Heterogeneous Data-Centric Architectures for Modern Data-Intensive Applications: Case Studies in Machine Learning and Databases

Geraldo F. Oliveira\textsuperscript{\circledast} Amirali Boroumand\textsuperscript{\dagger} Saugata Ghose\textsuperscript{\circledast} Juan Gómez-Luna\textsuperscript{\circledast} Onur Mutlu\textsuperscript{\circledast}  
\textsuperscript{\circledast}ETH Zürich \hspace{1cm} \textsuperscript{\dagger}Google \hspace{1cm} \textsuperscript{\circledast}University of Illinois Urbana-Champaign

1. Motivation & Problem

Today’s computing systems require moving data back-and-forth between computing resources (e.g., CPUs, GPUs, accelerators) and off-chip main memory so that computation can take place on the data. Unfortunately, this data movement is a major bottleneck for system performance and energy consumption [1, 2]. One promising execution paradigm that alleviates the data movement bottleneck in modern and emerging applications is processing-in-memory (PIM) [2–12], where the cost of data movement to/from main memory is reduced by placing computation capabilities close to memory. In the data-centric PIM paradigm, the logic close to memory has access to data with significantly higher memory bandwidth, lower latency, and lower energy consumption than processors/accelerators in existing processor-centric systems.

Naively employing PIM to accelerate data-intensive workloads can lead to sub-optimal performance due to the many design constraints PIM substrates impose (e.g., limited area and power budget available inside 3D-stacked memories [6] or manufacturing limitations of combining memory and logic elements [6, 13]). Therefore, many recent works co-design specialized PIM accelerators and algorithms to improve performance and reduce the energy consumption of (i) applications from various application domains, such as graph processing [14–40], machine learning [1, 41–77], bioinformatics [78–100], high-performance computing [95, 101–112], databases [18, 19, 29, 46, 60, 113–130], security [131–140], data manipulation [29, 115, 141–146], and mobile workloads [1, 46]; and (ii) various computing environments, including cloud systems [13, 15, 113, 147–150], mobile systems [1], and edge devices [151, 152].

We showcase the benefits of co-designing algorithms and hardware in a way that efficiently takes advantage of the PIM paradigm for two modern data-intensive applications: (1) machine learning inference models for edge devices and (2) hybrid transactional/analytical processing databases for cloud systems. We follow a two-step approach in our system design. In the first step, we extensively analyze the computation and memory access patterns of each application to gain insights into its hardware/software requirements and major sources of performance and energy bottlenecks in processor-centric systems. In the second step, we leverage the insights from the first step to co-design algorithms and hardware accelerators to enable high-performance and energy-efficient data-centric architectures for each application.

2. Mensa: Accelerating Google Neural Network Models for Edge Devices

Modern consumer devices make widespread use of machine learning (ML). The growing complexity of these devices, combined with increasing demand for privacy, connectivity, and real-time responses, has spurred significant interest in pushing ML inference computation to the edge (i.e., in or near consumer devices, instead of the cloud) [153–155]. Due to their resource-constrained nature, edge computing platforms now employ specialized energy-efficient accelerators for on-device inference (e.g., Google Edge Tensor Processing Unit, TPU [156]; NVIDIA Jetson [157]; Intel Movidius [158]). At the same time, neural network (NN) algorithms are evolving rapidly, which has led to many types of NN models.

Despite the wide variety of NN model types, Google’s state-of-the-art Edge TPU [156] provides an optimized one-size-fits-all design (i.e., a monolithic accelerator with a fixed, large number of processing elements (PEs) and a fixed dataflow, which determines how data moves within the accelerator) that caters to edge device area and energy constraints. Unfortunately, it is very challenging to achieve high energy efficiency, computational throughput, and area efficiency for each NN model with this one-size-fits-all design.

We conduct an in-depth analysis of ML inference execution on a commercial Edge TPU, across 24 state-of-the-art Google edge NN models spanning four popular NN model types: (1) convolutional neural networks (CNNs), (2) long short-term memories (LSTMs) [159], (3) Transducers [160–162], and (4) recurrent CNNs (RCNNs) [163, 164]. The key takeaway from our extensive analysis of Google edge NN models on the Edge TPU is that all key components of an edge accelerator (i.e., PE array, dataflow, memory system) must be co-designed and co-customized based on specific layer characteristics to achieve high utilization and energy efficiency. Therefore, our goal is to revisit the design of edge ML accelerators such that they are aware of and can fully exploit the growing variation within and across edge NN models.

To this end, we propose Mensa, the first general HW/SW composable framework for ML acceleration in edge devices. The key idea of Mensa is to perform NN layer execution across multiple on-chip and near-data accelerators, each of which is small and tailored to the characteristics of a particular subset (i.e., family) of layers. Our experimental study of the characteristics of different layers in the Google edge NN models reveals that the layers naturally group into a small number of clusters that are based on a subset of these characteristics. This new insight allows us to significantly limit the number of
different accelerators required in a Mensa design. We design
a runtime scheduler for Mensa to determine which of these
accelerators should execute which NN layer, using informa-
tion about (1) which accelerator is best suited to the layer’s
characteristics, and (2) inter-layer dependencies.

