Report from Dagstuhl Seminar 24162

Hardware Support for Cloud Database Systems in the Post-Moore's Law Era

David F. Bacon[*1], **Carsten Binnig**[*2], **David Patterson**[*3], **and Margo Seltzer**[*4]

1 **Google – New York, US.** `dfb@google.com`
2 **TU Darmstadt, DE.** `carsten.binnig@cs.tu-darmstadt.de`
3 **University of California – Berkeley, US.** `pattrsn@cs.berkeley.edu`
4 **University of British Columbia – Vancouver, CA.** `mseltzer@cs.ubc.ca`

Abstract

The end of scaling from Moore's and Dennard's laws has greatly slowed improvements in CPU speed, RAM capacity, and disk/flash capacity. Meanwhile, cloud database systems, which are the backbone for many large-scale services and applications in the cloud, are continuing to grow exponentially. For example, most of Google's products that run on the Spanner database have more than a billion users and are continuously growing. Moreover, the growth in data also shows no signs of slowing down, with further orders-of-magnitude increases likely, due to autonomous vehicles, the internet-of-things, and human-driven data creation. Meanwhile, machine learning creates an appetite for data that also needs to be preprocessed using scalable cloud database systems. As a result, cloud database systems are facing a fundamental scalability wall on how to further support this exponential growth given the stagnation in hardware.

While database research has a long tradition of investigating how modern hardware can be leveraged to improve overall system performance – which is also shown by the series of past Dagstuhl Seminars – a more holistic view is required to address the imminent exponential scalability challenge that databases will be facing. However, applying hardware accelerators in a database needs a careful design. In fact, so far, no commercial system has applied hardware accelerators at scale. Unlike other hyper-scale applications such as machine learning training and video processing where accelerators such as GPUs and TPUs circumvent this problem, workloads in cloud database systems are typically not compute-bound and thus benefit less or not at all from such existing accelerators. This Dagstuhl Seminar thus aimed to bring together leading researchers and practitioners from database systems, hardware architecture, and storage systems to rethink, from the ground up, how to co-design database systems and compute/storage hardware. By uniting experts across these disciplines, the seminar sought to identify the architectural changes and system designs that could enable the order-of-magnitude improvements required for the next generation of applications.

Seminar April 14–19, 2024 – https://www.dagstuhl.de/24162
2012 ACM Subject Classification Information systems → Data management systems; Hardware
Keywords and phrases Databases, Modern Hardware, Cloud
Digital Object Identifier 10.4230/DagRep.14.4.54

* Editor / Organizer

1 Executive Summary

David F. Bacon (Google – Seattle, US, dfb@google.com)
Carsten Binnig (TU Darmstadt, DE, carsten.binnig@cs.tu-darmstadt.de)
David Patterson (University of California – Berkeley, US, pattrsn@cs.berkeley.edu)
Margo Seltzer (University of British Columbia – Vancouver, CA, mseltzer@cs.ubc.ca)

This Dagstuhl Seminar on the Future of Cloud Database Systems was convened to address the pressing challenges arising from the stagnation in hardware performance gains, historically driven by Moore's and Dennard's laws. As data continues to grow exponentially – propelled by the expansion of autonomous systems, the Internet of Things (IoT), and machine learning – there is an urgent need to rethink the co-design of database systems and hardware. This seminar brought together experts from database systems, hardware architecture, and storage systems to explore innovative approaches to overcoming these scalability bottlenecks and envisioning the future of cloud database systems.

A central theme of the seminar was the growing disconnect between the exponential increase in data and the slowing pace of hardware improvements, leading to what participants referred to as a "scalability wall." Addressing this challenge requires groundbreaking architectural changes in cloud database systems to support the next generation of applications. One significant area of focus was the potential role of AI-driven hardware and software in reshaping database management systems (DBMS). Participants explored whether AI hardware, such as GPUs and TPUs, could be adapted for database workloads, which traditionally are not compute-bound. Additionally, the concept of leveraging large language models (LLMs) as a new paradigm for databases was discussed, prompting further considerations of the future interplay between AI and DBMS.

To kickstart these discussions, several invited impulse talks were presented, each designed to set the stage for the working groups by exploring possible future scenarios for cloud database systems:

1. **AI Rules:** This talk examined a future where AI hardware and software dominate data centers, fundamentally altering the design and function of DBMS. The discussion centered on how DBMSs might need to evolve in a world where AI is integral to data processing and whether an LLM could serve as a database.
2. **A Disaggregated Future:** This presentations offered a perspective on a future where heterogeneous devices (compute, memory, storage) are connected via ultra-fast networks, creating a fully disaggregated cloud infrastructure. The talk prompted discussions on how DBMS could adapt to and thrive in such an environment.
3. **A Fully Reprogrammable Future:** The talk on this future envisioned a future where all hardware is reprogrammable and customizable at runtime, drastically changing how data processing and storage are handled. The implications for DBMS in such a highly flexible hardware environment were critically examined.
4. **The Pipe Dream:** This session explored the idea of "dreaming up" new DBMS hardware, revisiting the concept of a dedicated database machine. The discussion focused on whether this approach, which has failed in the past, could succeed in the context of modern cloud environments.

Following these impulse talks, the seminar divided into working groups to delve deeper into specific challenges:

1. **Working Group 1:** The Next Order of Magnitude focused on how database technologies can evolve to achieve order-of-magnitude improvements in performance, despite the slowdown in hardware advancements. This group was particularly concerned with managing the exponential growth of unstructured data feeding machine learning models.
2. **Working Group 2:** Memory-Centric DBMS Design advocated for a shift from processor-centric to memory-centric designs, emphasizing the optimization of data access in cloud environments as a solution to the performance bottlenecks caused by traditional architectural models.
3. **Working Group 3:** AI Hardware for Databases investigated how emerging AI hardware, like GPUs and TPUs, could be leveraged for cloud DBMS, even though database workloads typically do not benefit as much from compute-bound acceleration as other applications do.
4. **Working Group 4:** The last working group explored taking disaggregation to the extreme and considering its impact on systems for cloud DBMSs.

As the seminar progressed, participants emphasized the importance of cross-disciplinary collaboration and knowledge sharing. They worked together to draft a comprehensive paper for publication, summarizing the insights and innovations discussed. The seminar concluded with a focus on the need for continued innovation in both hardware and software to meet the demands of future cloud database systems.

In summary, the Dagstuhl Seminar provided a crucial platform for reimagining the future of cloud database systems in light of hardware stagnation. By bringing together leading experts from multiple disciplines and sparking deep discussions through targeted impulse talks, the seminar laid the groundwork for the architectural and system-level innovations necessary to overcome the scalability challenges posed by exponential data growth. The insights and collaborative efforts from this seminar will be instrumental in guiding the development of next-generation database systems.

2 Table of Contents

3 Overview of Impulse Talks

In the following, we provide the information about the (stage setting) impulse talks. While the initial talks had the goal to connect communities by providing the state-of-the-art regarding hardware and cloud databases, the other impulse talks were motivating possible futures how hardware and databases might evolve.

3.1 Computer Architecture 101

David A. Patterson (University of California – Berkeley, US)

The slowing of Moore's Law and the lack of new big ideas to improve CPUs mean slow improvement in general purpose computing in data centers. Innovations in packaging such as chiplets and high bandwidth memories help, but they do not restore the doubling of performance every 18 months that we enjoyed previously. Today the path to much faster general purpose computing that lifts all boats requires scaling out to more computers and more data centers. Similarly, the end of Dennard Scaling means faster computation uses more power. The maximum power of data centers CPUs and GPUs has risen from 300W, which could be air cooled, to 700W and 1200W, respectively, which requires liquid cooling. Domain Specific Architectures (DSAs) are the only path left for big gains in performance. Nevertheless, DSAs must follow Amdahl's Law: performance improvement to be gained from using a faster mode of execution is limited by the fraction of the time the faster mode can be used. Given the high expense to develop new hardware, which domains merit DSAs?

Deep Neural Networks (DNNs) easily justify their own chips and supercomputers. Through boldness and good fortune, NVIDIA dominates commercial DSAs for DNNs with an unconventional architecture and new programming language (GPU/SIMT and CUDA), although most hyperscalers also have their internal inhouse alternatives. DNN DSAs have leveraged on fast matrix multiply, high memory bandwidth, and new narrow data types. We have likely reached a historic tipping point in data center hardware. From the 1960s to the 2010s, conventional software was king and Moore's law still held, so CPUs dominated the hardware investment and deployment. Looking forward, Moore's Law is slowing and AI software now holds the throne, so we expect DNN DSAs will be the majority of investment and deployment.

3.2 Some Hardware Impacts on Cloud Databases

Mark D. Hill (University of Wisconsin-Madison, US)

This talk addresses three hardware impacts on cloud database systems. First, using accelerators will be necessary to get more rapid performance improvement than will be possible with slowly improving CPUs. To this end, I recommend that innovation first target existing accelerators like GPUs, TPUs, SmartNICs, data movers, encryption, and compression. This avoids the substantial investment that new accelerators require. Second, memory is and will likely be the bottleneck for many computations. With general purpose CPUs, database work should consider exploiting Compute eXpress Link (CXL) that enables more more memory to be attached to CPUs than possible with directly-attached DDR memory alone. For GPUs, high-bandwidth memory (HBM) will provide high bandwidth, but low capacity, leaving the challenge for how the next memory tier should evolve and be exploited. Third, the public cloud is evolving toward confidential compute wherein tenant data is protected cryptographically end-to-end. New database ideas and accelerators must be compatible with confidential compute to impact the public cloud.

