

A Program Context-Aware Data Separation Technique for Reducing Garbage Collection Overhead in NAND Flash Memory

Keonsoo Ha and **Jihong Kim**

Computer Architecture & Embedded Systems Lab.
School of Computer Science and Engineering
Seoul National University

**7th IEEE International Workshop on
Storage Network Architecture and Parallel I/O**
May 25, 2011

Outline

- **Introduction**
- **Motivation**
- **A Program Context-Aware Data Separation Technique**
- **Experimental Results**
- **Conclusions**

NAND Flash Memory

- NAND flash memory becomes an attractive storage solution



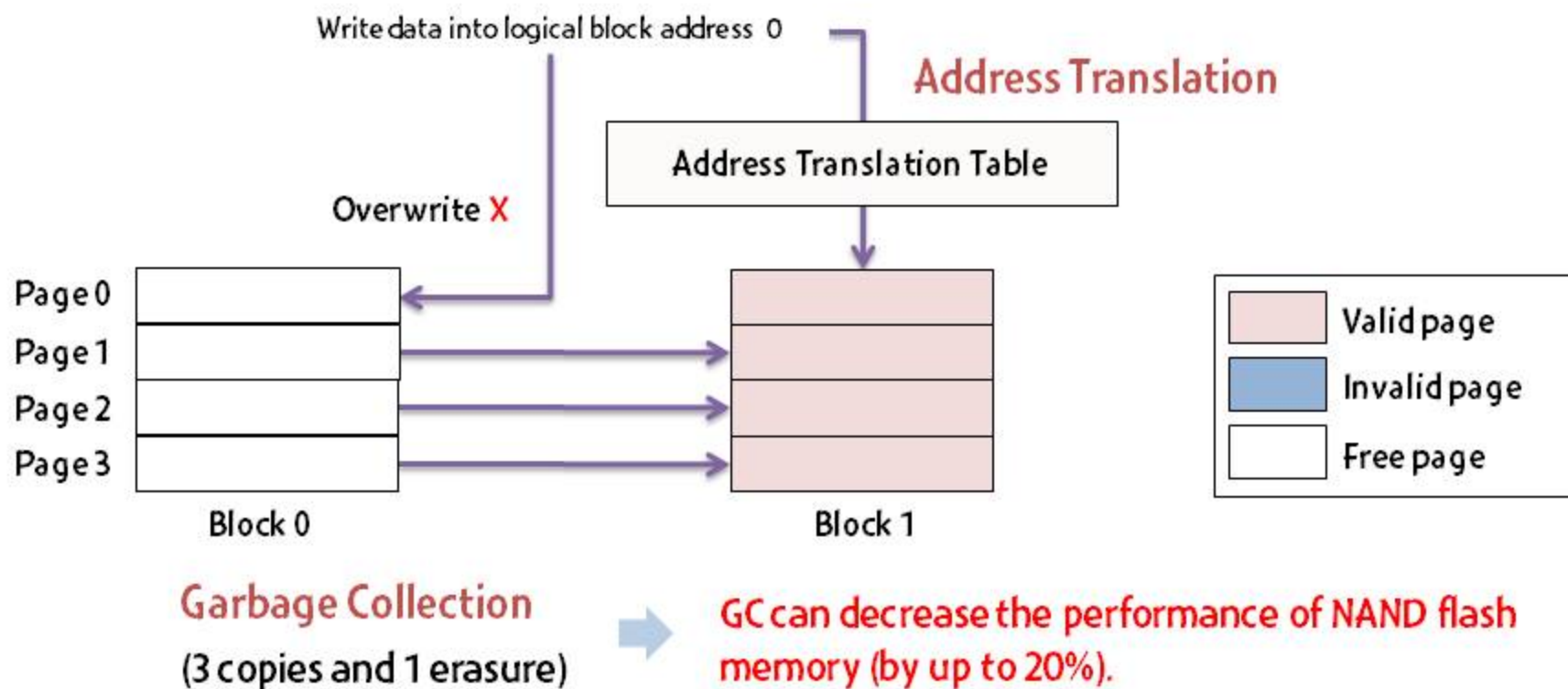
- Flash memory has unique characteristics
 - **Erase-before-write architecture**
 - **Asymmetric read/write and erase operation**



Flash Translation Layer (FTL) is used

Flash Translation Layer

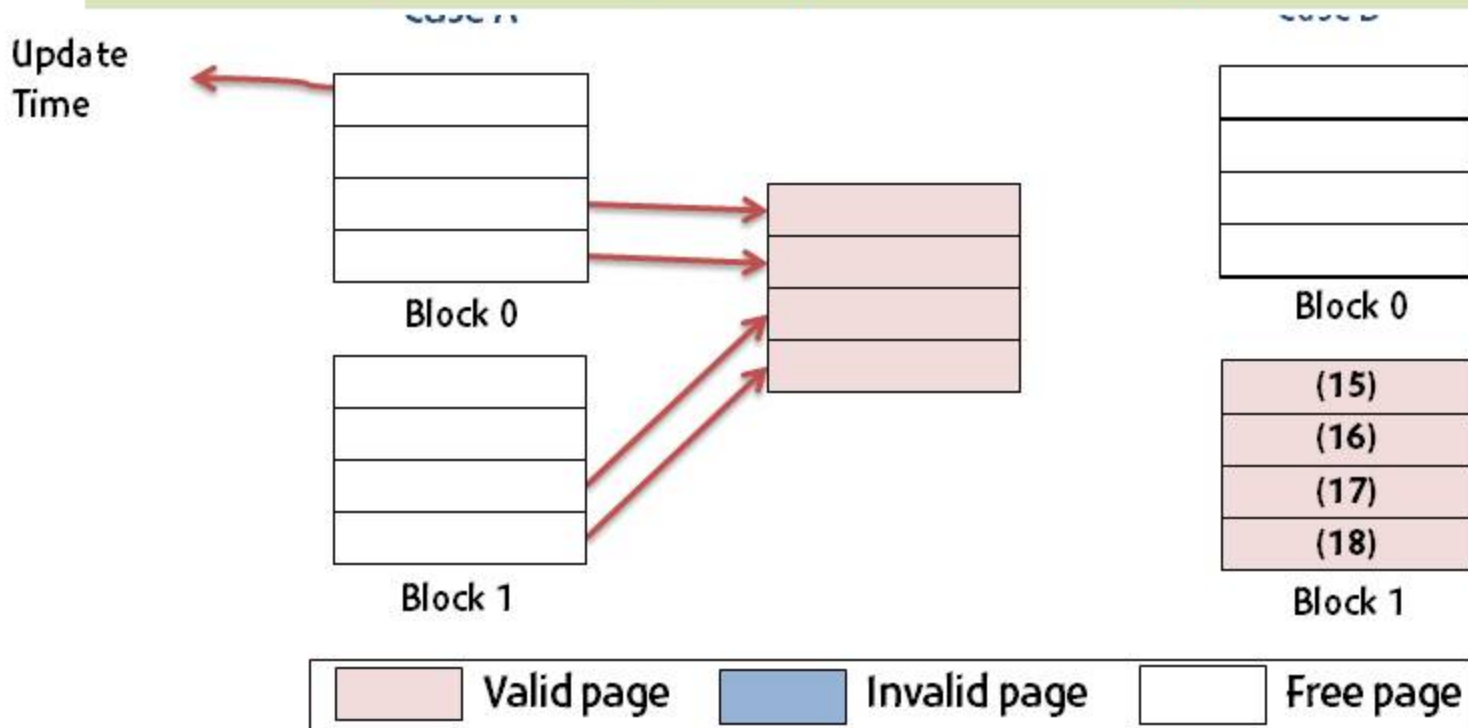
- Main functions of FTL
 - Address Translation
 - Garbage Collection (GC)





