Storage-Centric Computing for Modern Data-Intensive Workloads

Mohammad Sadrosadati
mohammad.sadrosadati@safari.ethz.ch

27 September 2023

Huawei STW 2023
Brief Self Introduction

- Mohammad Sadrosadati
  - Senior Researcher and Lecturer @ SAFARI Research Group, ETHZ
  - Postdoc @ IPM 2019–2021
  - PhD from Sharif University of Technology, 2014-2019
  - mohammad.sadrosadati@safari.ethz.ch

Research Area

- Computer Architecture
- Memory/Storage Systems
- Near-Data Processing
- Heterogeneous System Architecture
- Bioinformatics
- Interconnection Network
The Problem

Computing is Bottlenecked by Data
Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

- Data is increasing
  - We can generate more than we can process
Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks

1800x more compute
In just 2 years

Tomorrow, multi-trillion parameter models

Source: https://youtu.be/Bh13Idwcb0Q?t=283
Data is Key for Future Workloads

**In-memory Databases**
[Mao+, EuroSys’12; Clapp+ (Intel), IISWC’15]

**Graph/Tree Processing**
[Xu+, IISWC’12; Umuroglu+, FPL’15]

**In-Memory Data Analytics**
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

**Datacenter Workloads**
[Kanev+ (Google), ISCA’15]
Data Overwhelms Modern Machines

In-memory Databases

Graph/Tree Processing

Data → performance & energy bottleneck

In-Memory Data Analytics
[Clapp+ (Intel), IISWC’15; Awan+, BDCloud’15]

Datacenter Workloads
[Kanev+ (Google), ISCA’15]
Data is Key for Future Workloads

Chrome  
Google’s web browser

TensorFlow Mobile  
Google’s machine learning framework

VP9  
Video Playback  
Google’s video codec

VP9  
Video Capture  
Google’s video codec
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

VP9

Video Playback

Google’s video codec

VP9

Video Capture

Google’s video codec
Data is Key for Future Workloads

Developement of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

Genome Analysis

1 Sequencing

2 Read Mapping

3 Variant Calling

4 Scientific Discovery

Data → performance & energy bottleneck
We Need Faster & Scalable Genome Analysis

Understanding **genetic variations**, **species**, **evolution**, ...

Predicting the **presence** and **relative abundance** of **microbes** in a sample

Rapid surveillance of **disease outbreaks**

Developing **personalized medicine**

SAFARI

And, many, many other applications ...
Problems with (Genome) Analysis Today

Special-Purpose Machine for Data Generation

General-Purpose Machine for Data Analysis

FAST

SLOW

Slow and inefficient processing capability
Large amounts of data movement

SAFARI This picture is similar for many “data generators & analyzers” today
Beginner Reading on Genome Analysis

Mohammed Alser, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu

“From Molecules to Genomic Variations to Scientific Discovery: Intelligent Algorithms and Architectures for Intelligent Genome Analysis”
Computational and Structural Biotechnology Journal, 2022
[Source code]

Review

From molecules to genomic variations: Accelerating genome analysis via intelligent algorithms and architectures

Mohammed Alser*, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu*

ETH Zurich, Gloriastrasse 35, 8092 Zürich, Switzerland

FPGA-based Near-Memory Analytics


FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications

Gagandeep Singh♦ Mohammed Alser♦ Damla Senol Cali✶
Dionysios Diamantopoulos▽ Juan Gómez-Luna◊
Henk Corporaal* Onur Mutlu♦新陈代谢

♦ETH Zürich ✶Carnegie Mellon University
*Eindhoven University of Technology ▽IBM Research Europe
Near-Memory Acceleration using FPGAs

IBM POWER9 CPU

HBM-based FPGA board

Near-HBM FPGA-based accelerator

Two communication technologies: CAPI2 and OCAPI
Two memory technologies: DDR4 and HBM
Two workloads: Weather Modeling and Genome Analysis
Performance & Energy Greatly Improve

5-27× performance vs. a 16-core (64-thread) IBM POWER9 CPU

12-133× energy efficiency vs. a 16-core (64-thread) IBM POWER9 CPU

HBM alleviates memory bandwidth contention vs. DDR4
In-Storage Genome Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,

"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"


[Talk Slides (pptx) (pdf)]
[Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]
[Talk Video (17 minutes)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi\textsuperscript{1} Jisung Park\textsuperscript{1} Harun Mustafa\textsuperscript{1} Jeremie Kim\textsuperscript{1} Ataberk Olgun\textsuperscript{1} Arvid Gollwitzer\textsuperscript{1} Damla Senol Cali\textsuperscript{2} Can Firtina\textsuperscript{1} Haiyu Mao\textsuperscript{1} Nour Almadhoun Alserr\textsuperscript{1} Rachata Ausavarungnirun\textsuperscript{3} Nandita Vijaykumar\textsuperscript{4} Mohammed Alser\textsuperscript{1} Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich \textsuperscript{2}Bionano Genomics \textsuperscript{3}KMUTNB \textsuperscript{4}University of Toronto
Accelerating Basecalling + Read Mapping

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

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

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (25 minutes)]
[arXiv version]

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

Haiyu Mao\textsuperscript{1} Mohammed Alser\textsuperscript{1} Mohammad Sadrosadati\textsuperscript{1} Can Firtina\textsuperscript{1} Akanksha Baranwal\textsuperscript{1} Damla Senol Cali\textsuperscript{2} Aditya Manglik\textsuperscript{1} Nour Almadhoun Alserr\textsuperscript{1} Onur Mutlu\textsuperscript{1}

\textsuperscript{1}ETH Zürich \textsuperscript{2}Bionano Genomics

[SAFARI](https://arxiv.org/pdf/2209.08600.pdf)
More on Fast & Efficient Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
More on Fast & Efficient Genome Analysis …

Accelerating Genome Analysis
A Primer on an Ongoing Journey

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu
5 April 2022
SPMA Workshop Keynote @ EuroSys

https://www.youtube.com/watch?v=NCagwf0ivT0
Detailed Lectures on Genome Analysis

- Computer Architecture, Fall 2020, Lecture 3a
  - **Introduction to Genome Sequence Analysis** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5](https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5)

- Computer Architecture, Fall 2020, Lecture 8
  - **Intelligent Genome Analysis** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14](https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14)

- Computer Architecture, Fall 2020, Lecture 9a
  - **GenASM: Approx. String Matching Accelerator** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15](https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15)

- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - **Accelerating Genomics** (ETH Zürich, Fall 2020)
  - [https://www.youtube.com/watch?v=rqjI8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId](https://www.youtube.com/watch?v=rqjI8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId)

[SAFARI](https://www.youtube.com/onurmutlulectures)
Data Overwhelms Modern Machines …

- Storage/memory capability

- Communication capability

- Computation capability

- Greatly impacts robustness, energy, performance, cost
A Computing System

- Three key components
- Computation
- Communication
- Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.

Most of the system is dedicated to storing and moving data.

