

Storage-Centric Computing for Genomics and Metagenomics

Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

6 August 2024

FMS: the Future of Memory and Storage

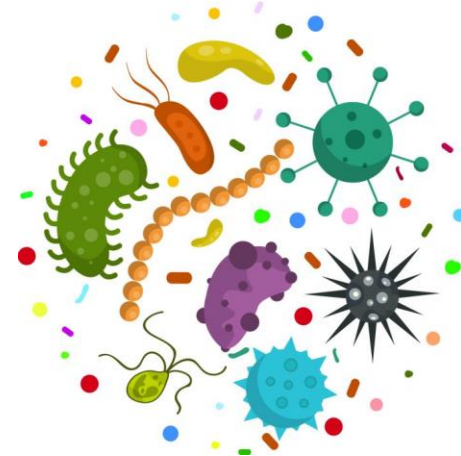
SAFARI

Quick Background & Motivation

We Need Faster & Scalable Genome Analysis



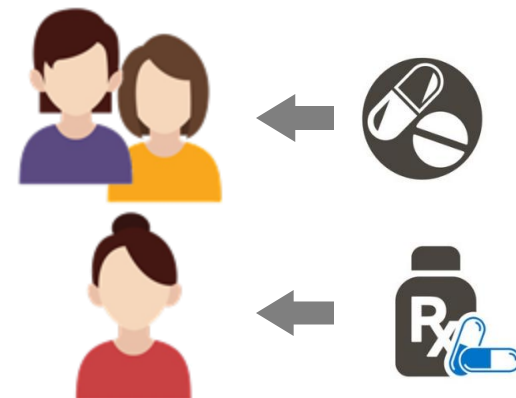
Understanding **genetic variations, species, evolution, ...**



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



Rapid surveillance of **disease outbreaks**

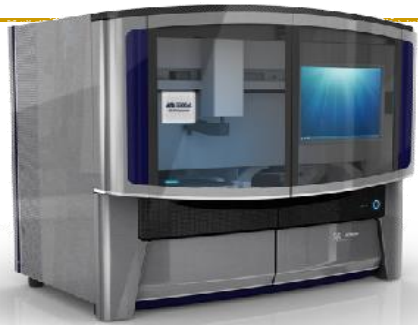


Developing **personalized medicine**

Genome Sequencers



Roche/454



AB SOLiD



Illumina MiSeq



Complete Genomics



Illumina HiSeq2000



Pacific Biosciences RS



Oxford Nanopore MinION



Illumina NovaSeq 6000



Ion Torrent PGM



Ion Torrent Proton



Oxford Nanopore GridION

SAFARI

... and more! All produce data with different properties.

High-Throughput Sequencers



Illumina MiSeq



Pacific
Biosciences
Sequel II

Oxford
Nanopore
PromethION



Oxford Nanopore MinION



Illumina NovaSeq 6000



Pacific Biosciences RS II



Oxford
Nanopore
SmidgION

... and more! All produce data with different properties.

Newer Genome Sequencing Technologies

Nanopore sequencing technology and tools for genome assembly: computational analysis of the current state, bottlenecks and future directions

Damla Senol Cali ✉, Jeremie S Kim, Saugata Ghose, Can Alkan, Onur Mutlu

Briefings in Bioinformatics, bby017, <https://doi.org/10.1093/bib/bby017>

Published: 02 April 2018 **Article history** ▼

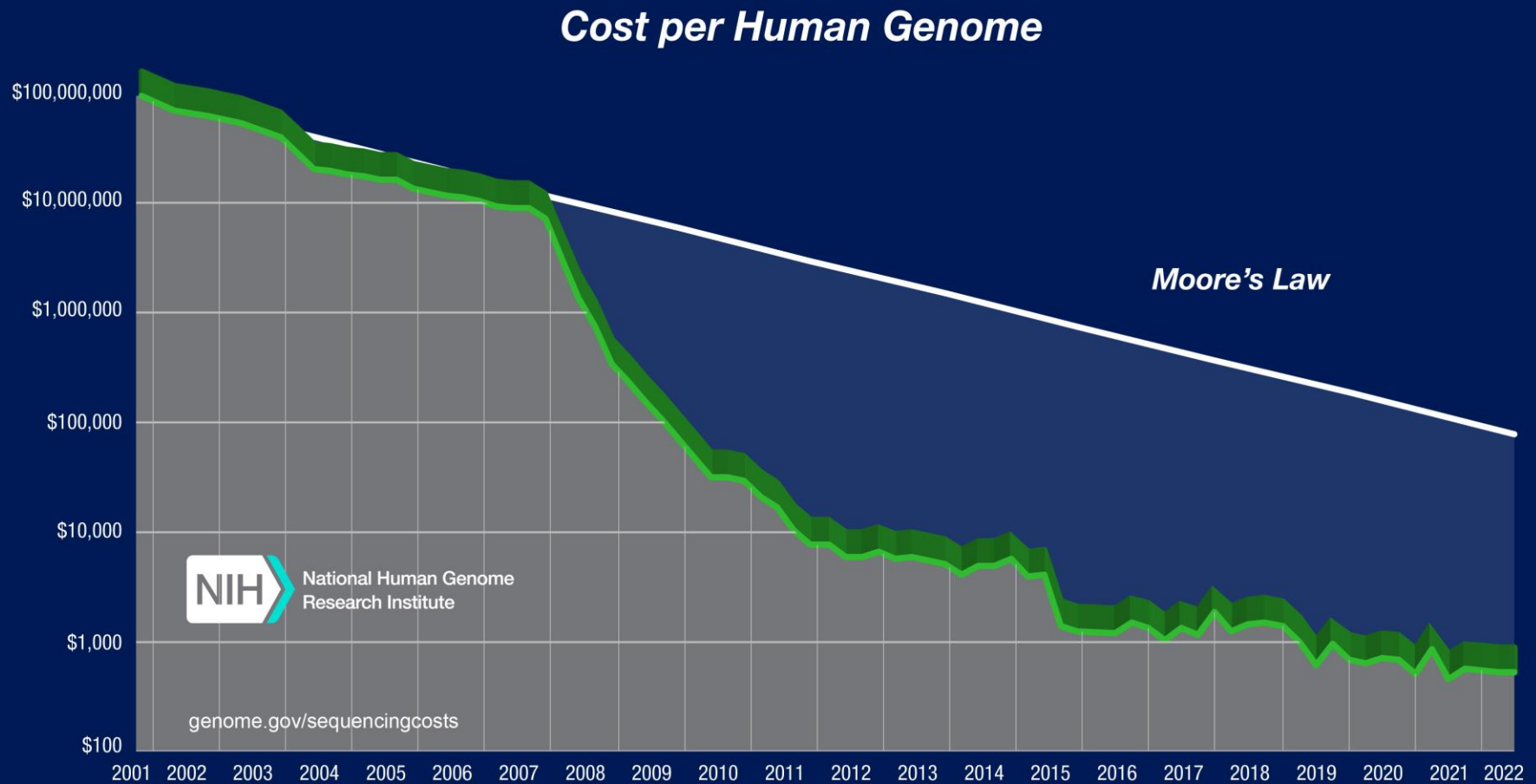


Oxford Nanopore MinION

Senol Cali+, "[Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions](#)," *Briefings in Bioinformatics*, 2018.

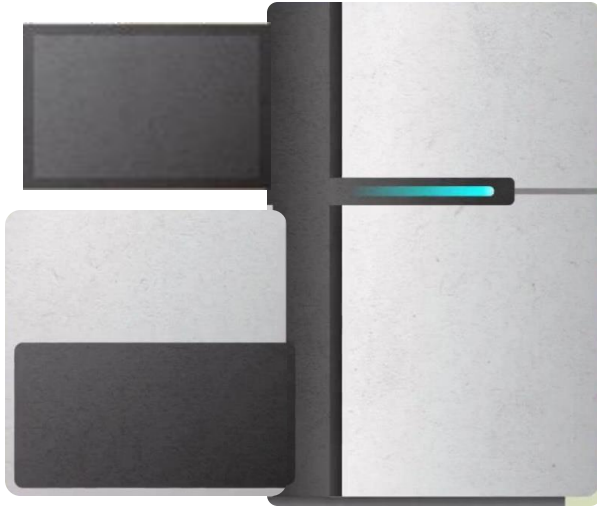
[[Open arxiv.org version](#)] [[Slides \(pptx\) \(pdf\)](#)] [[Talk Video at AACBB 2019](#)]

Genome Sequencing Cost Is Reducing



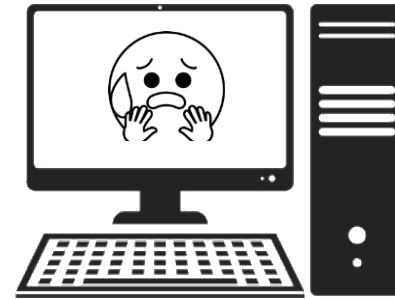
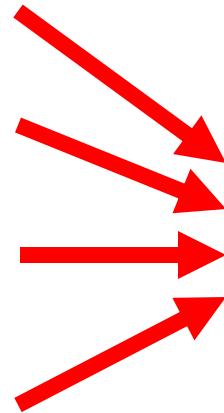
*From NIH (<https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data>)

Problems with (Genome) Analysis Today



Special-Purpose Machine
for **Data Generation**

FAST



General-Purpose Machine
for **Data Analysis**

SLOW

Slow and inefficient processing capability
Large amounts of data movement

Accelerating Genome Analysis [IEEE MICRO 2020]

- Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
"Accelerating Genome Analysis: A Primer on an Ongoing Journey"
IEEE Micro (IEEE MICRO), Vol. 40, No. 5, pages 65-75, September/October 2020.
[\[Slides \(pptx\)\(pdf\)\]](#)
[\[Talk Video \(1 hour 2 minutes\)\]](#)

Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

Saugata Ghose
University of Illinois at Urbana–Champaign and
Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and
Bilkent University

Accelerating Genome Analysis [DAC 2023]

- Onur Mutlu and Can Firtina,
"Accelerating Genome Analysis via Algorithm-Architecture Co-Design"
Invited Special Session Paper in Proceedings of the 60th Design Automation Conference (DAC), San Francisco, CA, USA, July 2023.
[\[arXiv version\]](#)

Accelerating Genome Analysis via Algorithm-Architecture Co-Design

Onur Mutlu Can Firtina
ETH Zürich

Simulating Storage: MQSim [FAST 2018]

- Arash Tavakkol, Juan Gomez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu,
"MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices"
Proceedings of the 16th USENIX Conference on File and Storage Technologies (FAST), Oakland, CA, USA, February 2018.
[[Slides \(pptx\)](#)] [[pdf](#)]
[[Source Code](#)]

MQSim: A Framework for Enabling Realistic Studies of Modern Multi-Queue SSD Devices

Arash Tavakkol[†], Juan Gómez-Luna[†], Mohammad Sadrosadati[†], Saugata Ghose[‡], Onur Mutlu^{†‡}
[†]*ETH Zürich* [‡]*Carnegie Mellon University*

<https://github.com/CMU-SAFARI/MQSim>

Simulating Memory: Ramulator 2.0

- Haocong Luo, Yahya Can Tugrul, F. Nisa Bostanci, Ataberk Olgun, A. Giray Yaglikci, and Onur Mutlu,
"Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator"
*Preprint on **arxiv**, August 2023.*
[\[arXiv version\]](#)
[\[Ramulator 2.0 Source Code\]](#)

Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator

Haocong Luo, Yahya Can Tuğrul, F. Nisa Bostancı, Ataberk Olgun, A. Giray Yağlıkçı, and Onur Mutlu

<https://arxiv.org/pdf/2308.11030.pdf>

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.

👤 440 followers 📍 ETH Zurich and Carnegie Mellon U... 🔗 <https://safari.ethz.ch/> ✉ omutlu@gmail.com

🏠 Overview 📁 Repositories 98 📁 Projects 📦 Packages 👤 People 13

📁 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 the...

● C++ ☆ 532 🍴 206

📁 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 publ...

● C ☆ 126 🍴 47

📁 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++ ☆ 268 🍴 143

📁 rowhammer Public

Source code for testing the Row Hammer error mechanism in DRAM devices. Described in the ISCA 2014 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_isca14.pdf.

● C ☆ 211 🍴 42

📁 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 ☆ 120 🍴 27

📁 Pythia Public

A customizable hardware prefetching framework using online reinforcement learning as described in the MICRO 2021 paper by Bera et al. (<https://arxiv.org/pdf/2109.12021.pdf>).

● C++ ☆ 109 🍴 34

<https://github.com/CMU-SAFARI/>

Genomics Course (Fall 2022)

Fall 2022 Edition:

https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=bioinformatics

Spring 2022 Edition:

https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=bioinformatics

Youtube Livestream (Fall 2022):

https://www.youtube.com/watch?v=nA41964-9r8&list=PL5Q2soXY2Zi8tFIQvdxOdizD_EhVAMVQV

Youtube Livestream (Spring 2022):

https://www.youtube.com/watch?v=DEL_5A_Y3TI&list=PL5Q2soXY2Zi8NrPDgOR1yRU_Cxxjw-u18

Project course

- Taken by Bachelor's/Master's students
- Genomics lectures
- Hands-on research exploration
- Many research readings

<https://www.youtube.com/onurmutlulectures>

SAFARI

Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials
W1	11.3 Fri.	YouTube Live	M1: P&S Accelerating Genomics Course Introduction & Project Proposals 📄 (PDF) 📄 (PPT)	Required Materials Recommended Materials
W2	18.3 Fri.	YouTube Live	M2: Introduction to Sequencing 📄 (PDF) 📄 (PPT)	
W3	25.3 Fri.	YouTube Premiere	M3: Read Mapping 📄 (PDF) 📄 (PPT)	
W4	01.04 Fri.	YouTube Premiere	M4: GateKeeper 📄 (PDF) 📄 (PPT)	
W5	08.04 Fri.	YouTube Premiere	M5: MAGNET & Shouji 📄 (PDF) 📄 (PPT)	
W6	15.4 Fri.	YouTube Premiere	M6: SneakySnake 📄 (PDF) 📄 (PPT)	
W7	29.4 Fri.	YouTube Premiere	M7: GenStore 📄 (PDF) 📄 (PPT)	
W8	06.05 Fri.	YouTube Premiere	M8: GRIM-Filter 📄 (PDF) 📄 (PPT)	
W9	13.05 Fri.	YouTube Premiere	M9: Genome Assembly 📄 (PDF) 📄 (PPT)	
W10	20.05 Fri.	YouTube Live	M10: Genomic Data Sharing Under Differential Privacy 📄 (PDF) 📄 (PPT)	
W11	10.06 Fri.	YouTube Premiere	M11: Accelerating Genome Sequence Analysis 📄 (PDF) 📄 (PPT)	

PIM Course (Fall 2022)

■ Fall 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=processing_in_memory

■ Spring 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2022/doku.php?id=processing_in_memory

■ Youtube Livestream (Fall 2022):

- <https://www.youtube.com/watch?v=QLL0wQ9I4Dw&list=PL5Q2soXY2Zi8KzG2CQYRNQOVD0GOBmKy>

■ Youtube Livestream (Spring 2022):

- <https://www.youtube.com/watch?v=9e4Chnwdovo&list=PL5Q2soXY2Zi-841fUYUYK9EsXKhQKRPyX>

■ Project course

- Taken by Bachelor's/Master's students
- Processing-in-Memory lectures
- Hands-on research exploration
- Many research readings

<https://www.youtube.com/onurmutlulectures>

SAFARI

PIM Review and Open Problem
Processing in Memory Course Meeting 1: Ex
1/13

A Modern Primer on Processing in Memory

Onur Mutlu^{a,b}, Saugata Ghose^{b,c}, Juan Gómez-Luna^a, Rachata Ausavarungnirun^d

SAFARI Research Group

^aCarnegie Mellon University
^bUniversity of Illinois at Chicago
^cUniversity of Illinois at Chicago-Champaign
^dKing Mongkut's University of Technology North Bangkok

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, "A Modern Primer on Processing in Memory", Invited Book Chapter in *Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann*, Springer, to be published in 2021.

Watch on <https://arxiv.org/pdf/1903.03988.pdf> 108

Spring 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	10.03 Thu.	Live	M1: P&S PIM Course Presentation (PDF) (PPT)	Required Materials Recommended Materials	HW 0 Out
W2	15.03 Tue.		Hands-on Project Proposals		
	17.03 Thu.	Premiere	M2: Real-world PIM: UPMEM PIM (PDF) (PPT)		
W3	24.03 Thu.	Live	M3: Real-world PIM: Microbenchmarking of UPMEM PIM (PDF) (PPT)		
W4	31.03 Thu.	Live	M4: Real-world PIM: Samsung HBM-PIM (PDF) (PPT)		
W5	07.04 Thu.	Live	M5: How to Evaluate Data Movement Bottlenecks (PDF) (PPT)		
W6	14.04 Thu.	Live	M6: Real-world PIM: SK Hynix AIM (PDF) (PPT)		
W7	21.04 Thu.	Premiere	M7: Programming PIM Architectures (PDF) (PPT)		
W8	28.04 Thu.	Premiere	M8: Benchmarking and Workload Suitability on PIM (PDF) (PPT)		
W9	05.05 Thu.	Premiere	M9: Real-world PIM: Samsung AxDIMM (PDF) (PPT)		
W10	12.05 Thu.	Premiere	M10: Real-world PIM: Alibaba HB-PNM (PDF) (PPT)		
W11	19.05 Thu.	Live	M11: SpMV on a Real PIM Architecture (PDF) (PPT)		
W12	26.05 Thu.	Live	M12: End-to-End Framework for Processing-using-Memory (PDF) (PPT)		
W13	02.06 Thu.	Live	M13: Bit-Serial SIMD Processing using DRAM (PDF) (PPT)		
W14	09.06 Thu.	Live	M14: Analyzing and Mitigating ML Inference Bottlenecks (PDF) (PPT)		
W15	15.06 Thu.	Live	M15: In-Memory HTAP Databases with HW/SW Co-design (PDF) (PPT)		
W16	23.06 Thu.	Live	M16: In-Storage Processing for Genome Analysis (PDF) (PPT)		
W17	18.07 Mon.	Premiere	M17: How to Enable the Adoption of PIM? (PDF) (PPT)		
W18	09.08 Tue.	Premiere	SS1: ISVLSI 2022 Special Session on PIM (PDF & PPT)		

SSD Course (Spring 2023)

Spring 2023 Edition:

- https://safari.ethz.ch/projects_and_seminars/spring2023/doku.php?id=modern_ssds

Fall 2022 Edition:

- https://safari.ethz.ch/projects_and_seminars/fall2022/doku.php?id=modern_ssds

Youtube Livestream (Spring 2023):

- https://www.youtube.com/watch?v=4VTwOMmsnJY&list=PL5Q2soXY2Zi_8qOM5Icpp8hB2Shtm4z57&pp=iAQB

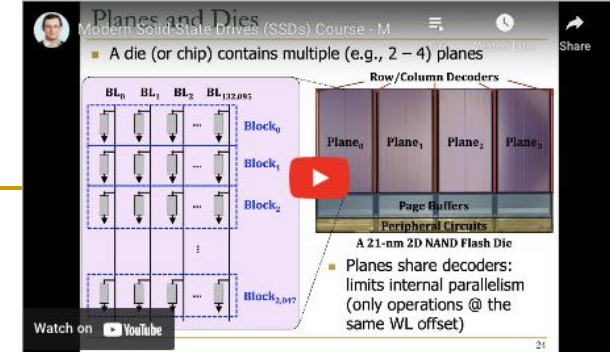
Youtube Livestream (Fall 2022):

- <https://www.youtube.com/watch?v=hqLrd-Uj0aU&list=PL5Q2soXY2Zi9BJhenUq4JI5bwhAMpAp13&pp=iAQB>

Project course

- Taken by Bachelor's/Master's students
- SSD Basics and Advanced Topics
- Hands-on research exploration
- Many research readings

<https://www.youtube.com/onurmutlulectures>



Fall 2022 Meetings/Schedule

Week	Date	Livestream	Meeting	Learning Materials	Assignments
W1	06.10		M1: P&S Course Presentation PDF PPT	Required Recommended	
W2	12.10	YouTube Live	M2: Basics of NAND Flash-Based SSDs PDF PPT	Required Recommended	
W3	19.10	YouTube Live	M3: NAND Flash Read/Write Operations PDF PPT	Required Recommended	
W4	26.10	YouTube Live	M4: Processing inside NAND Flash PDF PPT	Required Recommended	
W5	02.11	YouTube Live	M5: Advanced NAND Flash Commands & Mapping PDF PPT	Required Recommended	
W6	09.11	YouTube Live	M6: Processing inside Storage PDF PPT	Required Recommended	
W7	23.11	YouTube Live	M7: Address Mapping & Garbage Collection PDF PPT	Required Recommended	
W8	30.11	YouTube Live	M8: Introduction to MQSim PDF PPT	Required Recommended	
W9	14.12	YouTube Live	M9: Fine-Grained Mapping and Multi-Plane Operation-Aware Block Management PDF PPT	Required Recommended	
W10	04.01.2023	YouTube Premiere	M10a: NAND Flash Basics PDF PPT	Required Recommended	
			M10b: Reducing Solid-State Drive Read Latency by Optimizing Read-Retry PDF PPT Paper	Required Recommended	
			M10c: Evanescence: Architectural Support for Efficient Data Sanitization in Modern Flash-Based Storage Systems PDF PPT Paper	Required Recommended	
			M10d: DeepSketch: A New Machine Learning-Based Reference Search Technique for Post-Deduplication Delta Compression PDF PPT Paper	Required Recommended	
W11	11.01	YouTube Live	M11: FLIN: Enabling Fairness and Enhancing Performance in Modern NVMe Solid State Drives PDF PPT	Required	
W12	25.01	YouTube Premiere	M12: Flash Memory and Solid-State Drives PDF PPT	Recommended	

In-Storage Genomics & Metagenomics

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 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"**
Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Virtual, February-March 2022.
[[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 Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto

GenStore: A High-Performance and Energy-Efficient In-Storage Computing 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 Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto



<https://arxiv.org/abs/2202.10400>

In-Storage Metagenomics [ISCA 2024]

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

"MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing"

Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, July 2024.

