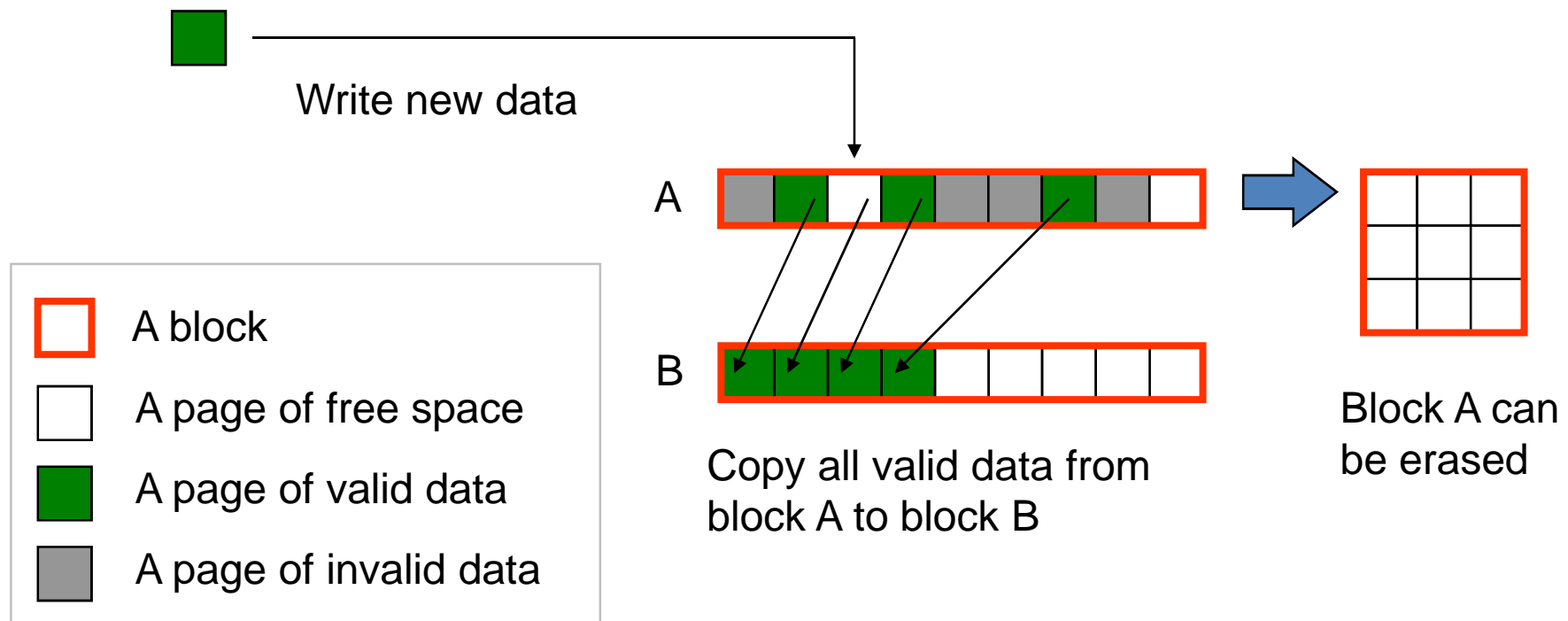
ES² Lab <http://esslab.tw>

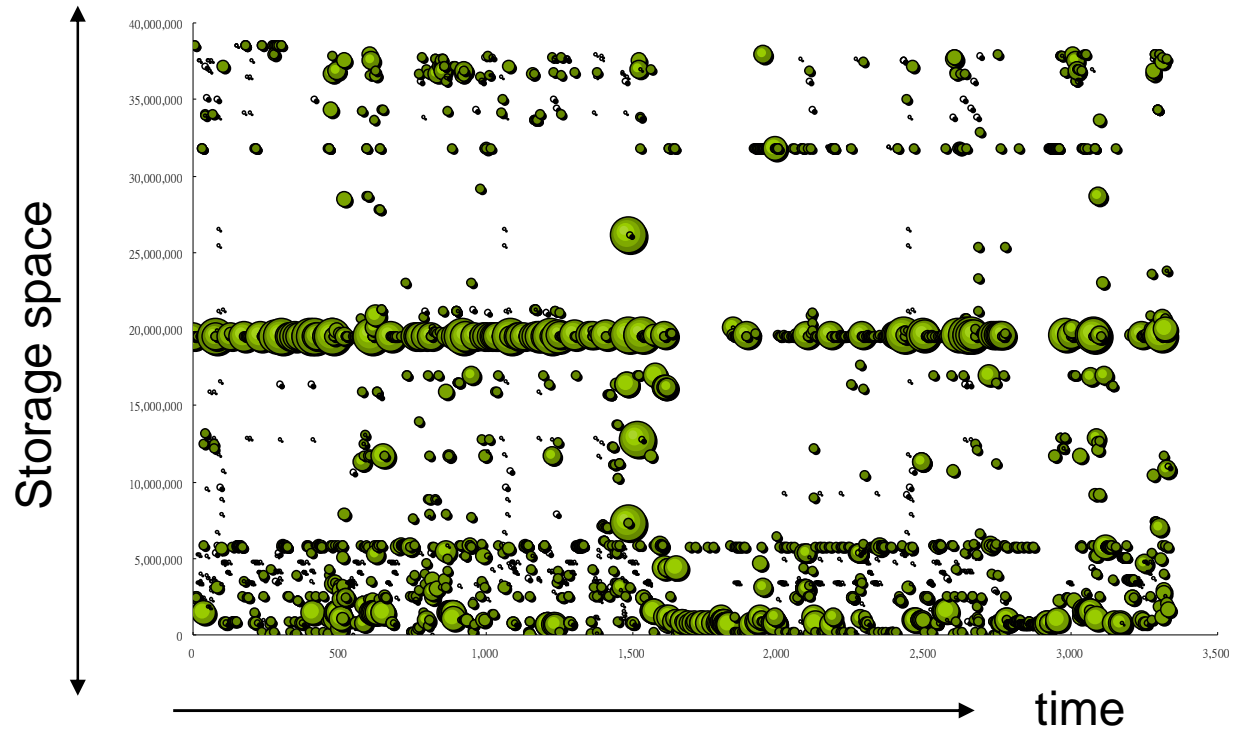
Introduction

- Flash memory is now a crucial component in the design of computer systems
 - It is widely deployed in embedded systems and even ordinary computers
- The management of flash memory is very different from that of disks because of
 - the need of garbage collection
 - its limited endurance

Introduction

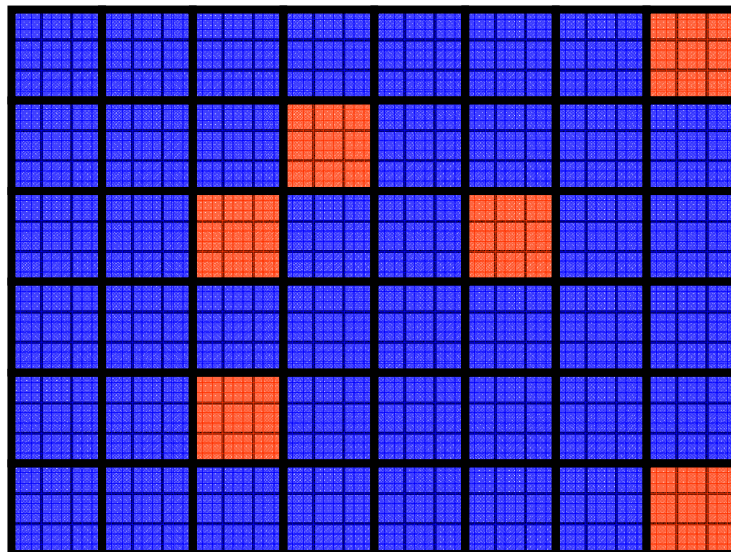
- Garbage collection
 - Free space can be written only once
 - Free space is reclaimed by garbage collection





