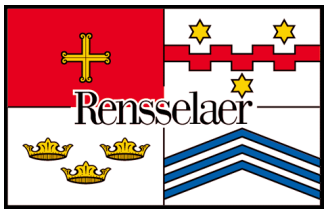
Leveraging Progressive Programmability of SLC Flash Pages to Realize Zero-overhead Delta Compression for Metadata Storage

Xuebin Zhang, Jiangpeng Li, Kai Zhao,
Hao Wang and Tong Zhang
xuebinzhang.rpi@gmail.com
ECSE Department, Rensselaer Polytechnic Institute (RPI)



Outline

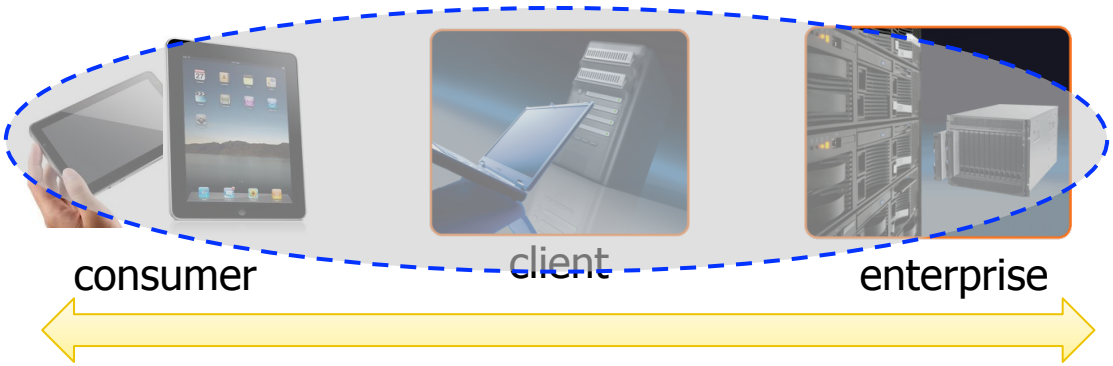
- Introduction and Background
- Progressive Programmability of SLC Flash Pages
- Metadata Update Solution using Delta Compression
- Evaluation Results and Practical Considerations
- Summary and Future Work

NAND Flash Memory

- Increasing Adoptions and Decreasing Cost of NAND Flash Memory



Increasingly wide adoption



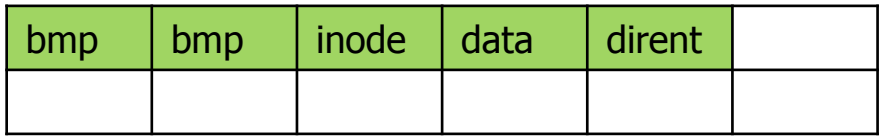
- Endurance: One of Existing Problems of NAND Flash Memory
  - Limited lifetime
  - Make it worse: Write amplification.

Why Metadata Storage

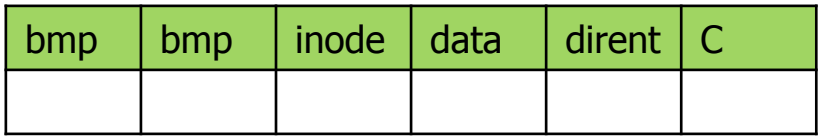
A simple example[1] in ext3:

