Report from Dagstuhl Seminar 25151

Disruptive Memory Technologies

Haibo Chen*[1], Ada Gavrilovska*[2], Jana Giceva*[3], and Olaf Spinczyk*[4]

1 **Shanghai Jiao Tong University, CN.** haibochen@sjtu.edu.cn
2 **Georgia Institute of Technology – Atlanta, US.** ada@cc.gatech.edu
3 **TU München – Garching, DE.** jana.giceva@in.tum.de
4 **Universität Osnabrück, DE.** olaf@uos.de

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 25151 "Disruptive Memory Technologies".

Memory is a central component in every computer system. Hardware evolution has lead to greater capacities and higher speeds, but essential properties of its hardware/software interface have been unchanged for decades: Main memories used to be passive, largely homogeneous, and volatile. These properties are now so firmly anchored in the expectations of software developers that they manifest in their products.

However, a wave of innovations is currently shattering these assumptions. In this sense, several new memory technologies are disruptive for the entire software industry. For example, new servers combine "high-bandwidth memory" with classic memory modules and "CXL" enables even more hybrid architectures (non-homogeneous). The "in-/near-memory" computing approaches abandon the Von Neumann architecture and promise huge performance improvements by allowing CPU-independent processing of data objects in or close to the memory (non-passive). Finally, "persistent memory" is available for servers and embedded systems (non-volatile).

Overall, the expectations are high. Computers could have lower energy consumption, more performance, improved reliability, and reduced costs. However, from the (system) software perspective it is largely unclear how to use the novel memory technology efficiently. The seminar tackled this problem by discussing the state and potential of disruptive memory technologies, the challenges for system and application software, and important research directions.

Seminar April 6–11, 2025 – https://www.dagstuhl.de/25151

2012 ACM Subject Classification Computer systems organization → Serial architectures; Hardware → Communication hardware, interfaces and storage; Information systems → Data management systems; Information systems → Information storage systems; Software and its engineering → Software organization and properties

Keywords and phrases data-centric computing, disaggregated memory, persistent memory (pmem), processing in memory (pim), system software stack

Digital Object Identifier 10.4230/DagRep.15.4.1

* Editor / Organizer

Disruptive Memory Technologies, *Dagstuhl Reports*, Vol. 15, Issue 4, pp. 1–27
Editors: Haibo Chen, Ada Gavrilovska, Jana Giceva, and Olaf Spinczyk

Dagstuhl Reports
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Executive Summary

Haibo Chen (Shanghai Jiao Tong University, CN)
Ada Gavrilovska (Georgia Institute of Technology – Atlanta, US)
Jana Giceva (TU München – Garching, DE)
Olaf Spinczyk (Universität Osnabrück, DE)

The continued increase in demand for data and data-intensive applications is exposing scaling limitations in the capacity and performance of traditional DRAM-based memory system designs. In response, new memory system designs are emerging, based on disaggregation, new heterogeneous memory components, and near-/in-memory processing. However, to fully leverage the benefits of the hardware innovation requires redesign of the software stack, from the low-level operating system and virtualization primitives managing hardware access, to programming and compiler support and application runtimes. This seminar gathered researchers from industry and academia working across the entire stack, to discuss the pressing challenges in this new memory landscape, and to identify the most promising paths forward. The five-day seminar was structured to guide the discussion across the different layers:

Day 1 set the stage by first discussing the driving challenges from the database and datacenter communities, and on surveying the state-of-the-art of the hardware technologies for processing-near memory, processing-in memory and for disaggregation (with emphasis on CXL).
Day 2 included a deep-dive in diverse use cases, including bioinformatics and drug discovery, database processing, visual analytics, and AI/ML, followed by presentations on lessons-learned from practical adoptions of memory disruptors such as persistent memory.
Day 3 focused on identifying the limitations of state-of-the-art software technologies in leveraging new memory capabilities, complemented by a panel discussion addressing challenges of cross-stack co-design to optimize their potentials, considering both general-purpose and domain-specific needs.
Day 4 explored vision talks outlining future research directions for Processing-in-Memory (PIM) , Processing-near-Memory (PNM) , hyper-heterogeneous computing, and software-hardware co-design, etc.
Day 5 concluded with collaborative proposal discussions, synthesizing insights to define a strategic roadmap for advancing these technologies.

The breakout sessions, visionary talks, and panel discussions proved highly effective in identifying the challenges and opportunities posed by disruptive memory technologies. These discussions were particularly impactful due to the diverse mix of participants, including experts from hardware, computer architecture, operating systems, databases, and parallel software domains, representing both academia and industry. For instance, industry leaders and academic researchers approached emerging memory technologies from complementary perspectives, and the ensuing vigorous debates helped broaden understanding across both sectors. Insights shared by practitioners provided researchers with valuable context to refine and prioritize critical research questions. Our report offers a comprehensive summary of the seminar's key discussions and outcomes.

2 Table of Contents

3 Overview of Talks

3.1 Vertical integration for more efficient memory/storage

Gustavo Alonso (ETH Zürich, CH)

In this talk I will argue that the best way to use the advances in hardware around storage and memory is to build vertically integrated systems. By that, I mean building end-to-end pipelines of processing elements from computational storage through smart NICs, near memory accelerators, processors in memory, programmable switches, etc. This type of architecture is possible in the context of the increasing use of scale up architectures as alternative to the scale out, elastic, highly distributed systems common in the cloud. Scale up systems provide a more compact system with higher density of storage, memory and computational power, making them ideal to explore data processing pipelines involving active elements all along the data path from storage to the processor registers.

3.2 Memory and Storage Challenges in ML Pipelines

Oana Balmau (McGill University – Montréal, CA)

The rapid expansion of machine learning (ML) workloads has introduced significant challenges for memory and storage systems. In this talk, we highlight critical pain points along the ML pipeline – spanning training, checkpointing, and inference – with a focus on storage inefficiencies and emerging opportunities for innovation.

During training, storage bottlenecks are primarily caused by the exponential growth of dataset sizes, which increasingly exceed system memory and must be accessed from persistent storage. While existing dataloaders serve as the bridge between storage and compute, they are poorly equipped to handle tiered storage architectures and exhibit high variability in sample preprocessing times. These inefficiencies lead to head-of-line blocking and underutilized GPUs, with utilization often falling to 30% in real workloads. There are multiple opportunities for improvement, including hardware-software co-design, processing-in-memory (PIM) capabilities, and intelligent caching mechanisms.

A second major challenge is the write-intensive nature of model checkpointing. Modern checkpoints encompass model weights, optimizer states, and various metadata, with storage footprints ranging from 100 GB for 7B models to 17 TB for trillion-parameter models. Checkpointing requires all-to-all synchronization across nodes and is prone to stragglers and failures, making efficient recovery a pressing problem. Persistent memory (PM) and emerging interconnects like CXL offer promising avenues to mitigate these bottlenecks.

In the inference phase, large language models (LLMs) introduce new memory pressures due to key-value (KV) caches. To meet stringent service-level objectives (e.g., 200–450ms time to first token), these caches must reside in GPU memory and are maintained per user. For example, serving 1200 users with a LLaMA 2 70B model under realistic workloads requires approximately 3 TB of KV cache memory – far exceeding the combined 640 GB available on 8 H100 GPUs. The situation is further exacerbated in Retrieval-Augmented

Generation (RAG) pipelines, where context lengths may reach 200K tokens and additional cache entries are needed for knowledge base documents. Efficient management of these caches – through tiered storage, cache offloading, or optimized vector search – is becoming a crucial frontier.

