Write Endurance in Flash Drives: Measurements and Analysis

Simona Boboila
Peter Desnoyers

Northeastern University

Motivation

- Flash used for many years in consumer devices (photography, media players, portable drives)
  - Parameters of flash not of interest to users (usually proprietary/undisclosed)
- But… only recently flash used for storage in laptops and desktops
  - Now we care!
    - efficient access to data (in intensively used storage)
    - consistent average performance (over large periods of time)
  - Understand flash internals:
    - harness its strengths
    - address its limitations: write endurance, garbage collection

Our work

- To uncover internals of flash we investigated real USB flash drives:
  - chip-level testing
  - analysis and simulation
  - reverse engineering
  - timing analysis
  - whole-device testing

  chip-level testing, analysis and simulation: In the paper

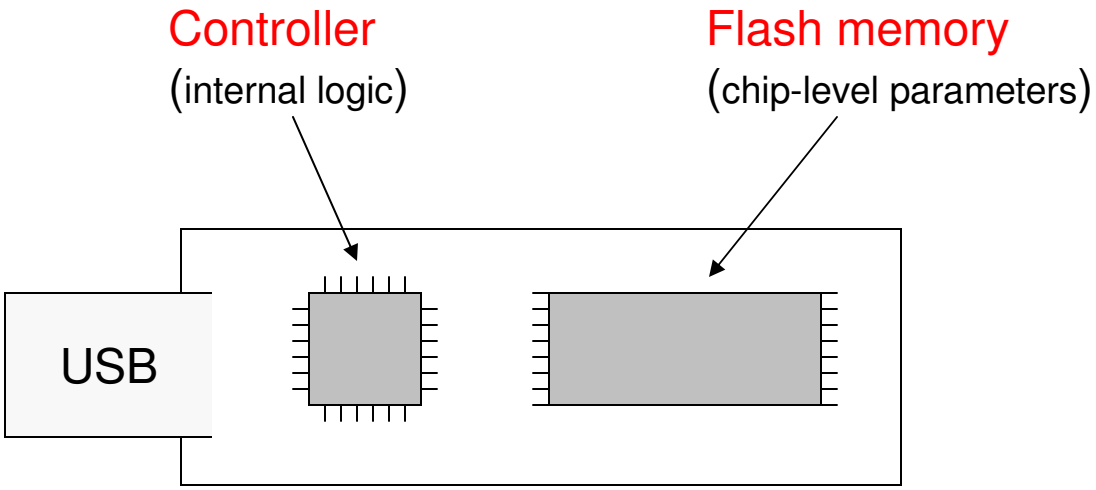
  reverse engineering, timing analysis, whole-device testing: Discussed next

- Why USB flash drives?
  - Device disassembling, destructive testing, reverse engineering more difficult to do for more sophisticated devices

Outline

- Device lifespan : predictions & measurements
- Timing analysis : non-intrusive investigation
- Scheduling : storage optimization for flash devices

USB flash drive

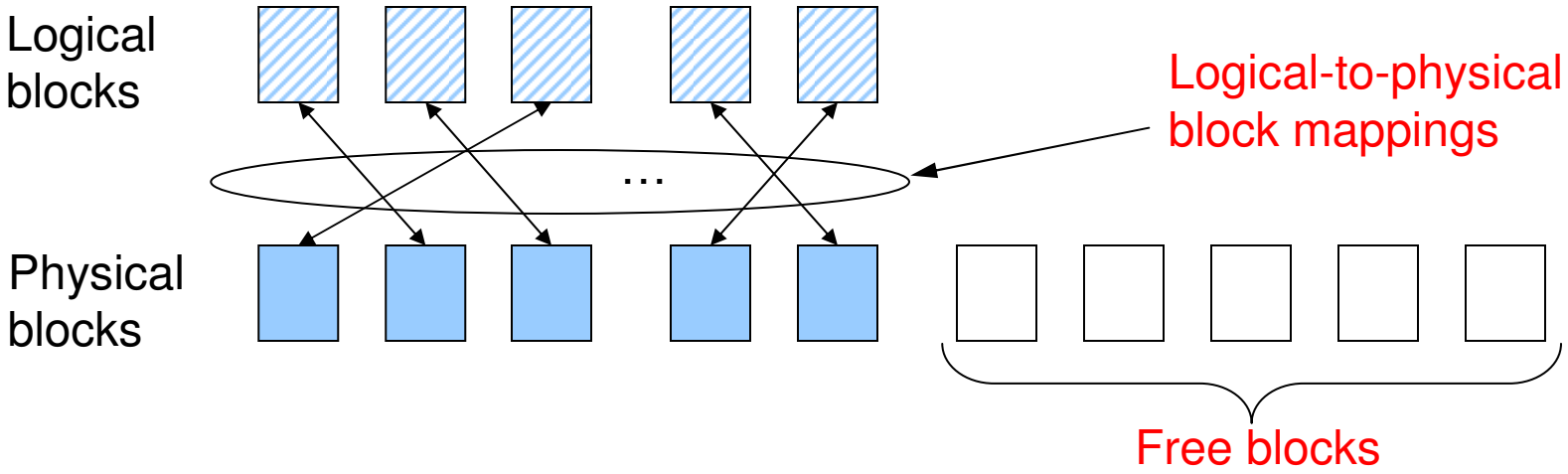


- Flash memory: chip-level parameters
- Controller: internal algorithms
  (implemented in the Flash Translation Layer, FTL)

