TECHNISCHE UNIVERSITÄT DARMSTADT

INF Informatik
Hochschule Reutlingen

Advances in Flashing the Database Storage

Ilia Petrov, Robert Gottstein, Sergey Hardock

Data Management Lab, Reutlingen University

Databases and Distributed Systems Group, TU-Darmstadt

Acknowledgements:
Alex Buchmann, Thorsten Peter, Paul Dubs, Todor Ivanov, Guilermo Almeida

Introduction

- Significant hardware evolution and emerging technologies

- Architectures of (database) systems lag behind
  - DB architectures are 20-30 years old
  - Algorithms based on 20 years old assumptions

- **Takeaway Message:**
- Simple hardware replacement: expensive, inefficient!
- Reevaluate algorithmic and architectural assumptions!
- Redesign needed!
- DBMS can control hardware directly:
  - Rethink existing abstractions and model
  - Simplify software stack (reduce redundancies)

Evolution of storage technology latencies

TECHNISCHE UNIVERSITÄT DARMSTADT
INF Informatik
Hochschule Reutlingen

(Source: IBM)

IBM 350 (*1956*):

50 Disk | 5MB | 1200 RPM | 1 IOPS | 1 Ton

100x

2000

100x

2010

100x

2012

Flash versus Hard Disk Storage

2x … 300x

- Asymmetry
- Parallelism
- Performance, Low access times
- Moore's law development
  - low Price/GB, Performance increasing
- Longevity
- Error Correction
- *Block Device Compatibility*

> 1000x

Amdahl's Law – Speedup [1]

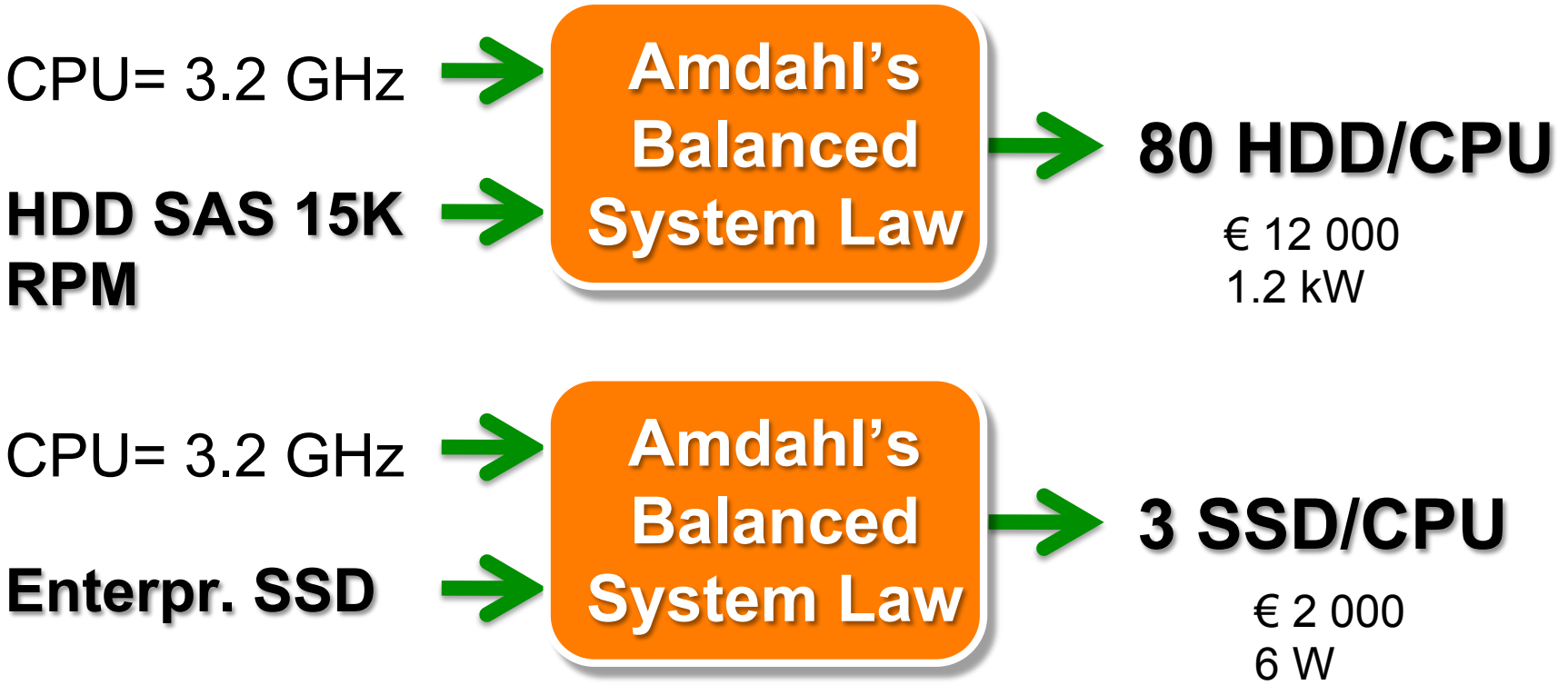
- An OLTP database performs IO approx. 60% of the time [Patterson]
- 100x faster IO-Subsystem?

If Flash SSDs are treated as HDD **replacements**: a 10x..1000x faster SSD will make DBMS run ~2.2x..2.5x faster! (Amdahl)**. Not Efficient!**

[1] Amdahl, Gene. "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities". In Proc. AFIPS Conference pp.483–485. 1967

Amdahl's Revised Balanced System Law [2]:

- A system needs 8 MIPS/MB/s IO
  - The instruction rate and IO rate workload dependent → OLTP, CPI=2.1
  - Assume 75% random write, 25% random read, 8KB page size, 3.2 GHz CPU



[2] Jim Gray, Prashant Shenoy, "Rules of Thumb in Data Engineering," ICDE 2000

In brief …

- Data-Intensive systems IO-Bound & built around HDD properties
  - HDD performance hits physical limits
  - Jim Gray: “Tape is Dead, Disk is Tape, Flash is Disk, RAM Locality is King”
- **New Storage Technologies – arrive on time!**
  - New performance characteristics.
  - Moore’s Law governs development?
- **Improvements needed to utilize Flash characteristics**
  - Algorithmic and Architectural
  - Hardware improvements
- **Hardware: Flash devices treated as HDD replacements and black-boxes**
  - Backwards compatibility → Cheep replacement
  - Legacy hardware and software interfaces

Flash SSDs – Basic Properties

TECHNISCHE UNIVERSITÄT DARMSTADT
INF Informatik
Hochschule Reutlingen

DBlab DVS

I/O Model: HDD and Flash

$$T_{access}(Addr., Blocks) = T_{positioning}(Addr.) + T_{transfer}(Blocks)$$

$$T_{access}(Addr., Blocks) = \overbrace{T_{seek}(Addr.) + T_{rotate}} + T_{transfer}(Blocks)$$

4KB → 5.5ms = 4ms + 1ms + 0.5ms

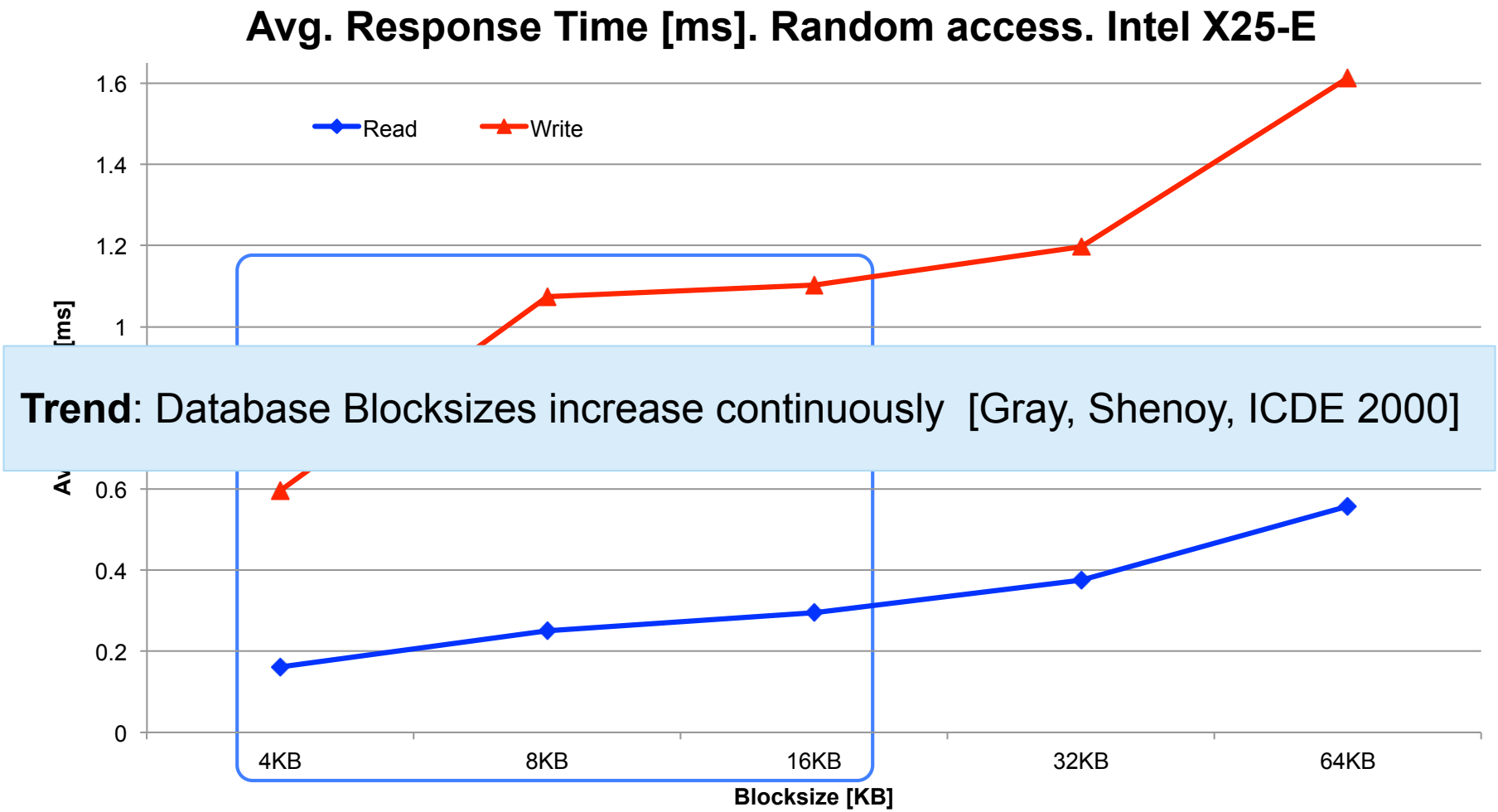
$T_{positioning} >> T_{transfer}$

Trend: Database Blocksizes increase continuously [Gray, Shenoy, ICDE 2000]

Trend: Random operations are expensive: avoid!

$T_{initialisation/Pos.} << T_{transfer}$

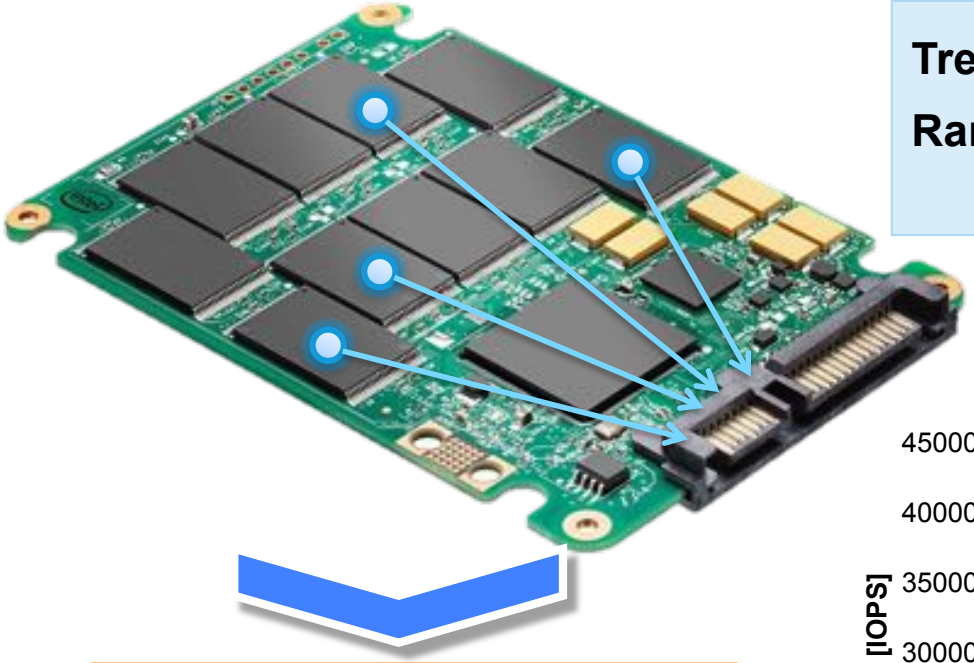
Flash SSD: Latencies and Blocksizes



Trend: Database Blocksizes increase continuously [Gray, Shenoy, ICDE 2000]

Flash SSDs: Low (read) latencies for small block sizes

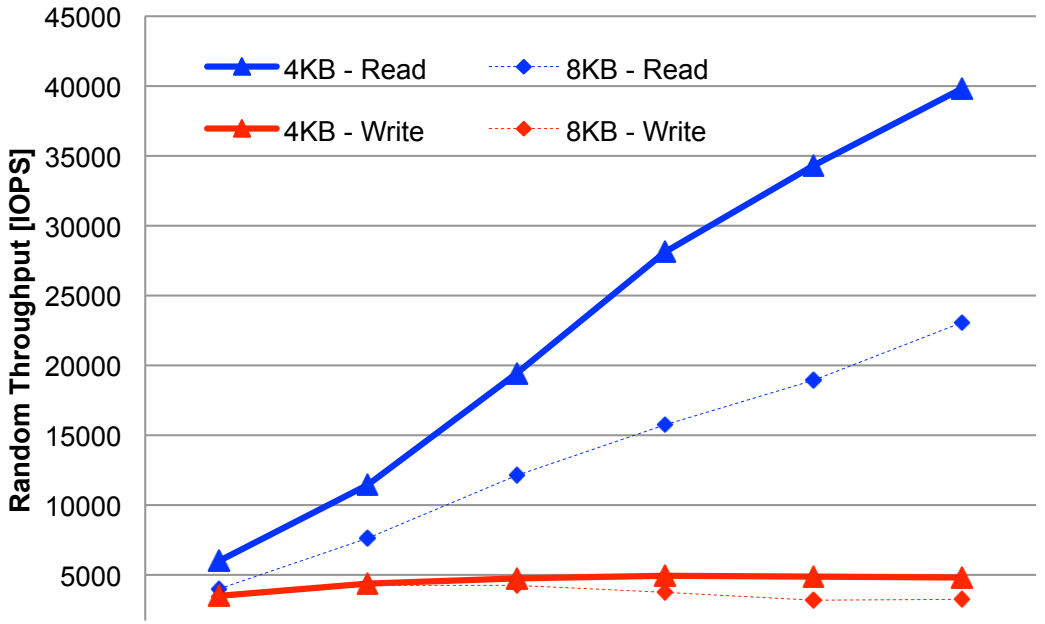
Flash SSD: Parallelism



Trend: Random operations are expensive: avoid!

