Disaggregated Database Systems

Jianguo Wang — PURDUE UNIVERSITY

Qizhen Zhang — UNIVERSITY OF TORONTO

Outline

- Introduction and motivation
- Storage disaggregation
- Additional discussions on PM

Jianguo Wang

- Memory disaggregation
- Additional discussions on CXL
- Future directions

Qizhen Zhang

About Me

- **Assistant Professor @ Purdue CS (01/2021 ~)**
- **Senior Researcher (07/2020 ~ 12/2020)**
  - Milvus Vector Database
- **Software Engineer (03/2019 ~ 07/2020)**
  - Amazon Aurora Cloud Database
- **PhD @ UC San Diego (09/2013 ~ 12/2018)**
- **Research Area: Database Systems for Non-traditional Architecture and Non-traditional Data**
  - Disaggregated Databases; Vector Databases

www.cs.purdue.edu/homes/csjgwang/

SIGMOD'23 Panel

Future of Database System Architectures

Gustavo Alonso
alonso@inf.ethz.ch
ETH
Zurich, Switzerland

Sam Madden
smadden@csail.mit.edu
MIT
Boston, United States

Natassa Ailamaki
anastasia.ailamaki@epfl.ch
EPFL
Lausanne, Switzerland

Swami Sivasubramanian
swami@amazon.com
AWS
Seattle, United States

Sailesh Krishnamurthy
sailesh@gmail.com
Google
Palo Alto, United States

Raghu Ramakrishnan*
raghu@microsoft.com
Microsoft
Redmond, United States

1 INTRODUCTION

Over the past two decades, we have experienced major technology disruptions on multiple fronts, none bigger than the emergence of cloud computing, which has led to fundamental changes in how database software is architected:

- Disaggregation of compute and storage, helping to disaggregate hardware and software
- Elastic resource allocation

VLDB'22 Panel

Cloud Data Systems: What are the Opportunities for the Database Research Community?

Magdalena Balazinska
University of Washington, Seattle

Surajit Chaudhuri
Microsoft Research

AnHai Doan
University of Wisconsin, Madison

Joseph M. Hellerstein
University of California, Berkeley

Hanuma Kodavalla
Microsoft

Ippokratis Pandis
Amazon Web Services

Matei Zaharia
Stanford University

Infrastructure Clouds continue to evolve. We have seen a transition from provisioned to serverless resource allocation. The disaggregation of storage and compute is common in infrastructure cloud offerings. Currently, disaggregation of memory is being actively explored. With the end of Dennard scaling, there is increasing interest in leveraging accelerators, leading to

TPCTC 2022 (VLDB 2022 Workshop) Panel: Disaggregated Database Management Systems

Date and Time: Sept 5, 10-11 am Australia Time, UTC/GMT +10 hours (Sept 4, 5-6 pm US Pacific Time)

Moderator: Shahram Ghandeharizadeh, USC

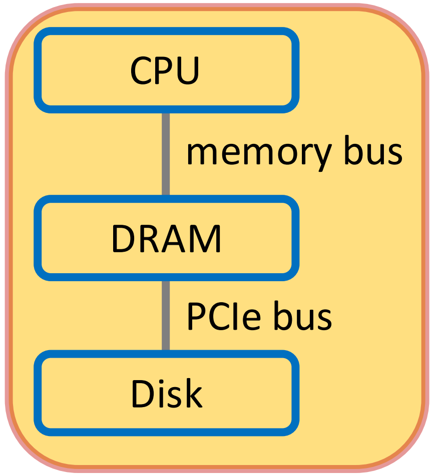
Panelists:

Phil Bernstein, Microsoft
Dhruba Borthaku, Rockset
Haoyu Huang, Google
Jai Menon, Fungible
Sumit Puri, Liqid

What're Disaggregated Databases?

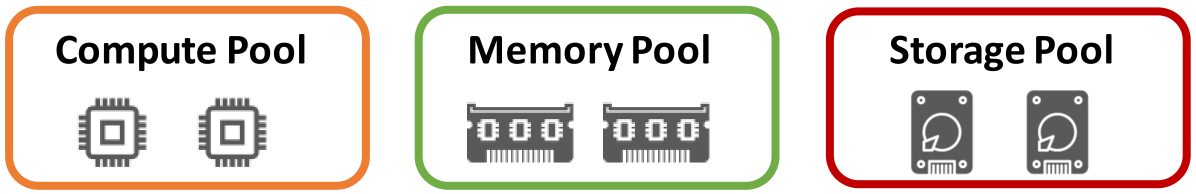
- Designed explicitly for resource disaggregation (RD)

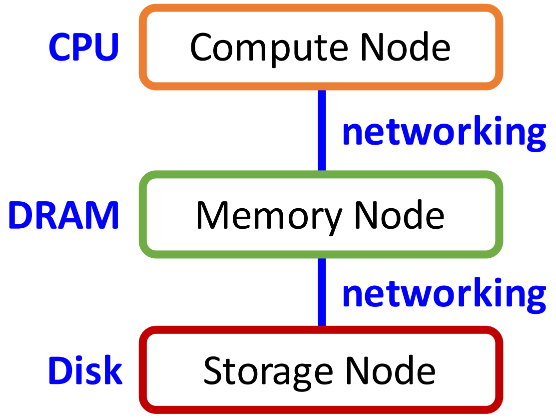
- RD is a new trend in cloud data centers that decouples resources (such as compute, storage, and memory) into separated resource pools connected via networking



Traditional Data Center

Monolithic "converged" servers with compute/memory/storage tightly coupled in physical servers



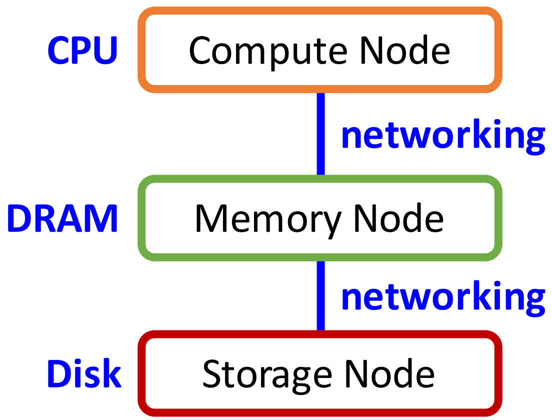


Disaggregated Data Center

Separate resources into resource pools connected via networking

Resource Disaggregation

- Compute Node
  - High computing power (e.g., 100s cores)
  - Limited local memory and local storage (flexible)
- Memory Node
  - Huge memory (e.g., 100s GBs)
  - Weak computing power (flexible)
- Storage Node
  - Huge storage (e.g., TBs)
  - Weak computing and limited memory (flexible)
- Resources are separated into different types of servers and each type may include limited (yet flexible) amount of other resources (partial disaggregation)



Benefits of Resource Disaggregation

- Independent and elastic resource scaling
  - Users can request any combinations of compute, memory, storage based on their needs
  - Very important in the cloud
- Increase resource utilization and reduce resource fragmentation
  - Due to resource pooling
  - Lead to lower cost
- Other benefits
  - Independent failure and upgrades, near-infinite pool of resources

Applications of Resource Disaggregation

- Many cloud vendors have adopted resource disaggregation to some extent and have re-designed their databases explicitly for resource disaggregation

Amazon Aurora

Microsoft® SQL Azure™ Socrates

Google AlloyDB

Alibaba Cloud PolarDB

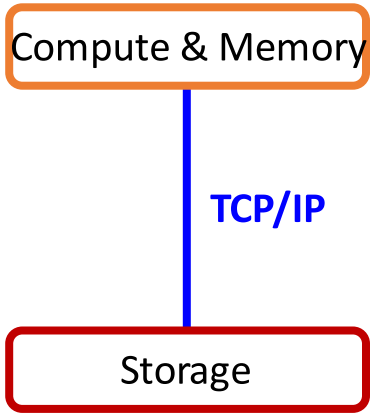
snowflake

Classification of Resource Disaggregation

Amazon Aurora

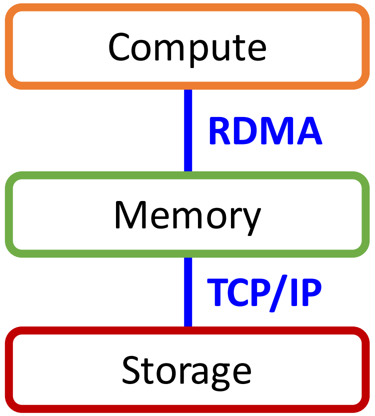
Microsoft SQL Azure Socrates

Alibaba Cloud PolarDB



Google AlloyDB

snowflake



Alibaba Cloud PolarDB

Storage Disaggregation

Only separate storage from compute & memory

Networking is usually based on TCP/IP

Widely used in cloud-native databases

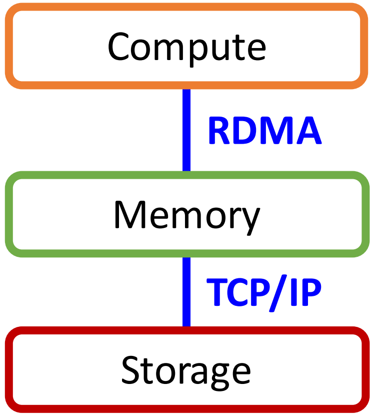
Memory Disaggregation

Further separate compute and memory

Networking is usually based on high-speed networks, e.g., RDMA

Under active investigation

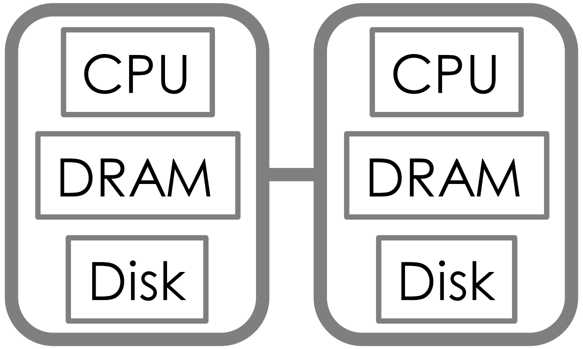
Implications of Resource Disaggregation to Database Systems



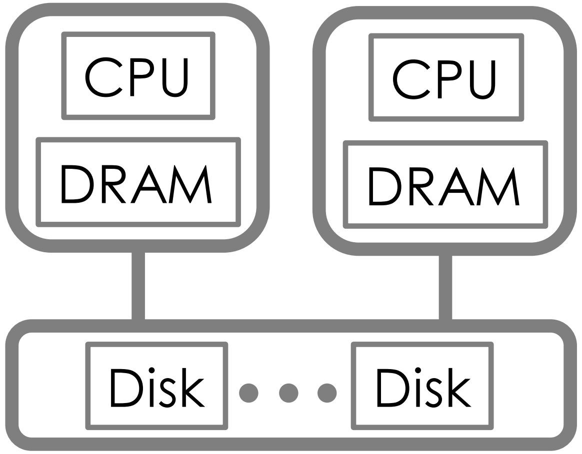
- What's the big deal to databases?
- Isn't same as traditional storage hierarchy?
- Implication 1: How to reduce networking overhead?
  - Use caching, compression (same as traditional storage hierarchy)
  - But can also use near-data computing (both the memory node and storage node have CPUs), e.g., log-as-the-database
    - Different from traditional storage hierarchy
  - This is partial disaggregation

Implications of Resource Disaggregation to Database Systems

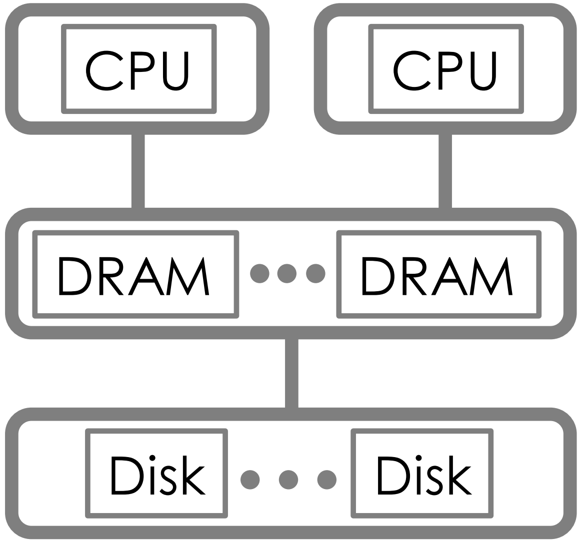
- Implication 2: How to distribute and share resources?
  - Resources are disaggregated ➔ enable sharing, e.g., different compute nodes can share the same storage or memory
  - Use distributed shared-storage (or shared-memory) architecture for distributed databases, instead of the golden standard of shared-nothing
  - Support much better elasticity and independent resource scaling ➔ key to implement serverless databases



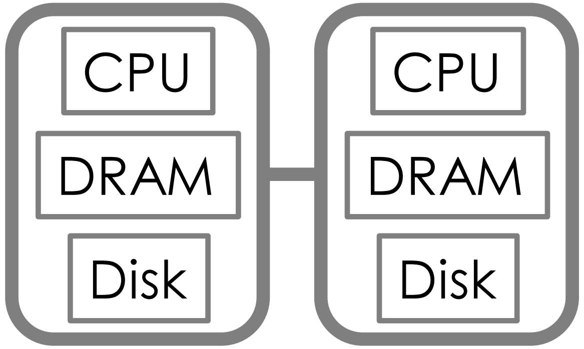
Distributed Shared-Nothing



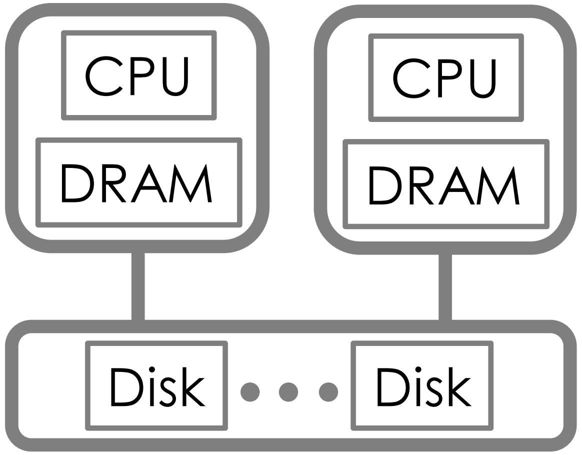
Distributed Shared-Storage



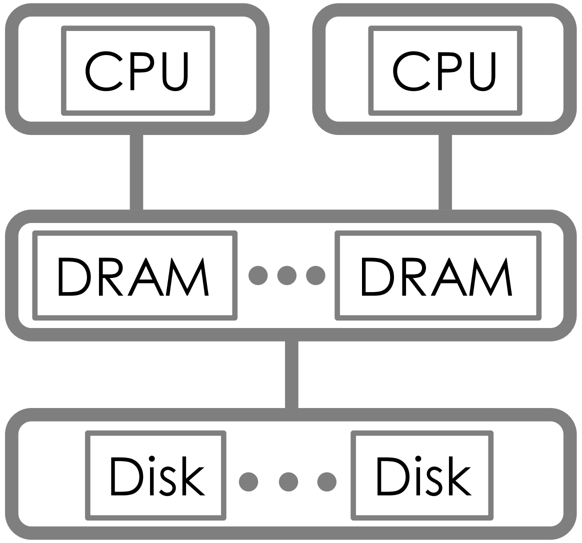
Distributed Shared-Memory



Distributed Shared-Nothing



Distributed Shared-Storage



Distributed Shared-Memory

R SYSTEMS

ORACLE

MySQL Cluster

PostgreSQL

Amazon Aurora

Microsoft SQL Azure Socrates

Google AlloyDB

The Case for Distributed Shared-Memory Databases with RDMA-Enabled Memory Disaggregation

Ruihong Wang, Purdue University, wang4996@purdue.edu
Jianguo Wang, Purdue University, csjgwang@purdue.edu
Stratos Idreos, Harvard University, stratos@seas.harvard.edu
M. Tamer Özsu, University of Waterloo, tamer.ozsu@uwaterloo.ca
Walid G. Aref, Purdue University, aref@purdue.edu

VLDB'23

Wang et al. The Case for Distributed Shared-Memory Databases with RDMA-Enabled Memory Disaggregation. VLDB 2023.

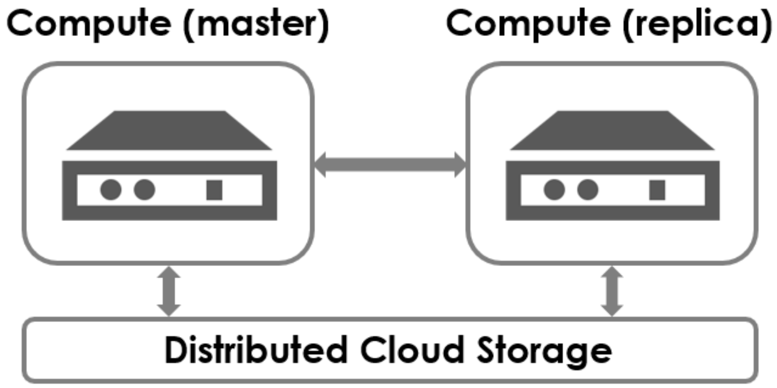
Outline

- Introduction and motivation
- **Storage disaggregation**
- Additional discussions on PM
- Memory disaggregation
- Additional discussions on CXL
- Future directions

- **OLTP databases**
  - Amazon Aurora
  - Microsoft Socrates
  - Google AlloyDB
  - Alibaba PolarDB
- **OLAP databases**
  - Snowflake
  - Amazon Redshift

Why Storage-Compute Disaggregation?

- Independent scaling of compute and storage
  - Best for any workload, lower cost
- Easy for serverless
  - Can shut down all compute nodes and start quickly
- Faster scaling & crash recovery
  - If you switch to a bigger instance (or a new instance due to crash), no need to move data ➔ much faster

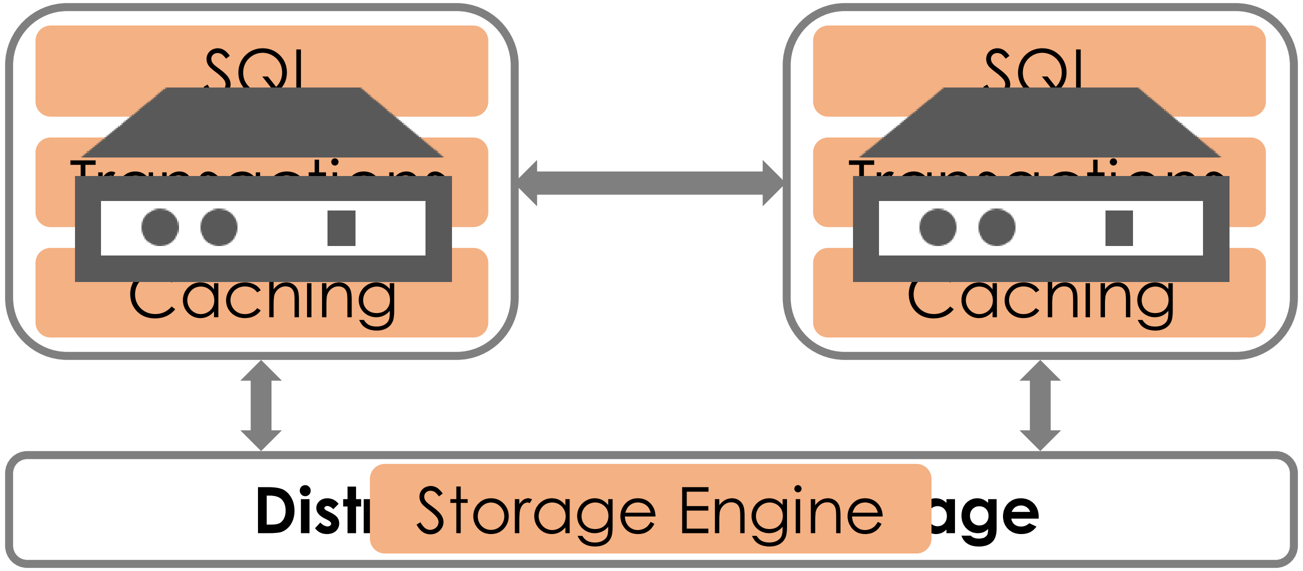


Aurora System Architecture

Designed for storage & compute disaggregation

Goal
Reduce network I/O

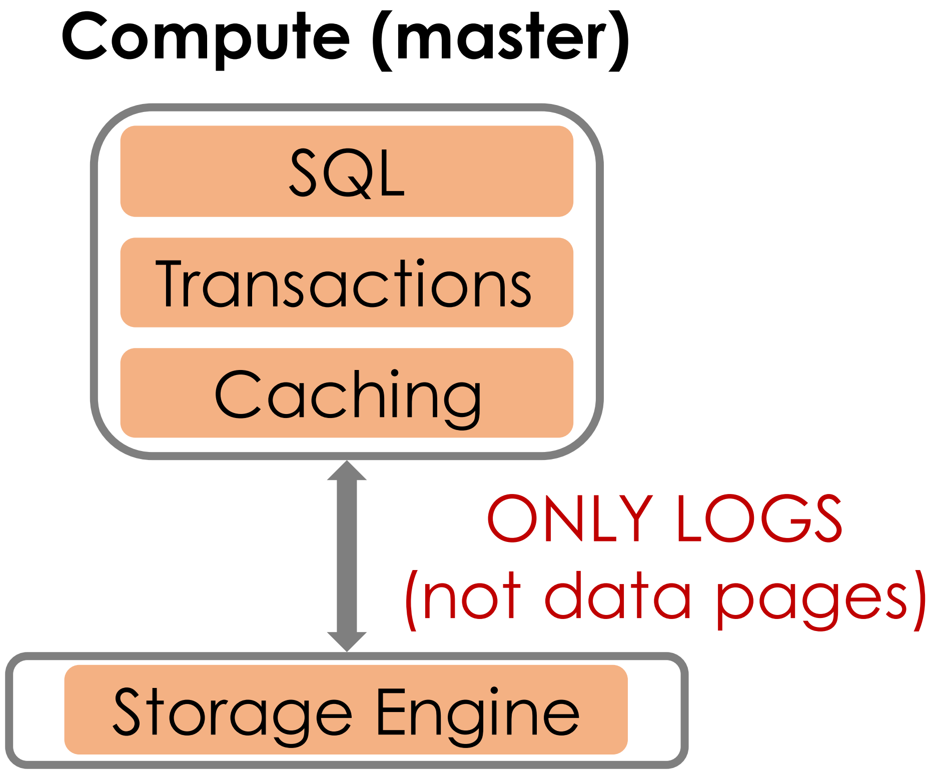
Compute (master) Compute (replica)



Verbitski et al. [Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases](). SIGMOD 2017.

Main Ideas in Aurora

- Log is the database
  - Only write logs on network
  - Push log applicator to storage tier
- Asynchronous processing
  - Materialize pages in background (storage engine)
- Buffer cache
  - To avoid network I/O
  - Can read pages upon cache miss



How Does Database Work with Logs?

