Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

ABSTRACT

Computing is bottlenecked by data. Large amounts of application data overwhelm 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 invited special session talk, we describe three major shortcomings of modern architectures in terms of 1) dealing with data, 2) taking advantage of the 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 show that handling data well requires designing architectures based on three key principles: 1) data-centric, 2) data-driven, 3) data-aware. We give several examples for how to exploit each of these principles to design a much more efficient and high 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 novel directions: 1) *processing using memory*, which exploits analog operational properties of memory chips to perform massively-parallel operations in memory, with low-cost changes, 2) *processing near memory*, which integrates sophisticated additional processing capability in memory controllers, the logic layer of 3D-stacked memory technologies, or memory chips to enable high memory bandwidth and low memory latency to near-memory logic. 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 guiding principles for future computing architecture and system designs. This accompanying short paper provides a summary of the invited talk and points the reader to further work that may be beneficial to examine.

I. INTRODUCTION

Existing computing systems process increasingly large amounts of data. Data is key for many modern (and likely even more future) workloads and systems. Important workloads (e.g., machine learning, artificial intelligence, genome analysis, graph analytics, databases, video analytics, online collaboration), whether they execute on cloud servers or mobile systems are all data intensive; they require efficient processing of large amounts of data. Today, we can generate more data than we can process, as exemplified by the rapid increase in the data obtained in astronomy observations and genome sequencing [1].

Unfortunately, the way they are designed, modern computers are not efficient at dealing with large amounts of data: large amounts of application data greatly overwhelm the storage capability, the communication capability, and the computation capability of the modern machines we design today. As such, data becomes a large performance and energy bottleneck, and it greatly impacts system robustness and security as well. As a prime example, we provide evidence that the potential for new genome sequencing technologies, such as nanopore sequencing [2, 113], is greatly limited by how fast and how efficiently we can process the huge amounts of genomic data the underlying technology can provide us with [3, 83, 113, 119, 143]. A similar observation can also be made for video analytics [163, 7] and machine learning [198-199, 7].

The processor-centric design paradigm (and the resulting processor-centric execution model) of modern computing systems is one prime cause of why data overwhelms modern machines [4, 5, 120]. With this paradigm, there is a clear dichotomy between processing and memory/storage: data has to be brought from storage and memory units to computation units (e.g., general-purpose processors or special-purpose accelerators), which are far away from the memory/storage units, before any processing can be done on the data. The dichotomy exists at the macro-scale (e.g., across the internet) as well as the micro-scale (e.g., within a single compute node, or even within a single CPU processing core). This processor-memory dichotomy leads to large amounts of data movement across the entire computing system, degrading performance and expending large amounts of energy. For example, a recent work [7] shows that more than 60% of the entire mobile system energy is spent on data movement across the memory hierarchy when executing four major commonly-used consumer workloads, including machine learning inference, video processing and playback, and web browsing. Similarly, due to the current processor-centric design paradigm, a large fraction of the system resources is dedicated to units that store and move data (i.e., to serve the computation units), and actual computation units constitute only ~5% of an entire processing node [8] – yet, even then, data access is still a major bottleneck due to the large latency and energy costs of accessing large amounts of data.

II. FUNDAMENTAL PRINCIPLES

Our starting axiom for an intelligent architecture is that it should handle (i.e., store, access, and process) data well. But, what does it mean for an architecture to handle data well? We posit (and later demonstrate with examples) that the answer lies in satisfying three major desirable properties (or principles): 1) data-centric, 2) data-driven, and 3) data-aware.

First, the system should ensure that data does not overwhelm its components. Doing so requires effort in intelligent algorithms, intelligent architectures and intelligent whole system designs that are co-optimized cross-layer (i.e., optimizations spanning across algorithms-architectures-devices), in a manner that puts data and its processing at the center of the design, minimizing data movement and maximizing the efficiency with which data is handled, i.e., stored, accessed, and processed (e.g., as exemplified in [4-38, 120]). We call this first principle *data-centric architectures*.

Second, an intelligent architecture takes advantage of the vast amounts of data and metadata that flow through the system, to continuously improve its decision making, by bettering both its policies and mechanisms based on online

978-3-9819263-5-4/DATE21/©2021 EDAA

learning and self-optimization. In other words, the architecture should make data-driven, self-optimizing decisions in its components (e.g., as exemplified in [39-51, 121]). We call this second principle *data-driven architectures*.

Third, an intelligent architecture understands and exploits various properties of each piece of data so that it can improve and adapt its algorithms, mechanisms, and policies based on the characteristics of data. In other words, the architecture should make *data-characteristics-aware* decisions in its components and across the entire system (e.g., as exemplified in [52-58, 107, 116, 11, 149]). We call this third principle *data-aware architectures*.

III. Existing Computing Architectures

Based on our qualitative and quantitative analyses, we find that existing computing architectures greatly fall short of handling data well. In particular they violate all of the three major desirable principles. We analyze each briefly next.

First, modern architectures are poor at dealing with data: they are designed to mainly store and move data, as opposed to actually compute on the data. Most system resources serve the processor (and accelerators) without being capable of processing data. As such, existing architectures are *processor-centric* as opposed to *data-centric*: they place the most value in the processor (not data) and everything else in the system is viewed as secondary serving the processor. We believe this is the wrong mindset and approach in designing a balanced system that handles data well: such a system should be data centric: i.e., data should be the prime thing that is valued and everything else in the system should be designed to 1) minimize data movement by enabling computation capability at and close to where data resides and 2) maximize the value and efficiency of processing data by enabling low-latency and low-energy access to as well as low-energy and low-cost storage of vast amounts of data. Doing so would eliminate the huge data access bottleneck of processor-centric systems, thereby improving performance, reducing energy consumption, alleviating off-chip bandwidth requirements (and hence area and cost), likely reducing system and hardware design complexity, as well as opening up new opportunities for improving system security and reliability by handling data more locally in or near where it resides.

