Deduplication in SSDs: Model and Quantitative Analysis

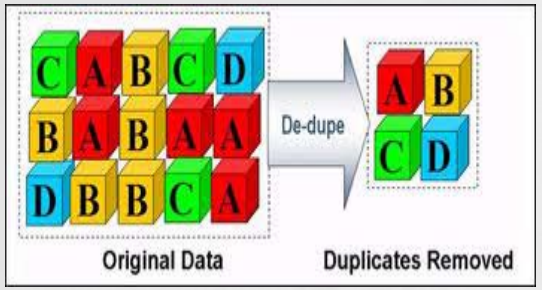
Jonghwa Kim, Sangyup Lee, Ikjoon Son, Jongmoo Choi, *Dankook University, Korea*
ChoongHyun Lee, *Massachusetts Institute of Technology*
Sungroh Yoon, *Korea University, Korea*
Hu-ung Lee, Sooyong Kang, Youjip Won, Jaehyuk Cha, *Hanyang University, Korea*

Contents

- Introduction
- Deduplication Overview
- Architecture for Deduplication on SSD
- Cost-benefit Model
- Three Acceleration Techniques
  - SHA-1 hardware logic (H/W approach)
  - Sampling based Filtering (S/W approach)
  - Recency based Fingerprint Management (S/W approach)
- Experimental Environments
- Experimental Results
- Conclusion

Introduction

- Deduplication
  - Remove duplicate writes
  - Reduce storage utilization
  - Improve I/O performance

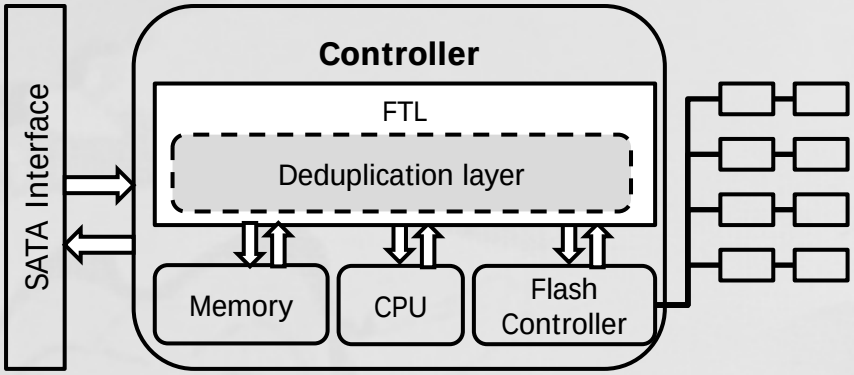


- Additional Benefits of Deduplication in SSD
  - Improve garbage collection overhead (especially highly utilized cases)
  - Enhance lifetime by reducing WAF (Write Amplification Factor)
  - Eliminate mapping overhead for deduplication using FTL

- Issues
  - Deduplication Overhead
    - How much is the deduplication overhead in SSD, especially considering the limited resources of SSD?
  - Duplication Rate
    - Are there enough duplicate data for obtaining performance gain in SSD workloads?

Architecture for Deduplication

- SSD overview

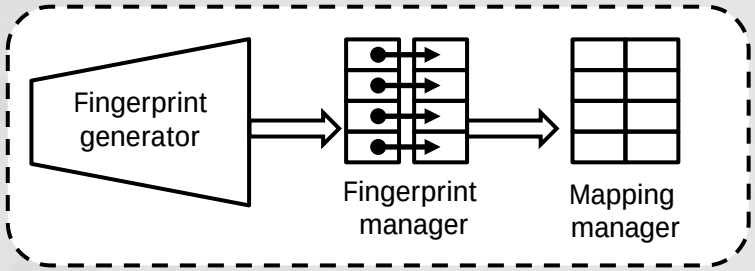


a) Conceptual structure of SSDs

- Main Components of SSD
  - SATA host interface
  - SSD controller
  - An array of flash chips.
- Characteristics of Flash Memory
  - Erase-before-write
  - Limited number of program/erase cycles.
- FTL (Flash Translation Layer)
  - Out-of-place update and wear-leveling mechanism

