The Cambridge Report on Database Research

October 19–20, 2023

ANASTASIA AILAMAKI, SAMUEL MADDEN, DANIEL ABADI, GUSTAVO ALONSO, SIHEM AMER-YAHIA, MAGDALENA BALAZINSKA, PHILIP A. BERNSTEIN, PETER BONCZ, MICHAEL CAFARELLA, SURAJIT CHAUDHURI, SUSAN DAVIDSON, DAVID DEWITT, YANLEI DIAO, XIN LUNA DONG, MICHAEL FRANKLIN, JULIANA FREIRE, JOHANNES GEHRKE, ALON HALEVY, JOSEPH M. HELLERSTEIN, MARK D. HILL, STRATOS IDREOS, YANNIS IOANNIDIS, CHRISTOPH KOCH, DONALD KOSSMANN, TIM KRASKA, ARUN KUMAR, GUOLIANG LI, VOLKER MARKL, RENÉE MILLER, C. MOHAN, THOMAS NEUMANN, BENG CHIN OOI, FATMA OZCAN, ADITYA PARAMESWARAN, IPPOKRATIS PANDIS, JIGNESH M. PATEL, ANDREW PAVLO, DANICA POROBIC, VIKTOR SANCA, MICHAEL STONEBRAKER, JULIA STOYANOVICH, DAN SUCIU, WANG-CHIEW TAN, SHIV VENKATARAMAN, MATEI ZAHARIA, STANLEY B. ZDONIK

HTTPS://ARXIV.ORG/ABS/2504.11259V1 15 APR 2025

Previous Reports

Seattle Report on Database Research. SIGMOD Rec. 48, 4 (Feb. 2020)

The Beckman report on database research. Commun. ACM 59, 2 (Jan. 2016)

The Claremont report on database research. SIGMOD Rec. 37, 3 (Sept. 2008)

The Lowell database research self-assessment. Commun. ACM 48, 5 (May 2005)

The Asilomar report on database research. SIGMOD Rec. 27, 4 (Dec. 1998)

Avi Silberschatz and Stan Zdonik. 1996. Strategic directions in database systems—breaking out of the box. ACM Comput. Surv. 28, 4 (Dec. 1996)

Avi Silberschatz,Mike Stonebraker, and Jeff Ullman. 1996. Database research: achievements and opportunities into the 21st century. SIGMOD Rec. 25, 1 (March 1996)

Avi Silberschatz, Michael Stonebraker, and Jeffrey D. Ullman. 1990. Database systems: achievements and opportunities. SIGMOD Rec. 19, 4 (Dec. 1990)

Future Directions in DBMS Research - The Laguna Beach Participants. SIGMOD Rec. 18, 1 (1989)

Positioning of the report

is **not** intended as an exhaustive survey of all technical challenges or industry innovations in the field

reflects the perspectives of senior community members on the **most pressing challenges** and **promising opportunities** ahead

Evolution Over The Past Five Years

Important advances in the database and data systems landscape for past five years

- new hardware
- cloud-based data systems
- continued adoption of statistical techniques, ML, and AI in both core data systems architecture and components

Rise of Large Language Models (LLMs)

- LLMs are still evolving and have yet to reach their full potential
- offer a promising solution to many complex data challenges involving natural language and unstructured data
- unlocked new possibilities for understanding human intentions and needs
- paving the way for more intuitive, natural language-based querying and analysis interfaces
- comprehend data, including video and text, and to ground structured data in broader general knowledge
- synthesize complex, multi-step data transformation programs

Outline

Core Data Systems
- Database Systems
- Cloud Data Systems
- New Hardware
- Learned Components, Autotuning, and Opportunities for ML-in-databases

Generative AI / Large Language Models
- LLMs for Database Systems
- Data Systems for LLMs

Responsible Data Management and Data Governance

Collaboration, Integration, and Human-Centric Data Issues
- Data Sharing and Collaboration
- Data Integration
- Human-Centered Systems
- Data Science and Data-Intensive Science

Database Systems I

Open-source database engines with roots in research continue to see increased adoption and commercialization

- **PostgreSQL** continues to be a popular choice for both on-premises and cloud deployments
- Apache **Spark** and Apache **Flink** as scalable data processing systems
- DuckDB - an embeddable analytical database engine
  - efficient query processing in local environments
  - increasingly popular in contexts where database engines were previously absent, including data science workflows and graphical user interfaces

Composable building blocks

- [Meta] Velox, Apache Arrow DataFusion, Apache Calcite
- reusable, high-performance components for data processing that can be easily integrated into different systems
- standardizing interfaces for common database operations and components

Bridging traditional **data management** and the **needs of data scientists**

- Dask, Modin, Polars [https://pola.rs/], Spark
- DBMS-style semantics and optimization have been brought to bear on data science workloads

DuckDB Applications

Mark Raasveldt and Hannes Mühleisen. 2019. DuckDB: an Embeddable Analytical Database. SIGMOD '19. ACM, New York, NY, USA. https://doi.org/10.1145/3299869.3320212

Atwal, R.J., Boncz, P.A., Boyd, R., Courtney, A., Döhmen, T., Gerlinghoff, F., Huang, J., Hwang, J., Hyde, R., Felder, E. and Lacouture, J., 2024. MotherDuck: DuckDB in the cloud and in the client. In CIDR.

