Scalable Data Management using GPUs with Fast Interconnects

A Brief Overview

Clemens Lutz[1,2]

Abstract:

Modern *database management systems (DBMSs)* are tasked with analyzing terabytes of data, employing a rich set of relational and machine learning operators. To process data at large scales, research efforts have strived to leverage the high computational throughput and memory bandwidth of specialized co-processors such as *graphics processing units (GPUs)*. However, scaling data management on GPUs is challenging because (1) the on-board memory of GPUs has too little capacity for storing large data volumes, while (2) the interconnect bandwidth is not sufficient for ad hoc transfers from main memory. Thus, data management on GPUs is limited by a *data transfer bottleneck*. In practice, CPUs process large-scale data faster than GPUs, reducing the utility of GPUs for DBMSs.

This paper provides an overview of the author's dissertation [Lu22a]. In our dissertation, we investigate how a new class of *fast interconnects* can address the data transfer bottleneck and scale GPU-enabled data management. Fast interconnects link GPU co-processors to a CPU with high bandwidth and cache-coherence. We apply our insights to process stateful and iterative algorithms out-of-core by the examples of a hash join and *k*-means clustering. Overall, GPU-enabled DBMSs are able to overcome the data transfer bottleneck by employing new out-of-core algorithms that take advantage of fast interconnects.

Keywords: relational database, GPU-accelerated data processing, data transfer bottleneck, NVLink, Compute Express Link (CXL), Infinity Fabric, Triton join, single-pass k-means, translation lookaside buffer (TLB)

The paper is structured as follows. First, we contextualize the role of GPUs today in Sect. 1, and outline how GPUs fit into DBMSs in Sect. 2. In Sect. 3, we then motivate the dissertation, and summarize the research challenges addressed in the dissertation in Sect. 4. We define what a fast interconnect is in Sect. 5. We summarize our findings in Sect. 6 and give a research outlook in Sect. 7.

1 The State of GPU-Enabled Data Management

Large-scale data management has become a pillar of science and industry, enabling new research fields, services, and business models [HTT09; La14]. Earth monitoring

[1] TU Berlin, Berlin, Germany, clutz@nvidia.com, https://orcid.org/0000-0002-6193-4734
[2] Work done at TU Berlin. The author is currently employed at NVIDIA, Santa Clara, CA.

satellites [CV21] and genome sequencers [Di18] generate terabytes of data every day. Online services such as Google Search [Go22] and Uber [FS21] are backed by petabytes of data. *Database management systems (DBMS)* routinely ingest and manage these large data volumes[Am16; Ar20; Da16; Gu15].

In order to continue scaling data management as the progression of Moore's Law slows down [BC11; Es11; Ho14; HP19], co-processors such as GPUs, FPGAs, and ASICs have been gaining adoption in research [Ba18; IKS20; Lo19; RBM22; Wu14] and industry [Ca16; Jo21; KS21; Le21; RM16; Sh21] over the past decade. The entry barrier to co-processors is now low with instant availability from all major cloud vendors, including Alibaba Cloud, Amazon EC2, Google Compute Engine, and Microsoft Azure. Despite the growing adoption and availability, in 2019 commercial *GPU-enabled DBMSs* occupied only a tiny 4.5–5.5‰ slice [Ma20; Ma21; Zi20] in the $46 billion DBMS market [Ga19]. GPU-enabled DBMSs are found mostly in the form of research prototypes [Br18; Ch19; Fu18; He13; Me16; PHH16], start-ups [Bl21; Br22; FA19; He21; Ki22; Om21; Ra21; SQ22], and peripheral products [JC19; Sh19]. Major DBMS vendors, such as Amazon, IBM, Microsoft, Oracle, SAP, and Snowflake, currently do not integrate co-processors into the core of their DBMS products. In contrast, there is wide-spread adoption in the deep learning [De17; Ng19] and high performance computing domains. For instance, 42% of the Top500 supercomputers support co-processors [To24].

2 The Design of GPU-Enabled Database Management Systems

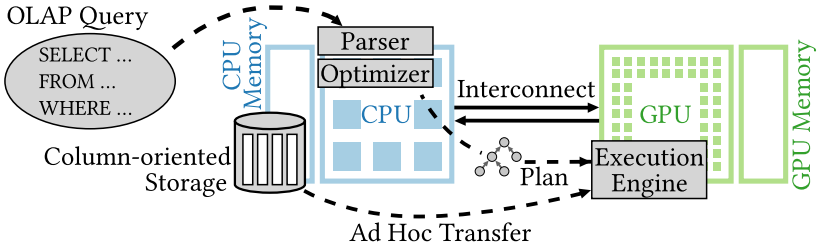

Fig. 1: Design of a GPU-enabled DBMS.

The state-of-the-art GPU-enabled DBMS prototypes and products have similarities in how they integrate GPUs and manage large data [BFT16; Ch19; Fu18; Le21; Me16; Ra20]. As a basis for further discussion, we extract relevant features into a general design. We present an overview in Figure 1. For an in-depth background, we refer the interested reader to the recent survey conducted by Rosenfeld et al. [RBM22].

GPU Use Cases. In principle, various tasks performed by a DBMS can be offloaded to a GPU, e. g., query execution [Go04], query optimization [HM12], transaction processing [BB22; HY11], stream processing [Ko16], and data loading [KLM21]. However, OLAP and machine learning queries are the most established use-cases for GPUs in DBMSs [RBM22; Re20].

OLAP and ML. *Online analytical processing (OLAP)* and machine learning queries are formulated ad hoc by the user to explore a data set [BKY19; CD97]. They are complex, i. e., consisting of scans, joins, and aggregations in the case of OLAP, and additionally linear algebra operators and iterative control flow in the case of machine learning. Although these workloads are read-heavy, the queries work on the entire data set. As a human is in the loop, response times should be short. Thus, the goal of GPU-enabled DBMSs is to achieve the high data rates necessary to query large data volumes in a timely fashion.

DBMS Design. From a design perspective, GPU-enabled DBMSs are column-oriented, in-memory DBMSs [Br14; He09; He13; YLZ13]. This DBMS design stores data in CPU memory and is optimized for analytical query processing [BZN05; KN11]. Storing large data sets in CPU memory is possible because modern servers have terabytes of memory capacity (i. e., *DRAM*) [21c; 21d; St18], and non-volatile memory technologies such as Intel Optane are capable of increasing the capacity by an order-of-magnitude [Ar19; Sh20]. Column-oriented in-memory DBMSs increase the data access performance relative to row-oriented data layouts [Br14; He09].

