SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2015

Developing Software for Persistent Memory

Thomas Willhalm, Karthik Kumar
Intel Corporation

Disclaimer

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

A "Mission Critical Application" is any application in which failure of the Intel Product could result, directly or indirectly, in personal injury or death. SHOULD YOU PURCHASE OR USE INTEL'S PRODUCTS FOR ANY SUCH MISSION CRITICAL APPLICATION, YOU SHALL INDEMNIFY AND HOLD INTEL AND ITS SUBSIDIARIES, SUBCONTRACTORS AND AFFILIATES, AND THE DIRECTORS, OFFICERS, AND EMPLOYEES OF EACH, HARMLESS AGAINST ALL CLAIMS COSTS, DAMAGES, AND EXPENSES AND REASONABLE ATTORNEYS' FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PRODUCT LIABILITY, PERSONAL INJURY, OR DEATH ARISING IN ANY WAY OUT OF SUCH MISSION CRITICAL APPLICATION, WHETHER OR NOT INTEL OR ITS SUBCONTRACTOR WAS NEGLIGENT IN THE DESIGN, MANUFACTURE, OR WARNING OF THE INTEL PRODUCT OR ANY OF ITS PARTS.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined". Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

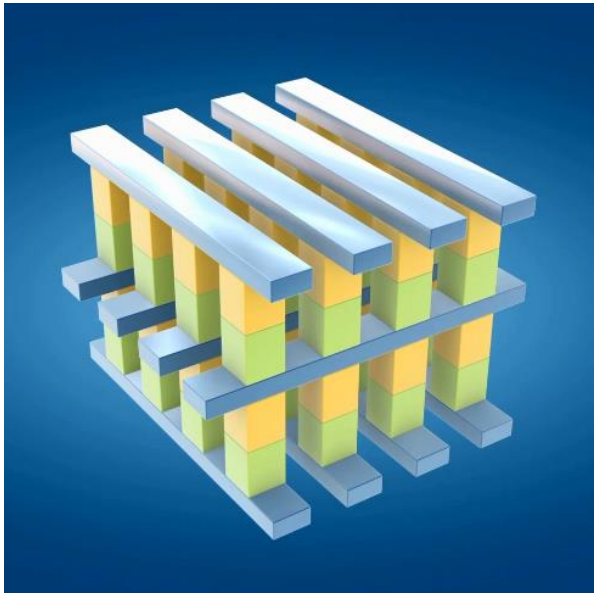
The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

This document contains information on products in the design phase of development. The information here is subject to change without notice. Do not finalize a design with this information.

Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Intel 3D XPoint Technology



- https://youtu.be/OaAjLyPtoyE

Key Messages at the Announcement

Intel and Micron begin production on a new class of non-volatile memory, creating the first new memory category in more than 25 years since the introduction of NAND Flash memory in 1989

- **Up to 1000 times faster[1] than NAND**
- **Up to 1000 times greater endurance[2] than NAND**
- **10 times denser than conventional memory[3]**

Built from the ground up, 3D XPoint™ technology uses an innovative, transistor-less memory cell architecture creating a three-dimensional checkerboard with perpendicular wires connecting submicroscopic columns. Each memory cell sits at the intersection of a word line and a bit line and can be addressed individually by selecting its top and bottom wire

1 Performance difference based on comparison between 3D XPoint technology and other industry NAND
2 Endurance difference based on comparison between 3D XPoint technology and other industry NAND
3 Density difference based on comparison between 3D XPoint technology and other industry DRAM

Agenda

- Technology Differentiators
  - Benefits and caveats to persisting data structures
  - Why programming to persistent memory is different
- Case study: Design of a Persistent Memory Enabled Database
  - Design decisions
  - Evaluation methodology

Technology Differentiators

	Differentiators for Persistent Memory Technologies	
+	Large capacity	Higher density will allow larger DIMMs
+	Persistence	Data stored in PMem will remain in memory after reboot
–	Higher latencies	Latencies higher than DRAM (100ns) but much less than latest gen-PCIe SSDs (10,000 ns)

What value do these technology differentiators offer?

Opportunity: Persistence in Memory Tier

For decades of computing...

Working store of data

Byte addressable
Low latency
High bandwidth

!=

Durable store of data

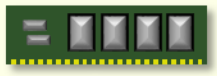
Persistent
Very Large Capacity

Restart time to load from disk
Convert from disk to memory format

Transactions wait until disk IO complete
Convert from memory to disk format

Now

Persistent Memory



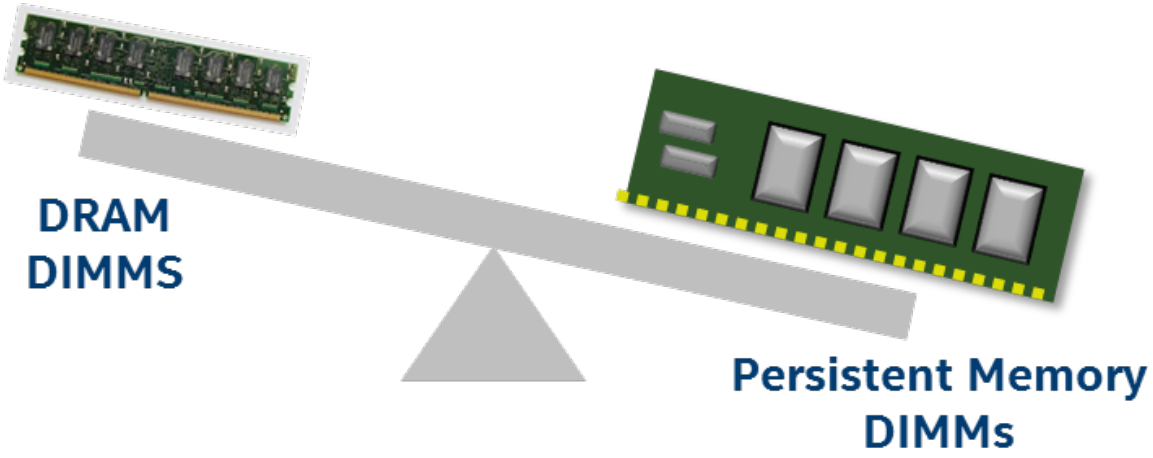
One store to rule them all!