Reducing Garbage Collection Overhead

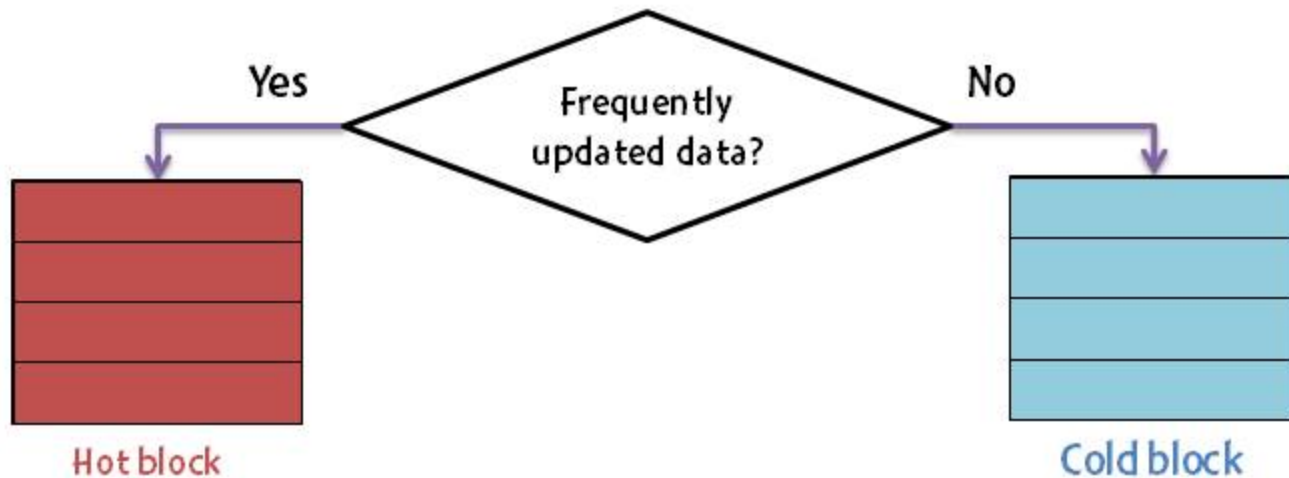
- Key requirement for reducing GC overhead
 - Gathering data with similar update times into the same block

In order to gather data with similar update times into the same block, efficient data separation technique is necessary



Existing Data Separation Technique

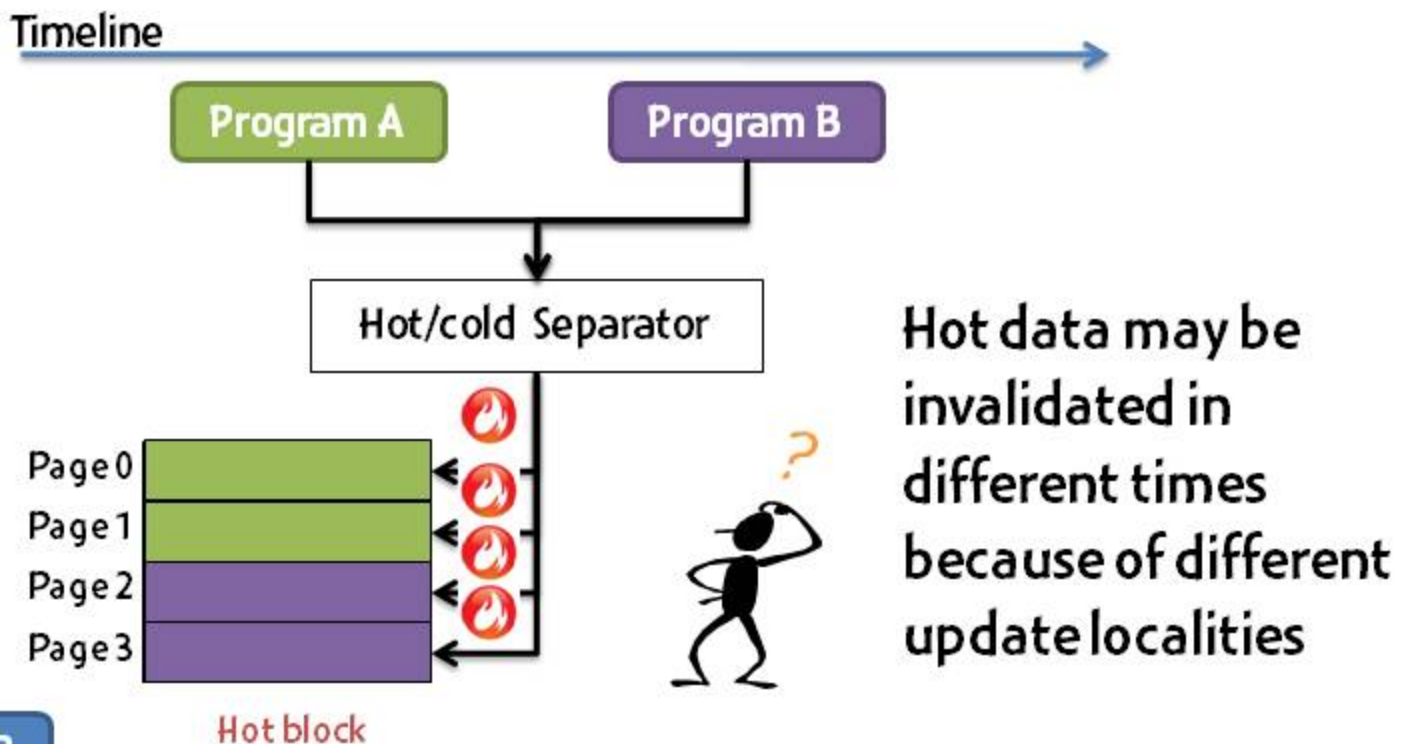
- Hot/cold Data Separation Technique
 - Key Assumption
 - Temporal locality of data updates
 - Classification based-on data update frequency
 - Hot: frequently updated data 
 - Cold: infrequently updated data 



