A CXL-Powered Database System: Opportunities and Challenges

Yunyan Guo
Tsinghua University
yunyanguo@tsinghua.edu.cn

Guoliang Li
Tsinghua University
liguoliang@tsinghua.edu.cn

***Abstract*—Compute Express Link (CXL) is emerging as a significant player in the landscape of modern database management systems (DBMS). CXL is an open industry-standard interconnect protocol between processors and devices such as memory buffers. Boasting high bandwidth, low latency, and support for coherency and memory semantics, CXL opens a new direction for addressing the limitations and bottlenecks faced by traditional distributed DBMS, particularly in large-scale data management, efficient query processing, and improving system availability. This paper explores the significant potential of employing CXL in constructing next-generation DBMS. Through a thorough analysis of CXL's key characteristics, this paper identifies emerging opportunities, particularly in buffer pool expansion, memory elasticity, swift data recovery, and index optimization. More importantly, this paper outlines a series of new challenges accompanying these opportunities, with the objective of inspiring cutting-edge approaches in future DBMS design that emphasize efficiency, reliability, and reduced total cost of ownership.**

***Index Terms*—Compute express link, database management system, memory disaggregation**

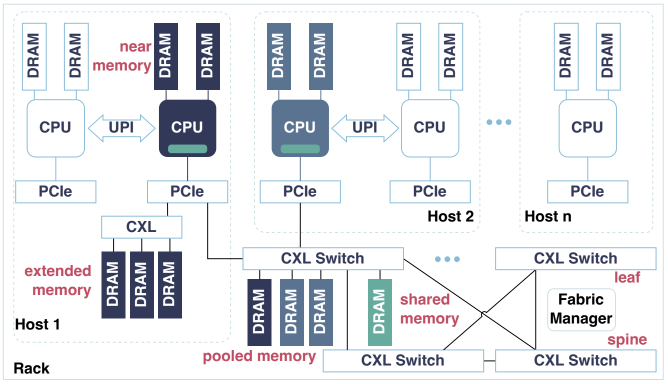

Fig. 1. Memory hierarchy with CXL protocols in a Rack: *Near memory* is closest to the computing units in NUMA nodes. *Extended memory* is connected through CXL protocols on PCIe slots. *Pooled memory* can be scheduled and allocated by *CXL switches*. *Shared memory* ensures coherency across multiple CPU caches. *Leaf switch* directly links to CXL devices, while the *spine switch* facilitates communication among leaf switches.

I. Introduction

Compute Express Link (CXL) [1]–[4] is an open industry-standard that defines a family of interconnect protocols between processors and devices, particularly memory buffers, catering to applications requiring substantial memory capacity [5]–[7]. Fig. 1 illustrates the various memory layers within a CXL-based rack. This depiction is a logic architecture that is derived from the recent CXL tutorial [8], with each component named according to the terminology in the lecture [9].

CXL has multiple advantages compared to other interconnect protocols. Crucially, it brings memory semantics and cache coherency support between CPU caches and CXL-based memory (as the same color in Fig. 1). Moreover, CXL has a high bandwidth that scales with PCIe bandwidth, but has significantly lower latency compared to PCIe/RDMA. These advantages together offer innovative solutions to longstanding challenges in database systems, particularly in areas of large-scale data management, query processing efficiency, and system availability. Despite extensive and in-depth research by database researchers on distributed DBMS, they have consistently been constrained by limited interconnect capacity, communication speed, and bandwidth, struggling to keep pace with growing demands. The rapid advancement of CXL and related hardware technologies makes large capacity, low latency, and quick recovery achievable, offering promising solutions to these persistent challenges.

Opportunity #1: Scaling buffer pool. Distributed DBMSs are often hampered by the scattered distribution of memory resources across diverse machines and devices, which complicates memory management [10] [11]. CXL-based memory, often referred to as *far memory*, has coherency and memory semantics. This enhancement allows CPUs to cache far memory similarly to near memory, and also enables far memory to cache data from the system memory effectively. This simplifies memory management and the construction of memory pooling [12] [13], opening up new opportunities for constructing large-capacity memory buffer pools in DBMS, simplifying memory resource management, and making the scaling of buffer pools feasible, significantly enhancing memory utilization efficiency and performance in database systems.

Opportunity #2: Elastic memory allocation. In traditional distributed database systems, the static allocation of memory resources to computational resources often leads to inefficiency and stranding, especially in tasks that are memory-

Guoliang Li is the corresponding author. This paper was supported by National Key R&D Program of China under Grant Number 2023YFB4503600, the China Postdoctoral Science Foundation under Grant Number 2022M711809, NSF of China (61925205, 62232009, 62102215), and Zhongguancun Lab.

intensive [14] [15]. However, CXL, as a high-speed memory interconnect protocol, uniquely enables the dynamic reassignment of far memory resources to different hosts over time, without having to reboot these hosts. This innovative feature enables DBMS to dynamically adjust memory allocation in response to the varying memory requirements of each query as it executes, ensuring optimal resource utilization throughout the query's life cycle. Consequently, this dynamic memory allocation mechanism not only maximizes resource efficiency, reducing waste, but also enhances the adaptability and responsiveness of the DBMS to diverse query workloads, significantly improving overall performance.

Opportunity #3: Fast data recovery from far memory. In traditional distributed DBMSs, fault recovery is a time-consuming process, primarily dependent on recovering data from disk transaction logs and synchronizing data across multiple nodes [16]. Fortunately, the high bandwidth and dynamic memory allocation of the CXL protocol hold promise for ultra-fast fault recovery in distributed compute-memory-storage disaggregation DBMSs. This includes, but is not limited to, swiftly reorganizing memory resources during failures and recovering data from backups in far memory, thereby reducing dependence on slower disk-based log recoveries. Additionally, the support for fine-grained memory sharing among multiple hosts, provided by the CXL switch, enables rapid node switching to maintain uninterrupted service, even in single-node failure scenarios, thus facilitating new avenues for quick and efficient data recovery.

Opportunity #4: Efficient index design. Index technology is intricately influenced by underlying technical details, including latency and bandwidth between memory and disk accesses, which shape structures such as B+ tree page indices, and communication costs among distributed nodes, impacting the maintenance of global secondary indices [17]. The CXL protocol, with its low latency, high bandwidth, and substantial capacity, poses novel requirements for index designs tailored to far memory data. Unfortunately, the multiple-layer far memory and elastic memory allocation in the CXL-based architecture present new challenges in index maintenance. The construction of an index for the CXL-based architecture can fully leverage the advantages of new hardware designs, enhancing the efficiency and scalability of index management, and thereby boosting the overall performance of DBMS.

Contribution. This paper presents a list of challenges when leveraging CXL protocols in DBMS design from four aspects.

- *Buffer Pool Management* (Sec. III): Designing and managing a hybrid buffer pool that integrates the multi-layered memory of CXL is a significant challenge.
- *Elastic Memory Scaling* (Sec. IV): Fully utilizing the elastic scalability of memory requires the development of appropriate strategies for distributed memory and data.
- *Database Recovery* (Sec. V): Fast database recovery using far memory necessitates maintaining a comprehensive transaction consistency mechanism to ensure data integrity before recovery can proceed.
- *Index Design* (Sec. VI): The advantages of low latency and high bandwidth support frequent data exchange, while also introducing new requirements for lightweight and fast index updates.

We argue that the next wave in DBMS innovation should consider CXL. Solving the challenges proposed in this paper will move DBMS design to the next frontier.

II. CXL Characteristics and Products

The evolution of specifications and products on CXL protocols reflects the rapid development [8].

A. CXL 1.1 Specification

The CXL 1.1 Specification [1] introduces significant advancements in cache coherency and memory semantics. CXL supports three protocols that include CXL.io, CXL.cache and CXL.mem. CXL.io is similar to PCIe 5.0. CXL.cache allows a device to cache data from the host memory. The host uses CXL.mem to access device-attached memory with load and store commands. Different combinations of these protocols demonstrate unique applicability to various device types. Particularly, in memory expanders (also called type 3 devices), the combination of CXL.io and CXL.mem plays a crucial role, allowing CPUs to cache memory from these devices. This paper focuses on exploring the potential of CXL-based memory devices in innovative DBMS design, aiming to enhance data management and processing efficiency. Additionally, investigating other CXL-based computing or storage devices, such as GPUs (type 2 devices), for enhancing DBMS design also holds certain significance, but is reserved for future exploration.