Byte addressable
Low latency
High bandwidth

Persistent
Large Capacity

Use-case where memory meets disk is a potential game changer for applications

Opportunity: Large Capacity



- Larger datasets in memory: Less paging, improved performance
- Scale-up vs. scale-out for in-memory solutions: Single coherent address space, avoid commit protocols for transactions

Large capacity: avoid disk accesses, benefit in-memory computing

Opportunity: Why Restart Time Matters

Logging, warehousing, processing information: lifeline of companies

Information availability depends on database availability (9s)

Minimize restart time to improve database availability

Availability	Annual DownTime
97%	11 days
98%	7 days
99%	3 days 15 hrs
99.9%	8 hrs 48 min
99.99%	53 min
99.999%	5 min
99.9999%	32 sec

- Each restart for an IMDB can take upto 1 hour to load TBs of data to memory.
- Dell study shows millions of dollars lost per hour due to downtime**
- Existing HA solutions increase the price *exponentially* for every nine

7

**http://tanejagroup.com/files/Compellent_TG_Opinion_5_Nines_Sept_20121.pdf

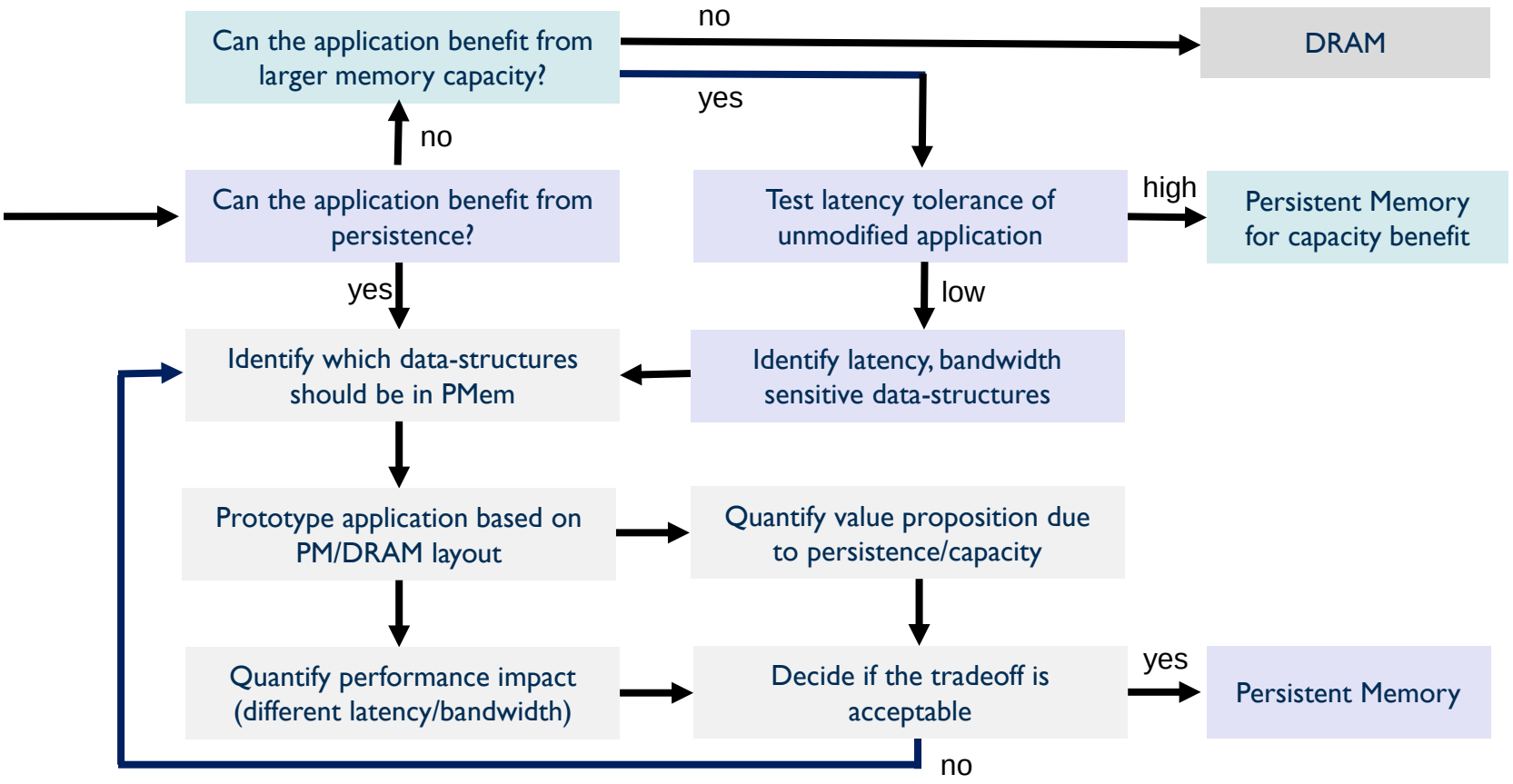
Caveat: Higher Memory Latencies

- Why not hold all data in PMem: higher latencies
- What are considerations for moving a data structure to PMem?

Data Layout and Size	Frequency of Access	Pattern of Access
• Can caching hide latency for data layout/size? Example: Arrays vs. linked lists	• Are data references frequent & performance-critical? Example: cold vs. hot stores	• Are data access patterns prefetch & cache friendly? Example: hash lookups vs column scans

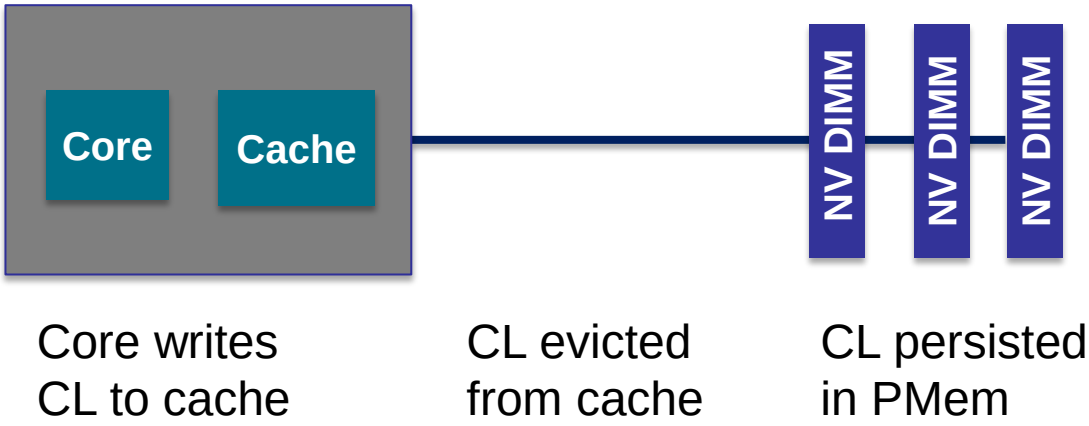
Need to identify application performance sensitivity for persisted data structures