Using our new insight about layer clustering, we develop
Mensa-G, an example design for Mensa optimized for Google
edge NN models. We find that the design of Mensa-G’s un-
derlying accelerators should center around two key layer char-
acteristics (memory boundedness, and activation/parameter
reuse opportunities). This allows us to provide efficient infer-
ence execution for all of the Google edge NN models using
only three accelerators in Mensa-G that we call Pascal, Pavlov,
and Jacquard. Pascal, for compute-centric layers, maintains
the high PE utilization that these layers achieve in the Edge
TPU, but does so using an optimized dataflow that both re-
duces the size of the on-chip buffer (16 × smaller than in the
Edge TPU) and the amount of on-chip network traffic. Pavlov,
for LSTM-like data-centric layers, employs a dataflow that
enables the temporal reduction of output activations, and en-
ables the parallel execution of layer operations in a way that
increases parameter reuse, greatly reducing off-chip memory
traffic. Jacquard, for other data-centric layers, significantly
reduces the size of the on-chip parameter buffer (by 32 ×) us-
ing a dataflow that exposes reuse opportunities for parameters.
As both Pavlov and Jacquard are optimized for data-centric
layers, which require significant memory bandwidth and are
unable to utilize a significant fraction of PEs in the Edge TPU,
we place the accelerators in the logic layer of 3D-stacked
memories [165–167] and use significantly smaller PE arrays
compared to the PE array in Pascal, unleashing significant
performance and energy benefits.

Our evaluation shows that compared to the baseline Edge
TPU, Mensa-G reduces total inference energy by 66.0%, im-
proves energy efficiency (TFLOP/J) by 3.0 ×, and increases
computational throughput (TFLOP/s) by 3.1 ×, averaged
across all 24 Google edge NN models. Mensa-G improves
inference energy efficiency and throughput by 2.4 × and 4.3 ×
over Eyeriss v2, a state-of-the-art accelerator.

We conclude that employing our Mensa framework and tai-
loring ML accelerators to the key characteristics of NN layers
can provide performance, energy, and area benefits to edge
devices. For more information on Mensa, a detailed descrip-
tion of Mensa-G’s accelerators, and our extensive evaluation
results, please refer to the full version of our paper [45, 168].

3. Polynesia: Accelerating Hybrid Transac-
tional/Analytical Processing using PIM

Many application domains, such as fraud detection [169–171],
business intelligence [172–174], healthcare [175, 176], per-
sonalized recommendation [177, 178], and IoT [177], have a
critical need to perform *real-time data analysis*, where data
analysis needs to be performed using the most recent version
of data [179, 180]. To enable real-time data analysis, state-
of-the-art database management systems (DBMSs) leverage
*hybrid transactional and analytical processing* (HTAP) [181–
183]. An HTAP DBMS is a single-DBMS solution that sup-
ports both transactional and analytical workloads [179, 181,
184–186].

Ideally, an HTAP system should have three properties [185]
to guarantee efficient execution of transactional and analytical
workloads. First, it should ensure that both transactional and
analytical workloads benefit from their own workload-specific
optimizations (e.g., algorithms, data structures). Second, it
should guarantee data freshness and data consistency (i.e.,
access to the most recent version of data) for analytical work-
loads while ensuring that both transactional and analytical
workloads have a consistent view of data across the system.
Third, it should ensure that the latency and throughput of both
the transactional workload and the analytical workload are the
same as if each of them were run in isolation.

We extensively study state-of-the-art HTAP systems and
observe two key problems that prevent them from achieving all
three properties of an ideal HTAP system. First, these systems
experience a drastic reduction in transactional throughput (up
to 74.6%) and analytical throughput (up to 49.8%) compared
to when transactional and analytical workloads run in isolation.
This is because the mechanisms used to provide data freshness
and consistency induce a large amount of data movement
between the CPU cores and main memory. Second, HTAP
systems often fail to provide effective performance isolation.
These systems suffer from severe performance interference
because of the high resource contention between transactional
workloads and analytical workloads. Therefore, our goal is
to develop an HTAP system that overcomes these problems
while achieving all three of the desired HTAP properties.

We propose a novel system for HTAP databases called *Pol-
ynesia*. The key insight behind Polynesia is to partition the
computing resources into two isolated new custom process-
ing islands: *transactional islands* and *analytical islands*. An
island is a hardware–software co-designed component spe-
cialized for specific types of queries. Each island consists of
(1) a replica of data for a specific workload, (2) an optimized
execution engine, and (3) a set of hardware resources that cater
to the execution engine and its memory access patterns.