Problems of Hot/cold Separator

Problem 1

Wide variations on future update times



Problem 2

If there is no clear temporal locality, hot/cold separator does not work

Contributions

- Evaluate performance gap of existing hot/cold separator with oracle predictor on *future* update times
 - Knowing future update time is a more important factor than update frequency
- Propose a new data separation technique based on *update times of data*
 - Predicts update times of data based on program contexts

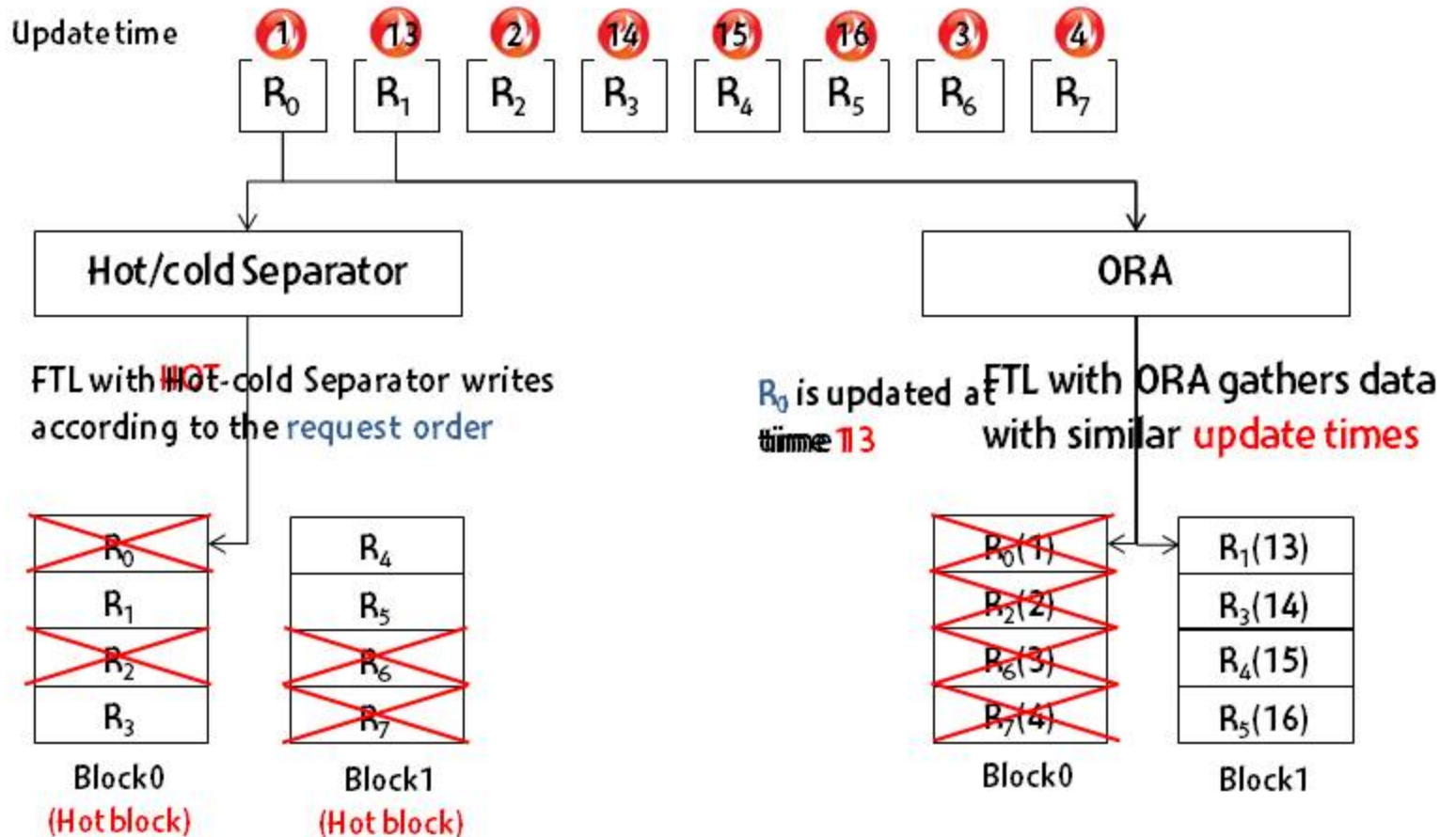
Outline

- Introduction
- **Motivation**
- **A Program Context-Aware Data Separation Technique**
- **Experimental Results**
- **Conclusions**

ORA: Oracle Predictor on Future Update Time

- **Perfect knowledge on future update times of data**
- Can sort data based on the future update times of data
- An FTL with ORA can gather data with similar update times into the same block
- Can be used as lower bound of GC overhead

