MegIS
High-Performance, Energy-Efficient, and Low-Cost
Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi
Mohammad Sadrosadati  Harun Mustafa  Arvid Gollwitzer  Can Firtina
Julien Eudine  Haiyu Mao  Joël Lindegger  Meryem Banu Cavlak
Mohammed Alser  Jisung Park  Onur Mutlu

SAFARI

ETH Zürich  POSTECH
Outline

Background
Motivation and Goal
MegIS
Evaluation
Conclusion
What is Metagenomics?

• **Metagenomics**: Study of genome sequences of diverse organisms within a shared environment (e.g., blood, ocean, soil)

• Overcomes the limitations of traditional genomics
  - Bypasses the need for culturing individual species in isolation
What is Metagenomics?

- **Metagenomics**: Study of genome sequences of diverse organisms within a shared environment (e.g., blood, ocean, soil)

Has led to groundbreaking advances

- Precision medicine
- Understanding microbial diversity of an environment
- Discovering early warnings of communicable diseases
Metagenomic Analysis

A large database containing information on many species

Metagenomic sample with species that are not known in advance

Preparation of Input Queries

Query K-mers

Presence/Absence Identification

V. cholerae
SARS-CoV-2
E. coli

Abundance Estimation

SAFARI (e.g., > 100 TBs in emerging databases)
Motivation

• Case study of the performance of metagenomic analysis tools
• With various state-of-the-art SSD configurations

I/O data movement causes significant performance overhead
Motivation

- Case study on the throughput of metagenomic analysis tools
- With Various state-of-the-art SSD configurations

I/O becomes an even larger overhead (by 2.7x) in systems where other bottlenecks are alleviated

I/O data movement causes significant performance overhead
I/O Overhead is Hard to Avoid

I/O overhead due to accessing large, low-reuse data is hard to avoid

- Sampling techniques to shrink database sizes

- Reduce accuracy to levels unacceptable for many use cases

- Keeping all data required by metagenomic analysis completely and always resident in main memory

- Energy inefficient, costly, unscalable, and unsustainable

  - Database sizes increase rapidly (doubling every few months)
  - Different analyses need different databases
Our Goal

*Improve metagenomic analysis* performance *by reducing large data movement overhead from the storage system in a cost-effective manner.*
Challenges of In-Storage Processing

Existing metagenomic analysis approaches cannot be implemented as an in-storage processing system due to **SSD hardware limitations**

- Long **latency of NAND flash** chips
- Limited **DRAM capacity** inside the SSD
- Limited **DRAM bandwidth** inside the SSD
MegIS: Metagenomics In-Storage

• First in-storage system for *end-to-end* metagenomic analysis

• **Idea:** Cooperative in-storage processing for metagenomic analysis
  
  - Hardware/software co-design between

![Diagram of MegIS-Enabled SSD and Host System](image-url)
MegIS’s Steps

**Step 1**: Preparation of Input Queries

- A large database containing information on many species

**Step 2**: Presence/Absence Identification

- Metagenomic sample with species that are not known in advance
- Query K-mers: GCTCA, CTCAT, TCATG, ...
- Presence/Absence: V. cholerae, SARS-CoV-2, E. coli

**Step 3**: Abundance Estimation

- A large database containing information on many species

SAFARI
MegIS Hardware-Software Co-Design

Host System

MegIS-Enabled SSD

- FTL
- Core
- Controller
- Channel#1
- Channel#N

SSD DRAM

- Standard Metadata
Task partitioning and mapping
• Each step executes in its most suitable system

Step 1: Host System

Step 2: MegIS-Enabled SSD
- FTL
- Cores
- Controller
- Channel#1
- Channel#N

Step 3: SSD DRAM
- Standard Metadata
**MegIS Hardware-Software Co-Design**

**Task partitioning and mapping**
- Each step executes in its most suitable system

**Data/computation flow coordination**
- Reduce communication overhead
- Reduce #writes to flash chips

**Step 1**
- Host System

**Step 2**
-MegIS-Enabled SSD
  - FTL
  - Cores
  - SSD Controller
  - Cntrl
  - Channel#1
  - Channel#N

**Step 3**
- SSD DRAM
  - Standard Metadata

---

**SAFARI**
MegIS Hardware-Software Co-Design

**Task partitioning and mapping**
- Each step executes in its most suitable system

**Data/computation flow coordination**
- Reduce communication overhead
- Reduce #writes to flash chips

**Step 1**
- Host System

**Step 2**
- MegIS-Enabled SSD
  - FTL
  - Cores
  - SSD Controller
  - Cntrl
  - Channel#1
  - ... Channel#N

**Step 3**
- SSD DRAM
  - Standard Metadata

**Storage-aware algorithms**
- Enable efficient access patterns to the SSD

**SAFARI**
MegIS Hardware-Software Co-Design

Task partitioning and mapping
- Each step executes in its most suitable system

Data/computation flow coordination
- Reduce communication overhead
- Reduce #writes to flash chips

