36th International Conference
on Massive Storage Systems
and Technology (MSST 2020)
October 26-30, 2020

Sponsored by Santa Clara University,
School of Engineering

PAPA: Partial Page-aware Page Allocation in TLC Flash SSD for Performance Enhancement

Imran Fareed, Mincheol Kang, Wonyoung Lee, and Soontae Kim

Embedded Computing Lab

Korea Advanced Institute of Science and Technology (KAIST)

Outline

- Background and Motivation
- Design of PAPA
- Evaluation
- Conclusion

Background and Motivation

	Read/Write [1][2]
MSB page	110/5500 us
CSB page	100/2000 us
LSB page	50/500 us

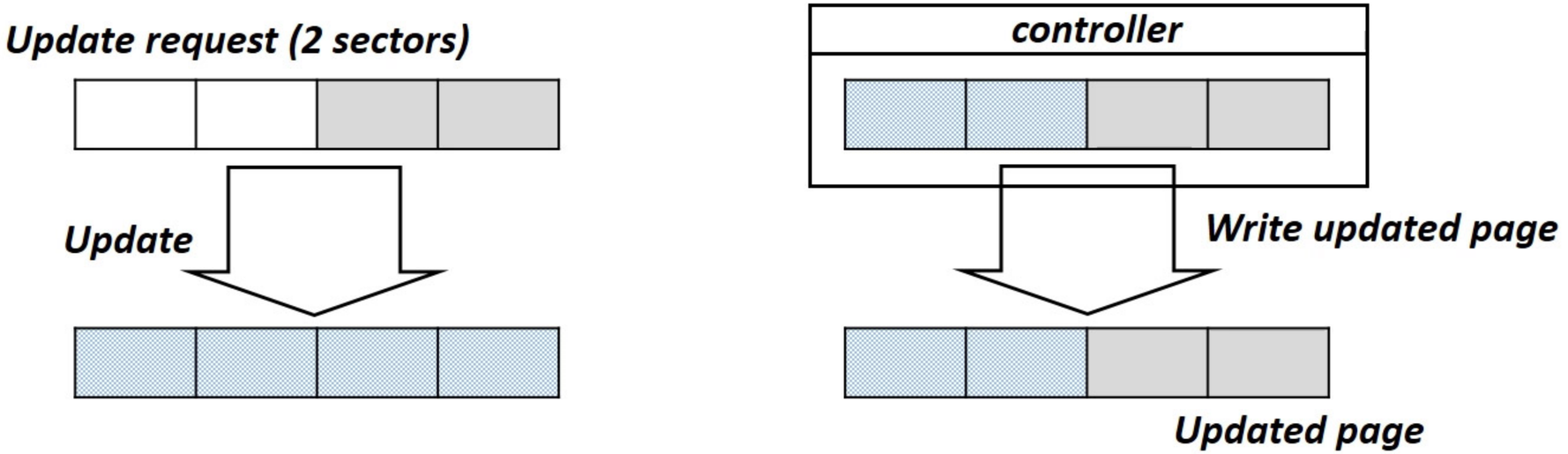
Partial page vs full page

Partial page full page

Partial page update is more costly than the full page update, owing to the read-modify-write property of the flash memory

