Fully Automatic Stream Management for Multi-Streamed SSDs using Program Contexts

Taejin Kim[1], Duwon Hong[1], Sangwook Shane Hahn[2], Myoungjun Chun[1], Sungjin Lee[3], Jooyoung Hwang[4], Jongyoul Lee[4], and Jihong Kim[1]

[1]Seoul National University, [2]Western Digital, [3]DGIST, [4]Samsung Electronics

17th USENIX Conference on File and Storage Technologies, 2019

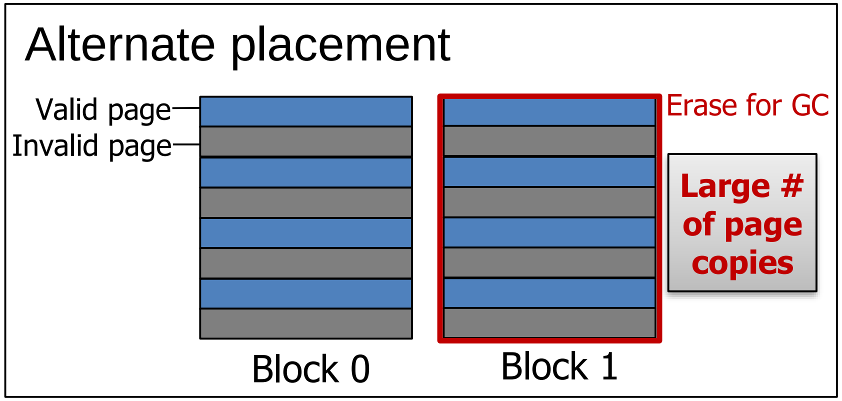
Outline

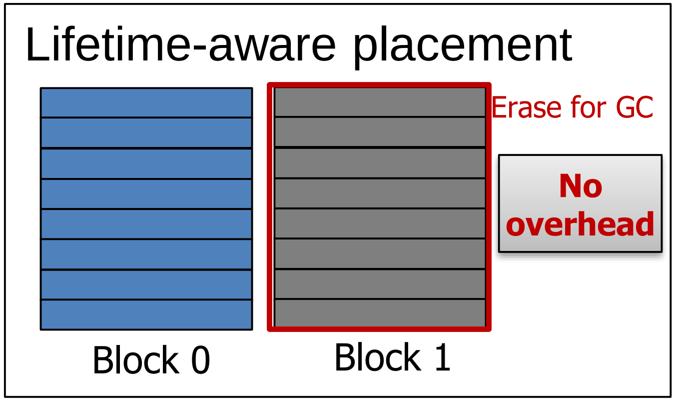
- Introduction & Motivation
- Automatic Stream Identification
- Design of PCStream
- Evaluations
- Conclusions

Garbage Collection Overhead in SSDs

- Garbage collection (GC) overhead
  - Reclaiming free space requires copying valid pages
  - Amplified writes shorten lifetime and reduce performance of SSDs

- How to minimize amplified writes
  - Prevent scattered page invalidation
  - Need to *know similar lifetime data* and *physically separate* them

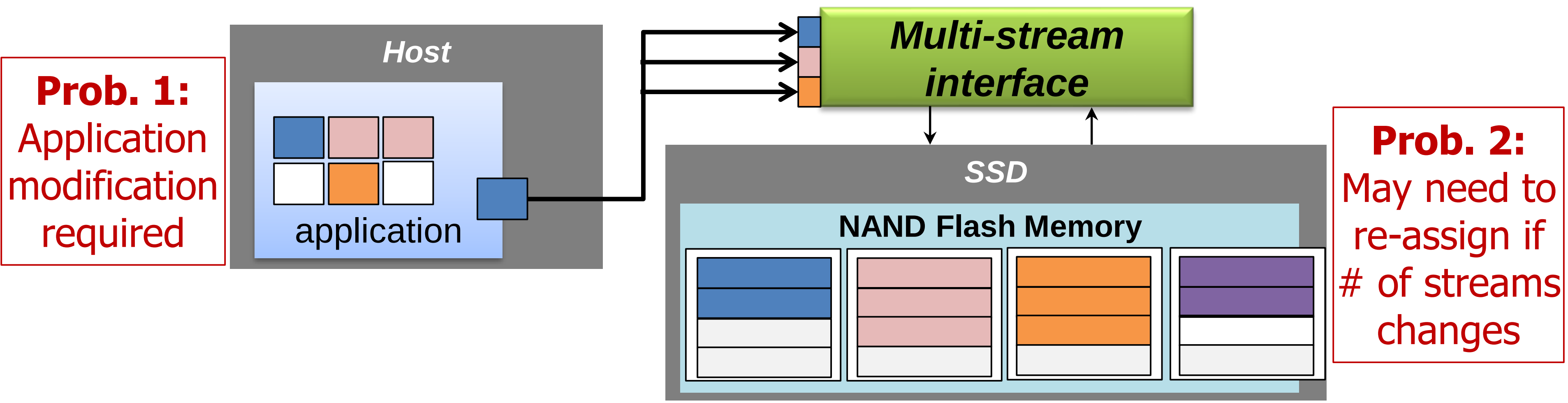




Placing data with similar lifetimes together can reduce GC overhead

Multi-stream: Minimize Write Amplification

- Data with different streams are *physically separated*
- Challenges of using the multi-stream feature
  - Host: Difficult to know **data lifetime** in advance
  - SSD: **# of supported streams** may be different across SSDs

