DFTL: A Flash Translation Layer Employing Demand-based Selective Caching of Page-level Address Mappings

Aayush Gupta, Youngjae Kim, Bhuvan Urgaonkar
Pennsylvania State University

Speaker: Sehwan Lee

Contents

Introduction

- Using Flash Memory in Enterprise-scale Storage
  - Reliability issue
  - Performance issue
- The Flash Translation Layer
  - Out-of-place updates
  - Page/Block-level FTL
  - Hybrid FTL
    - Poor garbage collection behavior
    - Too many tunable parameters
    - Not exploiting the temporal locality in access

Introduction

- Research Contributions
  - DFTL - purely page-mapped FTL
  - FlashSim to evaluate the efficacy of DFTL
  - Using a number of realistic enterprise-scale workloads

Background and Related Work

- Basics of Flash Memory Technology

Flash Type	Data Unit Size			Access Time		
	Page (Bytes)		Block (Bytes)	Page READ (us)	Page WRITE (us)	Block ERASE (ms)
	Data	OOB				
Small Block	512	16	(16K+512)	41.75	226.75	2
Large Block	2048	64	(128K+4K)	130.9	405.9	2

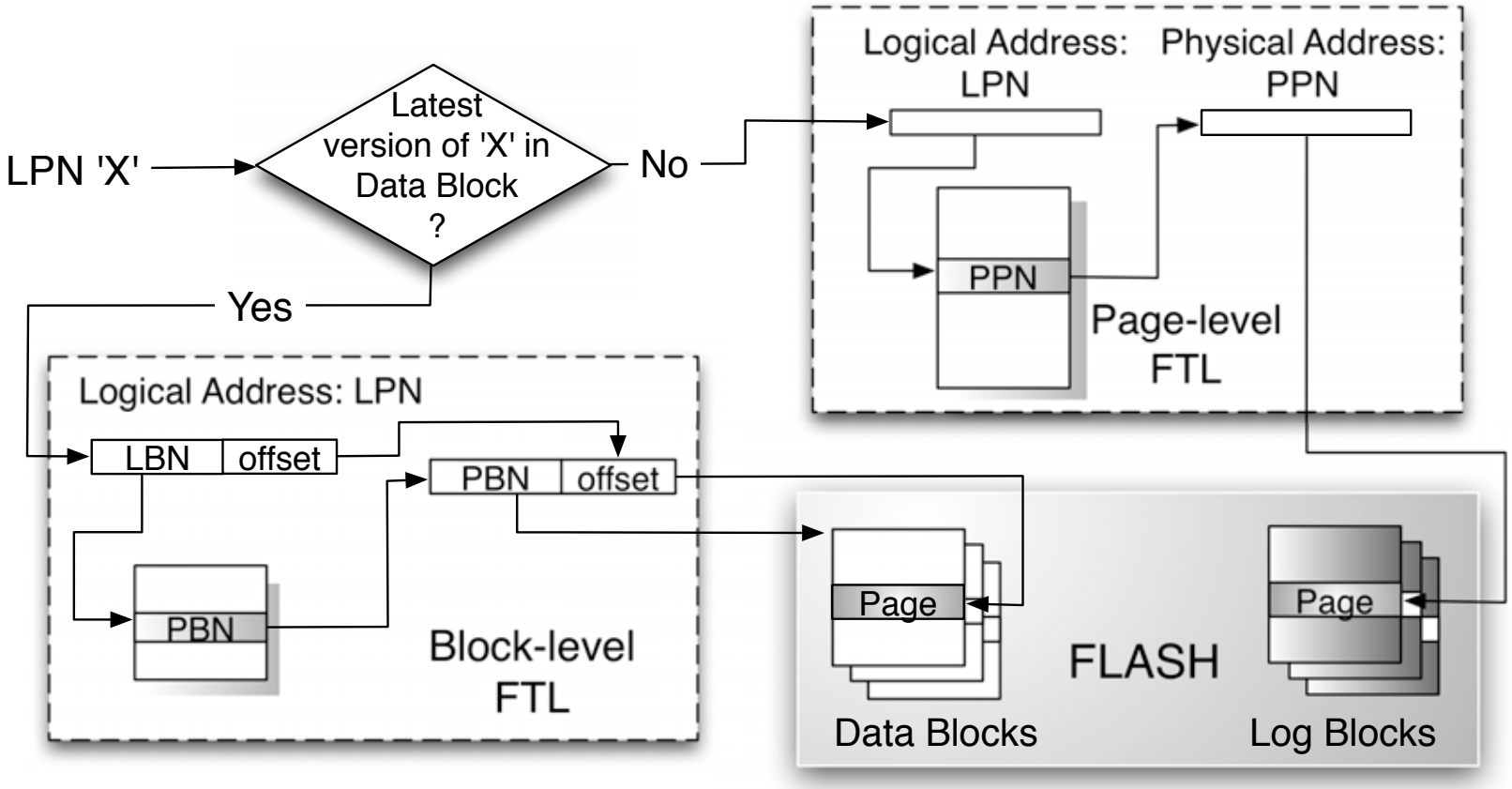
- 10K-1M erase operations -> wear-leveling issue

Background and Related Work

- Page-level and Block-level FTL Schemes
  - Page-level FTL scheme
    - Merit: compact and efficient utilization of blocks
    - Demerit: large translation table
  - Block-level FTL scheme
    - Merit: mall translation table
    - Demerit: garbage collection overheads + sequential programming constraint in a large block flash memory

