Data-Centric Architecture and Algorithm Co-design for Modern Data-Intensive Applications

Dr. Haiyu Mao
Postdoctoral Research Associate
Explosive Growth in Data

Global data created, consumed and stored (zettabytes)\[1\]

Modern Data-Intensive Applications

- Modern applications are data-driven
  - Gain insights from big data
  - Make decisions based on the insights gained

- Gain information from big text data e.g., 45TB for GPT-3
- Understand and generate language

- Gain information from massive reference genomes e.g., more than 2.5PB for The Cancer Genome Atlas (TCGA)
- Analyze individual genome to check cancer existence

Large Language Model (LLM)  Genome Analysis
Data Processing Bottleneck

- Generating data is **cheaper** than processing data

- Generating data is **faster** than processing data

48 Human whole genomes at 30× coverage in about 2 days

1 Human genome 32 CPU hours on a 48-core processor
Data Movement and Access Problem

- Traditional computer architecture separates **computation units** and **memory units**

  ![Diagram showing data movement between CPU, Main Memory, and Storage]

**Problem 1**

A large amount of data movement (TB – PB or more) between computation units and memory units.

**Problem 2**

Limited data access parallelism due to the limited number of pins (e.g., 228 on DDR4-DRAM).
Data Movement and Access Problem

- Data movement and access of traditional computer architecture when executing data-intensive applications: **Performance Bottleneck**

  ![Graph showing cache-bound cycles](image)

  **Half of cycles** are spent stalled on caches: waiting for the data from memory

Data Movement and Access Problem

Data movement and access of traditional computer architecture when executing data-intensive applications: **Energy Bottleneck**

77% of the total energy consumption are spent on data movement [1]

---

Problem Conclusion

Data movement and access problem causes significant performance bottleneck and energy consumption waste.

Unacceptable slow speed  Overheating and safety  Environment problems
Solution: Data-Centric Architecture

Traditional Computer Architecture:
Process data in the centric processor

Data-Centric Computer Architecture:
Process data where it resides
Main Data-Centric Architectures

- **Processing in Memory (PIM)**
- **In-Storage Processing (ISP)**
  - Processing near Memory (PNM)
  - Processing using Memory (PUM)
  - Processing near Flash (PNF)
  - Processing using Flash (PUF)

**SAFARI**
An Example of Processing using Memory

PUM for Vector-Matric Multiplication

PUM: Processing 1,000,000 calculations parallelly in 1000 ns.

Traditional Computer Architecture:
1. Read all data to CPU (e.g., 1 s)
2. Perform calculations (e.g., 2 s)
3. Write the data back to memory

PUM can be million times faster than traditional computer architecture.
Challenges of Data-Centric Architecture

- Limited flexibility

- Where to execute other operations?
  - CPU or other PUs

- How do PUM and other units cooperate?
  - New bottlenecks

→ End-to-end support for applications is challenging
Research Goal

Design comprehensive data-centric architectures for end-to-end data-intensive applications aimed at enhancing safety and quality of life.
Software and Hardware Co-design across the computer design hierarchy to achieve the design goals
Software and Hardware Co-Design

Computer Design Hierarchy

**Software**
- User Requirement
- Algorithm
- Program/Language
- System Software

**Hardware**
- SW/HW Interface
- Micro-architecture
- Logic
- Device

**Black box**

Traditional computer architecture refers to software/hardware interface and micro-architecture.

**Black box**
Software and Hardware Co-Design

Computer Design Hierarchy

- User Requirement
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture
- Logic
- Device

New Challenges & Opportunities

Data Intensive

Challenges

Emerging Technologies

Affect

Opportunities
Computer architecture in an expanded view:

- Co-design across the hierarchy
  -- From algorithms to devices
- Specialized within design goals
  -- Optimize computers in a holistic way
Research Work Overview

Computer Design Hierarchy

Software
- User Requirement
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture

Hardware
- Logic
- Device

Data-Centric Architecture
- Bioinformatics
- Non-Volatile Memory
- Machine Learning