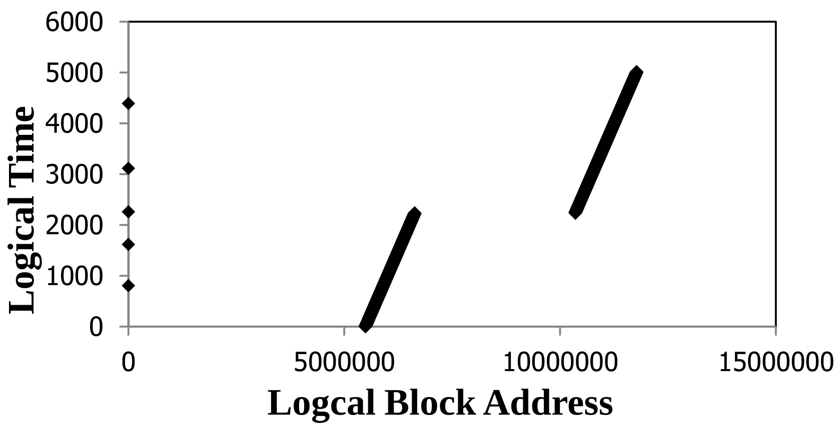


Automatic stream management is required

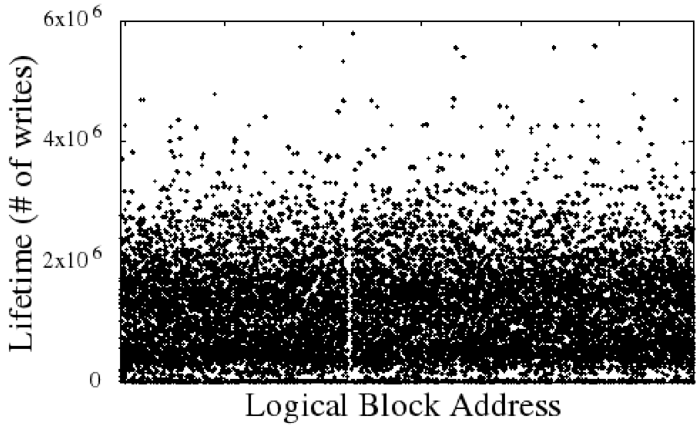
"The Multi-streamed SSD", J. Kang et al., in *HotStorage'14*

Existing Automatic Stream Management

- AutoStream assigns stream at device driver layer based on the access frequency of the same LBA
  - Applicability is limited only when LBA access locality is obvious
  - Does not work well when no apparent locality on LBA accesses (e.g., append-only, write-once patterns)



<Sequential Access Patterns>
(Multi-media)



<Append-only Lifetime Patterns>
(RocksDB)

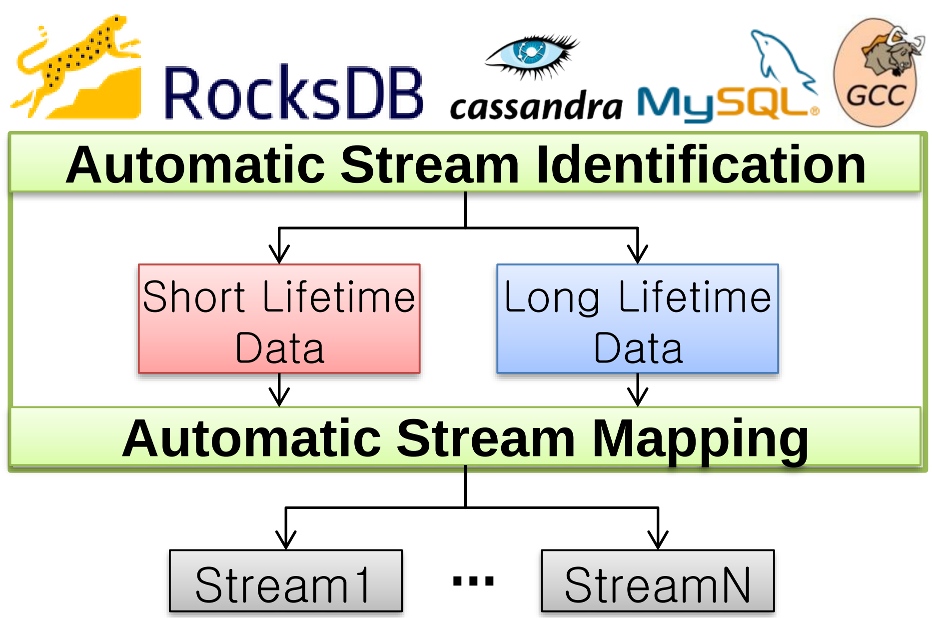
Data lifetime prediction for append-only workload is required

"AutoStream: Automatic Stream Management for multi-streamed SSDs", J. Yang et al., in *SYSTOR'17*

Design Goal of Proposed Work

- Must <u>*automatically*</u> work with <u>*general*</u> workloads
  - Streams are identified without modifying application
    - Data lifetimes should be estimated at a higher abstraction level than LBAs (I/O activities)
  - Streams should be allocated automatically
    - Similar lifetime data are mapped to the same stream

Automatic Stream Management Technique

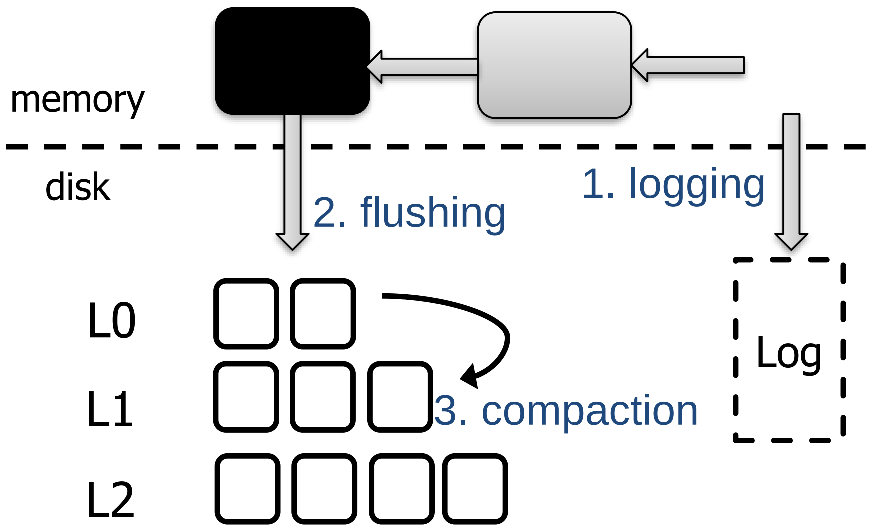


Outline

- Introduction & Motivation
- Automatic Stream Identification
- Design of PCStream
- Evaluations
- Conclusions

Stream Identification using I/O Activities

- I/O activities show distinct lifetime patterns
  - Ex) 3 activities in RocksDB shows distinct lifetime patterns
    - Logging: valid until data in memory are flushed (short lifetime)
    - Flushing: deleted when top level of LSM-tree is full (short lifetime)
    - Compaction: deleted when its level is full (long lifetime)



Data lifetime can be estimated by I/O activities

How to distinguish I/O activities?

