Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems

Jihong Kim

Seoul National University

NVRAMOS 2020

Evanesco: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems

Myungsuk Kim, Jisung Park, Geonhee Cho, Yoona Kim,
Lois Orosa, Onur Mutlu, and Jihong Kim

Seoul National University

SAFARI Research Group, ETH Zürich

ASPLOS 2020

Executive Summary

- **Motivation:** Secure deletion is essential in storage systems as modern computing systems process a large amount of security-sensitive data.

- **Problem:** It is challenging to support data sanitization in NAND flash-based SSDs.
  - **Erase-before-write property** → no overwrite on stored data
  - **Physical data destruction** → high performance & reliability overheads

- **Evanesco:** A low-cost data-sanitization technique w/o reliability issues
  - Uses on-chip access-control mechanisms instead of physically destroying data
  - Manages access-permission (AP) flags inside a NAND flash chip
    - Data is not accessible once the flash controller sets the data's AP flag to *disabled*.
    - An AP flag cannot be reset before erasing the corresponding data.

- **Results**
  - Provides the same level of reliability as an unmodified SSD (w/o data-sanitization support)
    - Validated w/ 160 real state-of-the-art 3D NAND flash chips
  - Significantly improves performance and lifetime over existing data-sanitization techniques
    - Provides comparable (94.5%) performance with an unmodified SSD

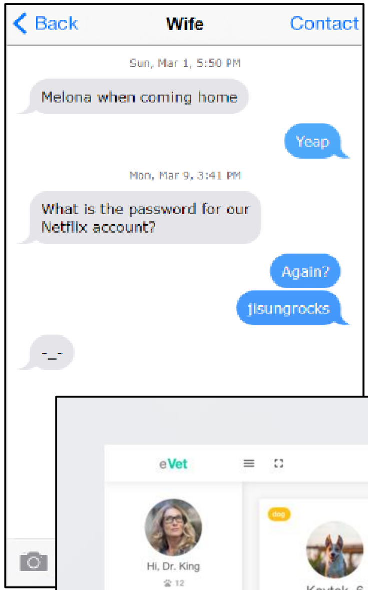
Outline

- **Secure Deletion in NAND Flash-Based SSDs**
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD
- **Evaluation**
- **Conclusion**

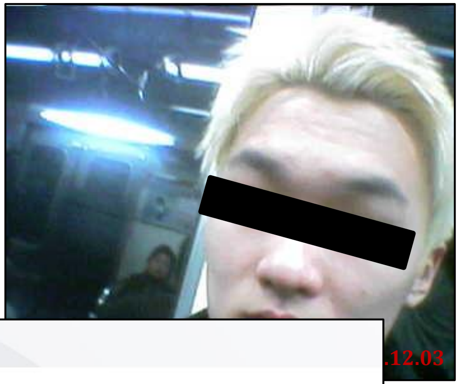
Secure Deletion in Storage Systems

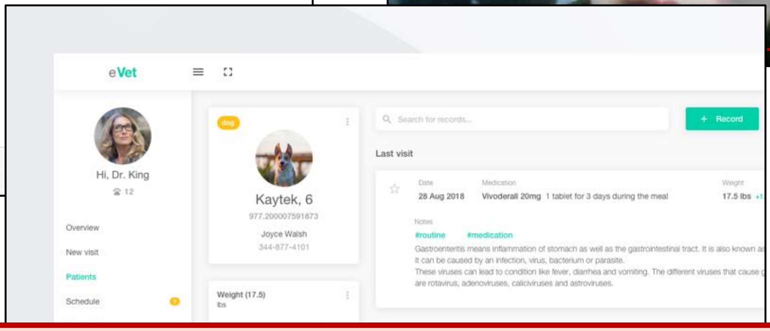
- Security-sensitive data is increasing in modern storage systems.

Private Message



Private Photo



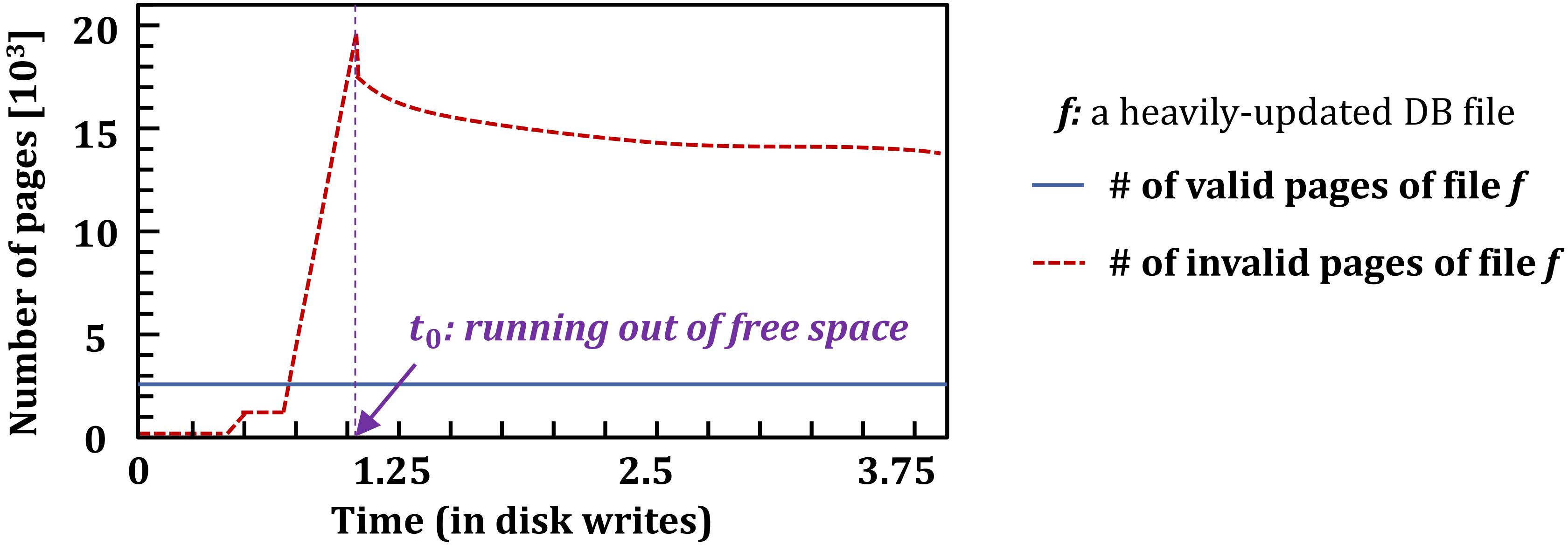


Once a user deletes security-sensitive data,
a storage system should guarantee its irrecoverability

Confidential Data (e.g., Medical Record)