Yet, system is still bottlenecked by memory.
Deeper and Larger Memory Hierarchies

AMD Ryzen 5000, 2020

Core Count: 8 cores/16 threads

L1 Caches: 32 KB per core

L2 Caches: 512 KB per core

L3 Cache: 32 MB shared

AMD increases the L3 size of their 8-core Zen 3 processors from 32 MB to 96 MB

Additional 64 MB L3 cache die stacked on top of the processor die
- Connected using Through Silicon Vias (TSVs)
- Total of 96 MB L3 cache
Deeper and Larger Memory Hierarchies

IBM POWER10, 2020

Cores:
15-16 cores, 8 threads/core

L2 Caches:
2 MB per core

L3 Cache:
120 MB shared
Deeper and Larger Memory Hierarchies

Apple M1 Ultra System (2022)
Data Overwhelms Modern Machines

Chrome

TensorFlow Mobile

Data → performance & energy bottleneck

Video Playback

Google’s video codec

Video Capture

Google’s video codec

VP9
Data Movement Overwhelms Modern Machines


62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand\textsuperscript{1} \hspace{1cm} Saugata Ghose\textsuperscript{1} \hspace{1cm} Youngsok Kim\textsuperscript{2} \\
Rachata Ausavarungnirun\textsuperscript{1} \hspace{1cm} Eric Shiu\textsuperscript{3} \hspace{1cm} Rahul Thakur\textsuperscript{3} \hspace{1cm} Daehyun Kim\textsuperscript{4,3} \\
Aki Kuusela\textsuperscript{3} \hspace{1cm} Allan Knies\textsuperscript{3} \hspace{1cm} Parthasarathy Ranganathan\textsuperscript{3} \hspace{1cm} Onur Mutlu\textsuperscript{5,1}

SAFARI
Axiom

An Intelligent Architecture
Handles Data Well
How to Handle Data Well

- **Ensure data does not overwhelm the components**
  - via intelligent algorithms, architectures & system designs: algorithm-architecture-devices

- **Take advantage of vast amounts of data and metadata**
  - to improve architectural & system-level decisions

- **Understand and exploit properties of (different) data**
  - to improve algorithms & architectures in various metrics
Corollaries: Computing Systems Today …

- Are processor-centric vs. data-centric

- Make designer-dictated decisions vs. data-driven

- Make component-based myopic decisions vs. data-aware
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
  "Intelligent Architectures for Intelligent Computing Systems"
  [Slides (pptx) (pdf)]
  [IEDM Tutorial Slides (pptx) (pdf)]
  [Short DATE Talk Video (11 minutes)]
  [Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com
Data-Centric Architectures
Processing Data
Where It Makes Sense
Process Data Where It Makes Sense

Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php
Processing in/near Memory: An Old Idea


IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 8, AUGUST 1969

Cellular Logic-in-Memory Arrays

WILLIAM H. KAUTZ, MEMBER, IEEE

Abstract—As a direct consequence of large-scale integration, many advantages in the design, fabrication, testing, and use of digital circuitry can be achieved if the circuits can be arranged in a two-dimensional iterative, or cellular, array of identical elementary networks, or cells. When a small amount of storage is included in each cell, the same array may be regarded either as a logically enhanced memory array, or as a logic array whose elementary gates and connections can be “programmed” to realize a desired logical behavior.

In this paper the specific engineering features of such cellular logic-in-memory (CLIM) arrays are discussed, and one such special-purpose array, a cellular sorting array, is described in detail to illustrate how these features may be achieved in a particular design. It is shown how the cellular sorting array can be employed as a single-address, multiword memory that keeps in order all words stored within it. It can also be used as a content-addressed memory, a pushdown memory, a buffer memory, and (with a lower logical efficiency) a programmable array for the realization of arbitrary switching functions. A second version of a sorting array, operating on a different sorting principle, is also described.

Index Terms—Cellular logic, large-scale integration, logic arrays logic in memory, push-down memory, sorting, switching functions.

Fig. 1. Cellular sorting array I.
Processing in/near Memory: An Old Idea


A Logic-in-Memory Computer

HAROLD S. STONE

Abstract—If, as presently projected, the cost of microelectronic arrays in the future will tend to reflect the number of pins on the array rather than the number of gates, the logic-in-memory array is an extremely attractive computer component. Such an array is essentially a microelectronic memory with some combinational logic associated with each storage element.
Challenge and Opportunity for Future

High Performance, Energy Efficient, Sustainable
(All at the Same Time)
The Problem

Data access is the major performance and energy bottleneck

Our current design principles cause great energy waste
(and great performance loss)
The Problem

Processing of data is performed far away from the data
A Computing System

- Three key components
- Computation
- Communication
- Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.

A Computing System

- Three key components
  - Computation
  - Communication
  - Storage/memory

Burks, Goldstein, von Neumann, “Preliminary discussion of the logical design of an electronic computing instrument,” 1946.
We Need A Paradigm Shift To …

- Enable computation with **minimal data movement**

- **Compute where it makes sense** *(where data resides)*

- Make computing architectures more **data-centric**
Many questions ... How do we design the:

- compute-capable storage/memory systems?
- processors & communication units?
- software & hardware interfaces?
- system software, compilers, languages?
- algorithms & theoretical foundations?
We Need to Think Differently from the Past Approaches
Processing inside Storage: Two Approaches

1. In-Flash Processing
2. Processing near Flash Memory
Approach 1: Processing Using Memory

- Take advantage of operational principles of memory to perform bulk data movement and computation in memory
  - Can exploit internal connectivity to move data
  - Can exploit analog computation capability
  - ...

Examples: RowClone, In-DRAM AND/OR, Gather/Scatter DRAM

- RowClone: Fast and Efficient In-DRAM Copy and Initialization of Bulk Data (Seshadri et al., MICRO 2013)
- Fast Bulk Bitwise AND and OR in DRAM (Seshadri et al., IEEE CAL 2015)
- Gather-Scatter DRAM: In-DRAM Address Translation to Improve the Spatial Locality of Non-unit Strided Accesses (Seshadri et al., MICRO 2015)
- "Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology” (Seshadri et al., MICRO 2017)
Vivek Seshadri, Donghyuk Lee, Thomas Mullins, Hasan Hassan, Amirali Boroumand, Jeremie Kim, Michael A. Kozuch, Onur Mutlu, Phillip B. Gibbons, and Todd C. Mowry,
"Ambit: In-Memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology"
Proceedings of the 50th International Symposium on Microarchitecture (MICRO), Boston, MA, USA, October 2017.
[Slides (pptx) (pdf)] [Lightning Session Slides (pptx) (pdf)] [Poster (pptx) (pdf)]
SIMDRAM Framework

- Nastaran Hajinazar, Geraldo F. Oliveira, Sven Gregorio, Joao Dinis Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gomez-Luna, and Onur Mutlu,

"SIMDRAM: An End-to-End Framework for Bit-Serial SIMD Computing in DRAM"


[2-page Extended Abstract]
[Short Talk Slides (pptx) (pdf)]
[Talk Slides (pptx) (pdf)]
[Short Talk Video (5 mins)]
[Full Talk Video (27 mins)]

SIMDRAM: A Framework for Bit-Serial SIMD Processing using DRAM

*Nastaran Hajinazar1,2
Nika Mansouri Ghiasi1

*Geraldo F. Oliveira1
Minesh Patel1
Juan Gómez-Luna1

Sven Gregorio1
Mohammed Alser1
Onur Mutlu1

João Dinis Ferreira1
Saugata Ghose3

1ETH Zürich   2Simon Fraser University   3University of Illinois at Urbana–Champaign
In-DRAM Lookup-Table Based Execution

- João Dinis Ferreira, Gabriel Falcao, Juan Gómez-Luna, Mohammed Alser, Lois Orosa, Mohammad Sadrosadati, Jeremie S. Kim, Geraldo F. Oliveira, Taha Shahroodi, Anant Nori, and Onur Mutlu,

"pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables"

Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.