Query Execution. Within the DBMS, GPUs extend the query execution engine [BFT16; Ch19; Fu18; Le21; Me16; Ra20]. The DBMS gains a runtime with operators specialized for GPU execution. During execution, the DBMS transfers the data from CPU memory to the GPU ad hoc and executes the query [BFT16; Ch19; Fu18; Le21; Me16; Ra20]. The other DBMS components required to execute a query, e. g., SQL parsing and query planning and optimization, are retained on the CPU [Br14; CSA19; He13].

Single Data Pass. Different execution paradigms exist to construct query execution engines. Some engines generate code just-in-time to emit operator pipelines [Br18; Ch19; Fu18; Pa20], others vectorize [Me16] or tile [PHH16; SMY20] the execution, and operator-at-a-time execution is yet another variant [Br14; He13; Le21]. The processing paradigm is immaterial to our thesis, but we must consider that only operator pipelines and vectorized/tiled engines are capable of processing queries in a single data pass [Fu18; SMY20], which avoids transferring data items over the interconnect multiple times.

3 Motivation

From a design standpoint, modern GPU-enabled DBMSs are able to achieve a high query execution performance. Furthermore, the performance of GPU hardware is rapidly increasing with each GPU generation [Su20; Sv21]. However, in order to sustain their query performance, GPU-enabled DBMSs require fast access to data.

Motivation 1: Data-intensive Query Processing. To out compete CPUs in performance benchmarks, GPU experts often assume that their input data are stored in the on-board *GPU memory* [Br18; FT20; He13; Pa20; PMS15; SMY20]. GPU memory provides high-bandwidth access to the data, but has only a limited storage capacity and therefore cannot

hold large data volumes [20b; 21b]. In practice, GPU-enabled DBMSs scale to large data volumes by storing data in *CPU memory* instead of in GPU memory [Br14; Ch19; Fu18; Le21; Me16; PMK14; Ra20; YLZ13]. CPU memory has sufficient capacity to store large data volumes [21a; 21c; 22; St21], as its capacity is two orders-of-magnitude greater than GPU memory [21b; 21e]. However, moving data from CPU memory to the GPU reduces query performance because the data are transferred over an interconnect. Consequently, database research points out that a data transfer bottleneck is the main reason behind the comparatively slow adoption of GPU-enabled DBMSs [BS10; Fu18; GH11; Me16; Ra20; SMY20; YLZ13].

Motivation 2: Stateful Data Processing. During query processing, DBMSs require additional memory to retain the intermediate state of the query [Br14; CSA19; Fu18]. Queries involving state, e. g., joins, are considered a strong point of GPU-enabled DBMSs, as keeping the state in GPU memory results in high query performance [He08; KML15; NEK18; Pa21; Ro19; RT17; Sa10; Si19; SJ17; To18; Ya17]. However, recent investigations reveal that a large state size incurs memory contention [BFT16; Me16] and causes commercial GPU-enabled DBMSs to fail query execution [Ch19; Ch20; FT20; Le21; Pa21]. Thus, current GPU-enabled DBMSs are not optimized to handle large state. We consider fragility in scaling the state size an obstacle for building production-ready DBMS products.

Motivation 3: Iterative Algorithms. In addition to relational queries, modern DBMSs [Bl22; He12; KBY17; Kh18; Pa17; Sc20; Sc21; Zh21] and specialized systems [Bo16; Bo20; Cr15; Ku19; Ro13; Sp17; Xi18] target machine learning queries. Machine learning queries differ from relational queries in that they iterate over the same data, i. e., *the working set*, multiple times [BKY19]. Although GPU are able to quickly compute machine learning queries, GPU execution strategies for, e. g., *k*-means, typically assume that the working set fits into GPU memory [An13; As15; JDJ21; KK20]. In effect, systems scale-out to multiple GPUs to manage large data sets, which increases the cost of processing queries.

Overall, GPU co-processing does not scale to large data volumes. Currently, GPUs are only able to speed-up short queries over small data sets, which have a small scope for improvement. Hence, an opportunity is lost for state-of-the-art DBMSs to achieve faster response times on those long-running, large-scale queries where performance matters.

4 Research Challenges Addressed

In their dissertation [Lu22a], the author investigates the scalability limitations of GPU co-processing and analyze how a faster interconnect helps us to overcome them. A new class of *fast interconnects* provide GPUs with high-bandwidth, cache-coherent access to main memory. Recent examples include NVLink 2.0 [17a] and C2C [Nv24a; Nv24b], Infinity Fabric [19a], and Compute Express Link 2.0 [20a]. Efficiently utilizing a fast interconnect in a GPU-enabled DBMS requires us to reevaluate fundamental design decisions in order to adapt to the new hardware properties.

Our dissertation consists of three main contributions that address the challenges stated below. Based on our insights, we have continued our research by following up on the dissertation. We summarize:

Challenge 1: Scalable Data-intensive Query Processing. Our first contribution focuses on the interconnect hardware [Lu20]. Specifically, we examine the principle limitations of GPU interconnects in the context of databases. We introduce fast interconnects, and show by the example of NVLink 2.0 that a fast interconnect improves the basic bandwidth and latency characteristics beyond PCI-e 3.0. Due to these improvements, we conclude that GPUs are now capable of efficiently processing large, out-of-core data sets. However, our measurements show that fast interconnects also lead to new challenges, such as handling operators with a large state efficiently.

Challenge 2: Scalable and Robust Stateful Data Processing. Guided by our findings, we propose the *Triton join* [Lu22b], a new join algorithm that efficiently handles large-scale joins for which the join state exceeds the GPU memory capacity. Although fast interconnects provide faster random-access bandwidth than PCI-e, the join throughput experiences a sharp performance drop when the join state exceeds the GPU memory capacity. Thus, large joins face the challenges of limited scalability and robustness. Previous approaches avoid the transfer bottleneck by preprocessing data on the CPU and thereby reducing the transfer volume. In contrast, we show that fast interconnects enable GPUs to efficiently partition data out-of-core, and are thus able to scale to a large join state using exclusively the GPU. With our Triton join, the principle limitation posed by the GPU memory capacity is solvable with a throughput of more than 50% vs. a join with a small state.