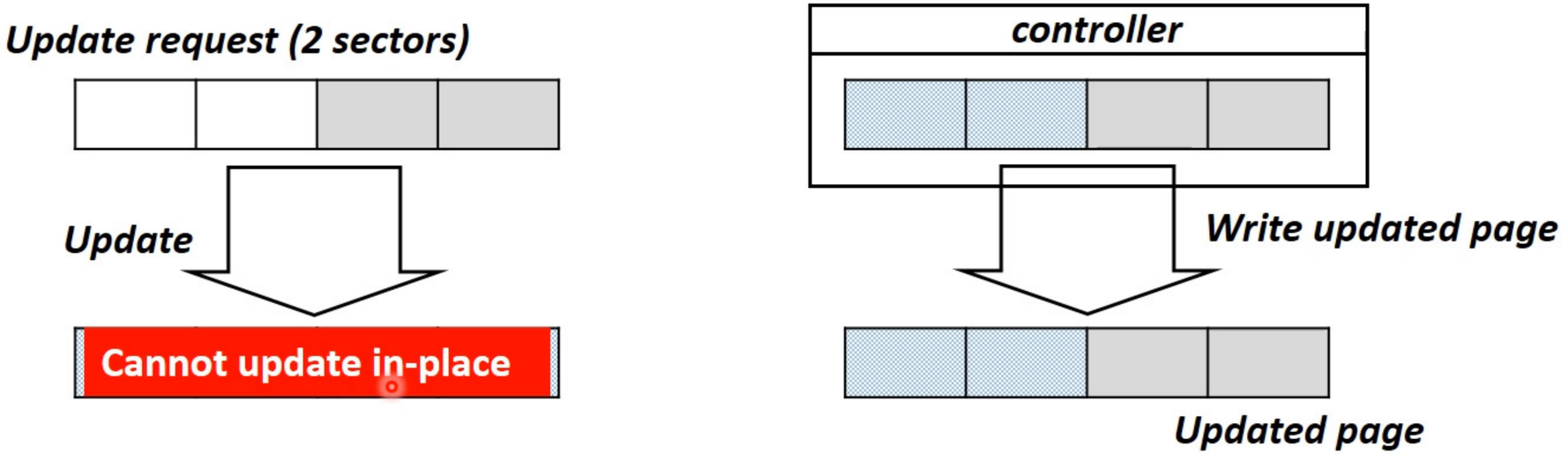
[1] W. Choi, M. Jung, and M. Kandemir, “Invalid data-aware coding to enhance the read performance of high-density flash memories,” in 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). IEEE, 2018, pp. 482–493. [3]
[2] W. Zhang, Q. Cao, H. Jiang, and J. Yao, “Improving overall performance of tlc ssd by exploiting dissimilarity of flash pages,” IEEE Transactions on Parallel and Distributed Systems, 2019.

Out-place update



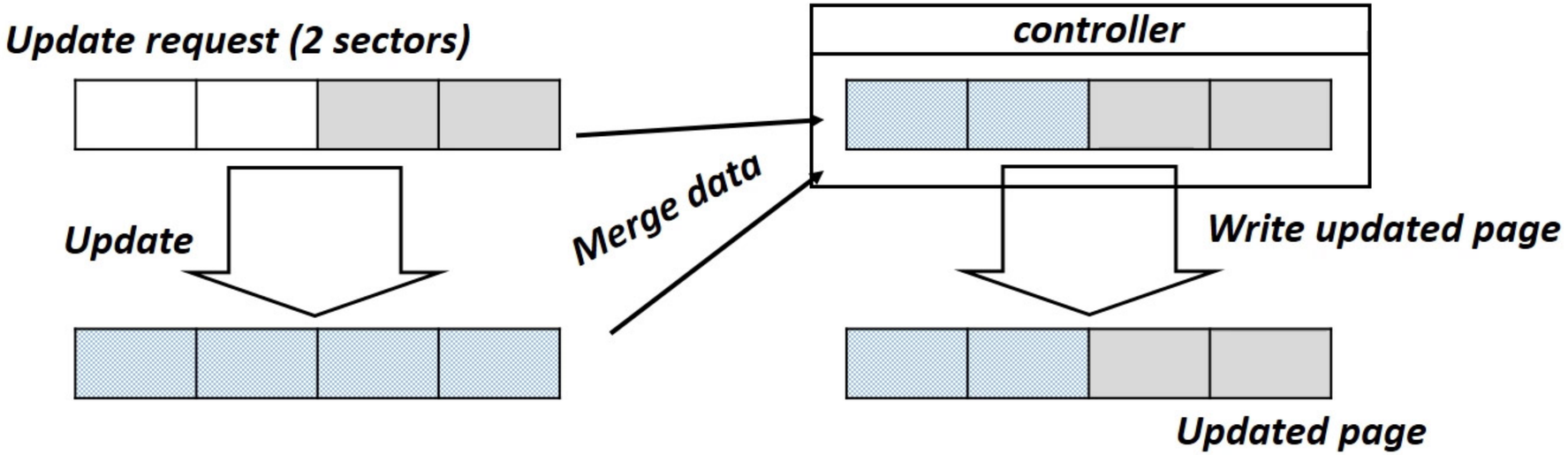
- In a partial update, old data is read, modified, and rewritten to a new page

Out-place update



- In a partial update, old data is read, modified, and rewritten to a new page

Out-place update



- In a partial update, old data is read, modified, and rewritten to a new page
  - ❖Extra read operation => Performance degradation

Partial-update with various page types

MSB page	110/5500 us	*highest cost*
CSB page	100/2000 us	*medium cost*
LSB page	50/500 us	*lowest cost*