3.3 Cloud Databases (OLTP) – Where are we and where are we going?

David F. Bacon (Google – New York, US)

There are two fundamental types of databases: operational (OLTP) and analytic (OLAP). My presentation focuses on Spanner, Google's largest OLTP database; Justin Levandoski will cover BigQuery, which is our analytic database, in the next session. Since this is an "opinionated overview" and I haven't worked on other Cloud database systems, I'll invite others in the audience to contribute details where other systems are different. Spanner stores 15 EiB of data and runs at over 4 billion QPS. Its data under management has been doubling roughly every year. Given the hardware trends described earlier in the talks by Dave Patterson and Mark Hill, we are at a crossroads where it will either cost dramatically more to store data each year, or we will have to dramatically slow the growth in data storage, or we will have to find new ways to storing and operating on data efficiently. As a co-organizer of this event, it was solving this problem that motivated me to help bring us together.

Strong Consistency Wins. Spanner is now used to store the data for virtually all of Google's largest consumer products, including Search, YouTube, Gmail, Drive, Photos, Maps, Meet, and so on. It also stores much of the data for the Google Cloud control plane and for Google's internal infrastructure. Finally, it is offered as a product as part of Google Cloud. Cloud Spanner is used by Uber, Walmart, Ford, and others.

Google was instrumental in launching the "NoSQL" movement for building its hyper-scale applications. A fundamental part of NoSQL (although the term has become somewhat fuzzy over time) was using relaxed consistency as a way of achieving scale. While this may still be true in parts of the industry, at Google we have come full circle: Spanner has solved the problems of strong consistency at scale. Meanwhile, we have repeatedly had the experience that building products without strong consistency may work well in the beginning, but as products evolve, add more features, and become more heavily relied upon by the public, that weak consistency becomes an Achilles heel. Each application wound up trying to hide weak consistency from users, and building custom protocols which are often ad-hoc and error-prone.

In 1976, when System R was introduced with the relational model, SQL was running at perhaps 20 QPS on 60 MB of data. Since then SQL has scaled by 11 decimal orders of magnitude in storage and 8 orders of magnitude in QPS. This is a testament to the remarkable power of its declarative programming model and its ability to express massive parallelism in an architecture-independent way.

The Scaling Wall? However, with the end of Moore and Dennard scaling, exponential data growth would cause almost exponential increase in cost, which is untenable. However, data growth shows no sign of slowing down. In fact, LLMs and related technologies seem to be accelerating data growth. At its current growth rate, Spanner could reach a zettabyte (1000 exabytes) by 2030, with an aggregate of 1 trillion QPS. Meanwhile, data is getting colder, on a per-byte basis. Spanner's storage is growing substantially faster than its QPS and CPU consumption. While more and more IOPs are moving to SSD, HDD IOPs are an increasingly large part of the total system cost.

No Silver Bullet. Unfortunately, there is no clear answer to these problems. Database systems are notoriously diverse in their computational load. Spanner's hottest function is roughly 2% of total CPU time, so there are no "kernels" amenable to hardware optimization. The biggest opportunity lies in data compression. Compression is a modest amount of total CPU time, but that is due to the fact that better compression is too expensive in software. If we had cheap compression available, we could compress more aggressively. The biggest savings would come not from the CPU time, but from the savings in HDD, SSD, and IOPs. Total system optimizations to make more effective use of the hardware, either by improving CPI or by reducing power consumption, are also fruitful areas for exploration. The best hope for hardware acceleration actually lies in creating new database paradigms and workloads, which have more computational density. While it isn't yet clear how, AI will clearly play a large role here.

Reliability Challenges at Exascale. Google has previously reported that a small but significant number of cores occasionally performs incorrect computation. Unlike memory or disk corruption, where there are long-standing mechanisms to detect and prevent corruption, we do not have experience with corruptions in the computational path. These corrupted computations can and do lead to corrupted data, and are therefore of enormous concern for Spanner. As we move towards exascale databases, we will need to tame this problem if we are to maintain the data integrity guarantees that our users expect.

3.4 Cloud Databases (OLAP) – Where are we and where are we going?

Justin Levandoski (Google – Seattle, US)

This talk provided an overview of current cloud-native analytics system architectures (e.g., Amazon Redshift [2], Snowflake [10], Google BigQuery [22]) that separate compute and storage. It then focused on the high-level architecture of BigQuery that also disaggregates memory for intermediate shuffle. This serverless and disaggregated design has several benefits, as it (1) allows for on-demand *scaling* of each resource, (2) allows for on-demand *sharing* of resources, and (3) adapts well to *multi-tenant* USge at *lower cost* – all of which is important to running a cloud data service at scale.

While the benefits and flexibility of compute/storage separate are well known, this talk reiterated the benefit of disaggregated memory for shuffle (also covered in [22]), which (1) reduced shuffle latency by and *order-of-magnitude*, (2) enabled *order-of-magnitude* larger shuffles, (3) reduced resource cost by 20% by allowing the system to avoid resource fragmentation, stranding, poor isolation of memory resources. This talk ended by introducing new workloads on the horizon for cloud data warehousing, focusing on unstructured data becoming a first-class citizen in these systems and the overall trend of traditional data warehouses becoming general purpose cloud data platforms for all data types [19].

3.5 The AI Future: ML for Systems

Tim Kraska (MIT – Cambridge, US)

Machine learning (ML) and Generative AI (GAI) is changing the way we build, operate, and use data systems. For example, ML-enhanced algorithms, such as learned scheduling algorithms and indexes/storage layouts are being deployed in commercial data services, GAI-code assistant help to more quickly develop features, ML-based techniques simplify operations by automatically tuning system knobs, and GenAI-based assistants help to debug operational issues. Most importantly though, Generative AI is reshaping the way users interact with data systems. Even today, all leading cloud providers already offer natural language to SQL (NL2SQL) features as part of their Python Notebook or SQL editors to increase the productivity of analysts. Business-line users are starting to use natural language as part of their visualization platforms or enterprise search, whereas application developers are exploring new ways to expose (structured) data as part of their GAI-based experiences using RAG and other techniques. Some even go so far and say that "English will become the new SQL" despite the obvious challenges that English is often more ambiguous. Arguably, industry is leading many of these efforts and they are happening at unprecedented speed – almost every week there is a new product announcement. Yet, a lot of the work feels ad-hoc and many challenges remain to make ML/GAI for systems in all these areas really practical despite all the product announcements. In this talk, I provide an overview of some of these recent developments and outline how the academic solution often differs from the ones deployed in industry. Finally, I list several opportunities for academia to not only contribute but also build a better, more grounded foundation.

3.6 The AI Future: Do we need Databases at all? Or Model = DB?

Carsten Binnig (TU Darmstadt, DE)

In recent years, the DBMS community has outlined a new direction of so-called learned DBMS components, where core components such as indexes or query optimizers are replaced by machine learning (ML) models. In this talk, I conducted a (somewhat extreme) thought experiment and asked the question: "Can we replace the entire DBMS with an ML model" or in short "DBMS = ML model". This direction is particularly interesting as "DBMS = ML model" would allow us to run DBMS natively on AI hardware and leverage advances of AI hardware in which massive investments are being made today. In addition, recent results on learned DBMS components have shown that they can significantly improve DBMS performance. However, it is still far from clear whether a complete DBMS can be replaced by a single ML model or whether this is simply impossible. Wait, is it really unclear whether "DBMS = ML model" can be true?

In my talk I implied that there is (at least) hope that "DBMS = ML model" can be true. For example, if we look at LLMs today, we can see that LLMs are already being used as a type of database because they support question-answering on external data sources, including tables, when used in combination with retrieval in what is called retrieval-augmented generation. Beyond the fact that we can thus use AI hardware as discussed above, using LLMs as DBMSs offers many other possibilities for modern DBMS workloads which need to deal with multimodal data (images and text). However, LLMs have many known weaknesses such as hallucinations and other issues that need to be addressed in order to utilize them for query answering and act as a replacement for DBMSs.

3.7 The AI Future: Where is AI HW going?

Holger Fröning (Universität Heidelberg – Mannheim, DE)

Graphics Processing Units (GPUs) and Deep Neural Networks (DNNs) synergize to advance computational capabilities. GPUs, originally for graphics rendering, accelerate DNN training and inference with parallel processing. DNNs' demanding computations benefit from GPUs' parallel architecture, driving efficiency and speed. As DNN complexity grows, so does the demand for more powerful GPUs, spurring GPU advancements. In turn, GPU evolution fuels DNN innovation, enabling breakthroughs in computer vision, natural language processing, and autonomous systems. This reciprocal relationship propels technological progress, shaping the future of AI and computational sciences.

In this light, this talk will review fundamentals of CMOS technology scaling, in particular considering the end of Dennard scaling. As a result, it is anticipated that overall performance in terms of operations per second is rather governed by power consumption budget and energy efficiency in operations per Joule. Furthermore, technology analysis suggests that data movements are more expensive than computations.

