Recent Issues in Flash-based DBMSs

Apr. 20, 2010

Sang-Won Lee

http://icc.skku.ac.kr/~swlee

Table of Contents

One FlashSSD beats Ten 15K rpm HDDs

But…

Flash Database Architectures

Page-Differential Logging

- "Page-Differential Logging: An Efficient and DBMS-Independent Approach for Storing Data into Flash Memory", SIGMOD 2010
  - The difference b/w old and new version of a page is very small
  - Sandforce-like approach ?
  - Assume page-mapping FTL ?
  - Differential = *<physical page ID, creation time stamp, [offset, length,changed data]+>.*
  - "At-most one differential" per page
- Physical changes vs. logical changes

IPL Basics, Beauty and Limitations

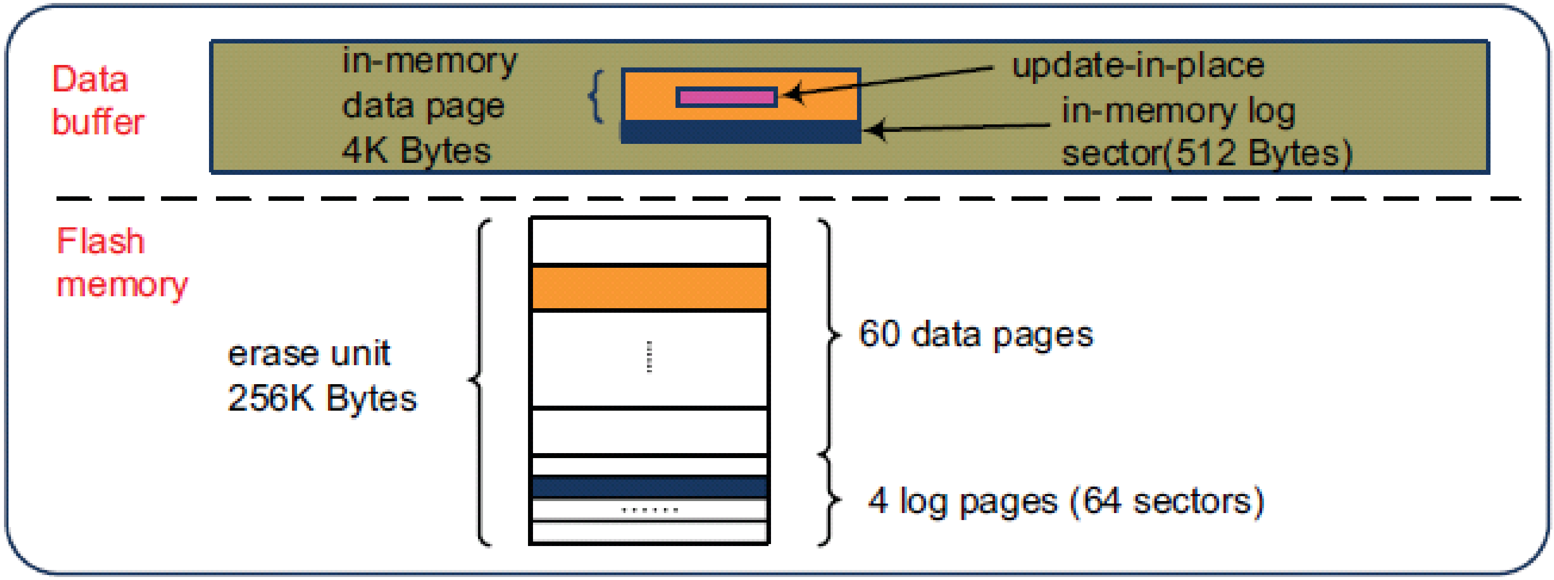
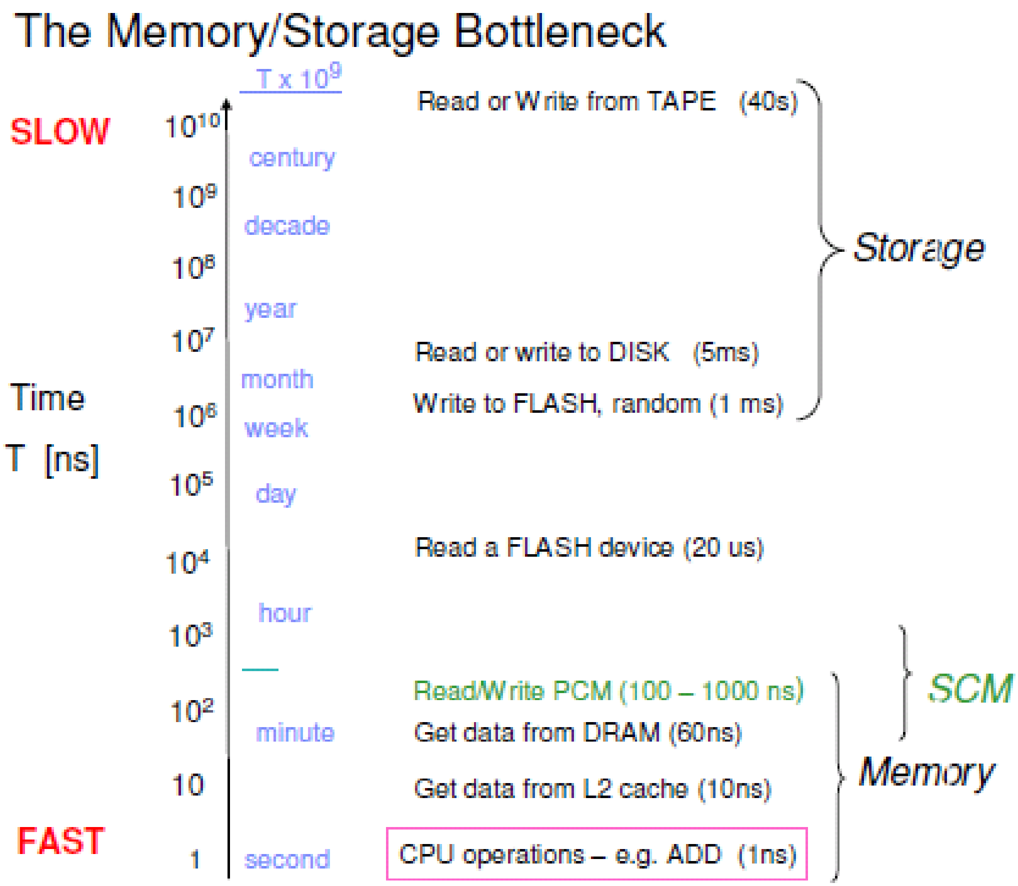


Figure 1. An illustration of the IPL method.

- Transactional extensions: submitted for publication
  - Multi-version concurrency control (SI) and recovery
- IPL: larger flash page, less efficient

SCM

- Source: FAST 2009 tutorial by Dr. Winfried W. Wilcke



IPL + SCM: Opportunities

- Source: “A Hybrid Solid-State Storage Architecture for the Performance, Energy Consumption, and Lifetime Improvement”, HPCA 2010