[[Slides \(pptx\)](#)] [[pdf](#)]

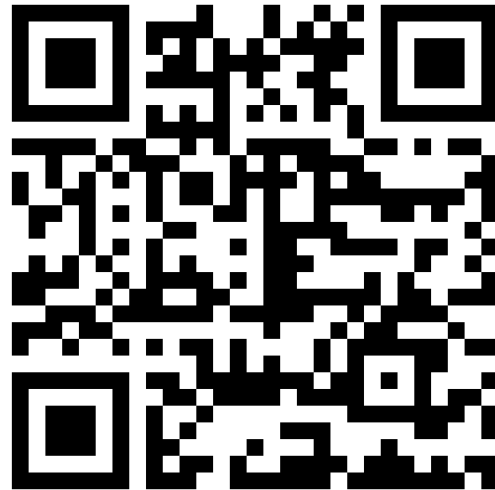
[[arXiv version](#)]

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

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



<https://arxiv.org/abs/2406.19113>

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

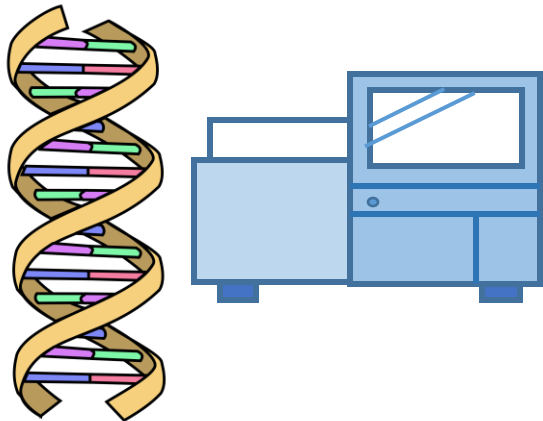
bionano
GENOMICS



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

Data Movement from Storage



Storage System

Main Memory

Cache

Alignment
Computation Unit
(CPU or Accelerator)

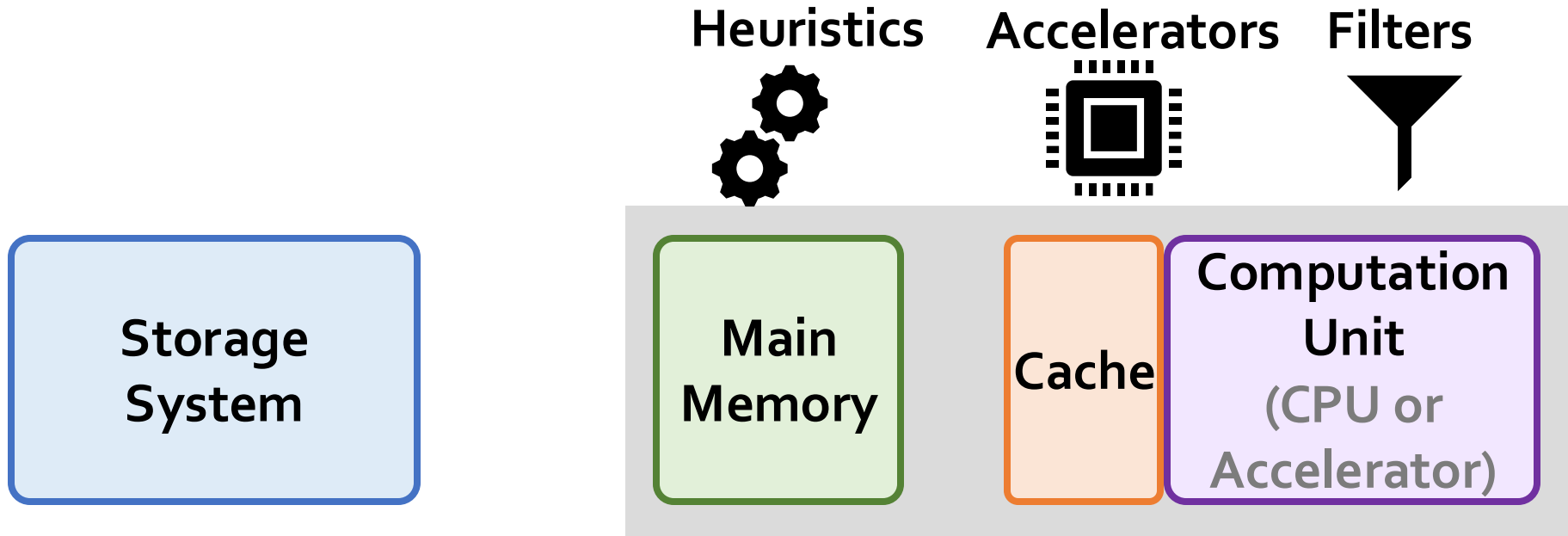


Computation overhead



Data movement overhead

Accelerating Genome Sequence Analysis



Computation overhead

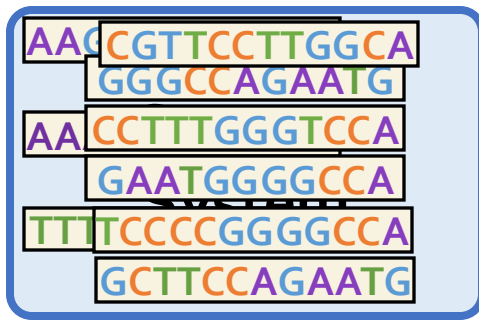


Data movement overhead

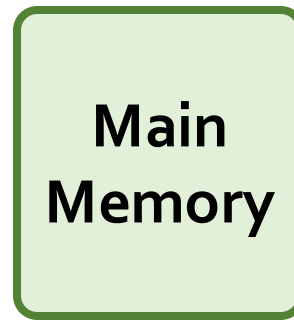
Key Idea



Filter reads that do not require alignment inside the storage system



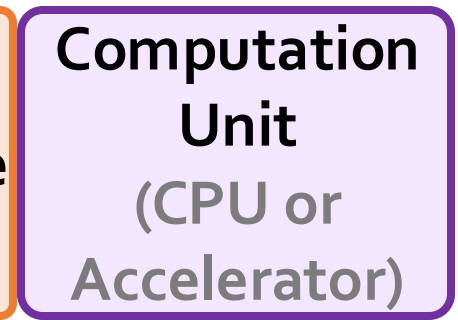
Filtered Reads



**Main
Memory**



Cache



**Computation
Unit
(CPU or
Accelerator)**

Exactly-matching reads

Do not need expensive approximate string matching during alignment

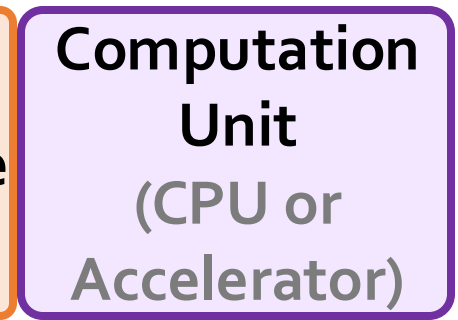
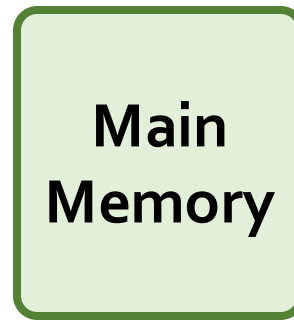
Non-matching reads

Do not have potential matching locations and can skip alignment

Challenges



Filter reads that do not require alignment inside the storage system



Filtered Reads

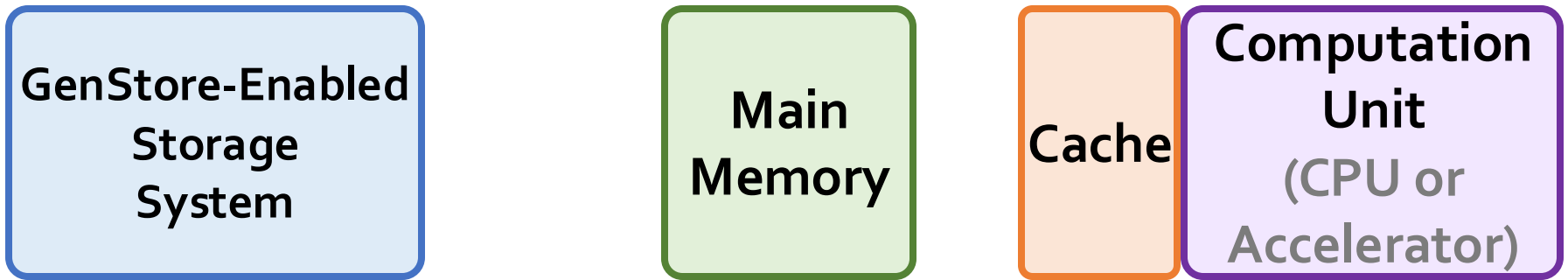
Read mapping workloads can exhibit different behavior

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

GenStore



Filter reads that do not require alignment inside the storage system



Computation overhead

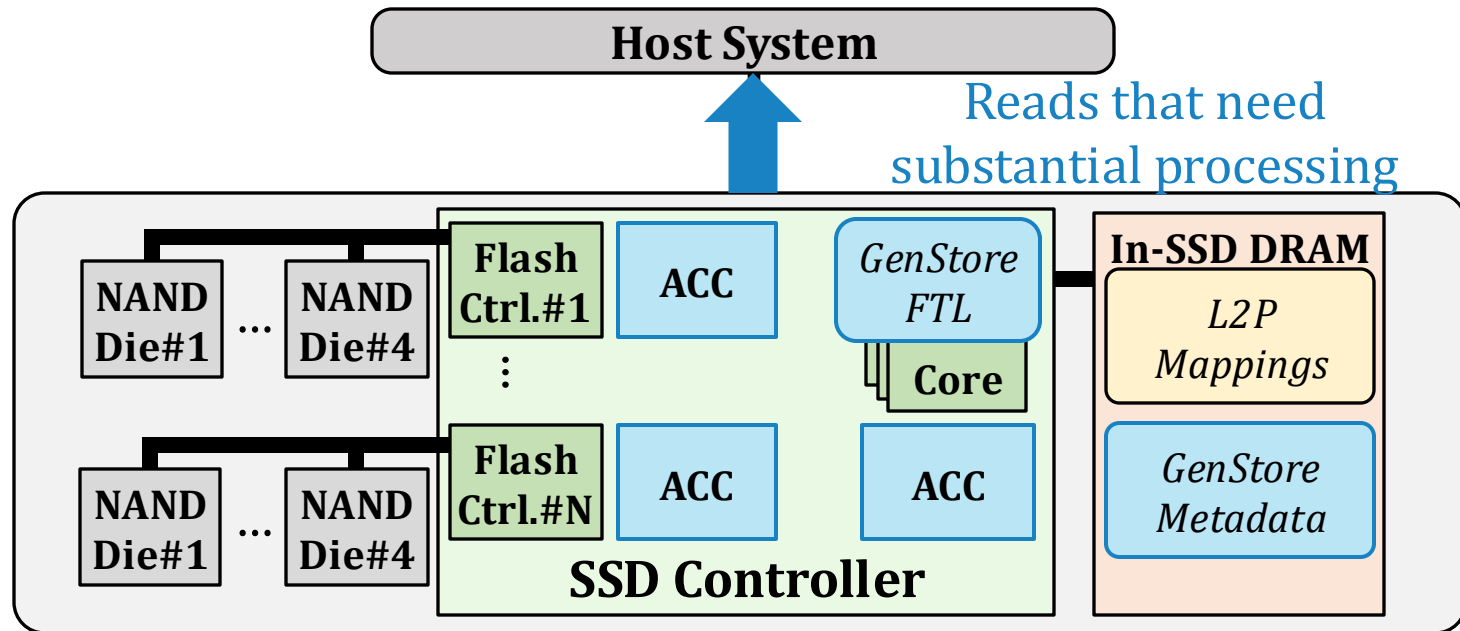


Data movement overhead

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

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

- 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



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



GenStore-EM: Data Structures

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...



Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
...	

Read-sized
K-mers

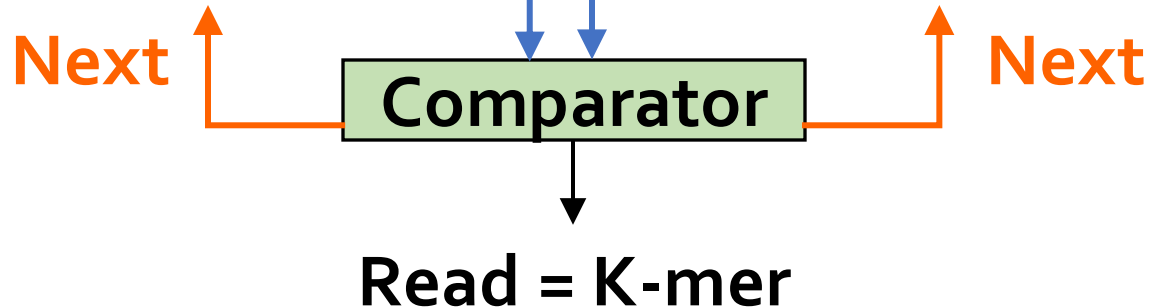
GenStore-EM: Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
...	



Exact match → Filter the read

GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
...	

Comparator

Next

Read > K-mer

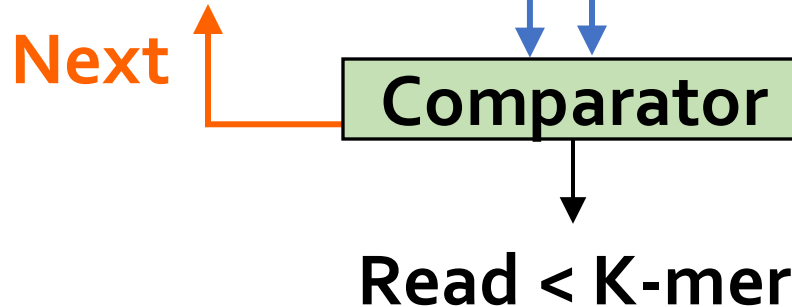
GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAAC	
AAAAAAAAAAAT	
...	



Not an exact match → Send to read mapper

GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	



Avoids random accesses



Simple low-cost logic

Comparator



Read < K-mer

Not an exact match → Send to read mapper

GenStore-EM: Optimization

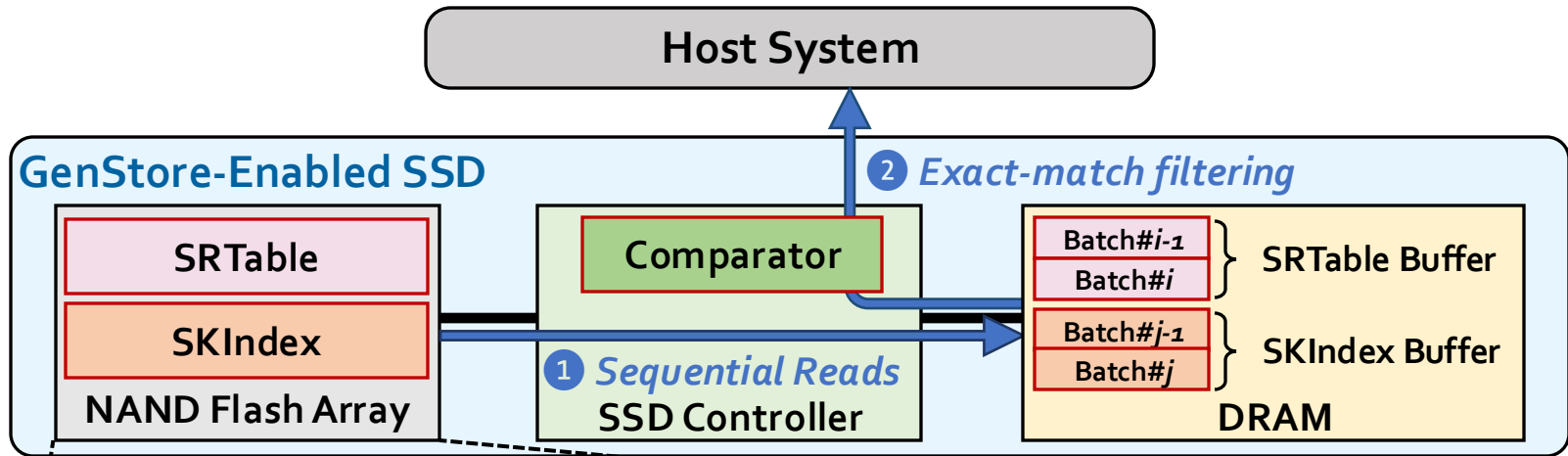
- Read-sized k-mer index takes up a **large amount of space** (126 GB for human index) due to the larger number of unique k-mers

Sorted K-mer Index

Strong Hash Value	Loc.
1	1, 8, ...
4	51
7	23, 37
16	...

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

GenStore-EM: Design



Steps 1 and 2 are **pipelined**.
During filtering, GenStore-EM sends the unfiltered reads to the host system.

Data is evenly distributed between channels, dies, and planes to **leverage the full internal bandwidth** of the SSD

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

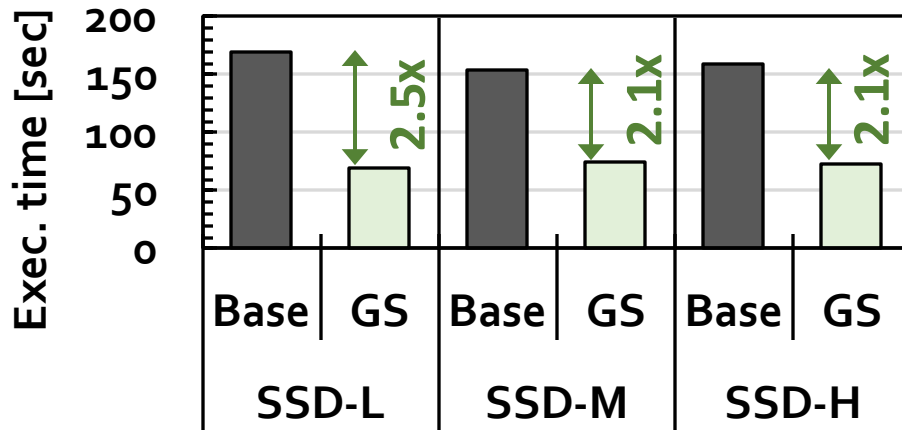
SSD Configurations

- **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)

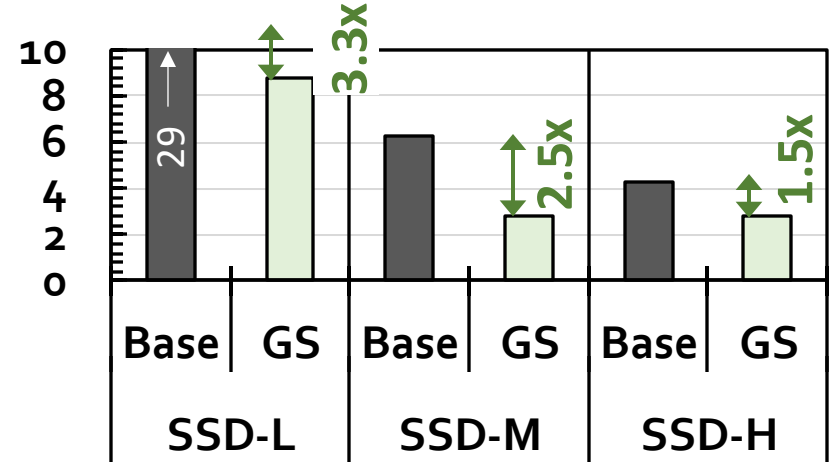
Performance – GenStore-EM

For a read set with 80% exactly-matching reads

With the Software Mapper



With the Hardware Mapper



2.1x - 2.5x speedup compared to the software Base

1.5x – 3.3x speedup compared to the hardware Base

On average 3.92x energy reduction

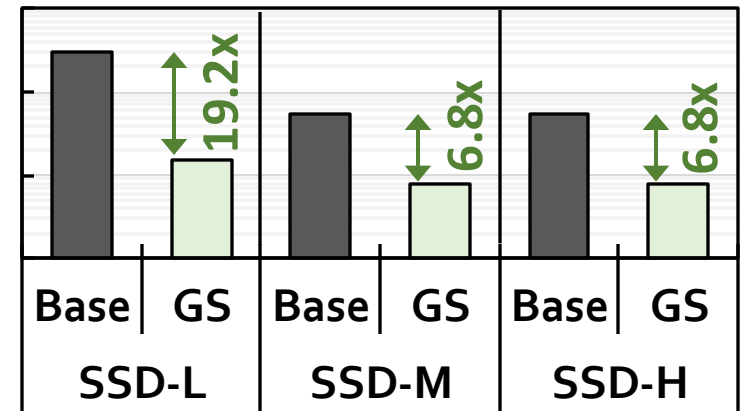
Performance – GenStore-NM

For a read set with 99.7% non-matching reads

With the Software Mapper



With the Hardware Mapper



22.4x – 27.9x speedup compared to the software Base

6.8x – 19.2x speedup compared to the hardware Base

On average 27.2x energy reduction

Area and Power

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

Logic unit	# of instances	Area [mm ²]	Power [mW]
Comparator	1 per SSD	0.0007	0.14
K -mer Window	2 per channel	0.0018	0.27
Hash Accelerator	2 per SSD	0.008	1.8
Location Buffer	1 per channel	0.00725	0.37375
Chaining Buffer	1 per channel	0.008	0.95
Chaining PE	1 per channel	0.004	0.98
Control	1 per SSD	0.0002	0.11
<i>Total for an 8-channel SSD</i>	-	0.2	26.6

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

GenStore Paper, Slides, Video [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"](#)**
Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), Virtual, February-March 2022.
[[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 Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto

GenStore: A High-Performance and Energy-Efficient In-Storage Computing 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 Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto



<https://arxiv.org/abs/2202.10400>

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

In-Storage Metagenomics [ISCA 2024]

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

"MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing"

Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, July 2024.