[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (26 minutes)]
[arXiv version]
[Source Code (Officially Artifact Evaluated with All Badges)]

pLUTo: Enabling Massively Parallel Computation in DRAM via Lookup Tables

João Dinis Ferreira§ Gabriel Falcao† Juan Gómez-Luna§ Mohammed Alser§ Lois Orosa§▼ Mohammad Sadrosadati§ Jeremie S. Kim§ Geraldo F. Oliveira§ Taha Shahroodi‡ Anant Nori* Onur Mutlu§

§ ETH Zürich † IT, University of Coimbra ▼ Galicia Supercomputing Center ‡ TU Delft * Intel

In-DRAM True Random Number Generation

- Ataberk Olgun, Minesh Patel, A. Giray Yağlıkçı, Haocong Luo, Jeremie S. Kim, F. Nisa Bostancı, Nandita Vijaykumar, Oğuz Ergin, and Onur Mutlu,
  "QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips"
  [Slides (pptx) (pdf)]
  [Short Talk Slides (pptx) (pdf)]
  [Talk Video (25 minutes)]
  [SAFARI Live Seminar Video (1 hr 26 mins)]

**QUAC-TRNG: High-Throughput True Random Number Generation Using Quadruple Row Activation in Commodity DRAM Chips**

Ataberk Olgun$^{§†}$, Minesh Patel$^§$, A. Giray Yağlıkçı$^§$, Haocong Luo$^§$, Jeremie S. Kim$^§$, F. Nisa Bostancı$^{§†}$, Nandita Vijaykumar$^{§⊙}$, Oğuz Ergin$^†$, Onur Mutlu$^§$

$^§$ETH Zürich  $^†$TOBB University of Economics and Technology
$^⊙$University of Toronto
In-DRAM True Random Number Generation

- F. Nisa Bostanci, Ataberk Olgun, Lois Orosa, A. Giray Yağılıkçı, Jeremie S. Kim, Hasan Hassan, Oguz Ergin, and Onur Mutlu,

"DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators"

Proceedings of the 28th International Symposium on High-Performance Computer Architecture (HPCA), Virtual, April 2022.
[Slides (pptx) (pdf)]
[Short Talk Slides (pptx) (pdf)]

DR-STRaNGe: End-to-End System Design for DRAM-based True Random Number Generators

F. Nisa Bostancı†§
Jeremie S. Kim§

Ataberk Olgun†§
Hasan Hassan§

Lois Orosa§
Oğuz Ergin†

A. Giray Yağılıkçı§
Onur Mutlu§

†TOBB University of Economics and Technology
§ETH Zürich

In-Flash Bulk Bitwise Execution

- Jisung Park, Roknoddin Azizi, Geraldo F. Oliveira, Mohammad Sadrosadati, Rakesh Nadig, David Novo, Juan Gómez-Luna, Myungsuk Kim, and Onur Mutlu,
"Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory"
Proceedings of the 55th International Symposium on Microarchitecture (MICRO), Chicago, IL, USA, October 2022.
[Slides (pptx) (pdf)]
[Longer Lecture Slides (pptx) (pdf)]
[Lecture Video (44 minutes)]
[arXiv version]

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

Jisung Park$^\text{§}^\text{¥}$ Roknoddin Azizi$^\text{§}$ Geraldo F. Oliveira$^\text{§}$ Mohammad Sadrosadati$^\text{§}$
Rakesh Nadig$^\text{§}$ David Novo$^\text{†}$ Juan Gómez-Luna$^\text{§}$ Myungsuk Kim$^\text{‡}$ Onur Mutlu$^\text{§}$

$^\text{§}$ETH Zürich  $^\text{¥}$POSTECH  $^\text{†}$LIRMM, Univ. Montpellier, CNRS  $^\text{‡}$Kyungpook National University

Flash-Cosmos
In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

Jisung Park, Roknoddin Azizi, Geraldo F. Oliveira, Mohammad Sadrosadati, Rakesh Nadig, David Novo, Juan Gómez Luna, Myungsuk Kim, and Onur Mutlu

MICRO 2022
Executive Summary

- **Background:** Bulk bitwise operations are widely used in many important data-intensive applications, e.g., databases, graph processing, cryptography etc.

- **Problem:**
  - Performance and energy efficiency of bulk bitwise operations are bottlenecked by
  1) data movement between storage and the compute unit in traditional systems
  2) data sensing (serial reading of operands) in prior in-flash processing (IFP) techniques
  - Prior IFP techniques provide low reliability during computation

- **Goal:** Improve performance, energy efficiency and reliability of bulk bitwise operations in in-flash processing

- **Key Idea:** Flash-Cosmos (Flash-Computation with One-Shot Multi-Operand Sensing) is an in-flash processing technique that improves performance, energy efficiency and reliability of bulk bitwise operations using two key techniques:
  - Multi-Wordline Sensing (MWS): Enables multi-operand bulk bitwise operations with a single sensing (read) operation
  - Enhanced SLC-mode Programming (ESP): Increases the voltage margin between the erased and programmed states to provide higher reliability during in-flash computation

- **Key Results:** Flash-Cosmos is evaluated using 160 real 3D NAND flash chips and three real-world workloads
  - Flash-Cosmos improves the performance and energy efficiency by 3.5x and 3.3x over state-of-the-art IFP technique while providing high reliability during computation
Bulk Bitwise Operations

- Hyper-dimensional Computing
- Cryptography
- Set Operations
- Graph Processing
- Genome Analysis
- Databases
- Web Search
Data movement between compute units and the memory hierarchy significantly affects the performance of bulk bitwise operations.
Conventional systems perform outside-storage processing (OSP) after moving the data to host CPU through the memory hierarchy.