- Write "hello" into text1.txt

  Effective data: 6 bytes vs. Flash write: 11 pages*4kB=44kB



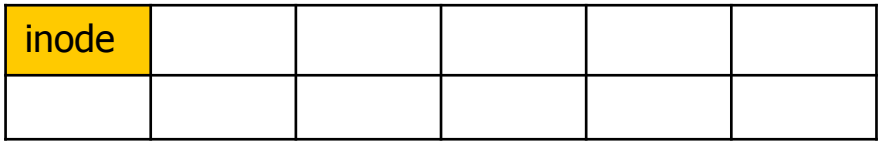
Data area



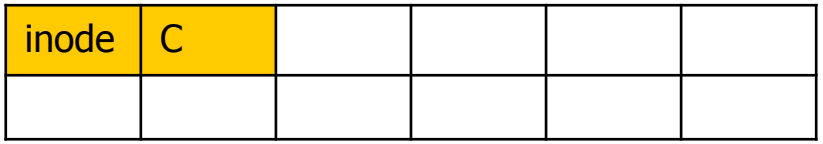
Journal area

- Read "happy" from text2.txt

  Flash write: 3pages*4kB=12 kB



Data area

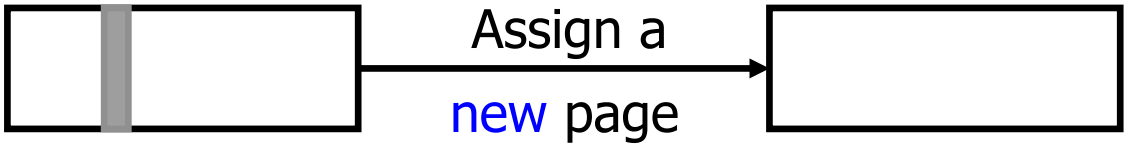


Journal area

[1] Y. Lu, J. Shu, and W. Zheng, "Extending the lifetime of flash based storage through reducing write amplification from file systems," Usenix FAST, 2013

Metadata Update Characteristics in NAND Flash

- Small (< 4KB) and Page-aligned
  - Around 80% write are smaller than 4KB, metadata are even smaller





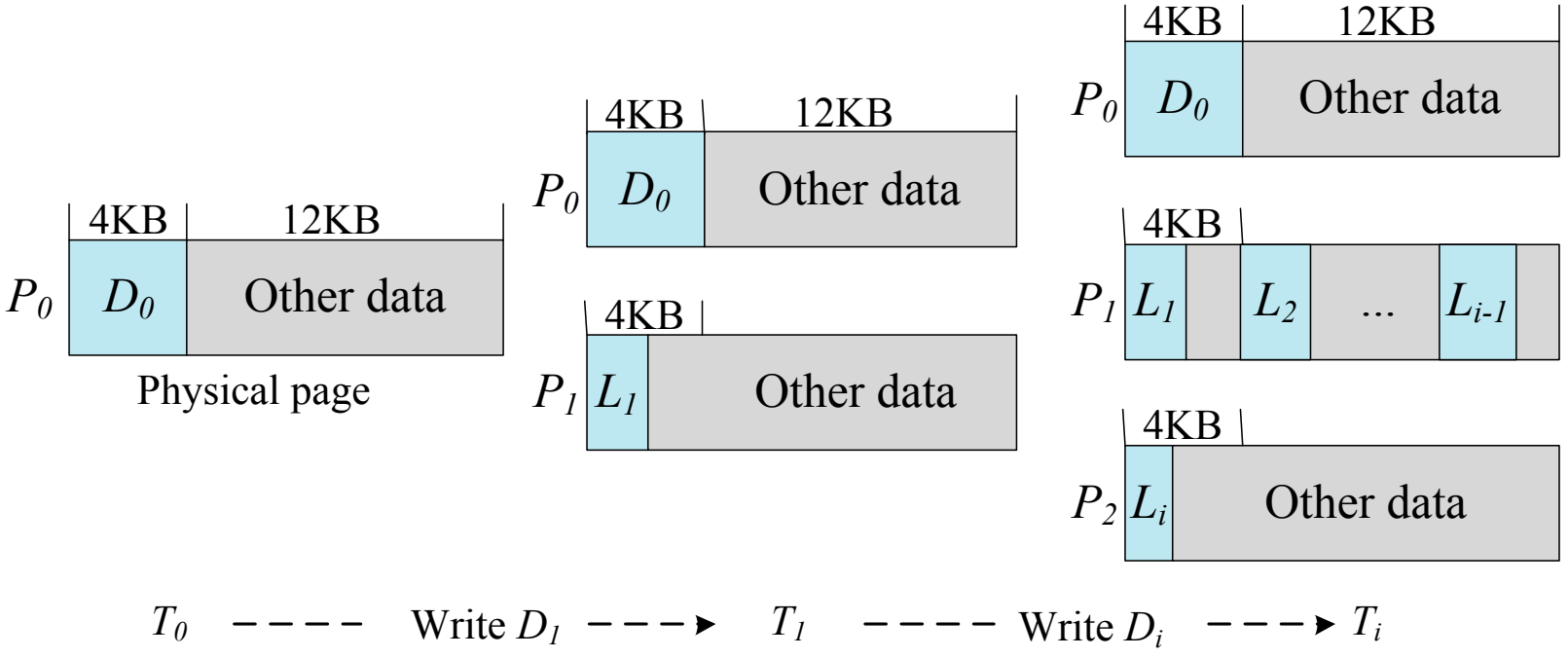
- Frequent update within a short time period (around 1s)

- Abundant redundancy between consecutive versions of metadata.
  - Metadata is highly structured.
  - Difference would always occur in the same locations.