[[Slides \(pptx\)](#)] [[pdf](#)]

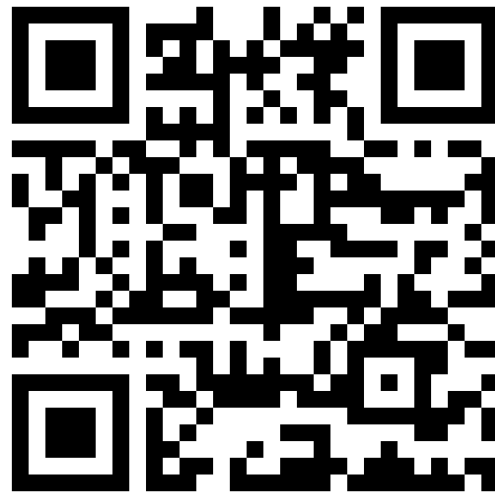
[[arXiv version](#)]

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

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



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

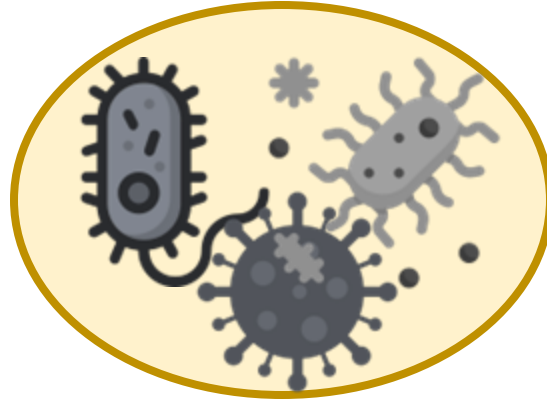
MegIS

Evaluation

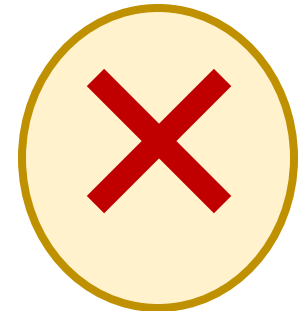
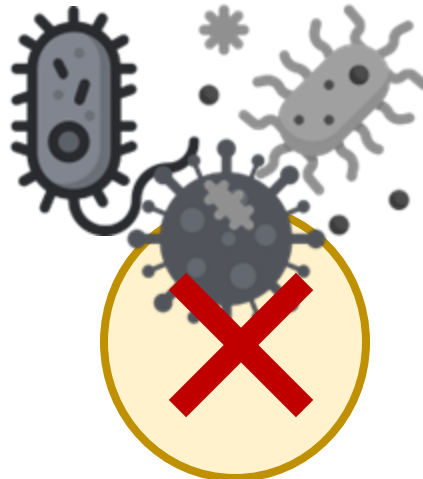
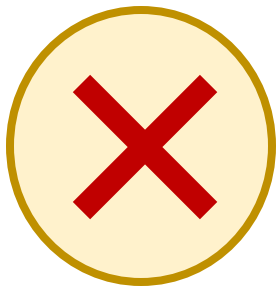
Conclusion

What is Metagenomics?

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

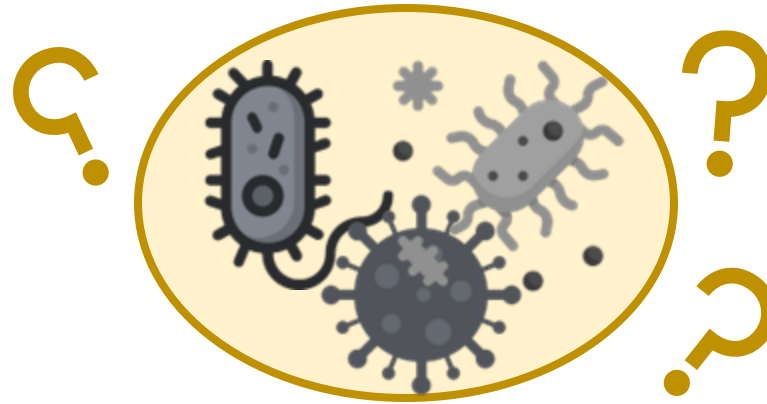


- **Overcomes the limitations of traditional genomics**
 - Bypasses the need for analyzing individual species in isolation



What is Metagenomics?

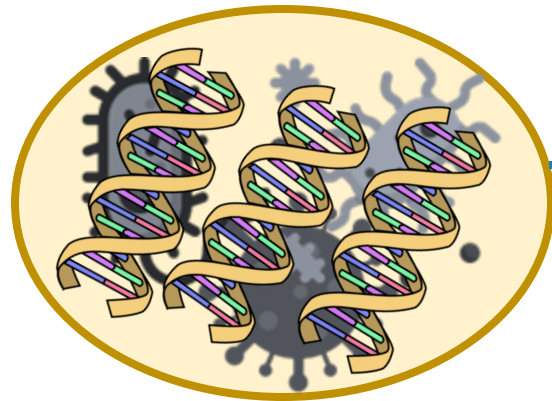
- ***Metagenomics***: Study of genome sequences of **diverse organisms** within a **shared environment** (e.g., blood, ocean, soil)



Has led to groundbreaking advances

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

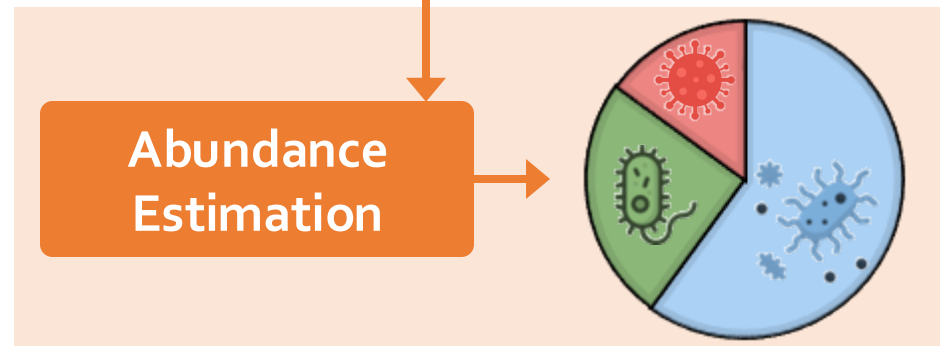
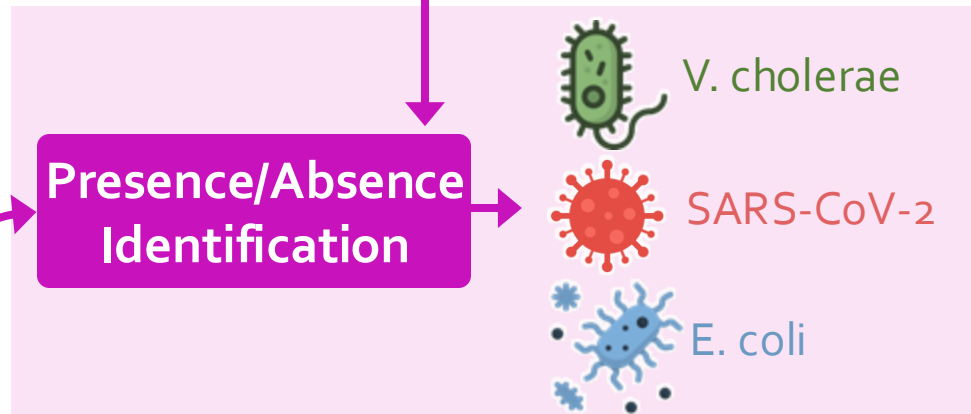
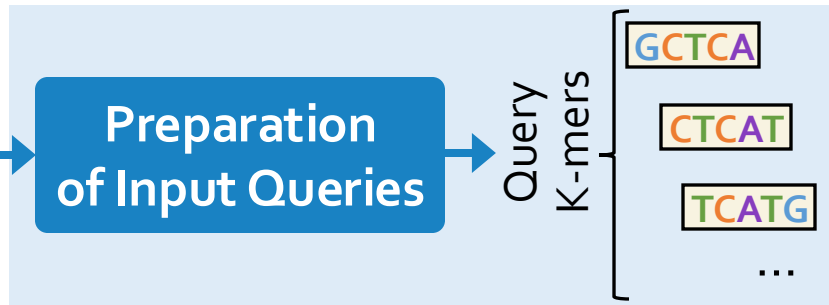
Metagenomic Analysis



Metagenomic sample with species that are **not** known in advance



A large database containing information on **many species**



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

Outline

Background

Motivation and Goal

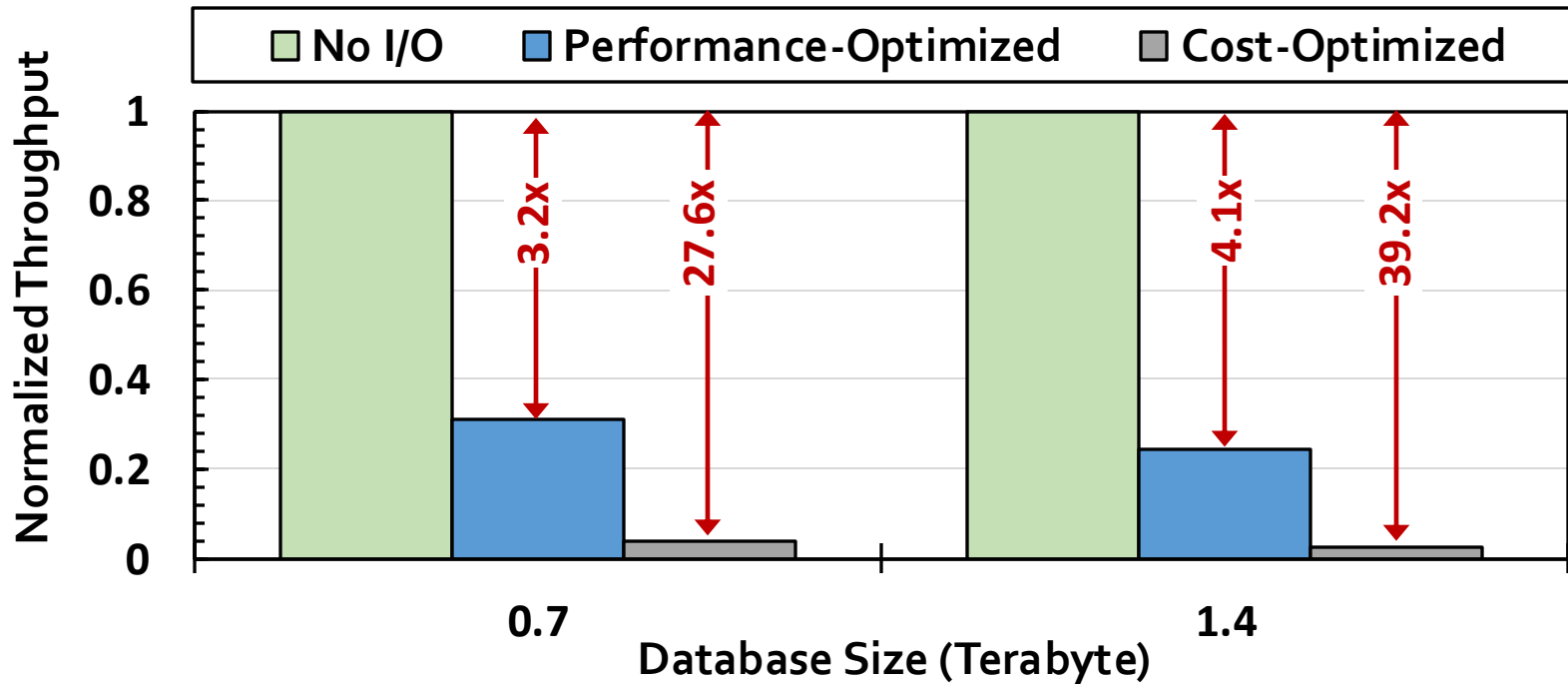
MegIS

Evaluation

Conclusion

Motivation

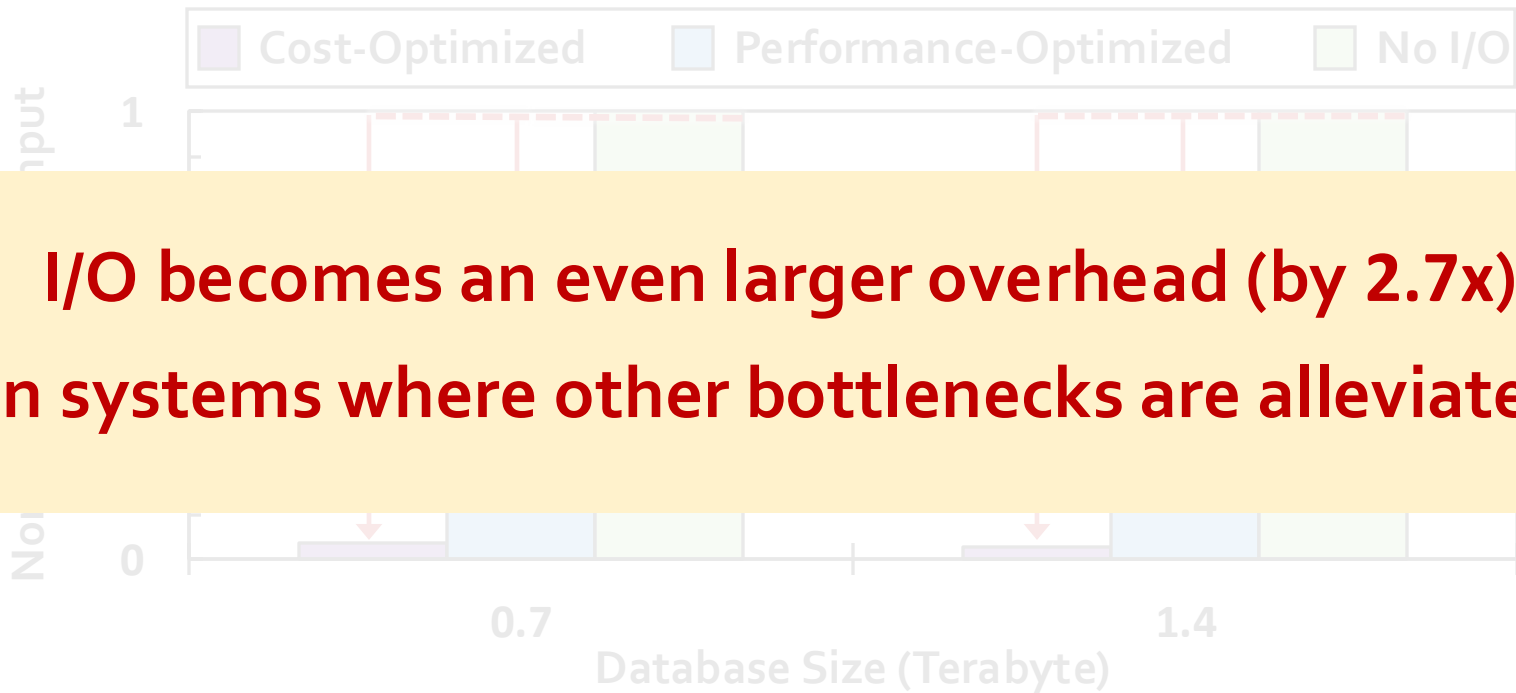
- Case study of the performance of metagenomic analysis tools
- With various state-of-the-art SSD configurations



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

[Wood+, Genome Biology'19], [Ounit+, BMC Genomics'15], [Kim+, Genome Research'16], ...

✗ *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**

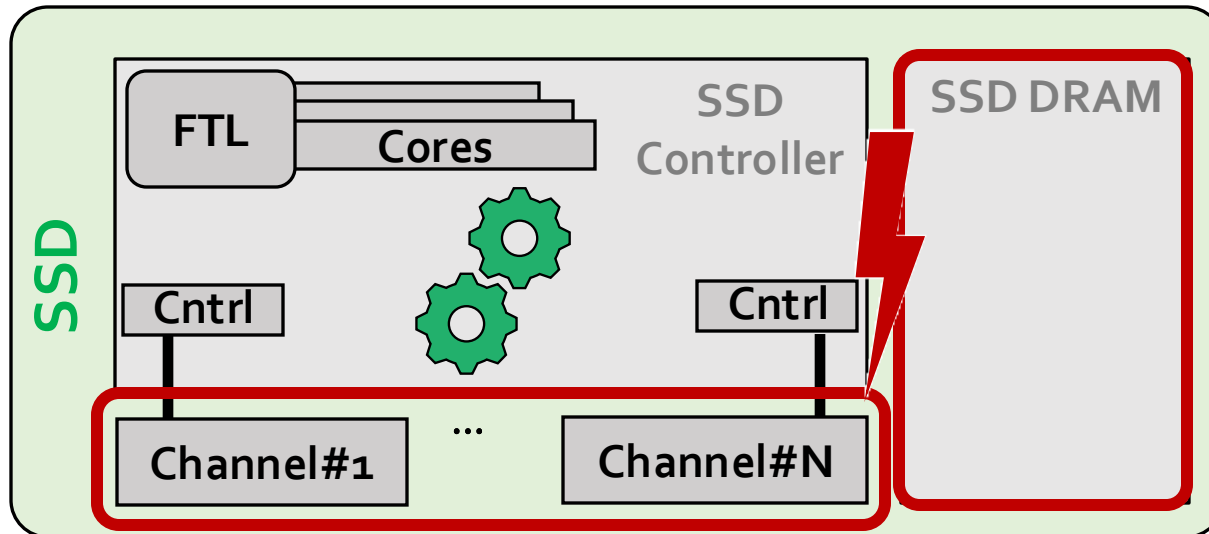
Our Goal

*Improve metagenomic analysis **performance**
by reducing large **data movement overhead**
from the storage system
in a **cost-effective** manner and with **high accuracy***

Challenges of In-Storage Processing

No metagenomic analysis tool can run in-storage due to SSD limits

- Long **latency of NAND flash** chips
- Limited **DRAM capacity** inside the SSD
- Limited **DRAM bandwidth** inside the SSD



Outline

Background

Motivation and Goal

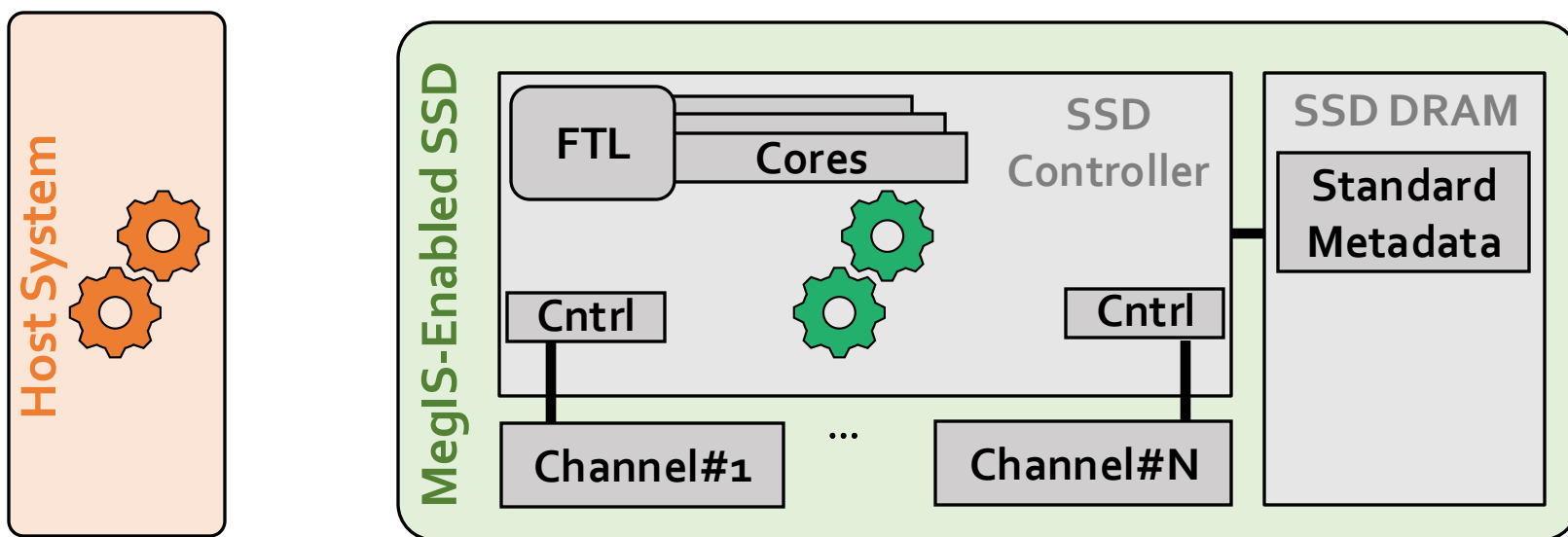
MegIS

Evaluation

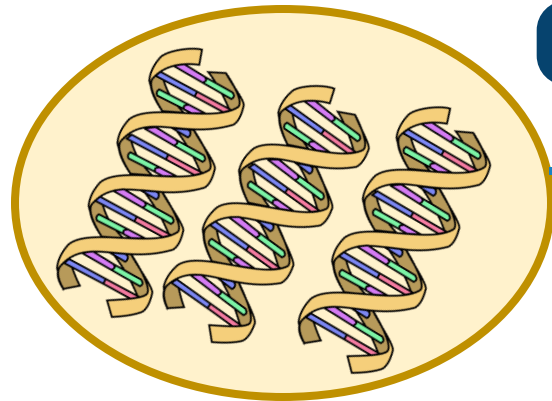
Conclusion

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



MegIS's Steps



Metagenomic sample with species that are **not** known in advance



A large database containing information on **many species**

SAFARI

Step 1

Preparation of Input Queries

Query K-mers

GCTCA
CTCAT
TCATG
...