The external I/O bandwidth of storage is the main bottleneck for data movement in OSP.
NDP for Bulk Bitwise Operations

**Our focus**
Large data sets that do **not fit** in main memory

**In-Flash**
(e.g., ParaBit⁵)

**In-Storage**
(e.g., Biscuit⁴)

**Near-Data Processing**

**Cache**
(e.g., Compute Cache¹)

**DRAM-based main memory**
(e.g., Ambit²)

**NVM-based main memory**
(e.g., Pinatubo³)

---

In-Storage Processing (ISP)

- ISP performs computation using an in-storage computation unit
- ISP reduces external data movement by transferring only the computation results to the host

Memory Bandwidth
tens to hundreds of GB/s

Storage External I/O Bandwidth
~ 8 GB/s

Storage Internal I/O Bandwidth
~ 9.6 GB/s

Data Movement Bottleneck
In-Storage Processing (ISP)

- ISP performs computation using the in-storage computation unit.
- ISP reduces external data movement by transferring only the computation results to the host.

Storage internal I/O bandwidth is the main bottleneck for data movement in ISP.
In-Flash Processing (IFP)

- IFP performs computation within the flash chips as the data operands are being read serially.
- IFP reduces the internal data movement bottleneck in storage by transferring only the computation results to the in-storage computation unit.
In-Flash Processing (IFP)

- IFP performs computation within the flash chips as the data operands are being read serially.
- IFP reduces the internal data movement bottleneck in storage by transferring only the computation results to the in-storage computation unit.

IFP fundamentally mitigates the data movement.
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique\cite{Gao+2021} performs bulk bitwise operations by controlling the latching circuit of the page buffer.

\cite{Gao+2021} Gao+, “ParaBit: Processing Parallel Bitwise Operations in NAND Flash Memory Based SSDs,” MICRO, 2021
Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique \[^1\] performs bulk bitwise operations by controlling the latching circuit of the page buffer.
Data Sensing Bottleneck in IFP

Data Sensing Bottleneck in IFP

- State-of-the-art IFP technique \(^1\) performs bulk bitwise operations by controlling the latching circuit of the page buffer.

**NAND Flash Chip**

Serial data sensing is the bottleneck in prior in-flash processing techniques.
Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing.
Reliability Issues in IFP

• Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing.
Reliability Issues in IFP

• Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing.

![Diagram of NAND Flash Chip and Page Buffer with data sensing and error markers]

**NAND Flash Chip**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Page Buffer**

Data Sensing
Reliability Issues in IFP

- Prior IFP approaches cannot leverage ECC and data-randomization techniques as computation is performed within the flash chips during data sensing.

Prior IFP techniques requires the application to be highly error-tolerant.
Our Goal

Address the bottleneck of state-of-the-art IFP techniques
(serial sensing of operands)

Make IFP reliable
(provide accurate computation results)
Our Proposal

• Flash-Cosmos enables
  • Computation on multiple operands using a single sensing operation
  • Provide high reliability during in-flash computation
NAND Flash Basics: A Flash Cell

A flash cell stores data by adjusting the amount of charge in the cell.

- **Erased Cell (Low Charge Level)**
  - Operates as a resistor

- **Programmed Cell (High Charge Level)**
  - Operates as an open switch

**Activation**
NAND Flash Basics: A NAND String

• A set of flash cells are **seriously connected** to form a NAND String.
NAND Flash Basics: Read Mechanism

• NAND flash memory reads data by **checking the bitline current**

![Diagram of Bitline (BL) and NAND String with Non-Target Cells serving as resistors regardless of stored data.](Image)
NAND Flash Basics: Read Mechanism

• NAND flash memory reads data by checking the bitline current

Bitline (BL)

Target Cells:
Operate as resistors (1) or open switches (0)

Non-Target Cells:
Operate as resistors regardless of stored data
NAND Flash Basics: Read Mechanism

- NAND flash memory reads data by checking the bitline current

Target Cells:
Operate as resistors (1)
or open switches (0)

Non-Target Cells:
Operate as resistors regardless of stored data

NAND String

BL\textsubscript{i}

BL\textsubscript{j}

Reads as ‘1’ if BL current flows
Reads as ‘0’ if BL current cannot flow
NAND Flash Basics: A NAND Flash Block

• NAND strings connected to different bitlines comprise a NAND block

A single wordline (WL) controls a large number of flash cells: High bit-level parallelism
NAND Flash Basics: Block Organization

- A large number of blocks share the same bitlines
Similarity to Digital Logic Gates

- A large number of blocks share the same bitlines

Cells in the same block are connected serially:
Similar to digital NAND

2-input NAND

\[(A \cdot B)'\]
• A large number of blocks share the same bitlines.

Cells in the same block are connected serially:
Similar to digital NAND

2-input NAND
\[(A \cdot B)'\]

Cells in different blocks are connected in parallel:
Similar to digital NOR

2-input NOR
\[(A + B)'\]
Talk Outline

Motivation

Background

Flash-Cosmos

Evaluation

Summary
Enables in-flash bulk bitwise operations on multiple operands with a single sensing operation using Multi-Wordline Sensing (MWS)
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**: Simultaneously activates multiple WLs in the same block
  - Bitwise AND of the stored data in the WLs

![Diagram of MWS](image)
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**: Simultaneously activates multiple WLs in the same block
  - Bitwise AND of the stored data in the WLs

Non-Target Cells: *Operate as resistors*
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**: Simultaneously activates multiple WLs in the same block
  - **Bitwise AND** of the stored data in the WLs

```
<table>
<thead>
<tr>
<th>BL1</th>
<th>BL2</th>
<th>BL3</th>
<th>BL4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
```

- **Target Cells**: Operate as resistors (1) or open switches (0)
- **Non-Target Cells**: Operate as resistors
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS:** Simultaneously activates multiple WLs in the same block
  - **Bitwise AND** of the stored data in the WLs

A bitline reads as ‘1’ only when all the target cells store ‘1’
→ Equivalent to the bitwise AND of all the target cells
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**: Simultaneously activates multiple WLs in the same block
  - **Bitwise AND** of the stored data in the WLs

![Diagram](image)

**Target Cell**: Operate as a **resistance (1)** or an **open switch (0)**

**Result**: 0 0 0 0
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS**: Simultaneously activates multiple WLs in the same block
  - Bitwise AND of the stored data in the WLs

A bitline reads as ‘1’ only when all the target cells store ‘1’
→ Equivalent to the bitwise AND of all the target cells

Operate as a resistance (1) or an open switch (0)

Result: 0 0 0 0

BL₁ BL₂ BL₃ BL₄
Multi-Wordline Sensing (MWS): Bitwise AND

- **Intra-Block MWS:** Simultaneously activates multiple WLs in the same block
  - Bitwise AND of the stored data in the WLs
<table>
<thead>
<tr>
<th>BL₁</th>
<th>BL₂</th>
<th>BL₃</th>
<th>BL₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Flash-Cosmos (Intra-Block MWS)** enables bitwise AND of multiple pages in the same block via a single sensing operation
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS**: Simultaneously activates multiple WLs in different blocks
- **Bitwise OR** of the stored data in the WLs
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS:** Simultaneously activates multiple WLs in different blocks

- Bitwise OR of the stored data in the WLs

Diagram:
- WL<sub>x</sub> in Block<sub>1</sub>
- WL<sub>y</sub> in Block<sub>i</sub>