The talk will conclude with a couple of research directions in the light of these observations.

3.8 The AI Future: AI rules (NOT)? Real-Time Intelligent Systems

Anastasia Ailamaki (EPFL – LaUSnne, CH)

Database systems face new hurdles with the rise of diverse hardware infrastructures and varying workloads. Evolving hardware, with its mix of different components and dynamic configurations, complicates how data moves within systems, and affects performance and robustness. At the same time, the surge in data-centric analytics and powerful AI models brings a wide range of workloads that systems must handle. The old way of designing systems based on predicted performance no longer works, leading to inefficiencies.

This talk analyzes the complexity of heterogeneous hardware and diverse workloads in database systems. It highlights the limitations of standard approaches and stresses the need for systems that can adapt without relying on fixed assumptions about hardware or workloads. The future is **real-time intelligent systems**, i.e., systems that adapt to changes in their execution environment in intelligent ways using ML, GenAI, abstraction and just-in-time code generation. The goal is modular, composable infrastructures which enable cross-optimizations dynamically, while preserving separation of concerns in system design.

3.9 A Disaggregated Heterogeneous Future: An Overview

Gustavo Alonso (ETH Zürich, CH)

The trend towards hardware specialization and disaggregation raises the question of how data processing engines will be take advantage of these developments or, at the least, adapt to them. In the talk, I point out that the biggest bottleneck in data processing is the data movement and new hardware can be used to turn the data path from storage to processing units into a series of active components that filter, reduce, transform, and pre-process the data. I give an example of how this can be implemented using smart storage (e.g., to project data out), a smart NIC on the storage system (e.g., to filter the data further), a controller in disaggregated CXL memory (e.g., performing an initial hashing of the data), a smart NIC on the computing node (e.g., decompression and decrypting the data), and an accelerator between memory and caches (e.g., hashing the data to partition the data across different processors). Such a pipeline of processing elements is feasible today and can be used as an experimental platform to inform hardware evolution and better understand how near-data-processing can be orchestrated to achieve the biggest possible gains.

3.10 A Disaggregated Heterogeneous Future: Building Cloud-native Data Systems for the Post-Moore Era

Jana Giceva (TU München – Garching, DE)

Considering the disaggregated cloud environment, there are many open questions on how to build accelerators for data intensive applications and the impact resource disaggregation may have on the whole system stack. In this talk I touch upon a few directions that are worthwhile discussing in this context and exploring further in the spirit of Dagstuhl. In the first part I introduce the idea of using operator primitives as a building block both for expressing various types of dataflows (beyond relational) and for enabling hardware acceleration across the data-path in the era of resource disaggregation and active hardware components. In the second part I discuss the systems challenges and opportunities for adopting such primitives in practice. For example, one idea is to start treating databases as domain specific compilers, so we can transform optimizations of the logical plan into compiler passes, before generating a physical plan (DAG) of tasks with binaries suitable to the target environment. Nevertheless, for the primitives to be fully adopted we need to find a way to address the heterogeneity of the underlying memory- and compute-models, the need for data transformations, and virtualizing the resources in the cloud context. And finally, in the third part I propose using a memory-centric view of the system as a programming model for fully disaggregated system.

3.11 A fully (Re-)Programmable Future: Cloud Databases and Hardware

Zsolt István (TU Darmstadt, DE)

In modern clouds, Resource Disaggregation has been adopted as a way of offering scalability and efficient resource utilization for large-scale applications. Provisioning CPU, memory, and storage resources independently for distributed data-intensive applications is a great enabler and cloud databases are already designed with disaggregation in mind. In this talk, we focus on an exciting opportunity that emerges in this context, namely, to dramatically increase the efficiency of cloud databases through the use the specialized and programmable hardware devices that already underpin resource disaggregation. To take advantage of this opportunity, however, we need to overcome several challenges. First, we need to find practical ways of co-designing databases and the offloaded hardware functionality. Second, programming such devices must become easier, to be able to achieve good performance without having to entirely re-design operators. In this talk I sampled from relevant related work and from our early results on overcoming these two challenges, getting us closer to a fully programmable future for cloud databases.

3.12 The Pipe Dream: Database Systems Chasing Hardware

Jignesh M. Patel (Carnegie Mellon University – Pittsburgh, US)

New hardware is typically designed to support new essential applications that are poorly served by existing commodity hardware or to address fundamental architectural constraints. Database systems are modular in their internal organization, allowing them to adapt to new hardware in ways other software applications generally cannot. Thus, database applications haven't become critical motivators for new hardware, and this trend will continue. To get high performance, database systems must adapt in two primary ways. First, methods that reduce data movement will be even more critical. This was the primary reason analytic database systems went from row to column stores. We can further slice columns by bits and develop bit stores to reduce data movement even more. The second observation is that AI will drive the development of new hardware and that database systems have no choice but to adapt to run efficiently on this new hardware, which includes GPUs today. Fortunately, this seems possible, given the modular abstractions in database systems. Thus, the future of high-performance database systems is in developing methods that intrinsically reduce data movement and enable them to run efficiently on available diverse commodity hardware.

3.13 The Pipe Dream: Hardware Acceleration For Databases

Lisa Wu Wills (Duke University – Durham, US)

In this talk, we ask the question of whether it is possible to accelerate databases by having a true hardware-software co-design. Surveying the hardware development landscape, hardware specialization is motivated by more efficient processing where efficiency is defined as higher performance, lower power, lower energy, and therefore lower total cost of ownership. We introduce the concept of using datatype acceleration to raise the level of abstraction when designing hardware and leveraging already-defined software method calls and containers to provide more efficiency. We showed a classic example of Q100 accelerating databases achieving one to two orders of magnitude of performance and energy efficiency using heterogeneous functional tiles and exploiting pipeline and data parallelism. We then pose three possible futures for exploration: 1) replacing some worthwhile software operations with hardware primitives, 2) using an FPGA to provide fused specialized units for common database operations, and 3) providing hardware support for primitives used in a data-centric computing world.

4 Working Group 1: The Next Order of Magnitude

Gustavo Alonso (ETH Zürich, CH, alonso@inf.ethz.ch)
Carsten Binnig (TU Darmstadt, DE, carsten.binnig@cs.tu-darmstadt.de)
Mark Hill (University of Wisconsin-Madison, US, markhill@cs.wisc.edu)
Ihab F. Ilyas (University of Waterloo, CA, ilyas@uwaterloo.ca)
Justin Levandoski (Google – Settle, US, levandoski@google.com)
Jignesh M. Patel (Carnegie Mellon University – Pittsburgh, US, jignesh@cmu.edu)
Holger Pirk (Imperial College, London, UK, pirk@imperial.ac.uk)
Tobias Ziegler (TU Darmstadt, DE, tobias.ziegler@cs.tu-darmstadt.de)

Data growth continues being exponential[1], especially regarding the unstructured data feeding machine learning and large language models. In the past, hardware advancements enabled keeping pace with this trend. However, with the slowing of Moore's Law [28], managing large data volumes solely through hardware improvements has become increasingly challenging. This raises a critical question: if data continues to grow exponentially, how can we evolve database technologies to achieve order-of-magnitude improvements in performance?

Data workload growth and addressing *Hill's law*

At a high level, database workloads can be divided into (1) *Online transaction processing (OLTP)* that handles transactional/operational workloads for a system-of-record and (2) *Online analytics processing (OLAP)* for large-scale data analytics and enterprise business

[1] `https://www.statista.com/statistics/871513/worldwide-data-created/`

reporting. We argue that OLTP workloads are unlikely to experience exponential growth. Several of the authors have spent decades in industry building and operating cloud-based transaction processing systems. For the vast majority of these workloads (the 99.999%), a single large machine typically suffices. Furthermore, transaction processing workloads easily partition (e.g., by use case) and can thus horizontally scale without needing to coordinate cross-partition transactions.

Conversely, analytics workloads are experiencing a renaissance in both data volume growth and workload types. Traditionally OLAP workloads dealt with precise structured tabular data that is aggregated and fed into business intelligence/reporting applications. While valuable, these "traditional" analytics/BI workloads are expected to grow modestly at 20-30% year-over-year in line with the current cloud data warehousing market. A key exponential growth area for OLAP will be unstructured data and the new workload types it will bring about at the intersection of AI/ML and analytics. There are three key trends driving this growth:

1. **Unstructured data growth and collection in the enterprise.** With the advent of cheap cloud object storage, it has become economical for enterprises to store unstructured data (e.g., pdfs, video, audio, images) in raw form. As one data point, the IDC expects 80% data to be unstructured by 2025[2], and is driving new use cases for analytics on images, audio, speech and text.
2. **Cloud data warehouse architectural shifts and customer expectations.** The data analytics industry is shifting toward a so-called "lakehouse" approach to data management, whereby data warehouses evolve into general-purpose data orchestration platforms for all data types. This architectural shift separates storage and compute. Data is now stored in a scalable storage layer and the data warehouse software provides the compute layer orchestrating IO and computing on that data as needed. This architecture can be generalized to handle all kinds of data including non-tabular unstructured data. This architecture meets customer expections of a single system (or the illusion of a single system) to seamlessly handle both traditional data warehousing and advanced analytics use cases [2, 19].
3. **Democratization of in AI/ML inference/extraction.** Powerful LLM capabilities and advances in AI/ML inference techniques have democratized the ability to extract valuable enterprise data from unstructured data (e.g., audio, video, images, pdfs). Hence, more structured, but less accurate, tabular data will be available to OLAP workloads based on these techniques.