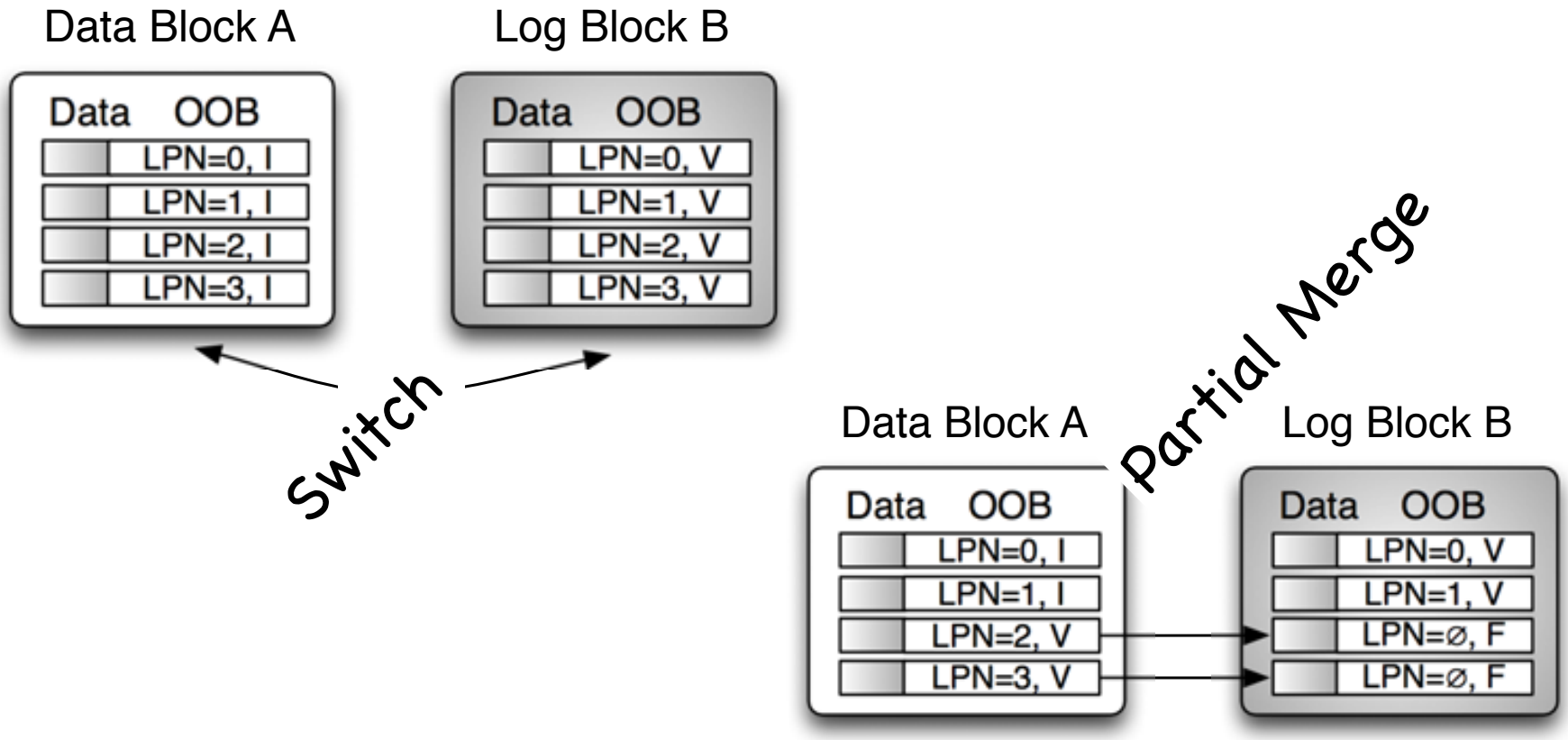
Background and Related Work

- A Generic Description of Hybrid FTL Scheme



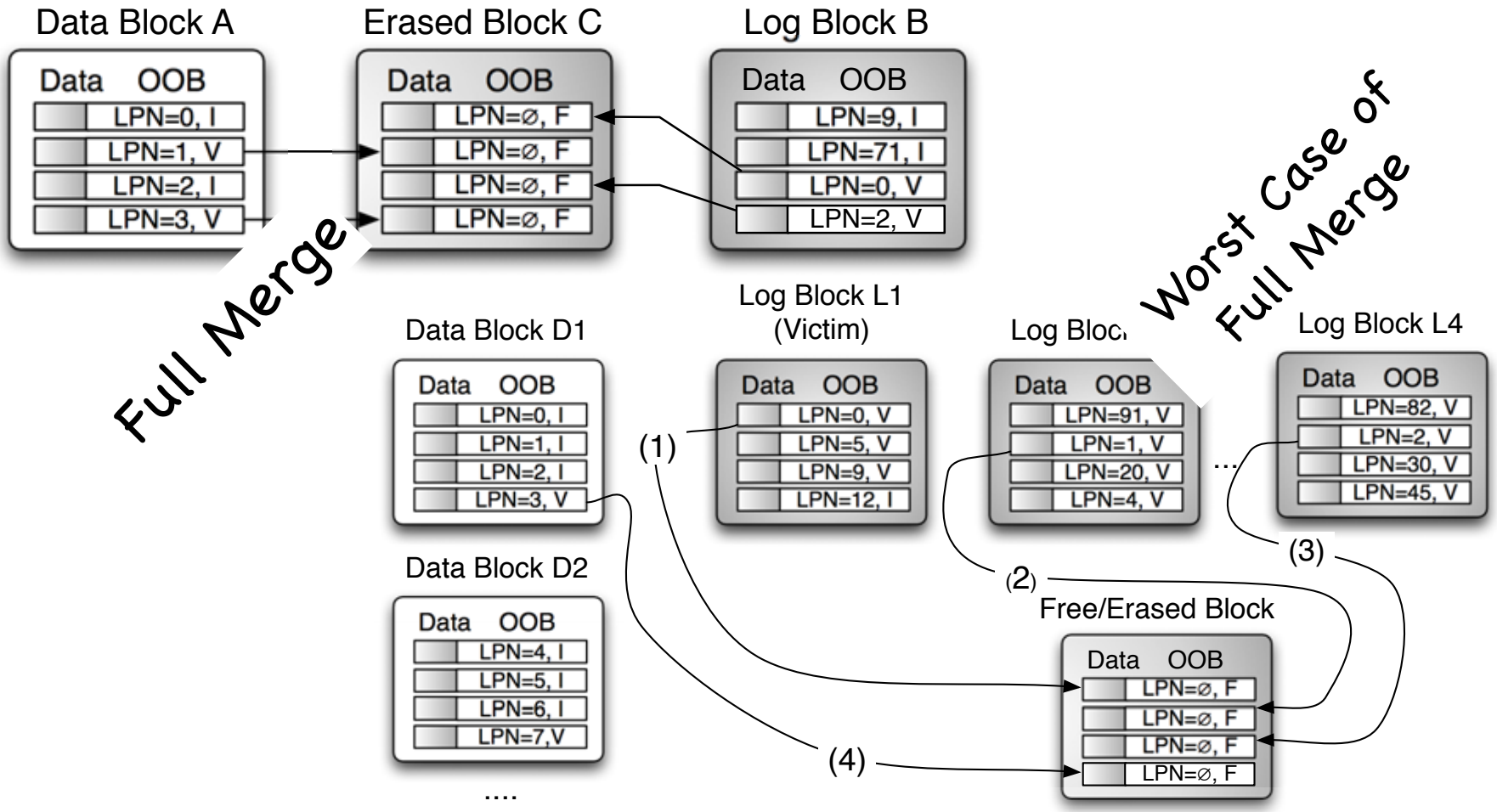
Background and Related Work

- Garbage Collection in Hybrid FTLs



Background and Related Work

Garbage Collection in Hybrid FTLs

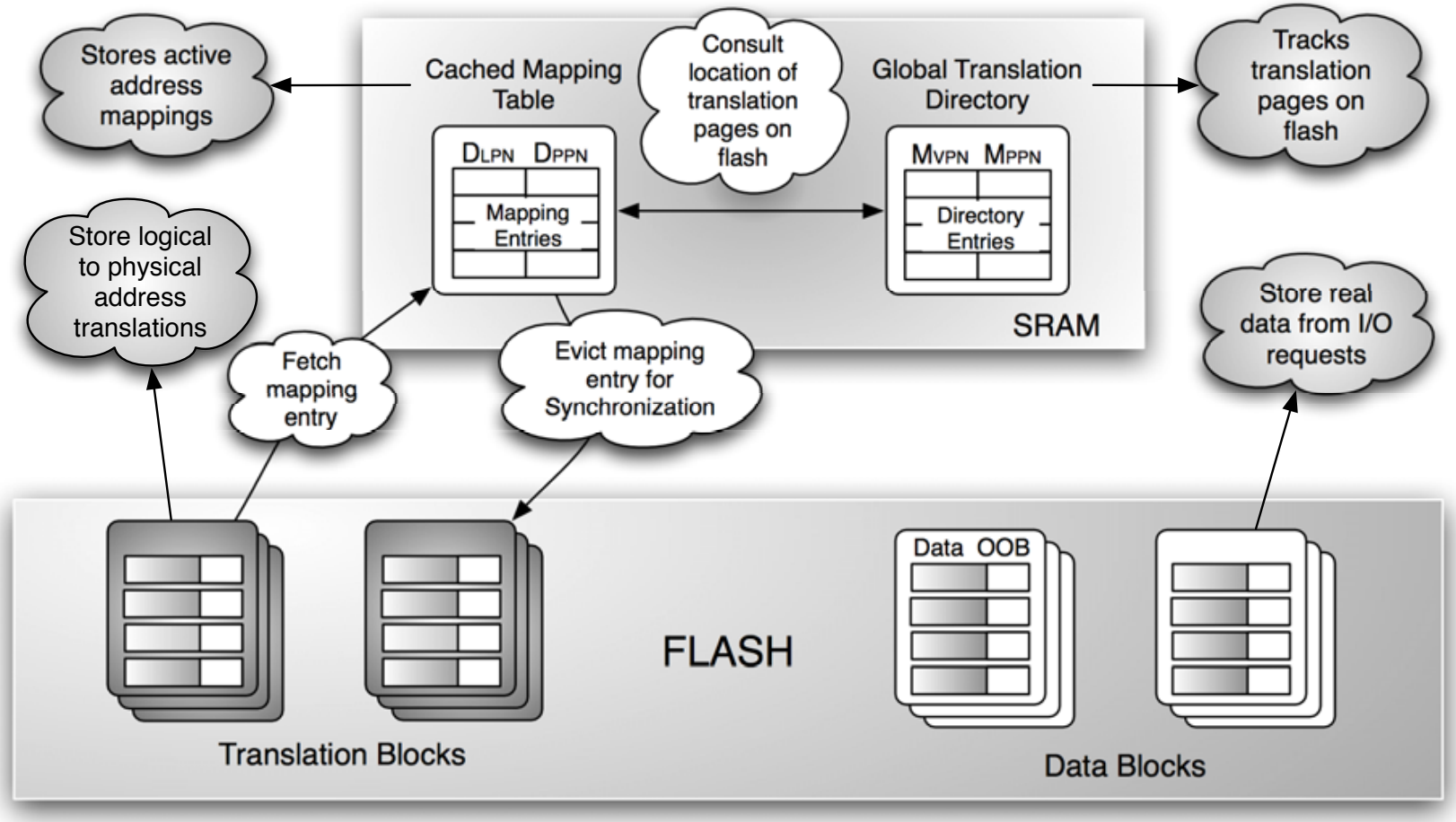


Background and Related Work

- State-of-The-Art FTLs
  - Block Associative Sector Translation (BAST)
    - data block:log block=1:m
    - "log block thrashing" when many small random writes
  - Fully Associative Sector Translation (FAST)
    - a sequential log block + random log blocks
  - SuperBlock FTL
    - using spatial locality in workloads
  - Locality-Aware Sector Translation (LAST)
    - FAST + hot/cold identification to reducing full merge costs

Design of DFTL

- DFTL Architecture
  - Data Pages and Translation Pages



Design of DFTL

- Logical to Physical Address Translation
  - Global Mapping Table (GMT) and Global Translation Directory (GTD)
    - GMT: the entire logical-to-physical address translation set in the flash memory
    - CMT: active mapping present in SRAM
    - GTD: translation set which keeps track of all these translation pages
  - Overhead in DFTL Address Translation
    - The worst case: two translation page reads + one translation page write

Design of DFTL

- Logical to Physical Address Translation
  - DFTL Address Translation Process

Input: Request's Logical Page Number ($\texttt{request}_{lpn}$), Request's Size ($\texttt{request}_{size}$)
Output: NULL
while $\texttt{request}_{size} \neq 0$ **do**
  if $\texttt{request}_{lpn}$ *miss in Cached Mapping Table* **then**
    if *Cached Mapping Table is full* **then**
      /* Select entry for eviction using segmented LRU replacement algorithm */
    end
    $Translation\_Page_{request} \leftarrow$ **consult_GTD**$(request_{lpn})$
    /* Load map entry of the request from flash into Cached Mapping Table */
    load_entry$(Translation\_Page_{request})$
  end
  $request_{type} \leftarrow$ Data Block
  $request_{ppn} \leftarrow$ **CMT_lookup**$(request_{lpn})$
  DFTL_Service_Request$(request)$
  $request_{size}$--
end