Products encompassing this iteration are brought to mass production between the second quarter of 2023 and the first quarter of 2024, including Intel Sapphire Rapids Xeon processors and Agilex 7 FPGAs for memory devices, Samsung's CXL Memory Expander.

B. CXL 2.0 Specification

The CXL 2.0 Specification [2], through the design of a single-layer CXL Switch, facilitates the connection between multiple hosts and multiple memory expansion devices. Its key innovation lies in the introduction of dynamic memory allocation functionality, allowing for the allocation and deallocation of memory resources across hosts without the need for system reboots. As shown in Fig. 1, the CXL memory space managed by the middle CXL Switch can be reallocated from Host 1 to Host 2 as needed, without requiring a reboot of Host 1. Pooling memory handled by CXL Switch significantly enhances the flexibility and efficiency of memory usage. If fully leveraged in the development of DBMS, this technology can effectively meet the dynamic and rapidly fluctuating memory demands associated with hybrid query workloads. Therefore, in designing DBMS on servers equipped with CXL Switch, prioritizing the utilization of pooled memory resources offered by CXL Switch is crucial for optimizing service efficiency.

This version is anticipated to enter mass production from the second quarter of 2024 to the first quarter of 2025. Presently, there are demonstrations available, such as the academically oriented DirectCXL [18], and from the industrial sphere, XConn Technologies demonstrates a Composable Memory System (CMS) that leverages their CXL 2.0 switch [19]. Coupled with CXL memory expanders from Samsung, software from MemVerge, and H3 Platform, a 2TB pooled CXL memory system for 8 hosts is built [20].

C. CXL 3.0 Specification

First, the CXL 3.0 Specification [3] significantly extends the boundary further by integrating CXL switches through the use of Fabric Manager, thereby achieving enhanced scalability with low latency. The CXL Fabric Manager facilitates the setup, deployment, and modification of the network environment, managing up to 4096 nodes (including CPU hosts and CXL memory devices) via port based routing. Fig. 1 illustrates a topology diagram for a CXL fabrics use-case, categorizing CXL switches into leaf type and spine type. The leaf switches connect to hosts and devices, whereas the spine switches link each leaf switch, ensuring that the communication path between any two hosts/devices involves no more than three CXL switches. This design, with multiple spines distributing the communication burden, guarantees low latency.

Second, it supports dynamically composable systems at the rack level or pod level, enabling fine-grained memory sharing across multiple hosts. By employing hardware coherency techniques, it facilitates shared coherent memory among hosts. A major enhancement in CXL 3.0 is its ability to back invalidate the Host's caches. Maintaining coherency for host-managed device-attached memory (HDM) is called enhanced coherency, replacing bias based coherency of previous generations. As depicted in Fig. 1, the CXL memory space in the right device of the middle CXL switch is shared by Host 1 and Host 2, each CPU caching it. Before Host 1 modifies any data within this shared space, this CXL switch ensures that Host 2's cache is invalidated. This complex process is managed jointly by hardware mechanisms and CXL protocols.

Products based on this version are expected to launch from the second quarter of 2025 through the end of 2026 or later. While this architecture is predominantly targeting high-performance computing cluster products, the capabilities introduced by CXL 3.0, such as dynamic system composition and fine-grained memory sharing, unlock significant opportunities to boost high concurrency, scalability, and availability in the next-generation DBMS. Fully leveraging the advantages of this new architecture requires DBMS designs to focus on efficient utilization of memory resources, harmonizing with anticipated performance features of future products to exploit CXL 3.0's advantages.

D. CXL 3.1 Specification

The latest (Dec 2023) CXL 3.1 Specification [4] primarily enhances fabric connectivity features, introducing global integrated memory (GIM) to facilitate communication across multiple hosts and improving fabric decoding and routing capabilities. Moreover, it incorporates security considerations, including support for the Trusted Execution Environment (TEE) Security Protocol (CXL-TSP), which strengthens confidential computing capabilities for directly attached CXL memory expander devices. This advancement significantly contributes to privacy protection and data security when building CXL-based DBMS, and the exploration of these advancements will be reserved for future discussions.

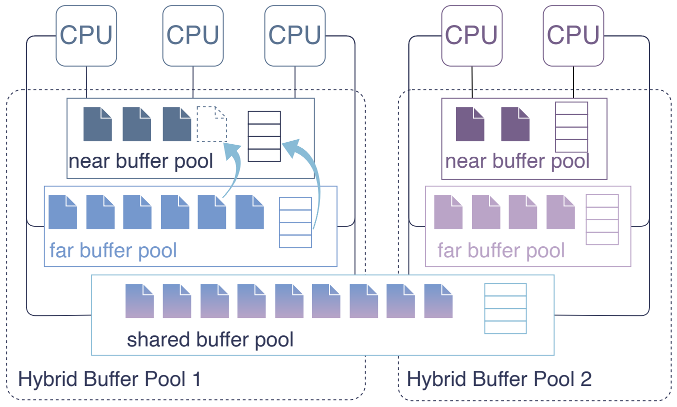

Fig. 2. Example of the *hybrid buffer pool* structure, including two hosts connected through a CXL switch. It is further divided into a *near buffer pool*, a *far buffer pool*, and a *shared buffer pool* that facilitates data sharing via the CXL switch. In DBMS, the memory management engine for the hybrid buffer pool is required to support dynamic allocation and migration of data pages across different buffer zones and to update the page directory to reflect the current location of data pages (as indicated by the two arrows in the figure), thereby improving data access efficiency and enhancing overall system performance.

In summary, the evolution of CXL protocols opens new avenues for the design and optimization of distributed database systems, particularly in areas such as memory disaggregation in cloud-native databases, shared memory management, elastic memory scaling, database fault recovery, transaction processing, and index optimization. This paper treats CXL 1.1, CXL 2.0, and CXL 3.0 as part of the same protocol family, considering their characteristics collectively.

III. Buffer Pool Management

In traditional distributed database systems, memory resources are distributed across various machines, multiple PCIe memory devices, and different NUMA (non-uniform memory access) nodes. Recently emerged memory disaggregation is striving to unify the management of these disaggregated memory resources, but the significant differences among various communication protocols, such as TCP/IP, RDMA, PCIe, and QPI, complicate this endeavor. The CXL, adding coherency and memory semantics on top of PCIe, enables CPUs to handle far memory data as easily as near memory data (but slower). This technological breakthrough significantly simplifies the management of disaggregated memory resources, enabling more efficient unified management of near memory, extended memory, pooled memory, and shared memory in DBMS. Moreover, it opens up new opportunities for constructing

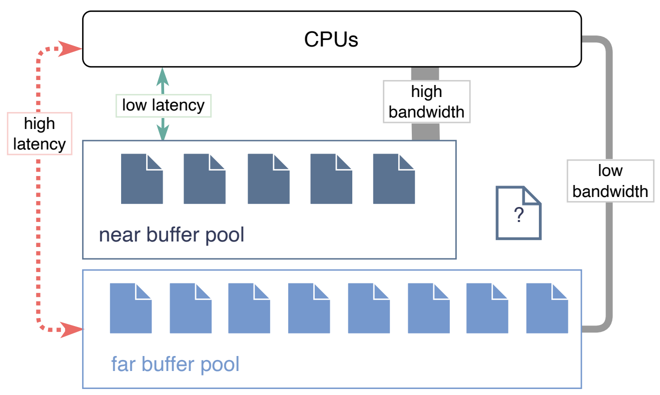

Fig. 3. Example of the bandwidth and latency characteristics between the Near Buffer Pool and the Far Buffer Pool on the host, as well as the direction of data flow. The diagram describes how CPUs interact with these two buffer pools through channels with different bandwidth (high/low bandwidth), latency (low/high latency), and capacity (small/large capacity) characteristics. This setup highlights the importance of optimizing data allocation based on varying performance requirements, within the constraints of system performance needs and resource limitations.

large-capacity memory pools and achieving efficient buffer pool data access within DBMS. Fig. 2 demonstrates the complex structure of building a hybrid buffer pool in DBMS using various types of memory, which necessitates the design of advanced data scheduling and management techniques to support efficient query processing.

A. Challenge #1: Hybrid Buffer Pool Construction.

As shown in Fig. 1, constructing a buffer pool necessitates considering various memory layers. Given the varied access characteristics (capacity, latency, bandwidth) of these memory layers, we propose constructing three distinct types of buffer pool components: (1) *Near buffer pool*, comprising the near memory of NUMA nodes; (2) *Far buffer pool*, consisting of extended memory from CXL memory devices and pooled memory allocated by CXL switches; and (3) *Shared buffer pool*, established on shared memory managed by CXL switches. These three components collectively form the hybrid buffer pool, whose logical structure is illustrated in Fig. 2.

