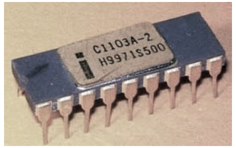
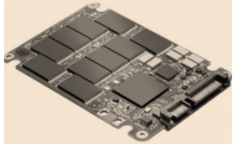
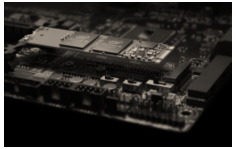
DATA MANAGEMENT ON NON-VOLATILE MEMORY

JOY ARULRAJ
Carnegie Mellon University

EVOLUTION OF MEMORY TECHNOLOGY







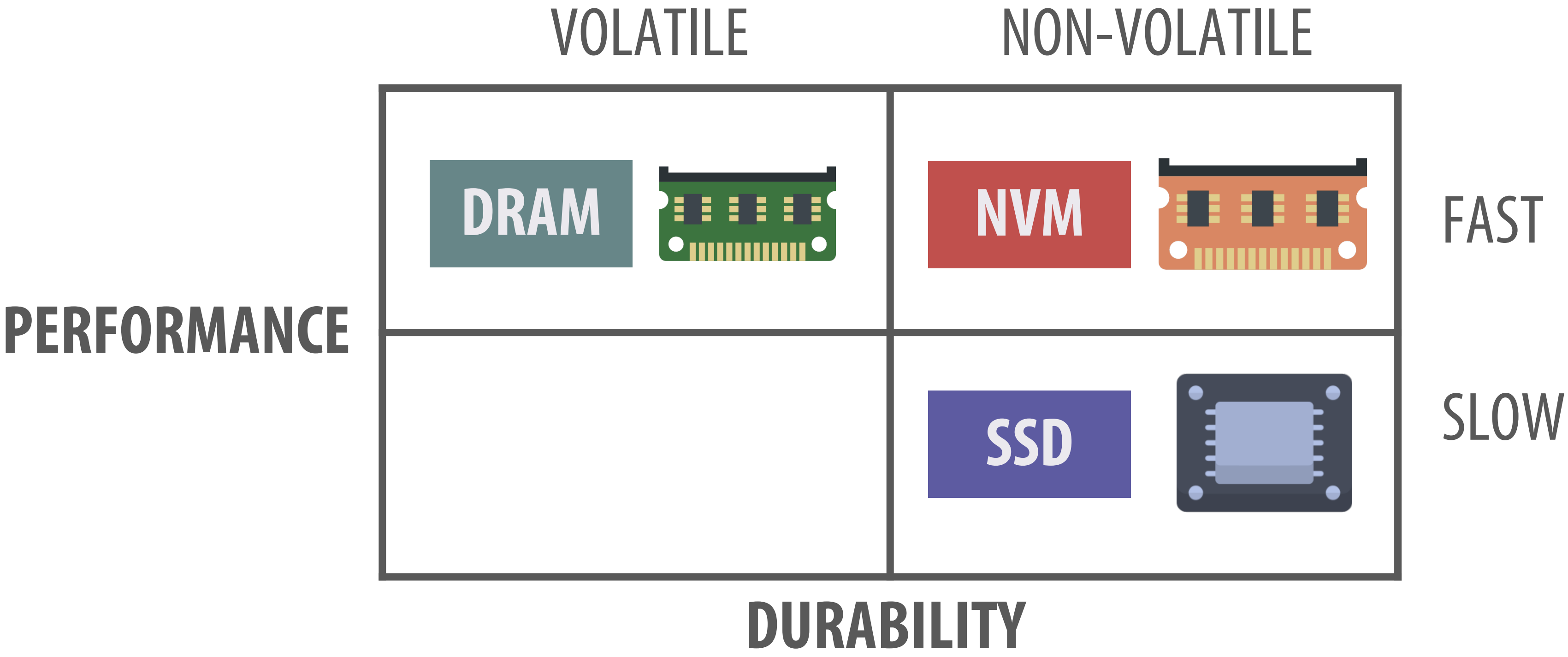
1960 1970 1980 1990 2000 2010 2020

DRAM

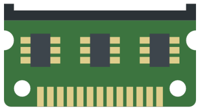
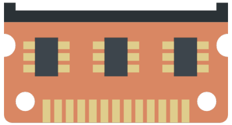
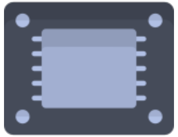
FLASH MEMORY

NON-VOLATILE MEMORY

NON-VOLATILE MEMORY [NVM]



DEVICE CHARACTERISTICS

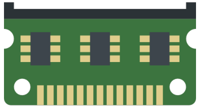




CHARACTERISTIC	DRAM	NVM	SSD
Device Latency	1x	10x	1000x
Byte-Addressability	✓	✓	✗
Durability	✗	✓	✓
High Capacity	✗	✓	✓
Cost/GB	100x	10x	1x

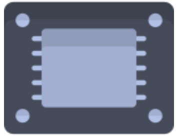
50 YEARS OF DATABASE SYSTEMS RESEARCH

1 SEPARATE MEMORY & STORAGE

DRAM

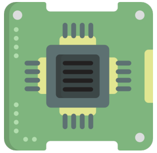


SSD

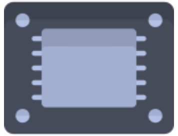


2 COMPUTE/STORAGE BALANCE

CPU

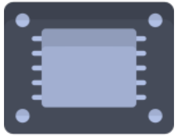


SSD



3 RANDOM VS. SEQUENTIAL

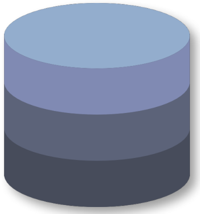
SSD



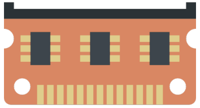
RESEARCH OVERVIEW

- How to manage data on NVM?
  - *Challenging because of NVM's unique characteristics*
  - *Important given the sudden shift in compute/storage balance*

DATABASE SYSTEM



NVM



#1: INDUSTRY STANDARDS

- Standardization of NVM technologies
  - *Design standards*
  - *Interface specifications*

JEDEC®

nvm EXPRESS

JUNE 2016

#2: OPERATING SYSTEM SUPPORT

- Major operating systems natively support NVM
  - *Linux 4.8*
  - *Windows 10*

Linux™ Windows 10

AUGUST 2015

#3: ARCHITECTURAL SUPPORT

- New assembly instructions in ISA updates
  - *Efficiently flush data from volatile CPU cache to NVM*
  - *Kaby Lake processor*

MAY 2016

HOW DO TODAY'S DATABASE SYSTEMS PERFORM ON NON-VOLATILE MEMORY?

NVM HARDWARE EMULATOR [INTEL]

- Emulates a wide range of NVM technologies
  - *Special CPU microcode*
  - *Supports recently added assembly instructions*

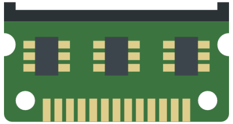
TODAY'S DATABASE SYSTEMS ON NVM

- Database System: MySQL
- Storage device performance
  - *NVM's performance compared to that of disk*
  - *I/O benchmark*
- Database system performance
  - *On NVM compared to that on disk*
  - *TPC-C benchmark*