Figure 1 shows the high-level organization of Polynesia,
which includes one transactional island and one analytical
island. Polynesia is equipped with a 3D-stacked memory
similar to the Hybrid Memory Cube (HMC) [166], where
multiple vertically-stacked DRAM layers are connected with
a *logic layer* using thousands of *through-silicon vias* (TSVs).
An HMC chip is split up into multiple *vaults*, where each vault
corresponds to a vertical slice of the memory and logic layer.
Polynesia meets all desired properties from an HTAP system in three ways. First, by employing processing islands, Polynesia enables workload-specific optimizations for both transactional and analytical workloads. Second, we design new hardware accelerators to add specialized capabilities to each island, which we exploit to optimize the performance of several key HTAP algorithms. This includes new accelerators and modified algorithms to propagate transactional updates to analytical islands and to maintain a consistent view of data across the system. Such new components ensure data freshness and data consistency in our HTAP system. Third, we tailor the design of transactional and analytical islands to fit the characteristics of transactional and analytical workloads. The transactional islands use dedicated CPU hardware resources (i.e., multicore CPUs and multi-level caches) to execute transactional workloads since transactional queries have cache-friendly access patterns [18, 19, 46]. The analytical islands leverage PIM techniques [6, 7, 187] due to the large data traffic analytical workloads produce. We equip the analytical islands with a new PIM-based analytical engine that includes simple in-order PIM cores added to the logic layer of a 3D-stacked memory [48, 166, 167], software to handle data placement, and runtime task scheduling heuristics. Our new design enables the execution of transactional and analytical workloads at low latency and high throughput.

In our evaluations, we show the benefits of each component of Polynesia, and compare its end-to-end performance and energy usage to three state-of-the-art HTAP systems (modeled after Hyper [188], AnkerDB [189], and Batch-DB [185]). Polynesia outperforms all three, with higher transactional throughput (2.20×1.15×1/1.94×; mean of 1.70×) and analytical throughput (3.78×/5.04×2.76×; mean of 3.74×), while consuming 48% lower energy than the prior lowest-energy HTAP system.

We conclude that Polynesia efficiently provides high-throughput real-time data analysis, while meeting all three desired HTAP properties. For more information on Polynesia design, including our tailored algorithms and hardware units to maintain data freshness and data consistency, and the design of our analytical engine, as well as our extensive evaluation results, please refer to the full version of our paper [117, 190]. We open-source Polynesia and the complete source code of our evaluation [191].

4. Conclusion & Future Work

This paper summarizes our recent works on accelerating emerging data-intensive applications with processing-in-memory (PIM). We showcase the performance and energy benefits that PIM provides for edge neural network models and hybrid transactional/analytical processing systems. Our proposed PIM architectures outperform state-of-the-art solutions by co-designing algorithms and hardware while reducing area costs. We hope our work inspires the design of novel, high-performance, and energy-efficient data-centric PIM architectures for many other emerging applications.

Acknowledgments

We thank SAFARI Research Group members for valuable feedback and the stimulating intellectual environment they provide. We acknowledge the generous gifts provided by our industrial partners, including ASML, Facebook, Google, Huawei, Intel, Microsoft, and VMware. We acknowledge support from the Semiconductor Research Corporation and the ETH Future Computing Laboratory.

This invited extended abstract is a summary version of our prior works, Mensa [168, 192] (published at PACT 2021) and Polynesia [117, 190] (published at ICDE 2022). Presentations describing Mensa and Polynesia can be found at [193] and [194], respectively.

References

[1] A. Boroumand et al., “Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks,” in ASPLOS, 2018.
[2] O. Mutlu, “Memory Scaling: A Systems Architecture Perspective,” in IMW, 2013.
[3] G. F. Oliveira et al., “DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks,” arXiv:2105.03725 [cs.AR], 2021.
[4] G. F. Oliveira et al., “DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks,” IEEE Access, 2021.
[5] S. Ghose et al., “The Processing-in-Memory Paradigm: Mechanisms to Enable Adoption,” in Beyond-CMOS Technologies for Next Generation Computer Design, 2019.
[6] O. Mutlu et al., “A Modern Primer on Processing in Memory,” arXiv:2012.03112 [cs.AR], 2021.
[7] S. Ghose et al., “Processing-in-Memory: A Workload-Driven Perspective,” IBM JRD, 2019.
[8] O. Mutlu et al., “Enabling Practical Processing in and near Memory for Data-Intensive Computing,” in DAC, 2019.
[9] O. Mutlu and L. Subramanian, “Research Problems and Opportunities in Memory Systems,” SUPERFRI, 2014.
[10] O. Mutlu, “Intelligent Architectures for Intelligent Computing Systems,” in DATE, 2021.
[11] O. Mutlu, “Intelligent Architectures for Intelligent Machines,” in VLSIDAT, 2020.
[12] O. Mutlu, “Main Memory Scaling: Challenges and Solution Directions,” in More Than Moore Technologies for Next Generation Computer Design. Springer-Verlag, 2015.
[13] F. Devaux, “The True Processing in Memory Accelerator,” in HCS, 2019.
[14] J. Ahn et al., “PIM-Enabled Instructions: A Low-Overhead, Locality-Aware Processing-in-Memory Architecture,” in ISCA, 2015.
[15] J. Ahn et al., “A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing,” in ISCA, 2015.
[16] L. Nai et al., “GraphPIM: Enabling Instruction-Level PIM Offloading in Graph Computing Frameworks,” in HPCA, 2017.
[17] L. Song et al., “GraphR: Accelerating Graph Processing Using ReRAM,” in HPCA, 2018.
[18] A. Boroumand et al., “LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory,” IEEE CAL, 2017.