Random Read ≈ Sequential Read
for high parallelism and typical DBMS block sizes

IOPS, Parallel Random I/Os, X25-E



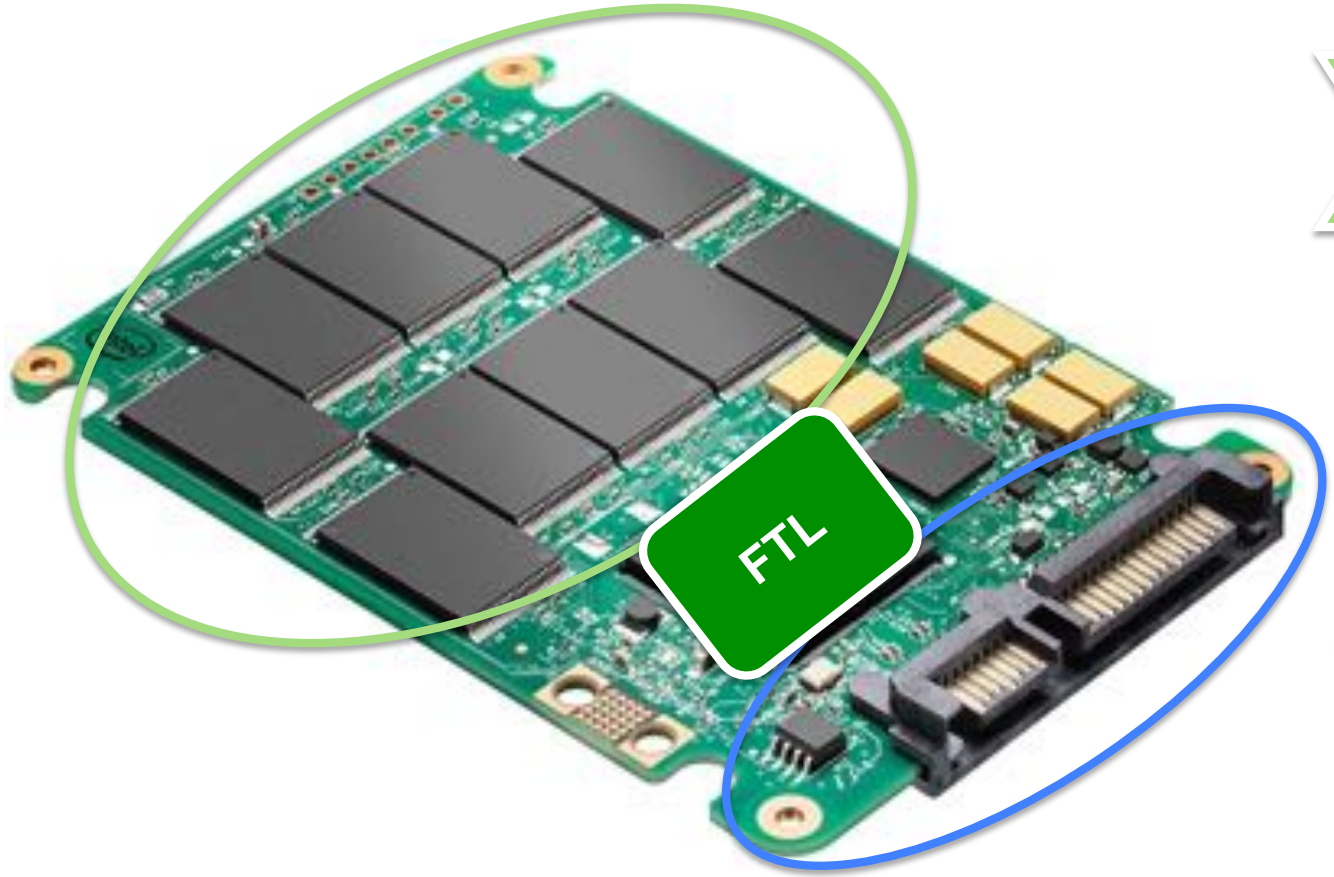
Flash SSDs: Very high parallelism due to the Flash characteristics and internal structure

What about different types of access: read, write?

Access Operations (Read/Write)

- **HDDs**: Read and Write operations from/to logical addr. (LBA)
  - Symmetric: $T_{READ} = T_{WRITE}$
- **Flash memories:** Read, Write and **Erase** operations of **different granularity** from/to a physical addresses
  - Read and Write(Program) – physical pages
  - Erase – physical block (32 to 256 physical pages)
  - Asymmetric: $T_{READ} << T_{WRITE} << T_{ERASE}$
    Erase-before-rewrite, Endurance
- **Flash SSDs:** Read and Write from/to logical addresses (LBA)
  - Asymmetric: $T_{READ} << T_{WRITE}$
    $T_{Sequential\ READ} \approx T_{Random\ READ}$ $T_{Sequential\ WRITE} << T_{Random\ WRITE}$
  - SSD Performance is state-dependent and unpredictable

Flash SSD Operations



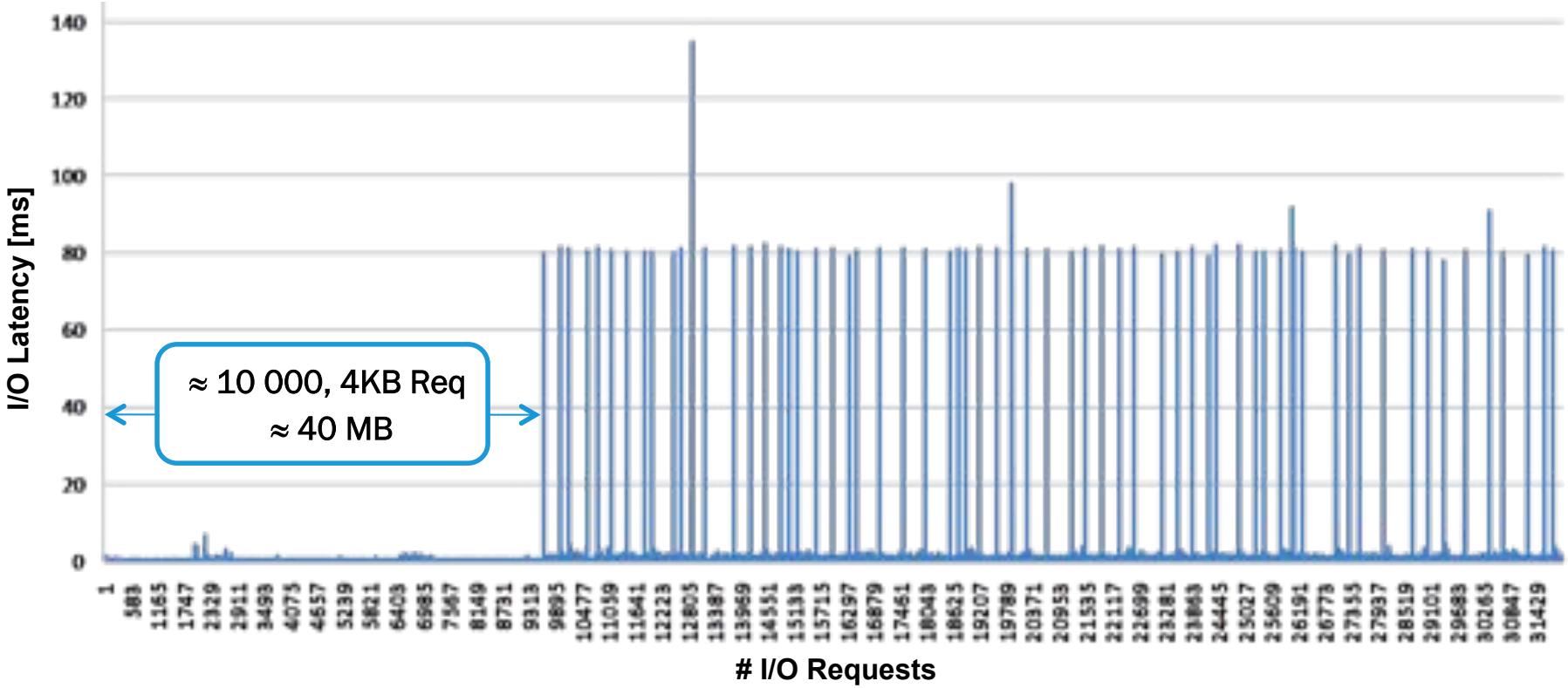
Samsung K9K4G08U1M	
Read	25 us
Write	200 us
Erase	2 ms

Intel X25E SLC SSD	
Rand.Read	167 us
Rand.Write	455 us

SSDs backwards compatible to HDD (Block device interface, FTL):

- Masks flash characteristics
- Performance slowdown and ...

Flash SSD Operations: Cost of FTL and State Dependence



- … Unpredictable, state-dependent performance:
  - Background processes
  - Limited on-device resources
- Redundant functionality - at different layers on the I/O path

Flash SSD Operations: Cost of FTL and State Dependence

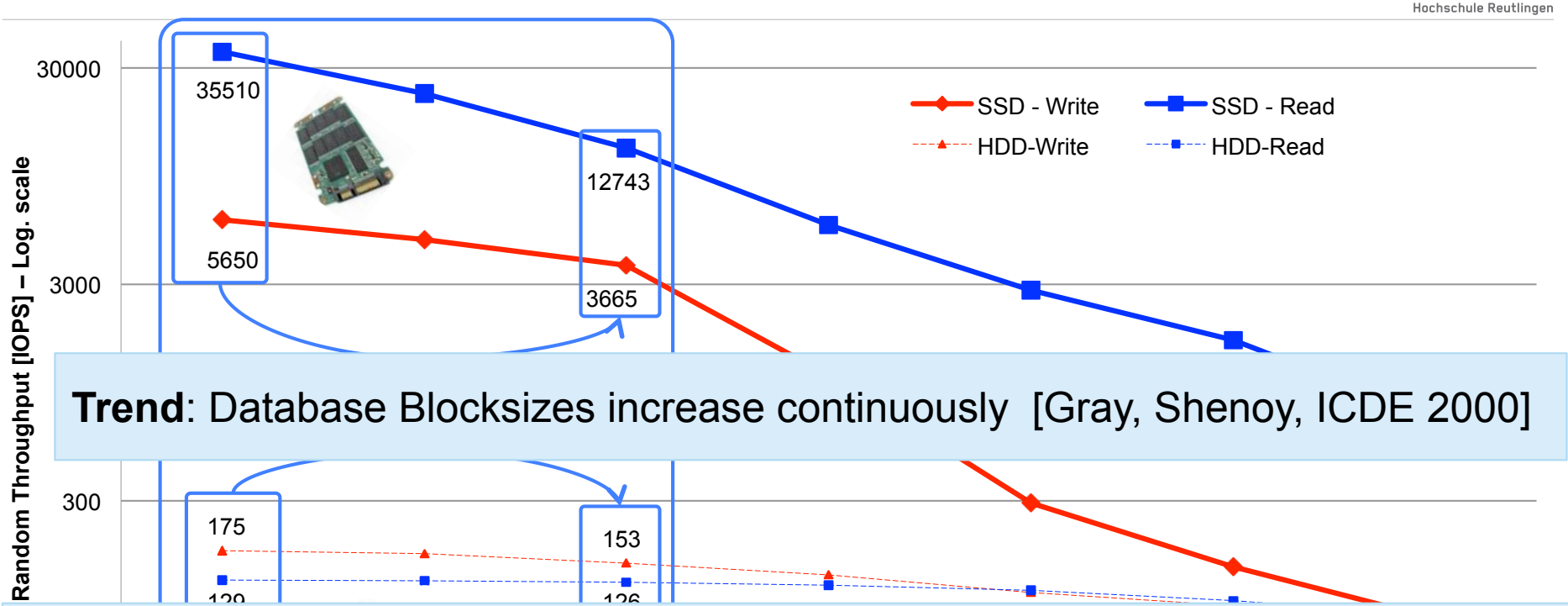
TECHNISCHE UNIVERSITÄT DARMSTADT
INF Informatik
Hochschule Reutlingen

- Excerpt from **Datasheet, Vendor 1**:

Peak Random Performance	Read: 80,000 IOPS Write: 36,000 IOPS	Read: 50,000 IOPS Write: 29,000 IOPS
Sustained Random Performance*	Read: 57,000 IOPS Write: 2,200 IOPS	Read: 50,000 IOPS Write: 1,900 IOPS

- Excerpt from **Datasheet, Vendor 2**:
  - … Typical I/O performance numbers as measured using Iometer with a queue depth of 32 and write cache enabled. Iometer measurements are performed on a **8GB span**. … (Drive size: 512GB)

In Brief …



Trend: Database Blocksizes increase continuously [Gray, Shenoy, ICDE 2000]

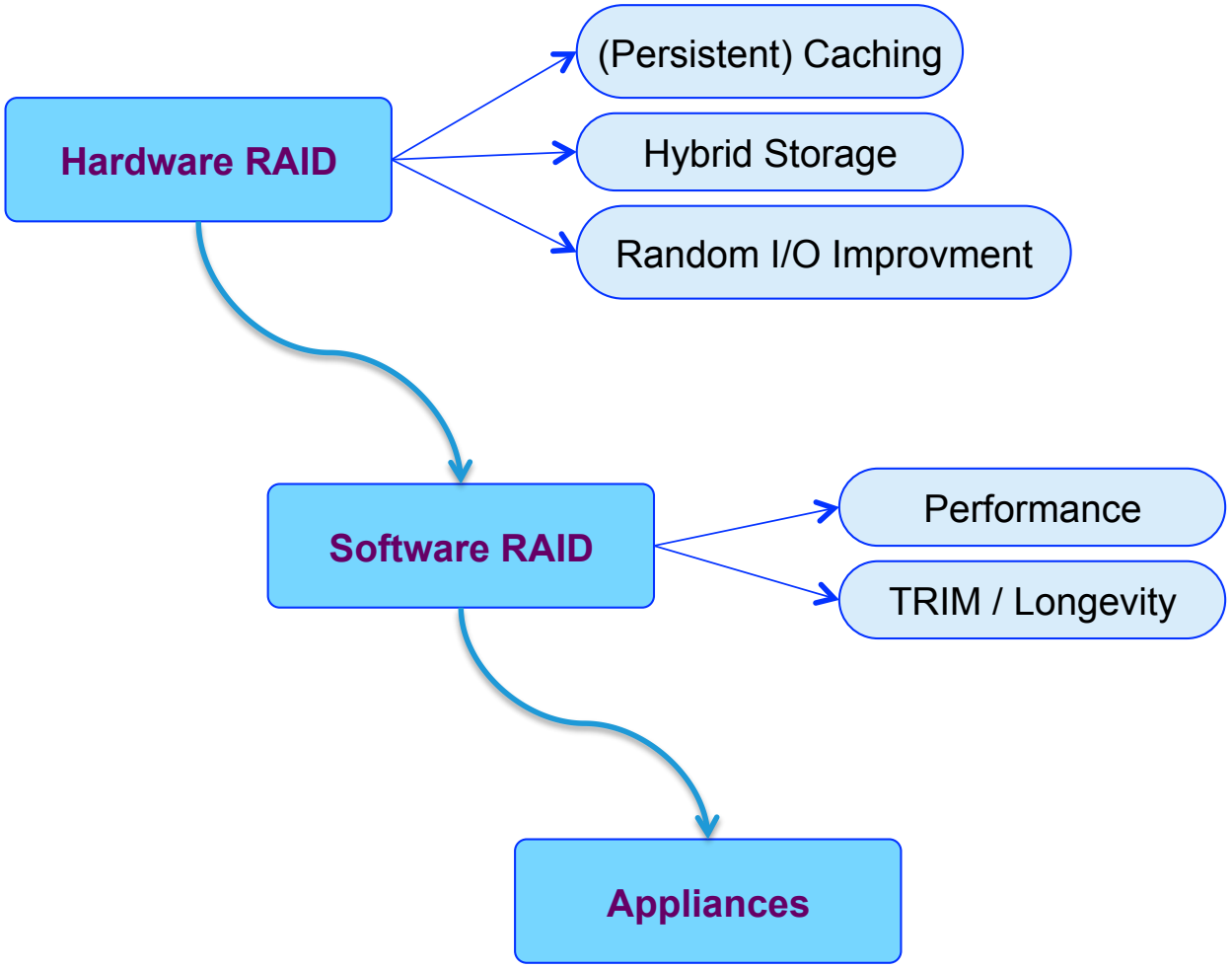
Influence on: Page Size, Page Formats, Storage Manager, …

T. Ivanov, I. Petrov, A. Buchmann. **Page Size Selection for OLTP Databases on SSD Storage**. In Proc. SBBS 2010. Belo Horizonte, Brasil, October, 2010

I. Petrov, R. Gottstein, T. Ivanov, D. Bausch, A. Buchmann. **Page Size Selection for OLTP Databases on SSD RAID Storage**. In Journal of Information and Data Management,Vol.2,No 1. 2011