Motivation Example



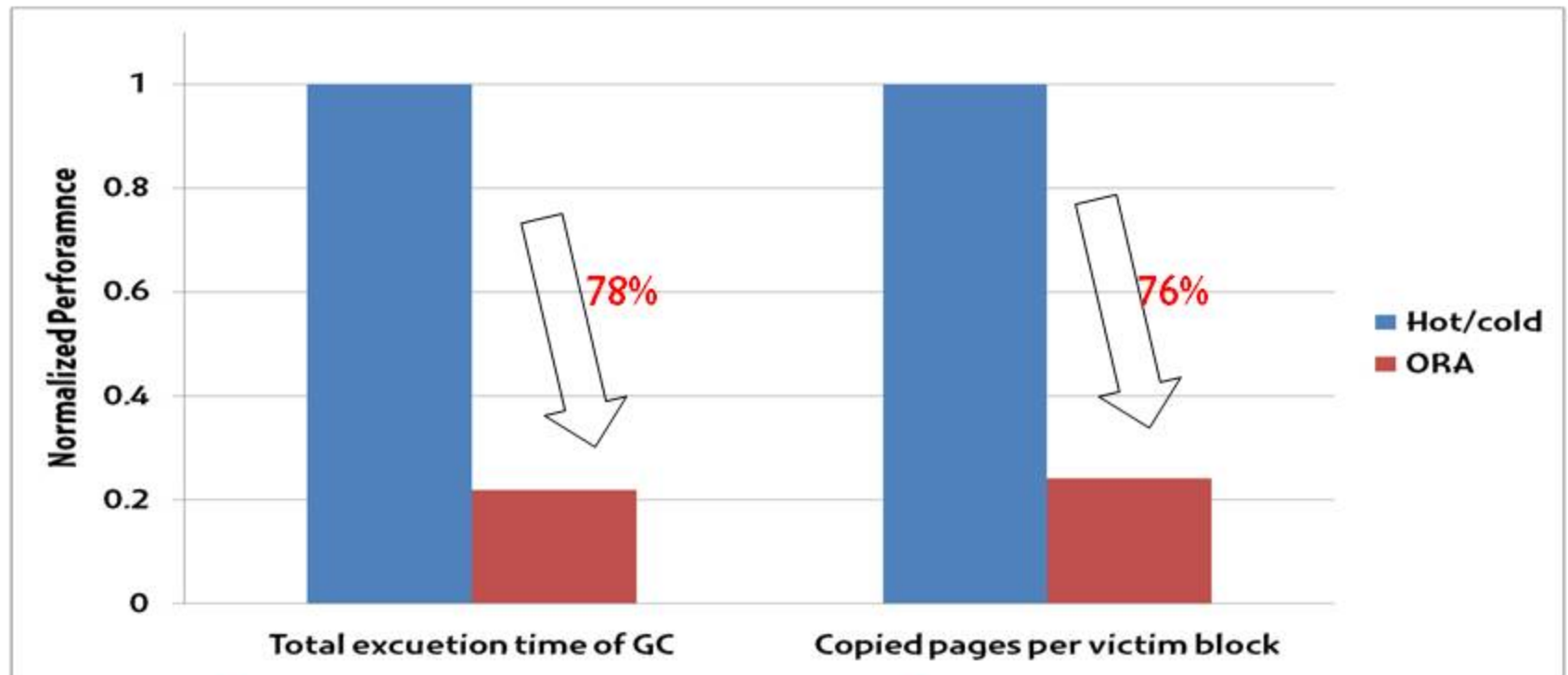
If a GC process was triggered at time 10,

4 copies + 2 erasures

1 erasure

Hot/cold Separator vs. ORA

- ORA can reduce GC overhead significantly



Update time is a more important factor in data separation technique than frequency of updates

Outline

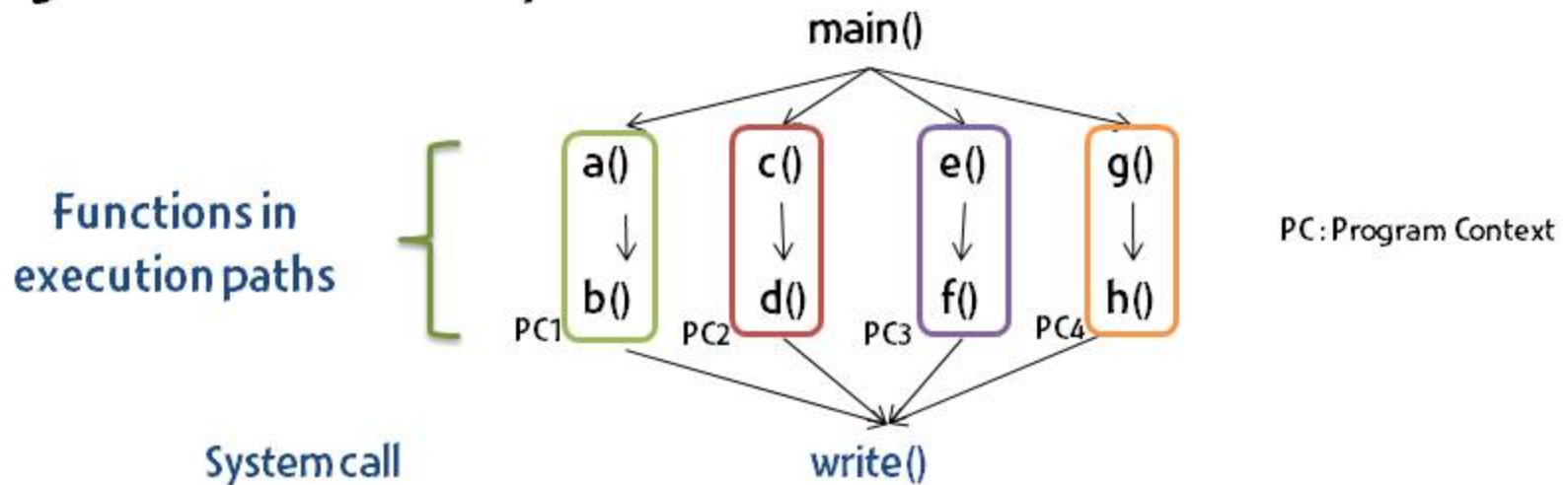
- Introduction
- Motivation
- A Program Context-Aware Data Separation Technique
 - Basic idea
 - Program Context and Update Behavior
 - Program Context-Based Heuristic
- Experimental Results
- Conclusions

Basic Idea

- **Predicts update times of data based on program behavior**
- **A program behaves similarly when the same program context is executed**
- **A program context is used to predict update times of data**