In the paper

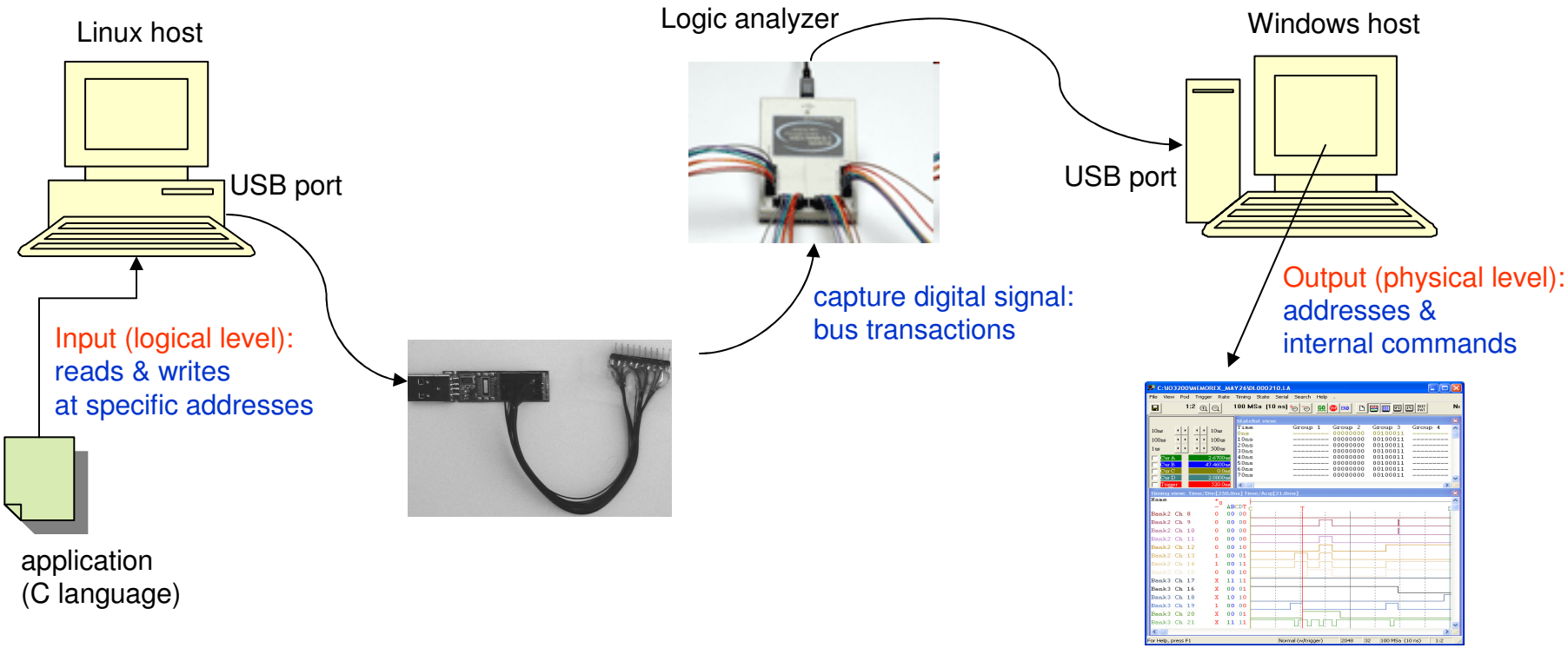
Discussed next

Flash Translation Layer (FTL)

