Introduction to New Hardware for Distributed Databases

NVM RDMA

Presented By: Xing Wei

E-Mail: simba_wei@stu.ecnu.edu.cn

Overview

Overview — Part 1

Faster Processor

Larger Memory

Higher Bandwidth

Overview — Part 1

In a nutshell……

Traditional Memory, CPU and Disk are going crazy!

Crazy hardware no longer means powerful performance under the traditional architecture!

Hardware is such a mysterious thing, even taken as a trouble maker!

Overview — Part 1

In-memory join evaluation from 09 to 12

Kim et al. PVLDB'09	• Hash joins faster than sort merge joins • Will change when SIMD wide enough • Showed tuning to multicore, SIMD
Blanas et al. SIGMOD'11	• No need for tuning a has join • No need for careful partitioning • Hardware hides complexity
Albutiu et al. PVLDB'12	• Sort merge join better already • No need to use SIMD

emmm ……

Overview — Part 1

Main-Memory Computing Systems

Main-Memory Storage Systems

Data Center

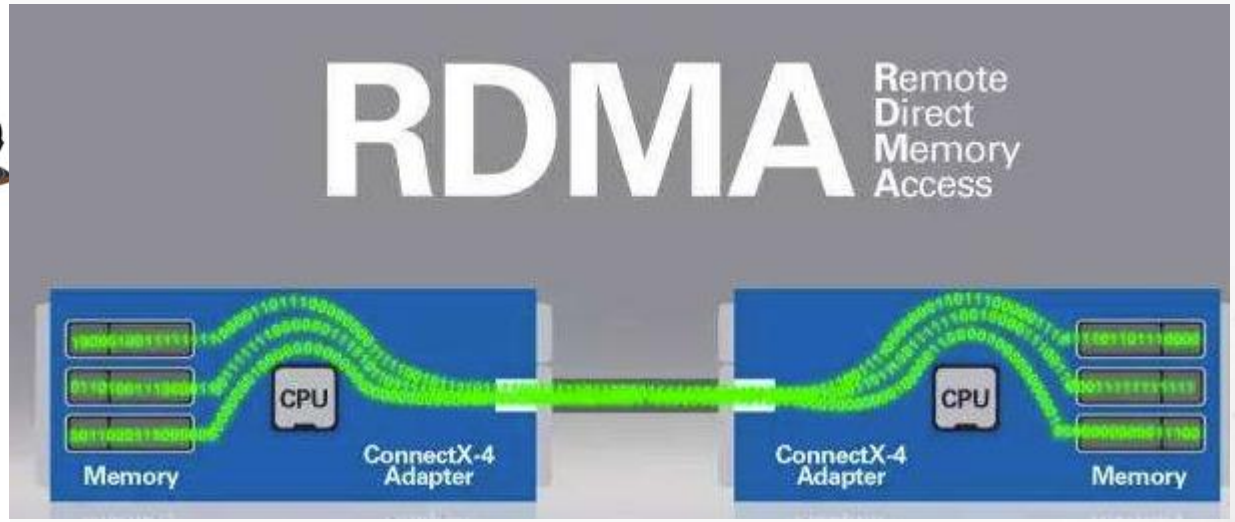
❖Large Application Demands

- –Memory is volatile
- –HDD/SSDs have high latency
- –Network has high latency and limited bandwidth

Overview — Part 1

New hardware breaks the bottleneck and becomes a Game Changer !

Non-volatile Memory (NVM)/
Storage-class Memory (SCM)/
Persistent Memory (PM)

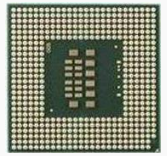


Remote Direct Memory Access (RDMA)

Hardware Features

Features — Part 2

Cache



DRAM

Fast, but volatile

Critical Performance Gap

Tap is dead, disk is tap!

----Jim Gray, Dec. 2006

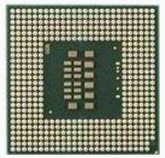
SSD

Persistent, but slow

Disk

Features — Part 2

Cache



DRAM

Fast, but volatile

NVM

Fast & Persistent

SSD

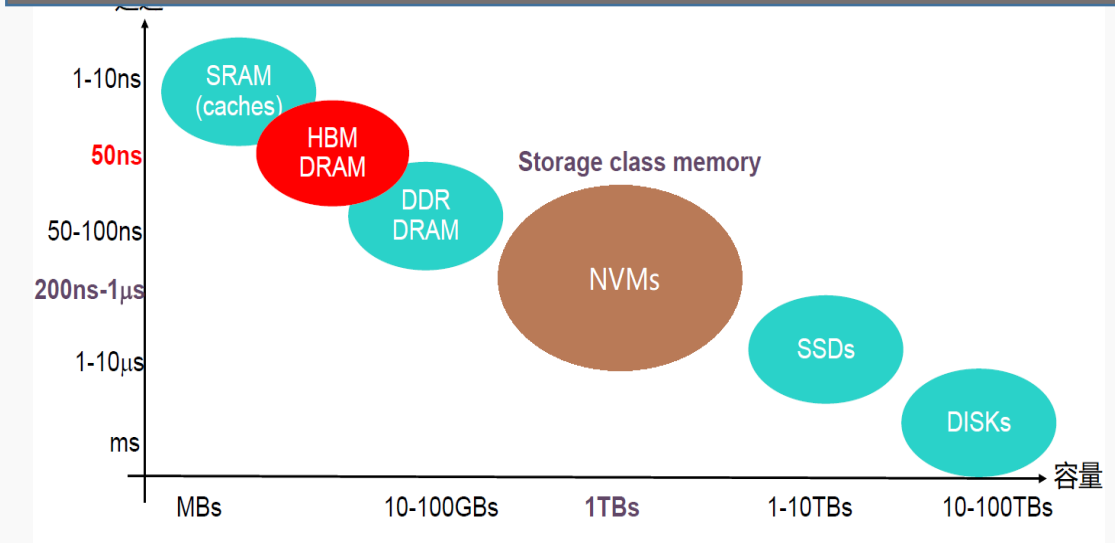
Persistent, but slow

Disk

Features — Part 2

	HDD	DRAM	NAND SLC Flash	PCRAM SLC	STTRAM	ReRAM
Data retention	Y	N	Y	Y	Y	Y
Cell Size	N/A	6-10F^2	4-6F^2	4-12F^2	6-50F^2	4-10F^2
Access Granularity (B)	512	64	4192	64	64	64

NVM blurs the boundary between DRAM and external storage!



Features — Part 2

Non-Volatile Memory

Compared with disks

- Byte-addressable vs. Block-addressable
- Faster read/write latencies

Compared with DRAM

- Consume less power
- Persist storage even when power is lost
- Own higher scalability and higher storage density

