Disaggregated Data Systems – State-of-the-Art and Open Challenges

Alexander Krause
alexander.krause@tu-dresden.de
Technische Universität Dresden
Dresden, Germany

Johannes Pietrzyk
johannes.pietrzyk@tu-dresden.de
Technische Universität Dresden
Dresden, Germany

Alexander Boehm
alexander.boehm@sap.com
SAP SE
Walldorf, Germany

Abstract

This tutorial aims to review disaggregated systems research in the context of database systems. We cover the exploitation of disaggregated storage in the industry landscape, contemporary research aimed towards database applications, and the current trend of data path computing. Our journey through disaggregated systems is concluded with a statement about open challenges in all three categories.

Keywords

Disaggregated Systems, Disaggregated Memory, CXL, RDMA, Near Data Processing

1 Motivation and Relevance

Disaggregated systems promise to break the fixed resource ratios of server form factors and expose compute, storage, and memory as independently and thus elastically scalable building blocks. While not yet fully implemented, this vision is already visible in production cloud deployments: durable storage is routinely provisioned and managed separately from compute, while traditionally "local" resources (notably DRAM and NVMe SSDs) are increasingly targeted for pooling and remote access. For data management systems, the consequence is an even stronger emphasis of managing data movement and control-path overhead—bytes on the fabric, CPU cycles in protocol/I/O stacks, and extra round-trips on critical access paths.

Disaggregation opens novel operating points for data systems - or more generally - data-intensive platforms: elastic scaling and rapid reconfiguration, reduced resource stranding through better demand matching, and new recovery/migration strategies enabled by decoupling state from specific machines. Yet, nothing comes for free: remote access amplifies latency sensitivity, stresses caching and concurrency control, and forces explicit placement decisions: what stays close to compute, what can reside in a remote tier, and which parts of the processing pipeline should move toward the data.

Tutorial Scope: This tutorial is motivated by how recent research connects these infrastructure capabilities to data system-specific design questions. RDMA-style *Split* architectures and CXL-style *Pool* architectures provide different performance envelopes for memory disaggregation, and their implications propagate upward from OS-level mechanisms through buffer management, access paths, and query execution. In parallel, a renewed push toward *data-path computing* (near-storage and near-network processing) aims to mitigate the "disaggregation tax" by reducing the number of transferred bytes and host-side overhead. Our goal is to provide a structured overview of this evolving design space, summarize state-of-the-art approaches, and highlight the open challenges that must be addressed to make disaggregation a foundation for future data systems. To achieve that, our tutorial is divided into three parts: (i) a brief review of disaggregated systems history in the industry landscape, (ii) contemporary research, and (iii) current research directions.

Intended Audience: This tutorial is aimed at researchers and practitioners alike, and at anyone interested in working with hardware disaggregation. This tutorial should help (i) to make disaggregated system research accessible to a wider audience and (ii) to understand the current rise of and hype around CXL and memory sharing.

2 Disaggregated Infrastructure

In the first part of our tutorial, we give a brief historical perspective on the evolution of the hardware ecosystem. With the advent of cloud computing, infrastructure deployments are increasingly moving away from static, on-premise setups and towards infrastructure-as-a-service (IAAS) offerings in the cloud. A key characteristic of cloud-based offerings is the *separation of compute and storage*. This means that customers pick a certain shape of virtual machine, which defines the CPUs and memory resources available. Durable storage is added separately, e.g., in the form of block devices that are mounted into the VMs, or by leveraging object storage offering large storage volumes with GET/PUT semantics at a comparatively low price. The separation of compute and storage virtually allows arbitrary combinations of CPU/DRAM and disk space, e.g., it is possible to create a small, virtual server with only one virtual CPU (vCPU), but to attach terabytes of persistent storage to it.