Architecture for Deduplication

- Internals of Deduplication Layer

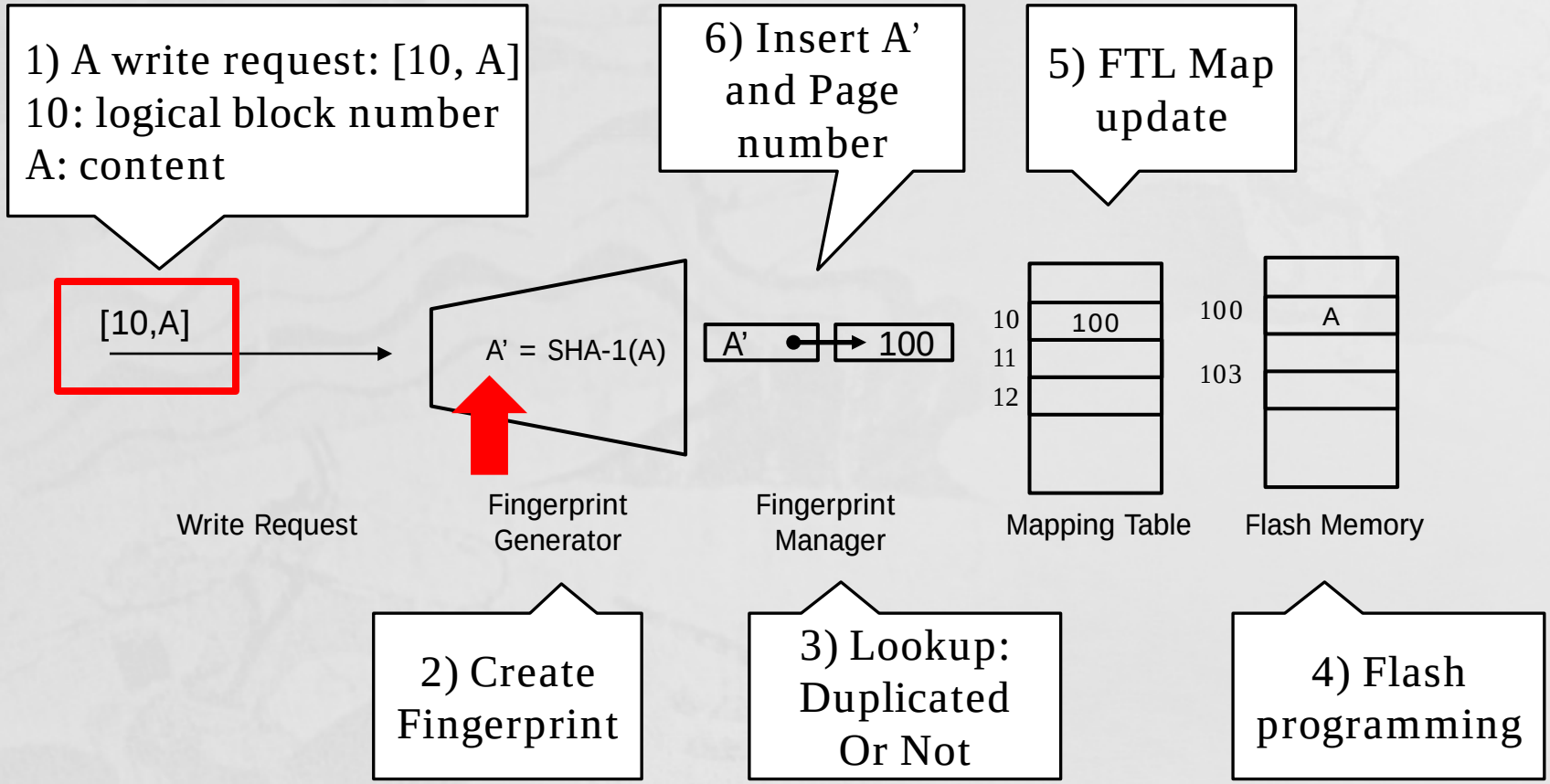


b) Deduplication layer

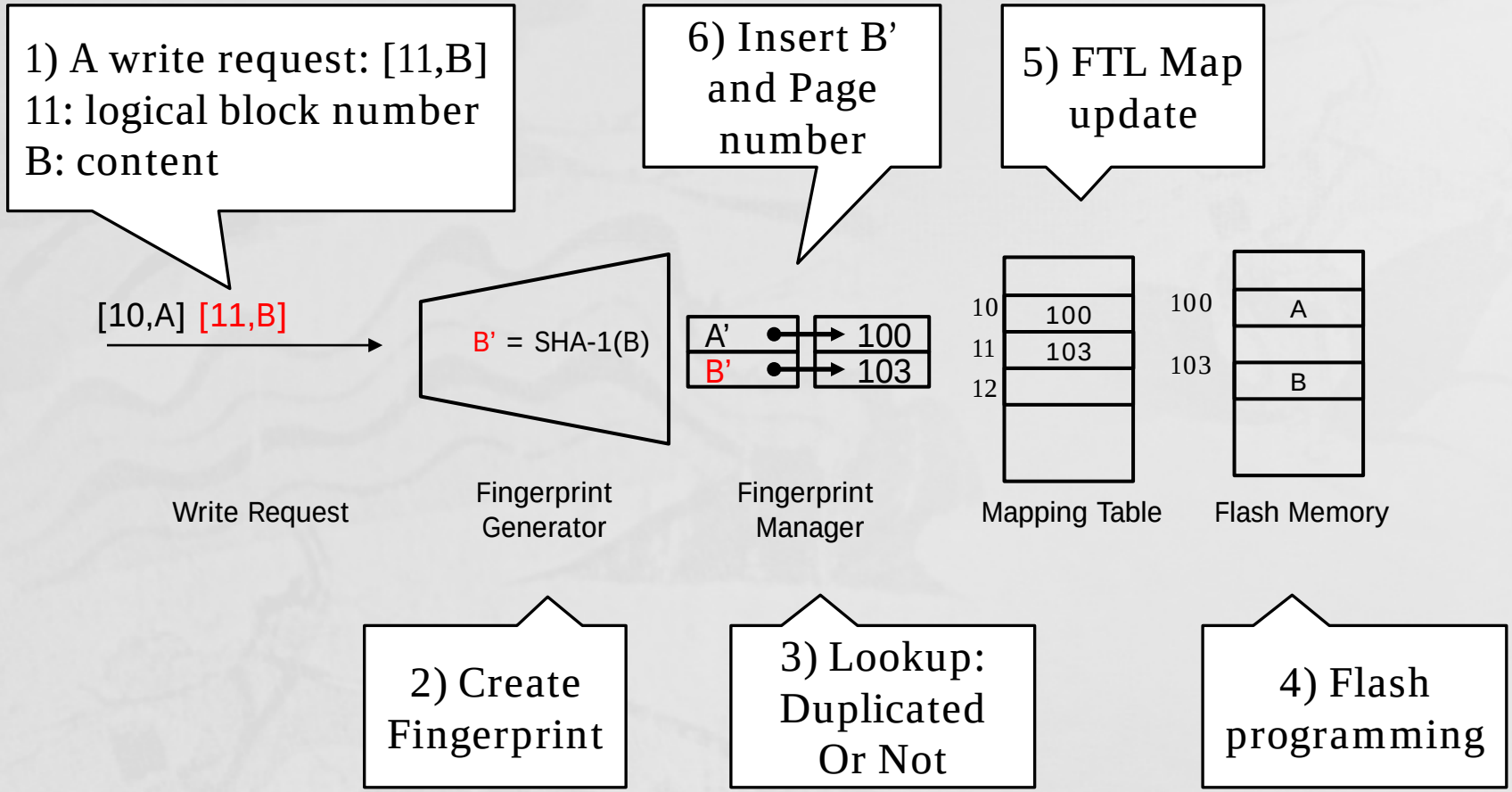
- Fingerprint Generator
  - Fixed chucking vs. Variable chucking ➔ Fixed chucking (4KB)
  - Diverse cryptographic hash functions ➔ SHA-1
    - generate a 160-bit hash value from 4KB data
- Fingerprint Manager
  - # of fingerprints: Full vs. Partial ➔ Partial (Recent X fingerprints with LRU)
- Mapping Manager
  - Make use of the mapping table of FTL (page-level mapping)

