

Towards Flash-based Enterprise Databases

Sep. 22nd, 2008

Sang-Won Lee

Sungkyunkwan University, Korea



SKKU VLDB Lab.

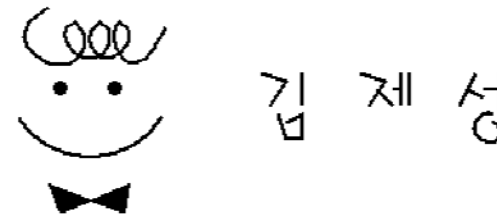
▪ Research Directions

- **Vision:** “Flash is disk, disk is tape, and tape is dead.”(by Jim Gray)
- **Strategy:** “Do only what only you can do.”(by E. W. Dijkstra)
- **Research Goal:** “Database systems on flash memory” pioneer

▪ Recent Achievements

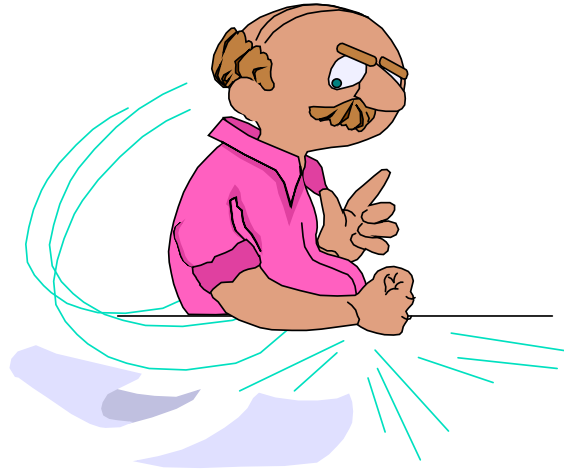
- **FAST:** One of the most efficient FTL mechanisms (ACM Transactions on Embedded Computing Systems, 2007)
- **IPL:** An innovative database storage scheme on flash memory (ACM SIGMOD 2007)
- A Case for Flash Memory SSD in Enterprise Database Applications (ACM SIGMOD 2008)

Flash-based Database Research: Motivations



Real Motivations

SCI, SCI, SCI, ...



The VERY Real Motivations

- Rule of the game in storage market is changing
- Hardware: Dr. Masuoka, Samsung, Toshiba, ..
- Hwang's Law vs. Moore's Law
- SW: OS, File System, FTL, DBMS, ...

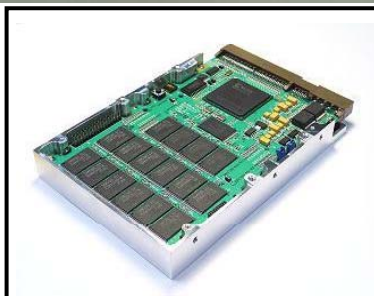


Flash Memory Market

- Mobile phone, digital camera, iPod, MP3 etc (1 ~ 10 G)
 - Flash **already wins** over harddisk
- PC, Laptop : over 10G
 - 2008: **the war begins**
- Enterprise server storage: 100G ~ 100T
 - 2009 ~ 2010?
 - In my opinion, flash is more attractive in enterprise rather than PC / Laptop market

Flash Memory Solid State Disk (SSD)

- Provides a block-device interface identical to a hard disk, but uses flash memory as a storage medium



50 years champion

VS

A new challenger!



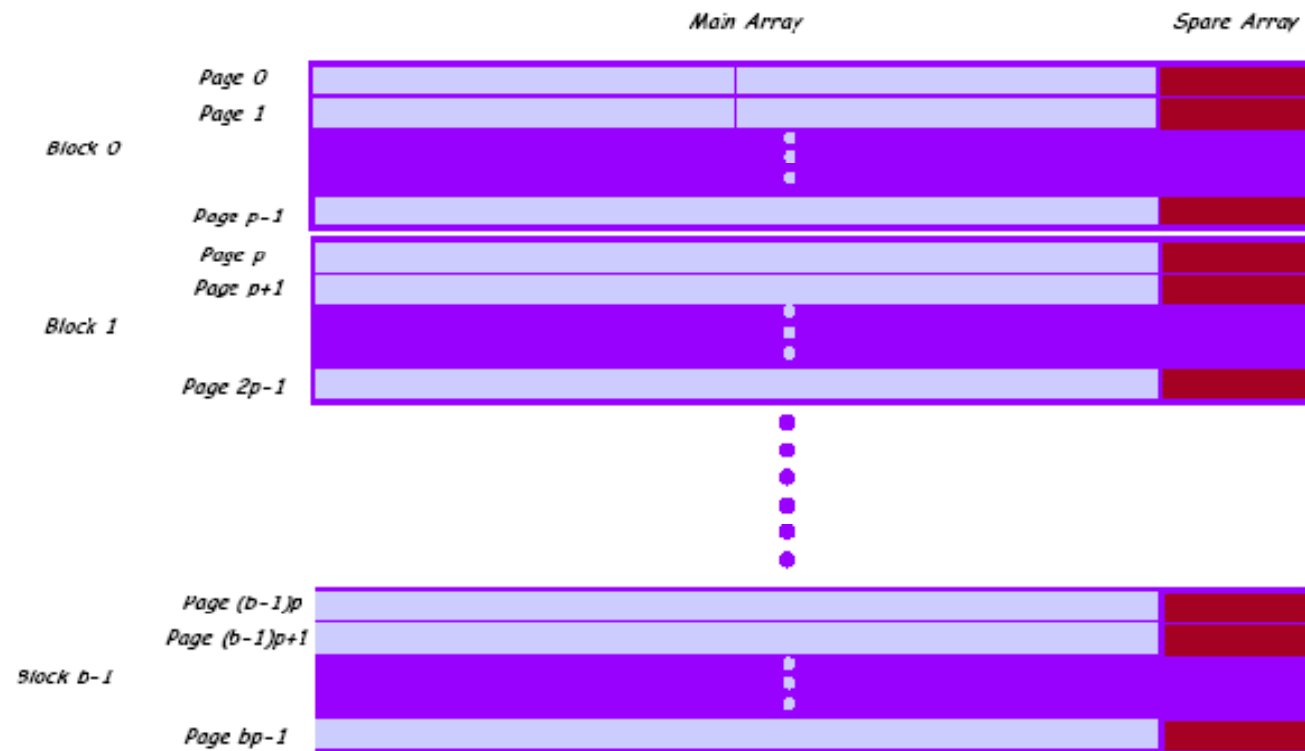
Identical
Interface



A Quick Intro to Flash Memory



Flash Organization



	Page Size	Block Size
Small Block NAND	512 B	16 KB
Large Block NAND	2 KB	128 KB

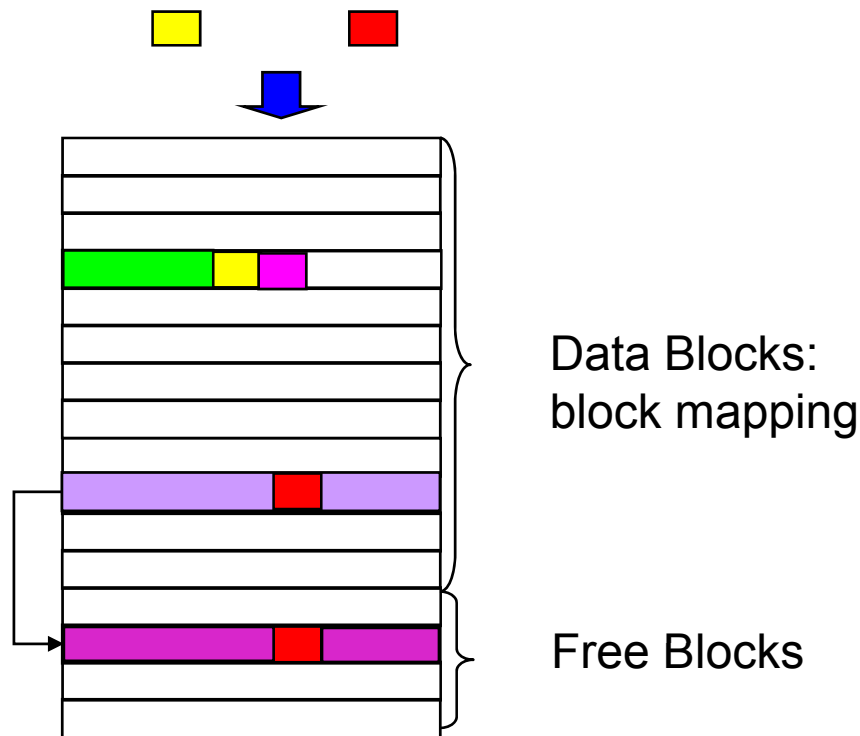
Flash Performance: Chip Level

Media	Access time		
	Read	Write	Erase
Magnetic [†] Disk	12.7 ms (2 KB)	13.7 ms (2 KB)	N/A
NAND Flash [‡]	80 μ s (2 KB)	200 μ s (2 KB)	1.5 ms (128 KB)

- Flash characteristics
 - Erase-before-write (No in-place update)
 - Asymmetric read/write speed
 - No mechanical latency

Sequential Append-only Writes vs. Random Overwrites

- E.g. 2K page write: new write vs. overwrite in a naïve way



- New write: 0.2ms
- Overwrite
 - 63 2K-reads = 6.3ms
 - 63 2K-writes = 12.6ms
 - 1 2K-write = 0.2ms
 - 1 erase = 1.5ms
 - Total = 20.6ms

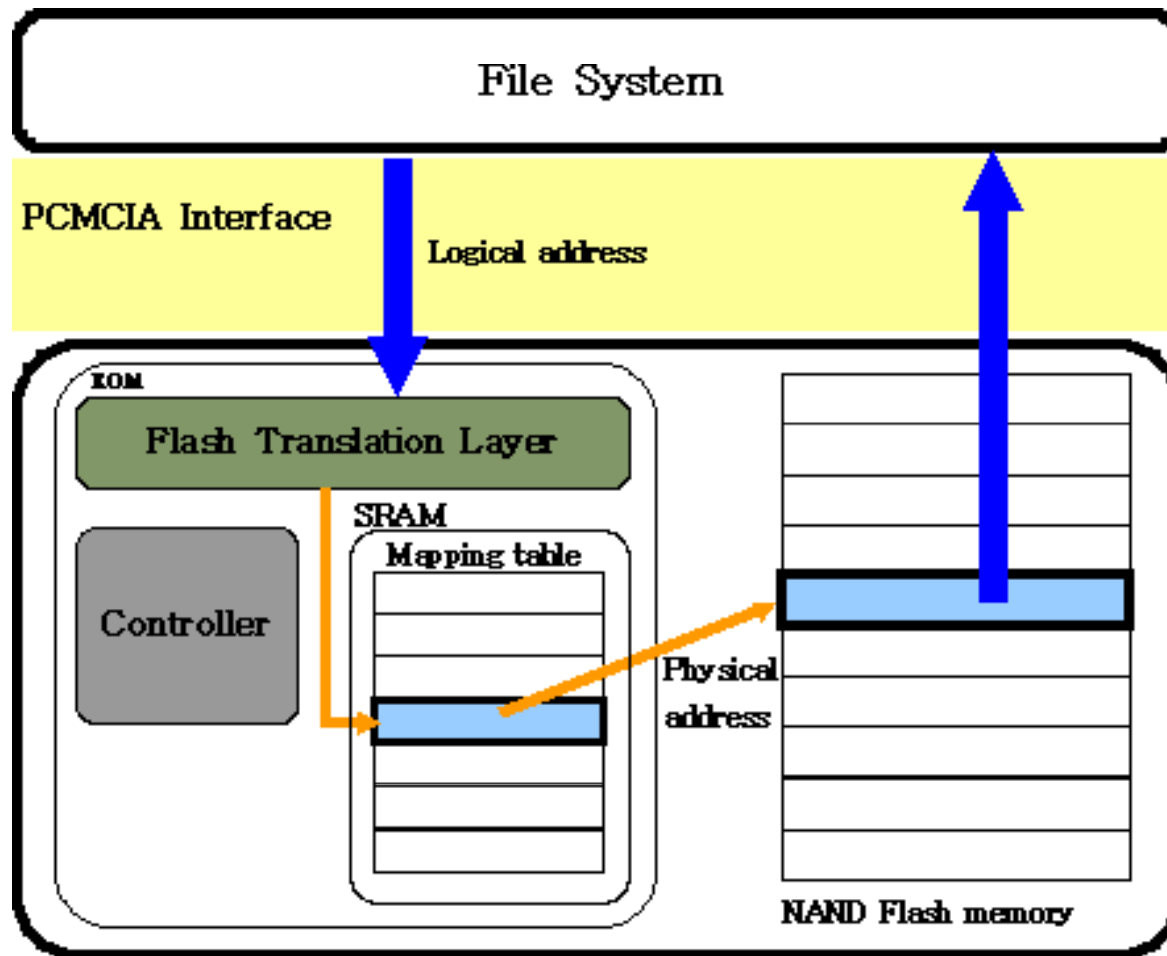
Flash Performance: SSD Level

Table 1: Measurements of a current flash disk's performance.

Device	Sequential	Random 8KB	Price \$	Power	iops/\$	iops/watt
SCSI 15k rpm	75 MBps	200 iops	500\$	15 watt	0.5	13
SATA 10k rpm	60 MBps	100 iops	150\$	8 watt	0.7	12
Flash- read	53 MBps	2,800 iops	?? 400\$	0.9 watt	7.0	3,100
Flash - write	36 MBps	27 iops	?? 400\$	0.9 watt	0.07	30

(Source: Jim Gray's slide)

NAND FlashSSD Architecture

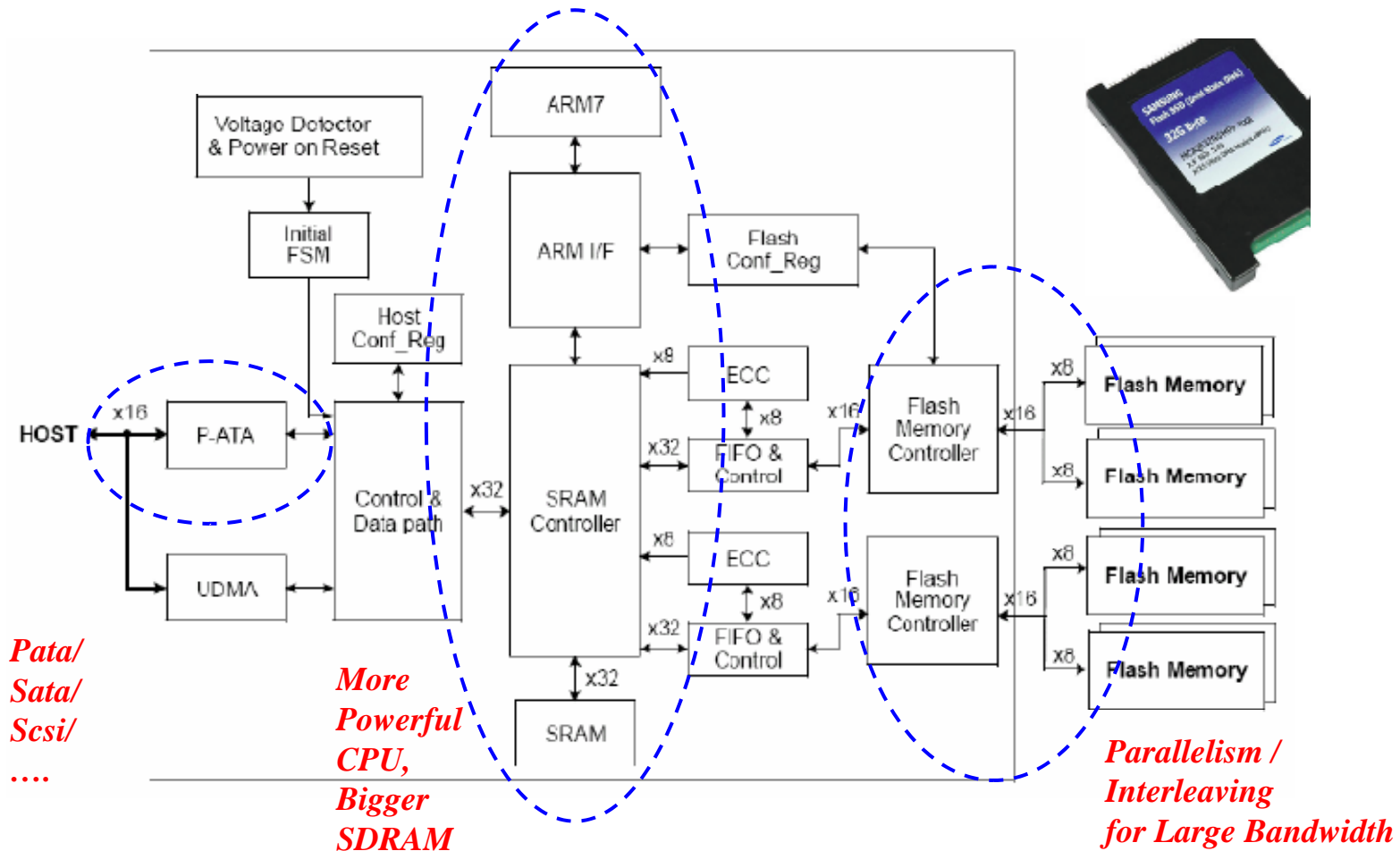


Flash Translation Layer (FTL)