Challenge 3: Scalable Iterative Algorithms. In our final contribution [Lu18a; Lu18b], we demonstrate that iterative machine learning algorithms are able to process large data sets by examining k-means and proposing a scalable GPU execution strategy. State-of-the-art execution strategies accelerate k-means by extracting the compute-intensive parts and offloading these computations to the GPU. However, as execution is split into a CPU phase and a GPU phase, this execution strategy incurs data transfer overhead and misses optimization opportunities. On each iteration, the CPU and the GPU transfer the model over the interconnect, and separately pass over the data. These overheads exacerbate as the algorithm requires tens of iterations to converge [El03]. As a result, the state-of-the-art strategy executes slower than a CPU-only baseline. In contrast, we avoid the model transfer overhead by proposing a new centroid update algorithm optimized for GPUs. In a second optimization step, this enables us to fuse all phases of k-means into a single data pass per iteration. Thus, our new GPU-only execution strategy efficiently scales k-means to large data volumes.

Follow-Up Work. Enabled by fast interconnects, new use cases for the GPU emerge. For example, data loading involves streaming data from an I/O device such as network interface or disk to the GPU, and parsing the data format. We demonstrate that the DBMS can

offload parsing of complex formats such as CSV to the GPU [KLM21]. In a similar vein, high-bandwidth network interfaces are pressuring stream processing engines to become more efficient [Ze19]. Due to algorithmic advances, compute-intensive stream joins are approaching the limit of PCI-e when processed on GPU [Mi21; Nu24]. Fast interconnects open the door to further improvements.

5 Fast Interconnects: A Definition

Up until this point, we have given an intuitive understanding of what the term *fast interconnect* means through examples. To cover the changing hardware landscape, we formalize the term by providing a definition. We base our definition on a set of hardware properties, and reason why these properties facilitate efficient data management. For the interested reader, our dissertation contains a deep-dive on GPU and interconnect hardware technology.

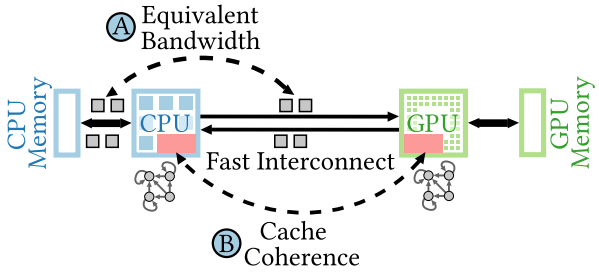

Fig. 2: The distinguishing properties of fast interconnects.

Definition. We define a "fast interconnect" to have two distinguishing properties which we illustrate in Figure 2: Ⓐ the aggregate bidirectional interconnect bandwidth approximately matches the per-socket CPU memory bandwidth and Ⓑ the interconnect is cache-coherent. In our definition, cache-coherence means that the hardware natively supports a system-wide address space, data-dependent accesses to pageable memory, and system-wide atomic memory operations.

Rationale. We justify why our definition requires high bandwidth and the four cache-coherence sub-properties. One, sufficiently high interconnect bandwidth is necessary to level the playing field between the GPU and the CPU. Without high bandwidth, the GPU cannot efficiently access CPU memory (and vice-versa). Two, cache-coherence (excluding the subsumed properties) is necessary because CPUs transparently cache data. Without cache-coherence, programmers must manage caches manually to avoid accessing stale data. Three, without a system-wide address space, programmers must manually translate pointers in order to move them from one processor's address space to the next. Four, without pageable memory access, memory accessed from a different processor must be pinned beforehand. Five, without atomic memory operations, processors cannot share and mutate data at a fine granularity. Overall, these properties work together to make memory accesses and memory management faster and more convenient.

Practical Limitations. Processors are free to take advantage of only a subset of the interconnect's features. For example, IBM POWER9 CPUs do not achieve the peak bandwidth of NVLink 2.0 [IB18], and the L1 caches of current GPUs do not implement cache-coherence in hardware [CGF18; Mi17]. Consequently, the programmer must deal with these shortcomings, e. g., by managing the consistency of cached data in software. However, in this case, the hardware continues to provide *memory consistency* [Nv24b], which is complementary to cache-coherence [HP17]. Ideally, the interconnect supports all features to avoid restricting processors to the intersection of their feature sets.

Conclusion. In our fast interconnect definition, we specify properties that complement each other and form a basis on which DBMSs can tightly integrate co-processors. Thus, a fast interconnect provides DBMSs with the means to resolve the data transfer bottleneck from a hardware perspective.

6 Scaling the Data Volume with a Fast Interconnect

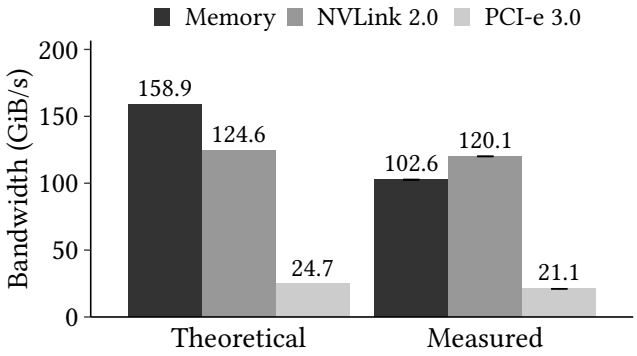

Fig. 3: NVLink 2.0 eliminates the GPU's main-memory access disadvantage compared to the CPU.

From a software perspective, however, DBMSs need to take advantage of the fast interconnect. We summarize the key findings of the dissertation.

Scalable Data Management using GPUs. We observe that GPUs can load data from CPU memory with bandwidth similar to the CPU, as shown in Fig. 3. Thus, offloading data processing on GPUs becomes viable even when the data is stored in CPU memory.

As a consequence of higher interconnect bandwidth, transfers are no longer the main bottleneck. In some cases, the high interconnect bandwidth shifts the bottleneck to other resources, such as random access bandwidth, TLB misses, and computation. For example, we have shown speedups of up to 6× over PCI-e 3.0 for hash joins operating on a data structure in GPU memory. In this case, performance is limited by random access bandwidth to GPU memory.

Due to the unified address space and lower latency, GPUs are able to operate on out-of-core data structures. In our evaluation, we showed up to 20× higher hash join throughput with NVLink 2.0 than with PCI-e 3.0. However, we recommend that GPUs should continue to operate in GPU memory if possible. Despite attaining speedups over PCI-e, operating within GPU memory is still 6.5× faster compared to transferring data over NVLink 2.0.