Step 2

Presence/Absence Identification



V. cholerae



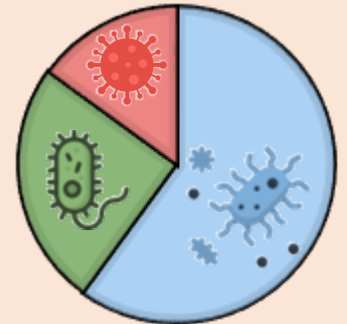
SARS-CoV-2



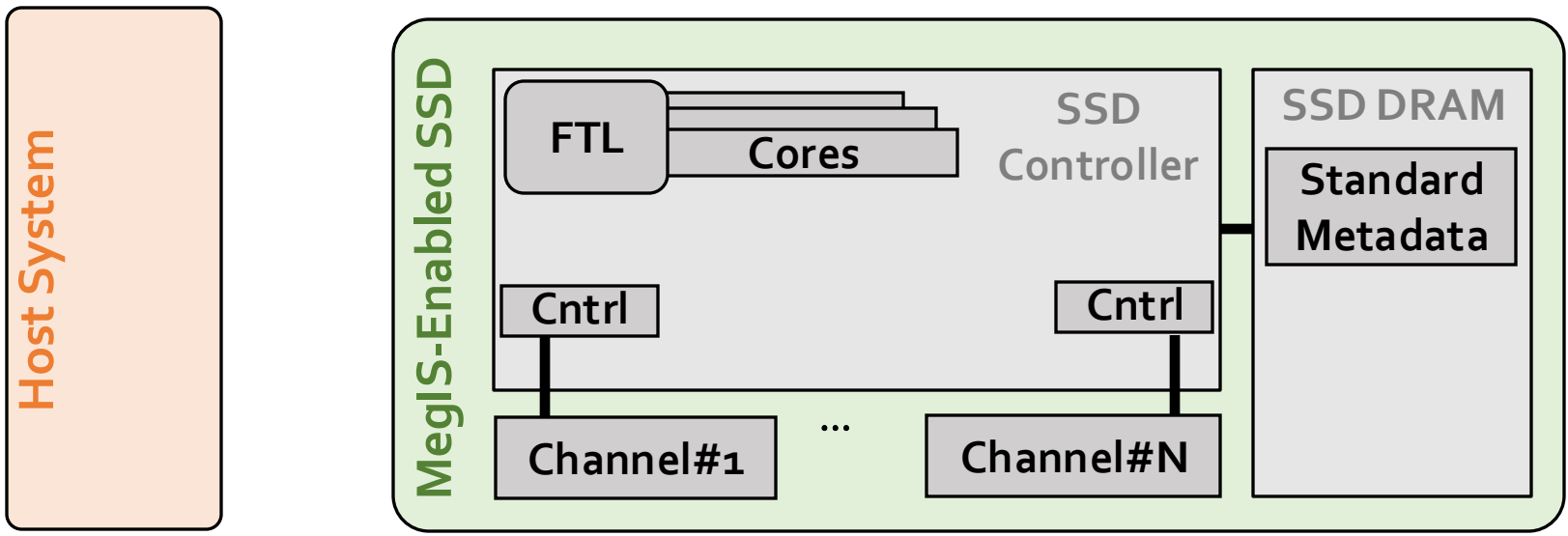
E. coli

Step 3

Abundance Estimation



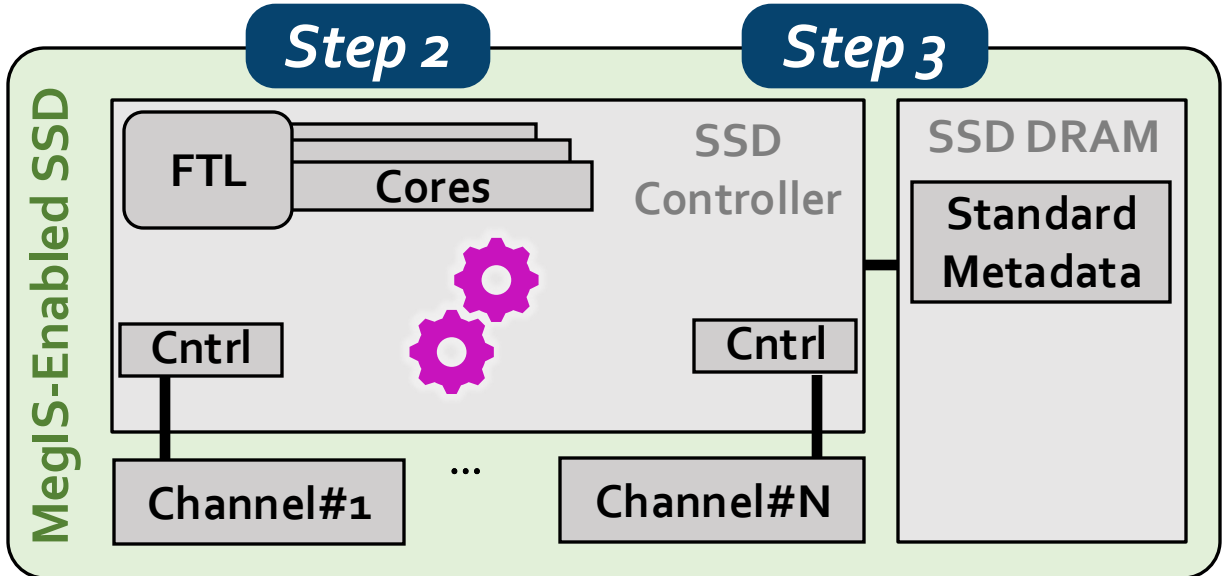
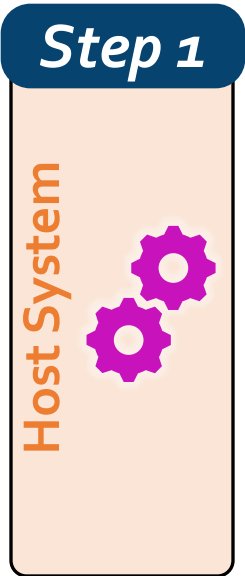
MegIS Hardware-Software Co-Design



MegIS Hardware-Software Co-Design

Task partitioning and mapping

- Each step executes in its most suitable system



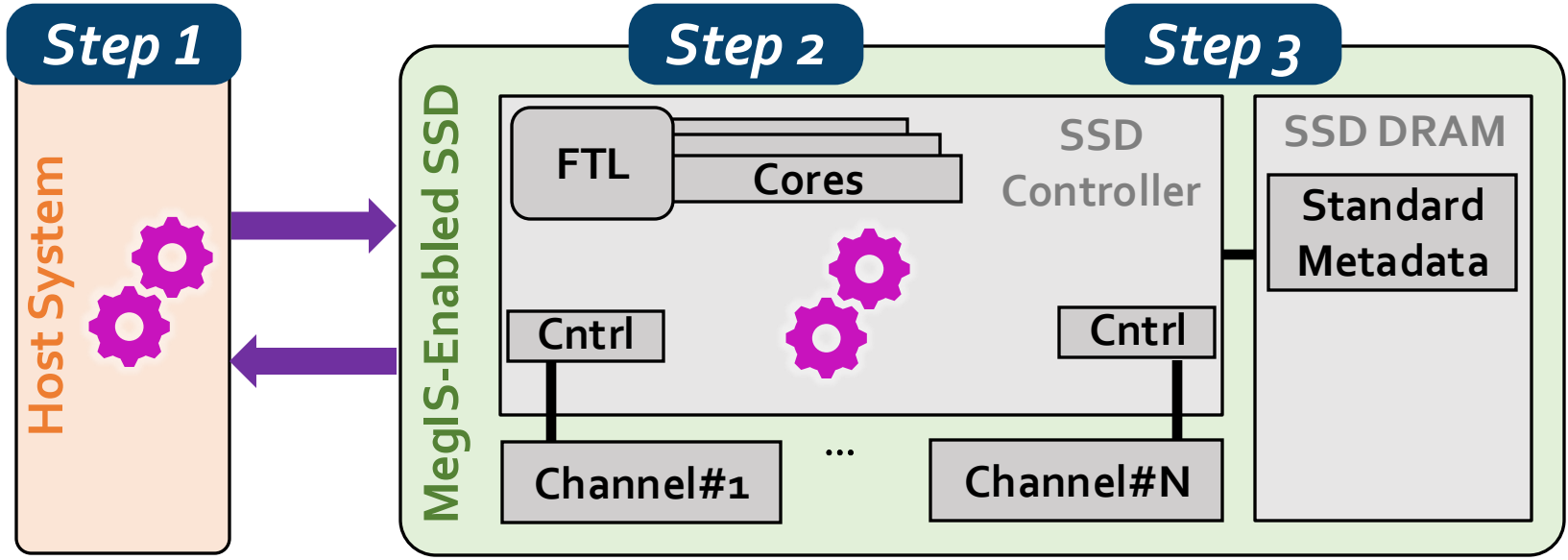
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- Each step executes in its most suitable system

Data/computation flow coordination

- Reduce communication overhead
- Reduce #writes to flash chips



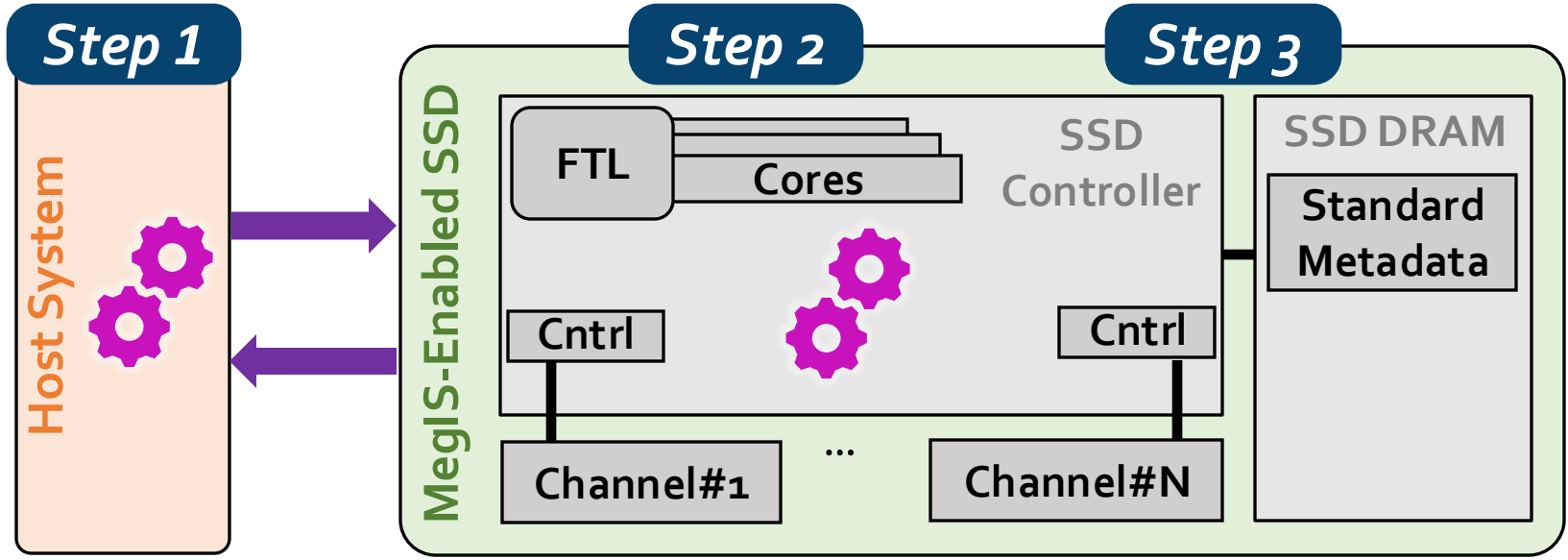
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- Each step executes in its most suitable system

Data/computation flow coordination

- Reduce communication overhead
- Reduce #writes to flash chips



Storage-aware algorithms

- Enable efficient access patterns to the SSD

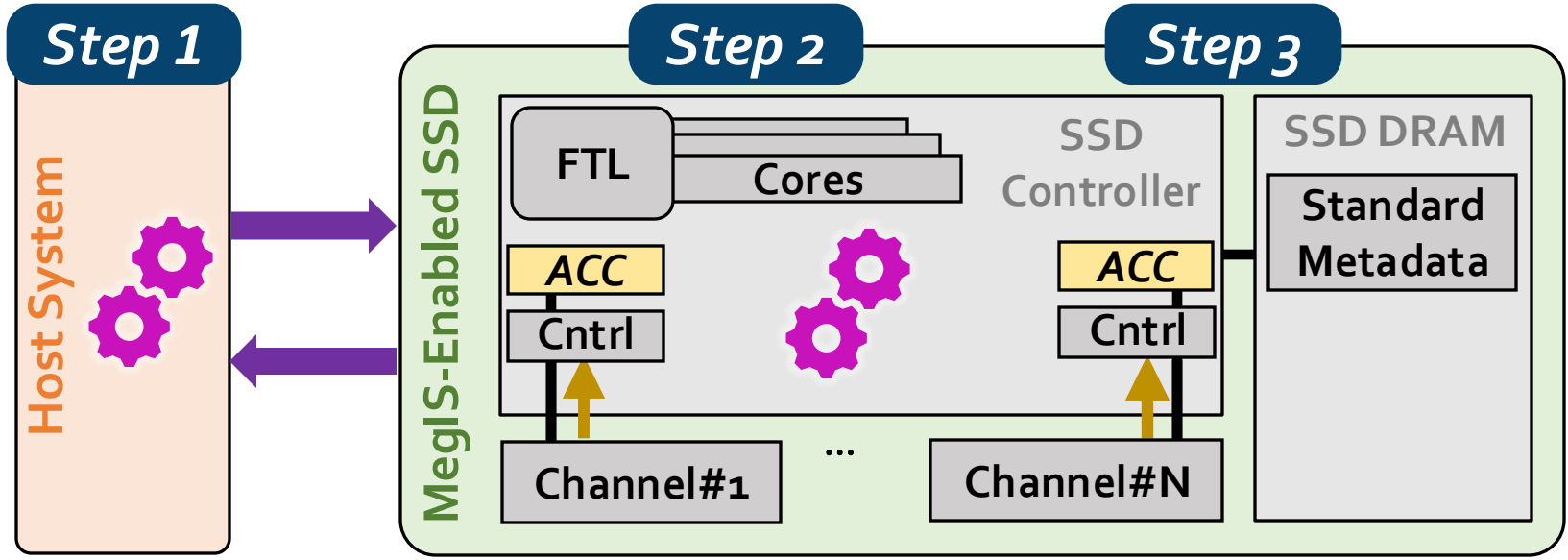
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- Each step executes in its most suitable system

Data/computation flow coordination

- Reduce communication overhead
- Reduce #writes to flash chips



Storage-aware algorithms

- Enable efficient access patterns to the SSD

Lightweight in-storage accelerators

- Minimize SRAM/DRAM buffer spaces needed inside the SSD

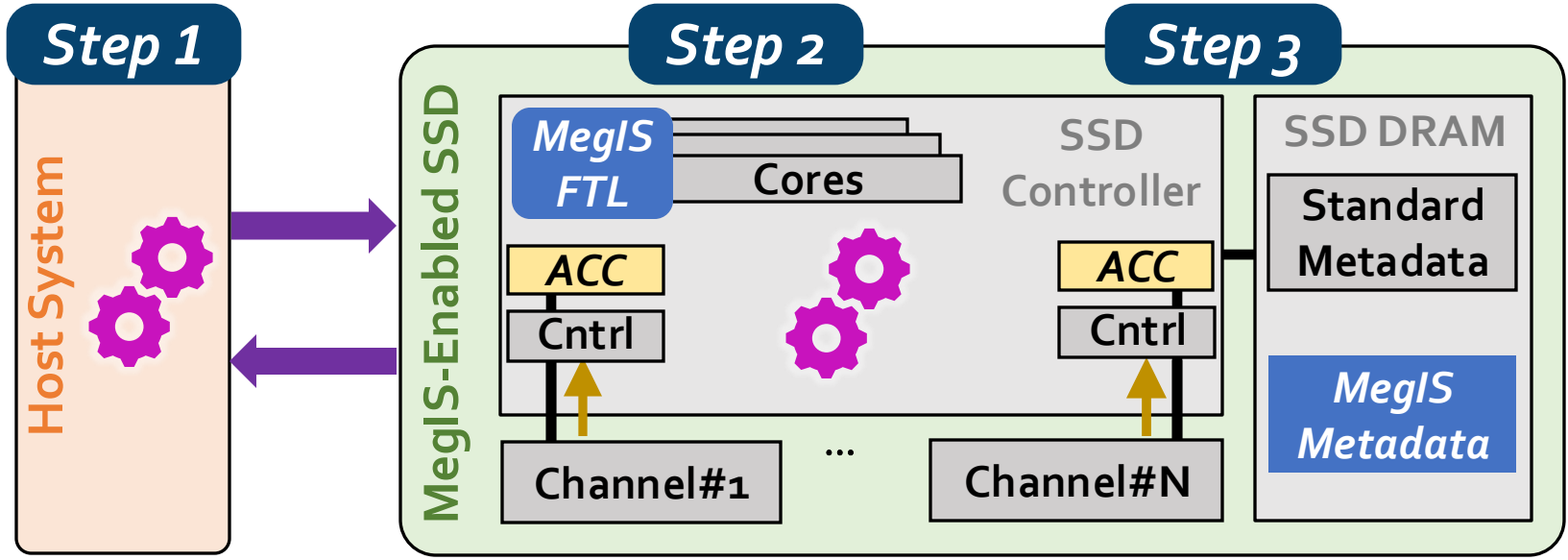
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- Each step executes in its most suitable system

Data/computation flow coordination

- Reduce communication overhead
- Reduce #writes to flash chips



Storage-aware algorithms

- Enable efficient access patterns to the SSD

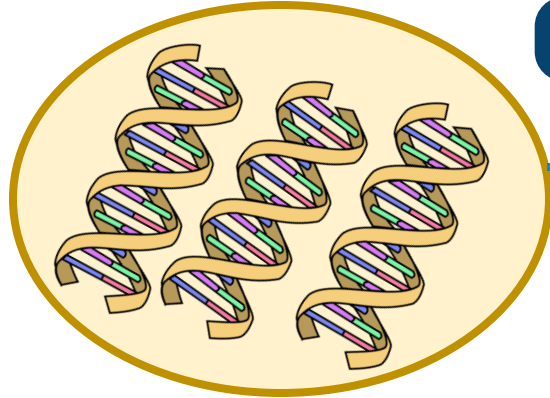
Lightweight in-storage accelerators

- Minimize SRAM/DRAM buffer spaces needed inside 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 1 Overview



Metagenomic sample with species that are **not known** in advance



A large database containing information on **many species**

Step 1

Preparation of Input Queries

Query
K-mers

GCTCA
CTCAT
TCATG
...