Building Large Flash Storage

State-of-the-Art:
Fast, Large, Reliable Flash Storage



State-of-the-Art: Fast, Large, Reliable Flash Storage

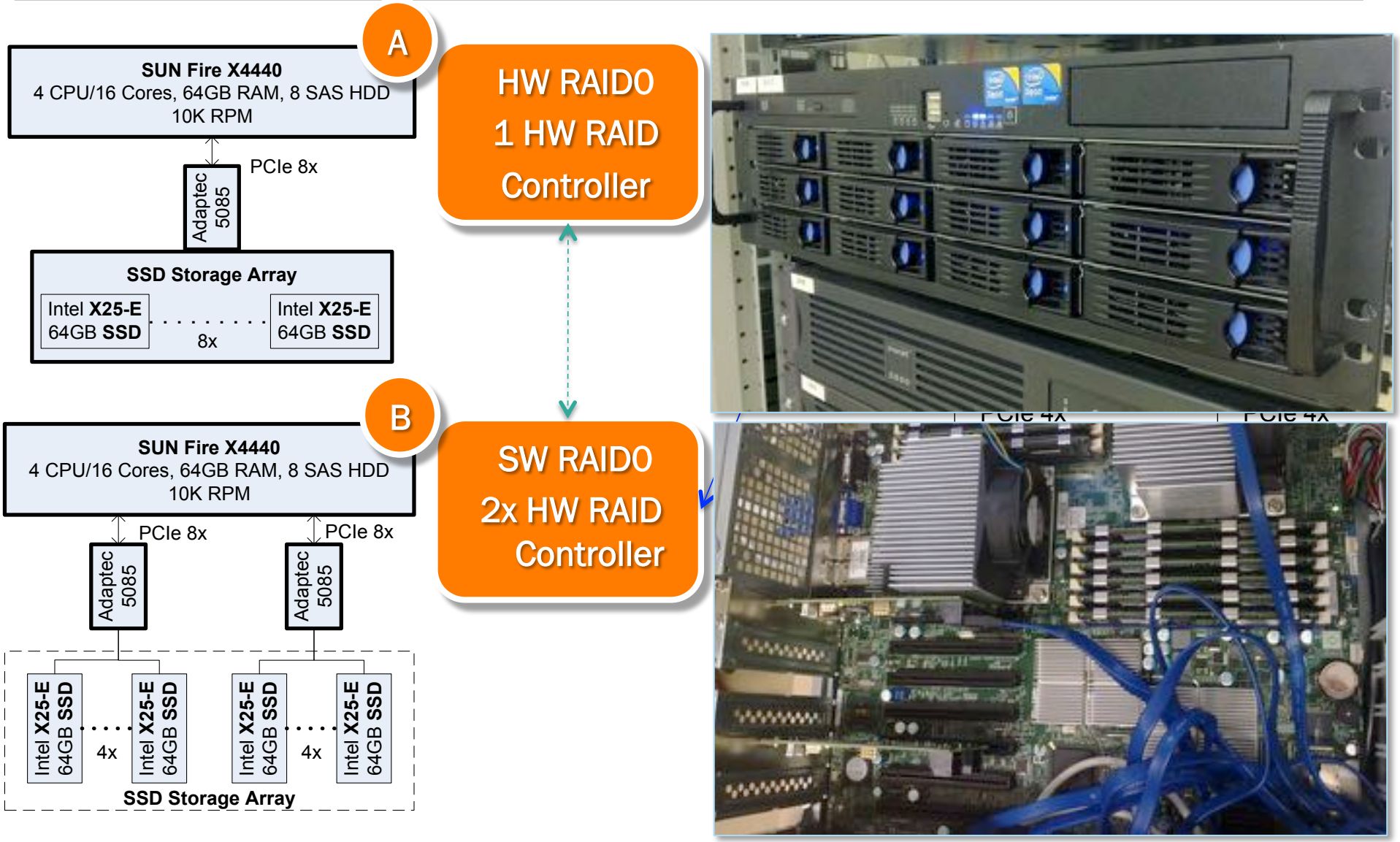
- **Hardware RAID**
  - S. Im, D. Shin, D. Shin. Flash-Aware RAID Techniques for Dependable and High-Performance Flash Memory SSD. IEEE Trans. Comput. 60, 1, 80-92. 2011
  - D. Yimo, L.Fang, C. Zhiguang, M. Xin. WeLe-RAID: a SSD-based RAID for system endurance and performance. In Proc. NPC'11
  - …
- ***Persistent Caching / Hybrid Storage / Improvements***
  - G. Soundararajan, V. Prabhakaran, M. Balakrishnan, T. Wobber. Extending SSD lifetimes with disk-based write caches. In Proc. USENIX FAST'10. 2010
  - R. Koller, L. Marmol, R. Rangaswami, S. Sundararaman, N. Talagala, M. Zhao. Write policies for host-side flash caches. In Proc. USENIX FAST'13. 2013.
  - …

State-of-the-Art: Fast, Large, Reliable Flash Storage

- **Software RAID**
  - J. He, A. Jagatheesan, S. Gupta, J. Bennett, A. Snavely. DASH: a Recipe for a Flash-based Data Intensive Supercomputer. In Proc. SC'10
  - J. He, J. Bennett, A. Snavely. DASH-IO: an empirical study of flash-based IO for HPC. In Proc. TeraGrid Conference 2010
  - N. Jeremic, G. Mühl, A. Busse, J. Richling. The pitfalls of deploying solid-state drive RAIDs. In Proc. SYSTOR '11.
  - N. Jeremic, G. Mühl, A. Busse, J. Richling. Enabling TRIM support in SSD RAIDs. Technical Report Informatik Preprint CS-05-11, Uni. Rostock, 2011. ISSN 0944-5900
  - B. Mao, H. Jiang, S. Wu, L. Tian, D. Feng, J. Chen, L. Zeng. HPDA: A hybrid parity-based disk array for enhanced performance and reliability. Trans. Storage 8(1), 2012
  - A.Kadav, M. Balakrishnan, V. Prabhakaran, D. Malkhi. Differential RAID: rethinking RAID for SSD reliability. SIGOPS Oper. Syst. Rev. 44, 1. 2010.

- **Appliances**
  - Violin Flash Storage Plattform
  - Exadata Smart Flash Cache Features
  - IBM FlashCache Storage Accelerator | IBM Blue Gene Active Storage (BGAS)

Configurations



Hardware/Software RAID

		A: 1 Controller (8 SSDs) RAID0 HW		B: 2 Controllers (4 SSD per Controller) RAID0 Simple Volumes		C: 2 SATA Controllers (4 SSD per Controller) Software RAID	
Quantity	blockSize	Read	Write	Read	Write	Read	Write
Seq.Throug-hput [MB/s]	256KB	672	397	1349	881	2018	1477
	512KB	670	398	1350	891	2027	1487
Rand.Throug-hput [IOPS]	4KB	24 828	10 198	51 142	23 661	279 776	41 882
	8KB	23 040	9 032	46 156	21 372	184 469	30 458

2x

12x/4x

Without HW RAID the aggregated throughput is reached. **C = ∑ (8 SSDs)**

Hardware RAID is significant bottleneck. Use software RAID and host based storage.

I.Petrov, G. Almeida, A. Buchmann, U. Graef. **Building Large Storage Based On Flash Disks**. In Proc. ADMS'10 at VLDB 2010. Singapore, September, 2010

Observations

Trend: New IO technologies most effective in host-based storage[3]. Reevaluate use of Network Storage.

- Evolving interfaces: SATA2 → PCIe → DIMM
  - Consider: PCIe, Memory Channel Storage, ULLtraDIMM, eXFlash-DIMM

eXFlash DIMM
Source: IBM

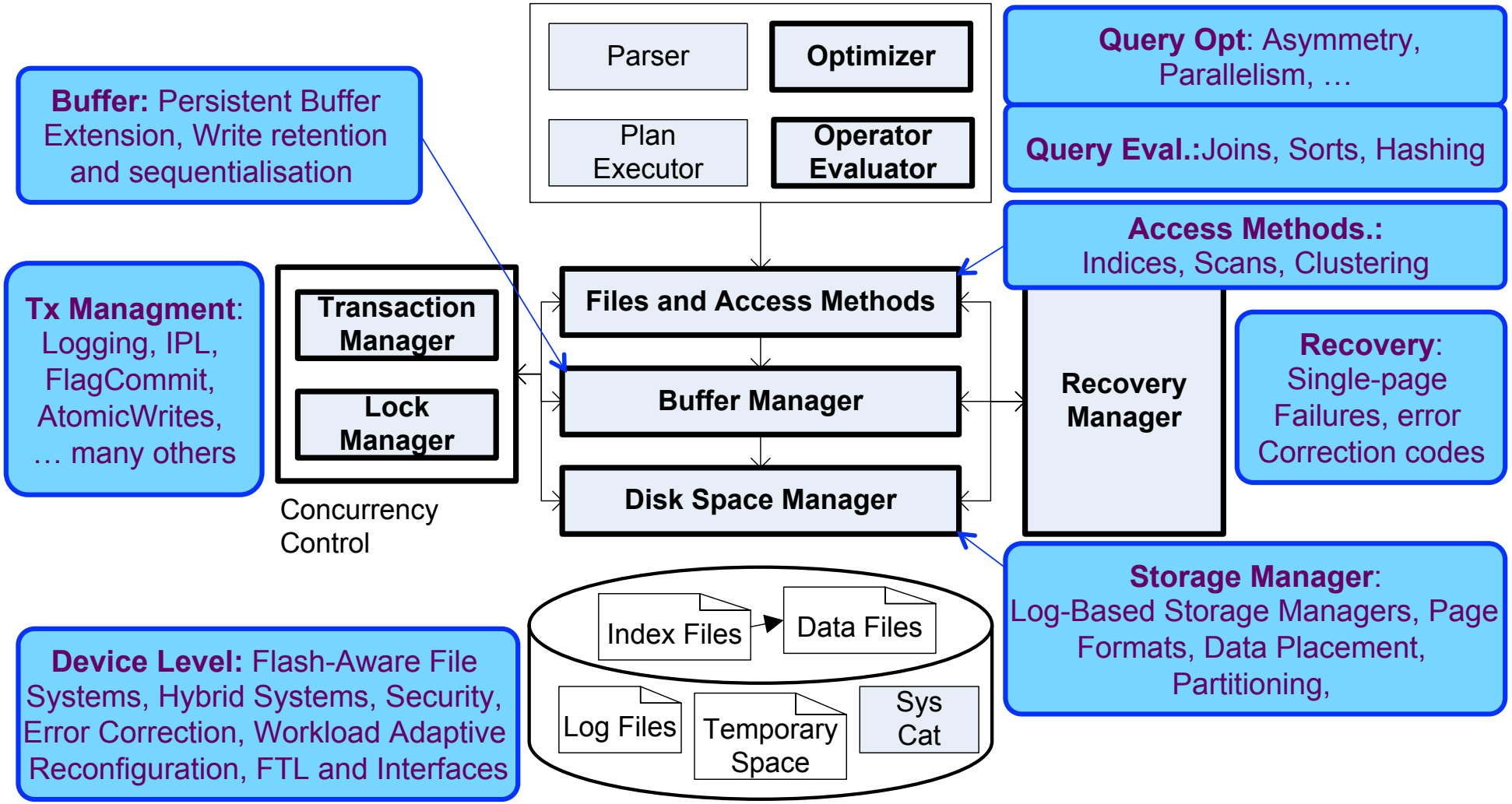
  - Reduce Backwards Compatibility
  - Device Management in software!
  - Better control (wear leveling, garbage collection), Parallelism
  - Adaptability
  - (-) Still: block device interface / FTL + Driver

Consider NoFTL

[3] Eleftheriou et al. Trends in Storage Technologies. IEEE Data Eng. Bull. 33, 2010

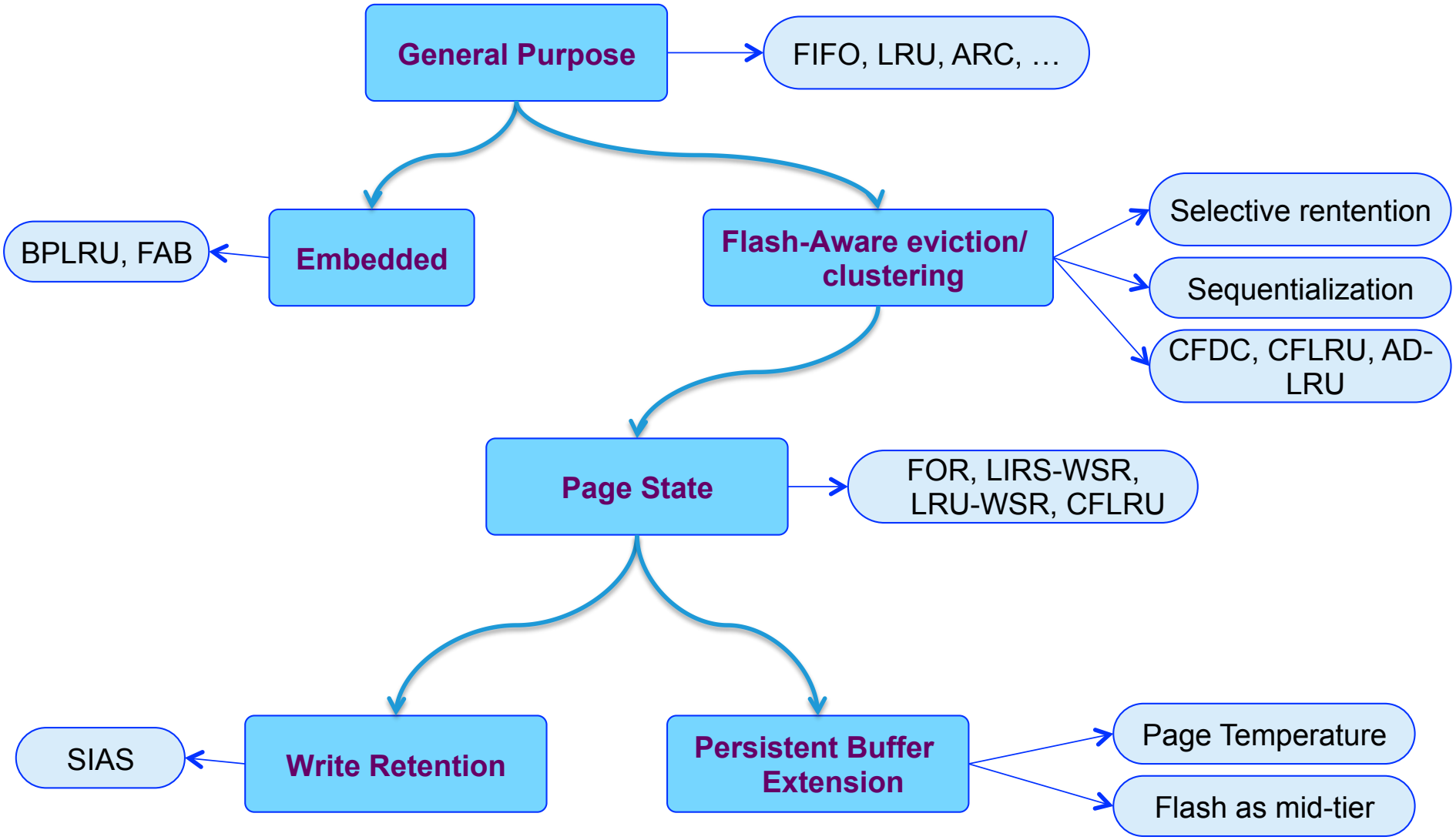
Algorithmic Improvements

Algorithmic Improvements



Flash-Aware Buffer-Management

State-of-the-Art: Flash-aware Buffer Management



State-of-the-Art: Flash-aware Buffer Management

- **General purpose**: FIFO, LRU, ARC
- **Special purpose**:
- ***Embedded***:
  - BPLRU: Kim, Hyojun and Ahn, Seongjun. BPLRU: a buffer management scheme for improving random writes in flash storage. In Proc. FAST'08
  - FAB: Jo, Heeseung and Kang, Jeong-Uk and Park, Seon-Yeong and Kim, Jin-Soo and Lee, Joonwon. FAB: Flash-Aware Buffer Management Policy for Portable Media Players. IEEE TCE, 52. 2006
- ***Clustering***:
  - CFDC:Yi Ou, theo Härder, Theo, Peiquan Jin. CFDC: a flash-aware buffer management algorithm for database systems. ADBIS'10. 2010
  - CFLRU: Seon-yeong Park, Dawoon Jung, and Kang, Jeong-uk and Kim, Jin-soo and Lee, Joonwon. CFLRU: a replacement algorithm for flash memory. CASES '06. 2006
  - AD-LRU: Peiquan Jin, Yi Ou, Theo Härder. AD-LRU: An efficient buffer replacement algorithm for flash-based databases. Data Knowl. Engineering. February, 2012
- ***Page state***:
  - FOR: Lv, Yanfei and Cui, Bin and He, Bingsheng and Chen, Xuexuan. Operation-aware buffer management in flash-based systems. SIGMOD '11. 2011
  - LIRS-WSR: Jung, Hoyoung and Yoon, Kyunghoon and Shim, Hyoki and Park, and Kang, Sooyong and Cha, Jaehyuk. LIRS-WSR: integration of LIRS and writes sequence reordering for flash memory. ICCSA'07. 2007
  - LRU-WSR: Jung, Hoyoung and Shim, Hyoki and Park, Sungmin and Kang, Sooyong and Cha, Jaehyuk. LRU-WSR: integration of LRU and writes sequence reordering for flash memory. IEEE TCE. 2008
  - CFLRU (find above under “Clustering”)