Second, modern architectures are poor at taking advantage of vast amounts of data (and metadata) available to them during online operation and over time. They are designed to make simple decisions based on fixed policies, ignoring massive amounts of easily-available data. This is because existing architectural policies make *human-driven* decisions as opposed to *data-driven* decisions, and humans, by nature, do not seem capable of designing policies and heuristics that consider hundreds, if not thousands, of different state attributes that may be useful to examine in a control policy that makes dynamic decisions. It is instructive to notice that a modern memory controller, for example, keeps executing *exactly* the same *fixed* policy for scheduling or power management (e.g., FR-FCFS [59, 60], PAR-BS [61] or some other heuristic-based policy [62-73, 117-118, 122-133]), during the *entire lifetime* of a system (for many many years!), regardless of the positive or negative impact of the decisions resulting from the policy at any given point of time on the system. The same is true for a modern prefetch controller, a cache controller, a network controller, and for many other hardware controllers in a system (e.g., [150-162, 200-214]). Each controller sees a vast amount of data and makes a vast number of decisions even in the timeframe of a single millisecond (let alone years), yet it is incapable of learning from that data and changing its policy to another dynamically-determined better policy because the policy it follows is rigid and hardcoded by a human. This is clearly not intelligent: for example, as humans, we have the capability to learn from the past and adapt our actions accordingly to not repeat the same mistakes as in the past or to choose the best policy/actions that we believe will provide the highest benefits in the future. Enabling similar intelligence and far-sightedness in controller and system policies in an architecture is necessary for obtaining good performance and efficiency (as well as better reliability, security and perhaps other metrics) under a variety of system conditions and workloads.

Third, modern architectures are poor at knowing and exploiting different properties of application and system data. They are designed to treat all data as the same (except for a small set of specialized hints that provide some opportunity to optimize based on data characteristics in a limited manner that is very specific to the particular optimization). As such, the decisions existing architectures make are *component-aware* decisions as opposed to *data-aware* decisions: a component's (e.g., a cache's or a memory controller's) structural and performance characteristics dominate the policies designed to control that component and the accessed/manipulated data's characteristics are rarely conveyed to the policies or even known. If the characteristics of the data to be accessed or manipulated were known, the decisions taken could be very different: for example, if we knew the relative compressibility of different types of data, e.g., different data types or different objects [55, 74-81, 135-138], different components in the entire system could be designed in a manner that adaptively scales their capability to match the compressibility of different data elements, in order to maximize both performance and efficiency. Modifying the architecture and its interface to become richer and more expressive, and to include rich and accurate information on various properties of data that is to be processed, is therefore critical to customizing the architecture to the characteristics of the data and, thus, enabling intelligent adaptation of system policies to data characteristics.

IV. Intelligent Computing Architectures

A major chunk of our invited talk describes in detail the characteristics of an intelligent computing architecture, by concrete examples and their empirical evaluation. This short paper does not go into detail, but provides a brief overview with references to other works that exemplify such architectures. Multiple detailed versions of this talk can be found online [82, 139-142]. We also refer the reader to recent detailed survey and overview papers we have written on the topic [120, 4].

Data-Centric

A data-centric architecture has at least four major characteristics. First, it enables processing capability in or near where data resides (i.e., in or near memory structures), as described in detail in [4-6, 8, 38, 120] and exemplified by [7-12, 14, 19, 20, 24, 27, 30, 34, 84, 108-113, 144-147]. Second, it provides low-latency and low-energy access to data, as exemplified by [11-13, 15-18, 21, 23, 31-33, 84-86]. Third, it enables low-cost data storage and processing (i.e., high capacity memory at low cost, via techniques like new memory

technologies, hybrid memory systems and/or compressed memory systems), as exemplified by [22, 87-96, 74, 76, 78, 107, 116]. Fourth, it provides mechanisms for intelligent data management (with intelligent controllers handling robustness, security, cost, etc.), as described in detail in [97-103, 116, 120] and exemplified by, e.g., [104-106, 116, 120, 179-190].

Our talk provides significant detail on providing processing capability in or near where data resides, focusing on processing in memory (PIM). There is a pressing need for enabling PIM in modern systems due to 1) *a bottom-up push*, i.e., circuit- and device-level memory technology scaling issues requiring intelligent main memory controllers to solve low-level scaling and reliability challenges, such as RowHammer [104-106, 99, 102], data retention [21, 167-170, 97, 191-193], energy consumption [171-172, 127, 132], enabling scalable emerging technologies [22, 87-93, 172-174] and 2) *a top-down pull*, i.e., systems and applications requiring near-data processing capability with minimal data movement to reduce the data access bottleneck and its large negative effect on performance [154-155, 164-165], energy [7, 166], and sustainability.