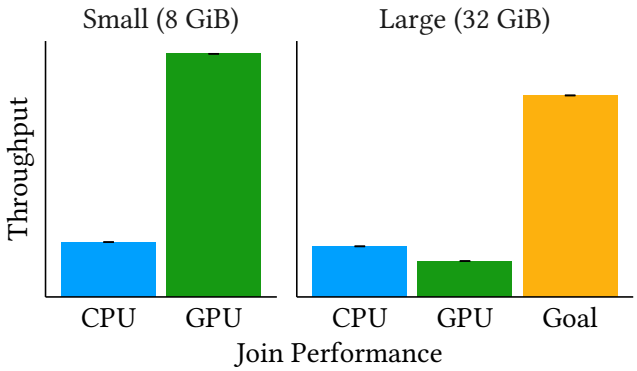

Fig. 4: Scaling to large data volumes needs an interconnect-conscious design.

A fast interconnect is necessary, but not sufficient. In Fig. 4, we showcase the principle. In the case of a small hash table, the interconnect hardware itself allows the GPU to outperform the CPU. However, when observing a large hash table, the GPU not only looses its advantage but experiences a slowdown. In contrast, the goal of our work is to consistently achieve a speedup.

For example, spilling data structures to CPU memory should be done with extreme caution. In the best case, a non-optimized hash join experiences irregular access patterns for a 2× slowdown compared to an optimized CPU radix join. In the worst case, the GPU hash join suffers from GPU TLB misses, which cause a further 400× slowdown compared to the TLB hit case.

However, a better algorithm can exploit that fast interconnects provide sufficient bandwidth to spill large state to CPU memory. e. g., with our Triton join, a 2× speedup over a strong CPU baseline is possible even when the state size exceeds the GPU memory capacity.

Interconnect-Conscious Design. As the bottleneck shifts, optimization becomes challenging. Multiple constraints can simultaneously affect different parts of the program. Simultaneously, the hardware properties of fast interconnects open the design space to create new algorithms.

We advocate an interconnect-conscious DBMS and algorithm design:

1. Regarding data access, DBMSs can take advantage of cache-coherence to directly access and mutate data anywhere in memory, potentially simplifying software design.

2. Regarding state access, awareness of the interconnect architecture can lead to higher performance. In case of the Triton join, the knowledge that interconnects are packet-oriented lead to *perfect coalescing* (i. e., tailoring the access pattern to fill each the packet with the maximum payload size).
3. Regarding data locality, end-to-end GPU execution eliminates overheads incurred by co-processing. This paradigm also enables exploiting the GPU's memory hierarchy to optimize execution.

Conclusion. Fast interconnects enable GPUs to cover a broader spectrum of database use-cases, but we require new algorithms to fully exploit the performance potential of fast interconnects. We have demonstrated efficient out-of-core data access, state access, and iterative algorithms.

7 Research Outlook

Our dissertation lays the foundation for research on scalable data management using GPUs with fast interconnects. At the time when the research for the dissertation was conducted, it was not clear whether fast interconnects would outlive a single hardware platform. As of this writing, we are headed towards the third GPU generation with a fast interconnect [17a; Nv24a; Nv24b], and major cloud service providers announcing availability [He24; Th24; Ve24]. This continued hardware evolution provides a perspective for further DBMS research.

In the following, we discuss five open research challenges.

DBMS Design. In our research, we have focused on joins and k-means to show that principle limitations of data management on GPUs can be solved using fast interconnects. Future work could broaden our findings to other relational operators such as selections, group-by aggregations, and set operators. We are currently investigating interconnect-conscious index structures to scale, e. g., highly selective queries, range queries, outer joins, and inequality joins. By lowering the penalty of data transfers, operator placement on heterogeneous processors could be reinvestigated. Our heterogeneous, morsel-driven work scheduling approach could be extended to operator pipelines, whereby the GPU could itself schedule work using native atomics. The trade-offs inherent to data compression should be reevaluated for fast interconnects and hardware-accelerated compression [Ab20; Nv24a; Rh18]. These innovations should be combined to improve the overall query performance of a DBMS.

Data Streaming. Fast interconnects are particularly interesting for data stream management due to the inherent network I/O. Modern network interface cards can achieve a bandwidth comparable to that of CPU memory, and thus require new approaches to ingest and process data [KLM21; Ze19]. However, individual streams are unlikely to reach these speeds. Taking advantage of the hardware would involve either complex streaming queries with many joins or scaling the number of queries. High data velocities could result in large windows that require spilling a large state to CPU memory via the interconnect. Furthermore, fast

interconnects can also integrate network interface cards into the system, which provides new research opportunities [RRW18].

Deep Learning. Neural network models such as BERT [De19] and GPT-3 [Br20] are affected by the data transfer bottleneck due to their large size. In effect, parameter servers must scale to a large model size with fast response times to parameter queries. An out-of-core parameter server could scale the model size by applying our hardware insights.

Data Loading. We notice that the data load time is typically not measured in research publications. However, benchmarks such as TPC-H reflect that real-world workloads often bulk load data before executing queries [17b, §4.3]. Loading converts the data format from, e. g., CSV, to a DBMS-specific format. We have shown promising results by leveraging GPUs to reduce the data load time [KLM21]. Future work could revisit our approach to load multiple data streams in parallel, create indices during loading, and validate the input format to guard against errors. Ultimately, this line of research would result in fast end-to-end query performance.

Storage Volume. Non-volatile memory and flash disks could be investigated to scale the data volume beyond CPU memory. Fast interconnects provide GPUs access to these storage technologies, which present additional challenges due to their read and write characteristics. In contrast, disaggregated memory scales to petabytes of space as well, but might involve complex data access paths [19b; St21].

Fast Interconnect Technologies. In our dissertation, we have evaluated NVLink 2.0 on account of its commercial availability. However, fast interconnects from multiple hardware vendors are becoming available (e. g., CXL). Future work could evaluate and compare the upcoming fast interconnect technologies.

In conclusion, we expect that continued research will evolve GPU-enabled DBMSs towards fast interconnects, and thereby expand the scope of GPUs in data management.

References

[17a] Nvidia Tesla V100 GPU architecture, WP-08608-001_v1.1, Nvidia, 2017, https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf.

[17b] Transaction Processing Performance Council. TPC-H. 2017, http://www.tpc.org/tpch.

[19a] AMD EPYC CPUs, AMD Radeon Instinct GPUs and ROCm open source software to power world's fastest supercomputer at Oak Ridge National Laboratory, AMD, 2019, https://www.amd.com/en/press-releases/2019-05-07-amd-epyc-cpus-radeon-instinct-gpus-and-rocm-open-source-software-to-power.

[19b] Compute Express Link specification, Revision 1.1, CXL, 2019, https://www.computeexpresslink.org.

[20a] Compute Express Link specification, Revision 2.0, CXL, 2020, https://www.computeexpresslink.org.

[20b] Nvidia A100 tensore core GPU architecture, Version 1.0, Nvidia, 2020, https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf.

