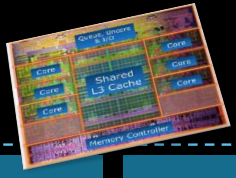
Storage

Motivation

- so far, we only talked about CPU caches and DRAM
- both are volatile (data is lost on power failure)
- traditional persistent storage:
  - disk
  - tape
- modern persistent storage:
  - solid state drive (flash)
  - storage class memory

Moving Mountains of Data



Core Register	Core L1 Cache	Core L2 Cache	Shared L3 Cache	DRAM	Storage Class Memory	Flash	HDD
Size	64KB	256KB	2-4MB	16-128GB	128GB-1TB	512GB-4TB	4-16TB
Speed	1ns	3-10ns	10-20ns	50-100ns	250-5,000ns	100,000ns-2,000,000ns	5-10,000,000ns
Cost				100x	20-25x	5x	1x

Source: Western Digital estimates

Storage Class Memory (SCM)

- also called non-volatile memory (NVM), NVRAM, persistent memory
- semiconductor-based technologies: Phase Change Memory (PCM), ReRAM, etc.
- promises:
  - similar performance as DRAM
  - byte-addressability
  - persistence
- not yet commercially available, but Intel has announced NVM-DIMMs

SCM Data Access

- connected like DRAM DIMMs (PCIe is too slow)
- mapped into address space of application
- automatically benefits from CPU caches and can be used as a (slower but cheaper) form of RAM (without re-writing software)
- How to utilize its persistence?
  - since writes are cached, they are not immediately persisted
  - write order is also a problem
  - solution: explicit write back instruction: `void _mm_clwb(void const* p)`
  - hardware guarantees that once a cache line is written back from cache, it is persisted
- SCM-aware data structures are an active research topic

Game Changer? Non-Volatile Memory (NVRAM)

Advantages

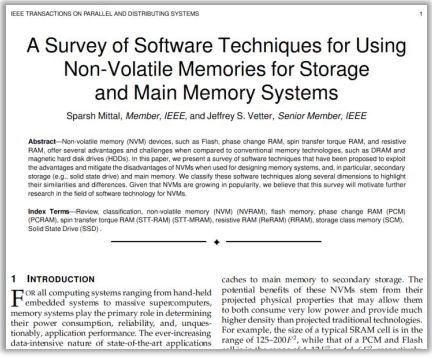
- ... does not consume energy if not used
- ... is persistent, byte-addressable
- ... x-times denser than DRAM

Drawbacks

- ... has higher latency than DRAM
  - Read latency ~2x slower than DRAM
  - Write latency ~10x slower than DRAM
- Number of writes is limited

	MRAM	DRAM	PCM	ReRAM	TLC NAND
Cost per Bit	~5	1	>0.5	>0.5	0.05
Read Latency	~1	1	10	100	1000
Write Latency	~1	1	50	1000	10000
Volatility	no	yes	no	no	no
Endurance	>1E15	> 1E16	1E6~1E8	1E6	1E3
Write Energy (J/bit)	0.1~100	1	0.1~10	0.1~10	10

Estimates provided by David Wang (Samsung)



IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTING SYSTEMS 1

A Survey of Software Techniques for Using Non-Volatile Memories for Storage and Main Memory Systems

Sparsh Mittal, *Member, IEEE*, and Jeffrey S. Vetter, *Senior Member, IEEE*

Abstract—Non-volatile memory (NVM) devices, such as Flash, phase change RAM, spin transfer torque RAM, and resistive RAM, offer several advantages and challenges when compared to conventional memory technologies, such as DRAM and magnetic hard disk drives (HDDs). In this paper, we present a survey of software techniques that have been proposed to exploit the advantages and mitigate the disadvantages of NVMs when used for designing memory systems, and, in particular, secondary storage (e.g., solid state drive) and main memory. We classify these software techniques along several dimensions to highlight their similarities and differences. Given that NVMs are growing in popularity, we believe that this survey will motivate further research in the field of software technology for NVMs.

Index Terms—Review, classification, non-volatile memory (NVM) (NVRAM), flash memory, phase change RAM (PCM) (PCRAM), spin transfer torque RAM (STT-RAM) (STT-MRAM), resistive RAM (ReRAM) (RRAM), storage class memory (SCM), Solid State Drive (SSD) .