Software Architecture: Persistent Memory



Systematically identify which data structures can benefit from Persistent Memory

Application is Responsible for Durability



- Need to regularly push stores out of processor caches to NVM
- Need to commit outstanding stores in volatile buffers to NVM

New instructions needed for apps to explicitly commit stores to durability

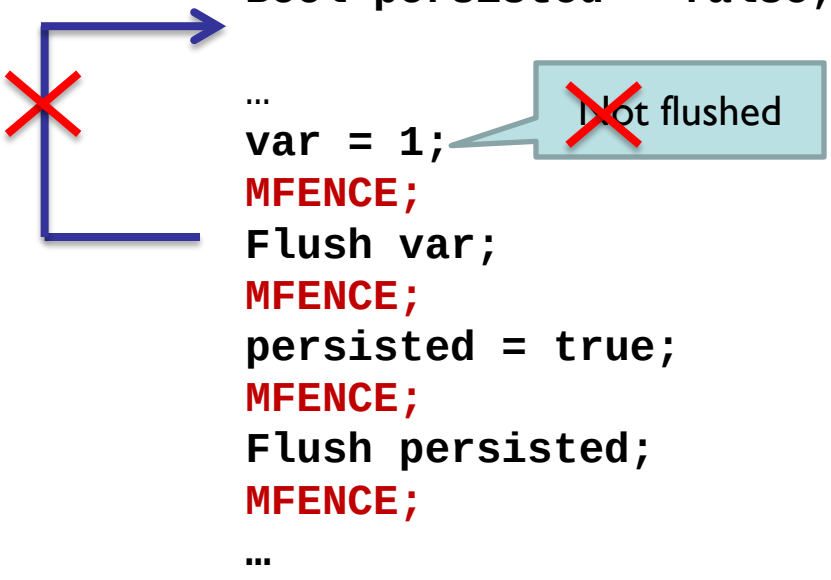
Flushing Writes from Caches

Instruction	Meaning
CLFLUSH addr	Cache Line Flush: Available for a long time
CLFLUSHOPT addr	Optimized Cache Line Flush: New to allow concurrency
CLWB addr	Cache Line Write Back: Leave value in cache for performance of next access

Flushing Writes from Memory Controller

Mechanism	Meaning
PCOMMIT	Persistent Commit: Flush stores accepted by memory subsystem
Asynchronous DRAM Refresh	Flush outstanding writes on power failure Platform-Specific Feature

Persisting Data on the Fly: Example



Before

	Cache	PMem
var	0	0
persisted	False	False

After (incorrect)

	Cache	PMem
var	I	0
persisted	True	True

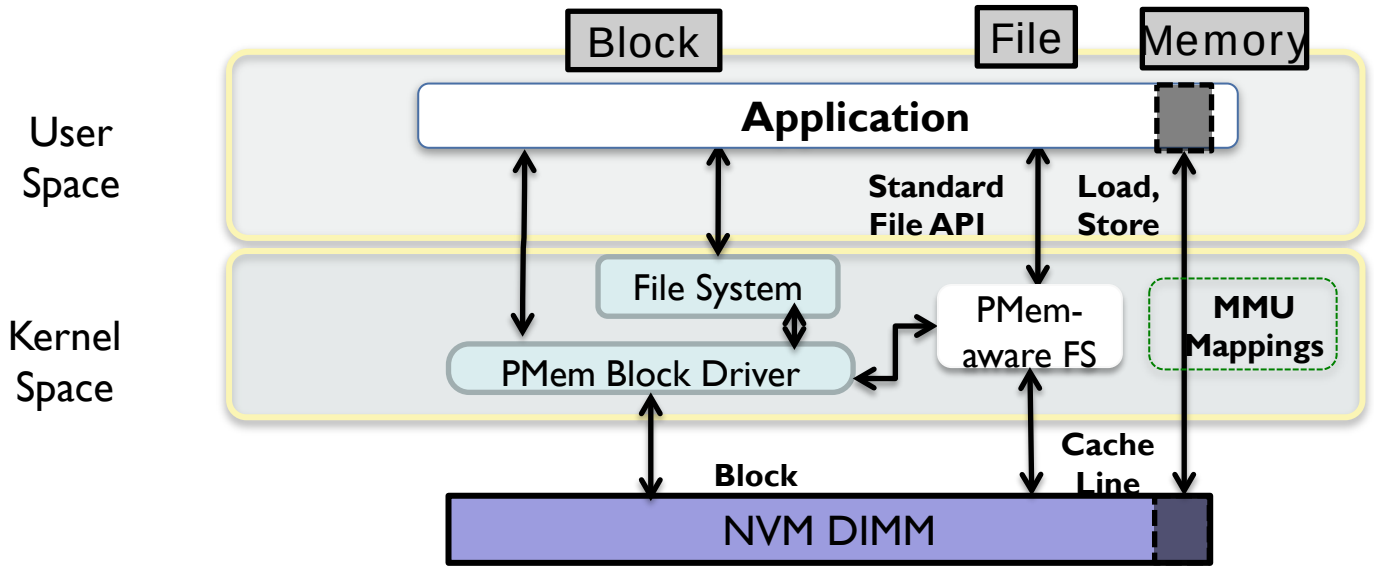
After (correct)

	Cache	PMem
var	I	I
persisted	True	True

New instructions CLFLUSHOPT and PCOMMIT required to make stores durable

Persistent Memory Aware Filesystems

- No buffering in DRAM on mmap → direct access to PMem
- Examples are PMFS (research), or ext4 and xfs + new "dax" mount (4.0 kernel onward)