[21a] 3rd gen Intel Xeon Scalable processors product brief, Intel, 2021, https://www.intel.com/content/dam/www/public/us/en/documents/a1171486-icelake-productbrief-updates-r1v2.pdf.

[21b] AMD CDNA 2 architecture, AMD, 2021, https://www.amd.com/system/files/documents/amd-cdna2-white-paper.pdf.

[21c] AMD EPYC 7003 series processors, Revision LE-77202-00 02/21, AMD, 2021, https://www.amd.com/system/files/documents/amd-epyc-7003-series-datasheet.pdf.

[21d] Intel Xeon Gold 6338 processor, Intel, 2021, https://ark.intel.com/content/www/us/en/ark/products/212285/intel-xeon-gold-6338-processor-48m-cache-2-00-ghz.html.

[21e] Nvidia A100 tensore core GPU, Nvidia, 2021, https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet-us-nvidia-1758950-r4-web.pdf.

[22] Ampere Altra Max datasheet, Document issue 0.91, Ampere Computing, 2022, https://amperecomputing.com/wp-content/uploads/2022/01/Altra_Max_Rev_A1_DS_v0.91_20220113.pdf.

[Ab20] Abali, B. et al.: Data compression accelerator on IBM POWER9 and z15 processors. In: ISCA. IEEE, Washington, DC, USA, pp. 1–14, 2020.

[Am16] Amazon AWS: AWS Snowmobile — Migrate or transport exabyte-scale data sets into and out of AWS, 2016, https://aws.amazon.com/snowmobile.

[An13] Andrade, G. et al.: G-DBSCAN: A GPU accelerated algorithm for density-based clustering. Procedia Computer Science 18, pp. 369–378, 2013.

[Ar19] Arafa, M. et al.: Cascade Lake: Next generation Intel Xeon Scalable processor. IEEE Micro 39 (2), pp. 29–36, 2019.

[Ar20] Armbrust, M. et al.: Delta Lake: High-performance ACID table storage over cloud object stores. PVLDB 13 (12), pp. 3411–3424, 2020.

[As15] Ashari, A. et al.: On optimizing machine learning workloads via kernel fusion. In: PPoPP. ACM, New York, NY, USA, pp. 173–182, 2015.

[Ba18] Balkesen, C. et al.: RAPID: In-memory analytical query processing engine with extreme performance per watt. In: SIGMOD. ACM, New York, NY, USA, pp. 1407–1419, 2018.

[BB22] Boeschen, N.; Binnig, C.: GaccO — A GPU-accelerated OLTP DBMS. In: SIGMOD. ACM, New York, NY, USA, pp. 1003–1016, 2022.

[BC11] Borkar, S.; Chien, A. A.: The future of microprocessors. Commun. ACM 54 (5), pp. 67–77, 2011.

[BFT16] Breß, S.; Funke, H.; Teubner, J.: Robust query processing in co-processor-accelerated databases. In: SIGMOD. ACM, New York, NY, USA, pp. 1891–1906, 2016.

[BKY19] Boehm, M.; Kumar, A.; Yang, J.: Data management in machine learning systems. Morgan & Claypool Publishers, San Rafael, CA, USA, 2019.

[Bl21] BlazingSQL: BlazingSQL, 2021, https://blazingsql.com.

[Bl22] Blacher, M. et al.: Machine learning, linear algebra, and more: Is SQL all you need? In: CIDR. www.cidrdb.org, pp. 1–6, 2022, https://www.cidrdb.org/cidr2022/papers/p17-blacher.pdf.

[Bo16] Boehm, M. et al.: Declarative machine learning — A classification of basic properties and types, 2016, arXiv: 1605.05826 [cs.DB].

[Bo20] Boehm, M. et al.: SystemDS: A declarative machine learning system for the end-to-end data science lifecycle. In: CIDR. www.cidrdb.org, pp. 1–8, 2020, http://cidrdb.org/cidr2020/papers/p22-boehm-cidr20.pdf.

[Br14] Breß, S.: The design and implementation of CoGaDB: A column-oriented GPU-accelerated DBMS. Datenbank-Spektrum 14 (3), pp. 199–209, 2014.

[Br18] Breß, S. et al.: Generating custom code for efficient query execution on heterogeneous processors. VLDB J. 27 (6), pp. 797–822, 2018.

[Br20] Brown, T. et al.: Language models are few-shot learners. In: NeurIPS. Vol. 33, Curran Associates, Inc., Red Hook, NY, USA, pp. 1877–1901, 2020, https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[Br22] Brytlyt: BrytlytDB, 2022, https://www.brytlyt.com/what-we-do/brytlytdb.

[BS10] Bakkum, P.; Skadron, K.: Accelerating SQL database operations on a GPU with CUDA. In: GPGPU. Vol. 425, ACM, New York, NY, USA, pp. 94–103, 2010.

[BZN05] Boncz, P. A.; Zukowski, M.; Nes, N.: MonetDB/X100: Hyper-pipelining query execution. In: CIDR. www.cidrdb.org, pp. 225–237, 2005, http://cidrdb.org/cidr2005/papers/P19.pdf.

[Ca16] Caulfield, A. M. et al.: A cloud-scale acceleration architecture. In: MICRO. IEEE Computer Society, 7:1–7:13, 2016.

[CD97] Chaudhuri, S.; Dayal, U.: An overview of data warehousing and OLAP technology. SIGMOD Rec. 26 (1), pp. 65–74, 1997.

[CGF18] Choquette, J.; Giroux, O.; Foley, D.: Volta: Performance and programmability. IEEE Micro 38 (2), pp. 42–52, 2018.

[Ch19] Chrysogelos, P. et al.: HetExchange: Encapsulating heterogeneous CPU–GPU parallelism in JIT compiled engines. PVLDB 12 (5), pp. 544–556, 2019.

[Ch20] Chu, H. et al.: Empirical evaluation across multiple GPU-accelerated DBMSes. In: DaMoN. ACM, New York, NY, USA, 16:1–16:3, 2020.

[Cr15] Crotty, A. et al.: An architecture for compiling UDF-centric workflows. PVLDB 8 (12), pp. 1466–1477, 2015.

[CSA19] Chrysogelos, P.; Sioulas, P.; Ailamaki, A.: Hardware-conscious query processing in GPU-accelerated analytical engines. In: CIDR. www.cidrdb.org, pp. 1–9, 2019, http://cidrdb.org/cidr2019/papers/p127-chrysogelos-cidr19.pdf.