Result:
- 1
- 1
- 1
- 0
Multi-Wordline Sensing (MWS): Bitwise OR

• **Inter-Block MWS**: Simultaneously activates multiple WLs in different blocks

  - Bitwise OR of the stored data in the WLs

A bitline reads as ‘0’ only when all the target cells store ‘0’
→ Equivalent to the bitwise OR of all the target cells
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS**: Simultaneously activates multiple WLs in different blocks
- **Bitwise OR** of the stored data in the WLs

\[
\begin{array}{c}
\text{WL}_x \text{ in Block}_1 \\
\text{WL}_y \text{ in Block}_i \\
\text{WL}_y \text{ in Block}_i \\
\end{array}
\]

Result: 1 1 1 1
Multi-Wordline Sensing (MWS): Bitwise OR

- **Inter-Block MWS**: Simultaneously activates multiple WLs in different blocks
  - Bitwise OR of the stored data in the WLs

A bitline reads as ‘0’ only when all the target cells store ‘0’
→ Equivalent to the bitwise OR of all the target cells
Multi-Wordline Sensing (MWS): Bitwise OR

- Inter-Block MWS: Simultaneously activates multiple WLs in different blocks
- Bitwise OR of the stored data in the WLs

Flash-Cosmos (Inter-Block MWS) enables bitwise OR of multiple pages in different blocks via a single sensing operation
Supporting Other Bitwise Operations

Exploit **Inverse Read**\(^1\) which is supported in modern NAND flash memory

Exploit **MWS + Inverse Read**

Use **XOR between sensing and cache latches**\(^2\) which is also supported in NAND flash memory

---

\(^{[1]}\) Lee+, “High-Performance 1-Gb-NAND Flash Memory with 0.12-µm Technology,” JSSC, 2002

\(^{[2]}\) Kim+, “A 512-Gb 3-b/Cell 64-Stacked WL 3-D-NAND Flash Memory,” JSSC, 2018
Flash-Cosmos: Overview

- Enables in-flash bulk bitwise operations on multiple operands with a *single* sensing operation using Multi-Wordline Sensing (MWS)

- Increases the reliability of in-flash bulk bitwise operations by using Enhanced SLC-mode Programming (ESP)
Enhanced SLC-Mode Programming (ESP)

- SLC-mode programming provides a large voltage margin between the erased and programmed states.
- Based on our real device characterization, we observe that SLC-mode programming is still highly error-prone without the use of ECC and data-randomization.
Enhanced SLC-Mode Programming (ESP)

- ESP further increases the voltage margin between the erased and programmed states.
- A wider voltage margin between the two states improves reliability by making the cells less vulnerable to errors.

![Diagram showing increased voltage margin in ESP]

---

SAFARI
Enhanced SLC-Mode Programming (ESP)

- ESP increases the voltage margin between the erased and programmed states.
- A wider voltage margin between the two states improves reliability during data sensing by making the cells less vulnerable to errors.

ESP improves the reliability of in-flash computation without the use of ECC or data-randomization techniques.
Enhanced SLC-Mode Programming (ESP)

- ESP increases the voltage margin between the erased and programmed states.
- A wider voltage margin between the two states improves reliability during data sensing by making the cells less vulnerable to errors.

ESP can improve the reliability of prior in-flash processing techniques as well.

<table>
<thead>
<tr>
<th># of cells</th>
<th>Erased</th>
<th>0 Prog.</th>
<th>0 Prog.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Increased voltage margin in ESP**
- **Threshold voltage**
Evaluation Methodology

• We evaluate Flash-Cosmos using

160 real state-of-the-art 3D NAND flash chips
Real Device Characterization

• We validate the **feasibility, performance, and reliability** of Flash-Cosmos

• 160 48-layer 3D TLC NAND flash chips
  • 3,686,400 tested wordlines

• Under worst-case operating conditions
  • 1-year retention time at 10K P/E cycles
  • Worst-case data patterns
Both intra- and inter-block MWS operations require no changes to the cell array of commodity NAND flash chips.

Both MWS operations can activate multiple WLs (intra: up to 48, inter: up to 4) at the same time with small increase in sensing latency (<10%).

ESP significantly improves the reliability of computation results (no observed bit error in the tested flash cells).
Evaluation Methodology

• We evaluate Flash-Cosmos using

  160 real state-of-the-art 3D NAND flash chips

  Three real-world applications that perform bulk bitwise operations
Evaluation with real-world workloads

• **Simulation**
  • MQSim [Tavakkol+, FAST’18] to model the performance of Flash-Cosmos and the baselines

• **Workloads**
  • Three real-world applications that heavily rely on bulk bitwise operations
  • **Bitmap Indices (BMI):** Bitwise AND of up to \( \sim 1,000 \) operands
  • **Image Segmentation (IMS):** Bitwise AND of \( 3 \) operands
  • **\( k \)-clique star listing (KCS):** Bitwise OR of up to \( 32 \) operands

• **Baselines**
  • **Outside-Storage Processing (OSP):** a multi-core CPU (Intel i7 11700K)
  • **In-Storage Processing (ISP):** an in-storage hardware accelerator
  • **ParaBit [Gao+, MICRO’21]:** the state-of-the-art in-flash processing (IFP) mechanism
Results: Performance & Energy

Flash-Cosmos provides significant performance & energy benefits over all the baselines

The larger the number of operands, the higher the performance & energy benefits.
More in the Paper

Flash-Cosmos: In-Flash Bulk Bitwise Operations Using Inherent Computation Capability of NAND Flash Memory

Jisung Park§∇ Roknoddin Azizi§ Geraldo F. Oliveira§ Mohammad Sadrosadati§
Rakesh Nadig§ David Novo† Juan Gómez-Luna§ Myungsuk Kim‡ Onur Mutlu§

§ETH Zürich  ∇ POSTECH †LIRMM, Univ. Montpellier, CNRS ‡Kyungpook National University

Flash-Cosmos: Summary

- First work to enable multi-operand bulk bitwise operations with a single sensing operation and high reliability.
- Improves performance by 3.5x/25x/32x on average over ParaBit/ISP/OSP across the workloads.
- Improves energy efficiency by 3.3x/13.4x/95x on average over ParaBit/ISP/OSP across the workloads.
- Low-cost & requires no changes to flash cell arrays.
Venice
Improving Solid-State Drive Parallelism at Low Cost via Conflict-Free Accesses

Rakesh Nadig*, Mohammad Sadrosadati*, Haiyu Mao, Nika Mansouri Ghiasi, Arash Tavakkol, Jisung Park, Hamid Sarbazi-Azad, Juan Gómez Luna, and Onur Mutlu
Multiple flash memory chips are connected to the SSD Controller using a shared channel.

I/O requests attempt to simultaneously access the flash chips using a single path.

Path Conflicts cause I/O requests to be transferred serially on the shared channel.

