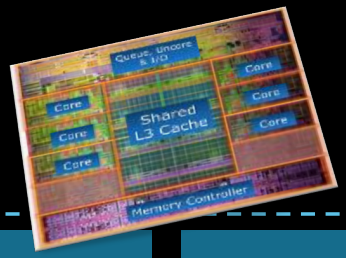
FRIEDRICH-SCHILLER-
UNIVERSITÄT
JENA

LeanStore

Prof. Dr. Viktor Leis

Professur für Datenbanken und Informationssysteme

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

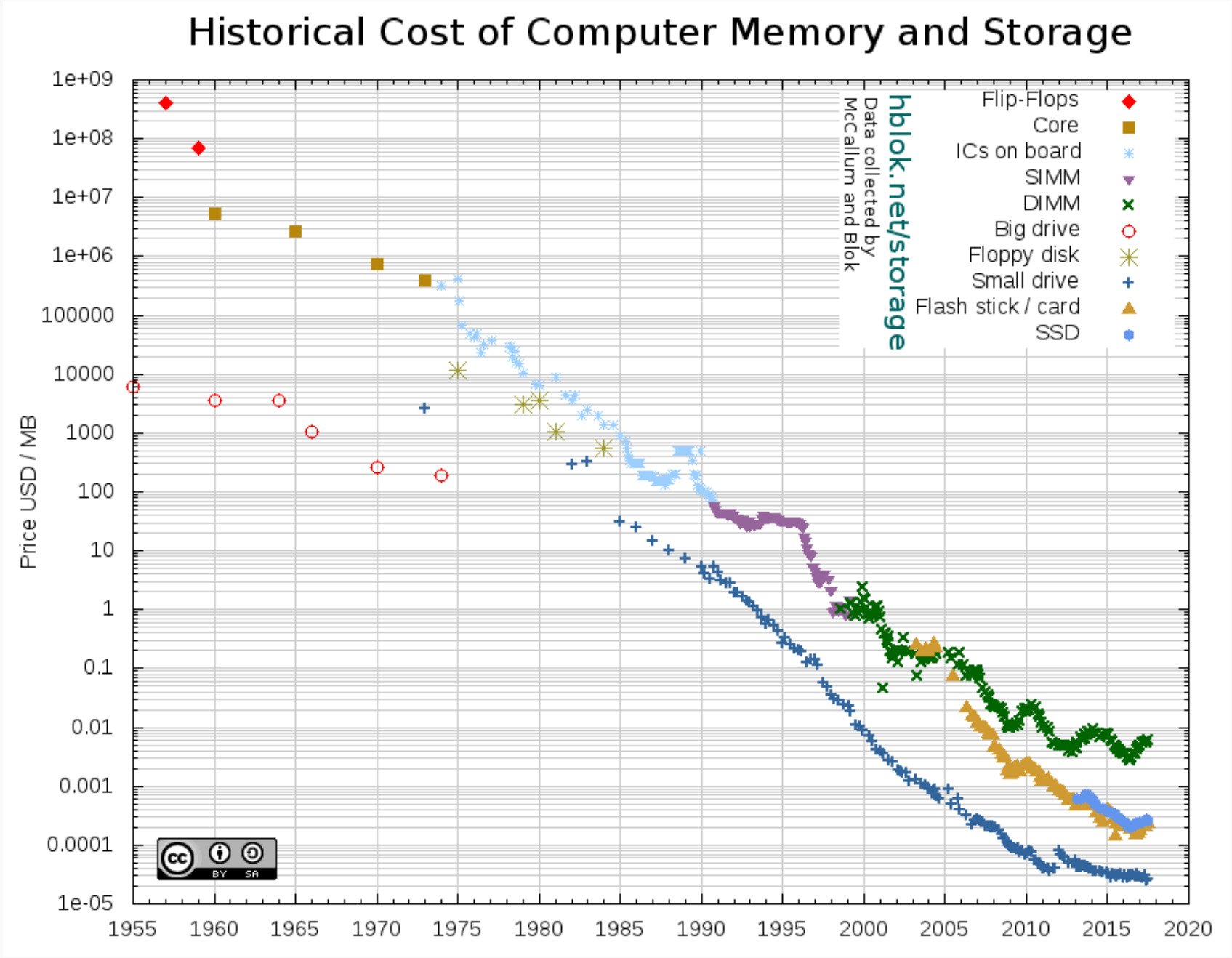
©2016 Western Digital Corporation or its affiliates. All rights reserved.