Realistic workloads
access storage with
spatial locality

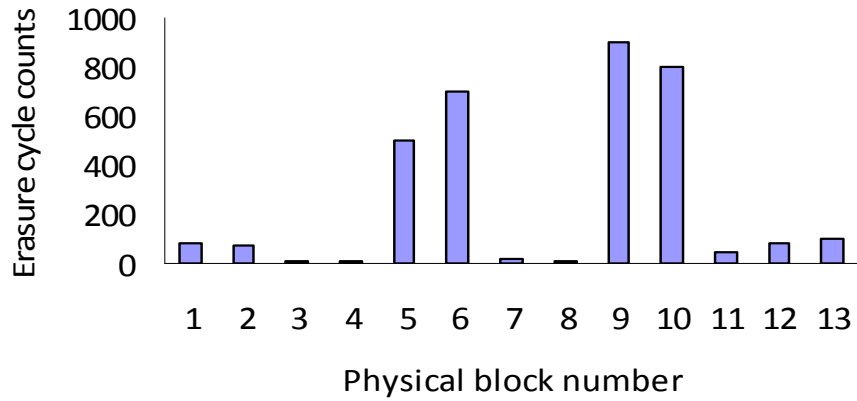
← Hot data



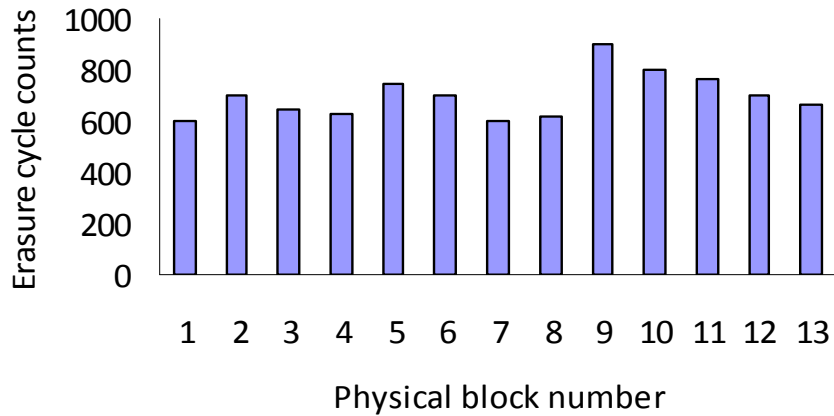
GC prefers blocks of a lot invalid data left
by hot data

GC does not prefer blocks of a lot of valid
and cold data

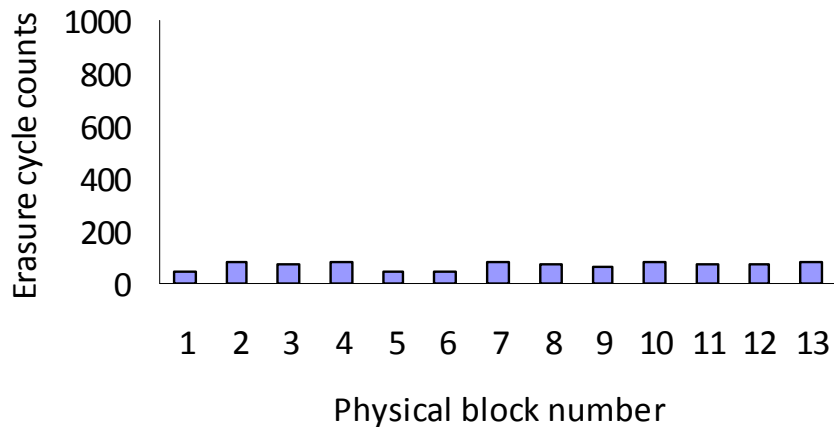
**Each individual block endures only 100k of
erasure cycles!**



(a) No wear leveling or wear leveling has no effect



(b) Wear-leveling activities introduce too much traffic



(c) The desired result. The overall lifetime is prolonged.

There may also be (a)+(b)...

Introduction

- The interests of wear leveling and garbage collection are **at odds with each other**
- Wear leveling intends to manipulate the preference of garbage collection
 - To stop wearing old blocks
 - To start wearing young blocks

Motivation

- What vendors say about wear leveling
 - Consider that 1GB flash is used and a 4KB file is updated every 5 seconds
 - The lifetime is

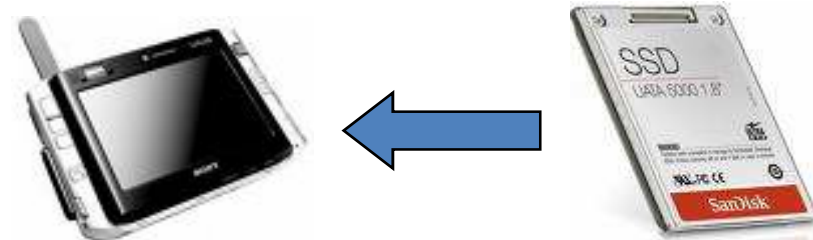
$$2,000,000 \times \frac{4000}{4} \times \frac{1}{1/5}$$

= 317 years



Motivation

- Consider the variety of applications
 - The considered applications are of **weak spatial locality** in accessing data
 - Digital cameras, portable media players, etc.
 - New-generation applications expose flash memory to strong spatial locality!!
 - Hard drives in **UMPC** have been replaced with **Solid-State Disks**



Motivation

- Consider new flash-memory technologies
 - Multi-level cell (**MLC**) flash is less durable
 - Block endurance is roughly degraded by an order of magnitude (100k→10k erasure cycles)
- Consider **scalability**
 - Gigabyte flash memory is very cheap now!
 - Management overheads of RAM space and/or CPU time