Read

Write

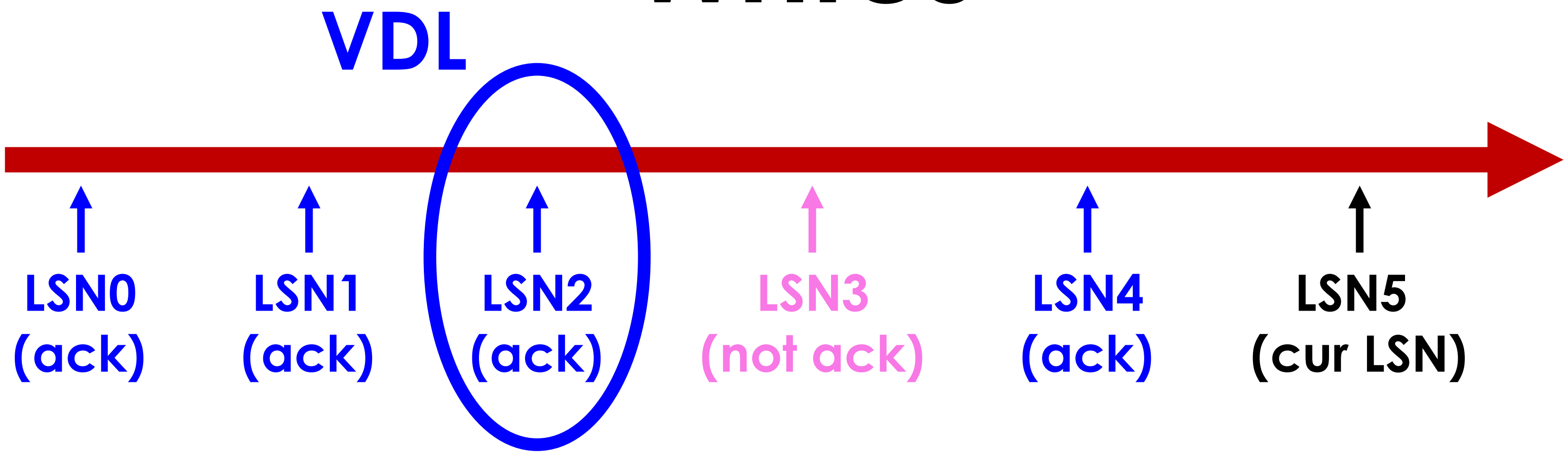
Cache

Commit

Replication

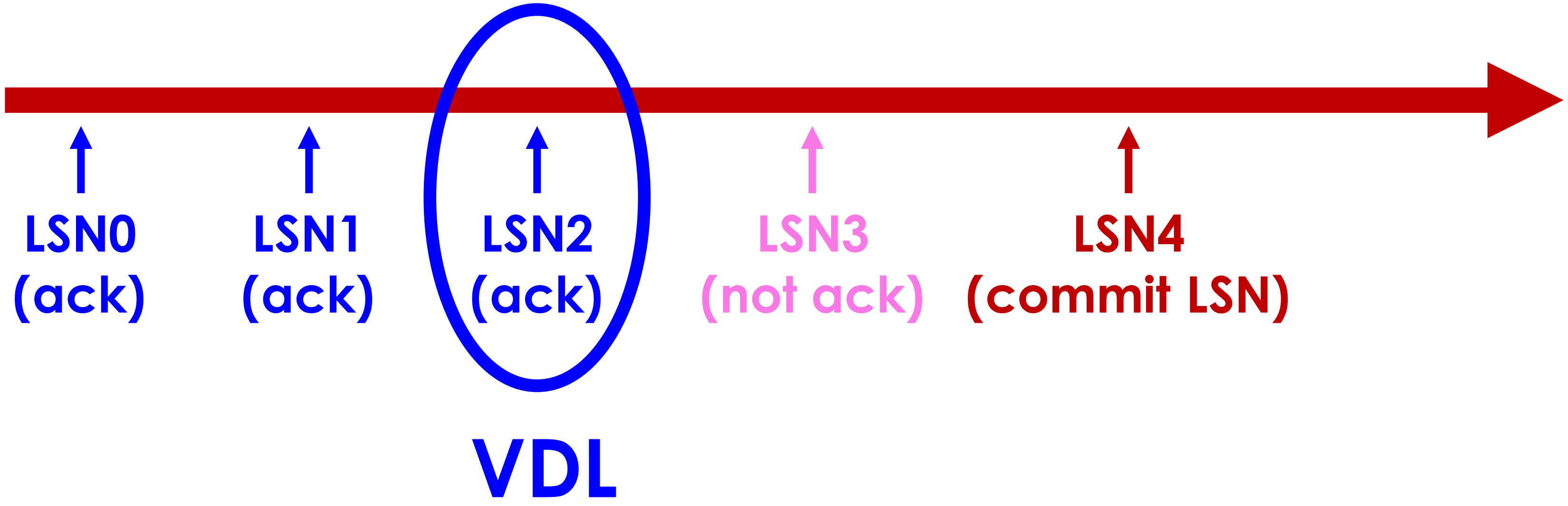
Recovery

Writes



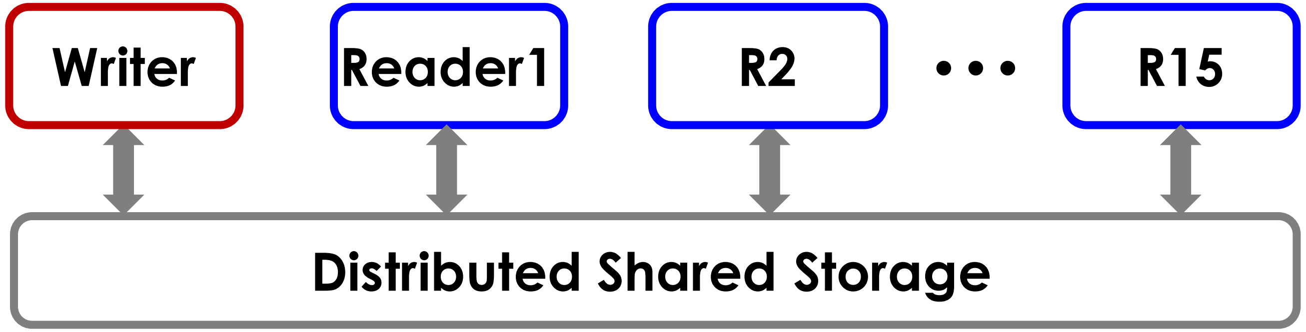
- DB generates logs for all trxns
- Writes (trxns) send logs to storage asynchronously
- Durability: each log is durable (ack) with 4/6 quorums
- Volume Durable LSN (VDL)
  - As log records can be lost, out of order
  - VDL: the largest one with all prior LSNs are durable

Transaction Commits



- Transaction commits asynchronously
- When a transaction commits, mark its commit LSN
- Commit only if VDL >= commit LSN

Replication: Scalability



- 1 writer and up to 15 reader instances
- How to keep data consistent between writer and readers?
  - The writer sends logs to readers at the same time
  - Once the reader receives logs, it will check if the page is in the cache
    - If yes, apply the log; Otherwise, discard it
  - Replication lag: 20ms

Reads (Caching)

- Each reader instance has a buffer (cache)
- Upon reads, check cache first
  - The cache is supposed to contain the latest data pages
  - Except replication lags
- What if the cache is full?
  - Always evict a clean page: a page that's durable (pageLSN <= VDL)
  - Why? Otherwise, need to write dirty pages to storage, which increases network overhead

Crash Recovery

Traditional Databases

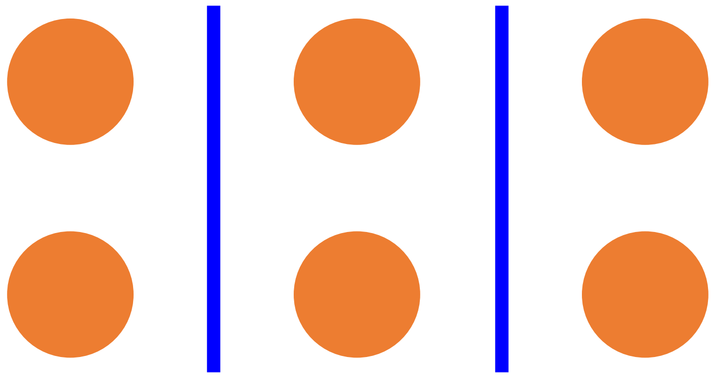
- Have to replay logs since the last checkpoint
- Typically 5 minutes between checkpoints
- Single-threaded in MySQL; requires a large number of disk accesses

Aurora

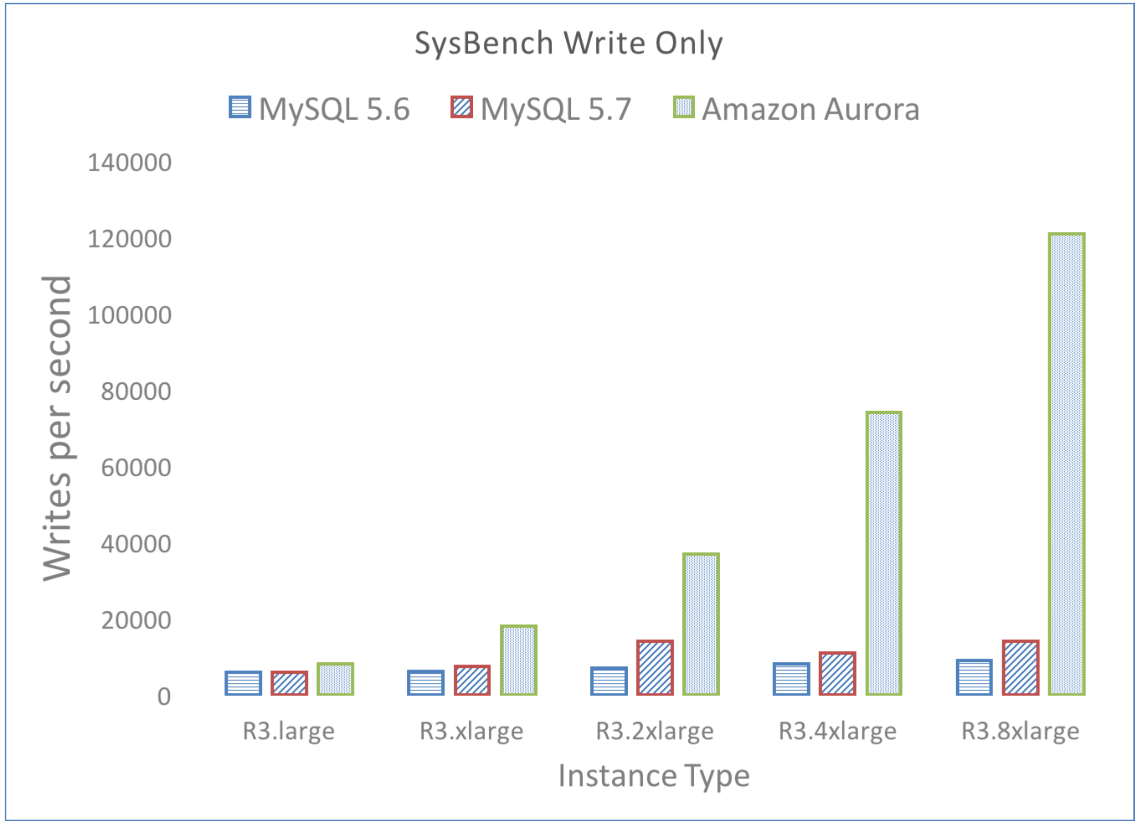
- No need to replay logs and generate pages during recovery (very fast)
- Just need to re-establish some states (e.g., VDL) ➔ make sure storage is consistent
- Generate pages asynchronously, in parallel
- Typically a few seconds

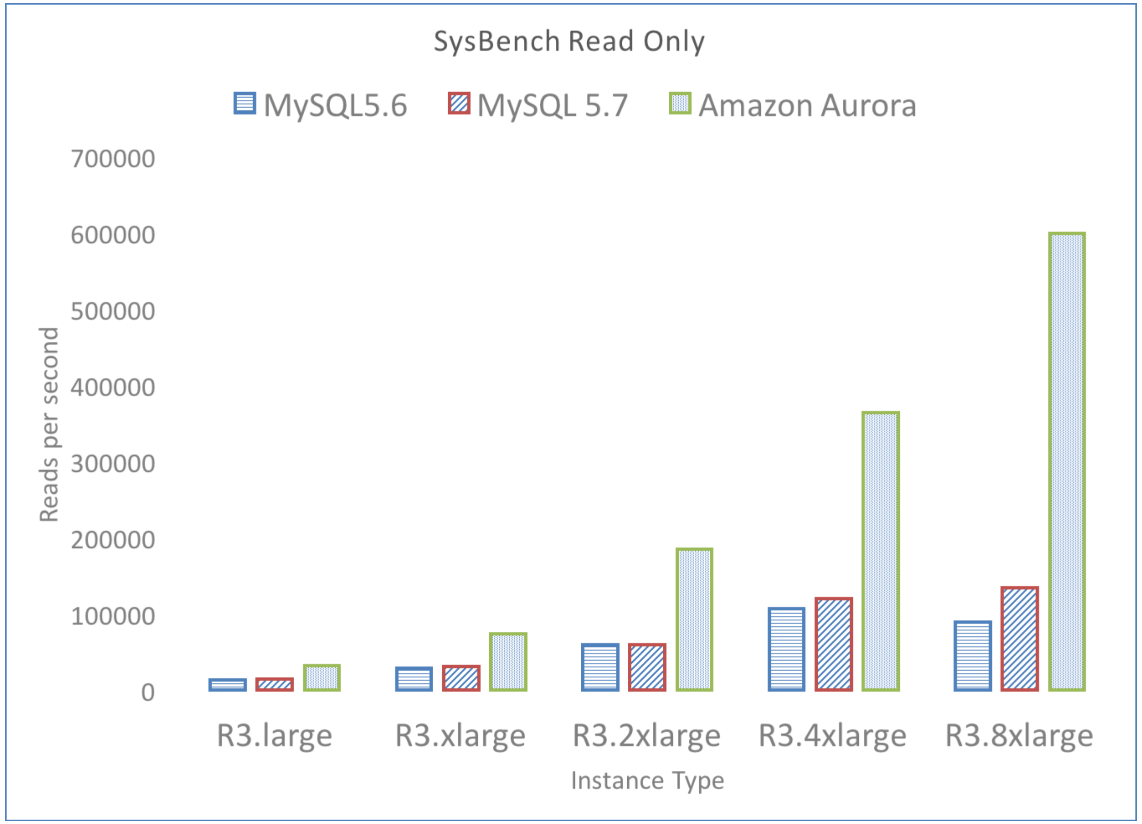
Aurora Storage

- Why not using S3 or EBS? Why developing a new distributed storage?
  - Highly available, 6-way replication
    - 2 copies per AZ
    - Write quorum 4/6; read quorum 3/6
  - Offloading logs (and allow other optimizations)



Experiments





Up to 5x faster than Cloud MySQL

Microsoft Socrates

- Similar to Amazon Aurora, it's also designed for storage-compute disaggregation
- Key difference: separate log service from page service
  - Philosophy: separate durability (implemented by logs) from availability (implemented by pages)
  - Allow customized optimizations for logs and pages
    - Durability does not require copies in fast storage
    - Availability does not require a fixed number of replicas

Antonopoulos et al. [Socrates: The New SQL Server in the Cloud]. SIGMOD 2019.

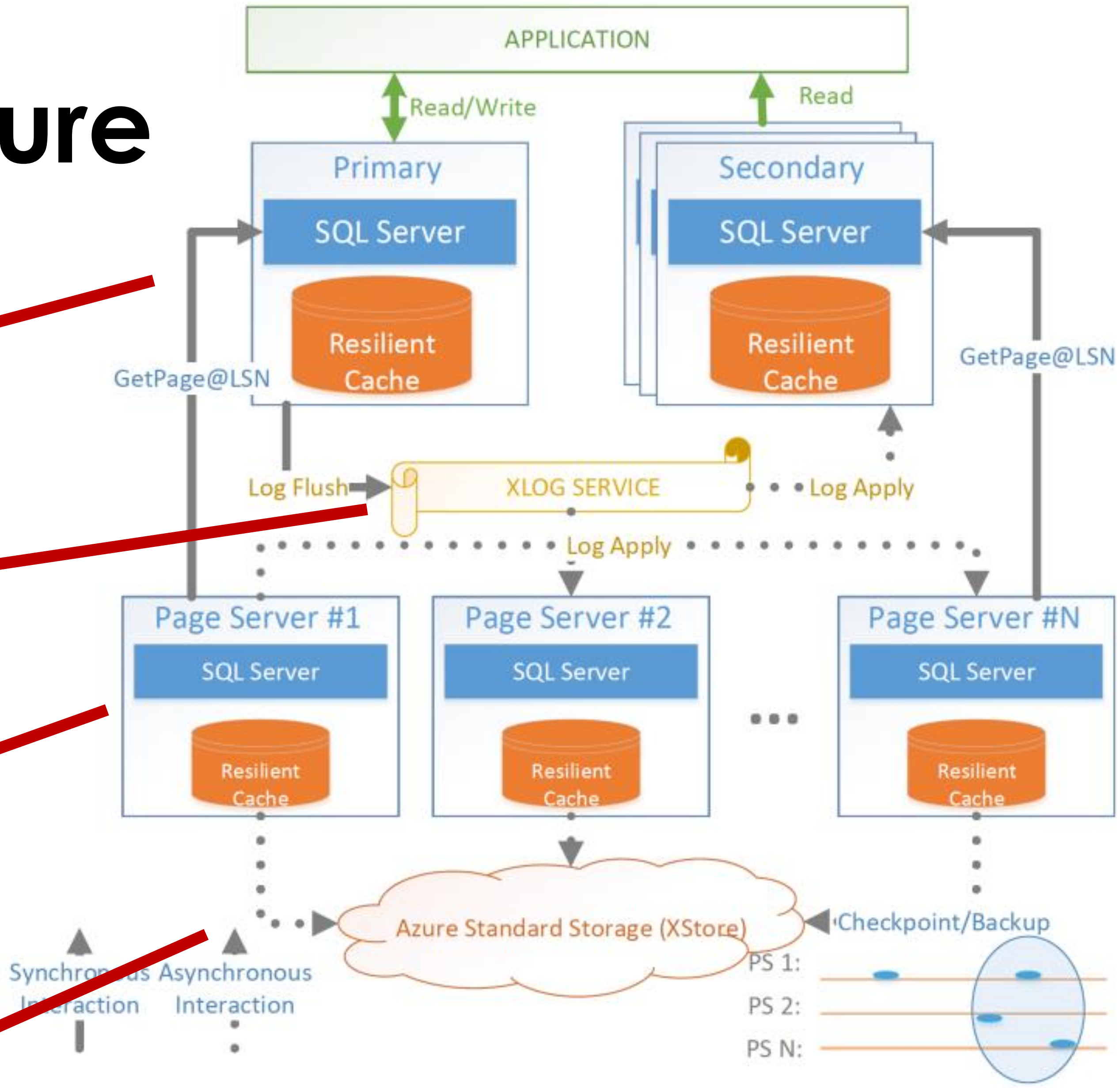
Socrates Architecture

Compute Layer

Log Layer

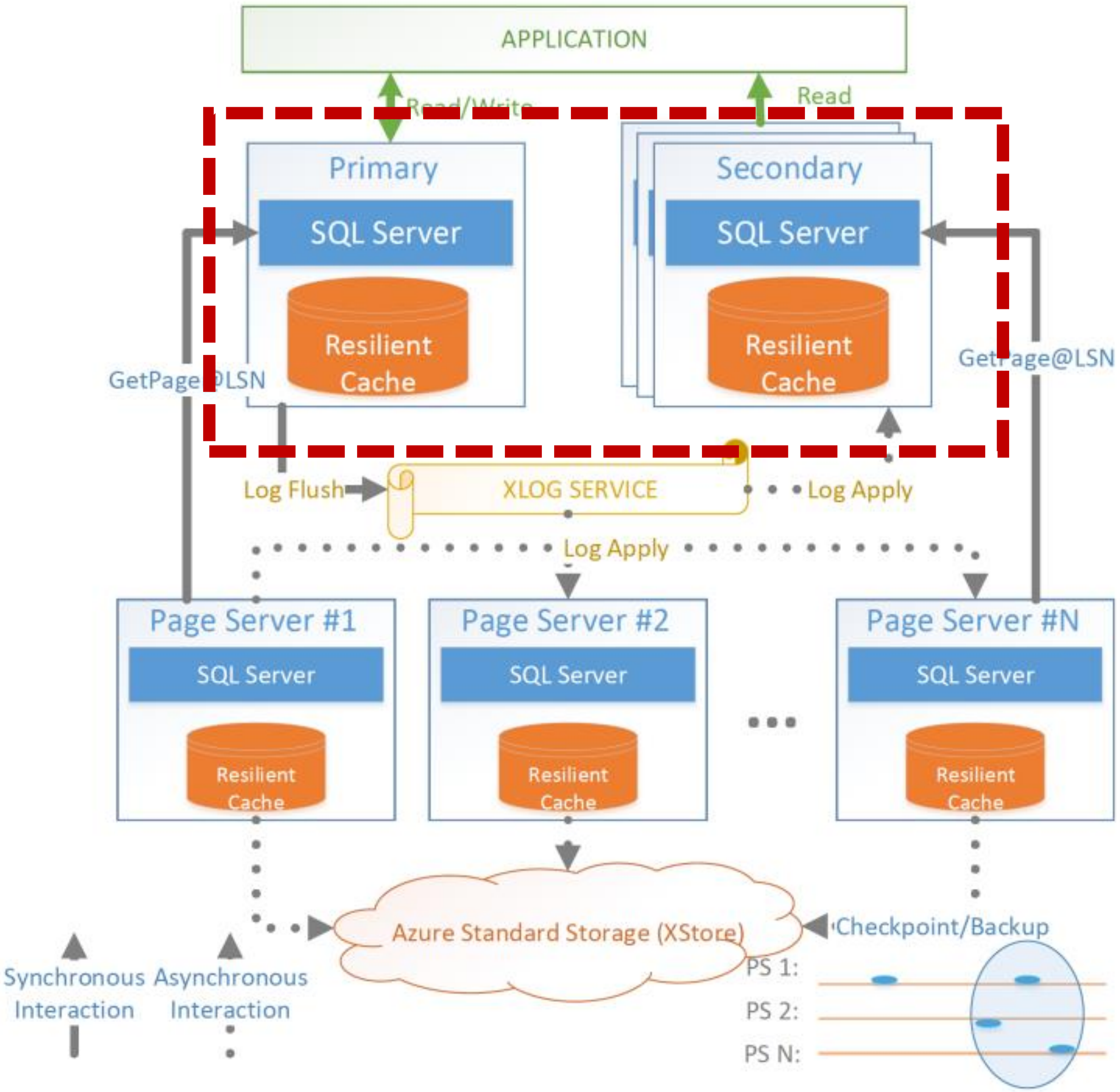
Storage Layer
(Page Servers)

Azure Storage
(Standalone Service)