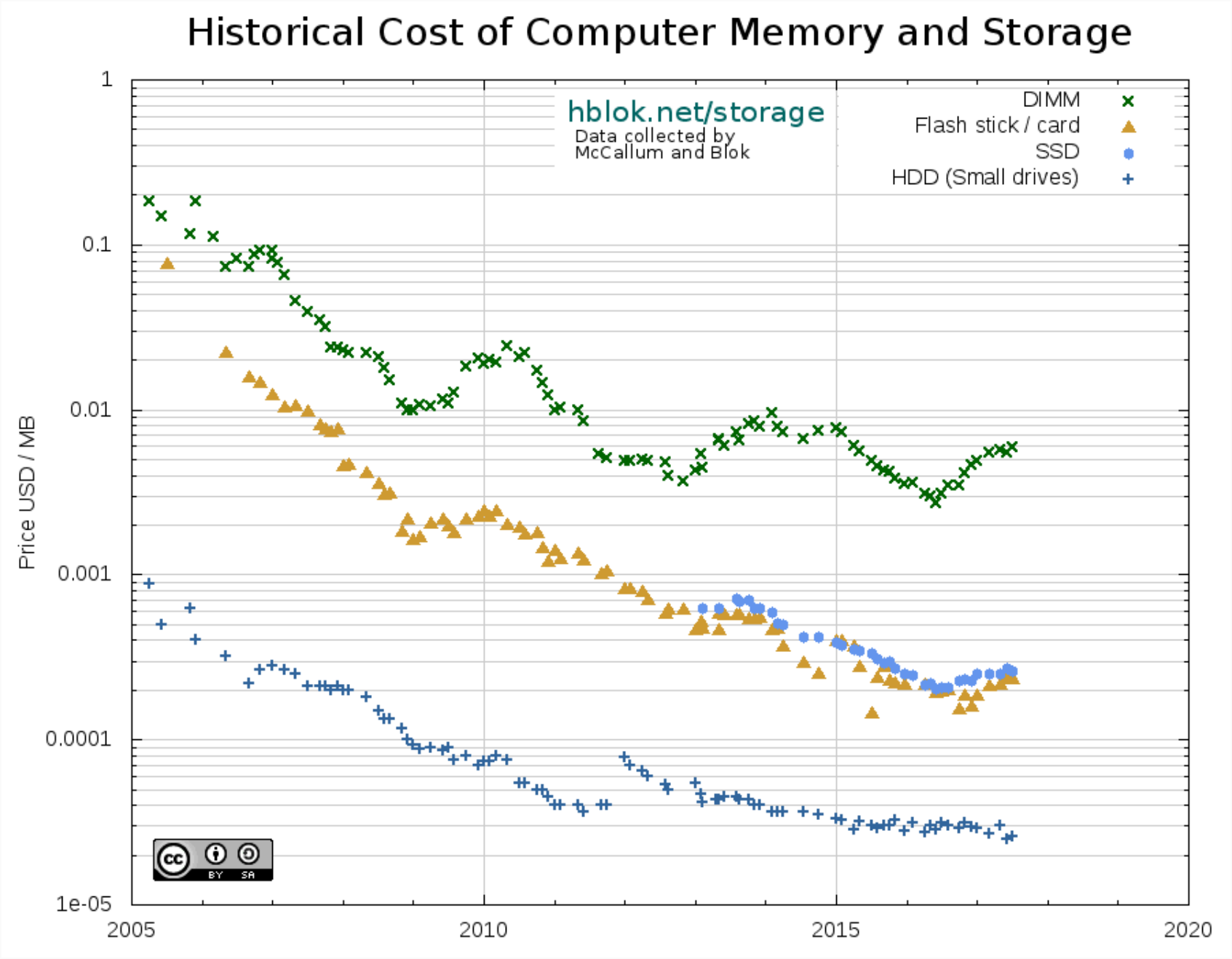
Solid State Drives (“Flash”)

- based on semiconductors (no moving parts)
- NAND gates
- available as SATA- or PCIe/M.2/U.2 attached devices (NVMe interface)
- e.g., Samsung 970 EVO:
  - 3.3/2.4 GB/s sequential read/write bandwidth
  - 500K/450K random 4KB read/write IOs/s (requires many concurrent accesses)
  - random read latency around 100 microseconds
  - around 200 EUR for 1 TB