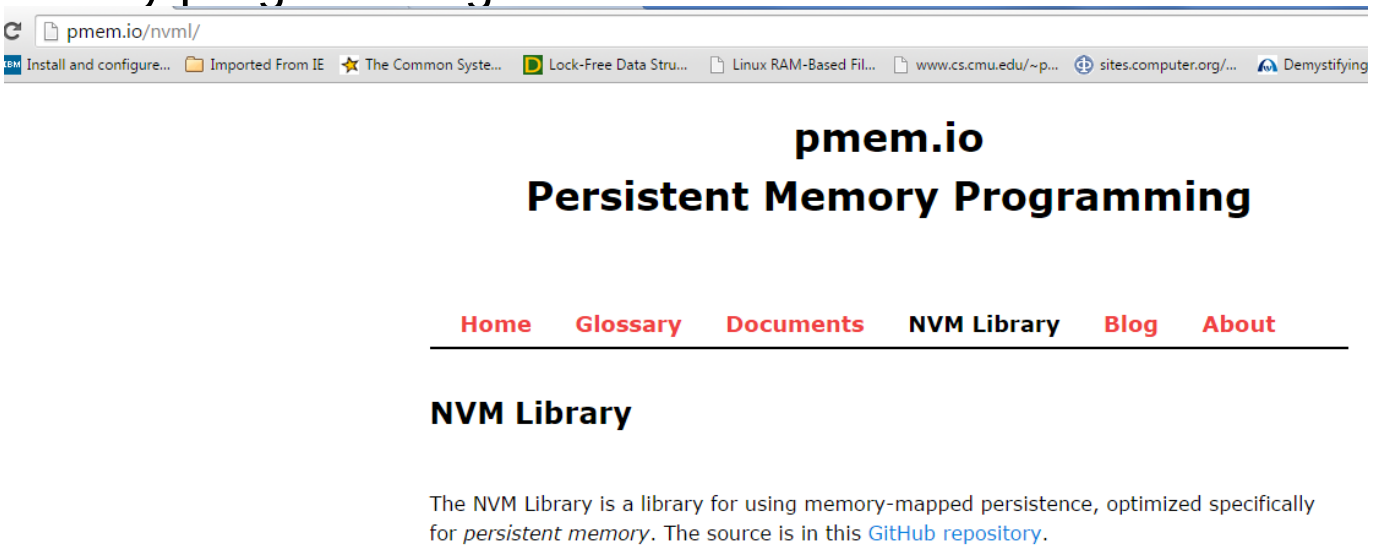


Byte addressable Access to PMem without kernel overhead for load/stores

ATTEND: Planning for the Next Decade of NVM Programming (Andy Rudoff)

Persistent Memory Programming

- Intel has released a set of open source persistent memory libraries
  - https://pmem.io/
- Example: libpmemobj provides transactional object store, providing memory allocation, transactions, and general facilities for persistent memory programming.



Website discusses library functions, caveats, suggestions on programming to Pmem

ATTEND: Solving the Challenges of Persistent Memory Programming (Sarah Jelinek)

Agenda

- Technology Differentiators
  - Benefits and caveats to persisting data structures
  - Why programming to persistent memory is different
- Case study: Design of a Persistent Memory Enabled Database
  - Design decisions
  - Evaluation methodology

Instant Recovery for Main-Memory Databases

Ismail Oukid*°, Wolfgang Lehner*, Thomas Kissinger*, Peter Bumbulis°,
and Thomas Willhalm +

*TU Dresden °SAP SE +Intel GmbH

CIDR 2015, California, USA,
January 5, 2015

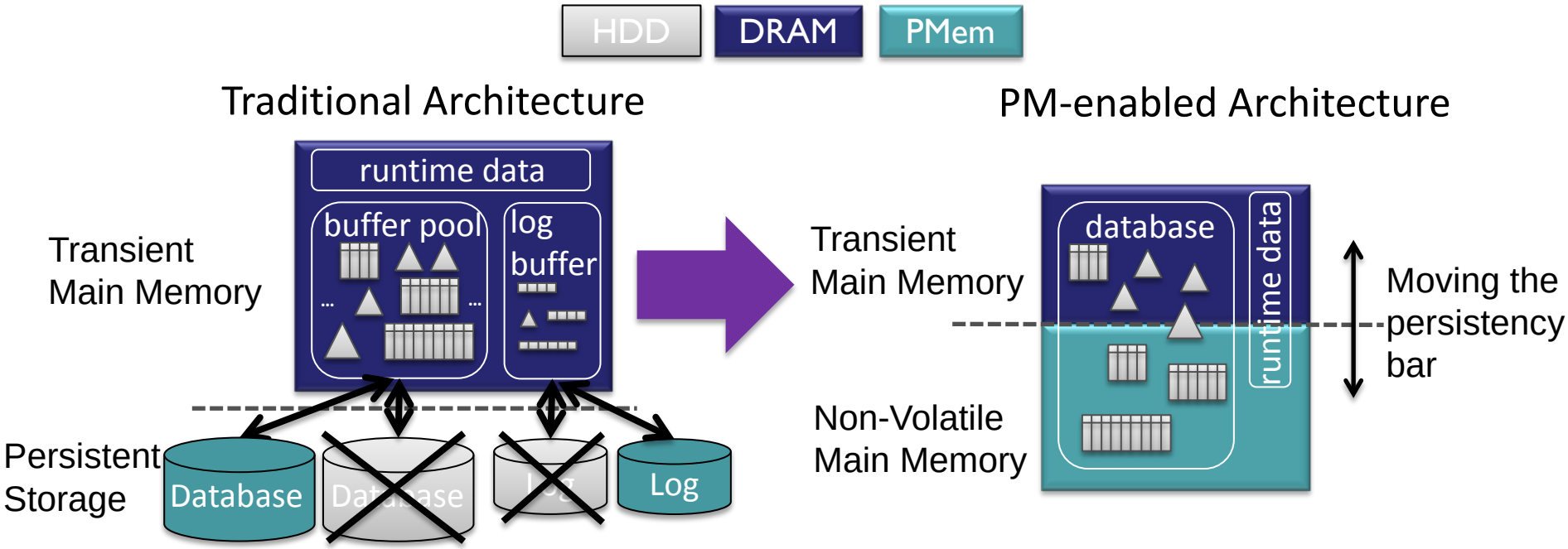
Design Considerations

Take full advantage of PMem (SCM)
to enable instant and point-in-time recovery

Eliminate data copy from storage	Eliminate log infrastructure	Dynamic decision making for secondary data structures
• Directly modify data in persistent memory	• Use concurrent and persistent data structures combined with concurrency scheme	• Use performance considerations to place secondary data structures in DRAM or PMem (SCM)