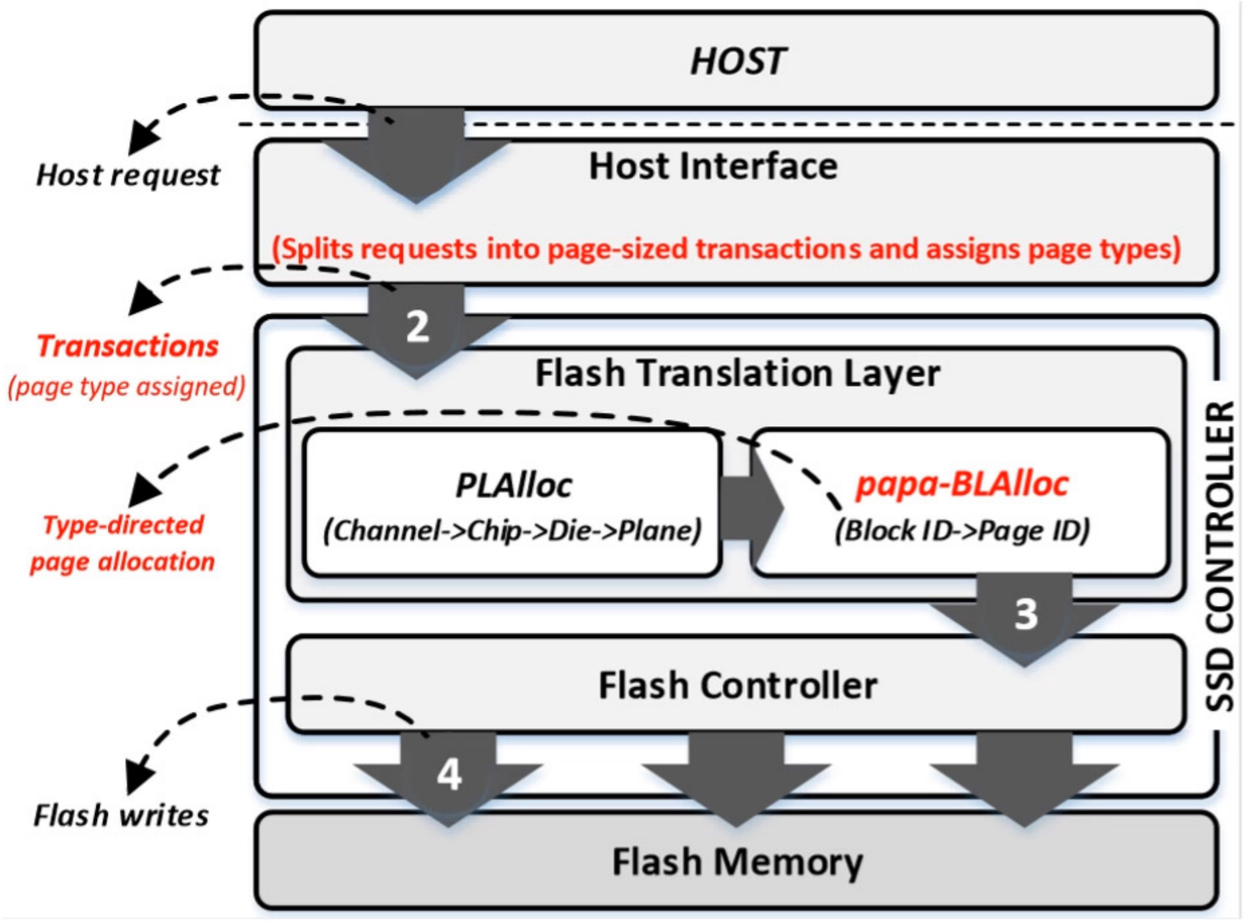
❖**MSB pages incur highest read/write latencies, thus should be avoided**