[CV21] Castriotta, A. G.; Volpi, F.: Copernicus Sentinel data access annual report Y2020, 2021, https://scihub.copernicus.eu/twiki/pub/SciHubWebPortal/AnnualReport2020/COPE-SERCO-RP-21-1141_-_Sentinel_Data_Access_Annual_Report_Y2020_final_v2.3.pdf.

[Da16] Dageville, B. et al.: The Snowflake elastic data warehouse. In: SIGMOD. ACM, New York, NY, USA, pp. 215–226, 2016.

[De17] Deloitte: Hitting the accelerator: the next generation of machine-learning chips, 2017, https://www2.deloitte.com/content/dam/Deloitte/global/Images/infographics/technologymediatelecommunications/gx-deloitte-tmt-2018-nextgen-machine-learning-report.pdf.

[De19] Devlin, J. et al.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: NAACL-HLT. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 4171–4186, 2019.

[Di18] van Dijk, E. L. et al.: The third revolution in sequencing technology. Trends in Genetics 34 (9), pp. 666–681, 2018.

[El03] Elkan, C.: Using the triangle inequality to accelerate k-means. In: ICML. AAAI Press, Menlo Park, CA, USA, pp. 147–153, 2003, https://www.aaai.org/Papers/ICML/2003/ICML03-022.pdf.

[Es11] Esmaeilzadeh, H. et al.: Dark silicon and the end of multicore scaling. In: ISCA. Pp. 365–376, 2011.

[FA19] FASTDATA.io: PlasmaENGINE, 2019, https://fastdata.io/plasma-engine.

[FS21] Fu, Y.; Soman, C.: Real-time data infrastructure at Uber. In: SIGMOD. ACM, New York, NY, USA, pp. 2503–2516, 2021.

[FT20] Funke, H.; Teubner, J.: Data-parallel query processing on non-uniform data. PVLDB 13 (6), pp. 884–897, 2020.

[Fu18] Funke, H. et al.: Pipelined query processing in coprocessor environments. In: SIGMOD. ACM, New York, NY, USA, pp. 1603–1618, 2018.

[Ga19] Gartner: Gartner Says the Future of the Database Market Is the Cloud, 2019, https://www.gartner.com/en/newsroom/press-releases/2019-07-01-gartner-says-the-future-of-the-database-market-is-the.

[GH11] Gregg, C.; Hazelwood, K. M.: Where is the data? Why you cannot debate CPU vs. GPU performance without the answer. In: ISPASS. IEEE Computer Society, Los Alamitos, CA, USA, pp. 134–144, 2011.

[Go04] Govindaraju, N. K. et al.: Fast computation of database operations using graphics processors. In: SIGMOD. ACM, New York, NY, USA, pp. 215–226, 2004.

[Go22] Google: How Google Search organizes information, 2022, https://www.google.com/intl/en/search/howsearchworks/how-search-works/organizing-information.

[Gu15] Gupta, A. et al.: Amazon Redshift and the case for simpler data warehouses. In: SIGMOD. ACM, New York, NY, USA, pp. 1917–1923, 2015.

[He08] He, B. et al.: Relational joins on graphics processors. In: SIGMOD. ACM, New York, NY, USA, pp. 511–524, 2008.

[He09] He, B. et al.: Relational query coprocessing on graphics processors. TODS 34 (4), 2009.

[He12] Hellerstein, J. M. et al.: The MADlib analytics library or MAD skills, the SQL. PVLDB 5 (12), pp. 1700–1711, 2012.

[He13] Heimel, M. et al.: Hardware-oblivious parallelism for in-memory column-stores. PVLDB 6 (9), pp. 709–720, 2013.

[He21] HeteroDB: PG-Strom, 2021, https://en.heterodb.com.

[He24] Hereth, N.: AWS and NVIDIA extend collaboration to advance generative AI innovation, Amazon, 2024, https://press.aboutamazon.com/2024/3/aws-and-nvidia-extend-collaboration-to-advance-generative-ai-innovation.

[HM12] Heimel, M.; Markl, V.: A first step towards GPU-assisted query optimization. In: ADMS. Pp. 33–44, 2012, http://www.adms-conf.org/heimel_adms12.pdf.

[Ho14] Horowitz, M.: Computing's energy problem (and what we can do about it). In: ISSCC. IEEE, Washington, DC, USA, pp. 10–14, 2014.

[HP17] Hennessy, J. L.; Patterson, D. A.: Thread-level parallelism. In: Computer architecture — A quantitative approach. 6th, Morgan Kaufmann, Cambridge, MA, USA, chap. 5, pp. 367–462, 2017.

[HP19] Hennessy, J. L.; Patterson, D. A.: A new golden age for computer architecture. Commun. ACM 62 (2), pp. 48–60, 2019.

[HTT09] Hey, T.; Tansley, S.; Tolle, K. M., eds.: The fourth paradigm: Data-intensive scientific discovery. Microsoft Research, Redmond, WA, USA, 2009.

[HY11] He, B.; Yu, J. X.: High-throughput transaction executions on graphics processors. PVLDB 4 (5), pp. 314–325, 2011.

[IB18] IBM POWER9 NPU team: Functionality and performance of NVLink with IBM POWER9 processors. IBM Journal of Research and Development 62 (4/5), 9:1–9:10, 2018.

[IKS20] István, Z.; Kara, K.; Sidler, D.: FPGA-accelerated analytics: From single nodes to clusters. Found. Trends Databases 9 (2), pp. 101–208, 2020.

[JC19] Johnston, D. B.; Caldwell, A.: Amazon Redshift reimagined: RA3 and AQUA, Amazon Web Services, 2019, https://d1.awsstatic.com/events/reinvent/2019/NEW_LAUNCH_Amazon_Redshift_reimagined_RA3_and_AQUA_ANT230.pdf.

[JDJ21] Johnson, J.; Douze, M.; Jégou, H.: Billion-scale similarity search with GPUs. IEEE Trans. Big Data 7 (3), pp. 535–547, 2021.

[Jo21] Jouppi, N. P. et al.: Ten lessons from three generations shaped Google's TPUv4i. In: ISCA. IEEE, pp. 1–14, 2021.

[KBY17] Kumar, A.; Boehm, M.; Yang, J.: Data management in machine learning: Challenges, techniques, and systems. In: SIGMOD. ACM, New York, NY, USA, pp. 1717–1722, 2017.

[Kh18] Khamis, M. A. et al.: AC/DC: In-database learning thunderstruck. In: DEEM. ACM, New York, NY, USA, 8:1–8:10, 2018.

[Ki22] Kinetica: Kinetica, 2022, https://www.kinetica.com.