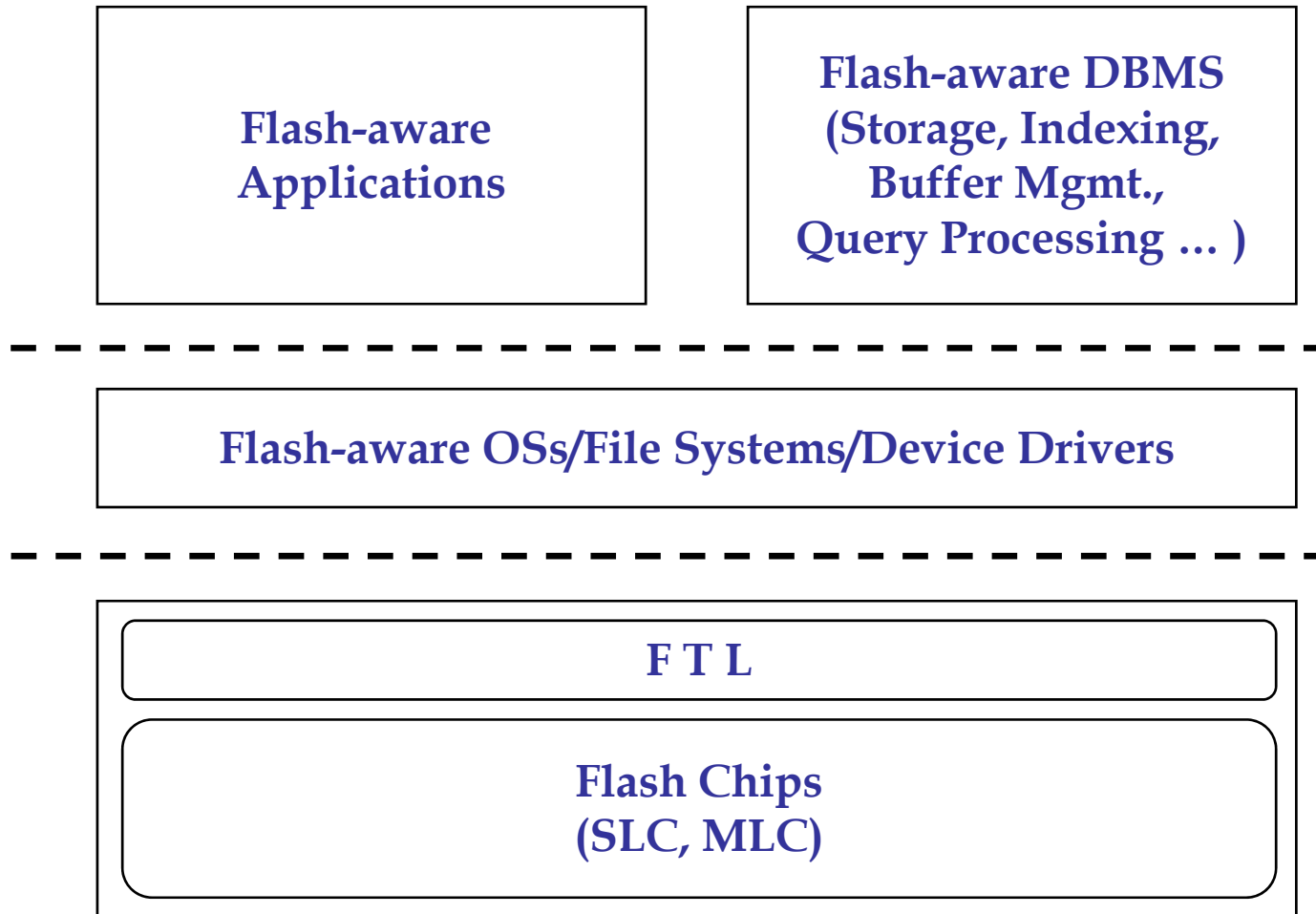
- A software layer that makes flash memory appear to the system like a disk drive
 - **Address mapping**: logical sector <-> physical sector
 - Garbage collection
 - Power-off recovery
 - Wear leveling and bad block management
 -

Flash SSD Block Diagram

- e.g. Samsung SSD – somewhat old model



Opportunities in Flash-based DBMS Research



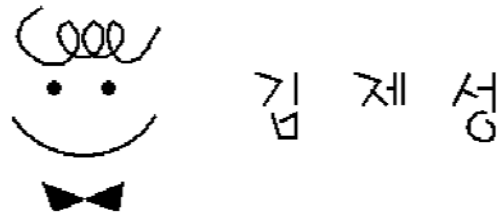
FAST:
**A Log Buffer based Flash Translation Layer
using Fully Associative Sector Translation**

Sang-Won Lee, et al.

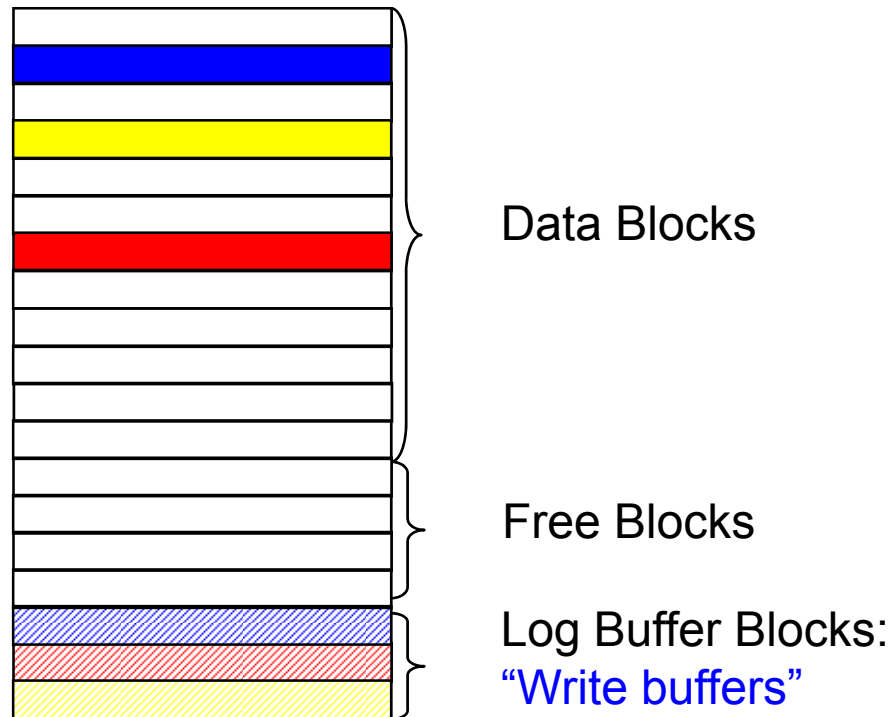
ACM Transactions on Embedded Computing Systems,

July 2007

Seoul National University FTL: Hybrid Mapping



Seoul National University FTL: Hybrid Mapping

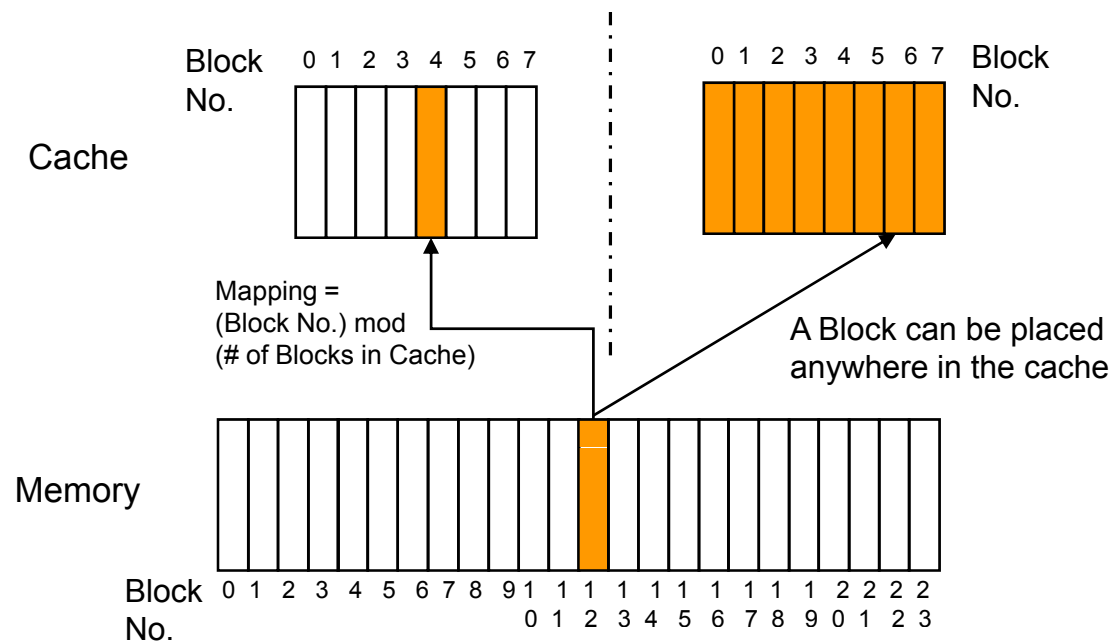


- Every 2K writes can be buffered in **0.2 ms**, not in **20.6 ms**
- When a log block is full, it should be merged
 - "Merge" operation

- based on "Locality" and "Log Structured File System" idea

Associativity in CPU Cache

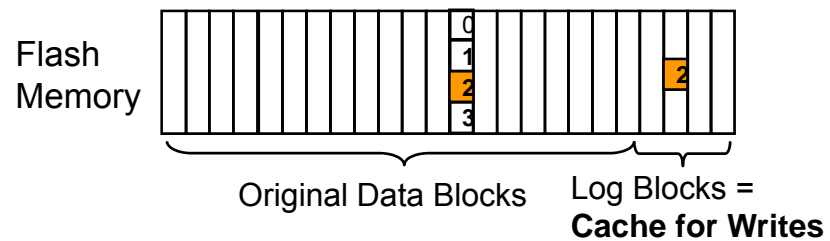
Directed mapping vs. Fully associative mapping



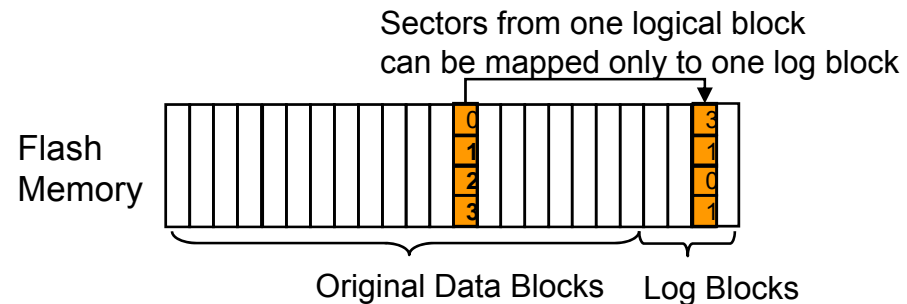
(a) Associativity between memory and CPU cache