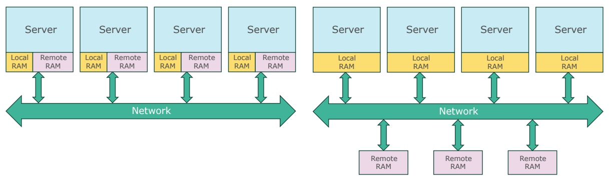
2.1 Implications for Data Systems

For many years, the database and distributed systems communities have researched the novel opportunities of disaggregated infrastructure. This effort has resulted in both novel system architectures and proposals for the incremental evolution of existing designs. Using some representative examples, we discuss three exemplary classes of data management designs and how they benefit from large-scale, disaggregated infrastructure.

Novel data management systems such as Map/Reduce [6], BigQuery [21, 22] and others [5, 7, 25, 33]. We highlight key elements of their designs and discuss the differences from traditional database systems, as well as their reception by the database community [8]. Recently, the idea of *serverless computing* and *serverless data processing platforms* gained attention. These solutions focus on offering a Software-as-a-Service (SaaS) data management solution to customers and hide the existence of underlying virtual machines and their shapes from the customer. Thus, we consider both industry and academia systems, such as AWS Athena [30], Lambda [23] or Skyrise [4].

Another popular approach for building cloud-native, disaggregated systems is the incremental evolution of existing data system architectures. One of the pioneers in this space is the

10.48786/edbt.2026.77



(a) Split (b) Pool

Figure 1: High-level system architectures for memory disaggregation (cf. Figure 2 of [9])

Aurora system [27, 28], which introduced the idea of componentizing the MySQL and PostgreSQL systems by separating their query processing frontends from their storage backends. A similar approach is found in several other industry systems, such as AlloyDB [24] or Socrates [1]. We highlight the key design aspects of these *evolved, traditional DBMS* and discuss their key advantages and challenges in these architectures.

2.2 Remaining Challenges

While the separation of compute and storage is a key building block to engineer the next generation data systems architectures, still, the availability and capacity of several other important hardware resources are tied to the virtual machine. This includes both local solid-state drives (SSDs) that are key for low-latency, persistent data storage, as well as main memory (DRAM).

The challenge with these local resources is that they are usually not fully utilized, as the available ratio in the server typically does not match the expected ratio needed by consuming applications (i.e., a in-memory database management system has higher DRAM requirements than a web server). This leads to the problem of *resource stranding*, either on the customer side, or for the cloud service provider.

Particularly, DRAM stranding is a very relevant problem due to the high DRAM price, and we highlight several different proposals how to address DRAM stranding proposed in the literature [17, 18]. For local SSDs, new network-based access protocols such as NVMe-over-TCP might be a way to also enable their disaggregation, as long as the consumers can tolerate the additional latency induced by remote communication.

3 Disambiguating Disaggregation

Memory disaggregation is sometimes confused with storage disaggregation. Both approaches separate hardware resources from the host machine on which they are ultimately used. Following up on the discussion about storage disaggregation from the first part of the tutorial, we will introduce the concept of disaggregated memory in the second part. Moving from the constraint of physically attached resources towards rack-organized hardware components is a key enabler for true software-defined systems, also known as disaggregated systems.

Memory disaggregation can be achieved in several ways, but typically through a network interconnect such as Ethernet or InfiniBand. A recent survey drafts disaggregated memory architecture as shown in Figure 1. Main memory can be shared among multiple machines, where the *Split* architecture allows servers to leverage Remote Direct Memory Access (RDMA) to directly access the DRAM that is located inside one discrete other server. Contrary, Compute Express Link (CXL) enables the attachment of byte-addressable main memory via PCIe, following the *Pool* architectural approach. The most recent CXL specification version