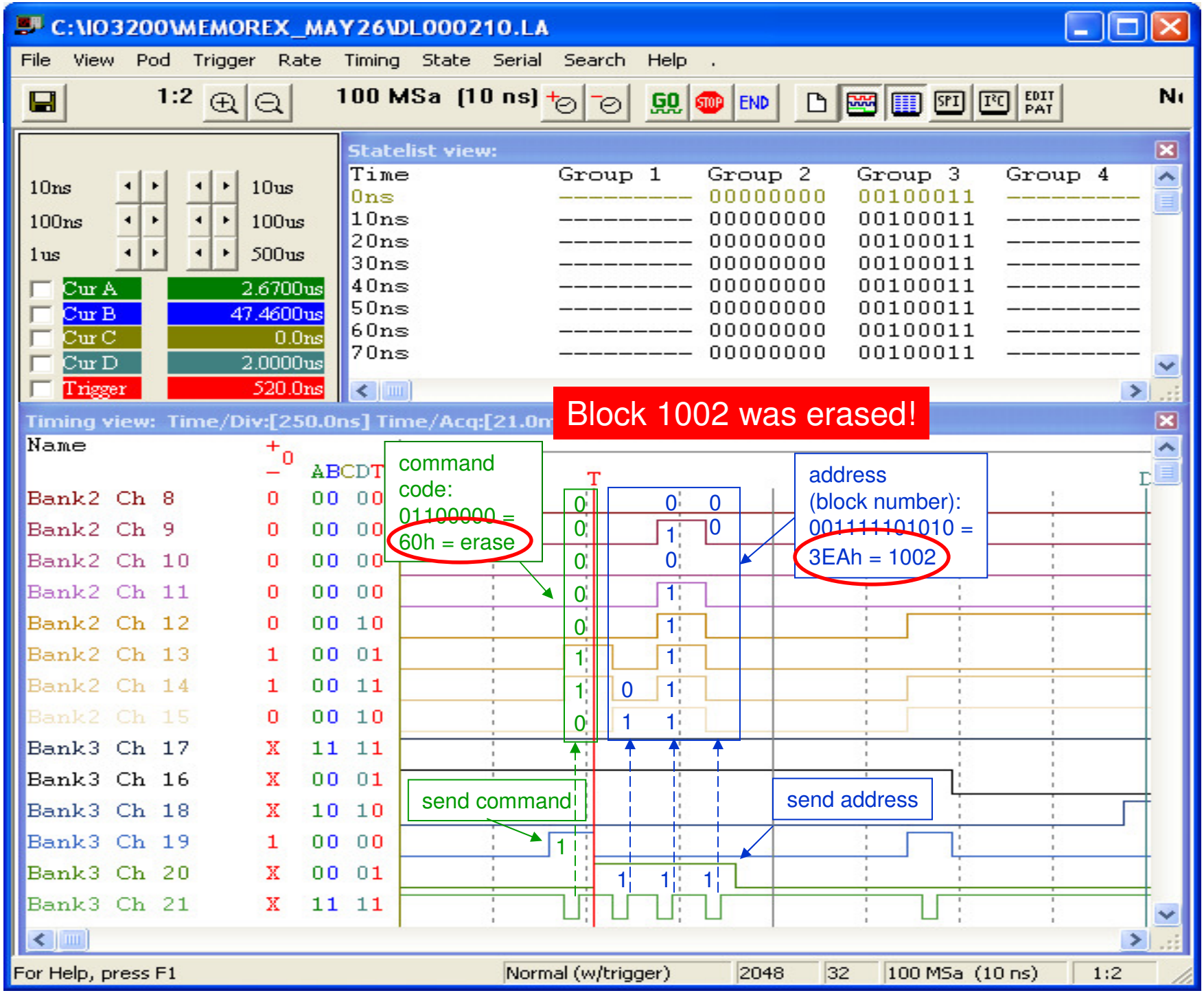


- Flash can not be overwritten (has to be erased before writing again)
  - FTL uses a pool of free blocks to accommodate new writes before old data is erased
  - Different granularity of program (page) vs. erase (block, ≥ 32 pages)
- Flash wears out in time (limited number of writes/erasures)
  - FTL distributes the number of writes/erasures evenly among physical blocks

Reverse engineering of FTL



- Input (logical level): reads/writes issued from a Linux USB host at specific logical addresses
- Output (physical level): internal commands and physical addresses captured with a IO-3200 logic analyzer