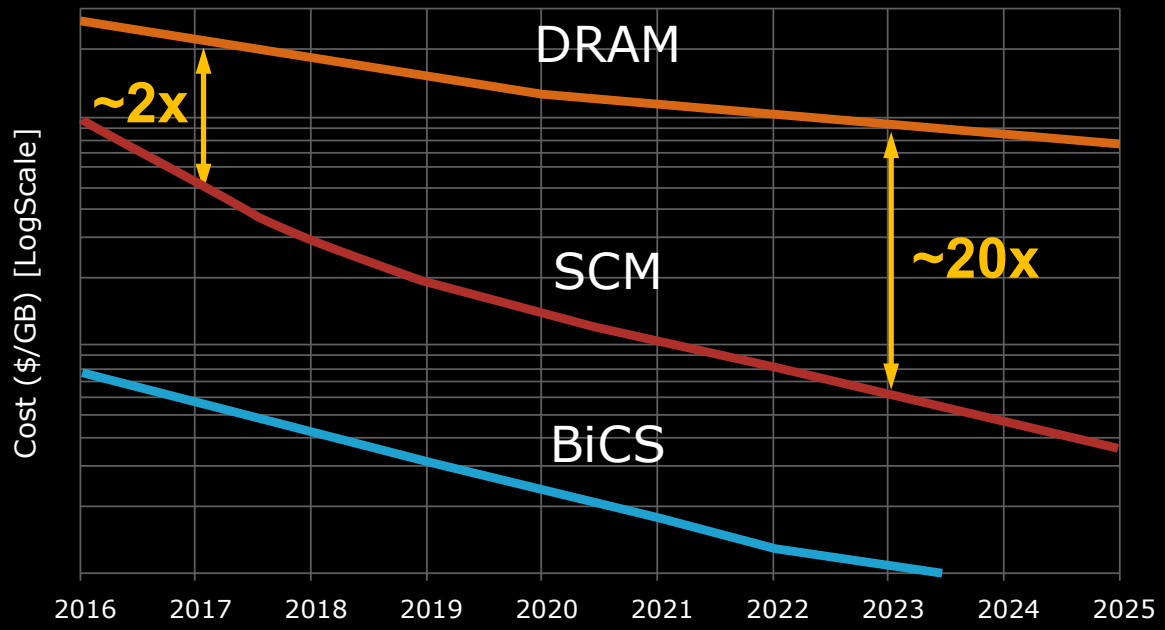
SSD Data Access

- 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
BY SA


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
BY SA

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

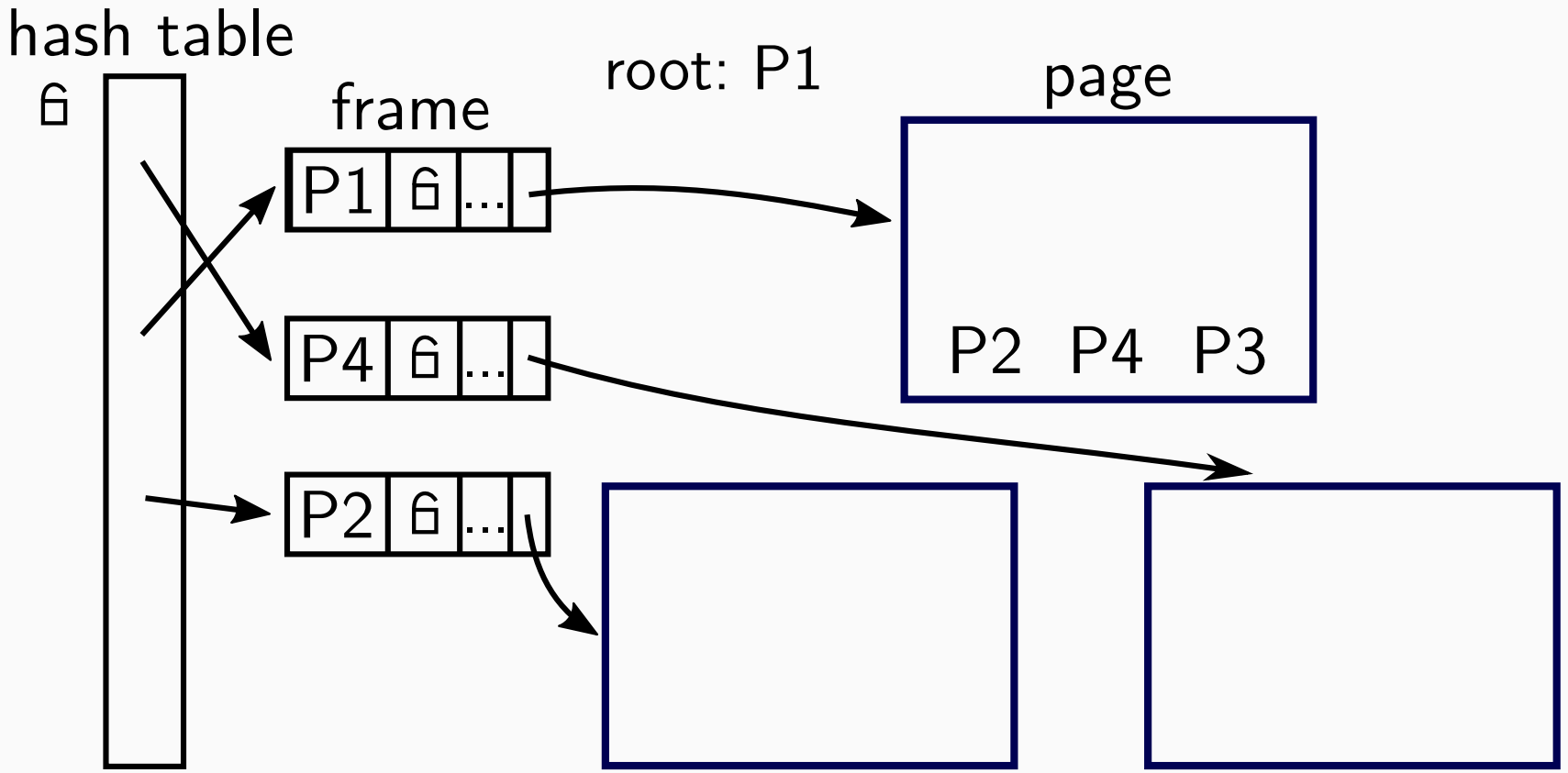
Western Digital

©2016 Western Digital Corporation or its affiliates. All rights reserved.