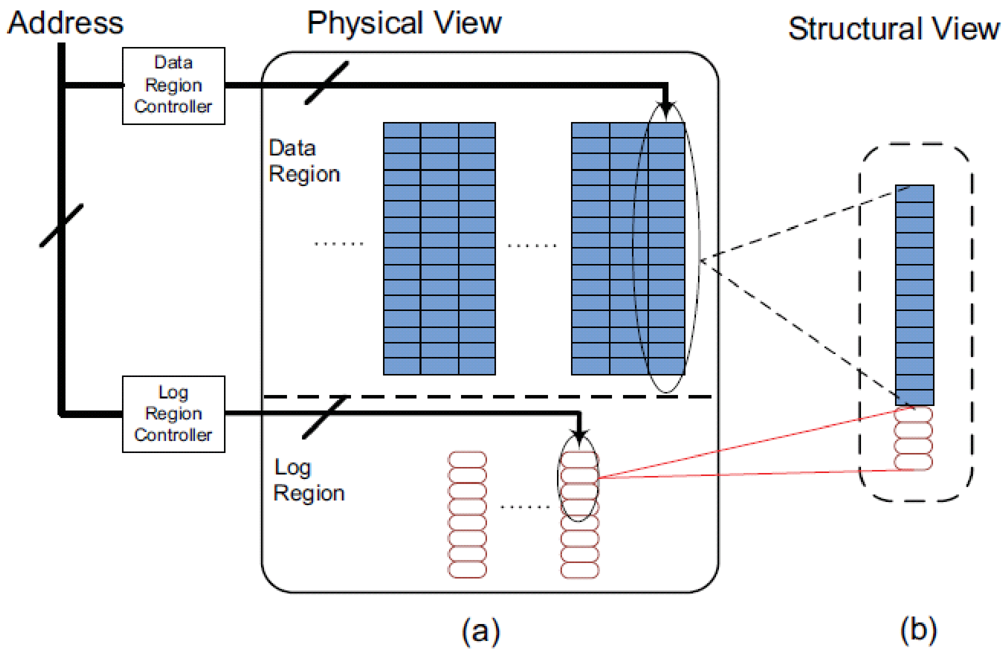
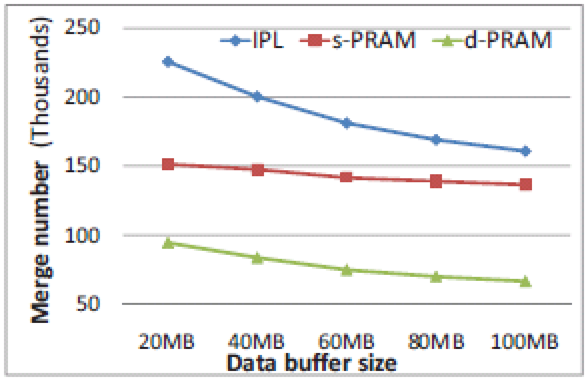
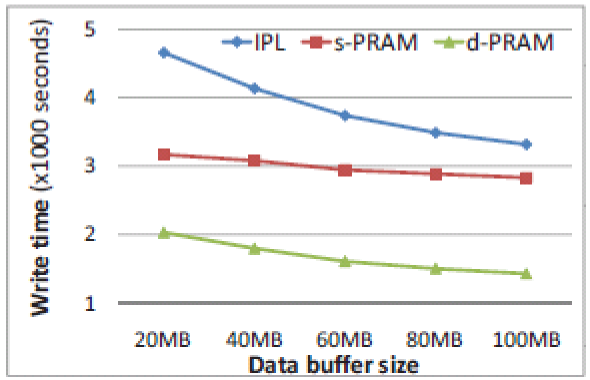


Figure 2. Physical and structural views of the hybrid architecture.

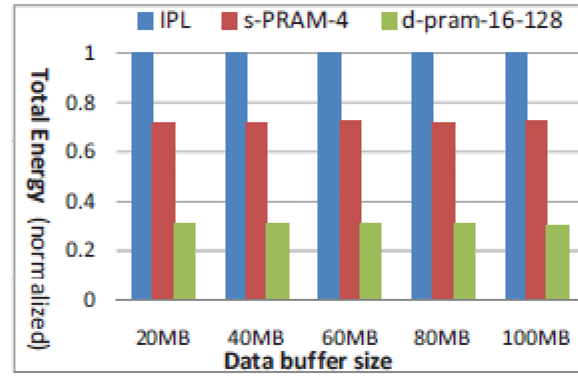
Better Performance, Energy, Lifetime



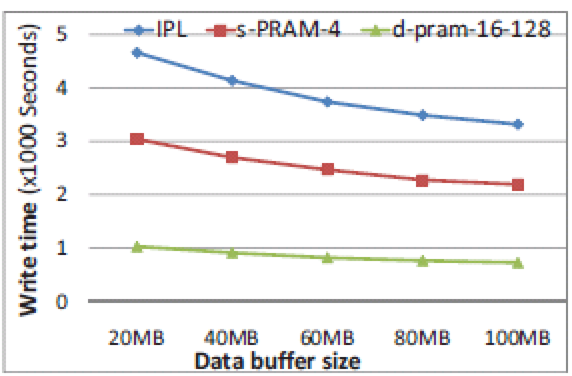
(b)



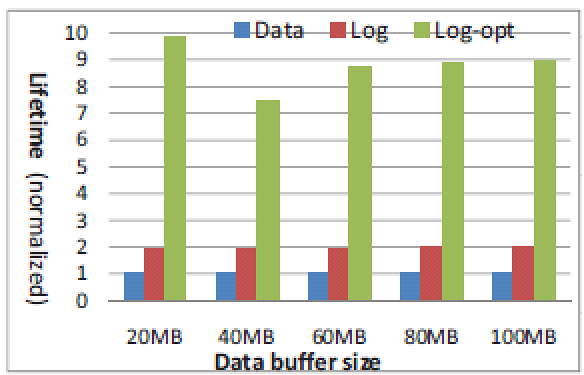
(c)



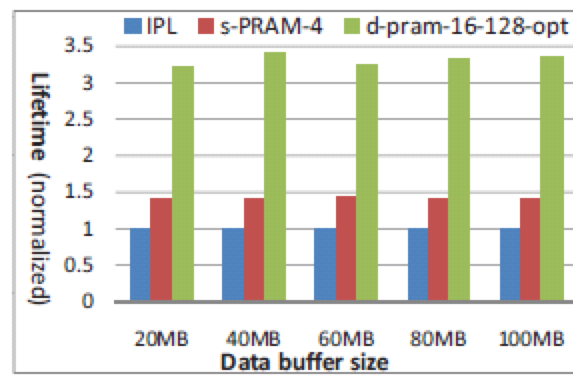
(c)



(a)



(b)



(c)

Why SandForce, IPL, PDL works?

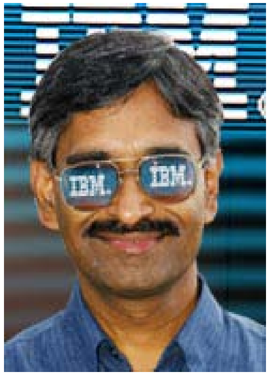
- In TPC-C, the average size of differentials is around 200B.

- 200B/4K = 5%

- Write amplication, performance, wearleveling…

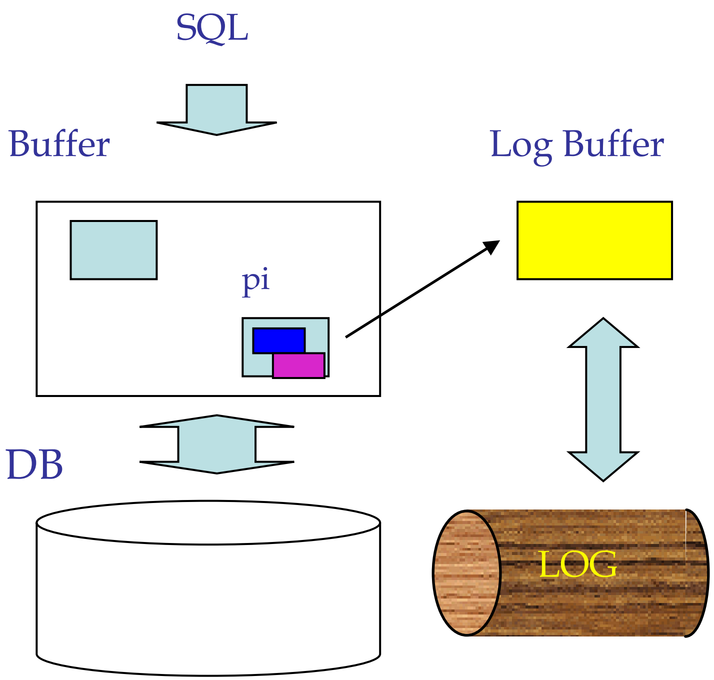
SCM Opportunities in DB

- "Implications of Storage Class Memories (SCM) on Software Architectures", HPCA 2010 West Workshop, C. Mohan @ IBM Almaden
  - PCM as disk, paging device, memory, extended memory



- SCM as log device
  - Should log records be written directly to PCM
  - Or, first to DRAM log buffers and then be forced to PCM (rather than disk)
- PCM replaces DRAM? Whole DB fits in PCM? No logging? ..
  - SafeRAM @ VLDB 1988

SCM as Log Device



Future SW Architecture for NVRAM??