Overview of Program Context

- A program context represents an execution path which generates write requests



- Identification

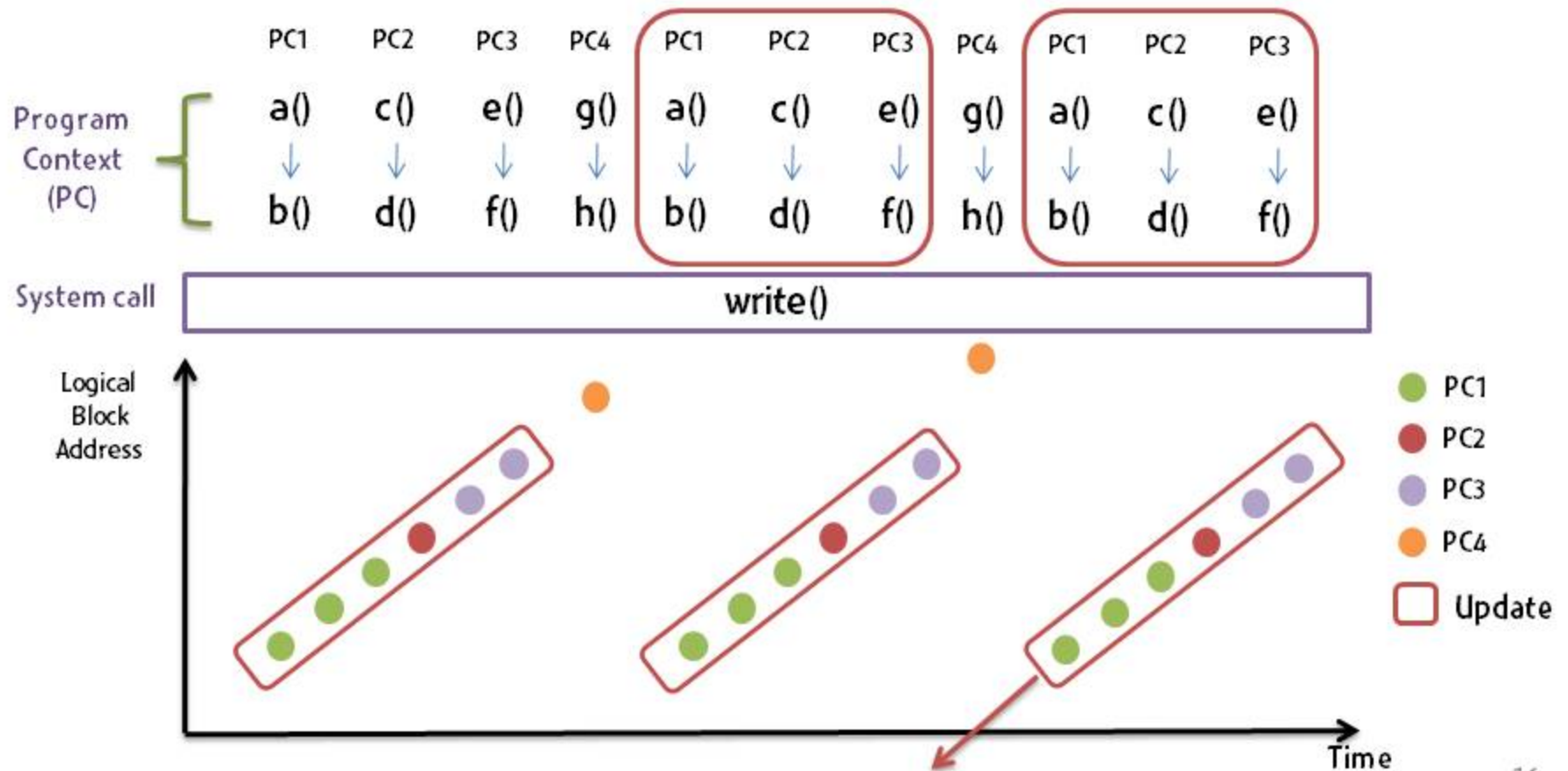
- Each program context is identified by **summing program counter values** of each execution path of function calls

Reference

Chris Gniady, and Ali R. Butt, and Y. Charlie Hu, "*Program Counter Based Pattern Classification in Buffer Caching*," OSDI, 2004

Program Context–Based Update Time Prediction

- We can indirectly predict future update times of data by exploiting program contexts