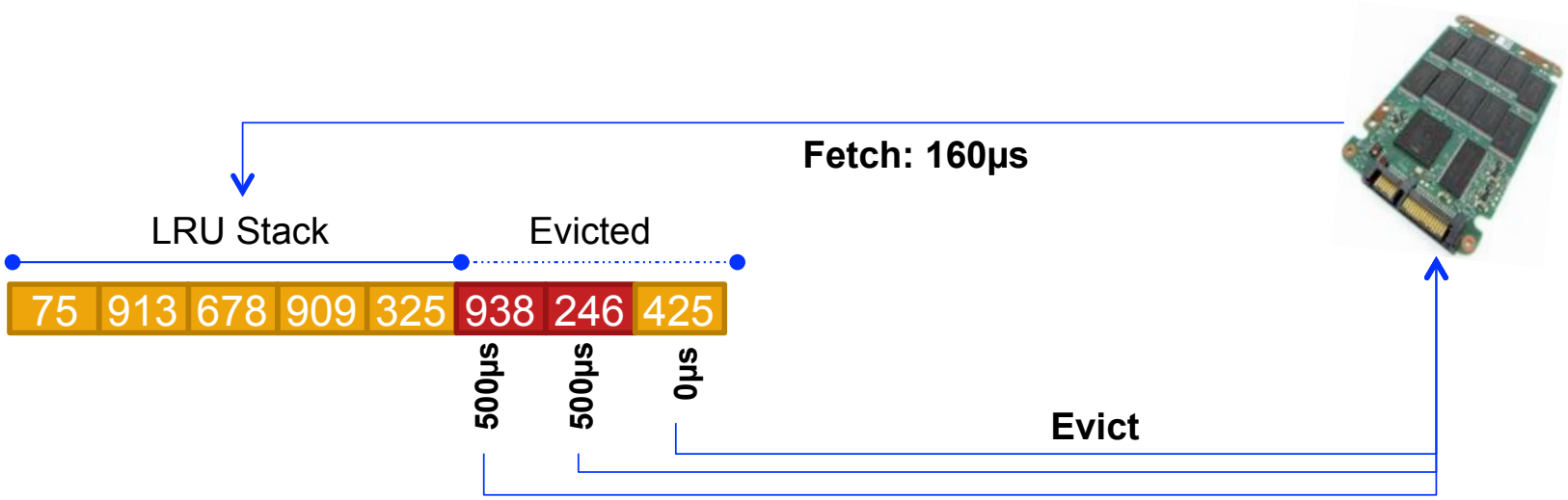
State-of-the-Art: Flash-aware Buffer Management

- **Persistent buffer extensions**:
  - Do, Jaeyoung and Zhang, Donghui and Patel, Jignesh M. and DeWitt, David J. and Naughton, Jeffrey F. and Halverson, Alan. Turbocharging DBMS buffer pool using SSDs. Proc. SIGMOD. 2011,
  - TAC: Mustafa Canim and George A. Mihaila and Bishwaranjan Bhattacharjee and Kenneth A. Ross and Christian A. Lang. SSD Bufferpool Extensions for Database Systems, In PVLDB Vol. 2 (3). 2010
  - Kang, Woon-Hak and Lee, Sang-Won and Moon, Bongki. Flash-based extended cache for higher throughput and faster recovery. Proc. VLDB Endow. 2012

- **Write-Retention**:
  - SIAS

Example: LRU

- **Access Trace**:
  - **R**425, **R**246, **R**938, **W**246, **R**909, **W**938, **R**325, **R**909, **R**678, **R**913, **R**75



Total **Read** cost: 7x160µs = **1120µs**
Total **Write** cost: 2x500µs + 2x160µs = **1320µs**

Eviction costs outweigh fetch costs! (with 2 out of 9 requests!)

FBARC

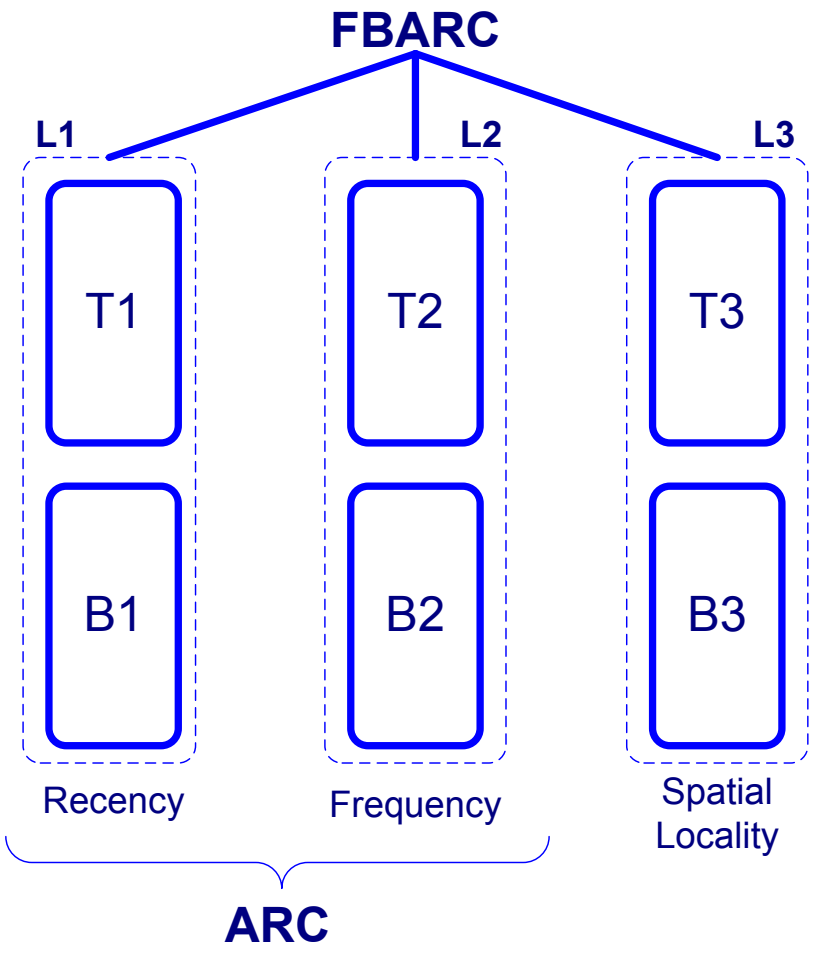
- Design tradeoff:
  i. Trade hitrate and computational intensiveness for
  ii. lower eviction costs to minimize the overall performance penalty
  - In line with present hardware trends

- Asymmetry considered first-class criterion besides hitrate!
  - Spatial locality to address write-aspects of asymmetry
  - Use semi-sequential writes and grid clustering

- We propose FBARC:
  - Based on ARC
  - Write-efficient and endurance-aware
  - High hitrate
  - Computationally efficient – static grid clustering
  - Workload adaptive
  - Scan-resistant

FBARC and ARC

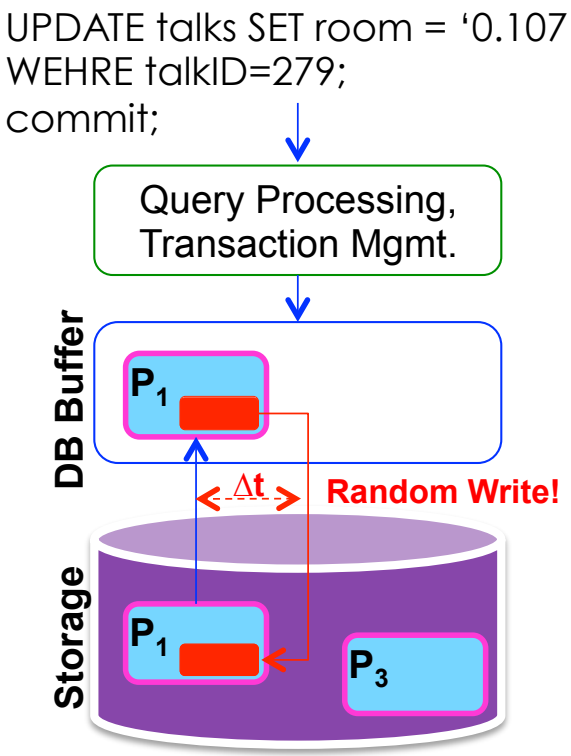
- ARC
  - 2 aspects of temporal locality
  - LRU organized lists
  - Buffered pages held in T-Lists
  - Metadata of evicted pages in B-Lists
- FBARC
  - Adds L3 to support spatial locality
  - T3 organized for clustering
  - B3 still LRU organized

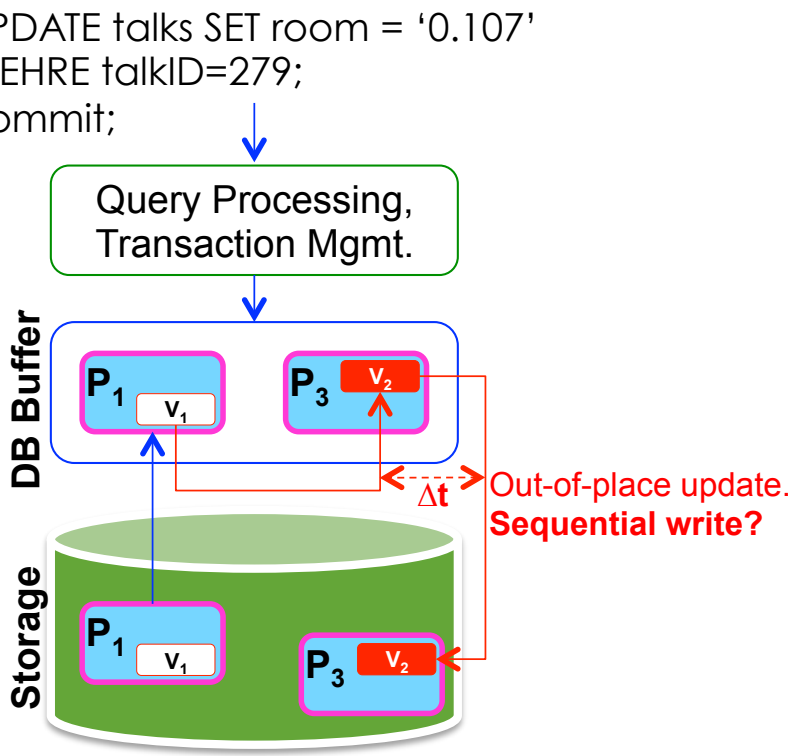


Transaction Management and Multi-Versioning

What can we do? Just one possibility …

- **Parallelism**: perform reads in parallel
- **Reads**: rethink → sequential reads ≈ random reads (for small blocks)
- **Writes**: trade more random reads for sequentialization of random writes



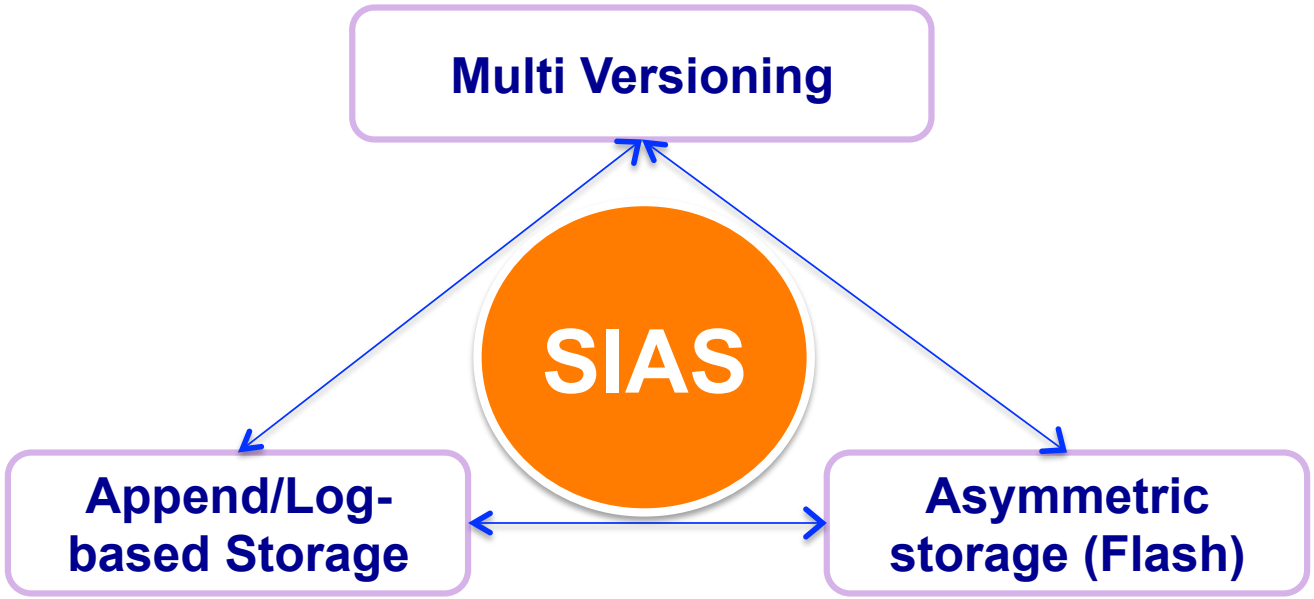


Classical approach. In-place Updates. Random Writes.

Multi-Versioning. Parallelism. Out-of-place Updates. Sequentialization.