Delta Compression

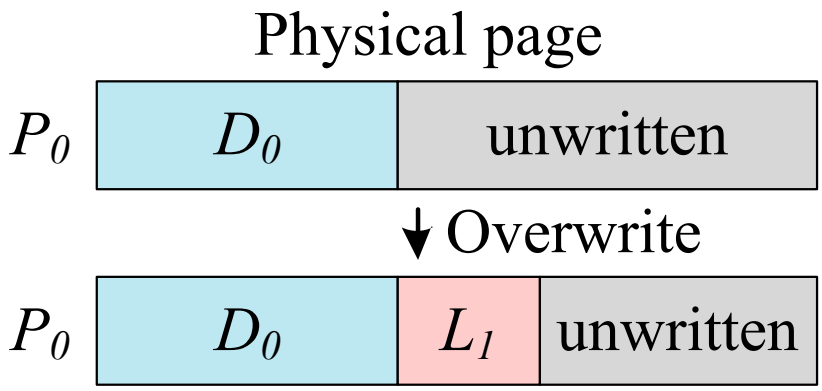
Delta Compression: Current Practice

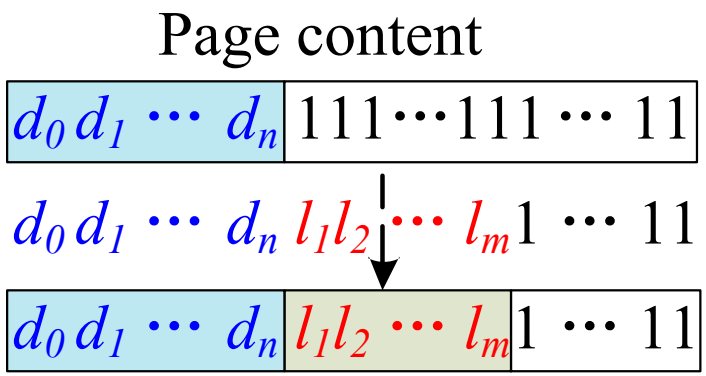


- Need to read multiple pages to recover the latest version.
- Need to keep a mapping table to store the related delta's location.

NAND Flash Progressive Programming

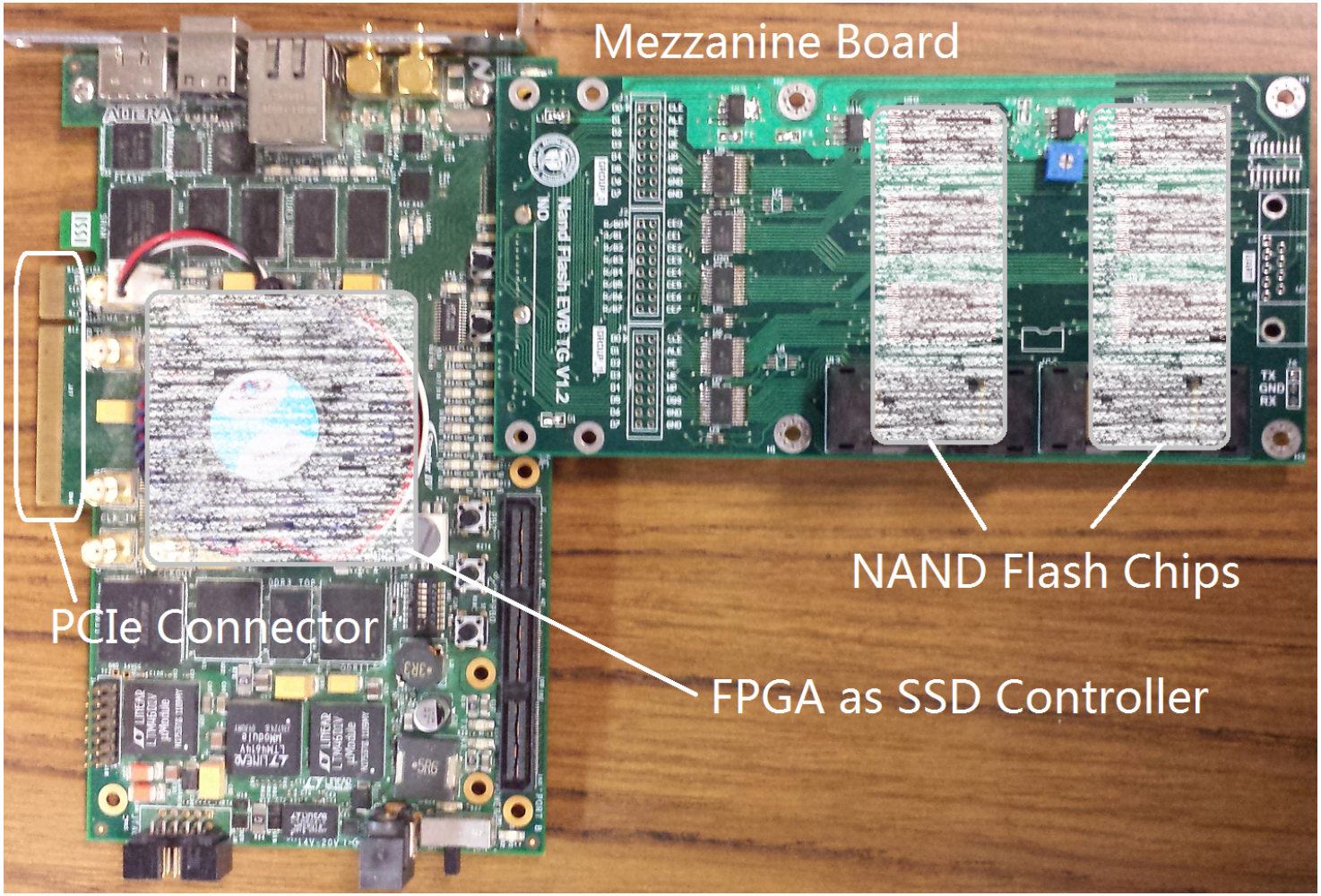
- Append new data in the same page (for SLC flash pages)
  1. Copy the original part
  2. Append new data
  3. Complement to a full page