This new reality will affect OLAP workloads in three ways. First, data curation and cleaning workloads will incur a significant jump in both compute and storage budget. Second, the volume of structured data generated from inference over unstructured data will grow exponentially compared to that traditionally ingested from structured/tabular sources. This structured extraction will take place directly within the analytics platform, as unstructured data has become a first-class citizen in these systems, e.g., through BigQuery object tables [19] or Snowflake directory tables[3]. This functionality gives rise to a new workload that is expected to dominate the storage and compute cost of modern OLAP stacks. Last but not least, we will soon embed this multi-modal data as large vector stores to enable features such as dense-retrieval, clustering and serving downstream models. It is probably too early to tell

[2] `https://solutionsreview.com/data-management/80-percent-of-your-data-will-be-unstructured-in-five-years`
[3] `https://docs.snowflake.com/en/user-guide/data-load-dirtables`

how these vector stores will change the overall data growth, but it definitely calls for a new scaling paradigm since data growth cannot simply be handled by hardware advancements anymore. We will call this observation *Hill's law.*

Hill's Law: The exponentially widening disparity between accelerating data growth and modest hardware improvements must be covered by exponentially better data reduction techniques.

Will hardware improvements be sufficient to handle data workload growth?

The answer is a qualified "no:" Hardware improvements may be able to cover the growth of OLTP and structured OLAP workloads, but will be woefully insufficient for straightforward handling of the exploding growth of exploratory OLAP processing of unstructured data.

A key hardware impact on data and other workloads occurs because some hardware parameters scale at different rates that others. One must pay particular attention to this now as 2D transistor scaling slows (Moore's Law).

- 2D logic scaling of general purpose cores on SOC package is slowing but will still proceed faster than scaling of DDR memory bandwidth off package and memory capacity per chip.
- Compute eXpress Link (CXL) will allow more memory bandwidth and capacity but at substantial cost. CXL also enables the hardware system designer to flexibility allocate a package's "lanes" to memory, I/O (PCIe), or accelerators.
- High-bandwidth memory (HBM) will continue to provide GPUs (and others) high bandwidth at high cost, but with limited capacity.
- SSD capacity will grow well due to its monolithic 3D implementation while its bandwidth growth will follow PCIe growth.
- Hard disk drive capacity and bandwidth will likely grow very slowly.
- Optical interconnect use will move closer to computing systems and will eventually terminate on SOC packages ("co-packaged" optics). This will unlock more bandwidth that can reach further, e.g., for disaggregation.

These disparate scaling trends will encourage hardware systems that carefully husband bandwidth (memory and I/O) and capacity (memory and hard drive) more than other resources. In the extreme, one can model computation as free.

With careful design, we expect hardware improvements may be more or less sufficient to cover the relatively slow growth of OLTP and Structured OLAP. However, substantial innovation in algorithms, software, and (co-designed) hardware will be needed to support exploding OLAP processing on exploding unstructured data.

First order principles to address Hill's Law

For cloud data platforms, the hardware developments outlined before and in particular the stagnation regarding bandwidth (both SSD and memory bandwidth) as well as memory capacity will have a significant impact on analytical data platform where data is growing at fast rates. In the following, we discuss first order principles that will help future cloud DBMS to get around these hardware limitations.

(1) Increase information density. A first principle that will help us to support the future growth in data without sacrificing performance is the principle that we need to increase information density per bit (IDB). An important aspect is that the information density needs to be increased along the full data access path from storage over memory until data hits the compute. This will have two important consequences: First, when we need to move data along the hierarchy from storage over memory to compute, we can significantly reduce the footprint of the data movement, which means we can "move more for less". Moreover, at the same time when keeping the information density also the footprint of intermediate data stored in memory will also be reduced which will help us to "store more for less".

(2) Consider computation free. Computation, unlike bandwidth, is projected to scale with the increasing number of processors that can be integrated into a server. So, we advocate for the following second core principle: *exploit the "free" computation capacity.* This principle is in particularly interesting since it closely aligns with our first principle, as it allows for a trade-off between computation and bandwidth and effectively use the information density per bit. For instance, by integrating more aggressive compression schemes such as heavy-weight compression along the full data access path until, we can keep data footprint low until it reaches the compute and use (free) computation to inflate data once it hits compute. Additional techniques to leverage this principle will be discussed in the following.

(3) Use better what we already have. As an alternative to reduce data footprint, as a third principle we can use resources better that we already have. Current infrastructure suffers from stranded memory resources, i.e., memory that is not fully utilized. Microsoft, for example, reported that up to 25% [20] of memory capacity is stranded. This under-utilization has long been an issue but has become even more critical as memory capacity now represents a constrained resource. As such, we should optimize the use of memory to avoid wastefulness and manage costs effectively. In particular, there are two ways forward: (1) We should adopt the principle of resource-constraint systems that memory is precious instead of trading memory for computation, e.g., by large pre-computed lookup tables. (2) more flexible database architectures, e.g., by using CXL to pool memory [20].

What can DBMS do to deal with the growth in structured data?

It is clear from the discussion above that storage devices will be more bandwidth-constrained than capacity-constrained in the future, and this discrepancy will only worsen over time. Further, the other hardware trend is that compute cycles are plentiful and nearly free. These driving factors open up new opportunities for efficiently scaling data management techniques by leveraging the first principles above.

Thus, methods like compression and encoding that intrinsically increase the IDB and thus reduce data sent across communication channels will be essential. Leveraging the nearly "free" compute cycles and "cheap" storage capacity, one can be far more aggressive in pre-computation, replication, and summarization of data and query results. Memory capacity, however, is likely to be an increasingly constrained resource, and one will have to throw away data far more aggressively in that layer. However, since analytic data systems are often bandwidth-constrained, designing methods that trade storage bandwidth for lower memory capacity will be critical.

What can DBMS do to deal with the growth in unstructured data?

Unstructured data use cases are relatively new and emerging quickly driven by Generative AI technologies. Coupled with the rapid changes in hardware, we can only speculate on the approaches needed to deal with this class of data applications.

At the data scales we have today and due to the influence of machine learning, there is the opportunity to take advantage of lossy compression and approximated computing as way to reduce the amount of data moving from storage to processing units. There might also be an opportunity to apply learning to data so that there is no need to process the data to find the answer to a query since the answer can be obtained through inference over a model (similar to the way some queries can be answered by looking at an index rather than at the data). These approaches would be a direct application of Hill's Law.

Similarly, data selection and model behavior attribution techniques [16, 24] operating over, for example, training data can help in reducing data movement, especially if the sampling and filtering can be directly done on computational storage, thereby providing hardware support for the improvements needed in Hill's Law. This will also require to revisit conventional data processing algorithms so that they can operate on approximated subsets of the data and low precision vector representations. Finally, given the size of vectors employed in ML and LLMs, a more efficient used of memory though different representations and data organizations will help in reducing the impact of limited I/O and memory bandwidth on our ability to process large amounts of data.

5 Working Group 2: A Case for Memory-Centric Design of Cloud Servers and DBMS

Anastasia Ailamaki (EPFL – Lausanne, CH, anastasia.ailamaki@epfl.ch)
Lawrence Benson (TU München, DE, lawrence.benson@tum.de)
Helena Caminal (Google – Sunnyvale, US, hcaminal@google.com)
Yannis Chronis (Google – Sunnyvale, US, chronis@google.com)
Jana Gieceva (TU München, DE, jana.giceva@in.tum.de)
David A. Patterson (University of California – Berkeley, US, pattrsn@cs.berkeley.edu)
Eric Sedlar (Oracle Labs – Redwood Shores, US, eric.sedlar@oracle.com)
Lisa Wu Wills (Duke University – Durham, US, lisa@cs.duke.edu)

The exponential growth of data in cloud-based systems, coupled with the plateauing of traditional processor performance gains, has created a fundamental bottleneck for database management systems (DBMS). The processor-centric architectural model, dominant for decades, is now hindered by slowing improvements dictated by Moore's Law and Dennard Scaling. This shift in the performance landscape requires a rethinking of DBMS design principles, moving the focus from pure computation to optimizing data access via memory and storage.

Conventional processor-centric architectures place the CPU as the central element, surrounded by statically allocated DRAM and connected to storage and networking via I/O buses. While this design served well when all components improved at similar rates, it is increasingly unsuited to modern workloads. Innovations like chiplets help mitigate limitations on the compute side, but they cannot compensate for the DRAM and data management challenges posed by modern datasets. Furthermore, the focus on computation in traditional design conflicts with the fundamentally data-oriented nature of DBMS operations.

In response to these challenges, we advocate for a memory-centric design approach. This model dissociates processors and data, allowing processors to access a shared pool of DRAM as needed. This improves DRAM utilization and reduces the cost impact of DRAM, which is increasingly expensive compared to other components. Additionally, memory-centric systems strategically leverage low-cost processors near storage devices. These "smart" storage nodes pre-process data on-site, performing operations like filtering, aggregation, and compression, thereby minimizing the need for costly data movement to higher system levels. In short, the emphasis is on moving the computation to the data, not the other way around.