Three main design considerations for instant and point-in-time recovery

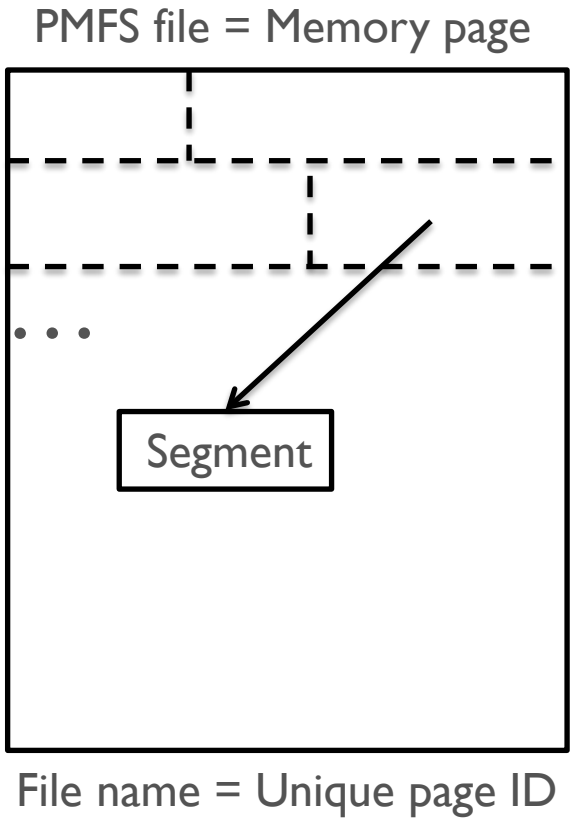
SOFORT: A PM-enabled architecture



SOFORT is a single-level column-store, i.e., the working copy is the durable copy

Implementation Consideration: PMAllocator

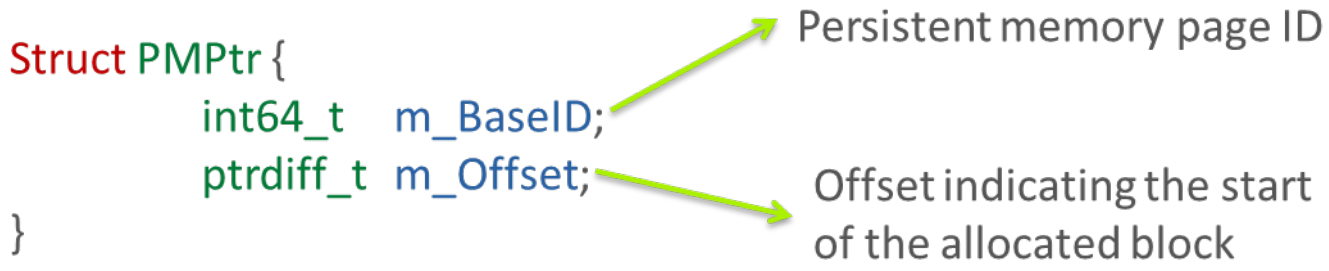
- PMAllocator:
  - Huge PMFS files as memory pages
  - Pages cut into segments for allocation
  - Persistent counter of memory pages
  - Mapping from persistent memory to virtual memory



New mechanisms for allocators required

Implementation Consideration: PMPtrs

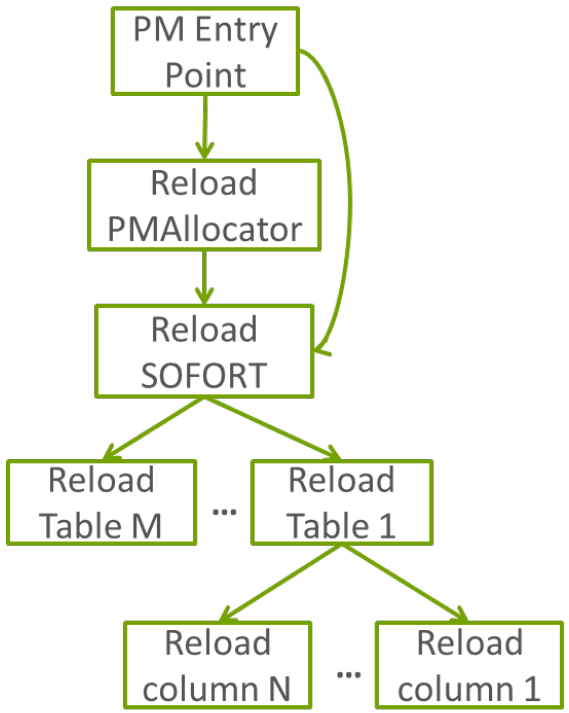
- Regular pointers are bound to the program‘s address space→ Cannot be used for recovery
- We propose persistent memory pointers



- PMPtrs can be converted (swizzled) to regular pointers and stay valid across failures

New mechanisms for handling pointers required

Implementation Consideration: Recovery Path



- A PMPtr to the PMAllocator
- A PMPtr to the storage engine

- Reload memory pages
- Reconstruct mapping to VM

- Rollback unfinished transactions
- Allocate ressources for recovery

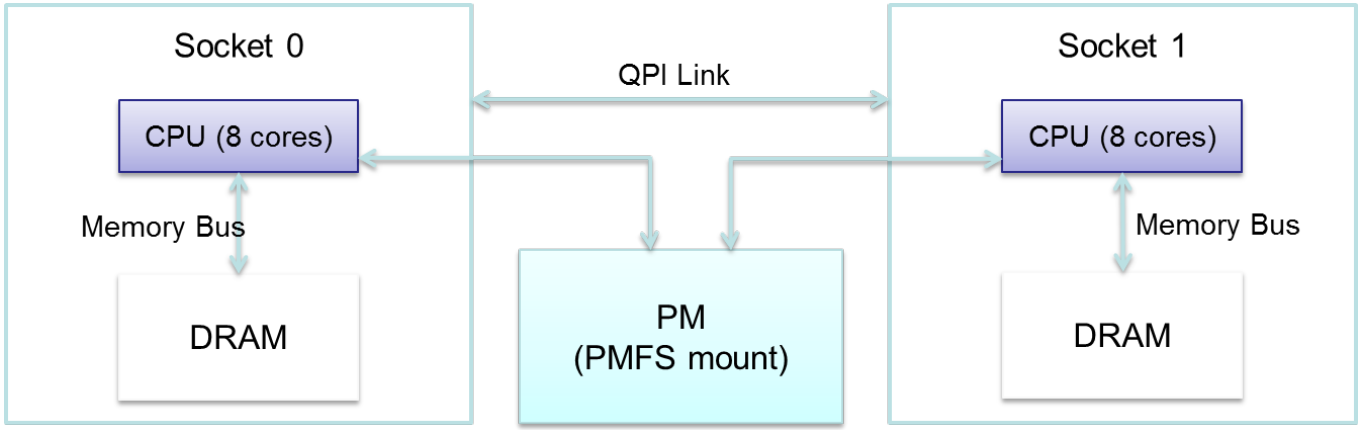
- Data structures sanity check
- Initiate columns recovery