- Need to learn from database??

- E.g. applications, file system, or OS should be able to capture the (logical or physical) differentials (or delta) and then write only the differentials, not the new version itself.
  - File as byte-stream vs. record-oriented page layout
  - Can we model the changes in PPT or work or save only the changes ?
    - ✓ What about multi-versioning? rollback?
- It is time to rethink the paradigm of overwrite or single version

FASTer FTL for OLTP Workloads

SNAPI 2010

Joint Work with Lim and Moon

Motivation

- **FAST**
  - Originally designed for random writes
  - With small log space, just high log block utilization and reduced log block thrashing
- Large scale SSD
  - For better performance, it can employ larger log space
- FAST criticized in DFTL
  - With 3%, performance and fluctuatic

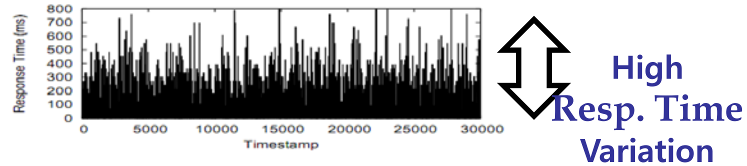


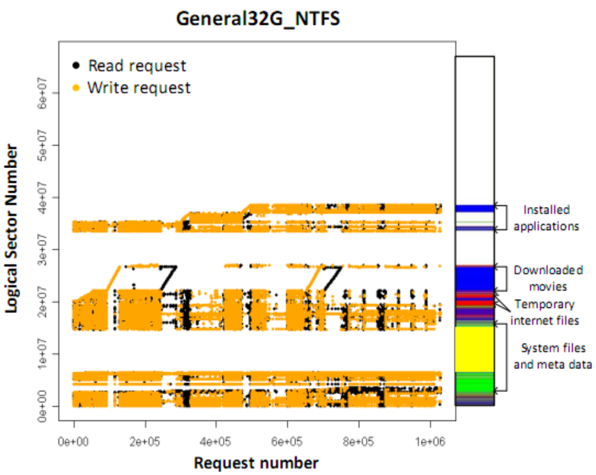
Figure 1: Response Time in FAST(Log Space: 3%)

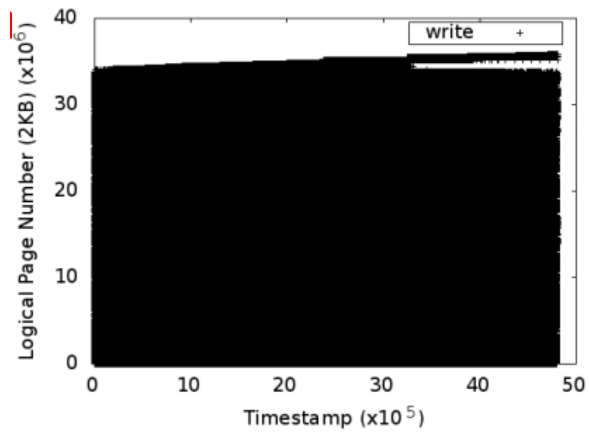
- Revisit FAST with OLTP workloads

Skewed Write Patterns in OLTP

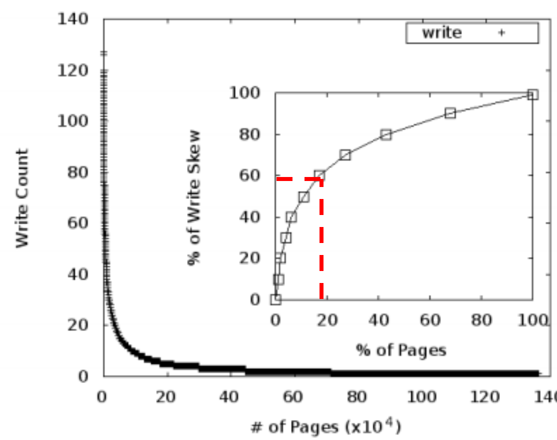
- Write pattern of PC and embedded applications
  - Small-scale range
  - Spatial(i.e, sequential write)/temporal(e.g, meta-data) locality
- How about OLTP applications?
  - Large-scale small random writes (few sequential writes)

Data Set:
8GB TPC-C
Mixed Workload





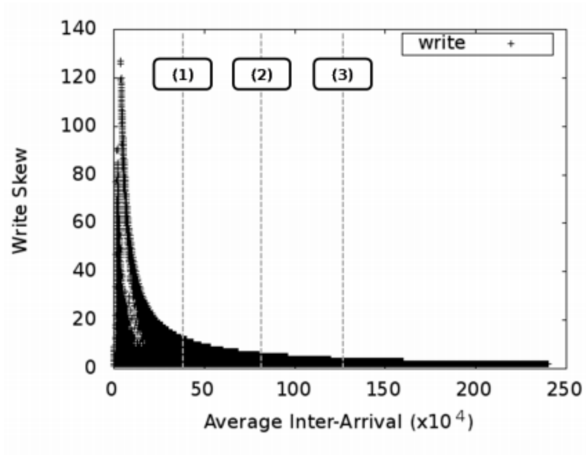
(a) Chronological order



(b) Write count order

Skewed Write Patterns in OLTP

- Temporal locality in OLTP: ‘write arrival interval’ per a page
  - Hot / cold page



(c) Average inter-arrival order

	(1)	(2)	(3)
Provisioning Space (%)	10	20	30
Cumulated Page Writes (%)	52	67	75

Table 2: Log Space vs. Write Buffering(Figure 2.(c))

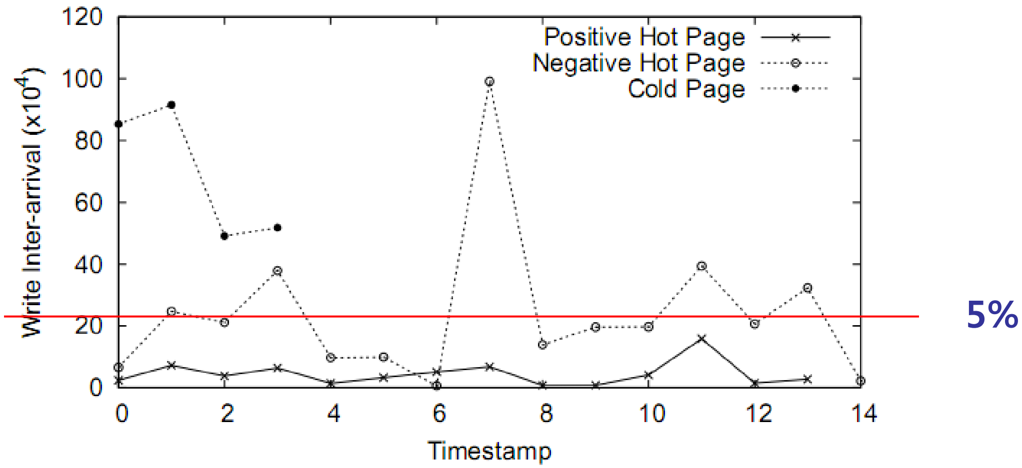


Figure 3: Write Intervals: Hot vs. Cold Page

FAST and Temporal Locality

- DFTL [Aayush Gupta, ASPLOS'09]
  - "FAST dose not provide any special mechanism to handle temporal locality in random streams."
  - With 3% over-provisioning, FAST shows poor performance and high variation