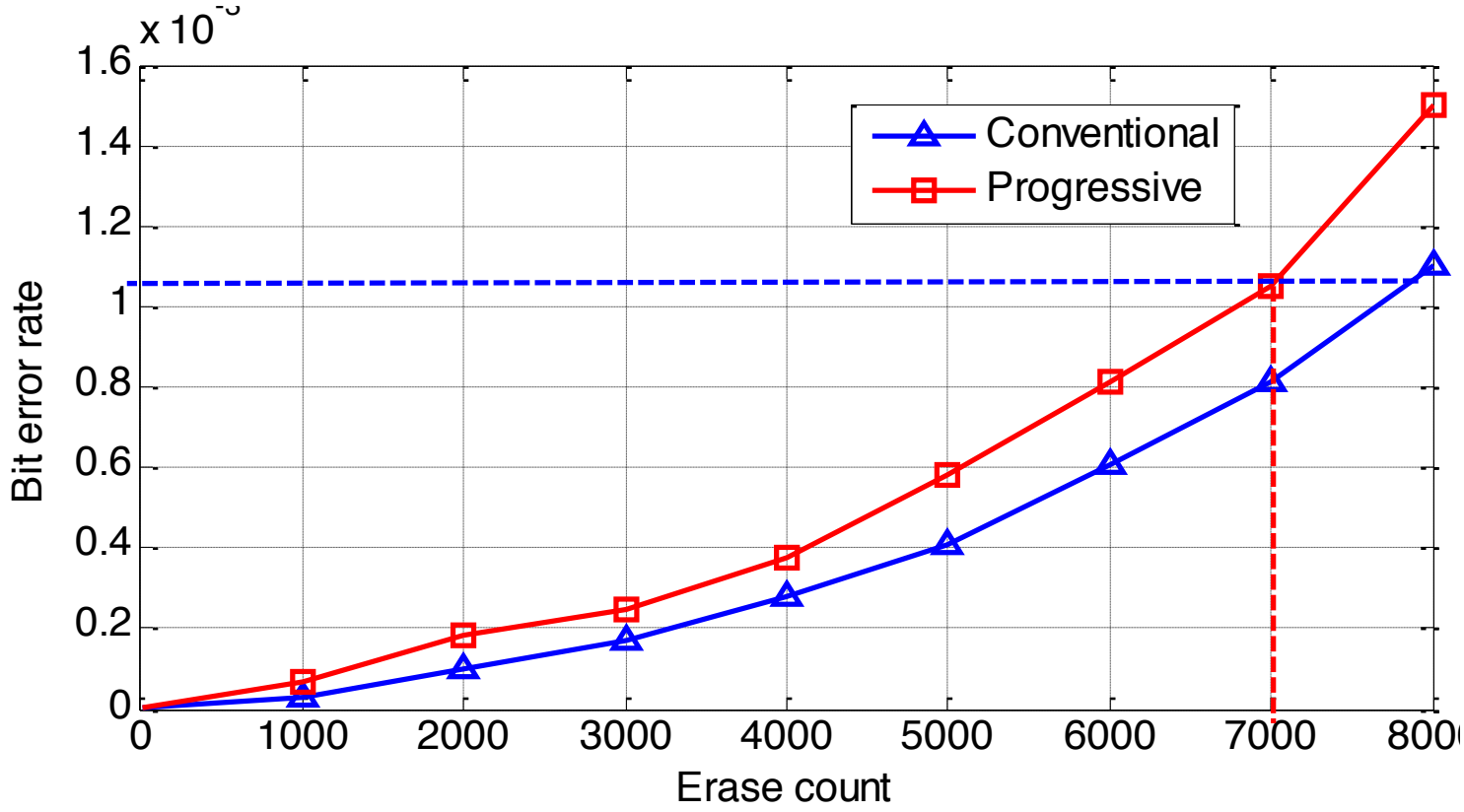
NAND Flash Progressive Programming

- Hardware Platform



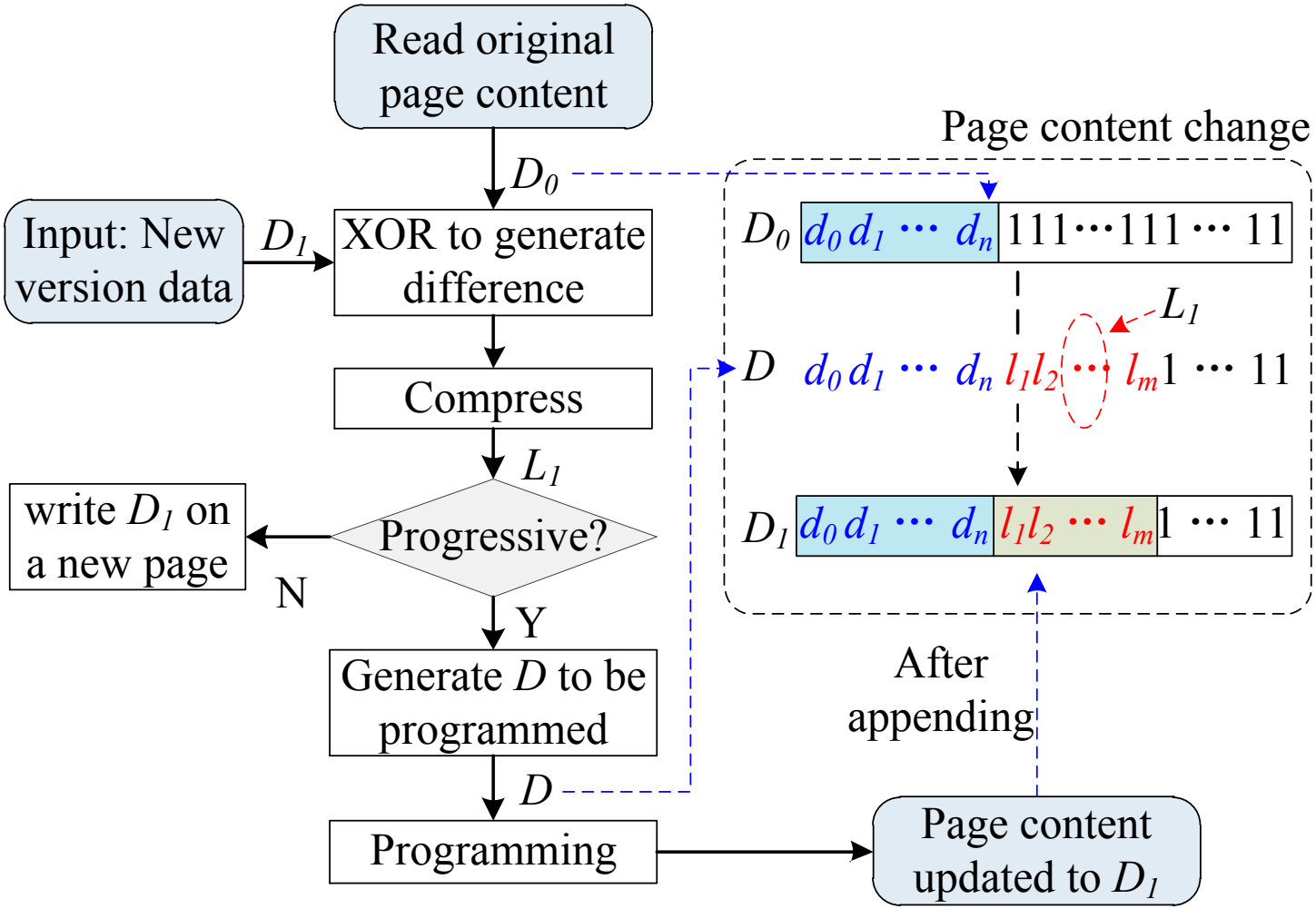