1 Introduction

For all computing systems ranging from hand-held embedded systems to massive supercomputers, memory systems play the primary role in determining their power consumption, reliability, and, unquestionably, application performance. The ever-increasing data-intensive nature of state-of-the-art applications

caches to main memory to secondary storage. The potential benefits of these NVMs stem from their projected physical properties that may allow them to both consume very low power and provide much higher density than projected traditional technologies. For example, the size of a typical SRAM cell is in the range of 125–200 F^2, while that of a PCM and Flash

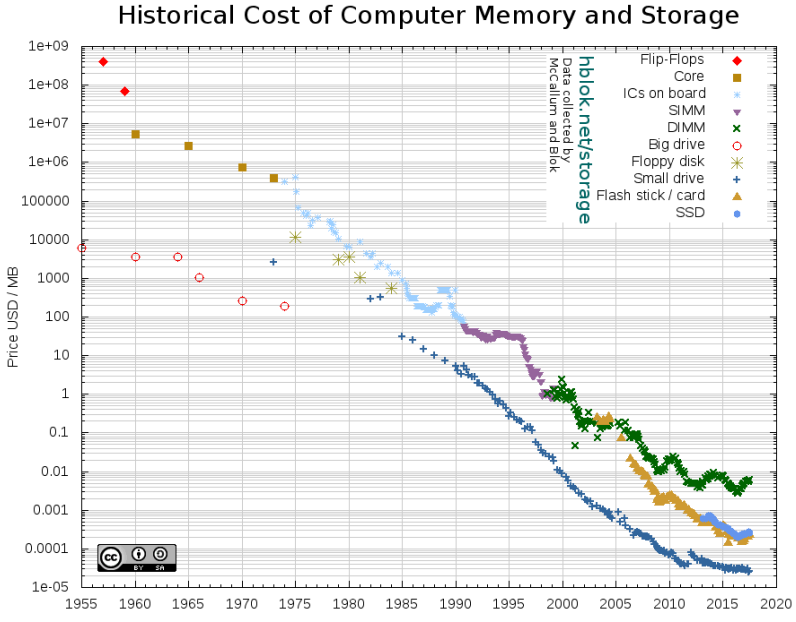
Parameter	NAND	DRAM	PCM	STT-RAM
Read Latency	25 µs	50 ns	50 ns	10 ns
Write Latency	500 µs	50 ns	500 ns	50 ns
Byte-addressable	No	Yes	Yes	Yes
Endurance	10^4–10^5	$>10^{15}$	10^8–10^9	$>10^{15}$

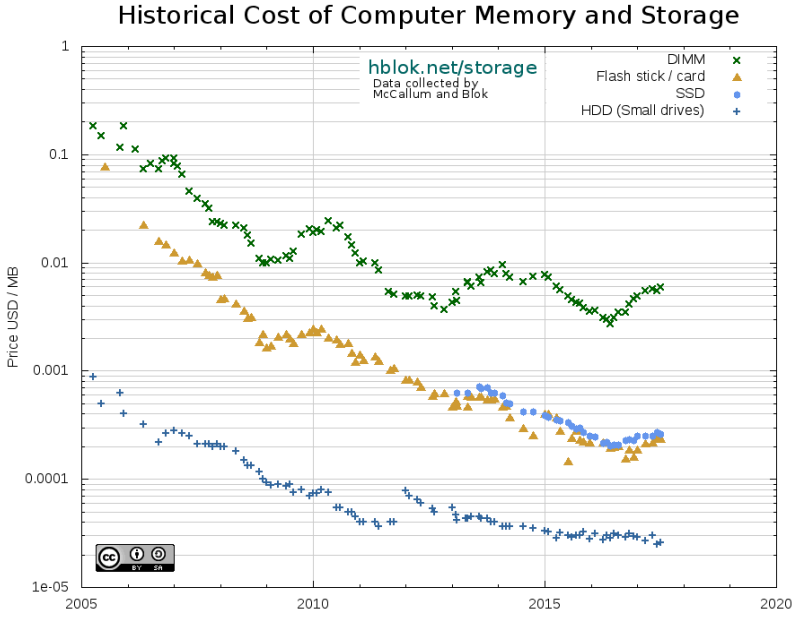
Solid State Drive

- based on semiconductors (no moving parts)
- NAND gates
- available as SATA- or PCIe/M.2- attached devices (NVMe interface)
- e.g., Intel P3700:
  - 2.8/1.9 GB/s sequential read/write bandwidth
  - 450K/150K random 4KB read/write IOs/s (requires many concurrent accesses)