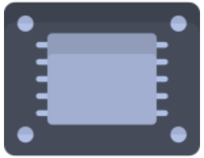
STORAGE HIERARCHIES

1 DISK-BASED HIERARCHY

DRAM

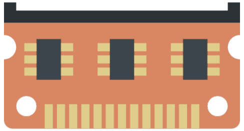


SSD



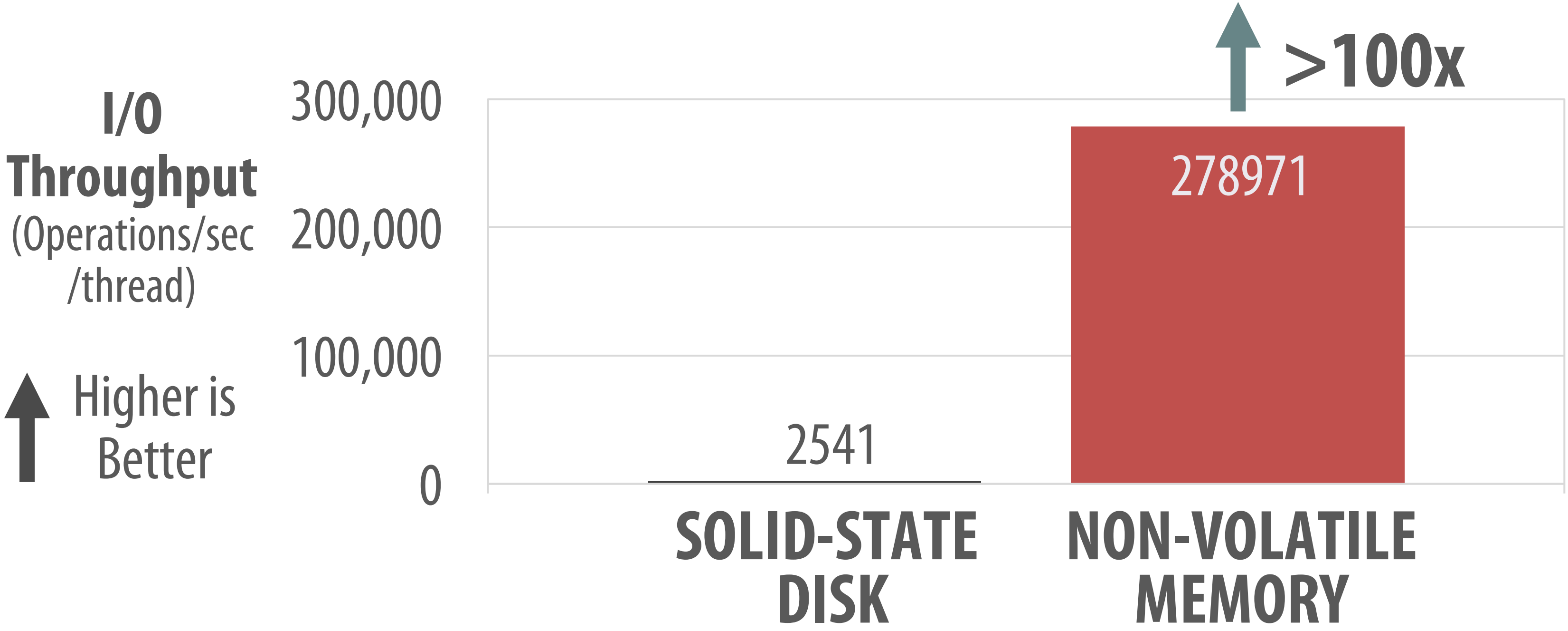
2 NVM-BASED HIERARCHY

NVM

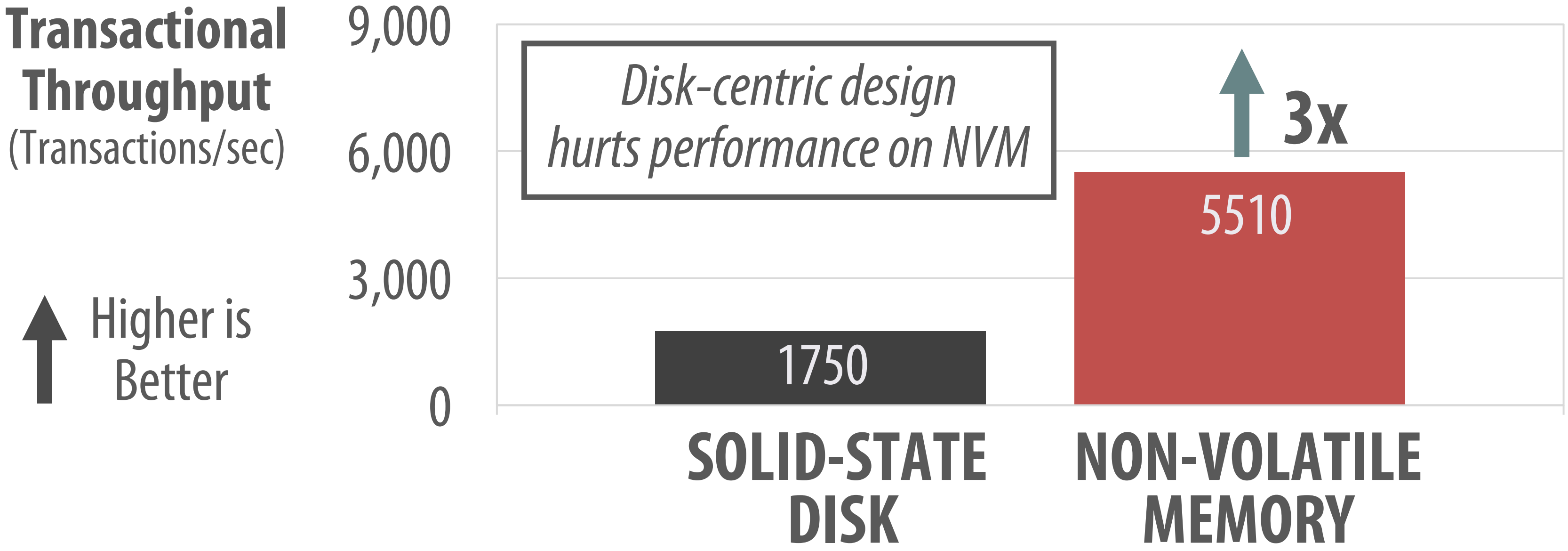


BOTH MEMORY & STORAGE

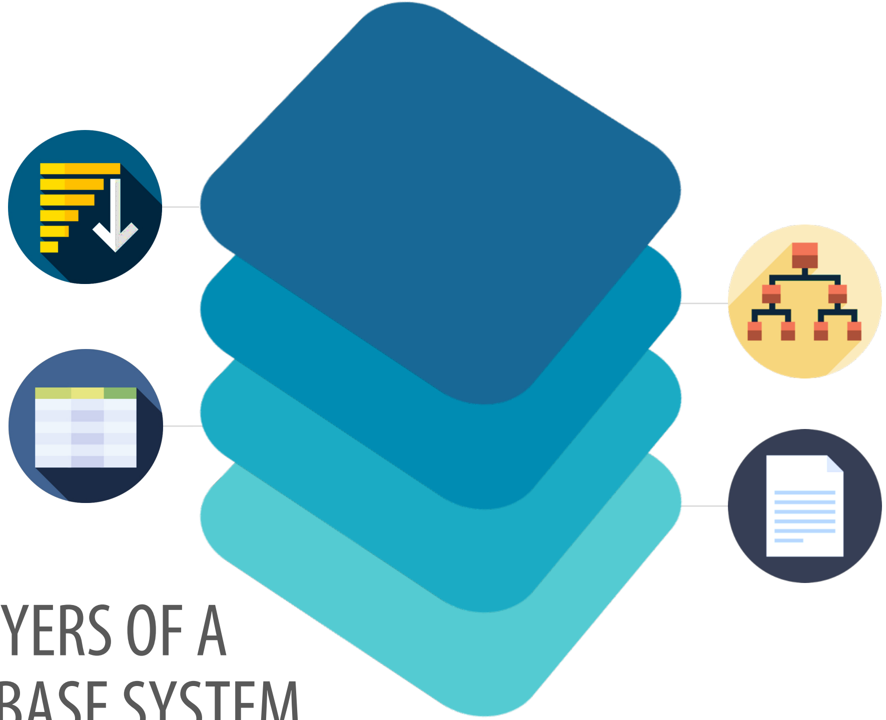
STORAGE DEVICE PERFORMANCE



DATABASE SYSTEM PERFORMANCE

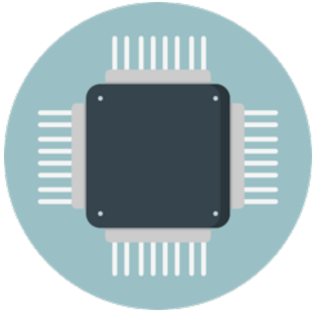


PELOTON NVM DATABASE SYSTEM



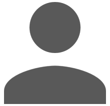
WRITE-BEHIND LOGGING

BZTREE INDEX



FUTURE DIRECTIONS

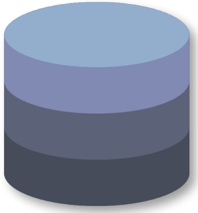
LOGGING & RECOVERY: MOTIVATION



SHOPPING APPLICATION

BEGIN TRANSACTION
Add Order
Update Stock
COMMIT TRANSACTION

DATABASE SYSTEM



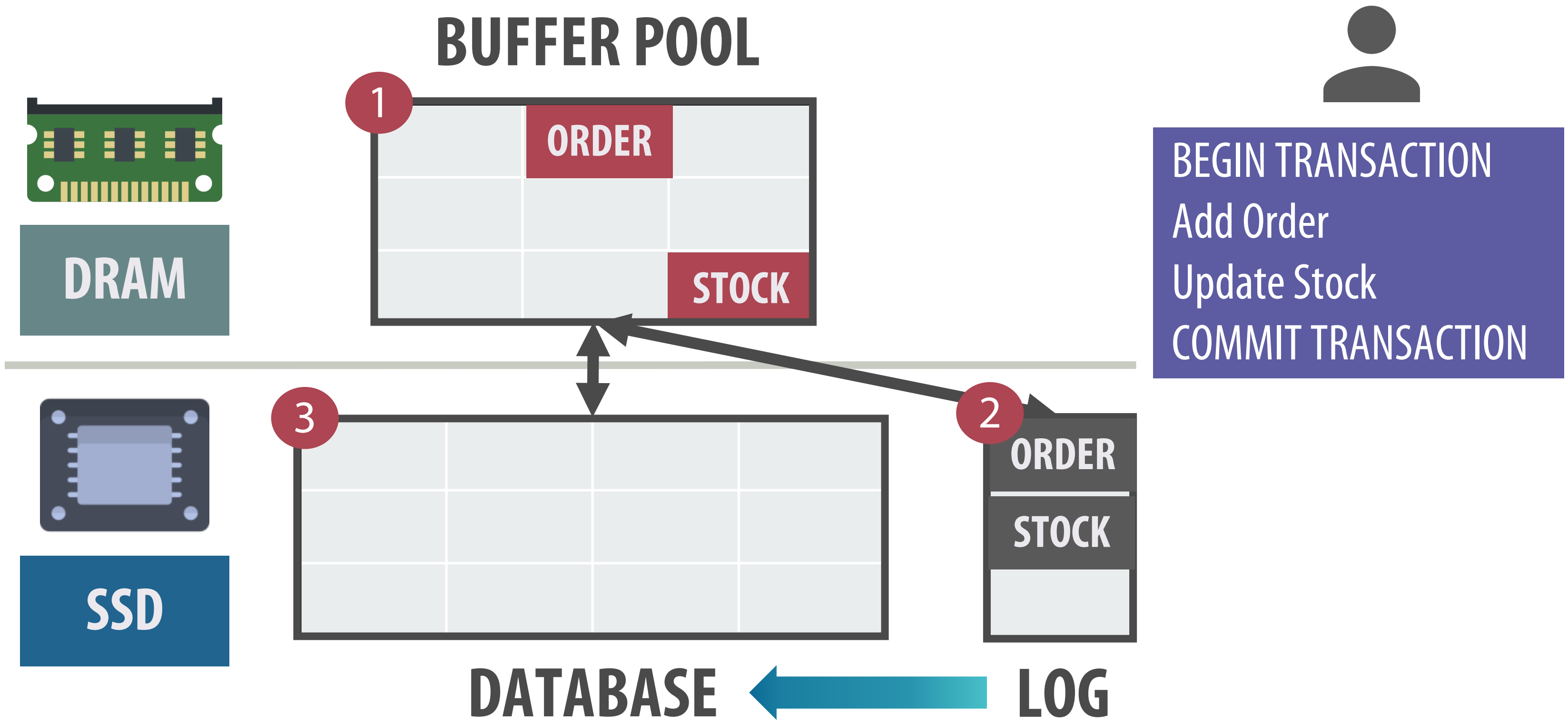
DURABILITY

FAULT-TOLERANCE

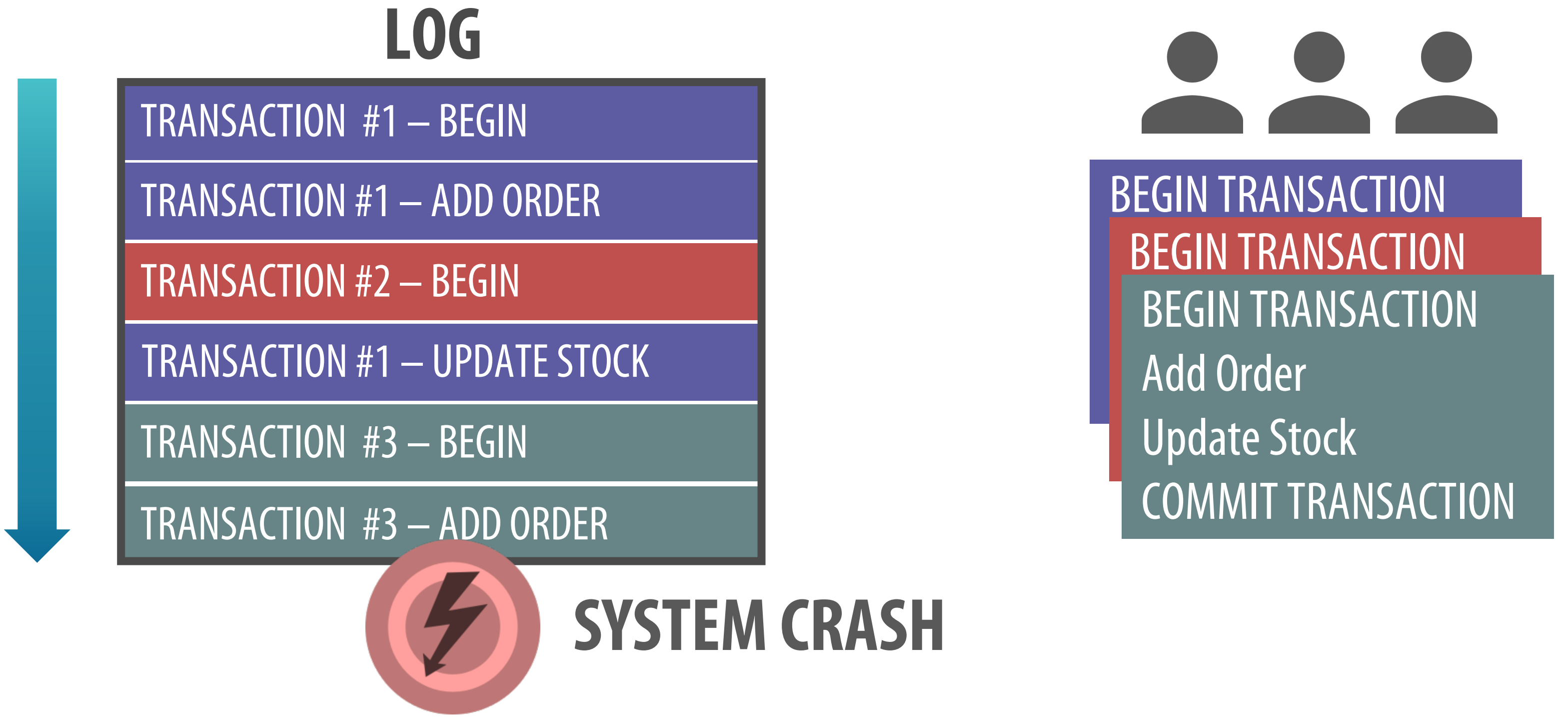
ATOMICITY

ALL-OR-NOTHING

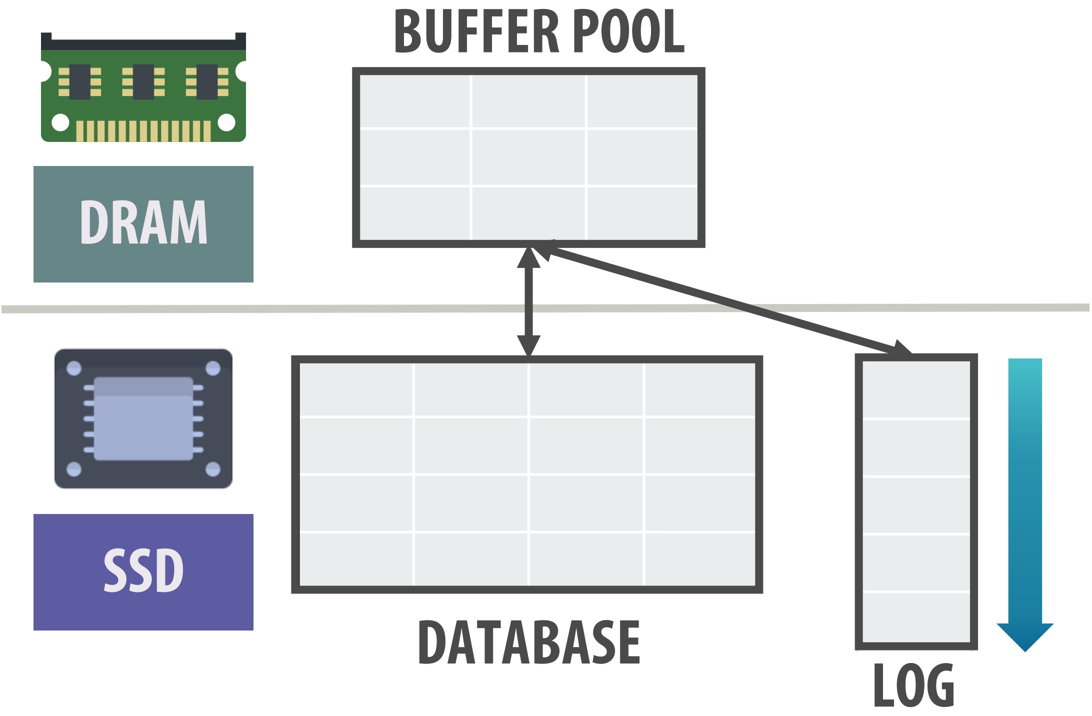
WRITE-AHEAD LOGGING: DURABILITY