NAND Flash Progressive Programming

- Validity checking of multiple programming before erasing
  - Conventional: one program – one erase
  - Progressive: eight program – one erase



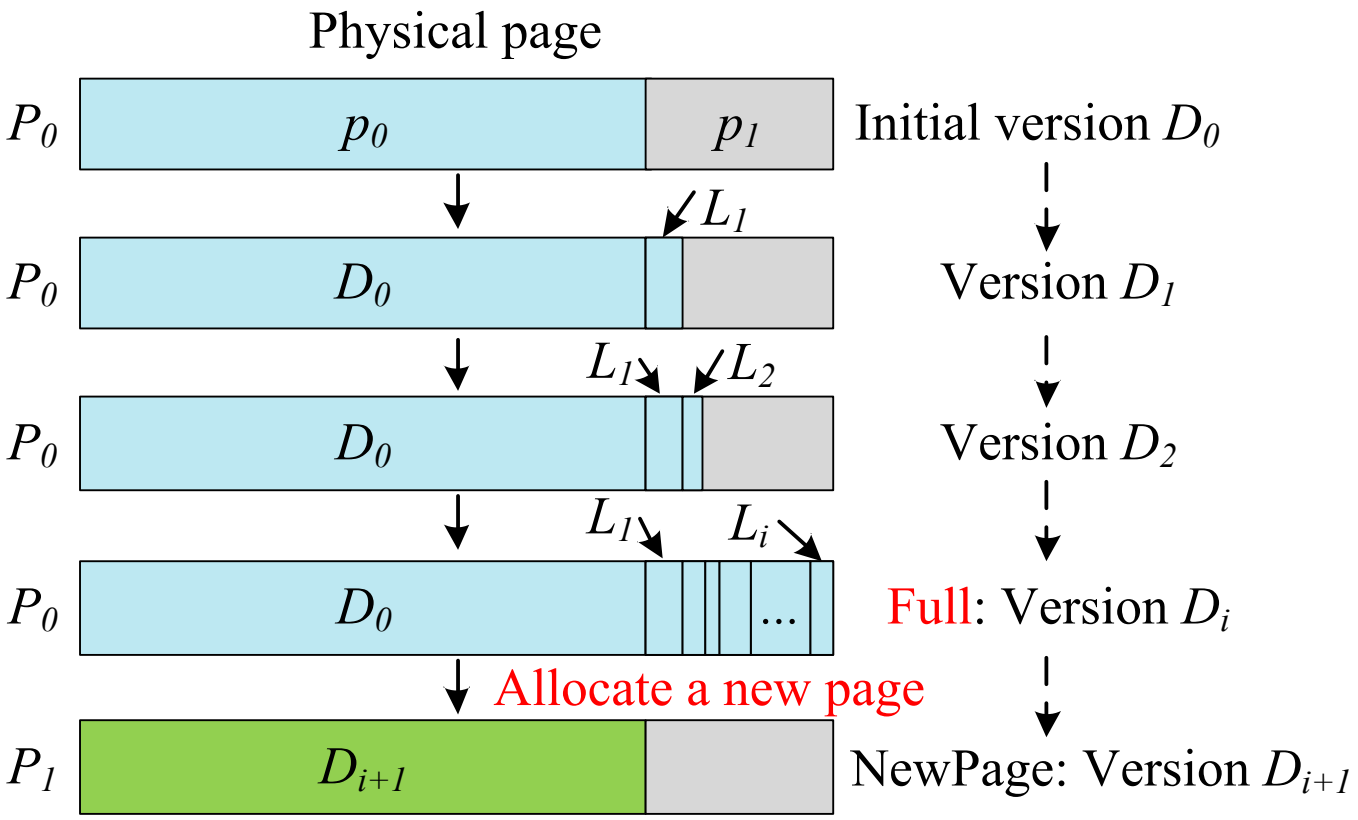
Use SLC to Realize Zero-overhead Metadata Storage