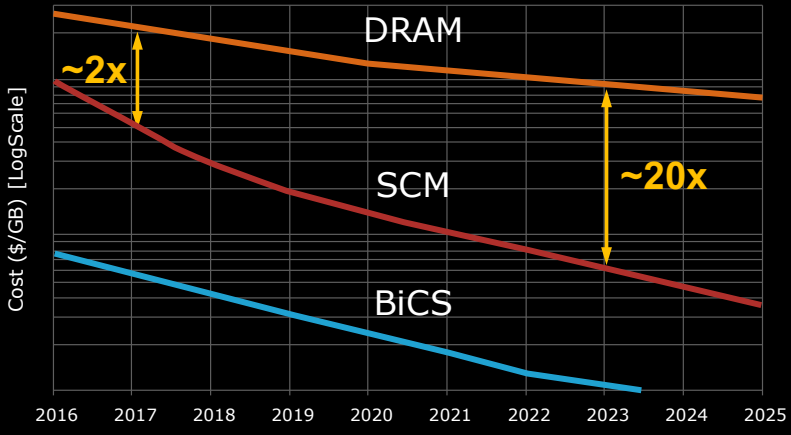
SSD Data Access

- data is stored on pages (e.g., 4KB, 8KB, 16KB), access is through the normal OS block device abstraction (i.e., `read`, `write`)
- physical properties:
  - pages are combined to blocks (e.g., 2MB)
  - it is not possible to overwrite a page, it is necessary to erase a block first
  - erasing is fairly expensive
- these properties are usually not exposed
- SSDs implement a flash translation layer (FTL) that emulates a traditional read/write interface (including in-page updates)


Historical Cost of Computer Memory and Storage
Price USD / MB
1e+09
1e+08
1e+07
1e+06
100000
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
1e-05
1955
1960
1965
1970
1975
1980
1985
1990
1995
2000
2005
2010
2015
2020
hblok.net/storage
Data collected by McCallum and Blok
Flip-Flops
Core
ICs on board
SIMM
DIMM
Big drive
Floppy disk
Small drive
Flash stick / card
SSD


Historical Cost of Computer Memory and Storage
hblok.net/storage
Data collected by
McCallum and Blok
DIMM
Flash stick / card
SSD
HDD (Small drives)
Price USD / MB
1
0.1
0.01
0.001
0.0001
1e-05
2005
2010
2015
2020

Cost Scaling for SCM Market Adoption



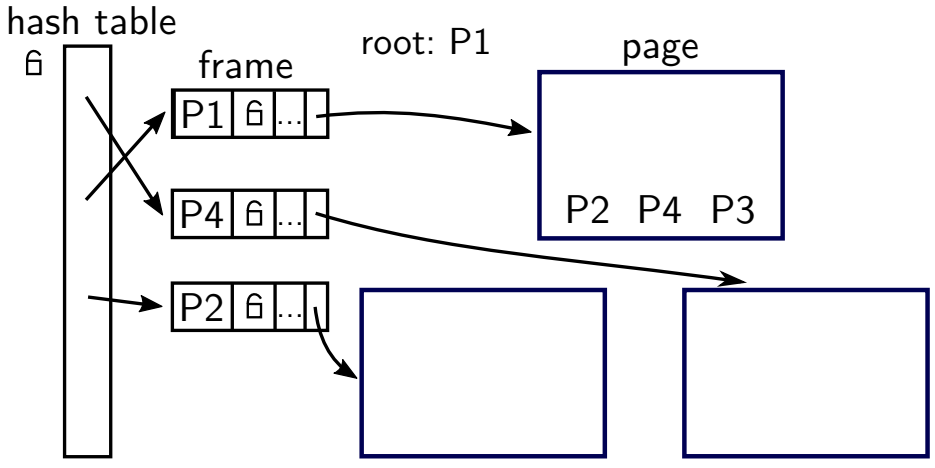
Note: Technology transition cadence assumed 18 months for all technologies
ReRAM & 3DXP greenfield fabs, NAND & DRAM existing fabs