These data are updated in a similar period when PC1, PC2, and PC3 are executed

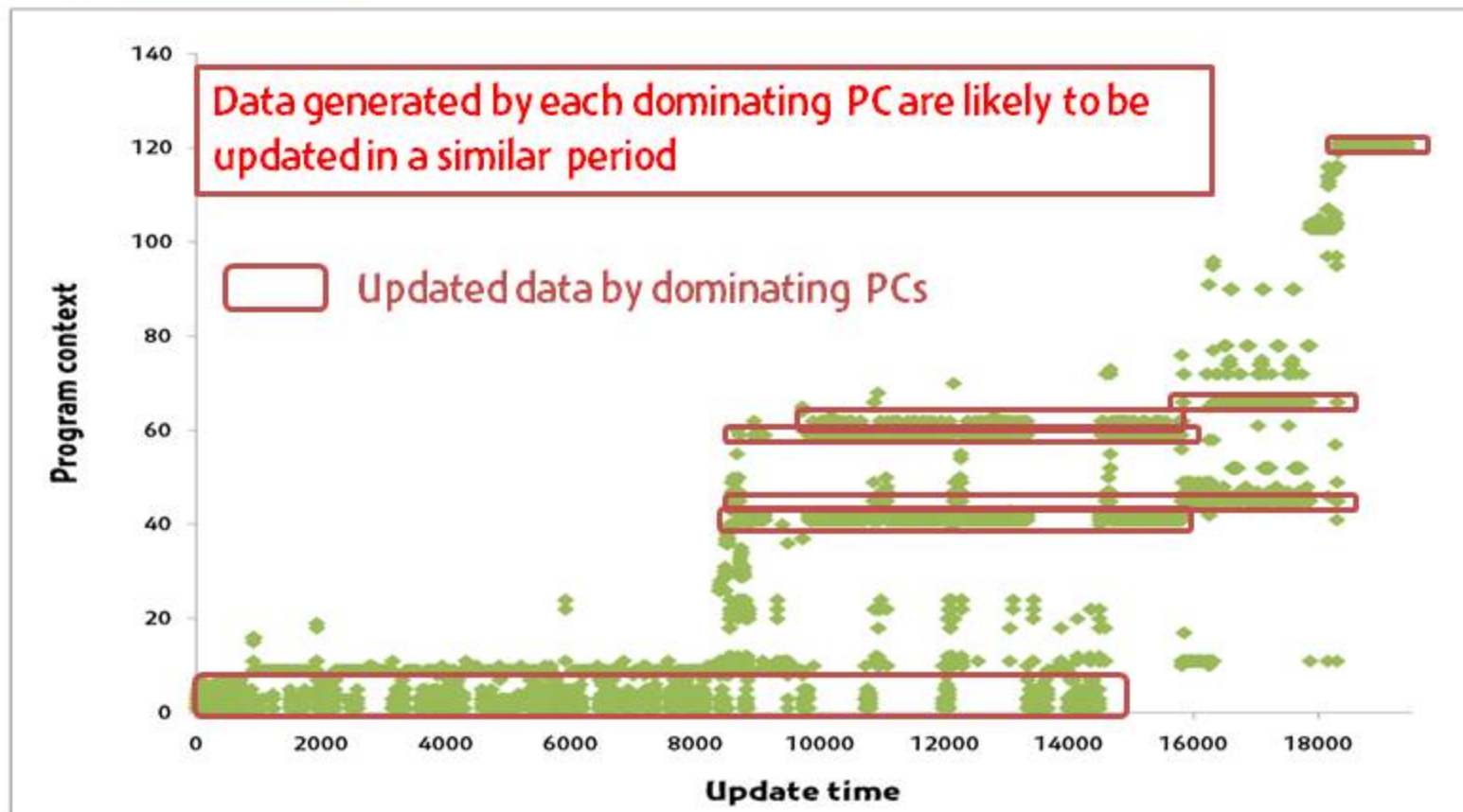
Program Context and Update Behavior

Observation 1

- A small number of PCs repeatedly generate a large number of update requests



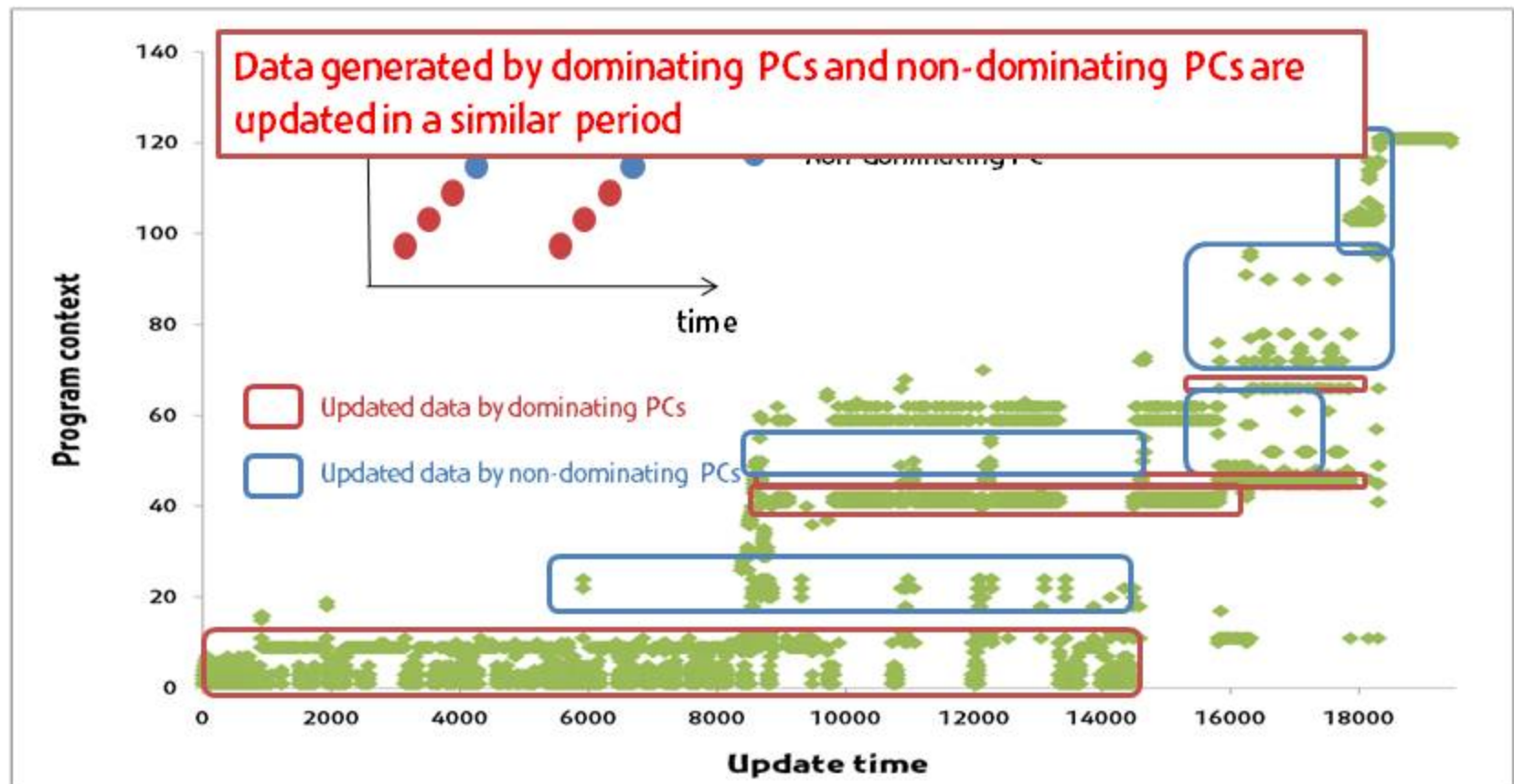
Dominating PC



Program Context and Update Behavior

Observation 2

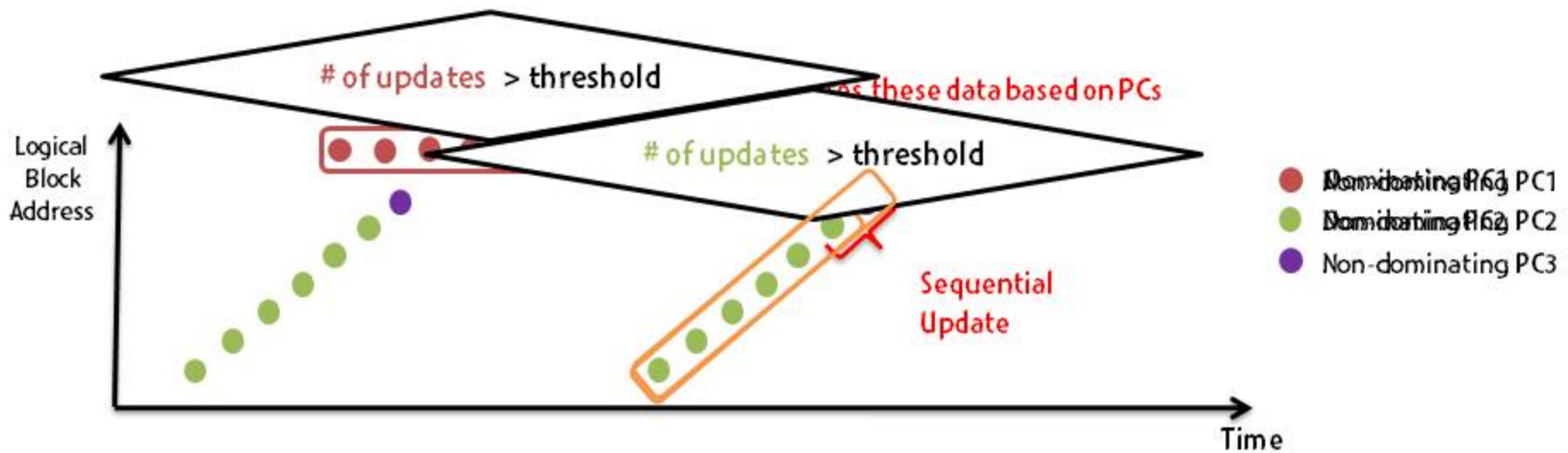
- Data generated by **non-dominating PCs** and **dominating PCs** form sequential update patterns



Program Context Based Data Separation Heuristic

- **Clusters data with similar update times**
 - Clusters data generated by a dominating PC
 - Clusters data generated by a non-dominating PC and adjacent dominating PC if update patterns of the data are sequential

Separating Data using Program Contexts



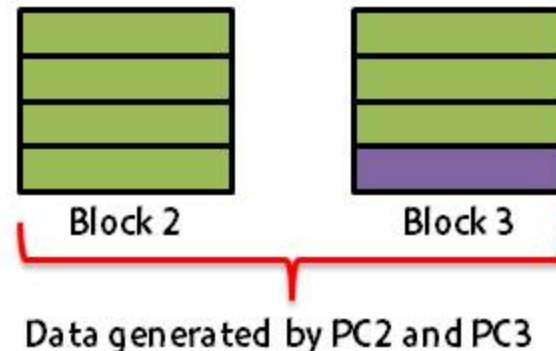