Architecture for Deduplication

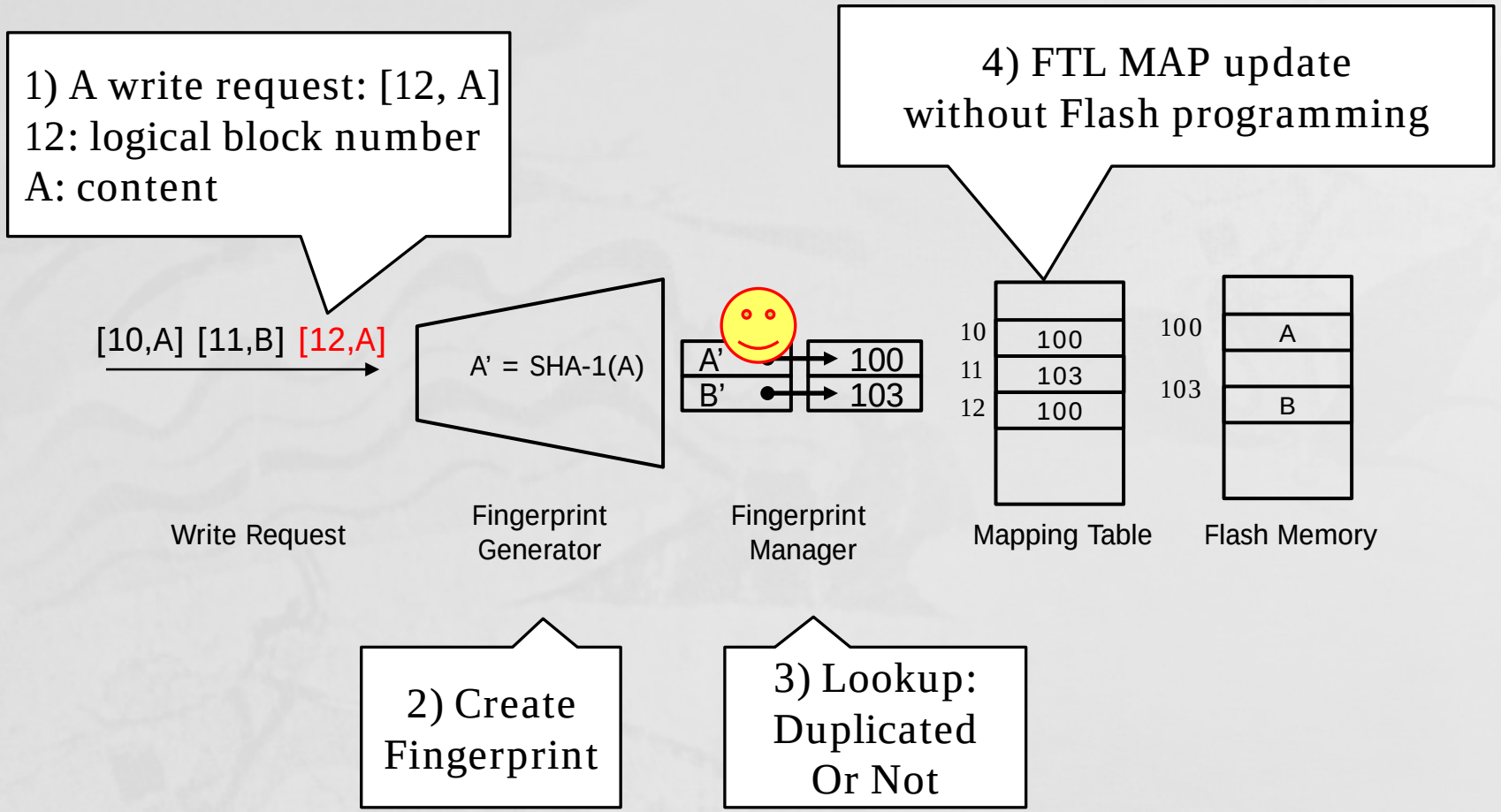
- Interactions among the Fingerprint generator, Fingerprint manager and Mapping manager.



Architecture for Deduplication



Architecture for Deduplication



Deduplication Model

- Question
  - How much duplication rate is required to obtain performance gain in SSD?
- Cost-benefit Analysis
  - Write latency

Without deduplication

$$Write_{latency} = FM_{program} + MAP_{manage} \quad (1)$$

With deduplication

$$Write_{latency} = (FP_{generator} + FP_{manage} + MAP_{manage}) \times Dup_{rate} + (FP_{generator} + FP_{manage} + MAP_{manage} + FM_{program}) \times (1 - Dup_{rate}) \quad (2)$$

Terminology

$Write_{latency}$	Write latency (elapsed time to handle a write request)	$FM_{program}$	Programming time on Flash memory
MAP_{manage}	Mapping table update time	$FP_{generator}$	Fingerprint generation time
FP_{manage}	Lookup time in the Fingerprint manager	Dup_{rate}	Duplication Rate (Ratio between the duplicate data and total written data)

Deduplication Model

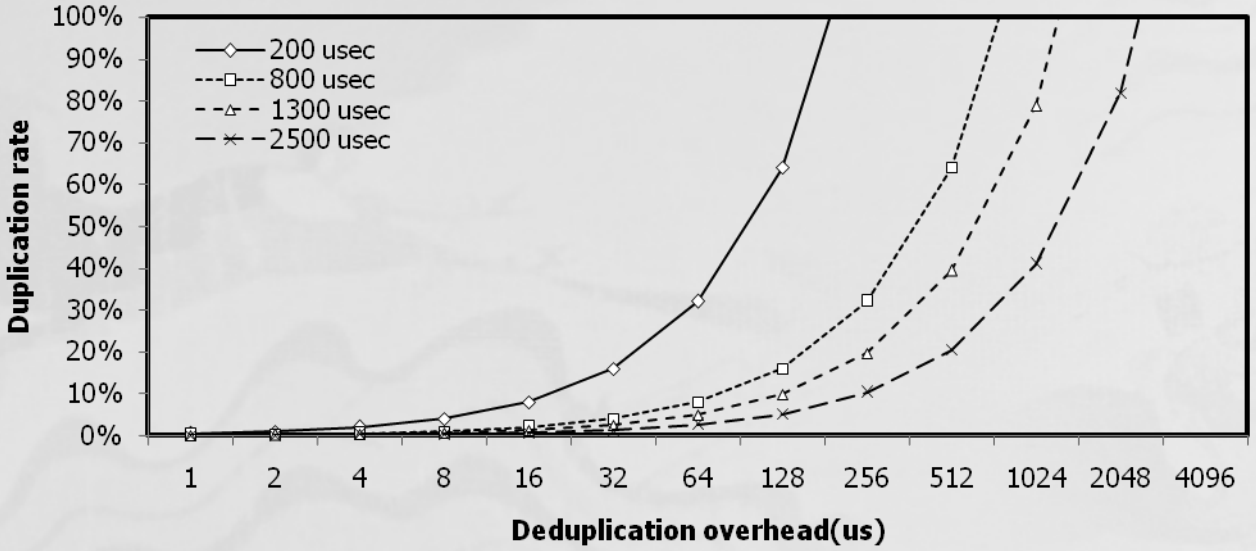
- To yield the performance gain
  - Equation 2) < Equation 1)