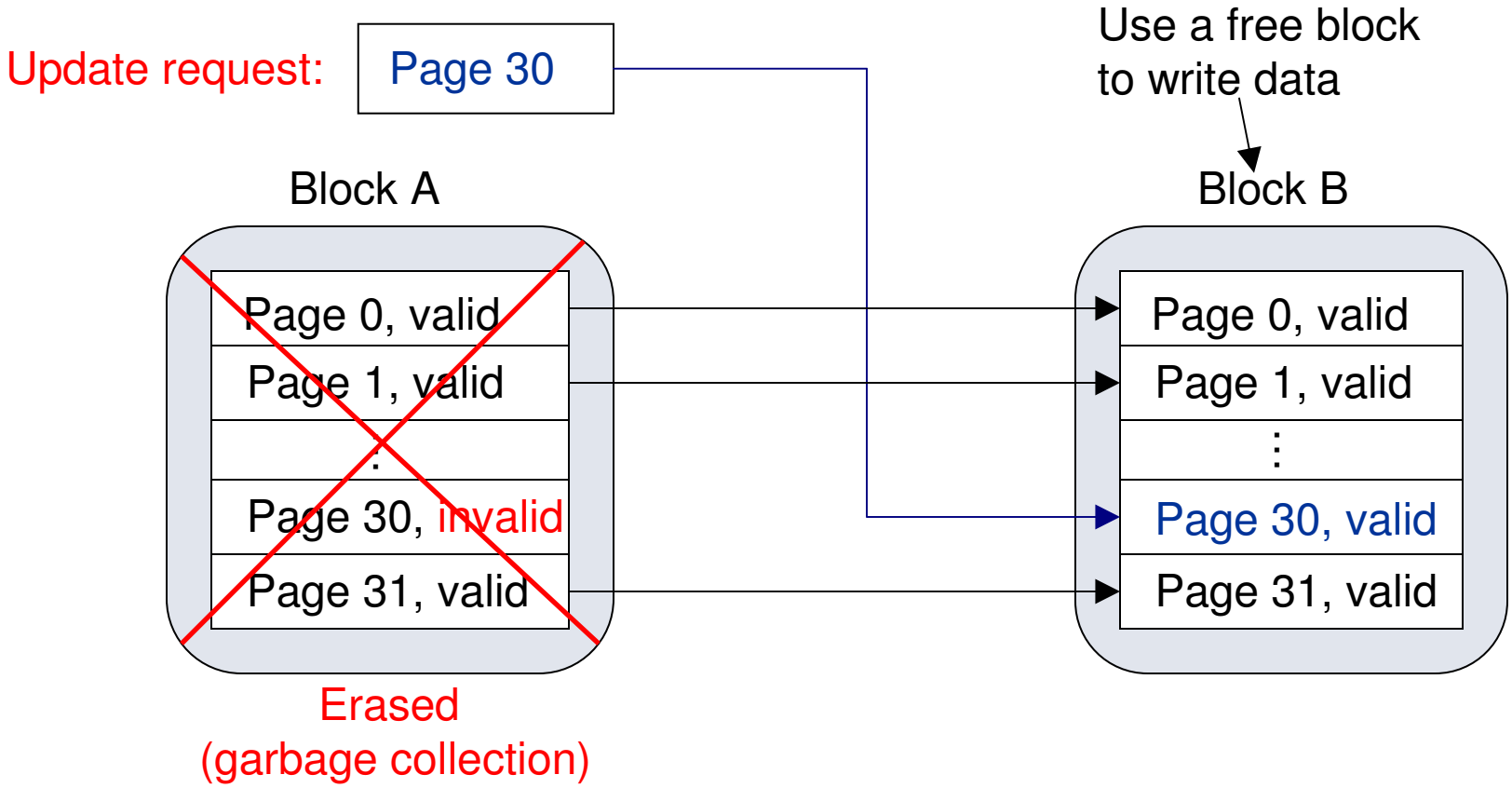

C:\IO3200\MEMOREX_MAY26\DL000210.LA
File View Pod Trigger Rate Timing State Serial Search Help
1:2
100 MSa (10 ns)
GO STOP END
Statelist view:
Time Group 1 Group 2 Group 3 Group 4
0ns 00000000 00100011
10ns 00000000 00100011
20ns 00000000 00100011
30ns 00000000 00100011
40ns 00000000 00100011
50ns 00000000 00100011
60ns 00000000 00100011
70ns 00000000 00100011
Cur A 2.6700us
Cur B 47.4600us
Cur C 0.0ns
Cur D 2.0000us
Trigger 520.0ns
Block 1002 was erased!
Timing view: Time/Div:[250.0ns] Time/Acq:[21.0n
Name
Bank2 Ch 8
Bank2 Ch 9
Bank2 Ch 10
Bank2 Ch 11
Bank2 Ch 12
Bank2 Ch 13
Bank2 Ch 14
Bank2 Ch 15
Bank3 Ch 17
Bank3 Ch 16
Bank3 Ch 18
Bank3 Ch 19
Bank3 Ch 20
Bank3 Ch 21
command code: 01100000 = 60h = erase
address (block number): 001111101010 = 3EAh = 1002
send command
send address
For Help, press F1
Normal (w/trigger)
2048
32
100 MSa (10 ns)
1:2