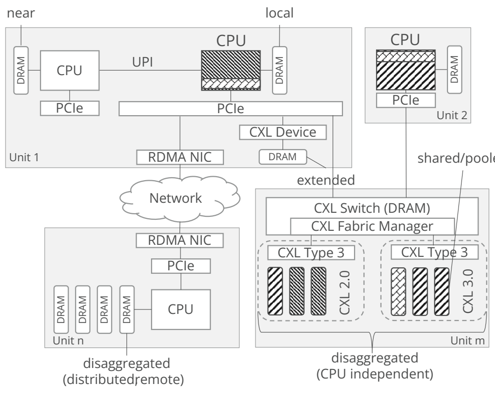


Figure 2: CXL-enabled memory disaggregation

4 was published in 2025, but until now, only CXL specification 2.0 conforming hardware is commercially available. With the CXL 2.0 specification, memory pools can be shared via a dedicated CXL switch, which enables main memory access over fabric. Figure 2 illustrates a disaggregated setup with three units: Unit 1 connects to Unit n through RDMA and to (memory-)Unit m via CXL. Once CXL 3.0 devices are generally available, multiple CPUs can even share the same DIMM.

3.1 Working up the Database Stack

This second part of the tutorial will present the general overview of a data system's software stack and with it, we will highlight memory disaggregation research throughout its layers. We first sketch operating system level techniques with Infiniswap [12] and Software-defined far memory [14]. Such techniques enable systems to use far memory as an extension for the local swap memory or to move cold, stale data to external memory.

We cover different research directions across the software stack, spanning from the buffer layer through the storage and access system up to the query processing layer. A first-class issue is data movement and near-data processing since the dawn of NUMA. PolarDBCXL [32] combats the asynchronicity and integration overhead, that is inherent to RDMA, by placing the database content into the far memory. Other approaches like Pipeline Grouping [10] show that naïve application of load/store can be outperformed by intelligent read/write operations, given enough base data overlap in concurrent query execution. Naturally, we cover holistic reviews of in-memory processing on genuine CXL hardware [31].

3.2 Open Challenges

CXL is a key enabler for true hardware disaggregation, but its cost may not offset its usability. Even major players like Google do not share a harmonic vision on the cost-benefit tradeoff for CXL, ranging from it being a godsend[1,2] or straight up way too expensive to become a true savior for cloud providers [16]. The latency penalty for CXL-attached memory varies greatly between directly attaching an add-in card to the PCIe socket or coupling multiple memory expansion devices behind a CXL switch. This poses a crucial memory access and data placement optimization criterion, especially for tiered memory setups.

[1]https://www.linkedin.com/posts/laurie-kirk_prediction-2026-is-going-to-be-the-year-activity-7412947514267598848-_PEf [accessed 03-Mar-2026]
[2]https://www.linkedin.com/posts/laurie-kirk_outside-of-the-datacenter-world-no-one-realizes-activity-7413329701861154816-QVwe [accessed 03-Mar-2026]

4 Computing on the Data Path

Disaggregation shifts the dominant cost from arithmetic to data movement and control-path overhead: bytes on the fabric, CPU cycles in protocol or I/O stacks, and extra round-trips. Near-Data Processing (NDP) addresses this "disaggregation tax" by executing selected computation on the data path—near storage, inside network devices (DPUs/SmartNICs), or near memory, so that less data (or fewer expensive control actions) must traverse remote boundaries. Recent research shows a consistent pattern: the largest and reliable gains come from (i) pushing down high-data-reduction work, (ii) using low-overhead interfaces, and (iii) ensuring correctness once updates cross the host-device boundary [2, 3, 15, 19, 29, 34]. Building on the disaggregated storage and memory models discussed in the previous tutorial parts, NDP can be viewed as "data-path computing": inserting computation along remote-access paths to reduce the number of transferred bytes and/or host CPU overhead. *Storage-side NDP* ranges from operator pushdown into storage servers to computational storage devices that run restricted kernels inside SSD/FPGA/SoC controllers. *Network-side NDP* leverages DPUs on the fast path to offload copying and request handling (and sometimes co-process data-parallel primitives), reducing the CPU load for remote I/O. *Memory-side NDP* (often implicit) redesigns access paths and data structures so that remote-memory latency/bandwidth constraints do not dominate performance. Hence, the third part of the tutorial sways the focus towards the question of how to exploit compute capabilites during data movement.