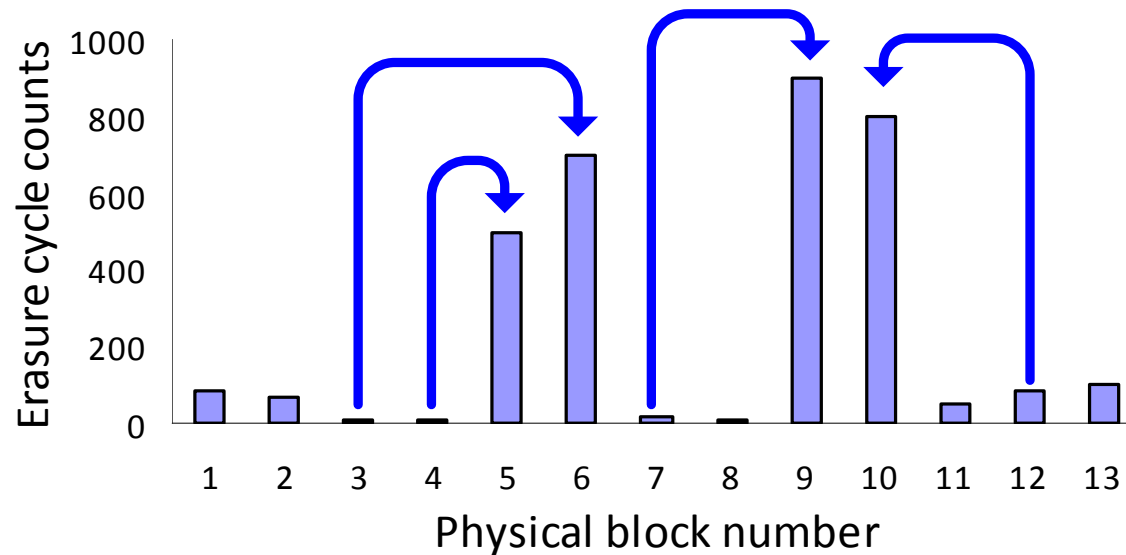
Motivation

- Existing approaches do not perform as good as they are claimed to be when **new technologies and applications are considered.**
- Instead of to develop yet another currently satisfiable solution,
- This work is focused on the **fundamental issues** of wear leveling:
 - Effectiveness (to evenly erase blocks)
 - Efficiency (to reduce traffic introduced by wear leveling)
 - Scalability (to have low resource requirements)

The Dual-Pool Algorithm

- Basic ideas (1 of 2): Cold-data migration

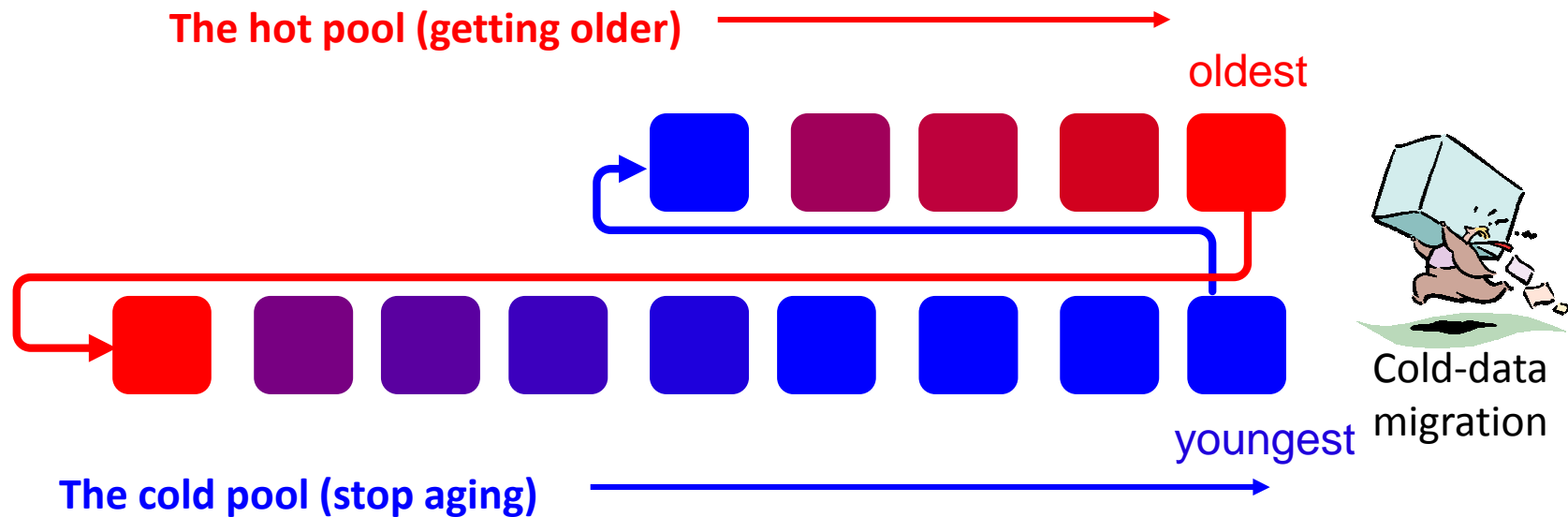
* The majority of data are cold, as the majority of blocks are young!!



- Migrating data from young blocks to old blocks
 - To **defrost** young blocks by moving cold data away
 - To **cool down** old blocks by moving cold data in

The Dual-Pool Algorithm

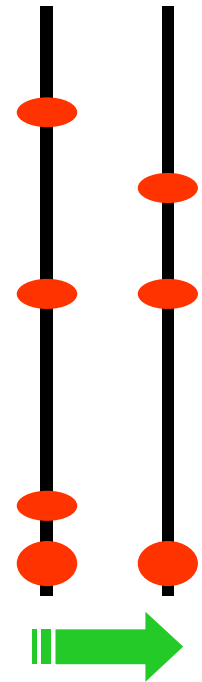
- Basic ideas (2 of 2): Hot-cold regulation



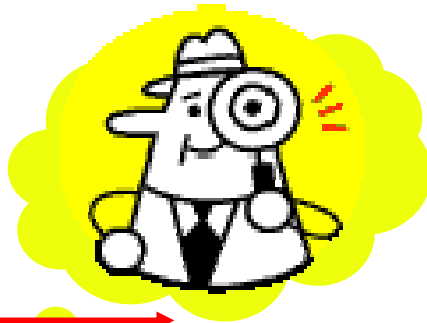
- Blocks involved in cold-data migration are protected against being involved again
 - until wear leveling takes effect

The Dual-Pool Algorithm

- Spatial locality in workloads may change from time to time
 - The basic algorithm forgets blocks in the hot pool if they stop aging
 - Similar things happen to the cold pool
 - Hot-pool resize and cold-pool resize are introduced to correct this problem



Hot pool resize



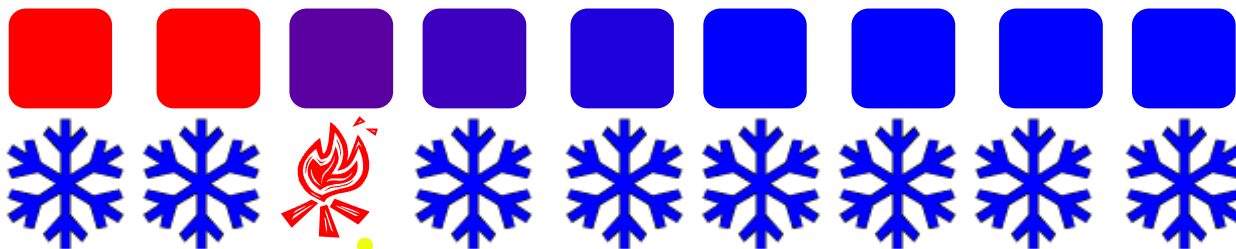
The hot pool (getting older)