- Data structures sanity check
- Reconstruct dictionary indexes

New mechanisms required to handle recovery path

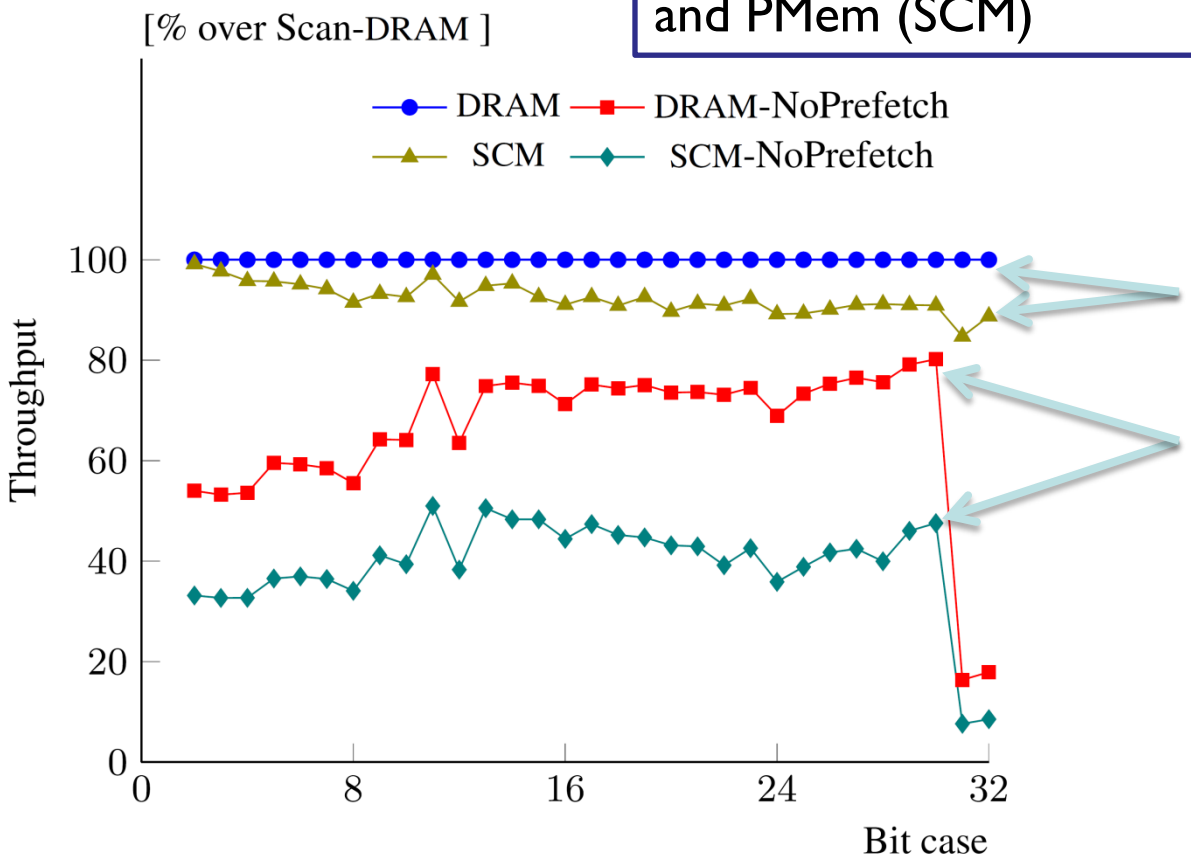
Evaluating Data Structure Latency Sensitivity

- **Hardware-based PMem simulation based on DRAM:**
  - Special BIOS, tunable DRAM latency with means of a microcode patch
  - Limitation: symmetric instead of asymmetric read/write latency
  - Avoiding NUMA effects: benchmark run on a single socket
  - **DRAM** Latency: **90ns**, **simulated PMem** latency: **200ns**



Evaluating Data Structure Latency Sensitivity

SIMD-Scan performance on DRAM and PMem (SCM)