$$(FP_{generator} + FP_{manage} + MAP_{manage}) \times Dup_{rate} + (FP_{generator} + FP_{manage} + MAP_{manage} + FM_{program}) \times (1 - Dup_{rate}) < FM_{program} + MAP_{manage}$$

$$\vdots$$

$$Dup_{rate} > \frac{FP_{generator} + FP_{manage}}{FM_{program}} \quad (3)$$

Deduplication Model



- Graph notation
  - X-axis: Deduplication overhead (FPgenerator + FPmanage), Y-axis: Duplication rate.
  - Each line: Flash memory programming time (200, 800, 1300 and 2500 us, respectively).
  - Area above the line: where we can obtain performance gain using deduplication.

- Implication
  - The minimum duplication rate decreases as the deduplication overhead decreases or as the program time becomes longer.
  - Workloads with the higher duplication rate can yield larger deduplication benefit.

Deduplication Overhead and Acceleration Techniques

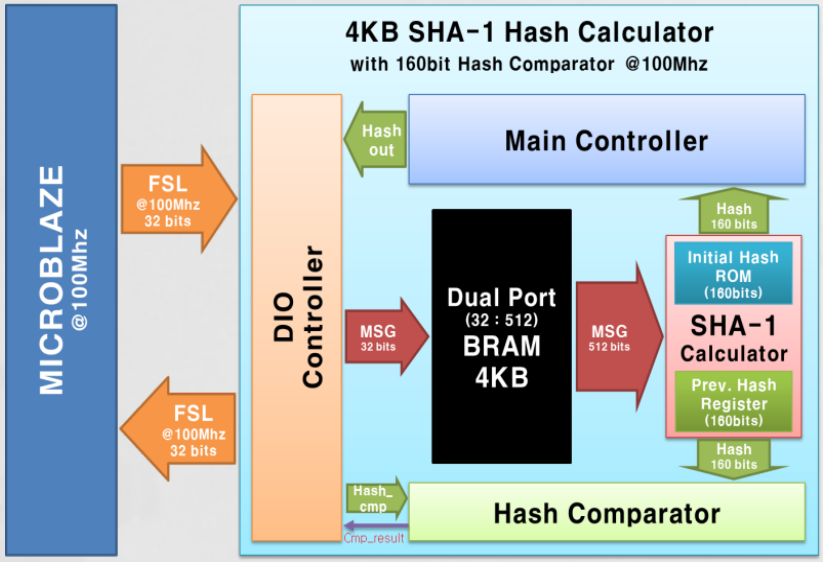
- Lesson from the deduplication model
  - Higher duplication rate, longer programming time, and lower deduplication overhead can give positive impacts on performance
  - The duplication rate is determined by the characteristics of workloads
  - The programming time is given by the specification of Flash memory
  - The key point is how to reduce the deduplication overhead (FPgenerator, FPmanage, and MAPmanage)

- Three Acceleration Techniques
  - SHA-1 Hardware logic (H/W approach)
  - Sampling based Filtering (S/W approach)
  - Recency based fingerprint management (S/W approach)

SHA-1 hardware logic

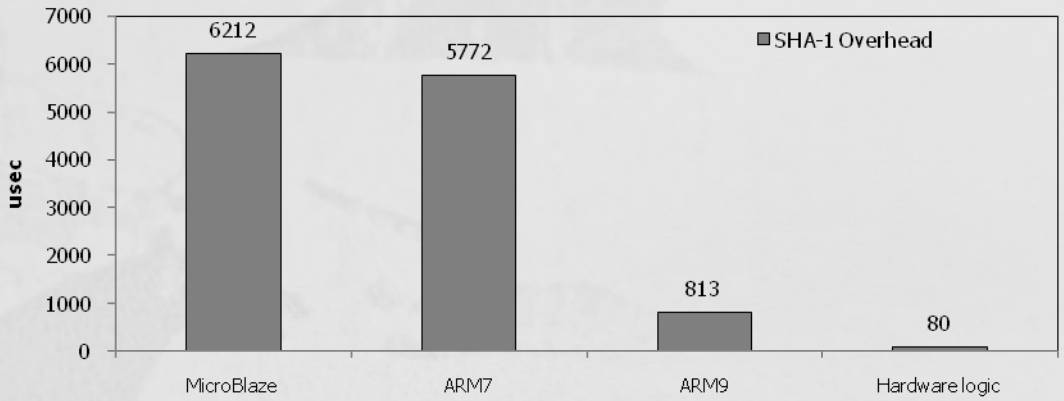
- Hardware acceleration technique
  - Xilinx Virtex6 XC6VLX240T FPGA
  - Verilog HDL 2001 for RTL coding

- Components
  - Main Controller
    - Govern the logic on the whole
  - DIO Controller
    - Data I/O Control unit
    - interfacing the logic with CPU.
  - 4KB Dual Port BRAM
    - Storing 4KB data temporary.
  - SHA-1 Calculator
    - Generating fingerprints using the standard SHA-1 algorithm.
  - Hash Comparator
    - Examines two fingerprints and returns whether they are same or not.