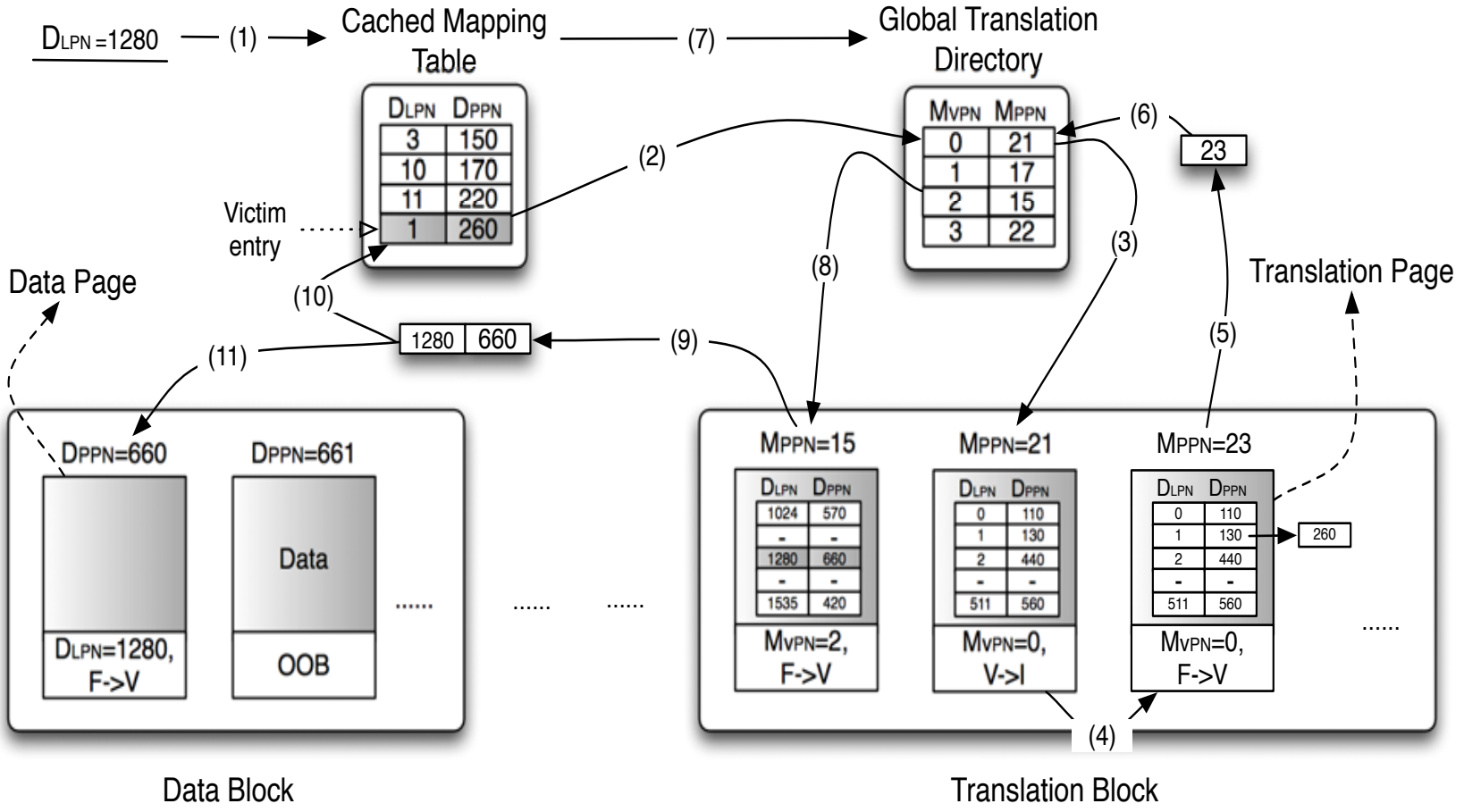
Design of DFTL

- Logical to Physical Address Translation
  - DFTL Address Translation Process

if *Cached Mapping Table is full* **then**
  /* Select entry for eviction using segmented LRU replacement algorithm */
  $victim_{lpn} \leftarrow \mathbf{select\_victim\_entry}()$
  if $victim_{last\_mod\_time} \neq victim_{load\_time}$ **then**
    /*$victim_{type}$: Translation or Data Block
    $Translation\_Page_{victim}$: Physical Translation-Page Number containing victim entry */
    $Translation\_Page_{victim} \leftarrow \mathbf{consult\_GTD}(victim_{lpn})$
    $victim_{type} \leftarrow$ Translation Block
    $\mathbf{DFTL\_Service\_Request}(victim)$
  end
  $\mathbf{erase\_entry}(victim_{lpn})$
end

Design of DFTL

- Logical to Physical Address Translation
  - DFTL Address Translation Process



Design of DFTL

- Read/Write Operation and Garbage Collection
  - Two blocks which is maintained by DFTL
    - Current Data Block + Current Translation Block
  - $GC_{threshold}$
    - a high watermark which invokes a garbage collection
    - victim in a translation block
      - copying the valid pages to the current translation block
      - updating the GTD
    - victim in a data block
      - copying the valid pages to the current data block
      - updating all the translation pages and CMT entries associated with them
  - Reducing overheads techniques
    - lazy copying
      - updating the CMT for those data pages whose mappings are present in it
    - batch updates

Design of DFTL: Demand-based Page-mapped FTL

- Comparison of Existing State-of-the-art FTLs with DFTL

	Replacement Block FTL[1]	BAST [2]	FAST [19]	SuperBlock [13]	LAST [20]	DFTL	Ideal Page FTL
FTL type	Block	Hybrid	Hybrid	Hybrid	Hybrid	Page	Page
Mapping Granularity	Block	DB-Block LB - Page	DB-Block LB-Page	SB-Block LB/Blocks within SB-Page	DB/Sequential LB - Block Random LB - Page	Page	Page
Division of Update Blocks (M)	-	-	1 Sequential + (M-1) Random	-	(m) Sequential-(M-m) (Hot and Cold)	-	-
Associativity of Blocks (Data:Update)	(1:K)	(1:M)	Random LB-(N:M-1) Sequential LB-1:1	(S:M)	Random LB-(N:M-m) Sequential LB-(1:1)	(N:N)	(N:N)
Blocks available for updates	Replacement Blocks	Log Blocks	Log Blocks	Log Blocks	Log Blocks	All Data Blocks	All Blocks
Full Merge	Yes	Yes	Yes	Yes	Yes	No	No

- Full Merge/Partial Merge
- Random Write Performance
- Block Utilization

The FlashSim Simulator

- FlashSim
  - A simulation framework for flash-based storage systems
  - Modular architecture with the capability to model a holistic flash-based storage environment
  - Simulating multiple parts
    - Device drivers+controllers+caches+flash devices+various interconnects
  - Implementing multiple scheme
    - a block based FTL scheme (replacement-block scheme)
    - FAST
    - DFTL
    - an idealized page-based FTL