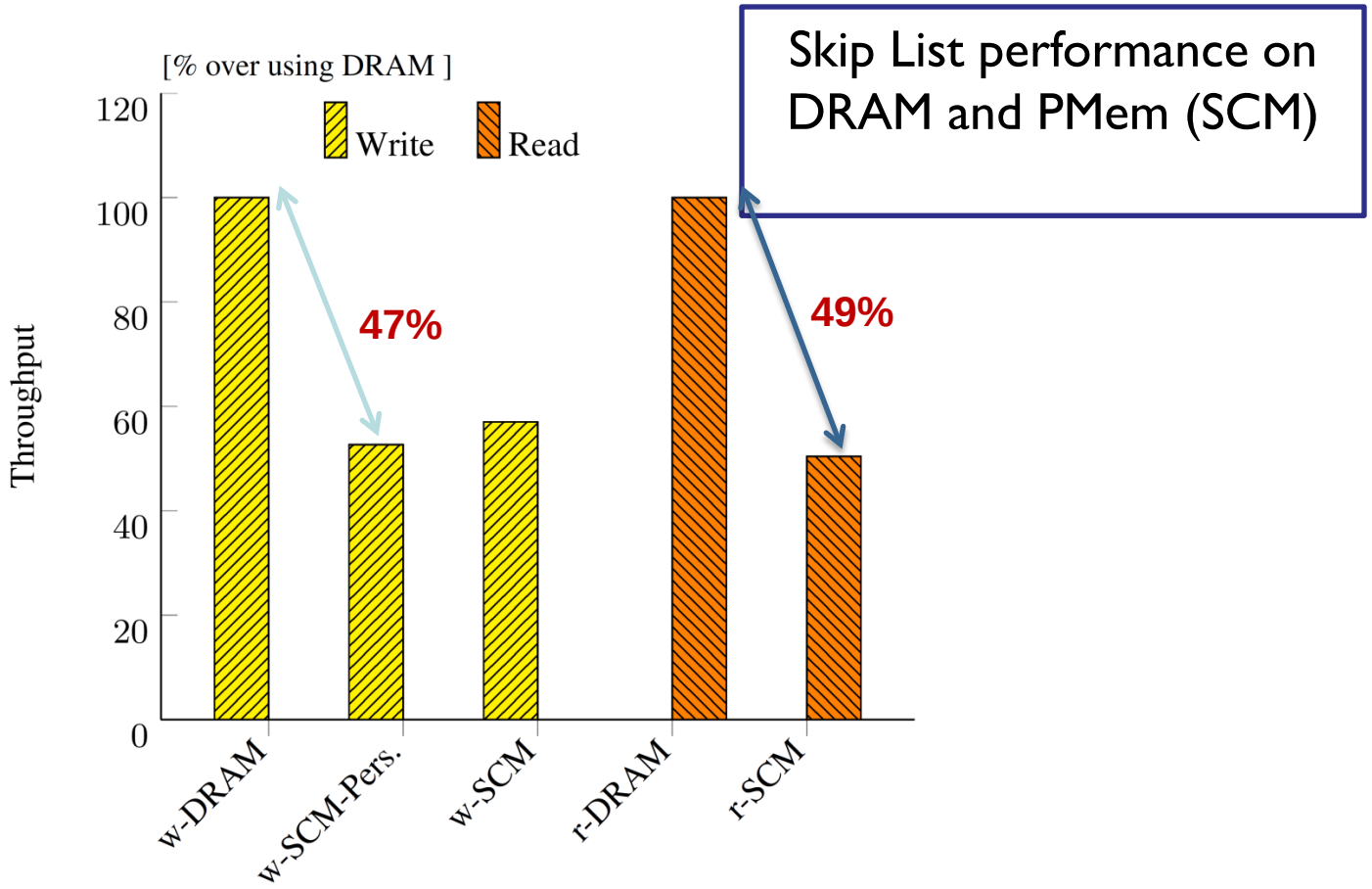


8% average slowdown

41% average slowdown

Workloads with sequential memory access patterns perform well on PMem (SCM)

Evaluating Data Structure Latency Sensitivity



Workloads with random memory access patterns do not perform well on SCM: We still need DRAM

Measuring the Value of Persistence

Recovery Area	Recovery response time	Recovery delta
• Maximum number of transactions that could have been executed if the database had not failed	• Average query response time during the recovery process	• Time it takes to achieve the pre-failure throughput

New metrics to quantify how persistence helps database recovery

Improving Recovery Performance

Synchronous Recovery

- Step 1: Recovery memory management
- Step 2: Recover primary data
- Step 3: Continue unfinished statements
- Step 4: Rebuild secondary data structures on DRAM
- Step 5: Start accepting user queries

Primary data already "loaded"

Restart time depends on the size of secondary data structures to be rebuilt

Instant Recovery

- Idea 1:

Use primary data to answer queries and rebuild secondary data structures asynchronously

Instant responsiveness

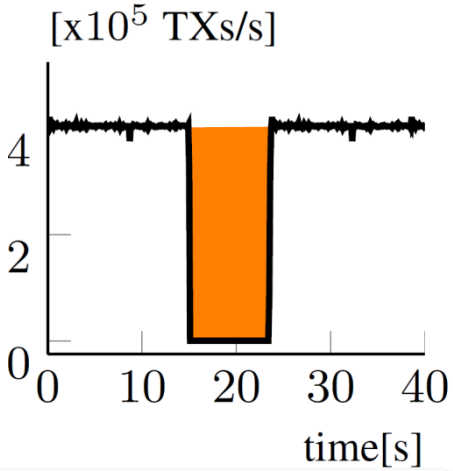
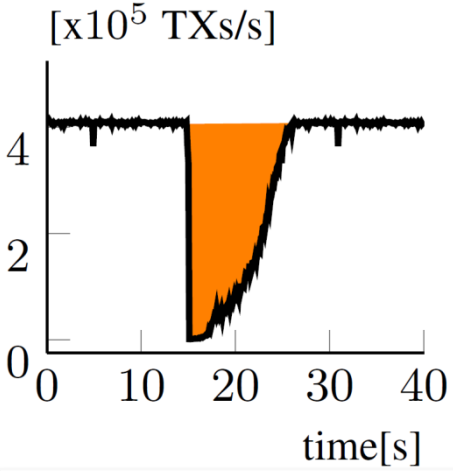
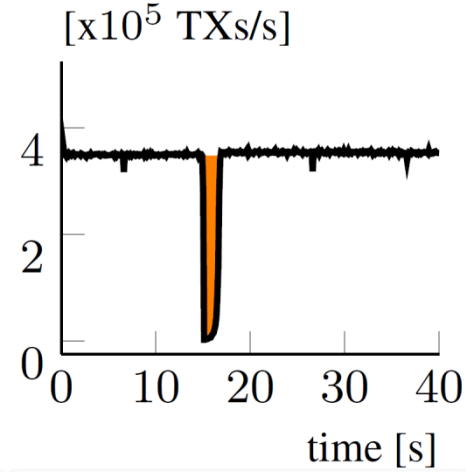
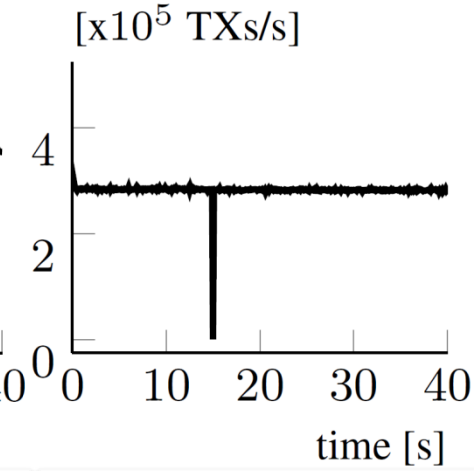
- Idea 2:

Persist part of or all secondary data structures in PMem (SCM)