- Procedures to update new version data



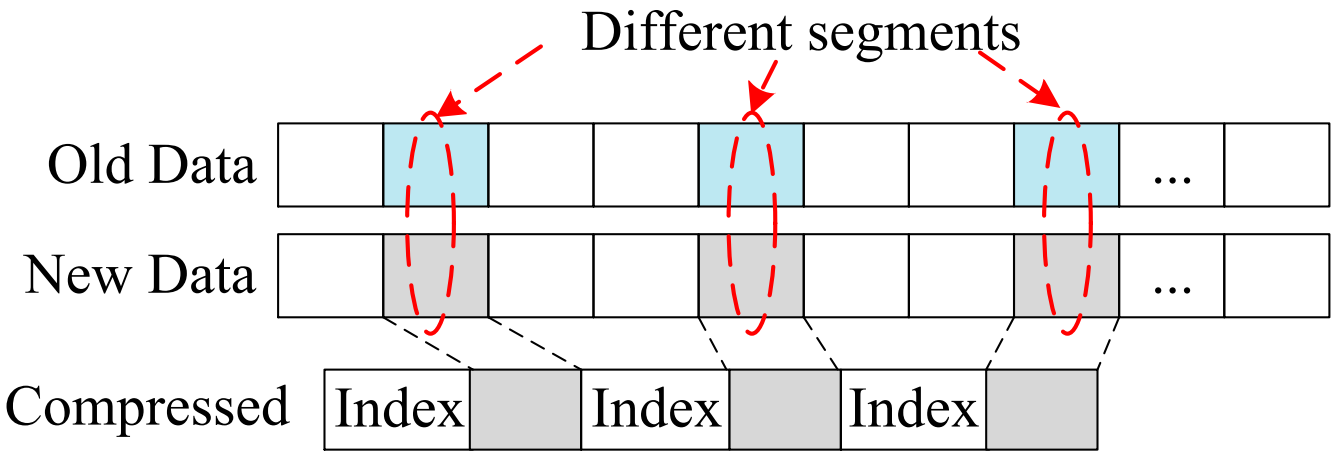
Use SLC to Realize Zero-overhead Metadata Storage