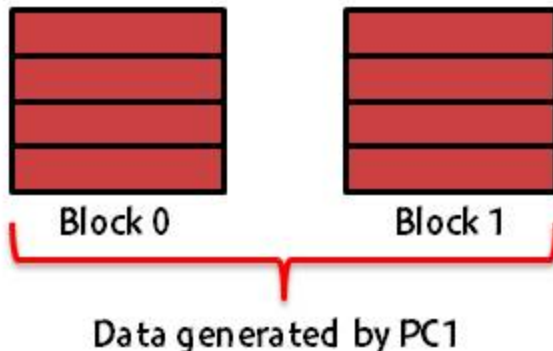
Simultaneously updated group 1

Data generated by PC1

Simultaneously updated group 2

Data generated by PC2 and PC3

FTL with our proposed data separator stores data based on simultaneously updated group



Outline

- Introduction
- Motivation
- **A Program Context-Aware Data Separation Technique**
- **Experimental Results**
- **Conclusions**

Experimental Environments (1)

- Used a trace-driven NAND flash memory simulator
 - Parameters

Flash Translation Layer	Mapping Scheme	Page-level mapping
	GC Triggering	5%
Flash memory	Read Time (1 page)	25usec
	Write Time (1 page)	200usec
	Erase Time (1 block)	1200usec

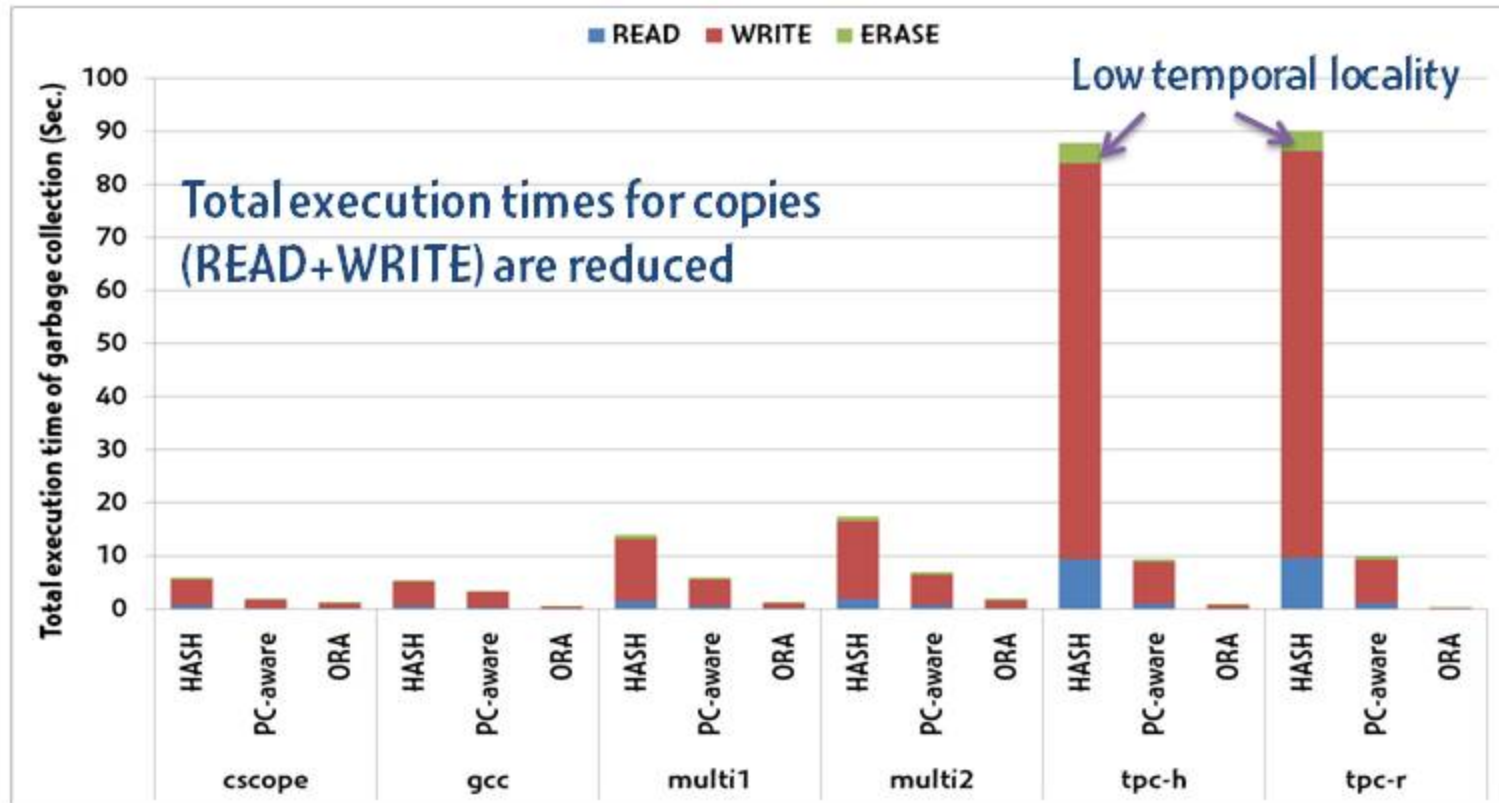
- Techniques for comparison
 - HASH: Hash-based hot/cold separation technique
 - ORA: Oracle predictor on future update times of data