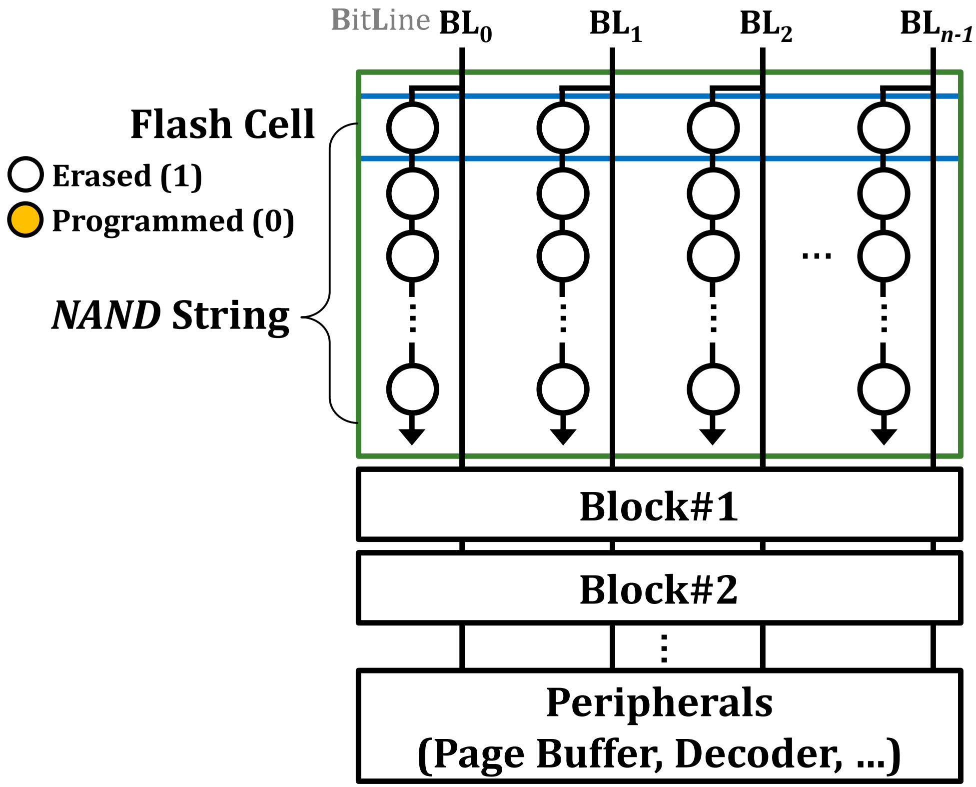
Data Versioning Problem

- Obsolete data in NAND flash-based solid-state drives (SSDs)
  - Old versions of updated or deleted files can remain in the SSD for a long time.



Updated or deleted data of a file can remain in SSDs due to unique features of NAND flash memory

NAND Flash Memory Organization & Operations

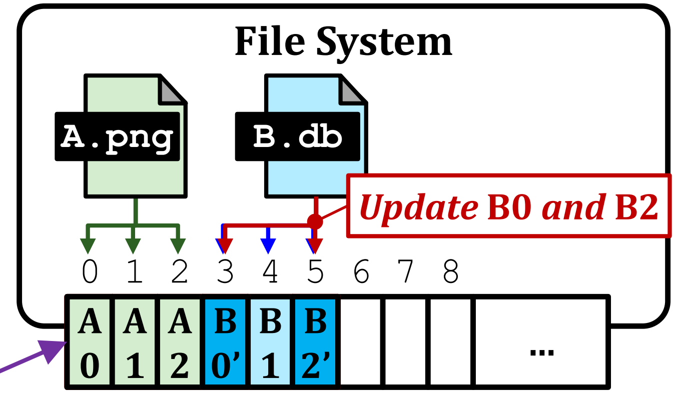


Page(s): Unit of read and program
(e.g., 131,072 cells: 16-KiB page)

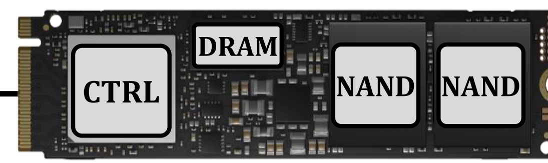
Block: Erase unit
(e.g., 576 pages: 9-MiB block)

***Erase-before-write*: A block needs to be erased before programming a page (i.e., no overwrite on a page)**

NAND Flash-Based SSD



Logical block-device view that supports overwrites



Flash-Based SSD

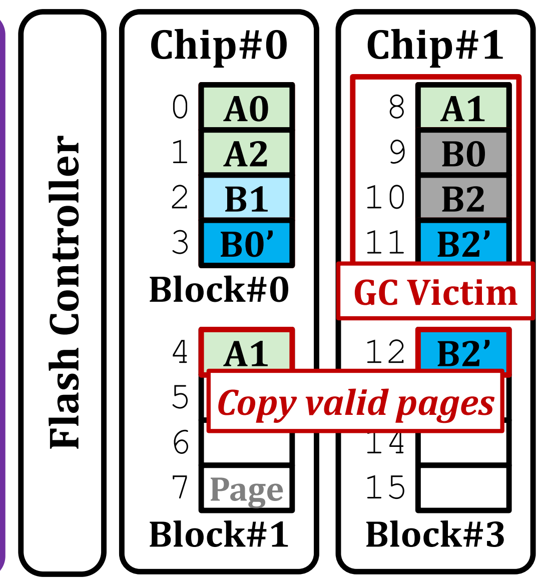
Flash Translation Layer (FTL)

- **Address translation**
  - Distributes host writes to fully exploit internal parallelism
  - Out-of-place updates
  - → **Logical-to-physical (L2P) mappings (e.g., LPA 1 → PPA 8)**

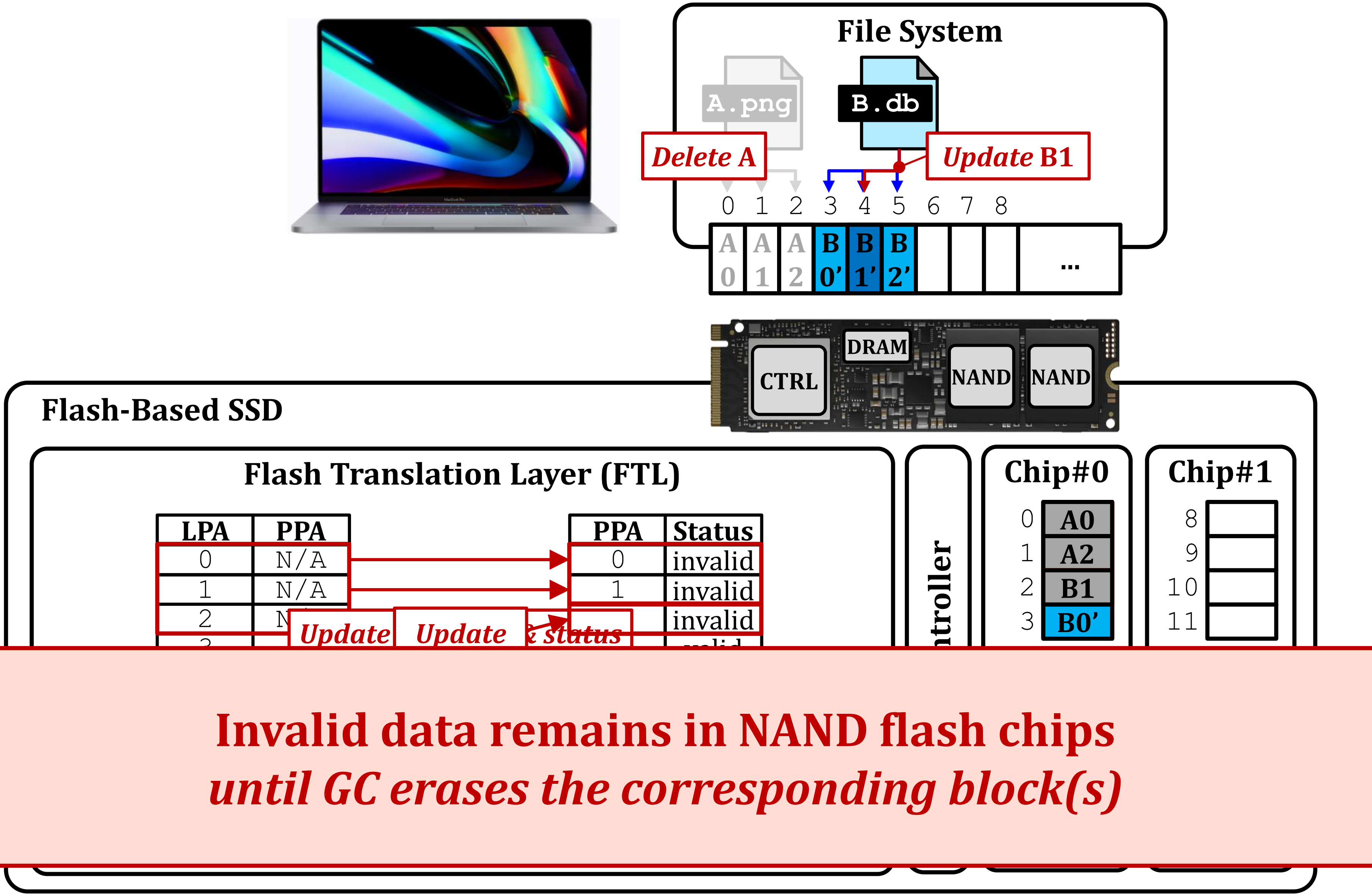
Logical Page Address

Physical Page Address

- **Garbage collection (GC)**
  - Reclaims free pages for future host writes
  - Selects a victim block w/ the smallest number of valid pages
  - Additional copy operations to move valid pages
  - → **Page-status information (e.g., B0: invalid)**