Instant recovery at peak performance

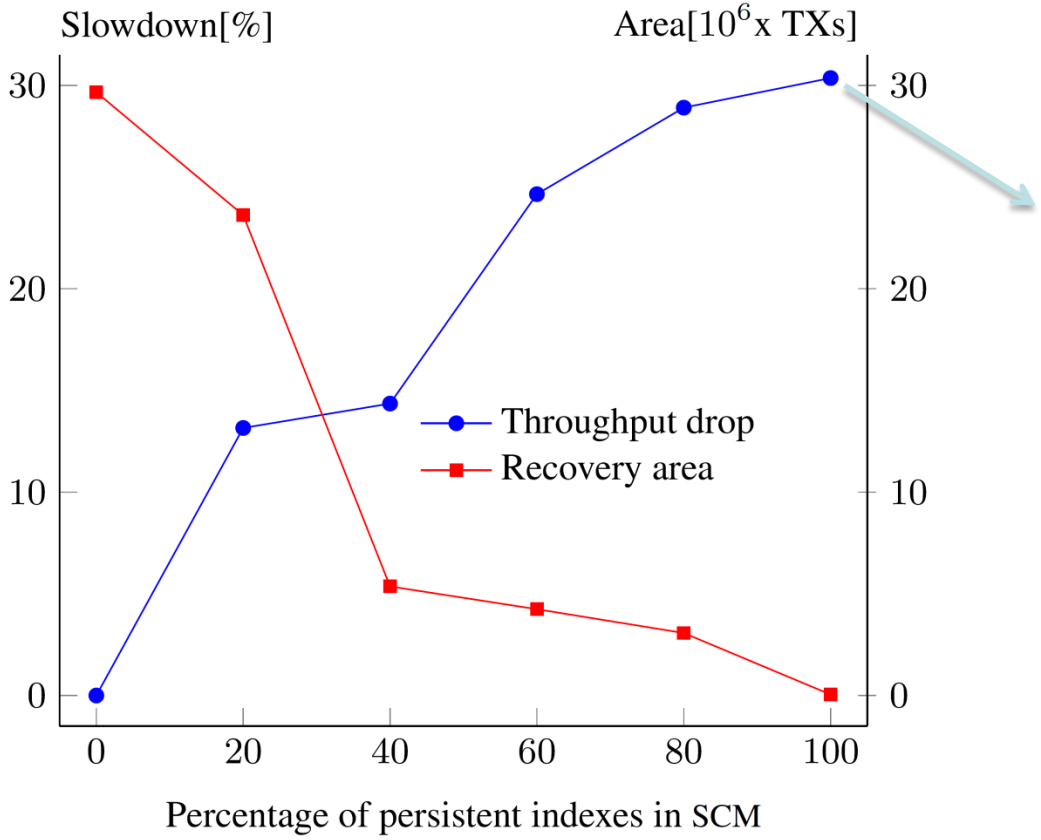
Performance Penalty on throughput

Evaluation: Recovery Time

Synchronous Recovery	Instant Recovery		
	0% indexes in SCM	40% indexes in SCM	100% indexes in SCM
			
First query accepted after **~8s,** i.e., Recovery delta = **8s**	Throughput: **-0%** Recovery area: **-16%** Recovery delta: **~8s**	Throughput: **-14%** Recovery area: **-82%** Recovery delta: **<2s**	Throughput: **-30%** Recovery area: **-99,8%** Recovery delta: **<5ms**

Different type of tradeoffs possible between throughput and recovery metrics

Evaluation: Throughput Vs. Recovery



Throughput drop limited to 30%

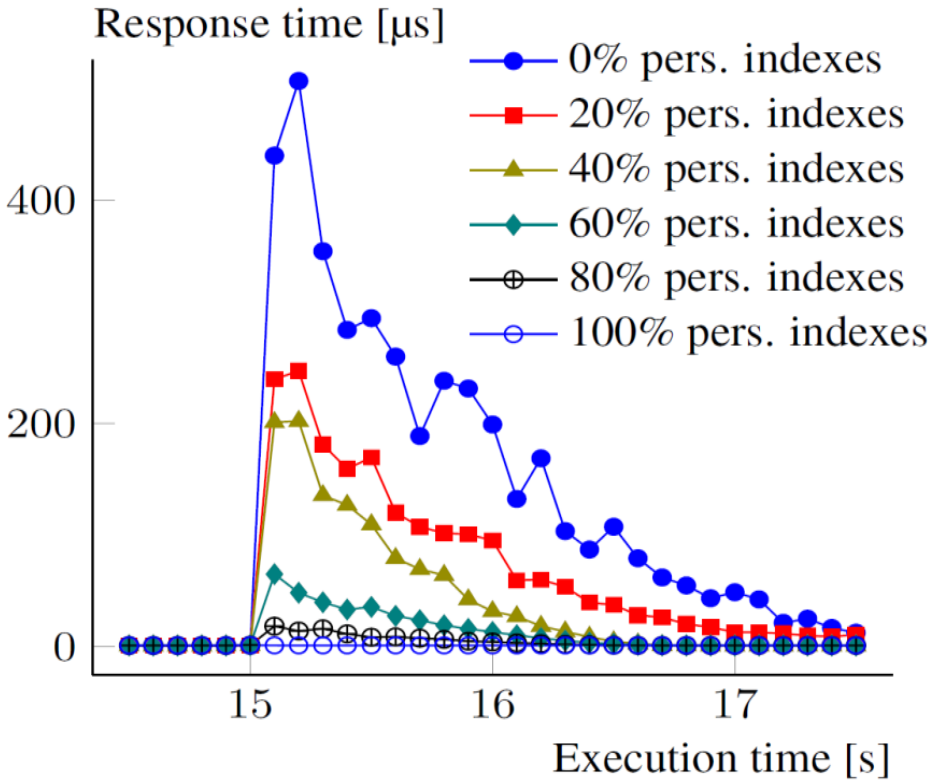
Curves are not linear: secondary data structures are not equally important for TATP

Taking advantage of a workload's characteristics leads to an optimal tradeoff

Takeaways

- Persistent Memory offers the game-changing ability to improve restart and recovery time, and improve capacity
- Design process involves deciding which data structures to persist
- Moving a data structure to PMem avoids the need to load from disk on restart altogether
- Tools, Libraries, Test platforms available

Evaluation: Average Response Time



Max. avg. (over 100ms) Response time:

- 0% pers. indexes: **506µs**
- 100% pers. indexes: **2µs**

Seek tradeoff depending on: throughput requirements, response time requirements, and desired recovery performance