Limits SSD parallelism and overall performance.
Delay Caused by Path Conflicts

• Case 1: Same Channel
  - Path conflict!
  - Flash chip 0
  - Flash chip 1
  - Served serially

• Case 2: Different Channels
  - Saved Cycles
  - Served in parallel

Channel 0:
- Flash chip 0
- CMD
- RD Operation
- Data Transfer

Channel 1:
- Flash chip 4
- CMD
- RD Operation
- Data Transfer
Performance Impact of Path Conflicts

Path conflicts increase the average I/O latency by 57% in our experiments on a performance-optimized SSD.

The performance overhead of path conflicts increases by 1.6x in our experiments for high-I/O-intensity workloads.
Prior Approaches

Baseline SSD

Packetized SSD (pSSD) [1]

Prior Approaches

Baseline SSD and Packetized SSD do not provide path diversity to flash chips
Baseline SSD and Packetized SSD do not provide path diversity to flash chips.
Prior Approaches

Baseline SSD and Packetized SSD do not provide path diversity to flash chips.

Packetized Network SSD and Network-on-SSD
1. do not effectively utilize the path diversity
2. incur large area & cost overheads

Our Goal

To fundamentally address the path conflict problem in SSDs by

1. increasing the number of paths to each flash chip (i.e., path diversity) at low cost

2. effectively utilizing the increased path diversity for communication between the SSD controller and flash chips
Our Proposal

Venice

A low-cost interconnection network of flash chips in the SSD

Conflict-free path reservation for each I/O request

A non-minimal fully-adaptive routing algorithm for path identification

Named after the network of canals in the city of Venice
https://en.wikipedia.org/wiki/Venice
Venice provides increased path diversity at low cost

No modifications to existing flash chips in Venice
Venice uses a small scout packet to reserve a conflict-free path for each I/O request.

Path reservation eliminates path conflicts by enabling conflict-free I/O transfer.
Mitigates path conflicts by efficiently utilizing the path diversity of the SSD interconnection network.

Improves performance by 1.9x/1.5x over the best-performing prior work on performance-optimized/cost-optimized SSD.
Venice: Summary

Mitigates path conflicts by efficiently utilizing the path diversity of the SSD interconnection network.

Improves performance by 1.9x/1.5x over the best-performing prior work on performance-optimized/cost-optimized SSD.

Reduces energy consumption by 46% on average over the most efficient prior work.

---

Energy Consumption (Norm. to Baseline SSD)

- pSSD
- pnSSD
- NoSSD
- Venice

MSR Cambridge
YCSB
Slacker
SYSTOR '17
YCSB RocksDB
AVG

61%
46%
Venice: Summary

- **Mitigates path conflicts** by efficiently utilizing the path diversity of the SSD interconnection network.

- **Improves performance** by 1.9x/1.5x over the best-performing prior work on performance-optimized/cost-optimized SSD.

- **Reduces energy consumption** by 46% on average over the most efficient prior work.

- **Low-cost** and requires no changes to commodity flash chips.
Venice
Improving Solid-State Drive Parallelism at Low Cost via Conflict-Free Free Accesses

Rakesh Nadig*, Mohammad Sadrosadati*, Haiyu Mao, Nika Mansouri Ghiasi, Arash Tavakkol, Jisung Park, Hamid Sarbazi-Azad, Juan Gómez Luna, and Onur Mutlu
Processing inside Storage: Two Approaches

1. In-Flash Processing
2. Processing near Flash Memory
Mindset: Memory as an Accelerator

- CPU core
- mini-CPU core
- video core
- imaging core
- GPU (throughput) core
- GPU (throughput) core
- GPU (throughput) core
- GPU (throughput) core
- LLC
- Memory Controller
- Memory Bus
- Memory
- Specialized compute-capability in memory

Memory similar to a "conventional" accelerator
In-Storage Genomic Data Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alser, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,

"GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"


[Lightning Talk Slides (pptx) (pdf)]
[Lightning Talk Video (90 seconds)]

GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi¹  Jisung Park¹  Harun Mustafa¹  Jeremie Kim¹  Ataberk Olgun¹
Arvid Gollwitzer¹  Damla Senol Cali²  Can Firtina¹  Haiyu Mao¹  Nour Almadhoun Alser¹
Rachata Ausavarungnirun³  Nandita Vijaykumar⁴  Mohammed Alser¹  Onur Mutlu¹

¹ETH Zürich  ²Bionano Genomics  ³KMUTNB  ⁴University of Toronto
GenStore: A High-Performance In-Storage Processing System for Genome Sequence Analysis

Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu

SAFARI

ETH Zürich  bionano GENOMICS  UNIVERSITY OF TORONTO
Genome Sequence Analysis

- **Genome sequence analysis** is critical for many applications
  - Personalized medicine
  - Outbreak tracing
  - Evolutionary studies

- Genome sequencing machines extract smaller fragments of the original DNA sequence, known as reads
Genome Sequence Analysis

- **Read mapping**: first key step in genome sequence analysis
  - Aligns reads to potential matching locations in the reference genome
  - For each matching location, the alignment step finds the degree of similarity (alignment score)

Reference Genome

```
...GCCCATATGGTTAAGCTTCCATGGAAATGGGCTTTCGTTTCCAGAATG...
```

- Differences

```
AAGCTTCCATGG
AAATGGGCTTT
```

- Differences

```
GCTTCCAGAATG
GCCCAAAATGGTT
```

- Calculating the alignment score requires computationally-expensive approximate string matching (ASM) to account for differences between reads and the reference genome due to:
  - Sequencing errors
  - Genetic variation
Genome Sequence Analysis

Data Movement from Storage

- Storage System
- Main Memory
- Cache
- Computation Unit (CPU or Accelerator)

Alignment

- Computation overhead
- Data movement overhead

SAFARI
Accelerating Genome Sequence Analysis

- Heuristics
- Accelerators
- Filters

Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

✓ Computation overhead

✗ Data movement overhead
Key Idea

Filter reads that do not require alignment inside the storage system

Filtered Reads

Exactly-matching reads
Do not need expensive approximate string matching during alignment

Non-matching reads
Do not have potential matching locations and can skip alignment

SAFARI
Challenges

*Filter reads that do not require alignment inside the storage system*

Read mapping workloads can exhibit different behavior

There are **limited hardware resources** in the storage system

SAFEAR
GenStore

Filter reads that do not require alignment inside the storage system

GenStore-Enabled Storage System

Computation overhead

Data movement overhead

GenStore provides significant speedup (1.4x - 33.6x) and energy reduction (3.9x – 29.2x) at low cost
Read Mapping Process

Reference

K-mer Locations

Read

K-mers

Seeding
Determine potential matching locations (seeds) in the reference genome

Seed Filtering (e.g., Chaining)
Prune some seeds in the reference genome

Alignment
Determine the exact differences between the read and the reference genome
Outline

Background

Motivation and Goal

GenStore

Evaluation

Conclusion
Motivation

- Case study on a real-world genomic read dataset
  - Various read mapping systems
  - Various state-of-the-art SSD configurations