WRITE-AHEAD LOGGING: ATOMICITY



WRITE-AHEAD LOGGING: RECOVERY PROTOCOL



③ UNDO CHANGES

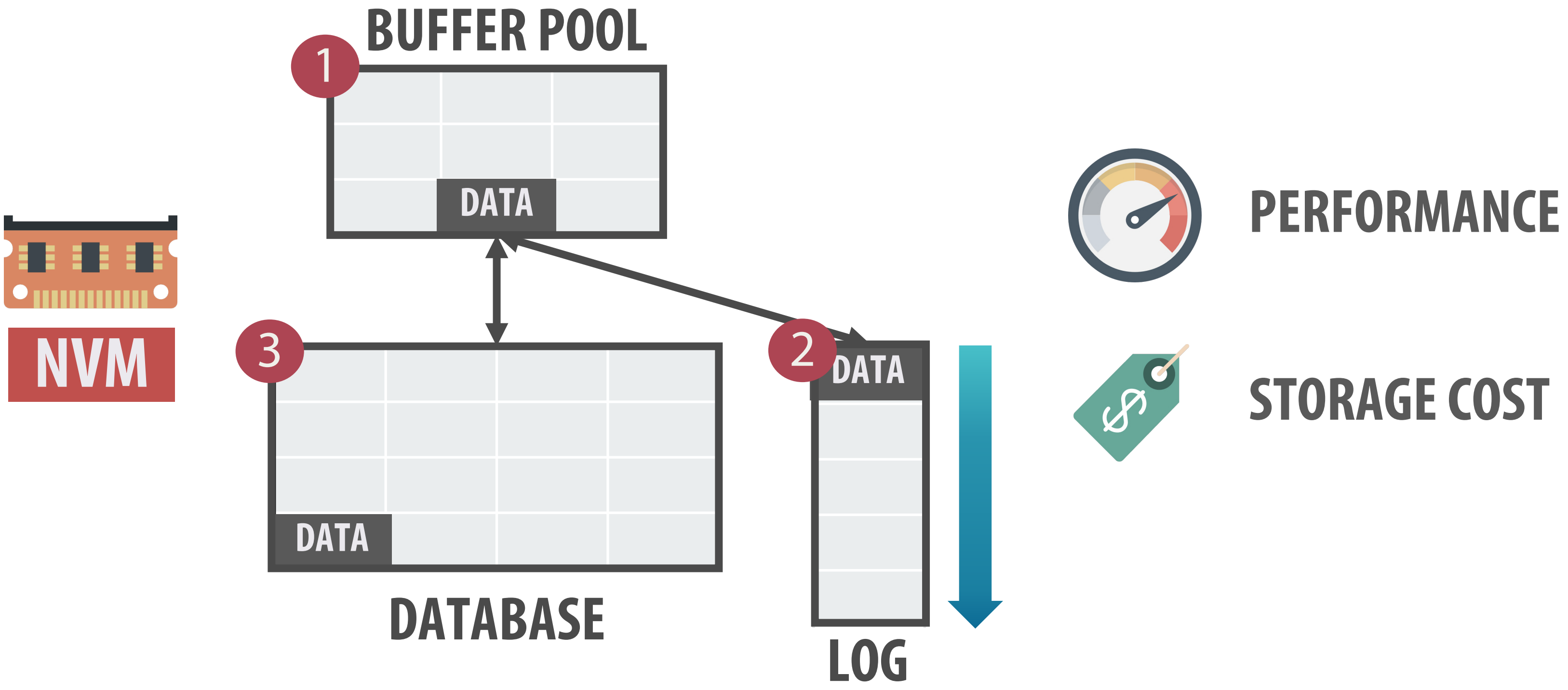
ACTIVE TRANSACTION TABLE

TXN ID	STATUS	LATEST CHANGE
TXN #2	RUNNING	LOG RECORD #7
TXN #3	RUNNING	---

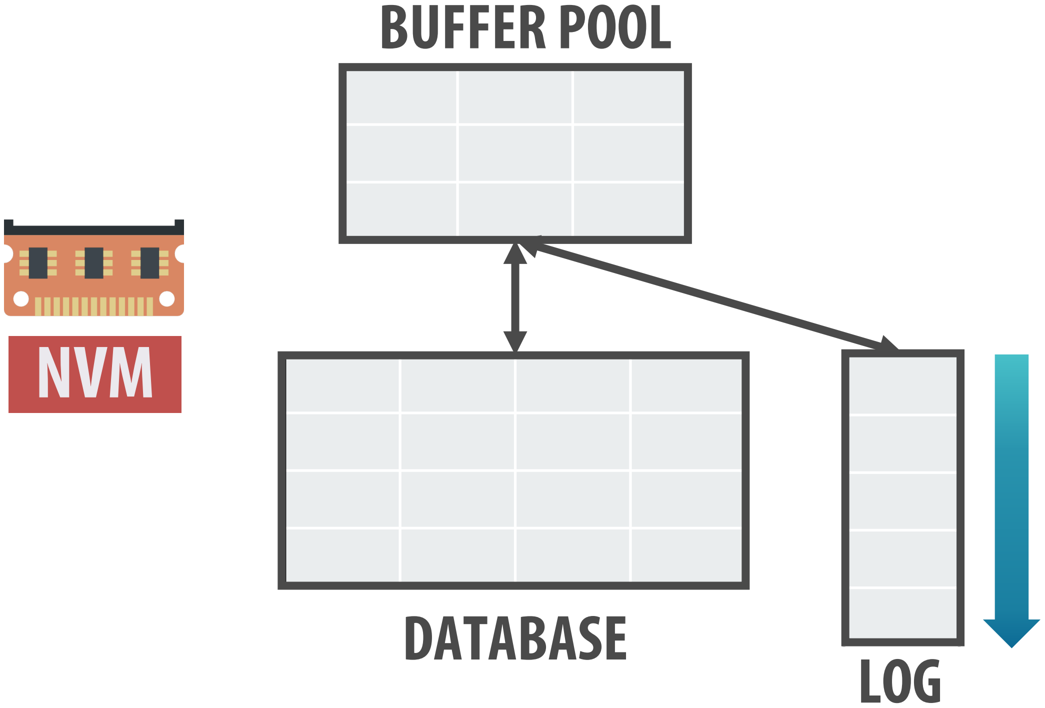
DIRTY PAGE TABLE

PAGE ID	CHANGE THAT DIRTIED PAGE
PAGE #30	LOG RECORD #5
PAGE #40	LOG RECORD #7

PROBLEM #1: DATA DUPLICATION



PROBLEM #2: SLOW RECOVERY



AVAILABILITY

LINEAR-TIME RECOVERY

NEEDS TO REDO LOG

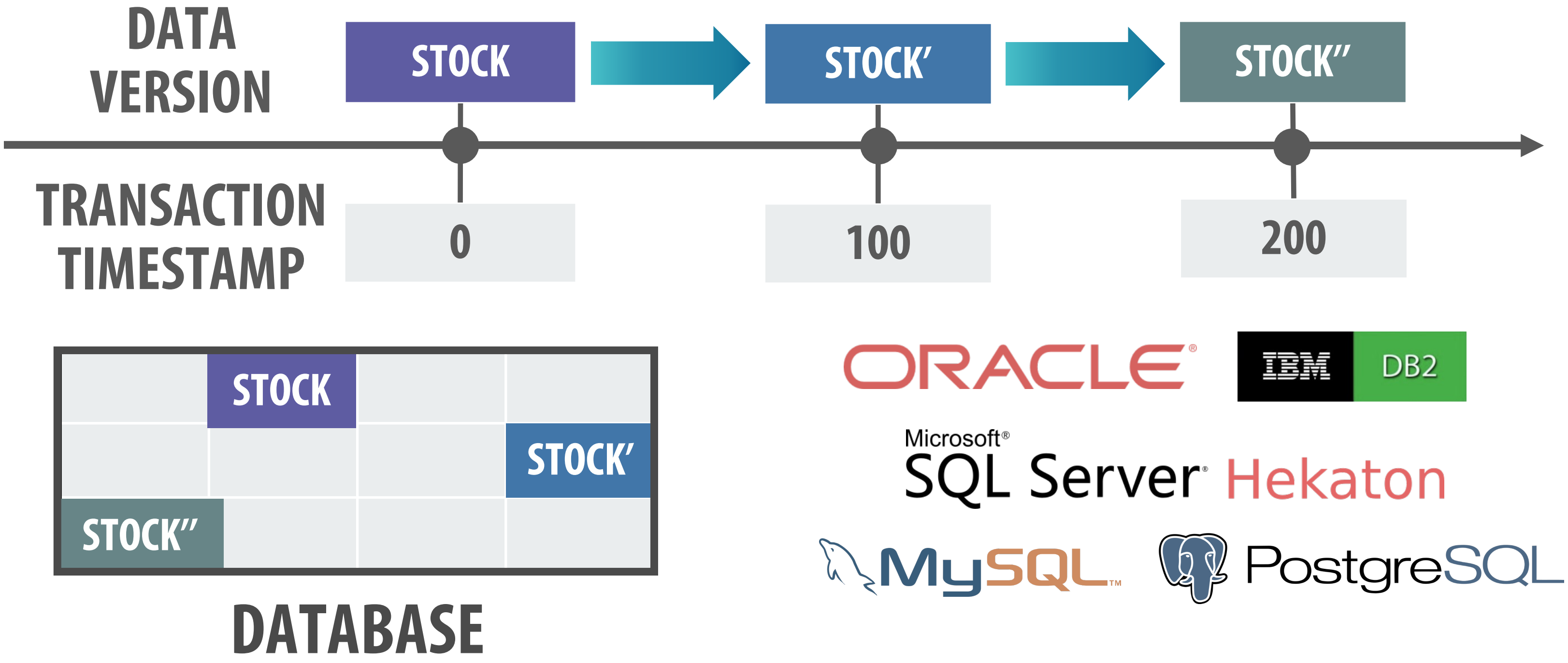
HOW TO IMPROVE PERFORMANCE AND AVAILABILTY ON NON-VOLATILE MEMORY?

WRITE-BEHIND LOGGING
VLDB 2017

WRITE-BEHIND LOGGING: OVERVIEW

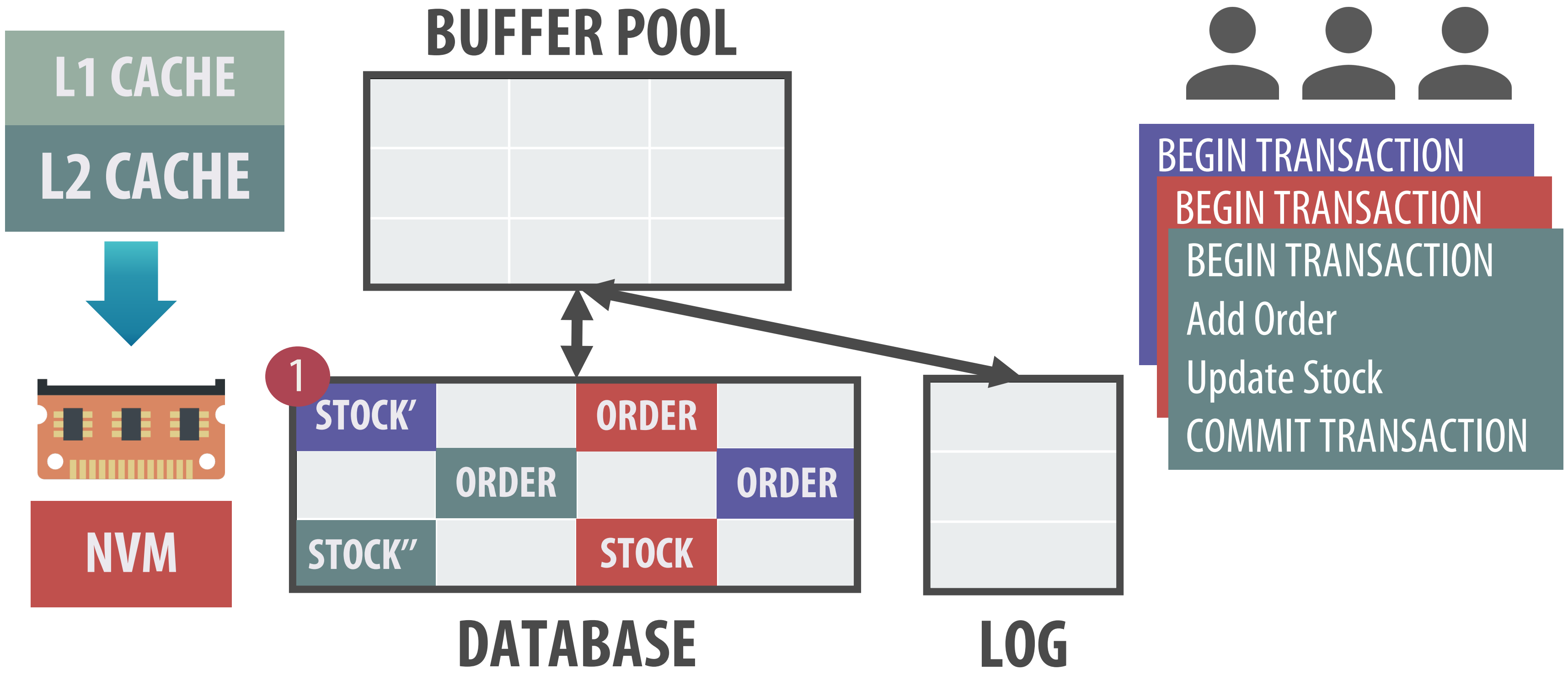
- NVM-centric design
  - *Improves availability by enabling instant recovery*
  - *Provides same guarantees as write-ahead logging*
- Key techniques
  - *Directly propagate changes to the database*
  - *Only record meta-data in log*

