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

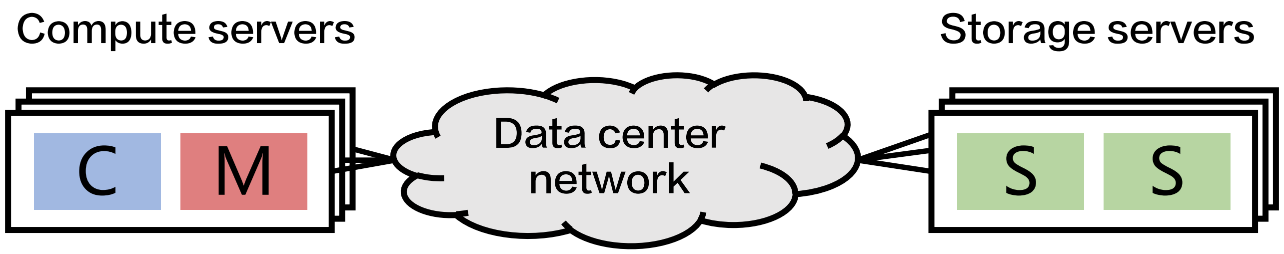
Covered Work

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

Introduction to memory disaggregation

Storage Disaggregation

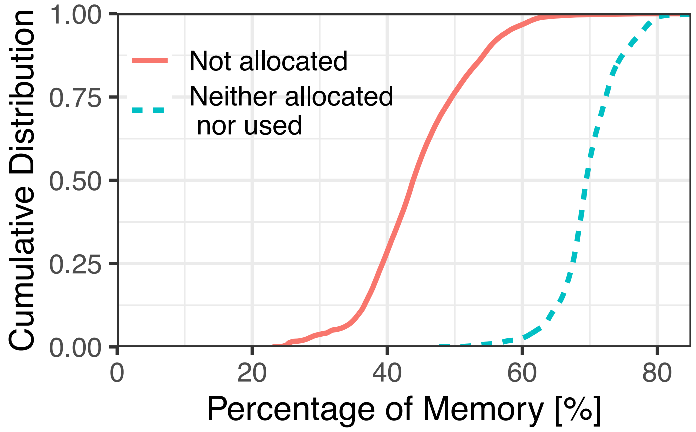
- Separating compute and storage



Storage Disaggregation

- Separating compute and storage

- Compute and memory are still coupled
  - Inflexible compute and memory allocation
  - Limited memory elasticity
  - S pute server failures