The ideal in-storage filter significantly improves performance by

1) reducing the computation overhead
2) reducing the data movement overhead
Motivation

- Case study on a real-world genomic read dataset
  - Various read mapping systems
  - Various state-of-the-art SSD configurations

Filtering outside SSD provides lower performance benefit since it

1) does not reduce the data movement overhead

2) must compete with read mapping for system resources

A HW accelerator reduces the computation bottleneck, which makes I/O a larger bottleneck in the system
Our Goal

Design an in-storage filter for genome sequence analysis in a cost-effective manner

Design Objectives:

Performance
Provide high in-storage filtering performance to overlap the filtering with the read mapping of unfiltered data

Applicability
Support reads with 1) different properties and 2) different degrees of genetic variation in the compared genomes

Low-cost
Do not require significant hardware overhead
GenStore

- **Key idea:** Filter reads that do not require alignment inside the storage system

- **Challenges**
  - Different behavior across read mapping workloads
  - Limited hardware resources in the SSD
Filtering Opportunities

- Sequencing machines produce one of two kinds of reads
  - **Short reads**: highly accurate and short
  - **Long reads**: less accurate and long

**Reads that do not require the expensive alignment step:**

**Exactly-matching reads**
- Do not need expensive approximate string matching during alignment
  - Low sequencing error rates (short reads) combined with
  - Low genetic variation

**Non-matching reads**
- Do not have potential matching locations, so they skip alignment
  - High sequencing error rates (long reads) or
  - High genetic variation (short or long reads)
GenStore

GenStore-EM for Exactly-Matching Reads

GenStore-NM for Non-Matching Reads
GenStore-EM for Exactly-Matching Reads

GenStore-NM for Non-Matching Reads
GenStore-EM

- Efficient in-storage filter for reads with at least one exact match in the reference genome
- Uses simple operations, without requiring alignment
- **Challenge:** large number of random accesses per read to the reference genome and its index

Expensive random accesses to flash chips

Limited DRAM capacity inside the SSD
GenStore-EM: Data Structures

- **Read-sized k-mers**: to reduce the number of accesses per each read

  Reader: GCCCAAAATGGTT

  K-mers: GCC, CCC, ...

  ✓ Only one index lookup per read

- **Sorted read-sized k-mers**: to avoid random accesses to the index

  ✓ Sequential scan of the read set and the index
## GenStore-EM: Data Structures

### Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAA</td>
<td></td>
</tr>
<tr>
<td>AAAAAAAAAAG</td>
<td></td>
</tr>
<tr>
<td>AAAAAAAAAC</td>
<td></td>
</tr>
<tr>
<td>AAAAAAAAACT</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

### Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAA</td>
<td></td>
</tr>
<tr>
<td>AAAAAAAAAAAC</td>
<td></td>
</tr>
<tr>
<td>AAAAAAAAAAT</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- **Sorted Read Table**
- **Sorted K-mer Index**
- **Read-sized K-mers**

**SAFARI**
GenStore-EM: Finding a Match

Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAAA</td>
<td>AAAAAAAAAAAAA</td>
<td>AAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAG</td>
<td>AAAAAAAAAAAAC</td>
<td>AAAAAAAAAAAAAT</td>
</tr>
<tr>
<td>AAAAAAAAAAACT</td>
<td>AAAAAAAAAAAAT</td>
<td>AAAAAAAAAAAAT</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAAA</td>
<td>AAAAAAAAAAAAA</td>
<td>AAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAAC</td>
<td>AAAAAAAAAAAAC</td>
<td>AAAAAAAAAAAAC</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAT</td>
<td>AAAAAAAAAAAAAT</td>
<td>AAAAAAAAAAAAAT</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Comparator

Read = K-mer

Exact match $\Rightarrow$ Filter the read

Next

SAFARI
GenStore-EM: Not Finding a Match

Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
<th>K-mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAA</td>
<td>AAAAAAAAAAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAG</td>
<td>AAAAAAAAAAAAAAAC</td>
</tr>
<tr>
<td>AAAAAAAAAACT</td>
<td>AAAAAAAAAAAAAAT</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAAAC</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAAT</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Comparator

Read > K-mer

Next
GenStore-EM: Not Finding a Match

Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAAG</td>
</tr>
<tr>
<td>AAAAAAAAAAACT</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAAAAAAAAAAA</td>
</tr>
<tr>
<td>AAAAAAAAAAAAAC</td>
</tr>
<tr>
<td>AAAAAAAAAAAAT</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

Comparator

Read < K-mer

Not an exact match ➔ Send to read mapper
GenStore-EM: Not Finding a Match

Sorted Read Table

<table>
<thead>
<tr>
<th>Read</th>
<th>K-mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>AAAA</td>
</tr>
</tbody>
</table>

Sorted K-mer Index

<table>
<thead>
<tr>
<th>K-mer</th>
<th>K-mer</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAA</td>
<td>AAAA</td>
</tr>
</tbody>
</table>

- ✅ Avoids random accesses
- ✅ Simple low-cost logic

Comparator

Read < K-mer

Not an exact match → Send to read mapper
GenStore-EM: Optimization

- Read-sized k-mer index takes up a large amount of space (126 GB for human index) due to the larger number of unique k-mers

Using strong hash values instead of read-sized k-mers reduces the size of the index by 3.9x
GenStore

GenStore-EM for Exactly-Matching Reads

GenStore-NM for Non-Matching Reads
GenStore-NM

- Efficient **chaining-based** in-storage filter to prune most of the non-matching reads

### Seeding
- Determine potential matching locations (seeds) in the reference genome

### Seed Filtering (e.g., Chaining)
- Prune some seeds in the reference genome

### Alignment
- Determine the **exact differences** between the read and the reference genome

**Challenge:** how to perform chaining inside the SSD

- Costly dynamic programming on many seeds in each read
- Particularly **challenging for long reads** with many seeds

SAFARI
GenStore-NM: Mechanism

- GenStore-NM uses a light-weight chaining filter
  - Selectively performs chaining only on reads with a small number of seeds
  - Directly sends reads that require more complex chaining to the host system

**Figures**

- Reads with a sufficiently large number of seeds are very likely to align to the reference genome
- Filters many non-aligning reads without costly hardware resources in the SSD
GenStore-NM: Mechanism

- GenStore-NM uses a light-weight chaining filter
  - Selectively performs chaining only on reads with a small number of seeds
  - Directly sends reads that require more complex chaining to the host system

Reads with a sufficiently large number of seeds are very likely to align to the reference genome

Details on GenStore-NM’s design are in the paper
Evaluation Methodology

Read Mappers
- **Base**: state-of-the-art software or hardware read mappers
  - Minimap2 [Bioinformatics’18]: software mapper for short and long reads
  - GenCache [MICRO’19]: hardware mapper for short reads
  - Darwin [ASPLOS’18]: hardware mapper for long reads
- **GS**: Base integrated with GenStore