oldest

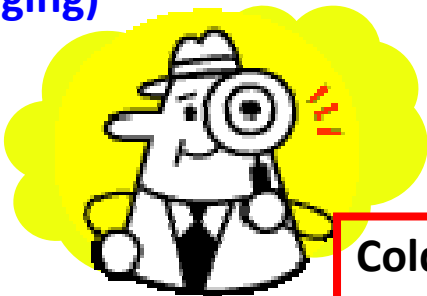


Cold-data migration



youngest

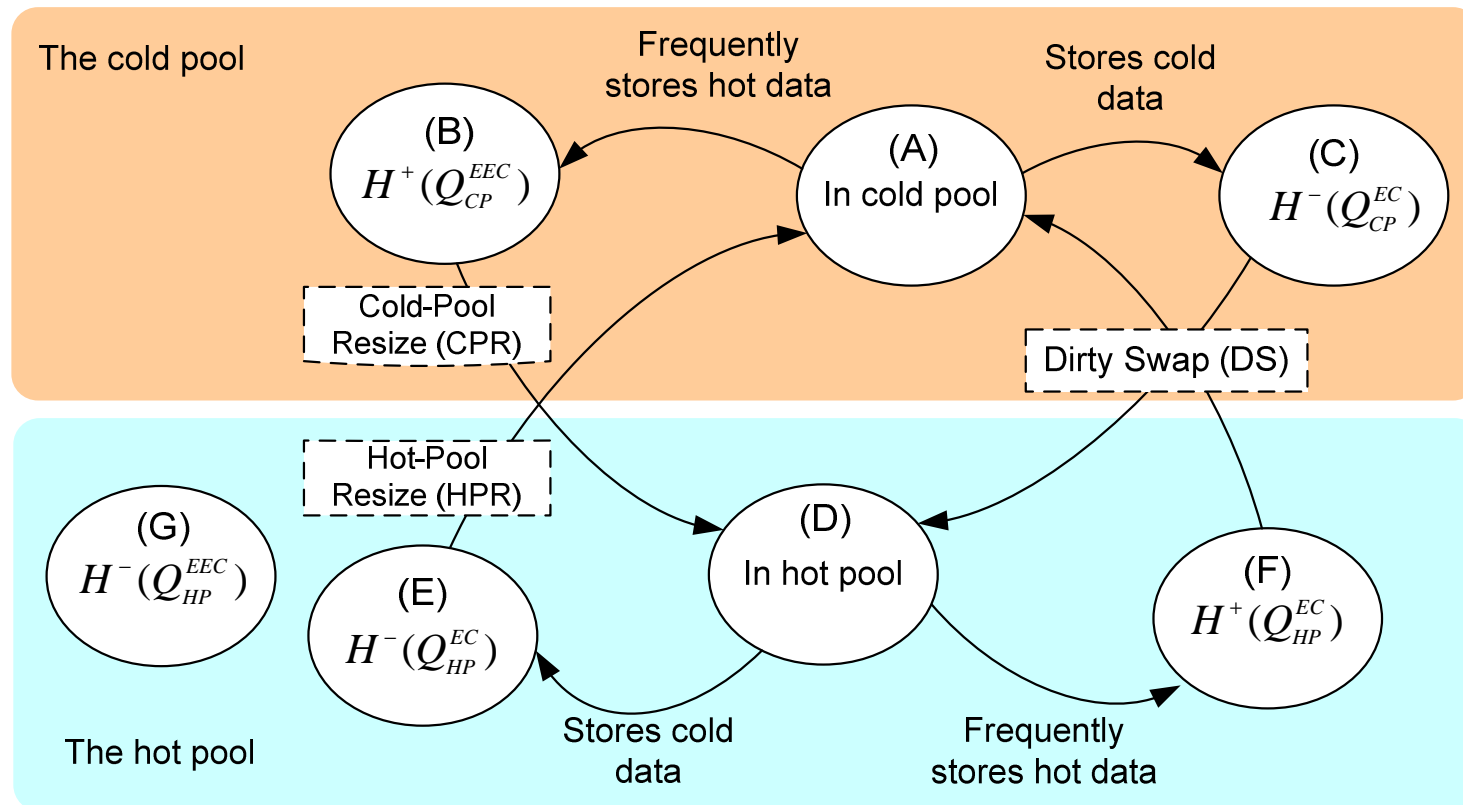
The cold pool (stop aging)



Cold pool resize

The Dual-Pool Algorithm

- State transition of blocks



Normally blocks are circulated among states $A \rightarrow C \rightarrow D \rightarrow F \rightarrow A$

Performance Evaluation

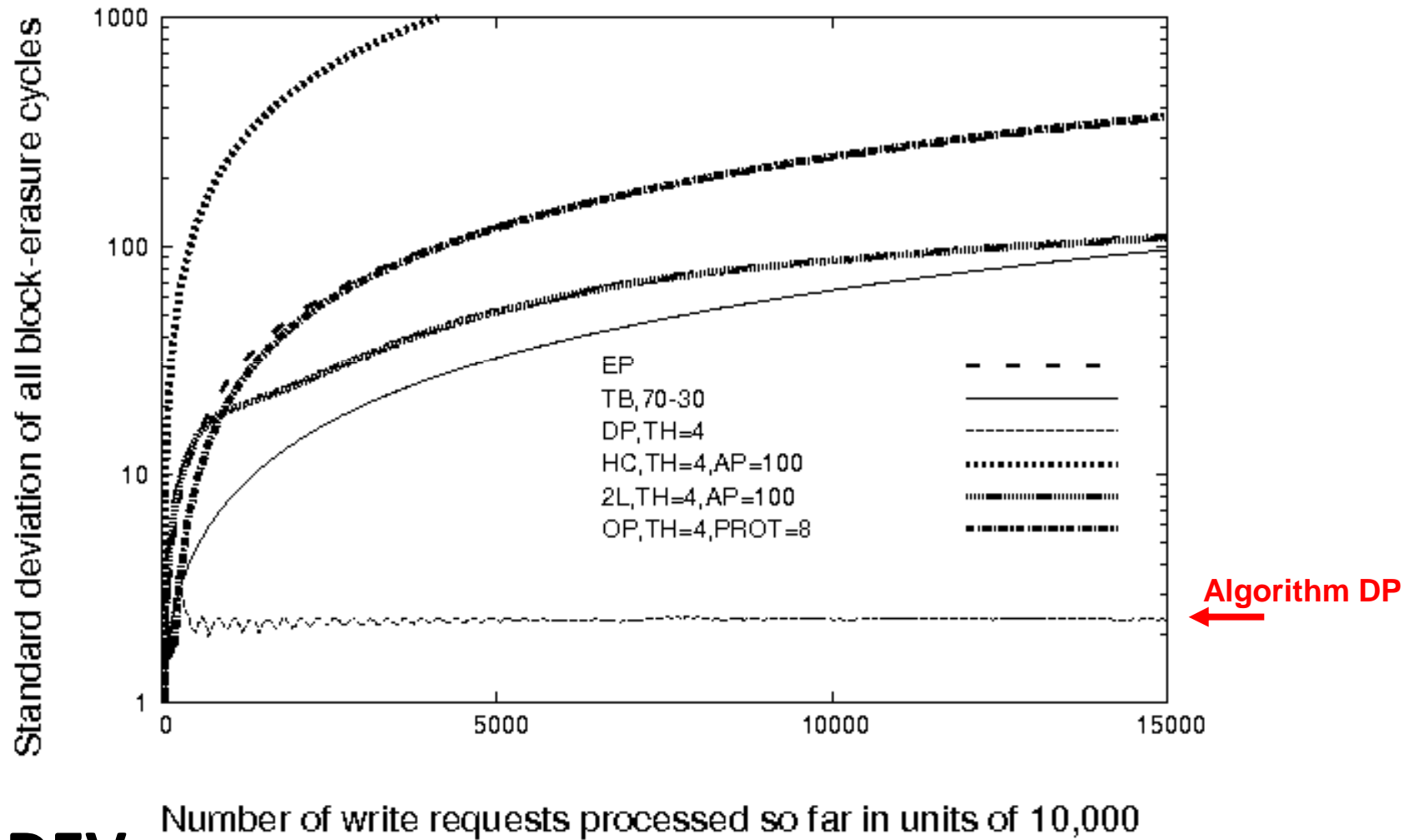
- 8 representative wear-leveling algorithms are chosen for performance comparison
- Two realistic workloads are considered
 - The disk of a UMPC
 - The storage of a multimedia appliance
- Two major performance metrics
 - Standard deviations of block-erasure cycles: (STDDEV)
 - The smaller the more even the erasure cycles are
 - Overhead ratios: (OR)
 - The ratio of overheads with WL to overheads without WL
 - The closer to 1.0 the lighter extra traffic is introduced by WL

?