Broaden

Non-Volatile Memory Machine Learning

Ph.D. First Author
Ph.D. Co-author
Postdoc First Author
Postdoc Co-author
Current Research

Computer Design Hierarchy

- User Requirement
- Algorithm
- Program/Language
- System Software
- SW/HW Interface
- Micro-architecture
- Logic
- Device

Computer architecture in an expanded view

Algorithm
And
Data-Centric
Architecture
Co-design
For
Genome Analysis

SAFARI
Talk Outline

- Research Background, Motivation, Goal, and Direction

- Data-Centric Architecture and Algorithm Co-design
  - For nanopore genome analysis

- Future Research Plan
Importance of Genome Analysis [1]

- Fast and accurate genome analysis is important for:
  - Personalized medicine
  - Surveillance of disease outbreaks
  - Predicting the presence of pathogens in an environment
  - Understanding genetic variations, species, and evolution

Gain insights by analyzing data generated from genome sequencing machine.

- DNA Sample
- Sequencing Machine
- Randomized Subsequences
- Analysis

Can only sequence small randomized fragments of the original DNA sequence.

Genome analysis
Reveal genome variants.
Nanopore Genome Sequencing

- Nanopore Genome Sequencing
  A widely-used sequencing technology
  - Portable sequencing devices
  - High-throughput
  - Cheap
  - ...

Nanopore sequencing device
Nanopore Genome Analysis Pipeline

1. Basecalling
   - Compute: AT
   - Storage: GCGTTC
   - Storage: TC AAAG

2. Read Quality Control
   - Compute: ATGGAC
   - Storage: GCGTTC

3. Read Mapping
   - Compute: TATGGAC TTTAGCAAAAC
   - Mapped
   - Unmapped

Genome sequencing

- Store mapping results
Limitation 1: Large Data Movement

- Using a human dataset in [NC’19] as an example:

Raw Signals → Basecalling → Reads → Read quality control → High-quality reads → Read mapping → Mapped reads

- Large data movement between genome analysis steps

3913 GB → 546 GB → 437 GB → 382 GB

Limitation 2: Wasted Computation

- Using a human dataset in [NC’19] as an example:

<table>
<thead>
<tr>
<th>Raw Signals</th>
<th>Basecalling</th>
<th>Reads</th>
<th>Read quality control</th>
<th>High-quality reads</th>
<th>Read mapping</th>
<th>Mapped reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td></td>
<td></td>
<td></td>
<td>79.5%</td>
<td></td>
<td>69.5%</td>
</tr>
</tbody>
</table>

- A considerable amount of computation on **useless data** due to:
  - Low-quality reads
  - Unmapped reads

Goal: Efficiently accelerate the entire genome analysis pipeline while minimizing data movement and useless computation

- We perform a study to quantify potential performance benefits
  - Results are normalized to the performance of GPU

<table>
<thead>
<tr>
<th>Normalized Speedup</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.7x</td>
<td>NVM-based PIM accelerators for separate basecalling and read mapping</td>
</tr>
<tr>
<td>6.1x</td>
<td>no data movement between the accelerators of analysis steps</td>
</tr>
<tr>
<td>9x</td>
<td>no data movement and no useless reads (ideal case)</td>
</tr>
</tbody>
</table>
GenPIP

- *First holistic in-memory accelerator for the genome analysis pipeline*, including basecalling, read quality control, and read mapping steps

- **GenPIP** has two key techniques
  - **Chunk-based Pipeline (CP)**
    - Enables **fine-grained pipelining** of genome analysis steps
    - Processes reads at **chunk** granularity (i.e., a subsequence; 300 bases)
  - **Early Rejection (ER)**
A read consists of four chunks: C1, C2, C3, C4

- **CP increases parallelism** by overlapping the execution of different steps at chuck granularity
- **CP reduces intermediate data** by computing on data as soon as data is generated
- **CP provides opportunities for ER** by analyzing a read at chunk granularity
GenPIP

- First holistic in-memory accelerator for the genome analysis pipeline, including basecalling, read quality control, and read mapping steps