Experimental Results

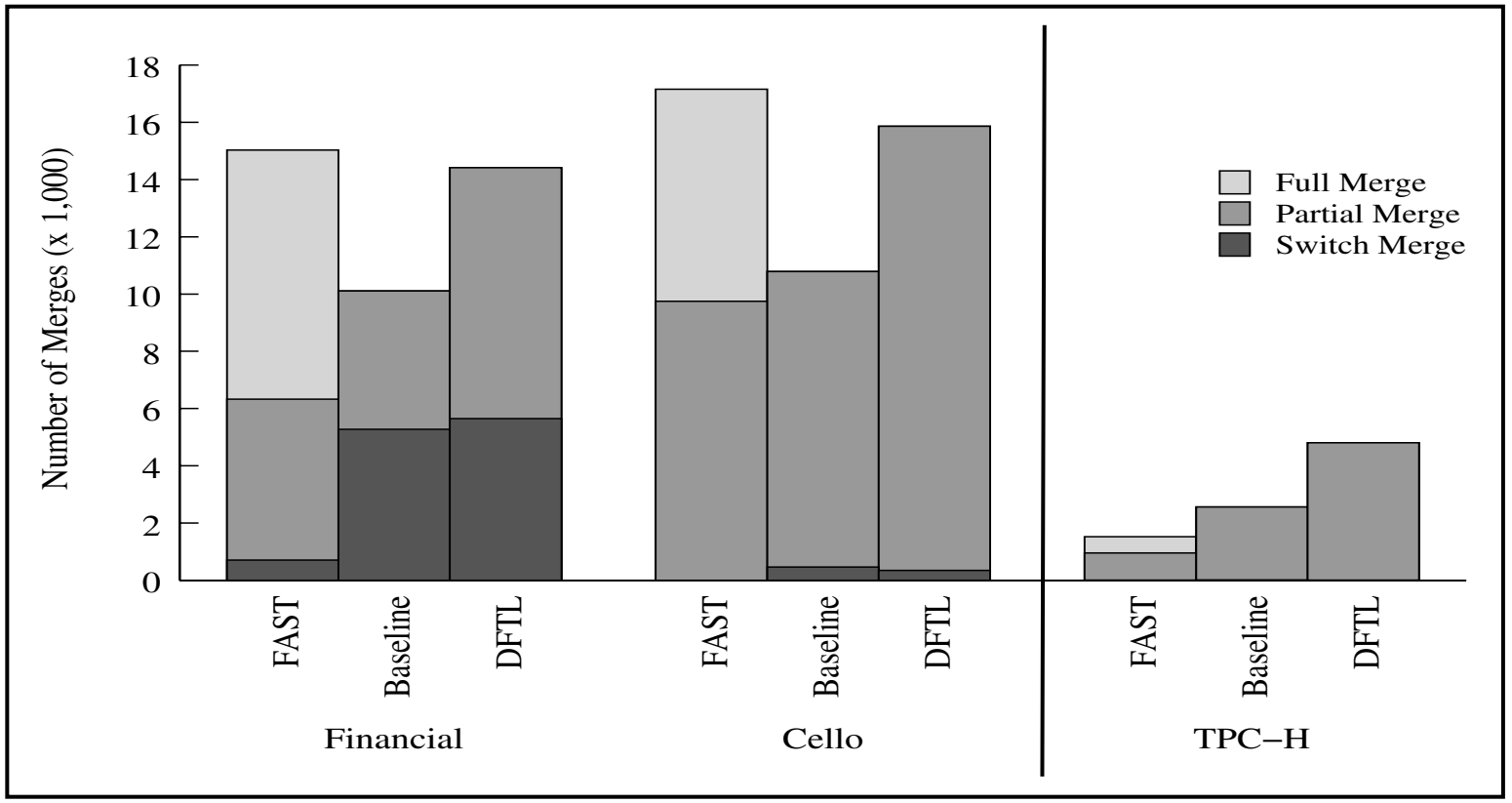
- Evaluation Setup
  - Workloads

Workloads	Avg. Req. Size (KB)	Read (%)	Seq. (%)	Avg. Req. Inter-arrival Time (ms)
Financial [25]	4.38	9.0	2.0	133.50
Cello99 [10]	5.03	35.0	1.0	41.01
TPC-H [28]	12.82	95.0	18.0	155.56
Web Search [26]	14.86	99.0	14.0	9.97

  - Performance Metrics
    - Device service time
    - indicators of the garbage collector's efficacy + response time as seen at the I/O driver

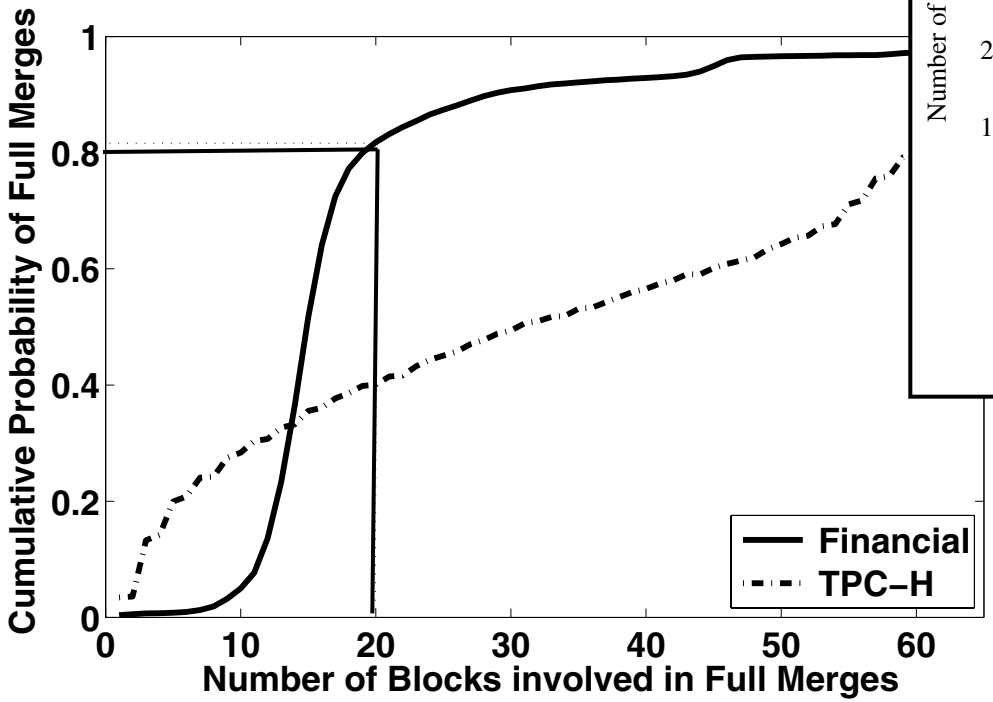
Experimental Results

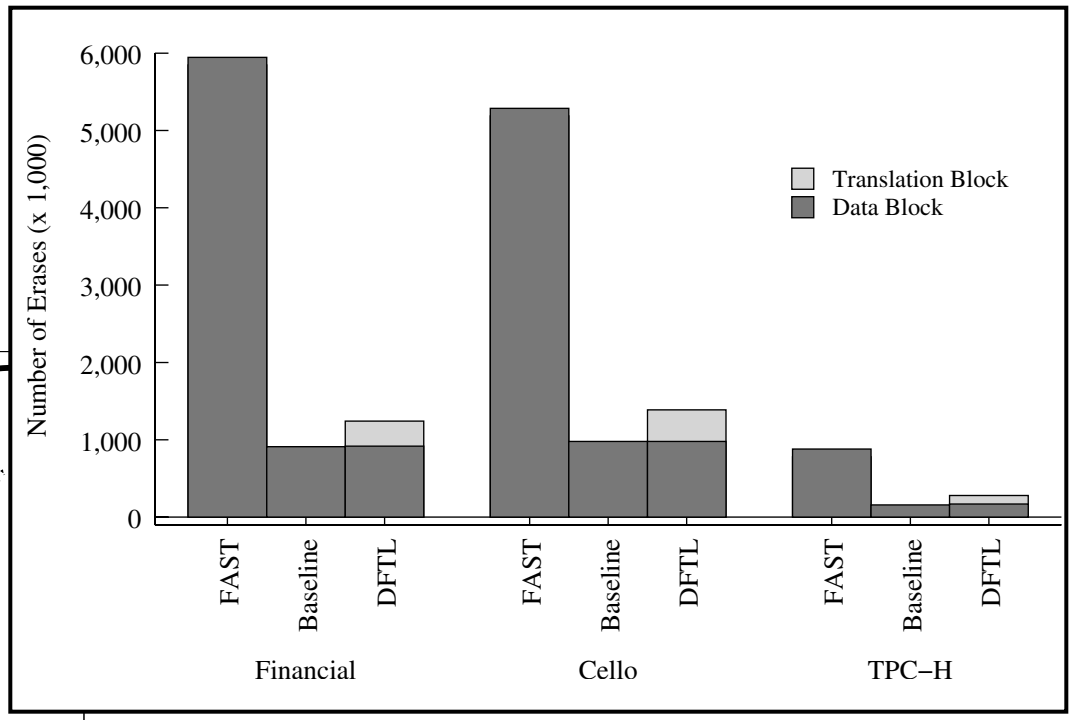
- Analysis of Garbage Collection and Address Translation Overheads
  - Switch Merges