To address these mounting challenges, MLCommons has launched the MLPerf Storage benchmark suite, which aims to systematically evaluate and improve the storage performance of ML pipelines. MLPerf Storage targets diverse scenarios, including training, checkpointing, and inference, by simulating real-world workloads and stressing different components of the storage hierarchy. This benchmark provides a common ground for researchers and practitioners to test new systems, co-design solutions, and drive forward innovations that ensure scalable and efficient ML pipelines.
More information on MLPerf Storage is available at `https://mlcommons.org/working-groups/benchmarks/storage/`.

3.3 CXL Pooling: What's Real and When?

Daniel Berger (Microsoft – Redmond, US)

This talk briefly reviews the broad opportunities promised by the CXL standard and then dives into pooling. Pooling and disaggregation at scale sound appealing but bring many implementation and deployment challenges. CXL switches are slow and expensive. Large multi-ported devices are cheaper and faster, but efforts to scale them have not been successful. We are essentially left with only two-ported pooling devices.

This talk proceeds with a proposal to build larger CXL pods out of these smaller devices. The key idea is simple: connect every CPU to multiple pooling devices, which in turn connect to different subsets of CPUs. Instead of connecting every device to every CPU ("fully connected pods"), our design connects every pair of CPUs to a shared pooling device ("partially connected pods"). This leads to cheap and potentially large pods but requires more complexity in the software stack. For example, CPUs need to carefully place data into the right pooling device that is accessible by those CPUs it seeks to communicate or collaborate with.

3.4 Programming and compiler abstractions for emerging computer architectures

Jerónimo Castrillón-Mazo (TU Dresden, DE)

Main reference Asif Ali Khan, Hamid Farzaneh, Karl Friedrich Alexander Friebel, Clément Fournier, Lorenzo Chelini, Jerónimo Castrillón: "CINM (Cinnamon): A Compilation Infrastructure for Heterogeneous Compute In-Memory and Compute Near-Memory Paradigms", in Proc. of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 4, ASPLOS 2024, Hilton La Jolla Torrey Pines, La Jolla, CA, USA, 27 April 2024 – 1 May 2024, pp. 31–46, ACM, 2024.
URL http://dx.doi.org/10.1145/3622781.3674189

Compute-in-Memory (CIM) is a promising non-von Neumann computing paradigm that promises unprecedented improvements in performance and energy efficiency. Moving past manual designs, automation will be key to unleash the potential of CIM for multiple application domains and to accelerate cross-layer design cycles. This talks reports on an ongoing effort to build a high-level compiler infrastructure for different CIM approaches, built with MLIR to abstract from individual technologies to foster re-use. This includes abstractions and optimizations flows for logic-in memory [2], content-addressable memories [3, 1], arithmetic operations in crossbars [4], and near-memory architectures.

References

1 Hamid Farzaneh, João Paulo Cardoso de Lima, Mengyuan Li, Asif Ali Khan, Xiaobo Sharon Hu, Jeronimo Castrillon. *C4CAM: A Compiler for CAM-based In-memory Accelerators*. Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'24), Volume 3, Association for Computing Machinery, pp. 164–177, New York, NY, USA, May 2024.

2 Hamid Farzaneh, João Paulo Cardoso De Lima, Ali Nezhadi Khelejani, Asif Ali Khan, Mahta Mayahinia, Mehdi Tahoori, Jeronimo Castrillon. *SHERLOCK: Scheduling Efficient and Reliable Bulk Bitwise Operations in NVMs*. Proceedings of the 61th ACM/IEEE Design Automation Conference (DAC'24), Association for Computing Machinery, New York, NY, USA, Jun 2024.

3 João Paulo C. de Lima, Asif Ali Khan, Luigi Carro, Jeronimo Castrillon. *Full-Stack Optimization for CAM-Only DNN Inference*. Proceedings of the 2024 Design, Automation and Test in Europe Conference (DATE), IEEE, pp. 1-6, Mar 2024.

4 Adam Siemieniuk, Lorenzo Chelini, Asif Ali Khan, Jeronimo Castrillon, Andi Drebes, Henk Corporaal, Tobias Grosser, Martin Kong. *OCC: An Automated End-to-End Machine Learning Optimizing Compiler for Computing-In-Memory*. In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD), IEEE Press, vol. 41, no. 6, pp. 1674-1686, Aug 2021.

3.5 Systems Software Meets Persistent Memory: Lessons and Experiences

Haibo Chen (Shanghai Jiao Tong University, CN)

A central challenge in storage research has long been balancing durability, speed, and capacity. Persistent memory, with its promise of durability, high-speed byte addressability, and scalable capacity, emerged as a transformative solution to this enduring tension. Over the past 12 years, our research and practical exploration has focused on rethinking the systems software stack to harness the potential of emerging hardware like persistent memory.

In this talk, I first examine our early efforts to redesign systems – including storage engines, file systems, distributed logging, and RDMA protocols – around nascent persistent memory prototypes. While these innovations demonstrated compelling performance gains in databases, file systems, and key-value stores, widespread adoption was hindered by immature hardware, cost inefficiency, and the need for intrusive changes to existing software stacks.

We then turn to recent advancements leveraging cutting-edge persistent memory and RDMA technologies to optimize transactional workloads in distributed, many-core database clusters. Our latest work achieves significant performance improvements by minimizing transactional overhead, demonstrating the viability of persistent memory-enabled systems in modern architectures.

Finally, I will reflect on our key lessons learned: the necessity of an evolutionary path for practical adoption, prioritizing solutions to critical problems over broad but shallow innovations, and the interplay between hardware maturity and software design. This journey underscores how emerging hard like persistent memory continues to reshape the frontier of storage research, demanding both technical rigor and strategic pragmatism in systems innovation.

3.6 50 years and Beyond – a look in the rearview mirror before looking forward

David Cohen (Solidigm – Santa Fe, US)

Amin Vahdat's "5th Epoch" keynote at the SIGCOMM 2020 a frame for more than 50 years of distributed computing. At the beginning of this period there were scale-up clusters but these gave way to roughly 40 years of x86-based, scale-out computing. In the wake of the End of Denard Scaling and the rise of Machine Learning, x86/Scale-Out is being disrupted by Accelerator-Centric, Scale-Up clusters that Harkin back to the beginning of the period. This talk looks at this evolution and questions assumptions that are biased toward x86/scale-out orthodoxy.

References

1 Amin Vahdat. *Coming of Age in the Fifth Epoch of Distributed Computing: The Power of Sustained Exponential Growth.* SIGCOMM 2020, keynote address.
2 P. Ranganathan and U. Holzle, *Twenty Five Years of Warehouse-Scale Computing.* In IEEE Micro, vol. 44, no. 05, pp. 11-22, Sept.-Oct. 2024, doi: 10.1109/MM.2024.3409469.