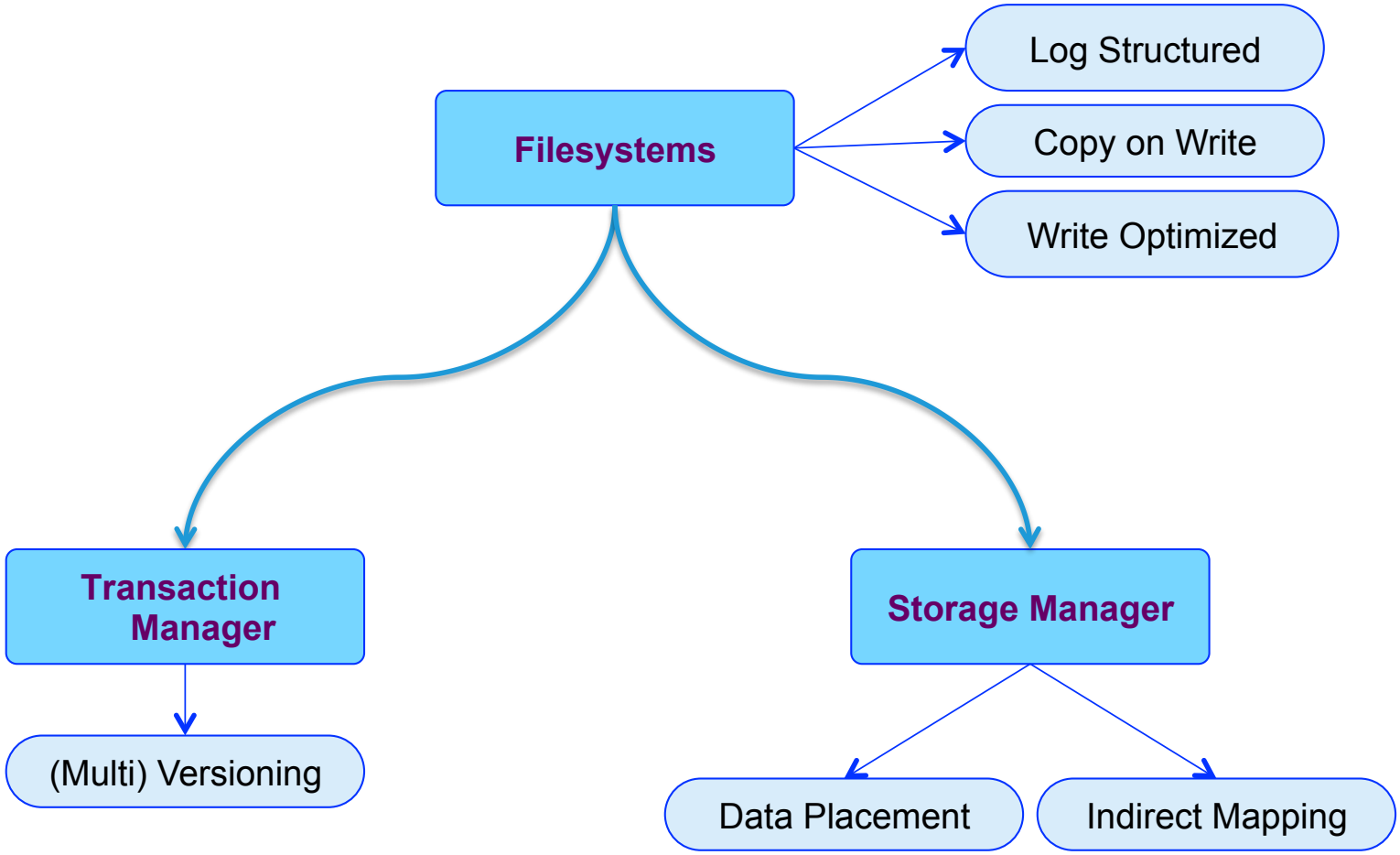
Multi-versioning …

- Multi-versioning is necessary, but NOT sufficient!
  - Helps to sequentialize writes→ Log-based DB Storage Manager needed
  - Multi-versioning improves Read performance→ Good for Flash
  - Log-based Storage Manager→complex reads,fast writes→good for Flash



Append-Based Storage Management

State-of-the-Art: Log Based Flash Storage



State-of-the-Art: Log-Based Storage

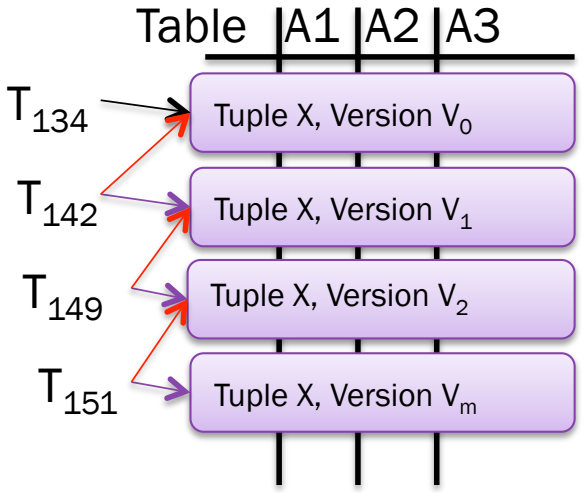
- **File systems**
  - M. Rosenblum. The Design and Implementation of a log-structured file system. U.C. Berkeley, Ph.D. Thesis Jun. 1992
  - Y. Wang, K. Goda, M. Kitsuregawa. Evaluating Non-In-Place Update Techniques for Flash-Based Transaction processing Systems. DEXA 2009.
  - R. Konishi, Y. Amagai, K. Sato et Al. The Linux Implementation of a Log-structured File System. NTT Cyber Space Lab.
  - S. Lee, K. Ha, K. Zang et Al. FlexFS: A Flexible Flash File System for MLC NAND Flash Memory. USENIX 2009.
  - J. C. Mogul, E. Argollo, M. Shah, P. Faraboschi. Operating System Support for NVM+DRAM Hybrid Main Memory. USENIX 2009.
  - W. K. Josephson, L. A. Bongo et Al. DFS: A Filesystem for Virtualized Flash Storage. Princeton EDU 2010.
  - O. Rodeh, J. Bacik, C. Mason. BTRFS: The Linux B-tree Filesystem. ACM 2013

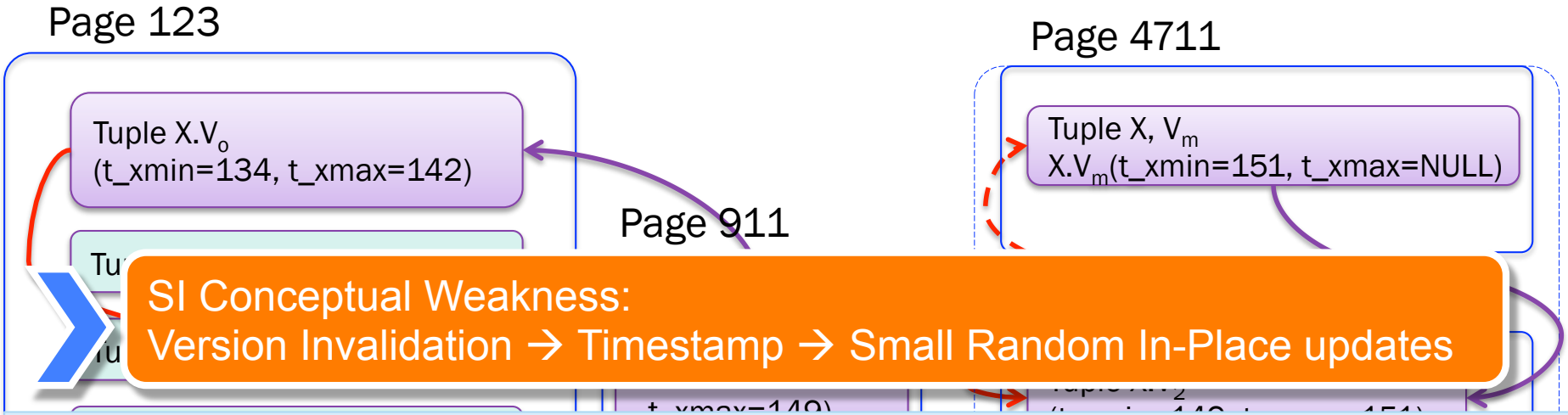
State-of-the-Art: Log-Based Storage

- **Storage Manager**
  - R. Stoica, M. Athanassoulis, R. Johnson and A. Ailamaki. Evaluating and Repairing write performance on Flash Devices. DaMoN 2009.
  - S. Lee, B. Moon. Design of a Flash Based DBMS: An In-Page Logging Approach. SIGMOD 2007.
  - S. Chen. FlashLogging: Exploiting Flash Devices for Synchronous Logging Performance. SIGMOD 2009.
- **Transactional Management**
  - P.A. Bernstein, C.W. Reis and S. Das. Hyder a transactional record manager for shared Flash. CIDR 2011.
  - J. Krueger, C. Kim, M. Grund et al. Fast Updates on read-optimized Databases using Multi-Core CPUs. In Proc. Of VLDB 2011.

Multi-Version Organisation: SI, MVCC

- **Observation:**
  - **Upon creation of a successor version**
  - **the predecessor is physically invalidated**
- Consequence: Random writes, write overhead, inefficient indexing
- Reason: version organisation





R. Gottstein, I. Petrov, A. Buchmann. **SI-CV: Snapshot Isolation With Co-Located Versions.** TPCTC'11 at VLDB 2011

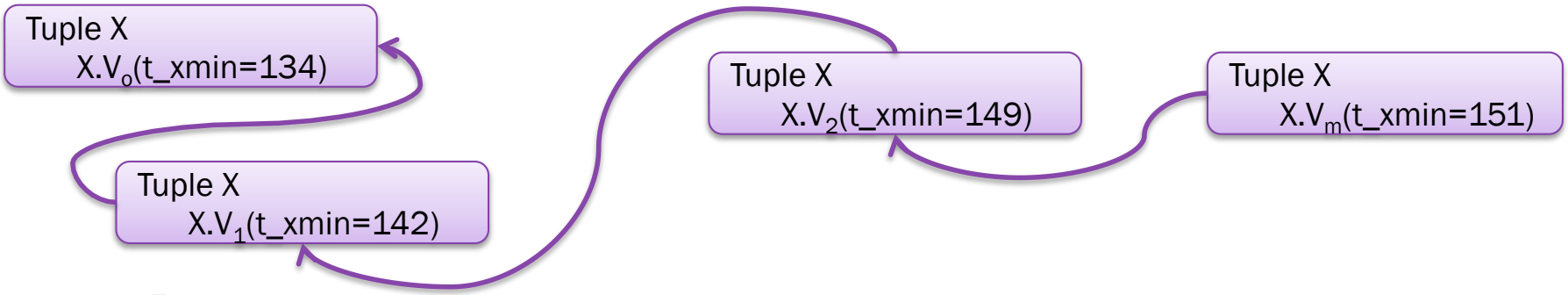
SIAS

- A version **should** be written **exactly once** in its lifetime!
  - Want: Reads are never blocked
  - Want: Append at transactional level

Tuple X
$X.V_2$(t_xmin=149, t_xmax=151)

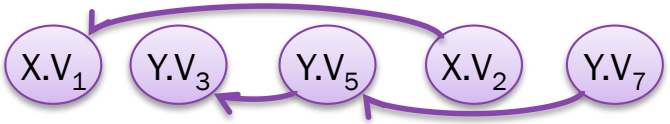
Tuple X
$X.V_m$(t_xmin=151, t_xmax=null)

- SIAS
  - Invalidation model: the existence of a successor invalidates predecessor
  - Backwards chaining of versions → No invalidation Timestamp
  - All tuple versions identified by a virtual ID
  - Combine with an Append Based Storage Manager

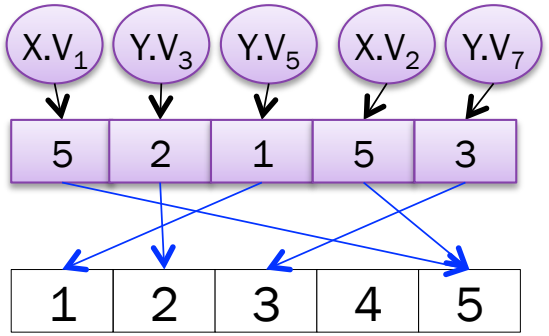


SIAS: Tuple-level Log-Based Storage

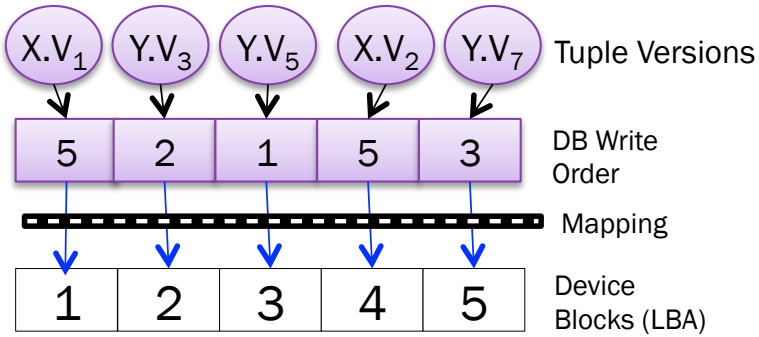
- Versions of Tuples X, Y:



- Storage Managers: in-place vs. page-level LbSM

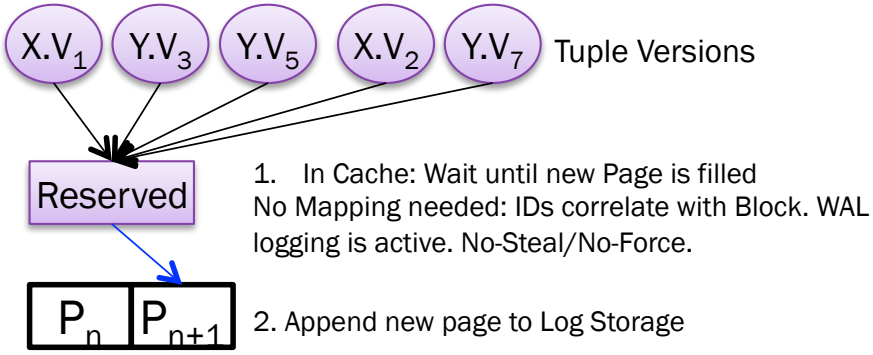


In-place approach

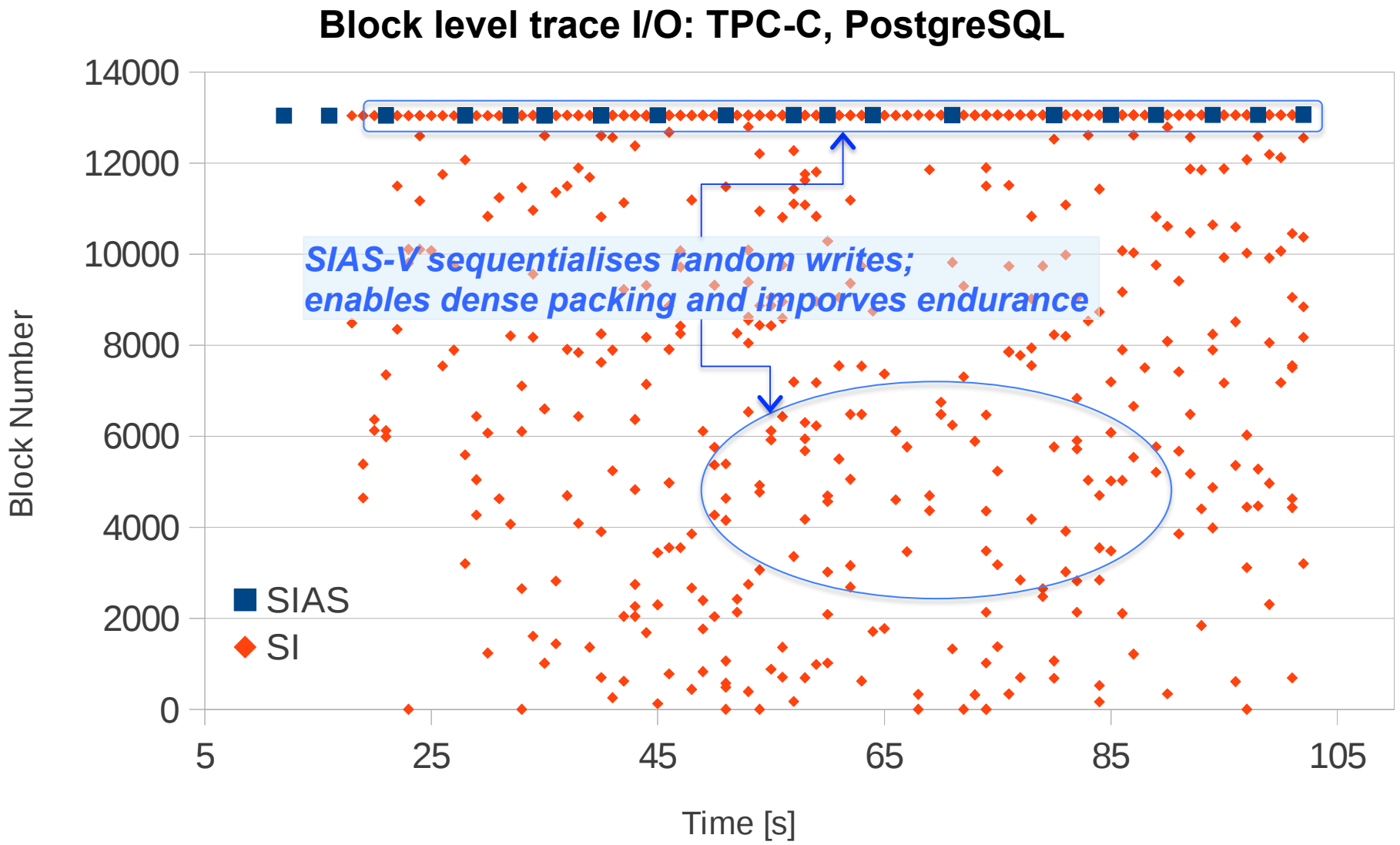


Page Log-Based Storage

- SIAS: Tuple-level LbSM
  - Append new tuple versions to a reserved page.
  - Write out only when page is full.