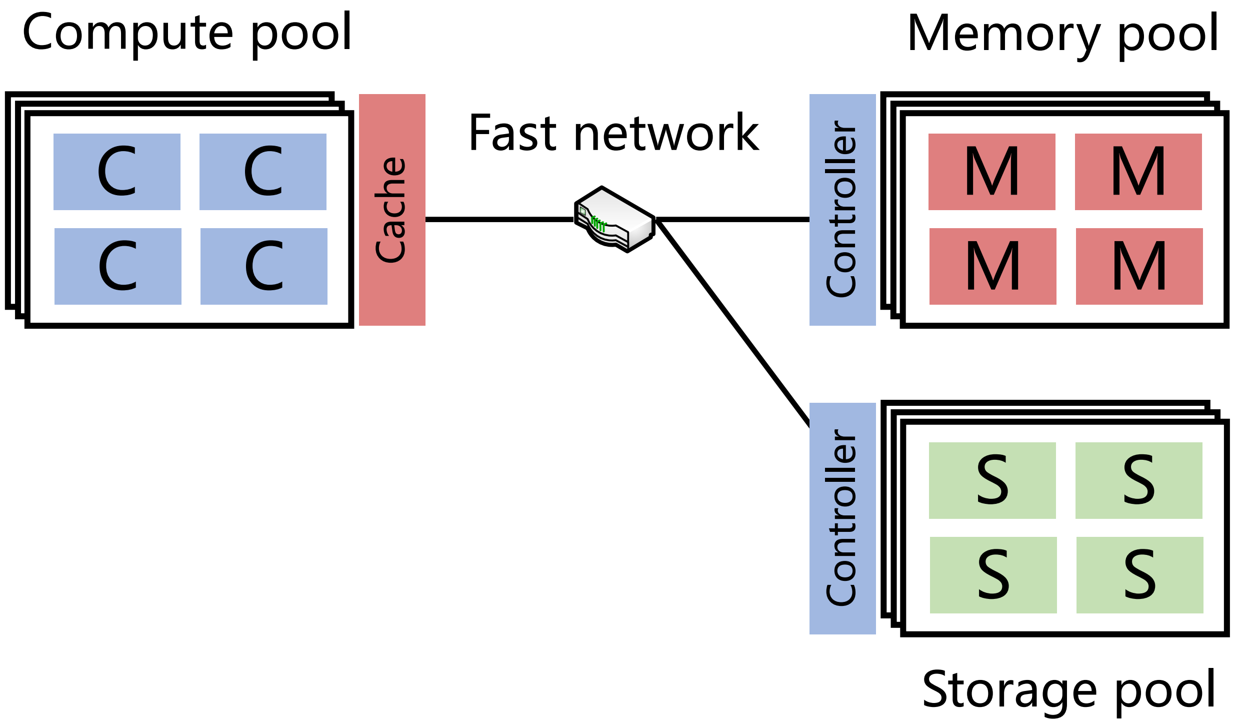


Unused memory in Azure

Translated to hardware cost

Memory Disaggregation

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



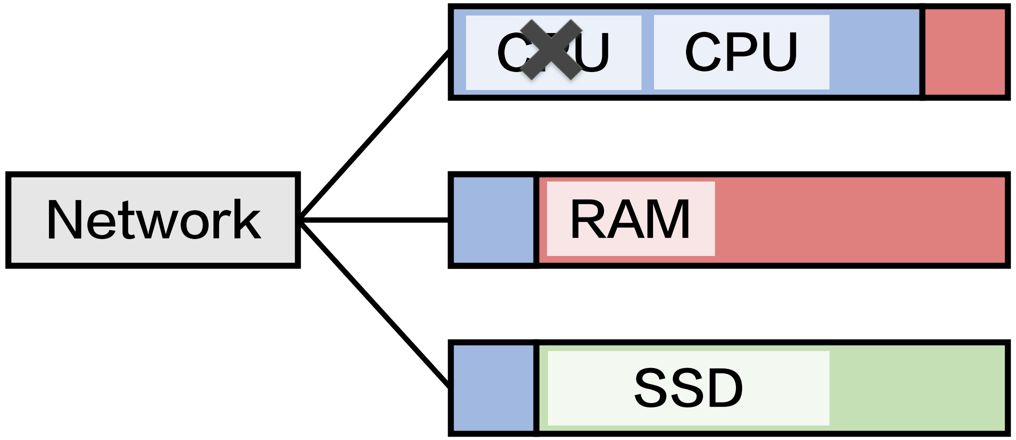
Memory Disaggregation

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

- Complete compute and data decoupling

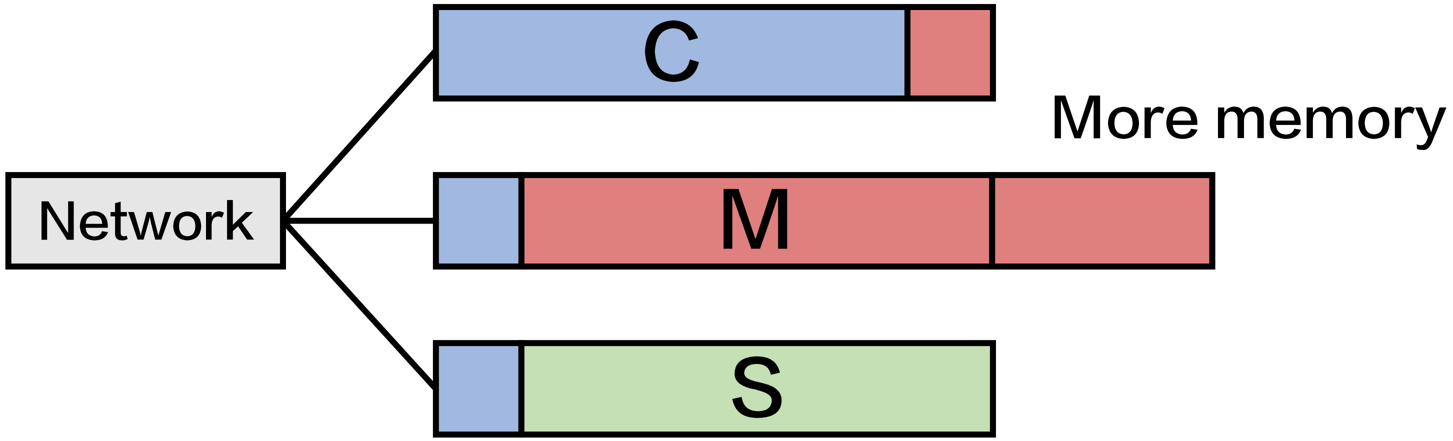
Operational Benefits

- Independent failures



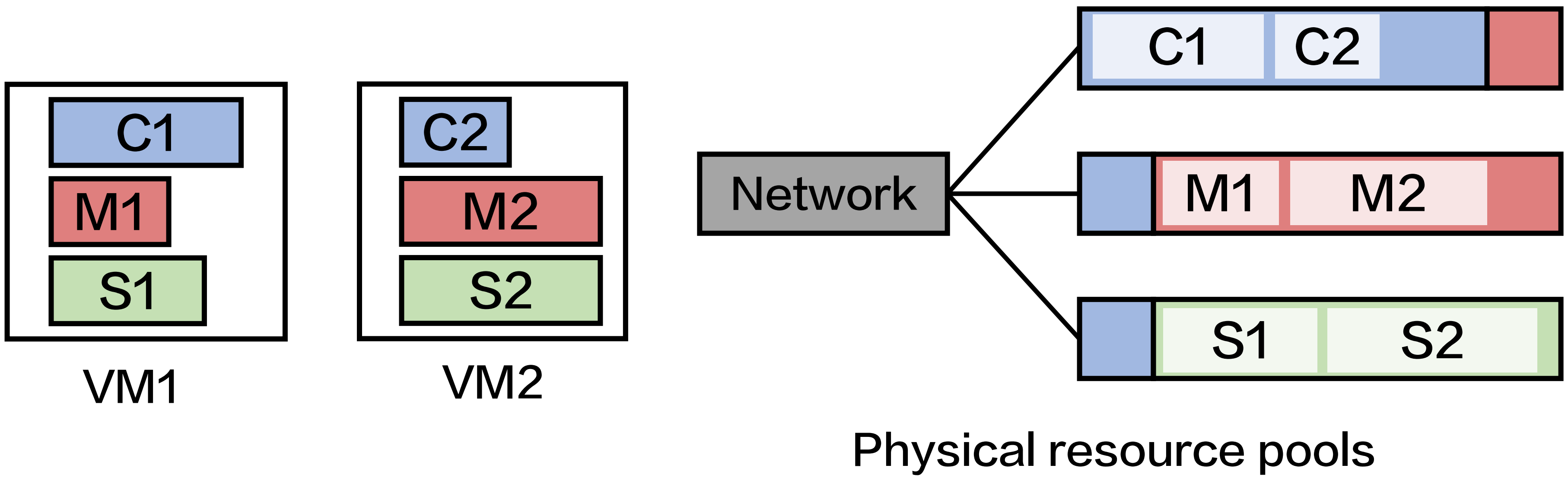
Operational Benefits

- Independent failures
- Independent expansion



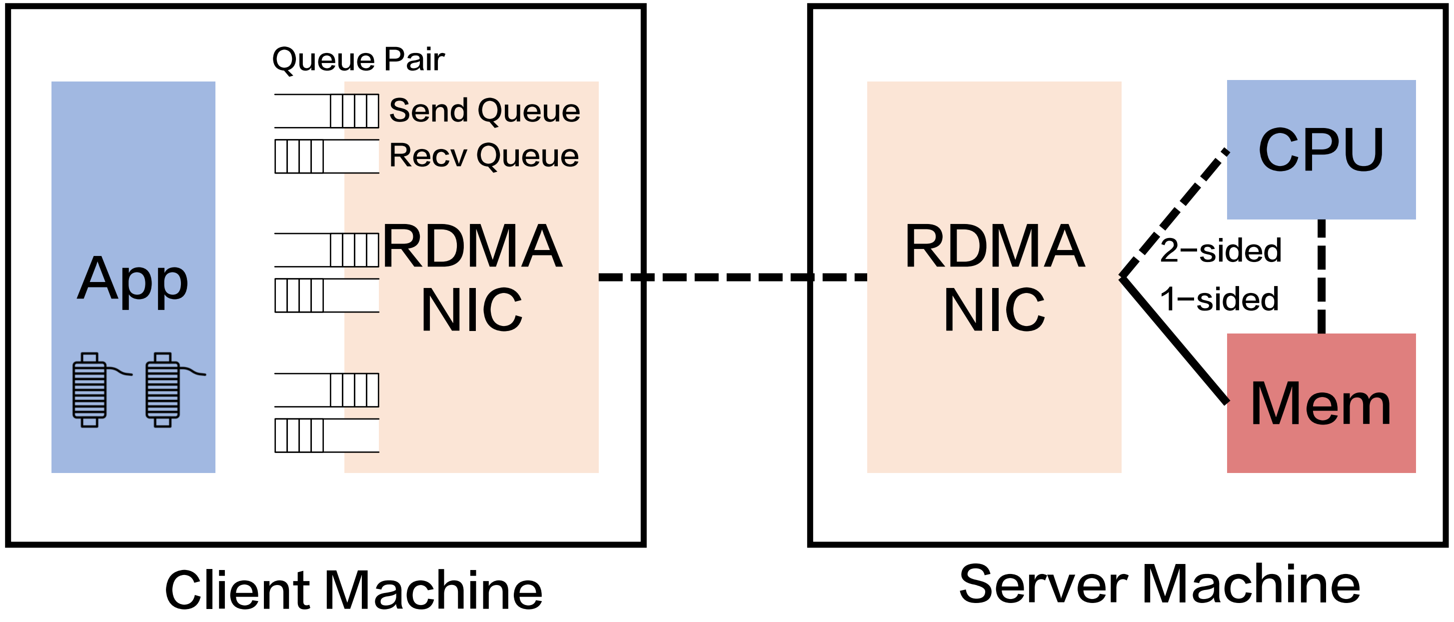
Operational Benefits

- Independent failures
- Independent expansion
- Independent allocation



Enabling Technique: RDMA

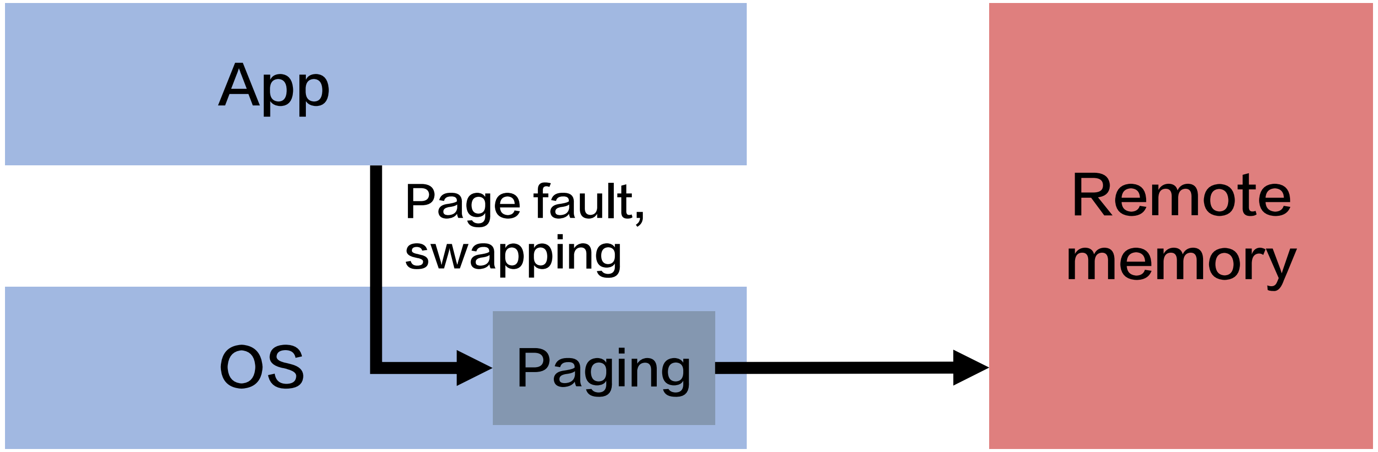
- Remote Direct Memory Access



Good fit
- Low CPU utilization
- High network speed

Types of Memory Disaggregation

- Kernel–space approaches

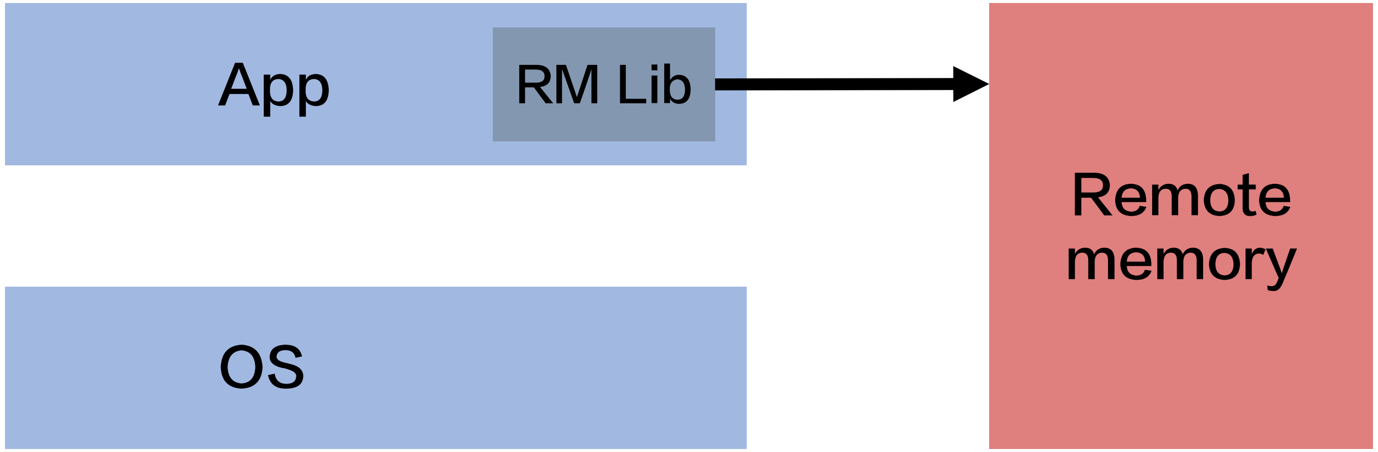


Pros
- Unmodified applications
- Transparent infra evolution

Cons
- High performance cost
- High development cost

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

Covered Work

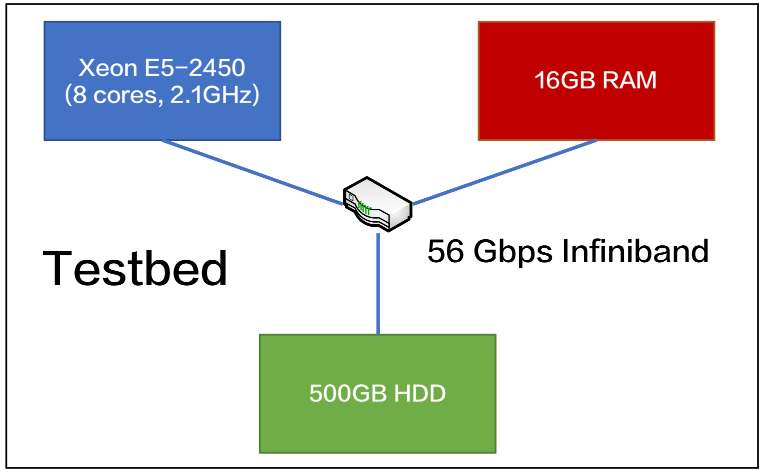
Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

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

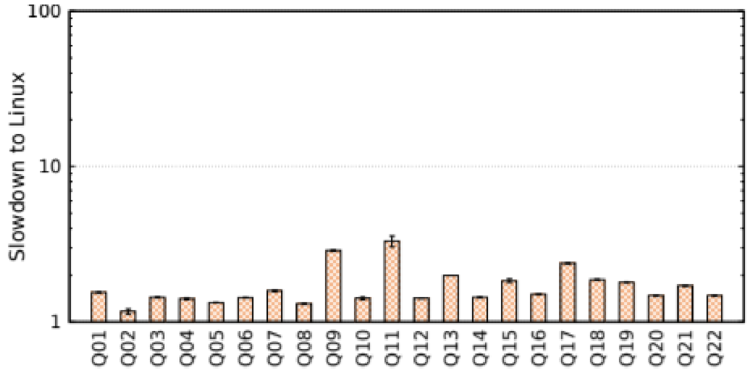
Disaggregation Cost

- 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)

Cost for In-memory Execution

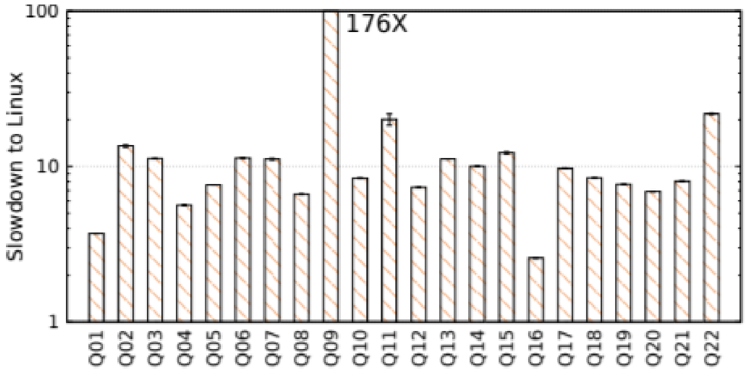
- MonetDB

1.7x slowdown



LegoOS (low degree of disaggregation)

18x slowdown



LegoOS (high degree of disaggregation*)