4.1 NDP Placement Along the Access Paths

At the *storage side* (operator pushdown into storage servers and computational storage devices), the state of the art focuses on byte reduction first: FPGA/SoC engines execute scan-centric kernels (filter/projection/lightweight aggregation) close to storage so that only reduced results traverse the fabric, with the host orchestrating transfers via DMA-friendly, streaming interfaces [26, 29]. Moving up the stack to the storage-engine / format boundary, the key shift is from "run code near data" to "run *DBMS-aware* code near data": systems must bridge engine-specific layouts and MVCC visibility into device-friendly streams and compact coordination metadata, and explicitly manage where version/visibility work is performed to avoid random-access amplification across the host–device boundary [15, 29]. At the *query/operator layer* on compute frontends, practical designs therefore push down primarily high–data-reduction operators (filters, partial aggregates, early projections) to cut shipped bytes [19, 29].

Beyond read-mostly pipelines, recent work begins to offload *stateful modifications* under explicit transactional contracts, using cache-coherent shared locking or coordination metadata paths to preserve correctness while bounding interference with foreground work [2, 3]. For *memory-side* NDP in memory - disaggregated Split (RDMA) and Pool (CXL fabric) settings, the dominant mechanism is often structural rather than "kernel offload": index and access-path designs reshape concurrency control, caching, and validation to reduce round-trips and remote synchronization, preventing pointer-heavy traversals from turning into control-path stalls dominated by remote latency [20].

Finally, at the *network/DPU datapath* inside storage backends, DPUs can remove protocol and copying overhead (e.g., zero-copy request handling and lightweight parsing/dispatch), but recent evidence emphasizes that the real frontier is co-processing and placement: benefits depend on configuration, input characteristics, and DPU resource limits, so static offload policies can underperform without adaptive split decisions [11, 34]. Overall, these results reinforce the tutorial's framing of NDP as data-path computing: dependable gains come from reducing transferred bytes and control-path work, and the open research problem is increasingly how to place and coordinate these fragments under contention and correctness constraints rather than merely identifying offloadable kernels [11, 13].

4.2 Open Challenges

A first research direction is end-to-end, adaptive placement: optimizers and runtimes should jointly model selectivity (data reduction), device saturation, reconfiguration cost, and contention and interference under shared devices [11, 13, 20, 34]. A second is portable correctness for update-capable NDP—minimal, reusable abstractions for MVCC visibility, locking, logging/recovery, and failure handling that can span heterogeneous near-data engines and varying fabrics (RDMA versus coherent CXL-class links) without per-platform redesign [2, 3, 15, 29]. A third is reusable representations and observability: canonical data formats, layout-aware transformations, and profiling/debugging support so that split execution remains understandable and verifiable as computation migrates into devices [11, 15, 34].

5 Biography

This is a joint tutorial, held by TU Dresden and SAP SE.

Alexander Krause is a PostDoc at TU Dresden's Database Research Group, chaired by Wolfgang Lehner. He is currently participating in the Reinhart Koselleck-Project of the German Research Foundation, which focuses on serverless data management principles. His research focuses on databases in the context of disaggregated systems by leveraging RDMA and CXL and he currently serves as an active member of the PVLDB Vol. 19 and SIGMOD 2027 review boards as well as a Proceedings Chair for the EDBT/ICDT 2026.

Johannes Pietrzyk is a PostDoc with the Database Research Group at TU Dresden. He currently works on CHORYS, a Horizon Europe project on open and programmable accelerators for data-intensive applications, with a particular focus on near-data processing, asynchronous data services, and RISC-V-based system design. His research interests include data-intensive systems, database architectures, hardware/software co-design, and high-performance data processing. Within CHORYS, he works on designing and evaluating system abstractions and mechanisms to enhance the performance of modern cloud and data platforms.