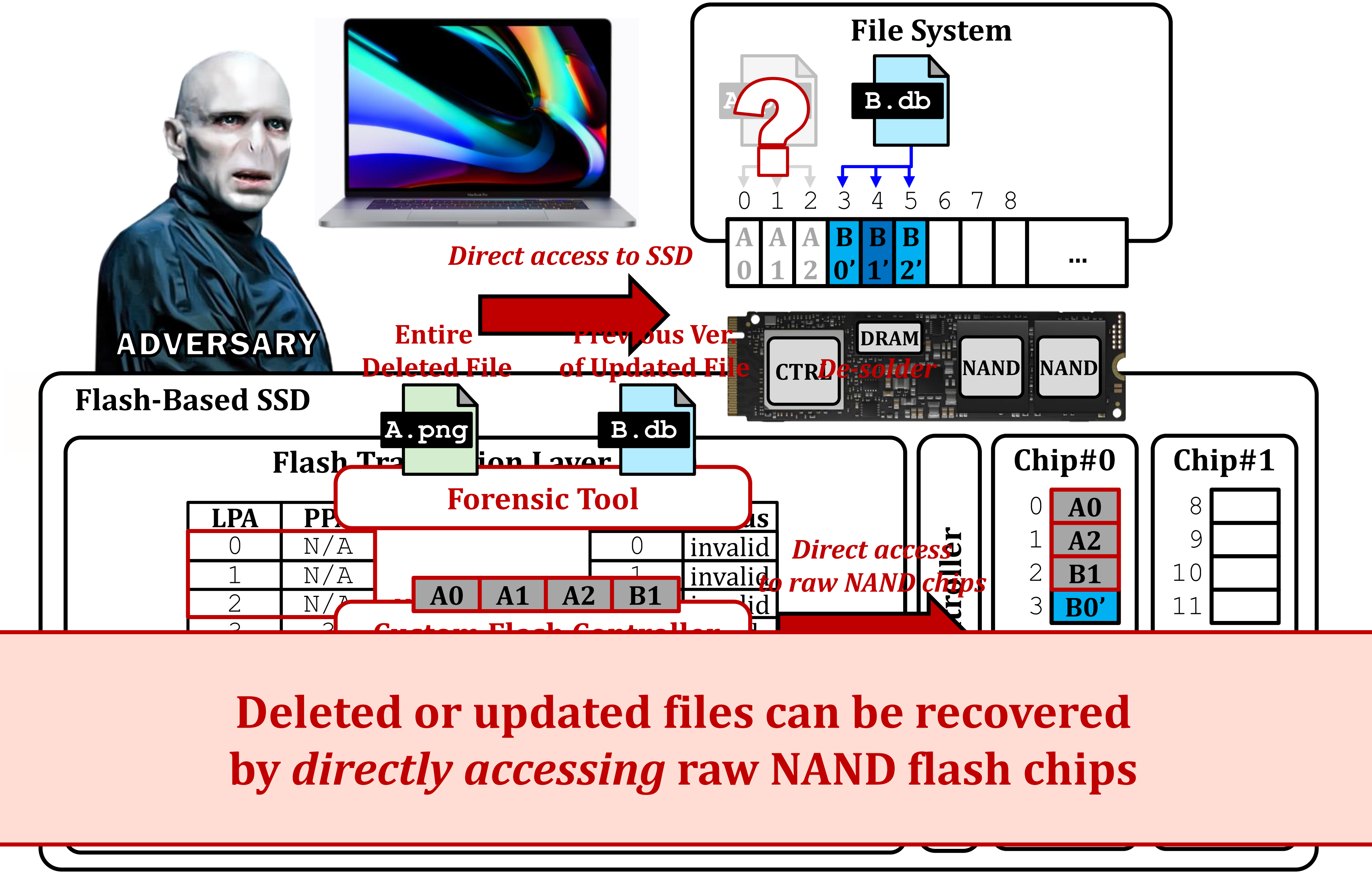
Data Deletion in NAND Flash-Based Storage Systems



Invalid data remains in NAND flash chips

until GC erases the corresponding block(s)

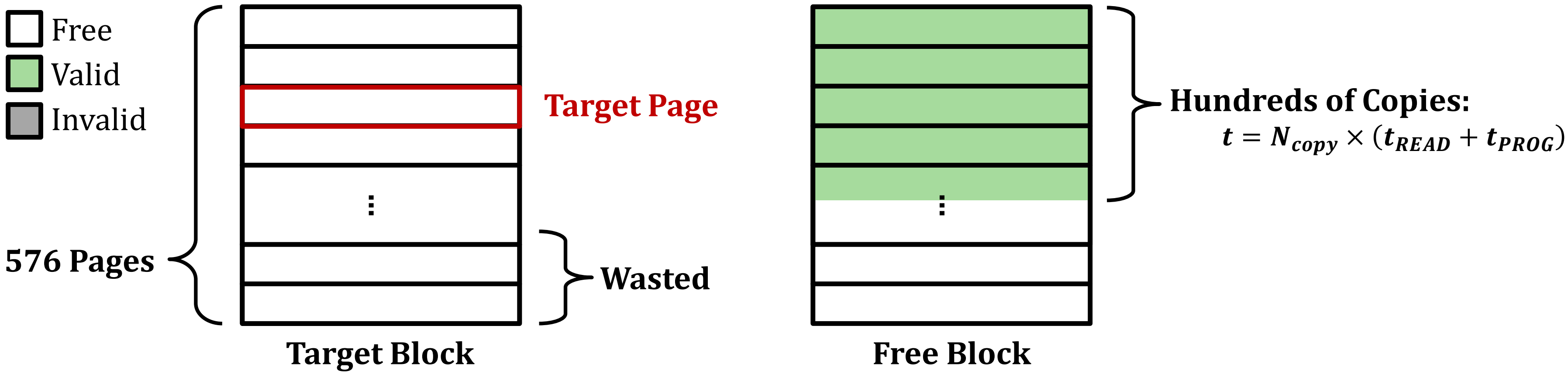
Security Vulnerability of NAND Flash-Based SSDs



Deleted or updated files can be recovered by *directly accessing* raw NAND flash chips

Existing Solution: Immediate Block Erasure

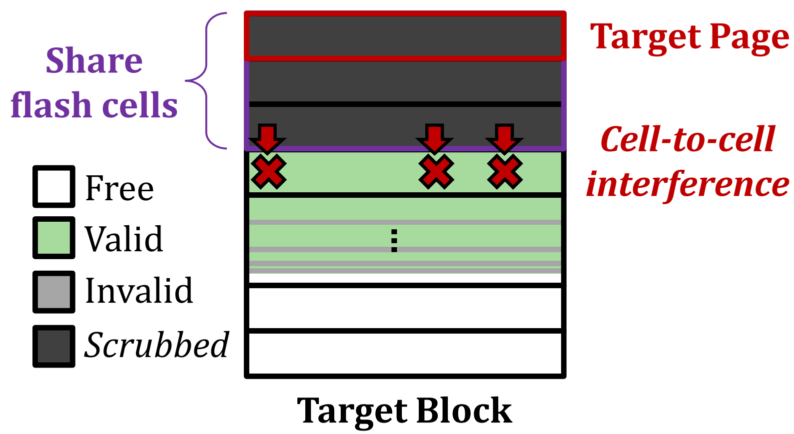
- Immediately erases the block that stores data to be sanitized
  - High performance and lifetime overheads due to *Erase-before-write property*
    - Needs to copy all the valid pages stored in the same block



Immediate block erasure:
High performance and lifetime overheads

Existing Solution: Reprogramming the Page

- Scrubbing [Wei+, FAST'2011]: Reprograms all the flash cells storing an invalid page
  - Destroys the page data w/o block erasure
  - Performance and lifetime overheads in *Multi-level cell* (MLC) NAND flash memory
    - Needs to copy all the valid pages stored in the same flash cells
  - Reliability issues: *cell-to-cell interference*