There are at least two new approaches to enabling processing-in-memory in modern systems. The first approach, *processing using memory (PUM)*, exploits the existing memory architecture and the operational principles of the memory circuitry to enable operations inside memory structures with minimal changes. PUM makes use of intrinsic properties and operational principles of the memory cells and cell arrays, by inducing interactions between cells such that the cells and/or cell arrays can perform useful computation. PUM architectures enable a wide range of different functions, such as data copy/initialization, bitwise operations, and simple arithmetic operations. We focus on how to minimally and practically change DRAM chips to perform fast and energy-efficient bulk data copy and initialization [84, 12, 147, 175] as well as bulk bitwise operations [6, 10, 109, 175]. Similar approaches are also applicable to SRAM, MRAM, RRAM and other NVM technologies [176-178].

The second approach, *processing near memory (PNM)*, involves adding or integrating computation units (e.g., accelerators, simple processing cores, reconfigurable logic) close to or inside the memory. Computation units can be placed in the logic layer of 3D-stacked memories, in the memory controller, or even inside memory chips. Recent advances in silicon interposers (in-package wires that connect directly to the through-silicon vias in a 3D-stacked chip) also allow for separate logic chips to be placed in the same die package as a 3D-stacked memory while still taking advantage of the TSV bandwidth.

Both PUM and PNM approaches can greatly accelerate real applications, including database systems, graph analytics, machine learning, genome analysis, GPU workloads, pointer-chasing-intensive workloads, data analytics, climate modeling, etc. Recent results show up to approximately two orders of magnitude improvement in energy and performance over conventional processor-centric systems. More functionality can be potentially integrated into a memory chip using PNM than using PUM, but both approaches can be combined to get even higher benefit from PIM. For both approaches, we describe and tackle relevant cross-layer research, design, and practical adoption challenges in devices, architecture, systems, and programming models in our talk. Our recent PIM overview work comprehensively analyzes modern PIM systems and issues [120, 4].

Data-Driven

A data-driven architecture enables the machine itself to learn the best policies for managing itself and executing programs. Controllers in such an architecture, when needed, are data-driven autonomous agents that automatically learn far-sighted policies. A prime example of such a controller is the *reinforcement learning based self-optimizing memory controllers* [39]. Such controllers can not only improve performance and efficiency under a wide variety of conditions and workloads but also reduce the hardware and system designer's burden in designing sophisticated controllers [39]. We believe an intelligent architecture will consist of a collection of such intelligent controllers that perform automatic data-driven online policy learning, including learning of how to best coordinate with each other to make decisions that benefit the overall system. Such machines learn the best policies over time and thus become better as they learn, adapting, evolving, and executing far-sighted policies. To enable such a machine, we need to revisit the design of all controllers (e.g., caching, prefetching, storage, memory, interconnect) and turn them into data-driven agents.

Data-Aware

A data-aware architecture understands what it can do with and to each piece of data (and associated computations on data), and uses this information about data characteristics to maximize system efficiency and performance. In other words, it customizes itself (i.e., its policies and mechanisms) to the characteristics of the data and computations it is dealing with. Such an architecture requires knowledge of various characteristics of different data elements and structures as well as computations. Many semantic or other characteristics of data (e.g., compressibility, approximability, sparsity, criticality, access and security semantics, locality, latency vs. bandwidth sensitivity, privacy requirements, data types, error vulnerability) are invisible or unknown to modern hardware and thus need to be communicated or discovered. We believe efficient and expressive software/hardware interfaces and resulting cross-layer mechanisms, as exemplified by X-Mem (Expressive Memory) [52, 53] and the Virtual Block Interface [56] as well as other works [54, 55, 57, 58, 107, 116, 11], are promising and critically-needed approaches to creating general-purpose data-aware architectures.

ACKNOWLEDGMENTS

An earlier version of this talk was delivered as a plenary keynote talk at the VLSI-DAT/TSA conferences [142], with an accompanying paper [148], which this paper is an extension of. The very first version of this talk was delivered as a keynote talk at the SRC-Mubadala-Khalifa Forum on The Future of Artificial Intelligence Hardware Systems in April 2019. We thank all of the members of the SAFARI Research Group, and our collaborators at Carnegie Mellon, ETH Zurich, and other universities, who have contributed to the various works we describe in this paper. Thanks also goes to our research group's industrial sponsors over the past ten years, especially ASML, Google, Huawei, Intel, Microsoft, NVIDIA, Samsung, Seagate, SRC, and VMware, who have supported various pieces of research that are described in this paper and the associated talk.

REFERENCES

[1] Z. D. Stevens et al., "Big data: astronomical or genomical?", PLoS Biology, 2015.

[2] D. Senol Cali et al., "Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions" BIB 2019.

[3] O. Mutlu, "Accelerating Genome Analysis: A Primer on an Ongoing Journey", Keynote Talk at HiCOMB-17, 2018.

[4] S. Ghose et al., "Processing-in-Memory: A Workload-Driven Perspective", IBM JRD 2019.

[5] O. Mutlu et al., "Processing Data Where It Makes Sense: Enabling In-Memory Computation", MICPRO, 2019.

[6] V. Seshadri and O. Mutlu, "In-DRAM Bulk Bitwise Execution Engine", Advances in Computers, 2020.

[7] A. Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks", ASPLOS 2018.

[8] O. Mutlu, "Enabling Computation with Minimal Data Movement: Changing the Computing Paradigm for High Efficiency", Design Automation Summer School Lecture, DAC 2019. https://people.inf.ethz.ch/omutlu/pub/onur-DAC-DASS-EnablingInMemoryComputation-June-2-2019.pptx

[9] J. Ahn et al., "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing", ISCA 2015.

[10] V. Seshadri et al., "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology", MICRO 2017.