Step 1
Host System

MegIS-Enabled SSD

Step 2

SSD Controller
- Cores
- SSD DRAM
  - Standard Metadata

Step 3

SSD DRAM

Storage-aware algorithms
- Enable efficient access patterns to the SSD

Lightweight in-storage accelerators
- Minimize SRAM/DRAM buffer spaces needed inside the SSD
MegIS Hardware-Software Co-Design

**Task partitioning and mapping**
- Each step executes in its most suitable system

**Data/computation flow coordination**
- Reduce communication overhead
- Reduce #writes to flash chips

**Step 1**
- Host System

**Step 2**
- MegIS-Enabled SSD
- Storage-aware algorithms
  - Enable efficient access patterns to the SSD
- Data mapping scheme and Flash Translation Layer (FTL)
  - Specialize to the characteristics of metagenomic analysis
  - Leverage the SSD’s full internal bandwidth

**Step 3**
- SSD DRAM
  - Standard Metadata
  - Lightweight in-storage accelerators
    - Minimize SRAM/DRAM buffer spaces needed inside the SSD

SAFARI
Step 1 Overview

**Step 1**
Preparation of Input Queries

**Step 2**
Presence/Absence Identification
- V. cholerae
- SARS-CoV-2
- E. coli

**Step 3**
Abundance Estimation

A large database containing information on many species

Metagenomic sample with species that are not known in advance
Step 1 Overview

MegIS works with **sorted data structures to avoid expensive random accesses to the SSD**

- **Extract k-mers** from the sample
- **Sort** the k-mers (database is sorted offline)

**MegIS executes Step 1 in the host system**

- Benefits from larger DRAM and more powerful computation
- Incurs **fewer writes** to NAND flash chips (than processing this step in the SSD)
- Enables **overlapping** Step 1 with Step 2

**To execute Step 1 efficiently in the host system, MegIS needs to:**

- Avoid significant overhead due to **data transfer time** between the steps
- Minimize **performance** and **lifetime** overheads even when host DRAM cannot hold all query k-mers
Step 1 Design

- Divide k-mers into **independent partitions** by their alphabetical range.
- Can overlap operations on different partitions.
- Pin partitions to the host system or the SSD.
- Avoid unnecessary movement of k-mers due to page swaps.

**Host CPU**

- **Query K-mers:** ACGTTACGATT...
- **K-mers:** ACGTT, CGTTA, GTTAC...

**Partition**

- **Partition A:** ACGTC
- **Partition C:** CATTA, CTATG, CCGCA...
- **Partition G:** GTTAC, GGTCC, GACAG...

**Host DRAM**

- **Overlap Step 1’s sorting with Data transfer and Step 2’s In-storage operations.**

**MegIS-Enabled SSD**

**MegIS-Enabled SSD**

- **Read Input Queries**

**SAFARI**
Step 2 Overview

**Step 1**
Preparation of Input Queries

**Query K-mer**
- GCTCA
- CTCAT
- TCATG
- ...

**Step 2**
Presence/Absence Identification

- V. cholerae
- SARS-CoV-2
- E. coli

**Step 3**
Abundance Estimation

A large database containing information on many species

Metagenomic sample with species that are not known in advance

SAFARI
Step 2 Overview

- **Identify the intersecting k-mers** between the query k-mers and the database k-mers

- **Retrieve the species IDs** of intersecting k-mers

**MegIS executes Step 2 in the SSD**

- Accesses **large data with low reuse**
- Involves **lightweight computation**

**To execute Step 2 efficiently in the SSD, MegIS needs to:**

- Leverage **internal bandwidth** efficiently
- Not require **expensive hardware inside the SSD** (e.g., large DRAM bandwidth/capacity and costly logic units)
Identifying the Intersecting K-mers

- **Challenge**: Limited internal DRAM bandwidth

![Diagram showing SSD Controller, Query K-mers from the Host, Database K-mers from Flash Chips, SSD DRAM, Intersection, MegIS-Enabled SSD.]
Identifying the Intersecting K-mers

- **Challenge**: Limited internal DRAM bandwidth

- **Compute directly on the flash data streams** [Zou+, MICRO’22]

- **Reduce buffer size based on application features**

---

**Diagram**

- **SSD Controller**
  - **Query K-mers from the Host**
  - **Write to DRAM**

- **SSD DRAM**
  - **Intersection**

- **Database K-mers**
  - **Channel #1**
    - AAAA
    - CAAAA
    - ...
    - TAACC