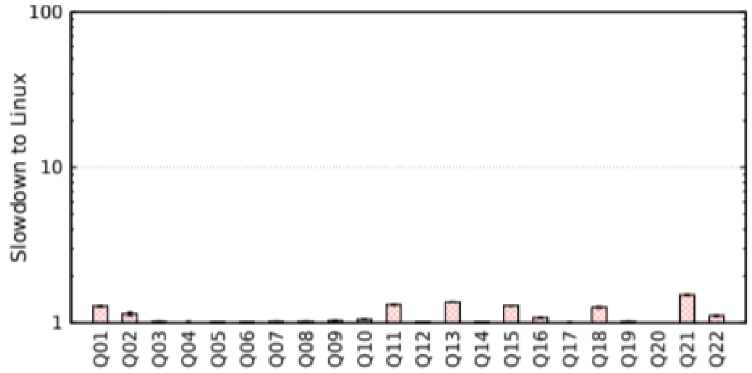
*low local memory size on compute node

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

Cost for Out-of-core Execution

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

1.08x slowdown



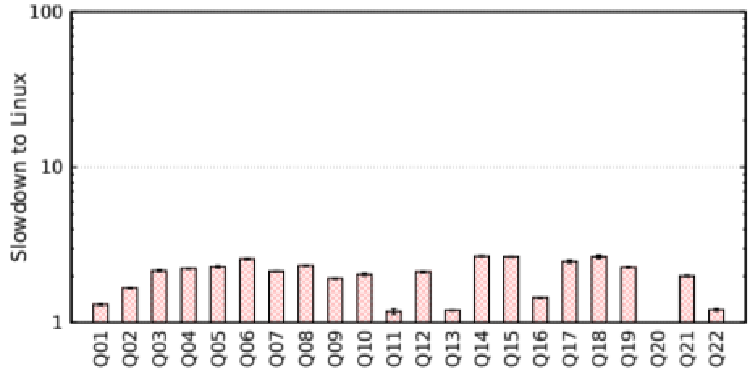
LegoOS (low degree of disaggregation)

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)

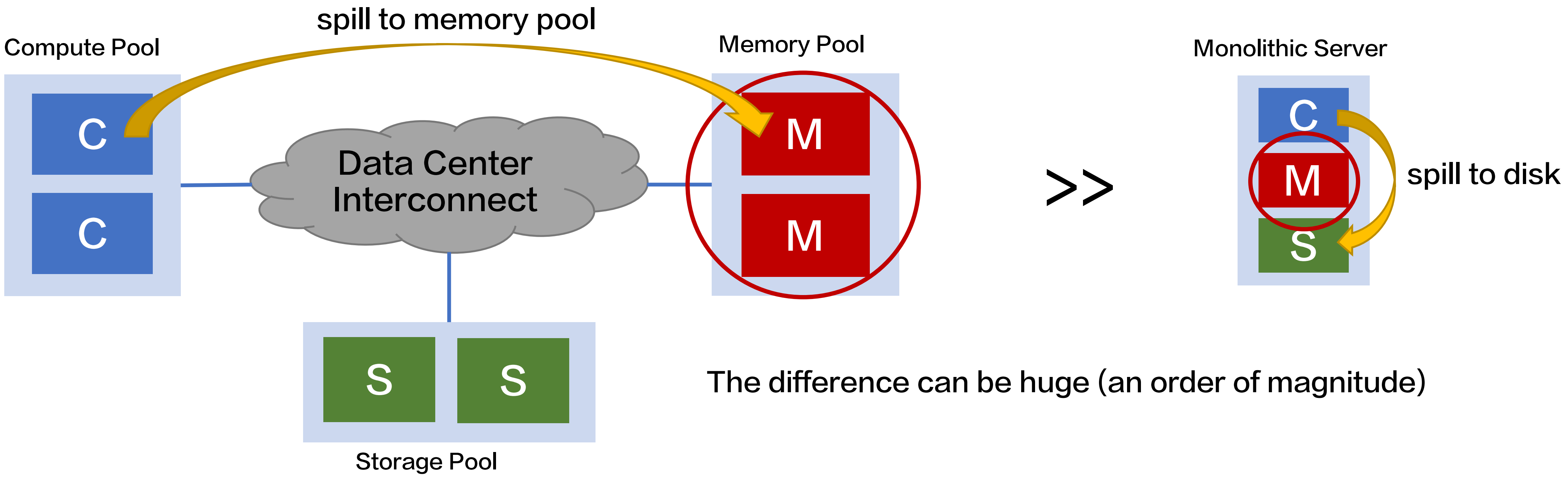
Summary of Disaggregation Cost

- 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



The difference can be huge (an order of magnitude)

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

Memory-disaggregated transactional systems

Covered Work

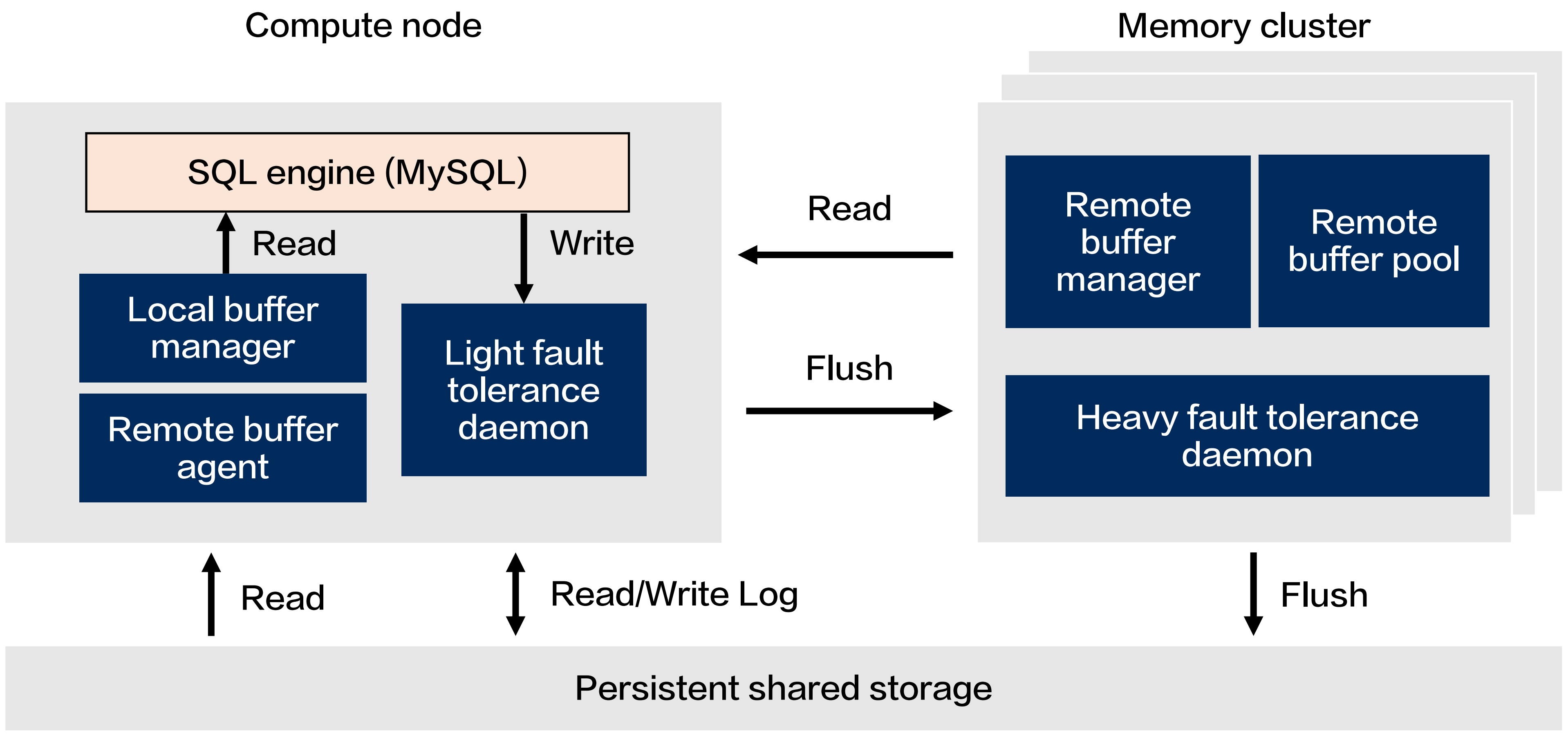
Understanding the effect on production DBMSs [VLDB 20]	*Implications*
LegoBase [VLDB 21]	*Transactions*
TELEPORT [SIGMOD 22]	*Analytics*
DirectCXL [ATC 22]	*CXL*

LegoBase

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

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

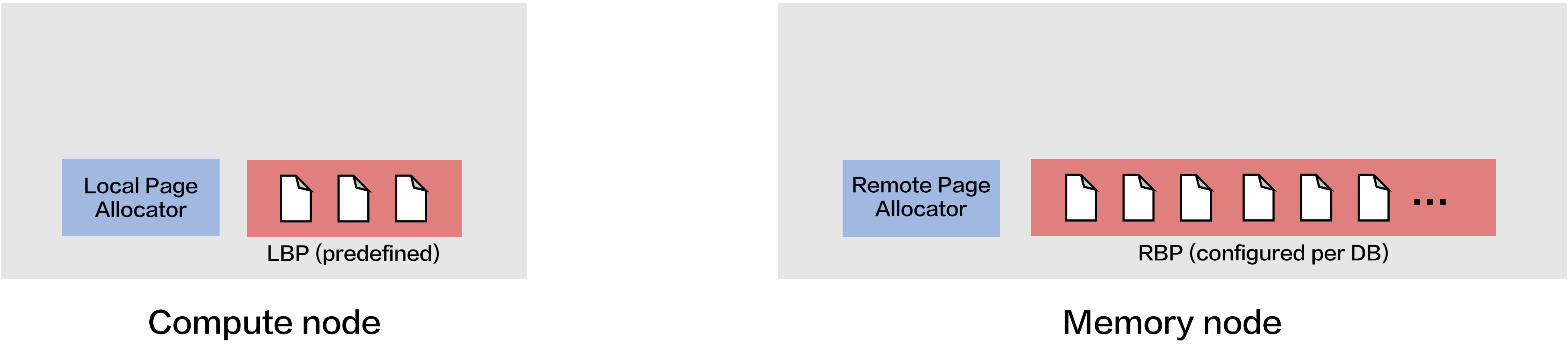
- Existing memory disaggregation has been OS-based
  - Infiniswap [NSDI 17], LegoOS [OSDI 18]
- Issue #2: low cache hit ratios with unified memory
  - Small but important data might be evicted, e.g., session info
  - OS LRU is less effective than DB-optimized LRU
  - Page size mismatch: 4KB in OS vs.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