3.7 A Programming Model for CXL

Michal Friedman (ETH Zürich, CH)

Joint work of Michal Friedman, Gal Assa, Ori Lahav, Lucas Burgi
Main reference Gal Assa, Michal Friedman, Ori Lahav: "A Programming Model for Disaggregated Memory over CXL", CoRR, Vol. abs/2407.16300, 2024.
URL http://dx.doi.org/10.48550/ARXIV.2407.16300

CXL (Compute Express Link) is an emerging open industry-standard interconnect between processing and memory devices that is expected to revolutionize the way systems are designed in the near future. It enables cache-coherent shared memory pools in a disaggregated fashion at unprecedented scales, allowing algorithms to interact with a variety of storage devices using simple loads and stores. Alongside unleashing unique opportunities for a wide range of applications, CXL introduces new challenges of data management and crash consistency. Alas, CXL lacks an adequate programming model, which makes reasoning about the correctness and expected behaviors of algorithms and systems on top of it nearly impossible. In this talk I'll present CXL0, the first programming model for concurrent programs running on top of CXL. I'll present a high-level abstraction for CXL memory accesses backed up by a formal definition of operational semantics on top of that abstraction.

3.8 Using Performance-Aware Behaviour Models for Hardware Characterization and Performance Prediction

Birte Kristina Friesel (Universität Osnabrück, DE)

Joint work of Birte Friesel, Olaf Spinczyk
Main reference Birte Friesel, Olaf Spinczyk: "Performance-Aware Behaviour Models for Feature-Dependent Runtime Attributes in Product Lines", in Proc. of the 19th International Working Conference on Variability Modelling of Software-Intensive Systems, VaMoS 2025, Rennes, France, February 4-6, 2025, pp. 131–135, ACM, 2025.
URL http://dx.doi.org/10.1145/3715340.3715435

Cost models are pervasive throughout the system stack, for instance to decide whether it is sensible to move data to faster memory prior to processing. Typically, they are embedded into scheduling or placement algorithms, and not designed to be understandable or usable outside of those. This vision talk proposes behaviour models as a common formalism for cost models in system software, and shows how they can help understand and predict the performance of database operations on UPMEM PIM memory modules. Behaviour models decompose runtime actions into individual steps within a state machine, and associate each step (i.e., transition) with a distinct performance model. Thus, they allow for efficient hardware characterization and reasoning about performance, while still being compatible with (and usable as) conventional cost models.

3.9 Input from the Data Management community – challenges and opportunities

Jana Giceva (TU München – Garching, DE)

For decades data management systems and their well-defined workloads have been a great source of inspiration for hardware acceleration and a testing ground for new technologies. In this talk, we give a short overview of the different types of data-intensive workloads and their specific requirements and characteristics. After a brief historical overview of prior customized approaches to accelerate their workloads over the years (e.g., the idea of the database machine, data appliances and custom data accelerators), we go into what changes today with resource disaggregation in the cloud. One proposal is to push compute along the data path so we can have processing close to where to data sits or as it moves. For that we argue for a novel abstraction layer based on declarative operator primitives that decouples semantics from imperative implementation and explicitly reasons about state, data placement and movement.

3.10 Don't forget the "else" workloads (they are the majority!)

Boris Grot (University of Edinburgh, GB)

Joint work of Amna Shahab, Mingcan Zhu, Artemiy Margaritov, Boris Grot
Main reference Amna Shahab, Mingcan Zhu, Artemiy Margaritov, Boris Grot: "Farewell My Shared LLC! A Case for Private Die-Stacked DRAM Caches for Servers", in Proc. of the 51st Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2018, Fukuoka, Japan, October 20-24, 2018, pp. 559–572, IEEE Computer Society, 2018.
URL http://dx.doi.org/10.1109/MICRO.2018.00052

The slowdown in technology scaling mandates rethinking of conventional CPU architectures in a quest for higher performance and new capabilities. As a step in that direction, this talk questions the value of on-chip shared last-level caches (LLCs) in server processors and argues for a better alternative leveraging 3D memory stacking technology and tight CPU/memory integration.

3.11 The Persistence of Big Memory: Unlocking Memory Hierarchy

Yu Hua (Huazhong University of Science & Technology – Wuhan, CN)

The artificial separation between volatile memory and persistent storage has long constrained system architectures, resulting in inefficient data movements across registers, cache, DRAM, and block storage. Big Memory flattens this hierarchy by making persistence an inherent property of the memory layer with the aid of networking technologies. The memory pooling in CXL creates a unified address space across disaggregated nodes, allowing memory resources to be dynamically composed and shared with local-DRAM latency. The zero-copy and

kernel-bypass mechanisms in RDMA mitigate storage stack overhead by enabling direct memory-to-memory transfers between hosts, bypassing traditional CPU and OS involvements. Byte-addressable memory semantics further replace block I/O as the fundamental access primitive. This transformation doesn't merely blur the line between memory and storage, and systematically creates a continuum where all data exist natively in persistent memory.

This architectural breakthrough stems from a fundamental insight: persistence must be the foundational property of memory systems. By establishing persistence as the first principle, Big Memory reconfigures the traditional storage-memory dichotomy. Instead of treating persistence as a storage-tier feature, it becomes a native capability of the memory layer. This core principle directly enables the unlocked memory hierarchy. When the memory is inherently persistent, the entire architecture of registers/cache/DRAM/storage coalesces into a flat, scalable memory plane. The result is a self-consistent architecture where persistence enables hierarchy flattening that in turn maximizes the benefits of persistent memory, creating a virtuous cycle of efficiency and performance.

3.12 Near-Storage and Near-Memory – Two Peas in a Pod

Sudarsun Kannan (Rutgers University – Piscataway, US)

In this talk, I will motivate and explore how operating systems must evolve to effectively support near-storage and near-memory processing in the face of increasing hardware heterogeneity. I will begin by providing a historical perspective on how these technologies have evolved over the past three to four decades, from early intelligent disks to today's commercial computational storage and near-memory accelerators. Traditional systems were designed with a binary view of memory and storage, but emerging workloads and devices demand more fine-grained data placement and processing strategies. I will describe our efforts to reduce data movement and processing overheads through techniques such as CISC-style I/O operations, horizontal caching, and metadata-aware offloading. By viewing near-storage and near-memory as complementary, we enable collaborative computation across tiers and improve performance. I will also highlight major system software challenges and briefly discuss the opportunities these technologies present for edge-aware systems.

References

1 Y. Ren, C. Min, and S. Kannan. *CrossFS: a cross-layered direct-access file system*. In Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation (OSDI'20). USENIX Association, USA, Article 8, 137–154, 2020.

2 J. Zhang, Y. Ren, and S. Kannan. *FusionFS: Fusing I/O operations using CISCOps in firmware file systems*. In Proceedings of the 20th USENIX Conference on File and Storage Technologies (FAST 22), pages 297- 312, Santa Clara, CA, February 2022. USENIX Association.

3 J. Zhang, Y. Ren, M. Nguyen, C. Min, and S. Kannan. *OmniCache: collaborative caching for near-storage accelerators*. In Proceedings of the 22nd USENIX Conference on File and Storage Technologies (FAST '24). USENIX Association, USA, Article 3, 35–50, 2024.

3.13 Thanks for the Memories: Lessons from Disruptive Memory Technologies

Kimberly Keeton (Google LLC – South San Francisco, US)

In this talk, I explore the factors that contribute to the success (or failure) of a disruptive memory technology. I begin by surveying the evolution of memory technologies over the last 75+ years. Looking across multiple generations, several themes emerge as to why technologies have faded, including: 1) its performance/cost/density/reliability wasn't as good as another (often newer) alternative; 2) production, supply chain, and/or operational challenges; 3) a lack of widespread adoption / insufficient use cases; and 4) limited compatibility. After exploring several concrete examples, I propose several ingredients in a recipe for maximizing the chances for success, including: 1) compelling use cases and/or benefits; 2) support for understanding technology characteristics; 3) support for developing the software ecosystem; and 4) open standards. I conclude by posing several questions for discussion by the group.