Analytical Workflows

- Lightweight Text Analytics Workflows with DuckDB
  - https://duckdb.org/2025/06/13/text-analytics
- Machine Learning Prototyping with DuckDB and scikit-learn
  - https://duckdb.org/2025/05/16/scikit-learn-duckdb
- Data Science ETL Pipelines with DuckDB
  - https://www.kdnuggets.com/data-science-etl-pipelines-with-duckdb

GUIs

- MEET THE NEW DUCKDB LOCAL UI: ANALYZE DATA VISUALLY, RIGHT WHERE IT LIVES
  - https://motherduck.com/blog/local-duckdb-ui-visual-data-analysis/
- Get a GUI for your Iceberg lakehouse with DuckDB UI from Motherduck
  - https://tower.dev/blog/get-a-gui-for-your-iceberg-lakehouse-with-duckdb-ui-from-motherduck

References

Pedro Pedreira, Orri Erling, Masha Basmanova, Kevin Wilfong, Laith Sakka, Krishna Pai, Wei He, and Biswapesh Chattopadhyay. 2022. **Velox**: meta's unified execution engine. Proc. VLDB Endow. 15, 12 (August 2022), 3372–3384. https://doi.org/10.14778/3554821.3554829

Andrew Lamb, Yijie Shen, Daniël Heres, Jayjeet Chakraborty, Mehmet Ozan Kabak, Liang-Chi Hsieh, and Chao Sun. 2024. Apache Arrow **DataFusion**: A Fast, Embeddable, Modular Analytic Query Engine. In Companion of the 2024 International Conference on Management of Data (SIGMOD '24). Association for Computing Machinery, New York, NY, USA, 5–17. https://doi.org/10.1145/3626246.3653368

Edmon Begoli, Jesús Camacho-Rodríguez, Julian Hyde, Michael J. Mior, and Daniel Lemire. 2018. Apache **Calcite**: A Foundational Framework for Optimized Query Processing Over Heterogeneous Data Sources. In Proceedings of the 2018 International Conference on Management of Data (SIGMOD '18). Association for Computing Machinery, New York, NY, USA, 221–230. https://doi.org/10.1145/3183713.3190662

References

Amandeep Khurana and Julien Le Dem. 2018. The Modern Data Architecture: The Deconstructed Database. login Usenix Mag. 43, 4 (2018). https://www.usenix.org/publications/login/winter-2018-vol-43-no-4/khurana

Pedro Pedreira, Orri Erling, Konstantinos Karanasos, Scott Schneider, Wes McKinney, Satyanarayana R. Valluri, Mohamed Zaït, and Jacques Nadeau. 2023. The Composable Data Management System Manifesto. Proc. VLDB Endow. 16, 10 (2023), 2679–2685. https://doi.org/10.14778/3603581.3603604

DataFusion: Modular Analytic Query Engine

Embeddable and extensible query engine written in Rust that

- uses Apache Arrow as its memory model (cache-efficient columnar layouts)

Extension points

- Scalar, Aggregate, and Window Functions
- Catalog
- Data Sources
- Execution Environment
- Query / Language Frontends
- Query Rewrites / Optimizer Passes
- Relational Operators

"DataFusion and DuckDB exhibit **similar scaling behavior**, and thus we conclude DataFusion's modular design and pull based scheduler do not preclude state of the art multi-core performance"

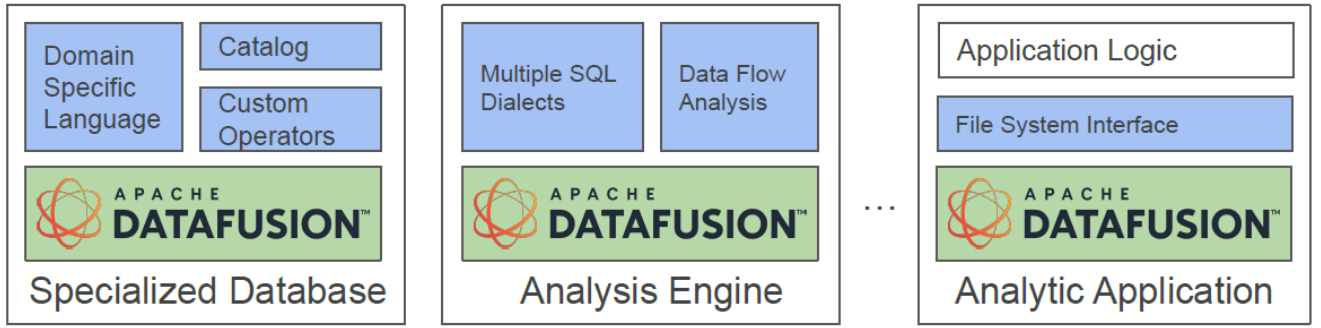


Figure 1: When building with DataFusion, system designers implement domain-specific features via extension APIs (blue), rather than re-implementing standard OLAP query engine technology (green).