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

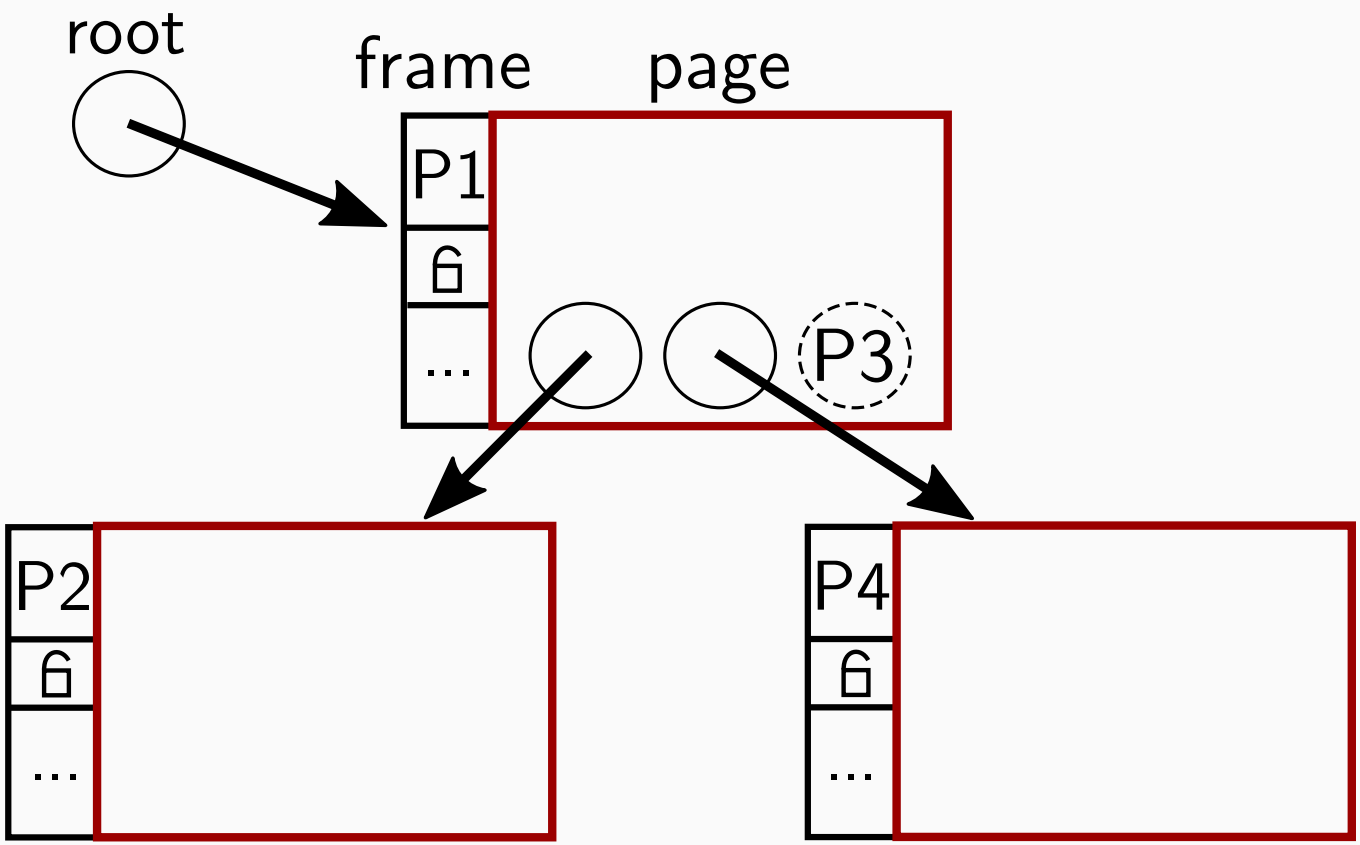
Hardware-Driven Desiderata

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

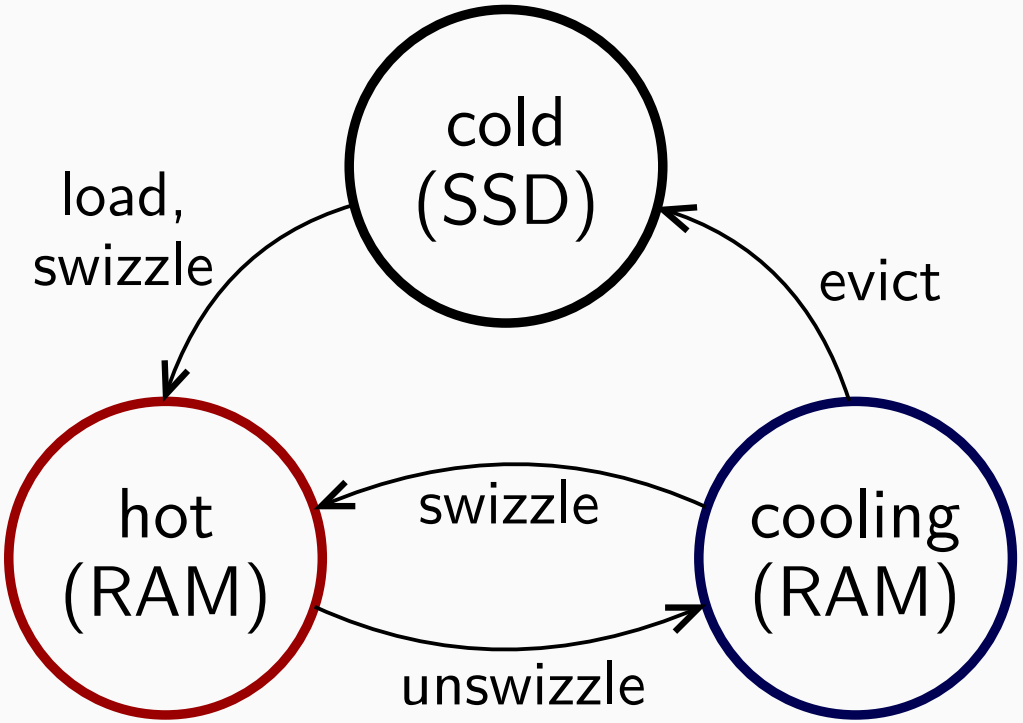
LeanStore: Key Techniques

1. pointer swizzling
2. low-overhead page replacement strategy
3. scalable optimistic synchronization

Pointer Swizzling



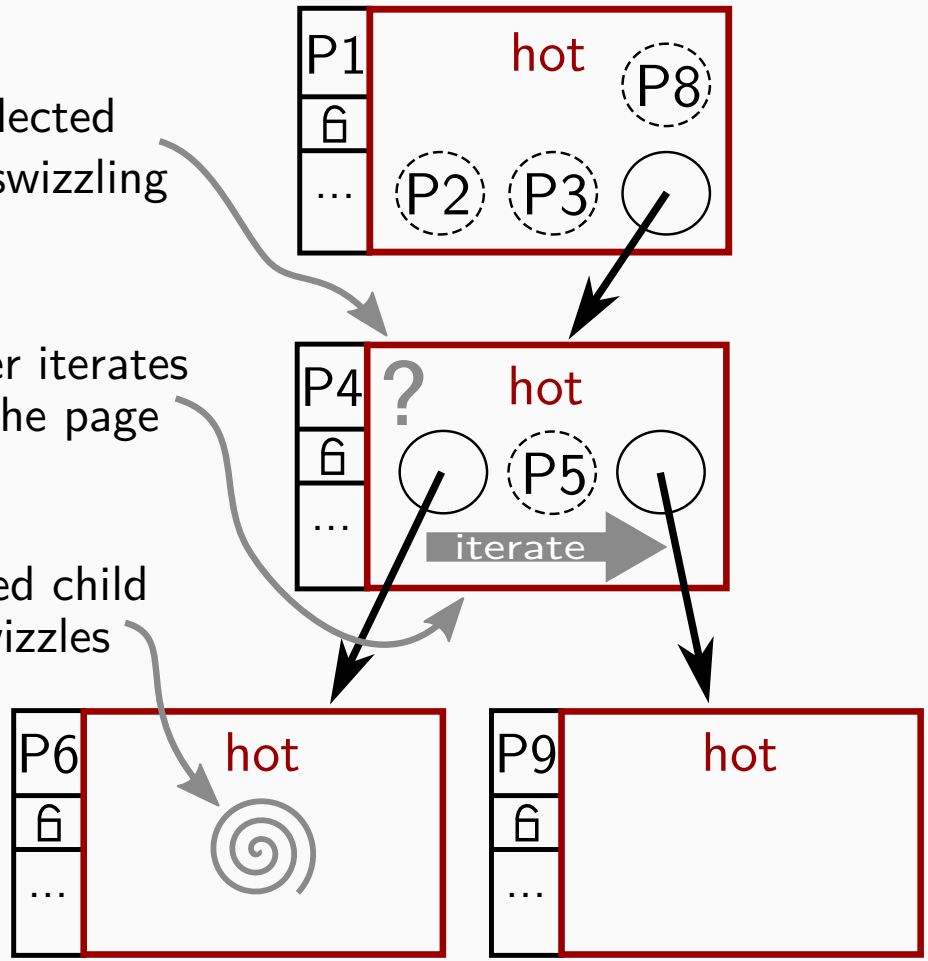
Replacement Strategy (1)