Performance Evaluation

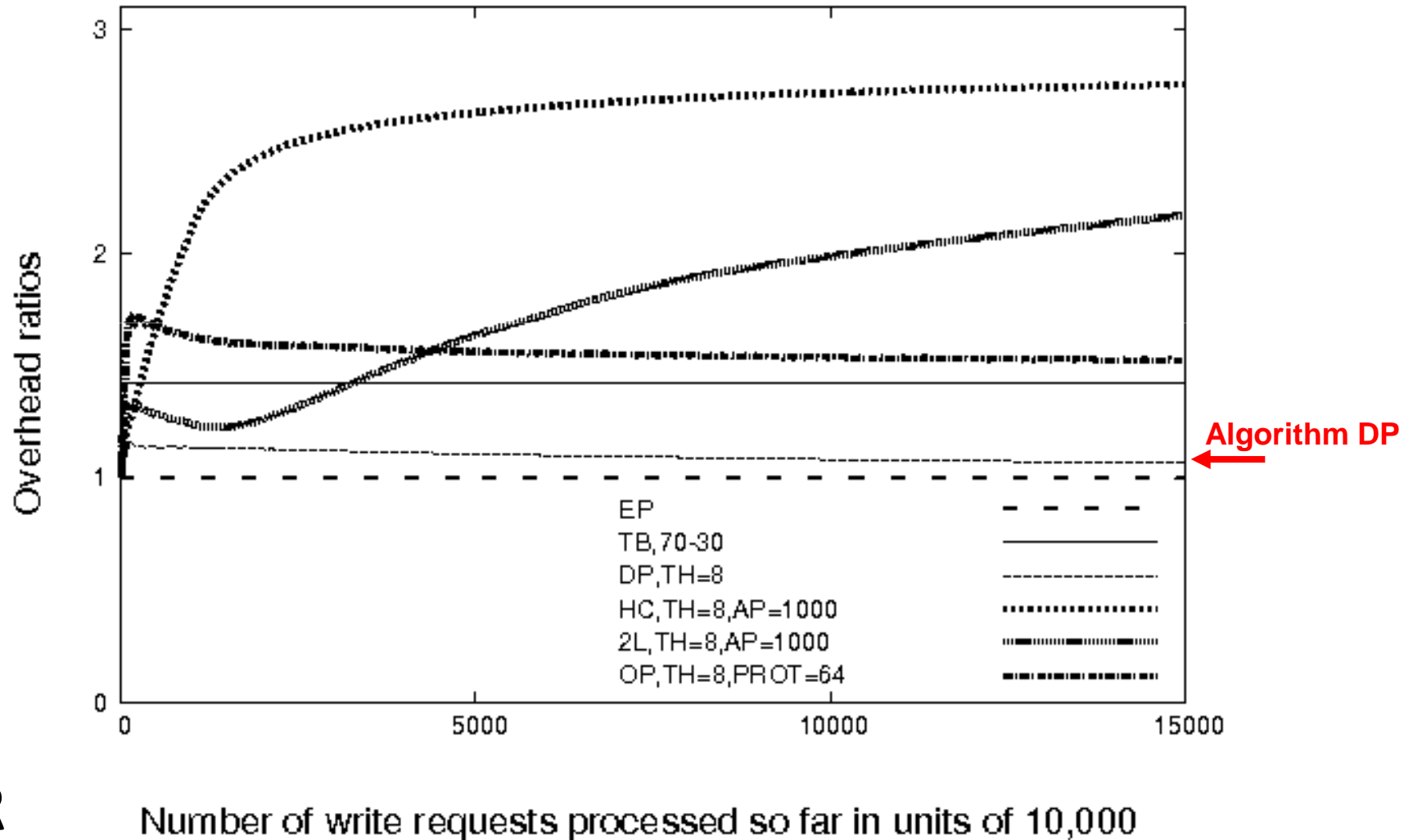
- **Algorithm EP**
 - ``SmartMedia Specification'', SSFDC Forum, 1999.
 - SanDisk Corporation, ``Sandisk Flash Memory Cards Wear Leveling''
- **Algorithm HC**
 - M-Systems, ``TruFFS Wear-Leveling Mechanism,''
- **Algorithm 2L**
 - ``Wear Leveling in Single Level Cell NAND Flash Memories,''
 - STMicroelectronics Application Note (AN1822), 2006.
- **Algorithm TB**
 - D. Woodhouse, ``JFFS: The Journaling Flash File System,''
 - C. Manning and Wookey, ``YAFFS Specification,''
- **Algorithm OP**
 - T. Gleixner, F. Haverkamp, and A. Bitvutskiy, ``UBI - Unsorted Block Images,''
- **Algorithm KL**
 - H. J. Kim and S. G. Lee, ``A New Flash-Memory Management for Flash Storage System,''
- **Algorithm CAT**
 - M. L. Chiang, Paul C. H. Lee, and R. C. Chang, ``Using Data Clustering To Improve Cleaning Performance For Flash Memory,''