Existing solutions incur
***performance, lifetime, and reliability* problems**
in modern NAND flash memory

Outline

- Secure Deletion in NAND Flash-Based SSDs

- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD

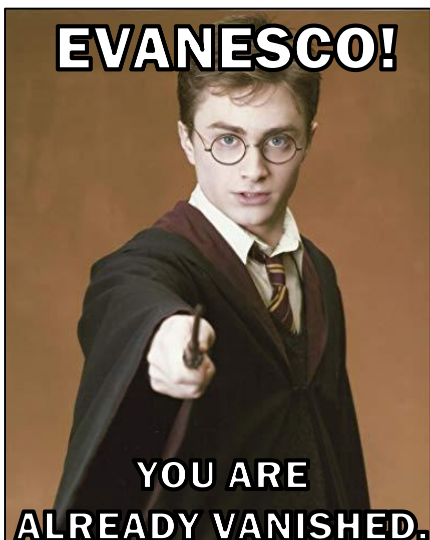
- Evaluation

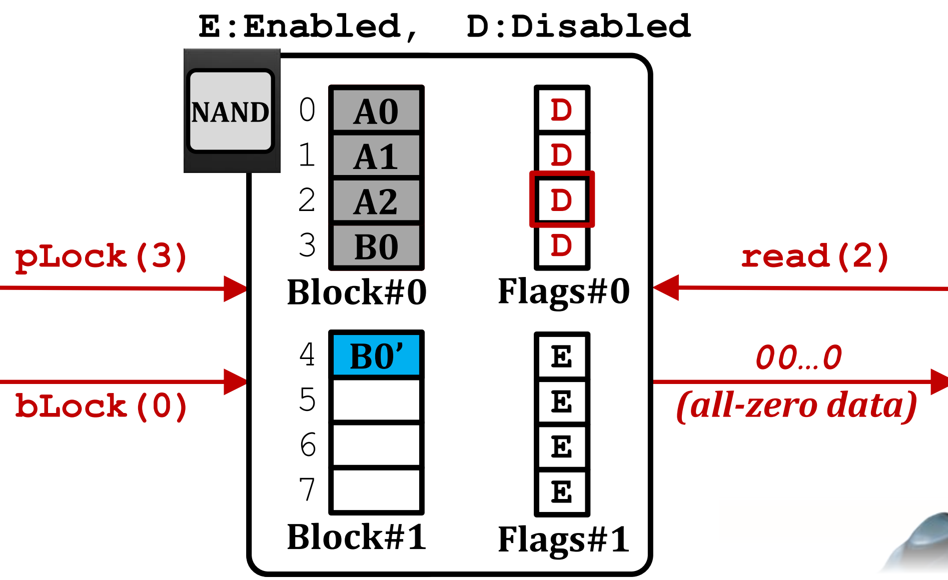
- Conclusion

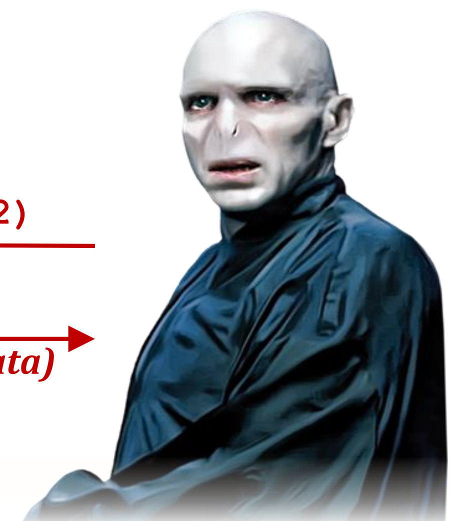
Evanesco: Access Control-Based Sanitization

- **Key idea:** Allow a NAND flash chip to be aware of data validity
  - Prevent access to invalid data at the chip level w/o destroying the data
    - → Low overhead: No copy operation to move valid pages stored in the same cells
    - → High reliability: No cell-to-cell interference to other valid pages

- Two new NAND flash commands: pageLock (pLock) and blockLock (bLock)
  - **pLock:** disables access to a page
  - **bLock:** disables access to all the page in a block







Outline

- **Secure Deletion in NAND Flash-Based SSDs**

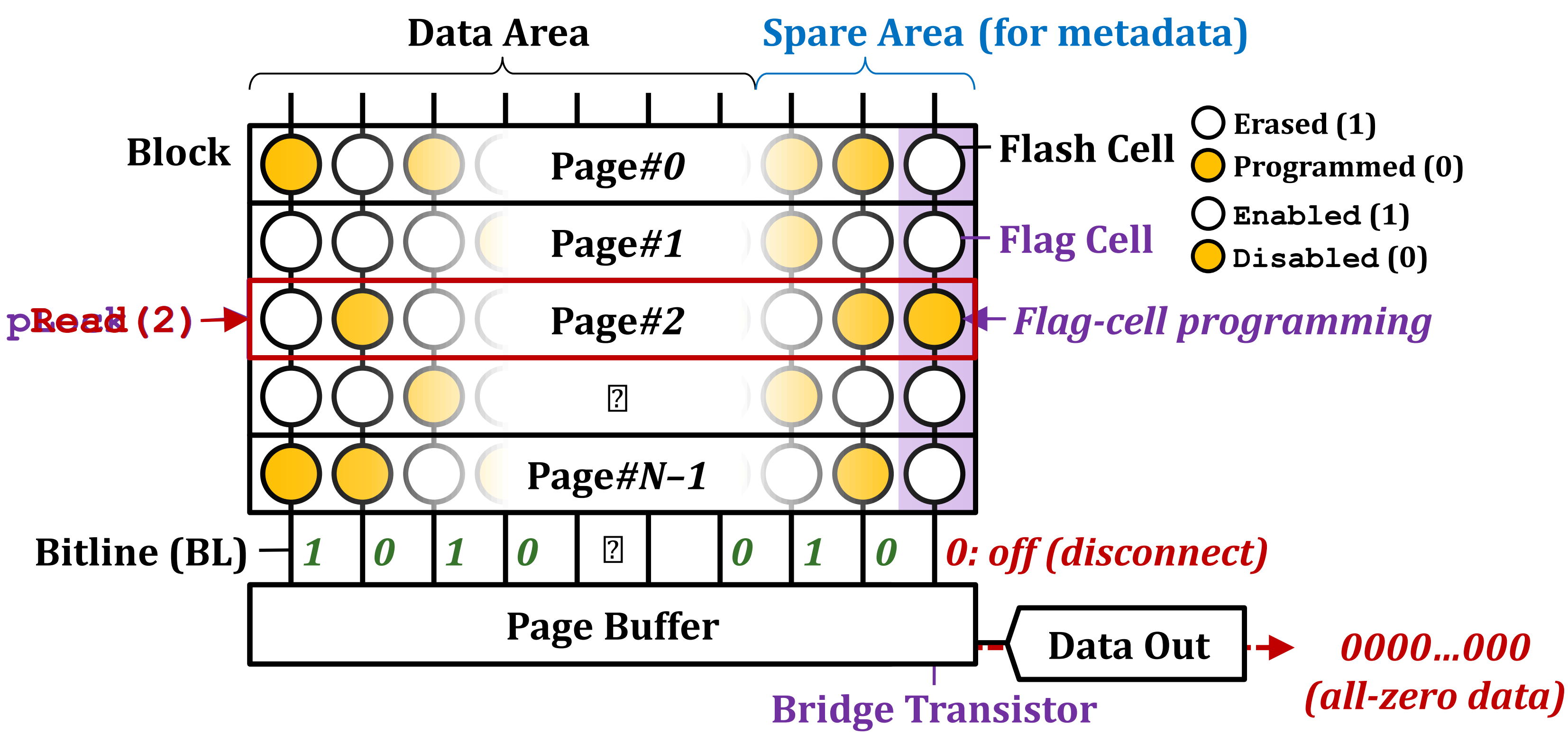
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD

- **Evaluation**

- **Conclusion**

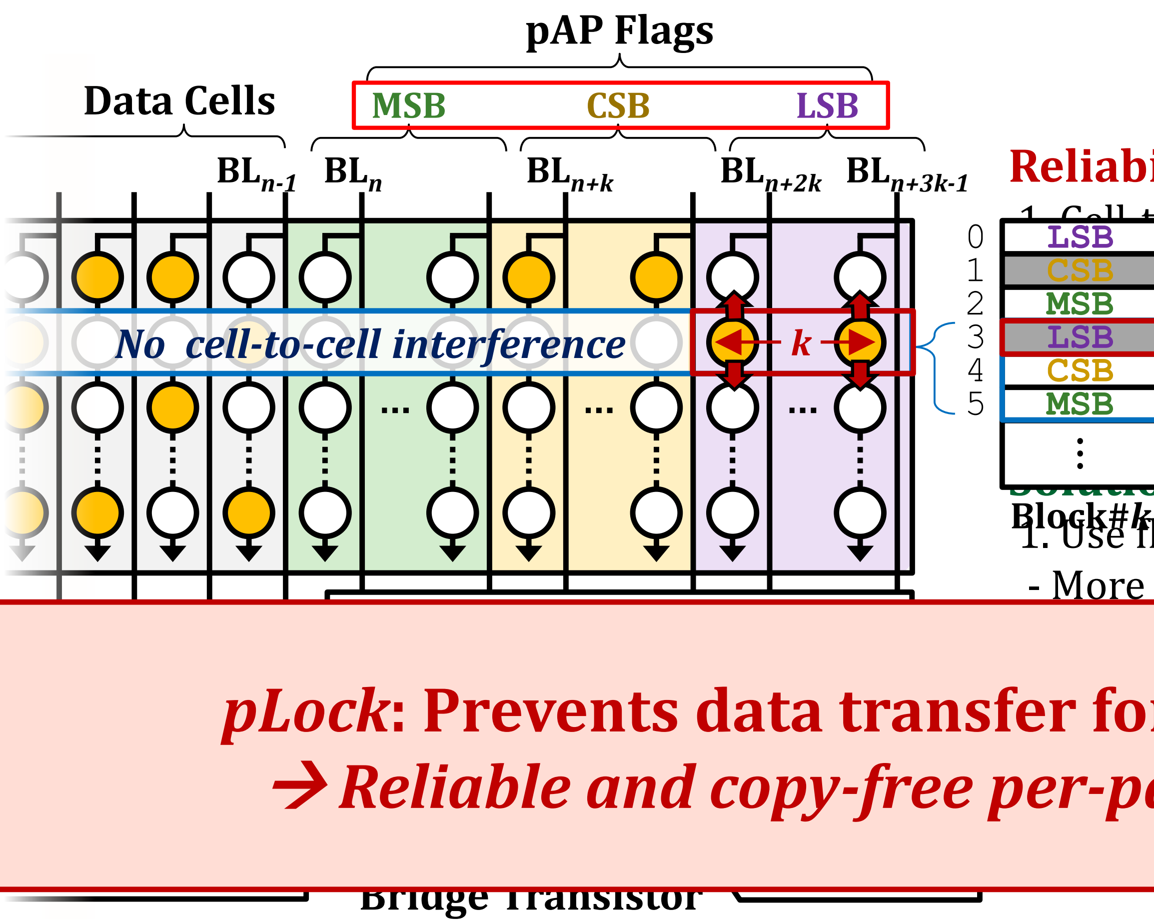
pLock: Page-Level Data Sanitization

- Implements page access-permission (pAP) flags using spare cells
  - A disabled page cannot be enabled until the entire block is erased.
  - No additional command to access a pAP flag: read with the page data at the same time
- Prevents transfer of data from a disabled page
  - The bridge transistor disconnects the page buffer from the data-out circuitry.



pLock: Implementation Details

- **Problem 1:** Multiple pages are stored in the same flash cell.
  - **Solution:** Use multiple flags for each row (e.g., 3 flags for triple-level cell (TLC) NAND)
- **Problem 2:** A flag cell can misbehave → unintentional disabling or enabling of a page
  - **Solution:** Use multiple flag cells for each pAP flag (*k*-modular redundancy scheme)



Reliability issues

1. Cell-to-cell interference b/w flag cells
   ... same NAND strings
   ...am disturbance due to high program
   ...e to the other cells at the same row

Solutions

1. Use flag cells in single-level cell (SLC) mode
   - More robust to interference and disturbance

***pLock*: Prevents data transfer for a disabled page**
→ *Reliable and copy-free per-page sanitization*

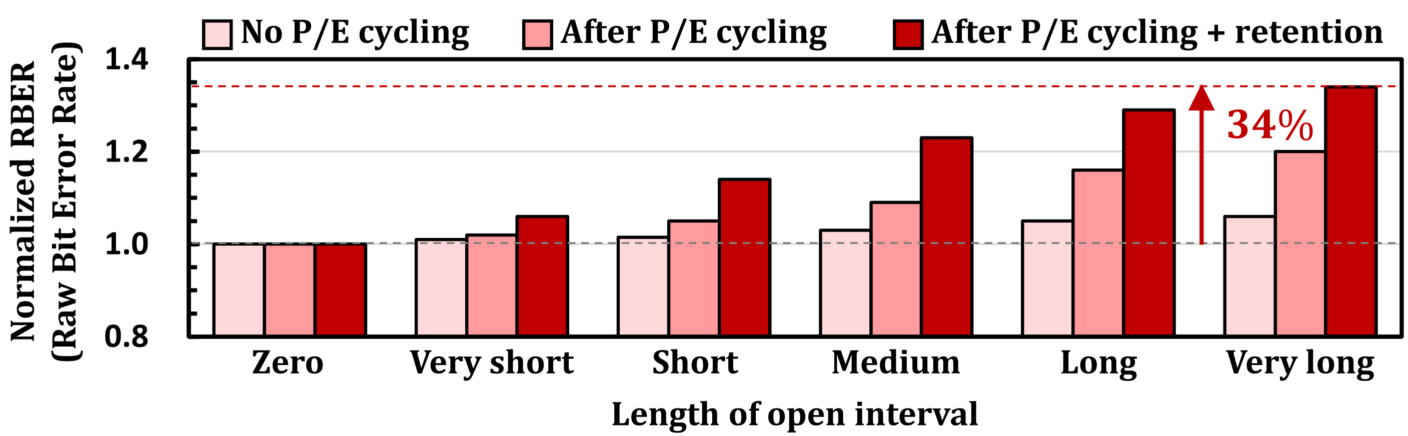
Outline

- **Secure Deletion in NAND Flash-Based SSDs**
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD
- **Evaluation**
- **Conclusion**