DATA VERSIONING USING TIMESTAMPS

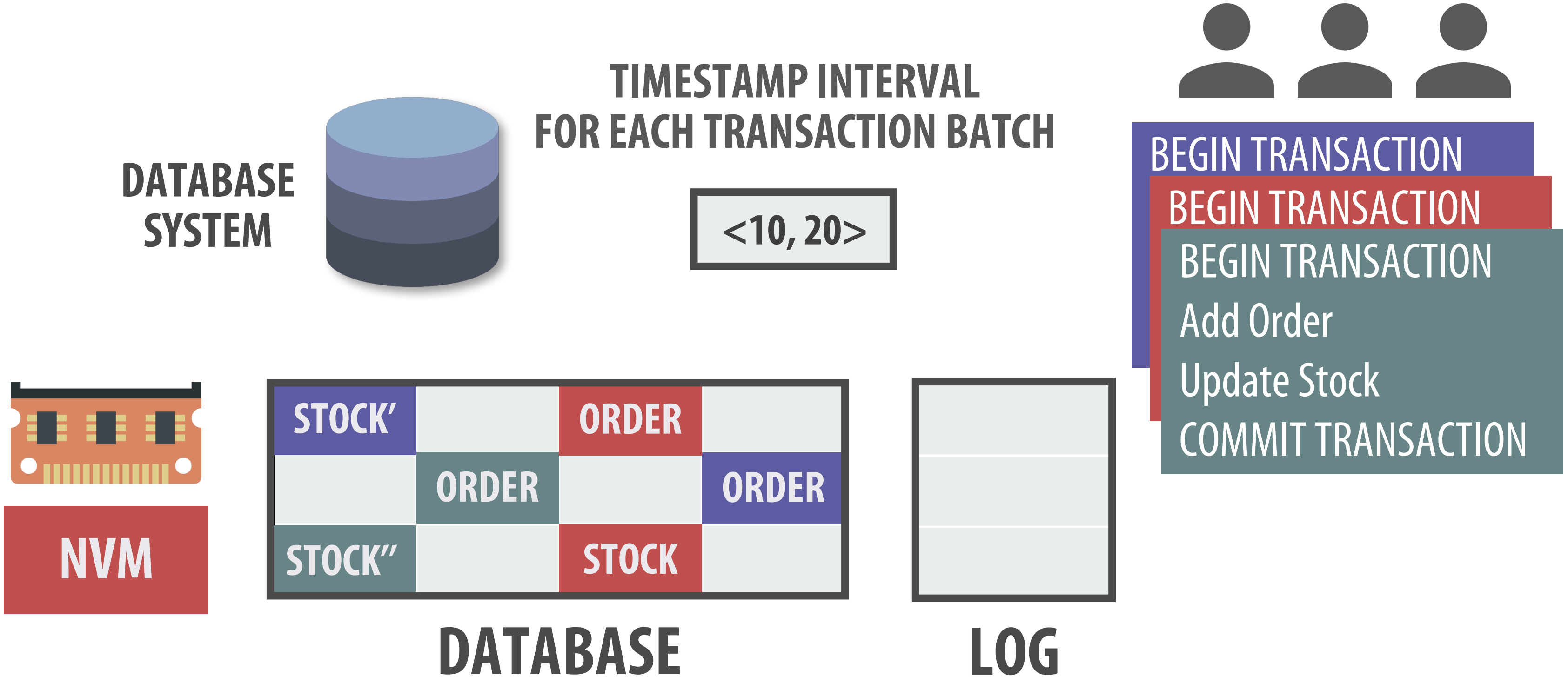


ORACLE
IBM DB2
Microsoft SQL Server Hekaton
MySQL
PostgreSQL

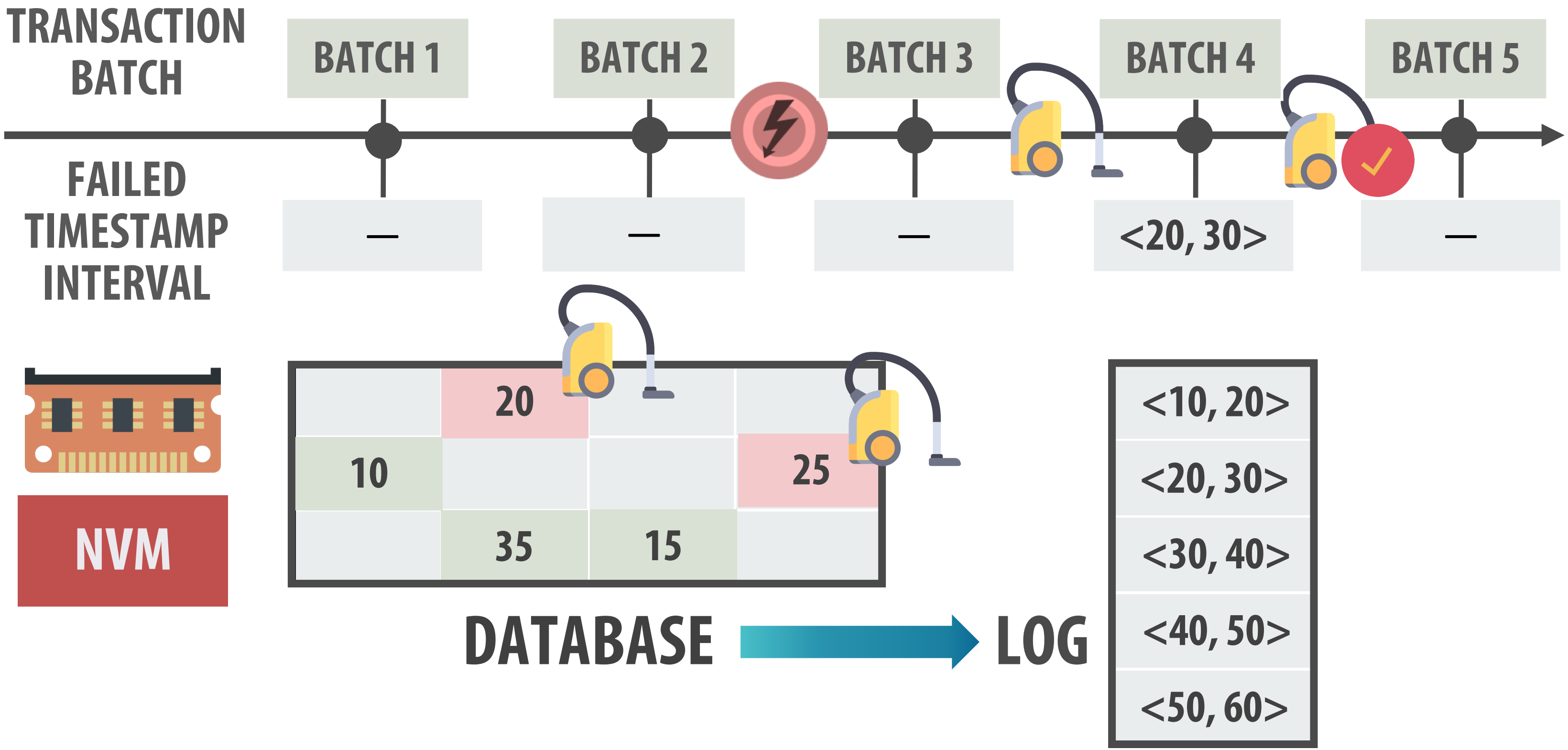
WRITE-BEHIND LOGGING: DURABILITY



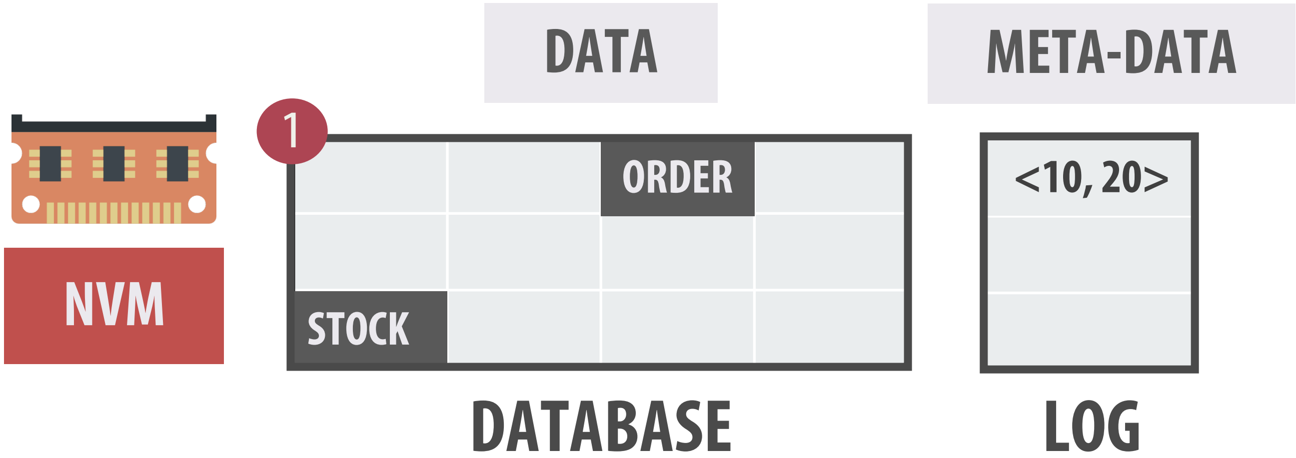
WRITE-BEHIND LOGGING: ATOMICITY



WRITE-BEHIND LOGGING: RECOVERY PROTOCOL



SOLUTION #1: NO DATA DUPLICATION



PERFORMANCE

STORAGE COST

SOLUTION #2: INSTANT RECOVERY

WRITE-AHEAD LOGGING | Linear-Time Recovery

1 ANALYSIS → 2 REDO → 3 UNDO

WRITE-BEHIND LOGGING | Constant-Time Recovery

1 ANALYSIS

AVAILABILITY

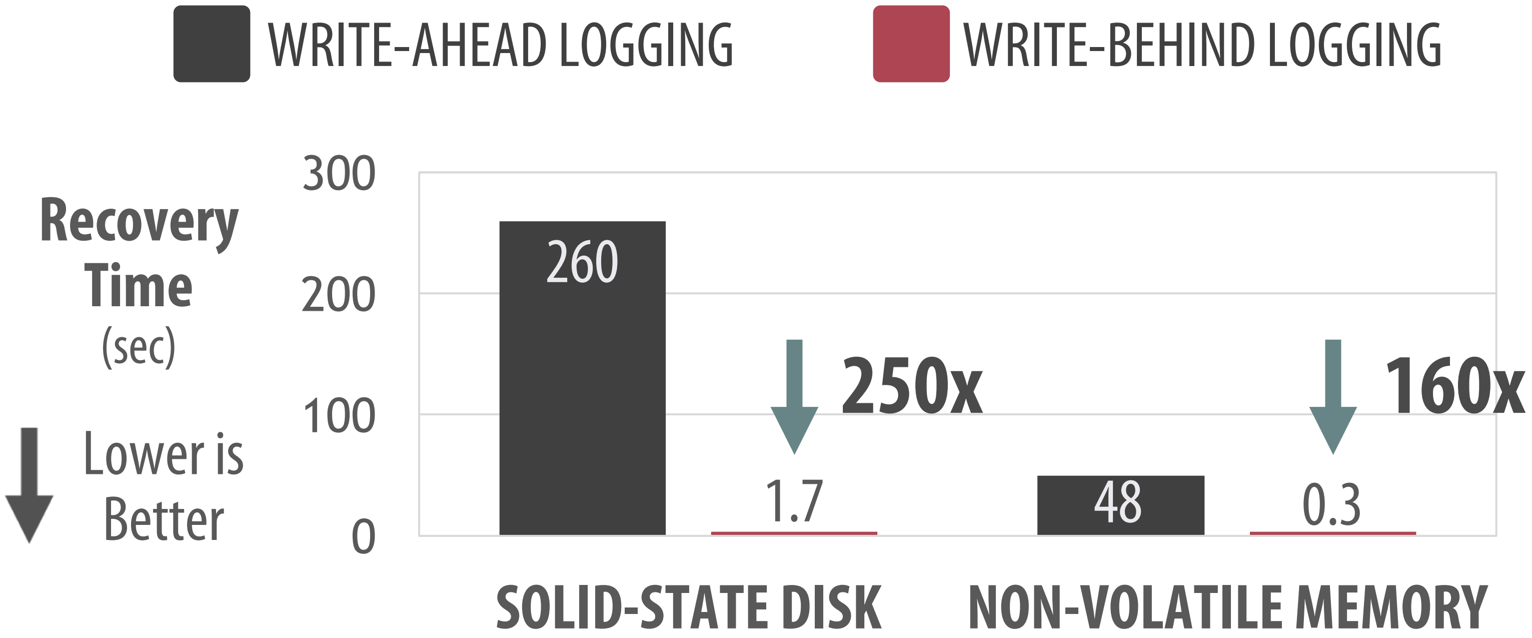
WRITE-BEHIND LOGGING

- Enables instant recovery from failures
- Eliminates data duplication
- Generalizes to single-versioned database systems
- Supports a multi-tier storage hierarchy
- Handles long lived transactions
- Copes with failures during recovery