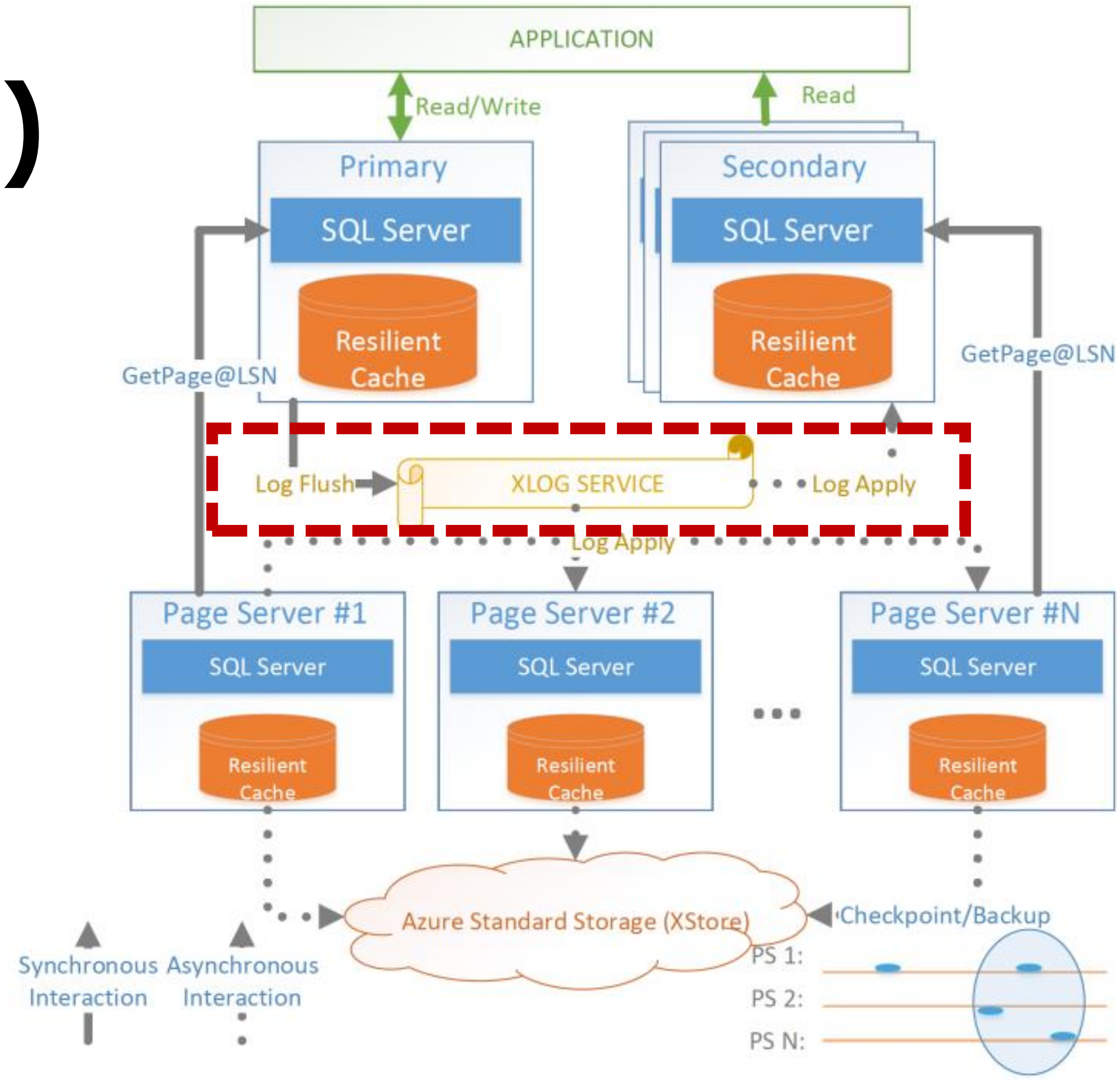
Compute Layer

- One primary node (writer) and multiple secondary nodes (readers)
- Compute nodes cache data in memory and local SSDs (ephemeral) if any
- Support SQL engine, optimization, buffer, transactions
- Send only logs to XLOG service



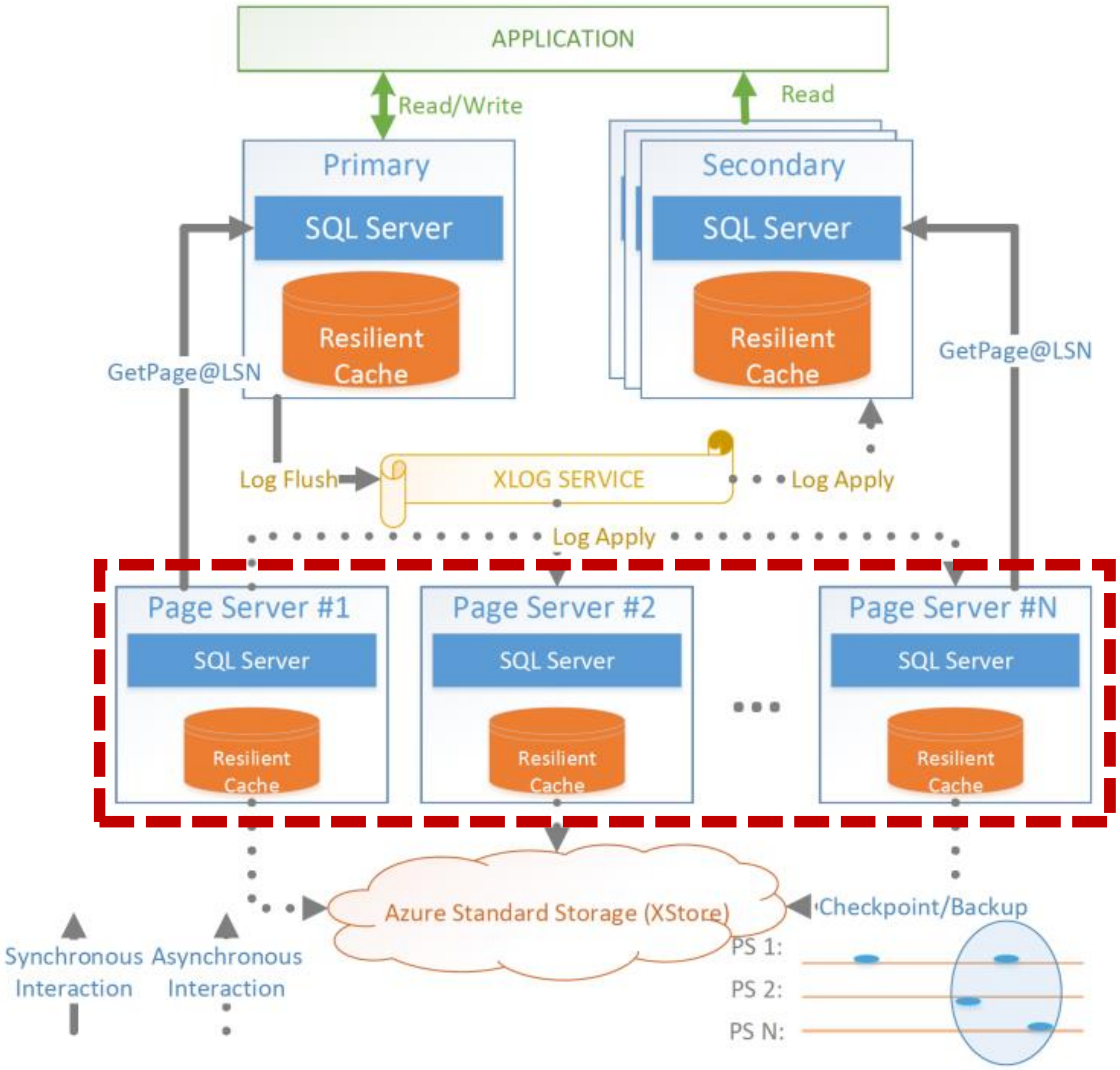
Log Layer (XLOG)

- For fast durability
- Expensive but small SSDs
  - Persist recent logs
  - Three replicas
- Logs are flushed to XStore
- Page servers consume the logs in an asynchronous way



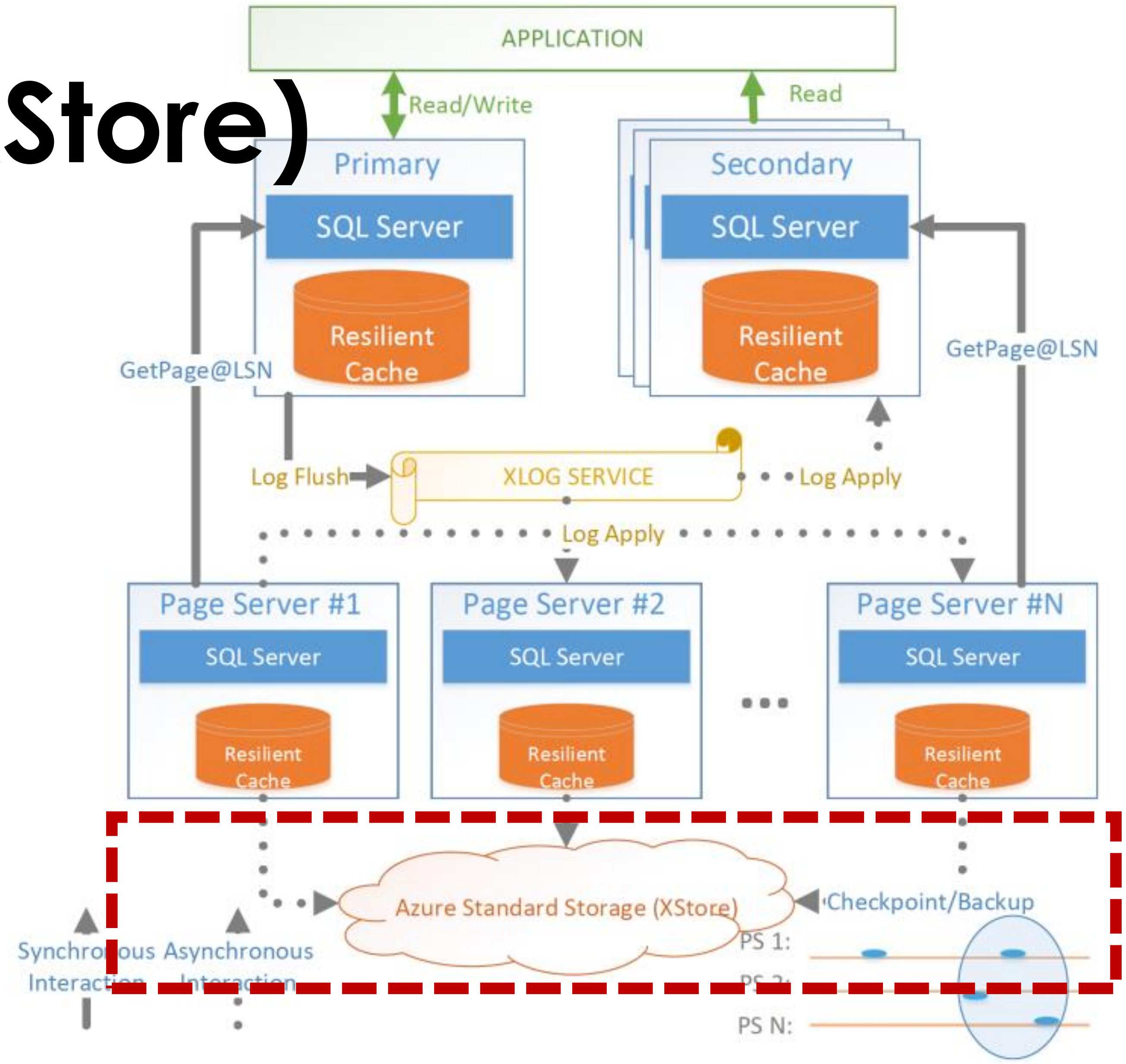
Storage Layer

- Store the actual pages
- Replay the logs
- Each page server stores a partition of the database
- Has local SSDs
- No replicas in this layer
  - Backup via the XStore



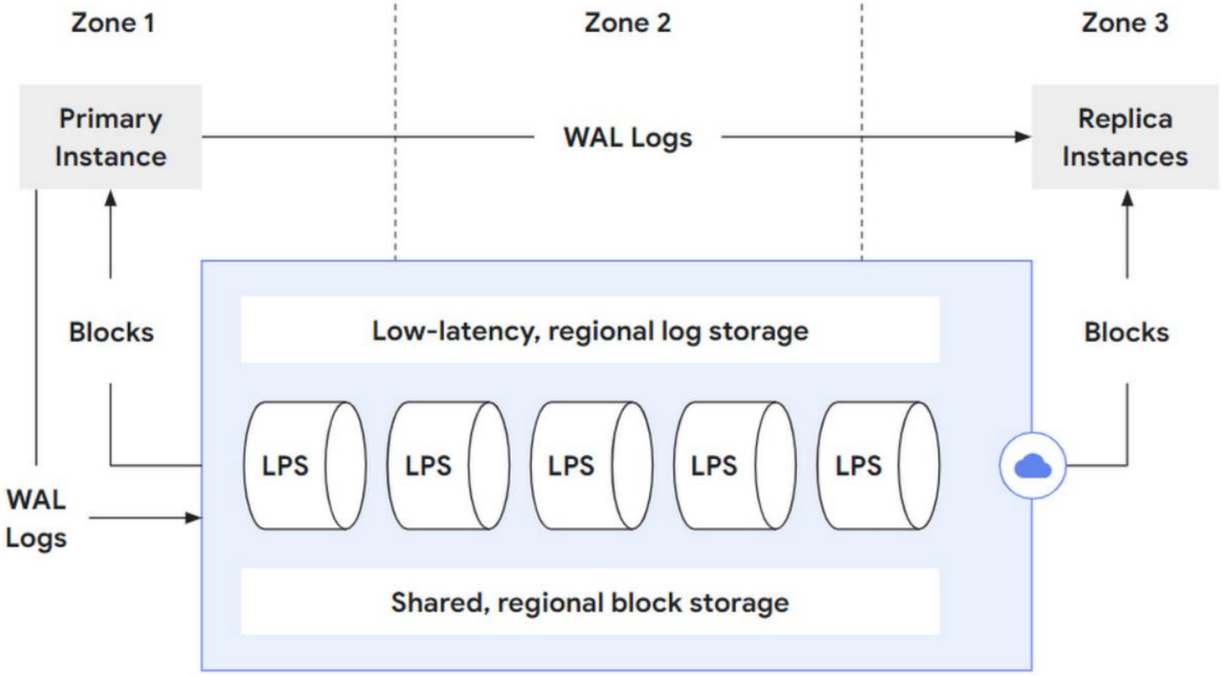
Azure Storage (XStore)

- Highly scalable, durable, and cheap storage service based on slow hard disks
- Compute nodes and page servers are stateless and they can fail at any time without data loss
- The "truth" of the database is stored in XStore and XLOG



Google AlloyDB Architecture

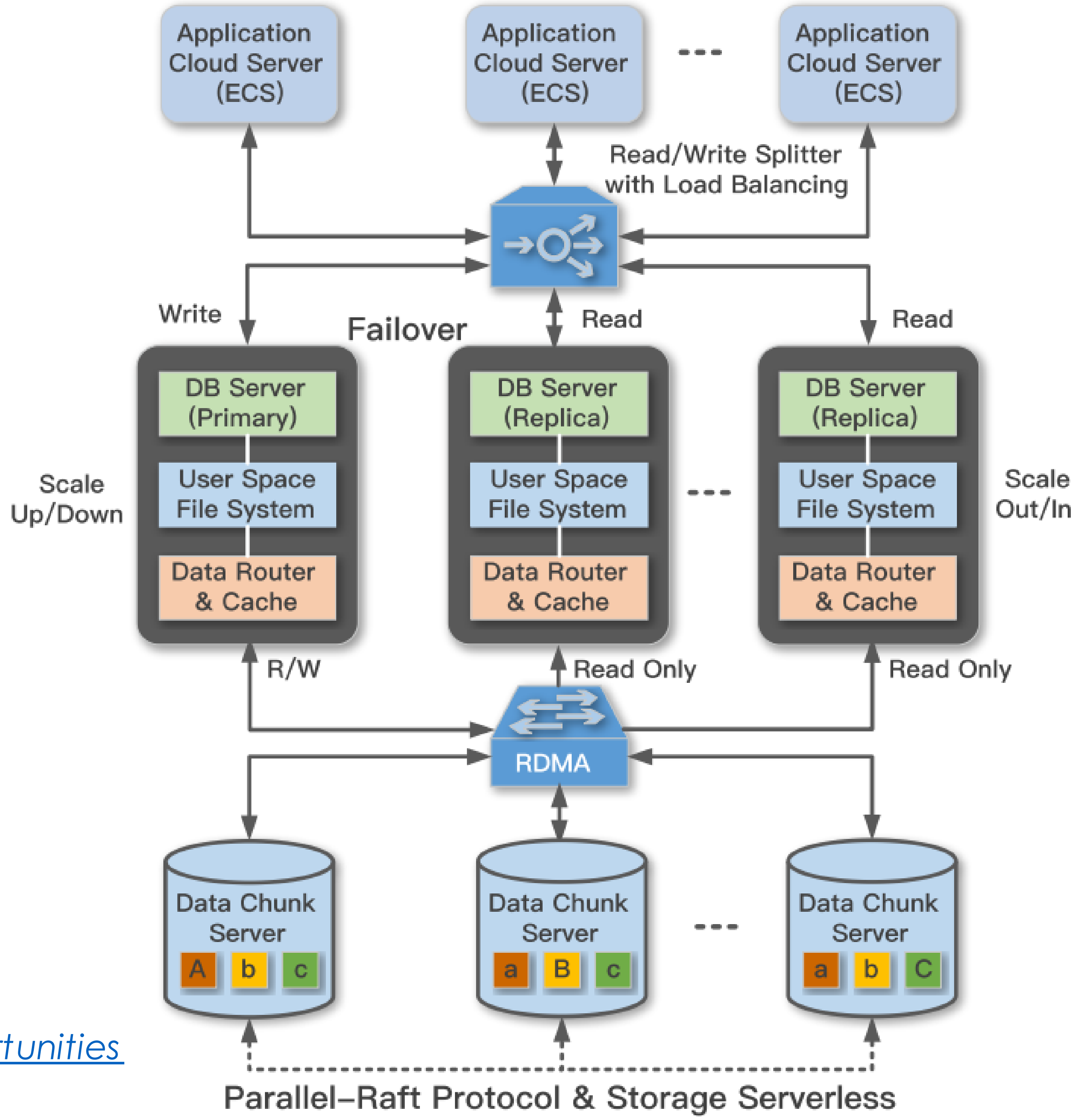
- Similar to Aurora
- Log-as-the-database
- One primary and multiple replicas
- The storage layer is based on GFS (Colossus)
- Public information is limited (no papers yet)
  - Product released in 2022



https://cloud.google.com/blog/products/databases/alloydb-for-postgresql-intelligent-scalable-storage

Alibaba PolarDB

- Similar to Aurora
- Differences
  - Send both data and logs
  - Use RDMA for fast data transfer
  - Based on PolarFS (no need log replay etc.)
  - Support memory disaggregation and HTAP



Li. [Cloud-Native Database Systems at Alibaba: Opportunities and Challenges](). VLDB 2019.

Outline

- Introduction and motivation
- **Storage disaggregation**
- Additional discussions on PM
- Memory disaggregation
- Additional discussions on CXL
- Future directions

- **OLTP databases**
  - Amazon Aurora
  - Microsoft Socrates
  - Google AlloyDB
  - Alibaba PolarDB
- **OLAP databases**
  - Snowflake
  - Amazon Redshift

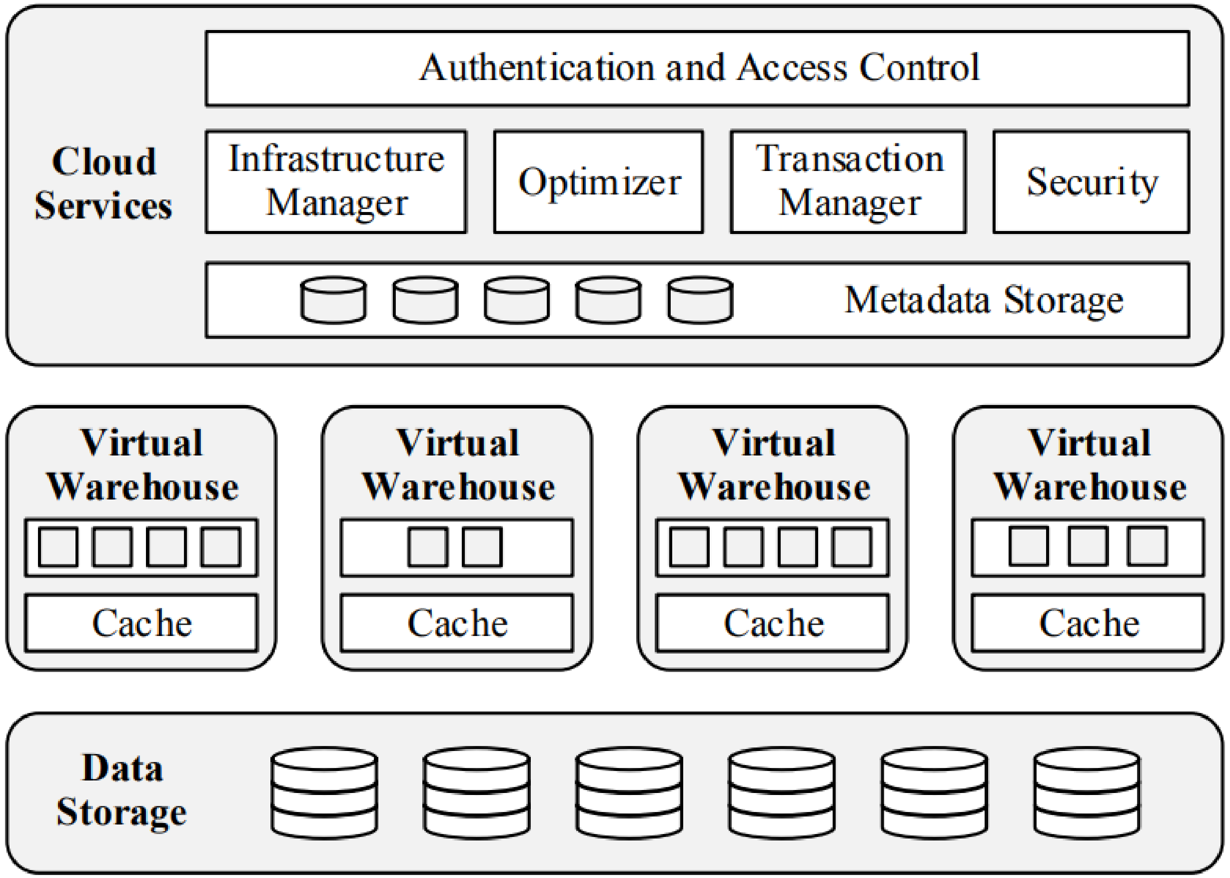
Snowflake Data Warehouse

Storage compute separation and distributed shared-storage

Control Layer

Compute Layer

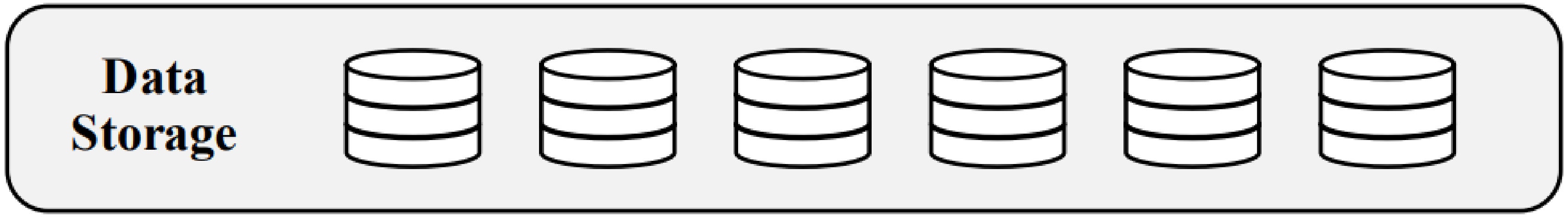
Storage Layer



Dageville et al. The Snowflake Elastic Data Warehouse. SIGMOD 2016.