Outline

- Background and Motivation
- **Design of PAPA**
- Evaluation
- Conclusion

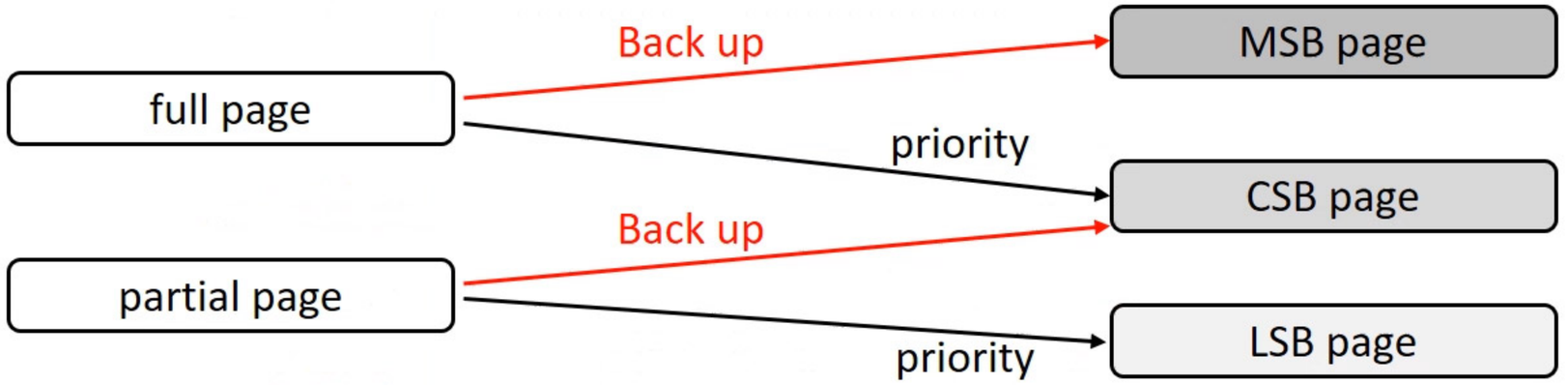
PAPA: Partial page-aware page allocation

Design



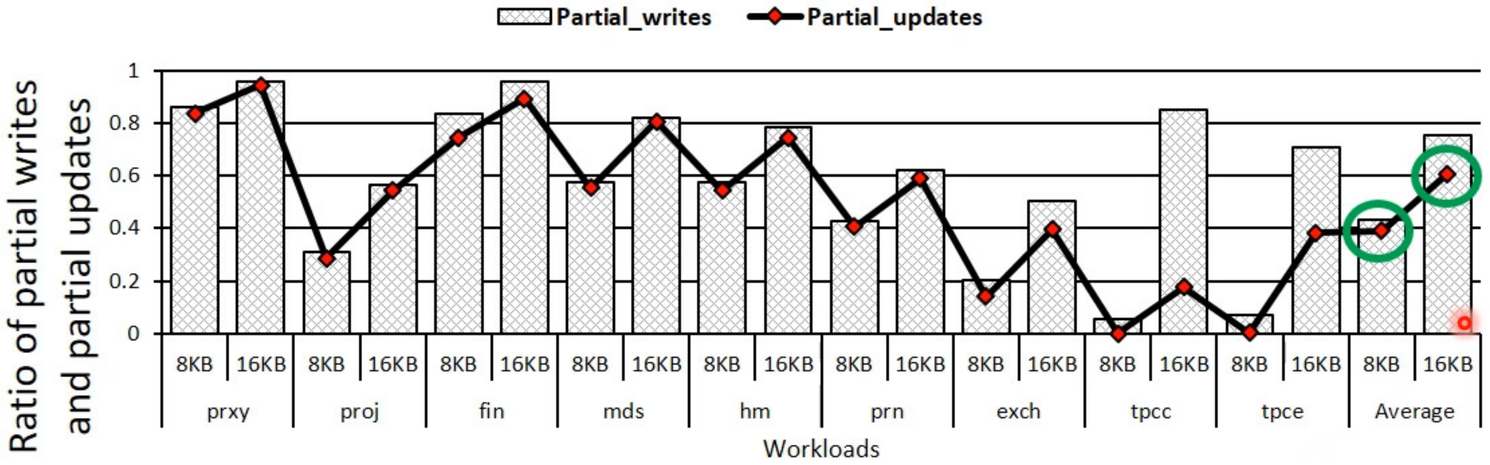
PAPA: Partial page-aware page allocation