Problem with Page-Level Sanitization

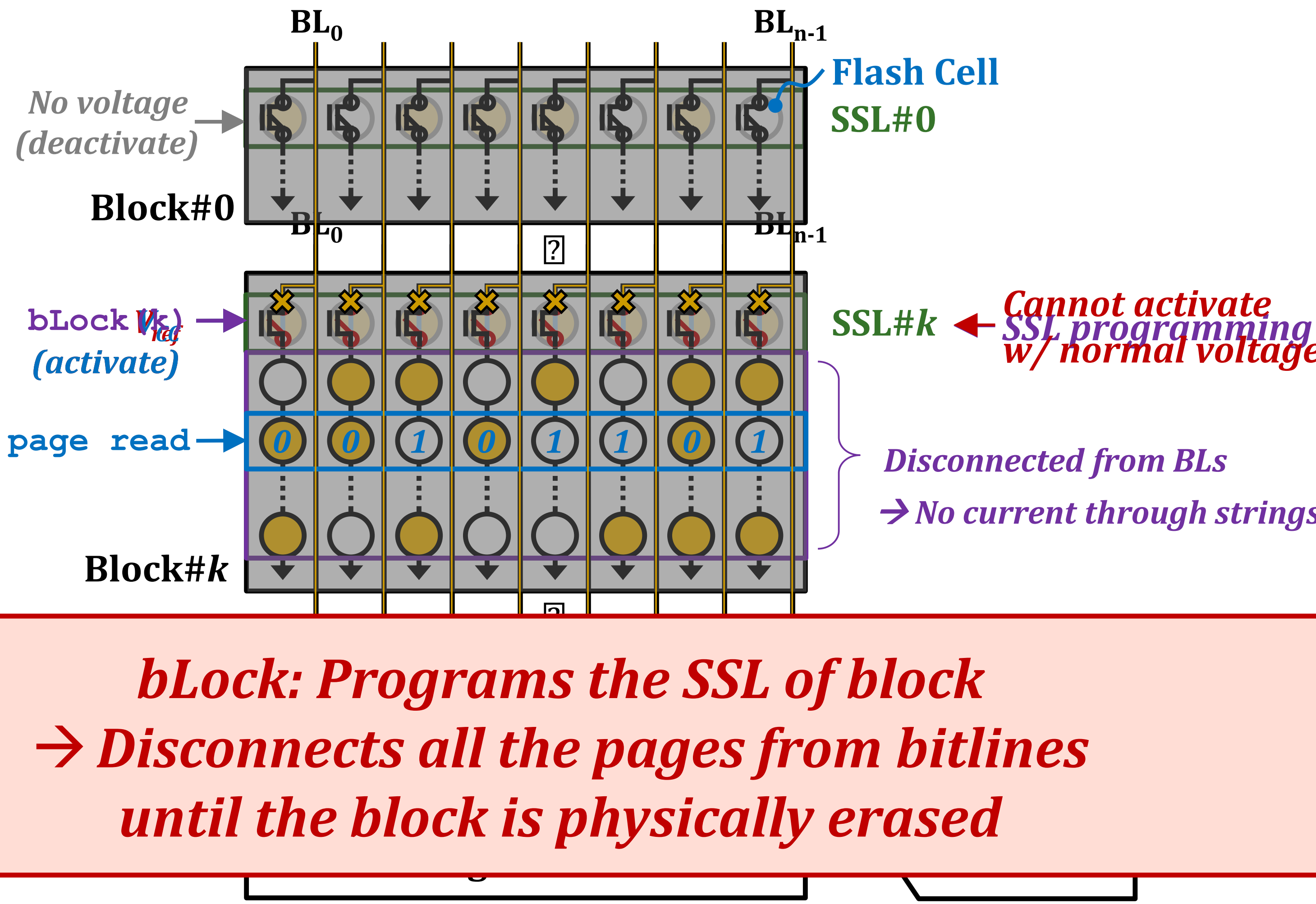
- Nontrivial performance overhead in invalidating an entire block
  - Deleting a 1-GiB video → 65,536 pLock operations (page size = 16 KiB)
  - Invalidating blocks in SSD management tasks (GC, wear-leveling, …)

- Immediate block erasure is not feasible in 3D NAND flash memory.
  - **Open-block problem:** Reliability degradation due to a long time interval b/w erasing and programming a block → A block should be erased *lazily*.



bLock: Block-Level Sanitization

- **Key idea:** Program the *string-select line* (SSL) of a block
  - 3D NAND flash memory implements an SSL using flash cells.
  - SSL programming: Disconnects all the pages from bitlines (i.e., from the page buffer)



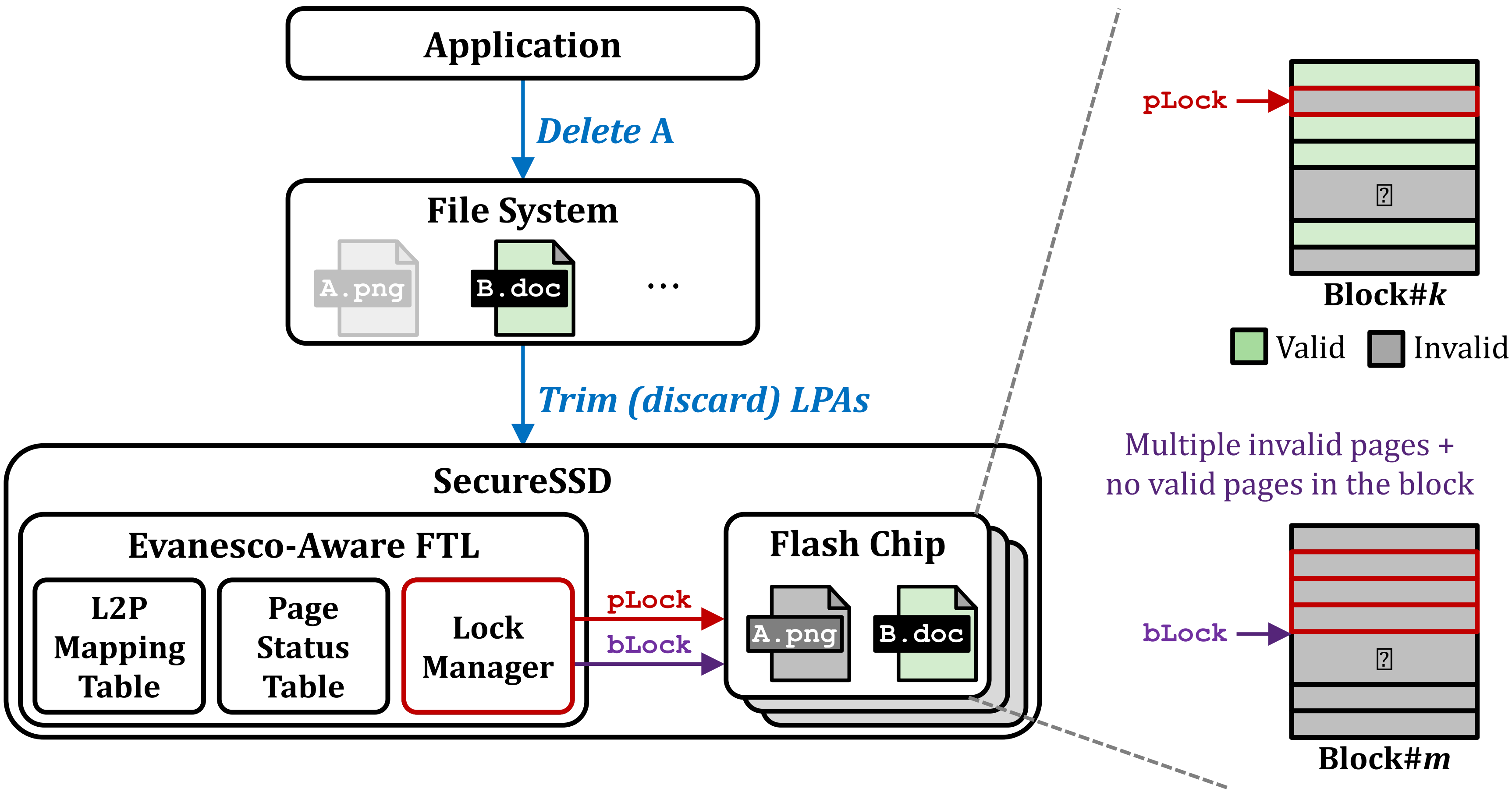
bLock: Programs the SSL of block
→ Disconnects all the pages from bitlines
until the block is physically erased

Outline

- **Secure Deletion in NAND Flash-Based SSDs**
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD
- **Evaluation**
- **Conclusion**