Program Context Can Distinguish Activities

- A program context (PC) is known to be an effective hint in separating data with similar update period
  - Represents a particular ***execution phase*** of a program
  - Identified by summing program counter values of each execution path of function calls
  - Ex) PC calculation with synthetic program
    - Execution path to the write system call

      Main () → func_B () → func_A () → write_data () → write ()

    - Addresses of program counter values in the stack

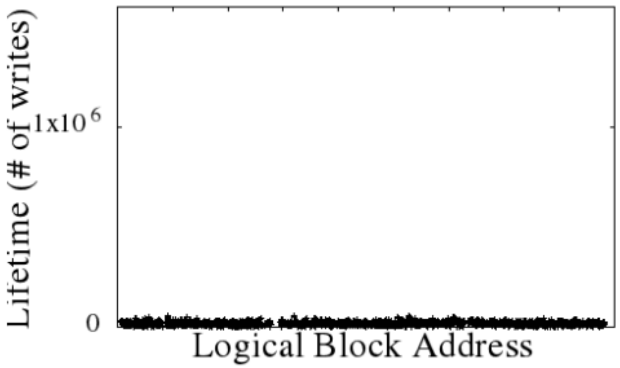
000000000040121e	main
00000000004012da	func_B
0000000000401024	func_A
0000000000400ffb	write_data
0000000000400d20	_start

    - Result **Summing all the values = 0x2408d00**

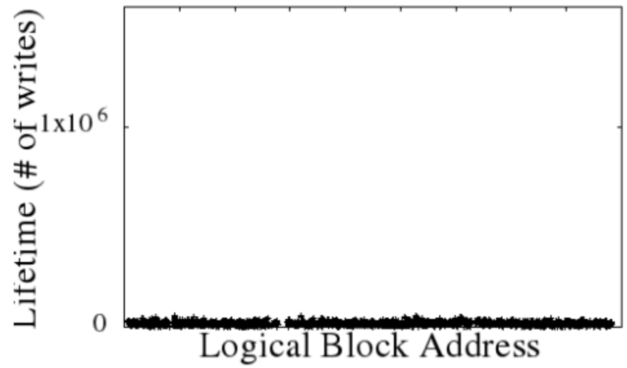
"A Program Context-Aware Data Separation Technique for Reducing Garbage Collection Overhead in NAND Flash Memory", K. Ha et al., SNAPI'11