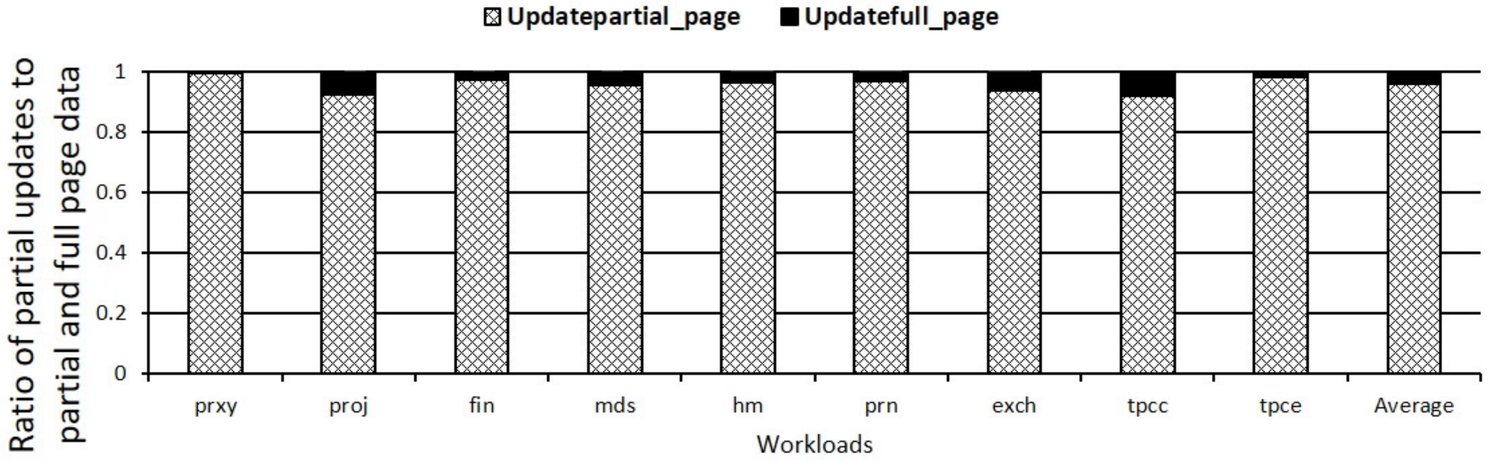
Type-directed page allocation



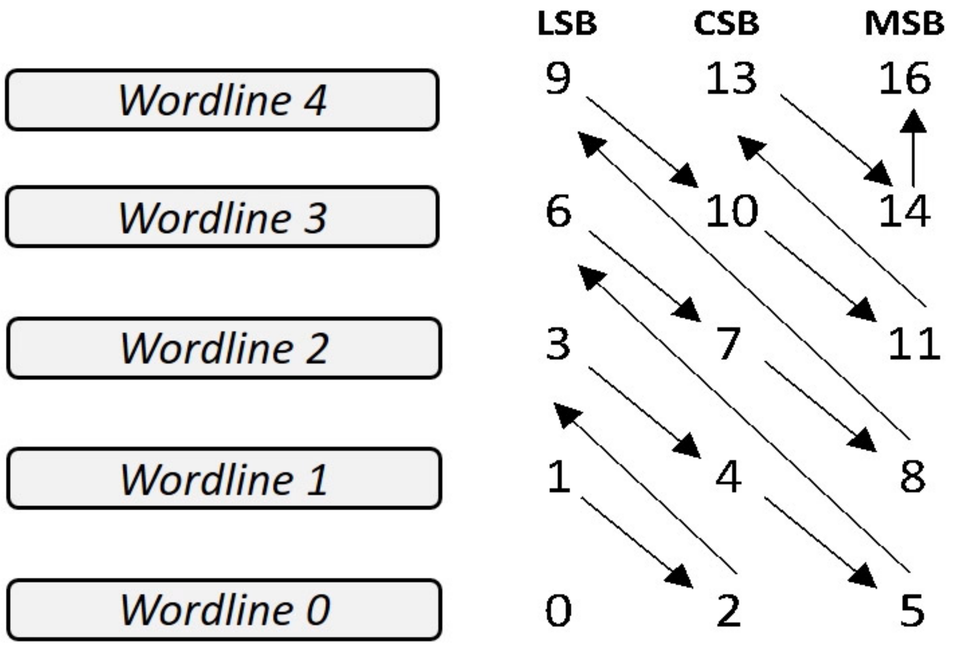
- Assign partial pages to low-cost LSB pages
- CSB pages are backup pages for partial writes
- Allocate full page writes to MSB pages
- CSB pages are backup pages for full page writes