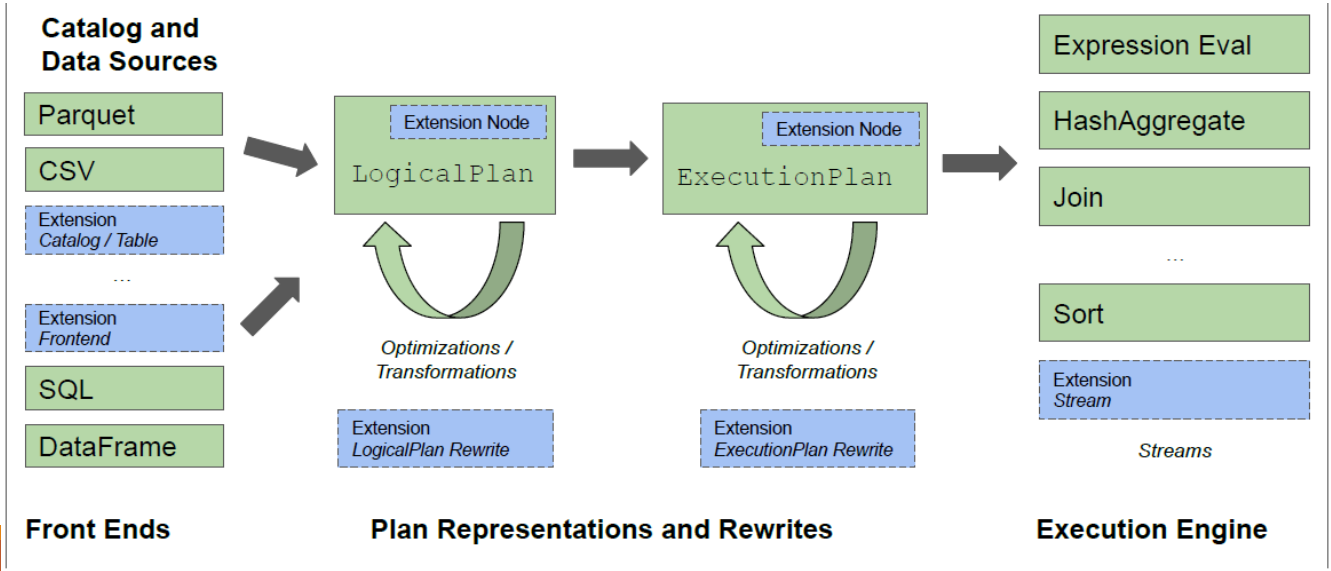


Figure 2: Architecture (Section 5). DataFusion's standard query engine subsystems (green) run queries "out of the box". Systems built on top of DataFusion customize behavior using extension APIs (blue).

Velox: Meta's Unified Execution Engine

The available data types, functions, and aggregates **vary** across these systems, and their behavior can be vastly **inconsistent** across engines

- 12 different implementations of the $substr$ with different parameter semantics (0- vs. 1-based indices), null handling, and exception behavior

Velox: an open source C++ database acceleration library

- Provides reusable, extensible, high-performance, and dialect-agnostic data processing components for building execution engines, and enhancing data management systems
- Does not provide a language frontend; instead, expects a fully optimized query plan as input and executes it locally using the resources available in the local host
- Does not provide a global query optimizer, but at execution time leverages adaptivity techniques, such as filter and conjunct reordering, dynamic filter pushdown, and adaptive column prefetching

Use cases: Presto, Spark, XStream, Scribe, FBETL, TorchArrow, F3,

Highlevel components

- **Type**: a generic type system including scalar, complex, structs, maps, arrays, tensors, …
- **Vector**: an Arrow-compatible2 columnar memory layout module, supporting multiple encodings
- **Expression Eval**: a fully vectorized expression evaluation engine
- **Functions**: APIs to build custom functions; simple (row-by-row) and vectorized (batch-by-batch) interface for scalar functions; for aggregate functions
- **Operators**: implementation of common data processing operators - TableScan, Project, Filter, Aggregation, Exchange/Merge, OrderBy, HashJoin, MergeJoin, Unnest
- **I/O**: a generic connector interface pluggable file format encoders/decoders and storage adapters
  - ORC, Parquet, S3, HDFS
- **Serializers**: a serialization interface targeting network communication
- **Resource Management**: a collection of primitives for handling computational resources, such as memory arenas and buffer management, tasks, drivers, and thread pools for CPU and thread execution, spilling, and caching

References

Dugré, M., Hayot-Sasson, V., & Glatard, T. (2023). Performance comparison of **Dask** and Apache **Spark** on HPC systems for neuroimaging. Concurrency and Computation: Practice and Experience, 35(21), e7635.

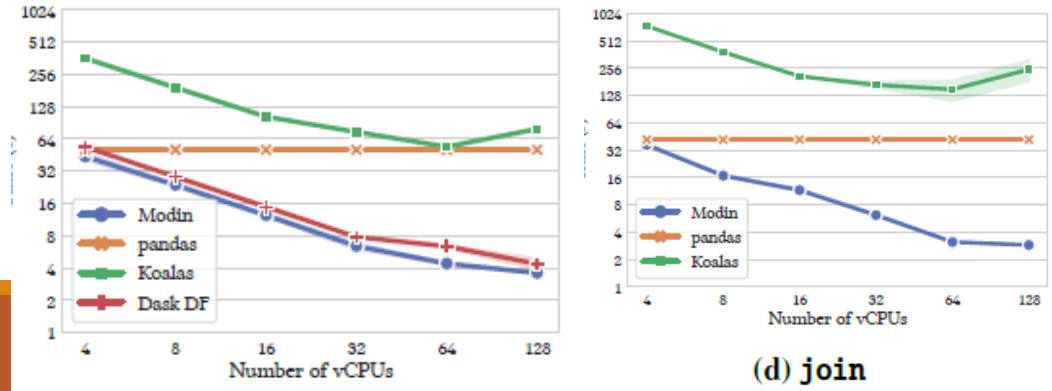
Devin Petersohn, Dixin Tang, Rehan Durrani, Areg Melik-Adamyan, Joseph E. Gonzalez, Anthony D. Joseph, and Aditya G. Parameswaran. Flexible Rule-Based Decomposition and Metadata Independence in **Modin**: A Parallel Dataframe System. PVLDB, 15(3): 739-751, 2022.