[11] H. Luo et al., "CLR-DRAM: A Low-Cost DRAM Architecture Enabling Dynamic Capacity-Latency Trade-Off", ISCA 2020.

[12] K. Chang et al., "Low-Cost Inter-Linked Subarrays (LISA): Enabling Fast Inter-Subarray Data Movement in DRAM", HPCA 2016.

[13] D. Lee et al., "Adaptive-Latency DRAM: Optimizing DRAM Timing for the Common-Case", HPCA 2015.

[14] K. Hsieh et al., "Accelerating Pointer Chasing in 3D-Stacked Memory: Challenges, Mechanisms, Evaluation", ICCD 2016.

[15] K. Chang et al., "Understanding Latency Variation in Modern DRAM Chips: Experimental Characterization, Analysis, and Optimization", SIGMETRICS 2016.

[16] K. Chang et al., "Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mechanisms", SIGMETRICS 2017.

[17] D. Lee et al., "Design-Induced Latency Variation in Modern DRAM Chips: Characterization, Analysis, and Latency Reduction Mechanisms", SIGMETRICS 2017.

[18] S. Ghose et al., "What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study", SIGMETRICS 2018.

[19] K. Hsieh et al., "Transparent Offloading and Mapping (TOM): Enabling Programmer-Transparent Near-Data Processing in GPU Systems", ISCA 2016.

[20] J. Ahn et al., "PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture", ISCA 2015.

[21] J. Liu et al., "RAIDR: Retention-Aware Intelligent DRAM Refresh", ISCA 2012.

[22] B. C. Lee et al., "Architecting Phase Change Memory as a Scalable DRAM Alternative", ISCA 2009.

[23] D. Lee et al., "Decoupled Direct Memory Access: Isolating CPU and IO Traffic by Leveraging a Dual-Data-Port DRAM", PACT 2015.

[24] V. Seshadri et al., "Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses", MICRO 2015.

[25] D. Lee et al., "Simultaneous Multi-Layer Access: Improving 3D-Stacked Memory Bandwidth at Low Cost", TACO 2016.

[26] H. Hassan et al., "ChargeCache: Reducing DRAM Latency by Exploiting Row Access Locality", HPCA 2016.

[27] M. Hashemi et al., "Accelerating Dependent Cache Misses with an Enhanced Memory Controller", ISCA 2016.

[28] M. Patel et al., "The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions", ISCA 2017.

[29] S. Khan et al., "Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content", MICRO 2017.

[30] J. S. Kim et al. "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies", BMC Genomics 2018.

[31] A. Das et al., "VRL-DRAM: Improving DRAM Performance via Variable Refresh Latency", DAC 2018.

[32] J. S. Kim et al., "Solar-DRAM: Reducing DRAM Access Latency by Exploiting the Variation in Local Bitlines", ICCD 2018.

[33] Y. Wang et al., "Reducing DRAM Latency via Charge-Level-Aware Look-Ahead Partial Restoration", MICRO 2018.

[34] J. S. Kim et al., "D-RaNGe: Using Commodity DRAM Devices to Generate True Random Numbers with Low Latency and High Throughput", HPCA 2019.

[35] H. Hassan et al., "CROW: A Low-Cost Substrate for Improving DRAM Performance, Energy Efficiency, and Reliability", ISCA 2019.

[36] S. Song et al., "Improving Phase Change Memory Performance with Data Content Aware Access", ISMM 2020.

[37] G. Singh et al., "NAPEL: Near-Memory Computing Application Performance Prediction via Ensemble Learning", DAC 2019.

[38] O. Mutlu et al., "Enabling Practical Processing in and near Memory for Data-Intensive Computing", DAC 2019.

[39] E. Ipek et al. "Self Optimizing Memory Controllers: A Reinforcement Learning Approach", ISCA 2008.

[40] D. A. Jimenez and C. Lin, "Dynamic Branch Prediction with Perceptrons," HPCA 2001.

[41] D. A. Jimenez, "Fast Path-Based Neural Branch Prediction", MICRO 2003.

[42] D. A. Jimenez, "Piecewise Linear Branch Prediction", ISCA 2005.

[43] D. A. Jimenez, "An optimized scaled neural branch predictor", ICCD 2011.

[44] E. Teran et al., "Perceptron learning for reuse prediction", MICRO 2016.

[45] E. Garza et al., "Bit-level perceptron prediction for indirect branches", ISCA 2019.

[46] E. Bhatia et al., "Perceptron-based prefetch filtering", ISCA 2019.

[47] L. Peled et al., "A Neural Network Prefetcher for Arbitrary Memory Access Patterns", TACO 2020.

[48] L. Peled et al., "Semantic locality and context-based prefetching using reinforcement learning," ISCA 2015.

[49] R. Bitirgen et al., "Coordinated management of multiple interacting resources in chip multiprocessors: A machine learning approach", MICRO 2008.

[50] J. Mukundan and J. F. Martinez, "MORSE: Multi-objective reconfigurable self-optimizing memory scheduler", HPCA 2012.

[51] J. F. Martinez and E. Ipek, "Dynamic Multicore Resource Management: A Machine Learning Approach", IEEE Micro 2009.

[52] N. Vijaykumar et al., "A Case for Richer Cross-layer Abstractions: Bridging the Semantic Gap with Expressive Memory", ISCA 2018.

[53] N. Vijaykumar et al., "The Locality Descriptor: A Holistic Cross-Layer Abstraction to Express Data Locality in GPUs", ISCA 2018.