Traditional buffer pool management techniques primarily focus on maximizing hit rates, aiming to increase access to local memory as much as possible, thereby minimizing high-latency access to local disks and remote memory in distributed DBMS. However, uniformly managing diverse memory types within a hybrid buffer pool, and applying a one-size-fits-all strategy, does not adequately balance access latency, capacity, and bandwidth to achieve optimal system performance, nor does it fully leverage the advantages of each memory type.

For instance, although extended memory boasts greater bandwidth, its access latency, considerably slower compared to near memory, may become a bottleneck when used in conjunction with near memory. Likewise, pooled memory controlled by CXL switches, while having a larger capacity, necessitates sharing bandwidth with other hosts, potentially affecting performance. Therefore, also as shown in Fig. 3, the crucial challenge in managing hybrid buffer pools lies in effectively allocating data to exploit the full potential of each memory type.

In addition, a shared buffer pool offers the potential for further compute-memory-storage disaggregation in cloud-native databases, especially in supporting efficient transaction processing and complex data analysis. While the existing compute-storage disaggregation cloud-native databases only decouple computing and storage nodes, but cannot decouple the memory, thus failing to support (1) multiple writes for multiple masters and (2) efficient shuffling for complicated operators, e.g. sorting, grouping, joining. Fortunately, CXL provides an opportunity to decouple the shared memory to make further disaggregation. However, the big challenge is to design a transparent shared memory layer to support efficient transactions and analytics, i.e., the user can be unaware of near memory and far memory.

In order to properly allocate data pages, transaction logs, index pages, and temporary tables in the hybrid buffer pool, the challenge lies in assessing the advantages and disadvantages of storing relational data, materialized views, or other data types in a specific type of buffer pool. This evaluation task involves considering various factors, including but not limited to data access frequency, read/write concurrency, data update frequency, as well as the size and life cycle of the data. Research into efficient, high-quality management techniques for predicting, assessing, and allocating data in hybrid buffer pools, especially in scenarios involving large volumes of data and high concurrency, will have a significant and profound impact on enhancing the efficiency and stability of database management systems.

B. Challenge #2: Dynamic Data Page Allocation.

Moving critical data pages closer to the corresponding processors is crucial, particularly in scenarios with dynamic workloads and data access patterns in high-concurrency queries. Consequently, dynamically allocating a data page to the most appropriate one of the three types of buffer pools in a hybrid buffer pool, during various stages of query execution, has emerged as a key area of research interest. The advantages of CXL, particularly its low latency and high bandwidth, enable more frequent movement of data pages among the three distinct types of buffer pools. This increased flexibility significantly enhances the capability to improve the DBMS performance across a range of workloads.

However, the frequent movement of data pages introduces additional challenges in maintaining an up-to-date page directory. To ensure the accurate and rapid location of data within the hybrid buffer pool, the page directory must be continuously updated in real-time to reflect the latest position of each page (Fig. 4). Therefore, in the management of hybrid buffer pools based on CXL protocols, a page directory system is required that is not only efficient but also highly adaptable to frequent data movements, addressing the dynamic data page allocation.

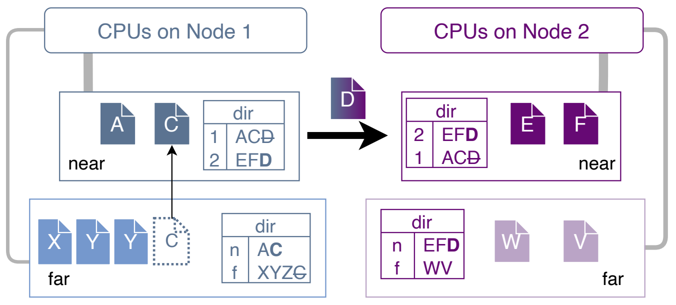

Fig. 4. Illustration of dynamic data page allocation within a hybrid buffer pool, highlighting the necessity of real-time updates across multiple page directories. The process of transferring data page D to Node 2's near buffer pool for optimized query access underscores the importance of simultaneous directory updates across various buffer zones, ensuring precise and immediate tracking of data.

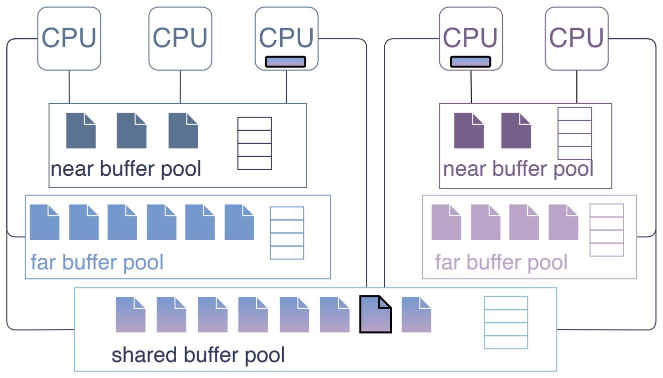

Fig. 5. Illustration of fine-grained memory sharing across multiple hosts ensuring cache consistency by CXL switch. When two hosts/nodes aim to write to the same data row within distinct transactions, DBMS can manage that row in the shared buffer pool to maintain consistency across transactions by CXL switch. This approach allows CXL switches to adeptly resolve part of conflicts arising in multi-write scenarios, thereby boosting processing efficiency.

This can be modeled as a graph partition problem, where the vertices represent tuples and an edge exists between two tuples if they are accessed by the same query. Page allocation aims to partition the tuples into different memory nodes while minimizing the number of cut edges.

Furthermore, in DBMS employing row-column hybrid storage, determining the optimal allocation of both row-stored and column-stored data pages within the hybrid buffer pool becomes an increasingly complex task. This critical decision requires consideration of various factors, including query patterns, data update frequencies, data movement, and the performance characteristics associated with maintaining consistency between row-stored and column-stored data pages. In a CXL-based hybrid buffer pool, making this decision is essential for optimizing the efficiency of HTAP (Hybrid Transactional/Analytical Processing).

In summary, effectively addressing the challenge of dynamic data page allocation within the hybrid buffer pool constructed using the CXL protocol is crucial for enhancing the query efficiency of DBMS. This involves not only strategies for moving data pages but also the efficient design and maintenance of the page directory.

C. Challenge #3: Fine-Grained Memory Sharing for Multi-Write Consistency.

In distributed DBMS, ensuring fine-grained data consistency during multi-write transaction processing presents a persistent core challenge. Despite numerous complex attempts, traditional approaches still involve a compromise in query efficiency in order to ensure data consistency. Fortunately, the innovative design of the CXL protocol offers a breakthrough. By supporting fine-grained memory sharing across host boundaries, it significantly reduces communication latency (to sub-microsecond levels) during the processing of kilobyte-scale data page updates [8]. This advancement enhances the response speed and processing capacity of the shared buffer pool in databases, offering a new solution path to this longstanding challenge.

CXL, equipped with an efficient data sharing mechanism, offers a more expedited approach to updating fine-grained data in the shared buffer pool (Fig. 5). This is crucial in reducing idling and improving overall system performance. However, despite the high efficiency of CXL in shared memory data, additional mechanisms are essential to guarantee ACID for multi-write transaction processing. These mechanisms encompass fine-grained data synchronization, conflict resolution, and the maintenance of transaction atomicity and durability. Therefore, in deploying CXL protocols, it's crucial not just to focus on enhancing the data access efficiency of the shared buffer pool but also to pay attention to adapting and upgrading critical components such as transaction logs and page locks.

CXL brings new developmental opportunities for shared memory management (disaggregated shared memory, shared storage) and avoiding distributed transaction processing (shared nothing). However, CXL also introduces new implementation challenges in shared buffer pool management, elastic memory scaling, maintaining data consistency, and ensuring transaction integrity.

IV. Elastic Memory Scaling

In traditional distributed DBMS, especially those based on NUMA architecture, there is a common tight integration of computational resources (CPUs), with memory resources (DRAM). Within the NUMA architecture, processors are usually directly connected to their respective local/near memory, creating a fixed allocation ratio of resources. Although this design optimizes memory access speeds and boosts data processing efficiency in scenarios with high concurrency, it results in the inefficient use of computational resources in memory-intensive tasks, where memory resources are fully utilized but the associated computational resources are underutilized. Such static resource coupling does not adequately tackle the imbalance in resource demands, but CXL presents an innovative solution to overcome this challenge.