- GenPIP has two key techniques
  - Chunk-based Pipeline (CP)
    - Enables fine-grained collaboration of genome analysis steps by processing reads at chunk granularity (i.e., a subsequence of a read, e.g., 300 bases)
  - Early Rejection (ER)
    - Stops the execution on useless reads as early as possible by using a small number of chunks to predict the usefulness of a read
Early Rejection (ER)

- Predict and eliminate low-quality and unmapped reads from the genome analysis pipeline as early as possible

- Early-Rejection based on chunk quality scores (ER-QSR)
  - Predict low-quality reads using chunk quality scores

- Early-Rejection based on chunk mapping scores (ER-CMR)
  - Predict unmapped reads using chunk mapping scores
Implementation of CP and ER

CP and ER can be applied on different systems, e.g., CPU, GPU, and PIM

We implement CP and ER using PIM since PIM is more efficient to reduce the data movement between genome analysis steps

We also apply CP and ER on CPU and GPU baselines and observe speedup and energy savings
GenPIP Implementation

Raw signals from the sequencing machine

- In-memory Basecaller [Helix, PACT’20]
- In-memory Read Mapping [PARC, ASPDAC’20] + Our design

**Basecalling Module**
- GenPIP Controller
  - eDRAM
  - Average Calculator
  - ER Controller

**In-memory Read Mapping Module**
- Read Mapping Controller
  - Read mapping result

**PIM-CQS**
- PIM chunk quality score calculation

**Basequality score**
- Chunk quality score
- Chunk mapping score
- ER

**To storage**
GenPIP Implementation

Raw signals from the sequencing machine

- PIM enables fast data processing
  - Analyze data as soon as it is generated by previous step
  - Reduce the storage of intermediate data

- Tightly integrating the genome analysis steps
  - Reduces data movement
  - Eliminates useless computation

Basecalling Module → ER → GenPIP Controller → ER → Read Mapping Module → To storage
Evaluation Methodology

- **Performance, Area and Power Analysis:**
  - Simulation via Verilog HDL, NVSim [TCAD’12], and CACTI 6.5 [MICRO’07]
  - See methodology in the paper for more

- **Baselines:**
  - **CPU** (Intel Xeon Gold 5118 CPU)
  - **GPU** (NVIDIA GeForce RTX 2080 Ti GPU)
  - Optimistic integration of two PIM accelerators (Helix [PACT’20] and PARC [ASP-DAC’20])
    - Assumes **no data movement** between steps
    - Assumes intermediate data causes no overhead

- **Datasets:**
  - **Human** ([https://www.ebi.ac.uk/ena/browser/view/PRJEB30620](https://www.ebi.ac.uk/ena/browser/view/PRJEB30620))
Key Results – Performance

GenPIP provides 41.6x, 8.4x, and 1.4x speedup over CPU, GPU, and optimistic PIM.

Both CP and ER are critical to the speedup.
Key Results – Energy Efficiency

GenPIP provides 32.8x, 20.8x, and 1.37x energy savings over CPU, GPU, and optimistic PIM.

ER is especially critical to the energy efficiency.
Summary of GenPIP

- **Problem:** The genome analysis pipeline has **large data movement** between genome analysis steps and a significant amount of **wasted computation on useless data**.

- **Goal:** Tightly integrate genome analysis steps to reduce the data movement between steps and eliminate computation on useless data.

- **GenPIP:** The *first* in-memory genome analysis accelerator that **tightly integrates** genome analysis steps.
  - **GenPIP** has two key techniques:
    - A chunk-based pipeline
    - A new early-rejection technique

- **GenPIP outperforms** state-of-the-art software & hardware solutions using **CPU**, **GPU**, and **optimistic PIM** by **41.6x**, **8.4x**, and **1.4x**, respectively.
Communicate to Industry

GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping

The basecalling and the read mapping are the two most time-consuming steps in the genome analysis pipeline. The read quality control is a highly-recommended but optional step to...