[54] S. Koppula et al., "EDEN: Enabling Energy-Efficient, High-Performance Deep Neural Network Inference Using Approximate DRAM", MICRO 2019.

[55] K. Kanellopoulos et al., "SMASH: Co-designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations", MICRO 2019.

[56] N. Hajinazar et al., "The Virtual Block Interface: A Flexible Alternative to the Conventional Virtual Memory Framework", ISCA 2020.

[57] Z. Yu et al., "Labeled RISC-V: A New Perspective on Software-Defined Architecture", CARRV 2017.

[58] J. Ma et al., "Supporting Differentiated Services in Computers via Programmable Architecture for Resourcing-on-Demand (PARD)", ASPLOS 2015.

[59] S. Rixner et al., "Memory access scheduling", ISCA 2000.

[60] W. K. Zuravleff and T. Robinson, "Controller for a synchronous DRAM that maximizes throughput by allowing memory requests and commands to be issued out of order", U.S. Patent Number 5,630,096, May 1997.

[61] O. Mutlu and T. Moscibroda, "Parallelism-Aware Batch Scheduling: Enhancing both Performance and Fairness of Shared DRAM Systems", ISCA 2008.

[62] H. Usui et al., "DASH: Deadline-Aware High-Performance Memory Scheduler for Heterogeneous Systems with Hardware Accelerators", TACO 2016.

[63] O. Mutlu and T. Moscibroda, "Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors", MICRO 2007.

[64] Y. Kim et al., "ATLAS: A Scalable and High-Performance Scheduling Algorithm for Multiple Memory Controllers", HPCA 2010.

[65] Y. Kim et al., "Thread Cluster Memory Scheduling: Exploiting Differences in Memory Access Behavior", MICRO 2010.

[66] I. Hur and C. Lin, "Adaptive History-Based Memory Schedulers", MICRO 2004.

[67] I. Hur and C. Lin, "Memory scheduling for modern microprocessors", ACM TOCS 2007.

[68] C. Natarajan et al., "A study of performance impact of memory controller features in multi-processor server environment", WMPI 2004.

[69] S. Rixner, "Memory controller optimizations for web servers", MICRO 2004.