Experimental Environments (2)

- Benchmarks characteristics

Benchmarks	Scenario	The number of writes (unit: page)	The number of updates (unit: page)
cscope	Linux source code examination	17575	15398
gcc	Building Linux Kernel	10394	3840
viewperf	Performance measurement	7003	119
tpc-h	Accesses to database	23522	20910
tpc-r	Accesses to database	21897	18803
multi1	cscope + gcc	28400	19428
multi2	cscope + gcc + viewperf	35719	20106

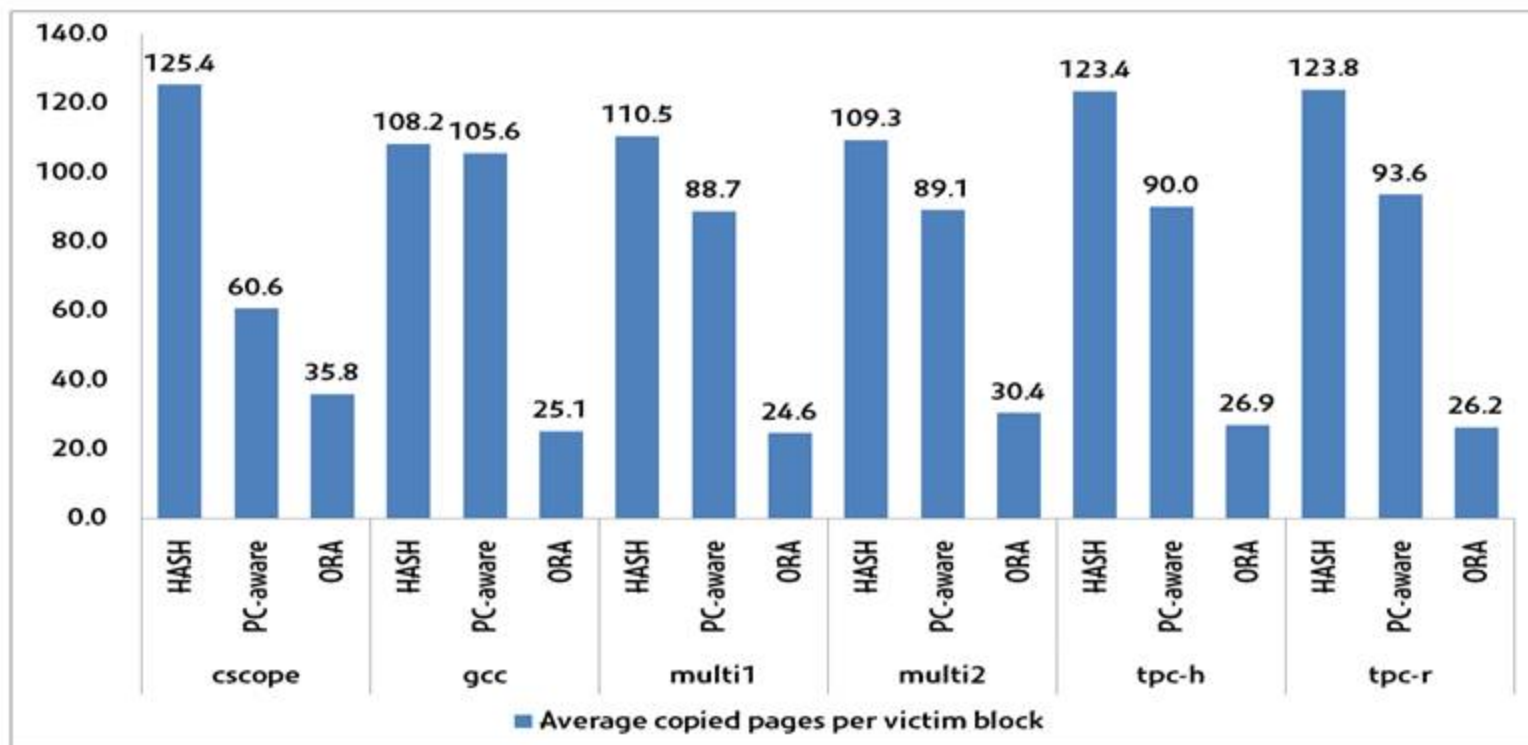
Result 1: Total Execution Time of GC



Reduces the total execution time of garbage collection on average 58% over HASH

Result 2: The number of copied pages per victim block

Reduces the number of copied pages per victim block on average 25% over HASH



Not as accurate as ORA in predicting update times of data

Conclusions

- **Evaluated the performance of existing hot/cold separator against ORA**
 - Update time is a more important factor than update frequency in separation technique
- **Proposed a novel program context-aware data separation technique**
 - Reduces GC overhead by about 58% over a hot/cold separator
- **Future works**
 - Improve our technique further to reduce performance gap with ORA
 - Exploit program context to explore efficient wear-leveling management