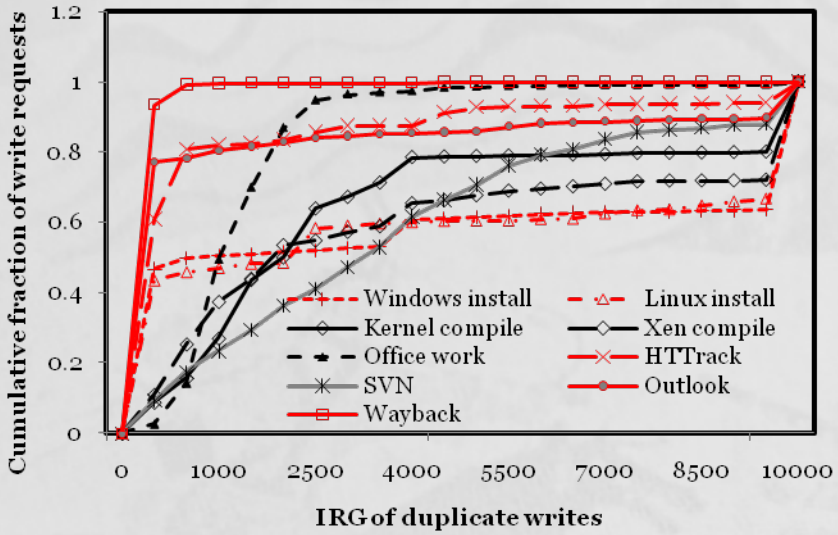
SHA-1 hardware logic

- Measurement of fingerprint generation overhead
  - Software version of the standard SHA-1 algorithm
    - 150MHz Xilinx Microblaze, 175MHz ARM7, 400MHz ARM9
    - 6212, 5772, 813 us per each SHA-1 calculation, respectively
  - SHA-1 hardware logic
    - 80 us on average
    - The deduplication model indicates that, when the programming time is 1300, we can expect the improvement of write latency with the minimum duplication rate of 5%.



Sampling-based Filtering

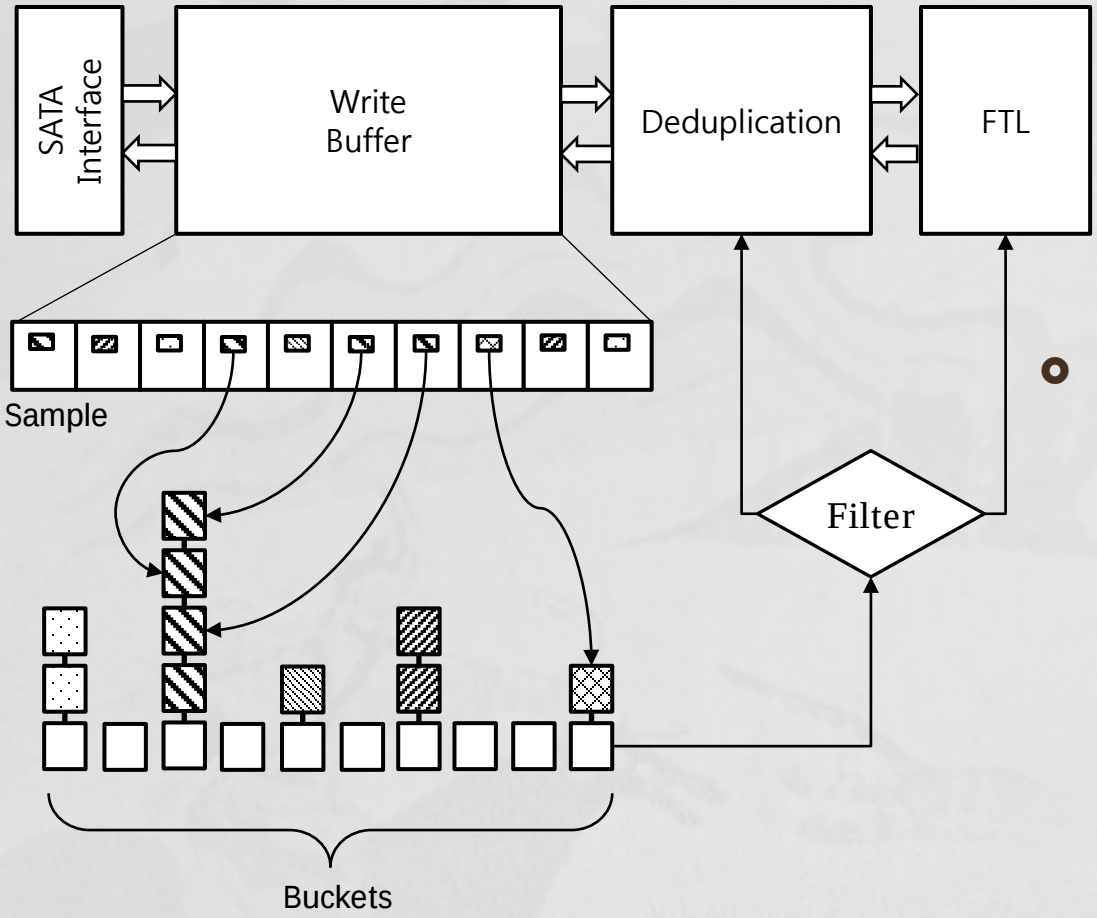
- Software acceleration technique
  - Selectively applies deduplication for write requests according to their duplicate possibilities
- IRG (Inter Referencing Gap)
  - Time difference between successive duplicate writes



- Red Group
  - most of the IRGs of duplicate writes are less than 500
- Black Group
  - more than 60% of IRGs are less than 4,000