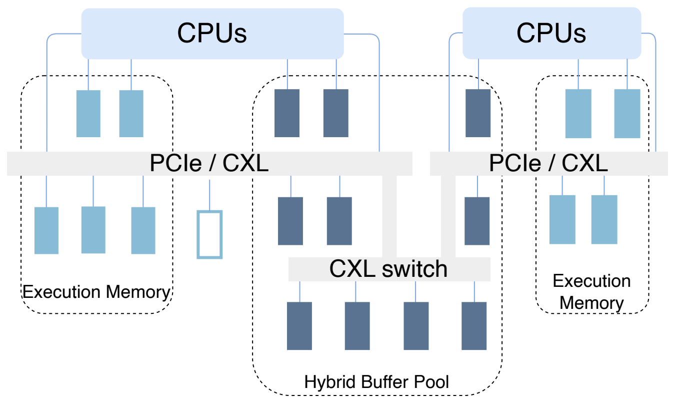

Fig. 6. A snapshot of execution memory and hybrid buffer pool during the dynamic memory allocation process. Execution memory, which demands lower access latency, primarily consists of near memory and can be expanded with extended memory as needed to support more complex processing tasks. The hybrid buffer pool encompasses various types of memory across multiple hosts, interconnected via CXL switches and PCIe/CXL connections.

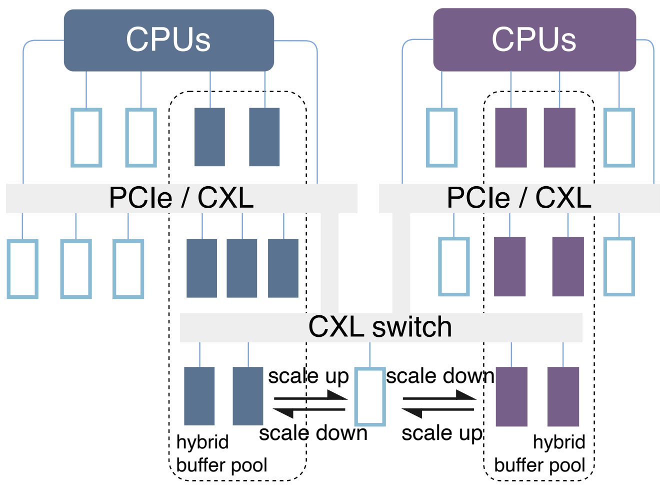

Fig. 7. Illustration of a Hybrid Buffer Pool based on the CXL protocol, demonstrating how it dynamically scales up and scales down to meet the fluctuating memory demands in HTAP and multi-modal data processing. The CXL switch plays a pivotal role in this architecture, dynamically allocating memory resources and coordinating communication between CPUs and CXL memory, thereby optimizing the efficiency of memory utilization.

CXL facilitates dynamic memory allocation, eliminating the need to reboot the host operating system or virtual management system. Consequently, this enables DBMS to flexibly allocate and reallocate memory resources across various processors and memory units, adapting to diverse workload requirements dynamically and saving CPU and memory consumption. In memory-intensive database operations such as large dataset management, high-concurrency read-write processing, and real-time data analysis, this adaptability not only enhances overall memory utilization but also alleviates performance bottlenecks. CXL brings flexibility and efficiency to the design and resource management of modern DBMS, enabling more effective handling of diverse data processing demands across various scales and types.

Fig. 6 depicts a snapshot of memory allocation during a dynamic change process. Both execution memory and the hybrid buffer pool's memory are dynamically allocated by the DBMS, optimizing the use of memory resources to enhance processing speed and overall system performance.

A. Challenge #4: Elastic Hybrid Buffer Pool.

The implementation of an elastic hybrid buffer pool presents a unique set of challenges, particularly in addressing the needs of HTAP. Notably, the memory space demand of the buffer pool fluctuates significantly during the conversion process between row-based and column-based storages. On the other hand, the particularity of prevalent multi-modal data processing also adds to the complexity of managing hybrid buffer pools. The interaction between different data types, such as vectors, graphs, and relational data, generates complex intermediate materialized tables, leading to diverse buffer pool memory requirements. Therefore, accurately predicting memory demands for a hybrid buffer pool is one of the primary challenges.

The significant and dynamically changing memory requirements across different query types render traditional static memory allocation strategies inadequate. Accurately anticipating these dynamic changes requires a deep understanding of workload characteristics and the application of advanced data analytics and machine learning techniques, which is a challenge in itself. Despite the significant advancements in dynamic memory allocation enabled by the CXL protocol, which is shown in Fig. 7, the intricacies of implementing hybrid buffer pool management in a DBMS remain notably complex.

A key challenge involves accurately predicting the latency costs tied to memory reallocation, which encompasses aspects like memory reallocation idling, data persistence, data loading, and information synchronization, among others. Such latencies directly influence the response time and overall performance of the DBMS, particularly in scenarios involving high-frequency, low-latency transaction processing where their impact is more significant. Therefore, it is crucial to ensure that any latency introduced through dynamic memory allocation is minimized in terms of its negative effect on user experience or maintained within tolerable limits. Therefore, while CXL offers opportunities for the flexible expansion of hybrid buffer pools, it also introduces a series of new complexities that need to be urgently addressed.

B. Challenge #5: Optimizing Near-Far Memory Allocation.

In distributed DBMS, especially when processing workloads such as HTAP and multi-model queries, coordinating the allocation strategy for execution memory and data memory (i.e., buffer pools) poses notable challenges. Opting for near memory on NUMA nodes typically reduces access latencies, whereas employing far memory through CXL protocol, despite offering increased storage capacity, may entail higher latency and bandwidth constraints. Therefore, it is important to adopt a

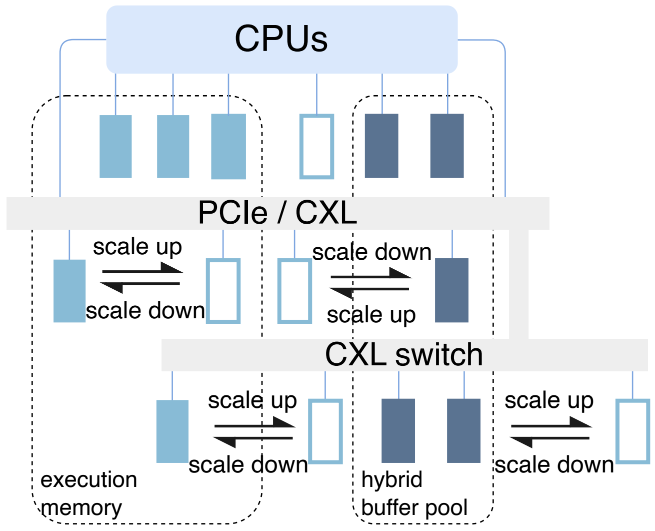

Fig. 8. Illustrates how a hybrid buffer pool supported by the CXL protocol dynamically expands and contracts to meet fluctuating memory demands. The diagram emphasizes the central role of the CXL switch in dynamic memory allocation, such as dynamically assigning memory spaces, originally not allocated to a particular host, to execution memory or the hybrid buffer pool, thereby ensuring the efficient use of memory resources.

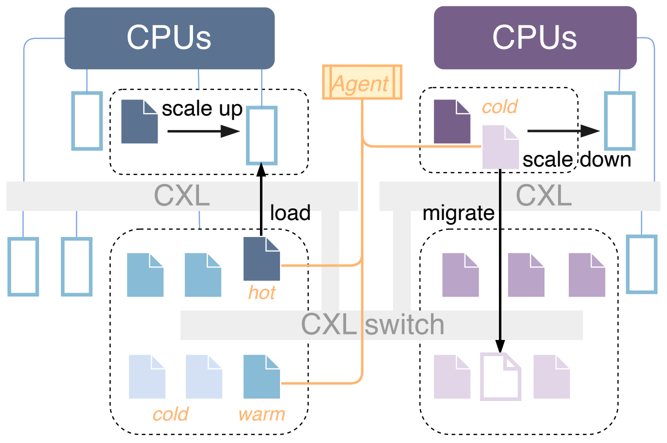

Fig. 9. Schematic of hot and cold page movements in the hybrid buffer pool during elastic scaling. The left side shows how the hottest pages from other zones (e.g., the far buffer pool) are preferentially selected for loading into the near buffer pool when scaling up; the right side illustrates the coldest pages being migrated to suitable locations, such as into the far buffer pool zone managed by the CXL switch, during scaling down. The efficiency of query processing, resulting from these decisions on page loading and migration, depends on the precision with which intelligent agents manage page temperature.

strategic approach to memory allocation, aiming to balance the advantages of near and far memory to fulfill specific workload requirements (as shown in Fig. 8).