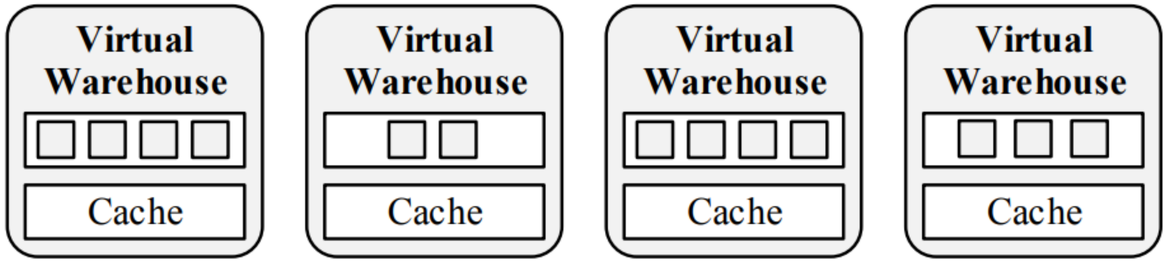
Storage Layer

- Based on S3 for high availability and durability
  - Slow but reliable and cheap
  - Rely on caching in the compute layer for high performance
- Partition table into files (micro-partitions)
  - Each file is around 16MB
- PAX hybrid columnar storage format within each file
- Storage is shared by all the compute nodes



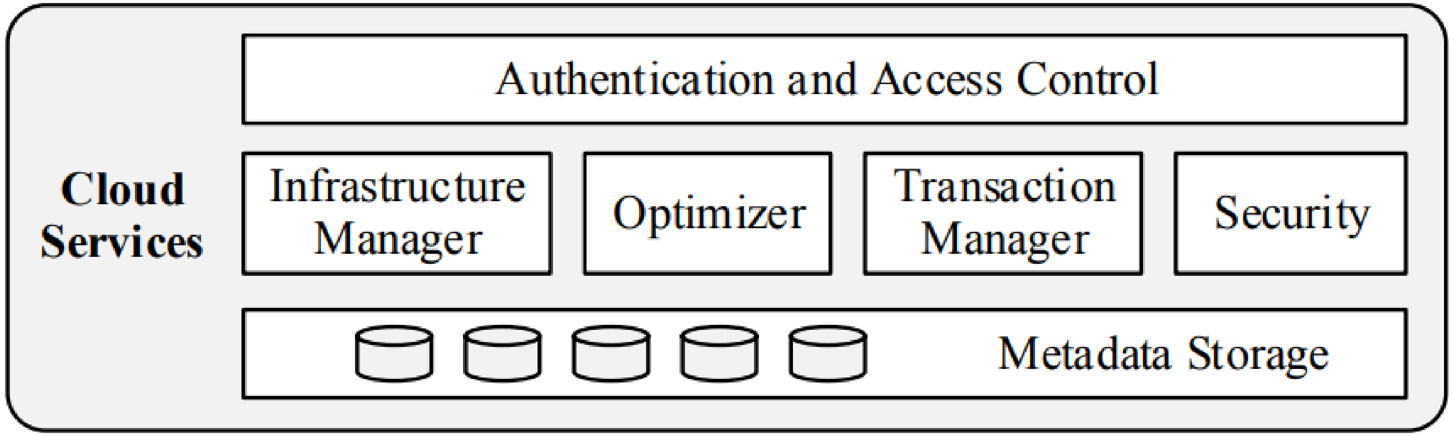
Computer Layer

- Virtual Warehouse (VW)
  - A set of EC2 instances (worker nodes) for the actual query processing and execution
  - Similar to MPP databases
- Elasticity
  - Created, destroyed, resized on demand
  - Users may shut down all warehouses when they have nothing to run
  - Sizing from X-Small to XX-Large



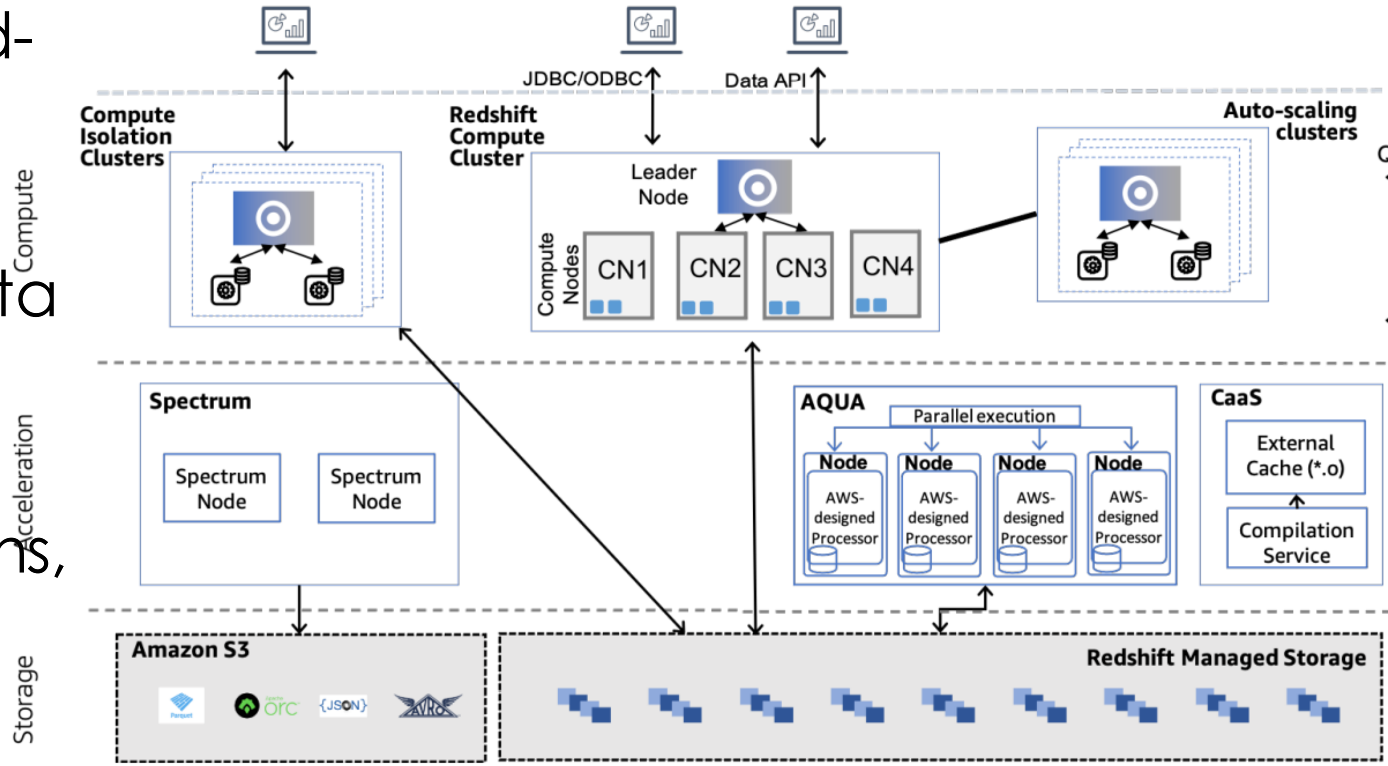
Control Layer

- The brain of the system to control and manage the system
- It's a collection of services that manage virtual warehouses, queries, transactions, concurrency control, multi-tenancy…
- Metadata information, e.g., min-max info for pruning



Amazon Redshift

- Initially, an MPP database (shared-nothing)
- Now support storage-compute separation with RMS to moves data from local storage to S3 automatically (storage scaling)
- Also introduces many optimizations, e.g., compression, query compilation, offloading, FPGA acceleration, ML…



Armenatzoglou et al. Amazon Redshift Re-invented. SIGMOD 2022.

Outline

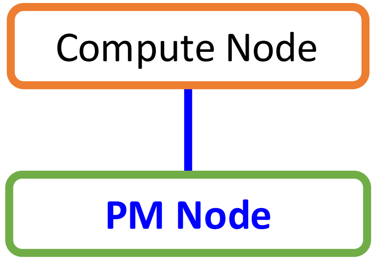
- Introduction and motivation
- Storage disaggregation
- **Additional discussions on PM**
- Memory disaggregation
- Additional discussions on CXL
- Future directions

Persistent Memory (PM)

- PM (or non-volatile main memory) is a new storage technology (many research papers in the last few years)
  - Performance is similar to DRAM
  - But durable as SSDs
- As we have storage disaggregation, how about PM disaggregation? What're the benefits?

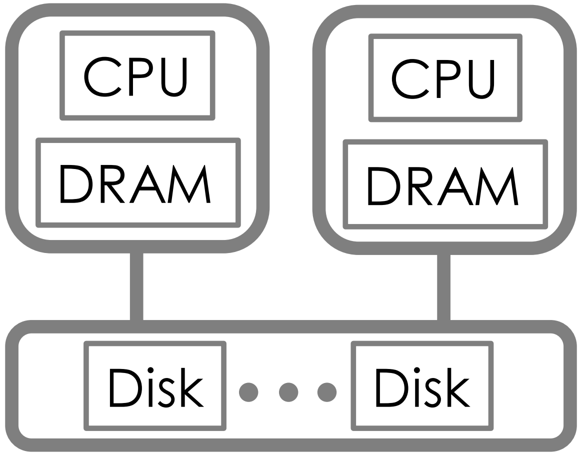
PM Disaggregation

- Besides of the benefits of storage disaggregation, e.g., independent and elastic scaling, what're new benefits?
  - PM server is expensive ➔ disaggregation enables sharing, which takes lower amortized cost
  - Can be cheaper overall as compute nodes do not need so much local memory anymore
    - Can leverage the cloud instances with leftover CPUs but limited memory
  - Can support faster recovery with huge data in PM (faster warm up)

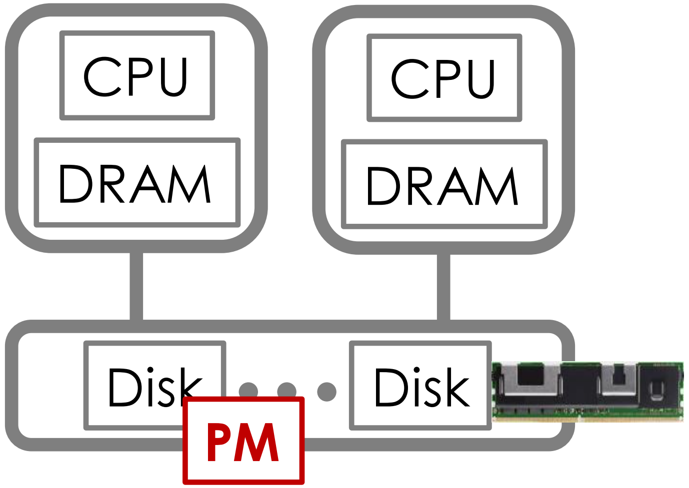


Challenges of PM Disaggregation

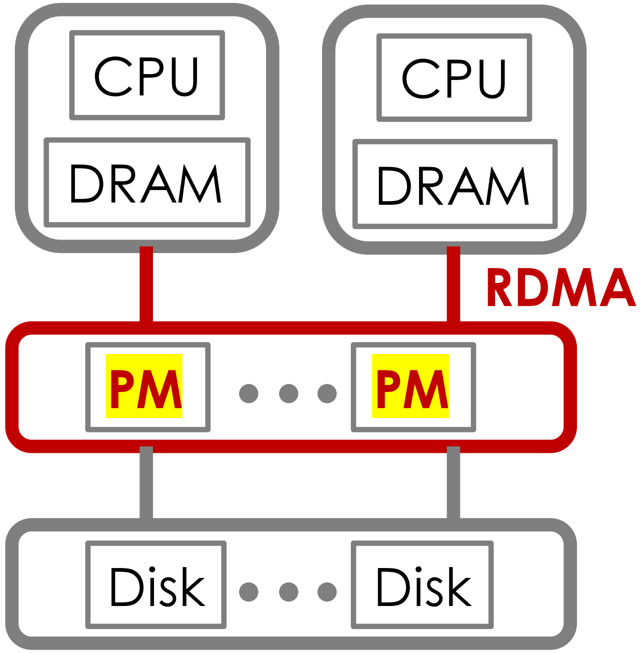
- Shall we have a dedicated PM node (layer)?



Storage Disaggregation



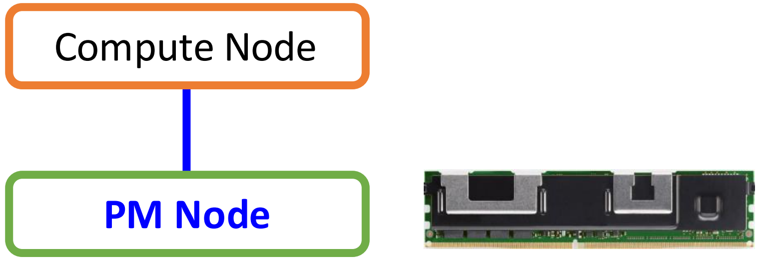
No Dedicated PM Layer
(Just add PM to existing storage servers)



With Dedicated PM Layer
(Need faster networking)

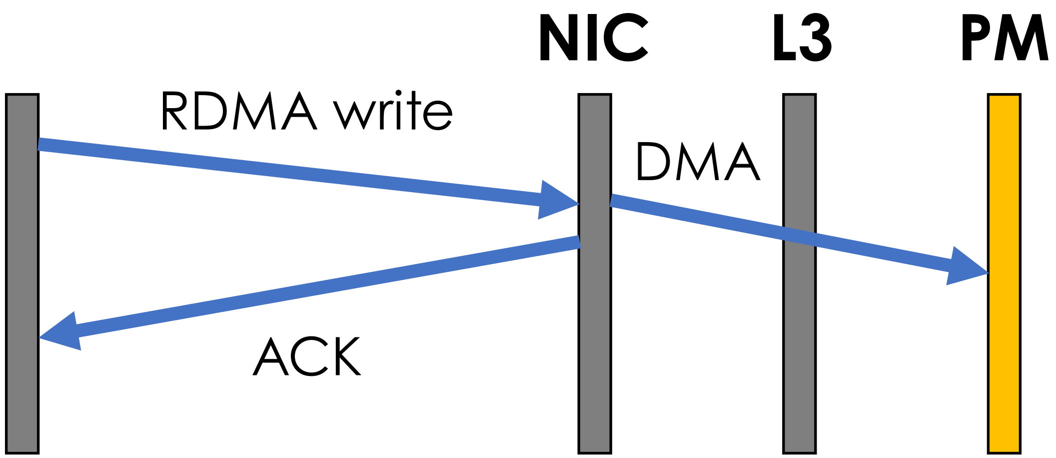
Challenges of PM Disaggregation

- How to leverage the CPU in the PM node?
  - Can be a lot (not limited CPU as in storage disaggregation)
  - E.g., Intel Optane PM needs high-end CPUs (3rd Gen Intel Xeon)



Challenges of PM Disaggregation

- Limited write bandwidth (still slower than reads)
- Remote persistency is tricky
  - Simple RDMA write to PM will not guarantee persistency
  - It requires one more RDMA read



Kalia et al. Challenges and Solutions for Fast Remote Persistent Memory Access. SoCC'20.

PilotDB: Persistent Memory Disaggregation for Cloud-Native Relational Databases

Compute Layer

PM Layer (Logs)

Storage Layer (Pages)

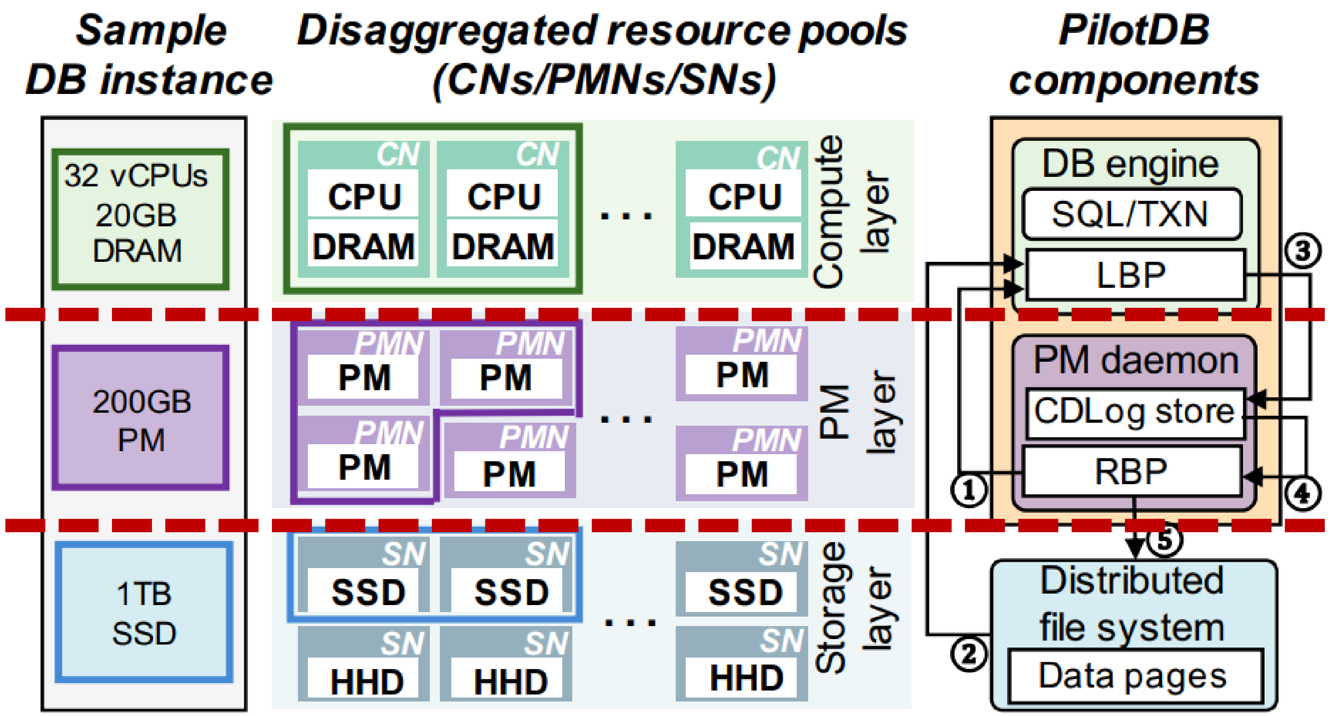
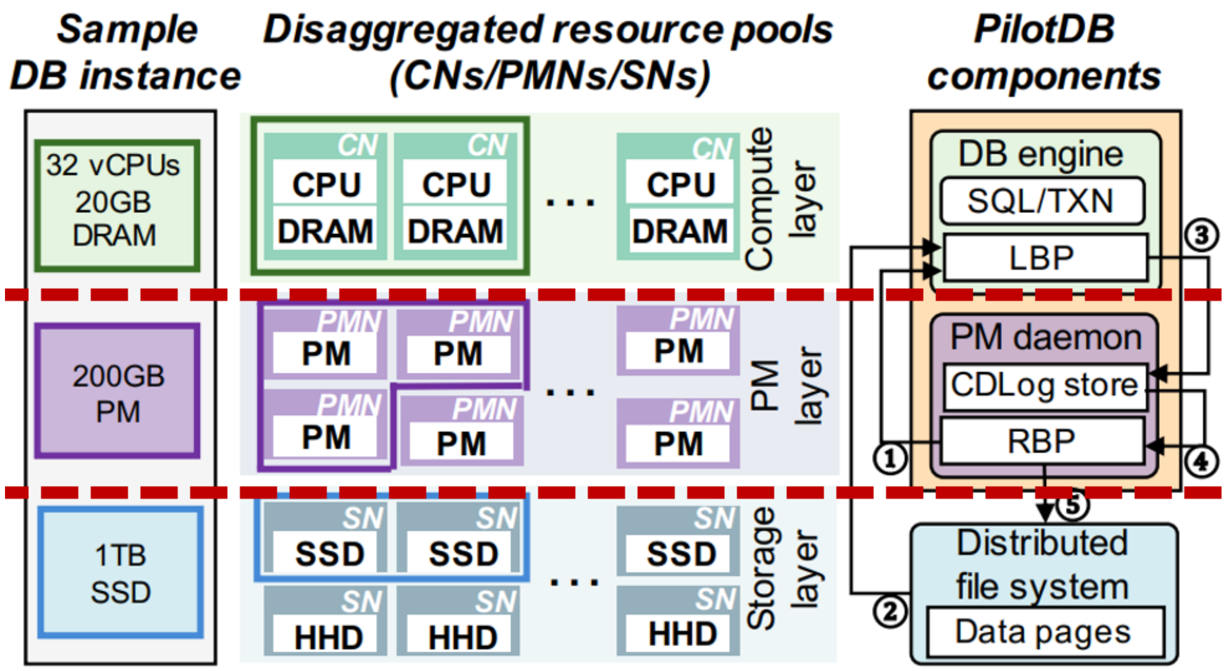


Figure 2: PilotDB architecture overview

Ruan et al. Persistent Memory Disaggregation for Cloud-Native Relational Databases. ASPLOS 2023.

PilotDB Architecture

- Reads
  - Check local buffer (LBP)
  - Then remote PM buffer (RBP)
- Write redo logs to PMN
- Replay the log in the PMN
  - To use the CPUs
- PMN flushes cold page to storage layer, when it is under space pressure



PilotDB Optimized RDMA Reads

- Compute node uses one-sided RDMA read to fetch pages from PM node
- But how to guarantee the page is already replayed?
  - The CN checks the LSN of the page against the LSN in PMT
  - If the page is outdated, the CN pulls relevant logs and performs replay

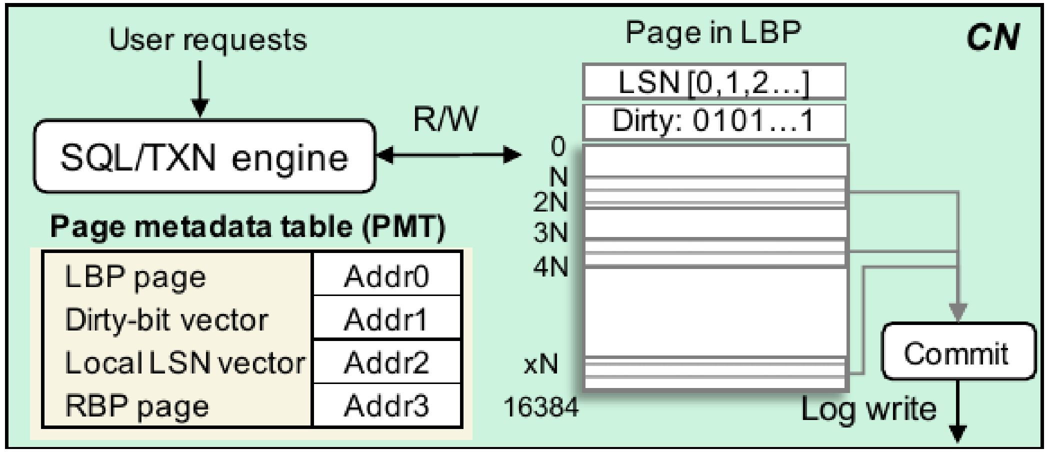


Figure 3: PilotDB CN architecture and page structure

PilotDB Replication and Recovery

- Replication
  - Logs are replicated in PM layer
  - Pages are stored only once (PM is still expensive)
- Recovery
  - If CN fails, recover quickly from PM layer (fast warm-up)
  - If PM fails:
    - If PM back online directly, just recover the connection and the system is good to go
    - If PM node not available anymore, refetch the page from the storage layer and reply the logs

Outline

- Introduction and motivation
- Storage disaggregation
- Additional discussions on PM
- **Memory disaggregation**
- Additional discussions on CXL
- Future directions

Jianguo Wang (Introduction and motivation; Storage disaggregation; Additional discussions on PM)

Qizhen Zhang (Memory disaggregation; Additional discussions on CXL; Future directions)

Memory-disaggregated DBMSs

Qizhen Zhang

University of Toronto

Outline

- Introduction to memory disaggregation
- Performance implications for DBMSs
- Memory-disaggregated transactional systems
- Memory-disaggregated analytical systems
- CXL-based memory disaggregation
- Future directions

Covered Work

Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

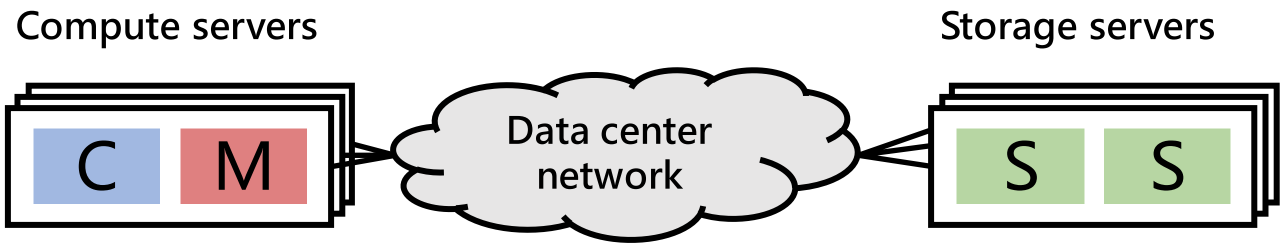
55

Introduction to memory disaggregation

56

Storage Disaggregation

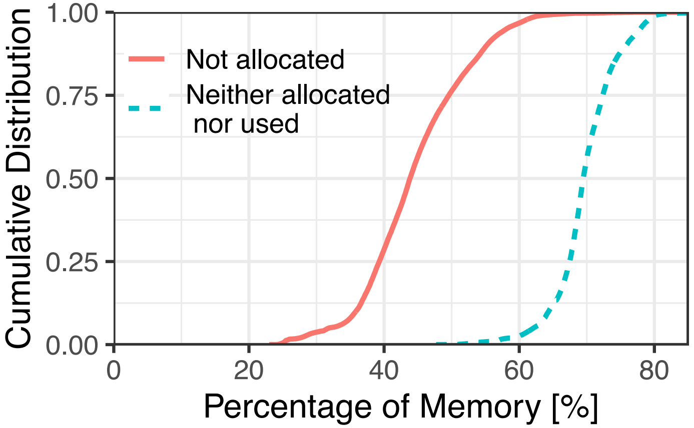
- Separating compute and storage



Storage Disaggregation

- Separating compute and storage

- Compute and memory are still coupled
  - Inflexible compute and memory allocation
  - L ... ticity
  - S ... mpute server failures