Modin: A Parallel Dataframe System

Dataframe systems (ex. pandas) are used by data scientists for data transformation, validation, cleaning, and exploration

Dataframe systems like are **non-interactive** on moderate-to-large datasets, and **break down** completely when operating on datasets **beyond main memory**

MODIN: ensuring scalability of dataframe operators, while also providing clear, consistent, and correct semantics to users



Formal basis: decomposition rules that allow us to

- rewrite operations on the original dataframe into analogous operations on vertical, horizontal, or block-based partitions of the dataframe
- while being able to concatenate the outputs to reproduce the results on the original operations

Metadata independence

- metadata is captured at a logical level, with the physical representation of the metadata being decoupled from the logical
- Independent type system for dataframes that natively supports mixed and unspecified types in a column, whereby we can defer type inference to only when it is needed

Database Systems II

Federation

Pulling data from diverse sources to answer complex queries

New challenges around automated infrastructure management and performance optimization

- techniques to push down query predicates to remote data sources can help to enhance performance by reducing data movement
- write propagation and data integration in such environments will be essential to ensure consistency and reliability
- data residency laws require data not to leave specific jurisdictions, such as the EU

Gu, Z., Corcoglioniti, F., Lanti, D., Mosca, A., Xiao, G., Xiong, J., & Calvanese, D. (2024). A systematic overview of **data federation systems**. Semantic Web, 15(1), 107-165.

Pushdown analysis. IBM Db2 Big SQL 7.1.0 Documentation. https://www.ibm.com/docs/en/db2-big-sql/7.1.0?topic=processing-pushdown-analysis

Introduction to **federated queries**. Google BigQuery Documentation. https://docs.cloud.google.com/bigquery/docs/federated-queries-intro

Cloud Data Systems

Cloud-native architectures have matured significantly

Traditional relational DBMSs, NoSQL, distributed SQL offerings

The industry has widely adopted the concept of disaggregated storage and compute, enabling a high degree of scalability and flexibility

Significant trend: **shared storage**, where any processing node can access any data element with a soft allocation of nodes to data partitions

- independent scaling of compute and storage
- separation of concerns between the data processing and durable storage layers

❑ Amazon Redshift, Google BigQuery, Databricks, Microsoft Fabric, Snowflake

Research directions

Database virtualization - a single database frontend automatically provisions and routes queries to the best infrastructure

Declarative infrastructure as a service

- interfaces are used for specifying more than queries but also the infrastructure upon which systems run
- search and optimization systems attempt to allocate infrastructure in the most cost-effective way

Collecting high-quality data to benchmark cloud systems and train machine learning models that will power adaptive and learned features of systems

- anonymized or aggregated workloads from operational environments
- synthetic benchmarks that accurately mirror real-world use cases (LLM to be applied?)

New Hardware

Hardware continues to evolve rapidly to cater to resource-hungry AI

Database community has made strides in leveraging improved hardware capabilities

- NVMe SSD
  - new storage engines that can fully utilize their high IOPS and low latency
- Persistent memory (e.g. Intel Optane DIMMs)
  - novel index structures that provide crash consistency without the overhead of traditional write-ahead logging
- CXL (Compute Express Link)
  - memory expansion and sharing techniques across servers
- GPU
  - massively parallel query processing, particularly for hash joins and sorting
- Tensor PU
- Data PU, Smart Network Interface Card (NIC)
- Specialized AI accelerators: FPGA, ASIC
  - data decompression and filtering

Research Directions

developing abstractions to leverage diverse accelerators

designing new data-centric accelerators

exploiting parallelism in GPUs for database operations

processing near memory (e.g., in Smart-NICs or DPUs) ?

References

Cheng Li, Hao Chen, Chaoyi Ruan, Xiaosong Ma, and Yinlong Xu. 2021. Leveraging **NVMe SSDs** for Building a Fast, Cost-effective, LSM-tree-based KV Store. ACM Trans. Storage 17, 4, Article 27 (November 2021), 29 pages. https://doi.org/10.1145/3480963

Lavinsky, B., & Zhang, X. (2022, July). PM-Rtree: A highly-efficient crash-consistent R-tree for **persistent memory**. In Proceedings of the 34th International Conference on Scientific and Statistical Database Management (pp. 1-11).

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 (May 2025), 3119–3133. https://doi.org/10.14778/3746405.3746432

Tobias Maltenberger, Ilin Tolovski, and Tilmann Rabl. 2025. Efficiently Joining Large Relations on **Multi-GPU Systems**. Proc. VLDB Endow. 18, 11 (July 2025), 4653–4667. https://doi.org/10.14778/3749646.3749720

Xuan Sun, Chun Jason Xue, Jinghuan Yu, Tei-Wei Kuo, and Xue Liu. 2021. Accelerating data filtering for database using **FPGA**. J. Syst. Archit. 114, C (Mar 2021). https://doi.org/10.1016/j.sysarc.2020.101908

References

Hu, J., Bernstein, P. A., Li, J., & Zhang, Q. (2024). DPDPU: data processing with **dpus**. arXiv preprint arXiv:2407.13658. CIDR 2025.

Tibbetts, N., Ibtisum, S., & Puri, S. (2025). A Survey on Heterogeneous Computing Using **SmartNICs** and Emerging Data Processing Units. Future Generation Computer Systems, 108207.