[KK20] Krulis, M.; Kratochvíl, M.: Detailed analysis and optimization of CUDA k-means algorithm. In: ICPP. ACM, New York, NY, USA, 69:1–69:11, 2020.

[KLM21] Kumaigorodski, A.; Lutz, C.; Markl, V.: Fast CSV loading using GPUs and RDMA for in-memory data processing. In: BTW. Vol. P-311. LNI, Gesellschaft für Informatik, Bonn, Germany, pp. 19–38, 2021.

[KML15] Karnagel, T.; Müller, R.; Lohman, G. M.: Optimizing GPU-accelerated group-by and aggregation. In: ADMS. Pp. 13–24, 2015, http://www.adms-conf.org/2015/gpu-optimizer-camera-ready.pdf.

[KN11] Kemper, A.; Neumann, T.: HyPer: A hybrid OLTP&OLAP main memory database system based on virtual memory snapshots. In: ICDE. IEEE Computer Society, Los Alamitos, CA, USA, pp. 195–206, 2011.

[Ko16] Koliousis, A. et al.: SABER: Window-based hybrid stream processing for heterogeneous architectures. In: SIGMOD. ACM, New York, NY, USA, pp. 555–569, 2016.

[KS21] Kuusela, A.; Smullen, C.: Video coding unit (VCU). In: HCS. IEEE, Washington, DC, USA, pp. 1–30, 2021.

[Ku19] Kunft, A. et al.: An intermediate representation for optimizing machine learning pipelines. PVLDB 12 (11), pp. 1553–1567, 2019.

[La14] Lasi, H. et al.: Industry 4.0. Bus. Inf. Syst. Eng. 6 (4), pp. 239–242, 2014.

[Le21] Lee, R. et al.: The art of balance: A RateupDB experience of building a CPU/GPU hybrid database product. PVLDB 14 (12), pp. 2999–3013, 2021.

[Lo19] Lottarini, A. et al.: Master of none acceleration: A comparison of accelerator architectures for analytical query processing. In: ISCA. ACM, New York, NY, USA, pp. 762–773, 2019.

[Lu18a] Lutz, C. et al.: Efficient and scalable k-means on GPUs. Datenbank-Spektrum 18 (3), pp. 157–169, 2018.

[Lu18b] Lutz, C. et al.: Efficient k-means on GPUs. In: DaMoN. ACM, New York, NY, USA, 3:1–3:3, 2018.

[Lu20] Lutz, C. et al.: Pump up the volume: Processing large data on GPUs with fast interconnects. In: SIGMOD. ACM, New York, NY, USA, pp. 1633–1649, 2020.

[Lu22a] Lutz, C.: Scalable data management using GPUs with fast interconnects, PhD thesis, Berlin, Germany: TU Berlin, 2022.

[Lu22b] Lutz, C. et al.: Triton join: Efficiently scaling to a large join state on GPUs with fast interconnects. In: SIGMOD. ACM, New York, NY, USA, pp. 1017–1032, 2022.

[Ma20] Maximize Market Research: Global GPU database market, 2020, https://www.maximizemarketresearch.com/market-report/global-gpu-database-market/22455.

[Ma21] Market Data Forecast: GPU database market, 2021, https://www.marketdataforecast.com/market-reports/gpu-database-market.

[Me16] Meraji, S. et al.: Towards a hybrid design for fast query processing in DB2 with BLU acceleration using graphical processing units: A technology demonstration. In: SIGMOD. ACM, New York, NY, USA, pp. 1951–1960, 2016.

[Mi17] Milic, U. et al.: Beyond the socket: NUMA-aware GPUs. In: MICRO. IEEE/ACM, New York, NY, USA, pp. 123–135, 2017.

[Mi21] Michalke, A. et al.: An energy-efficient stream join for the Internet of Things. In: DaMoN. ACM, New York, NY, USA, 8:1–8:6, 2021.

[NEK18] Nguyen, A.; Edahiro, M.; Kato, S.: GPU-accelerated VoltDB: A case for indexed nested loop join. In: HPCS. IEEE, Washington, DC, USA, pp. 204–212, 2018.

[Ng19] Nguyen, G. et al.: Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: A survey. Artif. Intell. Rev. 52 (1), pp. 77–124, 2019.

[Nu24] Nugroho, D. P. A. et al.: Benchmarking stream join algorithms on GPUs: A framework and its application to the state-of-the-art. In: EDBT. OpenProceedings.org, Konstanz, Germany, pp. 188–200, 2024.

[Nv24a] Nvidia: Nvidia Blackwell architecture technical brief, Version 1.1, Nvidia, 2024, https://resources.nvidia.com/en-us-blackwell-architecture/blackwell-architecture-technical-brief.

[Nv24b] Nvidia: Nvidia GH200 Grace Hopper Superchip architecture, Version 1.21, Nvidia, 2024, https://resources.nvidia.com/en-us-grace-cpu/nvidia-grace-hopper.

[Om21] OmniSci: OmniSciDB, 2021, https://www.omnisci.com/platform/omniscidb.

[Pa17] Passing, L. et al.: SQL- and operator-centric data analytics in relational main-memory databases. In: EDBT. OpenProceedings.org, Konstanz, Germany, pp. 84–95, 2017.

[Pa20] Paul, J. et al.: Improving execution efficiency of just-in-time compilation based query processing on GPUs. PVLDB 14 (2), pp. 202–214, 2020.

[Pa21] Paul, J. et al.: MG-Join: A scalable join for massively parallel multi-GPU architectures. In: SIGMOD. ACM, New York, NY, USA, pp. 1413–1425, 2021.

[PHH16] Paul, J.; He, J.; He, B.: GPL: A GPU-based pipelined query processing engine. In: SIGMOD. ACM, New York, NY, USA, pp. 1935–1950, 2016.

[PMK14] Pirk, H.; Manegold, S.; Kersten, M. L.: Waste not. . . Efficient co-processing of relational data. In: ICDE. IEEE Computer Society, Los Alamitos, CA, USA, pp. 508–519, 2014.

[PMS15] Pirk, H.; Madden, S.; Stonebraker, M.: By their fruits shall ye know them: A data analyst's perspective on massively parallel system design. In: DaMoN. ACM, New York, NY, USA, 5:1–5:6, 2015.

[Ra20] Raza, A. et al.: GPU-accelerated data management under the test of time. In: CIDR. www.cidrdb.org, pp. 1–11, 2020, http://cidrdb.org/cidr2020/papers/p18-raza-cidr20.pdf.

[Ra21] Rateup: RateupDB, 2021, http://www.rateup.com.cn/dist/#/product.