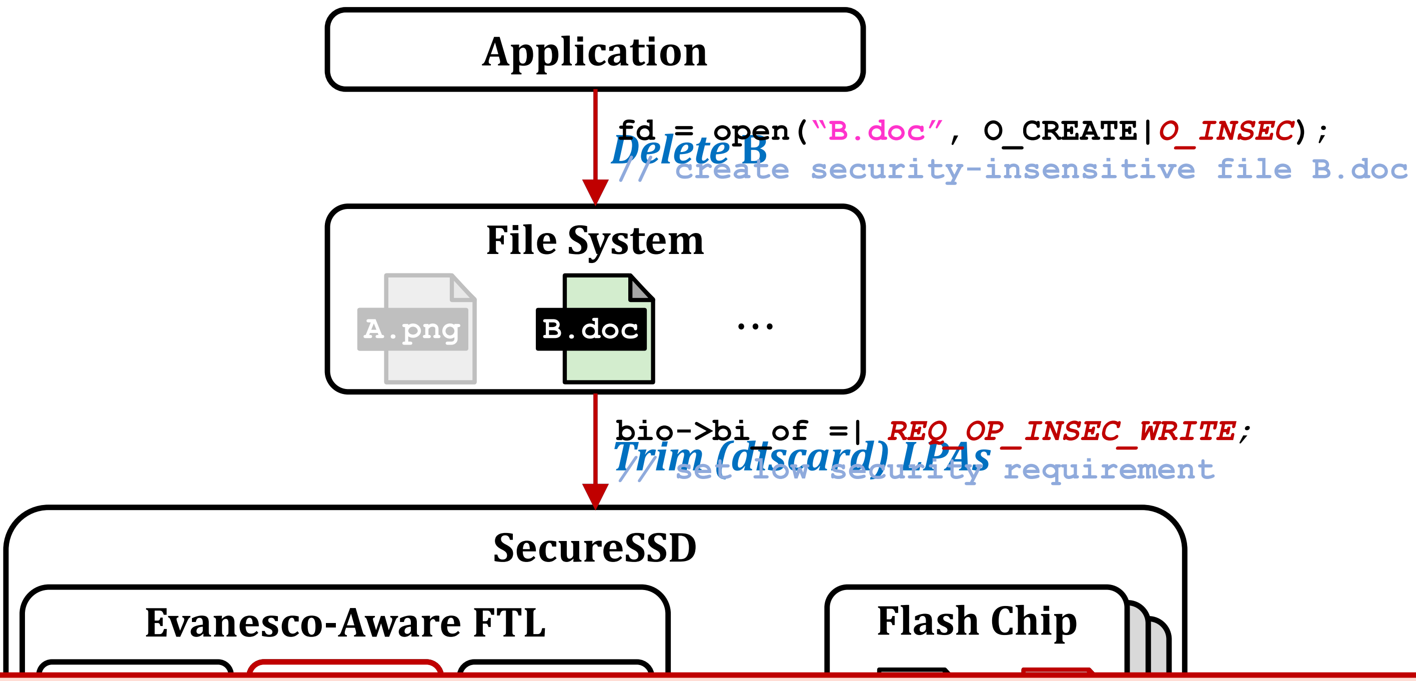
SecureSSD: An Evanesco-Enabled SSD

- An SSD that supports immediate data sanitization of updated or deleted data
  - Lock manager issues pLock and bLock commands depending on the block's status.



SecureSSD: Selective Data Sanitization

- SecureSSD avoids unnecessary pLock and bLock for security-insensitive data.
  - A user sets the security requirements of written data w/ extended I/O interfaces.



SecureSSD minimizes data-sanitization overheads

Outline

- **Secure Deletion in NAND Flash-Based SSDs**
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD
- **Evaluation**
- **Conclusion**

Methodology

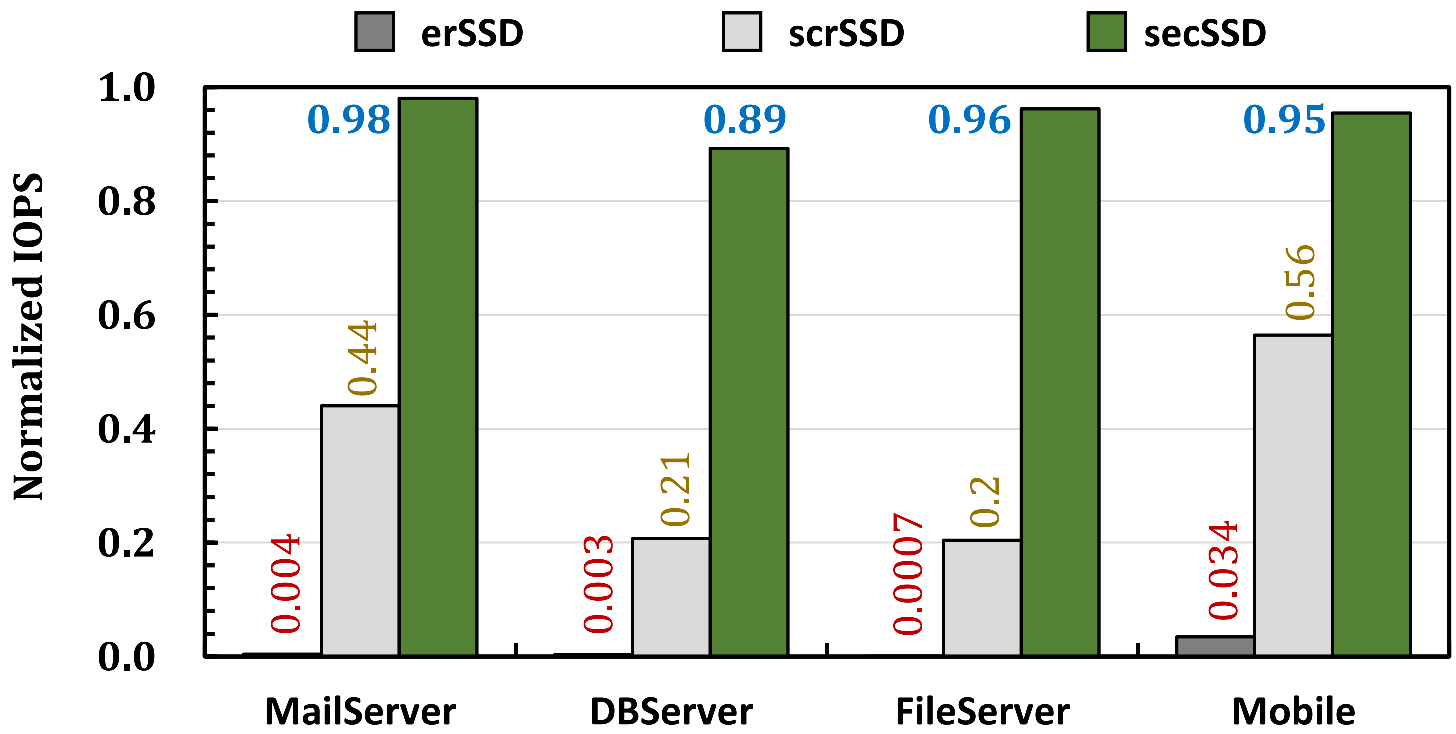
- **Design space exploration for pLock and bLock**
  - Using 160 real state-of-the-art 3D triple-level-cell (TLC) NAND flash chips
  - To find the best operation parameters w/o reliability degradation
    - **pLock:** 100-us latency w/ 9 flag cells per page
    - **bLock:** 300-us latency
    - tREAD = 100 us, tPROG = 700 us, tBERS = 3.5 ms

- **Simulator:** Open SSD-development platform (FlashBench [Lee+, RSP'2012])
  - 32-GiB storage capacity
  - 576 pages per block
  - 16-KiB page size

- **Compared SSDs**
  - **erSSD:** Erases the entire block after copying valid pages in the block
  - **scrSSD:** Performs scrubbing after copying valid pages in the same cells [Wei+, FAST'2011]