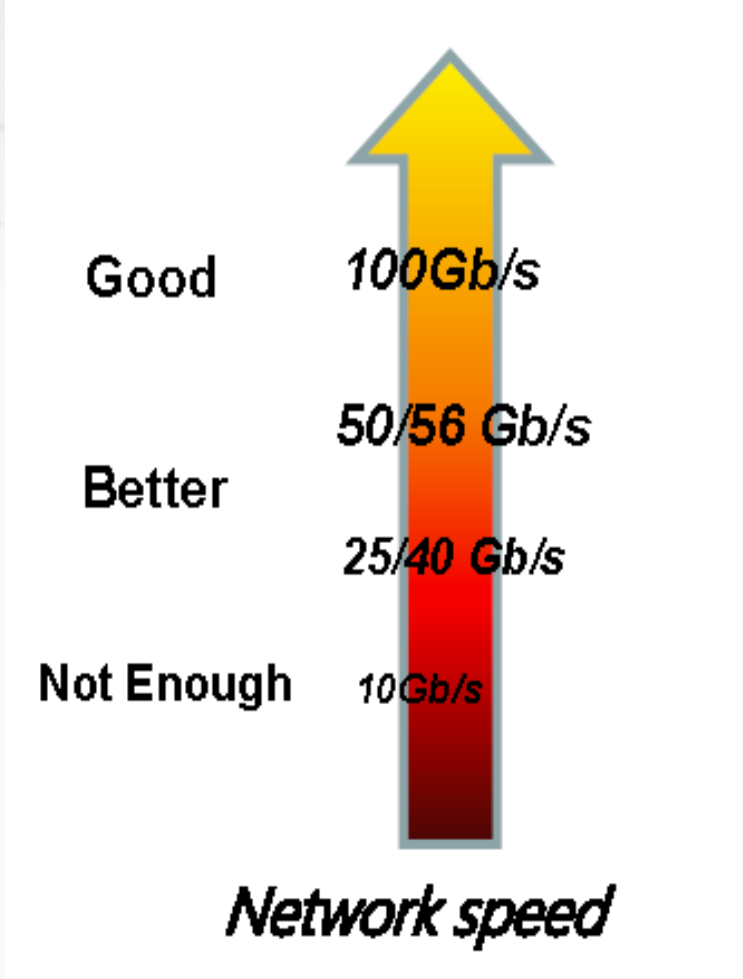
Features — Part 2

Non-Volatile Memory

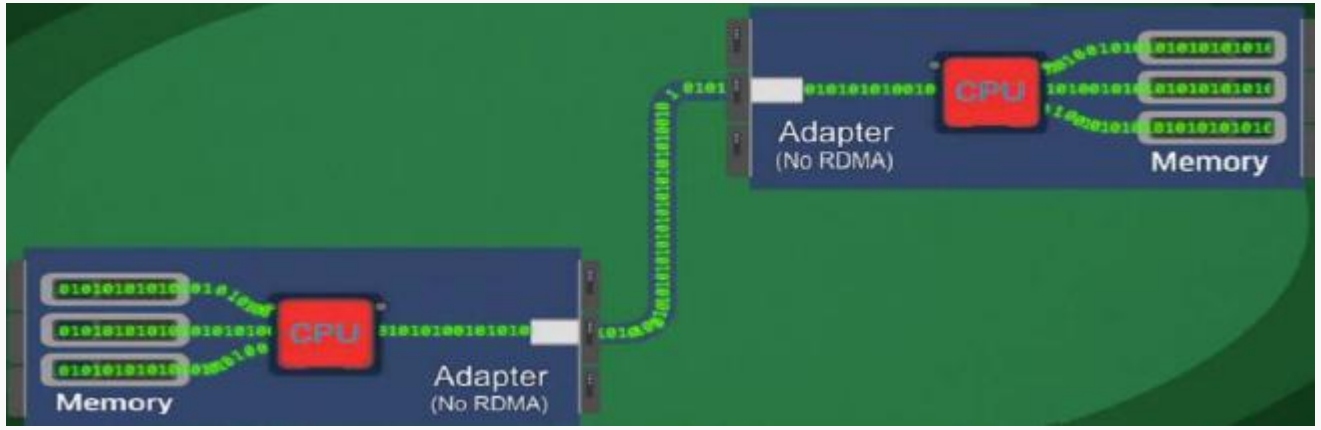
Shortcomings

- Read/Write asymmetry
- Write-wear operation decreases the service life

Features — Part 2

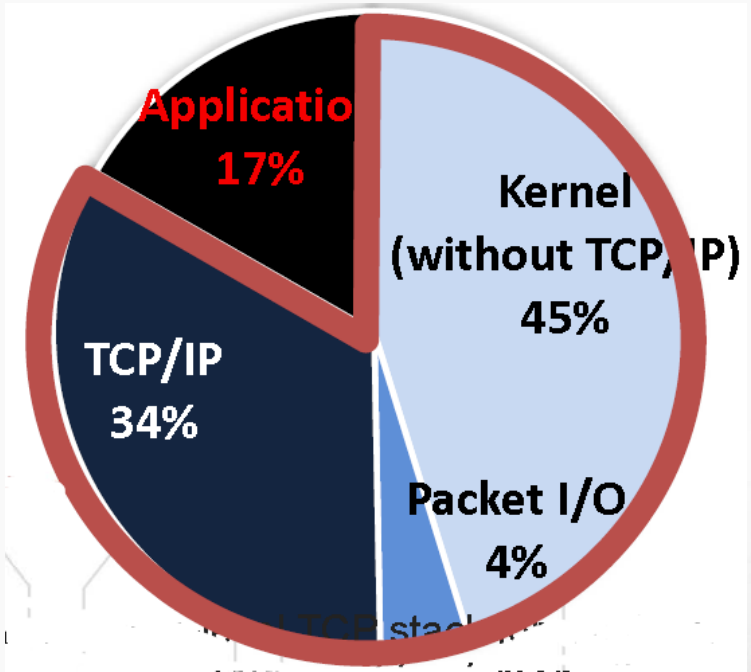


The speed we want



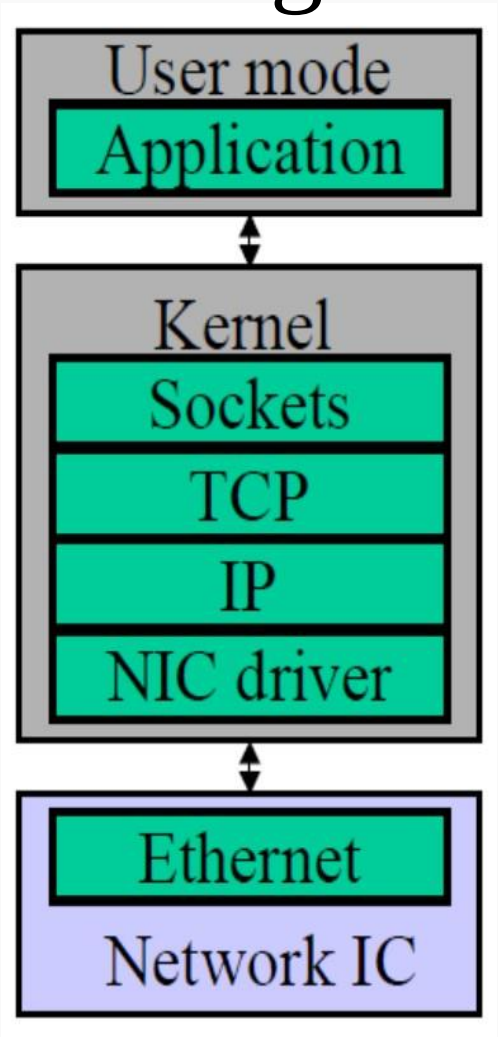
Traditional network transmission

Kernel costs 83%
CPU cycles



Features — Part 2

Message via Traditional Network



- **Send**
  - –Application buffer → Socket buffer
  - –Attach headers
  - –Data is pushed NIC buffer
- **Receive**
  - –NIC buffer → Socket buffer
  - –Parsing headers
  - –Data is copied into application buffer
  - –Application is scheduled(context switching)

Features — Part 2

How about Traditional Network?

- **Low Performance**
  - –Duplicate several copies
  - –Multiple abstractions between device driver and user apps
- **Low Flexibility**
  - –All protocol processing inside kernel
  - –Hard to support new protocols and message send/receive interfaces

Problem: Messages pass through the kernel

Features — Part 2

What is DMA ? ……

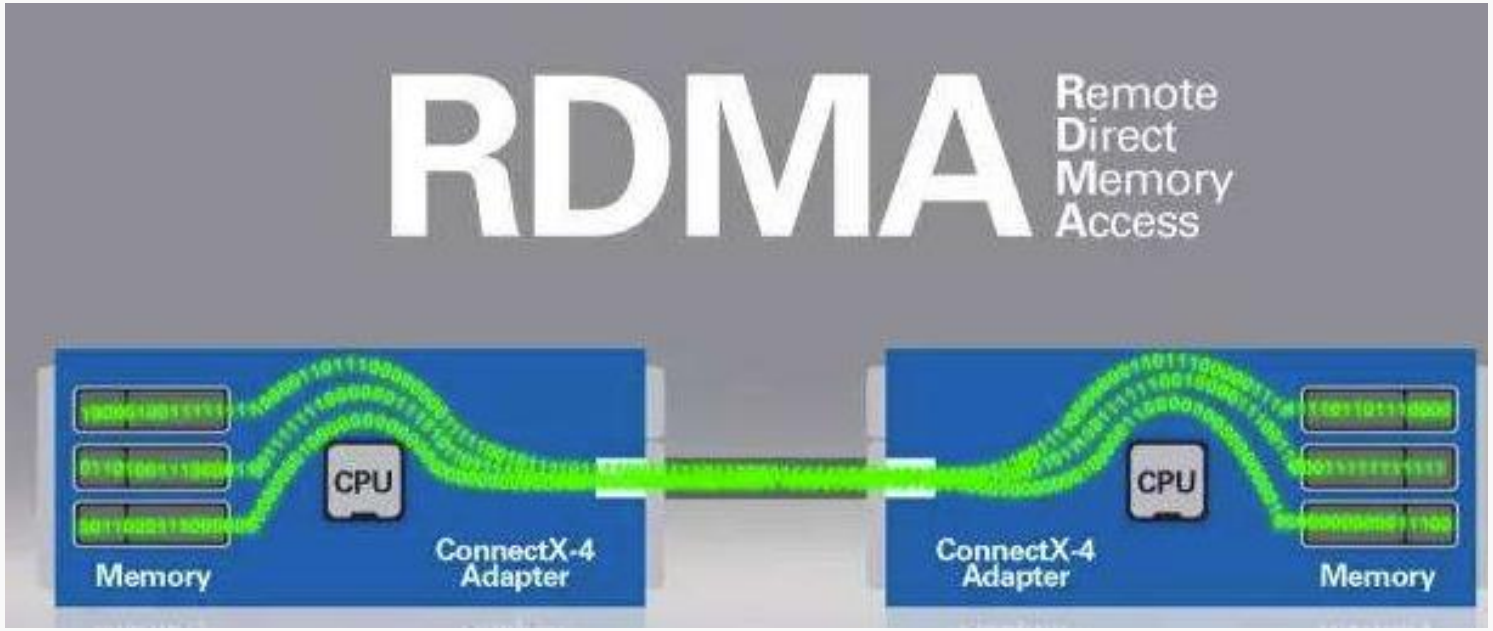
- **D**irect
  - –no Operating System Kernel involvement in transfers
  - –everything about a transfer offloaded onto Interface Card
- **M**emory
  - –transfers between user space application virtual memory
  - –no extra copying or buffering
- **A**ccess
  - –send, receive, read, write, atomic operations