SpanDB (2021)

LSM-tree-based KV store that adapts RocksDB to utilize **NVMe SSD**

The bulk of data are hosted on on cheaper and larger SSDs

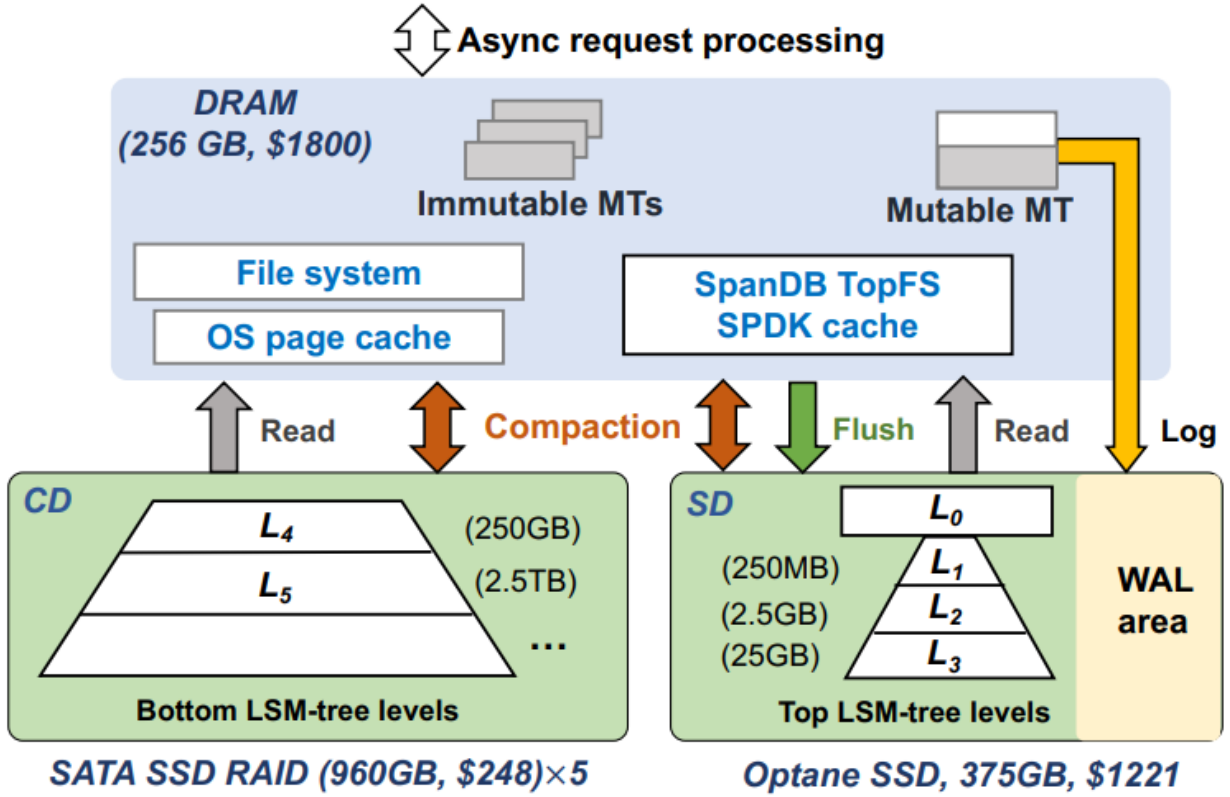
Write-ahead logs (WAL) and top levels of the LSM-tree are relocated to a much smaller and faster NVMe SSD

High-speed, parallel WAL writes

TopFS - a file system to enable live data migration between fast and slow disks

Evaluation

- SpanDB simultaneously improves RocksDB's throughput by up to 8.8x and reduces its latency by 9–58%
- Compared with KVell, a system designed for high-end SSDs, SpanDB achieves 96–140% of its throughput, with a 2.3–21.6 lower latency, at a cheaper storage configuration



Persistent Merged R-tree

High-efficient insert, delete, and search operations for high-dimensional datasets using persistent memory

Partitioned data structure

- non-leaf nodes are stored in DRAM
- leaf nodes are stored in persistent memory

Interleaved mapping approach

- maps contiguous data blocks in persistent memory to interleaved bits in bitmap groups in different cache lines to reduce cache line reflushes

Supports lock-free insertion using persistent multi-word compare and swap operations to eliminate locking overhead

Reduces the latency of insertion by up to 77.6% and 80% for the uniform and zipfian datasets respectively compared to the state-oftheart persistent R-trees while maintaining crash consistency

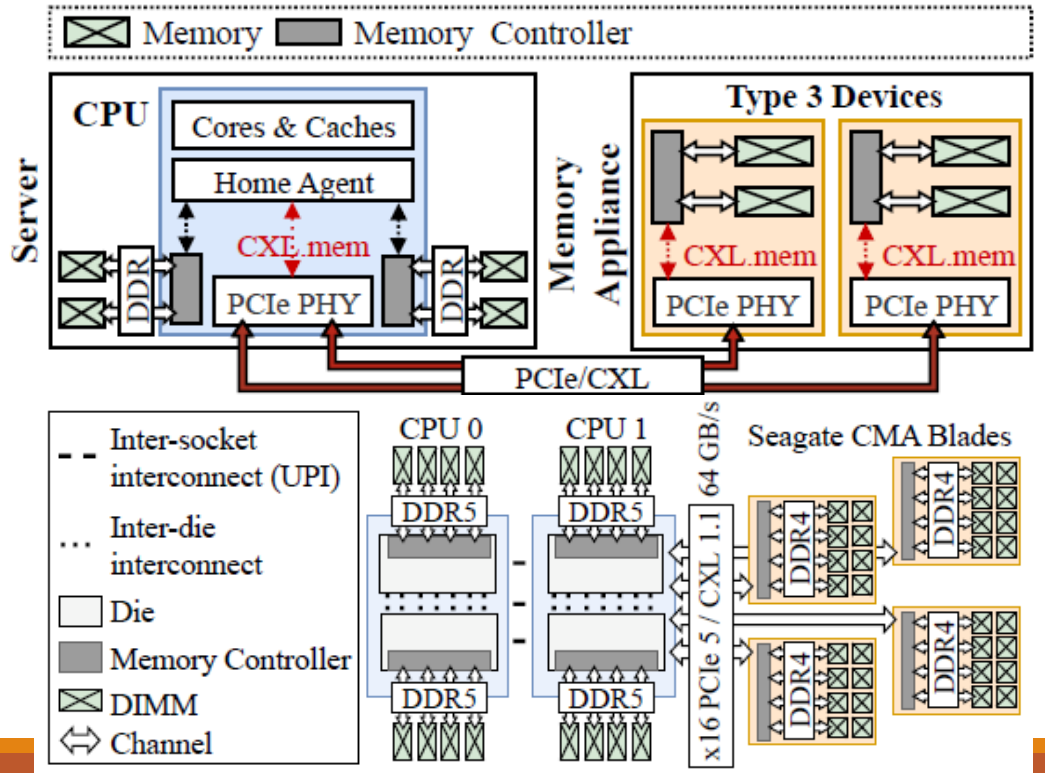
Reduces the search time by 19.2% compared to FBR-tree