Poster | 14 Jul 2023 | POSTERS BIOINFORMATICS DATA ANALYSIS

Adoption

I am actively communicating to the industry and have attracted industry people’s interests.
Talk Outline

- Research Background, Motivation, Goal, and Direction
- Data-Centric Architecture and Algorithm Co-design
  - For nanopore genome analysis
- Future Research Plan
Summary of My Research

- **Background and motivation:** data-intensive applications
  - Genome analysis *(nanopore genome analysis)*
  - Machine learning
  - ...

- **Promising solution:** data-centric architecture
  - Processing in memory/storage
  - Quantum computing
  - ...

- **Challenges of adopting data-centric architecture**
  - Flexibility, end-to-end support for applications, reliability

- **Research direction and goal**
  - Software and hardware co-design *(algorithm-architecture)*
  - Maximize benefits within design goals
On-going Work: Raw-signal Analysis

Nanopore Raw Signal

Deep Neural Network

Reference Genome

...GCTATATGTG...

Map

TATATG

Bases

A larger data volume of raw signals
Processing a larger data
Moving and storing a larger data

Genome analysis on bases

Genome analysis on raw signals

Raw-Signal analysis can avoid expensive basecalling based on DNNs \[1\]

Near-term Research Directions (1-3 years)

- **Reliable** processing using memory/flash systems
  - ECC for computation
  - Data mapping to tolerate errors in memory cells

- **Flexible** processing in memory system
  - Support several important algorithms in a specific domain

- **Heterogeneous** processing in memory and storage system
  - Minimize data movement and maximize data access parallelism

- **Data-centric architecture and modern application co-design for portable devices**
  - Processing on compressed data
  - Minimize the intermediate data storage
Near-term Research Directions (1-6 years)

- **Variant-calling** acceleration to enable fast identification of variations (e.g., identify diseases)
  - Identify the bottleneck of variant-calling in state-of-the-art systems

- **Quantum computing** as a promising data-centric architecture
  - Quantum computing for data-intensive applications
    - Genome analysis
    - Machine learning
    - ...

- **Cancer-related** genome analysis acceleration
  - Large data-centric architectures for immense data
Long-term Research Direction

Algorithms are developing and applied in applications fast:

- **Algo_1**: Designed in January 2020, Used in February 2020

Architecture for Algo_1 designed in January 2021

- **Algo_N**: Used in January 2023

Architecture for Algo_1 used in January 2023

Architectures are developing and applied (design-tape-out) slowly:

Domain-specific architectures are out-of-date when they are used.
Long-term Research Direction

- Data-centric architectures for future applications

- Analyze modern applications to predict future applications
- Design data-centric architectures for future applications

<table>
<thead>
<tr>
<th>Algo_N Used</th>
<th>Jan. 2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predict Algo_2N*</td>
<td>Jan. 2023</td>
</tr>
<tr>
<td>Architecture for Algo_2N* Designed</td>
<td>Jan. 2024</td>
</tr>
<tr>
<td>Algo_2N Used</td>
<td>Jan. 2026</td>
</tr>
<tr>
<td>Architecture for Algo_2N* Used</td>
<td>Jan. 2026</td>
</tr>
</tbody>
</table>

Time
Thanks! Q&A

- **Postdoctoral Researcher** at ETH Zurich, since **September 2020**
  - With Prof. Onur Mutlu (SAFARI Research Group)
  - Mentor and Guest lecturer

- **Group Associate** in the ETH Future Computing Laboratory, since **May 2021**
  - Handle the cooperation between ETH groups and industry
  - Apply funding as a PI (current grant: 80,000 CHF)

- **Ph.D.** at Tsinghua University, **August 2015 - July 2020**
  - With Prof. Jiwu Shu (Storage Group)
  - Outstanding Ph.D. graduate in Beijing

- **Research Interests (Overview)**
  - **Computer Architecture, Memory Systems, Bioinformatics, AI/ML**
  - Modern data-intensive applications and algorithms
  - Data-centric architectures (processing in/using memory/storage)
  - Software and hardware co-design
  - Non-volatile memory and storage (architecture, system, security)