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)

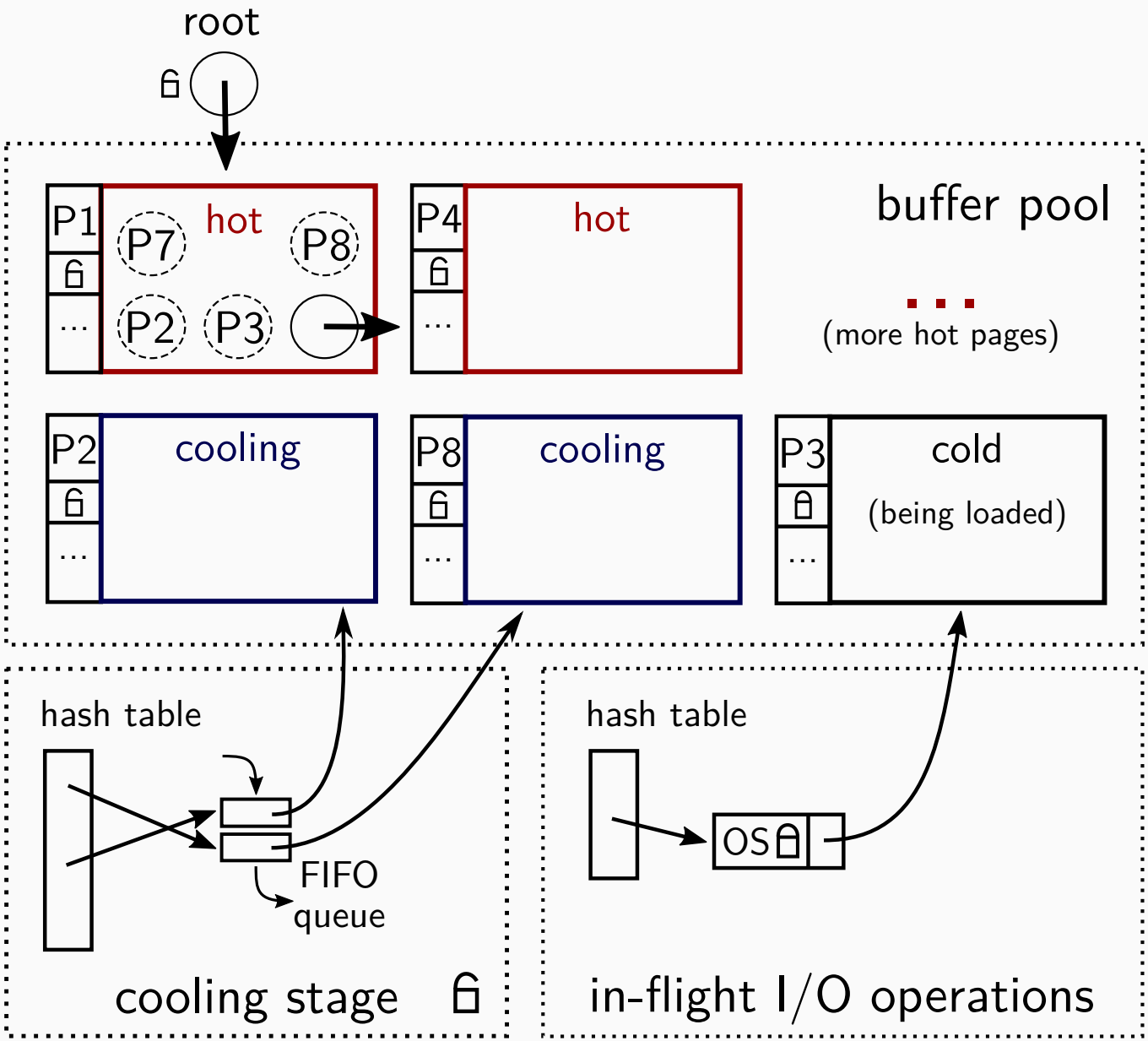
1. P4 is randomly selected for speculative unswizzling
2. the buffer manager iterates over all swips on the page
3. it finds the swizzled child page P6 and unswizzles it instead