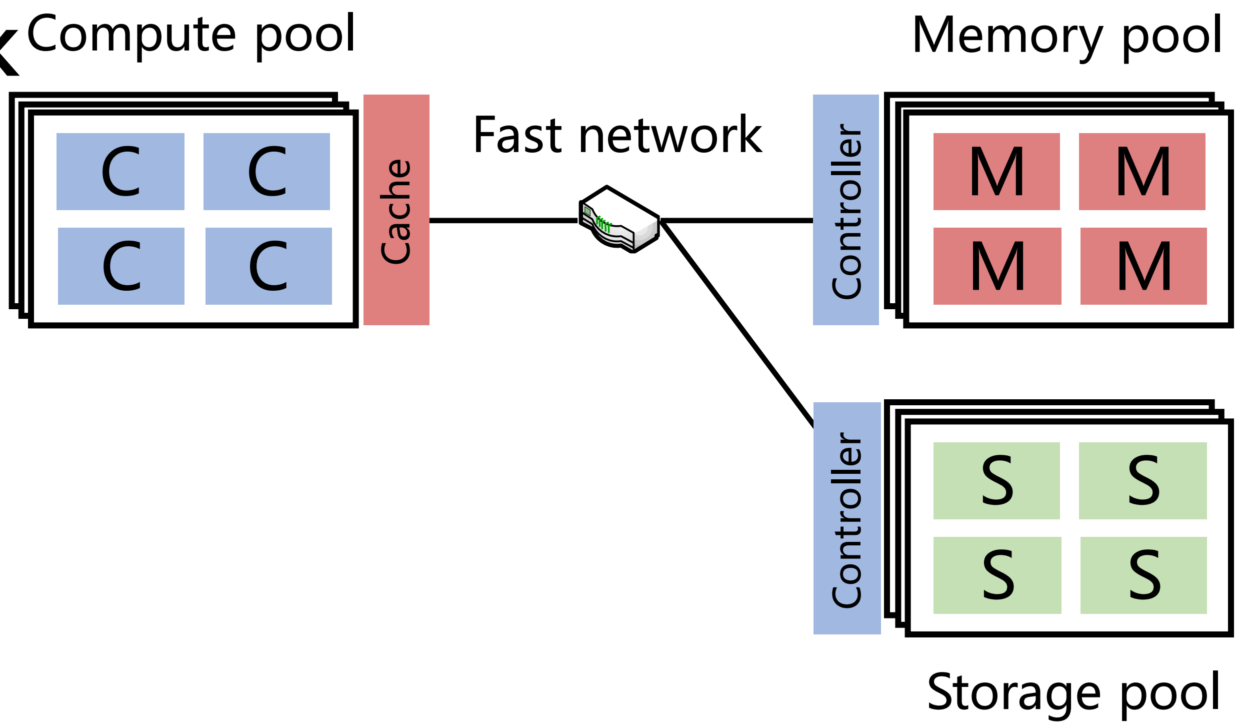


Unused memory in Azure:

Translated to hardware cost

Memory Disaggregation

- Separate compute, memory, and storage into resource pools that are connected by a fast network



Memory Disaggregation

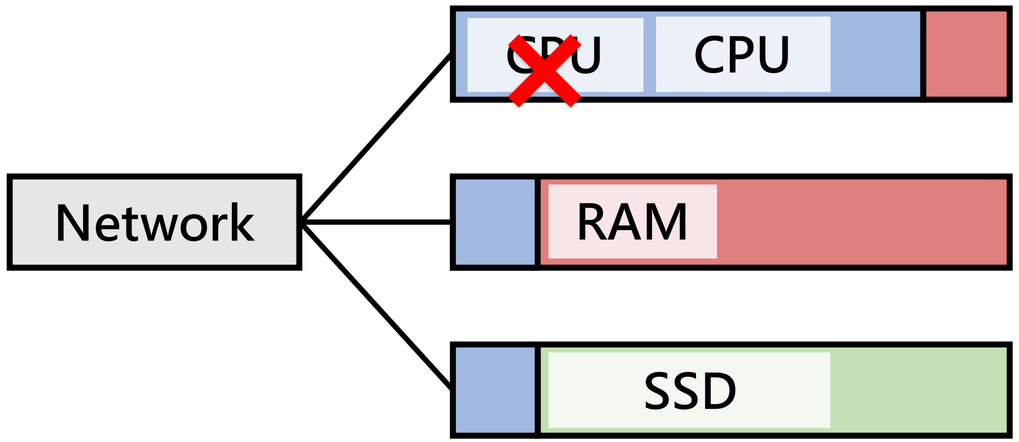
- Separate compute, memory, and storage into resource pools that are connected by a fast network

- Complete compute and data decoupling

60

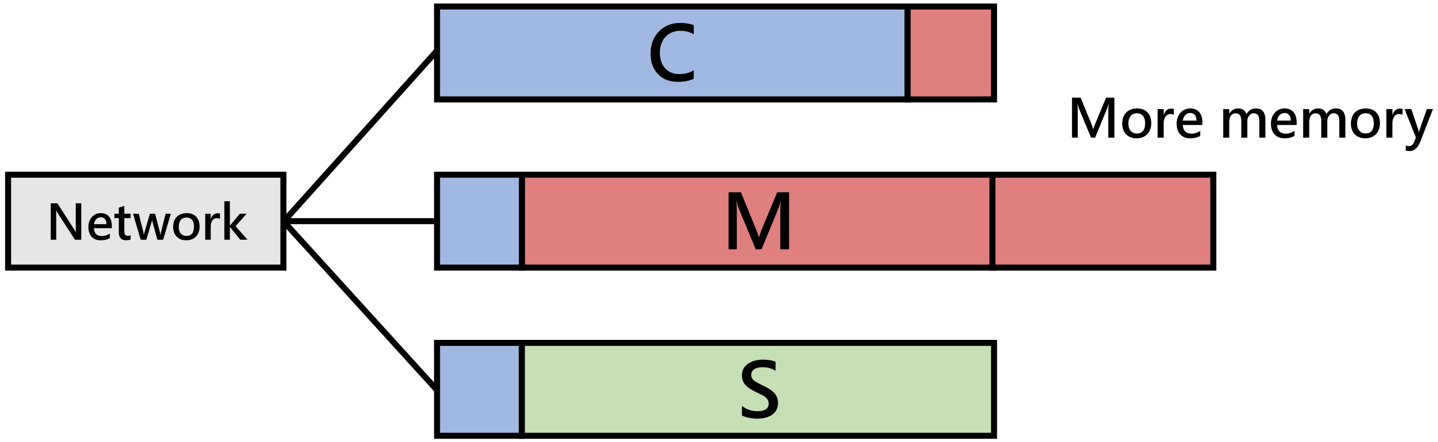
Operational Benefits

- Independent failures



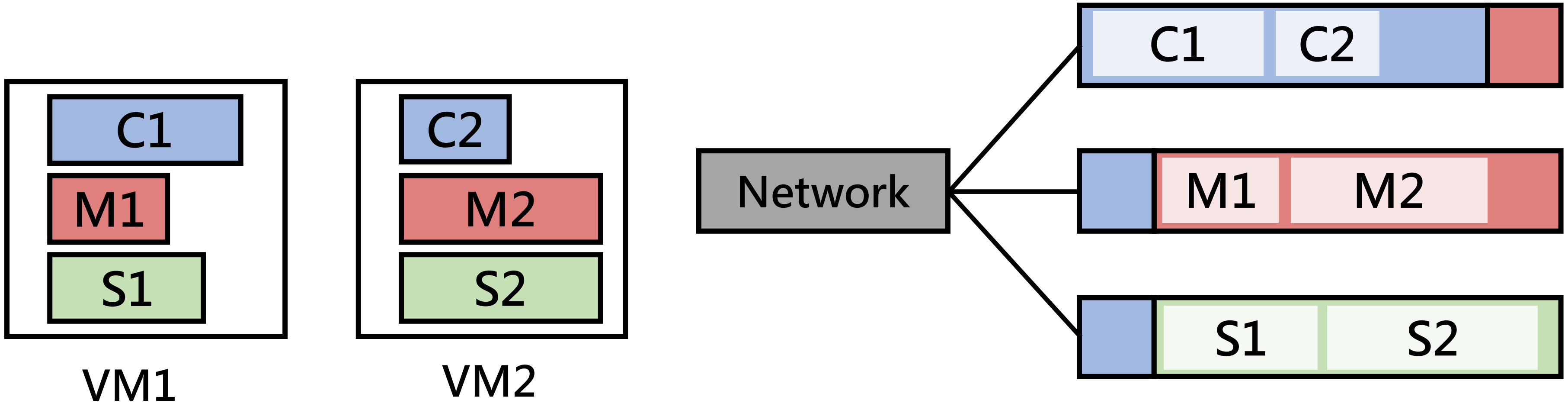
Operational Benefits

- Independent failures
- Independent expansion



Operational Benefits

- Independent failures
- Independent expansion
- Independent allocation

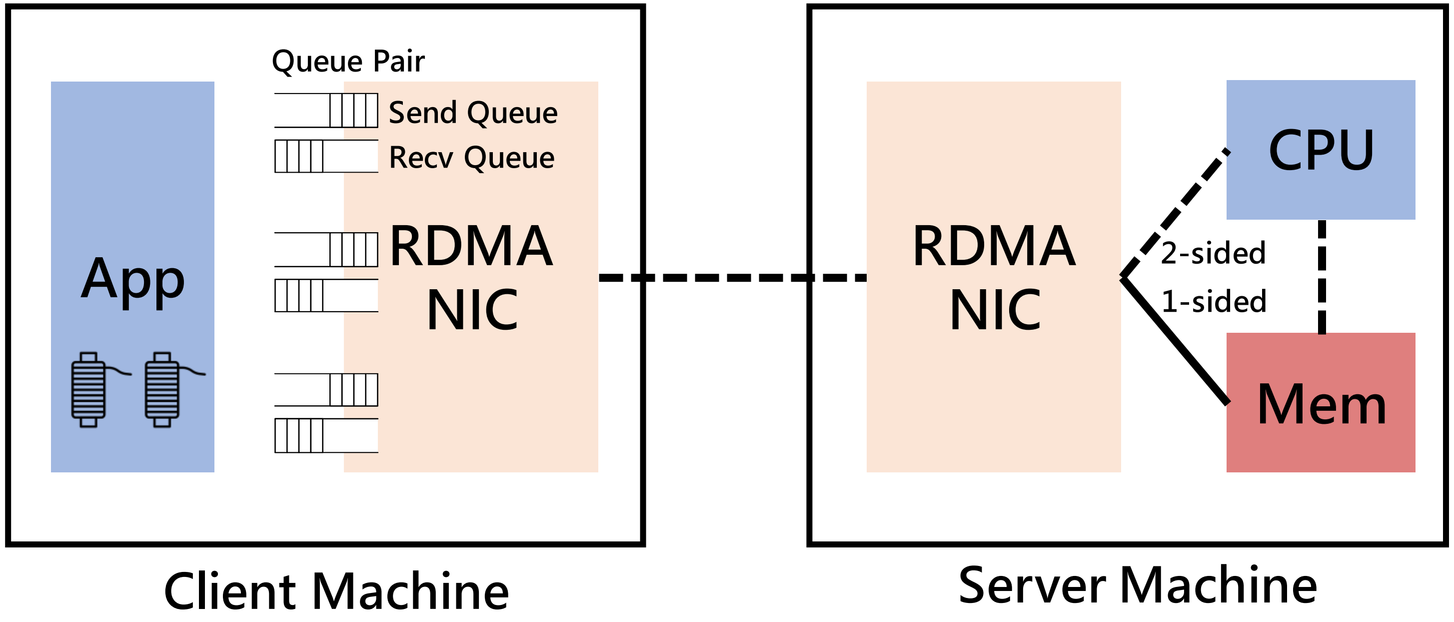


Physical resource pools

 63

Enabling Technique: RDMA

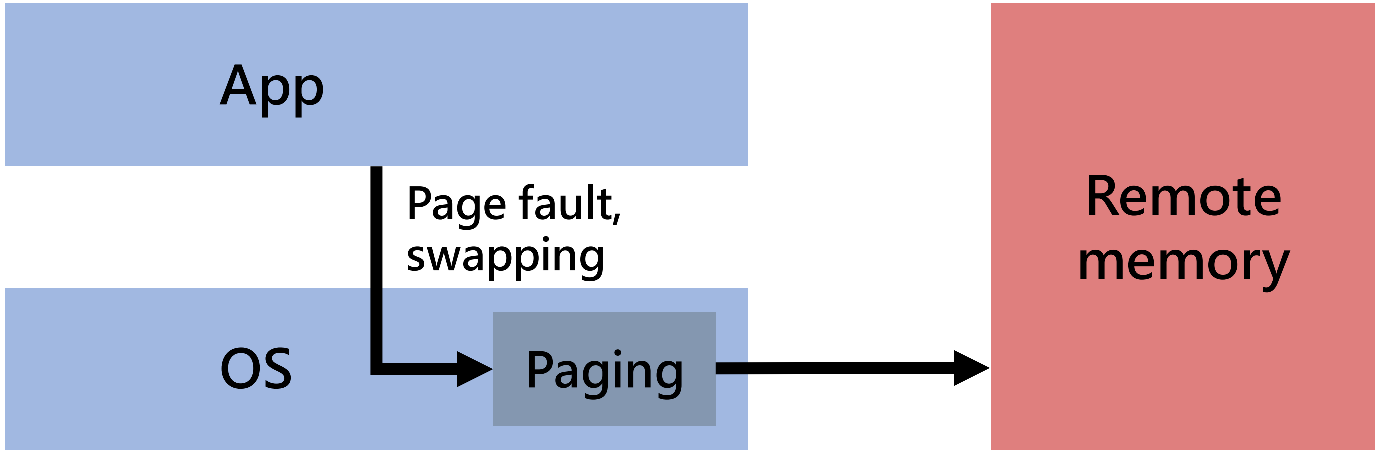
- Remote Direct Memory Access



Good fit
- Low CPU utilization
- High network speed

Types of Memory Disaggregation

- Kernel-space approaches



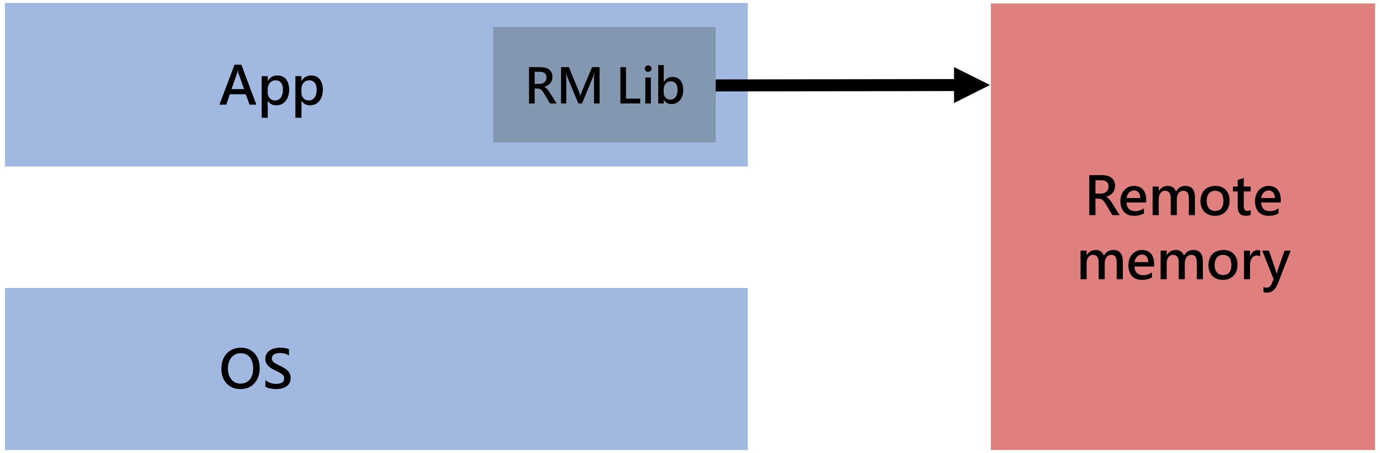
Pros
- Unmodified applications
- Transparent infra evolution

Cons
- High performance cost
- High development cost

65

Types of Memory Disaggregation

- User-space approaches



Pros
- No kernel overhead
- Fine-grained control
- Customized optimizations

Cons
- Application modifications

Implications for DBMSs

- Performance overhead
  - Memory access becoming network communication
- Data consistency
  - Consistent and concurrent remote memory access
- Remote memory abstraction
  - Offering remote memory with RDMA
- Reliability
  - Partial failures of compute and memory

Performance Implications for DBMSs

68

Covered Work

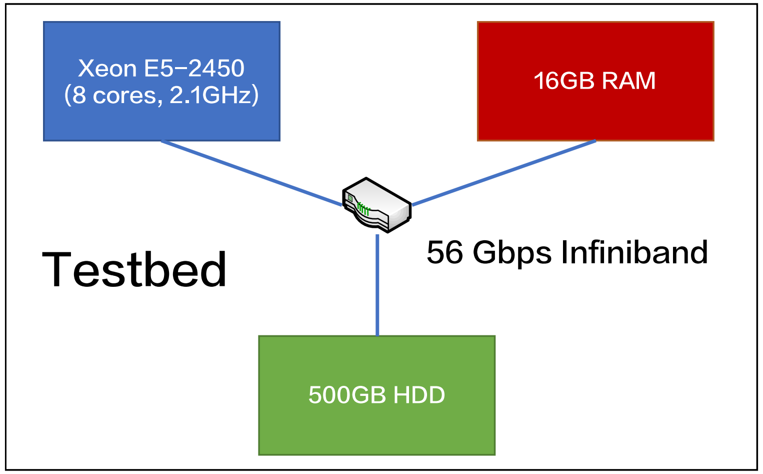
Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

Methodology of Study

- Evaluate production DBMSs
  - MonetDB
  - PostgreSQL

in a real cluster
  - Inifiniband network
  - LegoOS

with complex queries
  - All 22 TPC-H queries



LegoOS [OSDI 2018]

	MonetDB	PostgreSQL
Execution	In-memory	Out-of-core
Storage	Column-based	Row-based
Architecture	Client/Server	Client/Server
Buffer Pool Size	min(Capacity, Demand)	Customizable

Understanding the Effect of Data Center Resource Disaggregation on Production DBMSs
Q. Zhang et al., VLDB 2020

Disaggregation Cost

- What is the cost of memory disaggregation for complex queries?

- Evaluate DBMS performance slowdown in a disaggregated OS compared to Linux with the same hardware capacity
  - In-memory execution
  - Cold out-of-core execution (disk I/O involved)
  - Hot out-of-core execution (data cached)

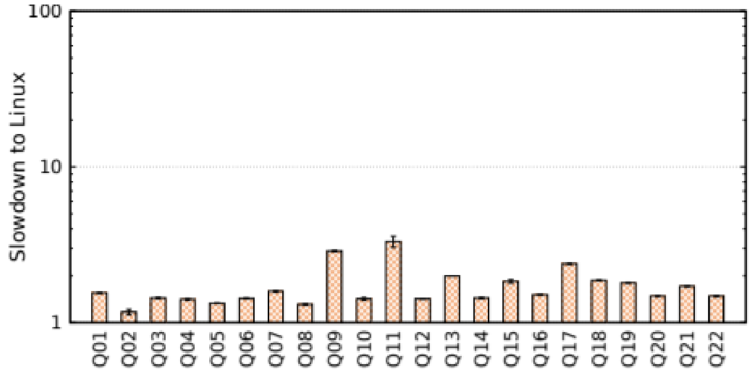
Understanding the Effect of Data Center Resource Disaggregation on Production DBMSs
Q. Zhang et al., VLDB 2020

71

Cost for In-memory Execution

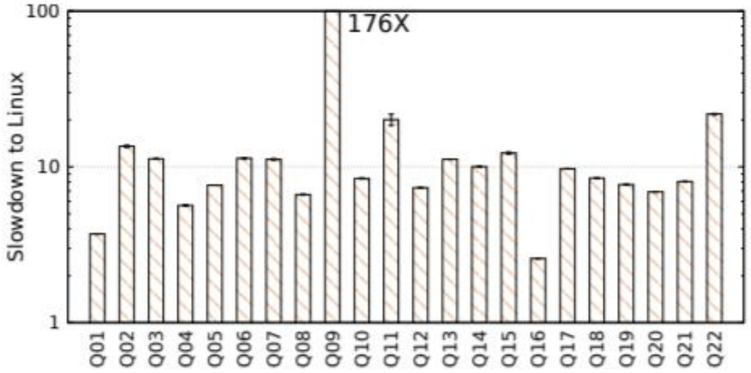
- MonetDB

1.7x slowdown



LegoOS (low degree of disaggregation)

18x slowdown



LegoOS (high degree of disaggregation*)

*low local memory size on compute node

Findings

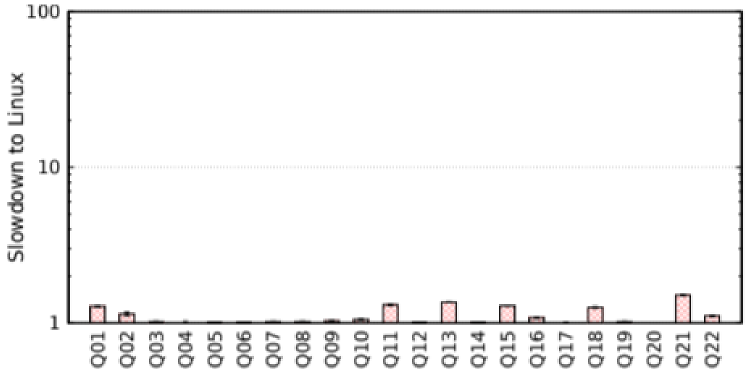
1. This confirms the cost of disaggregation for complex queries
2. The cost increases with the degree of disaggregation
3. The slowdown can be higher than 100x

Understanding the Effect of Data Center Resource Disaggregation on Production DBMSs
Q. Zhang et al., VLDB 2020

Cost for Out-of-core Execution

- PostgreSQL (cold, disk I/O is involved)

1.08x slowdown



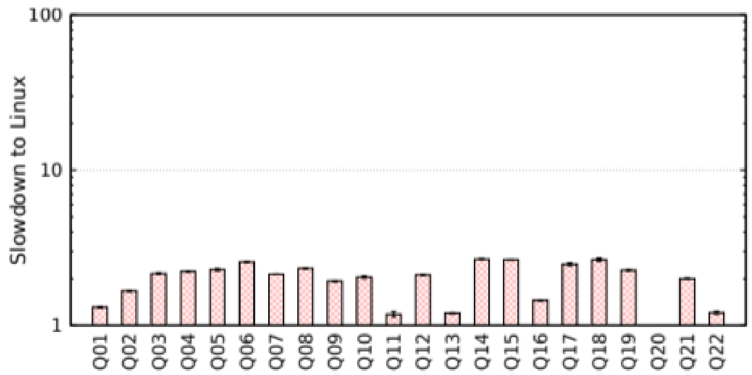
LegoOS (low degree of disaggregation)

Finding - most queries experience no cost from disaggregation

Cost for Out-of-core Execution

- PostgreSQL (hot, data is cached)