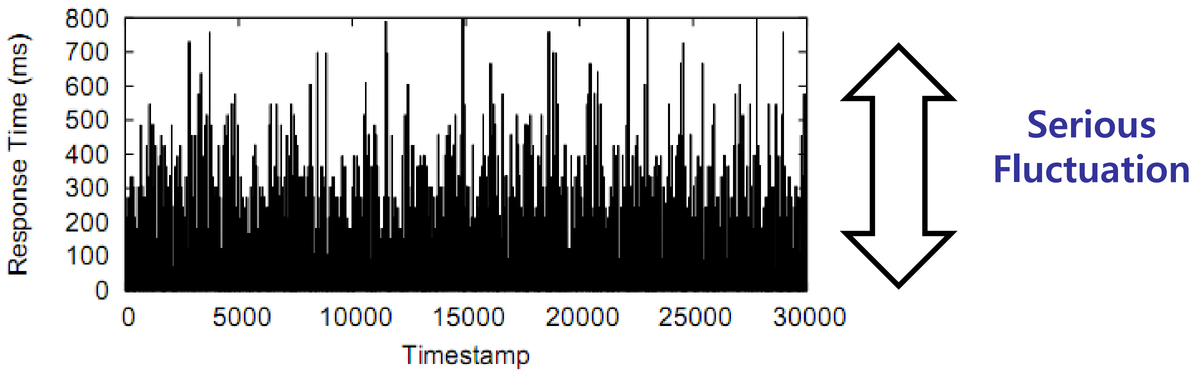
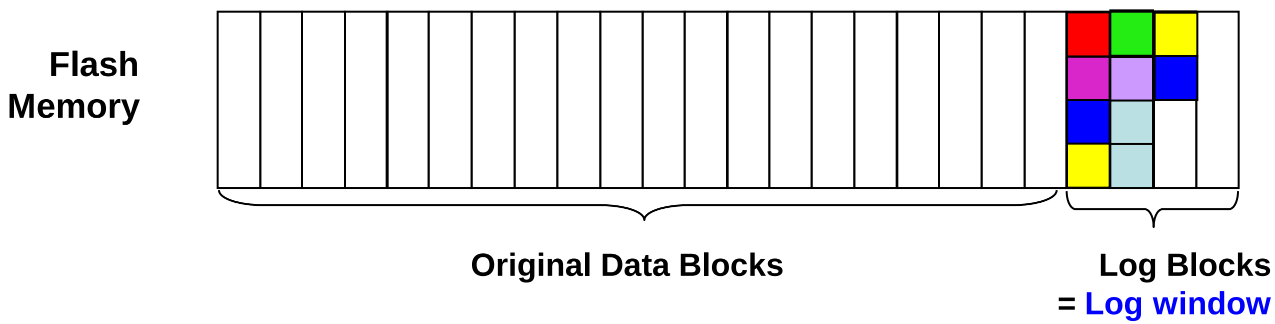
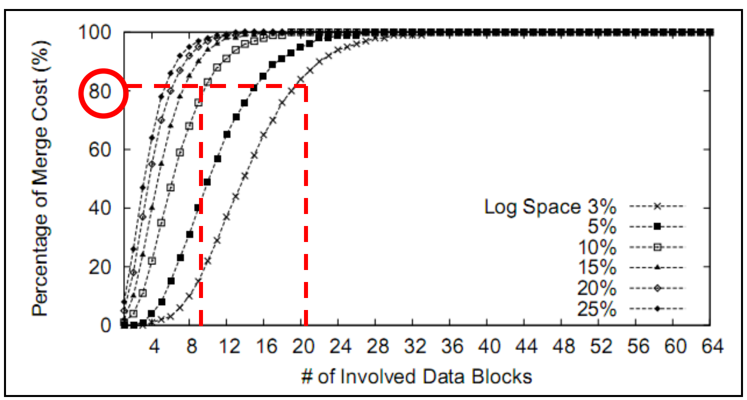


Figure 1: Response Time in FAST(Log Space: 3%)

FAST and Temporal Locality

- Log window ⇧→ data invalidation ⇧→ performance ⇧ & fluctuation of response time ⇩





<Merge Cost Estimation in FAST>

Log Space (%)	3	5	10	15	20	25
Avg. Response Time (ms)	6.12	4.23	2.85	2.49	2.18	1.79
Std. Deviation (ms)	49.51	35.29	20.45	17.08	14.69	11.41

<Temporal locality of OLTP Write patterns in FAST>

FASTer FTL for OLTP Workloads

- FASTer FTL
  - Second chance policy
  - Isolation area
  - No complex processing and meta info. Management overhead
  - Performance improvement
    - ✓ 20~40% than FAST
    - ✓ Even wins Greedy in some(?) cases (pure-page mapping)
  - More uniform response time

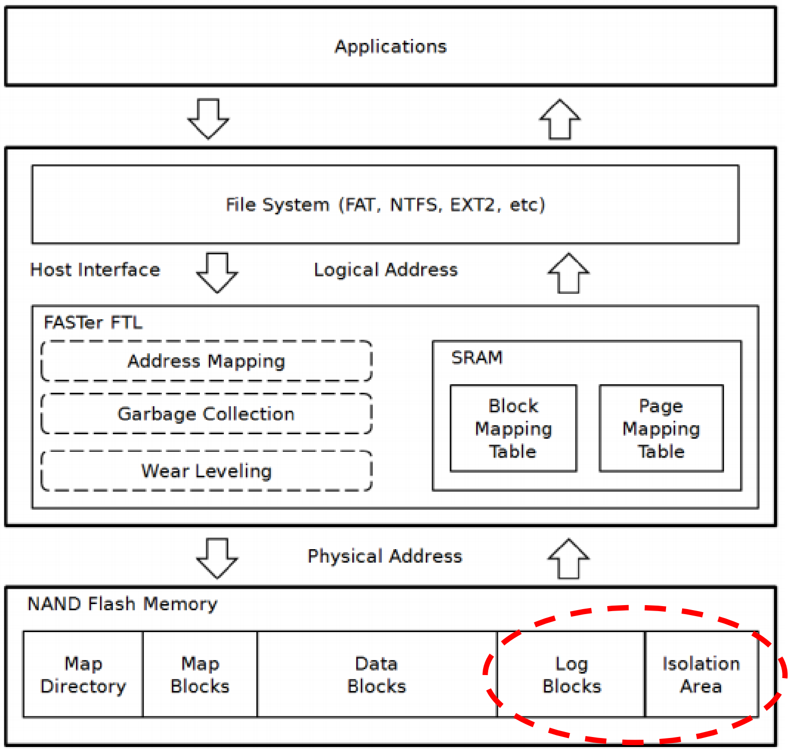
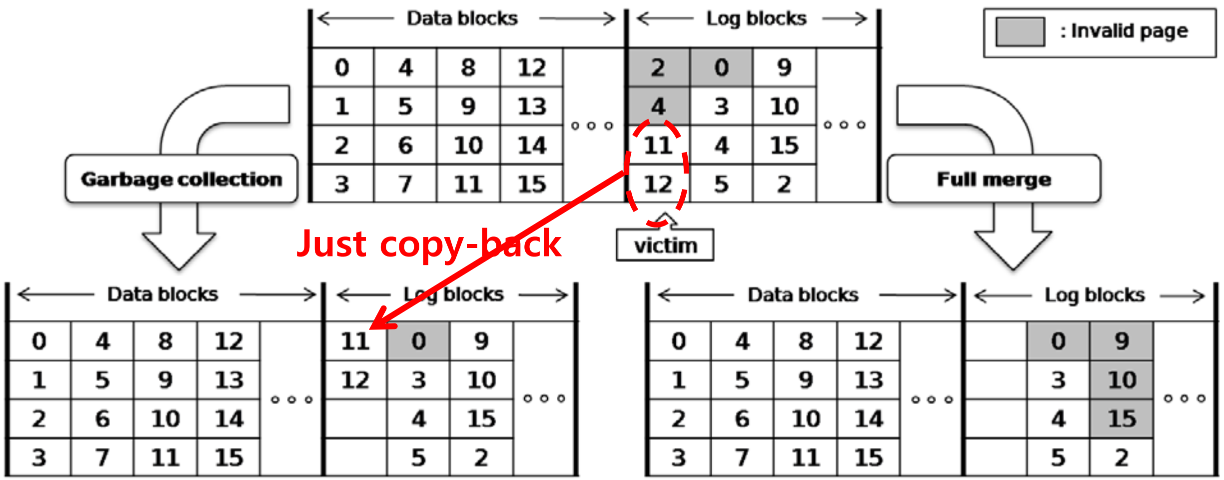


Figure 5: FASTer FTL Architecture

Second Chance Policy

- Give another chance to page in victim block, instead of immediate merge

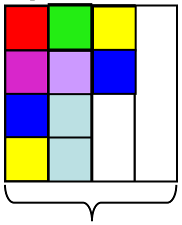


Second Chance Policy(2)

- Pros:
  - If a warm page is invalidated by the second chance, we can avoid costly merges.