The CXL protocol equips DBMSs with dynamic memory allocation capabilities, facilitating the expansion and sharing of far memory across various database instances in response to specific workload demands. This strategy significantly boosts system performance, notwithstanding the inherent bandwidth and latency constraints associated with far memory. During distributed elastic scaling, it becomes imperative for DBMS to strategically allocate near memory and far memory for execution memory and data memory, respectively. This careful data allocation is aimed at striking an optimal balance among performance, cost, and scalability factors, ultimately achieving a comprehensive optimum for the aggregate workload. This also enables high performance of HTAP processing.

Moreover, the utilization of the CXL protocol in distributed or disaggregated DBMS introduces a fundamental issue: the effective construction of a distributed memory access cost model that is sensitive to the latencies of near and far memory accesses. The key challenge is to devise a unified latency model that consistently represents the delay characteristics at different memory layers. Establishing such a model is not only critical for dynamic memory allocation under the CXL protocol but also lays a theoretical foundation for query optimization in distributed and disaggregated DBMS.

C. Challenge #6: Automatic Hot-Cold Data Tiering.

When designing dynamic elastic buffer pool scaling for DBMS, intelligent layering of hot and cold data becomes crucial. It not only enhances the flexibility of memory resources but also introduces a range of new challenges in data management. DBMS must be able to rapidly adapt to the dynamic adjustments of hybrid buffer pool size, making effective data layering decisions. While scaling up the hybrid buffer pool, the system needs to prioritize loading "hot" data pages into the newly added memory space. Conversely, when scaling down the hybrid buffer pool, it becomes necessary to select "cold" data pages that can be migrated or removed. This process requires DBMS to perform real-time analysis of data access patterns and frequencies to ensure efficient and precise decision-making.

Furthermore, with the elastic scaling of the hybrid buffer pool supported by CXL protocols, maintaining data locality and minimizing data access latencies becomes increasingly complex. This involves not only dealing with multi-layer memory differences but also requires an intelligent agent to continuously monitor, predict, and manage data access patterns. Fig. 9 illustrates that intelligent agents for data page tiering make critical decisions during the elastic scaling of buffer areas. Additionally, the application of machine learning models to predict data access characteristics and classify hot, cold, and shared data is key to addressing the challenges brought by distributed elastic memory allocation.

Overall, while CXL endows the hybrid buffer pool with dynamic elastic scaling capabilities, it also brings new challenges in intelligent data layering. These challenges primarily revolve around efficiently managing data within dynamically changing memory resources, maintaining data locality, and implementing precise hot and cold data layering and processing.

V. Database Recovery

In traditional distributed DBMS, fault recovery is typically a complex and time-consuming process. This process relies on transaction log rollbacks (undo) for atomicity and redos

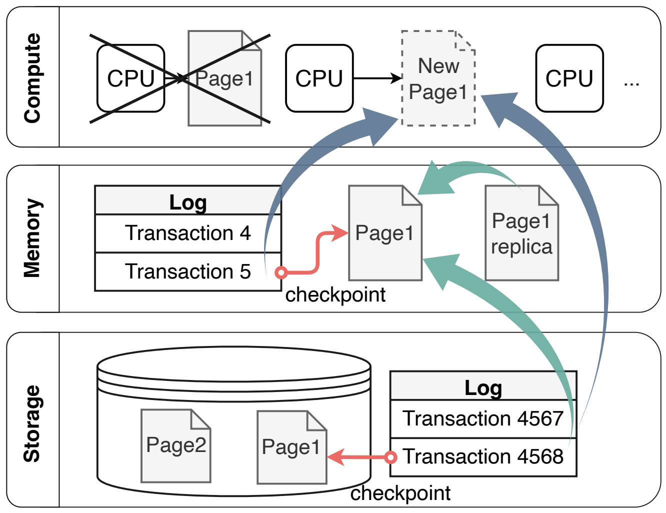

Fig. 10. Dual Checkpoint for two tiers data recovery. The architecture employs a three-layer separation of computing, memory, and storage. The memory layer consists of pooled and shared memory resources managed and communicated via CXL switches, based on the CXL protocol.

for durability, as well as data synchronization and consistency checks across multiple nodes. Particularly, recovering while maintaining data consistency across nodes often depends on slow and complex communication protocols, such as TCP/IP. However, the new features of the CXL protocols bring new opportunities for fast fault recovery in distributed DBMS. For instance, memory resources can be quickly restructured or reallocated for a failure, allowing the system to rapidly transfer critical data from affected nodes to active ones, or to use extended memory as an immediate data backup space, thereby reducing the time required to recover data from disk logs. Furthermore, the fine-grained memory sharing among multiple hosts supported by the CXL switch provides a new mechanism for rapid data recovery in distributed DBMS, where other nodes can quickly access necessary data through shared memory even when a single node fails.

A. Challenge #7: Dual Checkpoint Mechanisms.

In traditional distributed DBMS, data recovery techniques primarily focus on tightly coupled processor-memory-storage nodes, where failures are recovered from the storage by replaying the logs, leading to low recovery speed due to the high latency of disk-based log replay. To address this problem, we can disaggregate the DBMS into compute node, memory node, and storage node based on resource allocation to further improve the elasticity. The recovery of compute nodes can utilize memory checkpoints (if the memory nodes are active) or storage checkpoints, and thus the recovery speed can be significantly improved. The recovery of memory nodes can utilize their replicas or storage nodes. To achieve this, fully leveraging the efficient far memory access facilitated by CXL protocols becomes crucial for designing rapid data recovery techniques for both compute nodes and memory nodes.

The unified memory semantics and dynamic memory allocation capabilities of CXL enable the development of innovative recovery approaches. For instance, memory node data pages can be used to restore compute node data, while storage node logs assist in memory node data recovery. These processes, harnessing the low latency and high bandwidth advantages of CXL, substantially accelerate data recovery. For example, in the event of a compute node failure, data can be swiftly restored from remote memory nodes, reducing system downtime. Similarly, the fine-grained memory sharing supported by CXL enhances cross-node data recovery efficiency, allowing quick access to essential data in the event of single node failures.

However, implementing these designs effectively requires addressing the challenges posed by dual checkpoint mechanisms. As illustrated in Fig. 10, setting dual checkpoints in far memory and disk logs involves maintaining data consistency and synchronization across both levels. This adds complexity to data management and increases the requirements for data consistency. While CXL protocols enhance memory access efficiency, they must also ensure effective data exchange with disk storage, minimizing performance costs associated with data synchronization. Thus, effectively implementing a dual checkpoint mechanism, balancing recovery speed with data consistency, and utilizing the CXL protocol for flexible memory resource management and low access latency, pose significant challenges in ensuring rapid and accurate system recovery in case of failures.

B. Challenge #8: Synchronizing Dirty Pages in Shared Memory.

In traditional distributed DBMS, utilizing dirty page lists for cross-node data recovery presents a complex task, especially in high-concurrency and data-intensive query processing requirements. The low latency, high bandwidth, and large capacity features of the CXL protocol have revolutionized the maintenance of dirty page lists, making their storage in CXL extended memory feasible. This design not only reduces access latency but also enhances the frequency of updates, significantly improving the efficiency of using far memory data to recover from failures of near memory.

However, storing dirty page lists in CXL extended memory introduces new challenges, particularly considering the risk of failure in the extended memory itself. To address this risk, pooled memory under the CXL protocol becomes an ideal choice for backing up dirty page lists. This multi-tiered backup strategy ensures that even in the event of extended memory failure, dirty page information can be rapidly recovered from pooled memory, guaranteeing continuity and efficiency in data recovery (as shown in Fig. 11). Overall, the CXL protocol offers new possibilities for efficient fault recovery based on far memory dirty page lists, while also introducing new challenges in designing and maintaining these lists, especially in ensuring efficiency and consistency in the recovery process.