2x slowdown



LegoOS (low degree of disaggregation)

Findings
1. Hot execution has higher cost than cold execution
2. The slowdown is even higher than in-memory execution (1.7x)

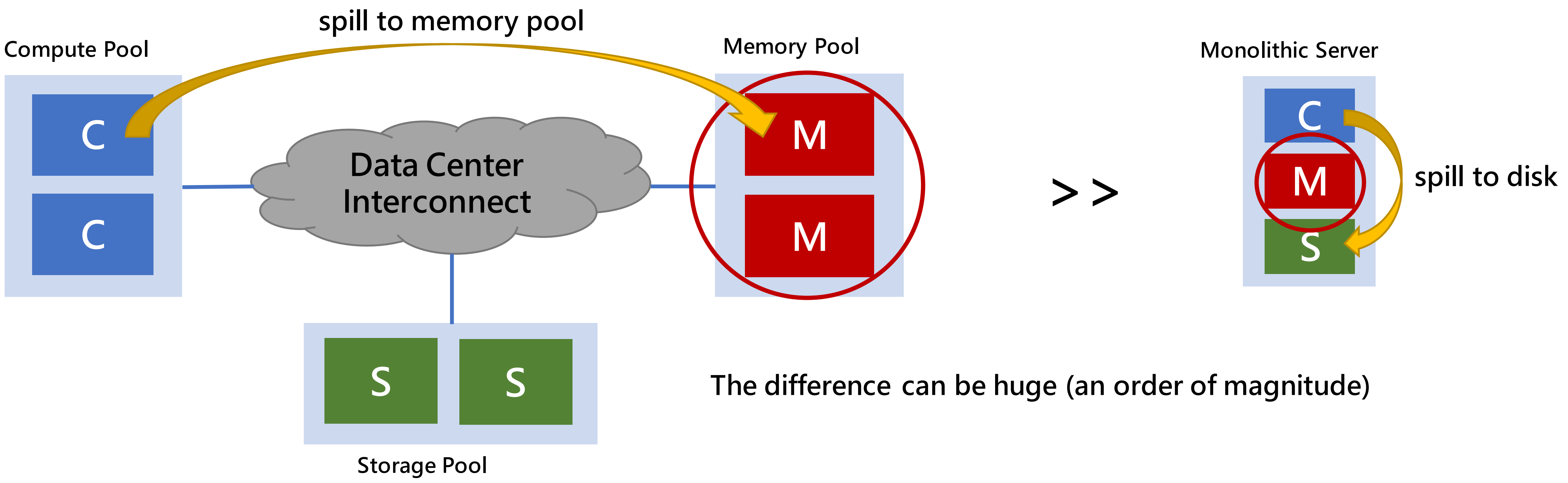
Summary of Disaggregation Cost

- In-memory execution
  - Moderate if working set fits into compute-local memory
  - Significant, otherwise

- Out-of-core execution
  - Dominated by other factors (disk I/O, cache design, etc.), and thus less sensitive to (the degree of) disaggregation

Another Perspective: Elasticity

- Consolidates the same type of resources
- Provides the opportunity of DBMSs using "infinite" resources without any application modifications



Understanding the Effect of Data Center Resource Disaggregation on Production DBMSs
Q. Zhang et al., VLDB 2020

Memory-disaggregated transactional systems

Covered Work

Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

LegoBase

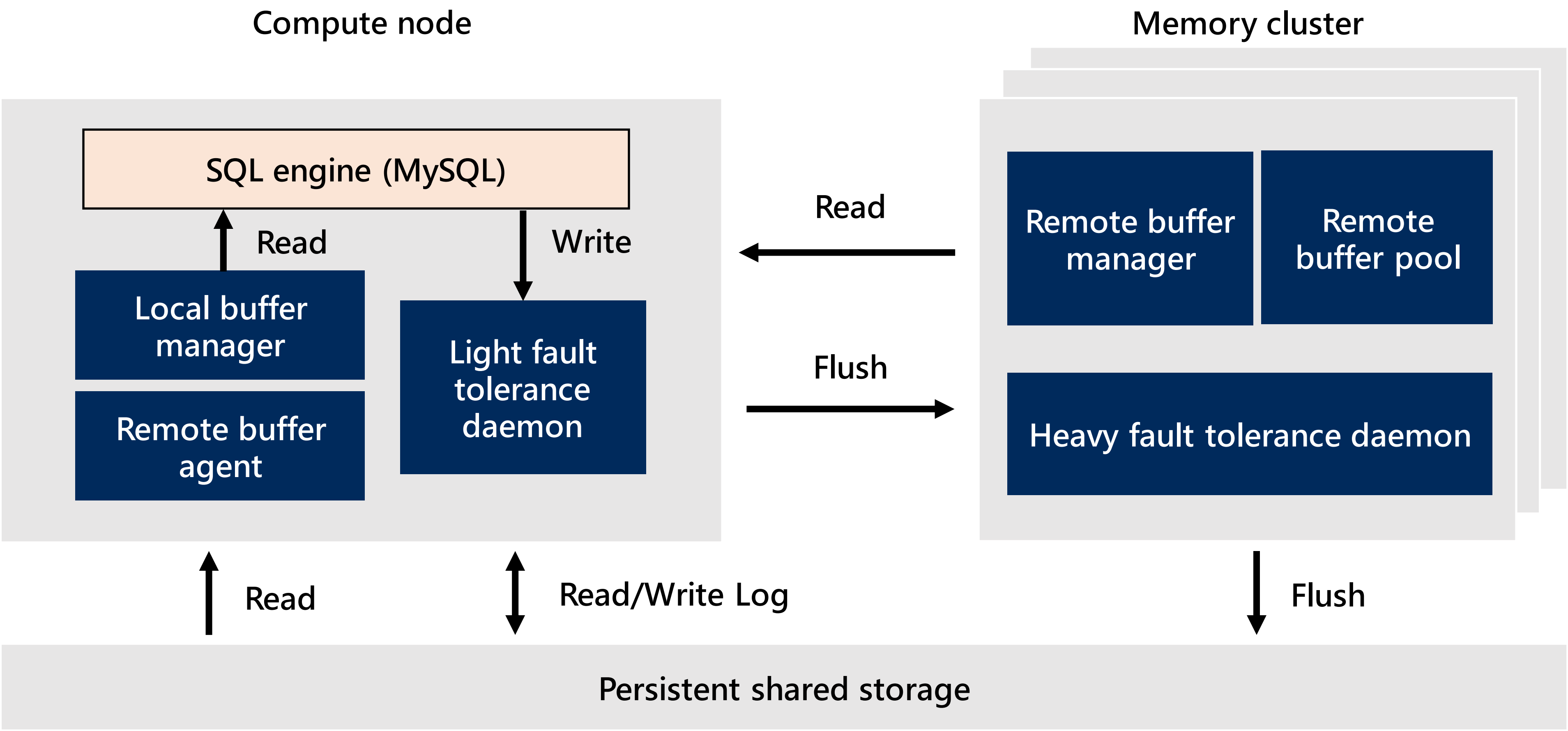
A transactional DB design for memory disaggregation with tiered memory management and recovery

Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

LegoBase

Primary contributions

- Moves memory management back to DBMS
- Provides a two-tier fault tolerance protocol



Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

Memory Management Motivation

- Existing memory disaggregation has been OS-based
  - Infiniswap [NSDI 17], LegoOS [OSDI 18]
- Issue #1: OS overhead on remote memory access
  - 4KB page transfer: 4-6 μs RDMA vs. 40 μs Infiniswap

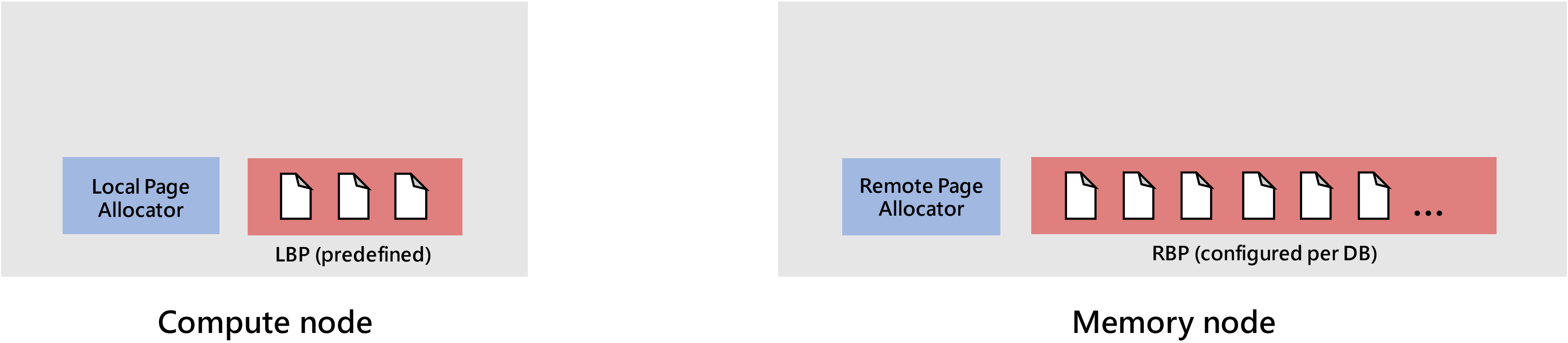
Memory Management Motivation

- Existing memory disaggregation has been OS-based
  - Infiniswap [NSDI 17], LegoOS [OSDI 18]
- Issue #2: low cache hit ratios with unified memory
  - Small but important data might be evicted, e.g., session info
  - OS LRU is less effective than DB-optimized LRU
  - Page size mismatch: 4KB in OS vs. 16KB in DBMS

Splitting Buffer Pool

Local Buffer Pool (LBP) vs. Remote Buffer Pool (RBP)

– LBP is a cache of RBP

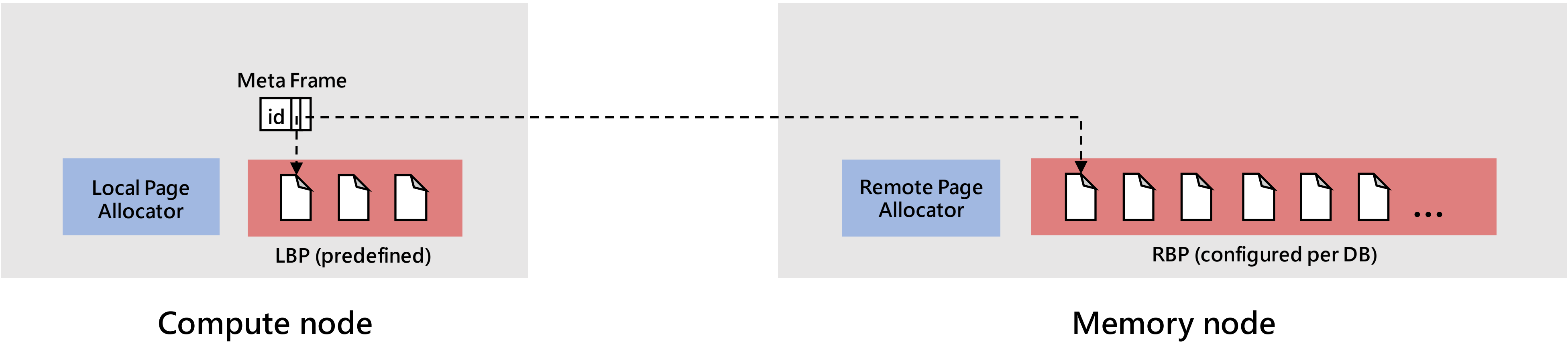


Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

Page Organization

Every page has a meta frame

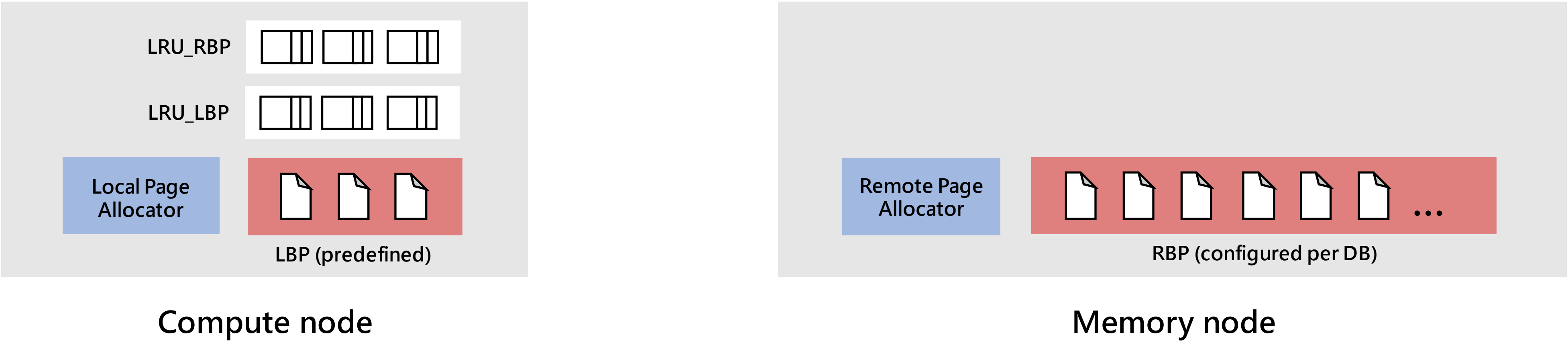
- Page id, local address, and remote address



Page Organization

Two LRU lists of meta frames on the compute node

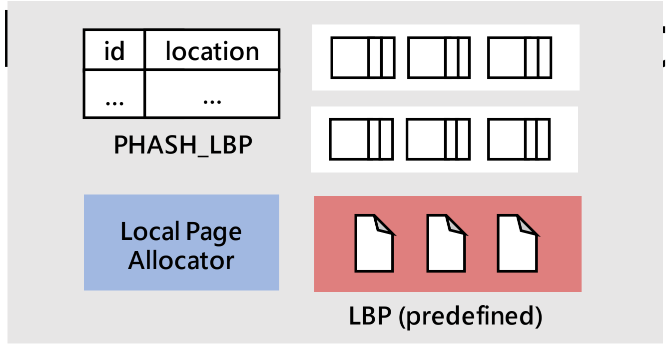
- LRU_LBP: MySQL-style LRU for local pages
- LRU_RBP: caching remote address for evicted pages



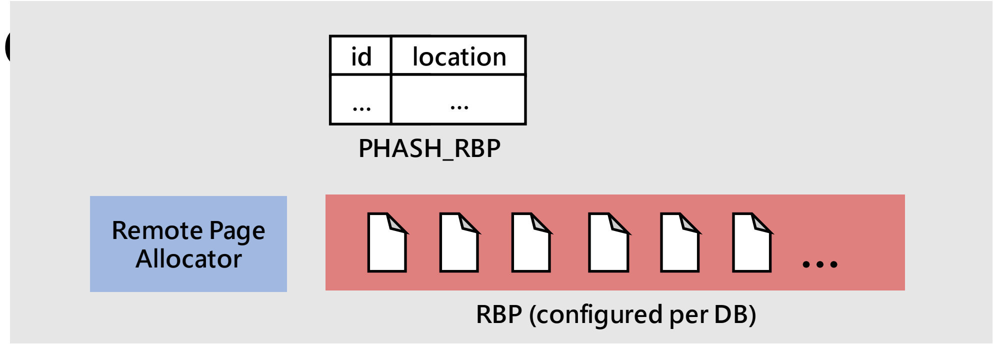
Page Lookup

Locating pages with hash lookups

- PHASH_LBP: pointing to the locations in the two LRU lists
- ing to lo



Compute node

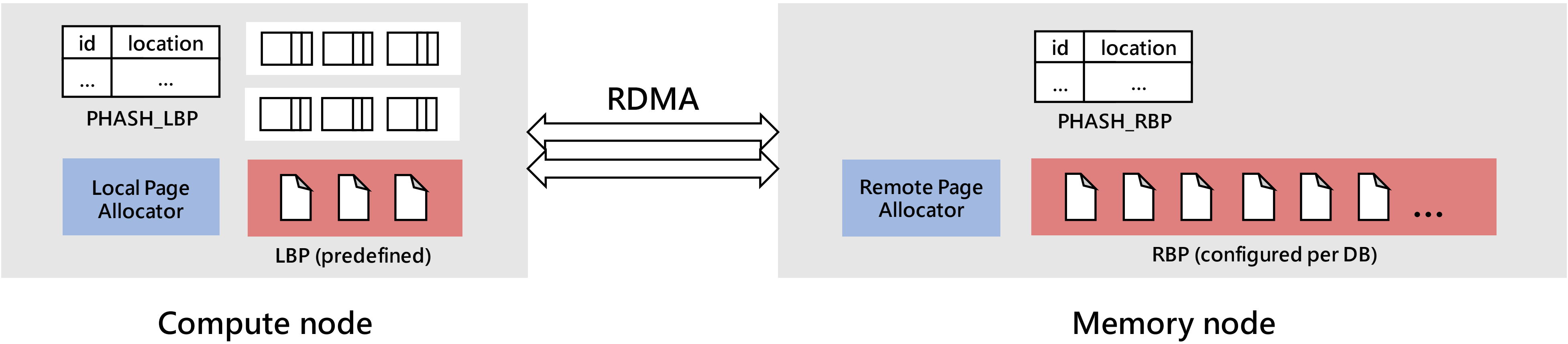


Memory node

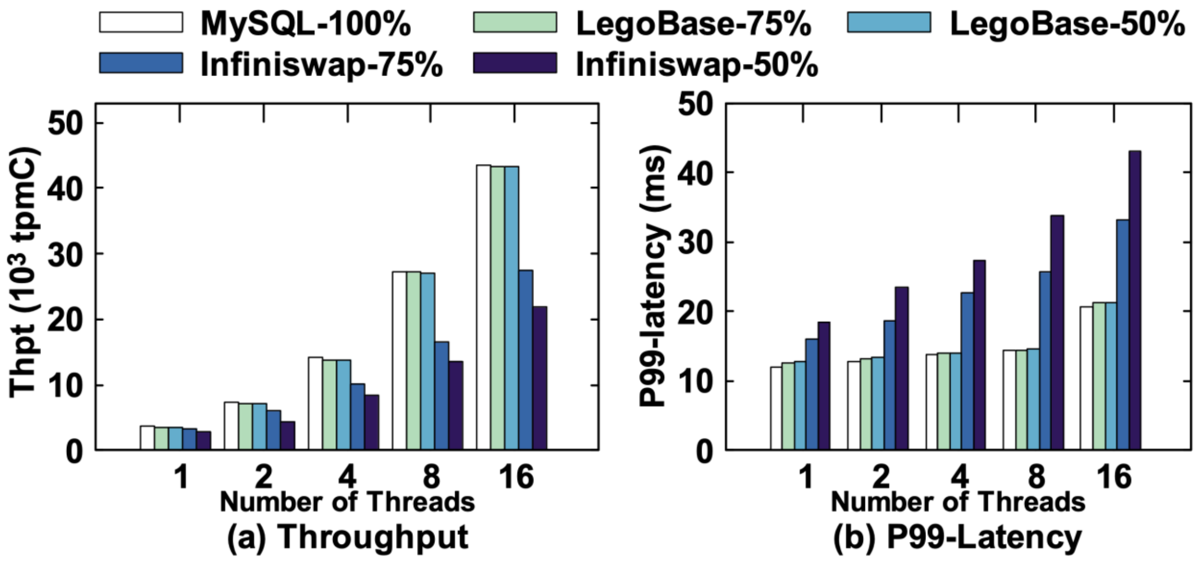
User-space Paging

Direct RDMA access from compute to memory

- Register and DeRegister: BP cache misses and evictions
- Read and Flush: compute cache misses and evictions



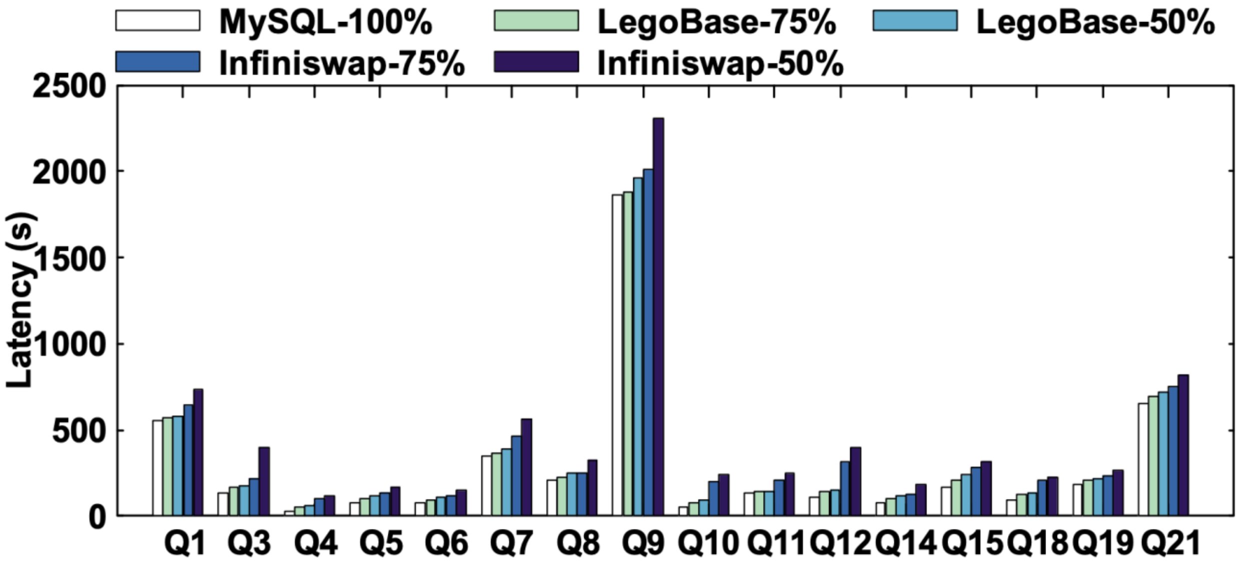
Result (TPC-C)



LegoBase outperforms Infiniswap

– Up to 2× on throughput and 2.3× on tail latency (p99)

Result (TPC-H)