Achieves better scalability for both insertion and search up to 32 threads

CXL Memory Performance for In-Memory Data Processing

Database operation performance with data interleaved across multiple CXL memory devices

- Composable Memory Appliance (CMA) Blade prototypes - FPGA-based memory expansion solution



CXL for Sequential Accesses: data structures that are primarily read randomly (hash tables) or frequently written should be placed in CPU memory, while sequentially accessed data (frequently scanned columns) can be placed in CXL memory

Attaching multiple CXL memory expansion devices to a CPU increases the overall memory bandwidth

CXL Memory Expansion devices can be **cheaper** than using only CPU memory

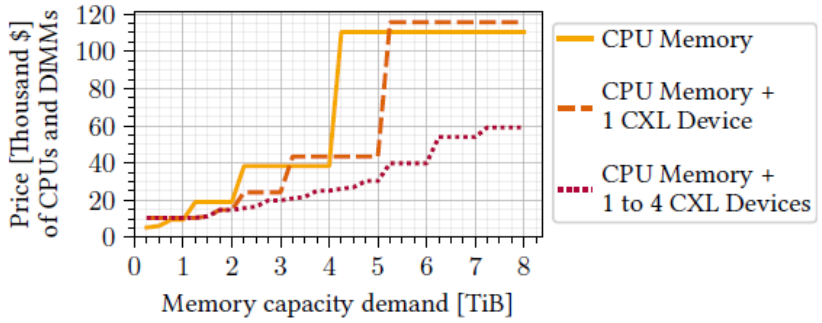


Figure 14: Cheapest CPU and memory configurations with and without CXL memory devices.

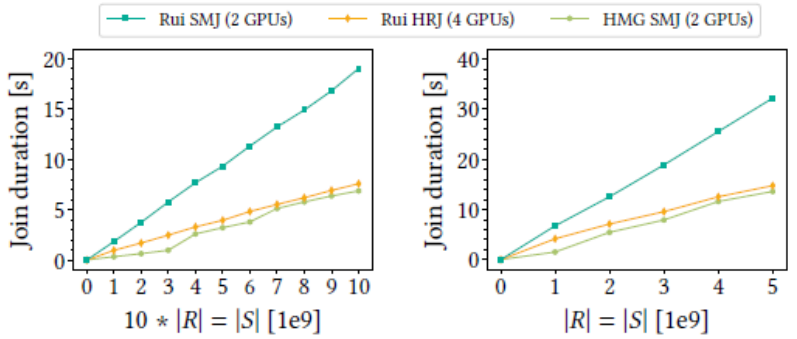
Efficiently Joining Large Relations on Multi-GPU Systems

Few existing multi-GPU algorithms either

- fail to exploit the high-speed P2P interconnects between the GPUs or
- to handle large out-of-core data atively

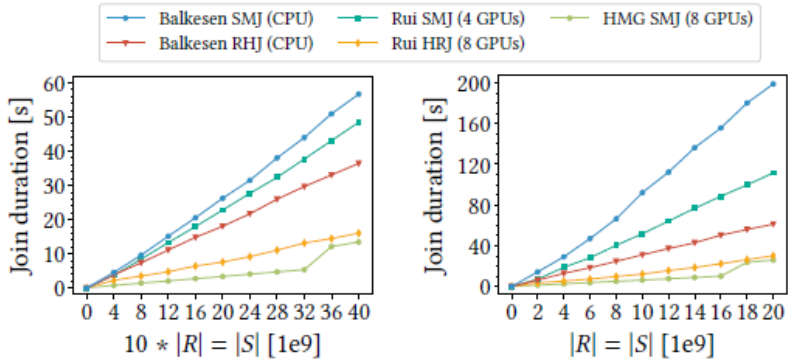
Heterogeneous multi-GPU **sort-merge join** for large out-of-core data exceeding the combined GPU memory capacity

- a multi-GPU-accelerated merge- or radix partitioning-based sort phase
- a parallel CPU-based multiway merge phase
- a hybrid join phase that combines a CPU merge path partition with a multi-GPU-accelerated join strategy



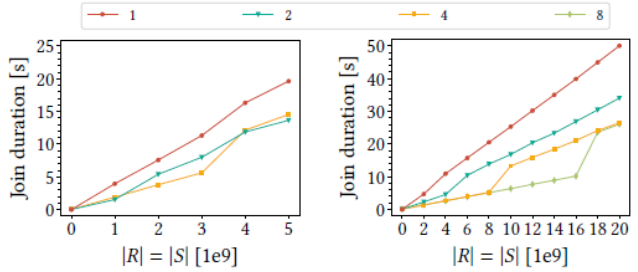
(a) Workload A (b) Workload B

Figure 11: Baseline comparison on the IBM AC922



(a) Workload A (b) Workload B

Figure 12: Baseline comparison on the NVIDIA DGX H100



(a) IBM AC922 (b) NVIDIA DGX H100

Figure 15: Scalability for increasing numbers of GPUs

Data Processing with DPUs

Challenges in the cloud

Compute inefficiency

- CPU speed has been increasing rather slowly over the past decade
- data systems frequently invoke compute-heavy subroutines: compress and encrypt
- can data systems still rely on CPUs to sustain good performance on these compute tasks?