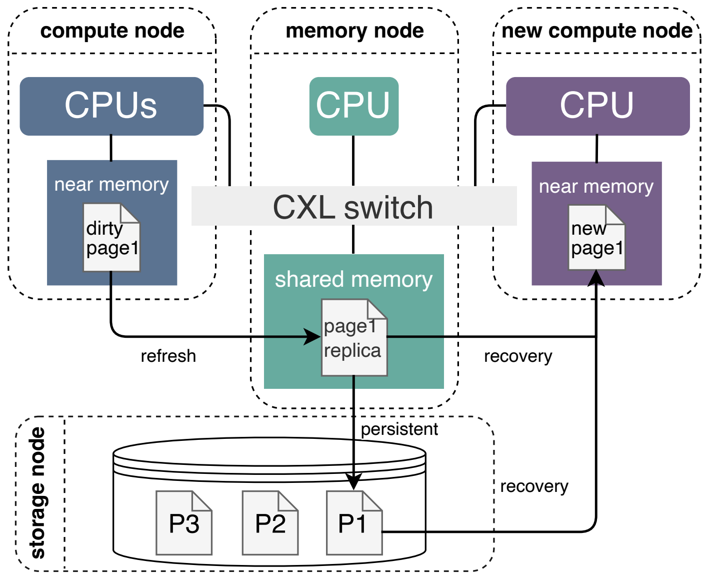

Fig. 11. Dirty page synchronization and recovery process. It illustrates the dynamic interaction between compute nodes, memory nodes, and storage nodes. Normally, dirty pages from compute nodes are frequently synchronized to memory nodes with CXL switches, and the memory nodes are responsible for persisting to storage nodes. Upon failure of a compute node, rapid data recovery for a new compute node is facilitated through efficient data synchronization and recovery from memory nodes via CXL switches, or from storage nodes when the memory node has also failed.

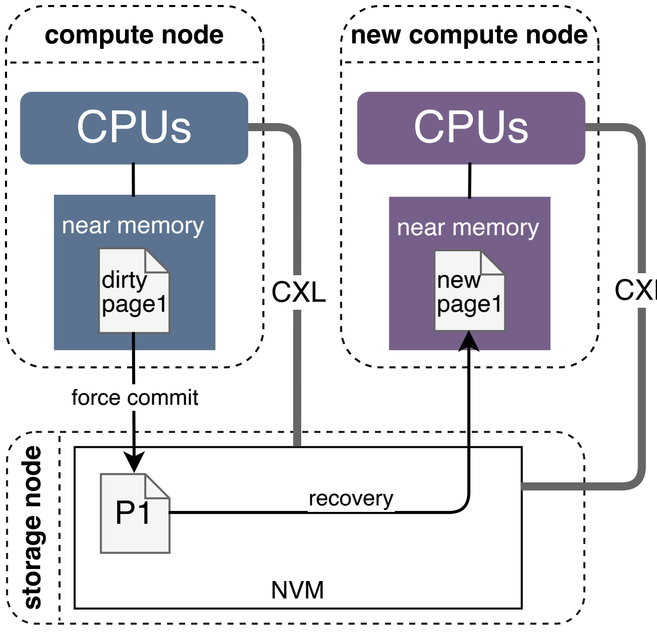

Fig. 12. Force commit with CXL protocol and persistent memory. Compute nodes synchronize dirty pages to storage nodes with persistent memory, and execute forced commit transactions via the CXL protocol. This mechanism allows for immediate persistence of data changes to NVM storage upon transaction completion, enabling rapid data recovery on new nodes in the event of system failures, while ensuring the efficiency and consistency of transaction processing.

C. Challenge #9: Force Commit with CXL and Persistent Memory.

While traditional distributed database systems commonly adopt the no-force commit strategy due to its high cost, the emergence of the CXL protocol opens new avenues for reconsidering the force-commit mechanism. According to the white paper on the CXL protocol, it can serve as an efficient transmission protocol between CPUs and Persistent Memory (PMEM). Within this technological framework, there is a potential shift towards a force-commit mechanism in database transaction processing, which involves immediate flushing of dirty pages upon transaction completion. The feasibility of this approach largely depends on the bandwidth and latency performance when accessing PMEM via the CXL protocol. The key challenge lies in assessing whether the force commit can maintain transaction processing speeds within an acceptable range for users.

The force commit mechanism based on CXL offers significant advantages. In the event of a system failure, all committed transactions can be rapidly recovered without the need for traditional redo operations, greatly enhancing the efficiency of data recovery and the overall reliability and stability of the system. However, the implementation of this mechanism requires a careful balance of the performance characteristics of the CXL protocol to ensure that the speed of transaction processing aligns with the demands of practical applications. Moreover, the redo logs are widely used between the master-standby mechanism and the cross-region disaster recovery mechanism. Thus, it is challenging if the redo logs are removed from the DBMS.

VI. Index Design

Long-standing latency challenges in distributed DBMS are being readdressed with the advent of the CXL protocols. These protocols, offering large capacity, low latency, and high scalability, unlock new possibilities in optimizing indexing strategies within disaggregated DBMS. Particularly, the novel features of CXL create unprecedented opportunities for accelerating index-based data access, while simultaneously introducing new challenges in index design.

Traditional single-node indexes have constraints on the limited size of the buffer pool, and thus are inefficient for a large volume of data. There are two possible ways to improve the scalability of indexes. The first partitions the data, builds a local index for each partition and constructs a global index across the partitions. However, this method involves distributed transactions and has high write latency. The second utilizes the shared memory to offer large memory space for improving index performance. Obviously, CXL provides opportunities for simplifying memory management and usage with near/far memory. However, there are several challenges in designing CXL-powered indexes.

A. Challenge #10: Memory Allocation for B+ Tree Nodes.

In the process of designing memory allocation strategies for B+ trees, it's essential to thoroughly analyze the relationship between the basic structure of B+ trees and their data access patterns. B+ trees consist of high-level non-leaf nodes and data-storing leaf nodes, with data within the leaf nodes being orderly stored. This structural design facilitates efficient data localization during index scans, allowing for swift top-down data location. During such operations, non-leaf nodes are accessed more frequently than leaf nodes, hence placing these frequently accessed nodes in near memory significantly reduces the need for far memory access, thereby enhancing the efficiency of frequent data queries. However, data operations extend beyond scans to include insertions and deletions, complicating the memory allocation strategy. Particularly during insertions, the access pattern inversely shifts, significantly increasing the frequency of certain leaf nodes being accessed. This indicates that memory placement strategies optimized for search operations are not suitable for insertions or deletions due to their differing node access requirements.

Therefore, formulating a memory allocation strategy for B+ tree nodes requires considering the ratio of different types of data operations within the workload, as well as the sequence of these operations. Additionally, it necessitates fully leveraging the bandwidth, latency, and capacity features of the CXL protocol to formulate the memory allocation strategy, including the cost of allocation and its potential benefits estimation. Such comprehensive consideration aims to achieve an optimal balance amid various data operations, avoiding frequent far memory access, and ultimately optimizing overall performance, presenting a significant challenge.

B. Challenge #11: Dynamic Memory Allocation for Data Modifications.

The dynamic memory allocation capability enhanced by CXL-powered DBMS opens up new possibilities for dynamically allocating memory space for data insertions and deletions. By reserving sufficient memory space for newly inserted data and promptly recycling memory space from deleted data, system memory usage efficiency can be significantly optimized. However, achieving this goal also faces challenges.

Firstly, DBMS must accurately predict the memory space required for each insertion or deletion operator. This involves forecasting the B+ tree nodes involved and their access frequencies to optimize the allocation between near and far memory. During this decision-making process, predicting the cost of node splitting, merging, and balancing mechanisms is crucial, as different mechanisms have varied costs associated with accessing different memory layers. Although the CXL switch offers a relatively lightweight dynamic memory allocation solution, the cost of memory scaling must still be carefully weighed.

Secondly, it is critical to predict and adapt to the trend of data modifications, enabling the DBMS to dynamically allocate memory based on real-time query patterns and upcoming data access frequencies. This strategy not only addresses the current workload but also anticipates future access patterns, ensuring that frequently accessed nodes remain in near memory to reduce latency, while less accessed nodes are allocated to far memory to utilize its larger capacity. This requires a finely balanced approach, taking into account the bandwidth and latency of the CXL protocol and the costs associated with dynamic memory scaling to optimize overall database performance.

By overcoming these challenges and effectively leveraging the capabilities of the CXL protocol, DBMS can support highly dynamic data workloads with unprecedented efficiency and flexibility, meeting the evolving needs of data-intensive applications. This significantly enhances the performance and scalability of index-based data management in DBMS.

C. Challenge #12: Enhancing Index Concurrency for Structure Updates.

Index, supporting concurrent reads, further enhances data access efficiency. However, the complex index structure updates, which include the writing processes involved in B+ tree node splitting, merging, and balancing, pose significant challenges to maintaining index consistency in high-concurrency data modification scenarios. Traditional DBMSs maintain index consistency and accuracy by minimizing lock contention. Optimistic Concurrency Control (OCC) technology, which assumes conflicts are rare and checks for violations at transaction commit, represents a promising approach. Nonetheless, effectively applying OCC to index structures requires careful design to minimize the cost of false positives and efficiently manage rollback mechanisms. Traditional locking mechanisms can become bottlenecks, severely hindering the performance and scalability of database systems.