SNU FTL = BAST

- **BAST**: Block Associative Sector Translation

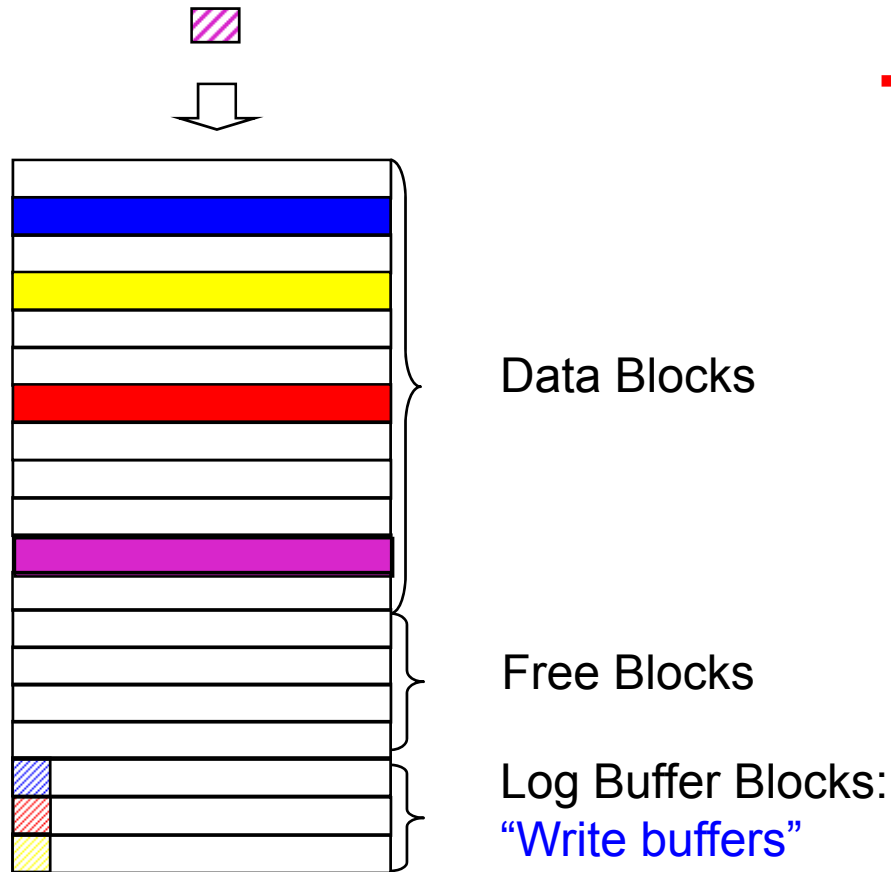


(b) Log blocks: **a cache for write operations**



(c) Block associativity in log block scheme

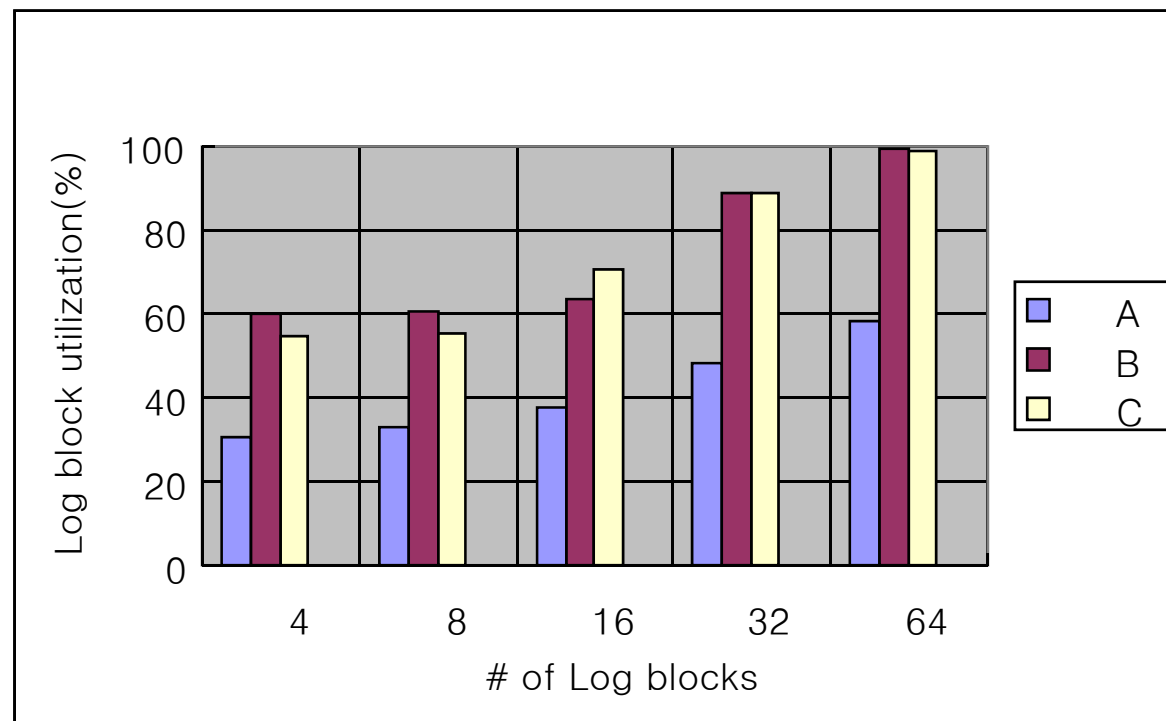
Disadvantages of BAST



- **What if** there are many hot blocks?
 1. Choose one **victim** log blocks
 2. Merge the victim block
 3. Assign a new log block for the new hot block

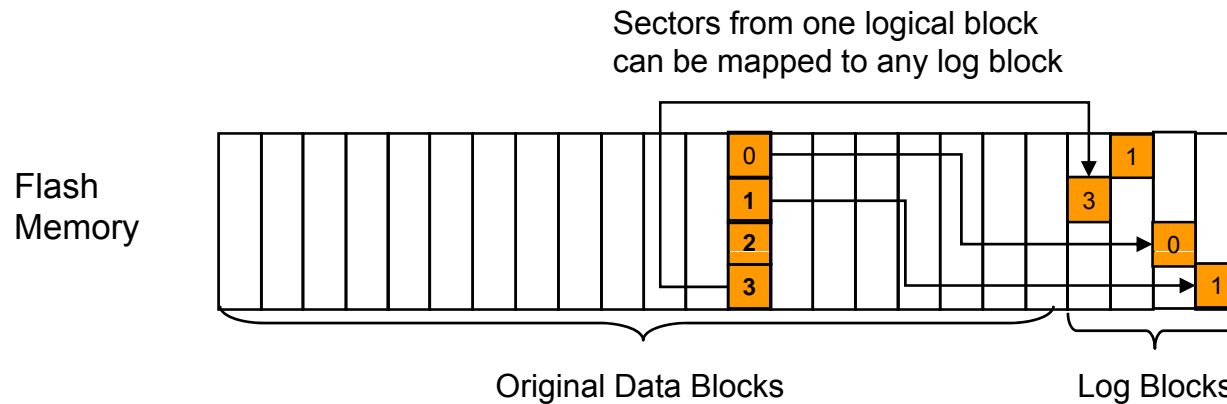
Disadvantages of BAST(2)

- Low space utilization of victim blocks
 - Log block thrashing



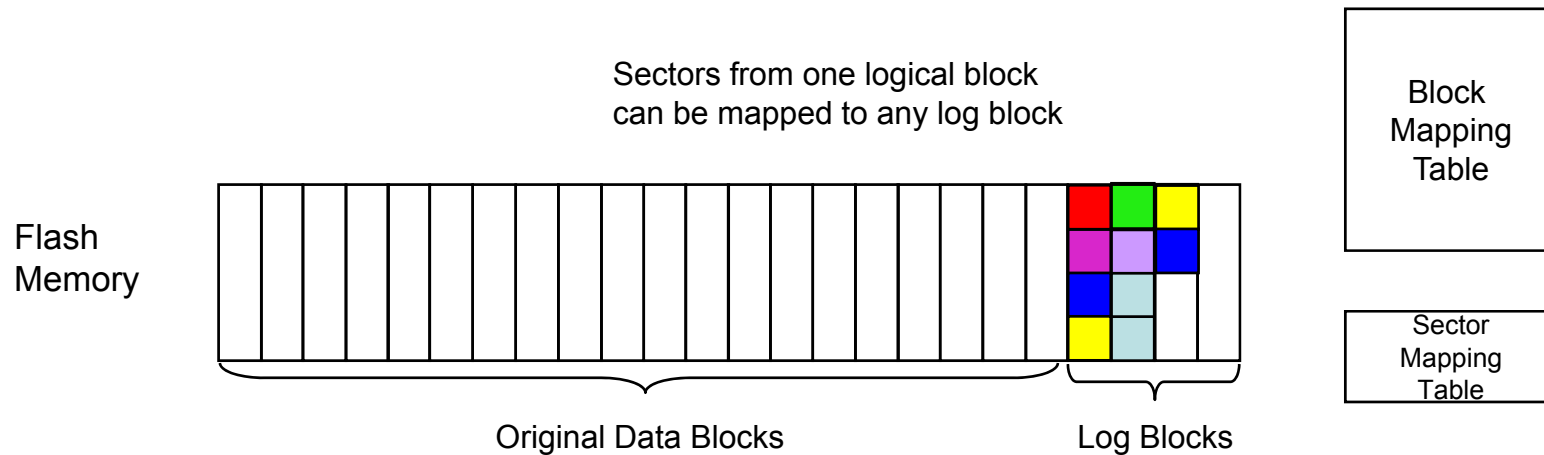
FAST

- What if we take “Fully Associative Sector Translation” in log buffer blocks?



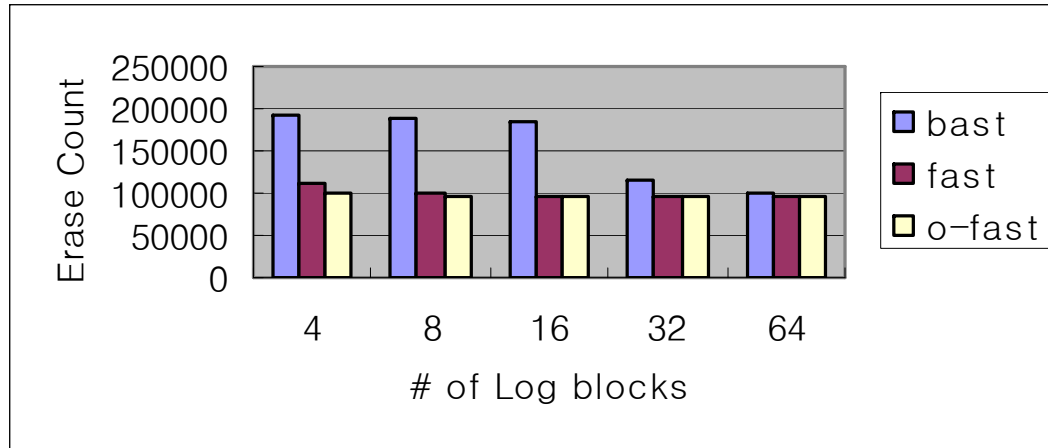
(d) Fully associativity b/w
logical sectors and log blocks

Sector Writes and Reads in FAST

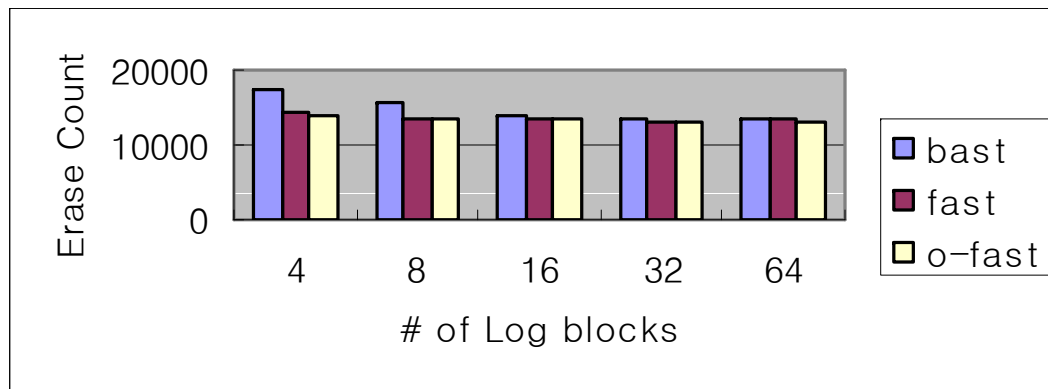


- Sector (over-)write
 - Append the write sector in log blocks
 - If the log blocks are full,
 1. Merge a log block in FIFO manner
 2. Get a free block and make it a new log block
- Sector read
 - Search the sector mapping table for the most recent sector
 - If not found, visit the original data blocks via block mapping table

Performance Evaluation

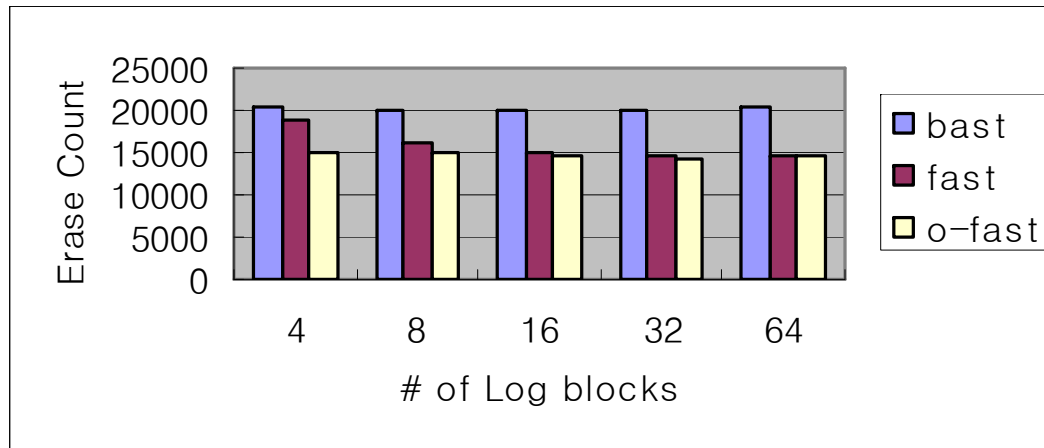


Digital
Camera

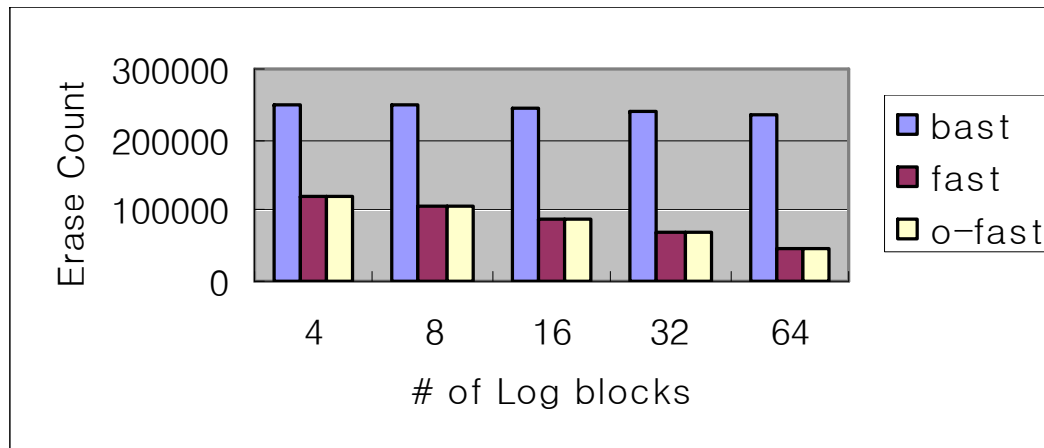


LINUX

Performance Evaluation(2)



Symbian



RANDOM

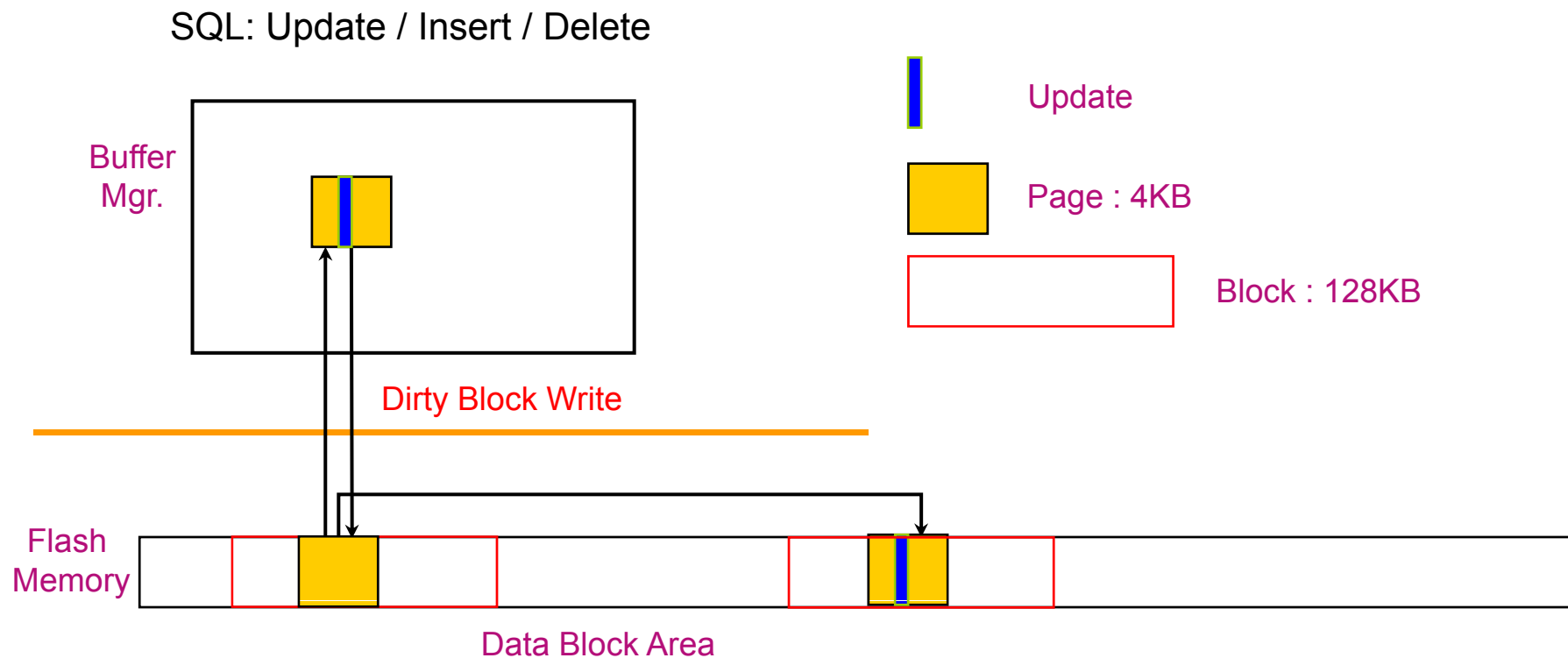
Design of Flash-Based DBMS: An In-Page Logging Approach