- **Channel #N**
  - AGTTT
  - CCGTG
  - ...
  - TTGGT

- **K-mer Register**
  - Multiple instances for each channel.
Retrieving the Species ID

- MegIS retrieves the species IDs of the **intersecting k-mers** by looking up a **sketch database**

<table>
<thead>
<tr>
<th>K-mer</th>
<th>Sketch Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAAAA</td>
<td></td>
</tr>
<tr>
<td>AAAAC</td>
<td></td>
</tr>
<tr>
<td>AATCC</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Space-Inefficient**

**K-mer Sketch Streaming**

- 7.5x Smaller
- 2.1x Larger

**Space-Efficient**

**Inefficient inside the SSD due to long NAND flash latency**

*K-mer Sketch Streaming is much more suitable for in-storage processing due to its streaming accesses*
Retrieving the Species ID

- MegIS retrieves the species IDs of the intersecting k-mers by looking up a sketch database.

Design details are in the paper.

K-mer Sketch Streaming is much more suitable for in-storage processing due to its streaming accesses.

SAFARI
Step 3

**Step 1**
Preparation of Input Queries

**Query K-mers**
- GCTCA
- CTCAT
- TCATG

**Step 3**
Abundance Estimation

**Step 3 and MegIS FTL are in the paper**

A large database containing information on many species.
<table>
<thead>
<tr>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outline</td>
</tr>
<tr>
<td>Background</td>
</tr>
<tr>
<td>Motivation and Goal</td>
</tr>
<tr>
<td>MegIS</td>
</tr>
<tr>
<td>Evaluation</td>
</tr>
<tr>
<td>Conclusion</td>
</tr>
</tbody>
</table>
Evaluation: Methodology Overview

Performance, Energy, and Power Analysis

Hardware Components
• Synthesized Verilog model for the in-storage accelerators
• MQSim [Tavakkol+, FAST’18] for SSD’s internal operations
• Ramulator [Kim+, CAL’15] for SSD’s internal DRAM

Software Components
Measure on a real system:
• AMD® EPYC® CPU with 128 physical cores
• 1-TB DRAM

Baseline Comparison Points
• Performance-optimized software, Kraken2 [Genome Biology’19]
• Accuracy-optimized software, Metalign [Genome Biology’20]
• PIM hardware-accelerated tool (using processing-in-memory), Sieve [ISCA’21]

SSD Configurations
• SSD-C: with SATA3 interface (0.5 GB/s sequential read bandwidth)
• SSD-P: with PCIe Gen4 interface (7 GB/s sequential read bandwidth)

SAFARI
Evaluation: Speedup over the Software Baselines

MegIS provides significant speedup over both Performance-Optimized and Accuracy-Optimized baselines.
Evaluation: Speedup over the Software Baselines

MegIS provides significant speedup over both Performance-Optimized and Accuracy-Optimized baselines.

MegIS improves performance on both cost-optimized and performance-optimized SSDs.

**SSD-P**

- Low: 1x
- Med: 3x
- High: 12.1x
- GMean: 12.1x

**Sample Genetic Diversity**

- Low
- Med
- High
- GMean
Evaluation: Speedup over the PIM Baseline

**MegIS provides significant speedup over the PIM baseline**
Evaluation: Reduction in Energy Consumption

- On average across different input sets and SSDs

MegIS provides significant energy reduction over the Performance-Optimized, Accuracy-Optimized, and PIM baselines.
Evaluation: Accuracy, Area, and Power

Accuracy

• Same accuracy as the accuracy-optimized baseline

• Significantly higher accuracy than the performance-optimized and PIM baselines
  - 4.6 – 5.2× higher F1 scores
  - 3 – 24% lower L1 norm error

Area and Power

Total for an 8-channel SSD:

• Area: 0.04 mm² (Only 1.7% of the area of three ARM Cortex R4 cores in an SSD controller)

• Power: 7.658 mW
Evaluation: System Cost-Efficiency

- **Cost-optimized system ($)**: With SSD-C and 64-GB DRAM
- **Performance-optimized system ($$$)**: With SSD-P and 1-TB DRAM

MegIS outperforms the baselines *even when running on a much less costly system*
**Evaluation: System Cost-Efficiency**

- **Cost-optimized system ($)**: With SSD-C and 64-GB DRAM
- **Performance-optimized system ($$$)**: With SSD-P and 1-TB DRAM

MegIS improves system **cost-efficiency** and makes metagenomics **more accessible** for wider adoption