3.14 Industry Talk – "Near-Memory Processing is Becoming a Reality"

Hoshik Kim (SK hynix – San Jose, US)

In this Industy talk session, Hoshik explained the background of growing demands for near-memory processing and the opportunities from industry perspective as well as the state-of-the-art of the technology and available products and prototypes for research communities.

The demands for near-memory processing are growing due to data-intensive characteristics of modern AI and data analytics workloads and gigantic energy consumption in data centers. The evolution of CXL and Custom HBM also opens a door to near-memory processing.

While near-memory processing has a great potential, Hoshik also highlighted that there are still many technical challenges ahead and innovations are required to commercialize the technology.

3.15 Memory-Centric Computing: Recent Advances in Processing-in-DRAM

Onur Mutlu (ETH Zürich, CH)

Main reference Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, Rachata Ausavarungnirun: "A Modern Primer on Processing in Memory", CoRR, Vol. abs/2012.03112, 2020.
URL https://arxiv.org/abs/2012.03112

Computing is bottlenecked by data. Large amounts of application data overwhelm the storage capability, communication capability, and computation capability of the modern machines we design today. As a result, many key applications' performance, efficiency, and scalability are bottlenecked by data movement. In this talk, we describe three major shortcomings of

modern architectures in terms of 1) dealing with data, 2) taking advantage of vast amounts of data, and 3) exploiting different semantic properties of application data. We argue that an intelligent architecture should be designed to handle data well. We posit that handling data well requires designing architectures based on three key principles: 1) data-centric, 2) data-driven, 3) data-aware. We give examples of how to exploit these principles to design a much more efficient and higher performance computing system. We especially discuss recent research that aims to fundamentally reduce memory latency and energy, and practically enable computation close to data, with at least two promising directions: 1) processing using memory, which exploits the fundamental operational properties of memory chips to perform massively-parallel computation in memory, with low-cost changes, 2) processing near memory, which integrates sophisticated additional processing capability in memory chips, the logic layer of 3D-stacked technologies, or memory controllers to enable near-memory computation with high memory bandwidth and low memory latency. We show both types of architectures can enable order(s) of magnitude improvements in performance and energy consumption of many important workloads, such as machine learning, graph analytics, database systems, video processing, climate modeling, genome analysis. We discuss how to enable adoption of such fundamentally more intelligent architectures, which we believe are key to efficiency, performance, and sustainability. We conclude with some research opportunities in and guiding principles for future computing architecture and system designs.

3.16 Some Experience with PMem, NVLink, CXL

Tilmann Rabl (Hasso-Plattner-Institut, Universität Potsdam, DE)

Main reference Felix Werner, Marcel Weisgut, Tilmann Rabl: "Towards Memory Disaggregation via NVLink C2C: Benchmarking CPU-Requested GPU Memory Access", in Proc. of the 4th Workshop on Heterogeneous Composable and Disaggregated Systems, HCDS 2025, Rotterdam, TheNetherlands, 30 March 2025, pp. 8–14, ACM, 2025.
URL http://dx.doi.org/10.1145/3723851.3723853

The memory hierarchy is becoming increasingly complex and heterogeneous. Modern servers include not only different levels of caches and main memory, but can also feature persistent memory, as well as, remote coherent memory through high speed interconnects. A recent addition is CXL, which is already supported in many new CPUs and a hot topic of research. Similarly, modern GPU servers often contain alternative high speed interconnects, such as NVLink and InfiniFabric, which serve a similar purpose.

In this presentation, we give an overview of our results evaluating persistent memory (PMem) in the form of Intel Optane and report on our experience in using it in a hybrid DRAM/PMem key-value store. We further evaluate the performance of CPU initiated GPU memory access through NVLink and show how fast interconnects enable fast GPU-based DB operations. We finally translate our findings to CXL and how first experiments on prototypical hardware.

3.17 Accelerating Bioinformatics Workloads

Tajana Simunic Rosing (University of California – San Diego, US)

Dramatic decreases in the cost of sequencing and other biological and chemical data acquisition has created a deluge of biomedical data. Personalized medicine is driven today by genomics, transcriptomics, proteomics, and metabolomics, which characterize the interactions between genes, RNA molecules, proteins, and metabolites respectively. These four "omics" disciplines are a key to understanding complex diseases including cancer, autoimmune disease, and response to infection, and are crucial for advances in medical diagnostics and therapeutics. Most omics data is generated by sequencing or mass spectrometry, using common tools and analysis pipelines that rely on CPU-based servers. Such systems spend a large majority of the time just moving data from storage and memory into the processors. For example, our recent profiling of algorithms used for analysis of mass spectrometry data shows that 80% of the run time is dominated by moving data from storage for preprocessing. We got similar results when profiling our COVID-19 genome sequence analysis pipeline.

Our team accelerated a number of key tools used for applications such as microbiome, COVID-19 analysis, protein and peptide mass spectrometry, using novel machine learning techniques, such as hyperdimensional computing, and in-memory and in-storage acceleration. Our accelerated microbiome and COVID-19 pipelines have been used by the UCSD medical center to support clinical investigations and the award-winning "Return to Learn" program that ensures student, faculty and staff safety on campus including sequencing tens of thousands of complete viral genomes from clinical cases, asymptomatic population screening, and wastewater. More recently, CDC has adopted some of our work as well. The in-memory sequence alignment reduced the time needed to go from the output of the sequencer to results by 1,900x, making it possible to get results in less than a second! Just the preprocessing step of mass spectrometry data analysis has been accelerated 180x by doing in-storage analysis. Hyperdimensional computing was used to accelerate mass spectrometry protein and peptide identification using in and near memory computing, and resulted in 5,000x speed up relative to the state of the art, at comparable accuracy. Such results go a long way toward ushering a new age of portable tools that can be used for personalized medicine in near real time to benefit individual patients today, not only in cohort studies that take years to analyze and benefit only future patients.

3.18 Suggestions for a PIM research agenda

Kevin Skadron (University of Virginia – Charlottesville, US)

After settling on a working definition of processing in memory (PIM) as placing logic within the memory die and processing near memory (PNM) as any logic outside the memory die (e.g., directly interfaced to the memory channel), discussions regarding future PIM research directions emphasized several topics.

One major issue was whether PIM will be more beneficial as an accelerator that is separate from the regular "host memory" ("accelerator-first") or as an additional feature within main memory ("memory first"). Some difficulties with the memory-first approach are that logic is costly in die area in a DRAM process, sacrificing memory capacity or severely limiting PIM functionality; that contention between PIM access and conventional memory read/write may negatively impact latency for the latter category; and that memory address interleaving is not friendly to PIM. However, requiring data to be copied from main memory to an accelerator-first PIM device sacrifices the opportunity to perform computing without moving data, and instead is limited to leveraging the internal parallelism of DRAM (which in itself is a powerful acceleration story). A hybrid approach is possible by mapping the accelerator-first PIM into the physical address space, or connecting it through CXL, but these likely sacrifice data access bandwidth and latency for conventional memory read/write.

Beyond these considerations, the compute placement (subarray vs bank-level), functionality (natively supported compute capabilities), and buffering (registers or scratchpad) are open areas of research. In addition, many applications will benefit from some mechanism for bank-to-bank and channel-to-channel communication without having to go through the host-side memory controller.

A further important consideration is how PIM should be programmed. The tight coupling of data placement and computation on that data are not well expressed in current programming languages.