Specifics of experiments

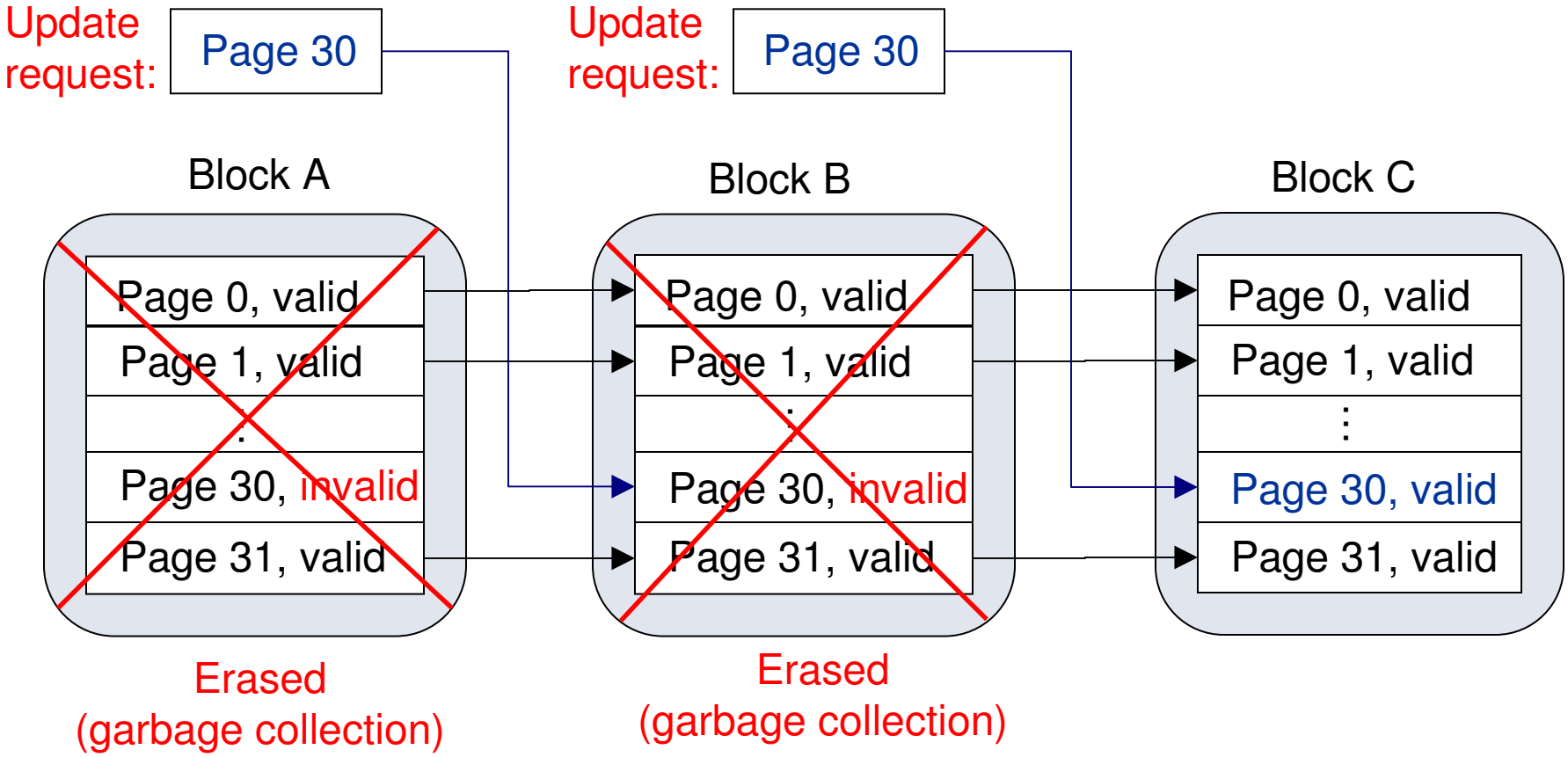
- Investigated USB drives:
  - Generic – 64MB, Hynix HY27US08121A
  - House – 2GB, Intel 29F16G08CANC1
  - Memorex – 512MB, Mini TravelDrive

- Writing pattern:
  - Step 1. Write all logical blocks completely.
  - Step 2. Overwrite some page.

Page update mechanism: Generic device



Successive updates: Generic device



- For Generic, one page update triggers a block erasure!!
- Only the list of free blocks is used: worn out faster!!

Predicting lifespan: Generic device

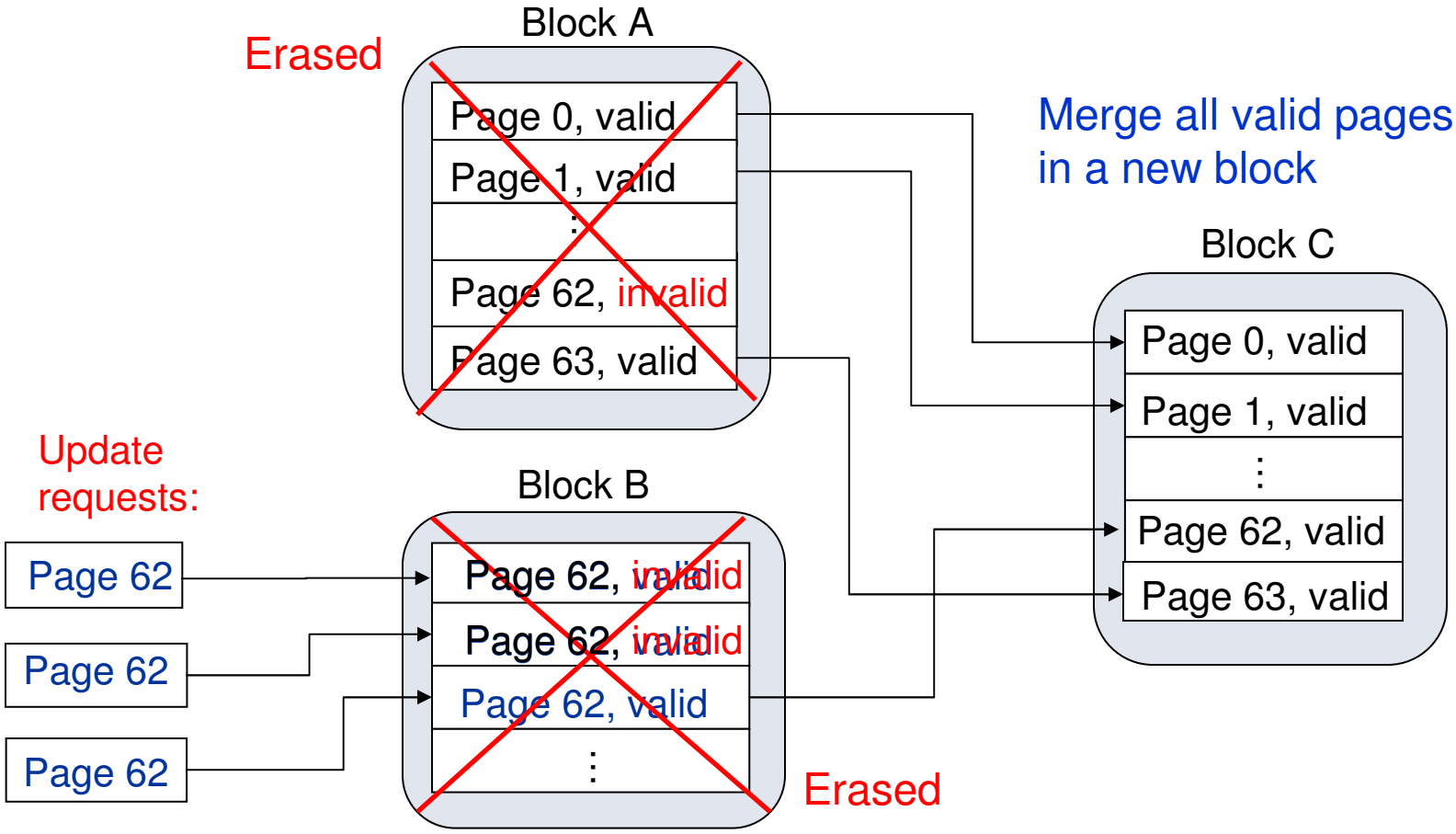
Can we predict the lifespan of the device?