Alexander Boehm is a Distinguished Engineer at SAP and one of the chief architects for the SAP HANA Cloud database management system. His specific focus is on system performance and core database topics. Additionally, he is working on the evolution of the HANA system, including novel hardware and cloud-based system deployments. Before re-joining SAP in 2024, Alexander was a Principal Engineer at Google Cloud and an Uber-Techlead for Google's AlloyDB for PostgreSQL system.

Acknowledgments

This work was partly funded by (1) by the German Research Foundation (DFG) under grant LE-1416/28-1 and (2) the European Union's Horizon research and innovation program under grant agreement no. 101189551. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Health and Digital Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Panagiotis Antonopoulos, Alex Budovski, Cristian Diaconu, Alejandro Hernandez Saenz, Jack Hu, Hanuma Kodavalla, Donald Kossmann, Sandeep Lingam, Umar Farooq Minhas, Naveen Prakash, Vijendra Purohit, Hugh Qu, Chaitanya Sreenivas Ravella, Krystyna Reisteter, Sheetal Shrotri, Dixin Tang, and Vikram Wakade. 2019. Socrates: The New SQL Server in the Cloud. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*. ACM, 1743–1756. doi:10.1145/3299869.3314047

[2] Arthur Bernhardt, Sajjad Tamimi, Florian Stock, Andreas Koch, and Ilia Petrov. 2025. Update NDP: On Offloading Modifications to Smart Storage with Transactional Guarantees in Near-Data Processing DBMS. *ACM Transactions on Database Systems* (2025). doi:10.1145/3774753

[3] Arthur Bernhardt, Sajjad Tamimi, Florian Stock, Tobias Vinçon, Andreas Koch, and Ilia Petrov. 2022. Cache-Coherent Shared Locking for Transactionally Consistent Updates in Near-Data Processing DBMS on Smart Storage. In *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*. OpenProceedings.org, 2:424–2:428. doi:10.48786/EDBT.2022.34

[4] Thomas Bodner, Daniel Ritter, Martin Boissier, and Tilmann Rabl. 2025. Skyrise: Exploiting Serverless Cloud Infrastructure for Elastic Data Processing. *Datenbank-Spektrum* 25, 1 (2025), 29–38. doi:10.1007/S13222-025-00496-7

[5] Matthias Brantner, Daniela Florescu, David A. Graf, Donald Kossmann, and Tim Kraska. 2008. Building a database on S3. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*. ACM, 251–264. doi:10.1145/1376616.1376645

[6] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*. USENIX Association, 137–150.

[7] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. 2007. Dynamo: amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*. ACM, 205–220. doi:10.1145/1294261.1294281

[8] David J. DeWitt. 2014. MapReduce: A major step backwards. https://api.semanticscholar.org/CorpusID:12492635

[9] Mohammad Ewais and Paul Chow. 2023. Disaggregated Memory in the Datacenter: A Survey. *IEEE Access* 11 (2023), 20688–20712. doi:10.1109/ACCESS.2023.3250407

[10] Andreas Geyer, Alexander Krause, Dirk Habich, and Wolfgang Lehner. 2023. Pipeline Group Optimization on Disaggregated Systems. In *13th Conference on Innovative Data Systems Research, CIDR 2023, Amsterdam, The Netherlands, January 8-11, 2023*. www.cidrdb.org.

[11] Dimitrios Giouroukis, Dwi P. A. Nugroho, Varun Pandey, Steffen Zeuch, and Volker Markl. 2025. Analyzing Near-Network Hardware Acceleration with Co-Processing on DPUs. *Proc. VLDB Endow.* 18, 13 (2025), 5689–5702.

[12] Juncheng Gu, Youngmoon Lee, Yiwen Zhang, Mosharaf Chowdhury, and Kang G. Shin. 2017. Efficient Memory Disaggregation with Infiniswap. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*. USENIX Association, 649–667.