Features — Part 2

Remote Direct Memory Access

What is Remote ? ……

One-Sided/Two-Sided Communication

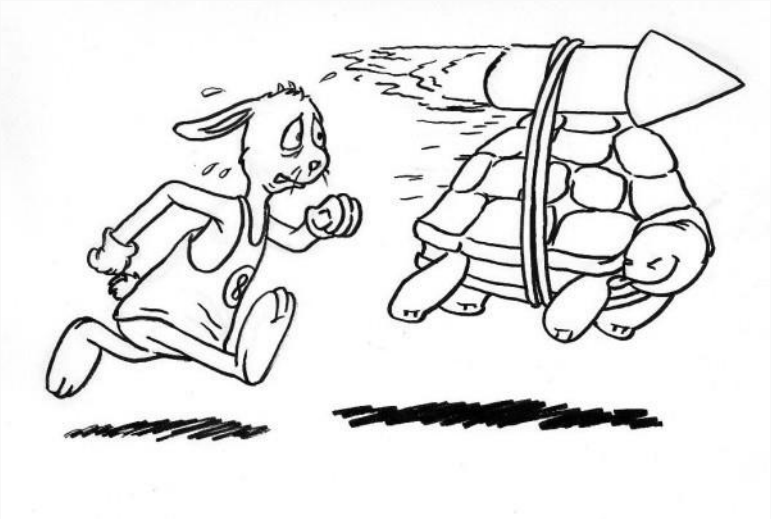


Data transfers between nodes via a network like DMA

Features — Part 2

Remote Direct Memory Access

- High throughput
- Low latency
- High messaging rate
- Low CPU utilization
- Low memory bus contention
- Message boundaries preserved
- Asynchronous operation

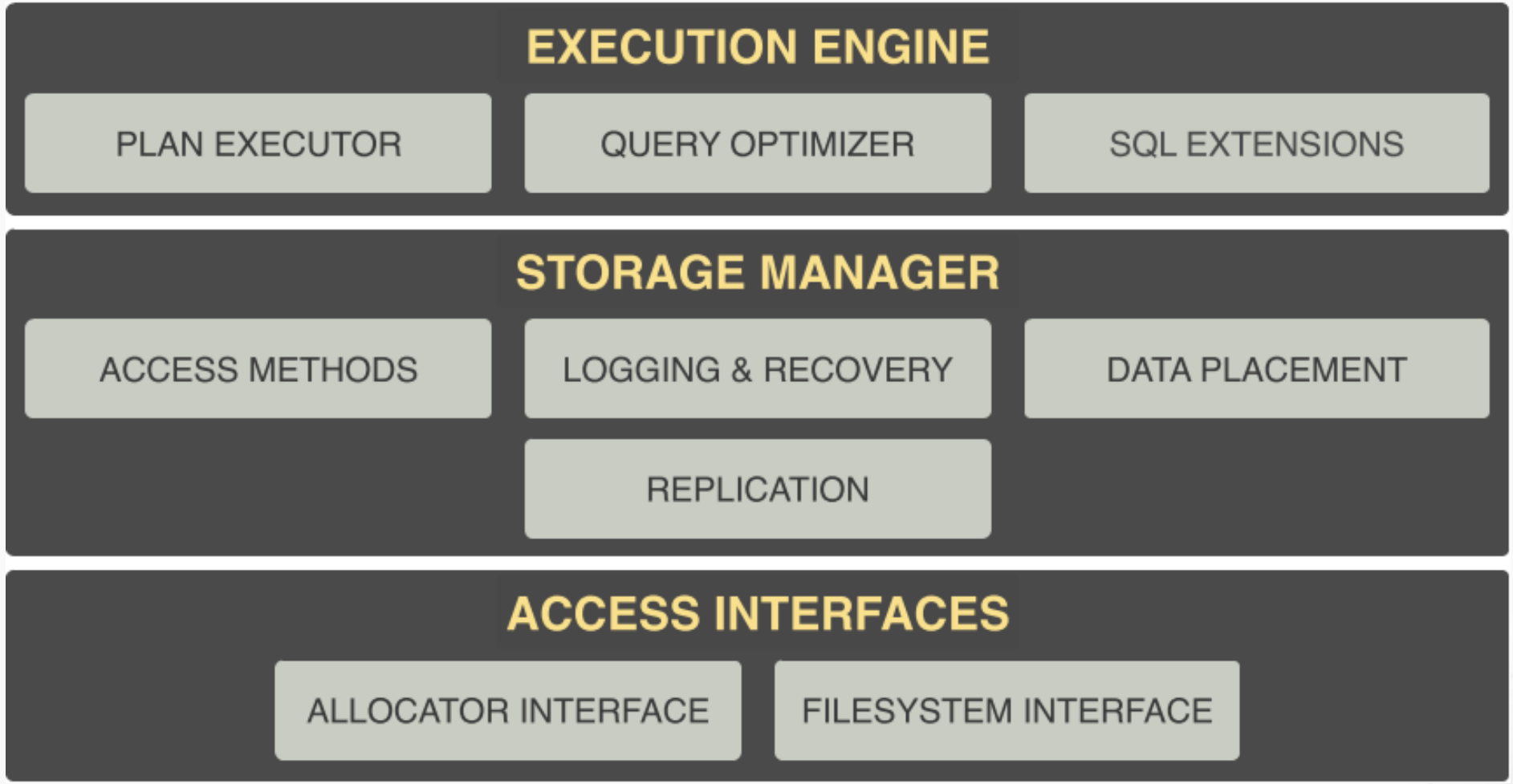


RDMA enables
Fast and *Lightweight* Communication!

Consideration

Consideration — Part 3

Which Components are Affected by NVM in DBMSs?



Consideration — Part 3

Execution Engine

Key Consideration:

write/read *asymmetry*

Consideration — Part 3

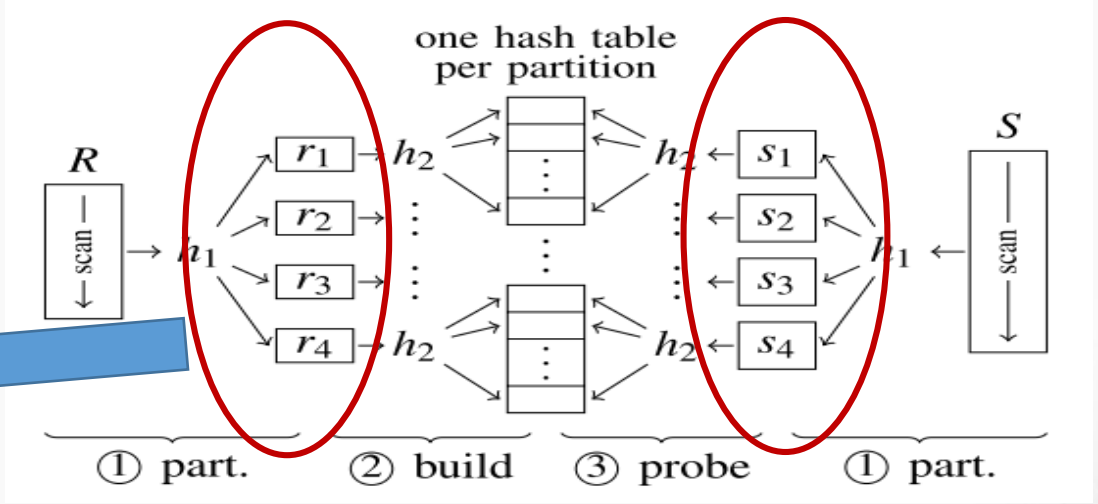
Execution Engine
— *Plan Execution*

❖**Rethink Query Processing Algorithm**

–Join Algorithms

–Sorting Algorithms

Write-Intensive



Write out the entire tables back on NVM

Consideration — Part 3

Execution Engine
— *Plan Execution*

❖**Rethink Query Processing Algorithm**

–Join Algorithms

–Sorting Algorithms

Write-Intensive

❖**Reduce the number of writes**

–Adjusting to avoid write-intensive job

–Designing write-limited algorithm

S. Chen, P. B. Gibbons, and S. Nath. Rethinking database algorithms for phase change memory. In CIDR 2015

Write-limited sorts and joins for persistent memory. PVLDB, 7(5), 2014

Consideration — Part 3

Execution Engine — *SQL Extension*

- **Allow user to control data placement**
  - –Certain performance-critical tables and MVs

```
ALTER TABLESPACE nvm_table_space DEFAULT ON_NVM;
```

Construct new extensions and standardize them!

  - –Exclude certain columns from being stored on NVM

```
ALTER TABLE orders ON_NVM EXCLUDE(order_tax);
```

Consideration — Part 3

Execution Engine

— *Query Optimizer*

❖Traditional Cost-Based Query Optimizer

– Distinguish between sequential & random accesses

– But not between reads and writes

$$\text{Cost(hash join)} = (\text{Cost}_{\text{sequential}} + \text{Cost}_{\text{random}}) * (\|\text{Inner-Table}\|_{\#\text{pages}} + \|\text{Outer-Table}\|_{\#\text{pages}})$$

❖NVM-Oriented Redesign

– Differentiate between reads and writes in cost model

$$\text{Cost(hash join)} = (\text{Cost}_{\text{sequential-reads}} + \text{Cost}_{\text{random-writes}}) * (\|\text{Inner-Table}\|_{\#\text{pages}} + \|\text{Outer-Table}\|_{\#\text{pages}})$$

Consideration — Part 3

Access Interfaces

- **Two Interfaces to Access NVM**
  - –Allocator interface (byte-level memory allocation)
  - –File system interface (POSIX-Compliant API)
- **Support in Major Operating Systems**
  - –From Windows Server 2016
  - –Linux 4.7