[RBM22] Rosenfeld, V.; Breß, S.; Markl, V.: Query processing on heterogeneous CPU/GPU systems. ACM Comput. Surv. 55 (1), 2022.

[Re20] Reuther, A. et al.: Survey of machine learning accelerators. In: HPEC. IEEE, Washington, DC, USA, pp. 1–12, 2020.

[Rh18] Rhu, M. et al.: Compressing DMA engine: Leveraging activation sparsity for training deep neural networks. In: HPCA. IEEE Computer Society, Los Alamitos, CA, USA, pp. 78–91, 2018.

[RM16] Root, C.; Mostak, T.: MapD: A GPU-powered big data analytics and visualization platform. In: SIGGRAPH. ACM, New York, NY, USA, 73:1–73:2, 2016.

[Ro13] Rossbach, C. J. et al.: Dandelion: A compiler and runtime for heterogeneous systems. In: SOSP. ACM, New York, NY, USA, pp. 49–68, 2013.

[Ro19] Rosenfeld, V. et al.: Performance analysis and automatic tuning of hash aggregation on GPUs. In: DaMoN. ACM, New York, NY, USA, 8:1–8:11, 2019.

[RRW18] Roberts, S.; Ramanna, P.; Walthour, J.: AC922 data movement for CORAL. In: HPEC. IEEE, Washington, DC, USA, pp. 1–5, 2018.

[RT17] Rui, R.; Tu, Y.: Fast equi-join algorithms on GPUs: Design and implementation. In: SSDBM. ACM, New York, NY, USA, 17:1–17:12, 2017.

[Sa10] Satish, N. et al.: Fast sort on CPUs and GPUs: A case for bandwidth oblivious SIMD sort. In: SIGMOD. ACM, New York, NY, USA, pp. 351–362, 2010.

[Sc20] Schüle, M. E. et al.: Freedom for the SQL-lambda: Just-in-time-compiling user-injected functions in PostgreSQL. In: SSDBM. ACM, New York, NY, USA, pp. 1–12, 2020.

[Sc21] Schüle, M. E. et al.: In-database machine learning with SQL on GPUs. In: SSDBM. ACM, New York, NY, USA, pp. 25–36, 2021.

[Sh19] Shen, J. et al.: Introducing AresDB: Uber's GPU-powered open source, real-time analytics engine, 2019, https://eng.uber.com/aresdb.

[Sh20] Shanbhag, A. et al.: Large-scale in-memory analytics on Intel Optane DC persistent memory. In: DaMoN. ACM, New York, NY, USA, 4:1–4:8, 2020.

[Sh21] Shaw, D. E. et al.: Anton 3: Twenty microseconds of molecular dynamics simulation before lunch. In: SC. ACM, New York, NY, USA, 1:1–1:11, 2021.

[Si19] Sioulas, P. et al.: Hardware-conscious hash-joins on GPUs. In: ICDE. IEEE, Washington, DC, USA, pp. 698–709, 2019.

[SJ17] Stehle, E.; Jacobsen, H.: A memory bandwidth-efficient hybrid radix sort on GPUs. In: SIGMOD. ACM, New York, NY, USA, pp. 417–432, 2017.

[SMY20] Shanbhag, A.; Madden, S.; Yu, X.: A study of the fundamental performance characteristics of GPUs and CPUs for database analytics. In: SIGMOD. ACM, New York, NY, USA, pp. 1617–1632, 2020.

[Sp17] Sparks, E. R. et al.: KeystoneML: Optimizing pipelines for large-scale advanced analytics. In: ICDE. IEEE Computer Society, pp. 535–546, 2017.

[SQ22] SQream: SQream DB, 2022, https://sqream.com/product/data-acceleration-platform/sql-gpu-database.

[St18] Starke, W. J. et al.: IBM POWER9 memory architectures for optimized systems. IBM Journal of Research and Development 62 (4/5), 3:1–3:13, 2018.

[St21] Starke, W. J. et al.: IBM's POWER10 processor. IEEE Micro 41 (2), pp. 7–14, 2021.

[Su20] Sun, Y. et al.: Summarizing CPU and GPU design trends with product data, 2020, arXiv: 1911.11313v2 [cs.DC].

[Sv21] Svedin, M. et al.: Benchmarking the Nvidia GPU lineage: From early K80 to modern A100 with asynchronous memory transfers. In: HEART. ACM, New York, NY, USA, 9:1–9:6, 2021.

[Th24] Thiagarajan, M.: Announcing world's largest, first zettascale AI supercomputer in the cloud, Oracle, 2024, https://blogs.oracle.com/cloud-infrastructure/post/worlds-largest-ai-supercomputer-in-the-cloud.

[To18] Tomé, D. G. et al.: Optimizing group-by and aggregation using GPU-CPU co-processing. In: ADMS. Pp. 1–10, 2018, http://www.adms-conf.org/2018-camera-ready/tome_groupby.pdf.

[To24] Top500: Top500 Highlights, 2024, https://www.top500.org/lists/top500/2024/11/highs/.

[Ve24] Vegas, M.: Microsoft adopts NVIDIA Blackwell to power the next frontier of AI supercomputing, Microsoft, 2024, https://techcommunity.microsoft.com/blog/azurehighperformancecomputingblog/microsoft-adopts-nvidia-blackwell-to-power-the-next-frontier-of-ai-supercomputin/4303541.

[Wu14] Wu, L. et al.: Q100: The architecture and design of a database processing unit. In: ASPLOS. ACM, New York, NY, USA, pp. 255–268, 2014.

[Xi18] Xin, D. et al.: Helix: Holistic optimization for accelerating iterative machine learning. PVLDB 12 (4), pp. 446–460, 2018.

[Ya17] Yabuta, M. et al.: Relational joins on GPUs: A closer look. IEEE Trans. Parallel Distrib. Syst. 28 (9), pp. 2663–2673, 2017.

[YLZ13] Yuan, Y.; Lee, R.; Zhang, X.: The yin and yang of processing data warehousing queries on GPU devices. PVLDB 6 (10), pp. 817–828, 2013.

[Ze19] Zeuch, S. et al.: Analyzing efficient stream processing on modern hardware. PVLDB 12 (5), pp. 516–530, 2019.

[Zh21] Zhang, Y. et al.: Distributed deep learning on data systems: A comparative analysis of approaches. PVLDB 14 (10), pp. 1769–1782, 2021.

[Zi20] Zion Market Research: GPU database market — Global industry analysis, 2020, https://www.zionmarketresearch.com/report/gpu-database-market.