Finally, participants discussed how PIM should be interfaced to the system, including virtual memory abstractions, and how PIM computation should be invoked.

3.19 Memory Management for the 21st Century

Michael Swift (University of Wisconsin-Madison, US)

The complexity of operating system memory management is increasing along with the complexity of compute, memory, acceleration, and I/O hardware. Current memory managers are typically monolithic code with brittle, inflexible policies.

In this talk I discuss our work on principled policies using a cost-benefit approach, showing that considering the cost of OS memory operations can dramatically reduce latency and improve performance. I also consider the challenge of modifying policies and show how to use the Linux virtual file interface (VFS) as an effective approach for introducing new memory management approaches. Finally, I address the challenges of tiered memory management in the OS, showing how lack of information can lead to poor policies that do not optimize tiering performance.

3.20 HYPNOS – Co-Design of Persistent, Energy-efficient and High-speed Embedded Processor Systems with Hybrid Volatility Memory Organization

Jürgen Teich (Universität Erlangen-Nürnberg, DE)

Joint work of Nils Wilbert, Stefan Wildermann, Jürgen Teich
Main reference Nils Wilbert, Stefan Wildermann, Jürgen Teich: "Hybrid Cache Design Under Varying Power Supply Stability – A Comparative Study", in Proc. of the International Symposium on Memory Systems, MEMSYS 2024, Washington, DC, USA, 30 September 2024 – 3 October 2024, pp. 257–269, ACM, 2024.
URL http://dx.doi.org/10.1145/3695794.3695819

This talk discusses the idea to introduce modern non-volatile memory (NVM) technology in the cache architecture of CPUs with the goal to save energy and and potentially save execution time of programs. Particularly for IoT devices that are subject to intermittend power outages, NVM caches sound promising. But in order to to not give up high clock rates (speed), a CPU cannot be built completely using NVM due to endurance limitations, hybrid solutions seem to be sound solution.

After an introduction to hybrid cache architectures, we present a design of a low-latency energy-efficient hybrid cache design for embedded systems and analyze different techniques on which data to keep in the volatile and which data to keep in the non-volatile caches lines and how to backup volatile data into non-volatile sections upon power outages.

From experimental results using a cycle-accurate simulation based on extensions of the GEM5 environment, we can see that different benchmarks do require different cache policies in order to save the highest amount of energy. It is also shown that current technologies of NVM can only be although not severely degrade the performance. For best exploitation of energy savings, we believe that hints coming from program analysis and compiler could provide an even better co-designed solution for such promising hybrid cache CPU architectures.

3.21 Memory Systems for Visual Computing: Challenges and Opportunities

Nandita Vijaykumar (University of Toronto, CA)

Visual computing has been revolutionized by emerging applications like multi-modal deep neural networks, generative AI, and differentiable rendering. These applications demand systems that can handle unprecedented challenges in efficiency, scalability, and security. In this talk, I will explore new challenges in efficiency by emerging visual computing applications and how innovations and technologies in memory systems can help address them.

3.22 Vision on Heterogeneous Hardware

Zeke Wang (Zhejiang University – Hangzhou, CN)

Joint work of Zeke Wang, Jie Zhang, Hongjing Huang, Yingtao Li, Xueying Zhu, Mo Sun, Zihan Yang, De Ma, Huajing Tang, Gang Pan, Fei Wu, Bingsheng He, Gustavo Alonso
Main reference Zeke Wang, Jie Zhang, Hongjing Huang, Yingtao Li, Xueying Zhu, Mo Sun, Zihan Yang, De Ma, Huajin Tang, Gang Pan, Fei Wu, Bingsheng He, Gustavo Alonso: "FpgaHub: Fpga-centric Hyper-heterogeneous Computing Platform for Big Data Analytics", CoRR, Vol. abs/2503.09318, 2025.
URL http://dx.doi.org/10.48550/ARXIV.2503.09318

Modern data analytics requires a huge amount of computing power to analyze on a massive amount of data. Therefore, heterogeneous data analytics platforms appear as an attempt to close the gap between compute power and data. For example, FPGA-based systems, GPU-based systems, SSD, and programmable switches.

Even though these heterogeneous systems achieve better performance than CPU-only homogeneous system due to the dying of Moore's law, we still identify that simple heterogeneous systems cannot harvest the potential of each heterogeneous device. For example, GPU-based approaches suffer from severe IO issues and programmable switch, e.g., P4 switch, has very limited compute and memory capacities.

To this end, we argue for hyper-heterogeneous computing for big data analytics. Therefore, we present FpgaHub, a hyper-heterogeneous computing platform for big data analytics. Our platform is centralized on FPGA-based SmartNIC, and explores its potential of co-optimization with any other heterogeneous devices, such as GPU, P4 switch, and SSD. The key idea of FpgaHub is to use FPGA to complement other heterogeneous devices via deep co-design, with a goal of "1+1 > 2".

3.23 CXL Memory – Where are we today?

Thomas Willhalm (Intel Deutschland GmbH – Feldkirchen, DE)

Today, the compute capabilities of servers are growing at a much faster pace than the memory capacity and bandwidth. This creates pressure on the number of pins and DIMM slots on a motherboard for realizing a balanced system. As a way to mitigate these limits, CXL memory (type 3) devices can expand the memory capacity and bandwidth using PCIe channels on current x86 processors. CXL memory devices come in various form factors, which provide different use cases: EDSFF in several sizes, number of PCIe lanes, and power envelopes as well as PCIe add-in cards, which also allow the (re-)usage of conventional DDR4 and DDR5 modules.

The latest Intel Xeon 6 processors support CXL 2.0 devices in two modes. By default, CXL memory is exposed as a separate NUMA node without cores assigned. OS and applications then manage the data placement based on the proximity information provided by the ACPI tables. In contrast to that, "Flat Memory Mode" provides a flat access to memory. The hardware is then managing the tiering between native DRAM and CXL memory on a cache line level transparently to the OS and application: Each cache line in the physical address space is either located in native DRAM or CXL memory. If the cache line is accessed, it will simply be returned in case it is located in native DRAM. In case the cache line is found in

CXL memory, the cache line will be moved to native DRAM, evicting some other cache line from native DRAM to CXL memory. Flat Memory Mode therefore requires a 1:1 ratio for native DRAM and CXL memory, but has the advantage that the total capacity of native DRAM and CXL memory can be accessed.

Beyond memory expansion, the CXL 3.0 standard will provide further capabilities for "memory pooling". This opens the door for innovative usage taking multiple hosts into account: dynamic memory allocation, data sharing, preserving data in the pool during restart, or producer-consumer access. These innovative use-case come with the burden that the whole stack from hardware, OS and hypervisor, device drivers and libraries, to the application level need to support the required capabilities. The best way to achieve broad adoption are solutions that work out-of-the-box and bring benefit for existing user scenarios but offer the required extensions for novel use cases. Potential targets are therefore distributed systems that could benefit from faster or simpler communication.