Consideration — Part 3

Access Interface

— *Allocator Interface*

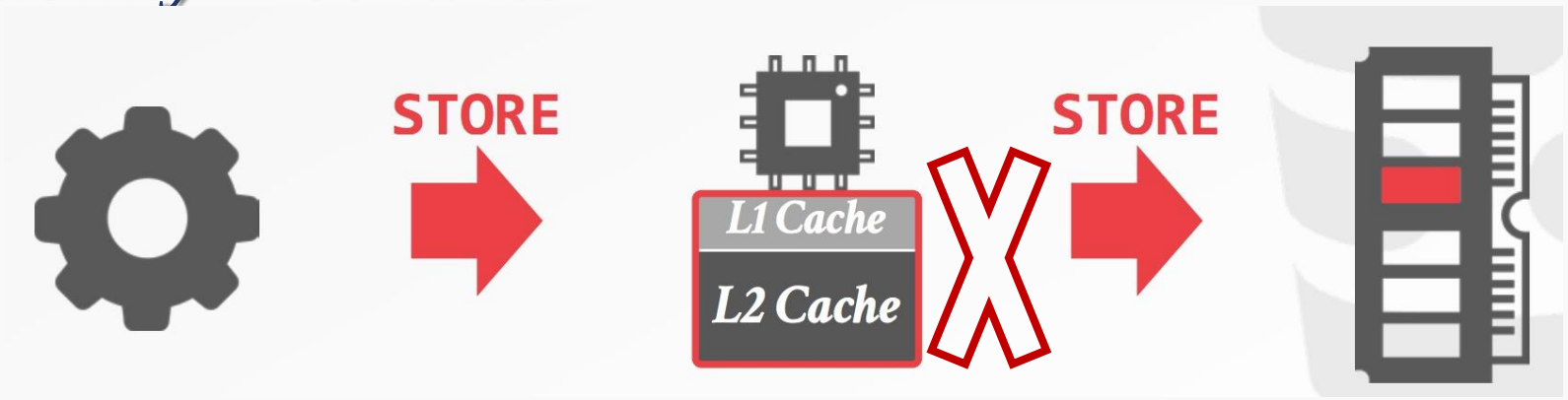
- **Treated as Regular DRAM Allocator**
  - –Dynamic Memory Allocation
  - –Metadata Management
- **Additional Features**
  - –Durability Mechanism
  - –Naming Mechanism
  - –Recovery Mechanism

Consideration — Part 3

Access Interface

— *Allocator Interface*

1、Durability Mechanism



Power is lost

❖**Two Steps Implements Persistence**

–Allocator first writes out dirty cache-lines (CLWB)

–Issues a memory fence (SFENCE) to ensure changes are visible

Consideration — Part 3

Access Interface

— *Allocator Interface*

2、Naming Mechanism

The allocator ensures that virtual memory addresses assigned to a memory-mapped region never change

Consideration — Part 3

Access Interface

— *Allocator Interface*

3、Recovery Mechanism

❖Atomicity of Memory Allocation

–No torn data writes

–No dangling references

❖Two Allocation Steps

–Reserve step: The DBMSs reserve the memory for storing ephemeral data. In case of a failure, the allocator reclaims the memory regions

–Before using this region, the DBMSs must activate the regions

Consideration — Part 3

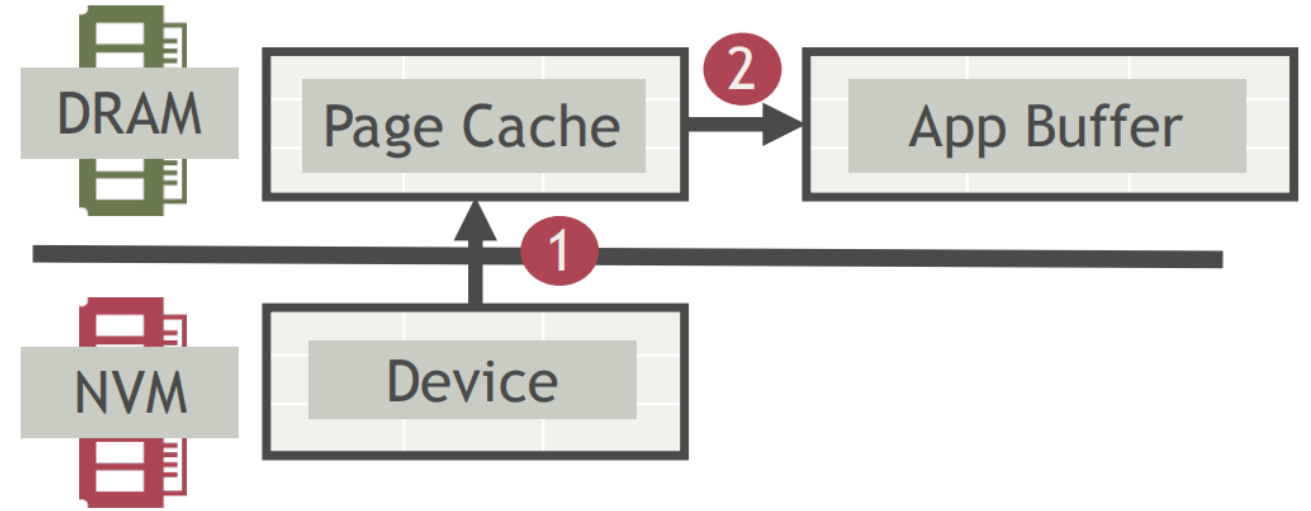
Access Interface

— *Filesystem Interface*

❖**Traditional block-based file system**

–File I/O: 2 Copies(Device → Page Cache → App Buff)

–Efficiency of I/O stack is not critical when hidden by disk latency.



However, NVM is byte-addressable and supports very fast I/O!

Consideration — Part 3

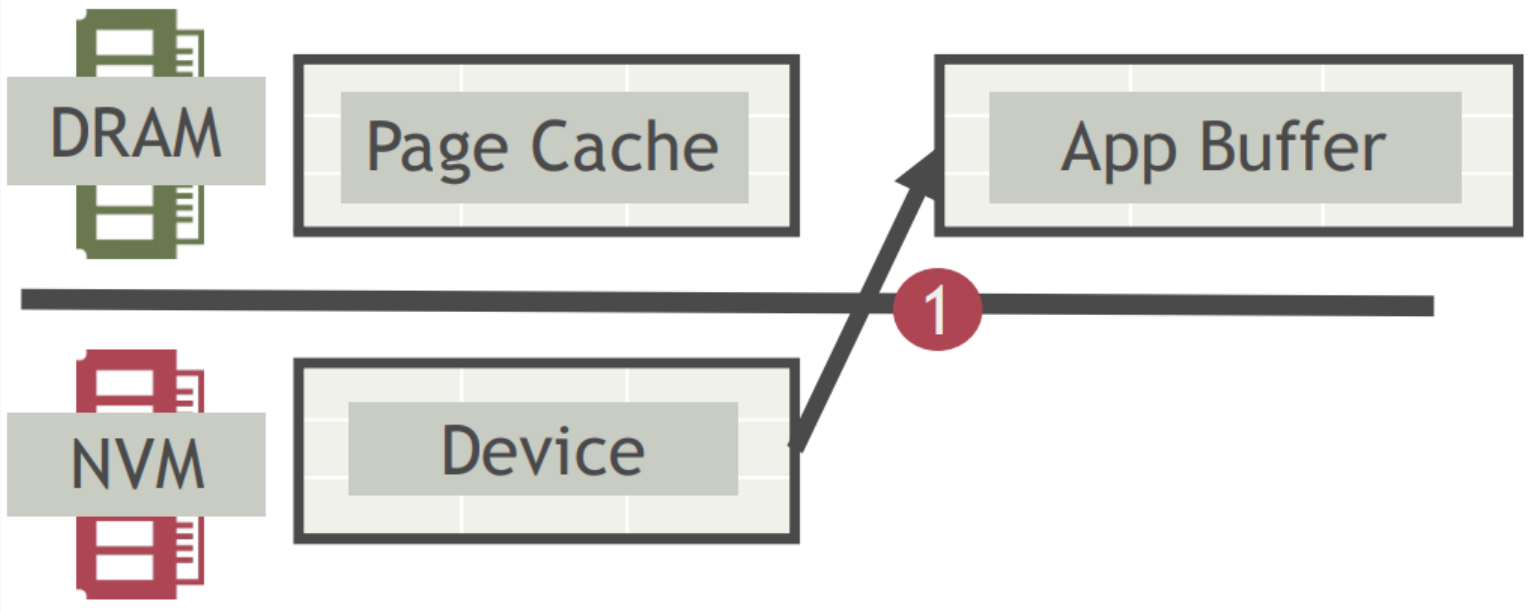
Access Interface

— *Filesystem Interface*

❖**Direct Access Storage (DAX)**

–DBMS can directly manage data by skipping page cache

–File I/O: 1 Copy (Device → App Buffer)

