P&S Genomics
Lecture 10a: GenPIP

Dr. Haiyu Mao

ETH Zurich
Spring 2024
6 May 2024
GenPIP

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

Haiyu Mao, Mohammed Alser, Mohammad Sadrosadati, Can Firtina, Akanksha Baranwal, Damla Senol Cali, Aditya Manglik, Nour Almadhoun Alserr, Onur Mutlu

MICRO 2022
Overview: Genome Analysis

- **Genome analysis**: Enables us to determine the order of the DNA sequence in an organism’s genome
  - Plays an important role in
    - Personalized medicine
    - Outbreak tracing
    - Understanding of evolution
    - ...

- Modern genome sequencing machines extract smaller randomized fragments of the original DNA sequence, known as *reads*
  - **Oxford Nanopore Technologies (ONT)**: A widely-used sequencing technology
    - Portable sequencing devices
    - High-throughput
    - Cheap

[ONT sequencing device](forbes.com)
Overview: Two Limitations

- Multiple steps in genome analysis
- Large data movement between multiple steps
- A lot of wasted computation done on data that is later discovered to be useless
Overview: GenPIP

GenPIP: A fast and energy-efficient in-memory acceleration system for the Genome analysis Pipeline via tight integration of genome analysis steps.

GenPIP has two key techniques:

- Chunk-based pipeline (CP)
  - Provides fine-grained collaboration of genome analysis steps

- Early rejection (ER)
  - Timely stops the execution on useless data by predicting which reads will not be useful

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

- Background and Motivation
  - GenPIP: Tight Integration of Genome Analysis Steps
    - Chunk-based Pipeline (CP)
    - Early Rejection (ER)
  - GenPIP Implementation
- Evaluation
- Conclusion
Genome Analysis Pipeline

1. Basecalling

2. Read Quality Control

3. Read Mapping

- Mapped
- Unmapped

Store mapping results

Low-quality
Basecalling

- Use deep neural networks to ensure the basecalling accuracy

- **Input:** Raw signal **chunks**
- **Process:** Translate raw signals to bases (i.e., A, C, G, T) and calculate each base quality
- **Output:** Assemble chunks into a long read
Read Quality Control

- **Input**: Base quality scores of a read from the basecalling step

- **Process**:  
  - Calculate the average of all base quality scores in a read as the read quality score  
  - Compare the read quality score to the threshold to decide whether this read is low-quality or high-quality

- **Output**: High-quality reads (discard low-quality reads)
Input: High-quality read passes the read quality control step

Process:
- Use subsequence in a read to query the hash table to get possible match locations
- Identify the candidate regions and output the chaining score
- Execute the alignment step if there is a chain

Output: Mapping information
Limitation 1: Large Data Movement

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

![Diagram showing data movement between genome analysis steps]

- Raw Signals: 3913 GB
- Basecalling: 546 GB
- Reads: 437 GB
- Read quality control
- High-quality reads
- Read mapping
- Mapped reads: 382 GB

Large data movement between genome analysis steps

Limitation 2: Wasted Computation

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

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

---

State-of-the-art Works

- NVM-based PIM is an efficient technique to reduce data movement by processing data using or near memory

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

- NVM-based PIM for vector-matrix multiplication operation [Helix, PACT’20]
- Vector-matrix multiplication is the dominant operation in the neural network applications
State-of-the-art Works

(a) Multiply-Accumulate operation

\[ I = I_1 + I_2 = V_1.G_1 + V_2.G_2 \]

(b) Vector-Matrix Multiplier

[Shafiee et al., "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars", ISCA 2016.]
State-of-the-art Works

\[
\begin{pmatrix}
  i_1 \\
  i_2 \\
  i_3 \\
  i_4
\end{pmatrix}
= \begin{pmatrix}
  O_1 \\
  O_2 \\
  O_3 \\
  O_4
\end{pmatrix}
\]
State-of-the-art Works

[Shafiee+, “ISACC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars”, ISCA 2016.]
State-of-the-art Works

- NVM-based PIM is an efficient technique to reduce data movement by processing data using or near memory

<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>
</table>

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>$B_1$</th>
<th>$B_0$</th>
<th>$A_1$</th>
<th>$A_0$</th>
<th>$B_1$</th>
<th>$B_0$</th>
<th>$C$</th>
<th>$S_1$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(a) Operands Layout

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>$B_1$</th>
<th>$B_0$</th>
<th>$C$</th>
<th>$S_1$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(b) Initialization

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_0$</th>
<th>$B_1$</th>
<th>$B_0$</th>
<th>$C$</th>
<th>$S_1$</th>
<th>$S_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

(c) Bit-0 completed

- NVM-based PIM for **search** and **addition** [PARC, ASPDAC’20]

- Search and addition are the dominant operations in the read mapping step
State-of-the-art Works

- NVM-based PIM is an efficient technique to reduce data movement by processing data using or near memory

- Reduce the data movement in a single genome analysis step
- Exacerbate the data movement overhead between analysis steps

No prior work tackles data movement between analysis steps and reduces useless computation
Goal and Opportunities

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

SAFARI
Outline

- Background and Motivation

- GenPIP: Tight Integration of Genome Analysis Steps
  - Chunk-based Pipeline (CP)
  - Early Rejection (ER)

- GenPIP Implementation

- Evaluation

- Conclusion
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)
Chunk-based Pipeline (CP)

- CP **increases parallelism** by overlapping the execution of different steps at chunk 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.