- Internal algorithm:
  - cycle through the list of free blocks
  - erase one block at each page update
- Predicted lifespan = $h \times m = 6 \times 10^7$
  - h = chip-level endurance
  - m = number of free blocks
- Measured lifespan = 7.7 x 10^7

Device lifespan ≈ Chip-level endurance + FTL algorithm

More complex FTL: House device

Less frequent garbage collection: Can accommodate several updates of a block into a single new block before erasing the old data



Use a free block to store new data

Predicting lifespan: House device

Can we predict the lifespan of the device?

- Internal algorithm:
  - cycle through the list of free blocks
  - accommodate *k* pages per block, 1≤ *k* ≤ block size
  - erase 2 blocks
- Predicted lifespan:
  - Uncertainty in tracing *k*

  - *h* = chip-level endurance,
  - *m* = number of free blocks,
  - *k* = number of pages written per block before erasing

(*) $$\frac{k \times h \times m}{2} \in \left[1.5 \times 10^{7}, \quad 9.6 \times 10^{8}\right], \quad with \quad k \in [1, \quad block\_size]$$

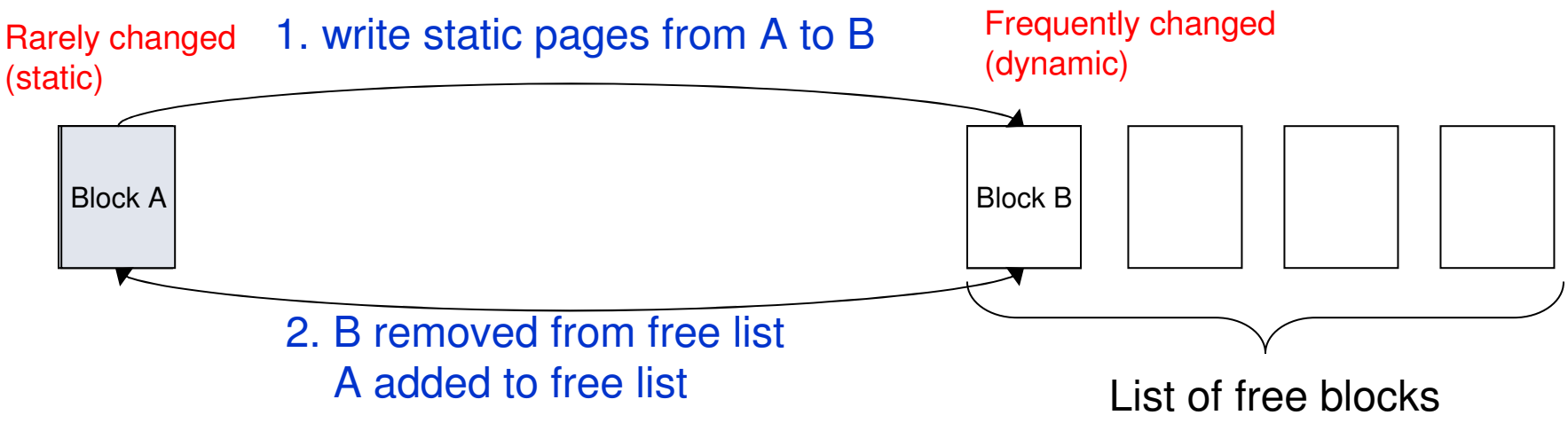
- Measured lifespan: 1.06 x 10^8

Device lifespan ≈ Chip-level endurance + FTL algorithm

(*) Refinement of the bound in the paper.