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

Splitting Buffer Pool

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

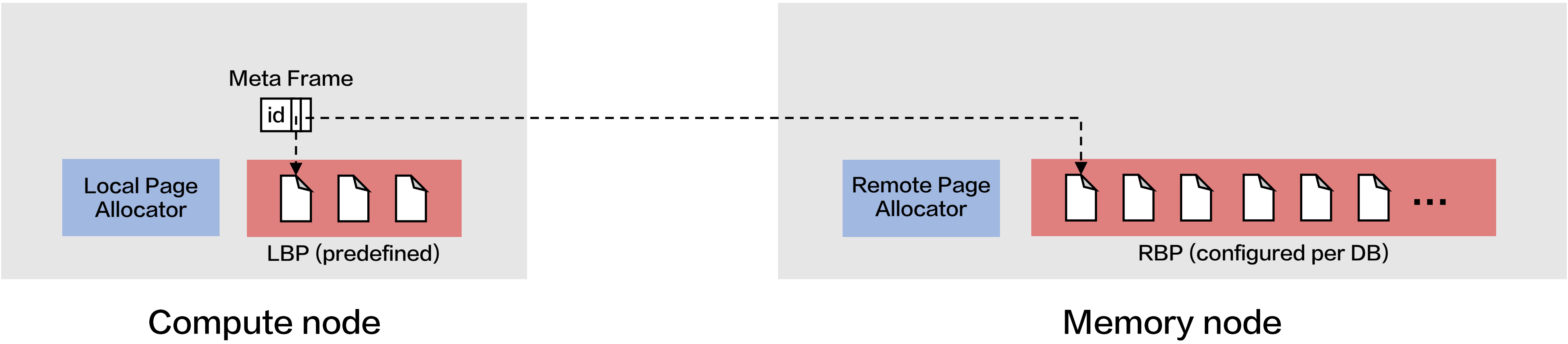
- LBP is a cache of RBP



Page Organization

Every page has a meta frame

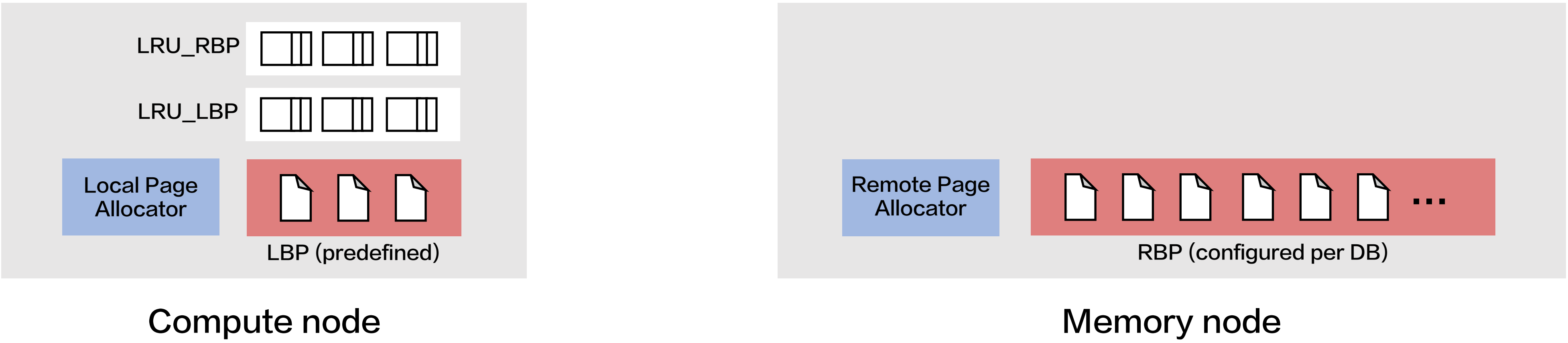
- Page id, local address, and remote address



Page Organization

Two LRU lists of meta frames on the compute node

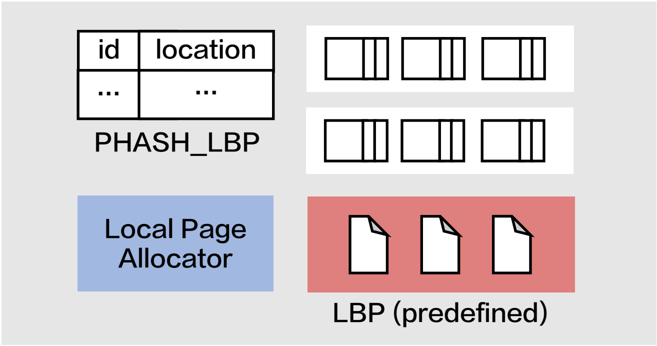
- LRU_LBP: MySQL-style LRU for local pages
- LRU_RBP: caching remote address for evicted pages



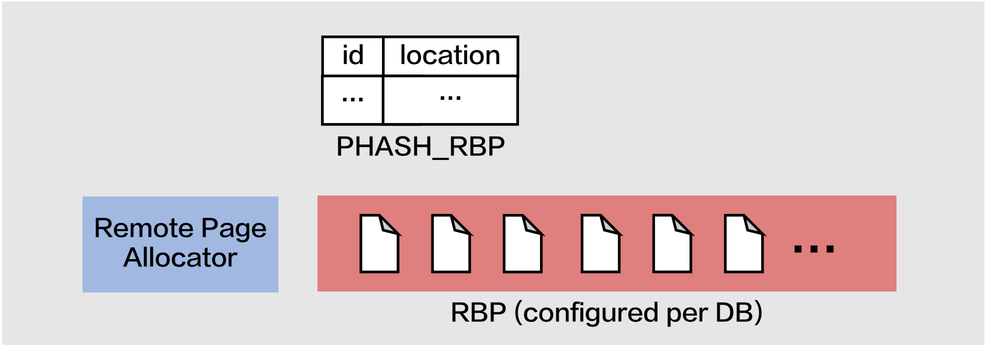
Page Lookup

Locating pages with hash lookups

- PHASH_LBP: pointing to the locations in the two LRU lists
- PHASH_RBP: pointing to local pages



Compute node

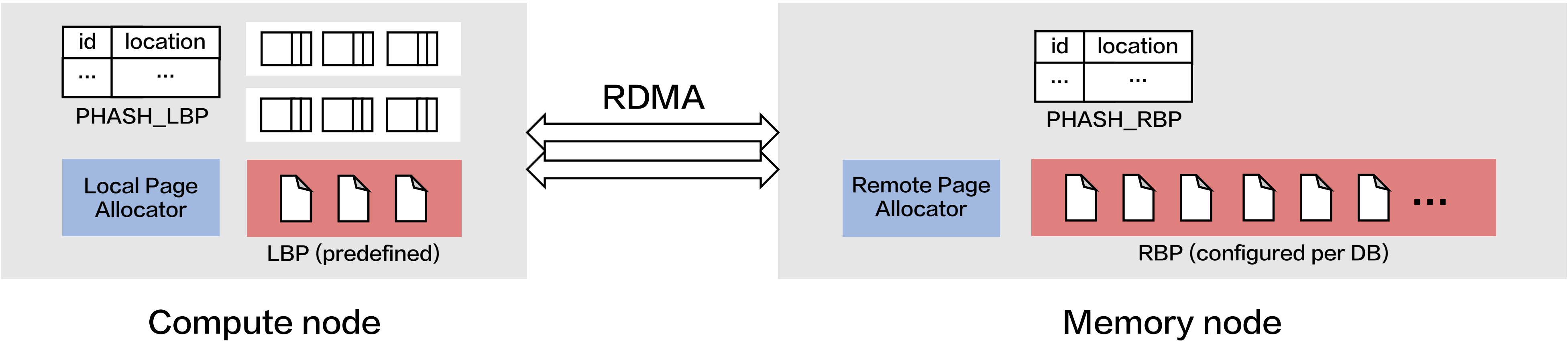


Memory node

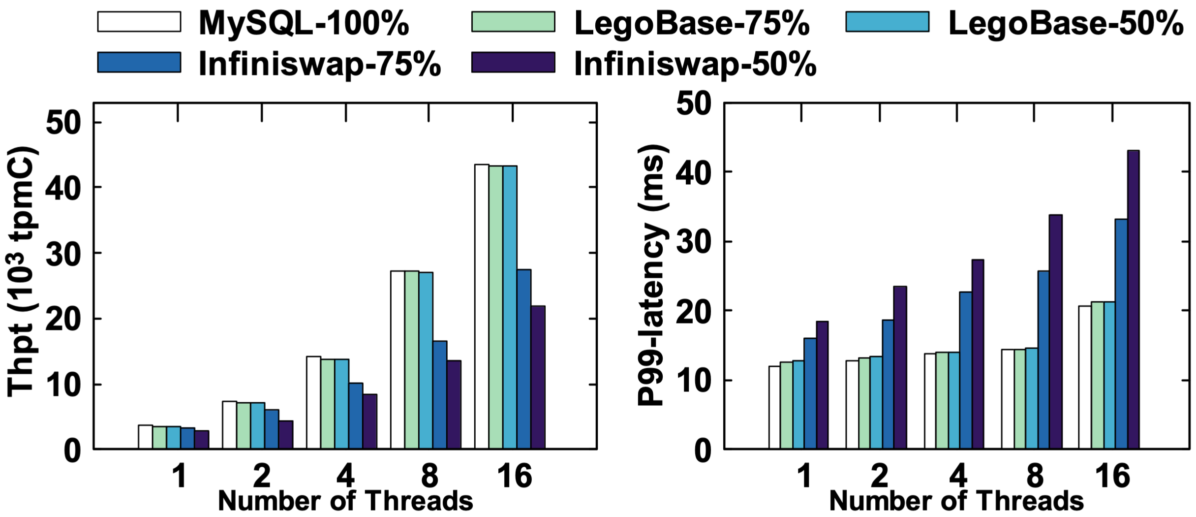
User-space Paging

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)

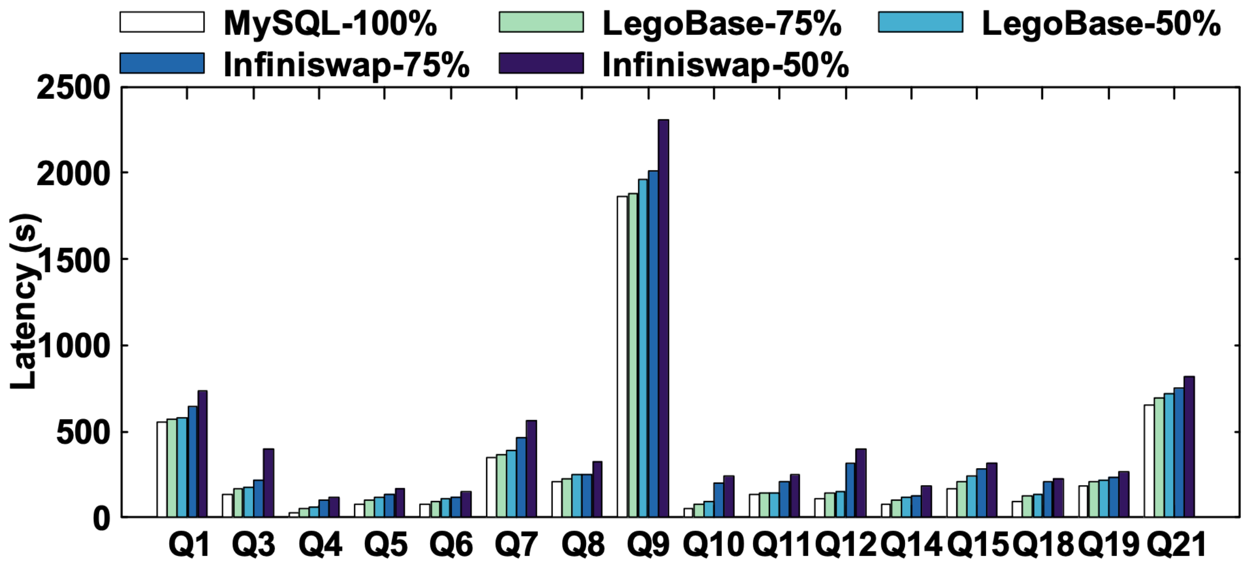


(a) Throughput
(b) P99-Latency

LegoBase outperforms Infiniswap

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

Result (TPC-H)