[70] L. Subramanian et al., "BLISS: Balancing Performance, Fairness and Complexity in Memory Access Scheduling", IEEE TPDS 2016.
[71] L. Subramanian et al., "The Blacklisting Memory Scheduler: Achieving High Performance and Fairness at Low Cost", ICCD 2014.
[72] R. Ausavarungnirun et al., "Staged Memory Scheduling: Achieving High Performance and Scalability in Heterogeneous Systems", ISCA 2012.
[73] K. J. Nesbit et al., "Fair Queuing Memory Systems", MICRO 2006.
[74] G. Pekhimenko et al., "Base-Delta-Immediate Compression: Practical Data Compression for On-Chip Caches", PACT 2012.
[75] N. Vijaykumar et al., "A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps", ISCA 2015.
[76] G. Pekhimenko et al., "Linearly Compressed Pages: A Low-Complexity, Low-Latency Main Memory Compression Framework", MICRO 2013.
[77] G. Pekhimenko et al., "A Case for Toggle-Aware Compression for GPU Systems", HPCA 2016.
[78] M. Ekman and P. Stenstrom, "A Robust Main-Memory Compression Scheme", ISCA 2005.
[79] A. Arelakis et al., "HyComp: a hybrid cache compression method for selection of data-type-specific compression methods", MICRO 2015.
[80] A. Arelakis and P. Stenstrom, "SC^2: A statistical compression cache scheme", ISCA 2014.
[81] G. Pekhimenko et al., "Exploiting Compressed Block Size as an Indicator of Future Reuse", HPCA 2015.
[82] O. Mutlu, "Intelligent Architectures for Intelligent Machines", Keynote Talk at 17th ChinaSys Workshop, December 2019. https://www.youtube.com/watch?v=n8Aj_A0WSg8
[83] M. Alser et al., "Shouji: A Fast and Efficient Pre-Alignment Filter for Sequence Alignment", Bioinformatics 2019.
[84] V. Seshadri et al., "RowClone: Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization", MICRO 2013.
[85] D. Lee et al., "Tiered-Latency DRAM: A Low Latency and Low Cost DRAM Architecture", HPCA 2013.
[86] Y. Kim et al., "A Case for Exploiting Subarray-Level Parallelism (SALP) in DRAM", ISCA 2012.
[87] B. C. Lee et al., "Phase Change Technology and the Future of Main Memory", IEEE Micro 2010.
[88] Y. Li et al., "Utility-Based Hybrid Memory Management", CLUSTER 2017.
[89] H. Yoon et al., "Row Buffer Locality Aware Caching Policies for Hybrid Memories", ICCD 2012.
[90] C. Wang et al., "Panthera: Holistic Memory Management for Big Data Processing over Hybrid Memories", PLDI 2019.
[91] J. Meza et al., "Enabling Efficient and Scalable Hybrid Memories Using Fine-Granularity DRAM Cache Management", IEEE CAL 2012.
[92] M. K. Qureshi et al., "Scalable high performance main memory system using phase-change memory technology", ISCA 2009.
[93] M. K. Qureshi et al., "Morphable memory system: a robust architecture for exploiting multi-level phase change memories", ISCA 2010.
[94] C-C. Chou et al., "CAMEO: A Two-Level Memory Organization with Capacity of Main Memory and Flexibility of Hardware-Managed Cache", MICRO 2014.
[95] V. Young et al., "Enabling Transparent Memory-Compression for Commodity Memory Systems", HPCA 2014.
[96] X. Yu et al., "Banshee: Bandwidth-Efficient DRAM Caching via Software/Hardware Cooperation", MICRO 2017.
[97] O. Mutlu, "Memory Scaling: A Systems Architecture Perspective", IMW 2013.
[98] O. Mutlu and L. Subramanian, "Research Problems and Opportunities in Memory Systems", SUPERFRI 2015.
[99] O. Mutlu and J. Kim, "RowHammer: A Retrospective", IEEE TCAD 2019.
[100] Y. Cai et al., "Error Characterization, Mitigation, and Recovery in Flash Memory Based Solid State Drives", Proc. IEEE 2017.
[101] Y. Cai et al., "Errors in Flash-Memory-Based Solid-State Drives: Analysis, Mitigation, and Recovery", Inside Solid-State Drives, 2018.
[102] O. Mutlu, "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser", DATE 2017.
[103] O. Mutlu et al., "Recent Advances in DRAM and Flash Memory Architectures", IPSI TIR, July 2018.
[104] Y. Kim et al., "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", ISCA 2014.
[105] J. S. Kim et al., "Revisiting RowHammer: An Experimental Analysis of Modern Devices and Mitigation Techniques", ISCA 2020.
[106] P. Frigo et al., "TRRespass: Exploiting the Many Sides of Target Row Refresh", S&P 2020.
[107] Y. Luo et al., "Characterizing Application Memory Error Vulnerability to Optimize Data Center Cost via Heterogeneous-Reliability Memory", DSN 2014.
[108] A. Boroumand et al., "CoNDA: Efficient Cache Coherence Support for Near-Data Accelerators", ISCA 2019.
[109] V. Seshadri et al., "Fast Bulk Bitwise AND and OR in DRAM", IEEE CAL 2015.
[110] A. Boroumand et al., "LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory", IEEE CAL 2016.
[111] M. Hashemi et al., "Continuous Runahead: Transparent Hardware Acceleration for Memory Intensive Workloads", MICRO 2016.
[112] A Pattnaik et al., " Scheduling Techniques for GPU Architectures with Processing-In-Memory Capabilities", PACT 2016.
[113] D. Senol Cali et al., "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis", MICRO 2020.
[114] H. Hassan et al., "SoftMC: A Flexible and Practical Open-Source Infrastructure for Enabling Experimental DRAM Studies", HPCA 2017.
[115] Y. Kim et al., "Ramulator: A Fast and Extensible DRAM Simulator", IEEE CAL 2015.
[116] J. Meza et al., "A Case for Efficient Hardware-Software Cooperative Management of Storage and Memory", WEED 2013.
[117] L. Subramanian et al., "MISE: Providing Performance Predictability and Improving Fairness in Shared Main Memory Systems", HPCA 2013.
[118] C. J. Lee et al., "Prefetch-Aware DRAM Controllers", MICRO 2008.
[119] M. Alser et al., "Accelerating Genome Analysis: A Primer on an Ongoing Journey", IEEE Micro, September/October 2020.
[120] O. Mutlu et al., "A Modern Primer on Processing in Memory", Invited Book Chapter in Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann, Springer, 2021. https://arxiv.org/abs/2012.03112
[121] L. N. Vintan and M. Iridon, "Towards a High Performance Neural Branch Predictor", IJCNN 1999.
[122] L. Subramanian et al., ""The Application Slowdown Model: Quantifying and Controlling the Impact of Inter-Application Interference at Shared Caches and Main Memory", MICRO 2015.
[123] S. P. Muralidhara et al., "Reducing Memory Interference in Multicore Systems via Application-Aware Memory Channel Partitioning", MICRO 2011.
[124] E. Ebrahimi et al., "Parallel Application Memory Scheduling", MICRO 2011.
[125] C. J. Lee et al., "Prefetch-Aware Memory Controllers", IEEE TC 2011.
[126] E. Ebrahimi et al., "Prefetch-Aware Shared Resource Management for Multi-Core Systems", ISCA 2011.
[127] H. David et al., "Memory Power Management via Dynamic Voltage/Frequency Scaling", ICAC 2011.
[128] I. Hur and C. Lin, "A Comprehensive Approach to DRAM Power Management", HPCA 2008.
[129] C. J. Lee et al., "DRAM-Aware Last-Level Cache Writeback: Reducing Write-Caused Interference in Memory Systems", HPS Technical Report 2010.
[130] E. Ebrahimi et al., ""Fairness via Source Throttling: A Configurable and High-Performance Fairness Substrate for Multi-Core Memory Systems", ASPLOS 2010.
[131] C. J. Lee et al., ""Improving Memory Bank-Level Parallelism in the Presence of Prefetching", MICRO 2009.
[132] Q. Deng et al., "MemScale: Active Low-Power Modes for Main Memory", ASPLOS 2011.
[133] B. Diniz et al., "Limiting the Power Consumption of Main Memory", ISCA 2007.
[134] V. Pandey et al., "DMA-aware Memory Energy Management", HPCA 2006.
[135] F. Zhang et al., "TADOC: Text Analytics Directly on Compression", VLDB Journal 2020.
[136] F. Zhang et al., "Enabling Efficient Random Access to Hierarchically-Compressed Data", ICDE 2020.
[137] F. Zhang et al., "Efficient Document Analytics on Compressed Data: Method, Challenges, Algorithms, Insights", VLDB 2018.
[138] F. Zhang et al., "Zwift: A Programming Framework for High Performance Text Analytics on Compressed Data", ICS 2018.
[139] O. Mutlu, "Intelligent Architectures for Intelligent Machines", Invited Talk at Texas State University Computer Science Seminar, Nov. 2019. https://www.youtube.com/watch?v=sJwO_BB4LaY
[140] O. Mutlu, "Memory-Centric Computing Sytems", Invited Tutorial at the 66th International Electron Devices Meeting (IEDM), Dec. 2019.
[141] O. Mutlu, "Intelligent Architectures for Intelligent Machines", Keynote Talk at ACM SYSTOR Conference, October 2020. https://www.youtube.com/watch?v=V6Sq7OiQD90