Sequentialization of Random Writes



Contributions

R. Gottstein, T. Peter, I. Petrov, A. Buchmann. **SIAS-V in Action: Snapshot Isolation Append Storage - Vectors on Flash**. EDBT 2014

R. Gottstein, I. Petrov, A. Buchmann**. Append Storage in Multi-Version Databases on Flash**. In BNCOD 2013, Oxford, UK (2013)

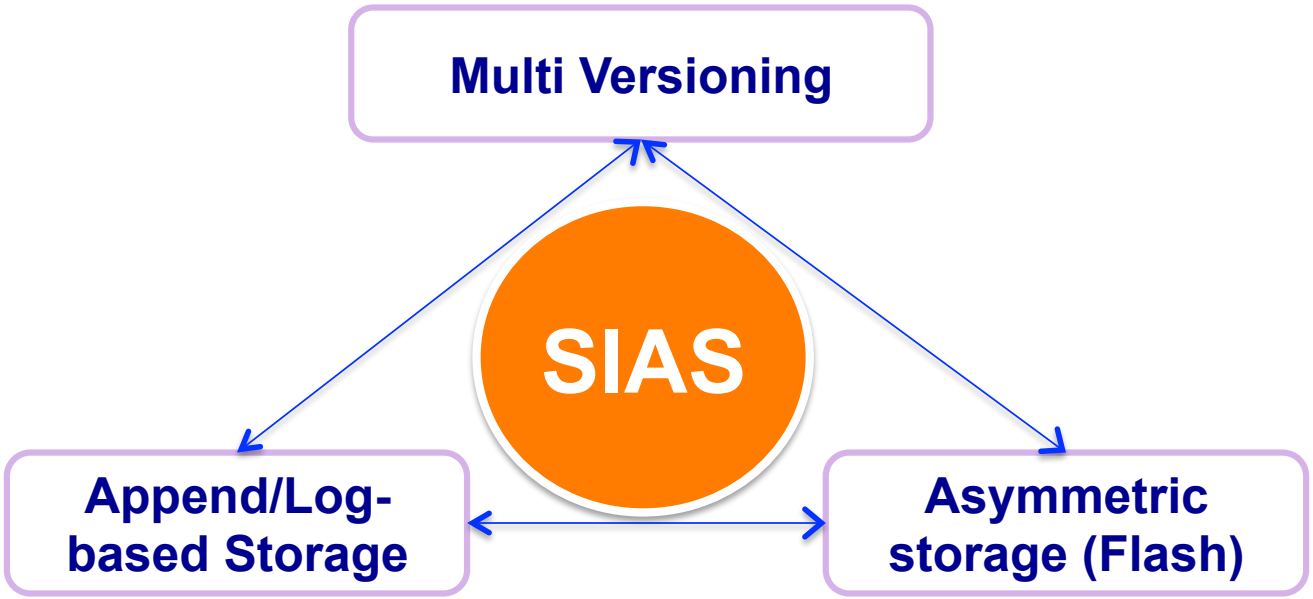
R. Gottstein, I. Petrov, A. Buchmann. **Read Optimisations for Append Storage on Flash**. IDEAS 2013.

R. Gottstein, R. Goyal, S. Hardock, I. Petrov, A. Buchmann. **MV-IDX: Indexing in Multi-Version Databases**. IDEAS 2014

R. Gottstein, R. Goyal, I. Petrov, S. Hardock, A. Buchmann. **MV-IDX: Multi-Version Index in Action**. BTW 2015

R. Gottstein, I. Petrov, A. Buchmann. **Multi-Version Databases on Flash: Append Storage and Access Paths**. IJAS, Vol. 6, Num. 3. 2013.

In brief …



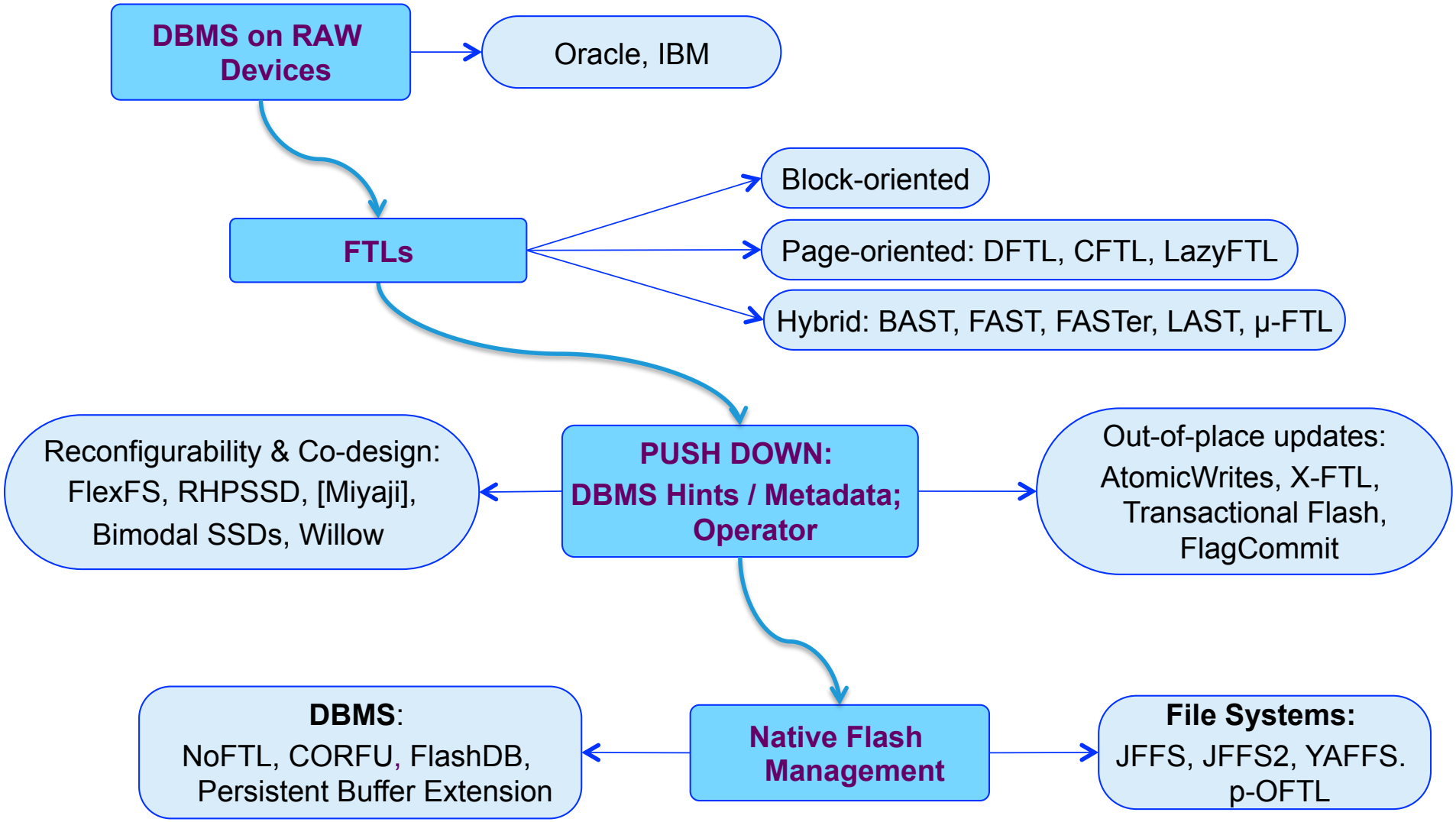
Traditionally	SIAS
In-Place Version Invalidation	Out-Of-Place & New Version Organisation
Page-based LbSM	Tuple-based LbSM Optimized for read AND write access Scales with increasing parallelism
Complex Reads	Utilize Flash Asymmetry

NoFTL: DB on Native Flash Storage

TECHNISCHE UNIVERSITÄT DARMSTADT
INF Informatik
Hochschule Reutlingen

DBlab DVS

State-of-the-Art: Flash Management



State-of-the-Art: Native Flash Management

- **DBMS on RAW Devices**
  - Oracle | IBM | …

- **FTL**
  - Aayush Gupta, Youngjae Kim, and Bhuvan Urgaonkar. DFTL: a flash translation layer employing demand-based selective caching of page-level address mappings. In Proc. of ASPLOS XIV. ACM, New York, NY, USA, 229-240, 2009.
  - Dongchul Park, Biplob Debnath, and David Du. CFTL: a convertible flash translation layer adaptive to data access patterns. In Proc. of SIGMETRICS '10. ACM, New York, NY, USA, 365-366, 2010.
  - Dongzhe Ma, Jianhua Feng, and Guoliang Li. LazyFTL: a page-level flash translation layer optimized for NAND flash memory. In Proc. of SIGMOD '11. ACM, New York, NY, USA, 1-12, 2011.
  - Jesung Kim, Jong Min Kim, S. H. Noh, Sang Lyul Min, and Yookun Cho. A space-efficient flash translation layer for CompactFlash systems. IEEE Trans. on Consum. Electron. 48, 2 (May 2002), 366-375.
  - Sang-Won Lee, Dong-Joo Park, Tae-Sun Chung, Dong-Ho Lee, Sangwon Park, and Ha-Joo Song. A log buffer-based flash translation layer using fully-associative sector translation. ACM Trans. Embed. Comput. Syst. 6, 3, Article 18 (July 2007).
  - Sang-Phil Lim, Sang-Won Lee, and Bongki Moon. FASTer FTL for Enterprise-Class Flash Memory SSDs. In Proc. of SNAPI '10. IEEE Computer Society, Washington, DC, USA, 3-12, 2010.
  - Sungjin Lee, Dongkun Shin, Young-Jin Kim, and Jihong Kim. LAST: locality-aware sector translation for NAND flash memory-based storage systems. SIGOPS Oper. Syst. Rev. 42, 6 (October 2008), 36-42.
  - Yong-Goo Lee, Dawoon Jung, Dongwon Kang, and Jin-Soo Kim. µ-FTL:: a memory-efficient flash translation layer supporting multiple mapping granularities. In Proc. of EMSOFT '08. ACM, New York, NY, USA, 21-30, 2008.

State-of-the-Art: Native Flash Management

- **Push Down**
  - **Out-of-place updates**
    - Xiangyong Ouyang, David Nellans, Robert Wipfel, David Flynn, and Dhabaleswar K. Panda. Beyond block I/O: Rethinking traditional storage primitives. In Proc. of HPCA '11, 2011.
    - Woon-Hak Kang, Sang-Won Lee, Bongki Moon, Gi-Hwan Oh, and Changwoo Min. X-FTL: transactional FTL for SQLite databases. In Proc. of SIGMOD '13, 2013.
    - Vijayan Prabhakaran, Thomas L. Rodeheffer, and Lidong Zhou. Transactional flash. In Proc. of OSDI'08, 2008.
    - On, Sai Tung, Xu, Jianliang, Choi, Byron, Hu, Haibo and He, Bingsheng. Flag Commit: Supporting Efficient Transaction Recovery in Flash-Based DBMSs. IEEE Trans. Knowl. Data Eng. 24 , no. 9 (2012): 1624-1639.
  - **Reconfigurability & Co-design**
    - Sungjin Lee, Keonsoo Ha, Kangwon Zhang, Jihong Kim, and Junghwan Kim. FlexFS: a flexible flash file system for MLC NAND flash memory. In Proc. of USENIX'09, 2009.
    - Ji-Yong Shin, Zeng-Lin Xia, Ning-Yi Xu, Rui Gao, Xiong-Fei Cai, Seungryoul Maeng, and Feng-Hsiung Hsu. FTL design exploration in reconfigurable high-performance SSD for server applications. In Proc. of ICS '09. 2009.
    - Kousuke Miyaji, Chao Sun, Ayumi Soga and Ken Takeuchi. Co-design of application software and NAND flash memory in solid-state drive for relational database storage system . Japanese Journal of Applied Physics ,Volume 53, Number 4S, 2014.
    - Philippe Bonnet, Luc Bouganim. Flash Device Support for Database Management. CIDR 2011: 1-8
    - Sudharsan Seshadri, Mark Gahagan, Sundaram Bhaskaran, Trevor Bunker, Arup De, Yanqin Jin, Yang Liu, and Steven Swanson. Willow: a user-programmable SSD. In Proc. of OSDI'14, 2014.

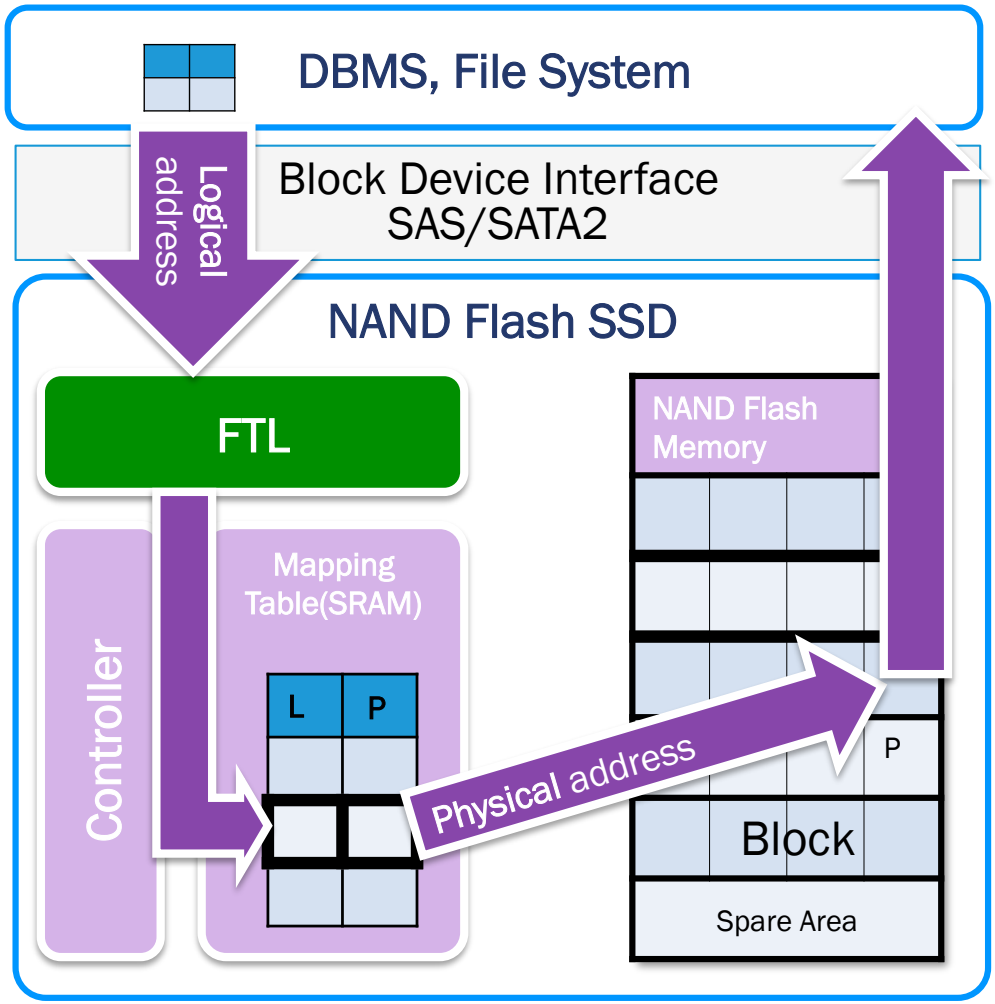
State-of-the-Art: Native Flash Management

- **Native Flash Management**
  - **DBMS**
    - Sergej Hardock, Ilia Petrov, Robert Gottstein, and Alejandro Buchmann. NoFTL: database systems on FTL-less flash storage. Proc. VLDB Endow. 6, 12 (August 2013), 1278-1281, 2013.
    - Mahesh Balakrishnan, Dahlia Malkhi, Vijayan Prabhakaran, Ted Wobber, Michael Wei, and John D. Davis. CORFU: a shared log design for flash clusters. In Proc. of NSDI'12. 2012.
    - Suman Nath and Aman Kansal. FlashDB: dynamic self-tuning database for NAND flash. In Proc. of IPSN '07. ACM, New York, NY, USA, 410-419, 2007.
    - Yi Ou, Theo Härder, and Peiquan Jin. CFDC: a flash-aware replacement policy for database buffer management. In Proc. of DaMoN '09. 2009.
  - **File Systems**
    - https://sourceware.org/jffs2/jffs2-html/
    - http://www.yaffs.net/
    - Wei Wang, Youyou Lu, and Jiwu Shu. 2014. p-OFTL: an object-based semantic-aware parallel flash translation layer. In Proc. of DATE '14. 2014.