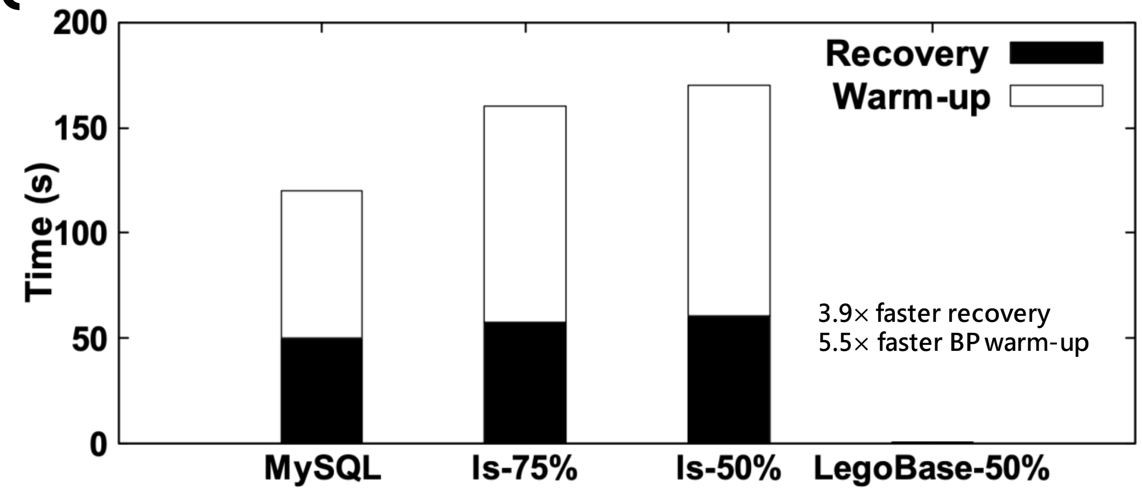


LegoBase query latency is close to monolithic MySQL

– But can be 2× higher for memory-intensive queries

Fault Tolerance Motivation

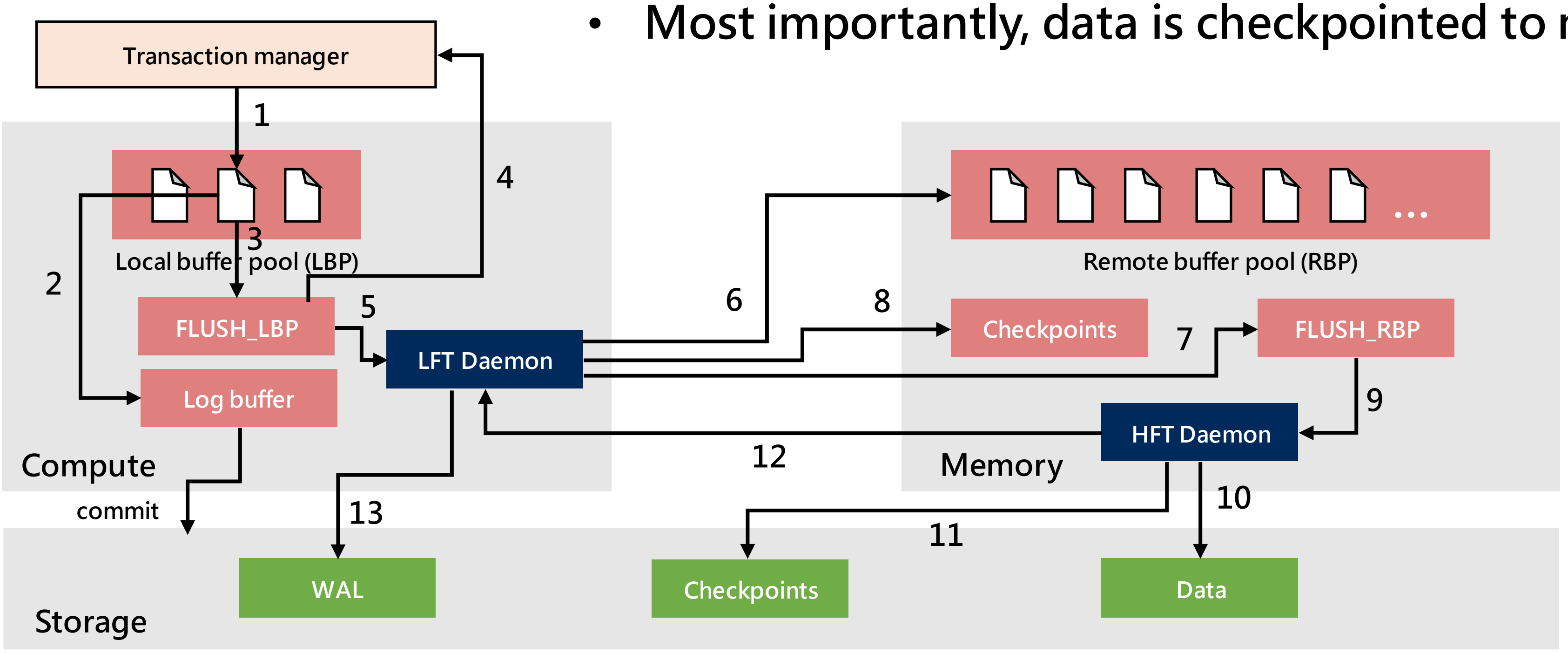
- Independent compute-memory failures introduce recovery opportunities
  - States saved in memory can speed up compute recovery



Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

Two-tier ARIES

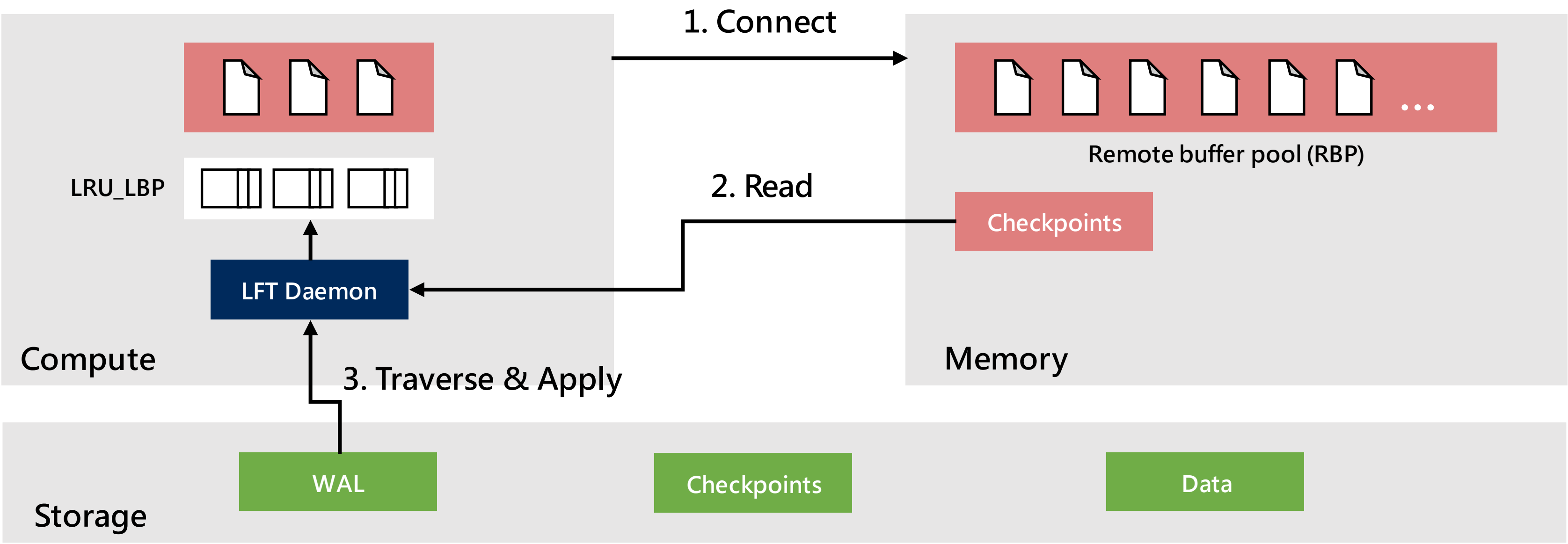
- Read the paper to figure this out
- Most importantly, data is checkpointed to memory



Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

If Compute Fails…

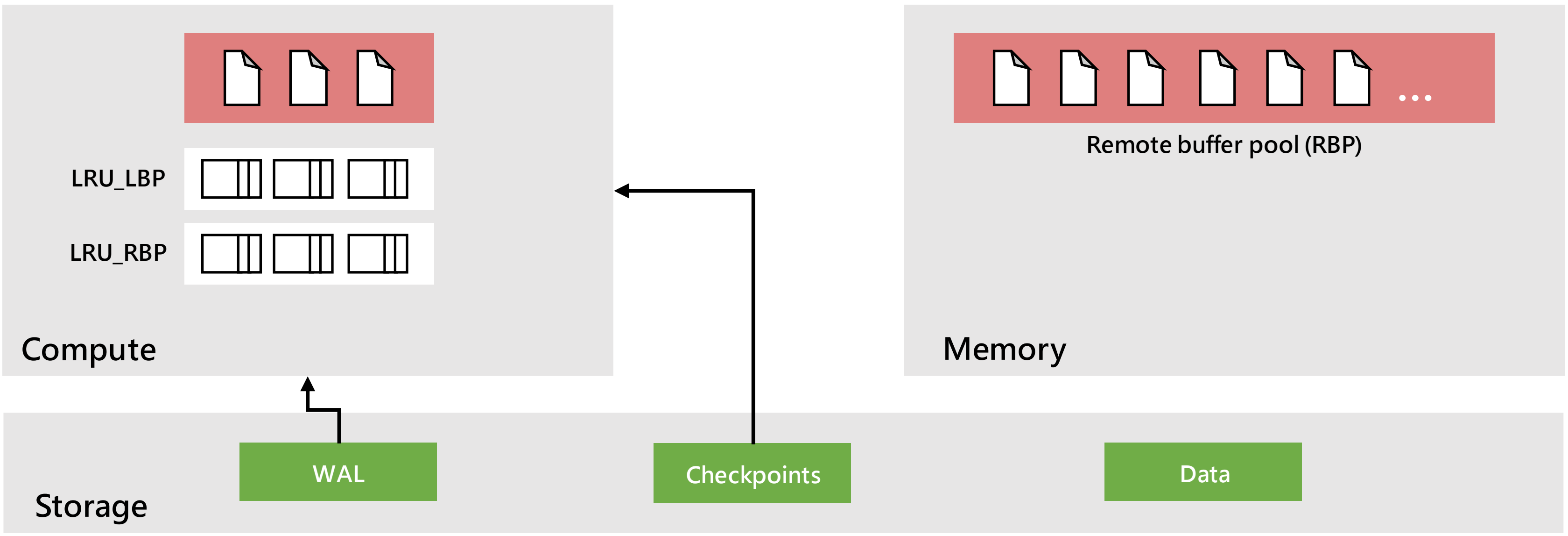
- Recover fast from tier-1 checkpoints



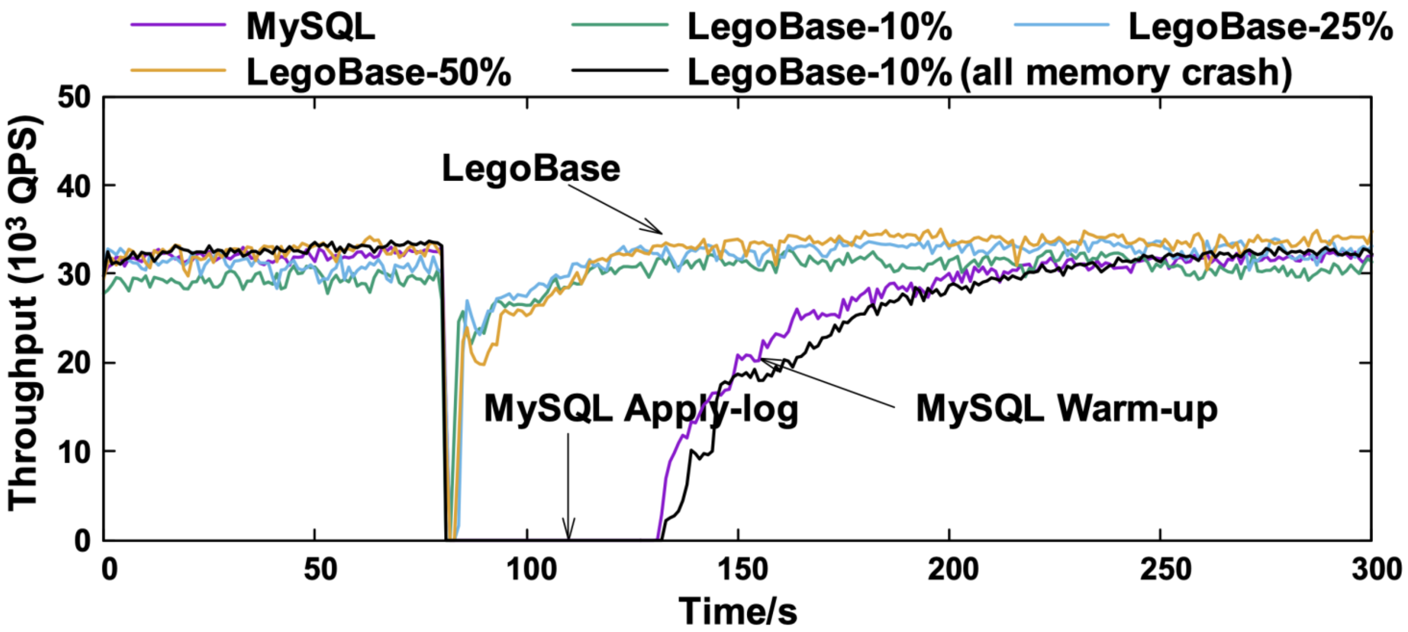
Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

If Both Fail…

- Recover slowly from tier-2 checkpoints



Result



Recovery time

– 50s for MySQL and LegoBase from tier-2
– 2s for LegoBase from tier-1

Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation
Y. Zhang et al., VLDB 2021

Summary

- MySQL customized for disaggregated memory
- DBMS-optimized memory management removes OS overhead and achieves more effective caching
- Two-tier fault tolerance leverages failure independence for fast recovery

Other Recent Work

- PolarDB Serverless [SIGMOD 21]: multi-compute
- Sherman [SIGMOD 22]: B+tree optimized for writes
- FlexChain [VLDB 23]: an XOV blockchain design
- dLSM [ICDE 23]: LSM indexing
- DSM-DB [VLDB 23]: distributed shared-memory DB

Memory-disaggregated analytical systems

Covered Work

Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

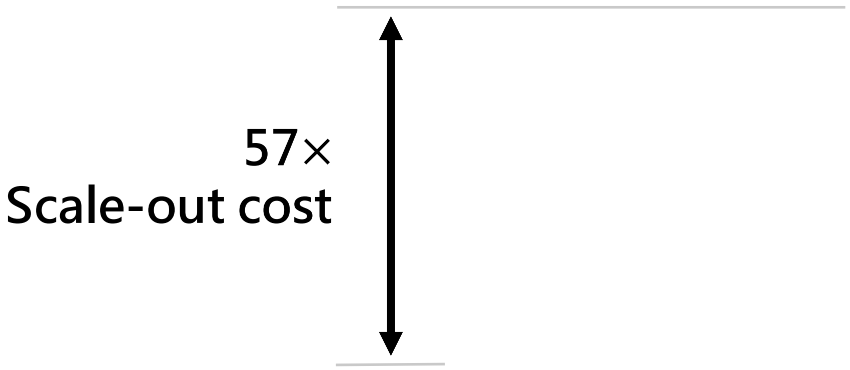
98

TELEPORT

A compute pushdown framework that moves operators from compute to memory

In-memory Query Performance

Monolithic vs. memory-disaggregated
MonetDB with TPC-H scale factor 50 (query 9)

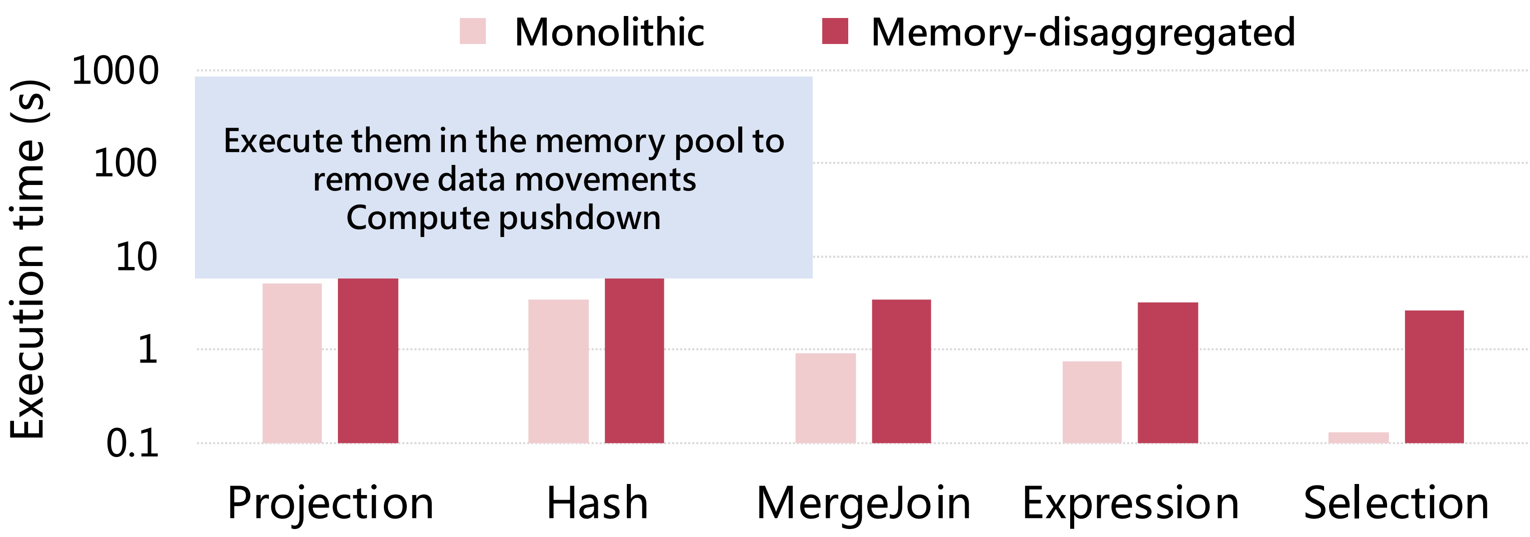


Can we remove most of this high "cost of disaggregation" to unlock all its benefits?

100

TELEPORT Motivation

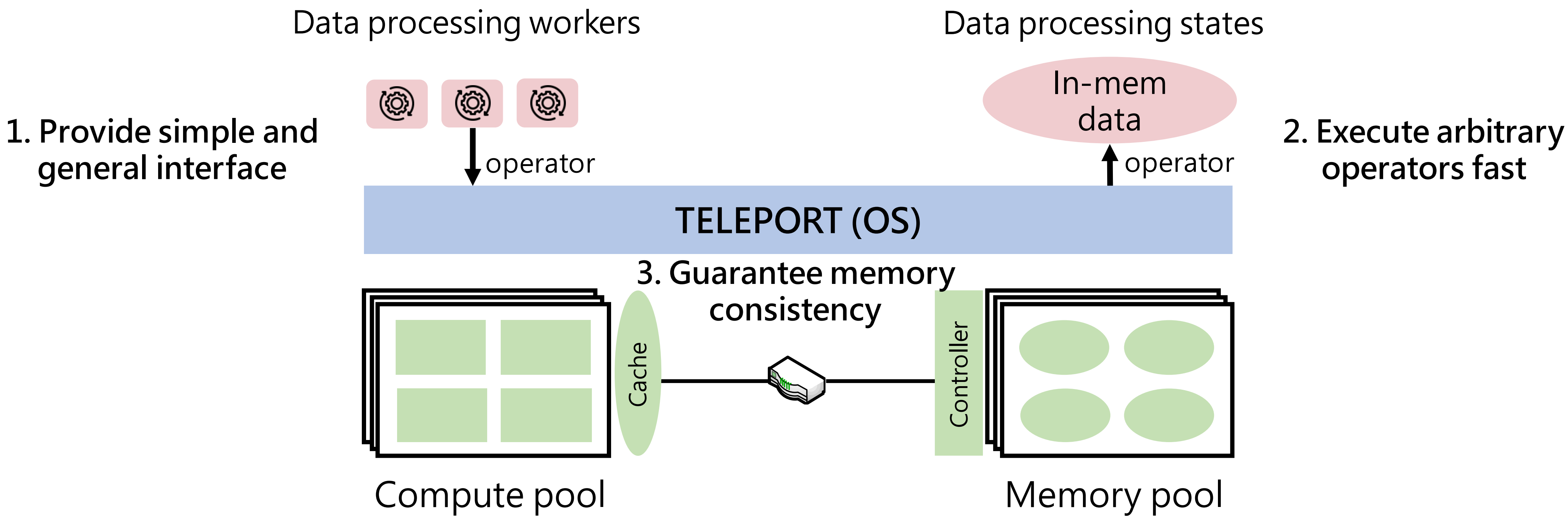
Monolithic vs. memory-disaggregated
MonetDB with TPC-H scale factor 50 (query 9)



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

TELEPORT Overview

- Compute pushdown framework for memory disaggregation



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

Compute Pushdown Interface

- **System call:** `pushdown(fn, arg, flags)`

  fn, arg: Function pointer / Argument pointer
  flags: Customization

```
void agg(table *input_table, double *result) {
   // implementation of aggregation
}

void main() {
   //...
   agg(t, r);
}
```

TELEPORT →

```
void agg(table *input_table, double *result) {
   // implementation of aggregation
}

void main() {
   //...
}
```

Compute Pushdown Interface

- **System call:** `pushdown(fn, arg, flags)`

  Function pointer / Argument pointer — Customization

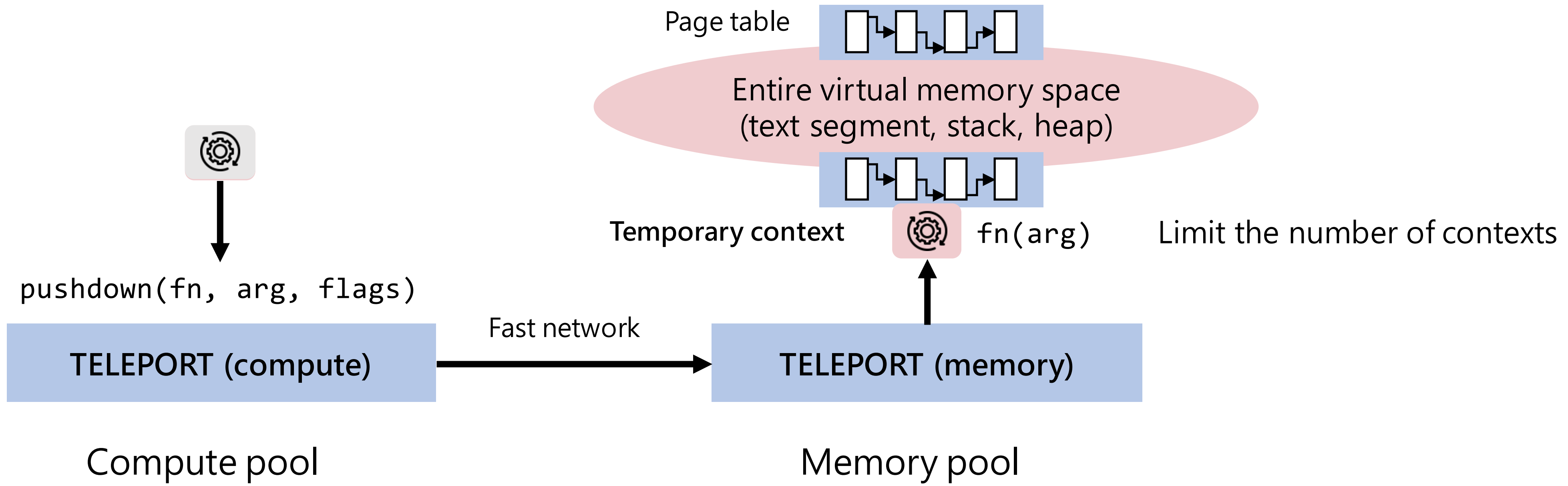
- **Ported MonetDB (in-memory DBMS, 400,000 lines in total)**
  - Projection, 117 lines
  - Aggregation, 214 lines
  - Selection, 302 lines
  - Hash, 75 lines

  To unlock all disaggregation benefits

  As well as PowerGraph (graph processing) and Phoenix (MapReduce)