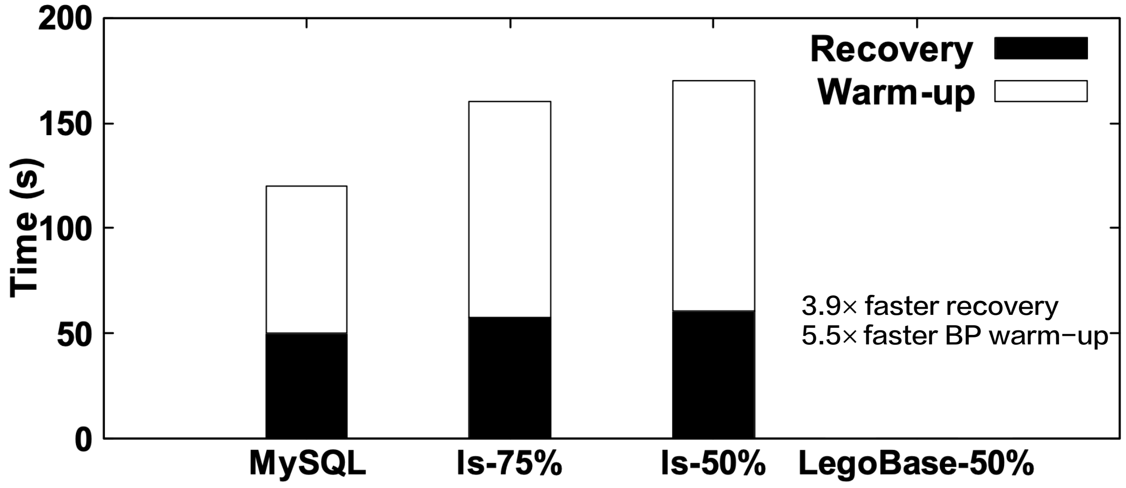


LegoBase query latency is close to monolithic MySQL

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

Fault Tolerance Motivation

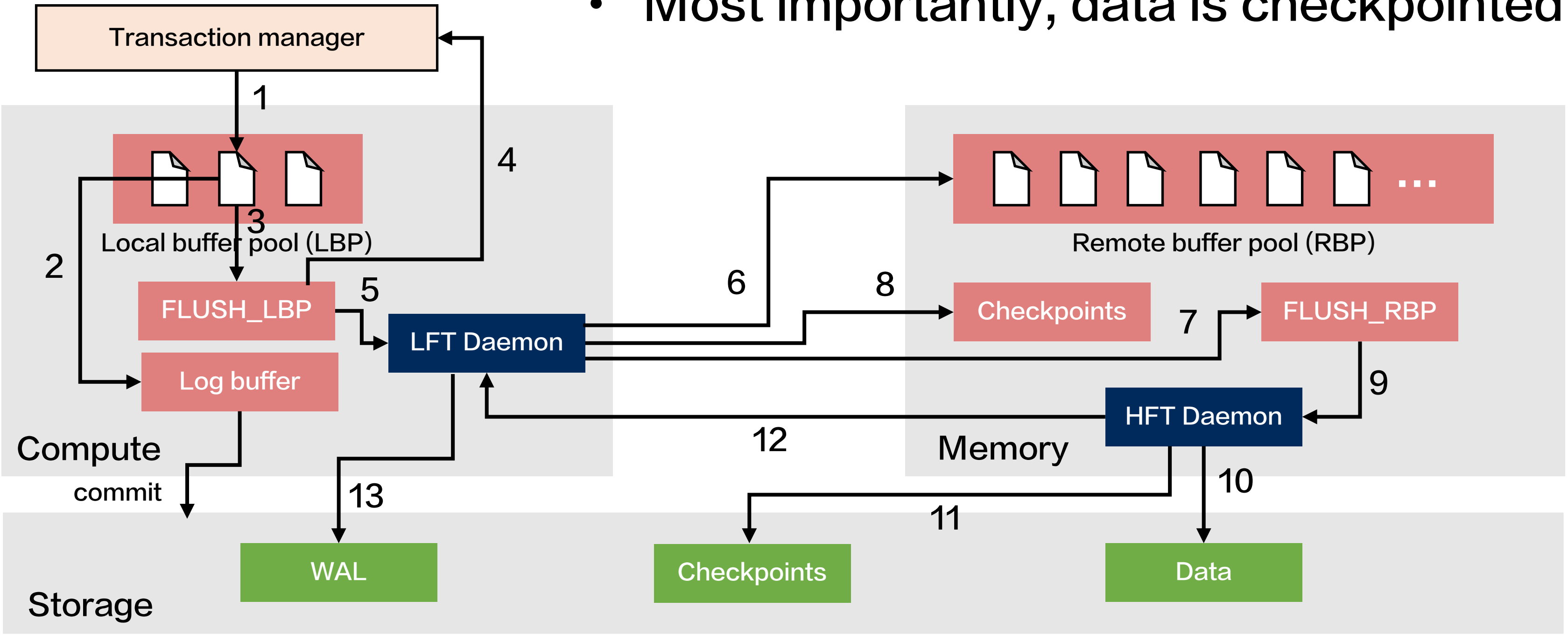
- Independent compute–memory failures introduce recovery opportunities
  - States saved in memory can speed up compute recovery



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

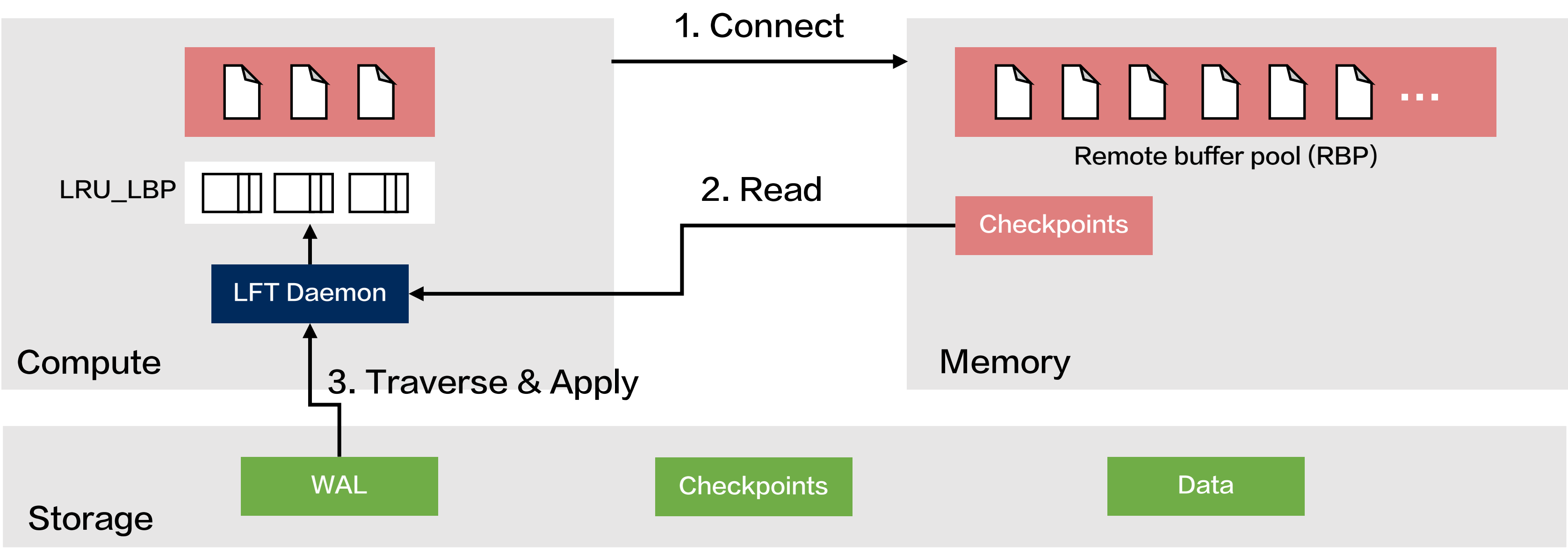
Two-tier ARIES

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



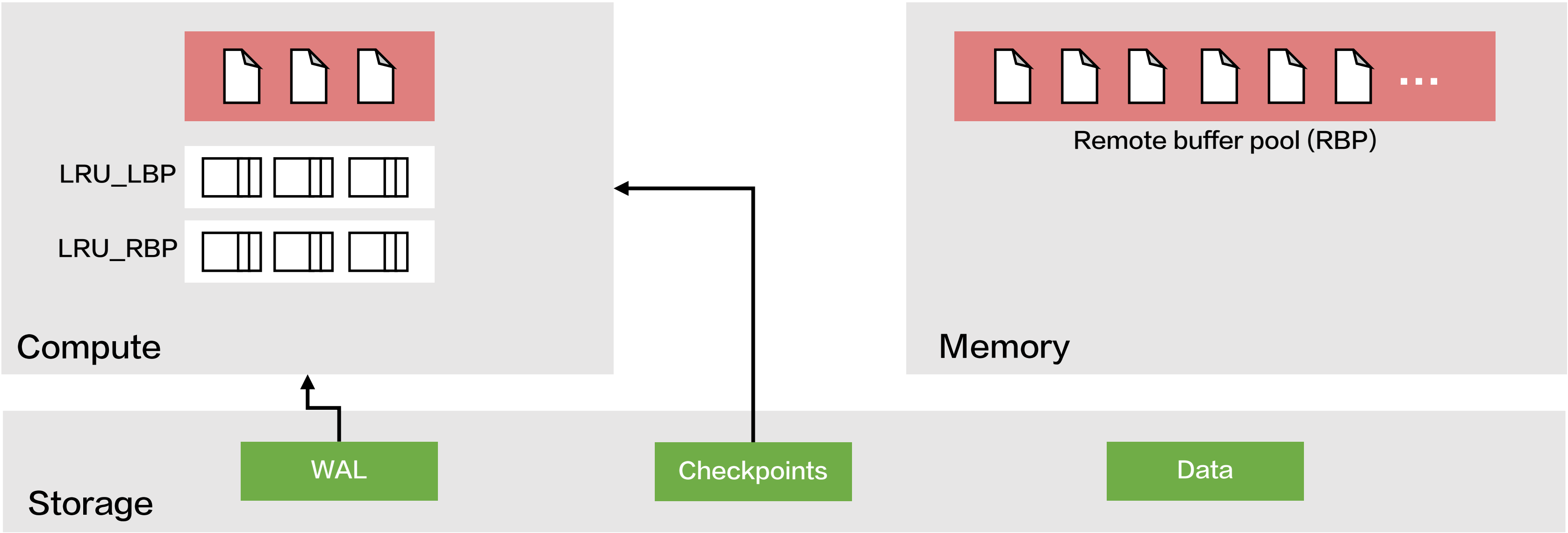
If Compute Fails...