Feasibility: Distinguishing Activities by PC

- Log data

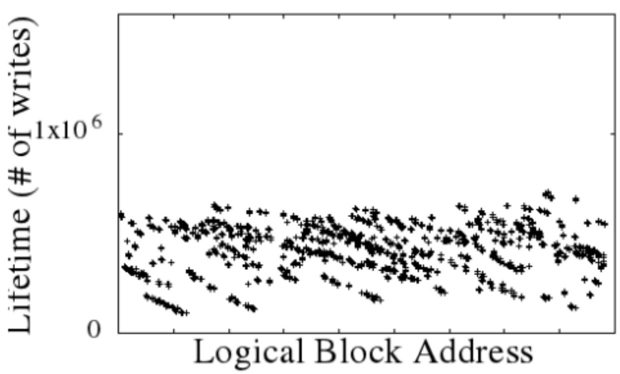


<manual>

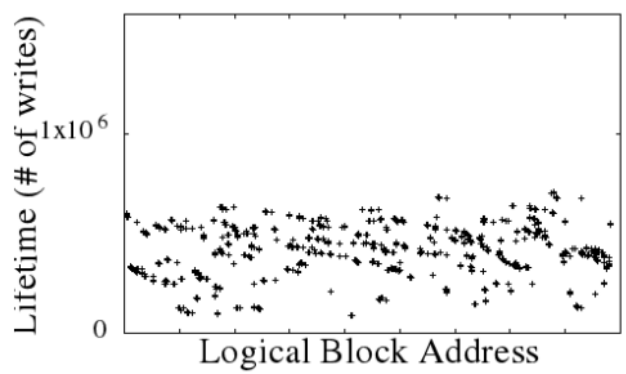


<PC #0>

- Flush data



<manual>

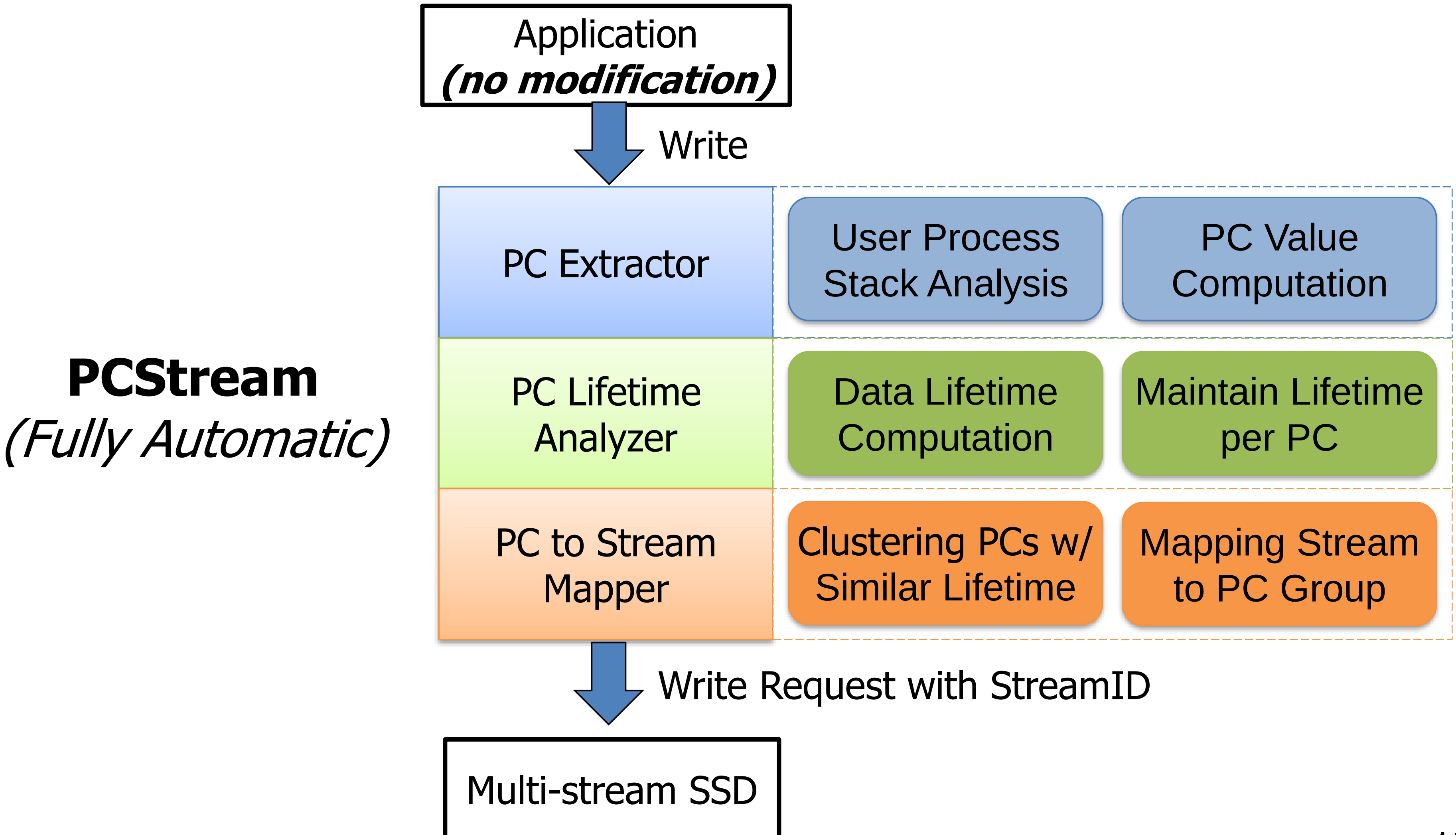


<PC #1>

Outline

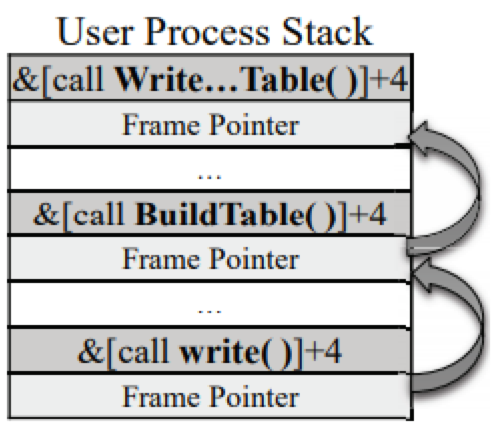
- Introduction & Motivation
- Automatic Stream Identification
- Design of PCStream
- Evaluations
- Conclusions

Overview of PCStream

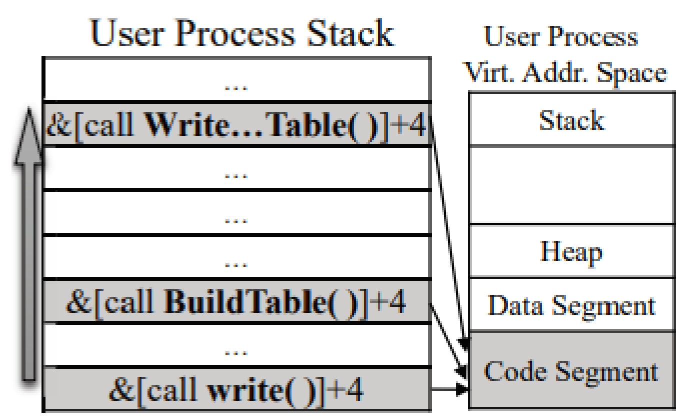


PC Extractor Module

- PC extraction with frame pointer
  - Recursively checking previous stack frames based on the frame pointer register (EBP)
- PC extraction without frame pointer
  - Frame pointer register is not available if omit-frame-pointer option is used by compiler
  - Scanning every word in the stack and check if it belongs to the process's code segment



<with frame pointer>

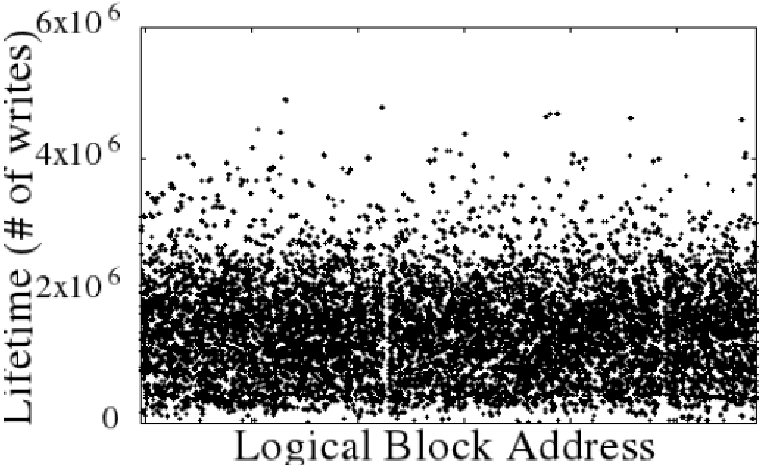


<without frame pointer>

PCs with Large Lifetime Variance

- Some PCs represent several I/O contexts
  - When multiple I/O contexts are covered by the same execuction to the write system call