- Cons:
  - If the copied page is cold page, we wasted copy time and a precious write buffer resource (reduced “effective log block utilization)

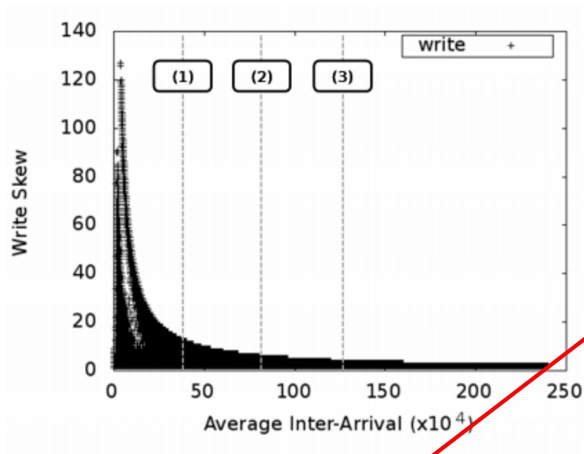
- Pros >> Cons



Log Blocks
= Log window

Second Chance Policy(3)

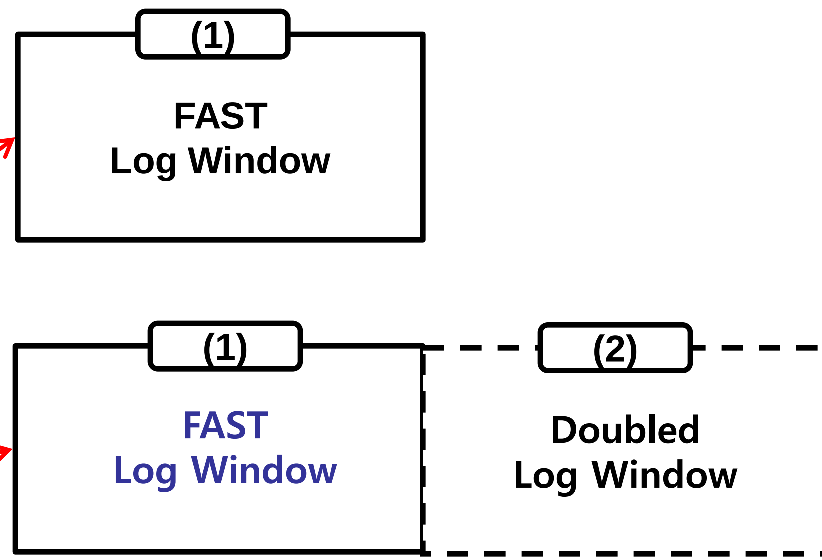
- Double the effective size of log window
- FASTer can skip numerous merges with doubled log window
- Exploit the temporal locality further

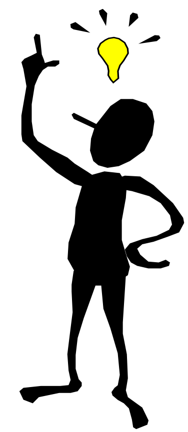


(c) Average inter-arrival order

	(1)	(2)	(3)
Provisioning Space (%)	10	20	30
Cumulated Page Writes (%)	52	67	75

Table 2: Log Space vs. Write Buffering(Figure 2.(c))





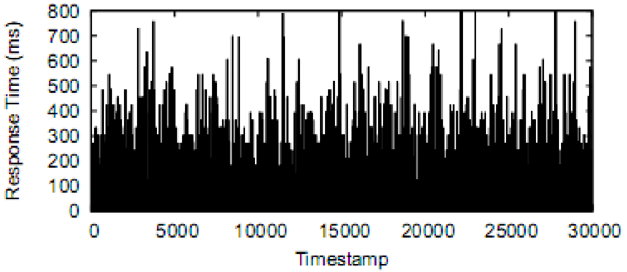
Second Chance Policy

Log Space (%)	3	5	10	15	20	25
Avg. Response Time (ms)	6.12	4.23	2.85	2.49	2.18	1.79
Std. Deviation (ms)	49.51	35.29	20.45	17.08	14.69	11.41

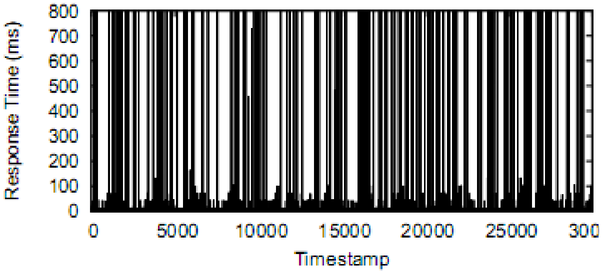
Table 3: Temporal Locality of OLTP Write Patterns in FAST

Log Space (%)	3	5	10	15	20	25
Avg. Response Time (ms)	4.73	2.73	1.92	1.24	1.33	1.09
Std. Deviation (ms)	15.11	11.06	8.78	6.32	6.68	5.56

Table 4: Temporal Locality of OLTP Write Patterns in FASTer



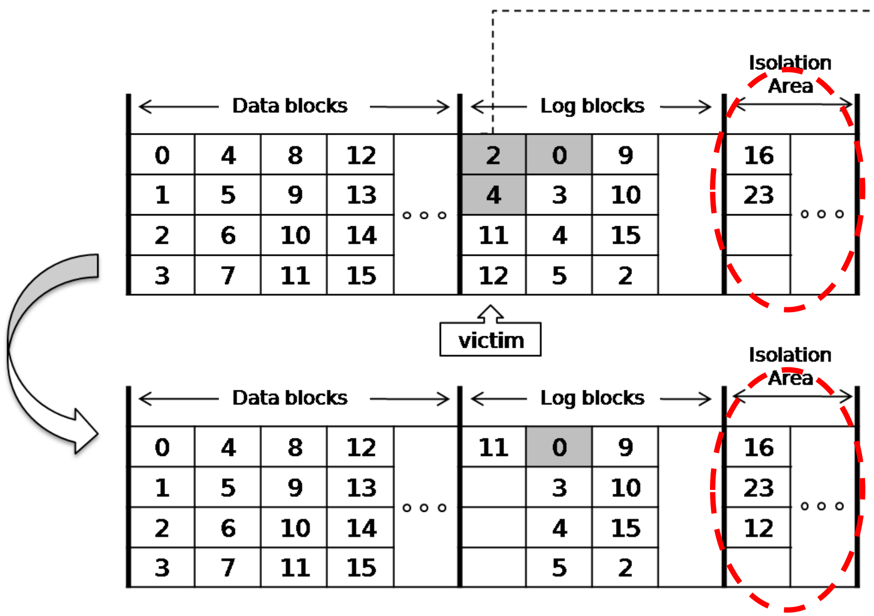
(a) Variation of Response Time in FAST



(b) Variation of Response Time in FASTer (left: no isolation area)

Isolation Area

- Isolation area
  - Write buffering for cold dirty pages
  - Merge progressively in the background
- More uniform response time than FAST



Page Mapping Table

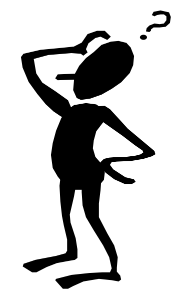
LPN	dataLPN	CBit
0	-	0
1	-	0
2	11	0
3	12	1