- Recover fast from tier-1 checkpoints

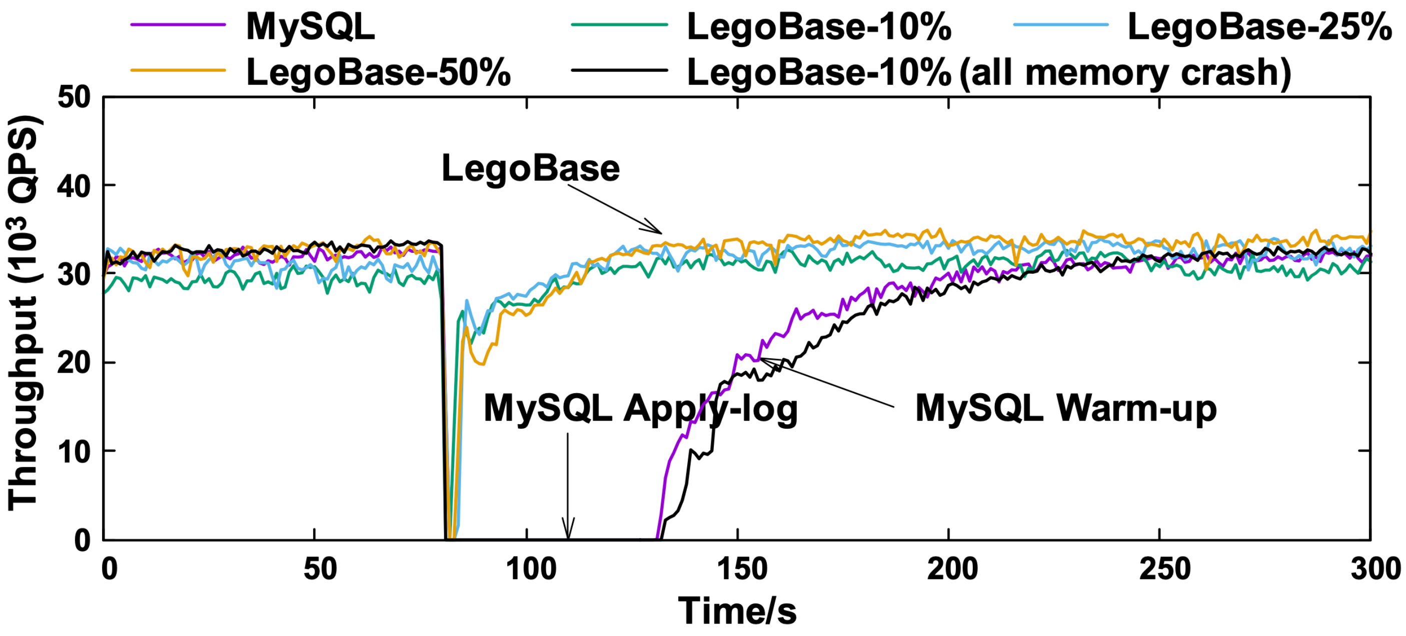


If Both Fail...

- Recover slowly from tier-2 checkpoints



Result



Recovery time

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

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*

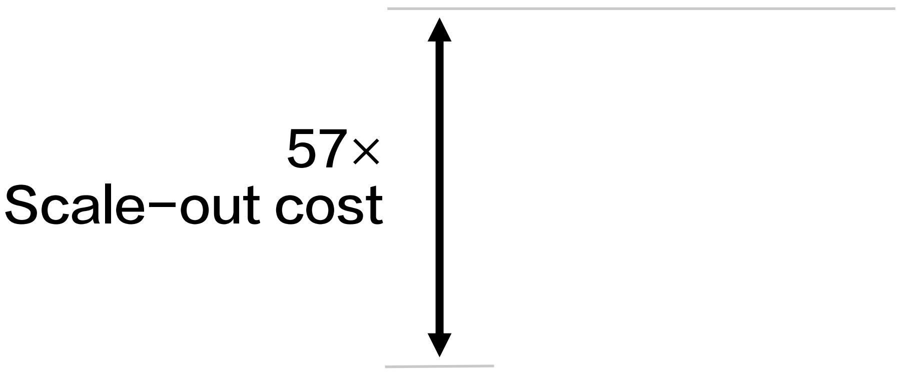
TELEPORT

A compute pushdown framework that moves operators from compute to memory

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

In-memory Query Performance

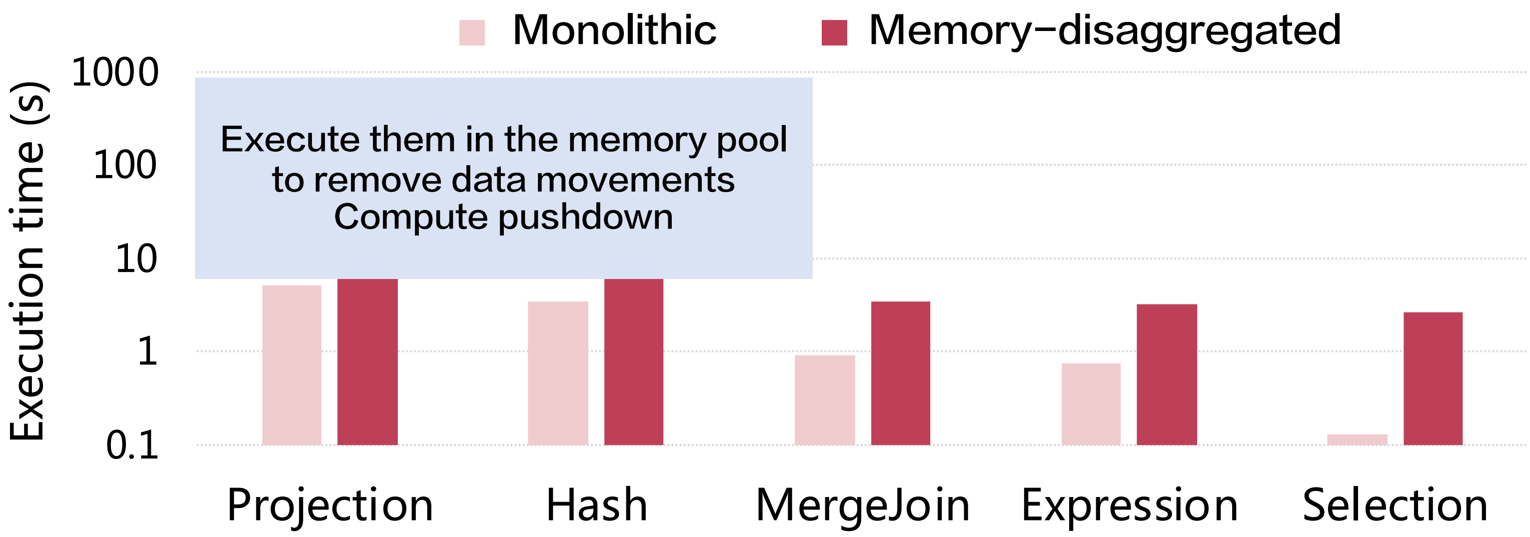
Monolithic vs. memory-disaggregated
MonetDB with TPC-H scale factor 50 (query 9)



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

TELEPORT Motivation

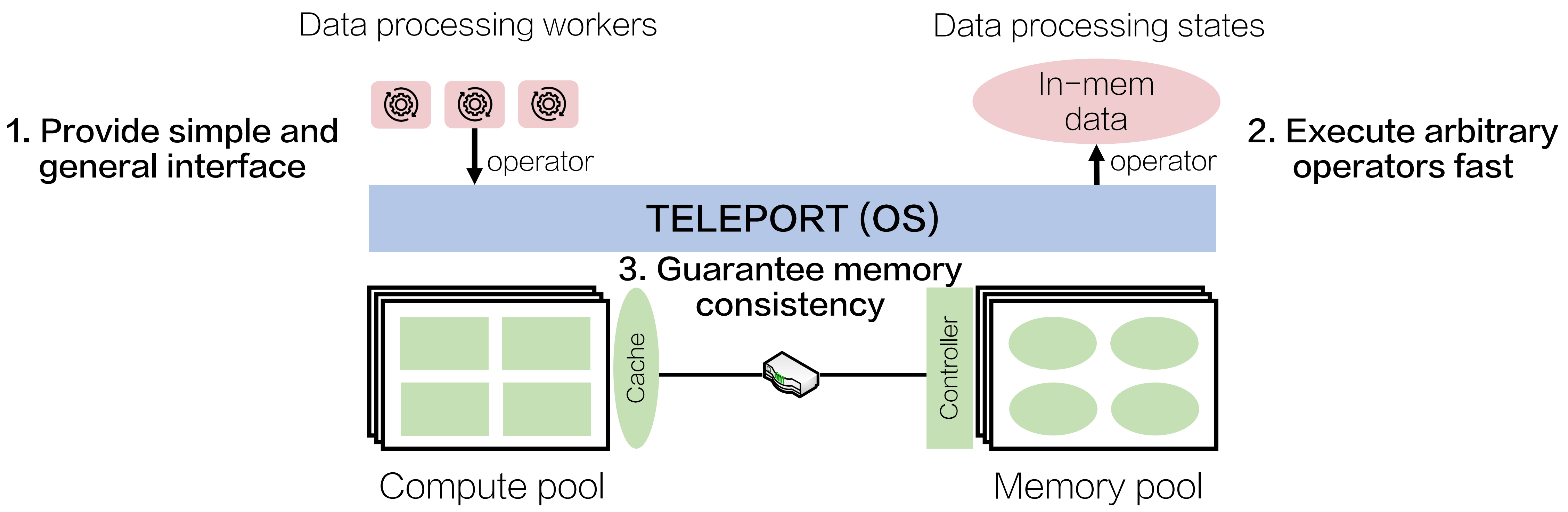
Monolithic vs. memory-disaggregated
MonetDB with TPC-H scale factor 50 (query 9)



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)`

  `fn, arg`: Function pointer, Argument pointer

  `flags`: Customization

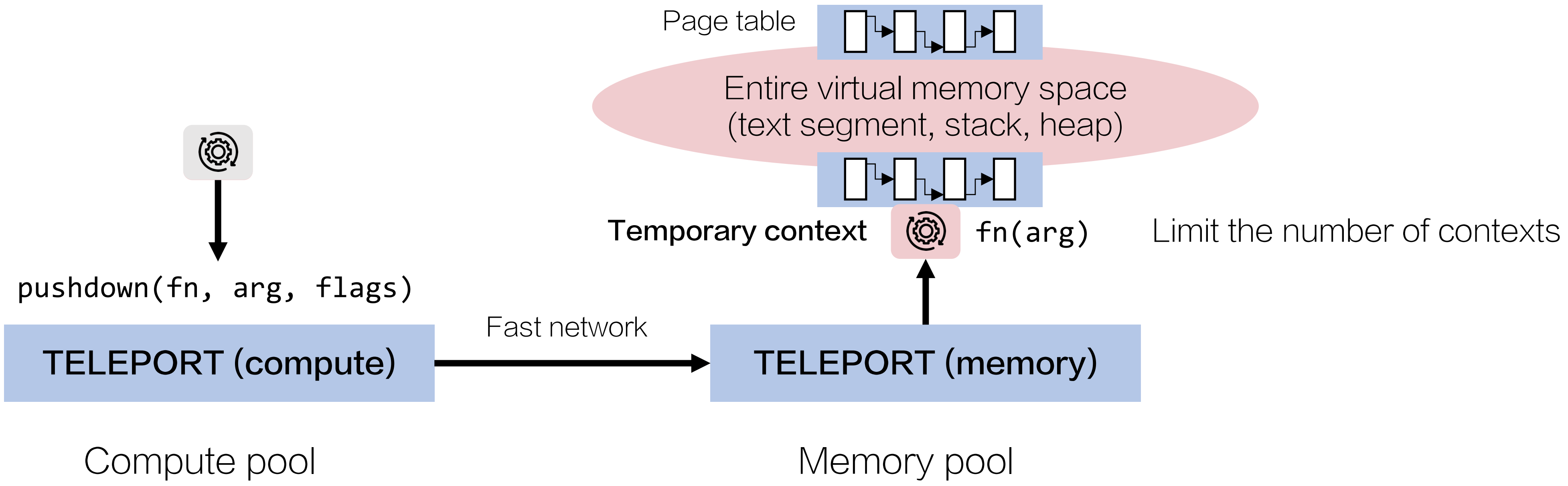
- Ported MonetDB (in-memory DBMS, 400,000 lines in total)
  - Projection, 117 lines
  - Aggregation, 214 lines
  - Selection, 302 lines
  - Hash, 75 lines