SIGMOD 2007

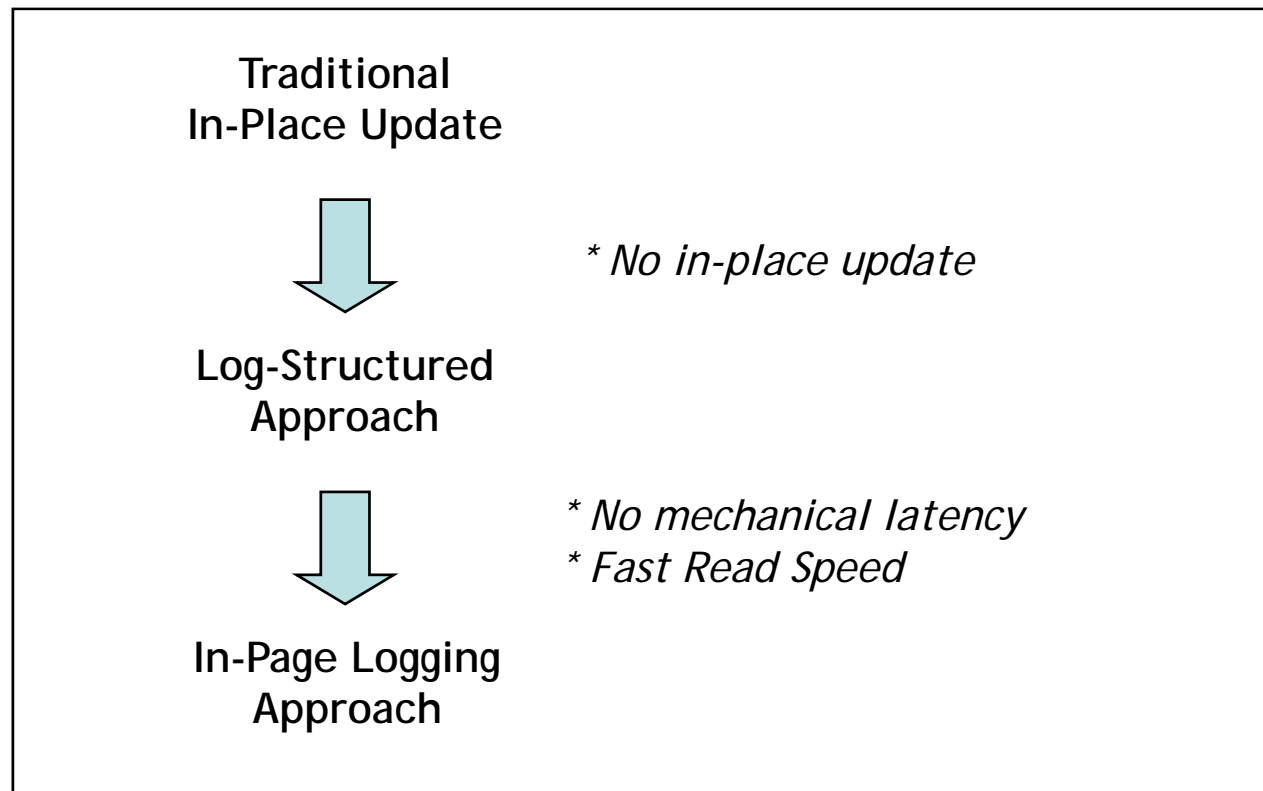
Sang-Won Lee,
Bongki Moon (Univ. of Arizona)

Traditional DBMS on Flash Memory: Problem

- (FTL) + pure disk-based DBMS – traditional in-place update
 - Because of no update-in-place characteristics, most block write results in a copy-back operation (> 20ms)

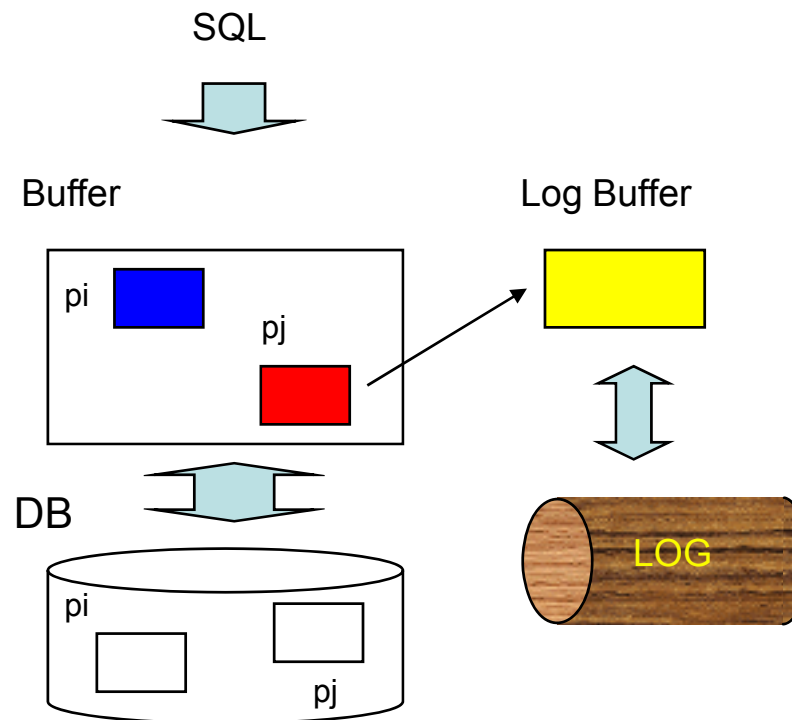


In-page Logging: Basic Ideas



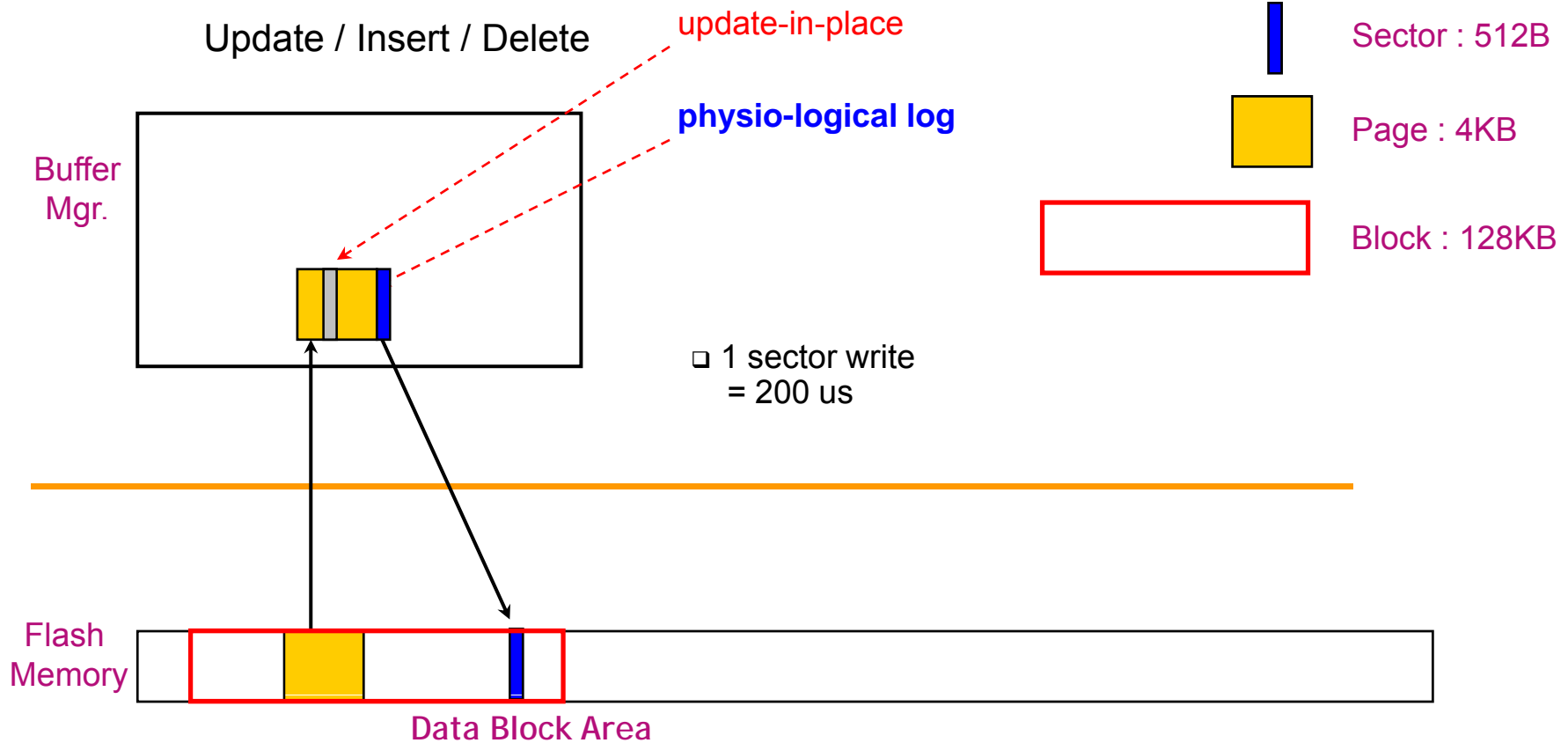
In-page Logging: Basic Ideas(2)

- Database update and log in traditional DBMS architecture



In-page Logging: Basic Ideas(3)

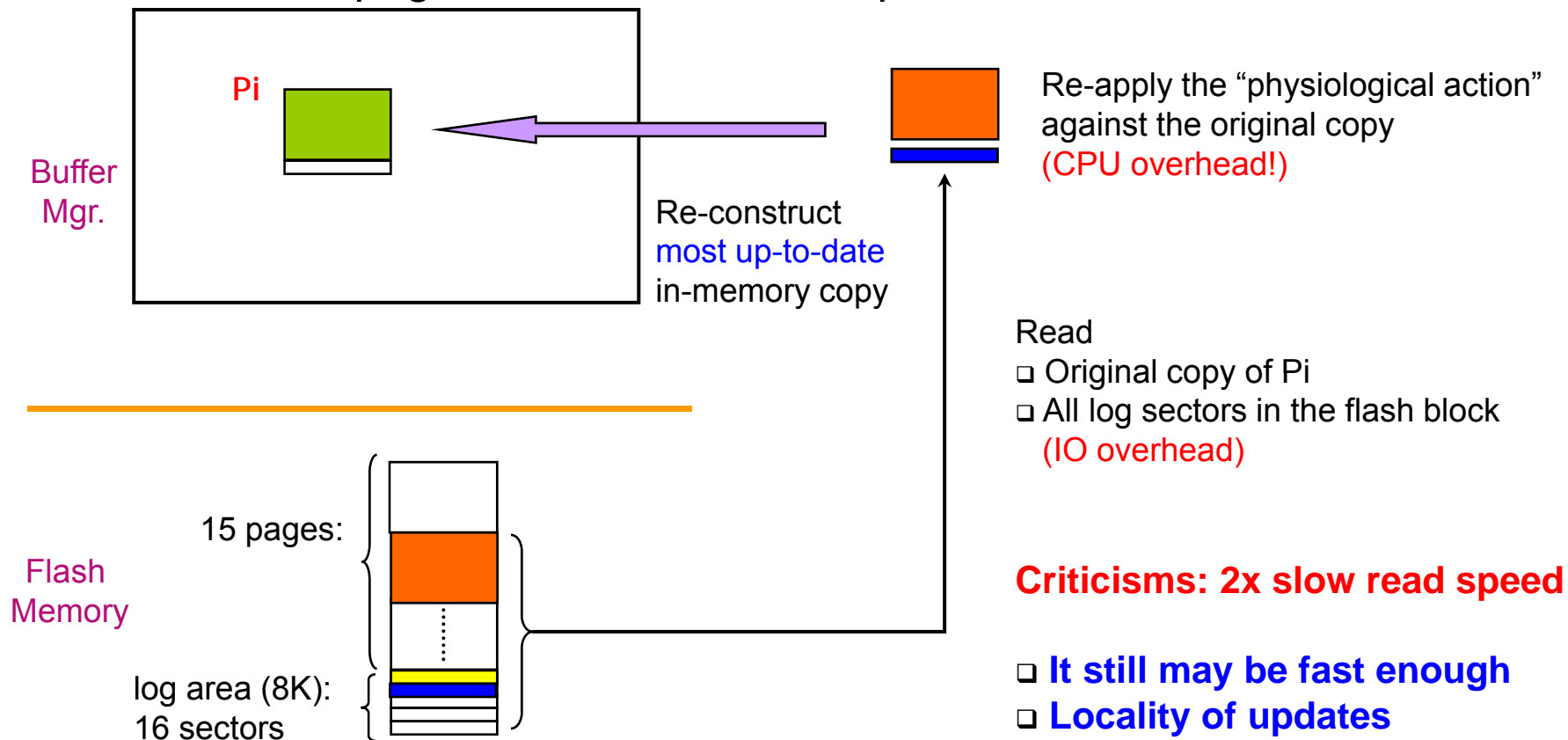
- Write performance: 100x faster in best case



In-page Logging: Basic Ideas(4)

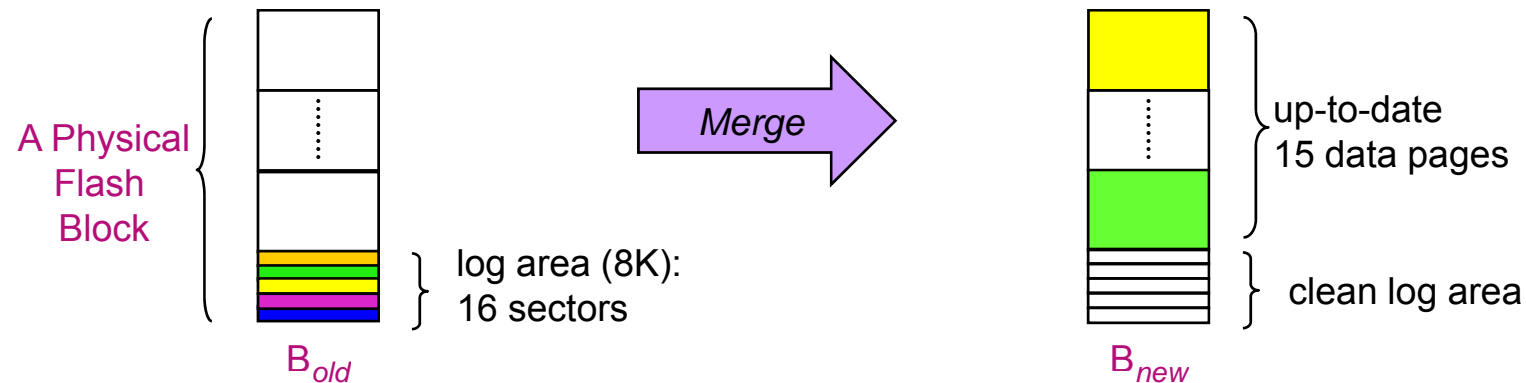
- Read

- When a page P_i is read in after replace-out



In-page Logging: Basic Ideas(5)