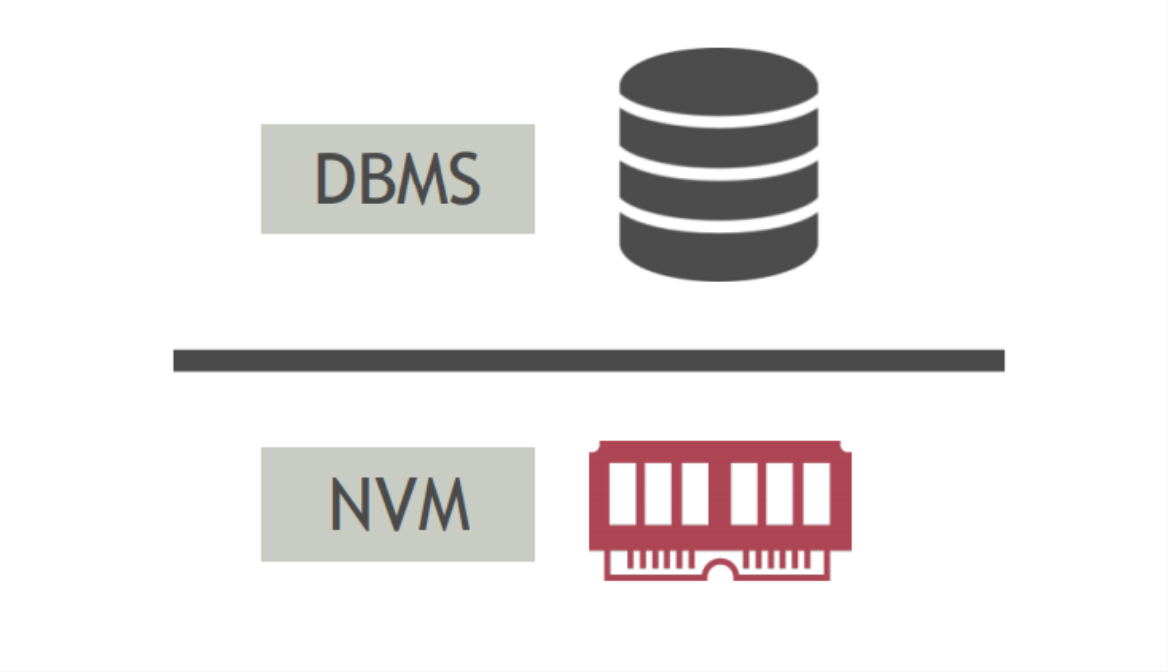


Consideration — Part 3

Storage Manager

❖**To keep things simple, NVM-only Storage Hierarchy**

–No volatile DRAM

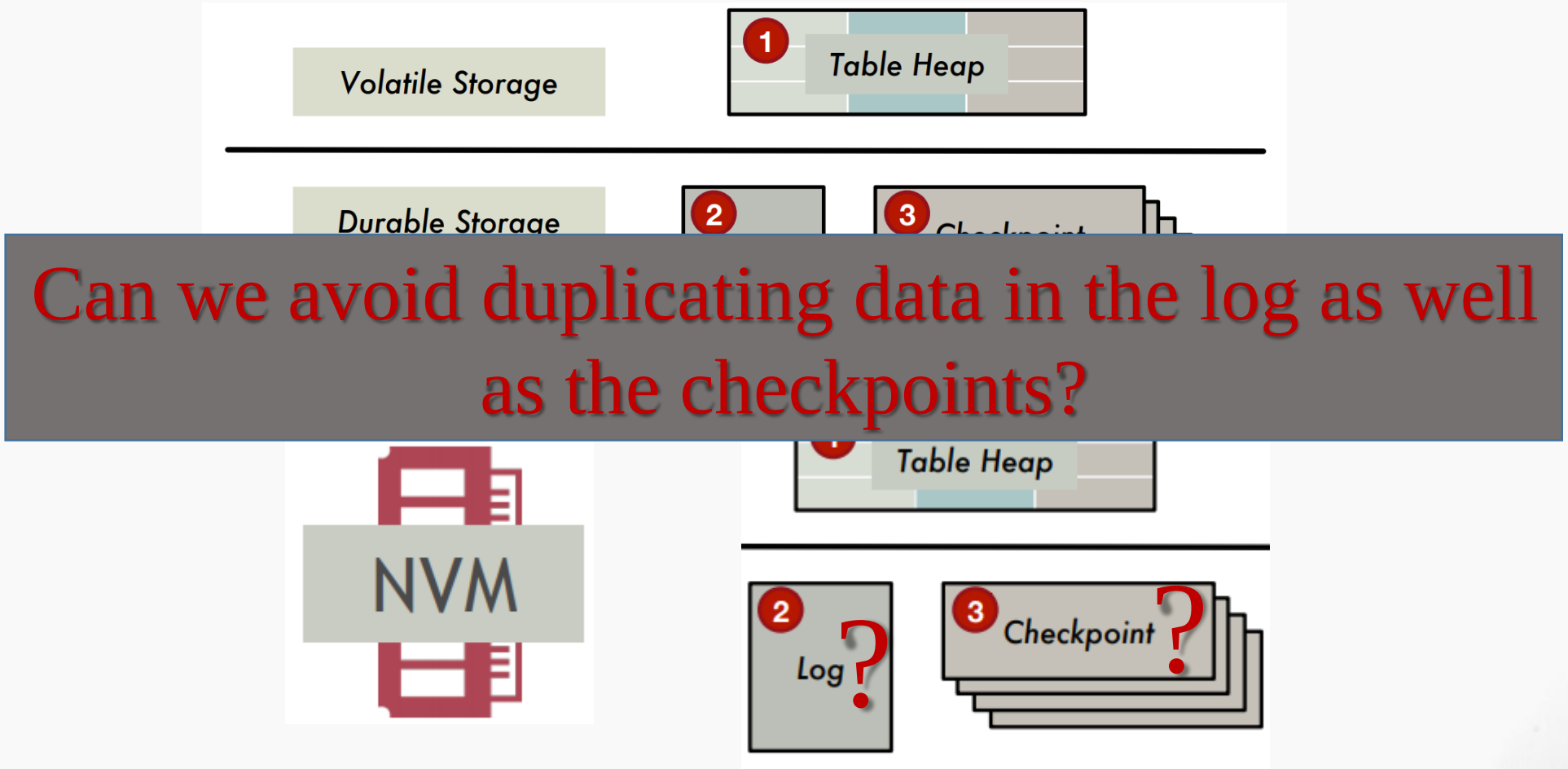


Consideration — Part 3

Storage Manager

— *Logging & Recovery*

❖Traditional Write-Ahead Logging(WAL)

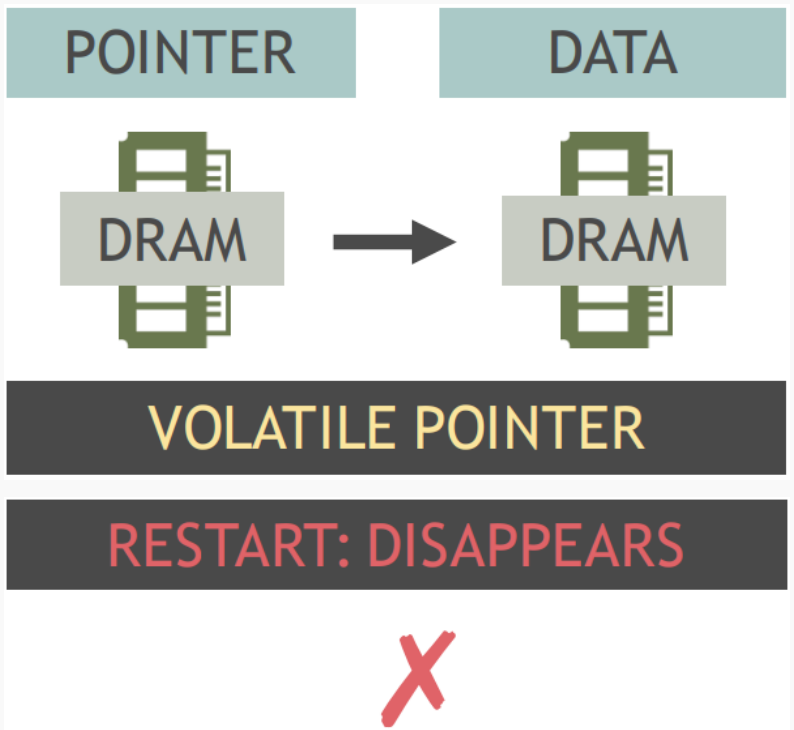


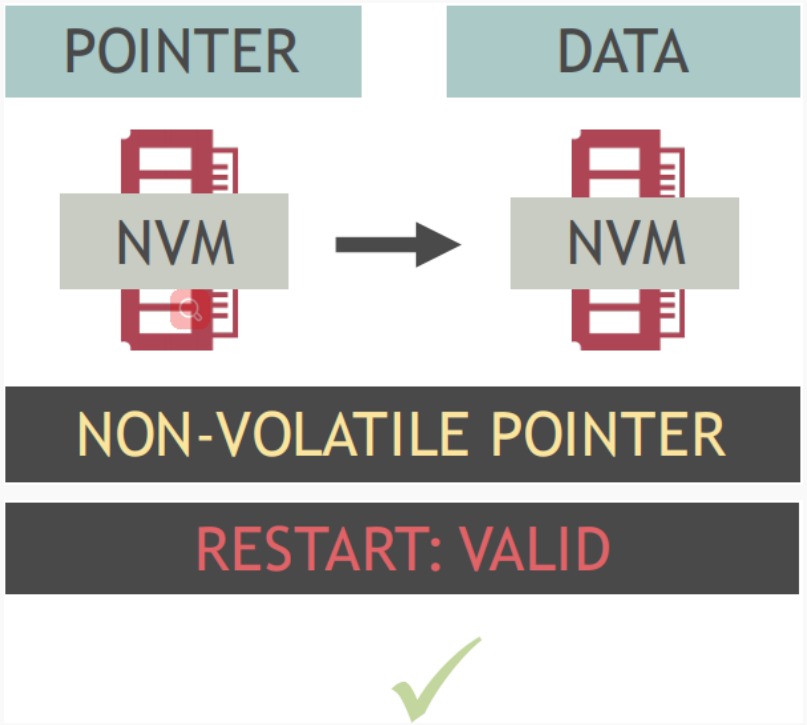
Consideration — Part 3

Storage Manager

— *Logging & Recovery*

❖Non-Volatile Pointer





Consideration — Part 3

Storage Manager

— *Logging & Recovery*

❖Non-Volatile Pointer

–Only store non-volatile tuple pointers in log records

Table Heap	
TUPLE ID	TUPLE DATA
100	XYZ
101	X'Y'Z'

Write-Ahead Log
TRADITIONAL MANAGER
INSERT TUPLE XYZ
UPDATE TUPLE XYZ → X'Y'Z'