  } To unlock all disaggregation benefits

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

Memory Pool Execution

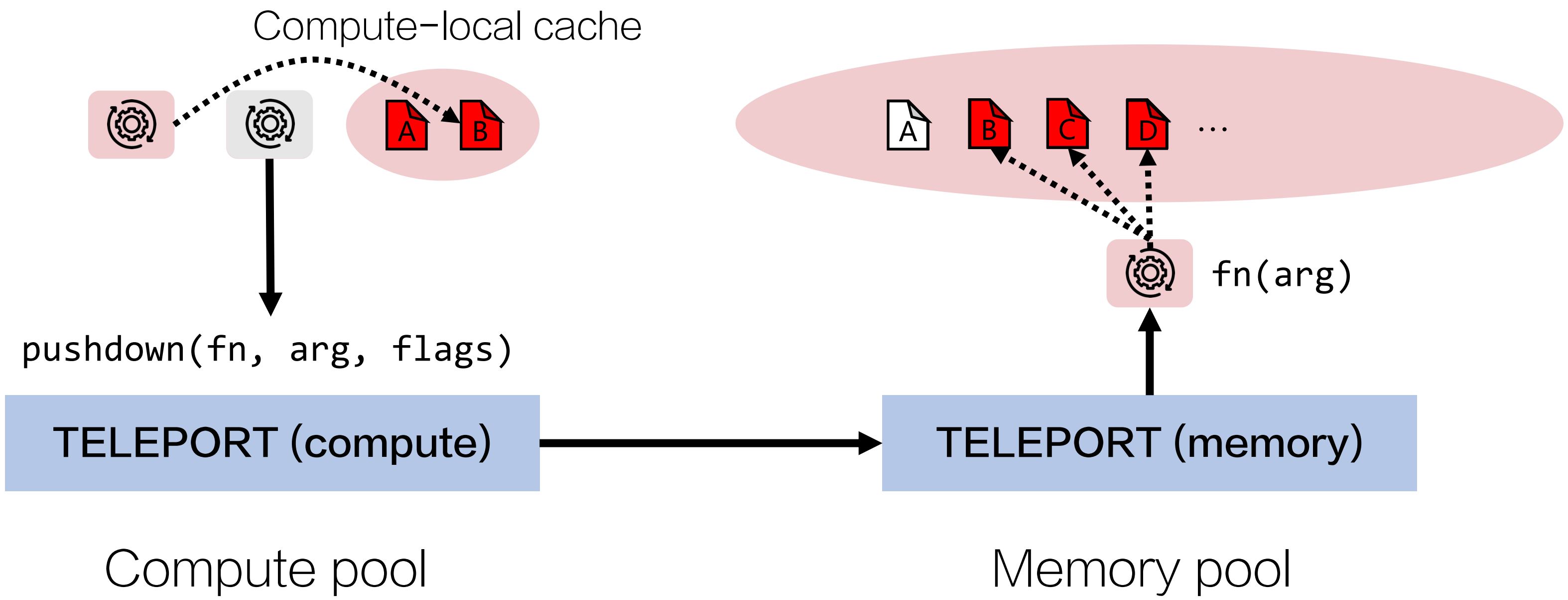
- Arbitrary and fast function execution
- Akin to POSIX vfork



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

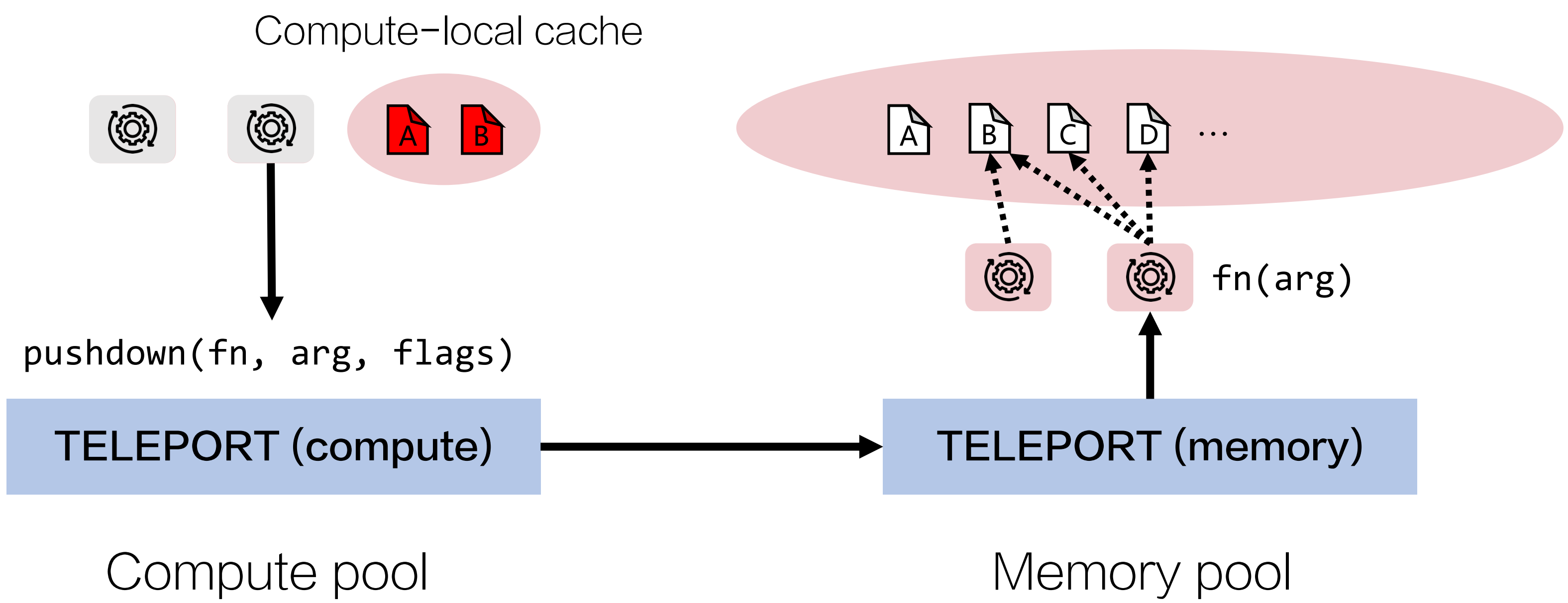
Data Synchronization

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



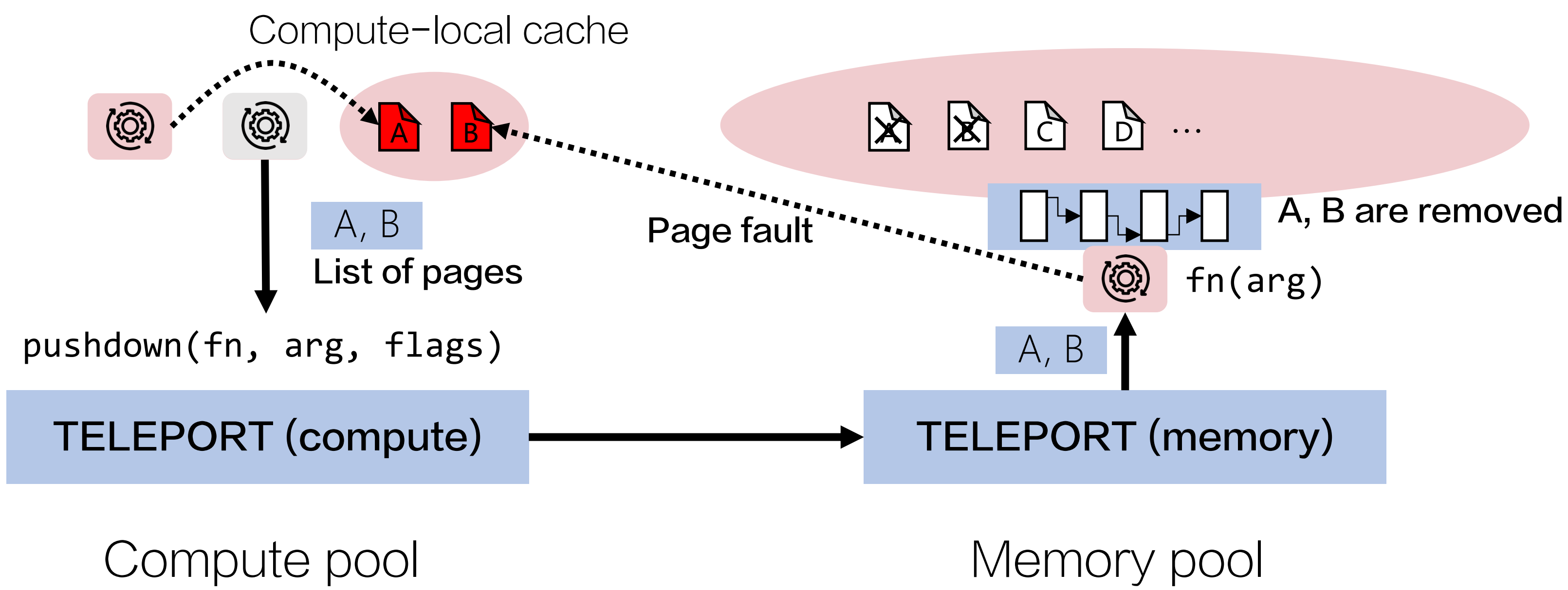
Baseline Approach

- 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



On-demand Coherence Protocol

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



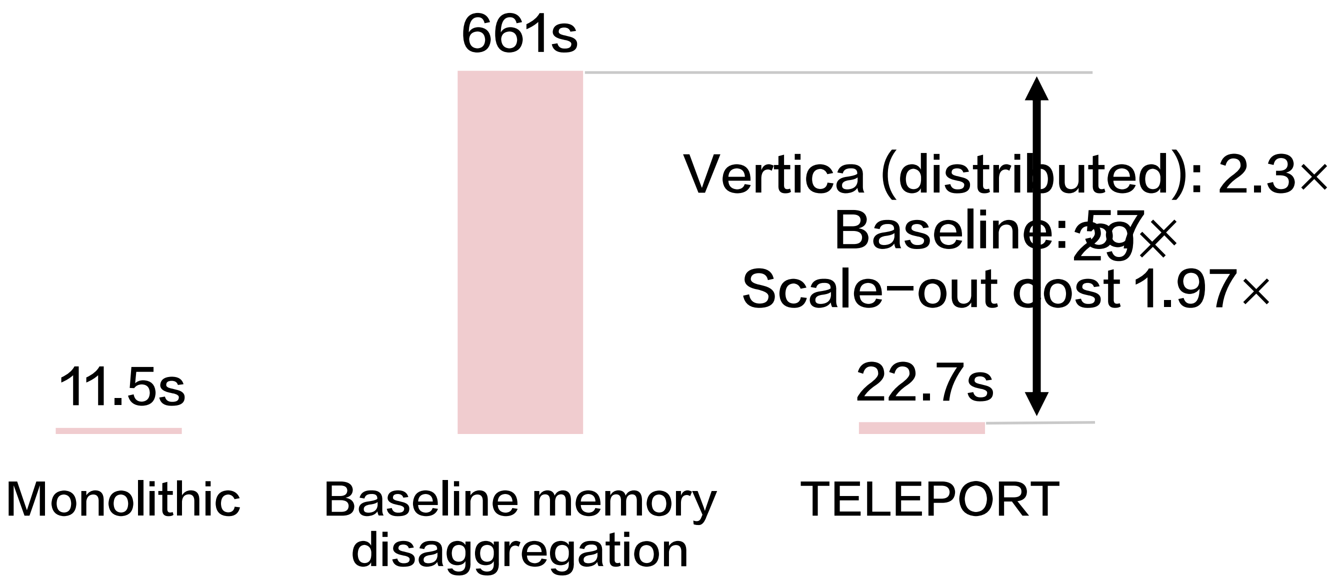
Evaluation Setup

- 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"