The CXL switch, by offering shared memory and cache coherence, paves new pathways for addressing concurrent index update challenges. While the CXL switch can effortlessly manage conflicts in shared CXL memory data under multi-write scenarios through hardware and CPU-memory communication mechanisms, the primary challenge lies in fully leveraging this advantage to develop an efficient and reliable index update strategy that includes the splitting, merging, and balancing of B+ tree nodes. Additionally, the adaptability of the index structure to the dynamic allocation capabilities of CXL memory introduces further complexity. Indices need to efficiently scale up, scale down, and re-balance in response to variations in data sets and workloads, all while supporting high concurrency. This demands sophisticated algorithms to dynamically adjust the index structure without compromising the overall database performance.

VII. Related Work

CXL for Database Systems. Recent research demonstrated the performance of utilizing CXL-based extended memory in In-Memory Database Management Systems (IMDBMS) [21] [15]. The first study [21] evaluated CXL memory with OLTP and OLAP workloads, finding that reasonable data allocation can mitigate the throughput impact. The second study [15] examined CXL memory pooling's potential limitations, observing that initial CXL versions had bandwidth issues for OLAP workloads. A recent tutorial [10] has concurred that CXL, as an interconnect technology surpassing RDMA in speed, offers two principal methodologies for utilization within IMDBMS. The first approach considers memory connected via CXL as part of a unified memory space with the host's local memory, while the second distinctively separates CXL memory from local memory, enabling explicit data management between the two to optimize performance. However, detailed perspectives on how CXL might transform the architectures of various disaggregated database systems remain under-explored. Researchers not only tested the performance enhancement of IMDBS through the use of CXL-based extended memory, but also demonstrated its application in data analytics platforms, e.g., Apache Spark [6]. By employing CXL extended memory for storing intermediate data in Spark Shuffle, it effectively reduced the latency of TeraSort operations. The DB community's attempts at utilizing CXL protocol represent a promising start. Thus it requires further in-depth study on how to utilize CXL to optimize database systems.

Difference between CXL and RDMA. Remote Direct Memory Access (RDMA) protocol enables direct data transfer from the memory of one host to another without CPU intervention, enhancing the efficiency of remote memory access. RDMA has supported the development of shared memory and compute-memory disaggregation architectures in distributed DBMS. However, when comparing RDMA with CXL, there are significant differences in bandwidth, latency, capacity, data access granularity, data consistency, and shared memory management.

RDMA offers larger access granularity, whereas CXL provides fine-grained data access. Thus, data organization and algorithms based on the RDMA protocol are not entirely suitable for memory management based on the CXL protocol and need to be redesigned. For example, data organization could shift from page-based to more fine-grained (a batch of) tuple-based granularities. Moreover, differences in bandwidth and latency imply that the types of workloads suitable for RDMA and CXL might differ, necessitating optimized execution strategies for the specific technology.

Most importantly, the cache coherency provided by the CXL protocol, along with future support for shared memory cache consistency, brings new opportunities for DBMS. This cache coherency feature, absent in RDMA, means the performance sacrifices made to ensure data consistency with RDMA protocol can be avoided in architectures based on the CXL protocol. Therefore, DBMS architectures, query optimization, and transaction processing based on the CXL protocol need to be redesigned to fully leverage its advanced features.

Memory Disaggregation. With the emergence of RDMA, researchers highlighted the necessity of redesigning the architecture of distributed DBMS to fully utilize high-performance networks [22]. The rise of disaggregated data centers prompts a shift towards disaggregated database designs [23], [24]. These advancements presented significant challenges [11]. In response, PolarDB Serverless [25], [26] was developed to enable each resource pool to independently adjust its size, introducing key optimizations like optimistic locking and index-aware prefetching to enhance performance and scalability. Farview [27] is implemented through FPGA-based smart NICs to support offloading queries by using operators such as selection, projection, aggregation, regular expression matching, and encryption. Redy [28] provides high-performance caches using remote memory through automatic RDMA configuration tuning. LegoBase [16] proposed a two-tier ARIES protocol to handle failures of compute nodes and the remote memory on memory disaggregation architecture. However, the difference between RDMA and CXL requires rethinking the DBMS-specific techniques on CXL-based memory disaggregation architecture.

CXL for Memory Pooling. Pond [14] leveraged the CXL protocol to develop a memory pooling system for cloud computing platforms, achieving both rapid response requirements and a reduction in memory costs. Another work [7] attempted to build a simulated CXL-enabled DRAM-SSD hybrid memory pool, with experiments showing no significant performance degradation for compute-intensive tasks. DirectCXL [18] provided a laboratory product that connects the operating system and software/hardware modules to realize memory pooling, which is of great significance. TPP [29] proposed an OS-level lightweight mechanism to identify and place hot/cold pages to appropriate memory tiers to improve the performance. However, they do not study how these memory pooling techniques affect database systems.

Beyond Memory. Firstly, within multi-core computing architectures, the CXL protocol has the potential to fully leverage computational resources, also offering more possibilities for solving challenges such as HTAP. Architectures based on the CXL protocol can also be viewed as ”expanded NUMA”, with the evolution from NUMA-aware to CXL-aware enhancing the performance and scalability of DBMS. Moreover, the advantages of the CXL protocol play a crucial role in promoting the development of record-based DBMS (as opposed to page-based DBMS), enabling fine-grained data processing. Lastly, the combination of the CXL protocol with SSDs [30]–[32], as well as interactions between CXL protocol and GPUs [33], brings new development opportunities for DBMS tailored to new hardware.

VIII. Conclusion

We argue that CXL has high potential to revolutionize the design of DBMS. The new features, high bandwidth, low latency, and support for cache coherency and memory semantics, provide a new direction for addressing the limitations and bottlenecks faced by traditional distributed database systems. This paper provides several challenges of buffer pool management, distributed elastic memory management, fast data recovery, and distributed index design. We believe that solving these challenges is a significant first step to inspire cutting-edge approaches in future DBMS design to achieve efficiency, reliability, and lower total cost of ownership. This paper will open up new research directions for CXL-powered database systems.

References

[1] Cxl consortium. introduction to compute express link. [Online]. Available: https://computeexpresslink.org/wp-content/uploads/2023/12/0c1418_d9878707bbb7427786b70c3c91d5fbd1.pdf

[2] Cxl consortium. compute express link 2.0 white paper. [Online]. Available: https://computeexpresslink.org/wp-content/uploads/2023/12/CXL2.0_White_Paper_November-2020_FINAL.pdf

[3] Cxl consortium. cxl 3.0 specification. [Online]. Available: https://computeexpresslink.org/wp-content/uploads/2023/12/CXL_3.0_white-paper_FINAL.pdf

[4] Cxl consortium. cxl 3.1 specification. [Online]. Available: https://computeexpresslink.org/wp-content/uploads/2023/12/CXL_3.1-White-Paper_FINAL.pdf

[5] M. Arif, K. Assogba, M. M. Rafique, and S. Vazhkudai, "Exploiting cxl-based memory for distributed deep learning," in *ICPP*. ACM, 2022, pp. 19:1–19:11. [Online]. Available: https://doi.org/10.1145/3545008.3545054

[6] S. Ryu, S. Kim, J. Jun, D. Moon, K. Lee, J. Choi, S. Kim, H. Kim, L. Kim, W. H. Choi, M. Nam, D. Hwang, H. Roh, and Y. Joo, "System optimization of data analytics platforms using compute express link (CXL) memory," in *BigComp*. IEEE, 2023, pp. 9–12. [Online]. Available: https://doi.org/10.1109/BigComp57234.2023.00011

[7] Q. Yang, R. Jin, B. Davis, D. Inupakutika, and M. Zhao, "Performance evaluation on cxl-enabled hybrid memory pool," in *NAS*. IEEE, 2022, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NAS55553.2022.9925356

[8] D. D. Sharma, R. Blankenship, and D. S. Berger, "An introduction to the compute express link (CXL) interconnect," *CoRR*, vol. abs/2306.11227, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2306.11227

[9] A. Geyer, D. Ritter, D. H. Lee, M. Ahn, J. Pietrzyk, A. Krause, D. Habich, and W. Lehner, "Working with disaggregated systems. what are the challenges and opportunities of RDMA and cxl?" in *BTW*, ser. LNI, vol. P-331. Gesellschaft für Informatik e.V., 2023, pp. 751–755. [Online]. Available: https://doi.org/10.18420/BTW2023-47