Step 2

Presence/Absence Identification



V. cholerae



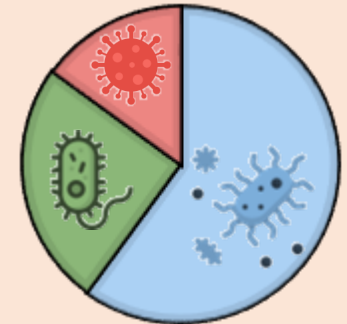
SARS-CoV-2



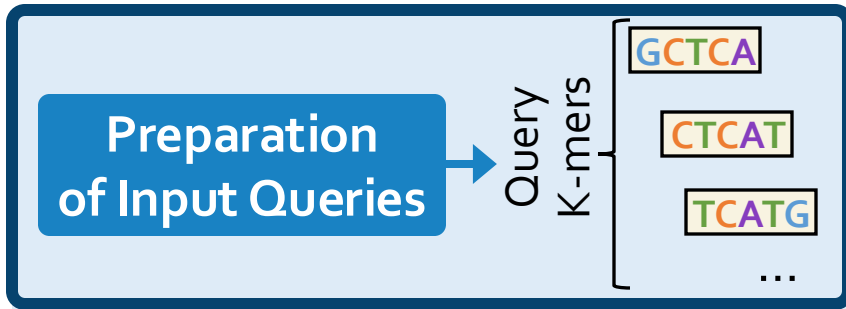
E. coli

Step 3

Abundance Estimation

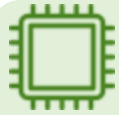


Step 1 Overview



MegIS employs 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

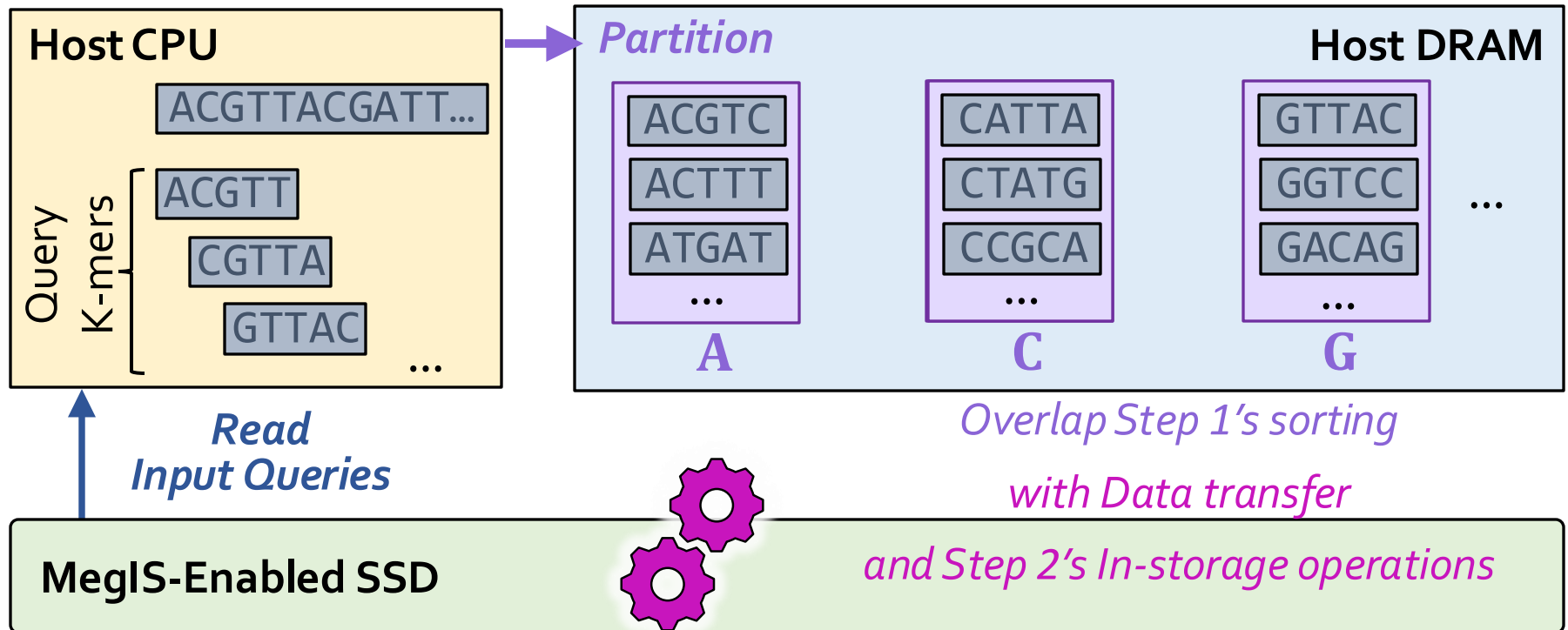
To execute Step 1 efficiently in the host system, MegIS needs to:

- 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

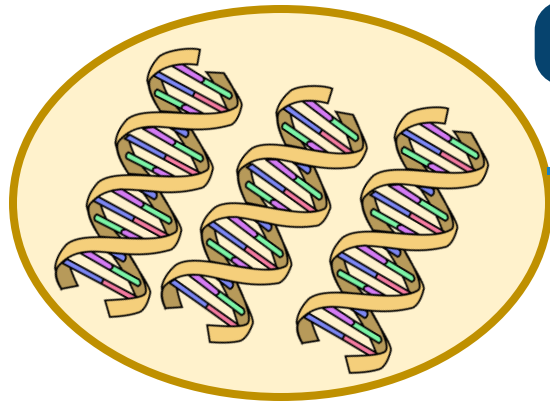
Step 1 Design

Divide k-mers into **independent partitions** by their alphabetical range

✓ Can overlap operations on different partitions



Step 2 Overview



Metagenomic sample with species that are **not known** in advance



A large database containing information on **many species**

Step 1

Preparation of Input Queries

Query K-mers

GCTCA
CTCAT
TCATG
...

Step 2

Presence/Absence Identification



V. cholerae



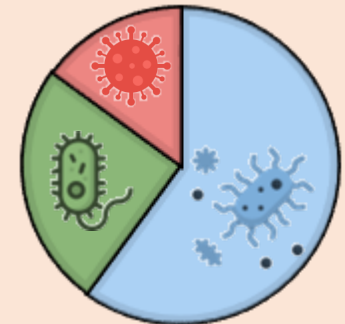
SARS-CoV-2



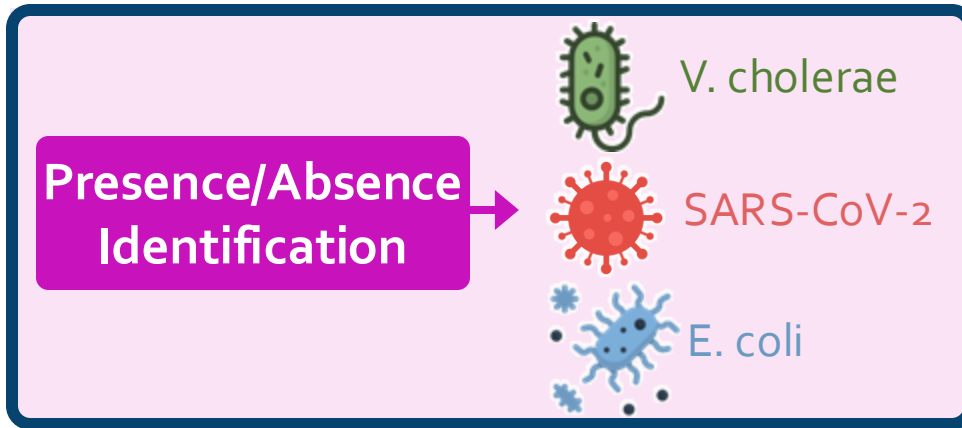
E. coli

Step 3

Abundance Estimation



Step 2 Overview



- **Identify the common k-mers** between the query k-mers and the database k-mers
- **Retrieve the species IDs** of the common k-mers



MegIS executes Step 2 in the SSD

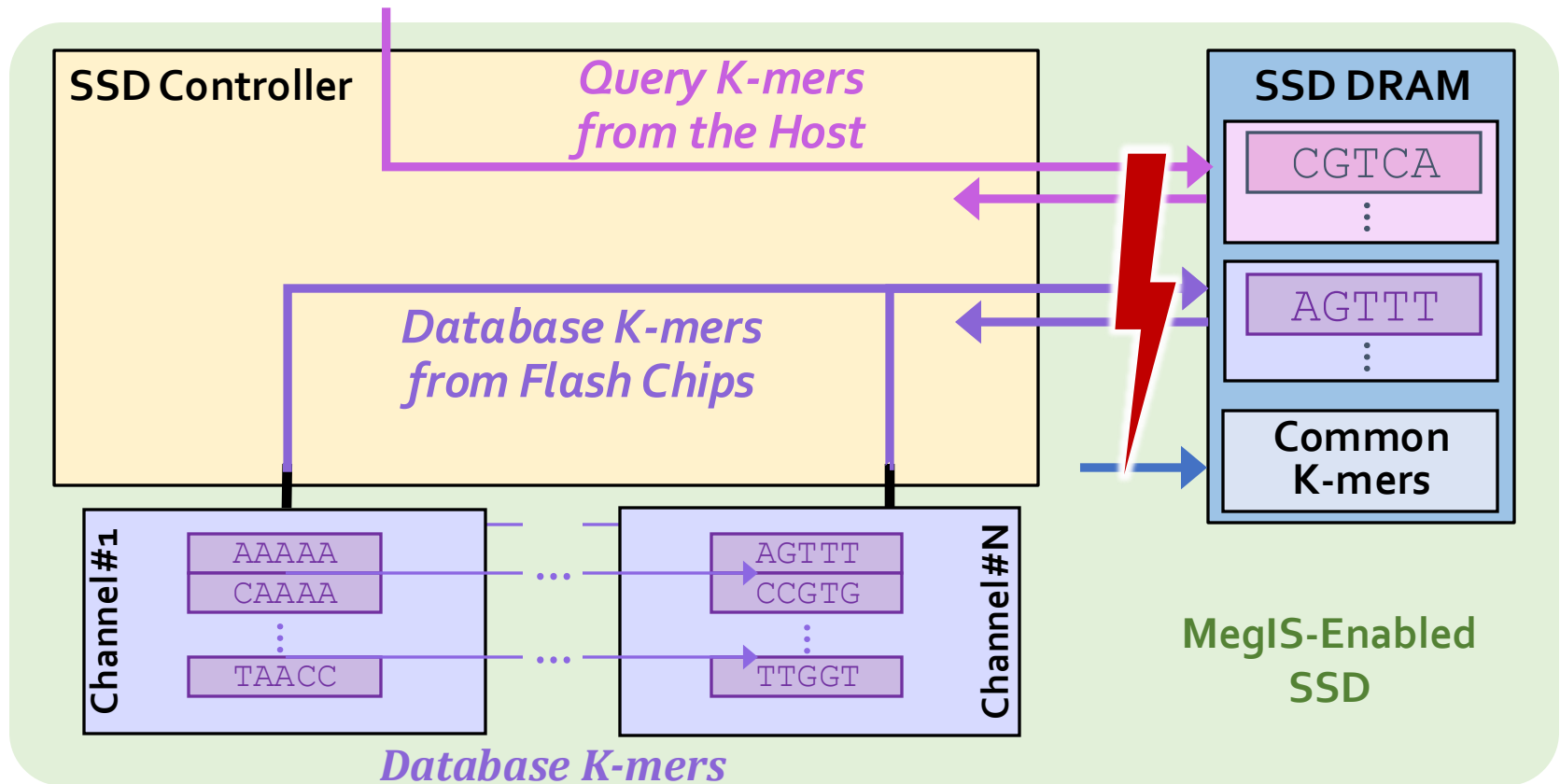
- Accesses **large data** with **low reuse**
- Involves **lightweight computation**

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)

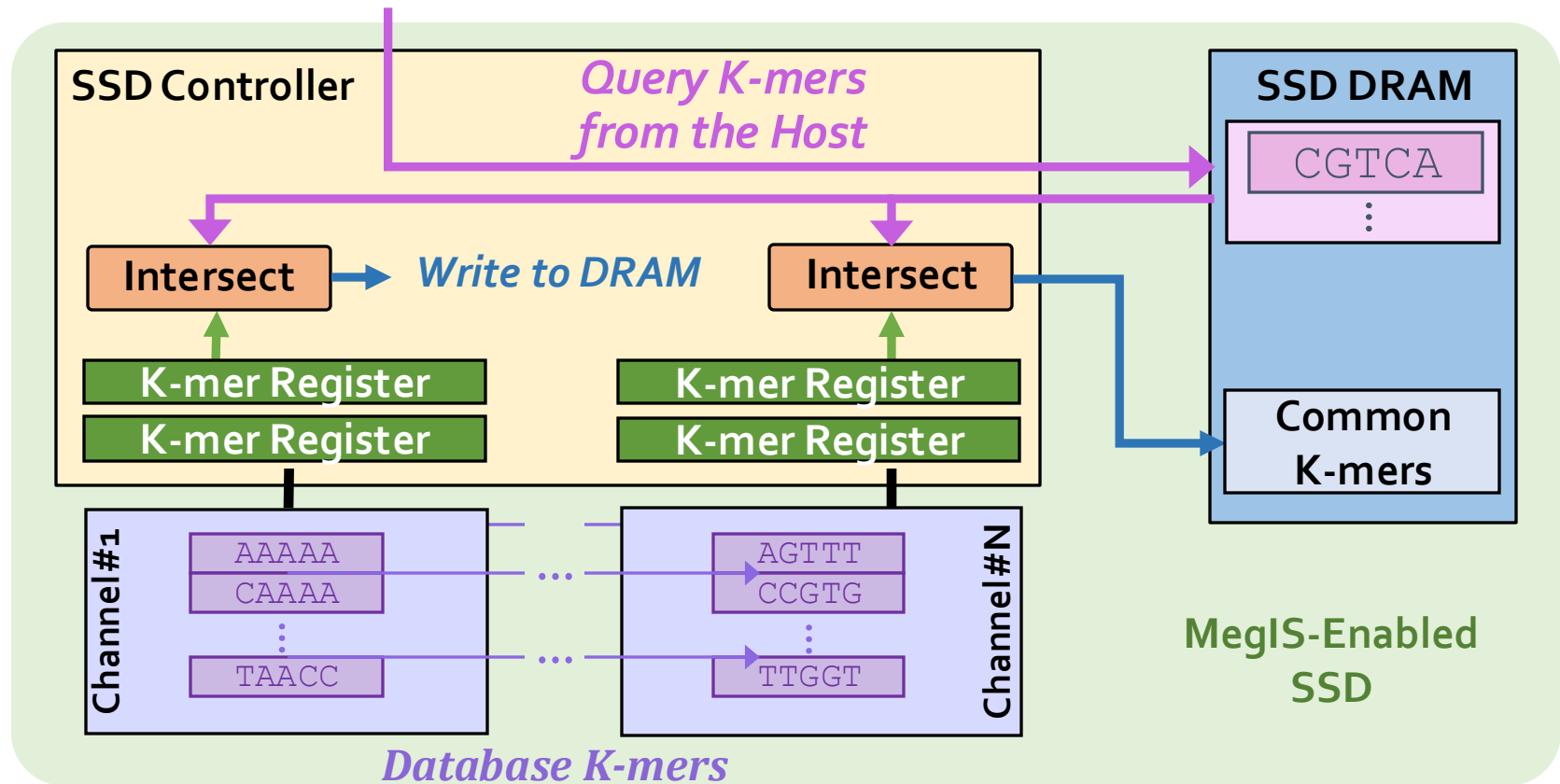
Step 2 Design: Identifying the Common K-mers

- **Challenge:** Limited internal DRAM bandwidth



Step 2 Design: Identifying the Common K-mers

- **Challenge:** Limited internal DRAM bandwidth
- ✓ *Compute directly on the flash data streams [Zou+, MICRO'22]*
- ✓ *Reduce buffer size based on application features*

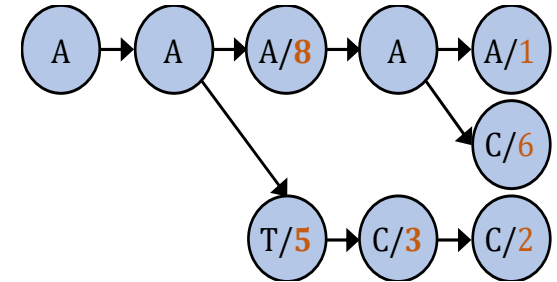


Step 2 Design: Retrieving the Species ID

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

K-mer	
AAAAA	
AAAAC	
AATCC	
...	

Space-Inefficient



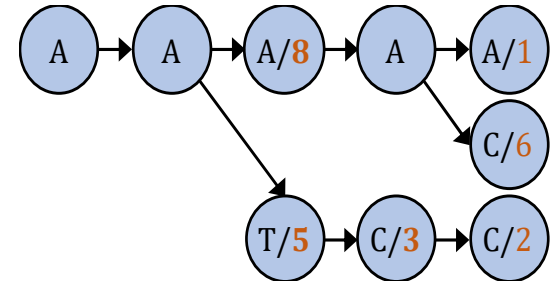
Space-Efficient

✗ *Slow inside the SSD
due to long
NAND flash latency*

Step 2 Design: Retrieving the Species ID

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

K-mer	ID
AAAAA	1,5
AAAAC	6
AATCC	2,9
...	...



Space-Inefficient

Space-Efficient

7.5x Smaller

2.1x Larger

K-mer Sketch Streaming

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

Step 2 Design: Retrieving the Species ID

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

K-mer	ID
AAAAA	15



Design details are in the paper

Space-inefficient

Space-Efficient

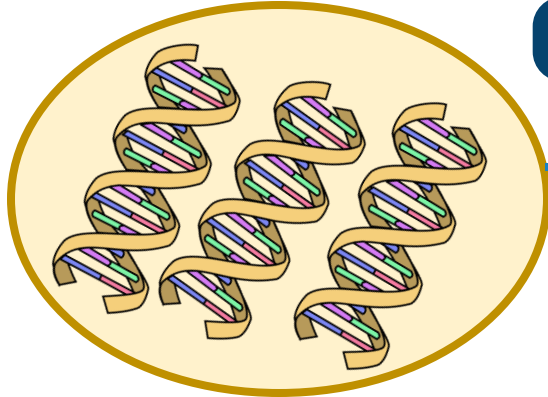
7.5x Smaller

2.1x Larger

K-mer Sketch Streaming

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

Step 3



Metagenomic sample with species that are **not known** in advance



A large database containing information on **many species**

Step 1

Preparation of Input Queries

Query K-mers

GCTCA
CTCAT
TCATG
...