3.24 Memory Requirements and Challenges in Large Language Models (LLMs)

Chun Jason Xue (MBZUAI – Abu Dhabi, AE)

Main reference Hongchao Du, Shangyu Wu, Arina Kharlamova, Nan Guan, Chun Jason Xue: "FlexInfer: Breaking Memory Constraint via Flexible and Efficient Offloading for On-Device LLM Inference", in Proc. of the 5th Workshop on Machine Learning and Systems, EuroMLSys 2025, World Trade Center, Rotterdam, The Netherlands, 30 March 2025- 3 April 2025, pp. 56–65, ACM, 2025.
URL http://dx.doi.org/10.1145/3721146.3721961

The deployment of Large Language Models (LLMs) still faces significant challenges due to their extensive memory requirements, especially on mobile and edge devices. For instance, even a quantized 7B-parameter model demands over 7GB of memory, exceeding the capacity of most mobile platforms. This talk explores the memory requirements and challenges in Large Language Models (LLMs), highlighting the shift from computational efficiency in traditional DNNs to memory efficiency as a critical bottleneck in modern LLMs. This talk discusses key challenges during inference, such as the growing model size and the high memory footprint of KV cache, along with optimization techniques like quantization, pruning, and offloading. The talk also introduces memory constraints in LLM training, including pre-training and post-training phases, and emphasizes solutions like parallel training and mixed precision. Additionally, it covers system-level optimizations such as virtual memory management and prefetching to enhance efficiency. The talk concludes by underscoring the importance of balancing memory and computational efficiency to advance LLM capabilities.

4 Working groups

4.1 Report of the Working Group on Hardware Coherence

Antonio Barbalace (University of Edinburgh, GB)

Hardware coherence is not the only solution for enabling inter-machine CXL memory sharing – it largely depends on the specific application. While many potential use cases for CXL memory sharing have been identified, including VM and container migration, Spark, key-value stores, databases, distributed AI/ML workloads (such as Ray), distributed file systems, high-frequency trading, and legacy system support, not all of them require hardware coherence. In general, the working group believes that the need for full address-space hardware coherence appears to be quite limited.

That said, certain scenarios may still benefit from some degree of hardware coherence. These include metadata management in distributed file systems, in-memory databases and transactional systems, high-frequency trading applications where low latency is critical, and support for legacy software that assumes hardware coherence. In these cases, hardware-based coherence can improve performance or simplify development, but for most other applications, software-based solutions or hybrid approaches may be more appropriate.

4.2 Report of the Working Group on Operating Systems Issues

David Cohen (Solidigm – Santa Fe, US), Frank Bellosa (KIT – Karlsruher Institut für Technologie, DE), Daniel Berger (Microsoft – Redmond, US), Yu Hua (Huazhong University of Science & Technology, CN), Kimberly Keeton (Google LLC – South San Francisco, US), Michael Swift (University of Wisconsin-Madison, US), and Thomas Willhalm (Intel Deutschland GmbH – Feldkirchen, DE)

An ad hoc working group on the role of the operating system for systems with disruptive memories met. The discussion covered topics such as how much should be delegated to applications, whether operating systems must get more disaggregated, what the right high-level abstractions are, how to secure and protect data, and what are the goals for operating system memory management.

4.3 Report of the Working Group on CXL Sustainability and Memory Fabric

Ada Gavrilovska (Georgia Institute of Technology – Atlanta, US), Gustavo Alonso (ETH Zürich, CH), Daniel Berger (Microsoft – Redmond, US), Thaleia Dimitra Doudali (IMDEA Software Institute – Madrid, ES), Hoshik Kim (SK hynix – San Jose, US), Alberto Lerner (University of Fribourg, CH), Ilia Petrov (Hochschule Reutlingen, DE), Tilmann Rabl (Hasso-Plattner-Institut, Universität Potsdam, DE), and Zeke Wang (Zhejiang University – Hangzhou, CN)

The breakout session brought seminar participants interested in two questions related to CXL memory: impact on sustainability and requirements for memory fabric design.

Regarding sustainability, the group discussed recent data published by industry [1] Operational cost of storage and memory, as well as of CPUs, are significant. Embodied cost associated with storage and memory are significant, whereas of CPUs not as much. The embodied costs of PNM are likely to be increased. Efficiency is important, but just energy efficiency (operational cost) is not sufficient. Operational costs are expected to go down substantially, for instance the hyperscalers are investing in offsetting these costs with renewables. However, embodied costs will remain important. It will be important to find ways to extend the lifetime on components, including of memory and storage, and to reuse them. Disaggregation solutions, powered by CXL, provide one way that can help in this regard [2].

Regarding the memory fabric design, the group first focused on contrasting current CXL-based memory fabric capabilities compared to other technologies (PCI, IB RDMA, NVLink, GPUDirect). The group then considered several LLM inference scenarios and worked through high level designs of fabric requirements that would be able to satisfy the data requirements of a high-end GPU system. The participants agreed such use-case driven approach provides a good way to drive the fabric design requirements and several of them planned to follow up on the initial conversation to derive more detailed requirements.

References

1 Wang et al. *Designing Cloud Servers for Lower Carbon*. ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, 2024, pp. 452-470, doi: 10.1109/ISCA59077.2024.00041.

2 Yuhong Zhong, Daniel S. Berger, Pantea Zardoshti, Enrique Saurez, Jacob Nelson, Antonis Psistakis, Joshua Fried, and Asaf Cidon. *My CXL Pool Obviates Your PCIe Switch*. In Proceedings of the 2025 Workshop on Hot Topics in Operating Systems (HotOS '25). Association for Computing Machinery, New York, NY, USA, 58–66. https://doi.org/10.1145/3713082.3730393

4.4 Report of the Working Group on Systems Support for Processing-Near-Memory

Ada Gavrilovska (Georgia Institute of Technology – Atlanta, US), Gustavo Alonso (ETH Zürich, CH), Birte Kristina Friesel (Universität Osnabrück, DE), Sudarsun Kannan (Rutgers University – Piscataway, US), Hoshik Kim (SK hynix – San Jose, US), Ilia Petrov (Hochschule Reutlingen, DE), Kai-Uwe Sattler (TU Ilmenau, DE), Kevin Skadron (University of Virginia – Charlottesville, US), Zeke Wang (Zhejiang University – Hangzhou, CN), and Chun Jason Xue (MBZUAI – Abu Dhabi, AE)

A working group on processing-near-memory (PNM) technologies gathered in one of the Day 2 breakout session to discuss the use cases, programming interfaces, and systems support for integration and efficient use.

Some part of the discussion was spent on differentiating PNM for processing-in-memory (PIM) technologies. PIM modifies the DRAM design, and/or operates at limited (bank) granularity. It's generally more challenging to integrate in systems. Multiple examples (e.g., vector-matrix dot product) show performance promise, but efficiency is not a clear win. It is well suited for examples where capabilities for flexible data access and data gather can be leveraged at the supported granularity, without the need to transform data (e.g., read/select from data records). In contrast, PNM adds computational capabilities at the channel level, like with CXL Computational Memory (CCM). It can be readily deployed. Some open question from industry include which specific functionality to integrate, are specialized cores or accelerators only adequate or is there a need for general purpose cores, and if so what kind.

Answering these questions requires consideration of use cases, and systems support for programming and integration of PNM in the OS and memory management stack. Several use cases were discussed.

Memory properties: PNM enhances the memory properties with new capabilities, such as for (de-)/compression, quantization, encryption/decryption, etc;

Data movement: PNM adds support for functionality common in data movement operations, such as for memory copying, initialization, etc. One tradeoff is that offloading these operations bypasses the CPU cache.

Data analytics: Existing industry prototypes already demonstrate the benefits of operations such as scan, filter, similarity, nearest neighbor, etc., common in data analytics applications.

LLM inference: Existing industry prototypes also demonstrate benefits for offloading select operations common in LLM inference; this is particularly useful for mobile or edge devices due to potential improvements in energy efficiency.