* DRAM data source: IDC ASP forecast with 45% GM assumed
** 13 nm: assumes EUV @ 1.4x i-ArF capex cost, 2160 w/day

Managing Larger-than-RAM Data Sets

- traditional databases use a buffer pool to support very large data sets:
  - store everything on fixed-size pages (e.g., 8KB)
  - pages are fixed/unfixed
  - transparently handles arbitrary data (relations, indexes)
- traditional systems are very good at minimizing the number of disk IOs
- however, in-memory systems are much faster than disk-based systems (as long as all data fits into RAM)

Canonical Buffer Management



- Effelsberg and Härder, TODS 1984

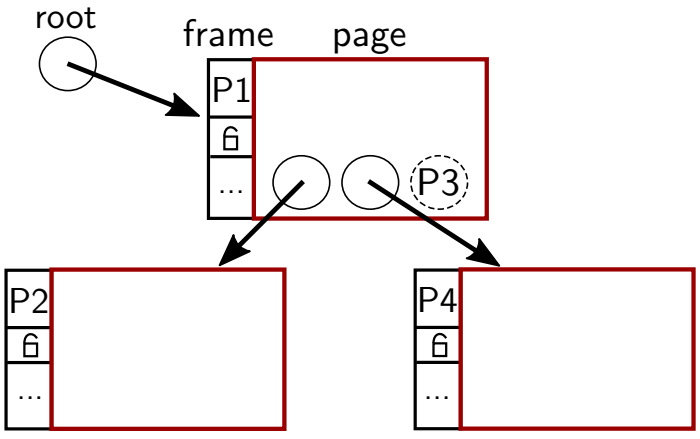
Managing Larger-Than-Memory Data Sets

- paging/mmap
  - no control over eviction (needed for OLTP/HTAP systems)
- Anti Caching
  - each tuple has a per-relation LRU list
  - move cold tuples to disk/SSD
  - indexes refer to hot and cold tuples
- Siberia
  - record tuple accesses in a log
  - identify cold tuples from log (offline)
  - move cold tuples to disk or SSD
  - in-memory index only stores hot (in-memory) tuples
  - bloom (or adaptive range) filter for cold tuples
- buffer managers
  - have full full control over eviction
  - handle arbitrary data structures (tables, indexes, etc.) transparently

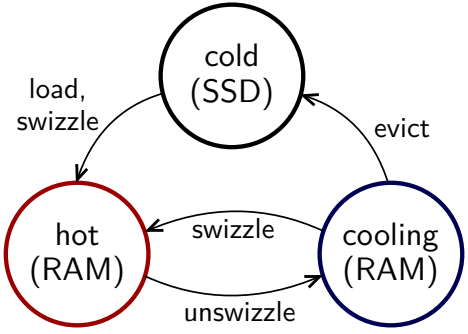
Hardware-Driven Desiderata

- fast and cheap PCIe-attached NVMe SSDs
  (1/10th of the price and bandwidth of DRAM)
  $\Rightarrow$ store everything (relations, indexes) on pages (e.g., 16 KB)
- huge main memory capacities
  $\Rightarrow$ make in-memory accesses very fast
- dozens (soon hundreds?) of cores
  $\Rightarrow$ use smart synchronization techniques

Pointer Swizzling

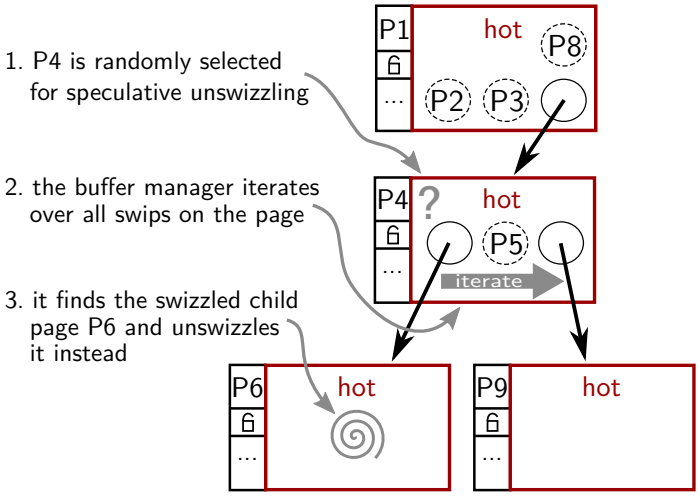


Replacement Strategy (1)