Sampling-based Filtering

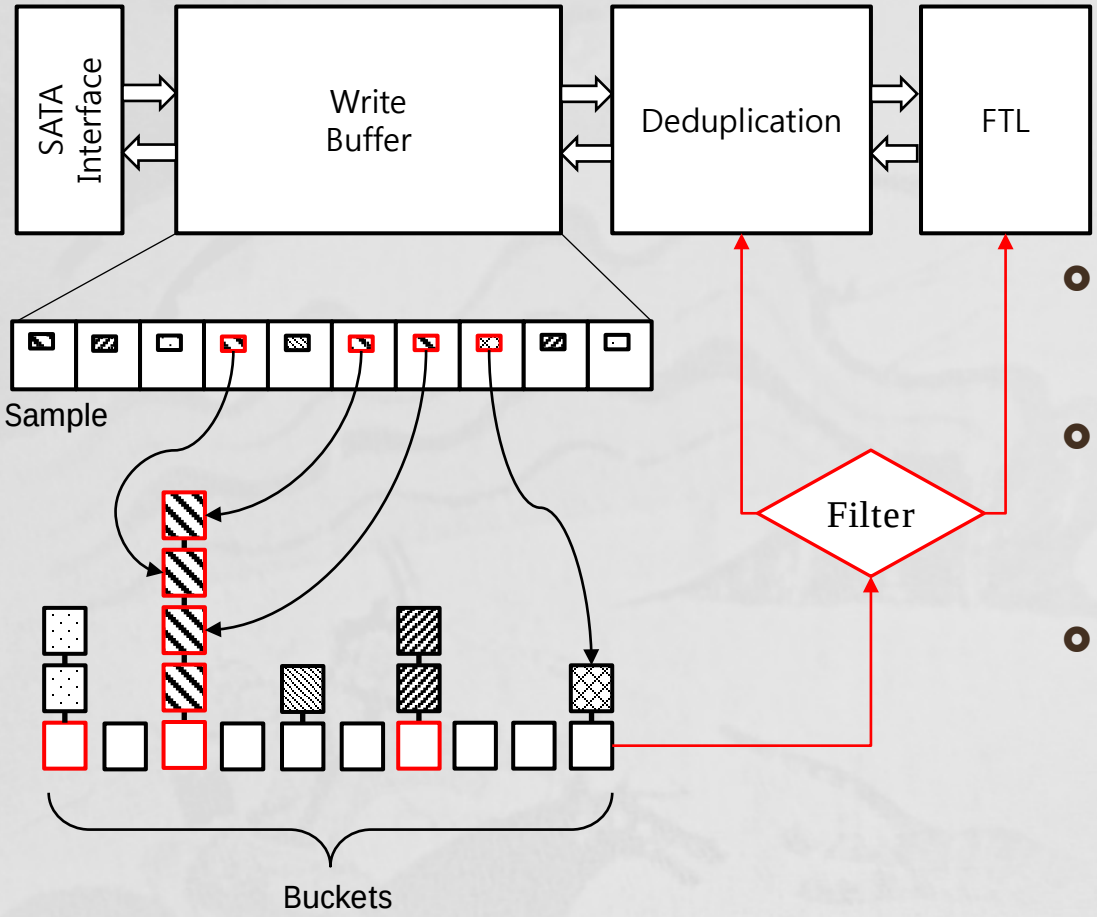
- Exploit write buffer in SSD



- OpenSSD
  - Write buffer: 32MB
  - Request: 8000 number s of 4KB pending write requests can be kept at maximum

Sampling-based Filtering

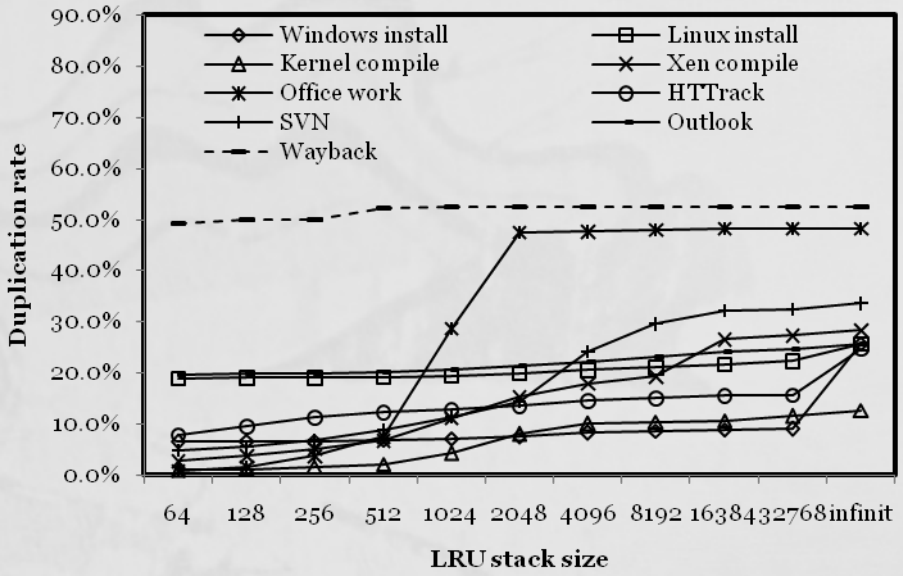
- Algorithm: Three steps



- Sampling
  - extract sample data from a randomly selected offset.
- Bucketing
  - classify write requests into buckets using sampled data as a hash index
- Filtering
  - apply deduplication only to the write requests in the buckets that have multiple requests

Recency-based Fingerprint Management

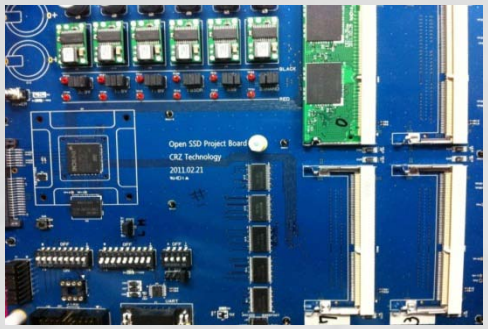
- To reduce FP manager overhead
  - The characteristics of SSD workloads with a viewpoint of the LRU stack
    - Strong temporal locality



  - Maintains recently generated fingerprints only, rather than managing all generated fingerprints

Experimental Environments

- Using Three Boards



OpenSSD

EZ-X5

Xilinx Vertex 6

	OpenSSD	EZ-X5	Xilinx Vertex 6
CPU	ARM7	ARM9	Microblaze
Clock	175MHz	400MHz	150MHz
SRAM	96KB	None	None
DRAM	64MB	64 MB	256MB
Flash	128GB	64 MB	16GB~
Purpose	Main experimental platform	For testing the Sampling-based filtering experiments	For testing the SHA-1 hardware logic experiments