Even more complex FTL: Memorex device

Static wear-leveling: periodically swaps static blocks with frequently updated blocks



Predicting lifespan: Memorex device

Can we predict the lifespan of the device?

- Internal algorithm:
  - cycle through the entire zone
  - accommodate up to a full block of pages before erasing
- Predicted lifespan = $z \times k \times h = 6.5 \times 10^{10}$
  - z = number of blocks per zone
  - *k* = number of pages per block
  - *h* = chip-level endurance
- Device did not break!

Outline

- Device lifespan : predictions & measurements
- Timing analysis : non-intrusive investigation
- Scheduling : storage optimization for flash devices

Timing analysis

- What can we figure out from timing analysis?
  - Garbage collection frequency
  - Writing patterns that trigger garbage collection
  - If static wear-leveling is used, and how frequently

  } In the paper

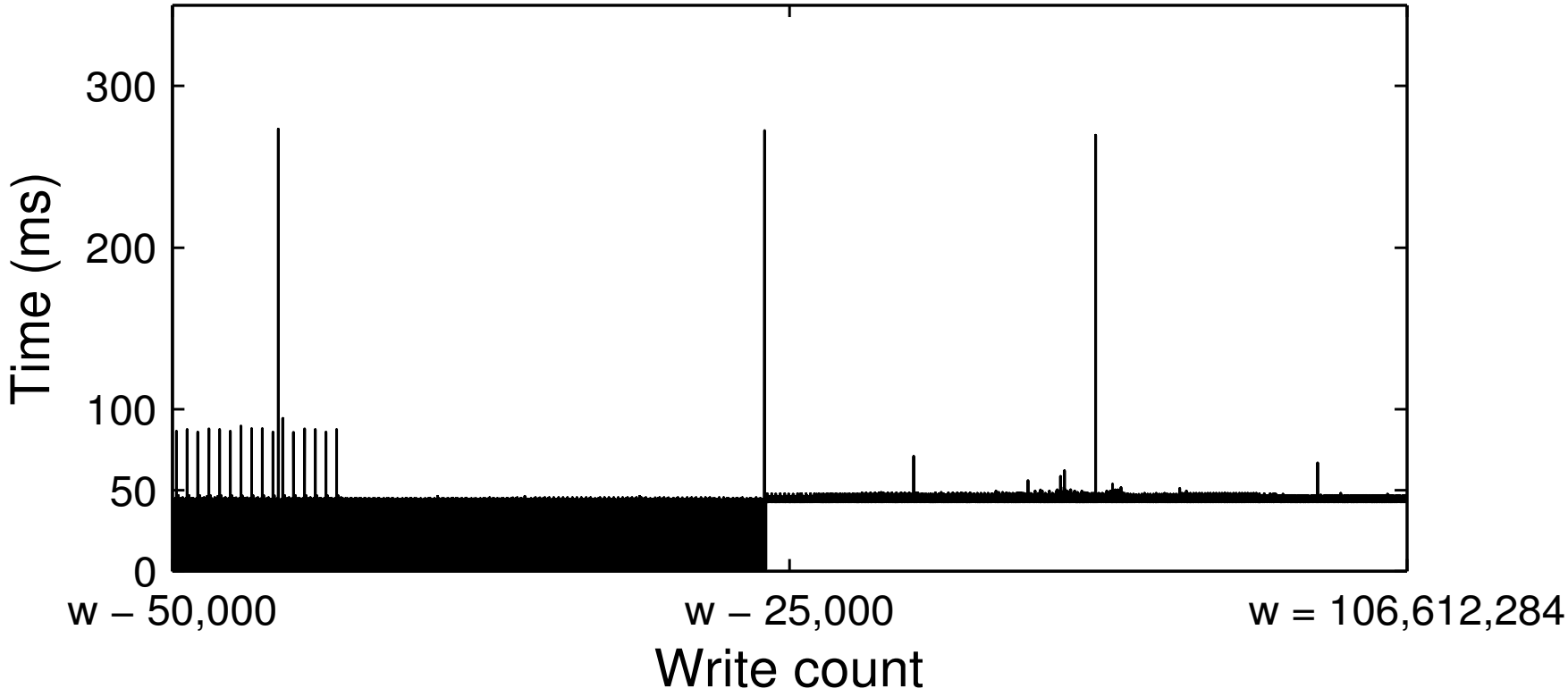
  - If the device is approaching its end of life

  } Discussed next

End-of-life signature: House device

Is the device approaching its end of life?