Why differentiate partial and full page writes

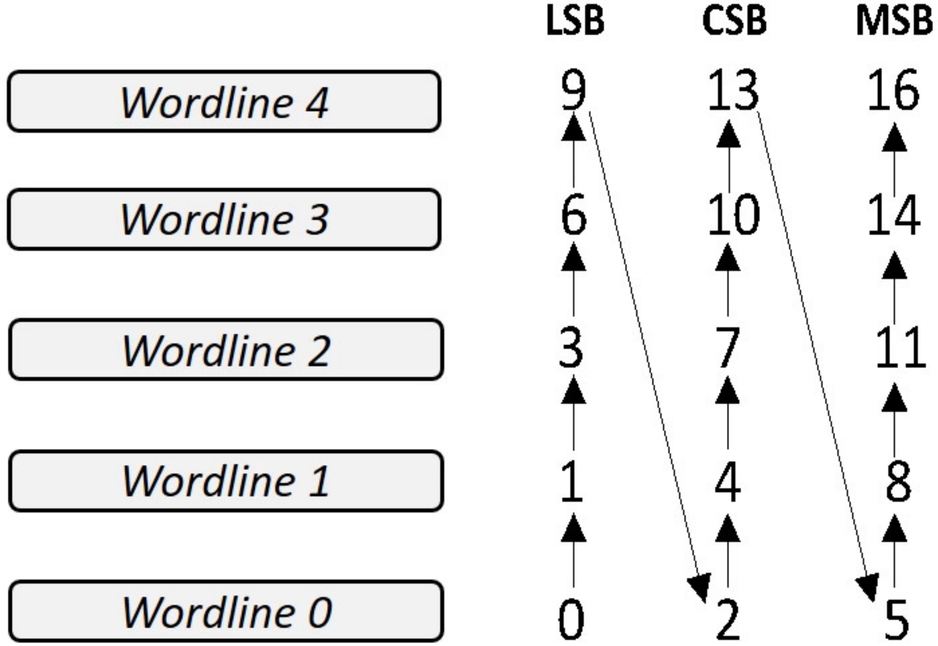




Fixed vs Relaxed program order

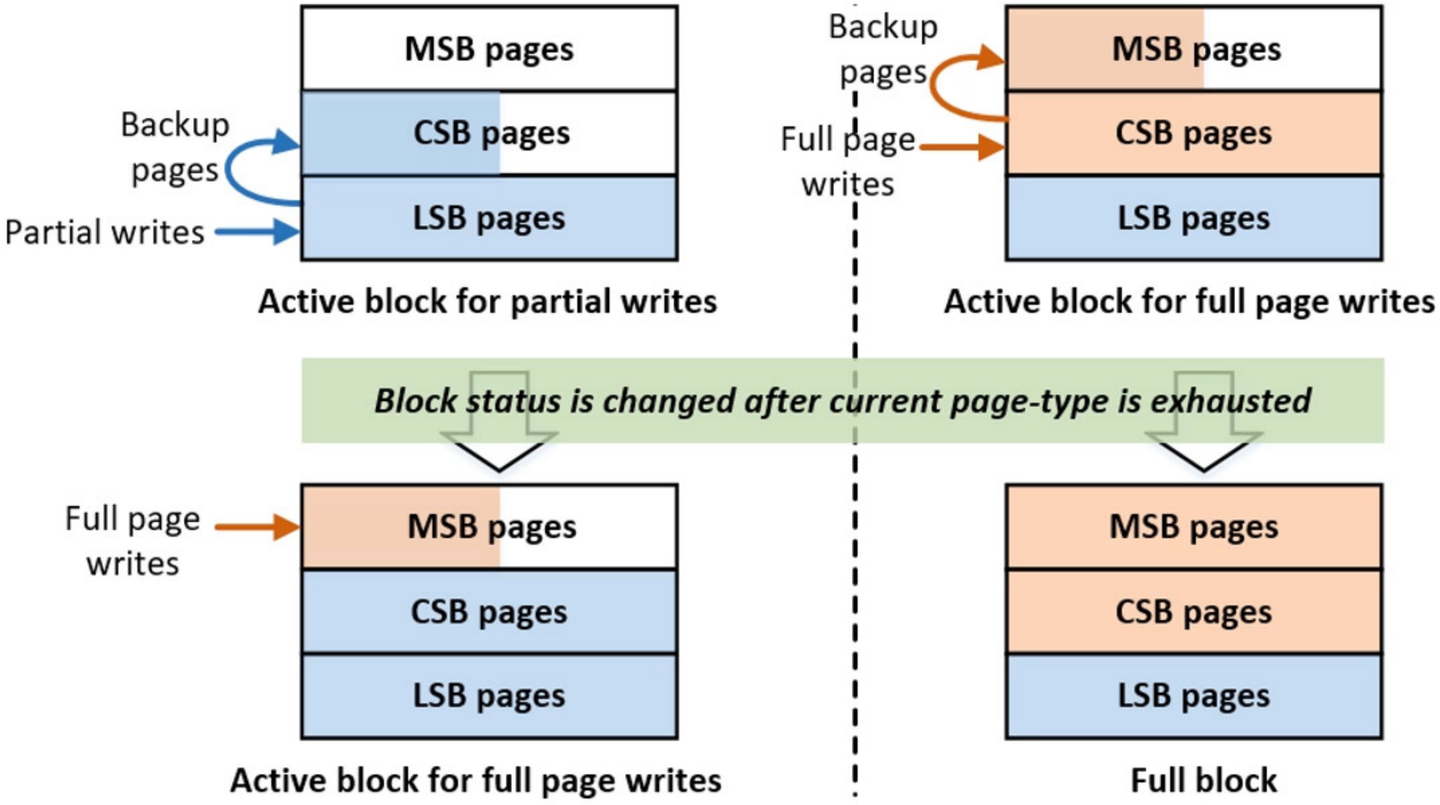


Fixed program order



Relaxed program order

Page allocation



Outline

- Background and Motivation
- Design of SLC
- **Evaluation**
- Conclusion

Experiment

Traces	Total request	Writes (%)	Reads (%)	Avg size (bytes)
prxy	1048576	95	5	2421
proj	1048582	16	84	29112
fin	5334987	77	23	4196
mds	1048576	87	13	7523
hm	1048498	73	27	6086
prn	1048579	86	14	12544
tpce	4510214	27	73	17701
tpcc	6286764	67	33	11942
exch	766362	69	31	12768

Characteristics of evaluated IO traces

Items	Value	Items	Value
Packages	2	Sector size	4 KB
Dies/package	2	DRAM cache	disabled