1. select **random** page as a potential candidate for eviction
2. cooling pages are organized in a **FIFO** queue (e.g., 10% of all pages are in cooling state)

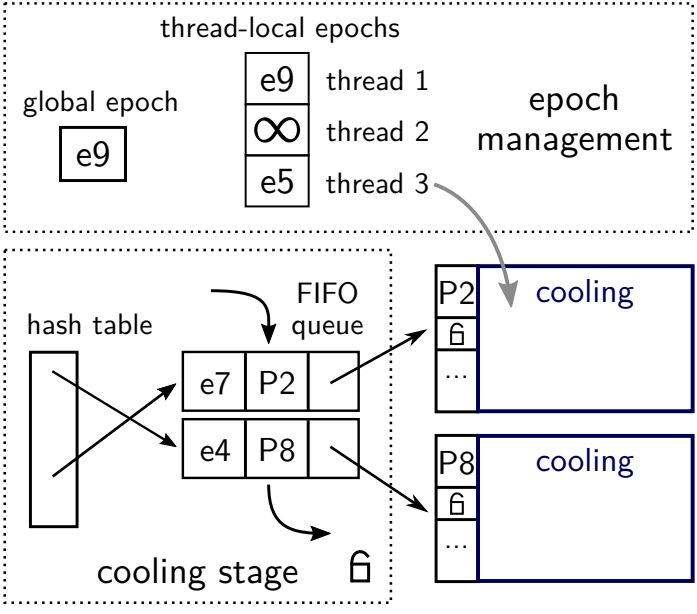
Replacement Strategy (2)



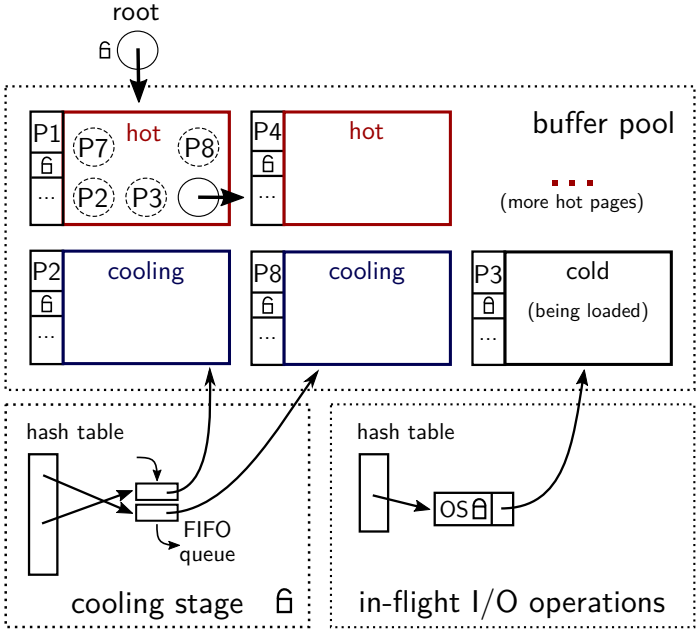
Scalable Optimistic Synchronization Primitives

- no hash table synchronization for accessing hot pages
- per-page optimistic latches
- epoch-based memory reclamation and eviction
- global latch for cooling stage and I/O manager

Epochs


thread-local epochs
global epoch
e9
e9
thread 1
∞
thread 2
e5
thread 3
epoch management
hash table
FIFO queue
e7
P2
e4
P8
cooling stage
P2
cooling
P8
cooling