Experimental Results

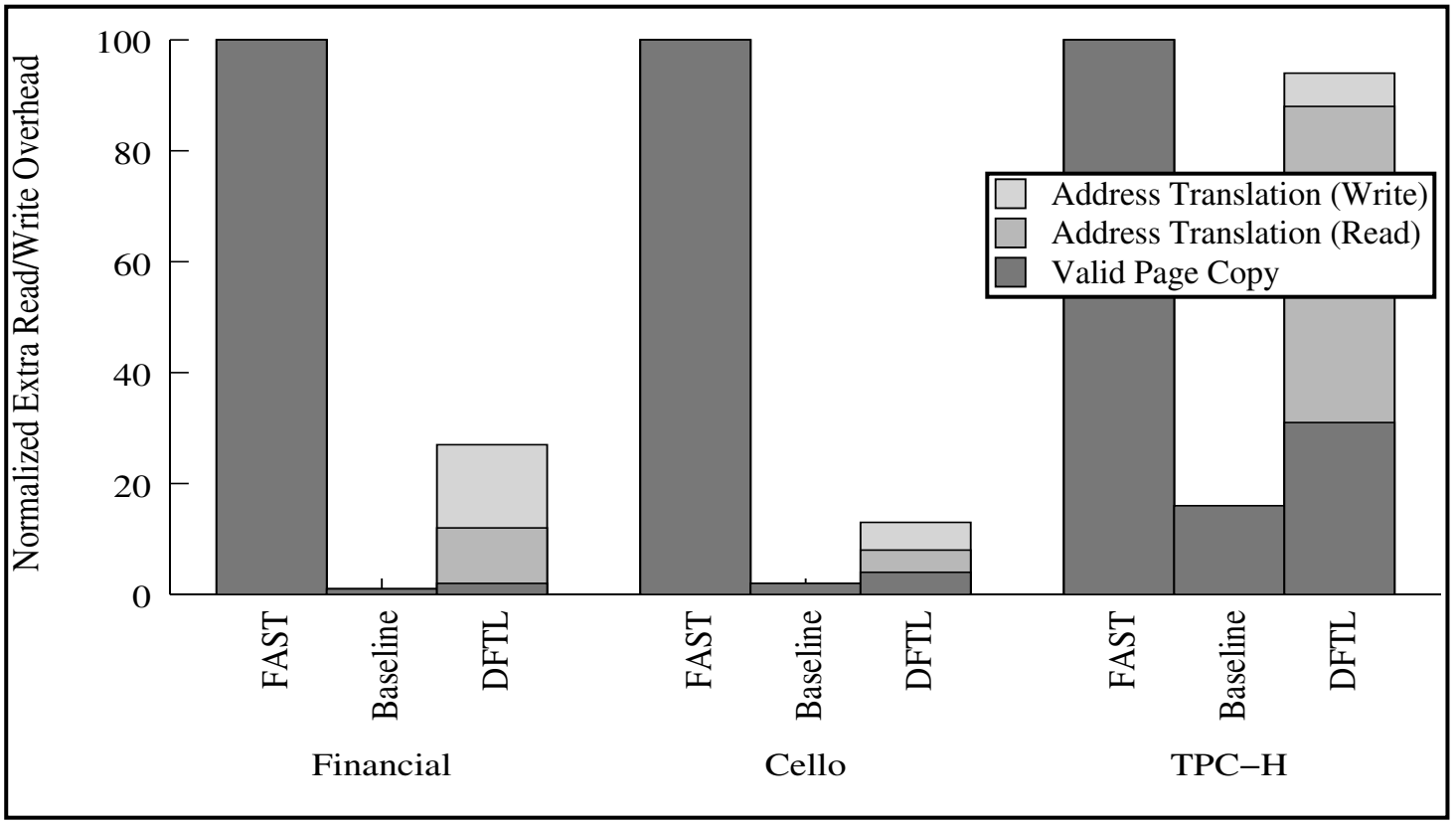
- Analysis of Garbage Collection and Address Translation Overheads
  - Full Merges





Experimental Results

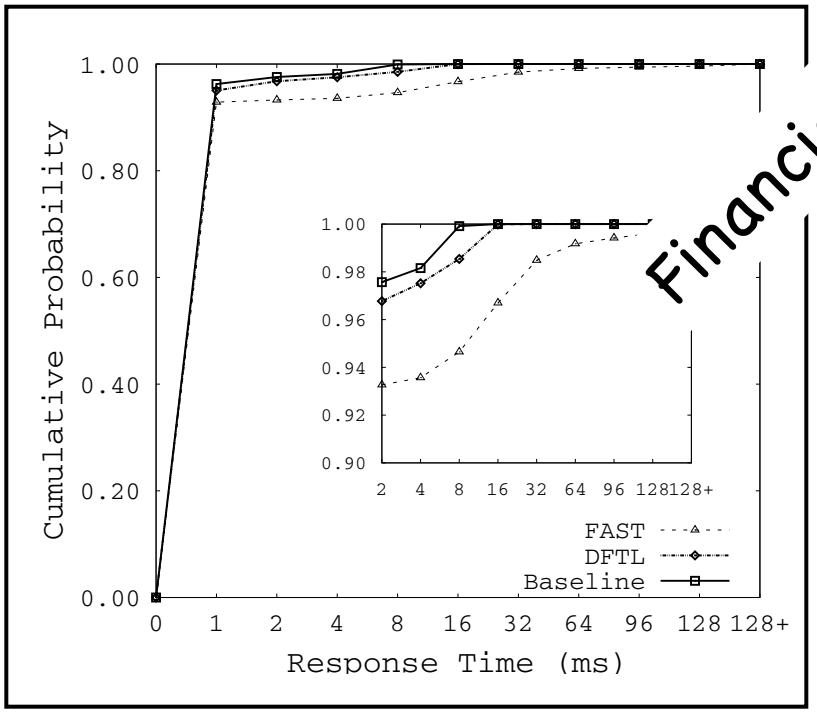
- Analysis of Garbage Collection and Address Translation Overheads
  - Translation and Valid Page Copying Overheads