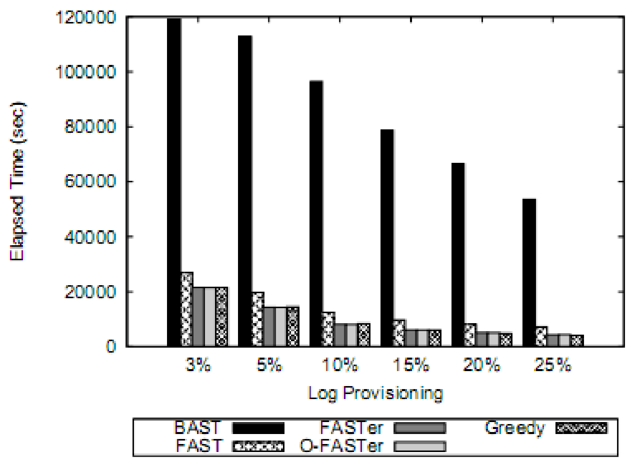
Page Mapping Table

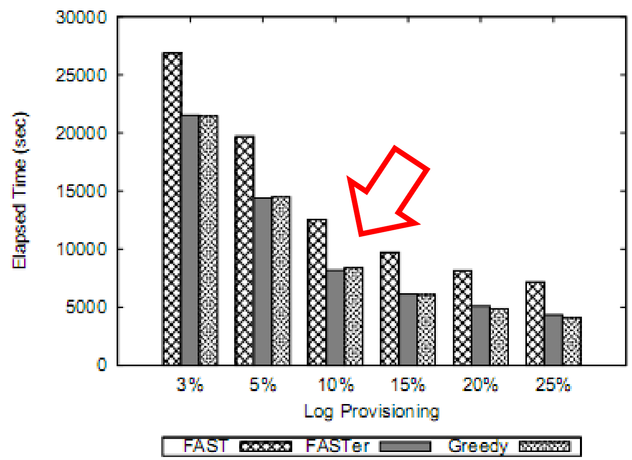
LPN	dataLPN	CBit
0	11	1
1	-	0
2	-	0
3	-	0

Performance Evaluation

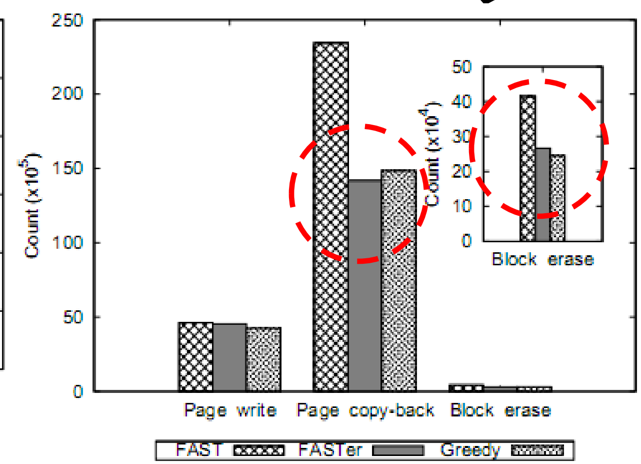
- FASTer w/ 10% > FAST w/ 20% log space
- W/ same log space, FASTer ~~ Greedy
  - With less address mapping information and SRAM







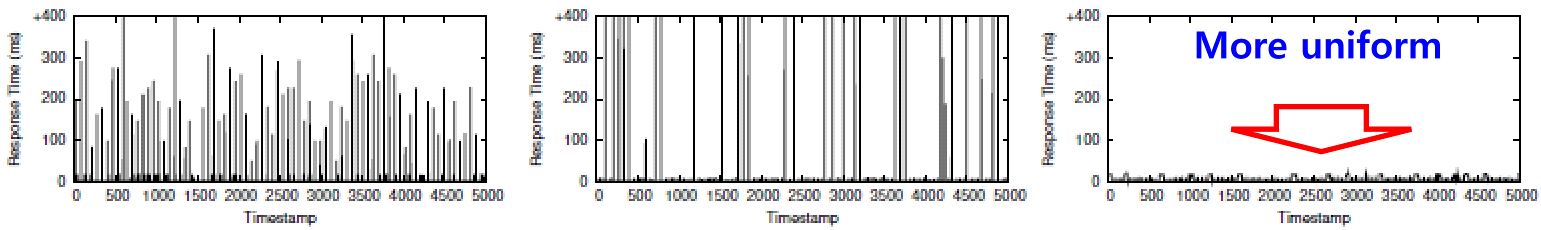
(c) Elapsed time



(d) Primitive Operation Counts(Log Space: 10%)

Performance Evaluation(2)

- FASTer also mitigate the average response time and variations with less provisioning



(a) FAST (b) *FASTer* (without and with an isolation area)

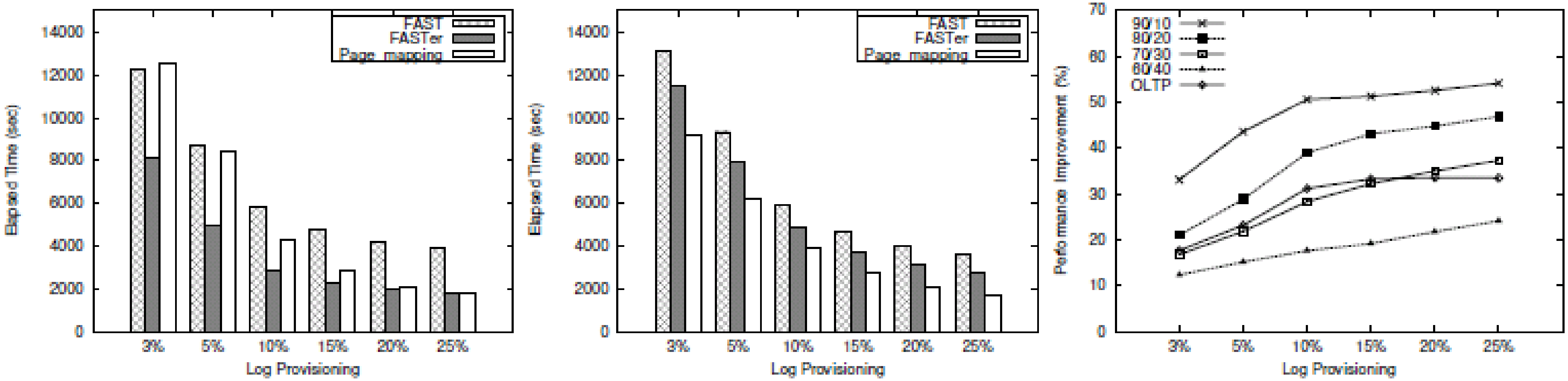
Figure 7. Response time variations with FAST and *FASTer* (log space : 3%)

Log Space (%)		3	5	10	15	20	25
Average Response Time (ms)	FAST	3.59	2.64	1.71	1.33	1.12	1.00
	FASTer (without isolation area)	3.11	2.11	1.24	0.92	0.76	0.66
	FASTer (with isolation area)	3.04	2.08	1.20	0.90	0.75	0.66
	Page mapping	3.00	2.05	1.20	0.89	0.72	0.61
Standard Deviation of Response Time (ms)	FAST	27.6	19.9	12.2	9.01	7.20	6.19
	FASTer (without isolation area)	38.9	30.8	21.6	17.3	14.6	12.7
	FASTer (with isolation area)	5.99	5.00	3.66	3.02	2.64	2.40
	Page mapping	5.73	4.74	3.44	2.77	2.32	2.01

Table II
RESPONSE TIME COMPARISON

Performance Evaluation(3)

- More skewed, better performance



(a) 90/10 (b) 60/40 (c) Improvement ratio between FAST and *FASTer*

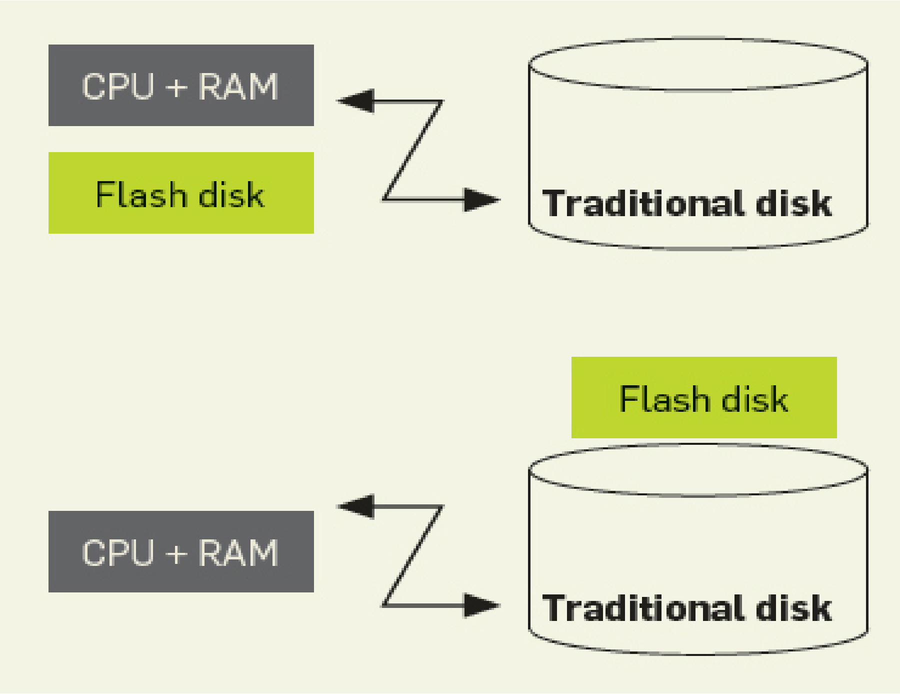
Figure 8. Performance comparison of non-OLTP workloads (synthetic workloads generated using a modified IOzone tool [2])