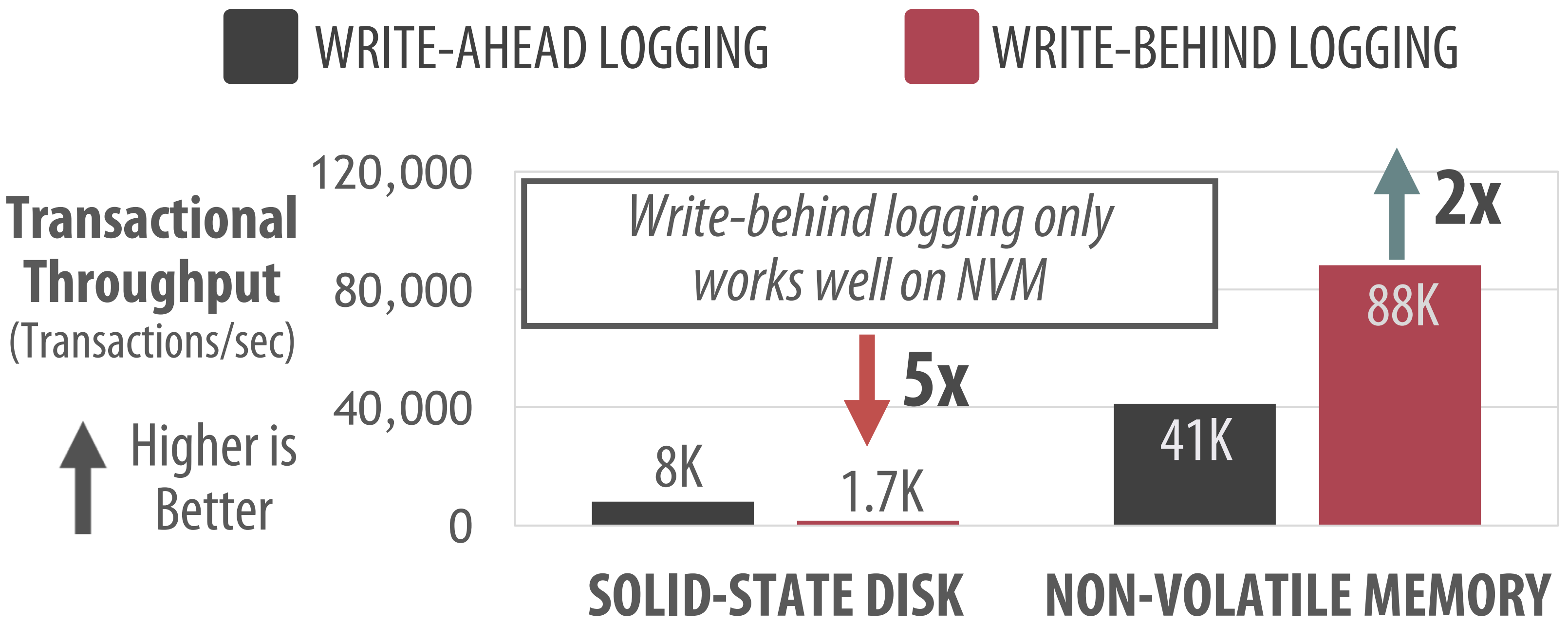
EVALUATION

- Logging Protocols: Write-Behind vs. Write-Ahead Logging
  - *Recovery Time*
  - *Database System Performance*
- Workload: TPC-C benchmark on Peloton
- Storage devices
  - *Solid-state disk*
  - *Non-volatile memory*

RECOVERY TIME



DATABASE SYSTEM PERFORMANCE



WRITE-BEHIND LOGGING: SUMMARY

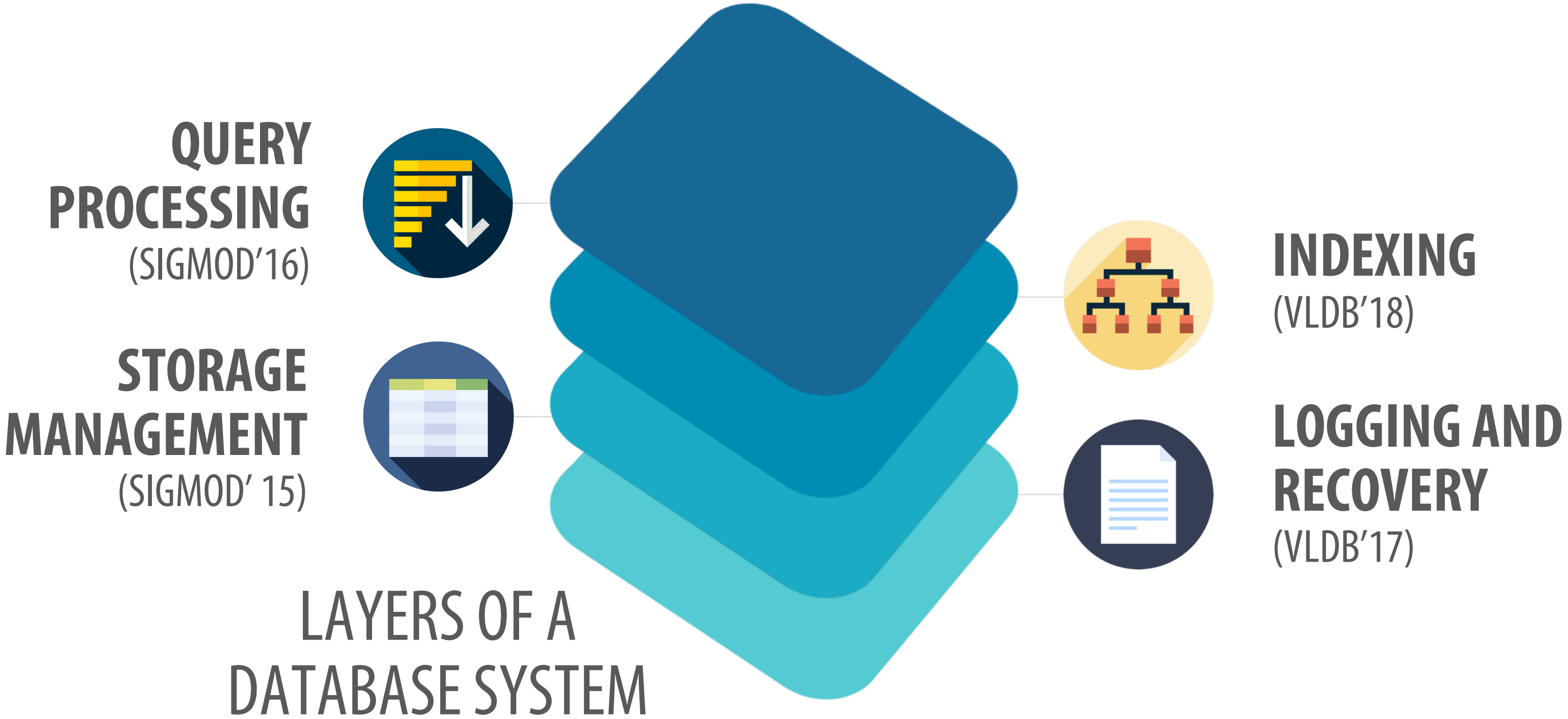
AVAILABILITY

PERFORMANCE

STORAGE COST

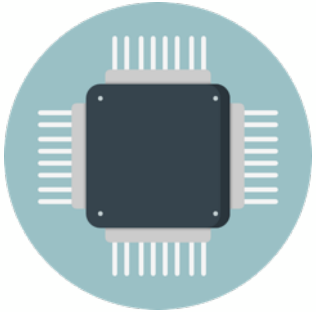
Advances the state of the art by shifting the complexity class of the recovery protocol on NVM