[10] J. Wang and Q. Zhang, "Disaggregated database systems," in *SIGMOD*. ACM, 2023, pp. 37–44. [Online]. Available: https://doi.org/10.1145/3555041.3589403

[11] R. Wang, J. Wang, S. Idreos, M. T. Özsu, and W. G. Aref, "The case for distributed shared-memory databases with rdma-enabled memory disaggregation," *Proc. VLDB Endow.*, vol. 16, no. 1, pp. 15–22, 2022. [Online]. Available: https://www.vldb.org/pvldb/vol16/p15-wang.pdf

[12] D. Boles, D. G. Waddington, and D. A. Roberts, "Cxl-enabled enhanced memory functions," *IEEE Micro*, vol. 43, no. 2, pp. 58–65, 2023. [Online]. Available: https://doi.org/10.1109/MM.2022.3229627

[13] D. Gouk, M. Kwon, H. Bae, S. Lee, and M. Jung, "Memory pooling with CXL," *IEEE Micro*, vol. 43, no. 2, pp. 48–57, 2023. [Online]. Available: https://doi.org/10.1109/MM.2023.3237491

[14] H. Li, D. S. Berger, L. Hsu, D. Ernst, P. Zardoshti, S. Novakovic, M. Shah, S. Rajadnya, S. Lee, I. Agarwal, M. D. Hill, M. Fontoura, and R. Bianchini, "Pond: Cxl-based memory pooling systems for cloud platforms," in *ASPLOS*. ACM, 2023, pp. 574–587. [Online]. Available: https://doi.org/10.1145/3575693.3578835

[15] D. Lee, T. Willhalm, M. Ahn, S. M. Desai, D. Booss, N. Singh, D. Ritter, J. Kim, and O. Rebholz, "Elastic use of far memory for in-memory database management systems," in *SIGMOD, DaMoN 2023*. ACM, 2023, pp. 35–43. [Online]. Available: https://doi.org/10.1145/3592980.3595311

[16] Y. Zhang, C. Ruan, C. Li, J. Yang, W. Cao, F. Li, B. Wang, J. Fang, Y. Wang, J. Huo, and C. Bi, "Towards cost-effective and elastic cloud database deployment via memory disaggregation," *Proc. VLDB Endow.*, vol. 14, no. 10, pp. 1900–1912, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1900-zhang.pdf

[17] R. Wang, J. Wang, P. Kadam, M. T. Özsu, and W. G. Aref, "dlsm: An lsm-based index for memory disaggregation," in *ICDE*. IEEE, 2023, pp. 2835–2849. [Online]. Available: https://doi.org/10.1109/ICDE55515.2023.00217

[18] D. Gouk, S. Lee, M. Kwon, and M. Jung, "Direct access, high-performance memory disaggregation with directcxl," in *ATC*. USENIX Association, 2022, pp. 287–294. [Online]. Available: https://www.usenix.org/conference/atc22/presentation/gouk

[19] Cxl 2.0 switch from xconn. [Online]. Available: https://www.youtube.com/watch?v=mUMo5fReiTk&t=127s

[20] Pooled cxl memory system at fms. [Online]. Available: https://tinyurl.com/yc7ycyvu

[21] M. Ahn, A. Chang, D. Lee, J. Gim, J. Kim, J. Jung, O. Rebholz, V. Pham, K. T. Malladi, and Y. Ki, "Enabling CXL memory expansion for in-memory database management systems," in *SIGMOD, DaMoN 2022*. ACM, 2022, pp. 8:1–8:5. [Online]. Available: https://doi.org/10.1145/3533737.3535090

[22] C. Binnig, A. Crotty, A. Galakatos, T. Kraska, and E. Zamanian, "The end of slow networks: It's time for a redesign," *Proc. VLDB Endow.*, vol. 9, no. 7, pp. 528–539, 2016. [Online]. Available: http://www.vldb.org/pvldb/vol9/p528-binnig.pdf

[23] Q. Zhang, Y. Cai, S. Angel, V. Liu, A. Chen, and B. T. Loo, "Rethinking data management systems for disaggregated data centers," in *10th Conference on Innovative Data Systems Research, CIDR 2020, Amsterdam, The Netherlands, January 12-15, 2020, Online Proceedings*. www.cidrdb.org, 2020. [Online]. Available: http://cidrdb.org/cidr2020/papers/p6-zhang-cidr20.pdf

[24] Q. Zhang, Y. Cai, X. Chen, S. Angel, A. Chen, V. Liu, and B. T. Loo, "Understanding the effect of data center resource disaggregation on production dbmss," *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1568–1581, 2020. [Online]. Available: http://www.vldb.org/pvldb/vol13/p1568-zhang.pdf

[25] W. Cao, Y. Zhang, X. Yang, F. Li, S. Wang, Q. Hu, X. Cheng, Z. Chen, Z. Liu, J. Fang, B. Wang, Y. Wang, H. Sun, Z. Yang, Z. Cheng, S. Chen, J. Wu, W. Hu, J. Zhao, Y. Gao, S. Cai, Y. Zhang, and J. Tong, "Polardb serverless: A cloud native database for disaggregated data centers," in *SIGMOD '21, Virtual Event, China, June 20-25, 2021*. ACM, 2021, pp. 2477–2489. [Online]. Available: https://doi.org/10.1145/3448016.3457560

[26] Y. Zhang, C. Ruan, C. Li, J. Yang, W. Cao, F. Li, B. Wang, J. Fang, Y. Wang, J. Huo, and C. Bi, "Towards cost-effective and elastic cloud database deployment via memory disaggregation," *Proc. VLDB Endow.*, vol. 14, no. 10, pp. 1900–1912, 2021. [Online]. Available: http://www.vldb.org/pvldb/vol14/p1900-zhang.pdf

[27] D. Korolija, D. Koutsoukos, K. Keeton, K. Taranov, D. S. Milojicic, and G. Alonso, "Farview: Disaggregated memory with operator off-loading for database engines," in *12th Conference on Innovative Data Systems Research, CIDR 2022, Chaminade, CA, USA, January 9-12, 2022*. www.cidrdb.org, 2022. [Online]. Available: https://www.cidrdb.org/cidr2022/papers/p11-korolija.pdf

[28] Q. Zhang, P. A. Bernstein, D. S. Berger, and B. Chandramouli, "Redy: Remote dynamic memory cache," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 766–779, 2021. [Online]. Available: https://www.vldb.org/pvldb/vol15/p766-zhang.pdf

[29] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. O. Kanaujia, and P. Chauhan, "TPP: transparent page placement for cxl-enabled tiered-memory," in *ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023*. ACM, 2023, pp. 742–755. [Online]. Available: https://doi.org/10.1145/3582016.3582063

[30] S. Lee, A. Lerner, P. Bonnet, and P. Cudré-Mauroux, "Database kernels: Seamless integration of database systems and fast storage via cxl," in *14th Conference on Innovative Data Systems Research, CIDR 2024, Chaminade, USA, January 14-17, 2024*. www.cidrdb.org, 2024. [Online]. Available: https://www.cidrdb.org/cidr2024/papers/p43-lee.pdf

[31] S. Yang, M. Kim, S. Nam, J. Park, J. Choi, E. H. Nam, E. Lee, S. Lee, and B. S. Kim, "Overcoming the memory wall with cxl-enabled ssds," in *2023 USENIX Annual Technical Conference, USENIX ATC 2023, Boston, MA, USA, July 10-12, 2023*. USENIX Association, 2023, pp. 601–617. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/yang-shao-peng

[32] M. Kwon, S. Lee, and M. Jung, "Cache in hand: Expander-driven CXL prefetcher for next generation CXL-SSD," in *Proceedings of the 15th ACM/USENIX Workshop on Hot Topics in Storage and File Systems, HotStorage 2023, Boston, MA, USA, 9 July 2023*. ACM, 2023, pp. 24–30. [Online]. Available: https://doi.org/10.1145/3599691.3603406

[33] S. Sano, Y. Bando, K. Hiwada, H. Kajihara, T. Suzuki, Y. Nakanishi, D. Taki, A. Kaneko, and T. Shiozawa, "GPU graph processing on cxl-based microsecond-latency external memory," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, SC-W 2023, Denver, CO, USA, November 12-17, 2023*. ACM, 2023, pp. 961–972. [Online]. Available: https://doi.org/10.1145/3624062.3624173