Performance Evaluation



STDDEV

Performance Evaluation

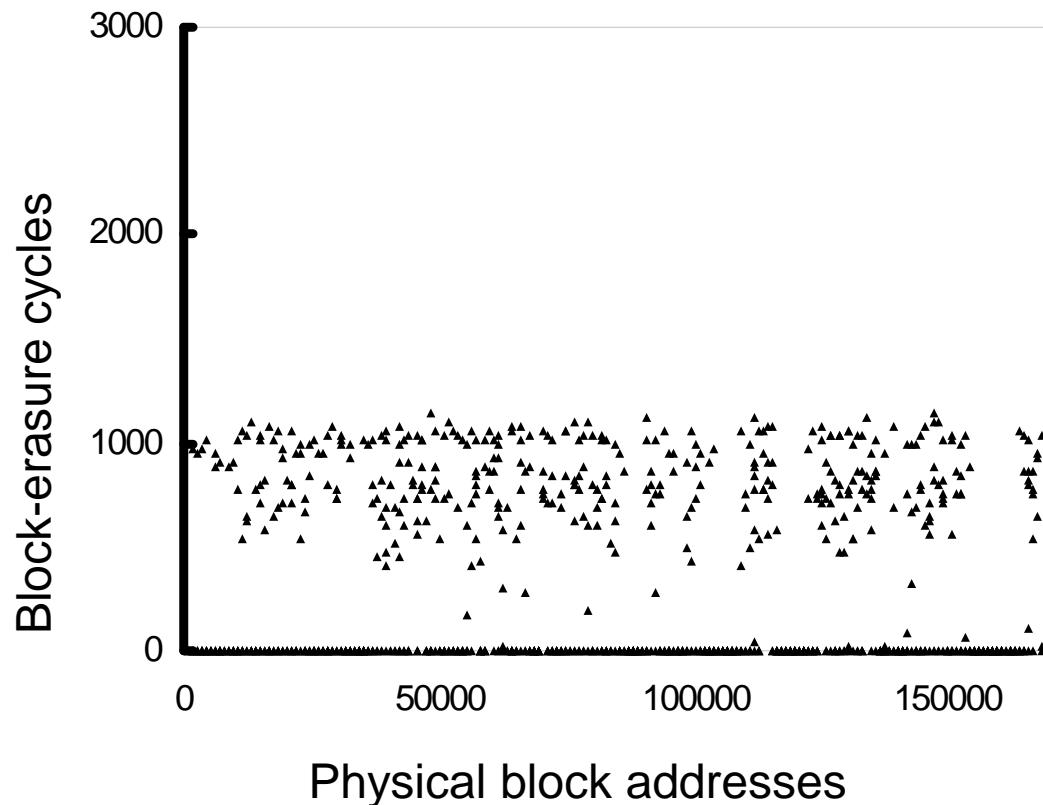


OR

Number of write requests processed so far in units of 10,000

Performance Evaluation

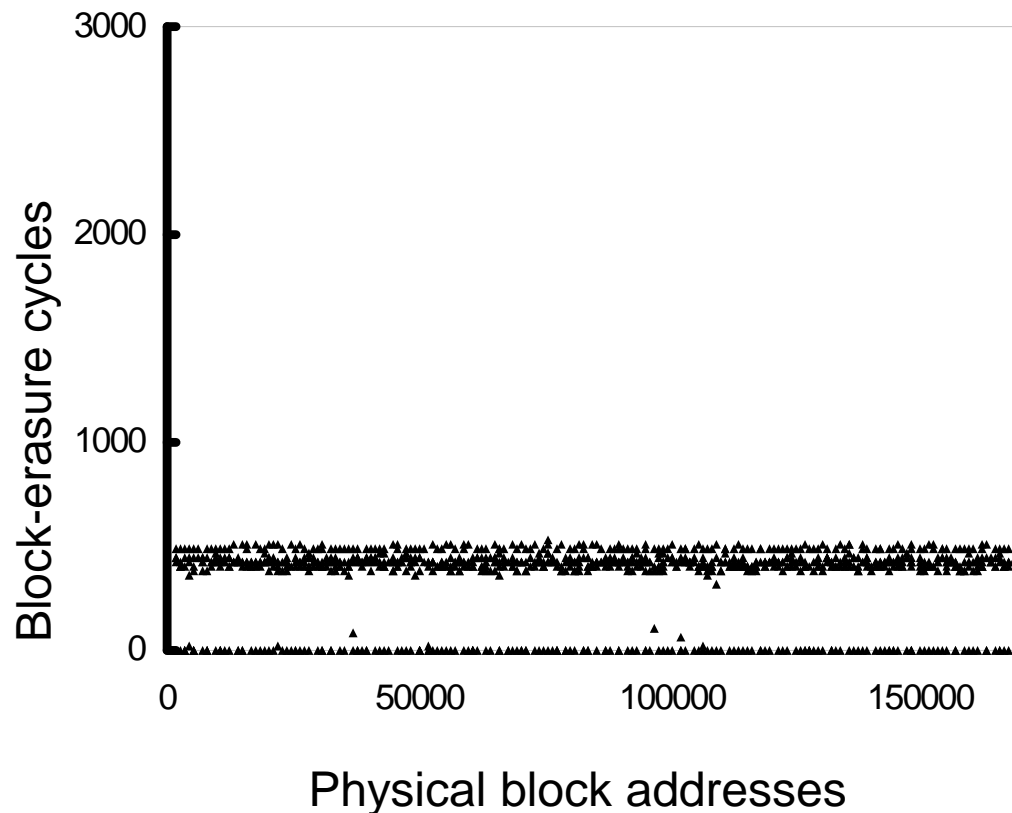
- Algorithm EP: out-place updates only
 - Sandisk and Smartmedia forum



- Many blocks never have chances to get erased

Performance Evaluation

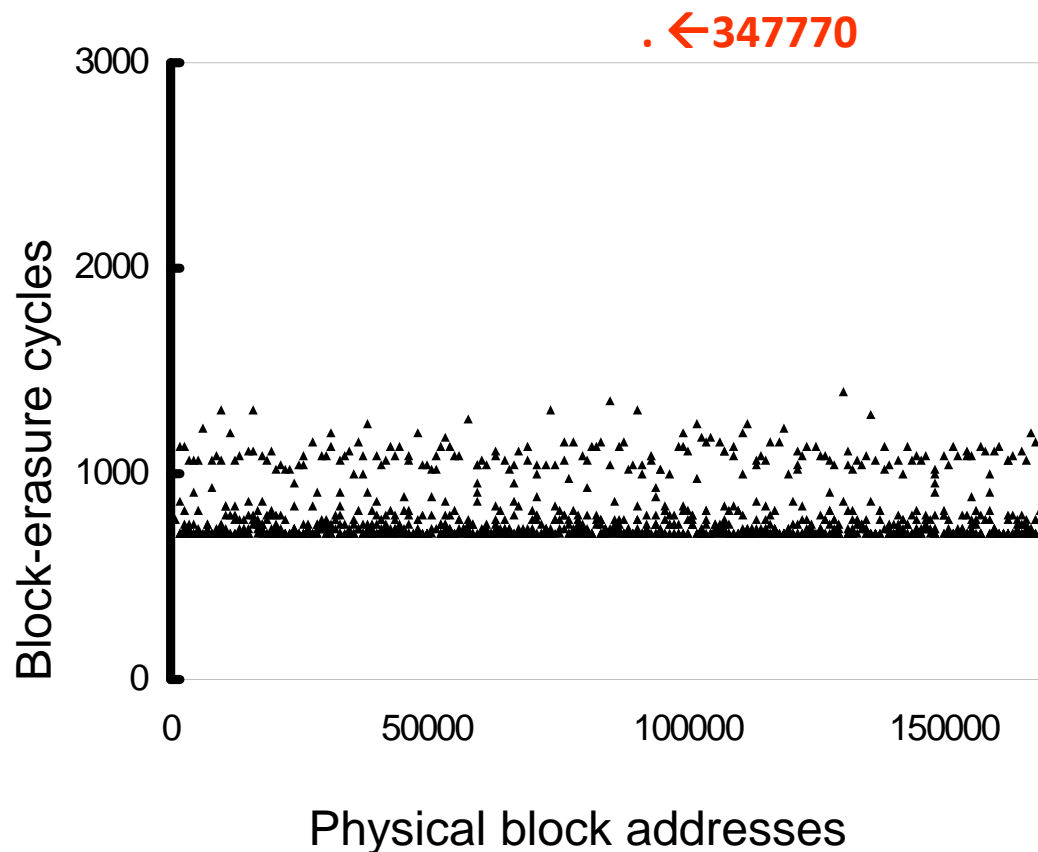
- Algorithm TB:
 - Out of 100 erasure operations, 1 goes to a block full of live data
 - JFFS2 and YAFFS



- Many blocks were never involved in wear leveling
 - These blocks have a **small amount of non-live data**
 - GC says no, because there are too many live data
 - WL says no, because they are not of all live data