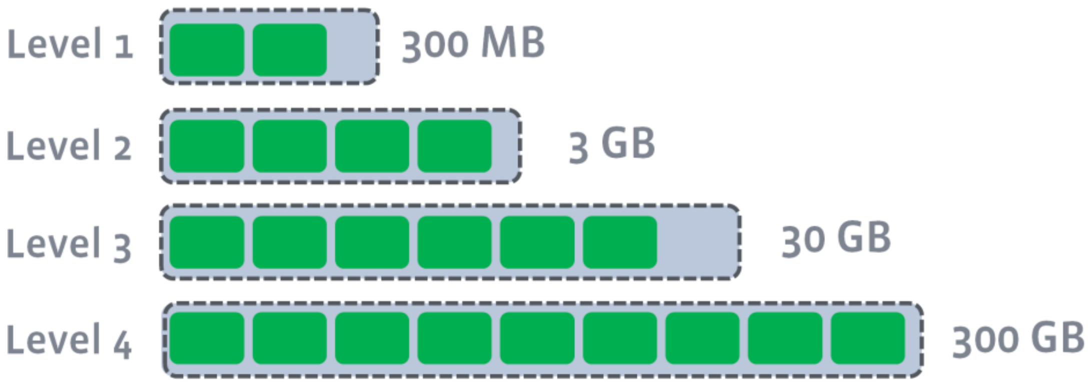
- Example: compaction at different levels of RocksDB
  - Regardless of compaction level, execution path to write system call is the same

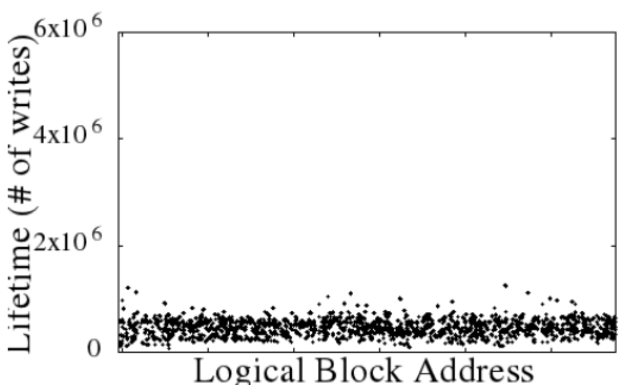


<Compaction PC>

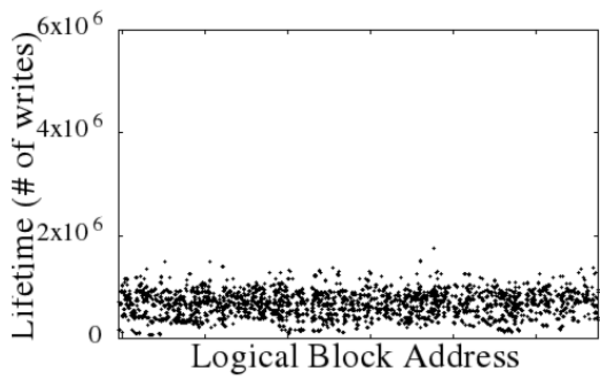
Lifetimes of Compaction Data per Level

- Higher level is smaller than lower level in LSM tree
  - Data in higher level are invalidated more frequently, shorter lifetime

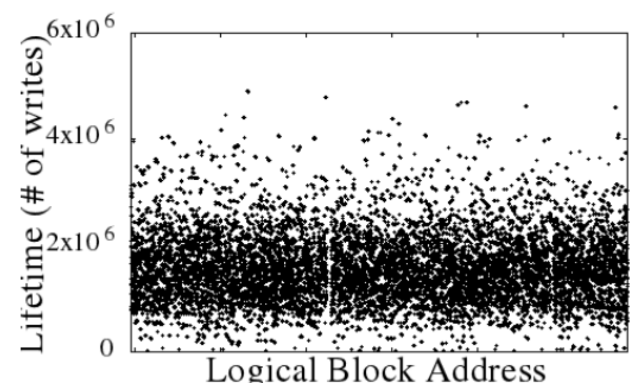




<Manual: L2>



<Manual: L3>



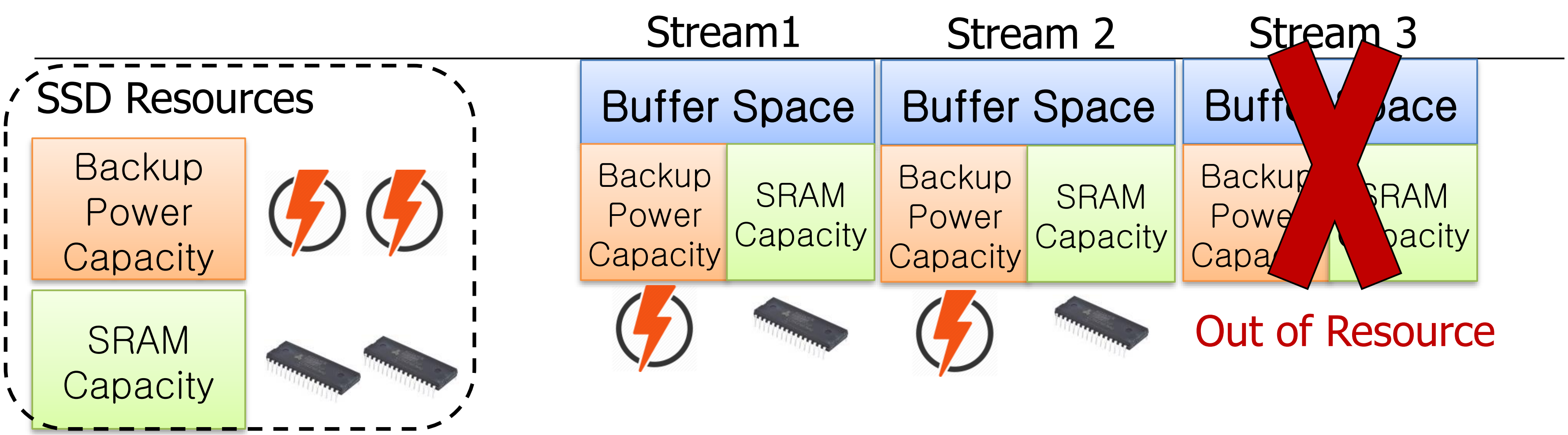
<Manual: L4 (Lowest)>

More streams are necessary for separating different lifetime data

Practical Limitations on Streams

- Host data should be buffered per stream
  - Hiding size difference between FTL and device