The recent CXL (Compute Express Link) industry standard enables realistic implementations of the memory-centric concept. CXL creates a cacheable shared memory space across multiple sockets, where at least some of the space is hardware coherent. While CXL introduces some latency and bandwidth trade-offs compared to local DRAM, the optimization of data locality and utilization more than compensate for these factors in DBMS workloads. For long-term scalability, CXL switches and optical interconnects could dramatically expand the size of the shared memory pool.

A memory-centric approach fundamentally transforms the design and optimization of DBMS systems. Here are key areas of change:

Query Execution

Traditional query optimization generates physical query execution plans from relational algebra expressions using an estimate of the available compute and memory resources as well as a cost model. Query processing relies on the optimizer's decisions to maximize efficiency through selecting specific operator implementations and implicit data movement (e.g., repartitioning, data shuffling with the exchange operator in a distributed setting). Both modules focus on minimizing processor cycles while mindfully using memory resources. Memory-centric computing makes data a first-class citizen. Query processing and optimization in memory-resident query execution engines will focus on pushing processing to and operating directly on data stored in local storage. Query optimization will produce query plans which make decisions about distribution of execution. The physical query execution plans will be produced locally on each node and may vary depending on node capabilities and local memory technologies (just-in-time mappings to local micro-architecture and code generation will produce the final query executions plan on each participating node during execution time, leveraging cache-conscious algorithms). When data from remote storage is needed, engines must push operator code to the processors attached to the remote storage. The integration of low-cost general-purpose cores (ARM/RISC-V) into those devices is a reality and recent innovation of tightly-integrated implementations of processing-in-storage will further increase the benefits of the memory-centric computing scheme. The more computation is executed close to storage the more we can reduce the pressure on the capacity and bandwidth of pooled DRAM. When the execution is placed near the data, instead of copying data between compute units, where possible it should be passed by reference. Memory-centric indexing needs to facilitate access to local data and use references to remote indexes to facilitate code transfer. Real-time adaptive query processing algorithms in combination with code-generated operators can bind the query processing logic with the specifics of memory microarchitecture, thereby optimizing for in-situ hardware characteristics. This makes query processing a natural fit to the memory pooling design, and enables more powerful operations to be offloaded to where the data actually sits (e.g., closer to storage).

Transaction Processing (OLTP)

Cloud-native transaction processing can significantly benefit from a memory-centric design as it allows for more efficient data access and processing by bringing computation closer to where the data resides in memory, thereby eliminating or reducing the need for frequent data movement between different memory locations and simplifies handling consistency. In addition, memory-centric designs enable better data partitioning strategies that minimize the need for cross-partition transactions.

Workload Infrastructure

Moving to a memory-centric system design does not have to make things complicated when reasoning about core infrastructure logic and management of resources. Whereas in a typical CPU-centric system, we have a coordinator and multiple compute nodes, in a memory-centric system the control plane can be placed on the host's root complex or a set of CPU nodes. The control plane's coordinator threads operate on shared system state and metadata, which can be placed on the coherent memory pool for ease of coordination, and to avoid leading to host-congestions [cite, Meta].

Data Pipelines

Databases are the first step in many data intensive tasks (e.g., Recommendation systems, Retrieval Augmented Generation, ML inference, sensor data monitoring) where the database output is the input to one (or more) "consumer" systems (e.g., Tensorflow model inference). The end to end task execution is organized in a data pipeline where each part of the pipeline uses the hardware and software platform suited to the operation it executes.

A memory centric design can enable a shift where CPUs and the accelerators that are increasingly becoming part of such data pipelines are "equidistant" from the data enabling simpler communication and minimizing the performance cliffs observed by data movement or by CPU and accelerators operating with vastly different memory budgets.

Conclusion

The conventional processor-oriented computer design, centered around a CPU, is becoming less attractive due to the slowing of Moore's Law and the increasing costs and constraints of DRAM. As a result, we should shift towards memory-oriented computer designs, where data plays a central role.

Memory-oriented computer designs dissociate processors from data, enabling computation wherever the data resides and minimizing data movement. By placing a greater emphasis on memory and storage components, memory-oriented designs optimize resource utilization and enable more efficient processing of data-intensive workloads. This approach also aligns with the rise of domain-specific architectures (DSAs), particularly for tasks such as Deep Neural Networks (DNNs), which are driving significant advancements in data center computing. For database systems specifically, a memory-centric design offers several advantages, including improved query processing, transaction management, and hybrid transactional-analytical processing. By partitioning query processing operations based on data location, pushing processing closer to the data, and optimizing memory capacity, memory-centric DBMSs can achieve higher performance and efficiency. Additionally, memory-centric designs facilitate the integration of data pipelines, enabling simpler communication and minimizing performance cliffs associated with data movement.

The adoption of memory-centric design principles represents a fundamental shift in how we approach computing architecture, particularly in cloud environments and database systems. By prioritizing memory and data access over traditional compute-centric approaches, memory-oriented designs offer the promise of greater performance, scalability, and efficiency in handling data-intensive workloads in the modern computing landscape.

6 Working Group 3: AI Hardware. What is in it for Cloud DBMSs?

David F. Bacon (Google – New York, US, dfb@google.com)
Holger Fröning (Universität Heidelberg – Mannheim, DE, holger.froening@ziti.uni-heidelberg.de)
Mark D. Hill (University of Wisconsin-Madison, US, markhill@cs.wisc.edu)
Holger Pirk (Imperial College London, GB, pirk@imperial.ac.uk)
Pinar Tözün (IT University of Copenhagen, DK, pito@itu.dk)
Tianzheng Wang (Simon Fraser University – Burnaby, CA, tzwang@sfu.ca)

AI hardware and deep neural networks (DNNs) synergize to advance computational capabilities. Graphics Processing Units (GPUs), originally for graphics rendering, accelerate DNN training and inference with parallel processing. DNNs' demanding computations benefit from GPUs' parallel architecture, driving efficiency and speed.

Similarly, DNN demands drive the design of specialized AI hardware such as Google's TPU [18], Intel's Gaudi [17] processor, and various other examples from industry and academia [5, 7, 3, 8, 6, 1, 11, 13, 30]. As DNN complexity grows, so does the demand for more powerful processors, spurring advancements in processor design. In turn, processor evolution fuels DNN innovation, enabling breakthroughs in computer vision, natural language processing, and autonomous systems. This reciprocal relationship propels technological progress, shaping the future of AI and computational sciences.

AI Hardware and Workloads

Given the synergy between AI hardware and DNNs, we make two main observations that mutually fuel each other. Firstly, the computational rule behind DNNs – including but not limited to convolutional layers, attention modules, and linear layers – is to a large extent dominated by the dot product operation. Seen as a set of operations that shares input operands, dot product operations exhibit high computational/arithmetic intensity i, calculated as the number of computations executed per byte fetched from memory. Secondly, analyzing the scaling trends reveals that compute performance scales much better than memory performance. In more detail, the ratio r of compute performance in operations per second and memory performance in bytes per second continues to grow. Notably, these two observations complement each other, as achieving peak performance on CMOS-based processors requires an increasing computational intensity i.

Furthermore, a growing ratio implies that overall execution costs are increasingly dominated by data movements such as fetching data from memory to the processor. In contrast, the contribution of the number of computations following such a data fetch to overall costs is becoming insignificant. Fundamentally, this suggests that it is unpromising to design algorithms based on the number of operations as it happens in classical Big O analysis. Instead, it is highly promising to revisit algorithmic alternatives that have no longer been pursued due to high computational costs.

Last, the energy (non-)proportionality of processors suggests that it is unpromising not to run a processor at peak utilization, as only then the best power efficiency can be achieved. As peak utilization requires utilizing all components, including compute and memory, such a situation is only achievable for a high ratio r. This in combination with other effects, such as large electrical surges when power variations occur, also indicates that computationally intensive workloads are desirable.

From Data-Intensive to Compute-Intensive

World-wide data production is increasing very quickly, and the amount of data stored by databases is increasing rapidly as well. For instance, Google's Spanner has been roughly doubling in size every year, expecting to reach a zettabyte by 2030 as mentioned earlier. While compute and I/O costs for data continue to rise rapidly, they are not rising nearly as fast as the demand for storage. This means that on a per-byte basis, data is becoming colder.

Databases have historically been I/O-dominated. They retrieve large amounts of data and do relatively small amounts of computation per byte. Increases in both storage volume and I/O cost have motivated more and more use of data compression, as one way of trading space for time.

These database trends are on a collision course with the hardware trends described in the previous section. The one resource that is becoming cheaper is specialized computation (in the form of GPUs and TPUs). Thus far we have not found a way to exploit this form of computation. Some work has explored how to process data on TPUs [14], however, as the amount of data grows exponentially we still lack approaches to managing data storage cost, and bringing substantially more data from storage to the computing resources still presents a major bottleneck.

Our fundamental thesis is therefore *we must switch the balance in database systems from being data-intensive to being compute-intensive*. If we can trade space for time, and move the compute time into specialized hardware (and in particular, specialized hardware being deployed for AI), then we can bring database growth back in line with technology growth.

