Towards Cost-Effective and Elastic Cloud Database Deployment via Memory Disaggregation

Yingqiang Zhang¹, Chaoyi Ruan¹,², Cheng Li², Xinjun Yang¹, Wei Cao¹, Feifei Li¹, Bo Wang¹, Jing Fang¹, Yuhui Wang¹, Jingze Huo¹,², Chao Bi¹,²
**The Rise of Cloud-Native Databases**

- Cloud-native databases
  - leverage modern cloud infrastructures
  - target high performance, high elasticity, and low price
  - serve as building block for cloud applications
Most of the time, the utilization of CPU is below 50%.

However, for short time periods, it reaches up to 91.27% at peak.

The traditional monolithic setup fails to meet the demands.
Related Work: Resource Disaggregation

- **Disaggregated databases**
  - Amazon Aurora [SIGMOD’ 17], PolarDB [SIGMOD’ 21]
  - They only focus on compute and storage disaggregation.

- **General disaggregation approaches**
  - LegoOS [OSDI’ 18], Infiniswap [NSDI’ 17]
  - High memory access and failure recovery overhead, due to being oblivious to database I/O characteristics and going through traditional I/O stack
Challenge 1: High disaggregation overhead

- Run workloads with Infiniswap
- Remote memory pool is accessed via 25Gbps network

Takeaway 1: Fast (remote) memory access advocates the co-design of database kernel and memory disaggregation.
Challenge 2: High fault recovery cost

- Run MySQL atop Infiniswap with TPC-C and varied remote memory ratios
- Crash the MySQL instance and measure the fail-over time costs

The pages cached remotely are not used, thus precluding fast recovery for local crashes.

Takeaway 2: Independent fault handling requires to bridge the gap between the monolithic fault tolerance protocol and memory disaggregation.
Database co-design

- Bypass OS kernel
- Retain MySQL sophisticated LRU
- Leverage DB access patterns and data layout

**Computer Node (cNode)**

- SQL Execution Engine
- Local memory
- LRU_RBP
- LRU_LBP

**Global memory cluster**

- Remote Buffer Pool
- Remote page Allocator

**Persistent Share Storage**
Local and Remote Memory Management

Two-level LRU: hold all metadata of local and remote pages, reduce communications with remote memory node.
Local and Remote Memory Management

Page access from local buffer pool
Page access from remote buffer pool
Local and Remote Memory Management

SQL Execution Engine

<table>
<thead>
<tr>
<th>PageID</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table1-p2</td>
<td>p2_ptr</td>
</tr>
<tr>
<td>Table1-p1</td>
<td>p1_ptr</td>
</tr>
</tbody>
</table>

Page access from persistent shared storage
New Fault Tolerance: Two-tier ARIES Protocol

- **Tier-1 checkpoints**
  - Compute node flushes fine-grained checkpoints to remote buffer at high speed via RDMA

- **Tier-2 checkpoints**
  - Remote mem node flushes big checkpoints to storage as usual for persistence
New Fault Tolerance: Two-tier ARIES Protocol
New Fault Tolerance: Two-tier ARIES Protocol

- Remote Buffer Pool (RBP)
- Local Buffer Pool (LBP)
- Log buffer
- FLUSH_LBP
- LFT Daemon
- HFT Daemon
- Checkpoints
- FLUSH_RBP
- Persistent Shared Storage
- Redo Log
- Checkpoints
- Data
New Fault Tolerance: Two-tier ARIES Protocol
New Fault Tolerance: Two-tier ARIES Protocol
New Fault Tolerance: Two-tier ARIES Protocol

Remote Buffer Pool (RBP)

Local Buffer Pool (LBP)

Tx Manager

Redo Log

Checkpoints

Data

Persistent Shared Storage
New Fault Tolerance: Two-tier ARIES Protocol

1. tx_dml
2. FLUSH_LBP
3. LFT Daemon
4. ack
5. Local Buffer Pool (LBP)
6. Remote Buffer Pool (RBP)
7. Checkpoints
8. FLUSH_RBP
9. HFT Daemon
10. Persistent Shared Storage
11. Redo Log
12. Checkpoints
13. Data

New Fault Tolerance: Two-tier ARIES Protocol
New Fault Tolerance: Two-tier ARIES Protocol

Tx Manager

Local Buffer Pool (LBP)

Remote Buffer Pool (RBP)

LFT Daemon

HFT Daemon

Redo Log

Checkpoints

Data

Persistent Shared Storage

log buffer

FLUSH_LBP

Checkpoints

... → ... → ...

FLUSH_RBP

page page page ...

page page page ...

purge

FLUSH_RBP

Checkpoints

Redo Log

Checkpoints

Data

Persistent Shared Storage

1. tx_dml

2. FLUSH_LBP

3. ...

4. ack

5. ...

6. ...

7. ...

8. ...

9. ...

10. ...

11. ...

12. ...

13. purge
Experimental Setup

- Hardware configuration
  - Two Intel Xeon CPU E5-2682 v4 processors
  - 512GB DDR4 DRAM
  - 25Gbps Mellanox ConnectX-4 network adapter

- Workloads
  - TPC-C (20GB) / TPC-H (40GB) / Sysbench / Production workloads

- Baseline systems
  - Monolithic MySQL-8.0
  - MySQL over Infiniswap
Compared to the monolithic MySQL setup, Infiniswap worsen by up to 2.01x (2.35\times) in Throughput (P99 Latency)

But, LegoBase brings only up to 3.82% (4.42%) loss in Throughput (P99 Latency)

LegoBase-75%’s latencies are closer to the ones of MySQL-100%

LegoBase outperforms MySQL over infiniswap: e.g., runs up to 75.7% faster for Q11
Fast State Recovery and Warm-Up

- When all/remote memory crashes, LegoBase resorts to the normal MySQL recovery process.
- But, with only local node crashes, LegoBase can re-use vast majority of remote pages, and significantly reduces the state recovery and warm-up time costs.
- This can also benefit efficient planned hardware re-configurations.
LegoBase is a novel memory-disaggregated cloud-native database architecture

LegoBase is able to scale CPU and memory capacities independently with comparable performance as the monolithic setup without using remote memory

LegoBase can achieve faster state recovery and is more cost-effective than state-of-the-art baselines
Thank you!

Yingqiang Zhang
yingqiang.zyg@alibaba-inc.com

Chaoyi Ruan
rcy@mail.ustc.edu.cn