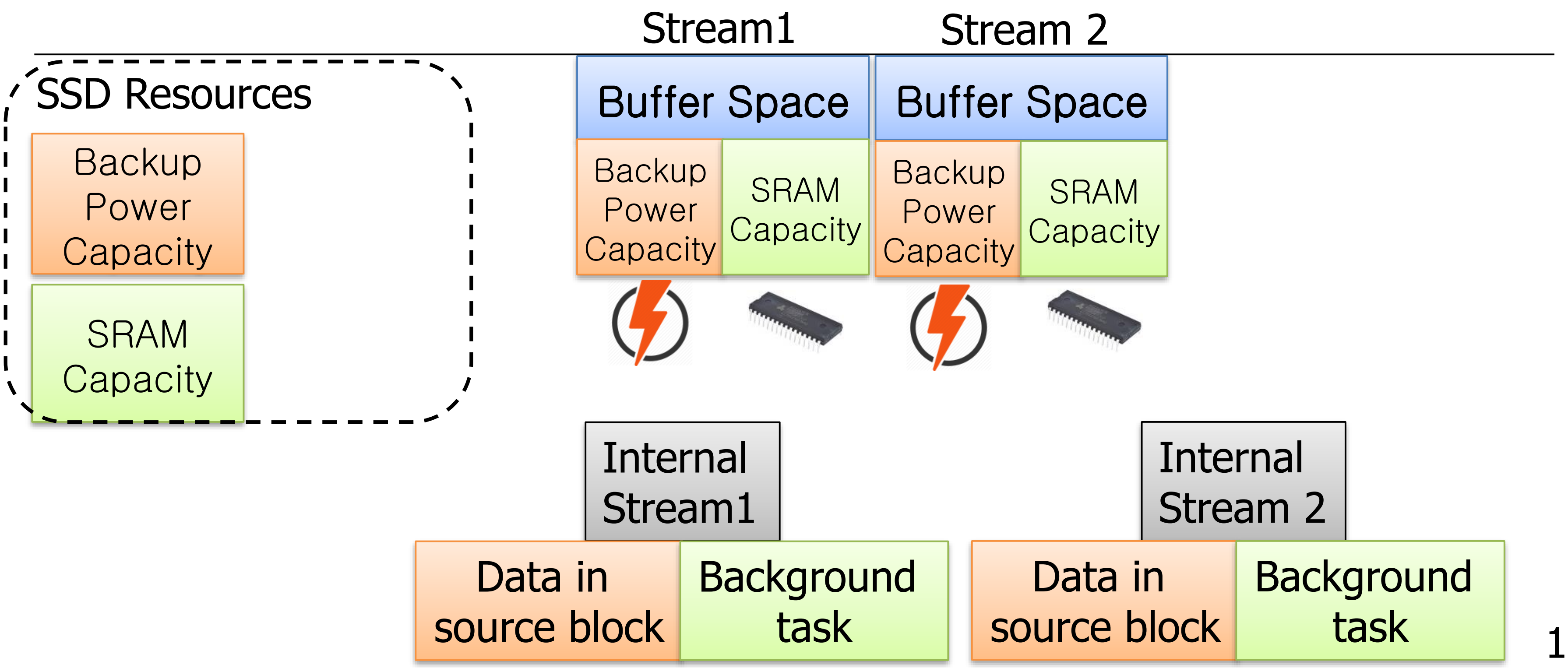
- Buffering data requires SSD resources
  - Backup power capacity: storing data for sudden power off
  - SRAM capacity: quick checkup of buffered data for read requests



Increasing number of streams is difficult

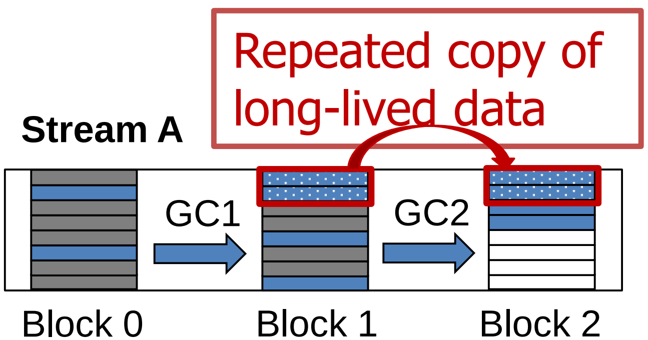
Internal Streams for GC

- Internal Stream: used only for data copy during GC
  - No backup power capacity: original data remains in source block
  - Slow (DRAM) memory: GC can be handled as background tasks
  - PCStream can effectively doubled # of available streams

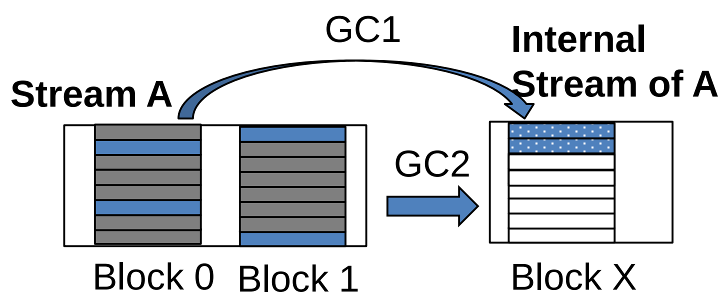


Two-Phase Stream Decision

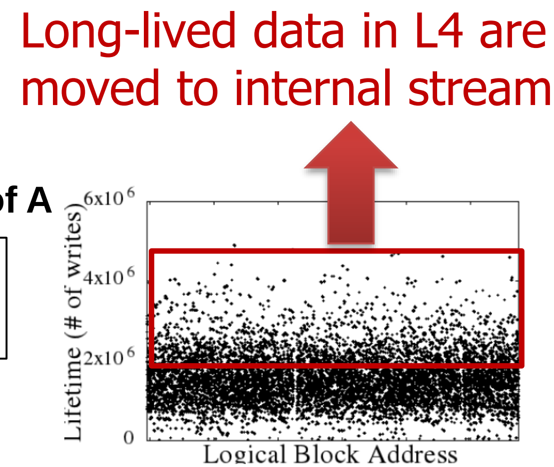
- For large variance PCs, apply 2-phase stream decision
  - 1st phase (host level): A stream is assigned to the PC
  - 2nd phase (device level): long-lived data are assigned to internal stream



<existing policy>



<2nd phase decision>



<Manual: L4 (Lowest)>

Using 2nd phase decision can avoid repeated copy of long-lived data in the large lifetime variance PC