In the following, we explore two basic approaches to solving this problem:

- Developing new forms of data compression that takes advantage of various properties of large language models (LLMs) [27, 29, 9, 4] and their use in emerging applications.
- Using more computationally expensive algorithms for database operations, and moving them into TPUs and GPUs.

Learned Compression

Compression is typically divided into lossless and lossy compression. We propose a third category, *learned compression*, which uses LLMs to compress data.

The fundamental observation is that much of the data growth is being driven by the storage of generated data, using prompts to LLMs. This presents an opportunity for databases to reduce their storage footprint by storing prompts rather than storing the "expansion" and re-generating the expansion on demand. This can lead to order-of-magnitude reduction in data storage and data transfer costs, at the expense of much more compute-intensive data retrieval. But that retrieval has now been moved to the GPU/TPU, where we can ride the commodity curve of capability and capacity.

Assuming the source data used to generate an LLM-based response is already in the database, and that we store a pointer to the input data, we describe three forms of learned compression:

- Use the LLM as a data source.
- Use the LLM to generate a summary of the information that is stored.
- Use the LLM to produce an approximation of the original input when it is retrieved.

Regenerating Auto-generated Information. A use of LLMs that has very rapidly entered widespread commercial use is automatic generation of text inside of productivity applications. For example, Gmail plugins can generate responses using GPT-4. In many cases, users simply accept the generated response and send it. The generated email is then stored in the "Sent" folder. However, we observe that the response can be encoded far more compactly as a triple consisting of (1) a "pointer" to the message to which the reply was generated (e.g. a message ID), (2) an identifier for the model that was used to generate it (e.g. "GPT-4"), and (3) the random seed used for the generative operation. Upon retrieval, this triple is combined with the original message and fed into the model, which should re-generate the identical response text.

The same approach can be applied for applications like auto-generated document summaries, translations and so on.

Delta Compression. It is quite common for the auto-generated email text to be modified by the user, and frequently those changes are small relative to the total size of the text. We can augment the technique above by storing a delta against the generated text. If the delta gets too large, we simply revert to storing the text itself.

Other Modalities. Images, audio, and video represent even larger opportunities for compression. In some cases this will be many orders of magnitude. One example would be *Generate an MP3 audio file saying "I'll Be Back" using Donald Duck's voice.* The database then only needs to store such prompt along with the random seed and model information, instead of full MP3 audio file copies.

Generating Column Data from Models. Another way of turning data-intensive operations into more compute-intensive ones is rather than storing all the data in a database to answer queries, we let models to either directly answer certain questions or generate materialized views of data for further data processing.

Even though it may be imprecise, LLMs have been good at answering common knowledge questions. For example, the answers to which or how many books are written by an author or the movies an actor stars are such questions. If we are to use a SQL query over a database to answer these questions, the data has to be loaded into a database with a predefined schema, such as:

```
-- A traditional SQL table:
CREATE TABLE Actors (
Actor STRING NOT NULL
Movie STRING NOT NULL
) PRIMARY KEY (Actor);
```

This results in higher space consumption either in storage or memory compared to keeping raw/unstructured data because of the explicit schema.

Rather than explicitly storing the data in a database with a schema, today we can effectively pose these questions as prompts to a LLM. Depending on the questions, an LLM deployed on a GPU or TPU can either answer it directly or generate a column or a table that can later be used by more traditional database operators. For example, we may turn the above predefined schema into the following form:

```
-- A SQL table generated from a model:
CREATE TABLE Movies (
Actor STRING NOT NULL,
Movie STRING NOT NULL AS (GENERATE FROM Gemini3.0
WITH "What movies featured the actor [Actor]?")
) PRIMARY KEY (Actor);
```

Also, a `COUNT *` query can be answered by the model without generating a table. If we expect follow-up questions after the initial prompt, it could be better to generate a table and materialize it. For example, a search of the movies an actor starred in may trigger a follow up question on keywords on the movie title. The materialized tables can be cached in accelerator memory if there is space or stored in a fast persistent storage medium (e.g., NVMe SSDs) for later reuse by other queries.

Re-inflation of Summarized Data. LLMs are also heavily used for both summarizing documents and generating longer documents given a summary. One can use these features to store the summarized versions of documents closer to the compute nodes with accelerators. Storing the summaries instead of the whole documents reduces the overall data size and makes caching of this data more feasible. When someone asks for the contents of the whole document, an LLM deployed on a hardware accelerator such as GPU or TPU can re-inflate this document using its summary.

Revisiting Compute-intensive Problems in Data Processing

Analytical query processors need to be conscious of the bottleneck shifting from compute to data access. Fortunately, there is precedent for such shifts, such as the move from sequential to parallel processing. To take that into account, we believe that algorithms that have been considered too compute-intensive in the past should be reinvestigated to determine if they are more competitive given the new balance in hardware. There are past examples that can serve as inspiration: For aggregation, sequential scans are optimal on sequential processors, while massively parallel prefix scans perform optimally on GPUs. Worst-case optimal join algorithms reduce the need for large intermediate result materialization at the cost of highly CPU-inefficient control paths. Neural networks were considered too expensive to evaluate upon their invention (in the 1960s) efficiently but received a boost in popularity when massively parallel processors became available. Another example from scientific computing is iterative methods used to solve systems of linear equations. While they share a common objective of minimal time, they differ in how they update the solution at each iteration. In the classical Gauss-Seidel method which was designed with sequential processing in mind, the updated value of one element is immediately used in the calculation of the next element. In the Jacobi method, in contrast, all components of the solution vector are updated simultaneously using the values from the previous iteration. In practice, this means that Gauss-Seidel converges faster with regard to the number of iterations. However, due to the algorithm's sequential nature, the execution is not in line with parallel processors. Given that all processors are highly parallel, it is thus much more promising to use the Jacobi method instead, even though it is work-inefficient with regard to the number of iterations.

We believe that more such "newly-relevant" techniques can be discovered for data-intensive operations such as relational analytics, graph processing and even transaction processing.

Non-Traditional Workloads

Finally, the increasing diversity of data management workloads creates challenges beyond traditional analytics and transaction processing: workloads such as spatial data processing, data cleaning or data integration will likely benefit from AI-supporting hardware. Developing creative solutions to map such workloads to the new hardware is an exciting research opportunity. Emerging domains such as vector databases are also a natural fit for AI-focused hardware.

7 Working Group 4: Incrementally Distributed

Nandita Vijaykumar (University of Toronto, CA, nandita@cs.toronto.edu),
Zsolt István (TU Darmstadt, DE, zsolt.istvan@tu-darmstadt.de),
Tilmann Rabl (HPI Potsdam, DE, Tilmann.Rabl@hpi.de),
Alexander Boehm (SAP HANA, DE, alexander.boehm@sap.com),
Margo Seltzer (University of British Columbia – Vancouver, CA, mseltzer@cs.ubc.ca)

Today's database landscape renders as a black and white image in which database systems are either single-node systems or distributed systems. Generally, the distributed systems offer scalability, but pay a penalty in terms of single-node performance. But single-node systems face obvious scalability challenges. The ideal would be a system with the simplicity and ease of operation of a single-node system capable of infinite scaling.

In lieu of being able to achieve our ideal, we currently provide two different solutions. First, we have small-scale distributed systems, such as Oracle RAC [26] and SAP HANA [12], that typically scale only to a double-digit number of geographically co-located machines. Second, cloud storage systems enable single-node systems to host databases larger than the storage available locally on a single node. Both of these architectures are point solutions that do not fully cover the range of possibilities between single-node and cloud-scale databases.

Resource disaggregation provides a logically centralized abstraction for a physically distributed reality, managed by infrastructure providers. We already make this a reality for persistent storage: cloud-scale storage systems provide practically infinite capacity from a single node. Emerging interconnect technology, such as Compute Express Link (CXL) [25], makes it possible for compute nodes to directly access more memory than is possible in a purely local configuration. In other words, in the same way that cloud storage systems allow for disaggregated storage; CXL allows for disaggregated memory. In addition to providing a simpler abstraction on which to build databases, such disaggregation solves problems for cloud providers: it makes use of stranded resources, thereby increasing cloud provider efficiency. This trend towards disaggregation introduces exciting, new software architectures.

We present a taxonomy that provides a framework in which to consider disaggregation more broadly. We ask what it might mean to disaggregate any combination of CPU, memory, network connectivity, storage, and custom accelerators. We consider different combinations of resource disaggregation and how such new configurations influence database management systems.

Considering disaggregation more broadly lets us optimize systems for different metrics. Rather than focusing on maximizing queries-per-second (QPS), we might ask for an architecture that maximizes QPS/core, QPS/byte, QPS/joule, etc. Alternately, these considerations allow us to consider software architectures that are impractical today, e.g., single-node databases with access to petabytes of main memory or systems with new forms of elasticity in memory, network bandwidth, or access to hardware acceleration. In other words, the golden age of computer architecture [15] should be enabling a new golden age in database system design.

All resources become disaggregated

In the past, scalable database architectures have introduced disaggregation for storage systems such as Oracle RAC that use dedicated and separately scalable storage networks. This separation of storage and compute improves adaptability and independent scalability of resources, enables better sizing decisions, and avoids underutilization. Meanwhile, disaggregated storage has become the standard way of persisting data in cloud environments. This allows even systems that were not designed with disaggregated storage in mind to benefit from features such as virtually infinitely scalable, highly available, and dynamically growing storage that looks like a traditional block device.