[13] Christian Knödler, Naeem Ramzan, and Ilia Petrov. 2025. hybridNDP: Dynamic Operation Offloading and Cooperative Query Execution in Smart Storage Settings. In *Proceedings 28th International Conference on Extending Database Technology, EDBT 2025, Barcelona, Spain, March 25-28, 2025*. OpenProceedings.org, 769–782. doi:10.48786/EDBT.2025.62

[14] H. Andrés Lagar-Cavilla, Junwhan Ahn, Suleiman Souhlal, Neha Agarwal, Radoslaw Burny, Shakeel Butt, Jichuan Chang, Ashwin Chaugule, Nan Deng, Junaid Shahid, Greg Thelen, Kamil Adam Yurtsever, Yu Zhao, and Parthasarathy Ranganathan. 2019. Software-Defined Far Memory in Warehouse-Scale Computers. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS 2019, Providence, RI, USA, April 13-17, 2019*. ACM, 317–330. doi:10.1145/3297858.3304053

[15] Kitaek Lee, Insoon Jo, Jaechan Ahn, Hyuk Lee, Hwang Lee, Woong Sul, and Hyungsoo Jung. 2023. Deploying Computational Storage for HTAP DBMSs Takes More Than Just Computation Offloading. *Proc. VLDB Endow.* 16, 6 (2023), 1480–1493. doi:10.14778/3583140.3583161

[16] Philip Levis, Kun Lin, and Amy Tai. 2023. A Case Against CXL Memory Pooling. In *Proceedings of the 22nd ACM Workshop on Hot Topics in Networks, HotNets 2023, Cambridge, MA, USA, November 28-29, 2023*. ACM, 18–24. doi:10.1145/3626111.3628195

[17] Feng Li, Sudipto Das, Manoj Syamala, and Vivek R. Narasayya. 2016. Accelerating Relational Databases by Leveraging Remote Memory and RDMA. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*. ACM, 355–370. doi:10.1145/2882903.2882949

[18] Huaicheng Li, Daniel S. Berger, Lisa Hsu, Daniel Ernst, Pantea Zardoshti, Stanko Novakovic, Monish Shah, Samir Rajadnya, Scott Lee, Ishwar Agarwal, Mark D. Hill, Marcus Fontoura, and Ricardo Bianchini. 2023. Pond: CXL-Based Memory Pooling Systems for Cloud Platforms. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2023, Vancouver, BC, Canada, March 25-29, 2023*. ACM, 574–587. doi:10.1145/3575693.3578835

[19] Shu Lin, Arunprasad P. Marathe, Per-Åke Larson, Chong Chen, Calvin Sun, Paul Lee, Weidong Yu, Jianwei Li, Juncai Meng, Roulin Lin, Xiaoyang Chen, and Qingping Zhu. 2022. Near Data Processing in Taurus Database. In *38th IEEE International Conference on Data Engineering, ICDE 2022, Kuala Lumpur, Malaysia, May 9-12, 2022*. IEEE, 1662–1674. doi:10.1109/ICDE53745.2022.00170

[20] Xuchuan Luo, Pengfei Zuo, Jiacheng Shen, Jiazhen Gu, Xin Wang, Michael R. Lyu, and Yangfan Zhou. 2023. SMART: A High-Performance Adaptive Radix Tree for Disaggregated Memory. In *17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023, Boston, MA, USA, July 10-12, 2023*. USENIX Association, 553–571.

[21] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, and Theo Vassilakis. 2010. Dremel: Interactive Analysis of Web-Scale Datasets. *Proc. VLDB Endow.* 3, 1 (2010), 330–339. doi:10.14778/1920841.1920886

[22] Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, Hossein Ahmadi, Dan Delorey, Slava Min, Mosha Pasumansky, and Jeff Shute. 2020. Dremel: A Decade of Interactive SQL Analysis at Web Scale. *Proc. VLDB Endow.* 13, 12 (2020), 3461–3472. doi:10.14778/3415478.3415568

[23] Ingo Müller, Renato Marroquín, and Gustavo Alonso. 2020. Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*. ACM, 115–130. doi:10.1145/3318464.3389758

[24] Ravi Murthy and Gurmeet (GG) Goindi. [n. d.]. AlloyDB for PostgreSQL under the hood: Intelligent, database-aware storage. https://cloud.google.com/blog/products/databases/alloydb-for-postgresql-intelligent-scalable-storage. Accessed: 2026-02-23.