Memory Pool Execution

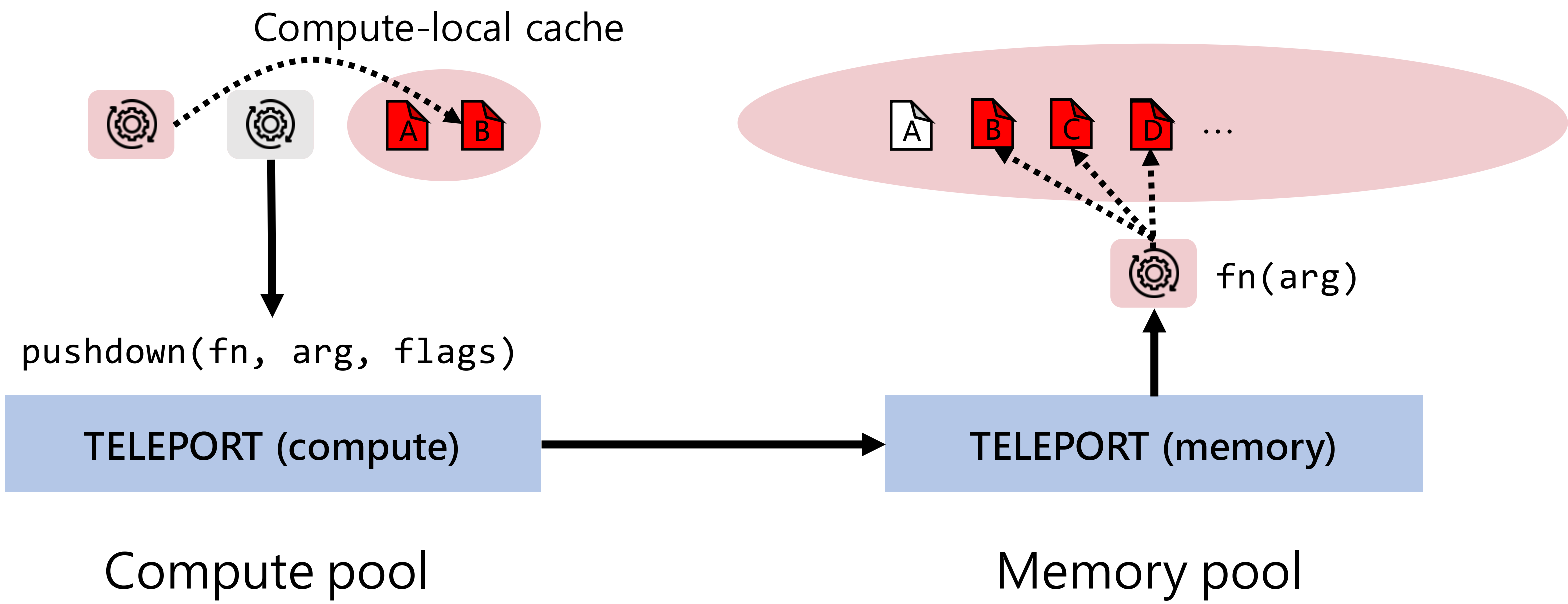
- Arbitrary and fast function execution
- Akin to POSIX vfork



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

Data Synchronization

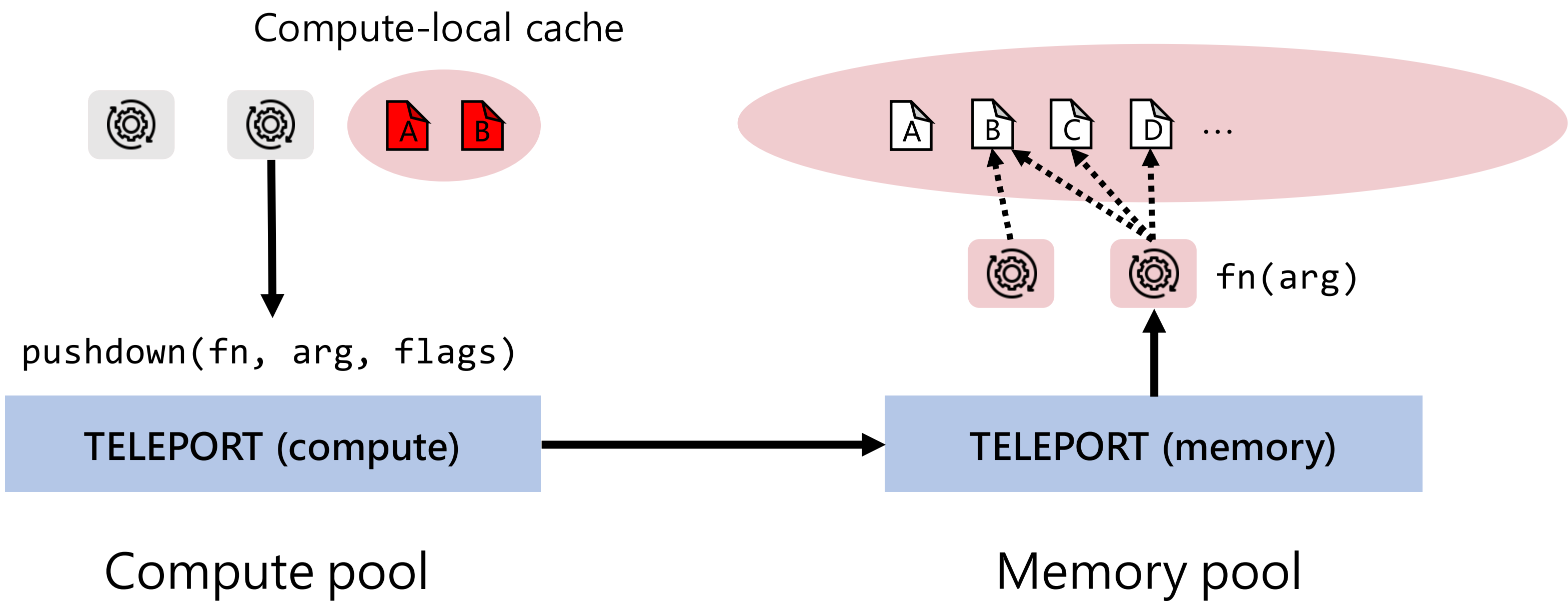
- Memory consistency between compute and memory
- Inconsistent time points: before pushdown after pushdown during pushdown
- Without proper synchronization, pushdown may be executed incorrectly



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

Baseline Approach

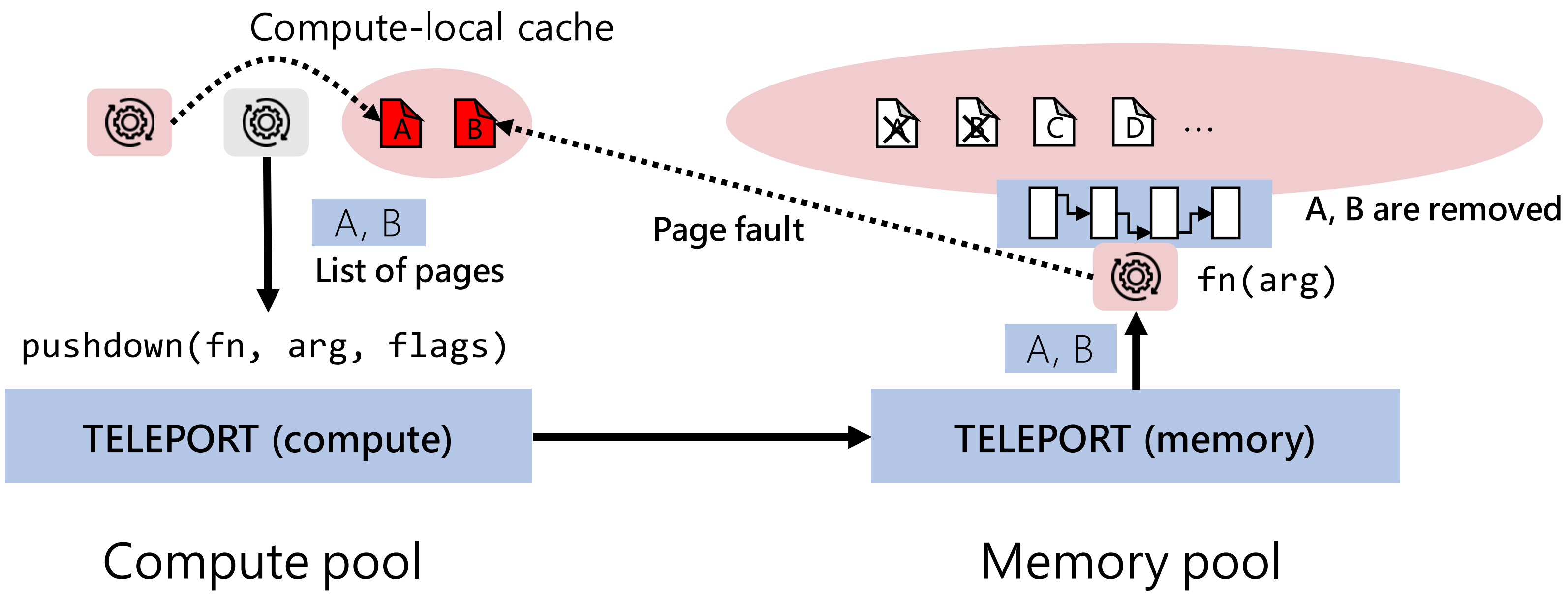
- Evict all local pages and push down all threads in the same process
- Performance issues
  - Not all compute-local pages are accessed in pushdown
  - Overwhelm memory pool' s limited compute resource



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

On-demand Coherence Protocol

- Synchronize pages only when they are needed
- Invariant: only one writable copy of a page between pools at any moment



Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

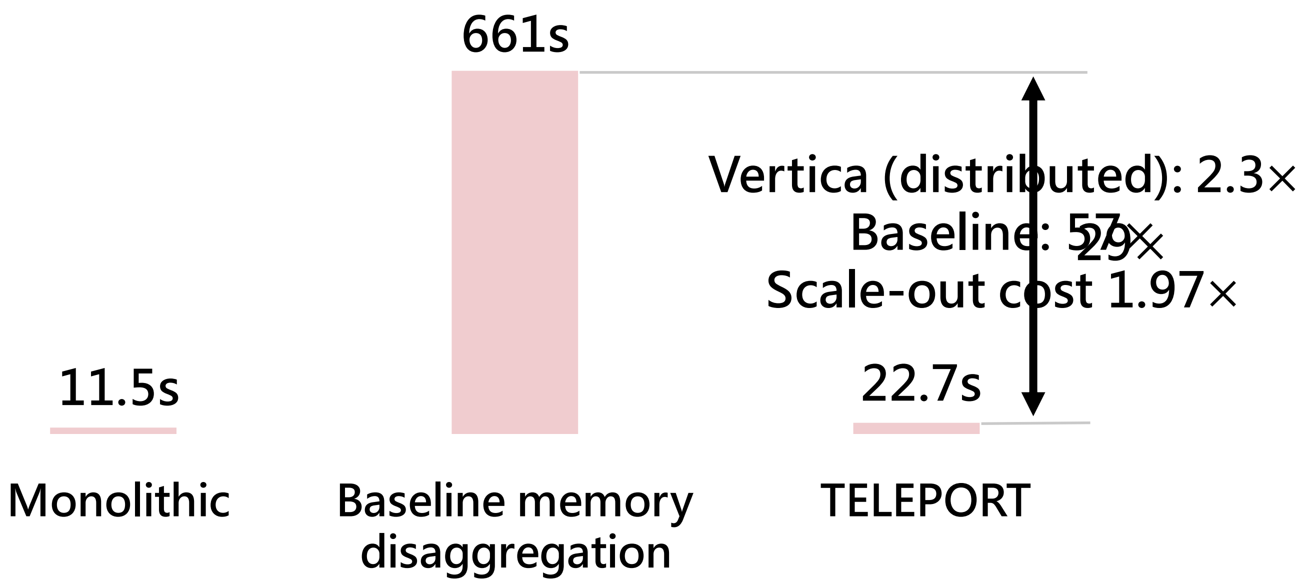
Evaluation Setup

- Compute: 8 CPU cores (16 threads) with 1 GB local cache
- Memory: 128 GB memory with 2 cores for pushdown
- Storage: 1 TB SSD

- Connected by an InfiniBand network: 56 Gbps bandwidth and 1.2 μs latency

TELEPORT Minimizes Overhead

MonetDB with TPC-H scale factor 50 (query 9)



TELEPORT removes most of the "cost of disaggregation"

Optimizing Data-intensive Systems in Disaggregated Data Centers with TELEPORT
Q. Zhang et al., SIGMOD 2022

Summary

- Memory disaggregation lacks good support for data-intensive applications, such as data analytics systems
- TELEPORT enables general and fast compute pushdown
- Distributing operators between compute and memory must take care of data consistency

Other Recent Work

- Google Big Query [VLDB 20]: large-scale shuffling through disaggregated memory
- Redy [VLDB 22]: utilizing stranded memory in cloud data centers as remote cache
- Farview [CIDR 22]: compute offloading with FPGAs for disaggregated memory

CXL-based memory disaggregation

113

Covered Work

Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirextCXL [ATC 22]	*CXL*

DirectCXL

An alternative approach to disaggregating memory using CXL

Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

Motivation: RDMA Cost

- Data is copied over the network
  - Network latency
  - DMA operations on both sides

- Data is copied between applications and NIC-registered memory regions

Compute eXpress Link (CXL)

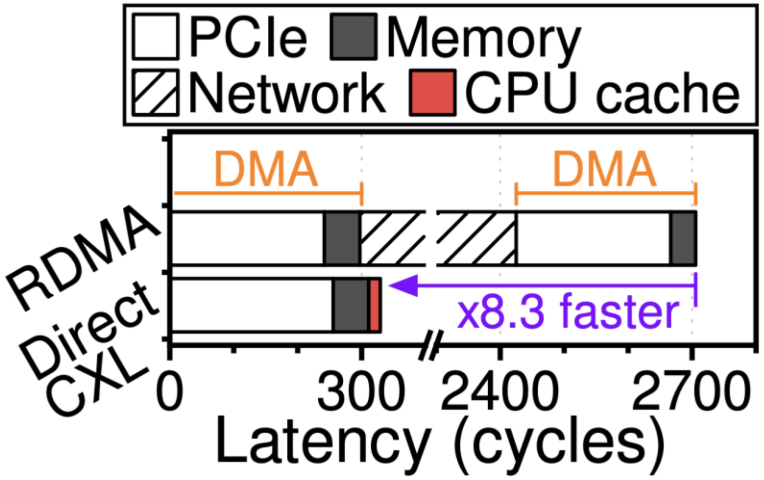
- Cache-coherent interconnects for connectivity between CPUs, accelerators, and I/O devices

- Supports all devices, from accelerators to memory
  - Type 1: device accessing host memory
  - Type 2: device and host accessing each other's memory
  - Type 3: host accessing device memory

Compared to RDMA

Direct PCIe access through load/store instructions

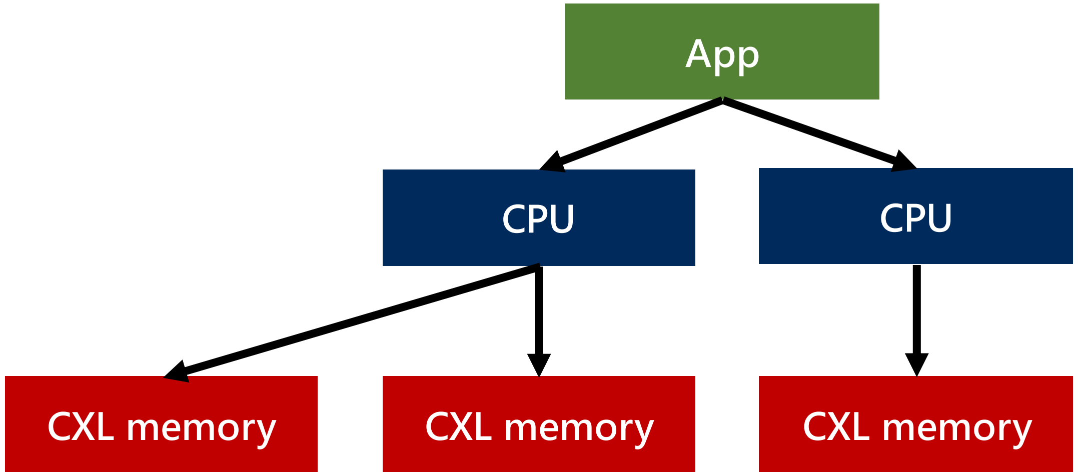
- No network latency
- No extra data copies



Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

Memory Disaggregation with CXL

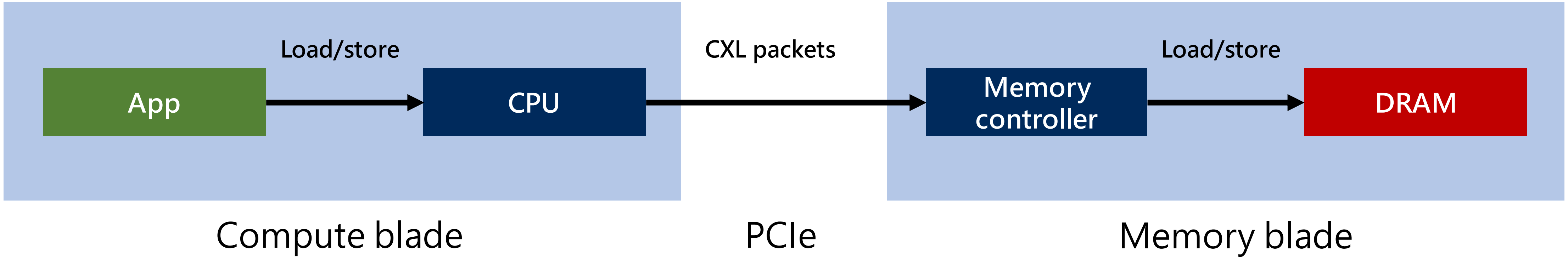
- How to enable direct access to CXL memory?
- How to enable flexible memory configuration?
- How to present CXL memory to applications?



Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

DirextCXL Design

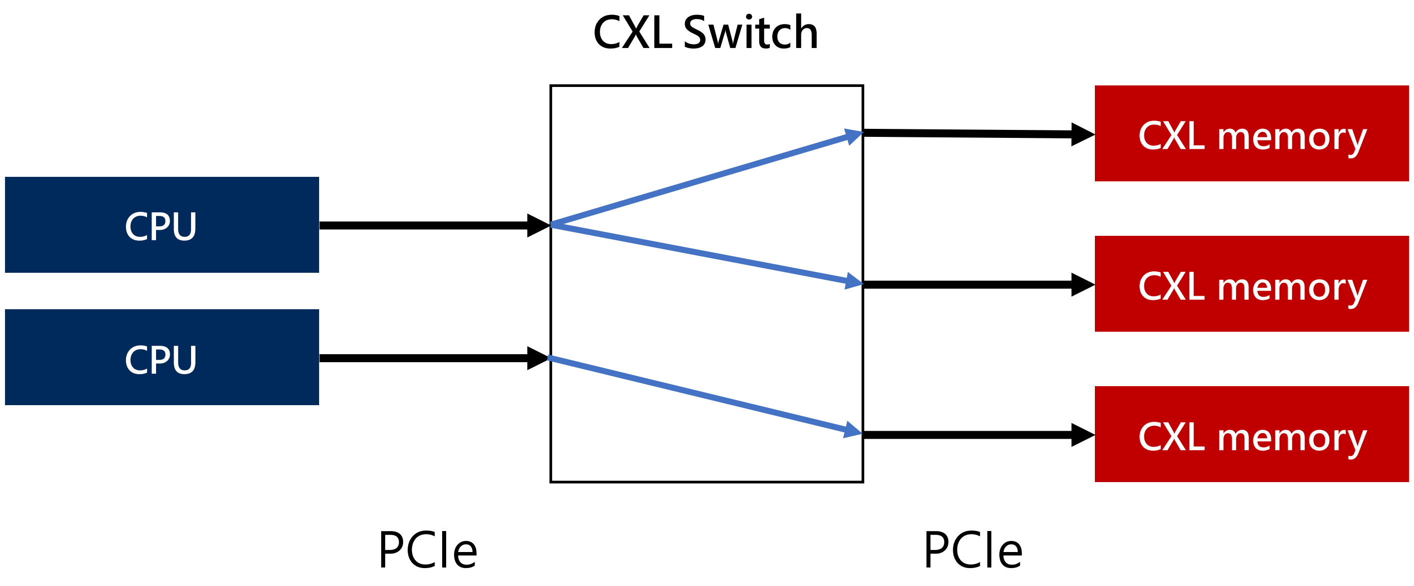
How to enable direct access to CXL memory?
- Convert load and store instructions to CXL packets
- An FPGA-based controller converts them back



Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

DirextCXL Design

How to enable flexible memory configuration?
- A CXL switch with a reconfigurable crossbar

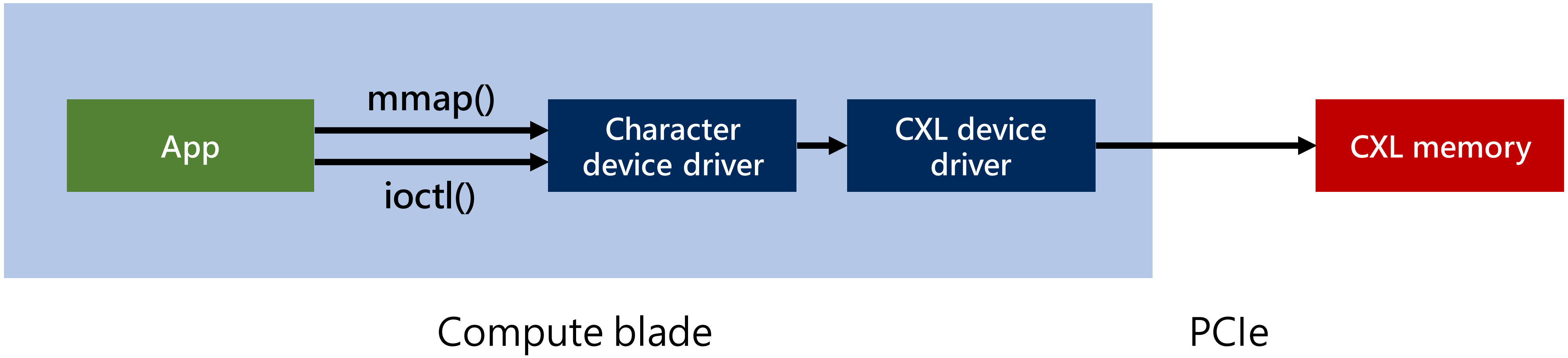


Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

DirextCXL Design

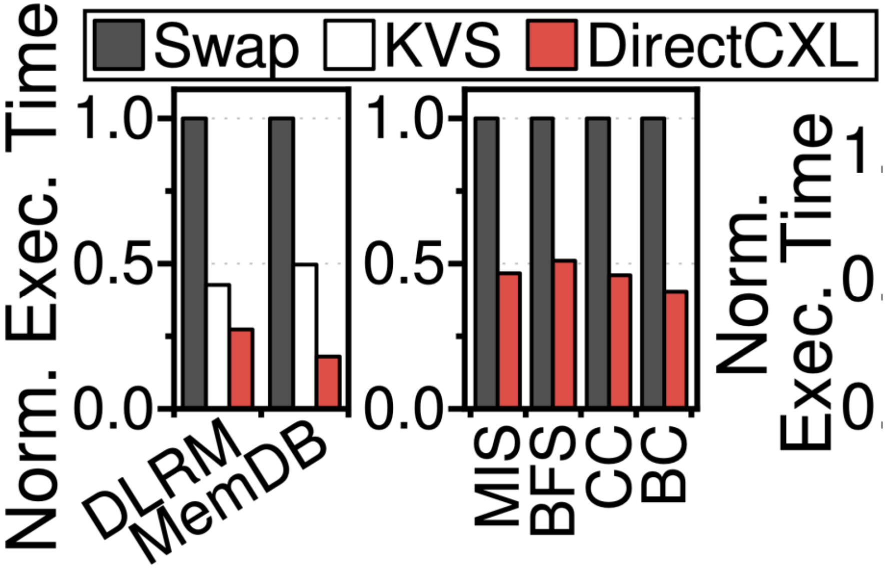
How to present CXL memory to applications?

– Leveraging Linux virtual memory system



Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

Result on Real Workloads



- DirectCXL outperforms RDMA
  - 3× faster than kernel-space RDMA (Swap)
  - 2.2× faster than user-space RDMA (KVS)

Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

Summary

- RDMA-based memory disaggregation incurs networking overhead and extra memory copies
- DirectCXL provides a CXL solution via direct PCIe access, a CXL switch, and a software runtime
- Application performance is significantly improved without modifications, showing CXL potentials

Other Recent Work

- SAP HANA on CXL-expanded memory [DaMon 22]: evaluating in-memory database system performance with CXL as the storage backend
- Active area in systems and architecture communities

Future directions of disaggregated DBMSs

126

Future Directions

- Comprehensive performance evaluation of disaggregated databases
- Scalable transactions in disaggregated databases
- Automatic resource provisioning
- CXL-optimized databases

127

Q & A

128