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

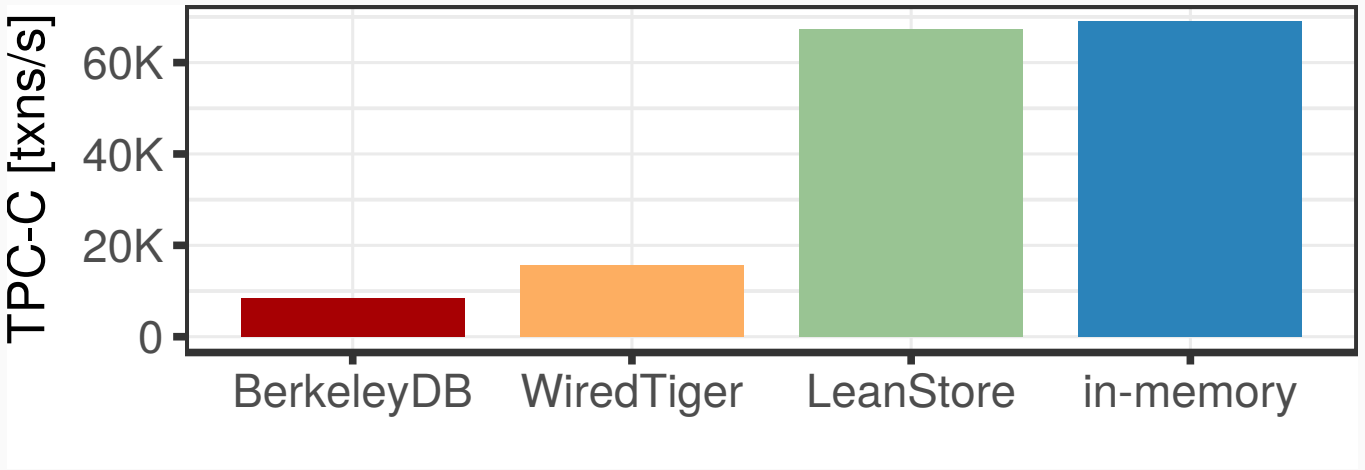
Putting It All Together



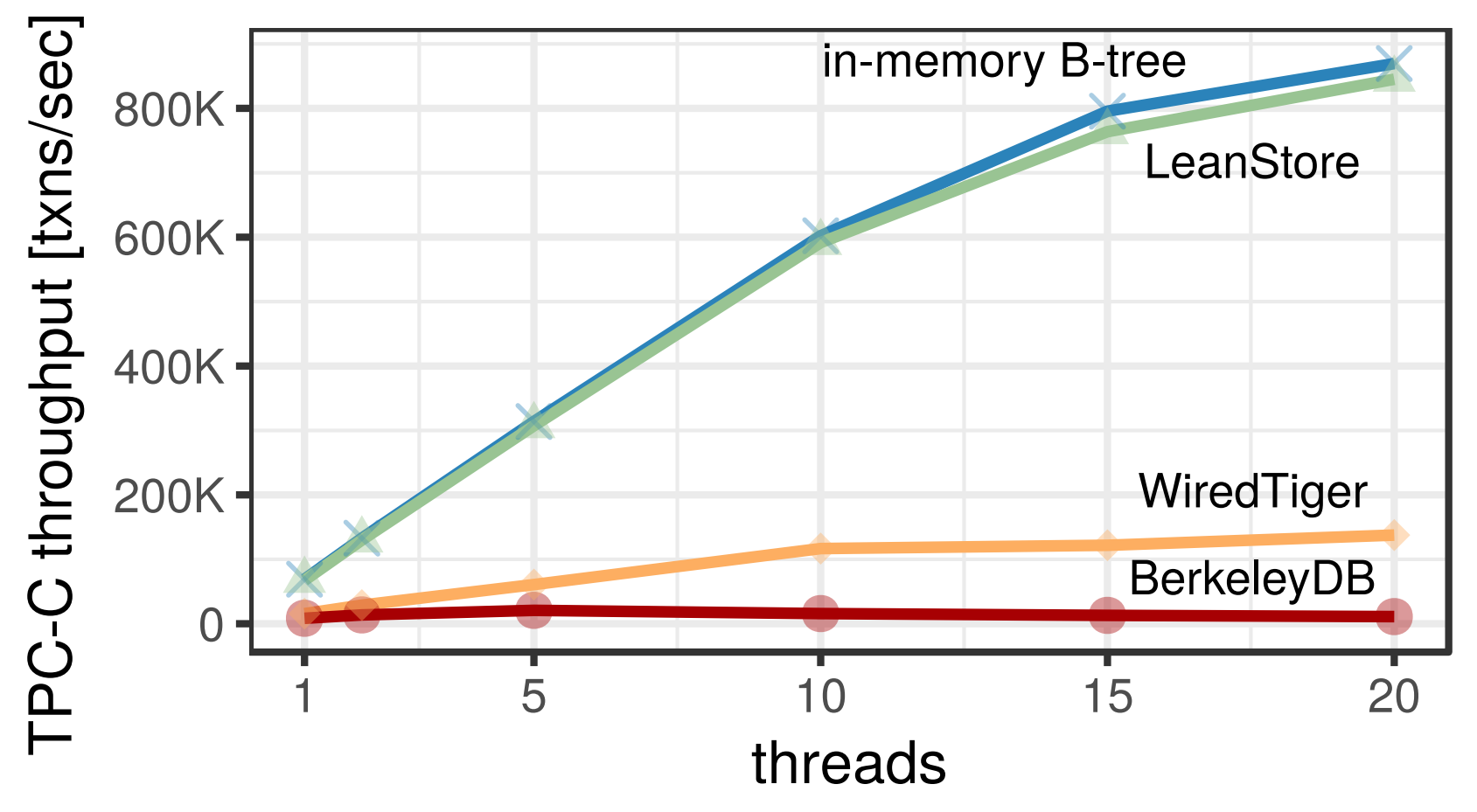
Evaluation