NVM-AWARE MANAGER
INSERT TUPLE 100
UPDATE TUPLE 100 → 101

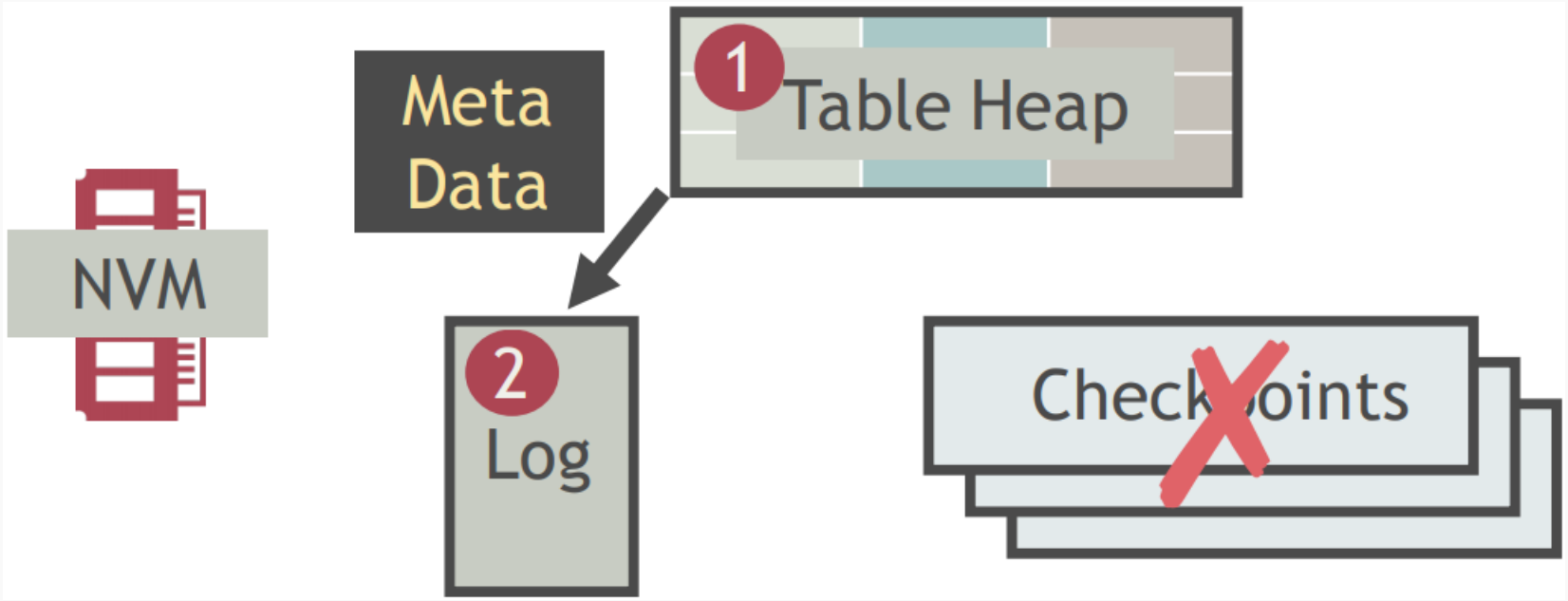
Consideration — Part 3

Storage Manager

— *Logging & Recovery*

❖Further Avoid Data Duplication

–Write-ahead meta-data logging



Consideration — Part 3

Storage Manager

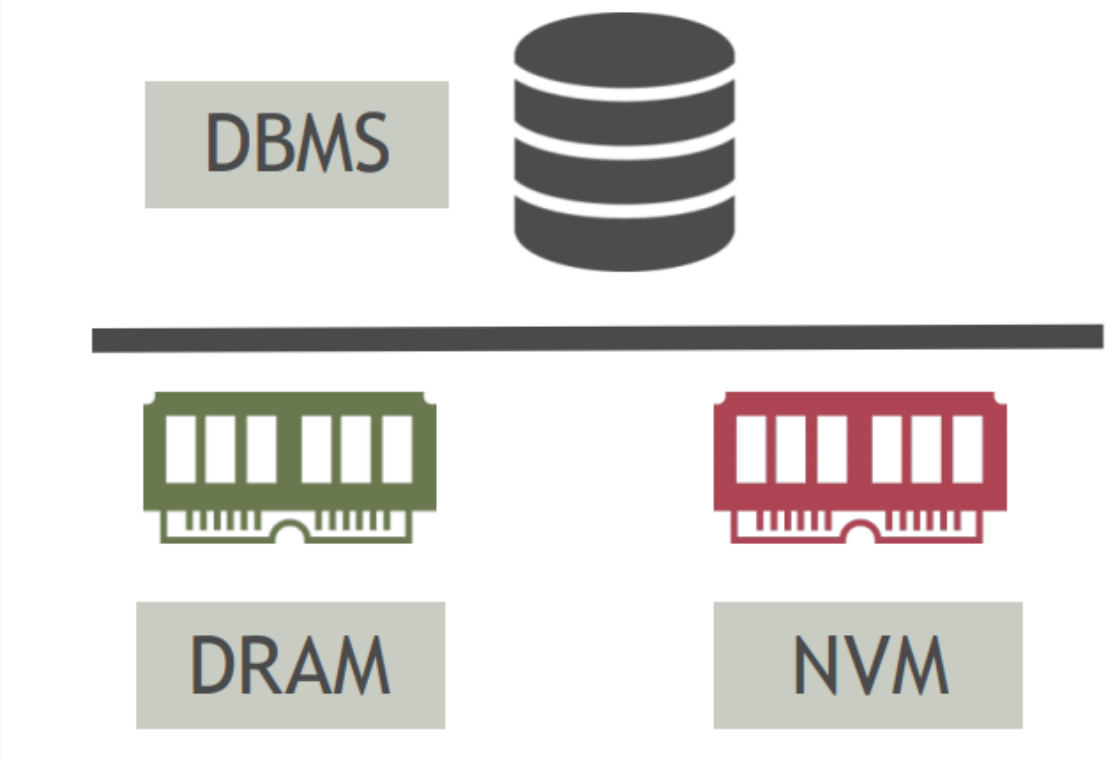
— *Logging & Recovery*

- **Targets at NVM-Only Hierarchy(WAL)**
  - –Leverages the durability of memory
  - –Skips duplicating data in the log and checkpoints
  - –Improves runtime performance
  - –Extends lifetime of the device

Consideration — Part 3

Storage Manager

- **Two-Tier Storage Hierarchy**
  - –Generalize the logging and recovery algorithms for this storage hierarchy

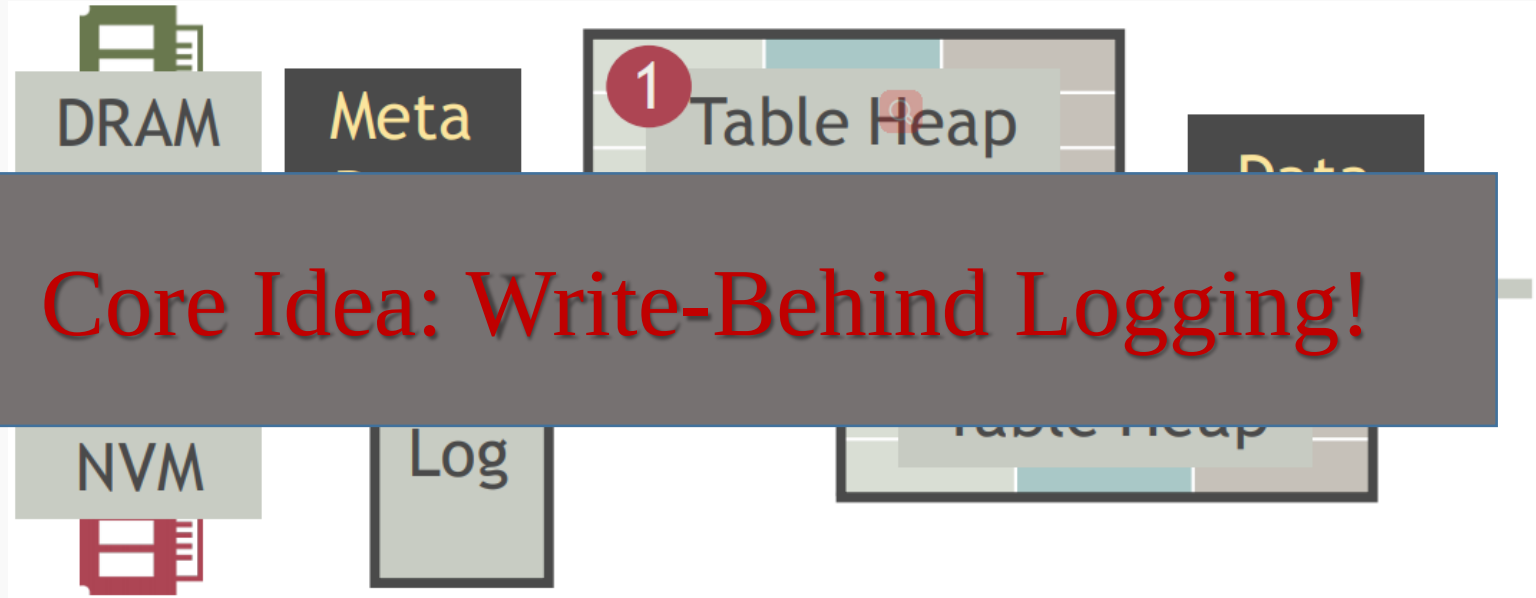


Consideration — Part 3

Storage Manager

— *Logging & Recovery*

❖**Write-Behind Logging (WBL)**



NVM supports fast random writes and directly write data to the multi-versioned database!