Step 2

Presence/Absence Identification



V. cholerae



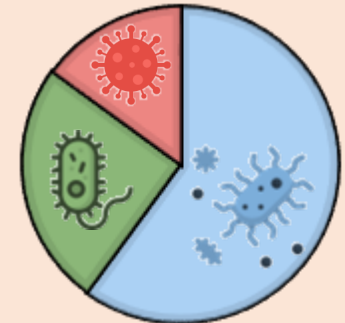
SARS-CoV-2



E. coli

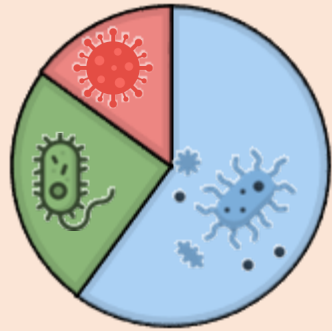
Step 3

Abundance Estimation



Step 3

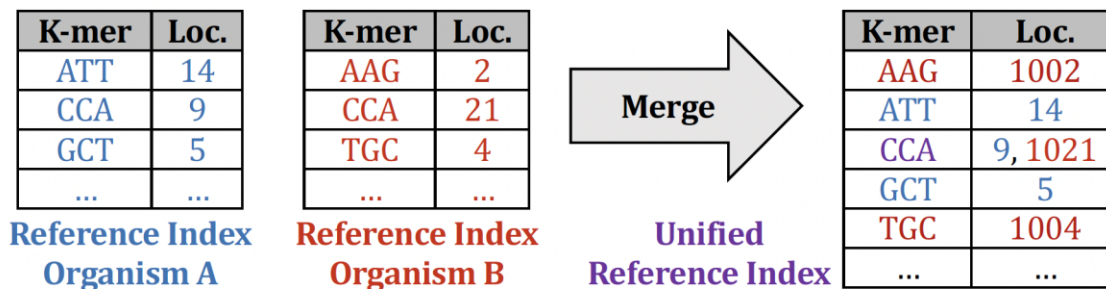
Abundance Estimation



MegIS performs additional analysis on species identified in the sample to estimate their abundance

MegIS can flexibly integrate with different approaches

1. **Lightweight statistical approaches:** Directly uses the output of Step 2
2. **More accurate and costly read mapping:** MegIS facilitates integration by preparing mapping indexes in the SSD



Step 3 and MegIS FTL are in the paper

Outline

Background

Motivation and Goal

MegIS

Evaluation

Conclusion

Evaluation Methodology Overview (I)

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)

Evaluation Methodology Overview (II)

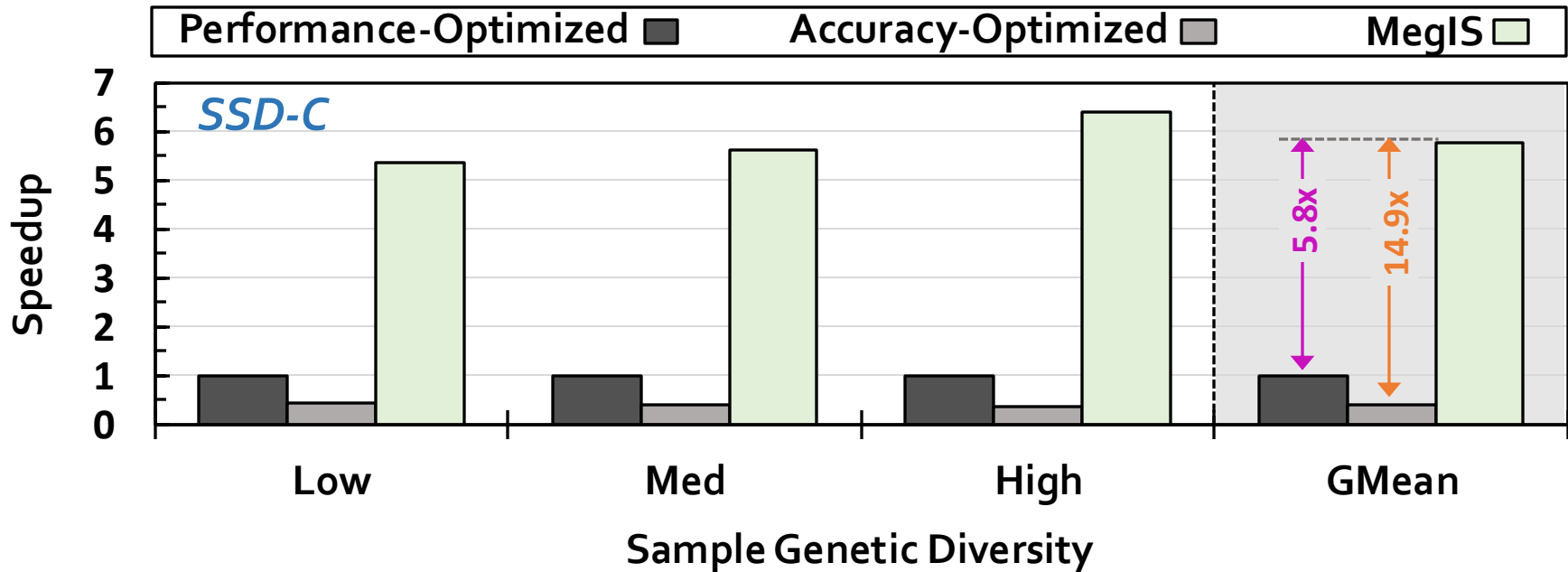
Metagenomic Analysis Task

- Finding species present in the sample
- Analysis of the abundance estimation task is in the paper

Metagenomic Samples

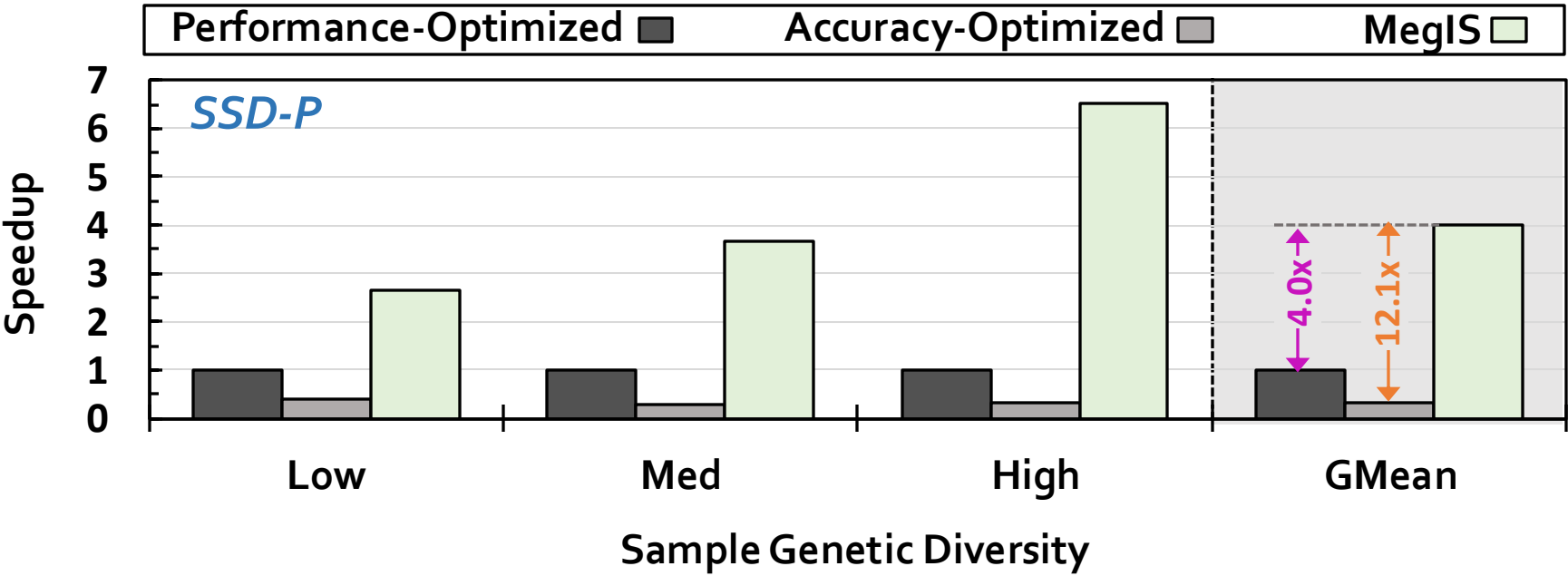
- With varying degrees of genetic diversity
 - Low
 - Medium
 - High

Speedup over Software (with Cost-Optimized SSD)



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

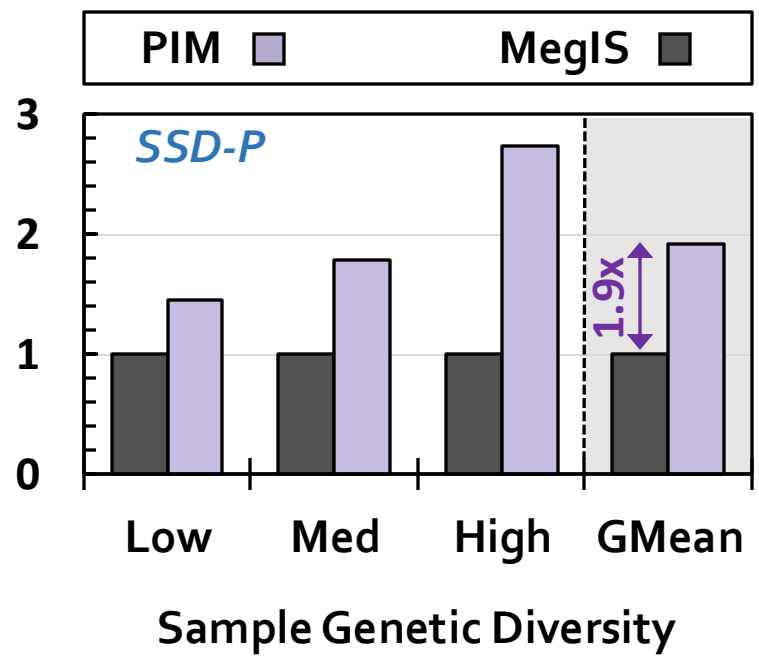
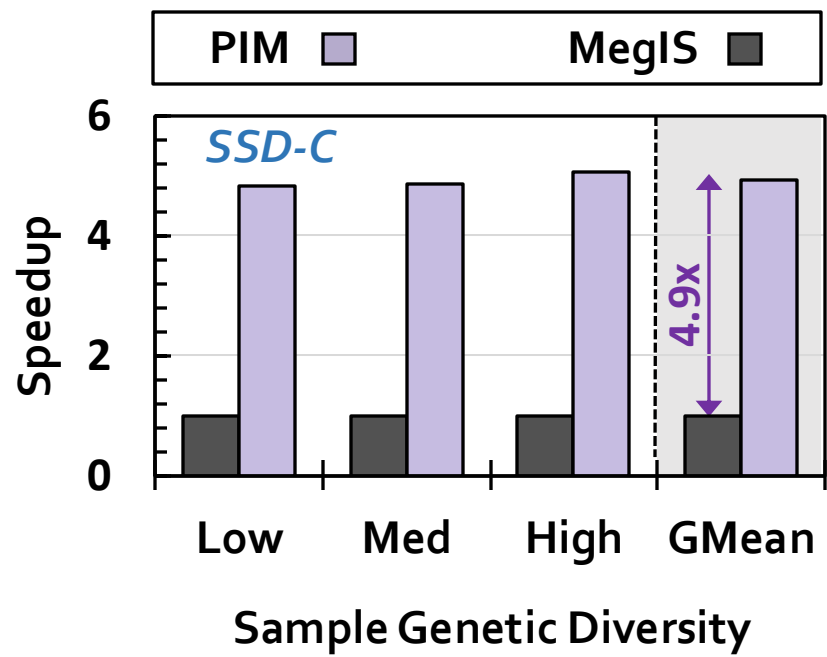
Speedup over Software (with Performance-Optimized SSD)



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

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

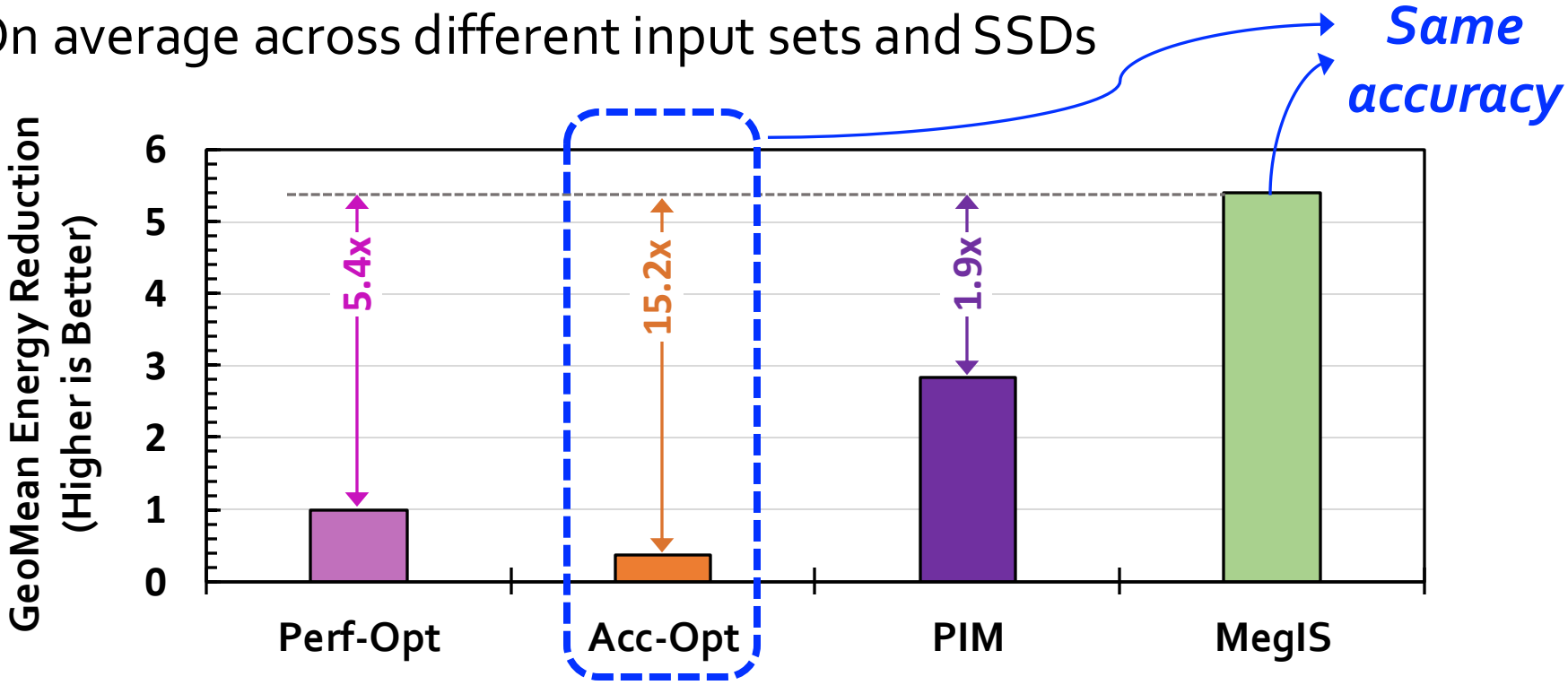
Speedup over the PIM Hardware Baseline



MegIS provides significant speedup over the PIM baseline

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

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 score
 - 3 – 24% lower L1 norm error

Area and Power

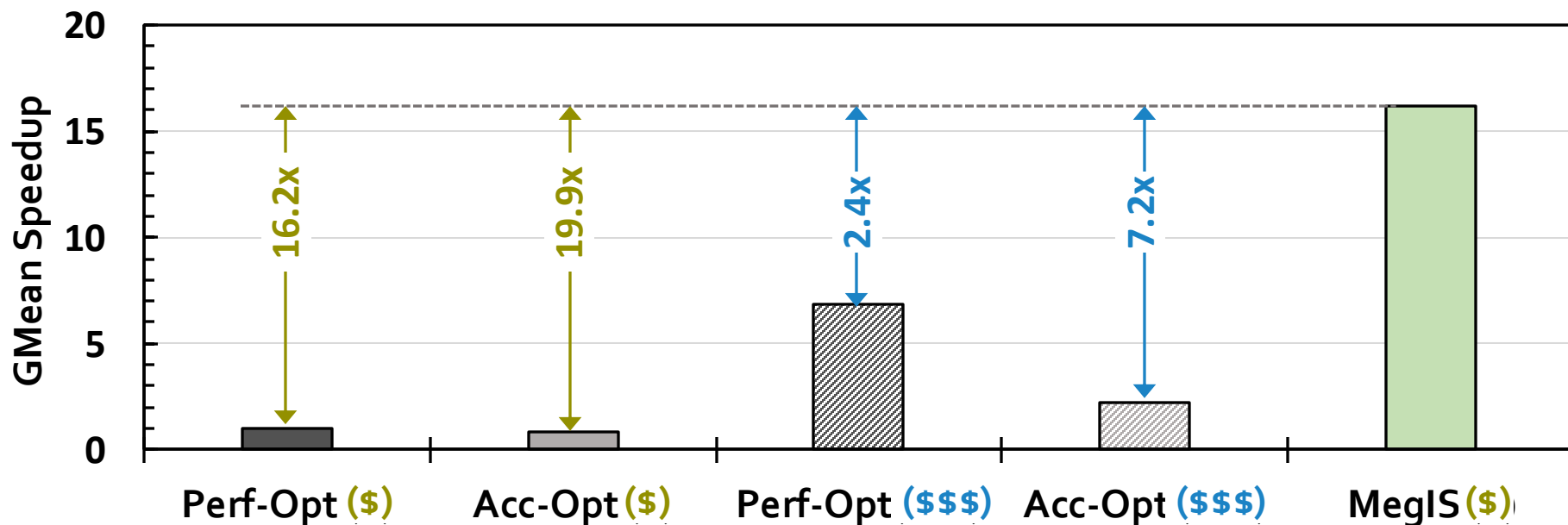
Total for an 8-channel SSD:

- **Area:** 0.04 mm²
- **Power:** 7.658 mW

*(Only **1.7%** of the area and **4.6%** of the power consumption of three ARM Cortex R4 cores in an SSD controller)*

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

System Cost-Efficiency

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

20

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

Perf-Opt (\$)

Acc-Opt (\$)

Perf-Opt (\$\$\$)

Acc-Opt (\$\$\$)

MegIS (\$)

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

More in the Paper

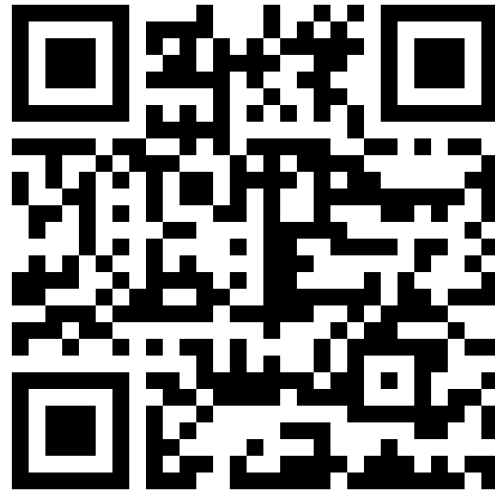
- MegIS's performance when running in-storage processing operations on the **cores existing in the SSD controller**
- MegIS's performance when using the same accelerators **outside SSD**
- **Sensitivity analysis with varying**
 - Database sizes
 - Memory capacities
 - #SSDs
 - #Channels
 - #Samples
- 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

<https://arxiv.org/abs/2406.19113>

Outline

Background

Motivation and Goal

MegIS

Evaluation

Conclusion

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
4.8× higher F1 score
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

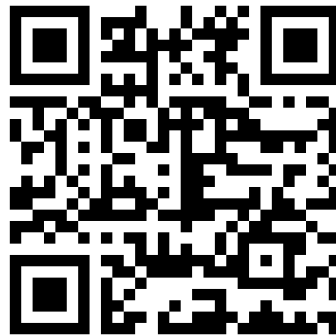


Small area/power

Area: 0.04 mm²
Power: 7.658 mW

MegIS

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



<https://arxiv.org/abs/2406.19113>

SAFARI

ETH zürich

POSTECH

In-Storage Metagenomics [ISCA 2024]

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

"MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing"

Proceedings of the 51st Annual International Symposium on Computer Architecture (ISCA), Buenos Aires, Argentina, July 2024.

[[Slides \(pptx\)](#)] [[pdf](#)]

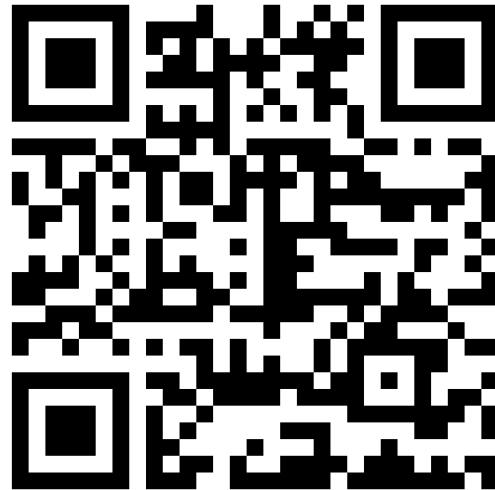
[[arXiv version](#)]

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

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



<https://arxiv.org/abs/2406.19113>

Storage-Centric Computing for Genomics and Metagenomics

Onur Mutlu

omutlu@gmail.com

<https://people.inf.ethz.ch/omutlu>

6 August 2024

FMS: the Future of Memory and Storage

SAFARI

Backup Slides

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

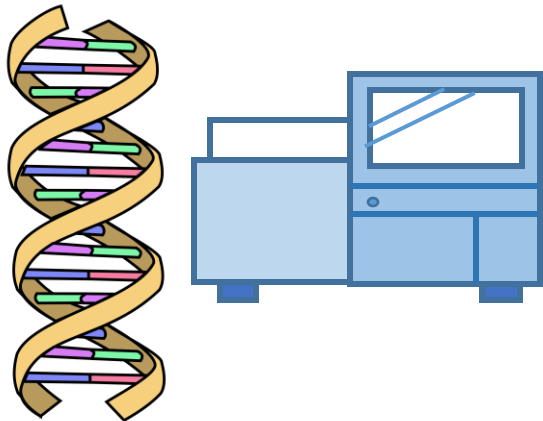
bionano
GENOMICS



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

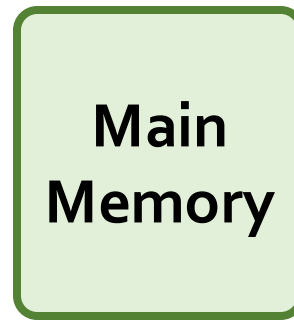
- **Read mapping:** first key step in genome sequence analysis
 - Aligns reads to potential matching locations in the reference genome
 - For each matching location, the alignment step finds the degree of similarity (alignment score)



- 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

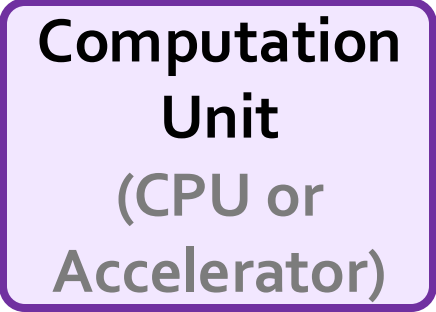
Genome Sequence Analysis

Data Movement from Storage



Alignment

Computation Unit
(CPU or Accelerator)