Outline

- Introduction & Motivation
- Automatic Stream Identification
- Design of PCStream
- Evaluations
- Conclusions

Experimental Setting

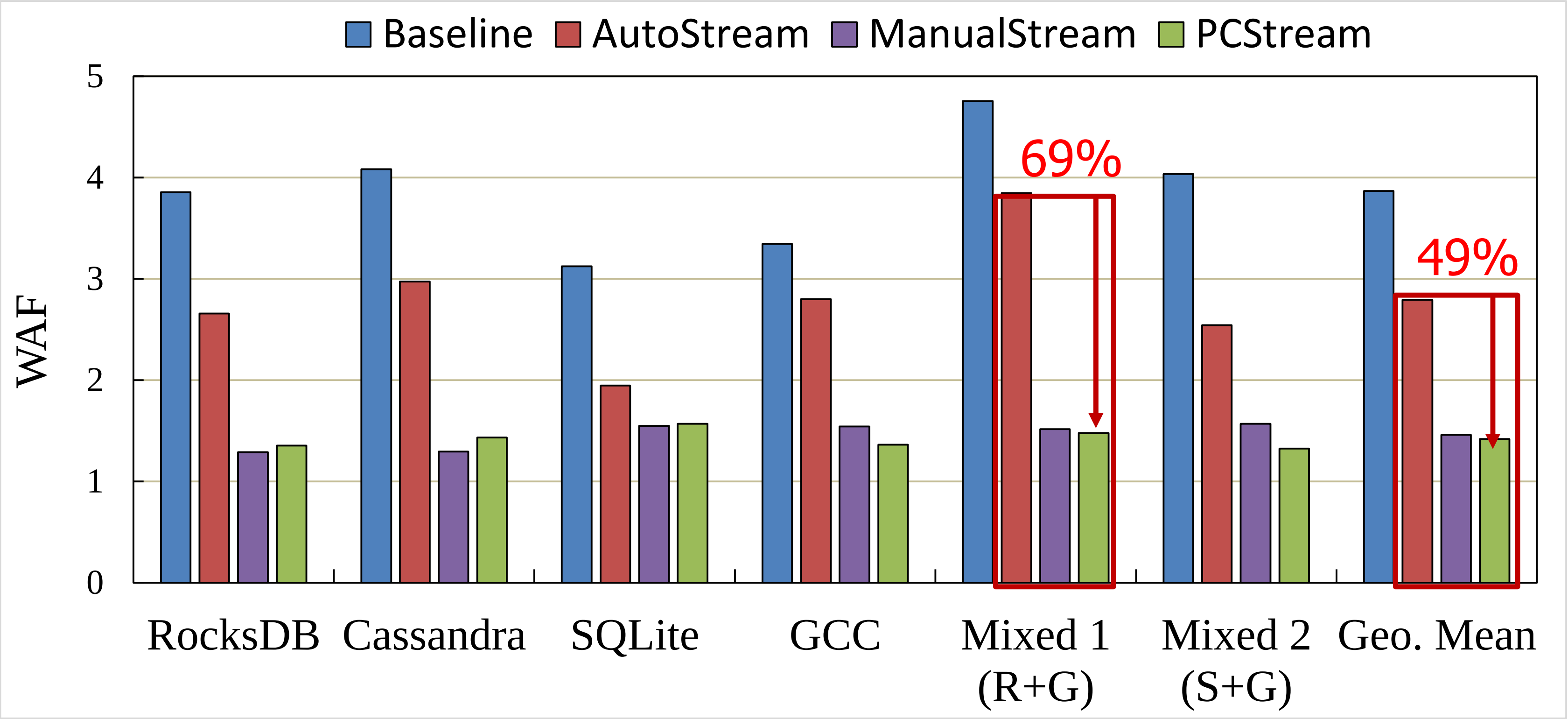
- Host
  - Linux kernel 4.5 with PCStream implementation
- SSD
  - Samsung PM963 SSD modified for internal stream support

PM963 SSD with 9 streams
(9 internal streams are added)

- Benchmark

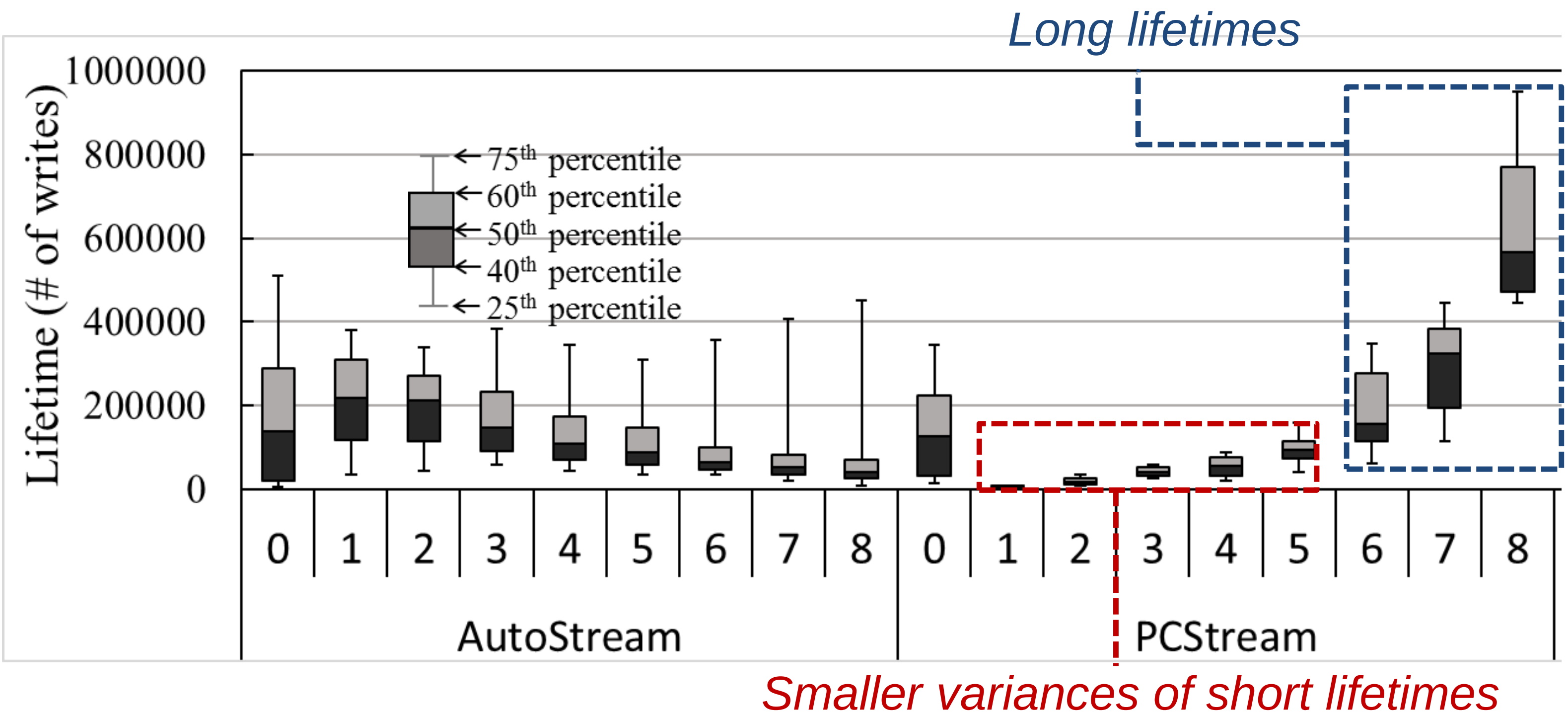
Benchmark	Type	Types of activities
RocksDB	Append-only	Logging, Flushing, Compaction
Cassandra	Append-only	Logging, Flushing, Compaction
SQLite	Updating	Logging, Updating DB table
GCC	Write-once	Outputting temp files, executable files
Mixed 1 (RocksDB+GCC)	A + W	
Mixed 2 (SQLite+GCC)	U + W	