A simple approach to exposing the PNM capabilities is through use of optimized libraries, similarly to what has been done for other hardware offload and acceleration engines. An interface that seems to generalize well to different use case is that of a function call with a pointer to data. In addition to programming support, there is need for compiler or runtime support to navigate tradeoffs in terms of function placement (e.g., to offload or not). In addition, even for fixed function computations, there may be need for data reshaping, to deal with specific requirements for data layout/data format support, for instance to offset

to appropriate data element in record, or when computation requires aggregation across memory channels (e.g., pointer to other memory, traversal of graph data structures, etc.). Finally, the programmability raises decisions on the control-plane interface, i.e., how does one program PNM. The group discussed whether eBPF provides a sufficiently adequate extension model, and observed additional work may be required.

Another set of important design decisions for PNM rests with the integration with the operating system, in terms of addressing, access control, reliance on TLB and MMU hardware, etc. One important question that must be considered is whether computational capabilities be integrated with all memory or only some parts of it. In the latter case, new support is needed to present a united view of memory with different capabilities (including local DRAM + CCM).

The working group discussed some of the possible design points and associated tradeoffs. Some of the breakout sessions in the later part of the seminar focused on a deep dive into some of these aspects (e.g., programming APIs and virtual memory integration).

4.5 Report of the Working Group on Drivers for Disruptive Memory Technologies

Jana Giceva (TU München – Garching, DE), Oana Balmau (McGill University – Montréal, CA), Jerónimo Castrillón-Mazo (TU Dresden, DE), Thaleia Dimitra Doudali (IMDEA Software Institute – Madrid, ES), Tajana Simunic Rosing (University of California – San Diego, US), Nandita Vijaykumar (University of Toronto, CA), and Huanchen Zhang (Tsinghua University – Beijing, CN)

The working group evaluated the application-level drivers from various domains to determine how they may influence the design and benefit from the development of future disruptive memory technologies. We discussed various aspects from performance bottlenecks across a diverse set of domains to current painful points and limitations. The main pursuit was identifying a common set of challenges. This short summary structures our discussion in two parts: the first one characterizes the most common domain workloads and their memory-related challenges, and the second one then tries to map some of these challenges to specific memory technologies that could help address them.

4.5.1 Domain Workload Analysis

The group discussed several classes of applications and domains (machine and deep learning, computer vision, data management, and genomics.) trying to identifying common and domain-specific memory bottlenecks:

Deep Learning and Large Language Models (LLMs)

- Training is in general quite memory-intensive and demands high-bandwidth.
- Inference can sometimes involve extremely large key-value (KV) caches, particularly in long-context or multi-session workloads.

Databases

- Vector databases and retrieval-augmented generation (RAG) systems, may require frequent updates to vector indices
- HTAP (Hybrid Transactional/Analytical Processing) systems need efficient support for in-memory updates without compromising snapshot consistency for analytics.
- Housekeeping tasks like index building, data partitioning, garbage collection, generating consistent snapshots from the logs, etc. are quite memory intensive and should not have to burn user-facing CPU cycles.

Genomics, Proteomics and Metabolomics

- Omic workloads are characterized by high-dimensional, large-volume datasets across heterogeneous formats.
- Preprocessing stages are costly and can last a few hours or even days.
- Prior to database search, the data needs to be clustered, which is currently impossible on regular basis due to large performance overhead
- Database search is massively parallel, focused on exact and nearest neighbour pattern match. For genomics alignment, dynamic programming and/or graph based methods are needed as well. For proteomics and metabolomics, extremely fast nearest neighbor pattern match is key.

Visual Computing

- 3D vision and interactive applications are sensitive to memory latency, especially for real-time training and inference.
- Generative AI for visual data (e.g., scene generation) requires fast, low-latency access to long sequences of data.

Graph Applications

- Graph traversal, pattern matching, and dynamic analytics require memory systems capable of handling irregular accesses, and with frequent updates also a consistent snapshot support.

4.5.2 Technology Mapping and Opportunities

In the second part of the session, the group discussed how emerging memory technologies could address some of the identified challenges:

CXL (Compute Express Link)

Enables memory pooling and scale-up architectures. These could be of general use for any domain that is constrained on the memory capacity (e.g., large scale data analytics, graph processing, etc.). With memory-extensions provided by CXL one can also support larger KV caches for LLM inference (10s of TBs possible), potentially extending down into storage layers. May alleviate memory bottlenecks in long-context generative workloads, including video generation and long sequence modeling in AI applications. The group did not identify many use-cases for coherency, or rather a need for coherency beyond small areas that are needed for coordination (e.g., cross-partition transactions in distributed databases).

Processing-in-Memory (PIM), Near-Data Processing (NDP), and Near-Storage Computing

These technologies are considered a great match for workloads that aim to reduce data movement. Some specific tasks that we identified may benefit the most are the following:

- Pattern matching (e.g., database search, nearest neighbor queries),
- Clustering (e.g. prior to database search for omics disciplines, for unsupervised learning)
- Irregular compute (pointer chasing),
- Housekeeping tasks (e.g., garbage collection, compaction, ML preprocessing),
- Sensor data summarization,
- Memory-bound operations like hash tables and joins in databases

Useful for omics disciplines, graph analytics, and databases for restructuring and lightweight computation close to data.

One potential concern was raised by the data management side, that accelerating individual tasks with PIM/PNM may have diminishing returns for workloads that exhibit a frequent set of accesses to common data regions and may in the long run benefit from caching. So the interplay between caching and processing close to where the original data resides may need to be explored with care.

Checkpointing and Persistent Memory

Managed Retention Memory (MRM) was brought up as a novel approach to simplifying checkpointing by enabling memory regions with flexible durability semantics.

4.6 Report of the Working Group on Programming Models for Disruptive Memory Technologies

Tianzheng Wang (Simon Fraser University – Burnaby, CA), Jerónimo Castrillón-Mazo (TU Dresden, DE), Birte Kristina Friesel (Universität Osnabrück, DE), Jana Giceva (TU München – Garching, DE), Yu Hua (Huazhong University of Science & Technology, CN), Kimberly Keeton (Google LLC – South San Francisco, US), Wolfgang Lehner (TU Dresden, DE), Kai-Uwe Sattler (TU Ilmenau, DE), Kevin Skadron (University of Virginia – Charlottesville, US), Olaf Spinczyk (Universität Osnabrück, DE), and Huanchen Zhang (Tsinghua University – Beijing, CN)

Traditional memory programming models (e.g., malloc, memcpy, madvise) have provided a robust abstraction for decades of general purpose computing. However, emerging memory technologies – including processing-in/near-memory, persistent memory or memory disaggregation via novel coherent protocols like CXL – expose fundamentally different performance characteristics (latency, bandwidth, granularity of accesses) and capabilities (coherency, ordering guarantees, persistency or even compute). These break the long-standing assumptions about address-spaces, pointer validity, update semantics, fault handling and expose a world where data and memory need to be decoupled from the traditional process-centric view/abstraction of a machine.

This report summarizes a recent discussion aiming to define new abstraction layers that can better support the development (and maintenance) of complex data-intensive applications (e.g., database systems and large-scale data processing) on top of modern memory technologies.