Hardware configuration

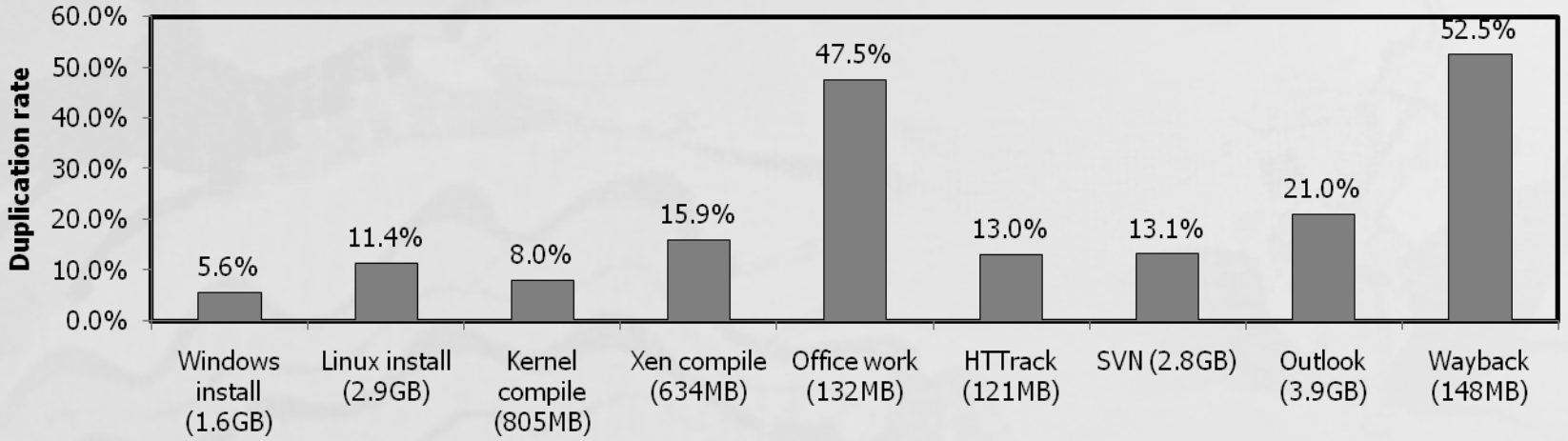
Experimental Environments

- 9 applications are used for the experiments

Name	Description	Total written data size
Windows install	Install Microsoft windows XP professional edition	1.6GB
Linux install	Install Ubuntu 10.10	2.9GB
Kernel compile	Compile the Linux kernel version 2.6.32	805MB
Xen compile	Compile the Xen version 4.1.1	634MB
Office work	Perform Microsoft Office applications (word and power point)	132MB
Outlook sync	Synchronize Gmail accounts	3.9GB
HTTrack	Download the contents of Dankook university web site	121MB
SVN	Using revisions of VirtualBox sources	2.8GB
Wayback machine	Download archived pages, composed of the first page of the Yahoo! Web site	148MB

Experimental Results

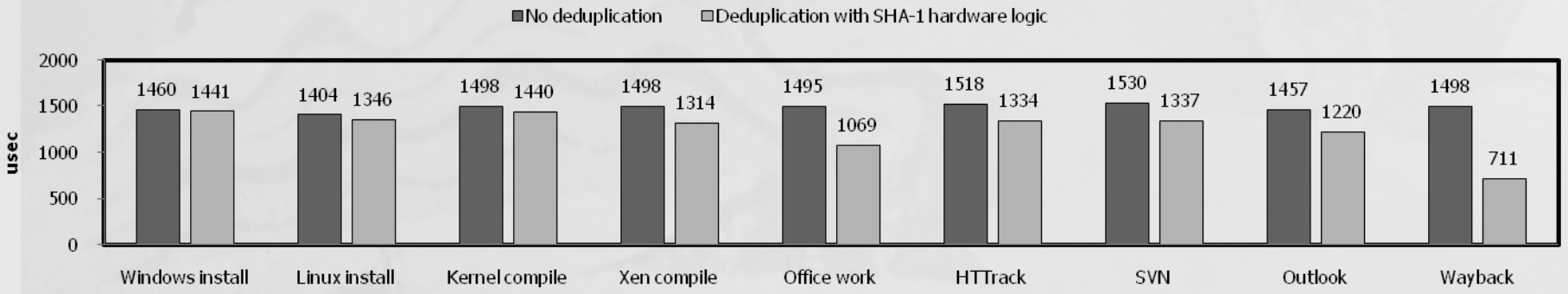
- Duplication Rate



- ranging from 5% to 52% with an average of 17%
  - Among the nine workloads, we can achieve the same duplication rate for each run from the windows install, Linux install, kernel compile and Xen compile workloads.
  - The duplication rate of the office work, HTTrack and outlook workloads depends on the user behavior and contents of a site/mail server.
  - The wayback machine shows the best duplication rate since it writes not only the modified data but also the unchanged data altogether for archiving.
  - The SVN saves modified data only in each revision

Experimental Results

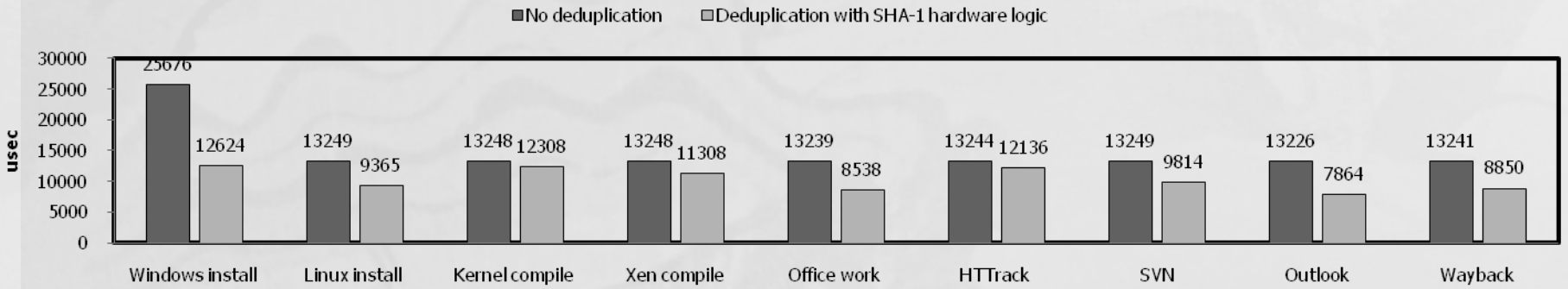
- Write latency
  - Experimental condition:
    - Use the SHA-1 hardware logic and recency-based FP management
    - Garbage collection is not invoked. (set utilization as 0% before experiments)



  - Observation
    - By applying deduplication, we can improve the write latency by up to 48% with an average of 15%.
    - Application with higher duplication rate yields larger benefit
    - The improvements are owing to the elimination of duplicate writes.