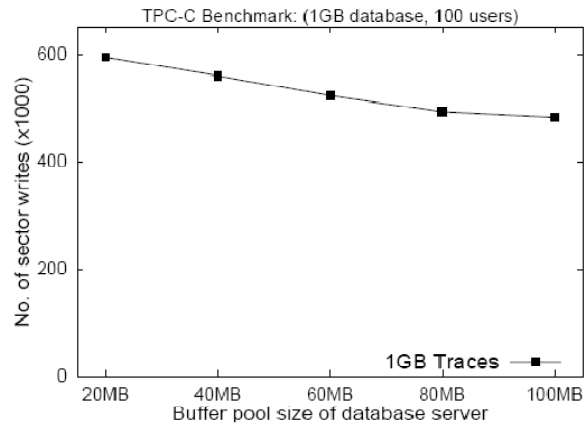
- Merge operation: new **internal** operation in IPL



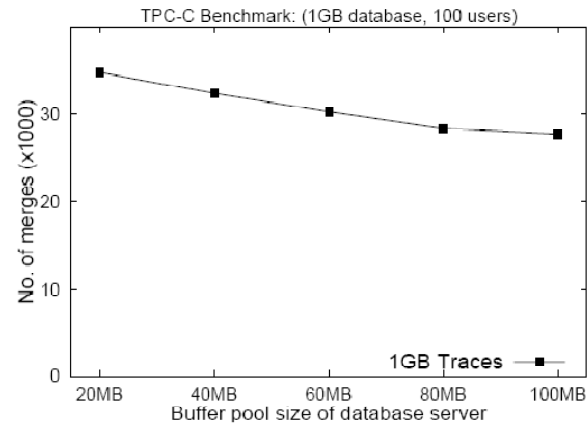
- Merge policies: eager vs. lazy merge
 - Refer to paper

IPL Simulation with TPC-C

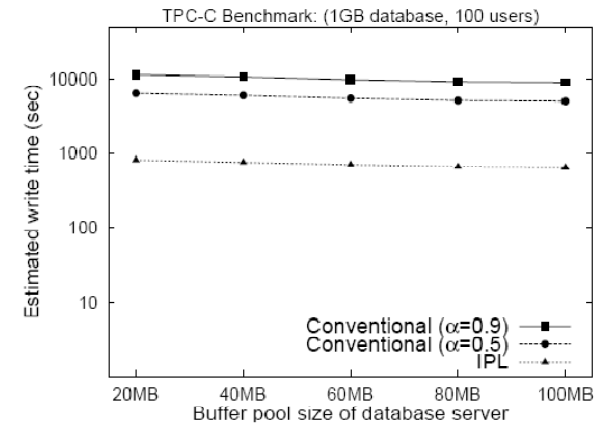
- Performance trend with varying buffer sizes (1G.100u, 8KB log area)



(a) Total number of sectors written

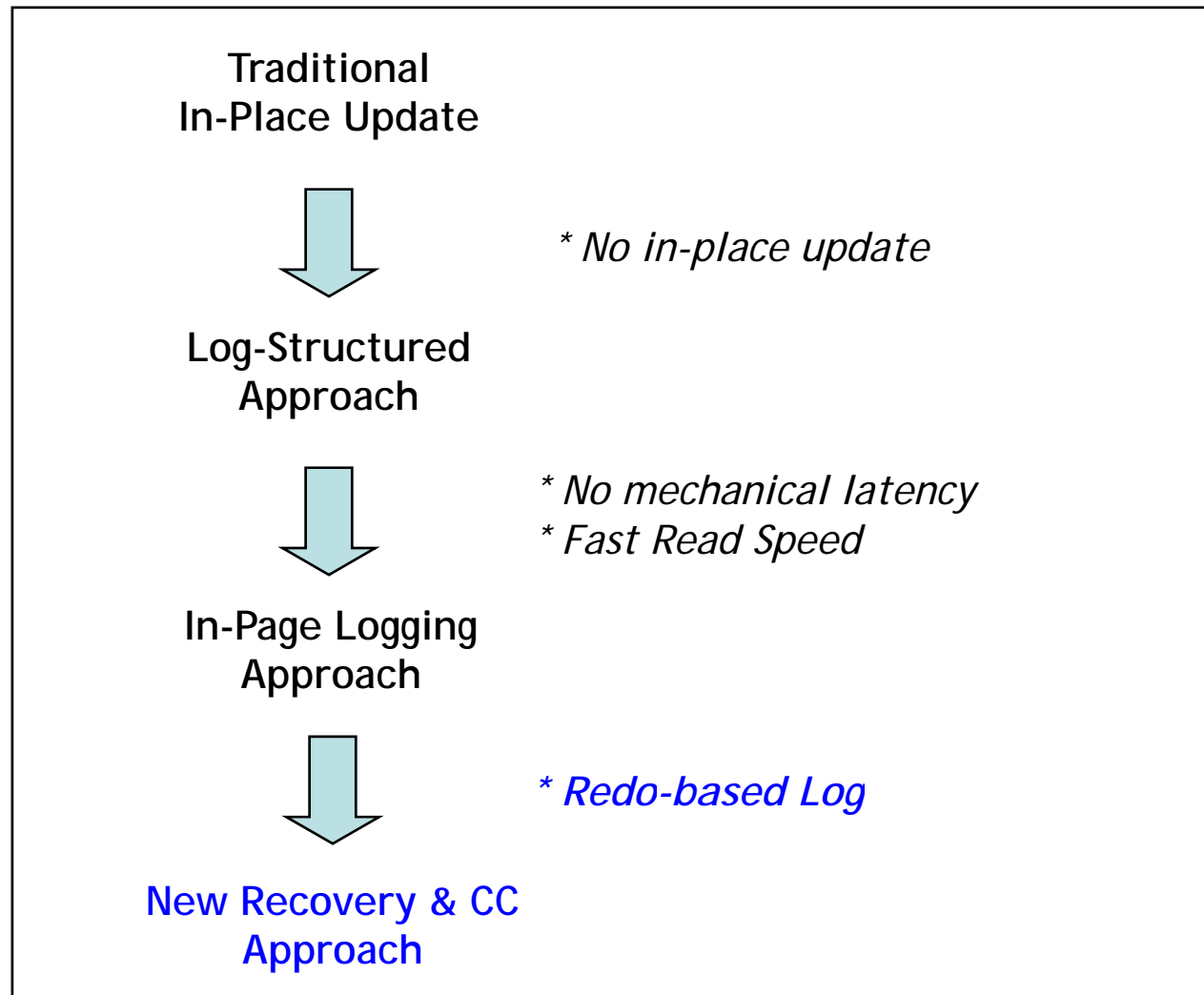


(b) Total number of merges performed



(c) Estimated write time

In-page Logging: The Unexplored Beauty

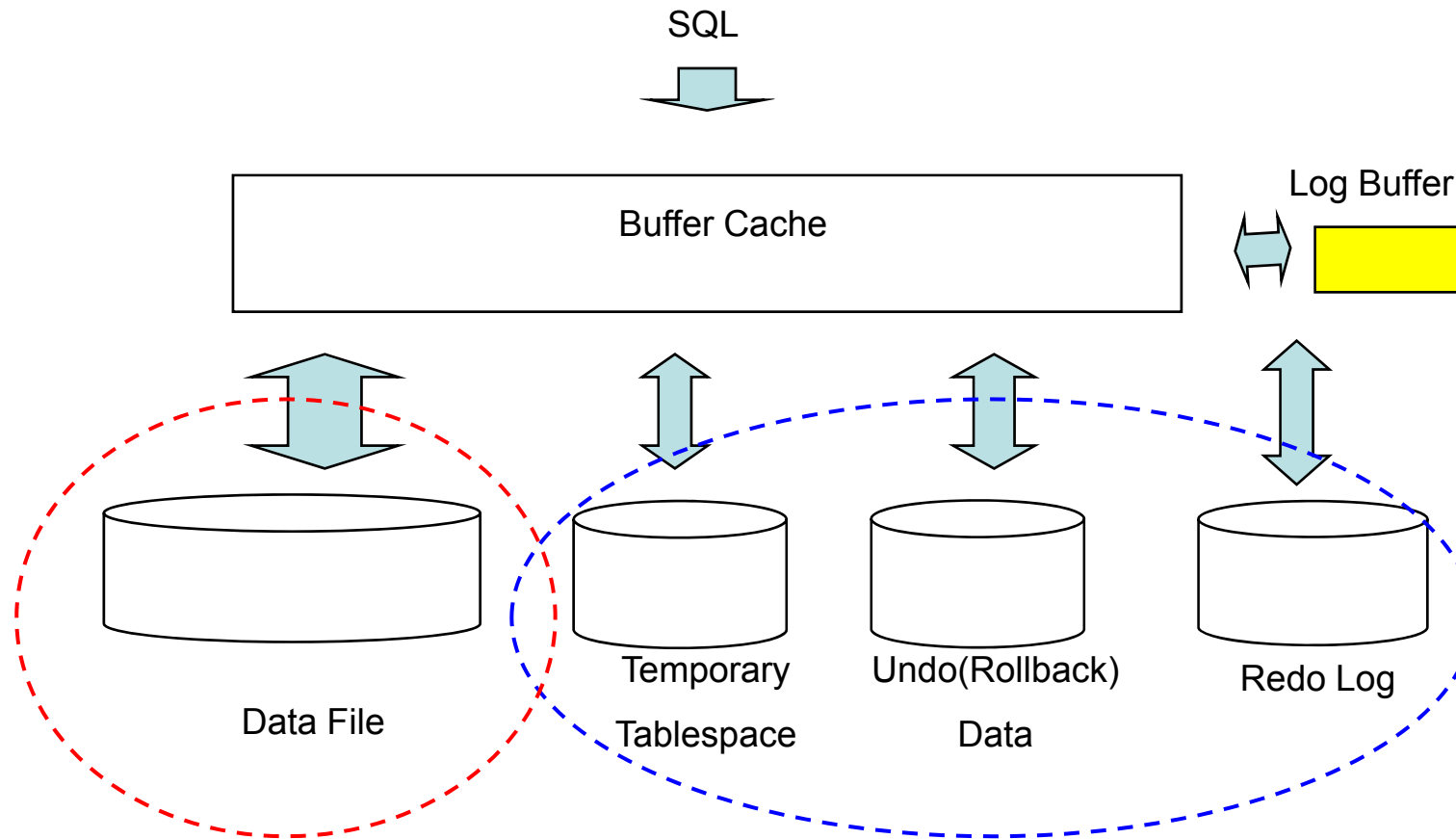


A Case for Flash Memory SSD in Enterprise Database Applications

SIGMOD 2008

**Sang-Won Lee, Bongki Moon, Chanik Park,
Jae-Myung Kim, Sang-Woo Kim**

Major I/Os in Database



- Access pattern: random
- Access pattern: “Append-only sequential writes, then random reads”
- Next challenge

Redo Log

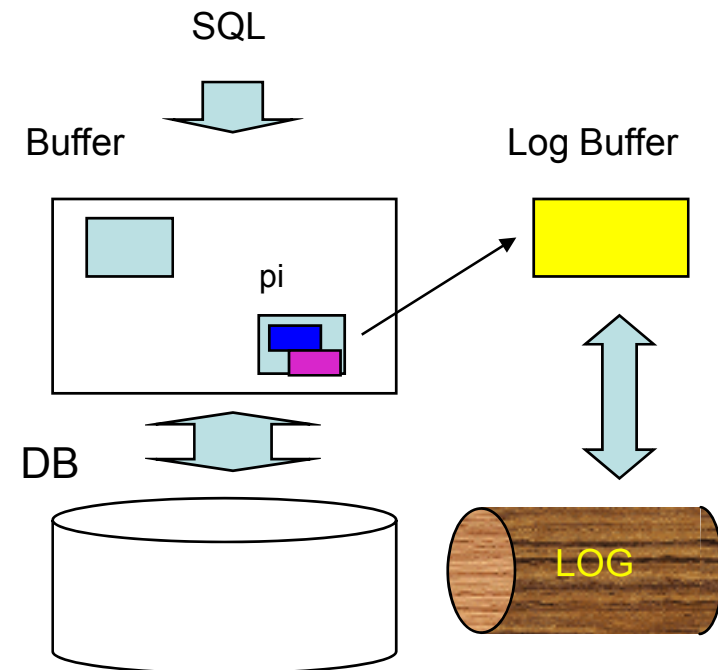
Transaction Concept and Log

`begin_transaction`

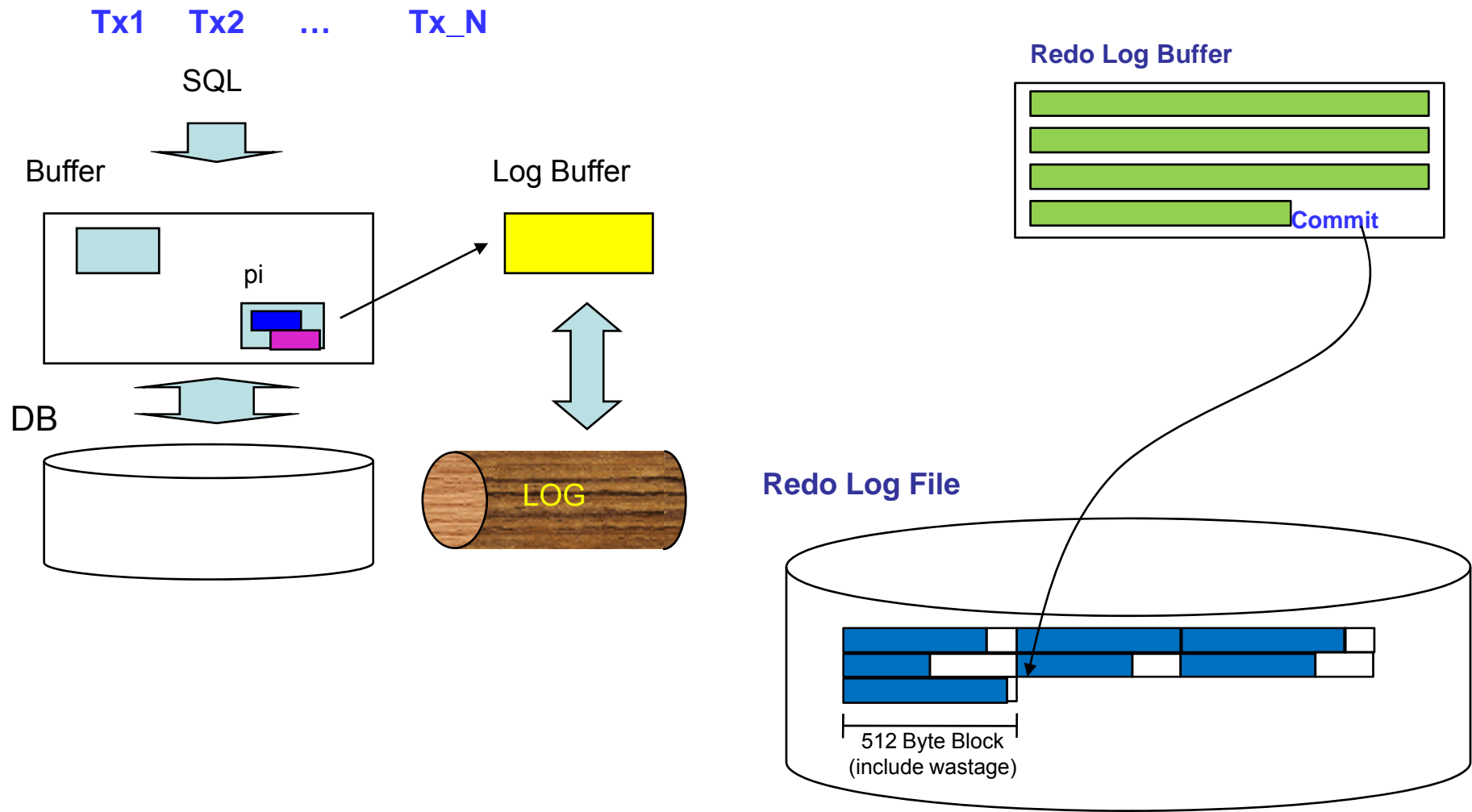
`select statements; // read`

`insert/update/deletes; // update`

`commit` or `rollback;`



Transaction Concept and Log (2)