- 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

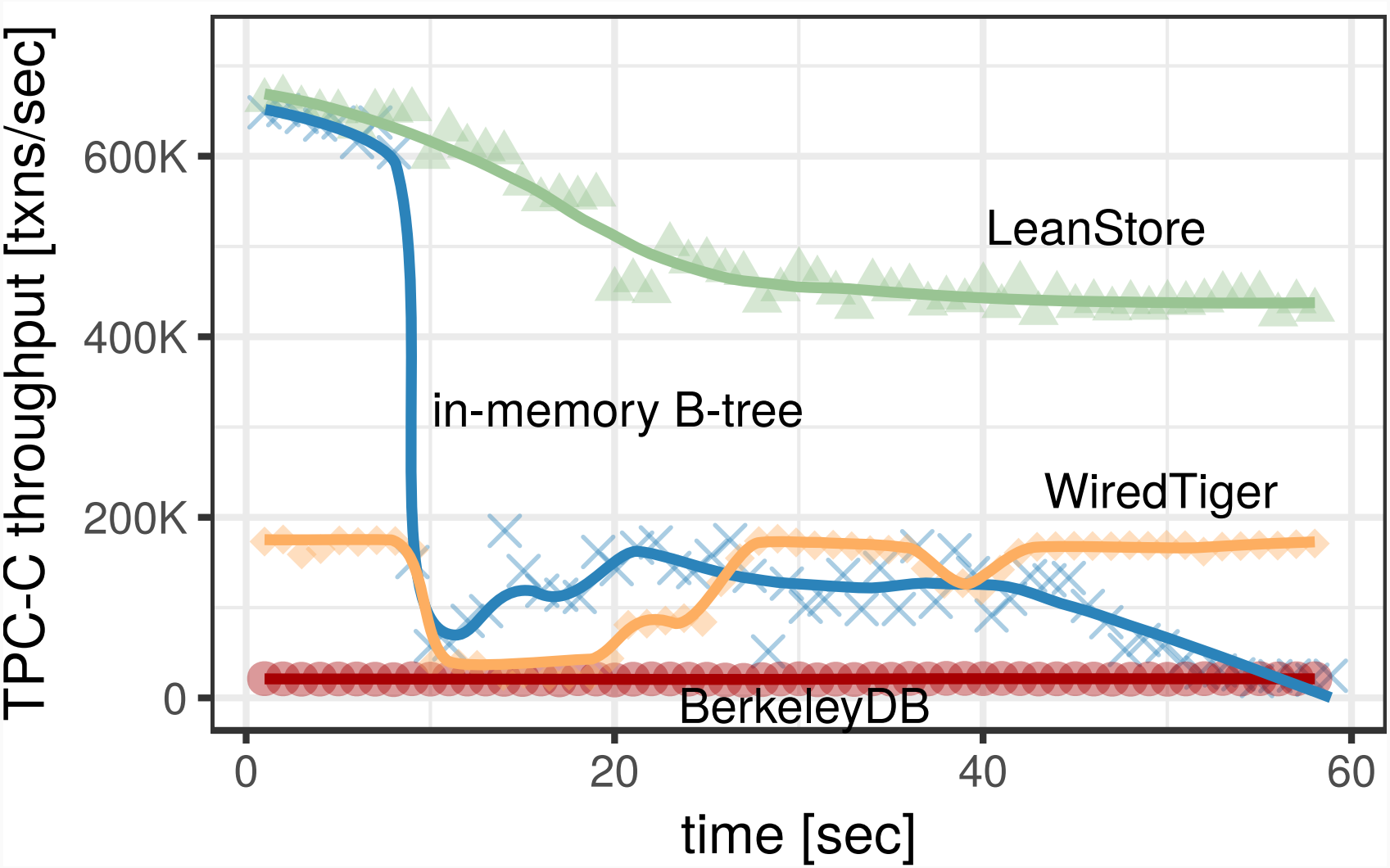
How Large Is the Overhead? (Single-Threaded)



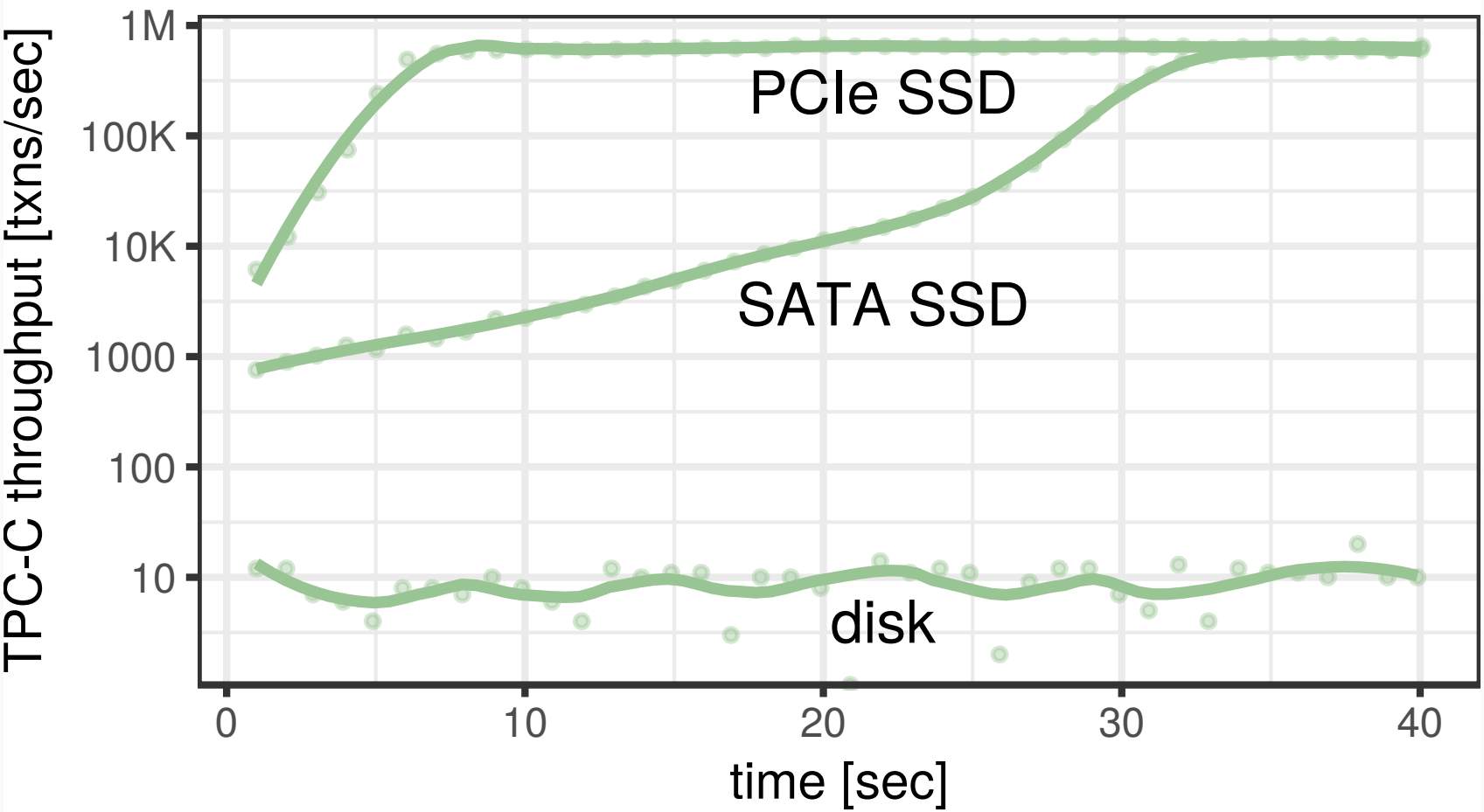
How Well Does It Scale?