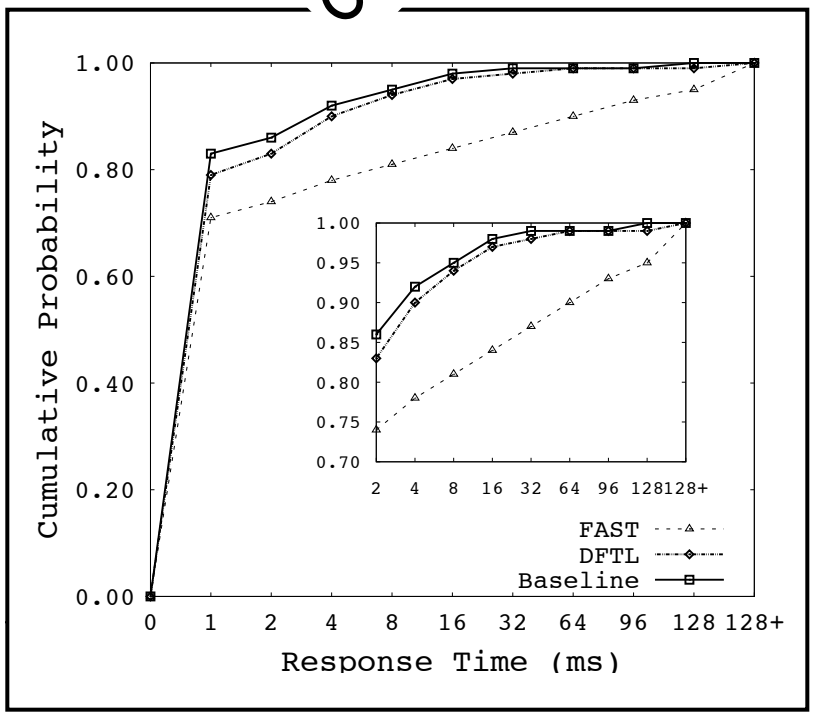
Experimental Results

- Performance Analysis

Financial Trace

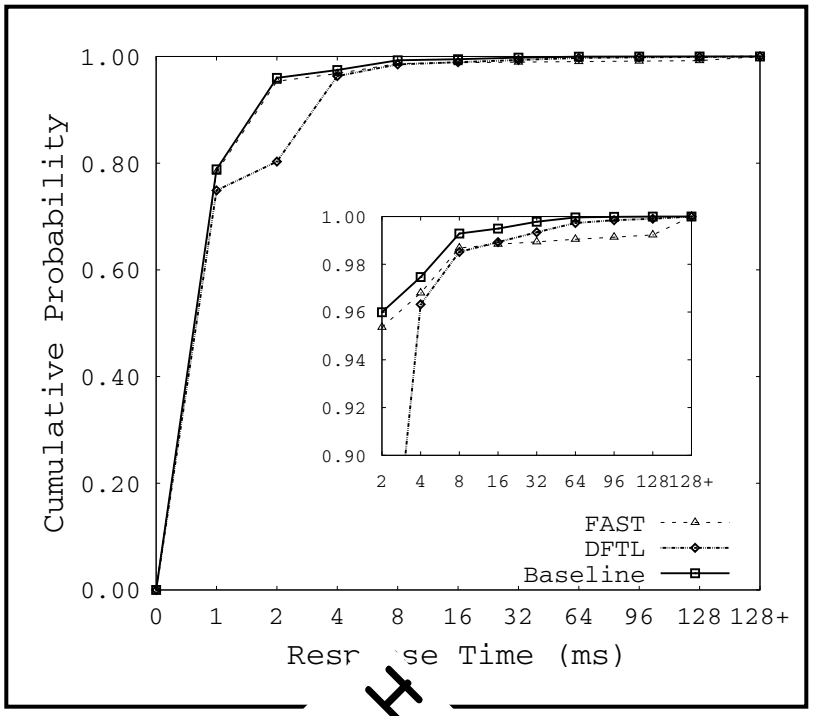


Cello99



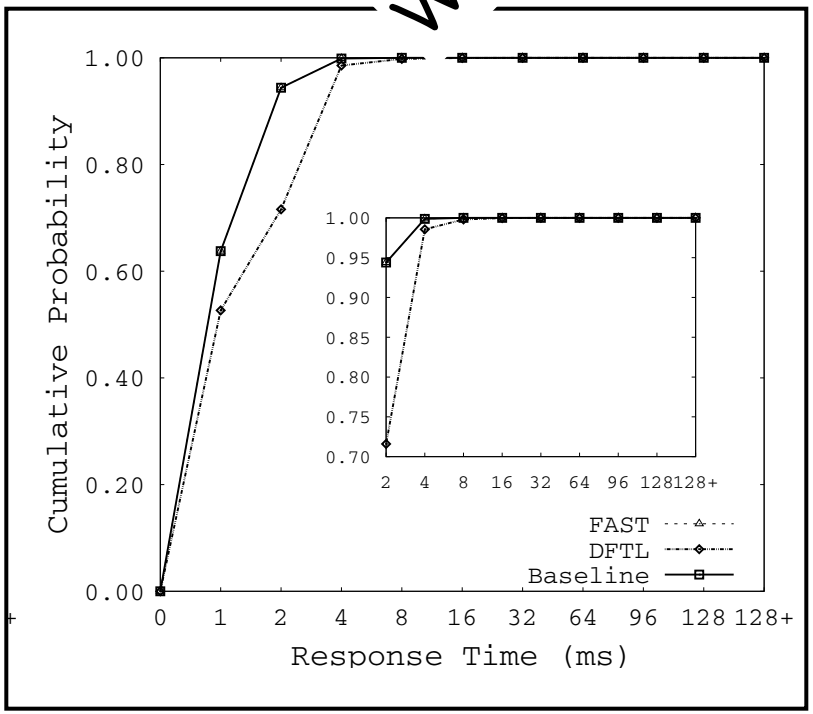
Experimental Results

- Performance Analysis



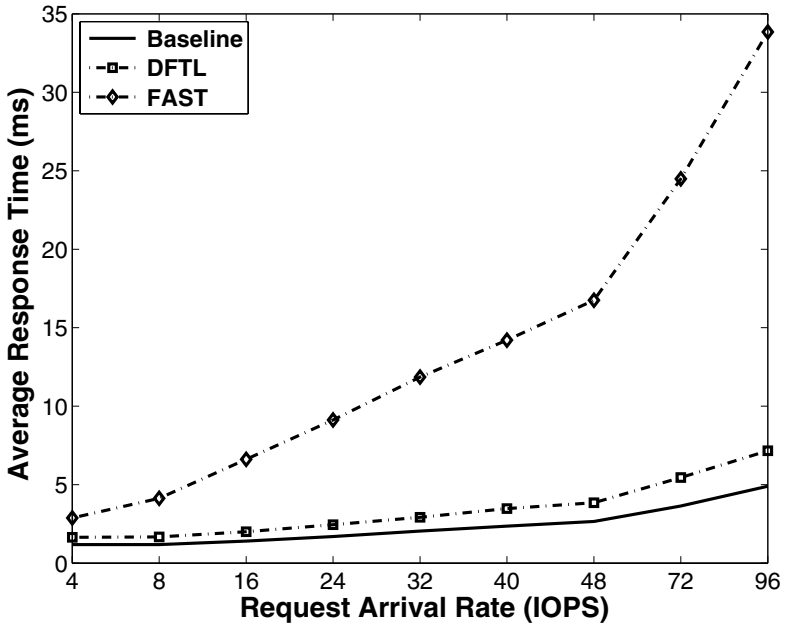
TPC-H

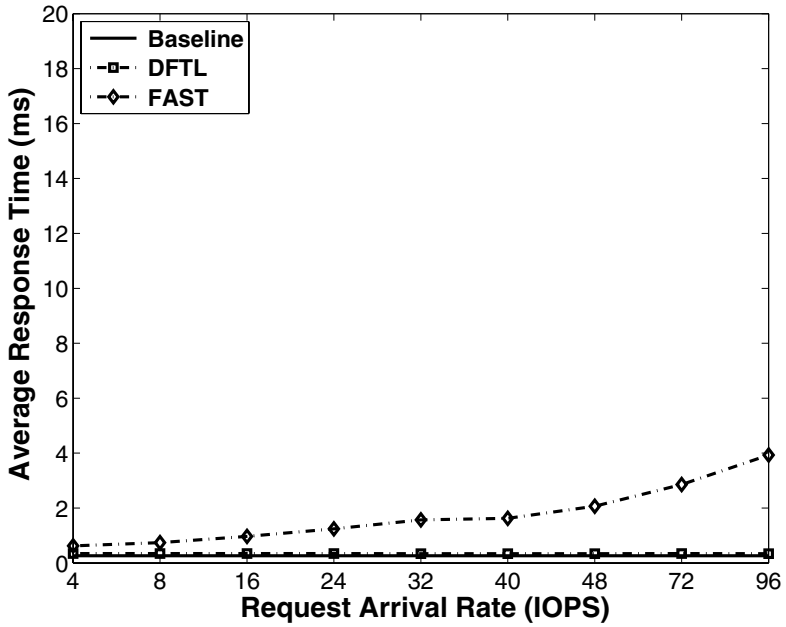
Web-Search



Experimental Results

- Exploring a Wider Range of Workload Characteristics

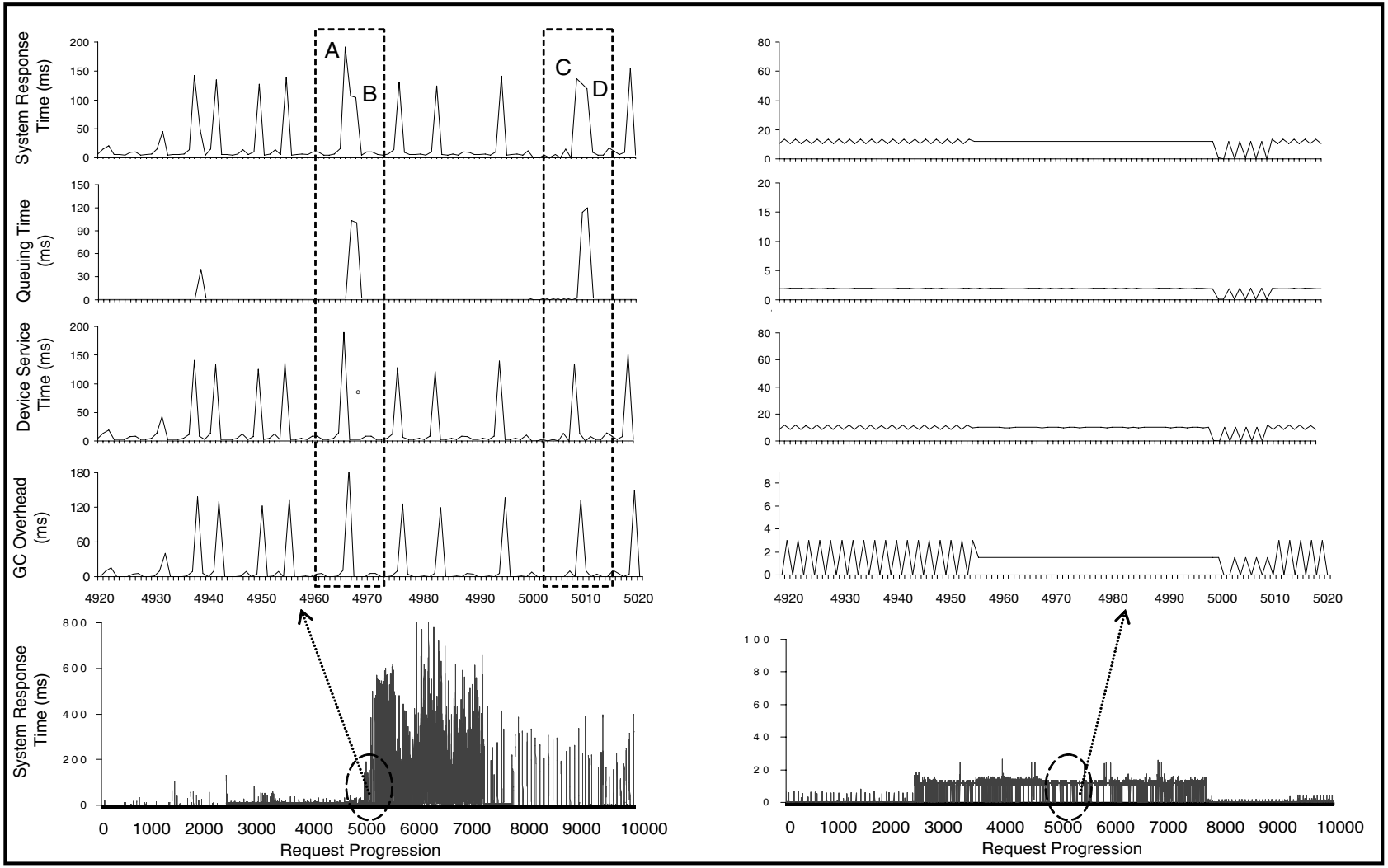