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

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

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)

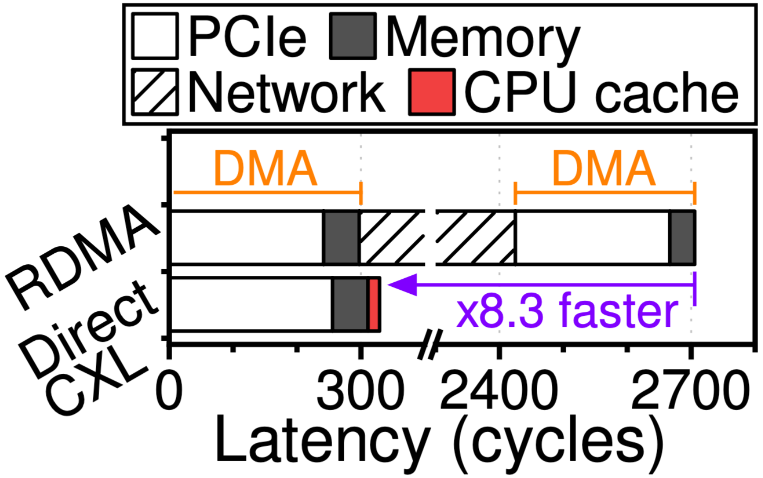
- Cache-coherent interconnects for connectivity between CPUs, accelerators, and I/O devices

- 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

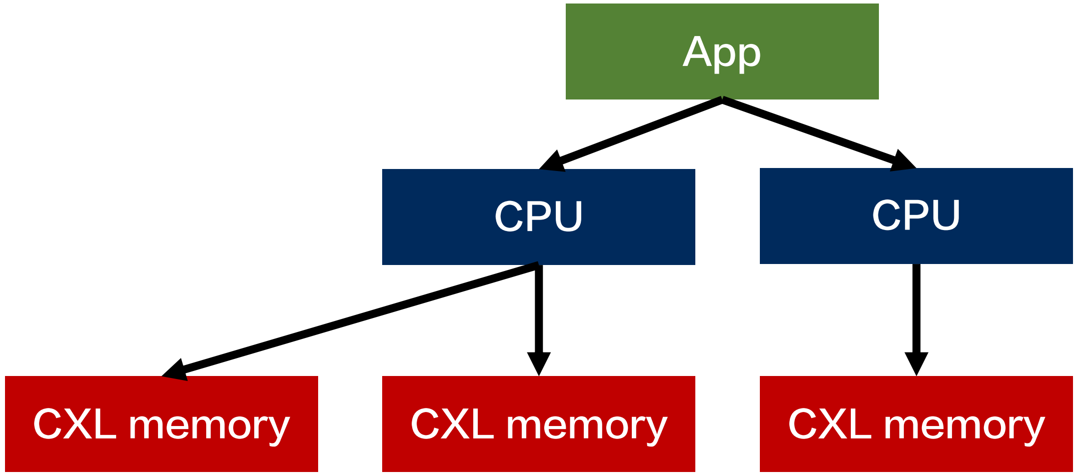
- No network latency
- No extra data copies



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?

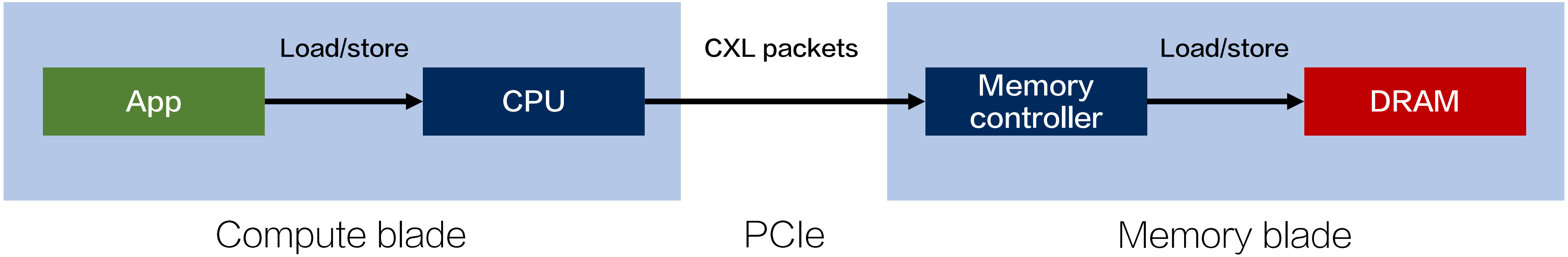


Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

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

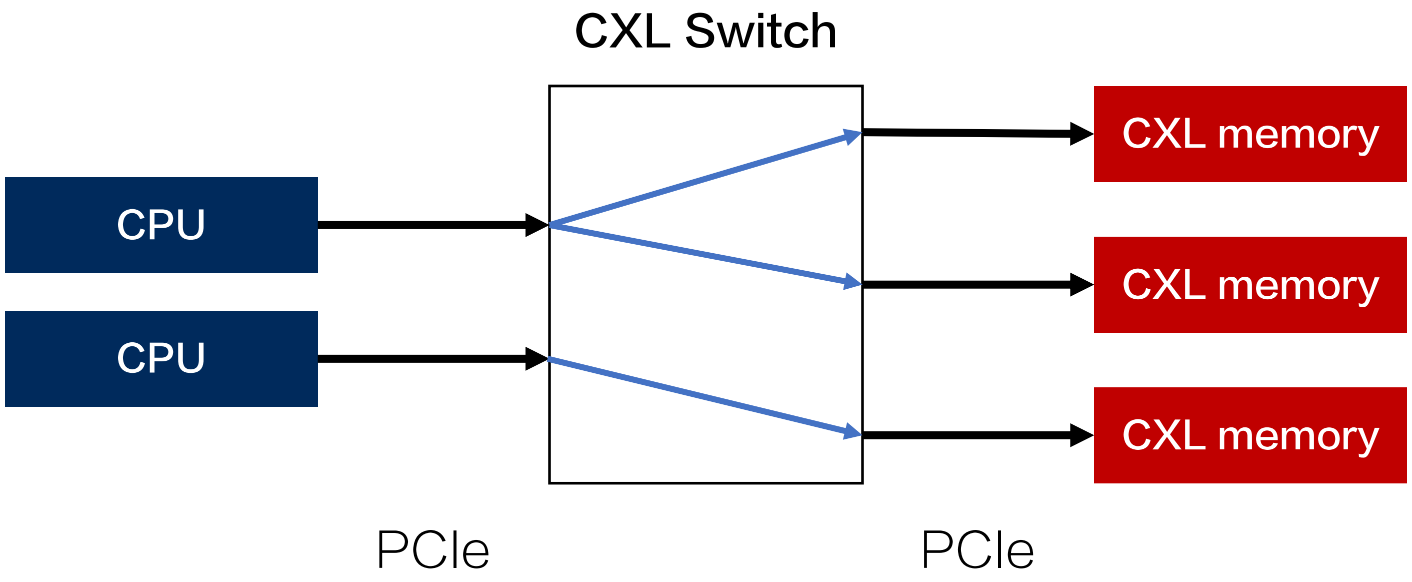


Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

DirextCXL Design

How to enable flexible memory configuration?

- A CXL switch with a reconfigurable crossbar

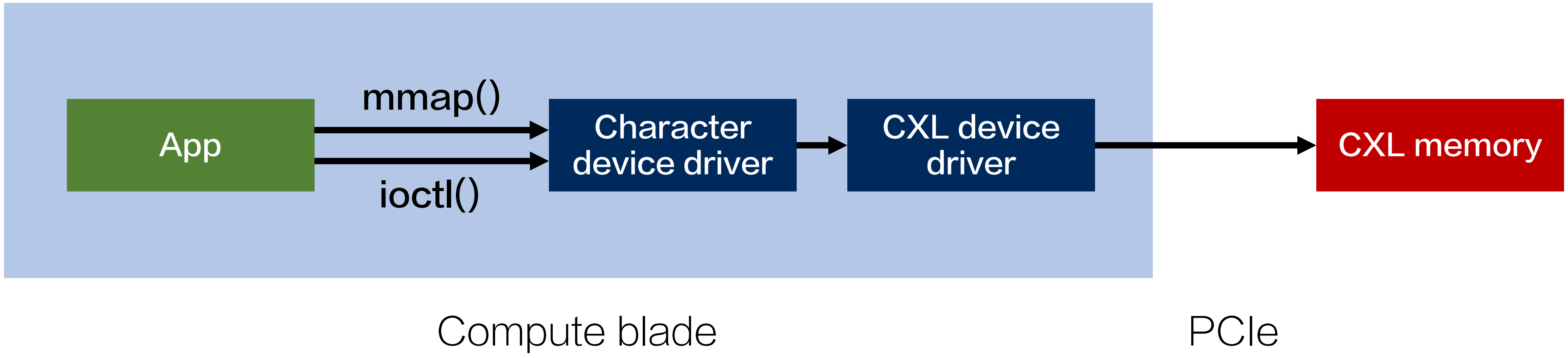


Direct Access, High-Performance Memory Disaggregation with DirectCXL
D. Gouk et al., ATC 2022

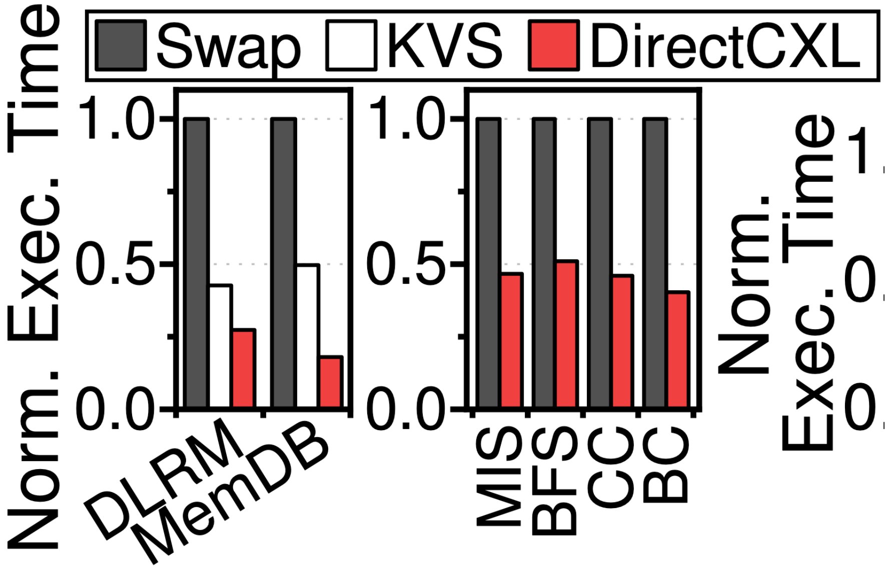
DirextCXL Design

How to present CXL memory to applications?

- Leveraging Linux virtual memory system



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

Future Directions

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

Q & A