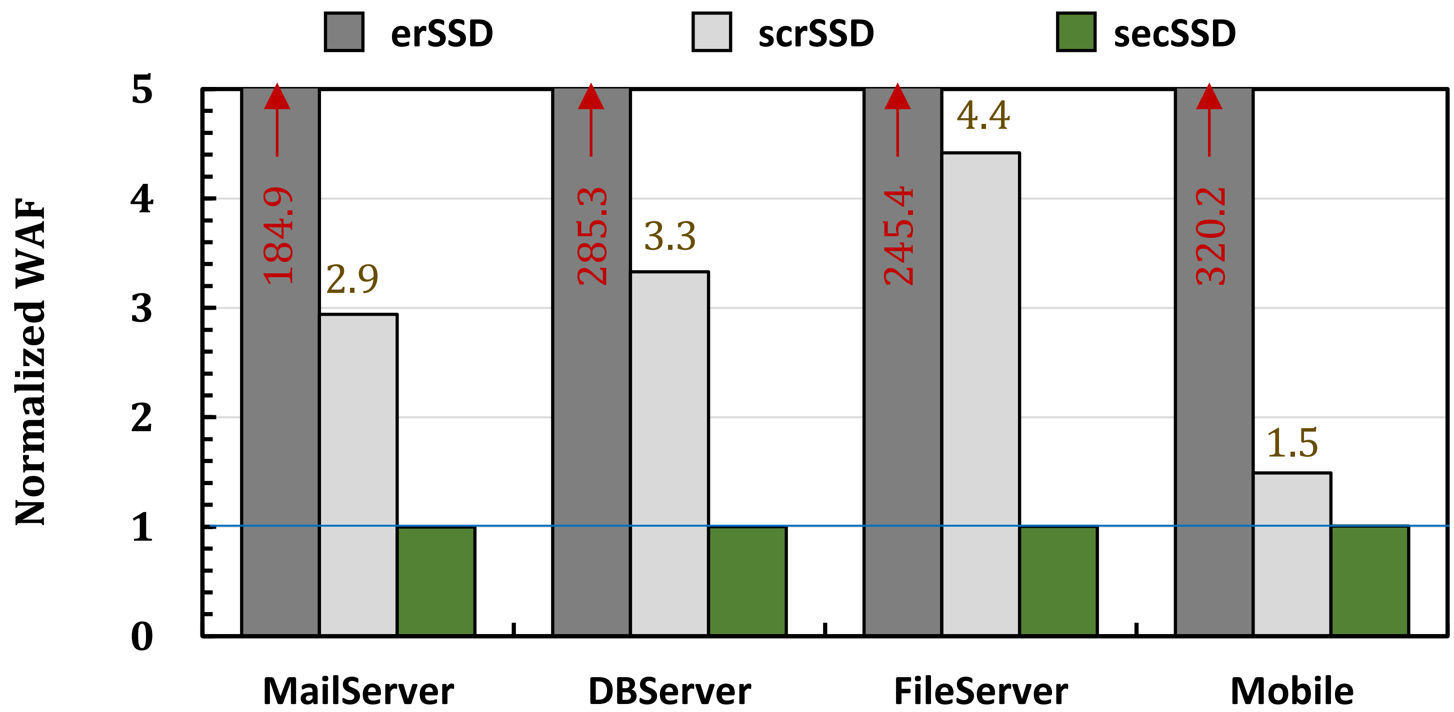
- **Workloads**
  - Three server workloads: MailServer, DBServer, FileServer
  - Mobile workload collected from an Android smartphone (Samsung Galaxy S2)

Results: Performance



SecureSSD significantly reduces performance overhead of data sanitization (11% slowdown at most)

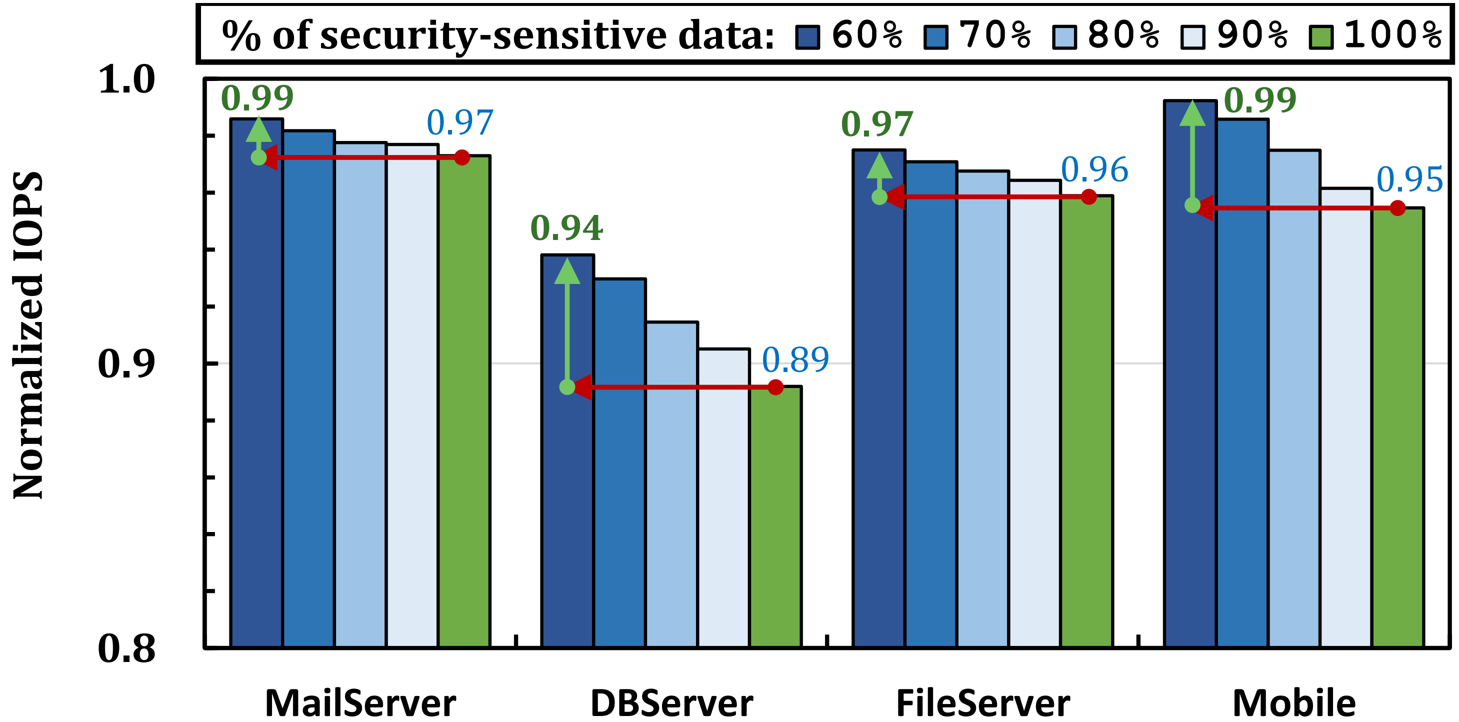
Results: Lifetime



$$Write\ Amplification\ Factor\ (WAF) = \frac{\#\ of\ logical\ pages\ written\ by\ the\ host\ system}{\#\ of\ physical\ pages\ written\ by\ the\ SSD}$$

No additional copy in SecureSSD: No lifetime overhead

Results: Effect of Selective Data Sanitization



Selective data sanitization minimizes performance overheads (6% slowdown at most with 60% security-sensitive data)

Other Analyses in the Paper

- **Empirical Study on Invalid Data in SSDs**
- **Reliability Issues in Physical Data Destruction**
- **Design Space Exploration for pLock and bLock**
- **Effectiveness of bLock command**

Outline

- **Secure Deletion in NAND Flash-Based SSDs**
- **Evanesco: Lock-Based Data Sanitization**
  - pageLock: Page-Level Data Sanitization
  - blockLock: Block-Level Data Sanitization
  - SecureSSD: An Evanesco-Enabled SSD
- **Evaluation**
- **Conclusion**

Conclusion

- **Challenges of data sanitization** in NAND flash-based SSDs:
  - **Erase-before-write property** → no overwrite on stored data
  - **Physical data destruction** → high performance & reliability overheads

- **Evanesco:** Uses on-chip access-control mechanisms
  - **pLock:** Page-level data sanitization
    - Implements the access-permission flag of each page using spare cells
  - **bLock:** Block-level data sanitization
    - Programs the SSL of a block to disconnect all pages
  - **SecureSSD:** An Evanesco-Enabled SSD
    - Supports selective data sanitization to reduce performance overheads

- **Results**
  - Provides the same level of reliability of an unmodified SSD
    - Validated w/ 160 real state-of-the-art 3D NAND flash chips
  - Significantly improves performance and lifetime over existing data-sanitization techniques
    - Provides comparable (94.5%) performance with an unmodified SSD