Consideration — Part 3

Storage Manager

— *Logging & Recovery*

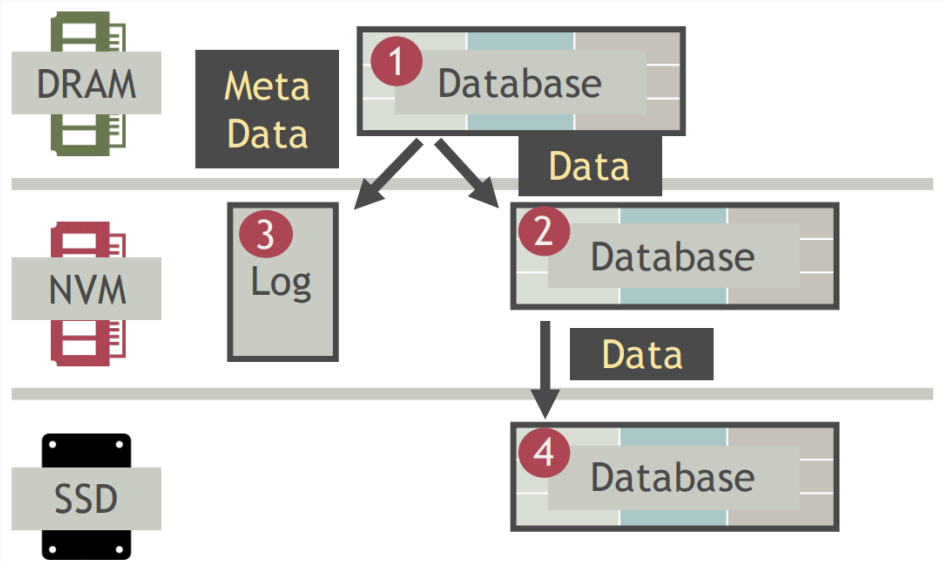
- **Write-Behind Logging**
  - –No need to record REDO log
  - –Recover the database almost instantaneously
  - –Extend the device lifetime

Consideration — Part 3

Storage Manager

— *Data Placement*

- **Cost of First-Generation NVM Devices**
  - –SSD is still going to be in the picture
- **Data Placement**
  - –Three-tier DRAM+NVM+SSD hierarchy



Consideration — Part 3

Storage Manager

— *Data Placement*

- **Unlike SSD, Can directly Read Data from NVM**
  - –No need to copy data over to DRAM for reading
- **Cache hot data in DRAM**
  - –Dynamically migrate cold data into SSD
  - –Keep warm data on NVM

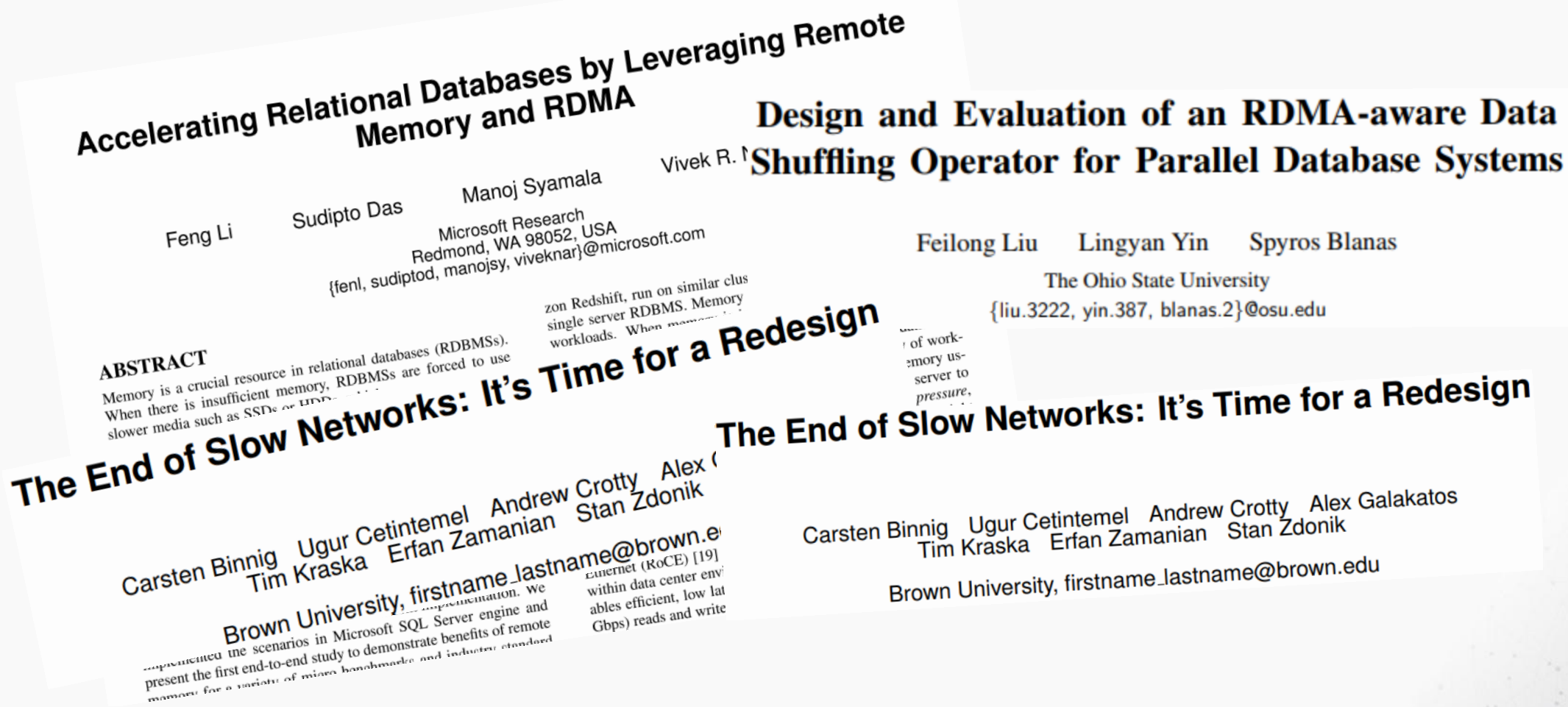
Consideration — Part 3

Storage Manager
— *Access Methods*

- **Read-Write Asymmetry & Wear-leveling**
  - –Write might take longer to complete compared to reads
  - –Excessive writes to a single NVM cell can destroy it
- **Write-Limited Access Methods**
  - –NVM-Aware B+ tree, Hash table

Consideration — Part 3

Rethinking Distributed Databases on High-Speed Networks



Accelerating Relational Databases by Leveraging Remote Memory and RDMA

Feng Li Sudipto Das Manoj Syamala Vivek R.

Microsoft Research
Redmond, WA 98052, USA
{fenl, sudiptod, manojsy, viveknar}@microsoft.com

ABSTRACT

Memory is a crucial resource in relational databases (RDBMSs). When there is insufficient memory, RDBMSs are forced to use slower media such as SSDs or HDD

zon Redshift, run on similar clus
single server RDBMS. Memory
workloads. When

The End of Slow Networks: It's Time for a Redesign

Carsten Binnig Ugur Cetintemel Andrew Crotty Alex
Tim Kraska Erfan Zamanian Stan Zdonik

Brown University, firstname_lastname@brown.e

implemented the scenarios in Microsoft SQL Server engine and present the first end-to-end study to demonstrate benefits of remote memory for a variety of micro benchmarks and industry standard

Ethernet (RoCE) [19]
within data center envi
ables efficient, low lat
Gbps) reads and write

Design and Evaluation of an RDMA-aware Data Shuffling Operator for Parallel Database Systems

Feilong Liu Lingyan Yin Spyros Blanas

The Ohio State University
{liu.3222, yin.387, blanas.2}@osu.edu

of work-
emory us-
server to
pressure,

The End of Slow Networks: It's Time for a Redesign

Carsten Binnig Ugur Cetintemel Andrew Crotty Alex Galakatos
Tim Kraska Erfan Zamanian Stan Zdonik

Brown University, firstname_lastname@brown.edu

Consideration — Part 3

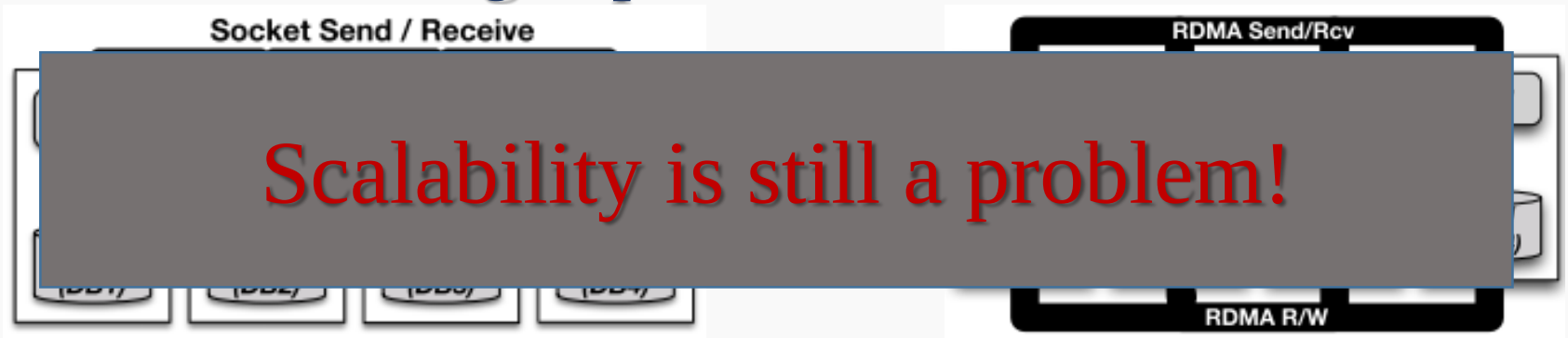
Rethinking Distributed Databases on High-Speed Networks