SSD Configurations
- **SSD-L**: with SATA3 interface (0.5 GB/s sequential read bandwidth)
- **SSD-M**: with PCIe Gen3 interface (3.5 GB/s sequential read bandwidth)
- **SSD-H**: with PCIe Gen4 interface (7 GB/s sequential read bandwidth)
For a read set with **80% exactly-matching reads**

With the Software Mapper

<table>
<thead>
<tr>
<th></th>
<th>SSD-L</th>
<th>SSD-M</th>
<th>SSD-H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Exec. time</td>
<td>200</td>
<td>150</td>
<td>100</td>
</tr>
<tr>
<td>GS Exec. time</td>
<td>50</td>
<td>75</td>
<td>100</td>
</tr>
</tbody>
</table>

- **2.5x** speedup compared to the software Base

With the Hardware Mapper

<table>
<thead>
<tr>
<th></th>
<th>SSD-L</th>
<th>SSD-M</th>
<th>SSD-H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Exec. time</td>
<td>200</td>
<td>150</td>
<td>100</td>
</tr>
<tr>
<td>GS Exec. time</td>
<td>25</td>
<td>35</td>
<td>50</td>
</tr>
</tbody>
</table>

- **3.3x** speedup compared to the hardware Base

**On average 3.92x energy reduction**
For a read set with 99.7% non-matching reads

With the Software Mapper

With the Hardware Mapper

22.4× – 27.9× speedup compared to the software Base

6.8× – 19.2× speedup compared to the hardware Base

On average 27.2× energy reduction
# Area and Power

- Based on **Synthesis** of **GenStore** accelerators using the Synopsys Design Compiler @ 65nm technology node

<table>
<thead>
<tr>
<th>Logic unit</th>
<th># of instances</th>
<th>Area [mm²]</th>
<th>Power [mW]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comparator</td>
<td>1 per SSD</td>
<td>0.0007</td>
<td>0.14</td>
</tr>
<tr>
<td>K-mer Window</td>
<td>2 per channel</td>
<td>0.0018</td>
<td>0.27</td>
</tr>
<tr>
<td>Hash Accelerator</td>
<td>2 per SSD</td>
<td>0.008</td>
<td>1.8</td>
</tr>
<tr>
<td>Location Buffer</td>
<td>1 per channel</td>
<td>0.00725</td>
<td>0.37375</td>
</tr>
<tr>
<td>Chaining Buffer</td>
<td>1 per channel</td>
<td>0.008</td>
<td>0.95</td>
</tr>
<tr>
<td>Chaining PE</td>
<td>1 per channel</td>
<td>0.004</td>
<td>0.98</td>
</tr>
<tr>
<td>Control</td>
<td>1 per SSD</td>
<td>0.0002</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**Total for an 8-channel SSD**

Only **0.006% of a 14nm Intel Processor**, less than **9.5% of the three ARM processors** in a SATA SSD controller
Other Results in the Paper

- Effect of read set features on performance
  - Data size (up to 440 GB)
  - Filter ratio

- Performance benefit of an implementation of GenStore outside the SSD
  - In some cases, it provides performance benefits due more efficient streaming accesses
  - Provides significantly lower benefit compared to GenStore

- More detailed characterization of non-matching reads across different read mapping use cases and species
Outline

- Background
- Motivation and Goal
- GenStore
- Evaluation
- Conclusion
Conclusion

- There has been significant effort into improving read mapping performance through efficient heuristics, hardware acceleration, accurate filters

- **Problem**: while these approaches address the computation overhead, none of them alleviate the **data movement overhead** from storage

- **Goal**: improve the performance of genome sequence analysis by effectively reducing unnecessary data movement from the storage system

- **Idea**: filter reads that do not require the expensive alignment computation in the storage system to fundamentally reduce the data movement overhead

- **Challenges**:
  - Read mapping workloads can exhibit **different behavior**
  - There are **limited available hardware resources** in the storage system

- **GenStore**: the **first** in-storage processing system designed for genome sequence analysis to reduce both the computation and data movement overhead

- **Key Results**: GenStore provides significant **speedup** (1.4x - 33.6x) and **energy reduction** (3.9x – 29.2x) at low cost
Concluding Remarks
Concluding Remarks

- We must design systems to be **balanced, high-performance, energy-efficient** (all at the same time) → intelligent systems
  - **Data-centric, data-driven, data-aware**

- Enable computation capability inside and close to memory/storage

- **This** can
  - Lead to **orders-of-magnitude** improvements
  - Enable new applications & computing platforms
  - Enable better understanding of nature
  - ...

- Future of **truly data-centric computing** is bright
  - We need to do research & design across the computing stack
Fundamentally Better Architectures

Data-centric

Data-driven

Data-aware
We Need to Revisit the Entire Stack

We can get there step by step
A Blueprint for Fundamentally Better Architectures

Onur Mutlu,
"Intelligent Architectures for Intelligent Computing Systems"
[Slides (pptx) (pdf)]
[IEDM Tutorial Slides (pptx) (pdf)]
[Short DATE Talk Video (11 minutes)]
[Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com
Funding Acknowledgments

- Alibaba, AMD, ASML, Google, Facebook, Hi-Silicon, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware, Xilinx
- NSF
- NIH
- GSRC
- SRC
- CyLab
- EFCL
- SNSF

Thank you!
Onur Mutlu’s SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory

https://safari.ethz.ch/safari-newsletter-january-2021/

Think BIG, Aim HIGH!

https://safari.ethz.ch
All are available at

https://people.inf.ethz.ch/omutlu/projects.htm

https://www.youtube.com/onurmutlulectures

https://github.com/CMU-SAFARI/
Open Source Tools: SAFARI GitHub

SAFARI Research Group at ETH Zurich and Carnegie Mellon University
Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

GitHub Repository:
https://github.com/CMU-SAFAI

Pinned Projects:

- **ramulator** (Public)
  A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in [this](#).
  - C++
  - 371
  - 173

- **MQSim** (Public)
  MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implement...
  - C++
  - 163
  - 97

- **prim-benchmarks** (Public)
  PrIM (Processing-In-Memory benchmarks) is the first benchmark suite for a real-world processing-in-memory (PIM) architecture. PrIM is developed to evaluate, analyze, and characterize the first publi...
  - C
  - 65
  - 25

- **rowhammer** (Public)
  - C
  - 196
  - 41

- **SparseP** (Public)
  SparseP is the first open-source Sparse Matrix Vector Multiplication (SpMV) software package for real-world Processing-In-Memory (PIM) architectures. SparseP is developed to evaluate and characteri...
  - C
  - 40
  - 7

- **SoftMC** (Public)
  SoftMC is an experimental FPGA-based memory controller design that can be used to develop tests for DDR3 SODIMMs using a C++ based API. The design, the interface, and its capabilities and limitatio...
  - Verilog
  - 91
  - 27

https://github.com/CMU-SAFAI/
Storage-Centric Computing for Modern Data-Intensive Workloads

Mohammad Sadrosadati
mohammad.sadrosadati@safari.ethz.ch

27 September 2023

Huawei STW 2023