PELOTON NVM DATABASE SYSTEM



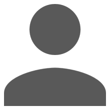
WRITE-BEHIND LOGGING

BZTREE INDEX



FUTURE DIRECTIONS

INDEXING DATA: MOTIVATION

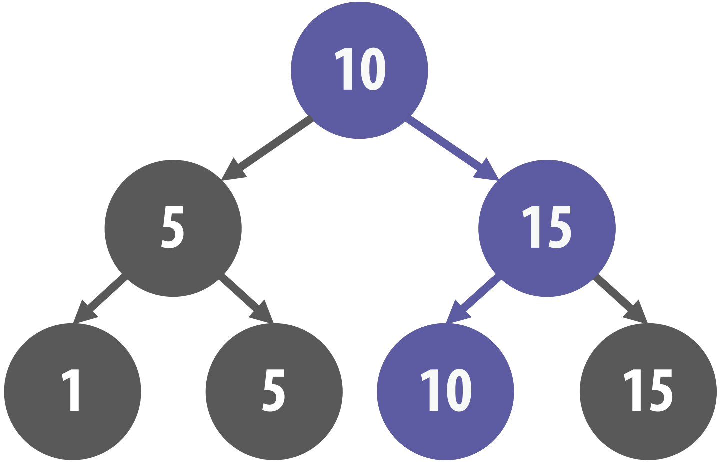


SHOPPING APPLICATION

BEGIN TRANSACTION
Update Stock By Stock ID
COMMIT TRANSACTION

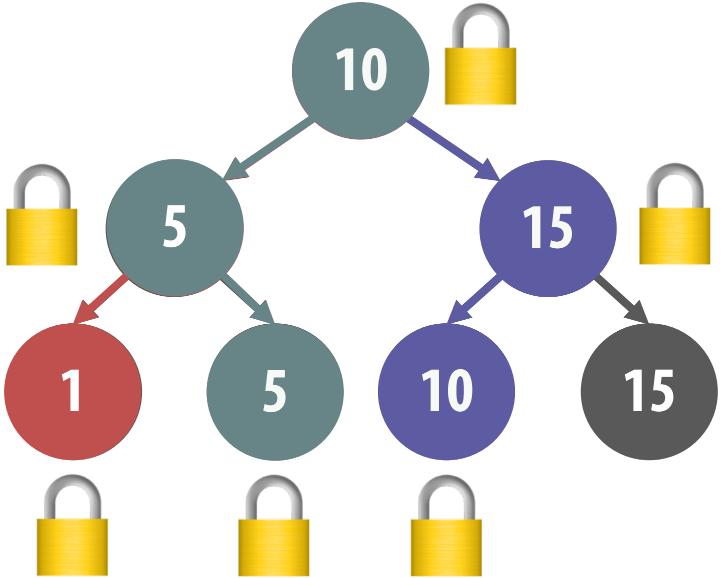
DATABASE SYSTEM

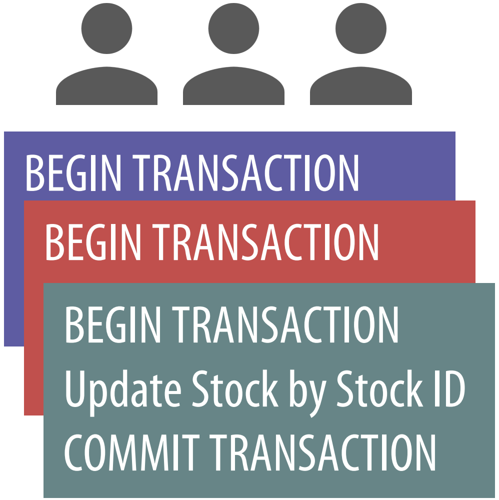
STOCK ID > 5 AND < 15



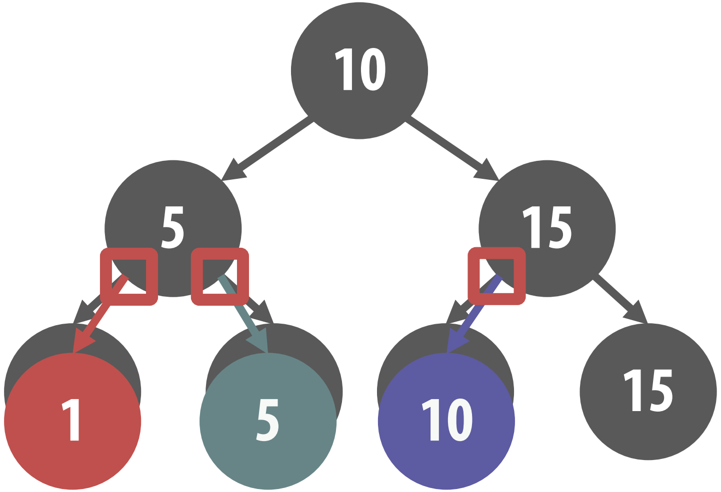
STOCK INDEX

SYNCHRONIZATION WITH LOCKS

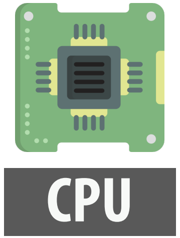




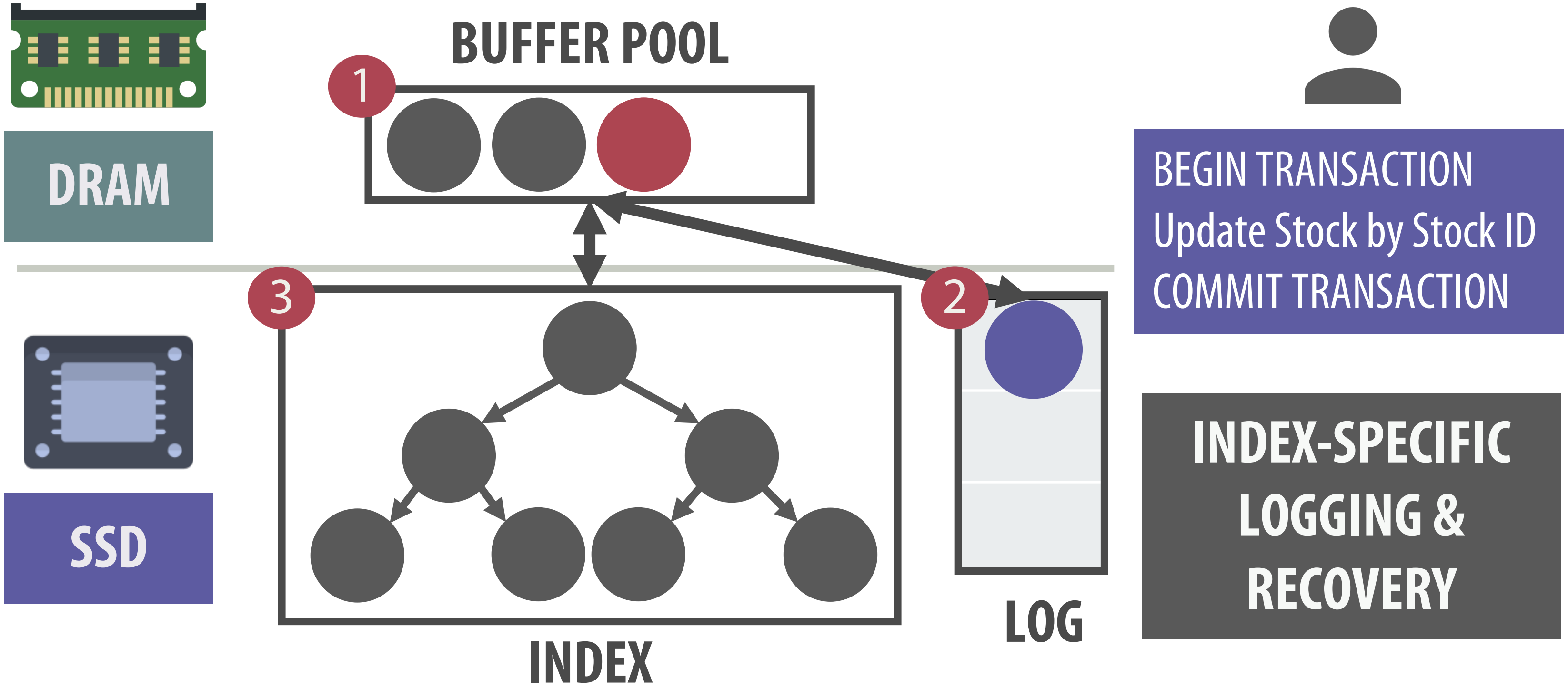
BWTREE: LOCK-FREE B+TREE [MICROSOFT]



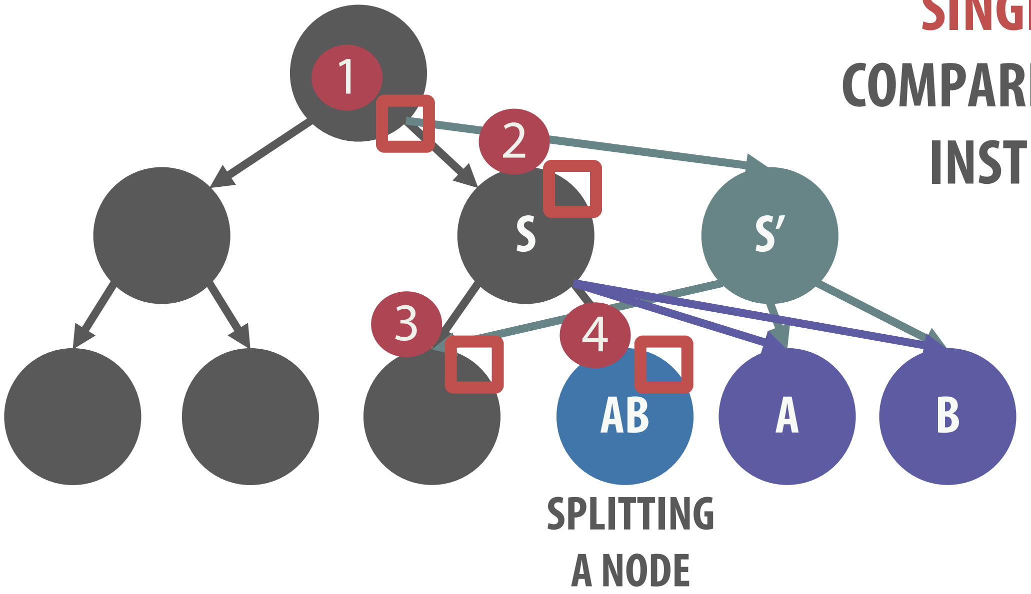
SINGLE-WORD
COMPARE-AND-SWAP
INSTRUCTION



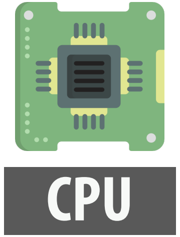
BWTREE: DURABILITY & ATOMICITY



PROBLEM #1: HIGH CODE COMPLEXITY



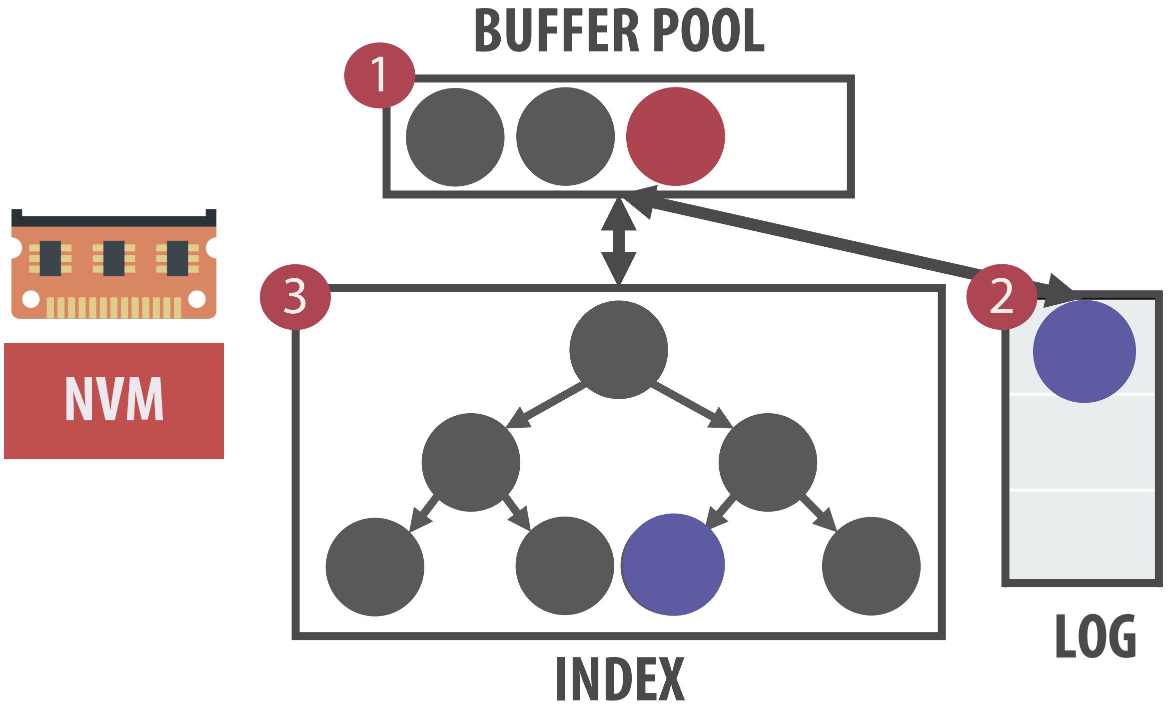
SINGLE-WORD
COMPARE-AND-SWAP
INSTRUCTION



LOCK-FREEDOM

INTERMEDIATE STATES

PROBLEM #2: INDEX-SPECIFIC PROTOCOL



DURABILITY & ATOMICITY

INDEX-SPECIFIC LOGGING & RECOVERY

HOW TO SIMPLIFY PROGRAMMING ON NON-VOLATILE MEMORY?

BZTREE: A HIGH-PERFORMANCE LATCH-FREE INDEX FOR NON-VOLATILE MEMORY
VLDB 2018

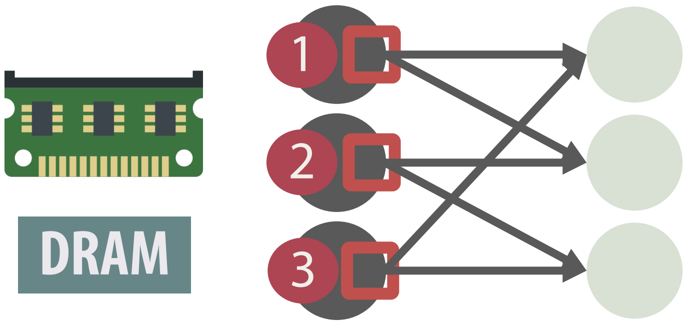
BZTREE: OVERVIEW

- NVM-centric design
  - *Uses a new software primitive to simplify programming*
  - *Provides same guarantees as disk-centric BwTree*
- B**z**Tree supersedes B**w**Tree
  - *But, we skipped B**x**Tree & B**y**Tree*
  - *Because we think it's the "last" index you will ever need!*
- Key techniques
  - *Offload programming complexity to the software primitive*
  - *Adopt a simpler NVM-centric architecture*