Today, DRAM contributes a significant fraction of the cost of data centers [21]. This memory is utilized differently and dynamically depending on the applications and operators running in the system. Database engine designers have started addressing this problem by sharing memory across nodes. Today, memory sharing is selectively used for specific operations, such as data shuffling in Google BigQuery [23], or to temporarily "borrow" memory from remote nodes using RDMA to avoid spilling data to disk in situations where not enough local memory is available [21]. However, there is no standard and easy-to-use way for cross-node memory sharing. Technologies such as CXL provide a new opportunity as they enable transparent memory capacity sharing within racks, with limited, but noticeable, overhead over local memory. This enables gracefully scaling memory requirements beyond single node capacity by utilizing either specialized memory instances or neighboring nodes' memory.

As a next step, after memory and storage are disaggregated, one can also think about disaggregating other resources, in particular, network and accelerators. Given increasing network bandwidth and varying communication patterns in applications, it is likely that systems frequently fail to utilize all their local network capacity. The NICs in some nodes will be overutilized while others are idle, similarly to the situation with memory today. If nodes are also interconnected with other technologies such as CXL, we see the potential of disaggregating networking, by sharing NICs across nodes. This enables new heterogeneous setups of network topologies, increasing the total bandwidth available, and improving utilization. Analogously, disaggregating accelerators makes it possible to share them for certain workloads rather than requiring a homogeneous overprovisioning on every node or requiring specialized node configurations. A fully disaggregated system will be able to efficiently use local and remote resources, scale each resource individually based on demand from a single system perspective, and optimize deployments from a multi-tenant perspective.

Database Taxonomy of Disaggregation

We claim that disaggregation will enable DBMS that can be incrementally distributed over time, regardless of whether they were initially designed as single-node systems with node-local resources or as distributed systems, where resources are spread out in a cluster.

Table 1 below gives an overview of the resource allocation for various database system designs and deployments. Node-local resources are depicted as an "L", and disaggregated/distributed resources are denoted by "D".

The classical single-node databases are built with the assumption that all resources are node-local (Line 1 in Table 1). By deploying them in a cloud-environment where storage is already disaggregated, but still offered as a block-device abstraction (e.g., AWS Elastic Block Storage (EBS), Google Persistent Disk (PD)), these systems implicitly adopt a distributed storage layer (Line 2). This brings them closer to multi-node DBMS such as Oracle RAC or Google AlloyDB (Line 3), which were initially designed for (small) clusters of machines leveraging a shared storage pool that enables data sharing across nodes. While Google's Spanner system (Line 4) is architected as a globally distributed system that significantly scales beyond the capabilities of multi-node DBMS, its use of disaggregated hardware is still similar to "classical" scale-out database deployments.

Table 1 Overview of the resource allocation for various database system designs and deployments.

		CPU	Network	Memory	Storage	Accelerators
1	Single Node (Postgres, MySQL)	L	L	L	L	L
2	Single Node on Cloud Infra (RDS, CloudSQL)	L	L	L	D	L
3	Multi-Node Systems (Oracle RAC, SAP HANA, AlloyDB, Aurora)	D	L	L	D	L
4	Distributed OLTP Systems (Spanner, Cockroach)	D	L	L	D	L
5	Distributed OLAP Systems (BQ, SPARK)	D	L	D	D	L
6	Future Analytical Systems (?)	L	D	D	D	D
7	Future Analytical Systems (?)	D	D	D	D	D

Today, there are only a few large-scale, distributed systems that are designed to make use of disaggregated memory. A prominent example is Google BigQuery [23] (Line 5), which uses a shared memory pool for data shuffling. With the proliferation of far memory, provided via CXL, existing multi-node systems can easily evolve in this direction, by getting (some of) their memory from remote CXL devices instead of local DRAM.

We envision that other resources such as network and accelerator cards will be disaggregated in the future, leading to potential new analytical systems that use communication channels other than Ethernet or RDMA for inter-node communication, thus allowing for the shared use of network and accelerator cards.

With the ever increasing trend of hardware disaggregation, we claim that all DBMS will eventually run on disaggregated storage, memory, and even make use of other disaggregated resources such as networking or accelerators. This is independent of whether the systems were initially designed for such a hardware configuration or not. Thus, database architects will need to adapt existing system architectures to the different hardware characteristics (e.g., increased latency, more capacity, changes in bandwidth) and leverage potential opportunities that come with disaggregation (e.g., increased elasticity, opportunities for data sharing).

Open Challenges & Opportunities

For database system architecture, disaggregation brings many opportunities. We see three steps toward DBMSs embracing a disaggregated future.

The low hanging fruit is to use disaggregated resources like local resources to expand single node systems, essentially, ignoring disaggregation. The prototypical example is memory expansion over CXL, which enables a system to extend local memory with slightly longer-latency far memory without changing the architecture. To avoid running into performance cliffs when crossing the single node boundary, awareness for local and far memory is beneficial. Otherwise, systems will continue to function as designed, but they may be notably slower. Without coping with disaggregation and optimizing across different workloads and database deployments, this will improve scalability but not utilization.

A next level is a cluster level view on the database and application deployments, which enables improving resource utilization by sharing resources across nodes with different requirements. Besides static, but disaggregated, mapping of resources, dynamic assignment can further improve utilization and reduce cost and energy USge.

When fully embracing disaggregation, it is possible to fluidly scale across resources, also scaling down to fractions of nodes. This would enable more fine-grain control in resource offerings from cloud providers, carbon efficiency from being able to consolidate and turn off unused pools of resources, and more flexibility in constructing hardware architectures for any database application. Being aware of hardware heterogeneity in disaggregated setups also enables efficient compute and data placement, new data structure designs, and innovative communication-less data exchange.

Carbon and Economic Efficiency

Data centers incur costs from provisioning sufficient compute resources (e.g., memory, CPU, storage) separately from provisioning power and cooling for these resources. Replacing and upgrading components further contribute to cost. In addition to direct economic costs, data centers create a significant carbon footprint from both power consumption and embodied carbon due to chip manufacturing for acquiring new components. As hardware technologies and applications evolve at a rapid pace, there is an increasing requirement for data centers to refresh older components.

Disaggregation offers the opportunity to deliver cost savings to both cloud providers and users, while enabling a reduction in carbon footprint. The reduction in carbon footprint via disaggregation can be accomplished in several ways. First, disaggregation can potentially reduce the overall requirements for hardware components by reducing resource stranding. The major benefits of disaggregation are in enabling better utilization of resources in the data center by addressing the bin packing problem, enabling flexible resource sharing, and more efficient resource utilization, potentially necessitating fewer resources overall. Second, disaggregation enables the use of older, lower performance components opportunistically in addition to newer higher performance components. This can potentially be done without sacrificing performance with novel system-level techniques to address performance differences. Providing different pricing points for the use of data center hardware can also motivate the use of older components. Third, disaggregation can enable flexibly turning off power to large pools of hardware resources in data centers during periods of low utilization. During these periods, all utilization of compute resources can be redirected to fewer pools of disaggregated resources and has the potential to significantly reduce the overall power consumption in data centers.

Economic Model: Elastic and Fine-grained Allocation

The initial promise of the cloud was elastic and fine-grained resource allocation. In today's cloud resource model, clients typically pick between pre-defined "bundles" of resources: even though there are plenty of different instance types in the cloud offering, these all link together CPUs, memory, network, and storage. In a disaggregated future, with resources "unbundled", clients can express their needs more specifically, in terms of CPU cores, memory, storage capacity, network bandwidth, etc., in smaller increments than what instance types allow for today. This latter way of requesting, allocating, and billing for resources is the fulfillment of the initial promise of clouds on granularity.

In terms of elasticity of resources, even though today we can scale out at VM increments. Within a single VM, storage capacity and bandwidth can be elastic, but memory and CPU allocations are static. In a disaggregated future we will have elasticity of CPUs, memory, and network bandwidth even within a single VM. This is an important improvement for single node databases, which will be able to allocate an expected amount of memory but have the ability to temporarily use more main memory, in case they face workload spikes.

A beneficial side-effect of allocating resources for applications in a more fine-grained and elastic manner is that it will bring to light the hidden "divisor" of performance metrics. Today, we reason in Queries Per Second (QPS), hiding the fact that in most workloads we should consider QPS/core, QPS/GB of memory, or QPS/aggregate power of the resources we run on. By being able to expose the divisor, scalability bottlenecks can be easier to identify and requirements can be more transparently communicated between users and service providers.

In terms of the pricing model of disaggregated resources, we believe that those cloud providers (existing or new) that will find an economic model that passes on the savings resulting from the more efficient use of hardware to their customers will be at a significant advantage over their competitors who stick to rigid instance types. For customers, once they can allocate and pay only for the resources they need, there will be little incentive to stick to the old model.

Call to Action

Disaggregation is happening. As database designers, we have three choices: we can ignore it, cope with it, or embrace it. In reality, we have no choice, we must adapt systems to work in a disaggregated world. Coping with disaggregation means redesigning our systems to effectively use multiple memory tiers. Embracing disaggregation means designing next generation architectures that seamlessly grow from the fastest single-node system to a fully, cloud-scale system. In designing these systems, we should demand that hardware designers and cloud providers give us the mechanisms we need. If a small amount of cache coherent memory is game-changing for databases, then we need to convince the hyperscalers to make it available; we should demand elasticity in all resources.