[25] Conor Power, Hiren Patel, Alekh Jindal, Jyoti Leeka, Bob Jenkins, Michael Rys, Ed Triou, Dexin Zhu, Lucky Katahanas, Chakrapani Bhat Talapady, Josh Rowe, Fan Zhang, Rich Draves, Ivan Santa, and Amrish Kumar. 2021. The Cosmos Big Data Platform at Microsoft: Over a Decade of Progress and a Decade to Look Forward. *Proc. VLDB Endow.* 14, 12 (2021), 3148–3161. doi:10.14778/3476311.3476390

[26] Sajjad Tamimi, Arthur Bernhardt, Florian Stock, Ilia Petrov, and Andreas Koch. 2025. DANSEN: Database Acceleration on Native Computational Storage by Exploiting NDP. *ACM Trans. Reconfigurable Technol. Syst.* 18, 1 (2025), 4:1–4:33. doi:10.1145/3655625

[27] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao. 2017. Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases. In *Proceedings of the 2017 ACM International Conference on Management of Data, SIGMOD Conference 2017, Chicago, IL, USA, May 14-19, 2017*. ACM, 1041–1052. doi:10.1145/3035918.3056101

[28] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, James Corey, Kamal Gupta, Murali Brahmadesam, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao. 2018. Amazon Aurora: On Avoiding Distributed Consensus for I/Os, Commits, and Membership Changes. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*. ACM, 789–796. doi:10.1145/3183713.3196937

[29] Tobias Vinçon, Christian Knödler, Leonardo Solis-Vasquez, Arthur Bernhardt, Sajjad Tamimi, Lukas Weber, Florian Stock, Andreas Koch, and Ilia Petrov. 2022. Near-Data Processing in Database Systems on Native Computational Storage under HTAP Workloads. *Proc. VLDB Endow.* 15, 10 (2022), 1991–2004. doi:10.14778/3547305.3547307

[30] AWS Webservices. [n. d.]. Amazon Athena. https://docs.aws.amazon.com/whitepapers/latest/big-data-analytics-options/amazon-athena.html. Accessed: 2026-02-23.

[31] Marcel Weisgut, Daniel Ritter, Pinar Tözün, Lawrence Benson, and Tilmann Rabl. 2025. CXL Memory Performance for In-Memory Data Processing. *Proc. VLDB Endow.* 18, 9 (2025), 3119–3133. doi:10.14778/3746405.3746432

[32] Xinjun Yang, Yingqiang Zhang, Hao Chen, Feifei Li, Gerry Fan, Yang Kong, Bo Wang, Jing Fang, Yuhui Wang, Tao Huang, Wenpu Hu, Jim Kao, and Jianping Jiang. 2025. Unlocking the Potential of CXL for Disaggregated Memory in Cloud-Native Databases. In *Companion of the 2025 International Conference on Management of Data, SIGMOD/PODS 2025, Berlin, Germany, June 22-27, 2025*. ACM, 689–702. doi:10.1145/3722212.3724460

[33] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: a unified engine for big data processing. *Commun. ACM* 59, 11 (2016), 56–65. doi:10.1145/2934664

[34] Qizhen Zhang, Philip A. Bernstein, Badrish Chandramouli, Jason Hu, and Yiming Zheng. 2024. DDS: DPU-optimized Disaggregated Storage. *Proc. VLDB Endow.* 17, 11 (2024), 3304–3317. doi:10.14778/3681954.3682002