I/O cost: high-bandwidth I/O is among the most common tasks in database systems

- the number of CPU cycles increases linearly with I/O throughput

Disaggregation overhead

- better flexibility in resource management at the expense of additional network I/O for storage accesses leading to higher access latency and even more CPU consumption

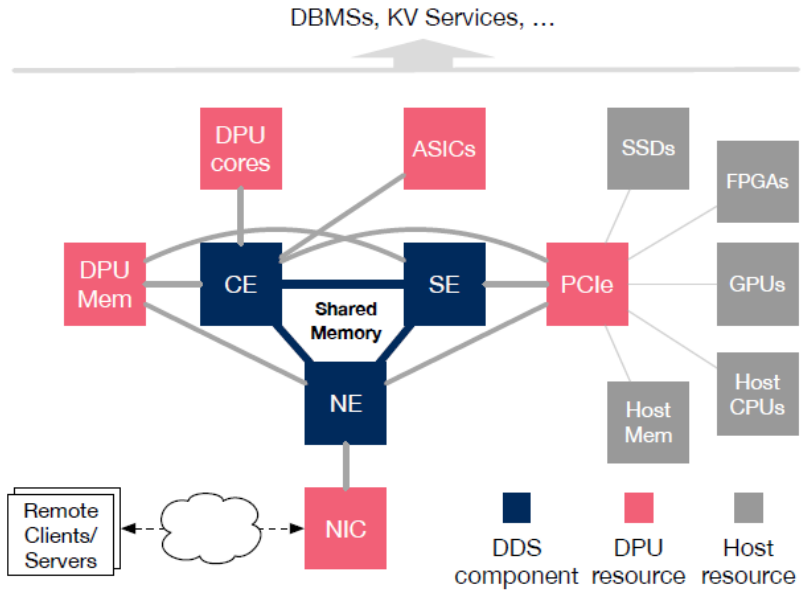


Figure 5: DPDPU components—Compute, Network, and Storage Engines (CE, NE, and SE)—and resources they access.

Compute Engine offers efficient and versatile computational power for data processing tasks orchestrated across DPU, host CPUs, GPUs, FPGAs, connected via PCIe

Network Engine handles network I/O utilizing the advanced networking facilities built in DPUs (high-speed interfaces, match-action offloading, and user libraries)

Storage Engine improves storage path efficiency, including requests from both local applications and those from remote clients

Learned Components, Autotuning, and ML-in-databases

Machine learning for DBMS internals

- **Query optimization**: cost models based on learning over data and workloads for complex, multi-join queries
- **Cardinality estimation:** deep neural networks and statistical models capture high-dimensional correlations in data distributions
- Reinforcement learning techniques are deployed to dynamically **adjust physical data organization** based on observed query workloads, with predictive I/O techniques demonstrating the potential to outperform traditional indexing methods
- **Cloud resource management**: ML models for serverless VM management have shown significant reductions in cold start times and resource costs

References

Bolin Ding, Rong Zhu, and Jingren Zhou. 2024. Learned **Query Optimizers**. Found. Trends databases 13, 4 (Sep 2024), 250–310. https://doi.org/10.1561/1900000082

Yang, Z. (2022). Machine learning for **query optimization** (Doctoral dissertation, University of California, Berkeley).

Bodra, D., & Khairnar, S. (2025). Machine learning-based **cloud resource allocation** algorithms: a comprehensive comparative review. Frontiers in Computer Science, 7, 1678976.

Generative AI / Large Language Models

Enabling more intuitive human interfaces for complex database systems

- auto-generating queries
- optimizing queries for performance
  - Song, M., & Zheng, M. (2024). A Survey of Query Optimization in Large Language Models. arXiv preprint arXiv:2412.17558.
- suggesting schema designs based on workload patterns and business requirements

Unclear whether LLMs are the right solution for many classic DBMS problems

- it seems unlikely that it will make sense to use LLM to solve query optimization problems
- APIs between data systems components will ever be replaced by “agents” interacting via natural language

Handling the high cost of inference at scale

- efficient computational stacks by combining data partitioning, caching, embedding indexes
- databases and provenance tools play a crucial role in validating outputs to reduce hallucinations

Moving beyond basic RAG methods by developing smarter, context-aware retrieval systems

LLMs for Database Systems

Text-to-SQL

- Spider Text-To-SQL Challenge [https://yale-lily.github.io/spider], ChatGPT 4 – rank 1-7

Tasks **beyond** Text-to-SQL

- Interpreting database manuals
- Tuning database parameters
- Aiding DBAs
- …

System design tasks

- composing database engines
- designing data pipelines

LLM Limitations

LLM inference is **unlikely** to suffice for **complex tasks**, necessitating robust pipelines that integrate verification steps and human oversight to ensure accuracy and reliability