Blocks/die	4096	Reads latency	*LSB/CSB/MSB 50/100/150 µs*
Pages/block	384	Writes latency	*LSB/CSB/MSB 500/2000/5500 µs*
Page size	16 KB	Erase latency	3000 µs

Simulation configurations

Implementation

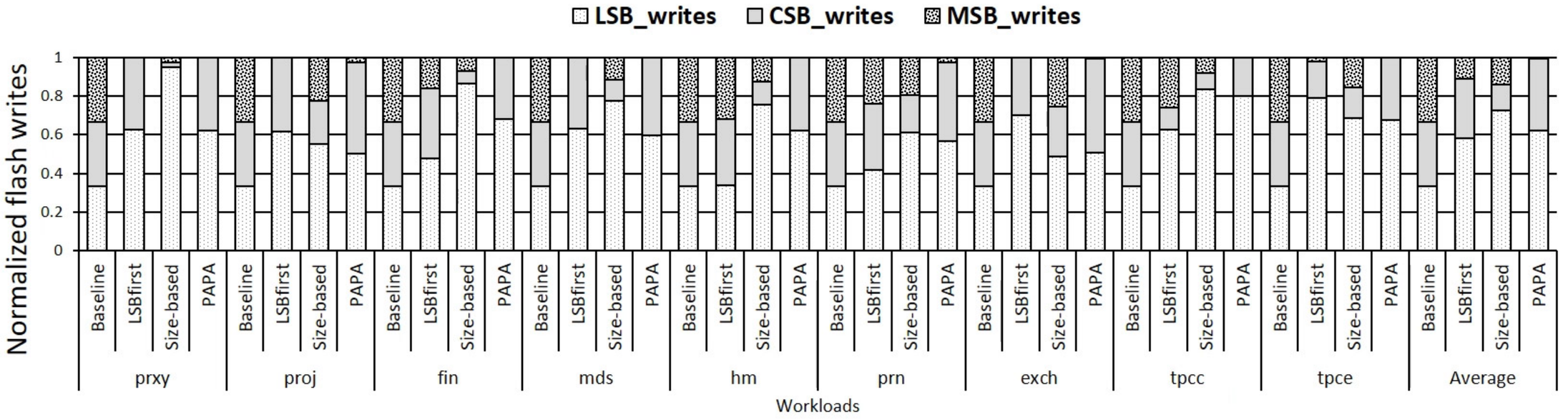
- Simulator: Flashsim [1]
- Workloads from SNIA [2] and OLTP applications [3]
- Evaluated schemes:-
  - Baseline: conventional program sequence
  - LSB_{first} scheme **(consume LSB pages first)**
  - Size-based scheme **(allocate LSB pages to small writes)**
  - Our proposed scheme **PAPA**

[1] Y. Kim, B. Tauras, A. Gupta, and B. Urgaonkar, “Flashsim: A simulator for nand flash-based solid-state drives,” in 2009 First International Conference on Advances in System Simulation. IEEE, 2009, pp. 125– 131.