Random write dominant workload vs. Sequential write dominant workload

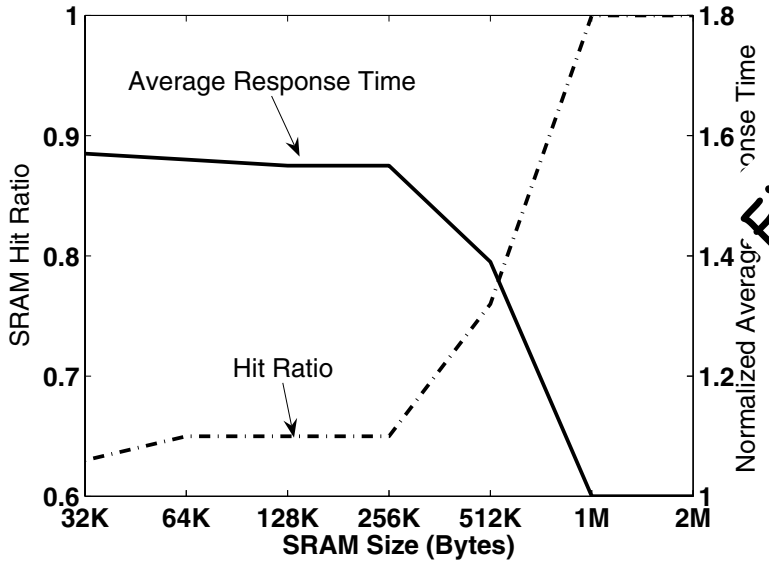
Experimental Results

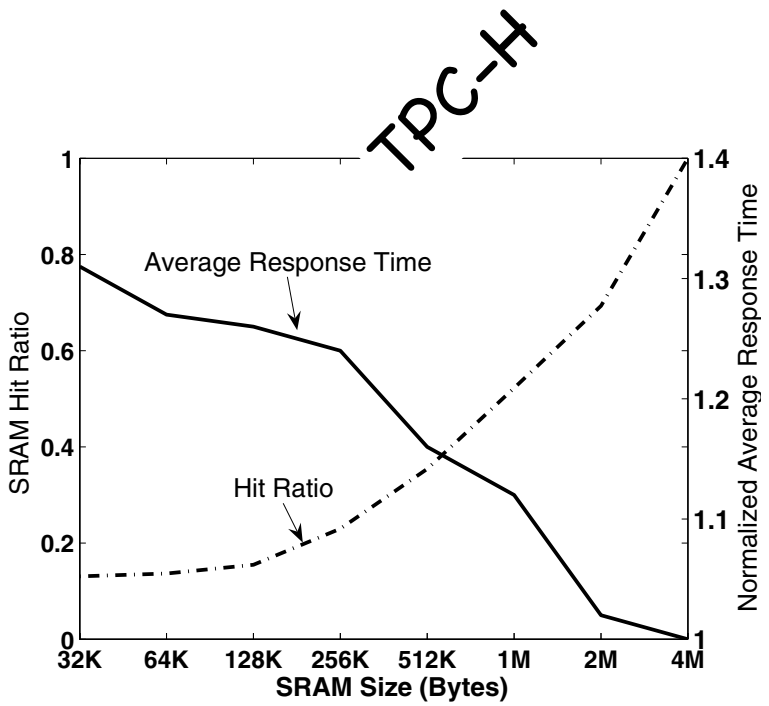
- Microscopic Analysis



Experimental Results

- Impact of SRAM size





Concluding Remarks

- We proposed a complete paradigm shift in the design of the FTL with our Demand-based Flash Translation Layer (DFTL)

- DFTL is...
  - improved performance
  - reduced garbage collection overhead
  - improved overload behavior
  - free from tunable parameters