We should embrace the entire spectrum from single-node to cloud-scale growth. Cloud providers should provide seamless elasticity in every dimension: CPU, storage, memory, network capacity, and hardware acceleration (including GPUs). Database providers should design systems that take full advantage of this elasticity. Together we can enable customers to optimize for metrics other than pure latency and bandwidth – customers might optimize for queries per second (QPS), QPS/core, QPS/byte, or QPS/watt.

References

1 Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, et al. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the twenty-fourth international conference on architectural support for programming languages and operating systems*, pages 715–731, 2019.

2 Nikos Armenatzoglou, Sanuj Basu, Naga Bhanoori, Mengchu Cai, Naresh Chainani, Kiran Chinta, Venkatraman Govindaraju, Todd J. Green, Monish Gupta, Sebastian Hillig, Eric Hotinger, Yan Leshinksy, Jintian Liang, Michael McCreedy, Fabian Nagel, Ippokratis Pandis, Panos Parchas, Rahul Pathak, Orestis Polychroniou, Foyzur Rahman, Gaurav Saxena, Gokul Soundararajan, Sriram Subramanian, and Doug Terry. Amazon redshift re-invented. In *SIGMOD*, pages 2205–2217. ACM, 2022.

3 Eunjin Baek, Dongup Kwon, and Jangwoo Kim. A multi-neural network acceleration architecture. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture*, ISCA '20, page 940 – 953. IEEE Press, 2020.

4 Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, US, 2020. Curran Associates Inc.

5 Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. Diannao: a small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '14, page 269 – 284, New York, NY, US, 2014. Association for Computing Machinery.

6 Yu-Hsin Chen, Joel Emer, and Vivienne Sze. Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. *ACM SIGARCH computer architecture news*, 44(3):367–379, 2016.

7 Yunji Chen, Tao Luo, Shaoli Liu, Shijin Zhang, Liqiang He, Jia Wang, Ling Li, Tianshi Chen, Zhiwei Xu, Ninghui Sun, and Olivier Temam. Dadiannao: A machine-learning supercomputer. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 609–622, 2014.

8 Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News*, 44(3):27–39, 2016.

9 Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sashank Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: scaling language modeling with pathways. *J. Mach. Learn. Res.*, 24(1), mar 2024.

10 Benoît Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, and Philipp Unterbrunner. The snowflake elastic data warehouse. In *SIGMOD*, pages 215–226, 2016.

11 Renhao Fan, Yikai Cui, Qilin Chen, Mingyu Wang, Youhui Zhang, Weimin Zheng, and Zhaolin Li. Maicc: A lightweight many-core architecture with in-cache computing for multi-dnn parallel inference. In *Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '23, page 411 – 423, New York, NY, US, 2023. Association for Computing Machinery.

12 Franz Färber, Norman May, Wolfgang Lehner, Philipp Große, Ingo Müller, Hannes Rauhe, and Jonathan Dees. The sap hana database–an architecture overview. *IEEE Data Eng. Bull.*, 35(1):28–33, 2012.

13 Soroush Ghodrati, Sean Kinzer, Hanyang Xu, Rohan Mahapatra, Yoonsung Kim, Byung Hoon Ahn, Dong Kai Wang, Lavanya Karthikeyan, Amir Yazdanbakhsh, Jongse Park, Nam Sung Kim, and Hadi Esmaeilzadeh. Tandem processor: Grappling with emerging operators in neural networks. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '24, page 1165 – 1182, New York, NY, US, 2024. Association for Computing Machinery.

14 Dong He, Supun C Nakandala, Dalitso Banda, Rathijit Sen, Karla Saur, Kwanghyun Park, Carlo Curino, Jesús Camacho-Rodríguez, Konstantinos Karanasos, and Matteo Interlandi. Query processing on tensor computation runtimes. *Proc. VLDB Endow.*, 15(11):2811 – 2825, jul 2022.

15 John L. Hennessy and David A. Patterson. A new golden age for computer architecture. *Commun. ACM*, 62(2):48 – 60, jan 2019.

16 Andrew Ilyas, Sung Min Park, Logan Engstrom, Guillaume Leclerc, and Aleksander Madry. Datamodels: Understanding predictions with data and data with predictions. In *ICML*, volume 162, pages 9525–9587, 2022.

17 Intel Corporation. Intel gaudi ai accelerators, 2024.

18 Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, US, 2023. Association for Computing Machinery.

19 Justin Levandoski, Garrett Casto, Mingge Deng, Rushabh Desai, Pavan Edara, Thibaud Hottelier, Amir Hormati, Anoop Johnson, Jeff Johnson, Dawid Kurzyniec, Sam McVeety, Prem Ramanathan, Gaurav Saxena, Vidya Shanmugam, and Yuri Volobuev. Biglake: Bigquery's evolution toward a multi-cloud lakehouse. In *SIGMOD*, 2024.

20 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. Pond: Cxl-based memory pooling systems for cloud platforms. In Tor M. Aamodt, Natalie D. Enright Jerger, and Michael M. Swift, editors, *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, ASPLOS 2023, Vancouver, BC, CA, March 25-29, 2023*, pages 574–587. ACM, 2023.

21 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. 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, page 574 – 587, New York, NY, US, 2023. Association for Computing Machinery.

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. Dremel: A decade of interactive SQL analysis at web scale. *PVLDB*, 13(12):3461–3472, 2020.

23 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. Dremel: a decade of interactive sql analysis at web scale. *Proc. VLDB Endow.*, 13(12):3461 – 3472, aug 2020.

24 Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: attributing model behavior at scale. In *ICML*, volume 202, pages 27074–27113, 2023.

25 Debendra Das Sharma, Robert Blankenship, and Daniel S. Berger. An introduction to the compute express link (cxl) interconnect, 2024.

26 Steve Shaw and Martin Bach. *RAC Architecture*, pages 63–95. Apress, Berkeley, CA, 2010.

27 Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

28 Thomas N. Theis and H.-S. Philip Wong. The end of moore's law: A new beginning for information technology. *Comput. Sci. Eng.*, 19(2):41–50, 2017.

29 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

30 Swagath Venkataramani, Vijayalakshmi Srinivasan, Wei Wang, Sanchari Sen, Jintao Zhang, Ankur Agrawal, Monodeep Kar, Shubham Jain, Alberto Mannari, Hoang Tran, Yulong Li, Eri Ogawa, Kazuaki Ishizaki, Hiroshi Inoue, Marcel Schaal, Mauricio Serrano, Jungwook Choi, Xiao Sun, Naigang Wang, Chia-Yu Chen, Allison Allain, James Bonano, Nianzheng Cao, Robert Casatuta, Matthew Cohen, Bruce Fleischer, Michael Guillorn, Howard Haynie, Jinwook Jung, Mingu Kang, Kyu-hyoun Kim, Siyu Koswatta, Saekyu Lee, Martin Lutz, Silvia Mueller, Jinwook Oh, Ashish Ranjan, Zhibin Ren, Scot Rider, Kerstin Schelm, Michael Scheuermann, Joel Silberman, Jie Yang, Vidhi Zalani, Xin Zhang, Ching Zhou, Matt Ziegler, Vinay Shah, Moriyoshi Ohara, Pong-Fei Lu, Brian Curran, Sunil Shukla, Leland Chang, and Kailash Gopalakrishnan. Rapid: Ai accelerator for ultra-low precision training and inference. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pages 153–166, 2021.

Participants

- Anastasia Ailamaki
EPFL – Lausanne, CH
- Gustavo Alonso
ETH Zürich, CH
- David F. Bacon
Google – New York, US
- Lawrence Benson
TU München, DE
- Carsten Binnig
TU Darmstadt, DE
- Alexander Böhm
SAP SE – Walldorf, DE
- Helena Caminal
Google – Sunnyvale, US
- Yannis Chronis
Google – Sunnyvale, US
- Holger Fröning
Universität Heidelberg – Mannheim, DE
- Jana Giceva
TU München – Garching, DE
- Mark D. Hill
University of Wisconsin-Madison, US
- Ihab Francis Ilyas
University of Waterloo, CA
- Zsolt Istvan
TU Darmstadt, DE
- Lana Josipovic
ETH Zürich, CH
- Tim Kraska
MIT – Cambridge, US
- Justin Levandoski
Google – Seattle, US
- Jignesh M. Patel
Carnegie Mellon University – Pittsburgh, US
- David A. Patterson
University of California – Berkeley, US
- Holger Pirk
Imperial College London, GB
- Tilmann Rabl
Hasso-Plattner-Institut, Universität Potsdam, DE
- Eric Sedlar
Oracle Labs – Redwood Shores, US
- Margo Seltzer
University of British Columbia – Vancouver, CA
- Pinar Tözün
IT University of Copenhagen, DK
- Nandita Vijaykumar
University of Toronto, CA
- Tianzheng Wang
Simon Fraser University – Burnaby, CA
- Lisa Wu Wills
Duke University – Durham, US
- Tobias Ziegler
TU Darmstadt, DE