Flash(SSD) as Extended Buffer Cache

On-going work

Flash: Extended Disk vs. Extended Buffer

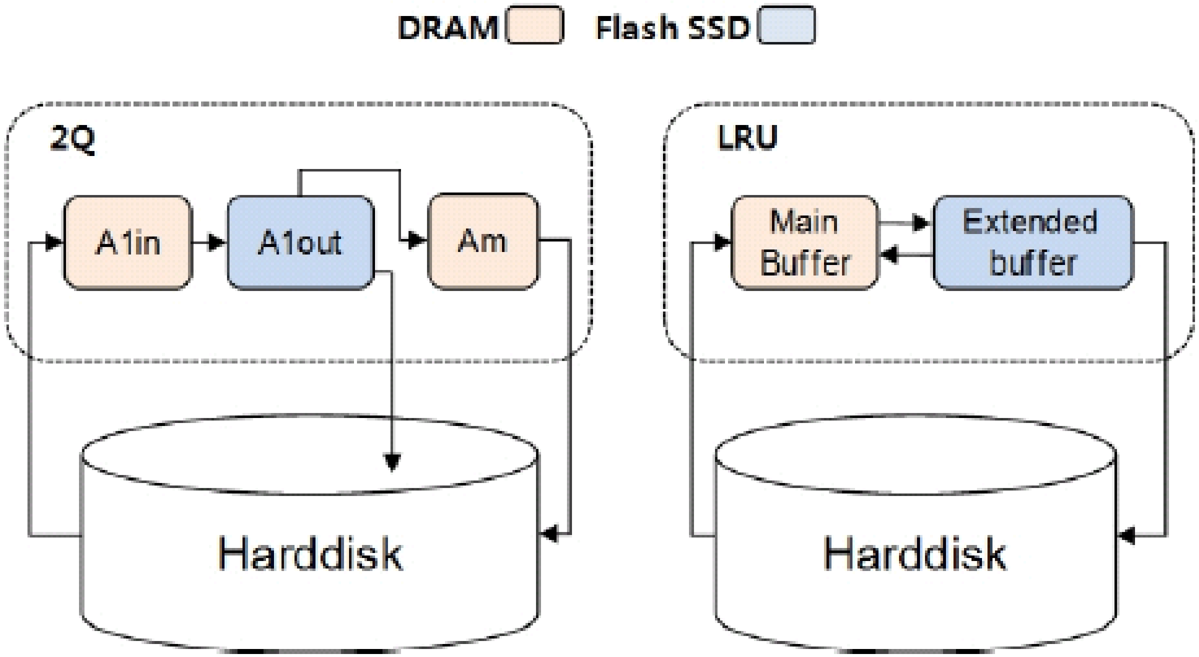
- Source: “The Five-Minute Rule 20 Years Later”, CACM 2009, Graefe



- “Flash as extended disk” approach: “Flashing up the storage layer”, VLDB 2009

Flash as Extended Buffer Cache

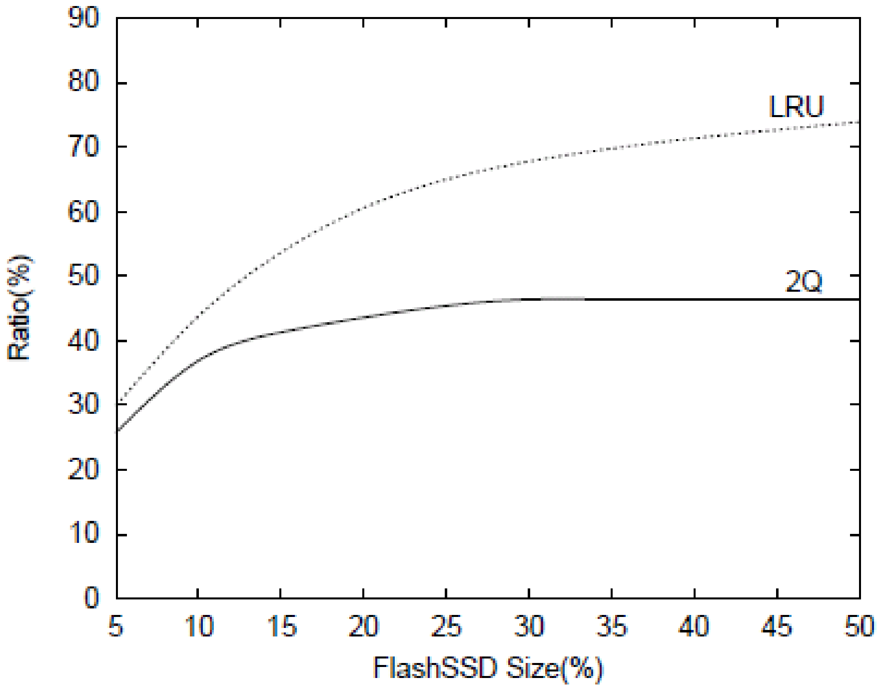
- LRU and 2Q



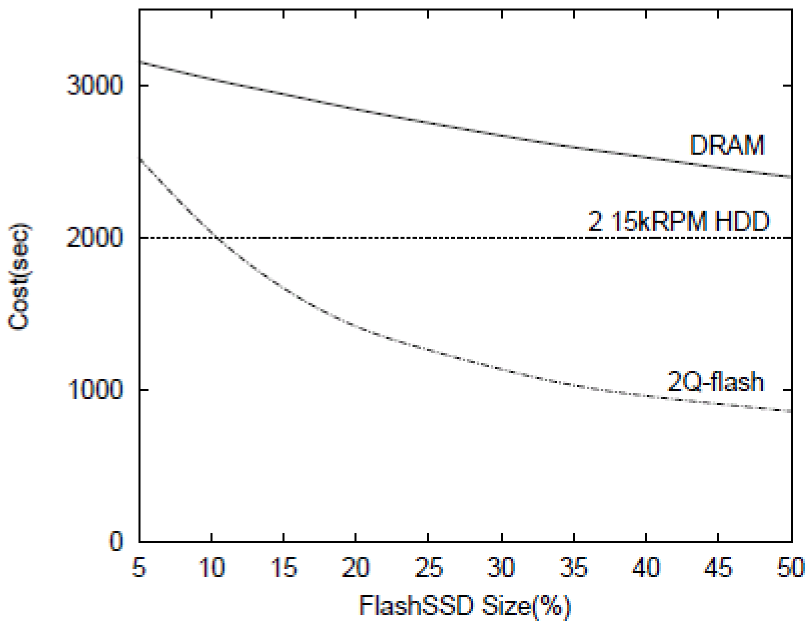
- Intel MLC SSD(80G, 250$): 30000 random reads, 3000 random

Flash as Extended Buffer Cache(2)