- At 25,000 operations before the end, all operations slow to 40 ms ≈ erasure at every write



Outline

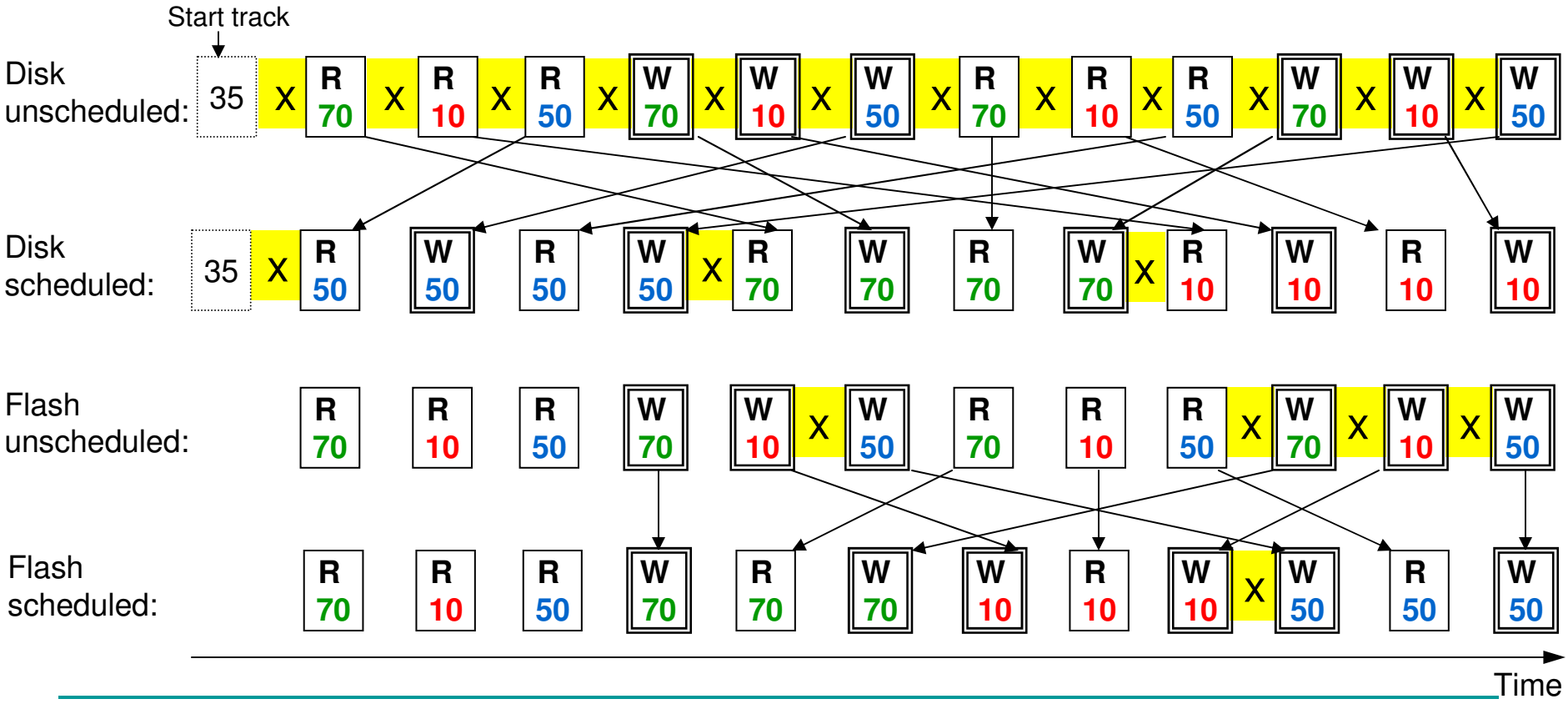
- Device lifespan : predictions & measurements
- Timing analysis : non-intrusive investigation
- Scheduling : storage optimization for flash devices

Latency problem: flash versus disk

- Latency:
  - Disk: mechanical (seek delays)
  - Flash devices: lack of free blocks (garbage collection delays)
- Solution: find an optimal scheduling to minimize latency
  - Disk:
    - Elevator algorithm: requests sorted by track number and serviced only in the current direction of the arm movement
  - Flash devices:
    - Key observation:
      - for writes issued to the same data block, FTL uses the same update block
      - for writes issued to different data blocks, FTL uses different update blocks
    - Solution:
      - Reorder data streams to service requests to the same data block consecutively
    - Result:
      - Use the free space compactly => reduce erasure frequency
    - No need to reschedule reads!!

An example: scheduling vs. no scheduling

- Address rounded to: track number (disk); block number (flash)
- X = seek (disk); garbage collection (flash)
- R = read; W = write
- Flash: 2 free blocks



Garbage collection overhead 4x smaller with scheduling vs. no scheduling!

Implications for storage systems

- Optimization of servicing requests:
  - Reduce garbage collection and improve performance
  - Internals of flash devices require a new scheduling paradigm for flash

- We expect our results to apply to:
  - Most removable devices (e.g. SD, CompactFlash, etc.) and low-end SSDs with little free space and RAM
  - Example: JMicron's JMF602 flash controller, used for many low-end SSDs: 8-16 flash chips, 16K RAM, 7% free space

Conclusions

- Lifespan of flash devices is a function of chip-level endurance and internal algorithms
- Flash exhibits specific timing patterns towards end of life
- New scheduling algorithms designed specifically for flash-based storage are necessary to extract maximum performance

Questions?