LLMs must learn to **interact with database APIs**, adapting to different system interfaces through prompting or in-context learning (examples in prompt)

LLM effectiveness in **handling relational and other structured data** remains an open question

- state-of-the-art models struggle with fundamental relational properties, such as the set-based nature of relations

Cross-modal embeddings may enhance LLMs' ability to process relational data, textual data, time series, nested tables, …

- Qian, S., Zhou, Z., Xue, D., Wang, B., & Xu, C. (2024). From linguistic giants to sensory maestros: A survey on cross-modal reasoning with large language models. arXiv preprint arXiv:2409.18996

Conversational interfaces and **query debuggers** will be crucial to helping users validate and trust LLM-generated queries

Data Systems for LLMs

Building data systems that support LLMs

- efficient infrastructures to manage large multi-modal data sets
- optimize fine-tuning
- ensure scalability, fault tolerance, and elasticity in native cloud environments of both training and serving systems
- accommodate new storage and access methods including text, code, images, video, and audio

Data quality, labeling, and metadata management for LLM training

- large-scale datasets contain both valuable content and low-quality or biased data
- effective tools are needed to filter, de-bias, and curate such data efficiently

Creating evaluation frameworks for LLM applications

- Effective tools to reliably **post-process output data**, **debug failures**, and adapt inputs to LLM application workflows by leveraging both **human feedback** and emerging **reinforcement learning** methods at scale

Supporting complex prompt engineering workflows

- involves metadata management and strategic data chunking
- LangChain, LlamaIndex tools

Novel indexing and retrieval methods for multi-modal data in RAG systems

Complex “agentic” AI workflows

- integrate multiple LLM inferences, retrieval steps, ML models, and external tools such as code executors or search engines
- improve functionality, robustness, and efficiency
- introduce new trade-offs in latency and accuracy

Responsible Data Management

RDM = integrating data management research into the area of responsible AI (RAI)

- Stoyanovich, J., Abiteboul, S., Howe, B., Jagadish, H. V., & Schelter, S. (2022). Responsible data management. Communications of the ACM, 65(6), 64-74.

Observation: decisions we make during data collection and preparation profoundly impact the accuracy, fairness, robustness, interpretability, and legal compliance of AI systems

Responsible data or AI system must consider both the data and system life cycles—from data provenance and validity to design goals, deployment impacts, and unintended consequences

Research directions

Metadata management tools for large-scale AI models

Methods for federated data management, privacy-preserving sharing, and interoperable standards

Methods to **assess data quality** in relation to specific downstream tasks and socially meaningful metrics e.g., fairness, accuracy, and robustness

Creating techniques to **improve data quality** through acquisition, cleaning, and preprocessing

Establishing lifecycle-aware **provenance tracking** to meet the diverse interpretability needs of data scientists, auditors, …

Data Sharing and Collaboration

Data sharing and collaboration are required and enable cross-organizational analytics

- Challenges in terms of privacy, governance, and query processing across distributed datasets in data lakes

Research questions

Enable easy and accurate discovery of relevant data and discard or avoid useless information

- with LLMs and RAG architectures, when too much data is indexed, much of what is retrieved is useless or irrelevant

Ensure that we are only collecting and retaining data what is needed

Overcome a tension between accessibility and privacy and questions about who has a right to retain data

- large organizations are accumulating immense volumes of data, giving them an advantage when making predictions or training models

Data Integration

True integration requires addressing

- semantic differences
- resolving entity-matching issues
- ensuring data quality
- ensuring consistency across sources

It is required to advance our understanding of the semantic and structural aspects of data integration

Putrama, I. M., & Martinek, P. (2024). **Heterogeneous data integration**: Challenges and opportunities. *Data in Brief*, *56*, 110853.

LLM ability to fully solve the complex problem of data integration is still **uncertain** and requires further investigation

LLM and AI may help to address schema matching, entity resolution, and data cleaning at scale

- we need to understand and mitigate potential biases In AI-assisted integration processes

How would an LLM be able to tell if two tables are related in a data lake with thousands of tables, each with millions to billions of tuples?

- ways to expose just enough data to facilitate integration are needed

Human-Centered Systems

Building systems that augment human ability to manage and analyze data while addressing the limitations of LLMs (hallucinations) through mechanisms including

- fact-checking
- database constraint maintenance
- user-verified results

Supporting users in spreadsheets, BI platforms, and computational notebooks remains critical

- enhance environments using database concepts such as indexing, declarative queries, and automated optimization

Leveraging LLMs for automated insight discovery and visualization

- Designing intuitive, natural-language-driven interfaces and explanatory tools
- Emphasizing human-in-the loop feedback and continuous learning mechanisms

Data Science and Data-Intensive Science

Increasing focus on end-to-end data pipeline and workflow systems

- data preparation, analysis and visualization, ML/AI
- pipelines are used both
  - in an exploratory mode, where they are iteratively developed and refined
  - for the deployment of live services
- increasingly sophisticated tools for
  - workflow and data pipeline management
  - data discovery
  - data integration and cleaning
  - synthetic data generation
  - metadata and log management
  - code and data versioning

The idea of a single, **universal language** or paradigm (e.g., extending SQL) covering all data programming needs is **unlikely**, due to the diversity and specialization of data science tasks

Efforts should focus on developing **interoperable systems** that allow different tools and languages to work together enhancing performance while respecting domain-specific workflows

A tension between improving existing widely used tools and advocating for cleaner or higher performance abstractions that may have a steeper adoption curve

- making pandas more scalable remains important
- there's potential in defining more streamlined, learnable, and optimized dataframe abstractions that unify ideas from Dask, Polars, Ibis, Spark