- Procedures to update new version data



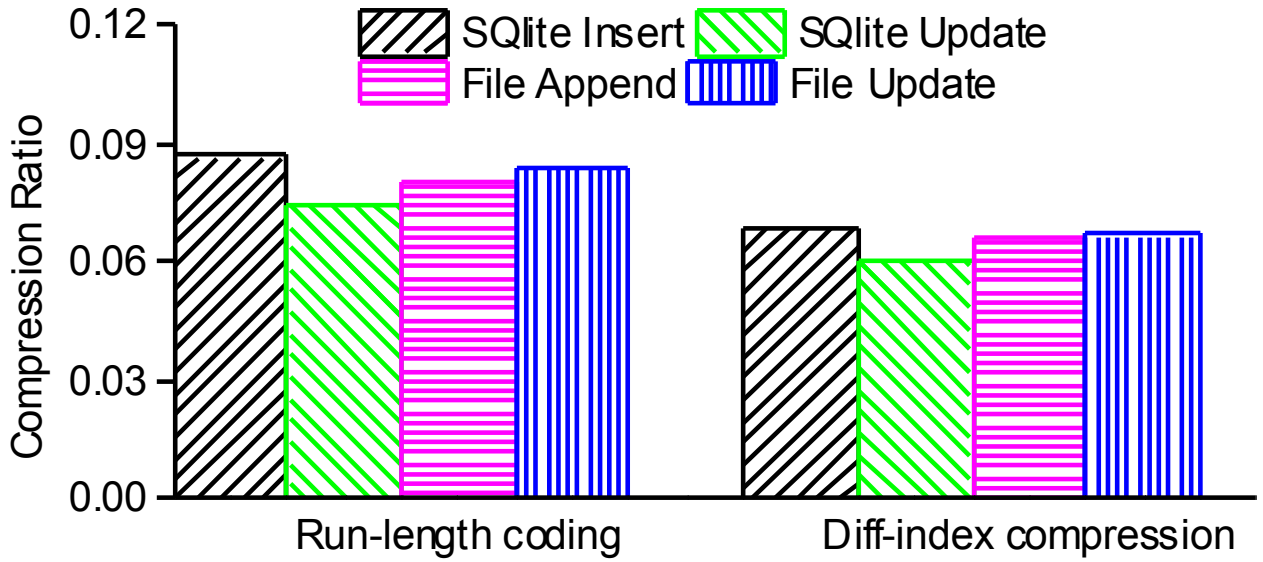
Implement the Delta Compression

- Bit level Run-length compression
- Difference-Index compression



Evaluation Results

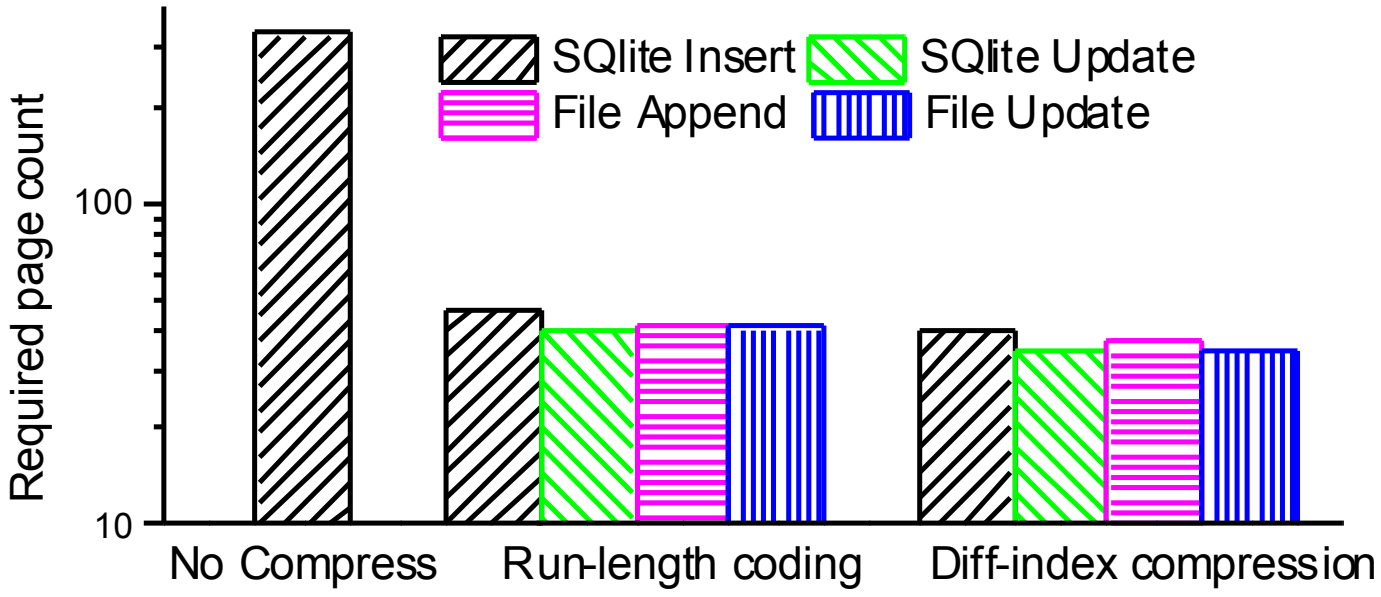
- Experiment Setup
  - Use benchmark to generate database/files operations
  - Implement a metaAnalyzer to grasp metadata from file system
  - Analyze the collected consecutive versions of metadata
- Compression Efficiency



Evaluation Results

- Write Footprint Reduction of Metadata

  To store 1000 consecutive versions:



Practical Considerations

- Read latency brought by decompression
  - Need multiple decompression to retrieve the latest version.
  - Decompression latency is much smaller(4us) than read latency(41us).
  - Decompression could be done by an ASIC module in controller.
- Changes on ECC management
  - Length of compressed delta should be written in the page.
- SLC mode space overhead compared to MLC mode
  - When multiple deltas are stored in the page, space overhead will be compensated.

Use SLC to Realize Zero-overhead Metadata Storage

- Conclusion
  - SLC Flash page can support "Progressive Programming": different portions of the same flash page can be programmed at different time.
  - Proposed solution can save data traffic written into disk.
  - Proposed solution can guarantee we do not need to read extra pages to retrieve the latest version.