- **High Bandwidth & Low Latency**
  - –Change the traditional architecture
  - –Query Processing: new cost models, new physical operators
  - –Scalability of Distributed Transaction Processing： 2-Phase Protocol

Consideration — Part 3

Rethinking Distributed Databases on High-Speed Networks

Connecting Separate RAMs via RDAM



Shared-Nothing Architecture

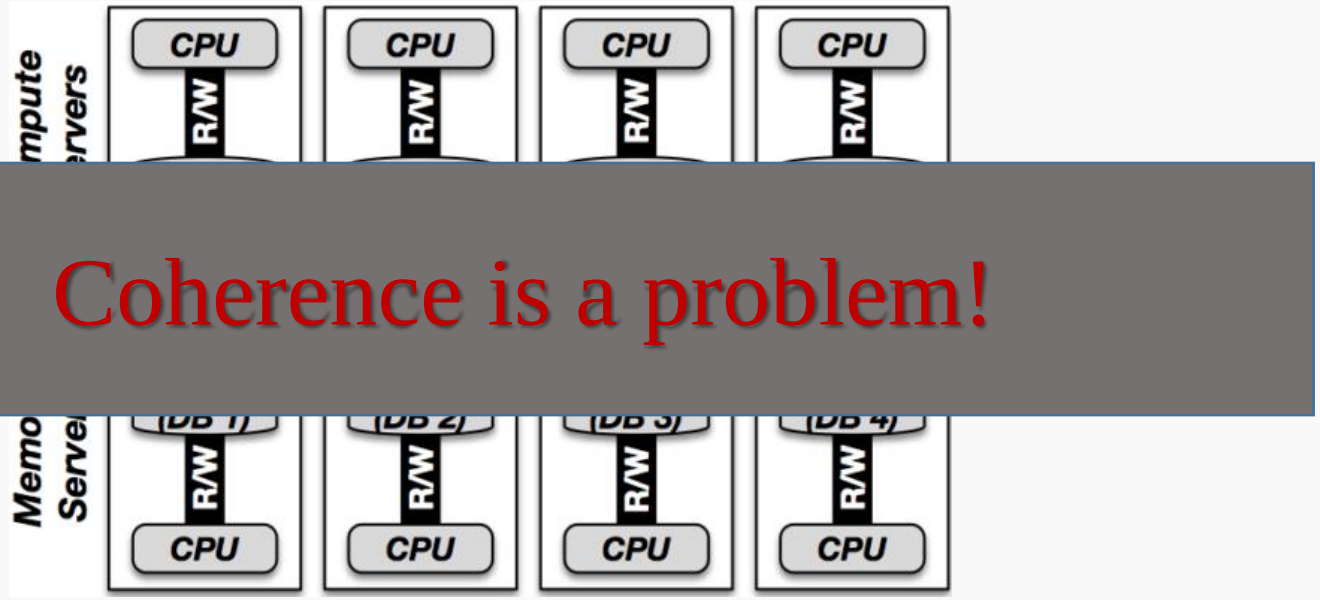
Shared-Memory Architecture

- **Break the Memory Barrier Between Severs**
  - –Extend the separate memory as a memory pool
  - –Support the Memory-Intensive Operators
  - –Form a new cache-level between local memory and HDD/SSDs

Consideration — Part 3

Rethinking Distributed Databases on High-Speed Networks

Network-Attached-Memory Architecture(NAM)



- **Decouple Compute and Storage**
  - –Scale the storage and Adjust computation resources easily

Consideration — Part 3

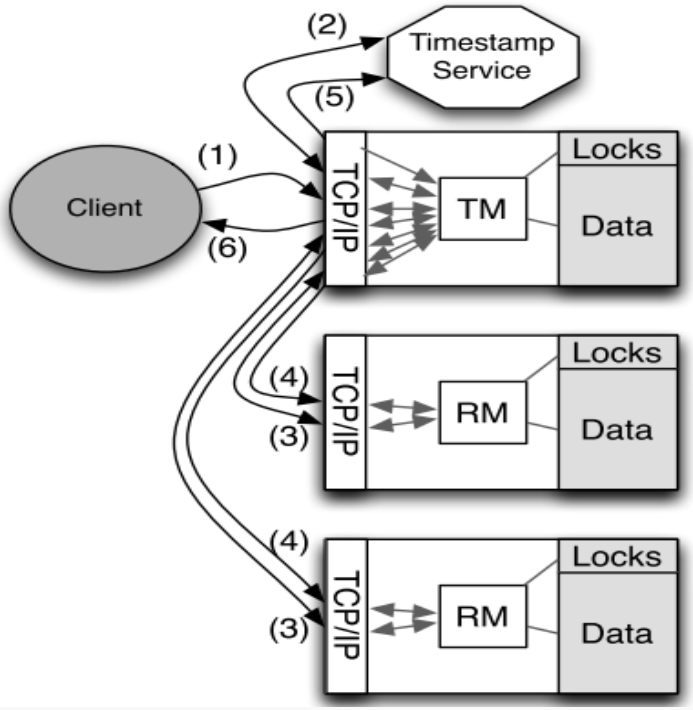
Rethinking Distributed Databases on High-Speed Networks

- **Query Processing via High bandwidth & Low Latency Network**
  - –Network is not the bottleneck
  - –Traditional distributed algorithms need to be redesigned
  - –Traditional cost model need to be redesigned

Consideration — Part 3

Rethinking Distributed Databases on High-Speed Networks

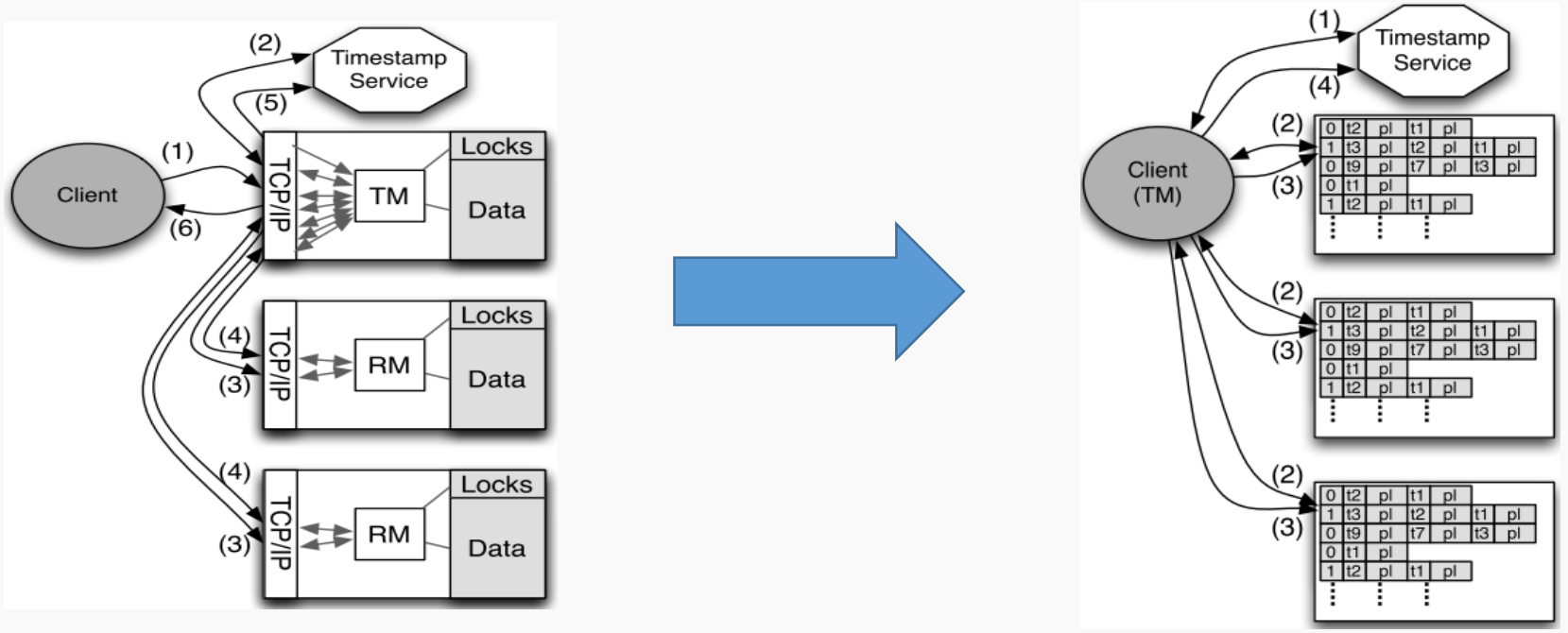
RDMA Improves the Scalability of Distributed Transactions



Traditional 2 PC Protocol

Consideration — Part 3

Rethinking Distributed Databases on High-Speed Networks



❖**Scale-Out become much easier**

–CPU-Intensive operations can be done by multiple clients

–Less communication and Higher Speed

System Design	Efficient Distributed Memory Management with RDMA and Caching	黄晨晨
	Octopus: an RDMA-enabled Distributed Persistent Memory File System	
Storage Level	Building a Bw-Tree Takes More Than Just BuzzWords	齐学成
	BzTree: A HighPerformance Latch-free Range Index for Non-Volatile Memory	
Query Level	Accelerating Relational Databases by Leveraging Remote Memory and RDMA	段惠超
	Rack-Scale In-Memory Join Processing using RDMA	

谢谢

THANK YOU