<table>
<thead>
<tr>
<th>Perf-Opt ($)</th>
<th>Acc-Opt ($)</th>
<th>Perf-Opt ($$$)</th>
<th>Acc-Opt ($$$)</th>
<th>MegIS ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MegIS outperforms the baselines even when running on a much less costly system
More in the Paper

• MegIS’s performance when running in-storage processing operations on the **cores existing in the SSD controller**

• MegIS’s performance when using the same accelerators **outside SSD**

• **Sensitivity analysis with varying**
  - Database sizes
  - Memory capacities
  - #SSDs
  - #Channels
  - #Samples

• MegIS’s performance for **abundance estimation**
More in the Paper

MegIS: High-Performance, Energy-Efficient, and Low-Cost Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi¹ Mohammad Sadrosadati¹ Harun Mustafa¹ Arvid Gollwitzer¹ Can Firtina¹ Julien Eudine¹ Haiyu Mao¹ Joël Lindegger¹ Meryem Banu Cavlak¹ Mohammed Alser¹ Jisung Park² Onur Mutlu¹

¹ETH Zürich ²POSTECH

- Database sizes
- Memory capacities
- #SSDs
- #Channels
- #Samples

• MegIS’s performance for abundance estimation

Conclusion

Metagenomic analysis suffers from significant storage I/O data movement overhead.

**MegIS**

The first in-storage processing system for end-to-end metagenomic analysis. Leverages and orchestrates processing inside and outside the storage system.

**Improves performance**
- 2.7×–37.2× over performance-optimized software
- 6.9×–100.2× over accuracy-optimized software
- 1.5×–5.1× over hardware-accelerated PIM baseline

**High accuracy**
- Same as accuracy-optimized software
- 4.8× higher F1 scores over performance-optimized/PIM

**Reduces energy consumption**
- 5.4× over performance-optimized software
- 15.2× over accuracy-optimized software
- 1.9× over hardware-accelerated PIM baseline

**Low area overhead**
- 1.7% of the three cores in an SSD controller

SAFARI
MegIS
High-Performance, Energy-Efficient, and Low-Cost Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi
Mohammad Sadrosadati  Harun Mustafa  Arvid Gollwitzer  Can Firtina
Julien Eudine  Haiyu Mao  Joël Lindegger  Meryem Banu Cavlak
Mohammed Alser  Jisung Park  Onur Mutlu

SAFARI
ETH zürich
POSTECH
Backup Slides
Executive Summary  *(I suggest not to present it due to time limits)*

**Problem:** Metagenomic analysis suffers from significant storage I/O data movement overhead

**Goal:** Improve metagenomic analysis performance by reducing storage I/O data movement overhead in a cost-effective manner

**Challenge:** While in-storage processing can be a promising direction, existing metagenomic analysis approaches cannot be implemented in the SSD due to SSD hardware limitations

**Idea:** Cooperative ISP for metagenomics

Capitalize on the strengths of processing both inside and outside the storage system

**MegIS:** The first in-storage processing system for end-to-end metagenomic analysis pipeline

An efficient pipeline between the SSD and the host system to *(i)* leverage and *(ii)* orchestrate the capabilities of both via:

- Task partitioning and mapping
- Data/computation flow coordination
- Storage-aware algorithms
- Lightweight in-storage accelerators
- Specialized data mapping scheme and Flash Translation Layer (FTL)

**Results:** Significant speedup *(1.5x – 100.2x)* and energy reduction *(1.9x – 25.7x)* with high accuracy and at low cost
Step 2.2: Retrieving Tax IDs

• KSS example when retrieving 5- and 4-mers
Step 2.2: Retrieving Tax IDs

- KSS example when retrieving 5- and 4-mers
**Area and Power**

- Based on *synthesis* of **MegIS** 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>Intersect (120-bit)</td>
<td>1 per channel</td>
<td>0.001361</td>
<td>0.284</td>
</tr>
<tr>
<td>k-mer Registers (2 x 120-bit)</td>
<td>1 per channel</td>
<td>0.002821</td>
<td>0.645</td>
</tr>
<tr>
<td>Index Generator (64-bit)</td>
<td>1 per channel</td>
<td>0.000272</td>
<td>0.025</td>
</tr>
<tr>
<td>Control Unit</td>
<td>1 per SSD</td>
<td>0.000188</td>
<td>0.026</td>
</tr>
<tr>
<td><strong>Total for an 8-channel SSD</strong></td>
<td>-</td>
<td><strong>0.04</strong></td>
<td><strong>7.658</strong></td>
</tr>
</tbody>
</table>

*Only 1.7% of the area of three 28-nm ARM Cortex R4 cores in a SATA SSD controller*