[142] O. Mutlu, “Intelligent Architectures for Intelligent Machines", Plenary Keynote Talk at VLSI-DAT/TSA, August 2020. https://www.youtube.com/watch?v=c6_LgzuNdkw
[143] M. Alser et al., “SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs”, Bioinformatics 2020.
[144] G. Singh et al., “NERO: A Near High-Bandwidth Memory Stencil Accelerator for Weather Prediction Modeling”, FPL 2020.
[145] I. Fernandez et al., “NATSA: A Near-Data Processing Accelerator for Time Series Analysis”, ICCD 2020.
[146] Y. Wang et al., “FIGARO: Improving System Performance via Fine-Grained In-DRAM Data Relocation and Caching”, MICRO 2020.
[147] S. H. S. Rezaei et al., “NoM: Network-on-Memory for Inter-Bank Data Transfer in Highly-Banked Memories”, IEEE CAL 2020.
[148] O. Mutlu, “Intelligent Architectures for Intelligent Machines”, VLSI-DAT 2020.
[149] S. Dustdar et al., “Rethinking Divide and Conquer - Towards Holistic Interfaces of the Computing Stack”, IEEE Internet Computing 2020.
[150] S. Srinath et al., “Feedback Directed Prefetching: Improving the Performance and Bandwidth-Efficiency of Hardware Prefetchers”, HPCA 2007.
[151] R. Bera et al., “DSPatch: Dual Spatial Pattern Prefetcher", MICRO 2019.
[152] E. Ebrahimi et al., “Techniques for Bandwidth-Efficient Prefetching of Linked Data Structures in Hybrid Prefetching Systems”, HPCA 2009.
[153] E. Ebrahimi et al., “Coordinated Control of Multiple Prefetchers in Multi-Core Systems”, MICRO 2009.
[154] O. Mutlu et al., “Runahead Execution: An Alternative to Very Large Instruction Windows for Out-of-order Processors”, HPCA 2003.
[155] O. Mutlu et al., “Techniques for Efficient Processing in Runahead Execution Engines”, ISCA 2005.
[156] K. J. Nesbit and J. E. Smith, “Data Cache Prefetching Using a Global History Buffer”, HPCA 2004.
[157] M. Shevgoor et al., “Efficiently Prefetching Complex Address Patterns”, MICRO 2015.
[158] J. Kim et al., “Path Confidence based Lookahead Prefetching”, MICRO 2016.
[159] M. K. Qureshi et al., “A Case for MLP-Aware Cache Replacement”, ISCA 2006.
[160] V. Seshadri et al., “The Evicted-Address Filter: A Unified Mechanism to Address Both Cache Pollution and Thrashing”, PACT 2012.
[161] M. K. Qureshi et al., “Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches”, MICRO 2006.
[162] M. K. Qureshi et al., “Adaptive Insertion Policies for High Performance Caching” ISCA 2007.
[163] K. Hsieh et al., “Focus: Querying Large Video Datasets with Low Latency and Low Cost”, OSDI 2018.
[164] R. L. Sites, “It’s the Memory, Stupid!”, MPR, 1996.
[165] S. Kanev et al., “Profiling a Warehouse-Scale Computer”, ISCA 2015.
[166] V. J. Reddi et al., “Web Search using Mobile Cores: Quantifying and Mitigating the Price of Efficiency”, ISCA 2010.
[167] J. Liu et al., “An Experimental Study of Data Retention Behavior in Modern DRAM Devices: Implications for Retention Time Profiling Mechanisms”, ISCA 2013.
[168] M. Patel et al., “The Reach Profiler (REAPER): Enabling the Mitigation of DRAM Retention Failures via Profiling at Aggressive Conditions”, ISCA 2017.
[169] S. Khan et al., “The Efficacy of Error Mitigation Techniques for DRAM Retention Failures: A Comparative Experimental Study”, SIGMETRICS 2014.
[170] U. Kang et al., “Co-architecting Controllers and DRAM to Enhance DRAM Process Scaling”, Memory Forum 2014.
[171] S. Ghose et al., “What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study”, SIGMETRICS 2018.
[172] E. Kultursay et al., “Evaluating STT-RAM as an Energy-Efficient Main Memory Alternative”, ISPASS 2013.
[173] B. C. Lee et al., “Phase Change Memory Architecture and the Quest for Scalability”, CACM 2010.
[174] H. Yoon et al., “Efficient Data Mapping and Buffering Techniques for Multi-Level Cell Phase-Change Memories”, ACM TACO 2015.
[175] F. Gao et al., “ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs”, MICRO 2019.
[176] S. Li et al., “Pinatubo: A Processing-in-Memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories”, DAC 2016.
[177] S. Kvatinsky et al., “MAGIC - Memristor-Aided Logic”, IEEE TCAS II 2014.
[178] S. Aga, “Compute Caches”, HPCA 2017.
[179] Y. Luo et al., “Improving 3D NAND Flash Memory Lifetime by Tolerating Early Retention Loss and Process Variation”, SIGMETRICS 2018.