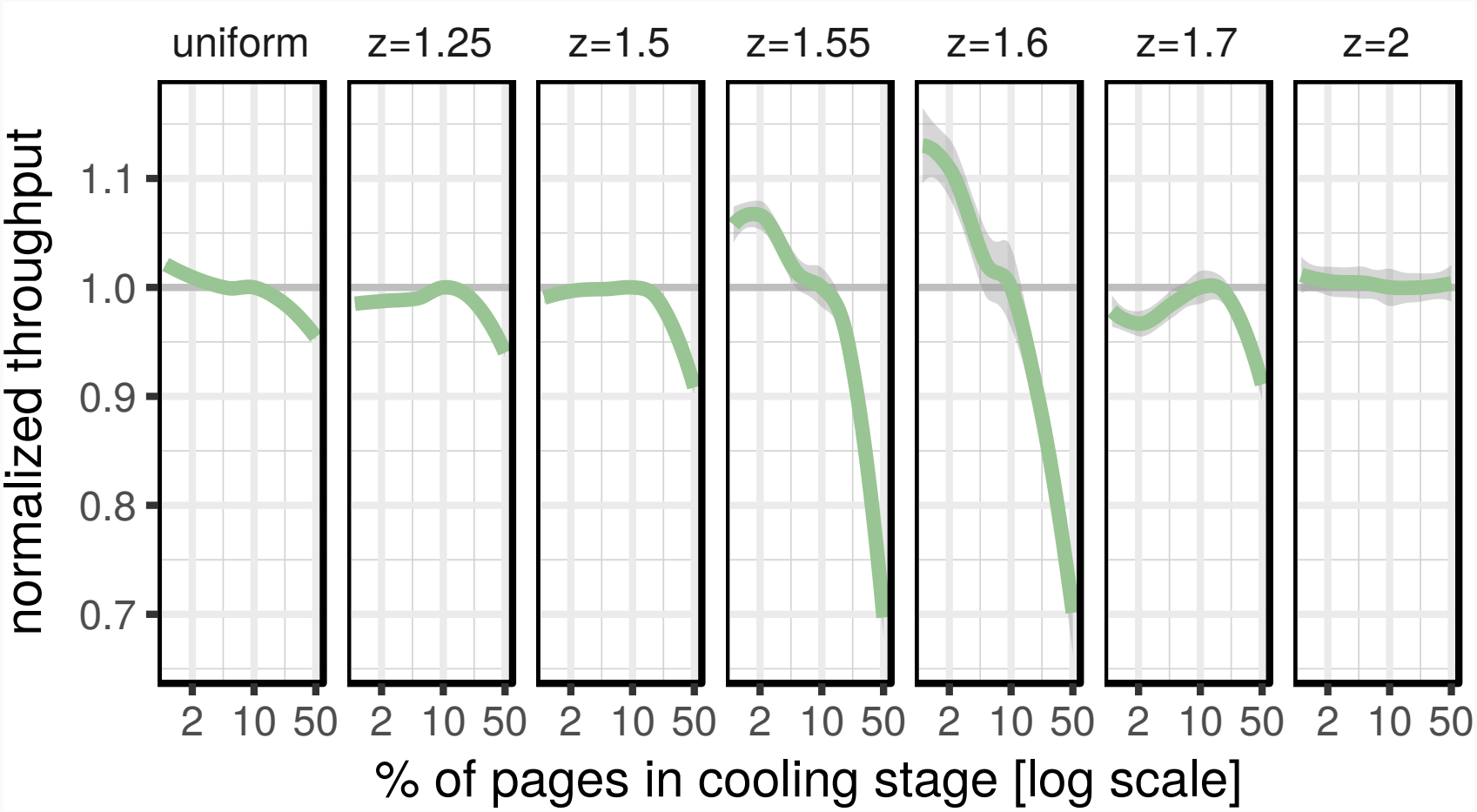
What Happens If The Data Does Not Fit Into RAM?



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