- Benefits: Preliminary results



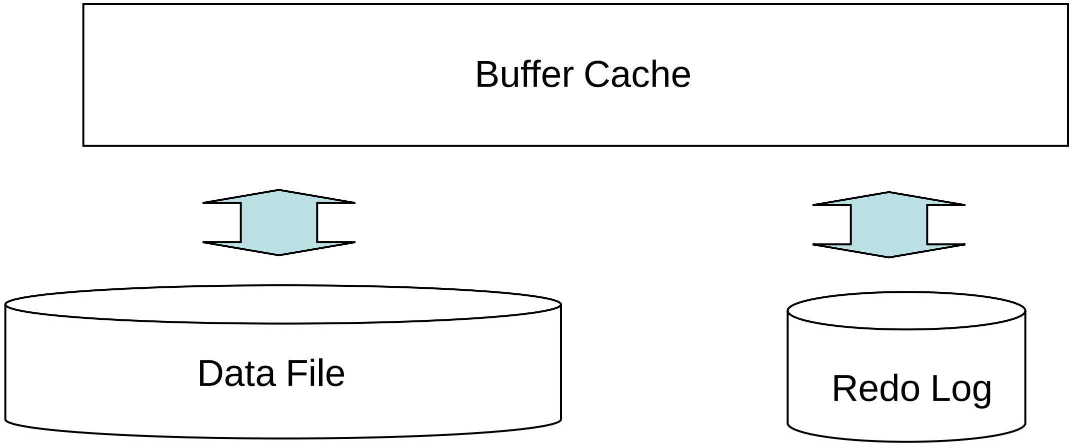
(b) Flash SSD Hit Ratio



A Case for Flash SSD in Database Recovery

On-going work

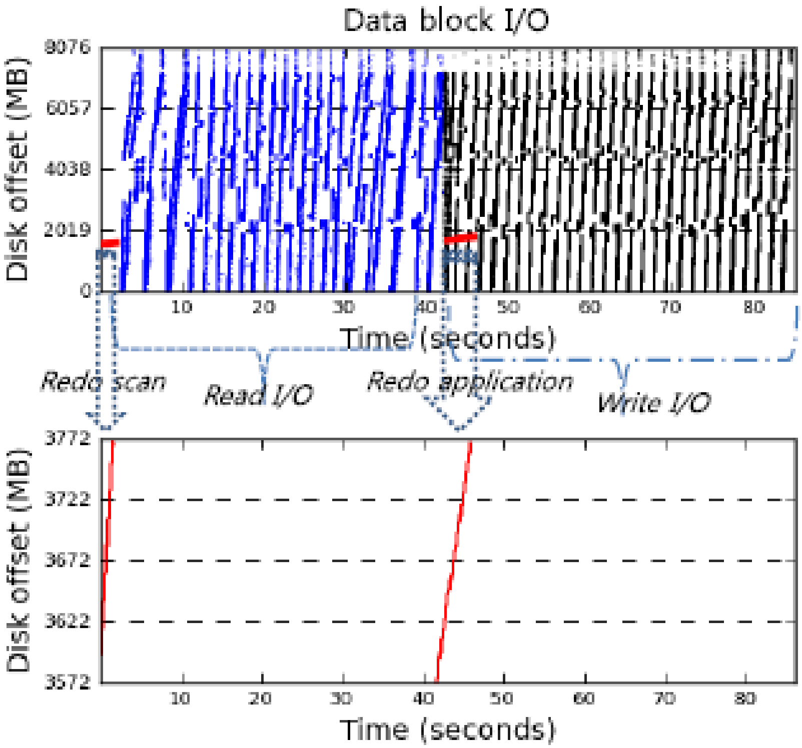
Database Recovery

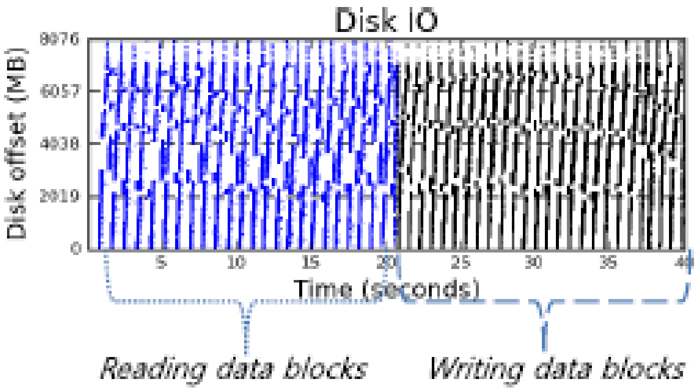


- 4 steps
  - Log scan: seq. scan + CPU
  - Read into buffers to be redo/undo: random IOs
  - Log apply: seq. scan + CPU
  - Write the updated pages to disk: random Ios
- Then vs. now

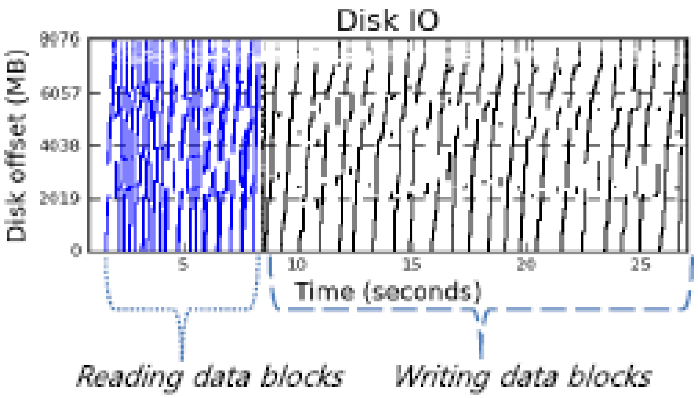
Recovery Performance

- Single 15K HDD, 8 HDDs vs. SLC SSD



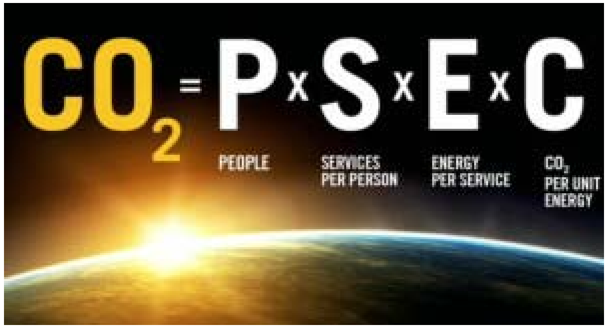


(a) Eight HDD arrays(15K.5-RPM)



Bill Gates

Bill Gates' TED SPEECH 2010



- P: People
- S: Services / person
- E: Energy / service
- C: CO2 / unit energy

- P 는 사람 수다. 빈곤퇴치에 성공할수록 이 숫자는 늘어날 것이다. 제 3세계의 보건 건강 문제가 해결될 것이고, 어린이들이 질병으로 죽어가는 일이 줄고 성인이 사소한 질병으로 목숨을 잃는 일이 줄어들 것이기 때문이다. S는 한 사람이 제공받는 의식주, 의료, 교육 등의 서비스 총량이다. 빈곤퇴치에 성공할수록 S 역시 늘어날 것이다. E는 서비스 1단위 생산에 드는 에너지다. 여기서부터는 좋은 소식이 있다. 기술 발전으로 에너지를 덜 사용하면서도 같은 삶의 질을 유지하는 방법이 늘어나고 있다. 석유를 덜 쓰는 하이브리드 자동차가 대표적 예다. 정작 빌 게이츠가 하고 싶었던 말은 C였다. 보다시피 빈곤을 퇴치할수록 탄소배출은 늘어날 수 밖에 없다. E에서 조금 절감해 볼 수 있지만, 제한적이다. 근본적인 해법은 에너지 생산 과정에서 탄소가 배출되지 않게 만드는 것일 수 밖에 없다는 것이 빌 게이츠의 이야기다. 위의 공식에서 명백하게 드러난다는 것이다. 빌 게이츠는 '테라파워'라는 새로운 아이디어 하나를 제시한다. 폐우라늄을 활용한 원자력 발전이다. 탄소배출이 적으면서도 싸게 공급될 수 있는 혁신적 기술이다. 그러나 이는 하나의 아이디어일 뿐이라고 스스로 말한다. 다만 그는, 인류의 양대 문제인 빈곤과 기후변화를 동시에 이겨내려면, 에너지 생산에 배출되는 탄소를 줄일 수 있는 혁신이 반드시 필요하다고 강조한다. 테라파워와 비슷한 아이디어가 계속 나와야 한다는 것이다.

(출처: Bill @ TED, www.ted.org, http://goodeconomy.hani.co.kr/blog/archives/788)

Storage Metrics in OLTP

- In OLTP databases
  - 2009, 1 flash SSD >> 10 15K rpm HDDs
  - 2010, 1 flash SSD >> 20 15K rpm HDDs

- Storage Metrics = f(Performance(IOPS) X Cost X Energy x Endurance X People X ?????)