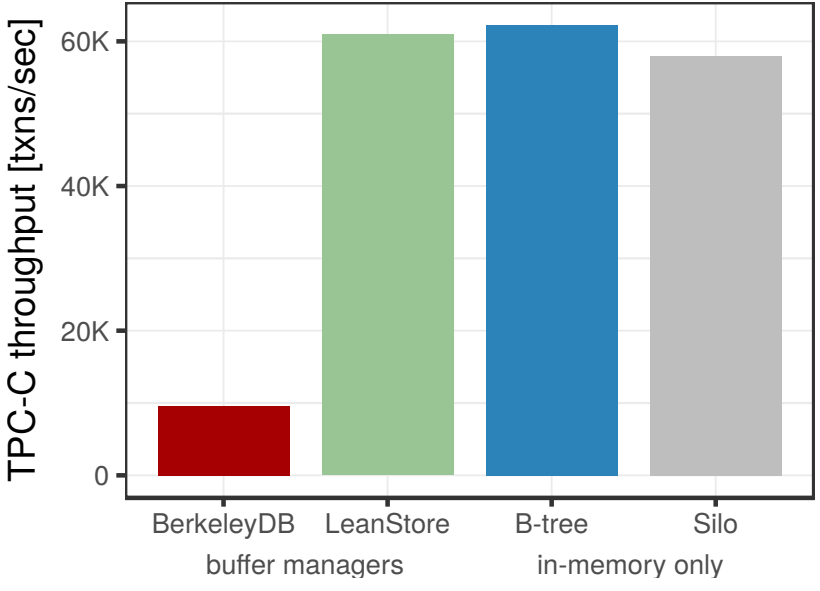
Putting It All Together



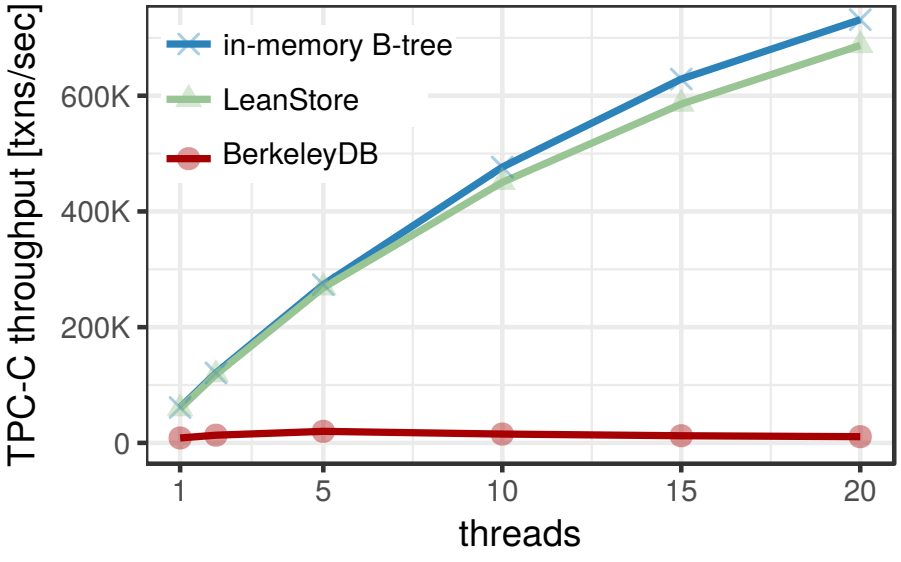
Evaluation

- 10 core Haswell EP
- Linux 4.8
- Intel NVMe P3700 SSD
- storage managers compared (no logging, no transactions, 16 KB pages):
  - **LeanStore**: B-tree on top of buffer manager
  - in-memory B-tree: same B-tree without buffer management
  - BerkeleyDB: B-tree on top of traditional buffer manager

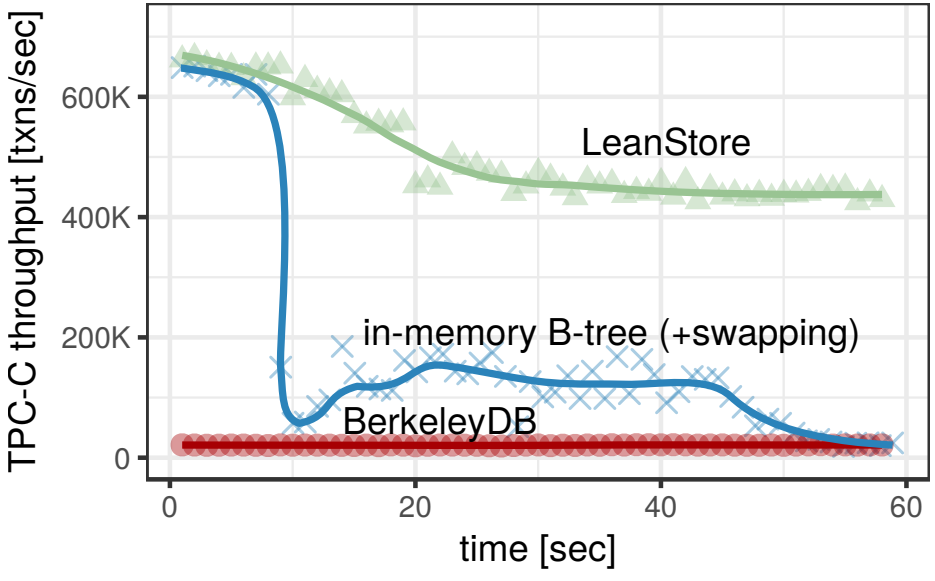
How Large Is the Overhead? (Single-Threaded)