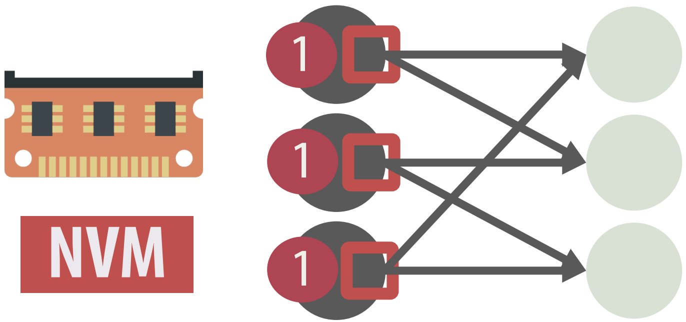
NVM-CENTRIC SOFTWARE PRIMITIVE

HARDWARE PRIMITIVE

SOFTWARE PRIMITIVE

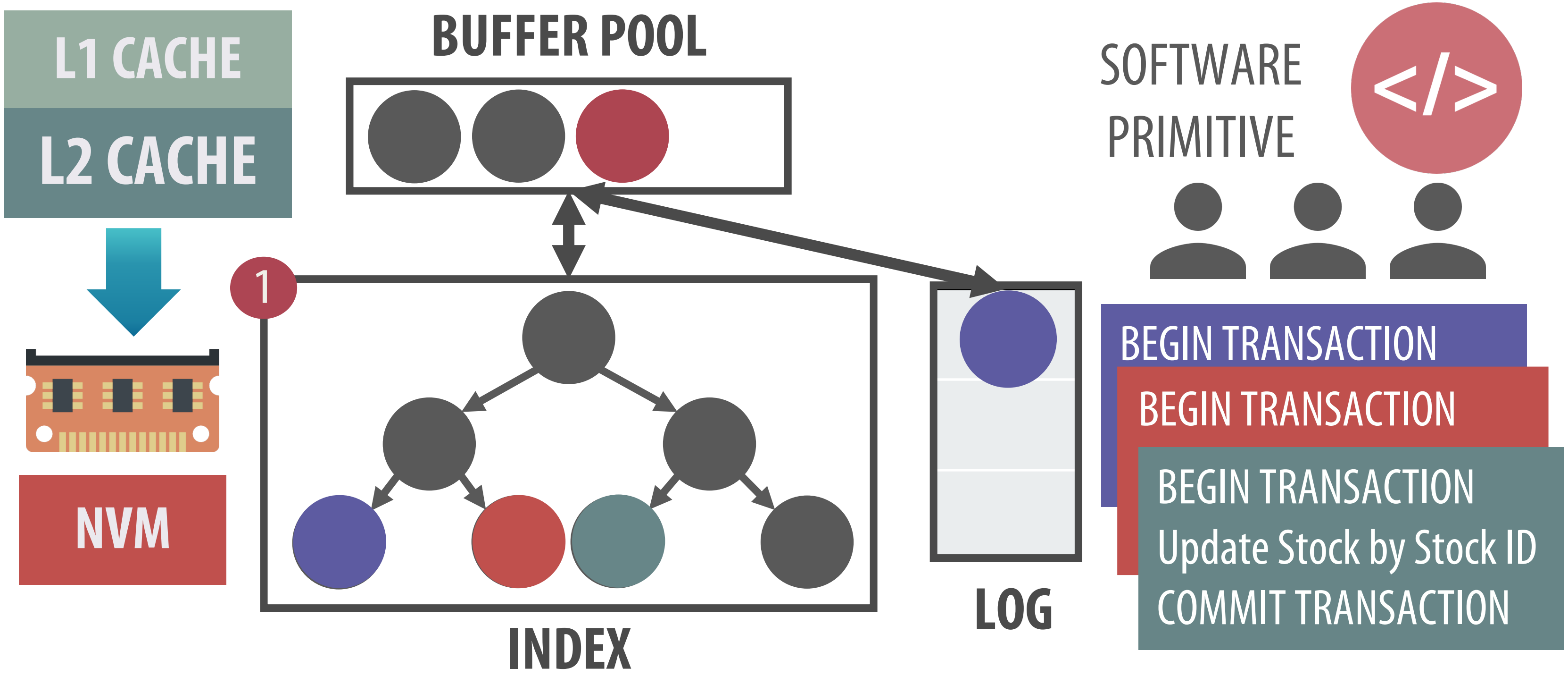


VOLATILE SINGLE-WORD COMPARE-AND-SWAP

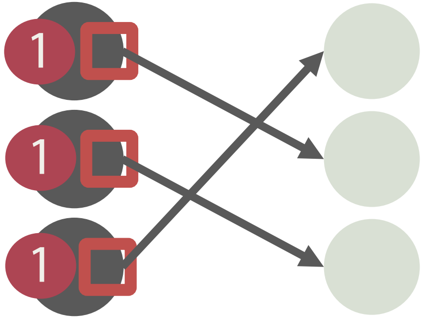


PERSISTENT MULTI-WORD COMPARE-AND-SWAP

BZTREE: NVM-CENTRIC ARCHITECTURE

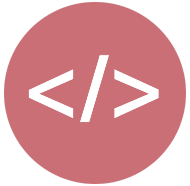


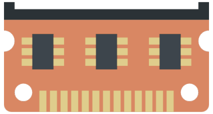
BZTREE: DURABILITY AND ATOMICITY



PERSISTENT MULTI-WORD COMPARE-AND-SWAP

SOFTWARE PRIMITIVE



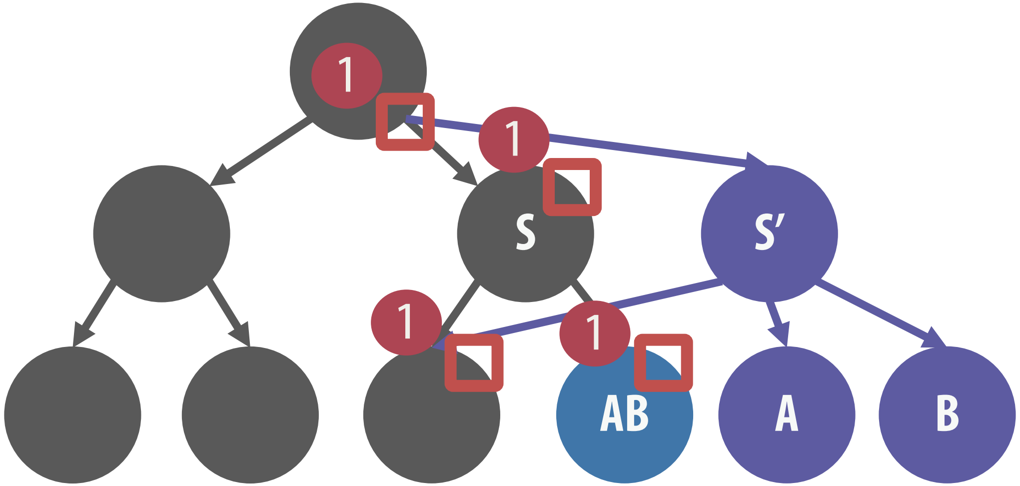


NVM

OPERATION TABLE

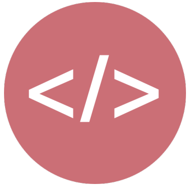
LOCATION	EXPECTED OLD VALUE	NEW VALUE	FLUSHED
0x100	OLD CHILD POINTER	NEW CHILD POINTER	1
0x200	OLD NODE STATUS	NEW NODE STATUS	1
0x300	OLD PARENT POINTER	NEW PARENT POINTER	0

SOLUTION #1: LOW CODE COMPLEXITY



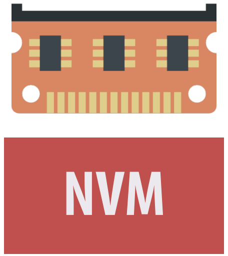
SPLITTING
A NODE

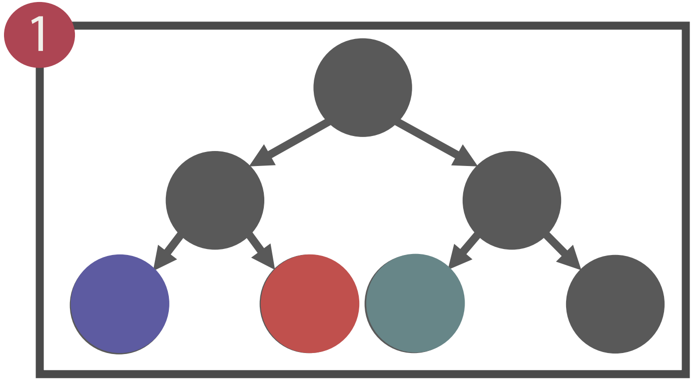
SOFTWARE
PRIMITIVE



EXPONENTIALLY FEWER INTERMEDIATE STATES

SOLUTION #2: NO INDEX-SPECIFIC PROTOCOL





INDEX

SOFTWARE PRIMITIVE </>

LOCATION	OLD VALUE	NEW VALUE	FLUSHED
0x100	OLD CHILD POINTER	NEW CHILD POINTER	1
0x200	OLD NODE STATUS	NEW NODE STATUS	1
0x300	OLD PARENT POINTER	NEW PARENT POINTER	0

DURABILITY & ATOMICITY

EVALUATION

- Index data structures: BzTree vs. BwTree index
  - *Code complexity*
  - *Index Performance*
- Benchmark: Yahoo Cloud Serving benchmark
  - *Read-mostly workload*
  - *Balanced workload*
- Storage devices
  - *Non-volatile memory (BzTree only works on NVM)*

CODE COMPLEXITY [NODE SPLIT PROTOCOL]

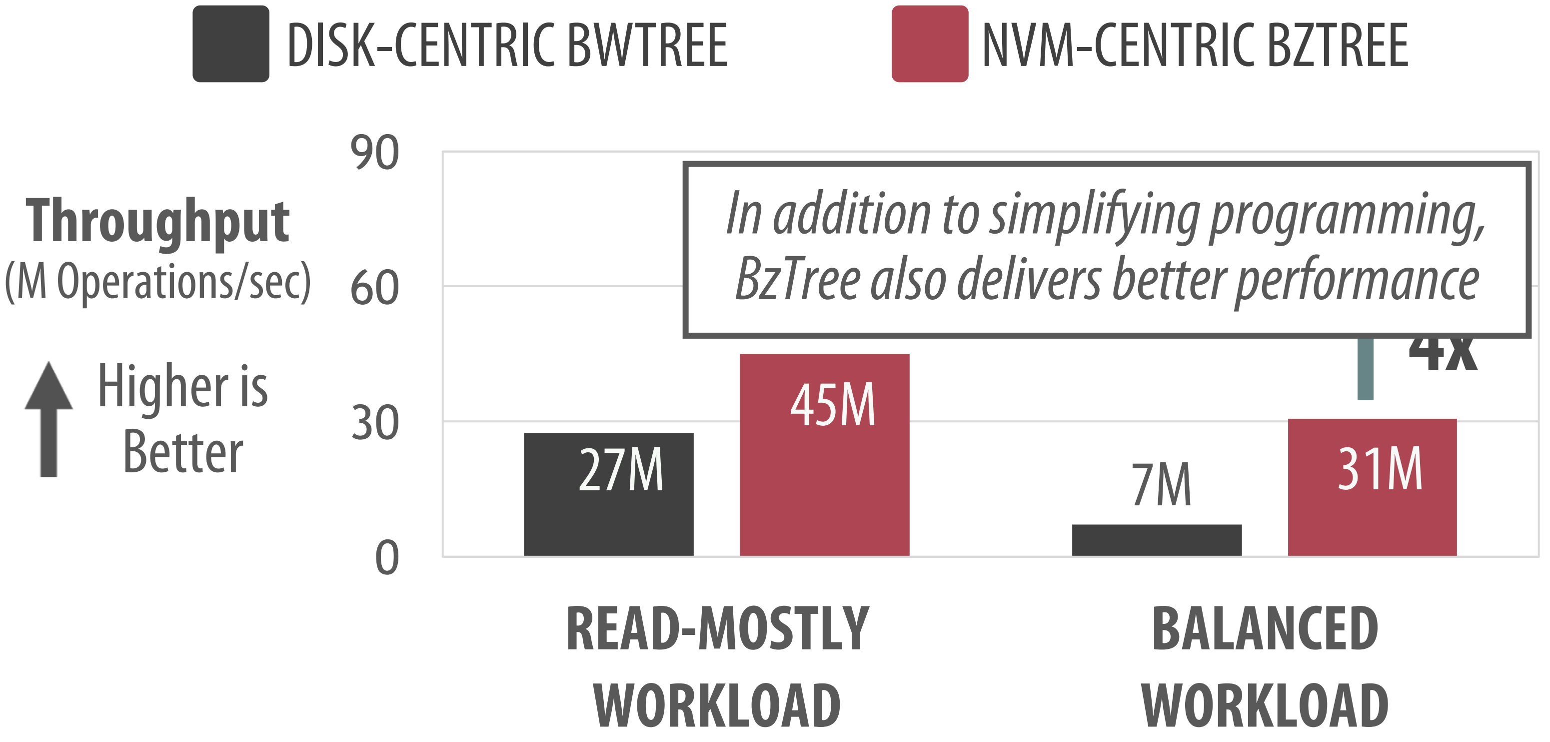
Lower is Better

CODE COMPLEXITY METRIC	BWTREE	BZTREE	
CYCLOMATIC COMPLEXITY	12	7	2x
LINES OF CODE	750	200	4x

1 FEWER INTERMEDIATE STATES

2 NO INDEX-SPECIFIC PROTOCOL

INDEX PERFORMANCE



BZTREE: SUMMARY

AVAILABILITY

PERFORMANCE

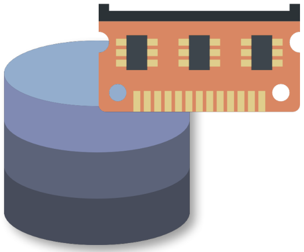
STORAGE COST

Advances the state of the art by illustrating a simpler way to design data structures for NVM

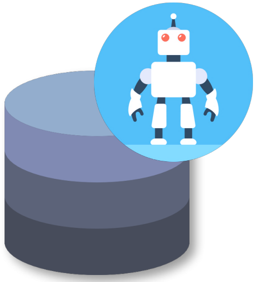
OTHER RESEARCH PROJECTS

CURRENT RESEARCH AGENDA

AREA #1:
NON-VOLATILE MEMORY
DATABASE SYSTEMS



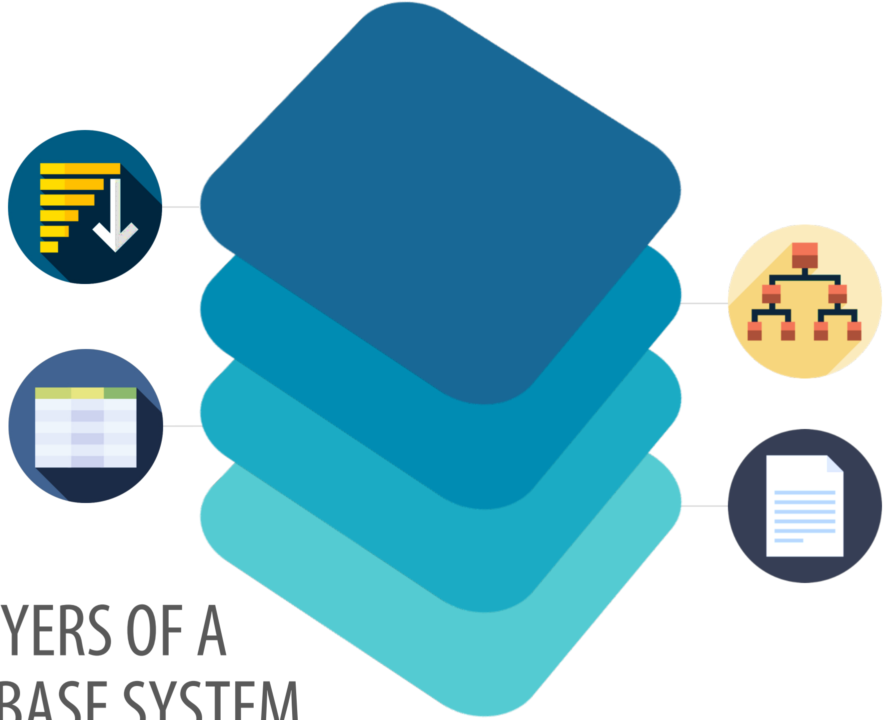
AREA #2:
SELF-DRIVING
DATABASE SYSTEMS



AREA #1: NVM DATABASE SYSTEM

QUERY PROCESSING
(SIGMOD'16)