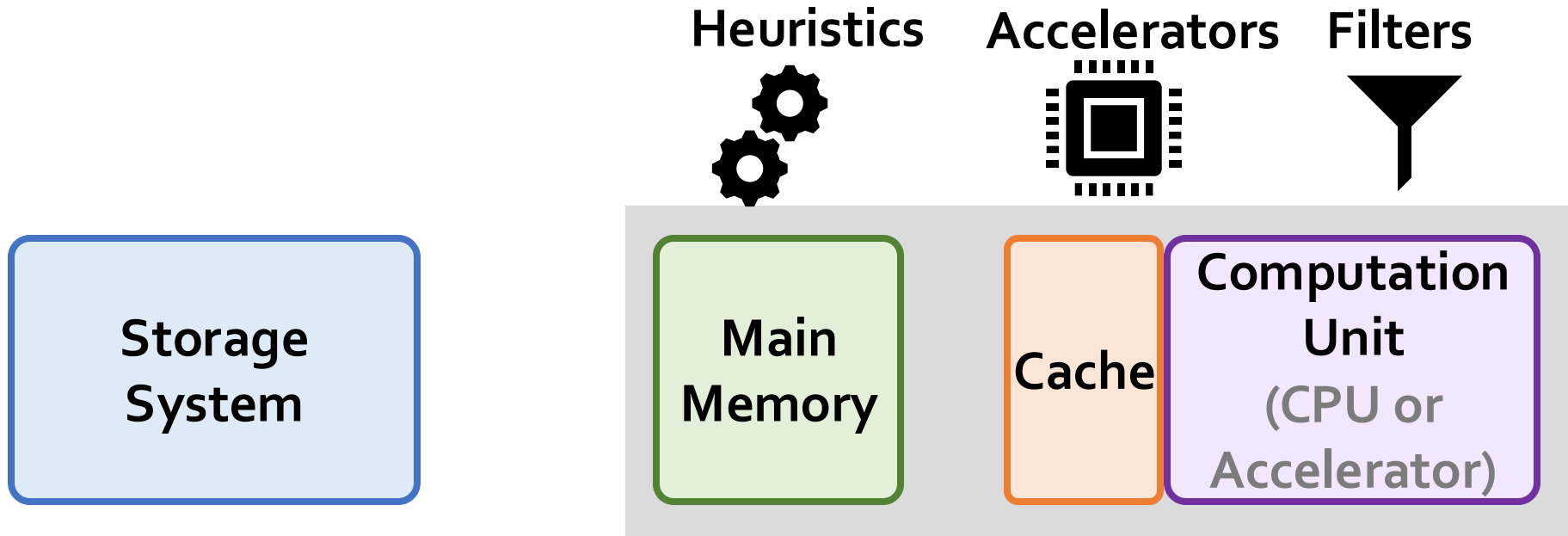


Computation overhead



Data movement overhead

Accelerating Genome Sequence Analysis



Computation overhead

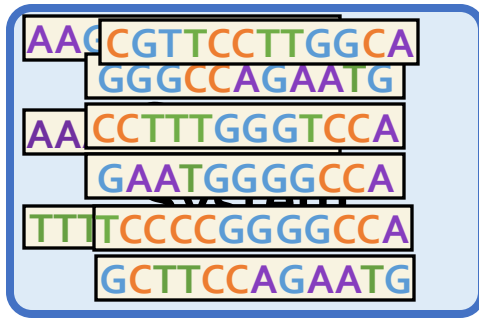


Data movement overhead

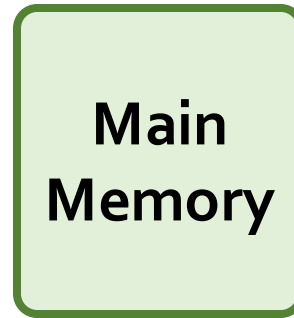
Key Idea



Filter reads that do not require alignment inside the storage system



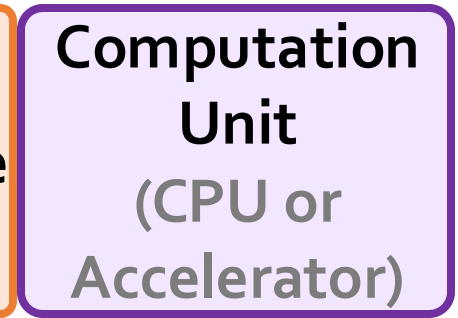
Filtered Reads



**Main
Memory**



Cache



**Computation
Unit
(CPU or
Accelerator)**

Exactly-matching reads

Do not need expensive approximate string matching during alignment

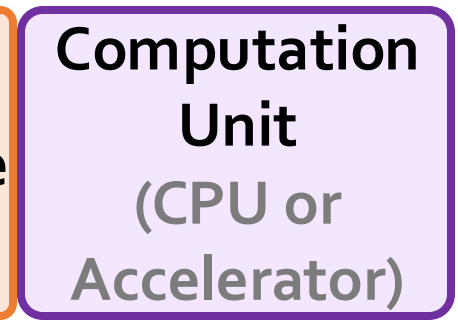
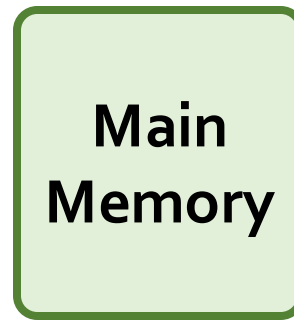
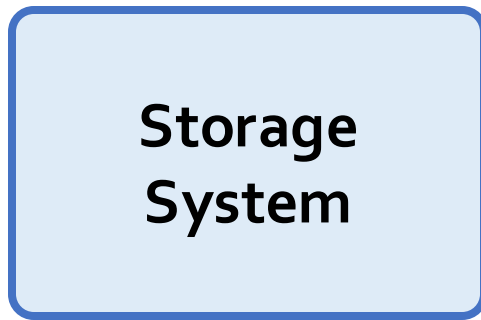
Non-matching reads

Do not have potential matching locations and can skip alignment

Challenges



Filter reads that do not require alignment inside the storage system



Filtered Reads

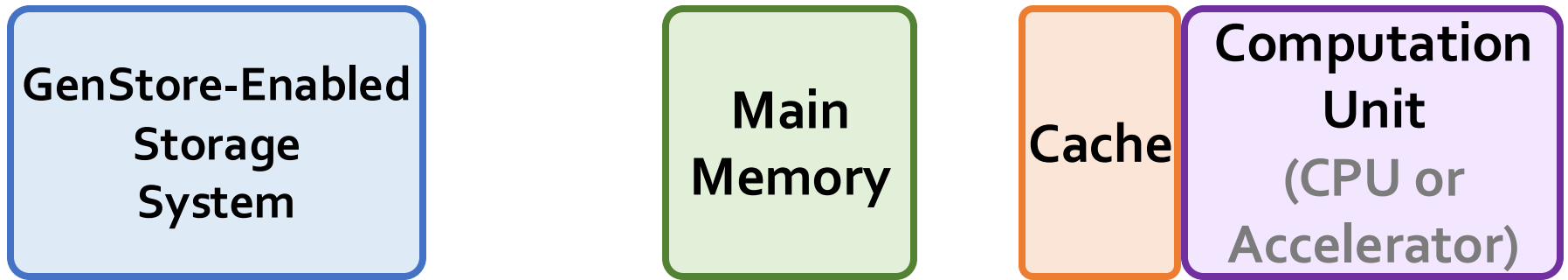
Read mapping workloads can exhibit different behavior

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

GenStore



Filter reads that do not require alignment inside the 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

Outline

Background

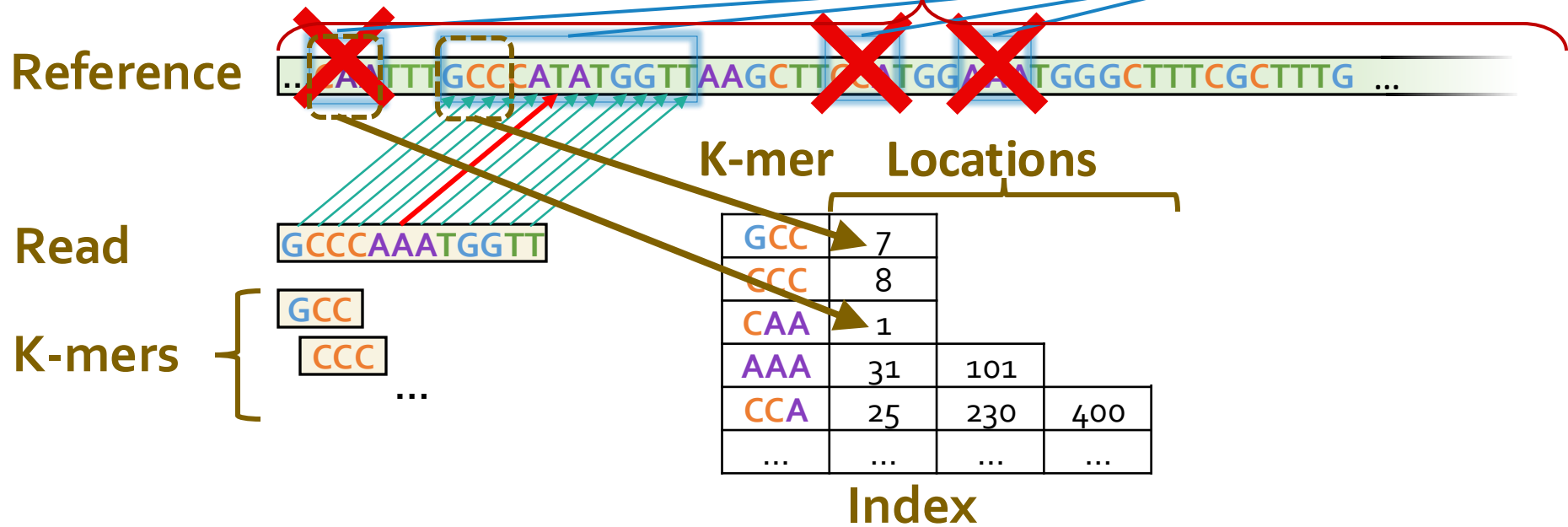
Motivation and Goal

GenStore

Evaluation

Conclusions

Read Mapping Process > 3 billion character Seeds



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

Conclusions

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**

Outline

Background

Motivation and Goal

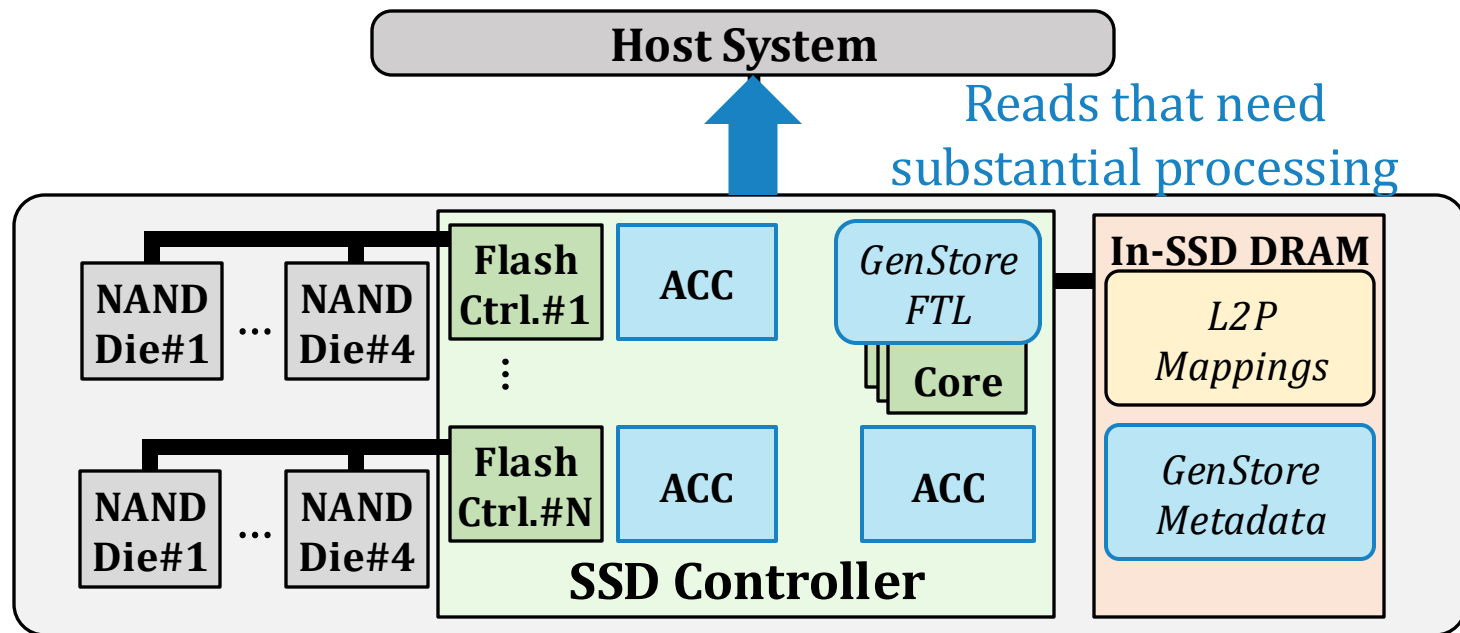
GenStore

Evaluation

Conclusions

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

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



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



GenStore-EM: Data Structures

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...



Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAG	
AAAAAAAAAAT	
...	

Read-sized
K-mers

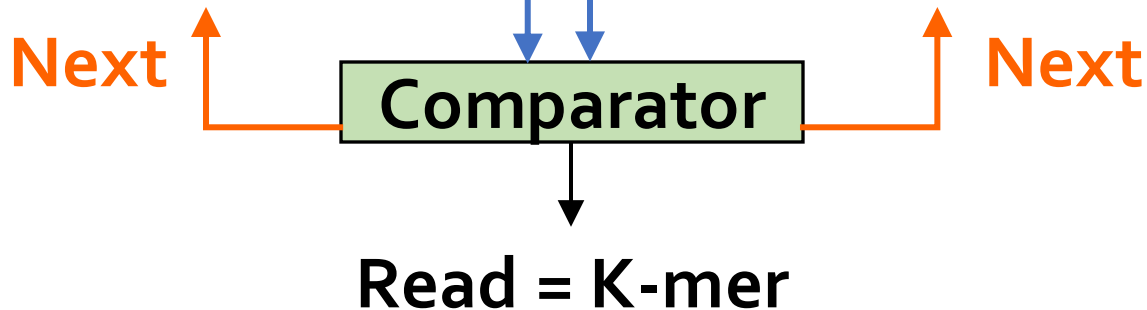
GenStore-EM: Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
...	



Exact match → Filter the read

GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
...	

Comparator

Next

Read > K-mer

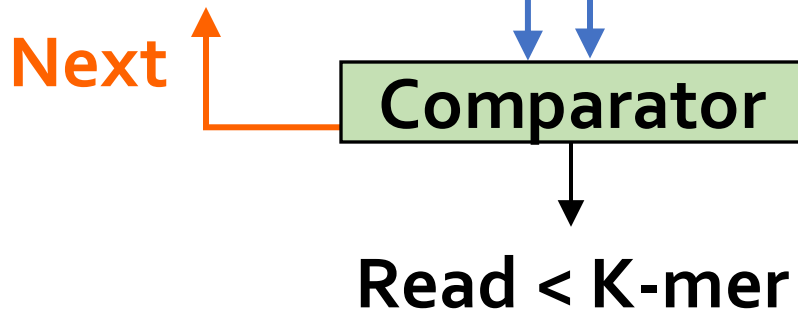
GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	
AAAAAAAAAAAC	
AAAAAAAAAAAT	
...	



Not an exact match → Send to read mapper

GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAAAA

Sorted K-mer Index

K-mer	
AAAAAAAAAAAA	



Avoids random accesses



Simple low-cost logic

Comparator



Read < K-mer

Not an exact match → Send to read mapper

GenStore-EM: Optimization

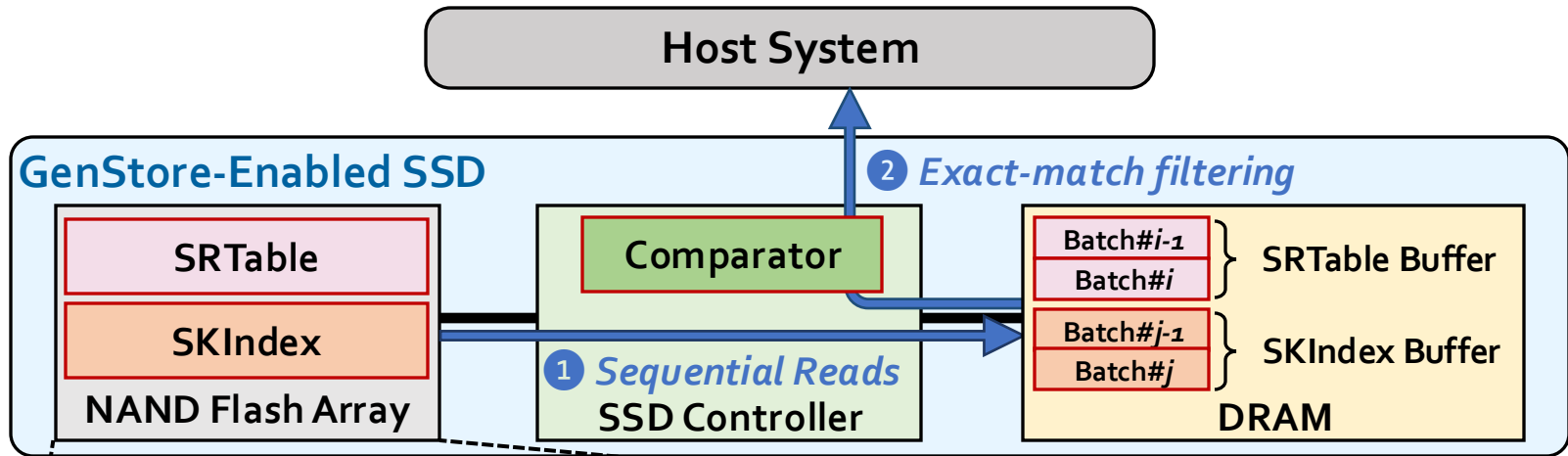
- Read-sized k-mer index takes up a **large amount of space** (126 GB for human index) due to the larger number of unique k-mers

Sorted K-mer Index

Strong Hash Value	Loc.
1	1, 8, ...
4	51
7	23, 37
16	...

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

GenStore-EM: Design



Steps 1 and 2 are **pipelined**.
During filtering, GenStore-EM sends the unfiltered reads to the host system.

Data is evenly distributed between channels, dies, and planes to **leverage the full internal bandwidth** of the SSD

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

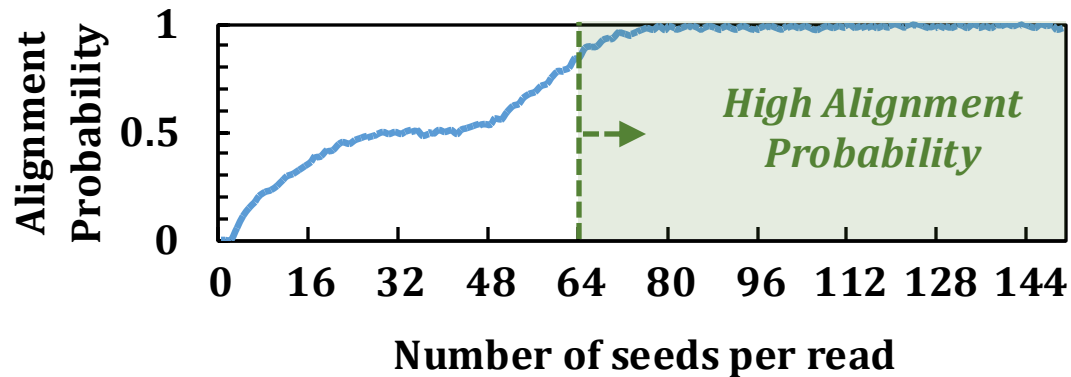
Seeding	Determine potential matching locations (seeds) in the reference genome
Seed Filtering (e.g., Chaining)	Prune some seeds in the reference genome
Alignment	Determine the exact differences between the read and the reference genome

- **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

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**



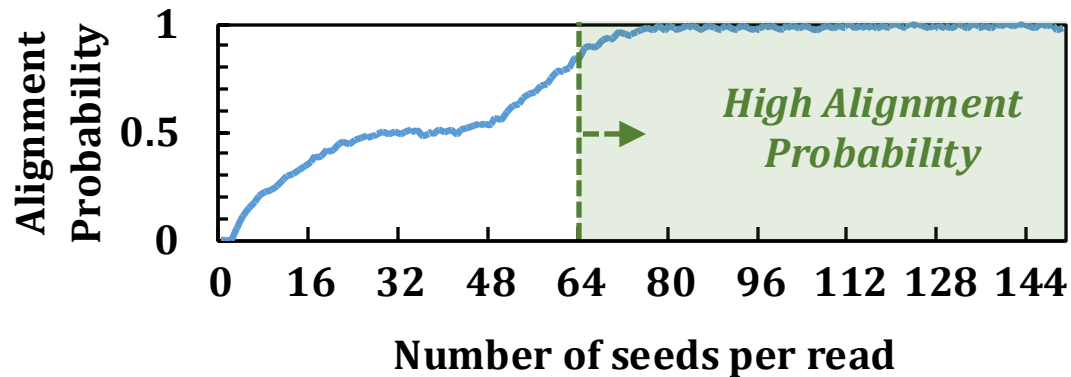
Reads with a sufficiently large number of seeds are very **likely to align** to the reference genome