Interfaces – Block Device Compatibility

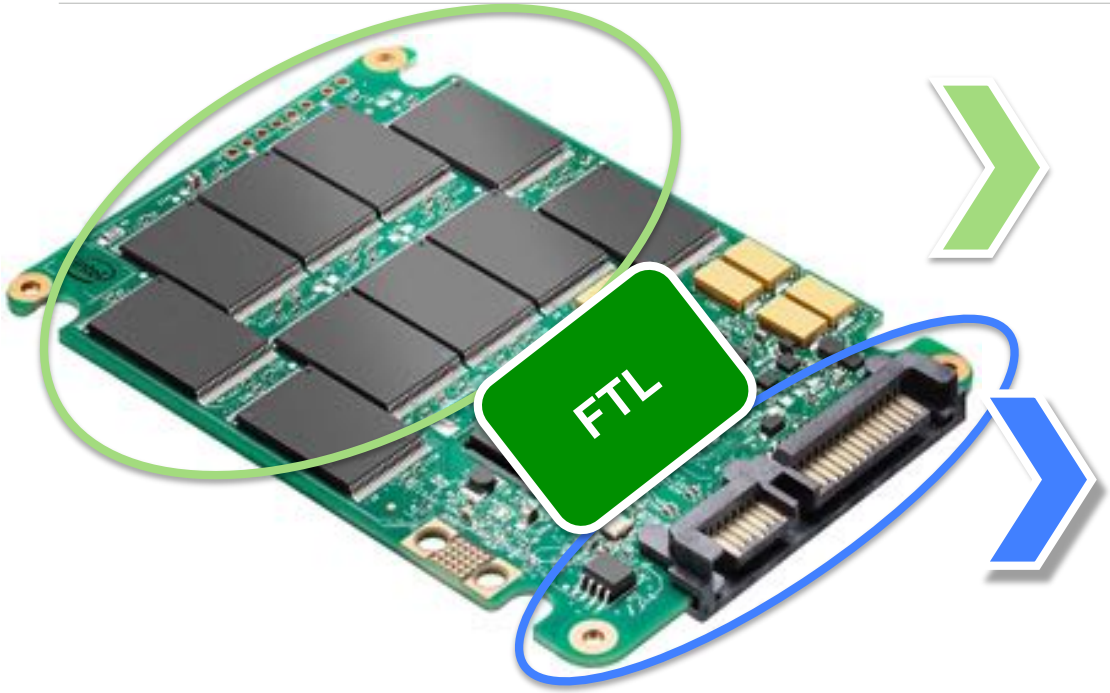
SSDs are backwards compatible to HDD. **Efficient?**
NAND Flash = SSD. **Or does it?**

FTL – Flash Translation Layer



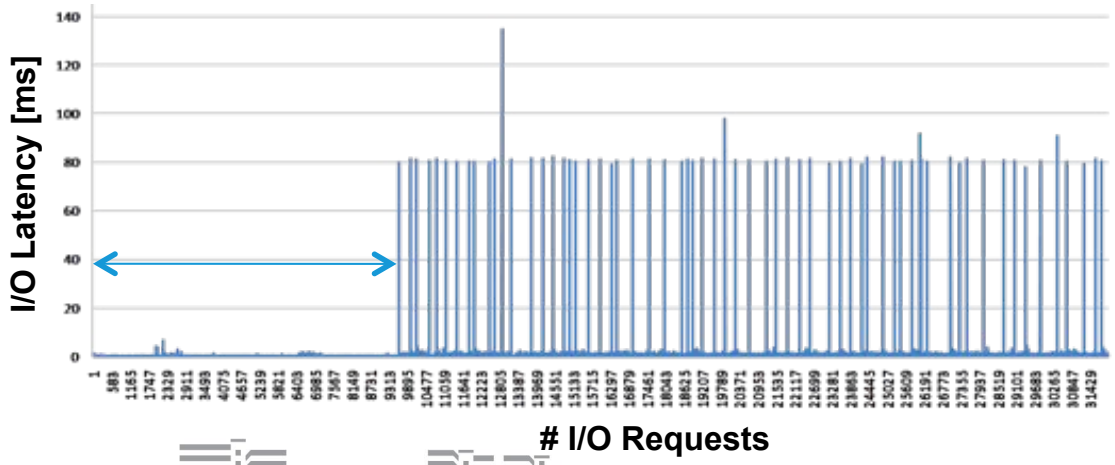
- On-device layer that ensures interface compatibility
- Erase-before-rewrite principle → address translation (mapping)
  - Block-level mapping
  - Page-level mapping (DFTL, CFTL, LazyFTL)
  - Hybrid mapping (e.g., BAST, FAST, FASTer)
    → Trade-off between space consumption and efficiency
- Background Processes:
  - Garbage Collection
  - Wear-leveling
  - Bad Block Management
  - Error Correction

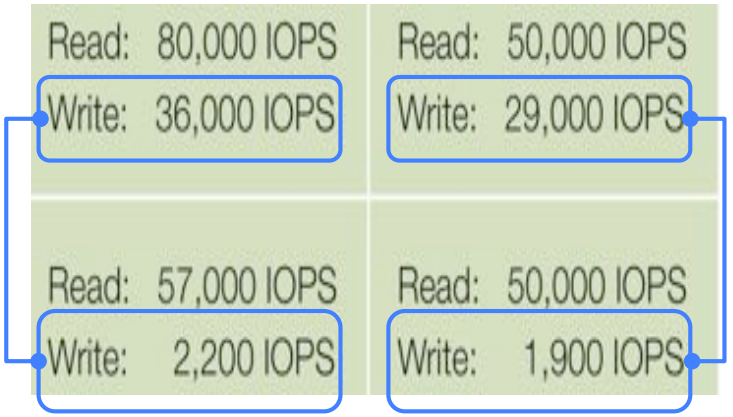
Remember:
Slowdown – Block Device Compatibility



Samsung K9K4G08U1M	
Read	25 us
Write	200 us
Erase	2 ms

Intel X25E SLC SSD	
Read	167 us
Write	455 us



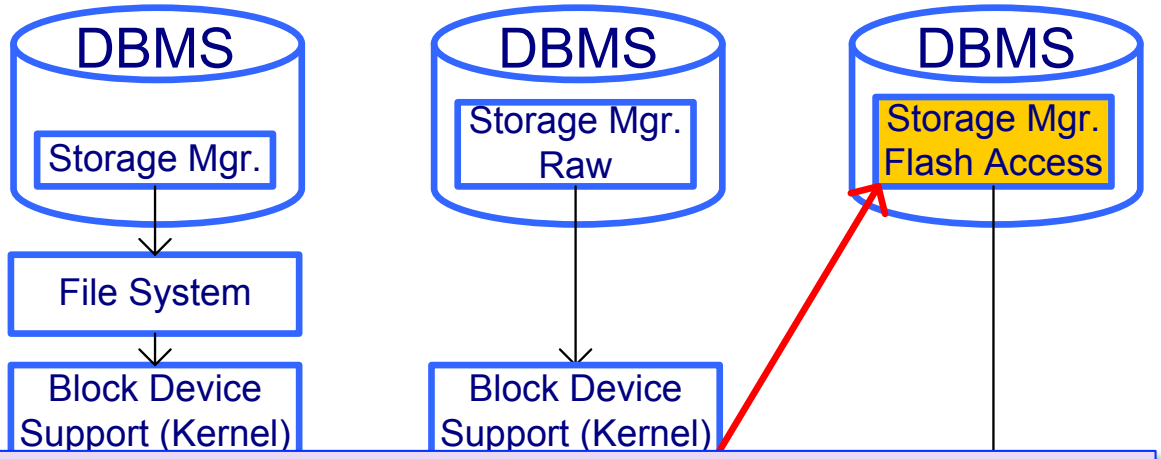


NoFTL: FTL-less Flash storage for DB

- Continuation of the long history of simplifying the I/O stack: DBMS on RAW storage

- DBMS operates directly on RAW NAND

- DBMS has full control over the NAND storage

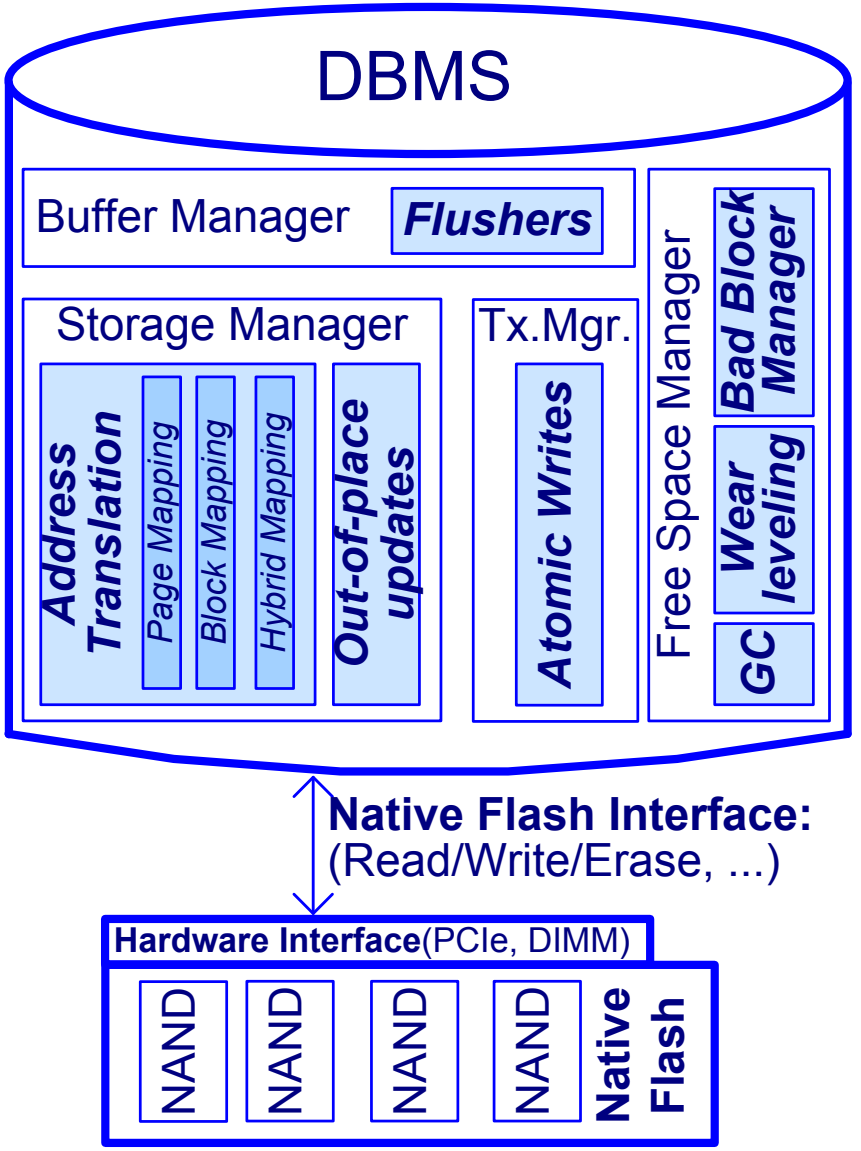


S. Hardock, I. Petrov, R. Gottstein, A. Buchmann. **NoFTL: Database Systems on FTL-less Flash Storage**. In VLDB 2013, Riva Garda (2013)

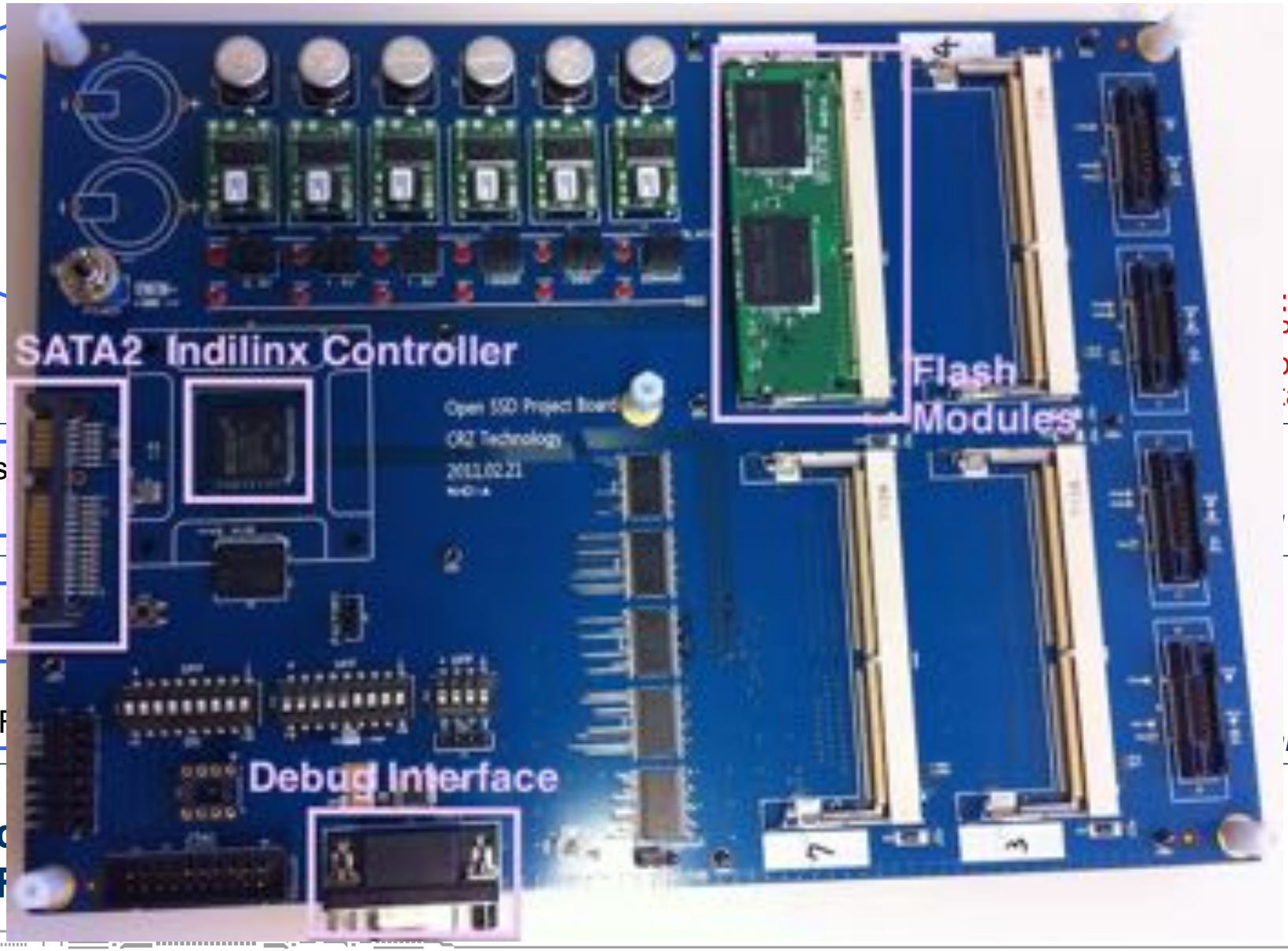
S. Hardock, I. Petrov, R. Gottstein, A. Buchmann. **NoFTL for Real: Databases on Real Native Flash Storage**. In Proc. EDBT 2015.