WAF Comparison



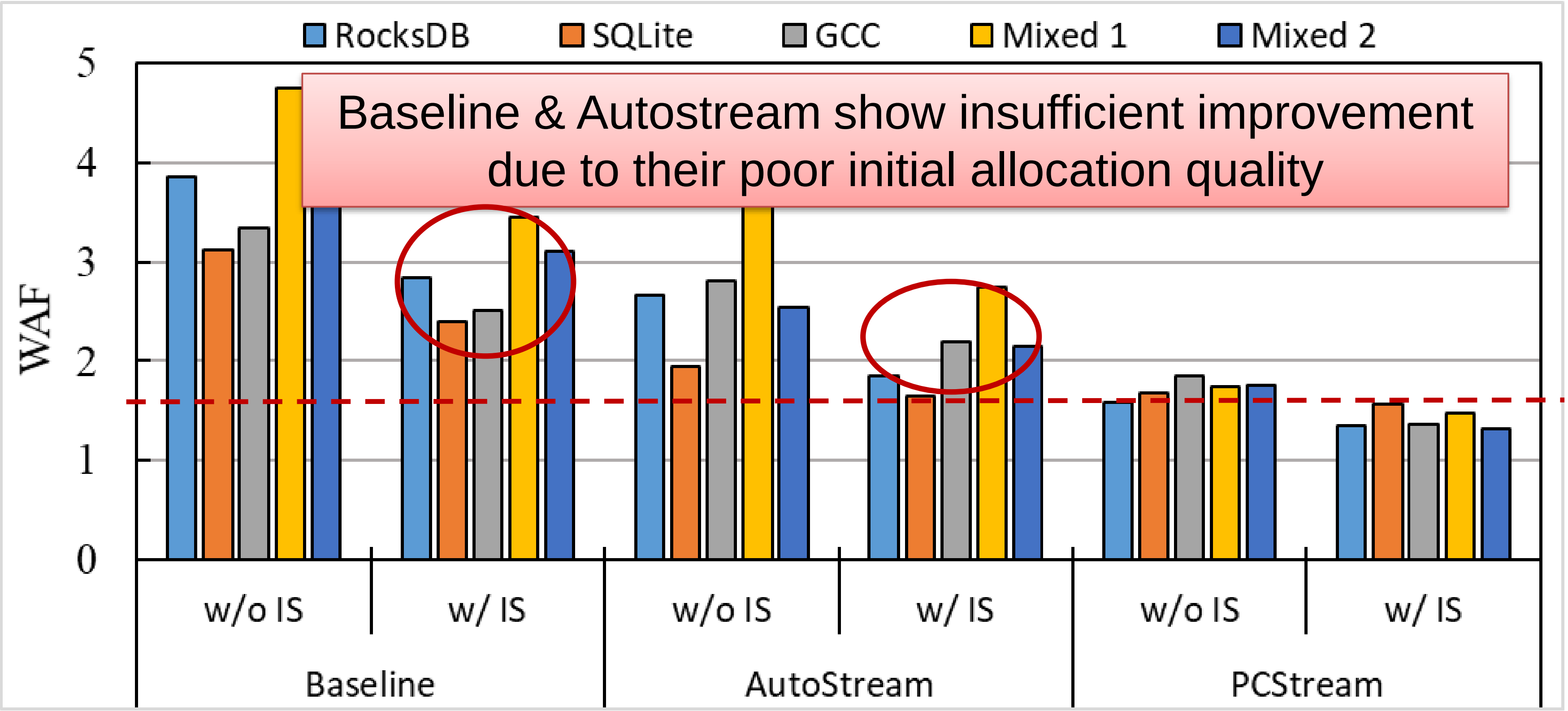
High efficiency of PCStream comes from
- LBA-oblivious data separation
- Internal streams

Per-stream Lifetime Distributions (Mixed 1)



Separating long lifetime data results better WAF reduction in small variance streams

Impact of Internal Streams



Conclusions

- We have presented the PCStream for improving performance and reducing WAF of multi-stream SSDs
  - Automatic stream management technique using program context to effectively estimate data lifetime
  - Internal stream can separate long-lived data from future short lifetime data
  - WAF was reduced by up to 69% over existing automatic technique

- Future work
  - Support applications based on indirect writes
    - Internal write buffer with flushing thread
    - mmap-related functions