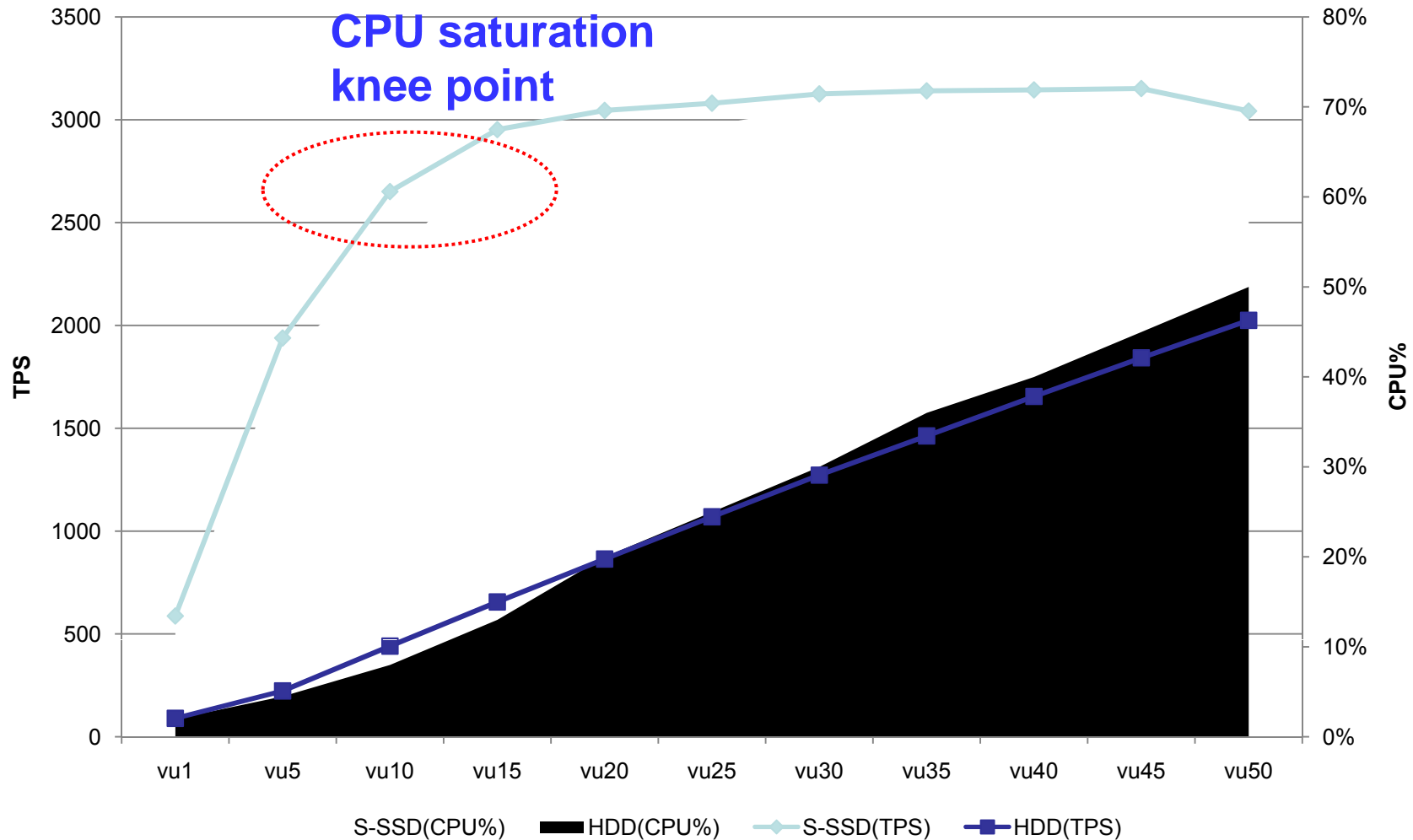


Transaction Commit: Final Performance Bottleneck

- Transaction response time = read / write + CPU + **commit**
 - Negligible read / write time
 - ✓ Large buffer cache or main memory DBMS
 - CPU time (dual core Intel CPU)
 - ✓ TPC-B: 50 us
 - ✓ TPC-C: ~ 200 us
 - Commit time = **8 ~ 10ms**
 - ✓ Final performance bottleneck

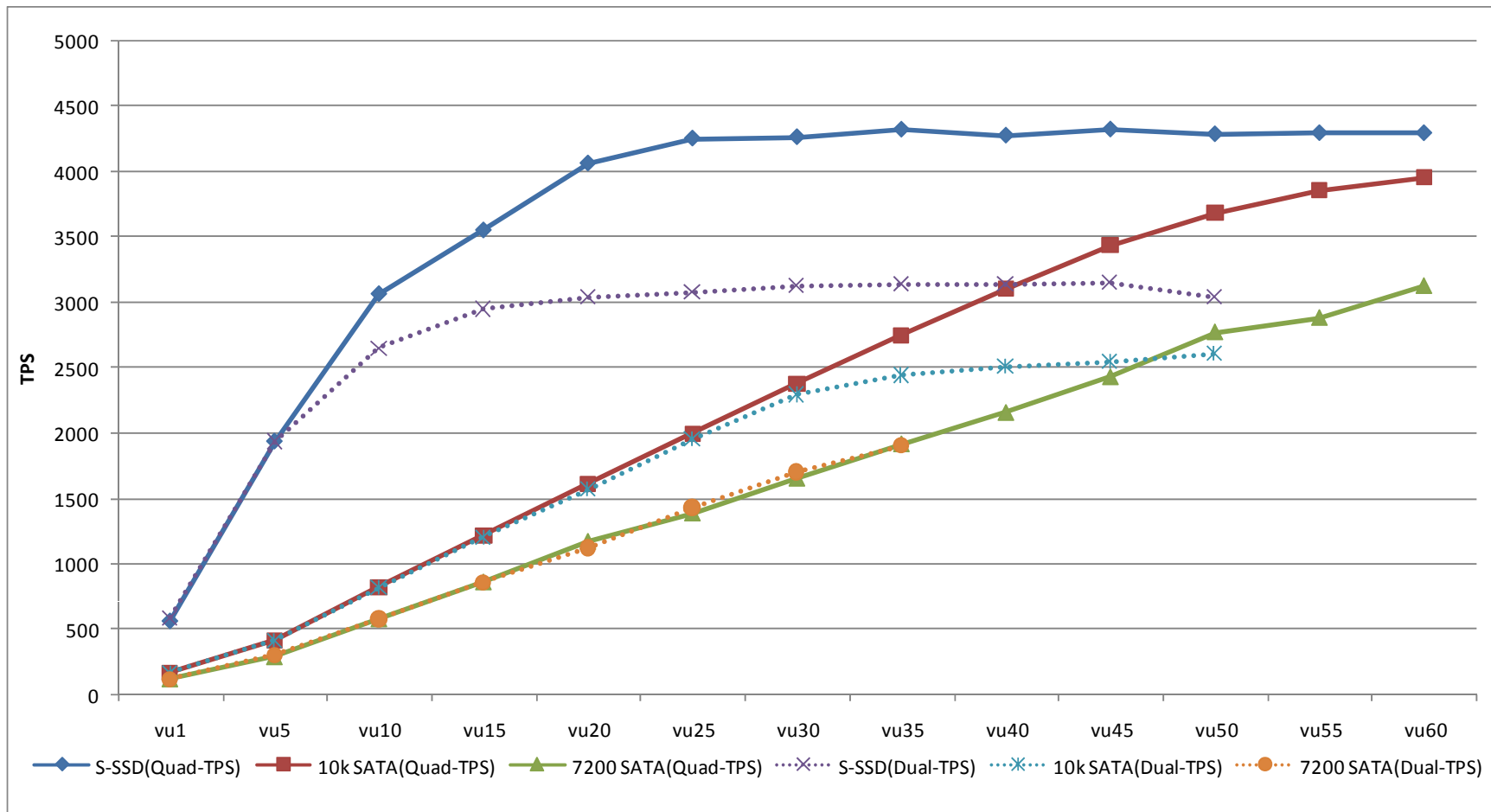
TPS in TPC-B: HDD vs. SSD

- TPS vs. CPU



TPS in TPC-B: HDD vs. SSD

- CPU-bound vs. IO-bound: 7200 vs. 10000 RPM SATA



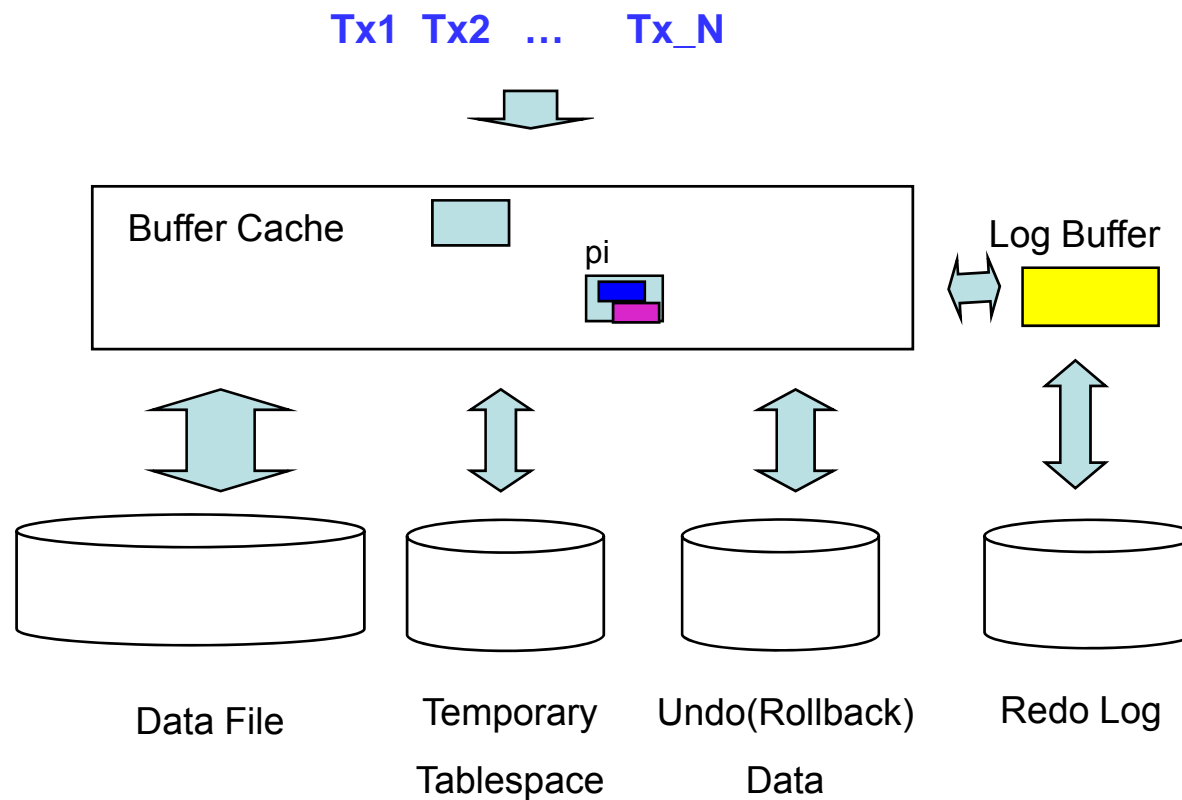
Flash as Log Device

- Communications of the ACM, July 2008
 - “Flash memory storage” Adam Leventhal @ Sun