[180] Y. Luo et al., “HeatWatch: Improving 3D NAND Flash Memory Device Reliability by Exploiting Self-Recovery and Temperature-Awareness”, HPCA 2018.
[181] Y. Cai et al., “Data Retention in MLC NAND Flash Memory: Characterization, Optimization and Recovery”, HPCA 2015.
[182] Y. Cai et al., “Flash Correct-and-Refresh: Retention-Aware Error Management for Increased Flash Memory Lifetime”, ICCD 2012.
[183] Y. Cai et al., “Error Patterns in MLC NAND Flash Memory: Measurement, Characterization, and Analysis”, DATE 2012.
[184] Y. Luo et al., “WARM: Improving NAND Flash Memory Lifetime with Write-hotness Aware Retention Management”, MSST 2015.
[185] Y. Cai et al., “Neighbor-Cell Assisted Error Correction for MLC NAND Flash Memories”, SIGMETRICS 2014.
[186] Y. Cai et al., “Program Interference in MLC NAND Flash Memory: Characterization, Modeling, and Mitigation”, ICCD 2013.
[187] Y. Cai et al., “Threshold Voltage Distribution in MLC NAND Flash Memory: Characterization, Analysis and Modeling”, DATE 2013.
[188] Y. Cai et al., “Vulnerabilities in MLC NAND Flash Memory Programming: Experimental Analysis, Exploits, and Mitigation Techniques”, HPCA 2017.
[189] Y. Cai et al., “Read Disturb Errors in MLC NAND Flash Memory: Characterization and Mitigation”, DSN 2015.
[190] Y. Luo et al., “Enabling Accurate and Practical Online Flash Channel Modeling for Modern MLC NAND Flash Memory”, IEEE JSAC 2016.
[191] M. Qureshi et al., “AVATAR: A Variable-Retention-Time (VRT) Aware Refresh for DRAM Systems”, DSN 2015.
[192] S. Khan et al., “PARBOR: An Efficient System-Level Technique to Detect Data-Dependent Failures in DRAM”, DSN 2016.
[193] S. Khan et l., “Detecting and Mitigating Data-Dependent DRAM Failures by Exploiting Current Memory Content”, MICRO 2017.
[194] H. Xin et al., “Shifted Hamming Distance: A Fast and Accurate SIMD-friendly Filter to Accelerate Alignment Verification in Read Mapping”, Bioinformatics 2015.
[195] M. Alser et al., “GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping", Bioinformatics 2017.
[196] H. Xin et al., “"Accelerating Read Mapping with FastHASH", BMC Genomics 2013.
[197] C. Alkan et al., “Personalized copy number and segmental duplication maps using next-generation sequencing”, Nature Genetics 2009.
[198] K. Hsieh et al., “Gaia: Geo-Distributed Machine Learning Approaching LAN Speeds”, NSDI 2017.
[199] K. Hsieh et al., “The Non-IID Data Quagmire of Decentralized Machine Learning", ICML 2020.
[200] T. Moscibroda and O. Mutlu, “A Case for Bufferless Routing in On-Chip Networks”, ISCA 2009.
[201] R. Das et al., “Application-Aware Prioritization Mechanisms for On-Chip Networks”, MICRO 2009.
[202] R. Das et al., “Aergia: Exploiting Packet Latency Slack in On-Chip Networks”, ISCA 2010.
[203] R. Das et al., “Application-to-Core Mapping Policies to Reduce Memory System Interference in Multi-Core Systems”, HPCA 2013.
[204] B. Grot et al., “Preemptive Virtual Clock: A Flexible, Efficient, and Cost-effective QOS Scheme for Networks-on-Chip”, MICRO 2009.
[205] C. Fallin et al., “CHIPPER: A Low-Complexity Bufferless Deflection Router”, HPCA 2011.
[206] O. Kayiran et al., “Managing GPU Concurrency in Heterogeneous Architectures”, MICRO 2014.
[207] C. Fallin et al., “MinBD: Minimally-Buffered Deflection Routing for Energy-Efficient Interconnect”, NOCS 2012.
[208] J. Zhao et al., “FIRM: Fair and High-Performance Memory Control for Persistent Memory Systems”, MICRO 2014.
[209] G. Nychis et al., “On-Chip Networks from a Networking Perspective: Congestion and Scalability in Many-core Interconnects”, SIGCOMM 2012.
[210] N. Vijaykumar et al., “A Case for Core-Assisted Bottleneck Acceleration in GPUs: Enabling Flexible Data Compression with Assist Warps”, ISCA 2015.
[211] J. A. Joao et al., “Utility-Based Acceleration of Multithreaded Applications on Asymmetric CMPs”, ISCA 2013.
[212] K. K. Rangan et al., “Thread Motion: Fine-grained Power Management for Multi-core Systems”, ISCA 2009.
[213] V. J. Reddi et al., “Voltage Emergency Prediction: Using Signatures to Reduce Operating Margins”, HPCA 2009.
[214] J. Haj-Yahya et al., “Techniques for Reducing the Connected-Standby Energy Consumption of Mobile Devices”, HPCA 2020.