STORAGE MANAGEMENT
(SIGMOD'15)



INDEXING
(VLDB'18)

LOGGING AND RECOVERY
(VLDB'17)

LAYERS OF A DATABASE SYSTEM

AREA #2: SELF-DRIVING DATABASE SYSTEM

STORAGE LAYOUT TUNING
(SIGMOD'16)

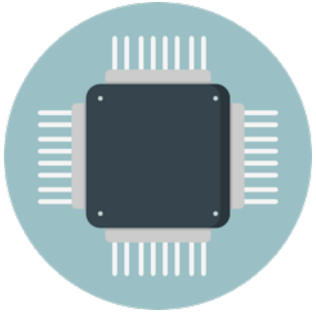
INDEX TUNING
(UNDER REVIEW)

SELF-DRIVING DESIGN
(CIDR'17)

SELF-DRIVING DATABASE SYSTEM

WRITE-BEHIND LOGGING

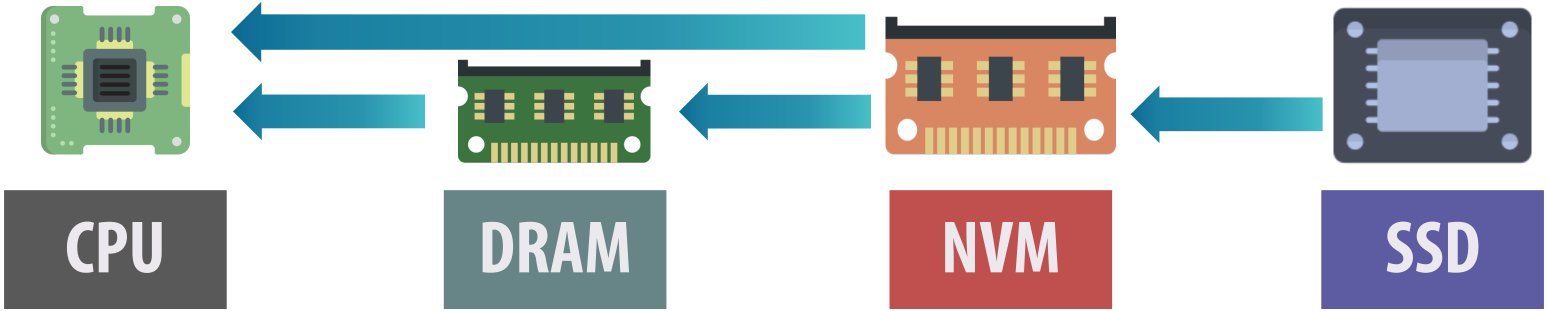
BZTREE INDEX



FUTURE DIRECTIONS

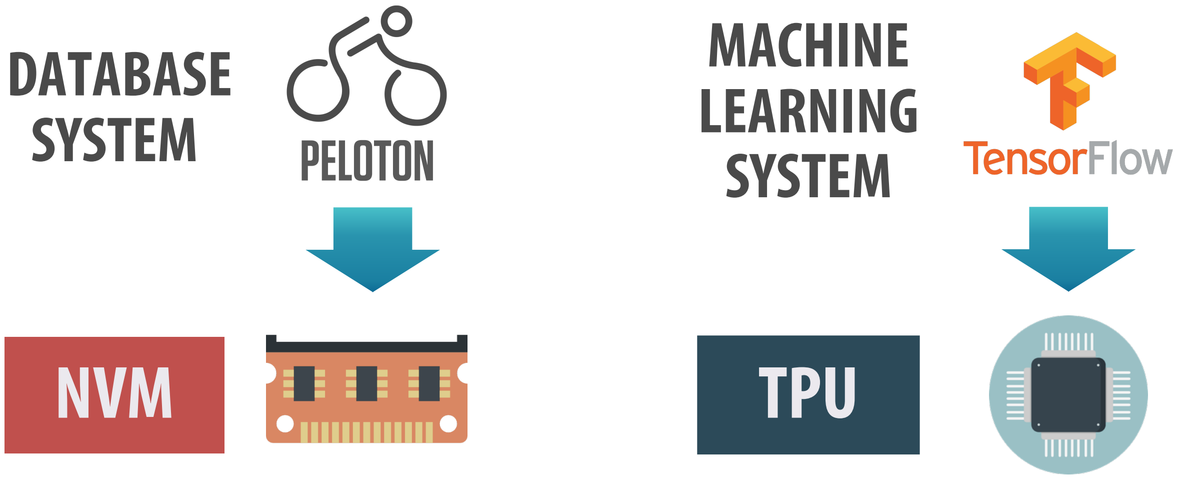
CROSS-MEDIA STORAGE MANAGEMENT

- Storage management tailored for a multi-tier hierarchy
  - *Industry collaboration: Intel Labs, Samsung Research*



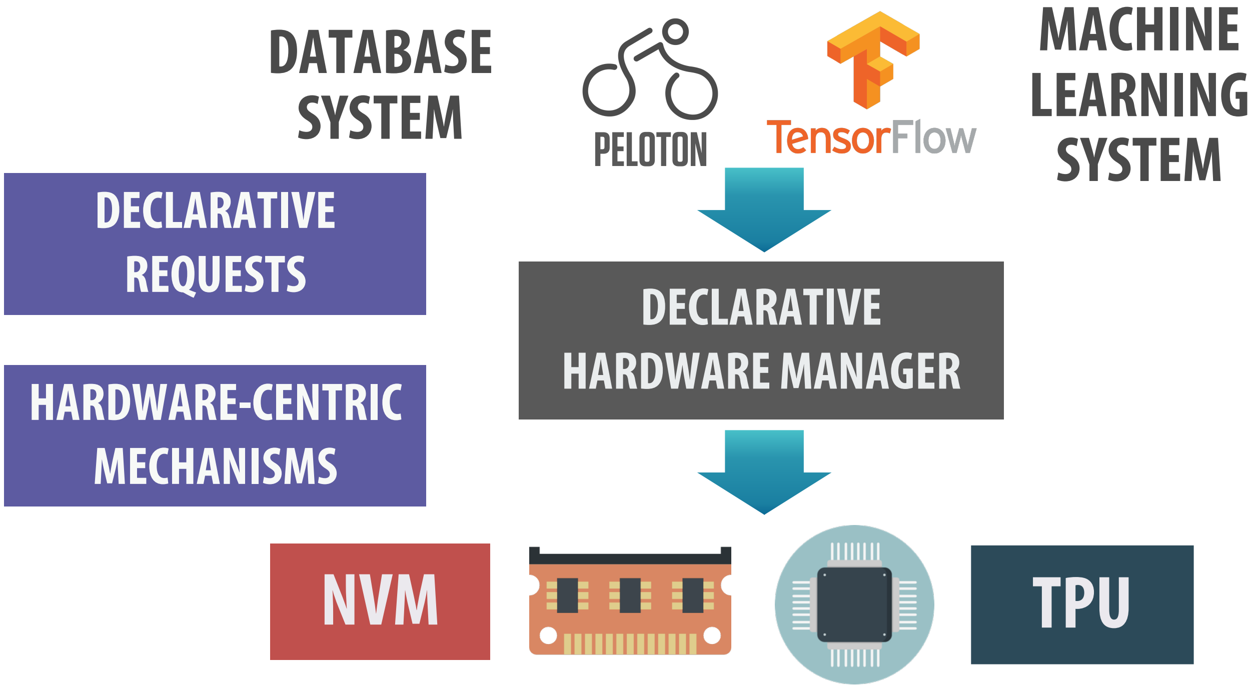
DECLARATIVE HARDWARE MANAGEMENT

- Most hardware-centric optimizations are system-specific



This approach does not scale!

DECLARATIVE HARDWARE MANAGEMENT



CONCLUSION

- Non-volatile memory invalidates age-old design assumptions
- Presented the design of a new NVM-centric database system
- Broader impact on other types of data processing systems

WRITE-BEHIND LOGGING

BZTREE INDEX

PELOTON

http://pelotondb.io

BACKUP SLIDES

DECLARATIVE STORAGE MANAGEMENT

DATABASE SYSTEM

PELOTON

TensorFlow

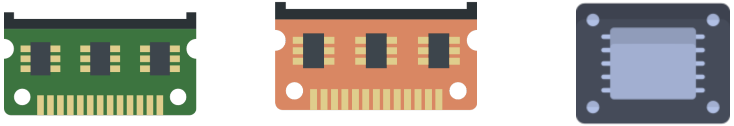
MACHINE LEARNING SYSTEM

DECLARATIVE STORAGE ALGEBRA

STORAGE-CENTRIC MECHANISMS

DECLARATIVE HARDWARE MANAGER

- DATA LAYOUT
- DATA MIGRATION
- DATA ORDERING
- DATA DISTRIBUTION
- DATA PACKING



DRAM

NVM

SSD

WRITE-BEHIND LOGGING: RELATED WORK

- NVM-centric logging, but only support linear-time recovery
  - **NVM Group Commit** [VLDB’13], **Passive Group Commit** [VLDB’14]
- NVM-centric logging with non-commodity hardware features
  - **MARS** [SOSP’13], **BPFS** [SOSP’09]

Write-behind logging enables constant-time recovery using only commodity hardware features

BZTREE: RELATED WORK

- NVM-centric indexing, but with index-specific recovery logic
  - **FP-Tree** [SIGMOD'16], **NV-Tree** [FAST'15]
- DRAM-centric indexing
  - **ART** [ICDE'13], **MassTree** [Eurosys'12]

BzTree illustrates a simpler way to design persistent data structures by obviating the need for index-specific recovery

RESEARCH IMPACT

- Research groups are shaping their systems for NVM
  - *Oracle, SAP HANA*
- Byte-addressable NVM is still not commercially available
  - *Intel shipped block-addressable NVM in 2017*
  - *Intel plans to ship byte-addressable NVM in 2019*
  - *Peloton will be the only open-source database system ready for NVM*

SHADOW PAGING

RECOVERY PROTOCOL CHARACTERISTIC	SHADOW PAGING	WRITE-BEHIND LOGGING
PROTOCOL TYPE	NO REDO/ NO UNDO	NO REDO/ UNDO
COMMIT OVERHEAD	HIGH	LOW
SYSTEM INTEGRATION	COMPLEX	SIMPLE
CONCURRENCY SUPPORT	NEED LOGGING	YES

VISTA RECOVERABLE MEMORY

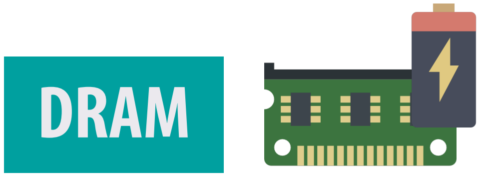
SYSTEM CHARACTERISTIC	RECOVERABLE MEMORY	WRITE-BEHIND LOGGING
UNDO MECHANISM	PHYSICAL UNDO	LOGICAL UNDO
CONCURRENCY SUPPORT	NO	YES
DBMS INTEGRATION	COMPLEX	SIMPLE

THE LOG IS THE DATABASE

SYSTEM CHARACTERISTIC	AMAZON AURORA	WRITE-BEHIND LOGGING
PROTOCOL TYPE	REDO/ UNDO	NO REDO/ UNDO
MATERIALIZATION	YES	NO
READ OVERHEAD	HIGH	LOW

BATTERY-BACKED DRAM

- Available only in specialized environments
  - *General-purpose database systems are not designed to leverage it*
- Limitations of battery-backed DRAM
  - *Physical form factor, Availability, Reliability, Cost*



MEDIA FAILURE

- Write-behind logging focuses on software failures
  - *Transaction failures, System failures*
  - *Software failures outnumber media failures 10-to-1*
- Media failure
  - *Replicating data to another machine's non-volatile memory*

SECURITY/PROTECTION

- Virtual memory protection mechanism
  - *All accesses should go through the TLB*
  - *Using write-permission bits in the page table*

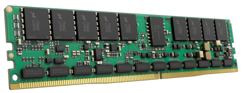
TRANSACTIONAL MEMORY

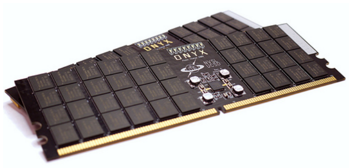
SYSTEM CHARACTERISTIC	SOFTWARE TRANSACTIONAL MEMORY	HARDWARE TRANSACTIONAL MEMORY	PERSISTENT MULTI-WORD CAS
DURABILITY	YES	NO	YES
PERFORMANCE	HEAVYWEIGHT	LIGHTWEIGHT	LIGHTWEIGHT
FALSE ABORTS	NO	YES	NO

READ-COPY-UPDATE

SYNCHRONIZATION PRIMITIVE CHARACTERISTIC	READ-COPY-UPDATE	PERSISTENT MULTI-WORD CAS
MULTI-LOCATION	NO	YES
DURABILITY	NO	YES
WRITERS USE LOCKS	YES	NO

CANDIDATE NVM TECHNOLOGIES





Faster than DRAM

FLASH-BACKED DRAM

PHASE-CHANGE MEMORY

Slower than DRAM

SPIN-TRANSFER TORQUE MRAM

MEMRISTOR

2D & 3D FLASH