NoFTL: Integration of Flash Management in DBMS

- **Observation**:
  Flash management can be coherently integrated into DBMS
- DBMS Storage Manager
  - Flash Address Translation
  - Out-of-place updates → wear leveling
- DBMS Free Space Manger
  - Flash Garbage Collection
  - Flash wear leveling and bad block mgt.
- DBMS Buffer Manager
  - Flash Writers | Persistent Buffer ext.
- Further exciting possibilities!



NoFTL: Architecture

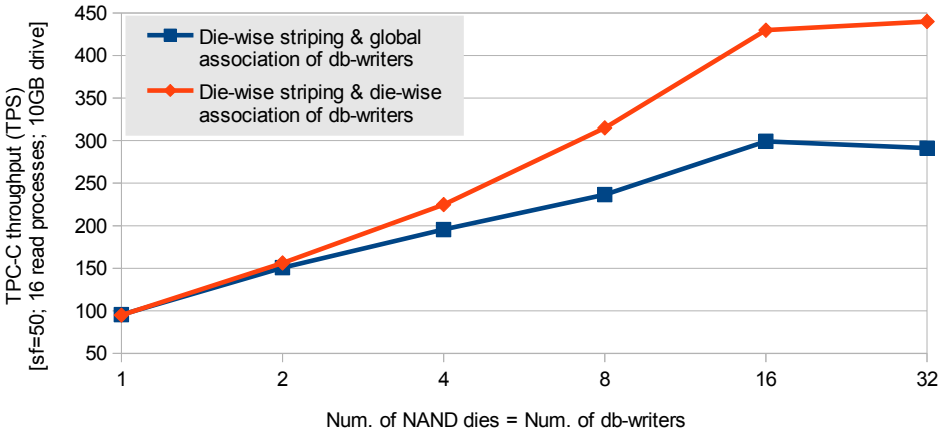


NoFTL: Experimental Evaluation

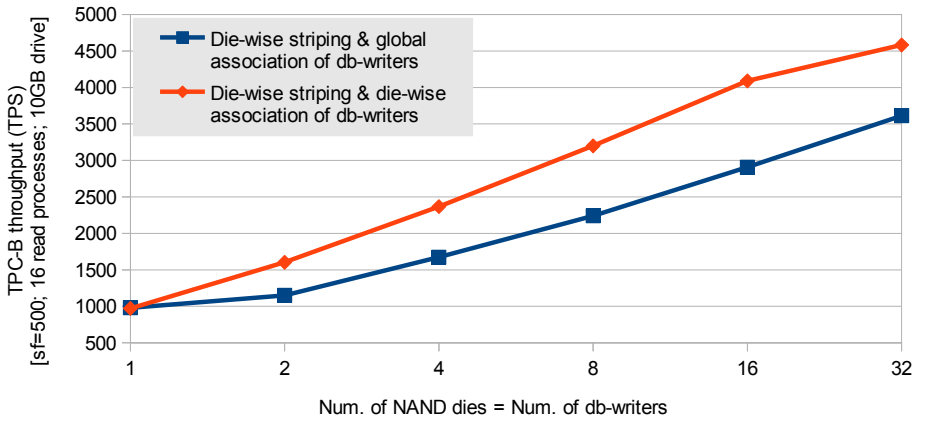
- Less FTL-specific I/Os
- Better utilization of I/O parallelism

- **TPC-C**: 2.4x + 1.5x
- **TPC-B**: 2.25x + 1.43x
- ~2x less GC I/Os
- ~1.7x less erases

	TPC-C		TPC-B	
	FASTer	NoFTL	FASTer	NoFTL
TPS	4.4	10.7	42.9	96.3
Trxs Attendet	16,088	38,758	154,343	346,793
READ 4KB	651,193	1,518,018	396,433	888,668
WRITE 4KB	395,134	555,596	408,320	591,273
COPYBACK	1,395,142	1,304,691	1,341,274	1,346,057
ERASE	14,059	14,533	13,736	15,135



IO type	TPC-C SF=30		TPC-B SF=350		TPC-E 1K Customers	
	Absolute	Relative	Absolute	Relative	Absolute	Relative
COPYBACK	16 465 930	1.98x	17 295 713	2.15x	1 805 540	1.97x
ERASE	129 317	1.73x	135 839	1.82x	14 231	1.68x
Off-line trace-driven testing. Traces were recorded on in-memory database running the benchmarks for 60 minutes.						

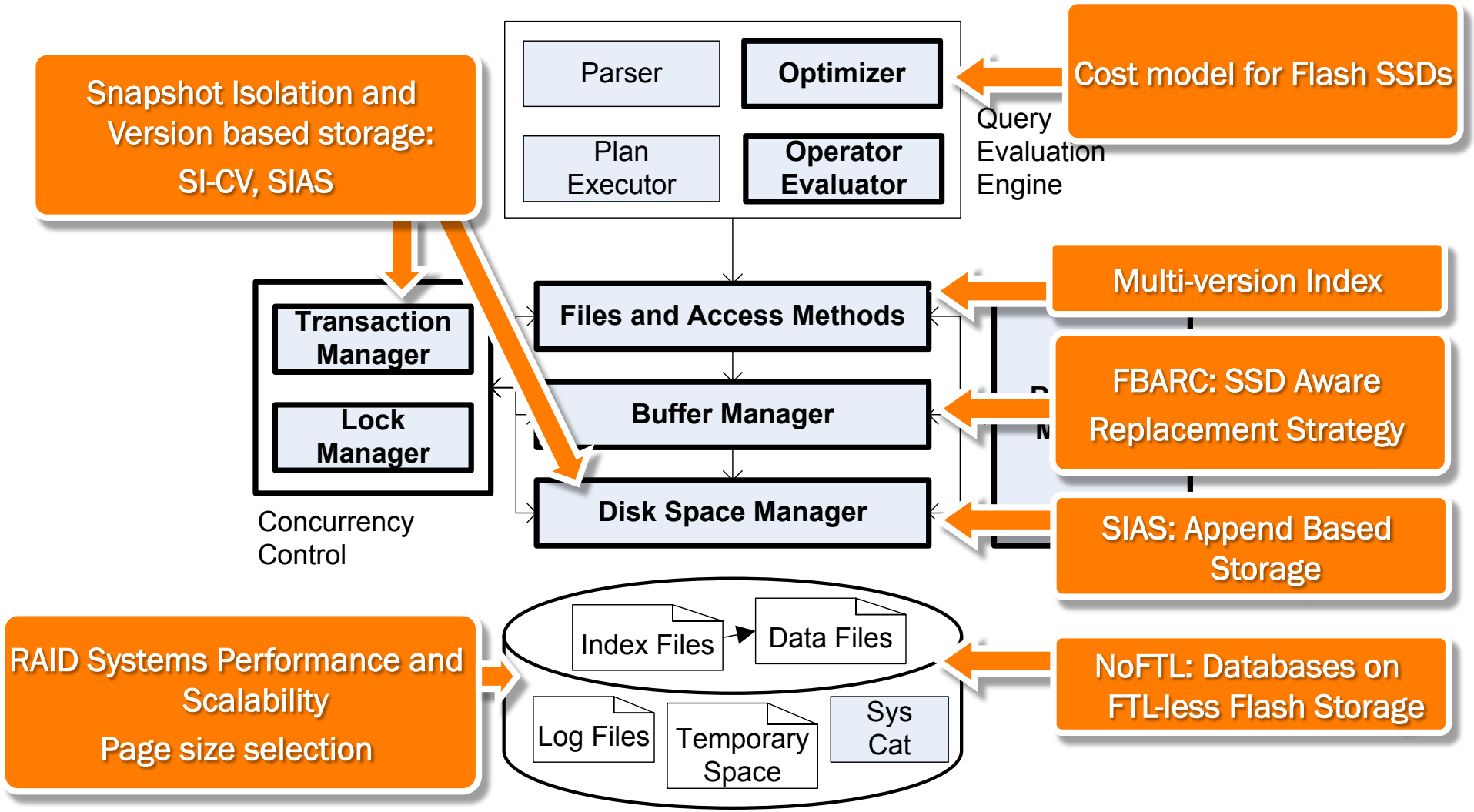


In brief …

- DBMS operates directly on native Flash
  - Continuation of ‘DB on Raw’ tradition
  - Integrate functionality in DBMS
  - Utilize DB algorithms, information and statistics
  - Utilize DBMS resources
- Performance, Predictability and Robustness, Direct Control
- DBMS algorithms and architecture
  - leaner I/O stack, less redundancies

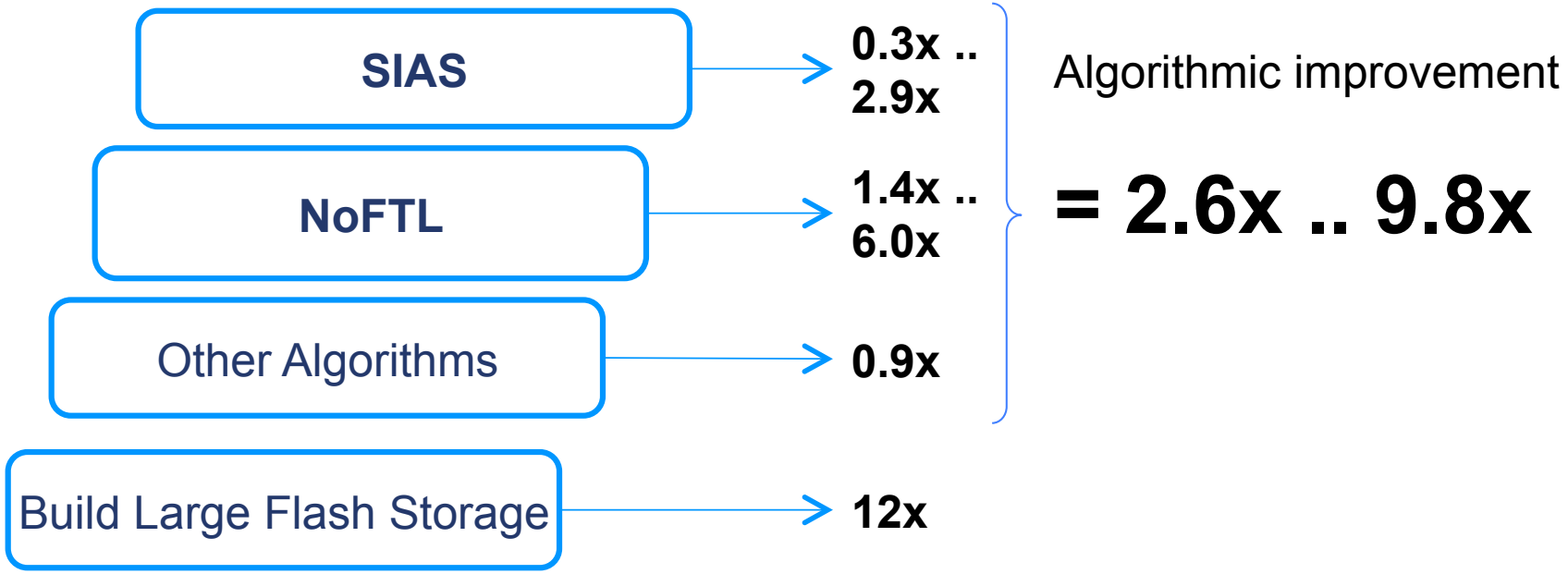
Summary

DFG FlashyDB: Research Impact



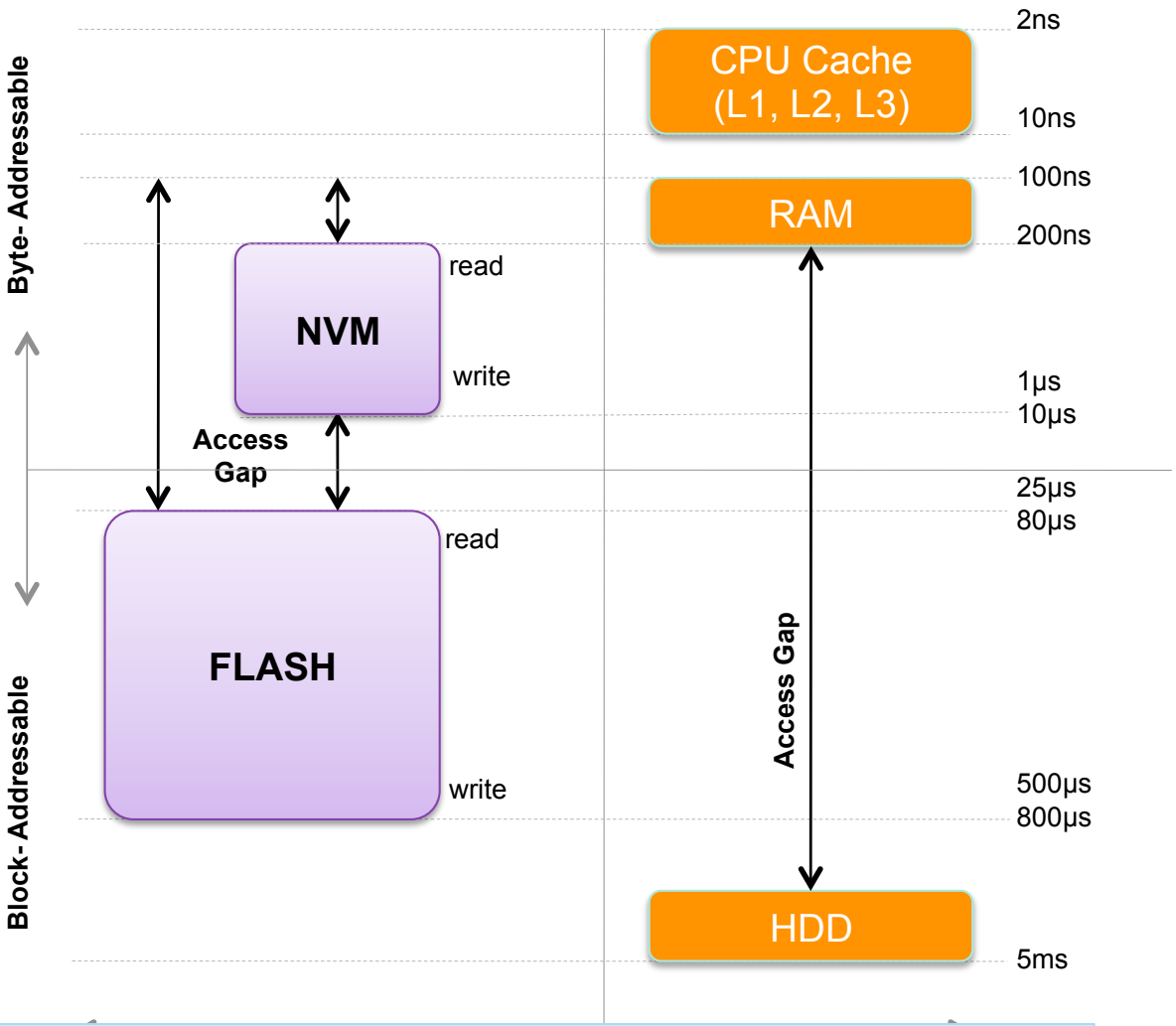
Summary

If Flash SSDs were treated as HDD replacements: a 10x..1000x faster SSD accelerates DBMS ~2.2x..2.5x! [Amdahl]. **Not Efficient!**



Outlook: Memory Hierarchy

- Memory Hierarchy changes
- Latency Continuum
- Characteristics
  - Asymmetry
  - Endurance
  - Parallelism
- ‘access gap’ shrinks
  - Operation-specific
- Data Placement gains importance



I.Petrov, D. Bausch, R. Gottstein, A. Buchmann., **Data-Intensive Systems on Evolving Memory Hierarchies**. In Proc. EEbS 2012, 42. GI Tagung, 2012

Outlook: What does the future bring?

TECHNISCHE UNIVERSITÄT DARMSTADT
INF Informatik
Hochschule Reutlingen

Large Main Memories
128 TB by 2022

Computing Power
1000 Core/CPU by 2022

Bandwidth
Memory: 2.5 TB/s
IO: 250 GB/s

Hardware Trends
[A. von Bechtolsheim]

Fast Persistent Storage
1TB Flash Chips by 2022

Non-Volatile Memories
512 TB by 2022

Andreas von Bechtolsheim. Technologies for Data- Intensive Computing. HTPS 2009

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

INF Informatik

Hochschule Reutlingen