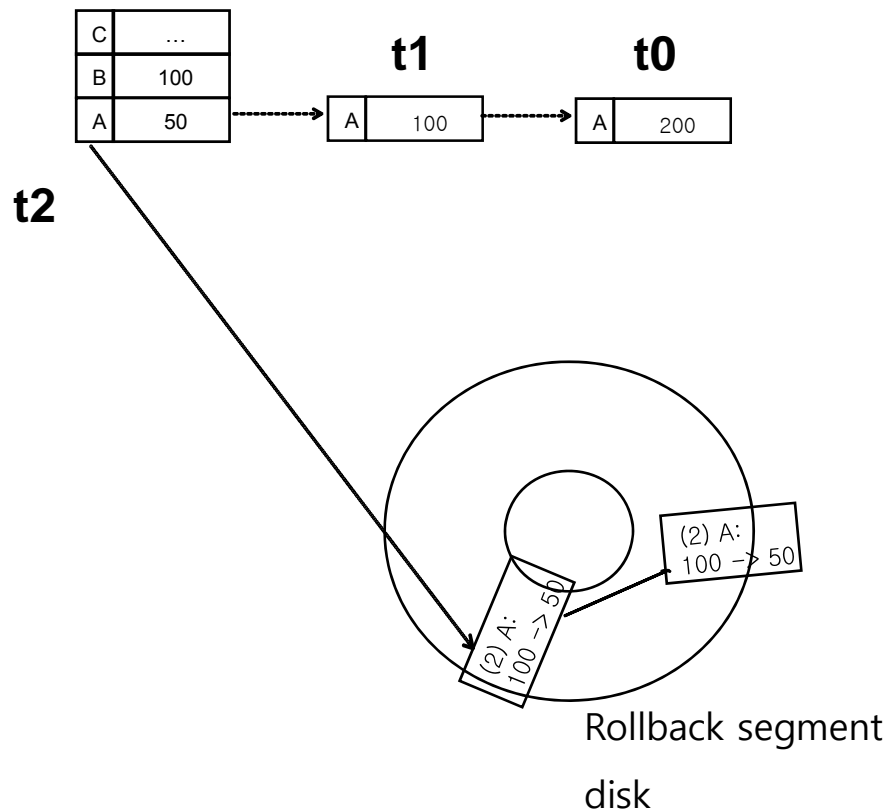
Rollback Segment

Undo Data in Oracle

- Undo tablespace and rollback segments



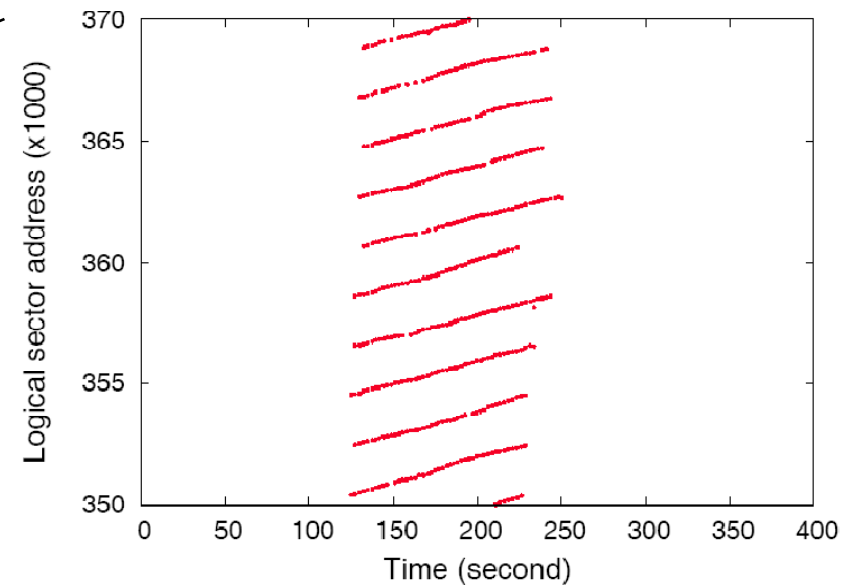
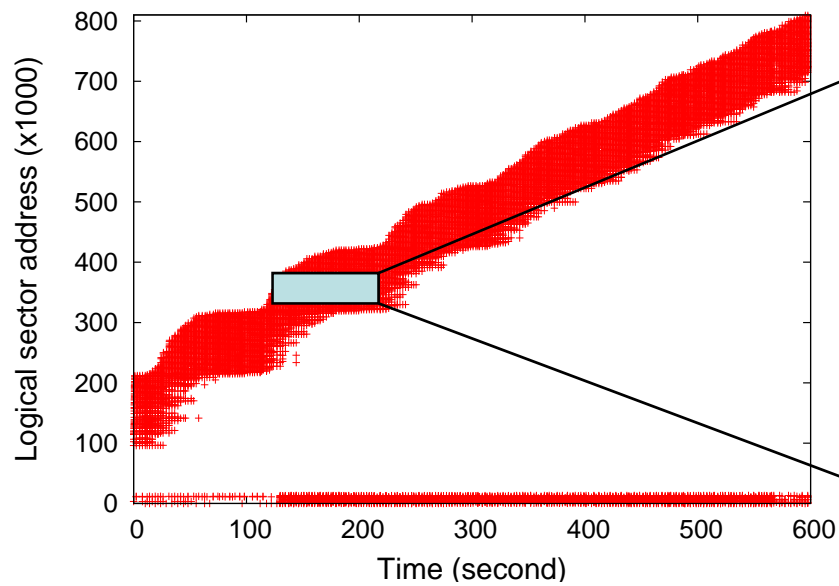
Rollback Segment



- Rollback segment
 - Stores undo(old) information
 - Transaction rollback; MVCC
- Each transaction is assigned to a rollback segment in round-robin
- Each rollback segment is written in (almost) **append-only** mode
 - Asynch mode by DBWR
- For read consistency, the rollback segments is randomly accessed

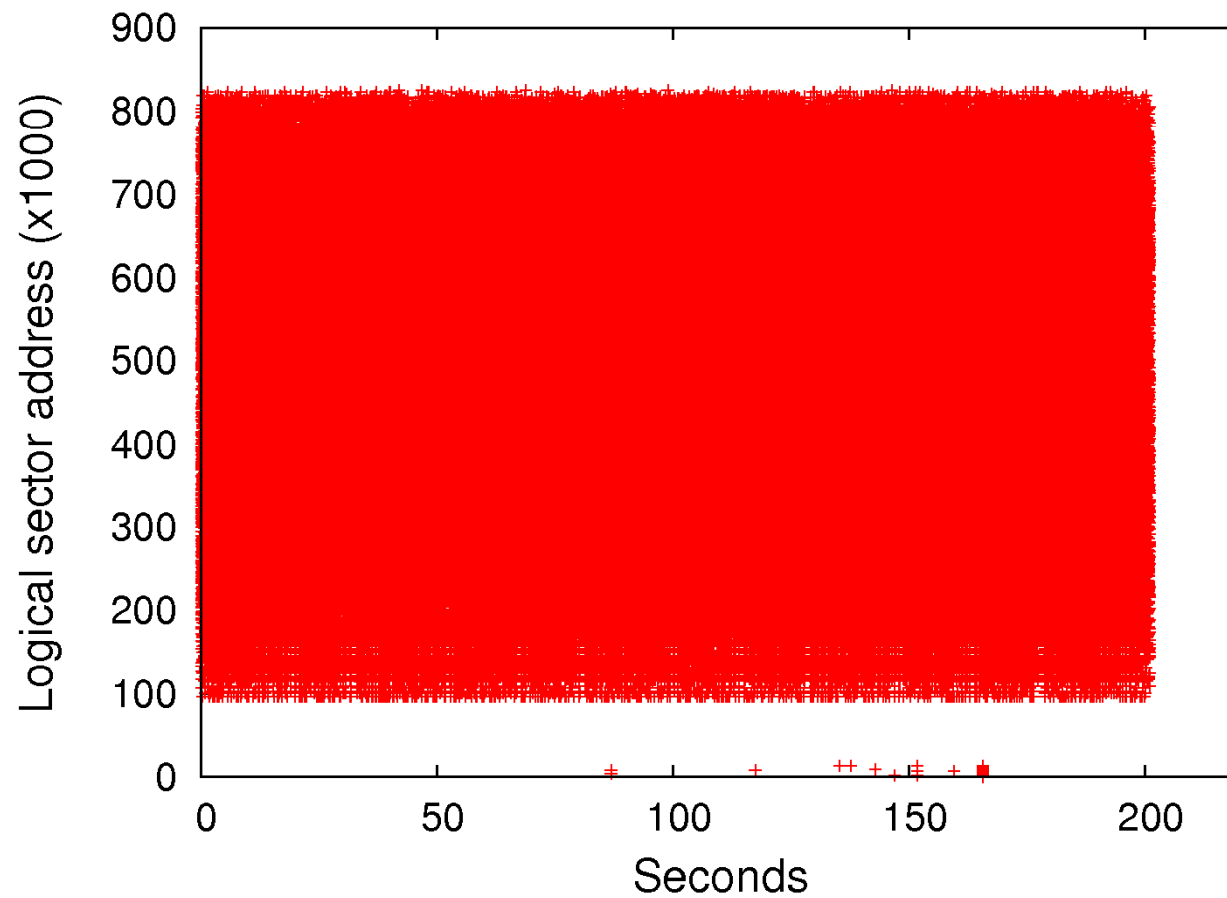
Undo Area I/O Pattern (Write)

- During TPC-C run
 - (Semi-)Sequential write, then random read



Undo Area I/O Pattern (Read)

- Full scan of a table while running TPC-C



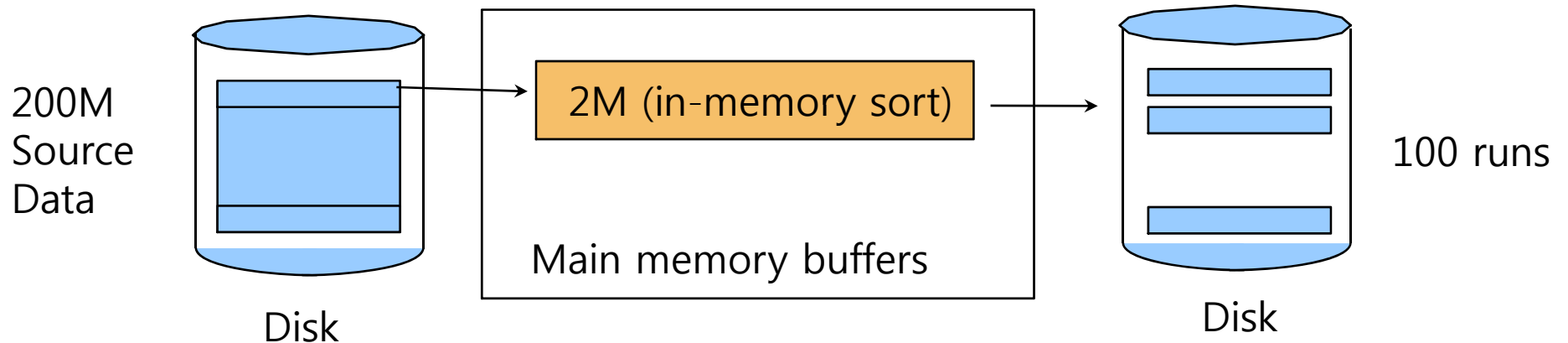
Temporary Tablespace

Temporary Tablespace

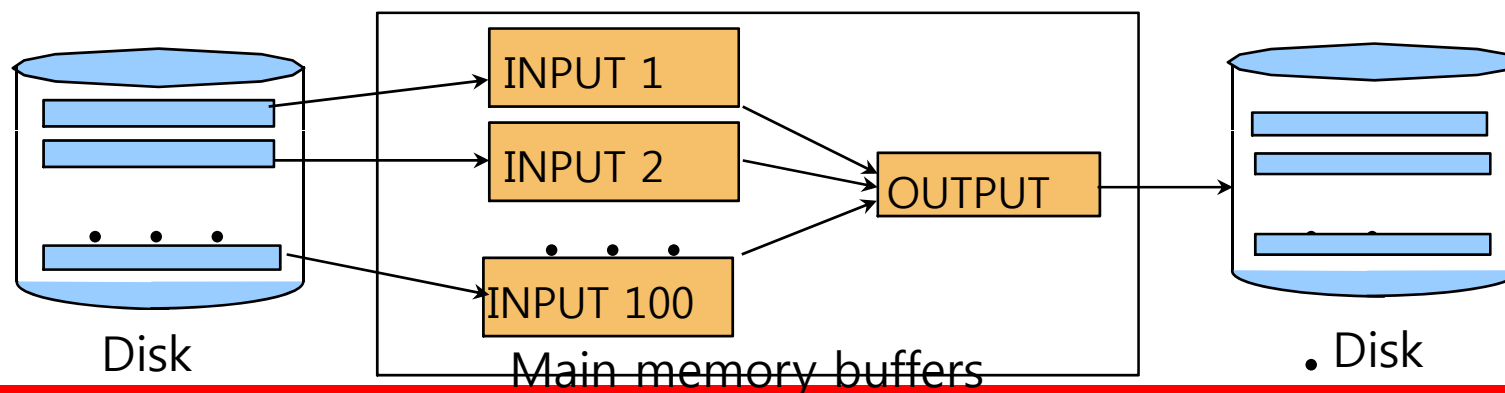
- Index creation
- Temporary table
- Sort / Sort merge join
- Hash join

External Merge Sort

- 1st phase: run generation

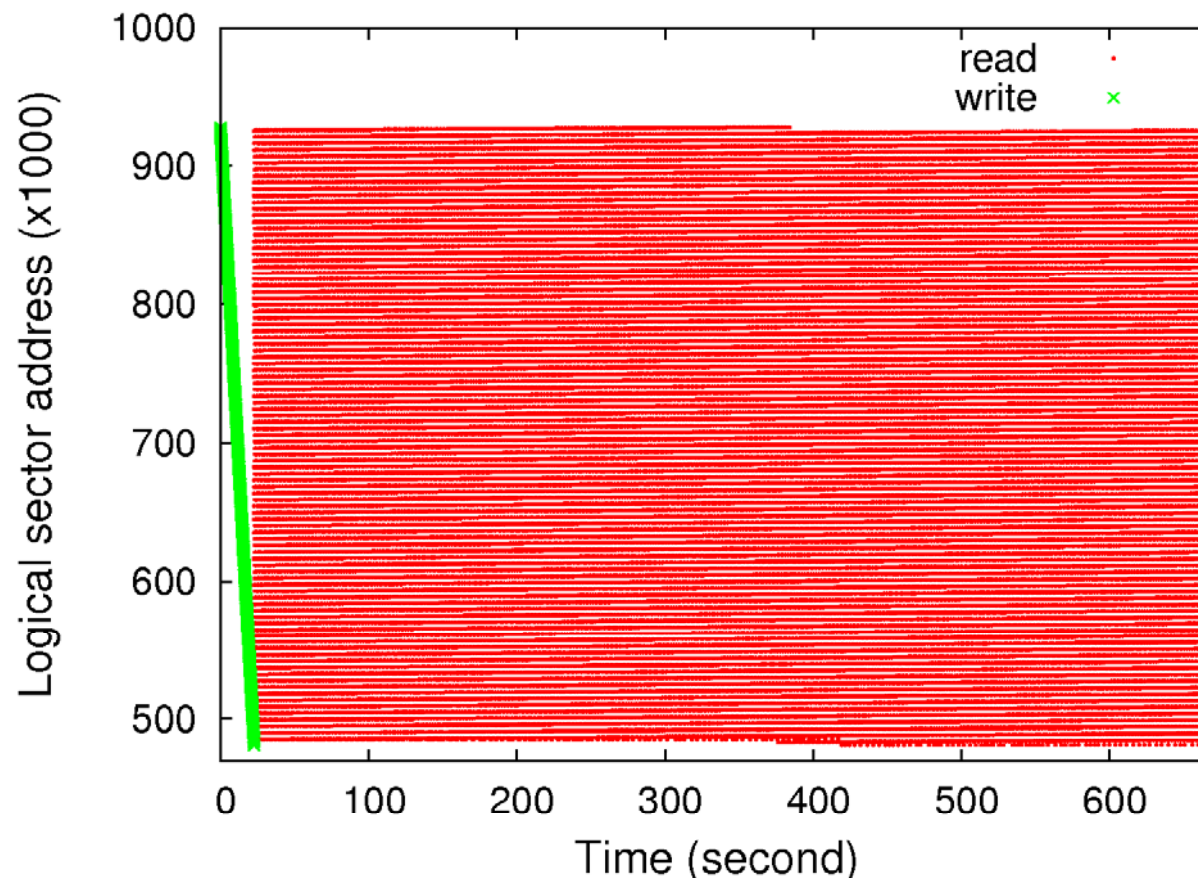


- 2nd phase: merge



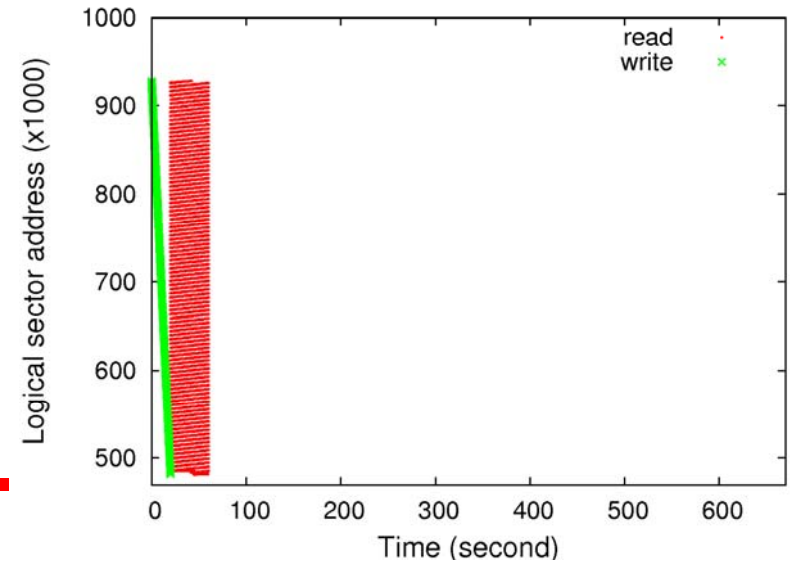
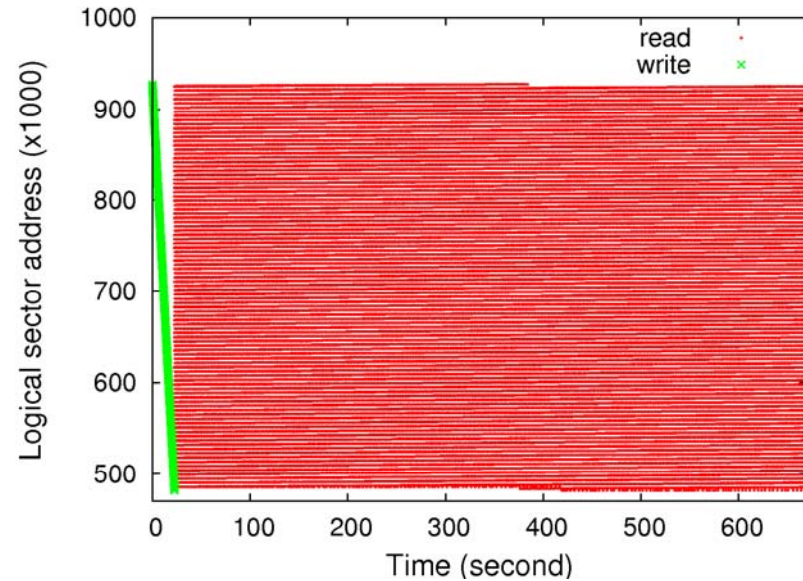
IO Patterns in External Merge Sort