Experimental Results

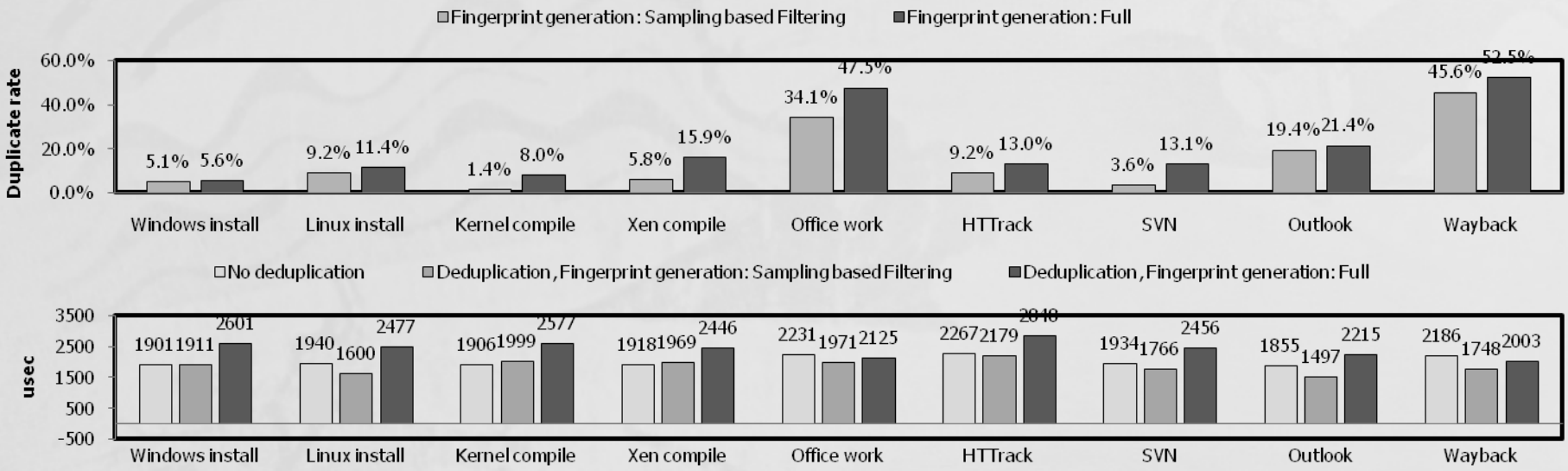
- Write latency
  - Experimental condition:
    - Use the SHA-1 hardware logic and recency-based FP management
    - Garbage collection is invoked during application execution (set utilization as 90% before experiments)



  - Observation
    - We can improve the write latency by up to 51% with an average of 27%.
    - When the garbage collection is involved, we can get larger performance gain since deduplication can not only eliminate duplicate writes but also reduce the garbage collection overhead.

Experimental Results

- Duplication rate and write latency
  - Experimental condition:
    - Use the Sampling-based filtering and recency-based FP management
    - Garbage collection is invoked during application execution (set utilization as 80% before experiments)

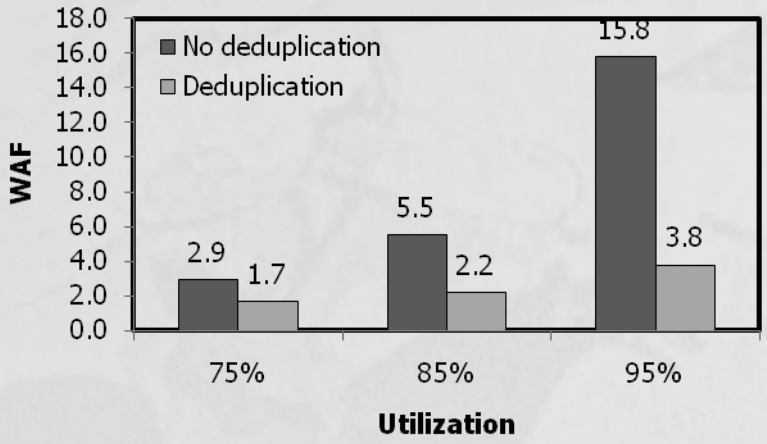


  - Observation
    - Achieving comparable duplication rate to the full FP generation
    - Obtain performance gain without SHA-1 hardware logic

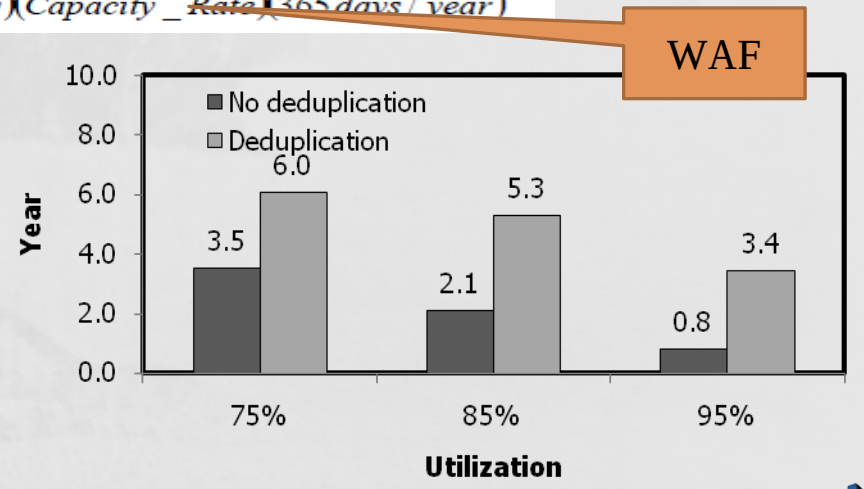
Experimental Results

- Reliability: WAF and Lifespan
  - WAF: Write Amplification Factor
    - Ratio of data actually written into Flash to data requested by the host
    - Deduplication reduce WAF by reducing both write traffic and GC overhead
  - Lifespan
    - Reduction of WAF can lead to enhance the lifetime of SSD
    - From Imation paper (www.csee.umbc.edu/~squire/images/ssd1.pdf)

$$Lifetime(years) = \frac{(SSD\_Capacity(GB))(P/E)(Percent\_Utilization)}{(Usage/Day)(Capacity\_Rate)(365\,days/year)}$$



Write Amplification Factor



Expected lifespan

Conclusions

- Deduplication architecture for SSD
  - FP generator, FP manager, MAP manager
- Deduplication model
  - Minimum duplication rate for performance gain under various deduplication overhead and programming time
- Three acceleration Techniques
  - SHA-1 hardware logic
  - Sampling-based filtering
  - Recency-based fingerprint management
- Experimental results
  - Deduplication is an effective solution for improving the write latency and lifespan of SSDs.

Discussion

- Question or Suggestion