Performance Evaluation

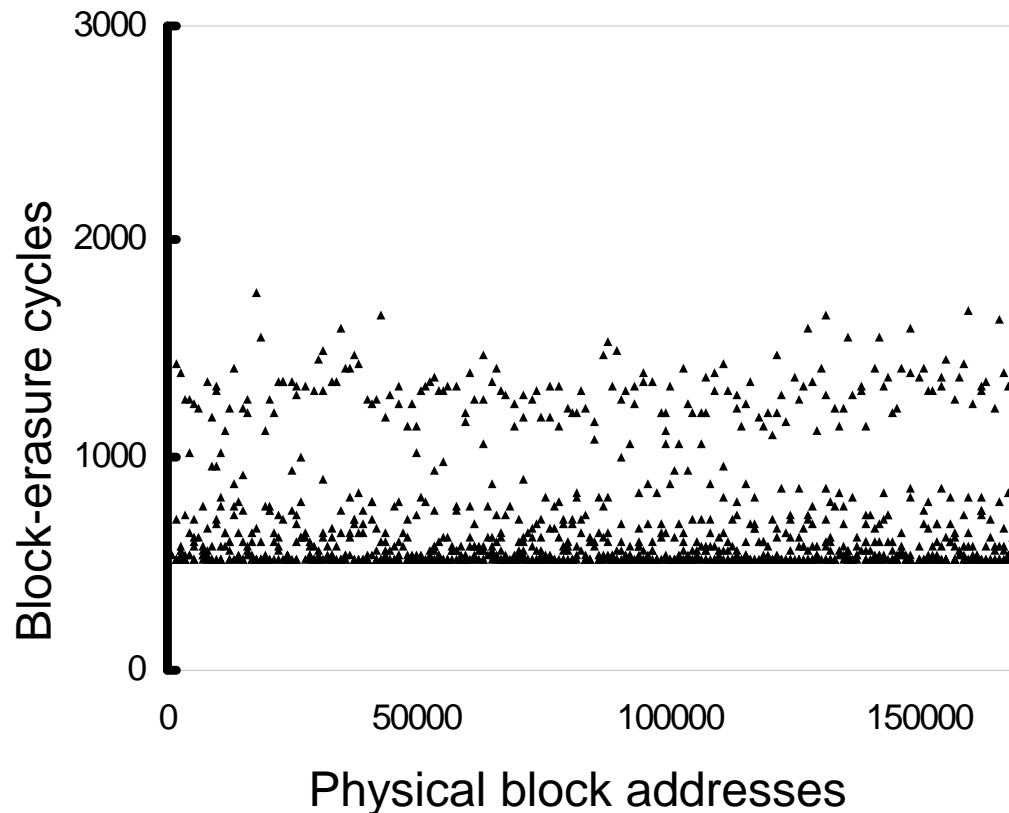
- Algorithm HC:
 - Swap data in the oldest block and the youngest block
 - M-System TrueFFS



- An old block was constantly involved in the swapping of data
 - The erasure cycle had even achieved **347,770** (beyond the boundary of the figure)

Performance Evaluation

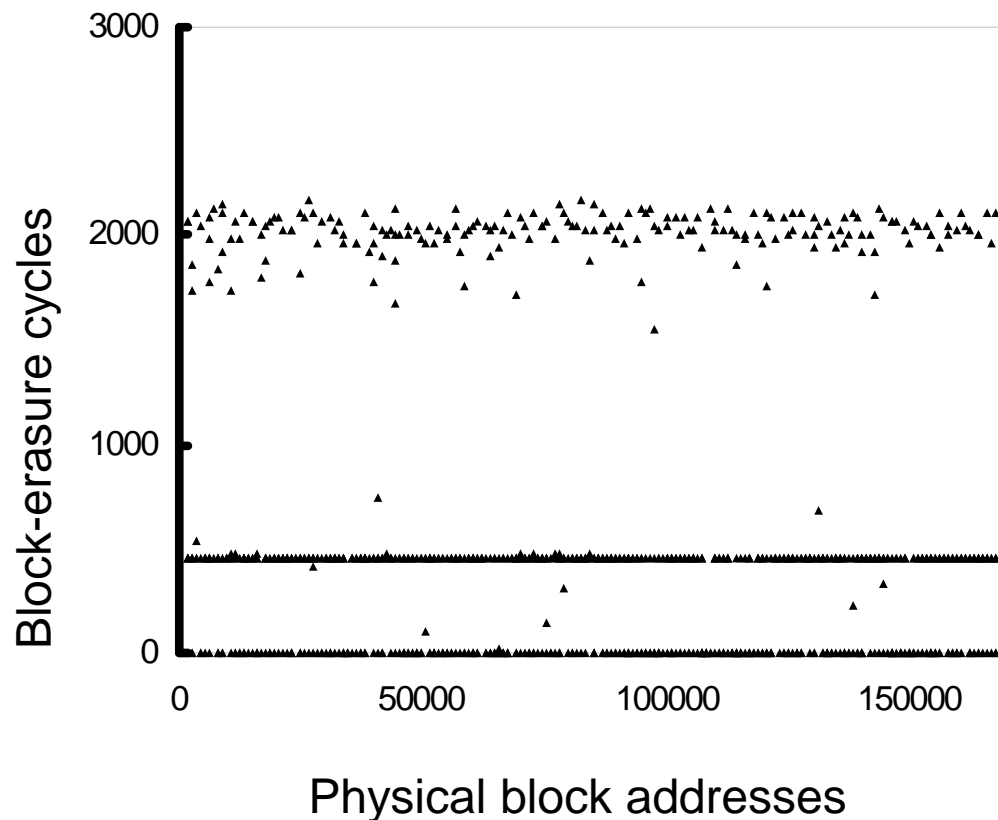
- Algorithm 2L
 - Move data away from the youngest block
 - STMicroelectronics and JFFS3



- One single erasure to the oldest block causes a lot of young blocks to be erased
- There were not enough old blocks to accommodate cold data
 - Cold data were pointlessly circulated among young blocks