4.6.1 Limitations of existing programming models

Current models provide only rudimentary support for the emerging memory landscape:

- Device-specific features: traditional interface APIs like malloc do not distinguish between memories with or without compute capabilities, nor do they express memory locations (local, remote, disaggregated), volatility or performance (latency, bandwidth).
- Device-specific constraints: some compute capabilities near/in memory may not share the global address space and existing memory semantics do not translate easily. For example, UPMEM's PIM requires manual partitioning and data movement.
- Disaggregated memory challenges: explicit data movement and non-cache coherence can be challenging to work with. It is also not clear how to capture the fault model and the guarantees (atomicity, ordering) that can be supported.
- Inflexible abstractions: lack the flexibility to express requirements and/or to allow applications to adapt and fully leverage the features of the emerging technologies.
- Lifetime of allocated objects: may not fit the traditional process-based model. For example the concept of a pointer in heterogeneous memory systems is complex analogous to persistent memory and the need for pointer swizzling, as virtual addresses might not be consistent after reboots or remapping.

4.6.2 Towards new-abstractions and programming models

We argue that now is a good time to embrace a declarative model, in which application developers specify what they want, and the runtime/compiler decides how and where it should run. This allows for a separation of concerns related to the execution runtime, and decouples the details of technology's capabilities and its relevant optimizations from the development of the application logic.

Similar ideas have been explored in the past for partitioned global address space (e.g., X10) or Intel's library/runtime for TBB which defers the placement/scheduling to the runtime.

We expect that this should enable:

1. better optimization across heterogeneous compute/memory sub-systems;
2. decoupling of the device-specific logic from the high-level application code;
3. simplify the programming for future, unknown architectures.

To achieve that we propose a two layer approach, where:

- A logical region-based abstraction layer is used by application developers to declaratively mark their requirements (e.g., persistent, encrypted, high bandwidth, coherent, etc.).
- A calibration layer captures the device characteristics (e.g., through reading through the specs and benchmarking) and stores the results in a catalog consumed by optimizers, compilers, and the runtime system's scheduler.

Upon a short analysis and discussion we concluded that a dataflow programming model can be a good fit for many data-intensive applications such as databases, streaming systems, big-data frameworks like Spark, and possibly also other ML-systems.

The key idea behind such a dataflow programming model is to explicitly model the (1) compute operators (kernels) and the (2) data transfers into the dataflow graph. One can mark the pipelines of operations that can be potentially merged (operator fusion) if the

data-transfer allows for streaming the data. In case of pipeline-breakers, one can explicitly mark the state (e.g., the hash-table).

This also can naturally support annotations, where developers can annotate performance/security goals of their program while the system can infer other requirements (e.g., persistence, memory locality, coherence, etc.).

Once the annotations are in place, the optimizer/runtime can select operators and memory regions using the cost model from the catalog of the calibration layer. For example, if the operational intensity is low it may choose to run the operation within a PIM/PNM device. Once the state and data transfer is explicit we can also reason about required data transformations.

4.6.3 Taxonomy of Memory Properties

To facilitate reasoning about the calibration layer and the type of properties that application developers may need to annotate (or the compilation framework to infer), we also thought of a list of taxonomy of memory properties:

Property	Examples
Performance	(latency, bandwidth)
Volatility	(non-volatile, volatile)
Active/Passive	(active (PIM) vs normal DRAM)
Location	local, NUMA, remote (CXL or RDMA)
Granularity	Byte-addressable vs. page-based (and size of packets/pages)
Coherency	(coherent vs. non-coherent)
Security	(encrypted or not)
Compression	(type of compression)
Ordering	(strong/weak memory model, explicit fencing, causal consistency, etc.)
Fault model	(atomic execution, exactly once, transient or partial fault, etc.)

The specific technologies that were discussed were:

Processing-in-memory (PIM): advantages for data processing is that it will reduce data movement and may result in better performance for workloads with sequential access patterns. The limitations we see is that it may not be beneficial for workloads that have strong data reuse and/or non-sequential patterns. Examples: UPMEM requires explicit data movements; lacks global address space; in the cloud set-up pushing predicate evaluation down to storage (e.g., with AWS S3-select) had problems with caching, and on a workload-level over longer period of time there were diminishing returns for the one-off performance improvement.

Memory extensions/pooling (CXL): offers good latency, coherency, but may introduce pointer consistency issues (what if a link fails?). It is not clear how much of the techniques and insights from RDMA will play a role in the context of CXL and how it should be used. For example, many benefits for remote memory pooling (over RDMA) can already be seen as reported by production systems like Google's BigQuery shuffling layer. At the same time, the lack of ordering guarantees on RDMA writes showed that it is impractical for synchronization purposes. How much of these can be avoided (or be customized/controlled) with novel protocols like CXL (or photonics) is to be seen. Nevertheless, the broader systems community already explored some of these questions in the context of far-memory and one needs to check them to see how it may influence the design of the proposed programming model.

Participants

- Gustavo Alonso
ETH Zürich, CH
- Oana Balmau
McGill University – Montréal, CA
- Antonio Barbalace
University of Edinburgh, GB
- Frank Bellosa
KIT – Karlsruher Institut für Technologie, DE
- Daniel Berger
Microsoft – Redmond, US
- Jerónimo Castrillón-Mazo
TU Dresden, DE
- Haibo Chen
Shanghai Jiao Tong University, CN
- Jian-Jia Chen
TU Dortmund, DE
- David Cohen
Solidigm – Santa Fe, US
- Christian Dietrich
TU Braunschweig, DE
- Thaleia Dimitra Doudali
IMDEA Software Institute – Madrid, ES
- Michal Friedman
ETH Zürich, CH
- Birte Kristina Friesel
Universität Osnabrück, DE
- Ada Gavrilovska
Georgia Institute of Technology – Atlanta, US
- Jana Giceva
TU München – Garching, DE
- Boris Grot
University of Edinburgh, GB
- Timo Hönig
Ruhr-Universität Bochum, DE
- Yu Hua
Huazhong University of Science & Technology – Wuhan, CN
- Sudarsun Kannan
Rutgers University – Piscataway, US
- Sanidhya Kashyap
EPFL – Lausanne, CH
- Kimberly Keeton
Google LLC – South San Francisco, US
- Hoshik Kim
SK hynix – San Jose, US
- Wolfgang Lehner
TU Dresden, DE
- Alberto Lerner
University of Fribourg, CH
- Till Miemietz
Barkhausen Institut – Dresden, DE
- Onur Mutlu
ETH Zürich, CH
- Ilia Petrov
Hochschule Reutlingen, DE
- Tilmann Rabl
Hasso-Plattner-Institut, Universität Potsdam, DE
- Tajana Simunic Rosing
University of California – San Diego, US
- Kai-Uwe Sattler
TU Ilmenau, DE
- Wolfgang Schröder-Preikschat
Universität Erlangen-Nürnberg, DE
- Kevin Skadron
University of Virginia – Charlottesville, US
- Olaf Spinczyk
Universität Osnabrück, DE
- Michael Swift
University of Wisconsin-Madison, US
- Jürgen Teich
Universität Erlangen-Nürnberg, DE
- Nandita Vijaykumar
University of Toronto, CA
- Tianzheng Wang
Simon Fraser University – Burnaby, CA
- Zeke Wang
Zhejiang University – Hangzhou, CN
- Thomas Willhalm
Intel Deutschland GmbH – Feldkirchen, DE
- Youjip Won
KAIST – Daejeon, KR
- Chun Jason Xue
MBZUAI – Abu Dhabi, AE
- Huanchen Zhang
Tsinghua University – Beijing, CN