- Sequential write, then random read



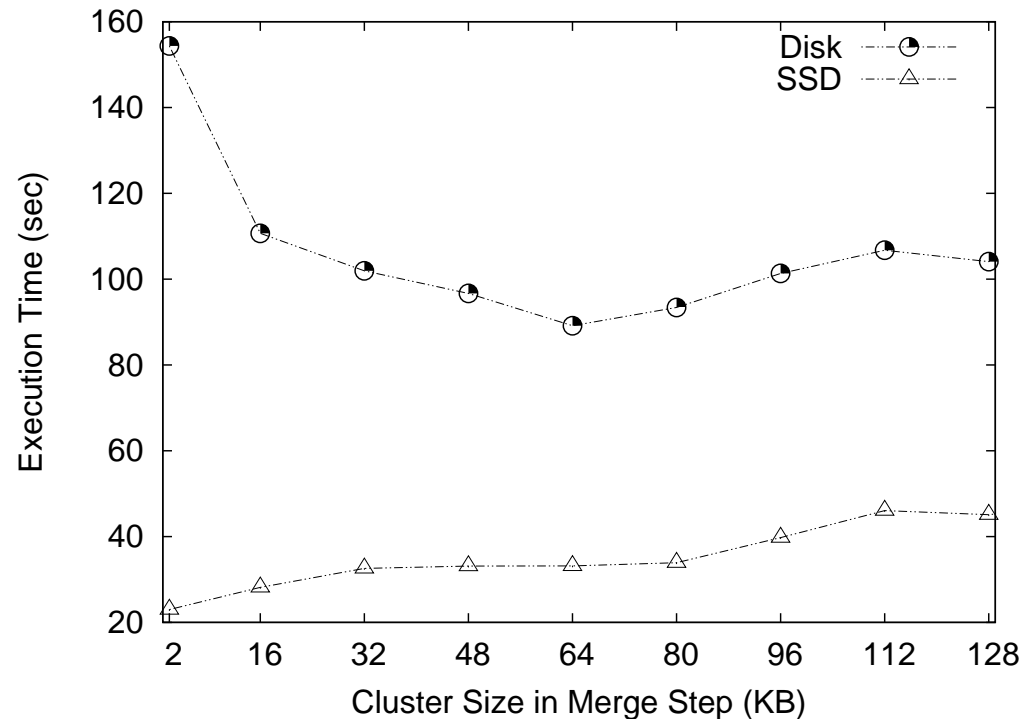
External Merge Sort

- Experiment set up
 - 200M data
 - ✓ 100B record
 - ✓ 10B sort key
 - In-memory sort area: 2M

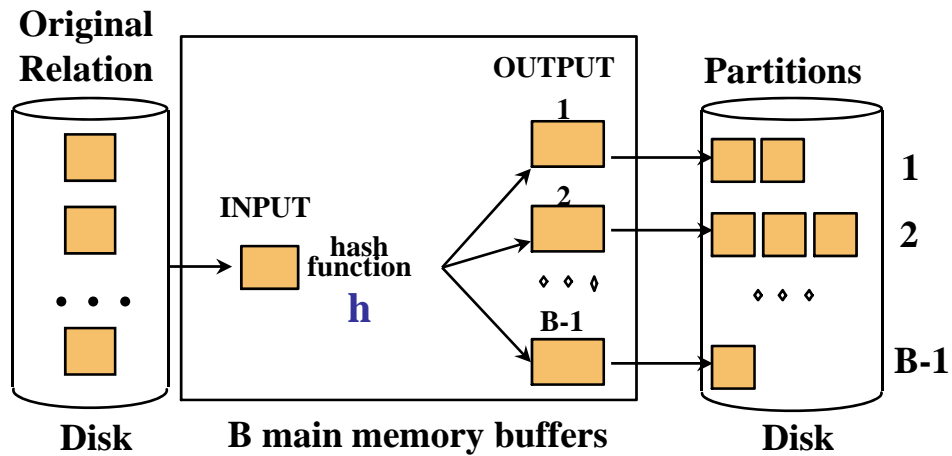


External Merge Sort

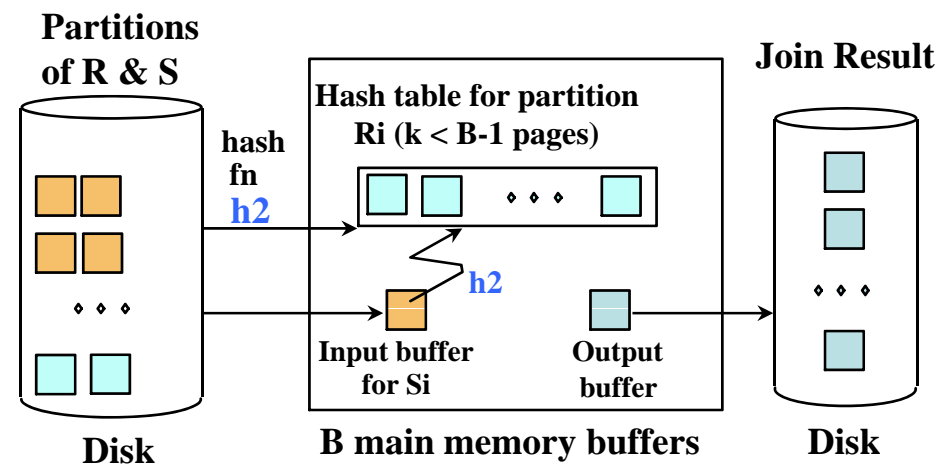
- Cluster
 - Unit of IO in merge step
- Change the cluster size
 - 2K ~ 128K
- Optimal cluster size
 - 64K (hdd) vs. 2K (sdd)
- Implications



Hash Join

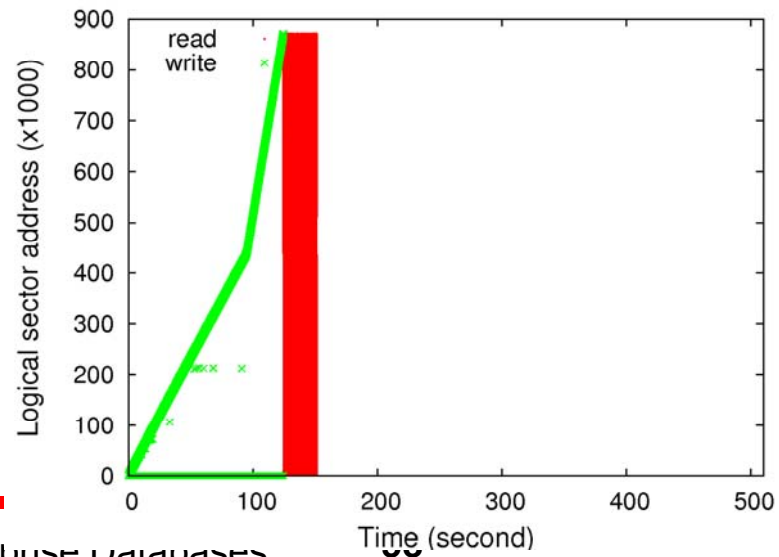
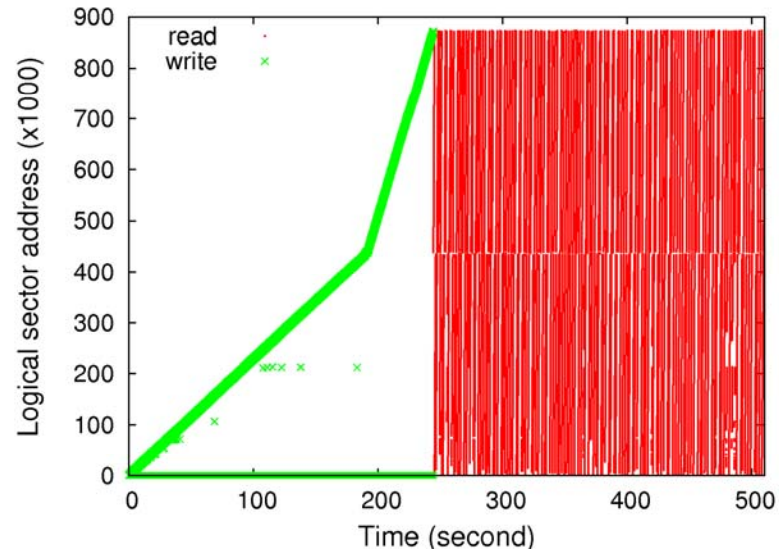


1. Partitioning (or building) phase



2. Probing (or matching) phase

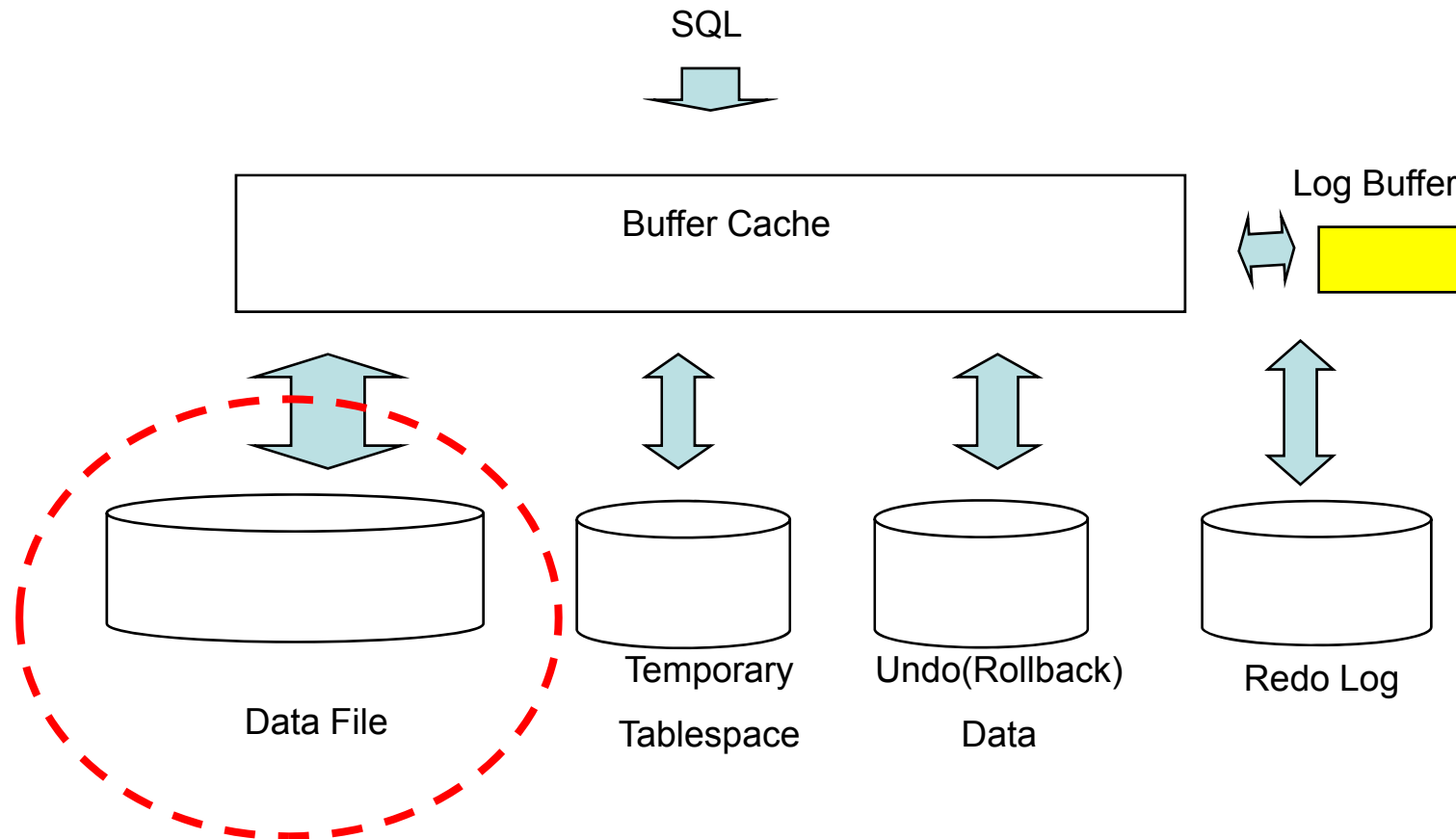
IO Patterns in Hash Join



FlashSSDs in Enterprise Databases



Next Challenges in Database



- Access pattern: **random writes**

Flash Perspectives from DB Community

- “There has been discussion of sea-changes in DBMS design arising from new storage technologies replacing disk. Flash memory appears to be both technologically viable and economically supported by a broad market. **Flash is the first new persistent storage medium to succeed in this regard in more than three decades.**” [Hellerstein, Stonebraker and Hamilton (2007)].



- Flash is disk, disk is tape, and tape is dead.



Some DB Community Papers

- DAMON 2008: from CPU to Storage in research
 - [Modeling the Performance of Algorithms on Flash Memory Devices](#) Ross, *Kenneth*
 - [Fast Scans and Joins using Flash Drives](#) Shah, *Mehul A.*; Stavros Harizopoulos; Janet L. Wiener; Goetz Graefe
- VLDB 2008
 - Flashing Up The Storage Layer, Ioannis Koltsidas (University of Edinburgh), Stratis Viglas (University of Edinburgh)
 - Online Maintenance of Very Large Random Samples on Flash Storage Suman Nath (Microsoft Research), Phillip Gibbons (Intel Research, Pittsburgh, USA)

Flash-based DBMS: Research Groups

- Ken Ross @ Columbia
- Sam Madden @ MIT
- Graefe group @ HP
- HKUST
- INRIA: PicoDBMS team
 - *PBFilter: Indexing Flash-Resident Data through Partitioned Summaries*
- Minesota
- Microsoft / Edinburg university (VLDB 2008)
- ...

*“If you want truly to understand something,
try to change it”*

Kurt Lewin