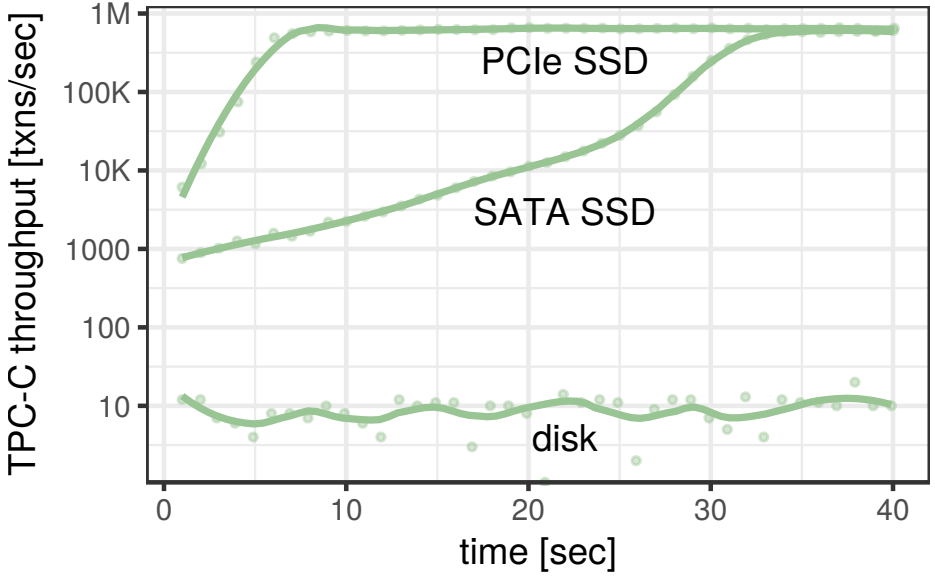
How Well Does It Scale?



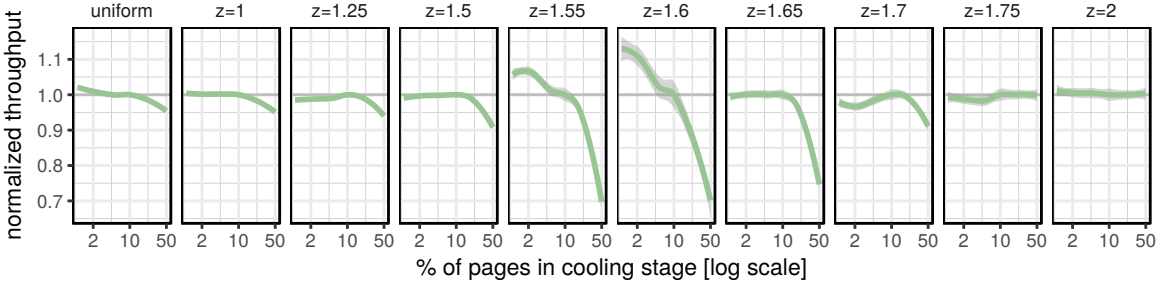
What Happens If The Data Does Not Fit Into RAM?



TPC-C Ramp Up With Cold Cache



Replacement Strategy Parameter



Conclusions

- in-memory performance is competitive with the fastest main-memory systems
- transparent management of arbitrary data structures (e.g., B-trees, hash tables, column-wise storage)
- scales well on multi-core CPUs