Performance Evaluation

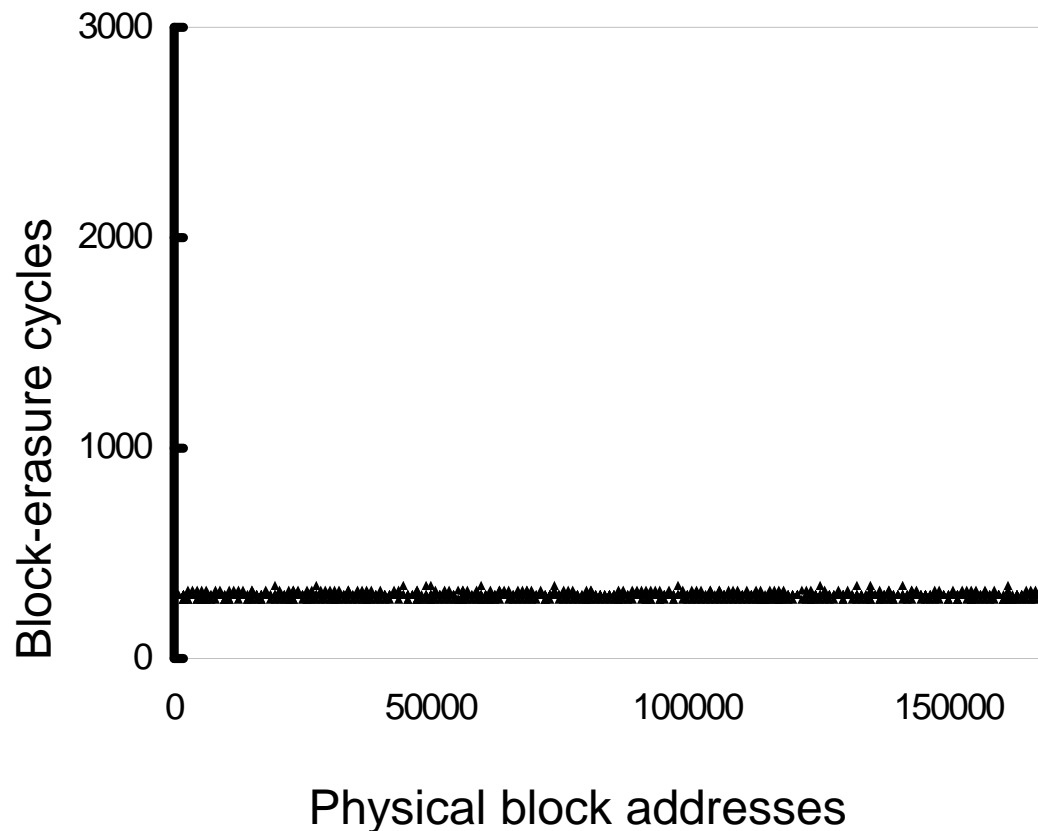
- Algorithm OP
 - Move data from the youngest block to the **erased** block
 - The erased block is then protected until a certain number of blocks are erased
 - UBI project of Linux MTD



- 3 stripes can visually be identified
- Upper stripe: blocks overly involved in wear leveling
- Middle stripe: blocks properly participated in wear leveling
- Bottom stripe: young blocks were not timely involved in wear leveling because algorithm OP's triggering condition was usually not satisfied

Performance Evaluation

- Algorithm DP
 - Cold-data migration and hot-cold regulation
 - Proposed in this work



- Block-erasure cycles were small, and they were close to one another

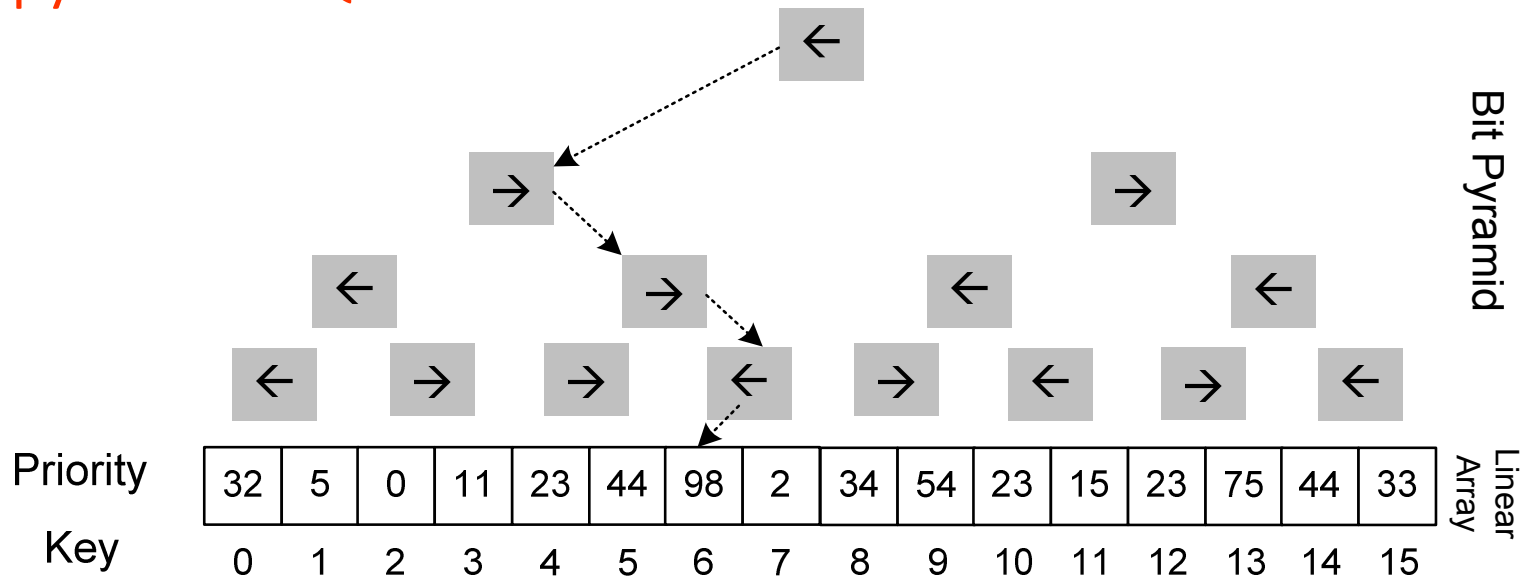
Performance Evaluation

- Garbage-collection policies that take wear leveling into consideration
 - Algorithm KL: Kim and Lee
$$(1 - l) \times \mu_i + l \times \frac{\epsilon_i}{\epsilon_{max} + 1}, \quad l = \begin{cases} \frac{2}{1 + e^{\frac{k\epsilon}{\Delta\epsilon}}} & \text{if } \Delta\epsilon \neq 0 \\ 0 & \text{, otherwise} \end{cases}$$
 - Algorithm CAT: Chiang et al.
$$\frac{\mu_i * a_i(t)}{(1 - \mu_i) * \epsilon_i}$$
 - O(n) time complexity and many floating-point operations!
 - Very slow! Did not complete one run of simulation in one month!

Performance Evaluation

- Scalability: 20G flash has 163,840 blocks
 - Blocks are prioritized in terms of wearing information
 - Need to search a block
 - Need re-prioritize a randomly selected block
- Random priority queues
 - A computational scalable implementation of random is needed!

pyramid RPQ



- Lookup queue head: $O(1)$
- Random priority update: $O(\log(n)\log(n))$
- Search an item: $O(1)$
- The bit pyramid is very small!
- Highly optimized for bit manipulating **machine instructions**
 - BT, BTS, and BTC in 386+
- Highly optimized for **cache organization**
 - A cache line (32 bytes) holds a small bit pyramid

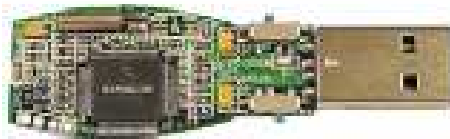
Performance Evaluation

- Scalability: To manage a 20 GB NAND flash memory

	RAM-space required by wear leveling (KB)	CPU cycles to complete one run of simulation (in units of 1,000,000)
HC	3840	26,683,489
2L	3840	43,908,891
DP	920	21,396,858
OP	5120	37,359,400
KL	640	Not completed in one month
CAT	1280	Not completed in one month

Implementation

- The proposed algorithm has been successfully implemented in
 - Firmware of a USB thumb drive
 - MTD subsystem of a Linux 2.4-based embedded computer



Freescale RDHCS12UF32TD
•MC9S12UF32 SoC
•128 MB NAND



Aiji System SMDK2410
•Samsung S3C2410
•Accepts Smartmedia Cards

Conclusion

- A new wear-leveling algorithm is proposed
 - Cold-data migration
 - Hot-cold regulation
- Different from prior work
 - Not being confined to current technologies and applications
 - In depth investigation of fundamental WL issues
- The proposed approach has been implemented and is proven being useful