A read consists of four chunks: **C1, C2, C3, C4**

[Diagram showing conventional and chunk-based pipelines with annotations for basecalling, QC, RM, and assembly.]
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**

---

**Flowchart**

1. **Basecall a small number of chunks**
   - **Check the average quality of these chunks**
   - **Pass**
   - **Fail**
   - **Stop analysis**

2. **Basecall more chunks**
   - **Map basecalled chunks so far**
   - **Check the mapping score**
   - **Fail**
   - **Pass**
   - **Basecall the remaining chunks**

3. **Execute the remaining computation in read mapping**

---

- **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
**ER based on Chunk Quality Scores**

- **Goal:** Accurately estimate the quality of the entire read by **checking the quality of a small number of sampled chunks**

Sample a small number of *non-consecutive* chunks evenly in a read to predict the read quality.
ER based on Chunk Mapping

Key insight of ER based on chunk mapping: A read probably cannot be mapped to the reference genome if enough consecutive chunks in this read cannot be mapped to the reference genome.

Mapping a small chunk provides too many possible mapping locations.

1. Sample a small number of consecutive chunks in a read.
2. Merge these small consecutive chunks into a big chunk.
3. Map this big chunk to the reference genome to predict whether the read can be mapped or not.
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
Outline

- Background and Motivation
- GenPIP: Tight Integration of Genome Analysis Steps
  - Chunk-based Pipeline (CP)
  - Early Rejection (ER)
- GenPIP Implementation
- Evaluation
- Conclusion
GenPIP Implementation

Raw signals from the sequencing machine

In-memory Basecaller
[Helix, PACT’20]

PIM-CQS
PIM chunk quality score calculation

Base calling Module

Base quality score

Signal chunk

Basecalled chunk

Chunk quality score

Average Calculator

ER Controller

GenPIP Controller

Quality score

Chunk

Chunk mapping score

Read Mapping Controller

Read mapping result

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

To storage

In-memory Seeding

A list of possible locations to the read mapping controller

Basecalled chunk from the GenPIP controller

Query String Generator (QSG)

A substring from a chunk

Generated query string

ReRAM-based RAM
(stores the possible locations i.e., values in the hash table)

ReRAM-based RAM
(stores the possible locations i.e., values in the hash table)

Generated query string

eDRAM Buffer

QSG

Addr.

Query String Generator (QSG)

Addr.

Query

Addr.

Query

Addr.

Query

Addr.

Query

Addr.

Query

Addr.
GenPIP Implementation

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

Raw signals from the sequencing machine

- Basecalling Module
  - Signal chunk
  - ER

- Controller
  - GenPIP
  - ER

- Read Mapping Module
  - In-memory
  - To storage
Outline

- Background and Motivation
- GenPIP: Tight Integration of Genome Analysis Steps
  - Chunk-based Pipeline (CP)
  - Early Rejection (ER)
- GenPIP Implementation
- Evaluation
- Conclusion
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:**
  - **E. coli** ([http://lab.loman.net/2016/07/30/nano pore- r9- data- release/](http://lab.loman.net/2016/07/30/nano pore- r9- data- release/))
  - **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.
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.
Early rejection based on the chunk quality scores technique uses **two and five** sampled chunks for the E. coli and human datasets, respectively.

Early rejection based on the chunk mapping technique uses **five and three** sampled chunks for the E. coli and human datasets, respectively.
More in the Paper

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

Haiyu Mao¹  Mohammed Alser¹  Mohammad Sadrosadati¹  Can Firtina¹  Akanksha Baranwal¹
Damla Senol Cali²  Aditya Manglik¹  Nour Almadhoun Alser¹  Onur Mutlu¹

¹ETH Zürich  ²Bionano Genomics

- Timely early rejection implementation
- In-memory seeding accelerator
- More comparison points
- Sensitivity analysis
- Area and power analysis

Outline

- Background and Motivation
- GenPIP: Tight Integration of Genome Analysis Steps
  - Chunk-based Pipeline (CP)
  - Early Rejection (ER)
- GenPIP Implementation
- Evaluation
- Conclusion
Conclusion

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

Haiyu Mao, Mohammed Alser, Mohammad Sadrosadati, Can Firtina, Akanksha Baranwal, Damla Senol Cali, Aditya Manglik, Nour Almadhoun Alserr, Onur Mutlu