[2] SNIAIOTTARepository.[n.d.].SNIARepository. http://iotta.snia.org/traces/130

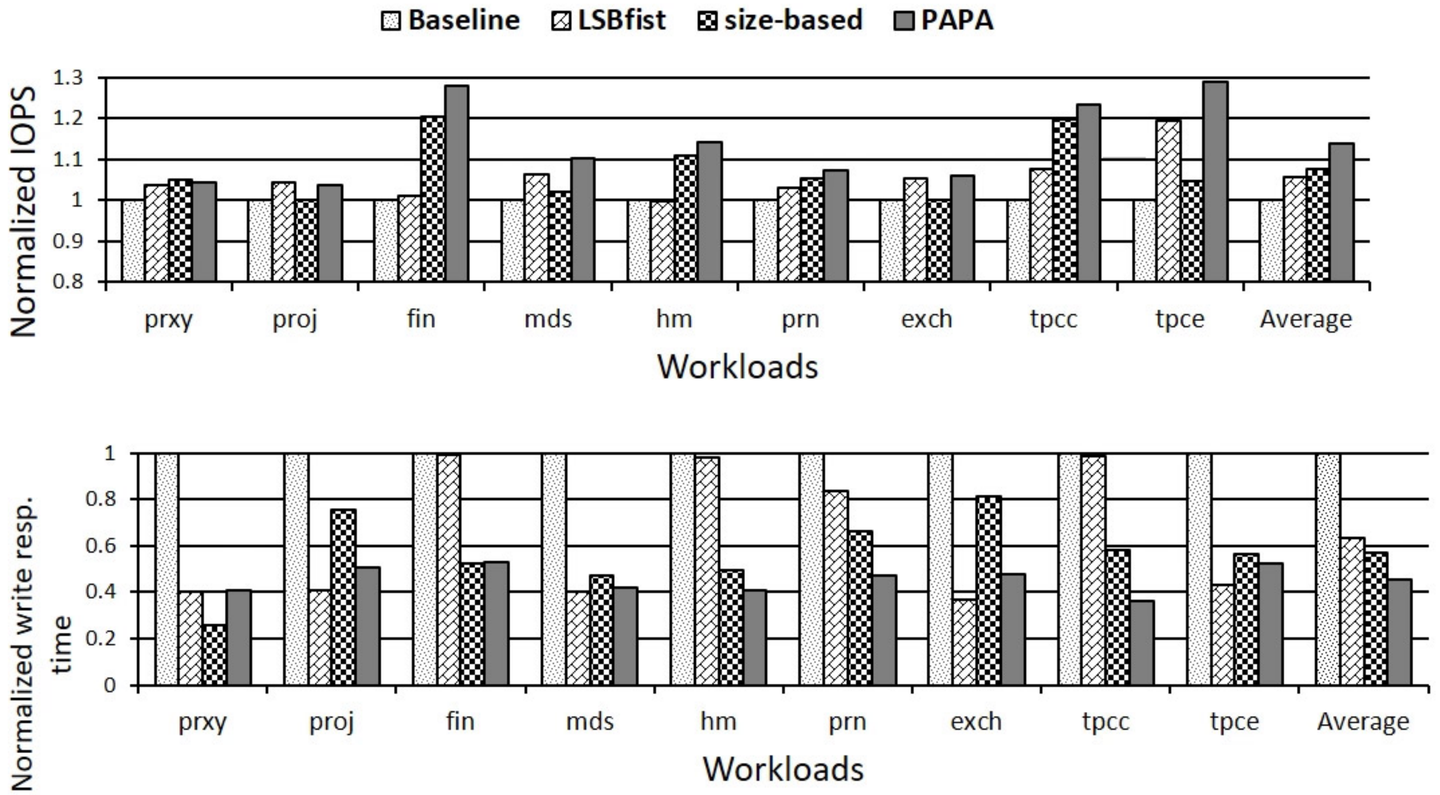
[3] Umasstracerepository.[n.d.]. Umasstracerepository. http://traces.cs.umass. edu/index.php/Storage/Storage

Results: Flash Writes Distribution



- ***PAPA*** allocates LSB and CSB pages to 62% and 37.2% writes, respectively

Results: performance improvement



- ***PAPA*** improves the throughput by **14%** on average and reduces the write response time by **55%**

Outline

- Background and Motivation
- Design of SLC
- Evaluation
- **Conclusion**

Conclusion

- Different types of pages in the TLC flash memory exhibit variable read/write latencies
- Partial updates incur additional latency (RMW latency)
- Conventional TLC programming designs follow a type-blind page allocation
- We proposed a relaxed program order which uses type-directed page allocation
- The throughput and write response time are improved by **14%** and **55%,** respectively

Thanks for your presence