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

Outline

Background

Motivation and Goal

GenStore

Evaluation

Conclusions

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

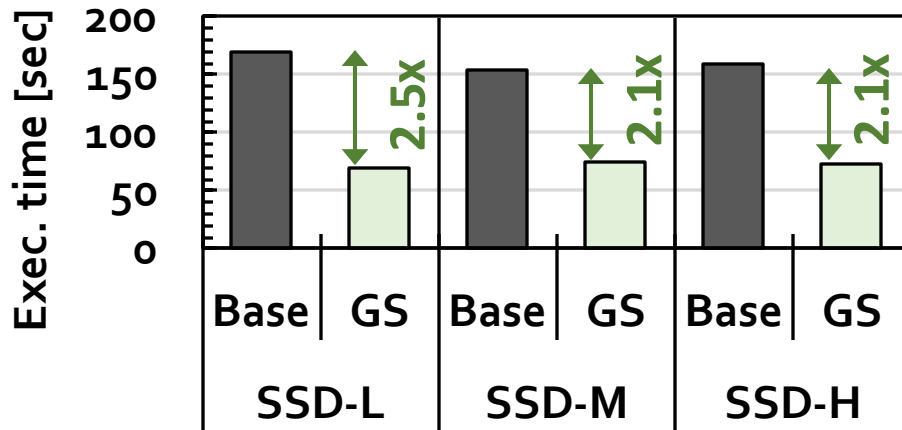
SSD Configurations

- **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)

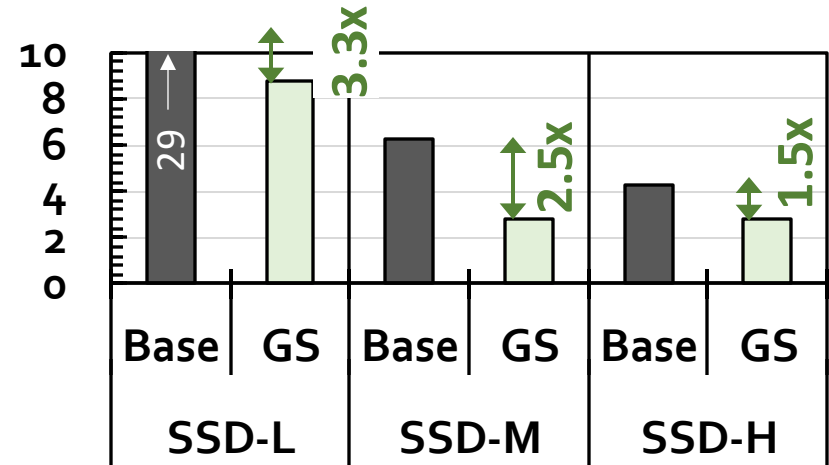
Performance – GenStore-EM

For a read set with 80% exactly-matching reads

With the Software Mapper



With the Hardware Mapper



2.1x - 2.5x speedup compared to the software Base

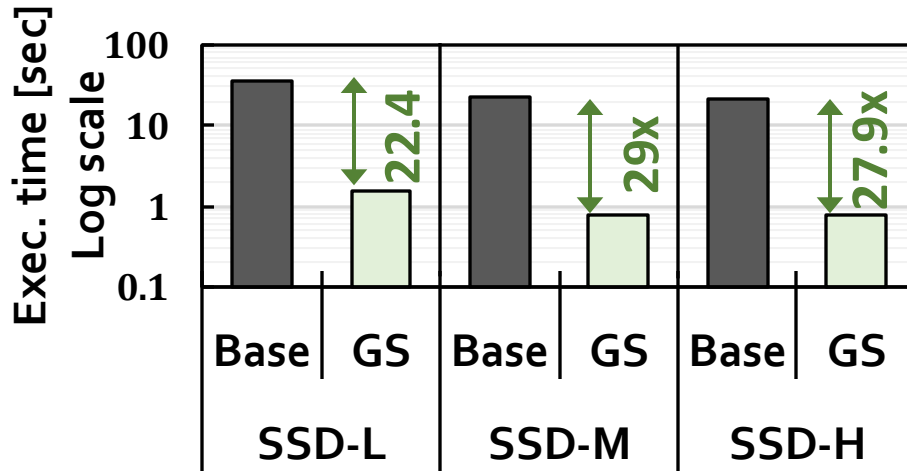
1.5x – 3.3x speedup compared to the hardware Base

On average 3.92x energy reduction

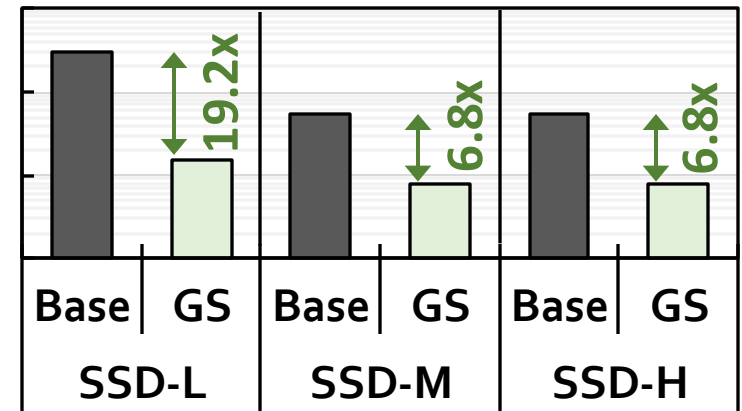
Performance – GenStore-NM

For a read set with 99.7% non-matching reads

With the Software Mapper



With the Hardware Mapper



22.4x – 27.9x speedup compared to the software Base

6.8x – 19.2x speedup compared to the hardware Base

On average 27.2x energy reduction

Area and Power

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

Logic unit	# of instances	Area [mm ²]	Power [mW]
Comparator	1 per SSD	0.0007	0.14
K -mer Window	2 per channel	0.0018	0.27
Hash Accelerator	2 per SSD	0.008	1.8
Location Buffer	1 per channel	0.00725	0.37375
Chaining Buffer	1 per channel	0.008	0.95
Chaining PE	1 per channel	0.004	0.98
Control	1 per SSD	0.0002	0.11
<i>Total for an 8-channel SSD</i>	-	0.2	26.6

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

More 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**

More in the Paper

GenStore: A High-Performance and Energy-Efficient In-Storage Computing 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 Alserr¹
Rachata Ausavarungnirun³ Nandita Vijaykumar⁴ Mohammed Alser¹ Onur Mutlu¹

¹ETH Zürich ²Bionano Genomics ³KMUTNB ⁴University of Toronto



<https://arxiv.org/abs/2202.10400>

Outline

Background

Motivation and Goal

GenStore

Evaluation

Conclusions

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**

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



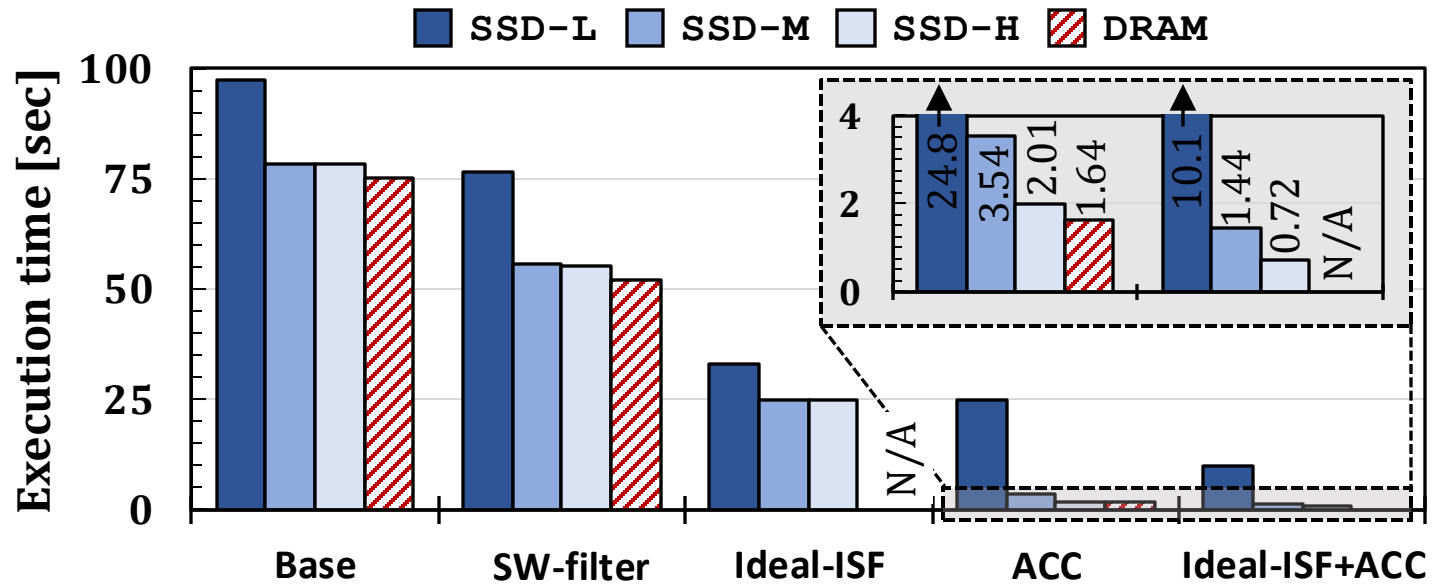
UNIVERSITY OF
TORONTO

GenStore Backup Slides

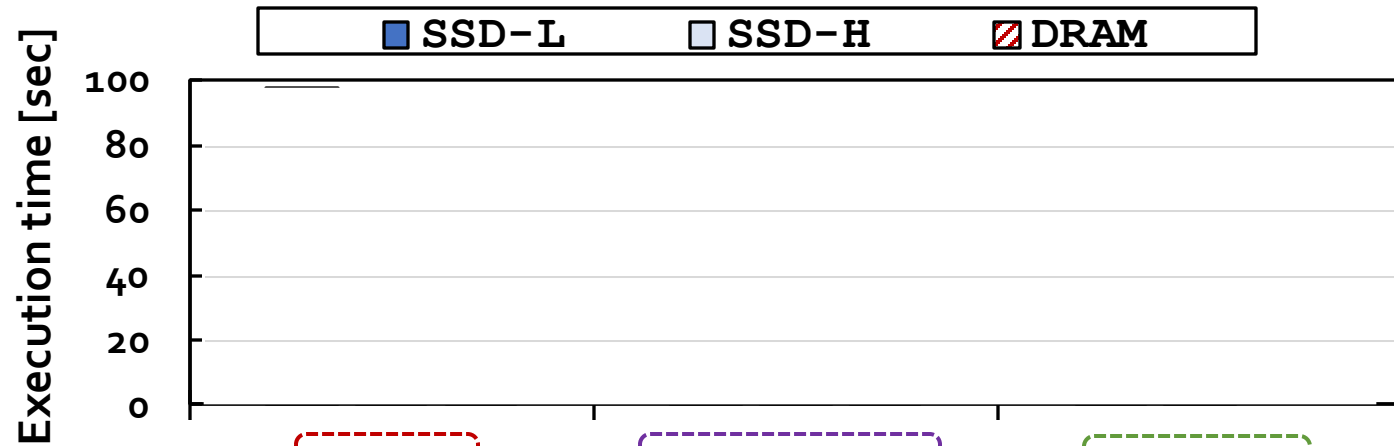
End-to-End Workflow of Genome Sequence Analysis

- There are **three key initial steps** in a standard genome sequencing and analysis workflow
 - Collection, preparation, and sequencing of a DNA sample in the laboratory
 - Basecalling
 - Read mapping
- Genomic read sets can be obtained by
 - Sequencing a DNA sample and **storing the generated read set into the SSD of a sequencing machine**
 - Downloading read sets from **publicly available repositories** and storing them into an SSD
- We focus on optimizing the performance of read mapping because sequencing and basecalling are performed only once per read set, whereas read mapping can be performed many times
 - Analyzing the differences between a reads from an individual and **many reference genomes of other individuals**
 - Repeating the read mapping step many times **to improve the outcome of read mapping**
- Improving read mapping performance is critical in almost all genomic analyses that use sequencing
 - 45% of the execution time when discovering **sequence variants in cancer genomics** studies
 - 60% of the execution time when profiling the species composition of **a multi-species (i.e., metagenomic) read**

Motivation



Motivation



State-of-the-art software read mapper, Minimap2

Base integrated with a software filter that prunes **80%** of exactly-matching reads

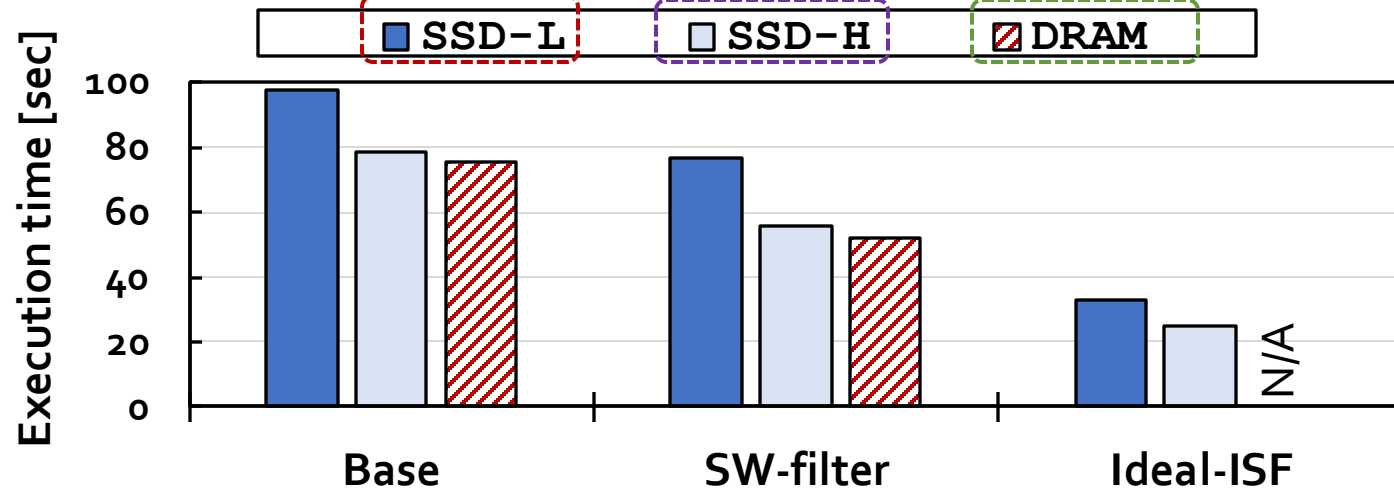
Base integrated with an ideal in-storage filter

Motivation

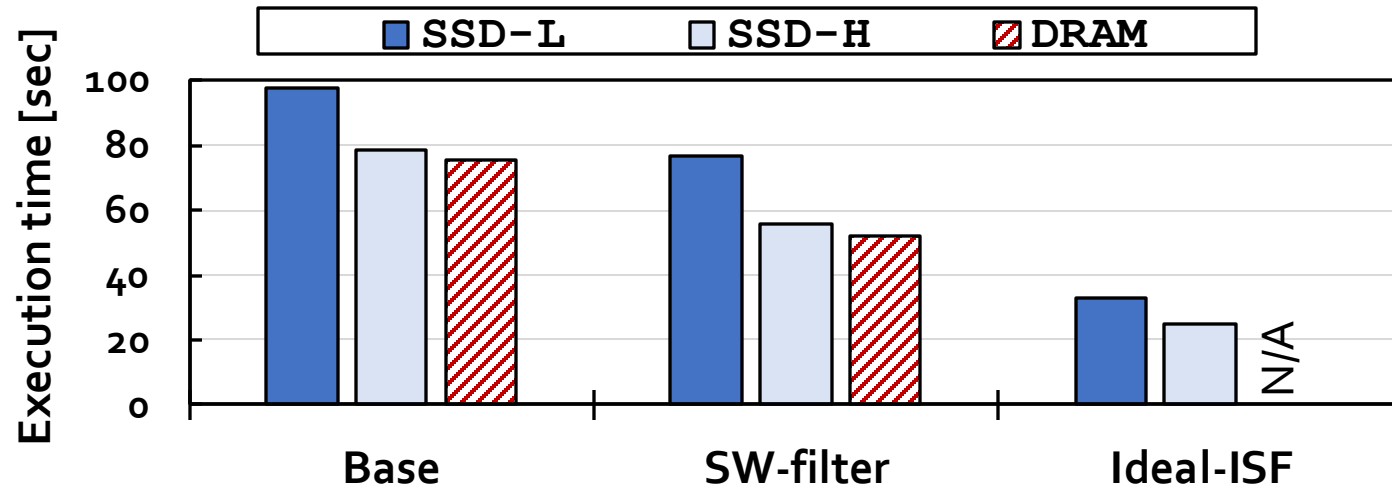
Low-end SSD with SATA₃ interface (0.5 GB/s)

High-end SSD with PCIe Gen₄ interface (7 GB/s)

Data preloaded in DRAM, with no I/O overhead



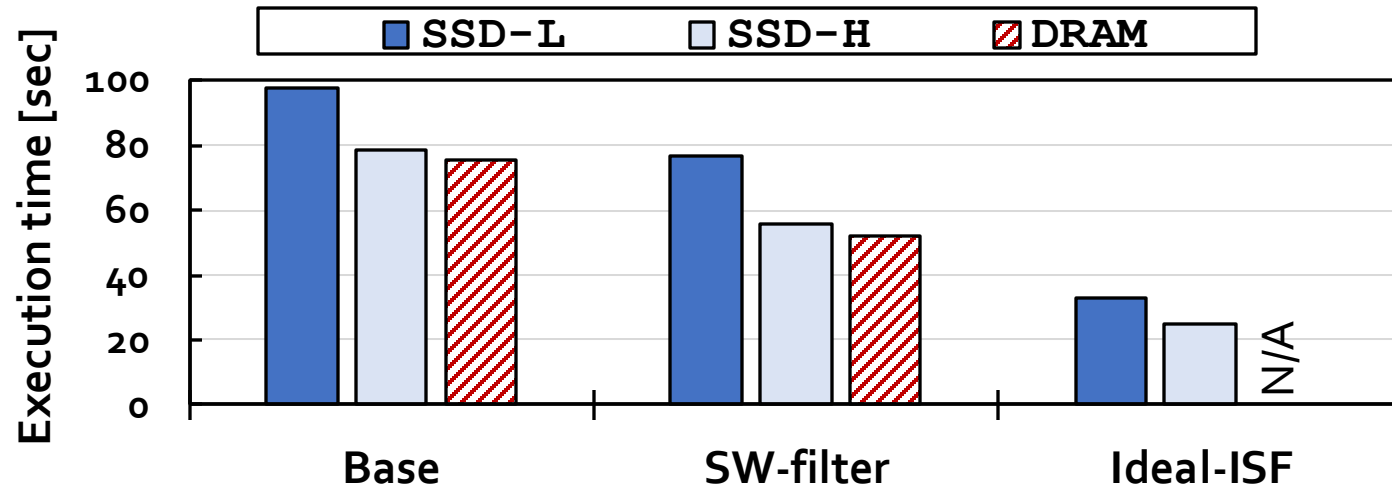
Benefits of Ideal In-Storage Filter



The ideal in-storage filter significantly improves performance by

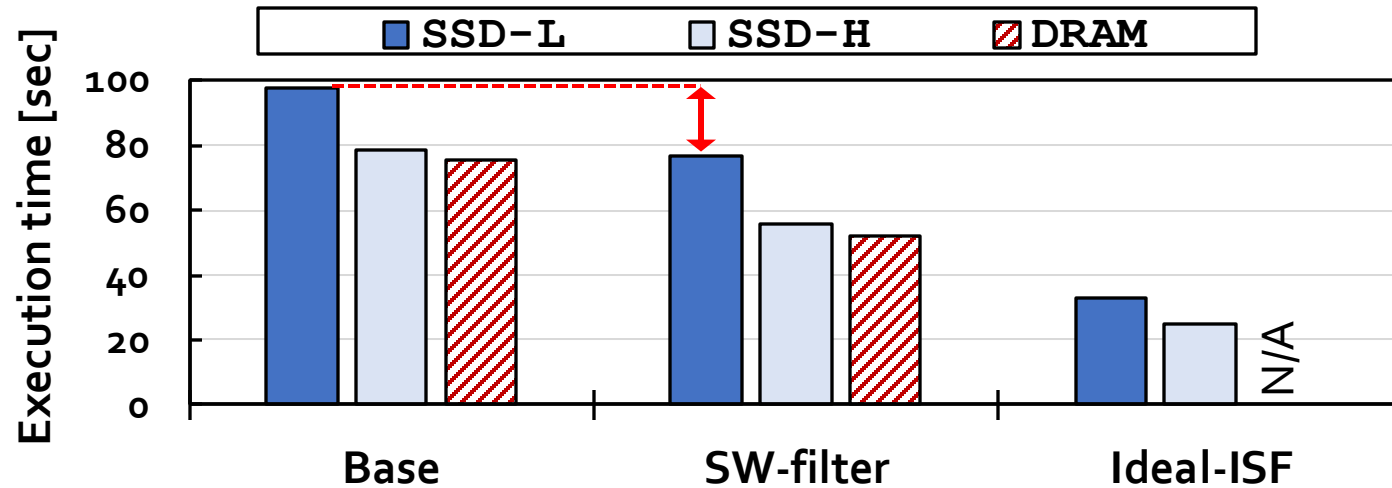
- 1) Reducing computation overhead
- 2) Reducing data movement overhead

Overheads of Software Mappers



I/O has a **significant impact** on application performance which can be alleviated at the cost of **expensive** storage devices and interfaces

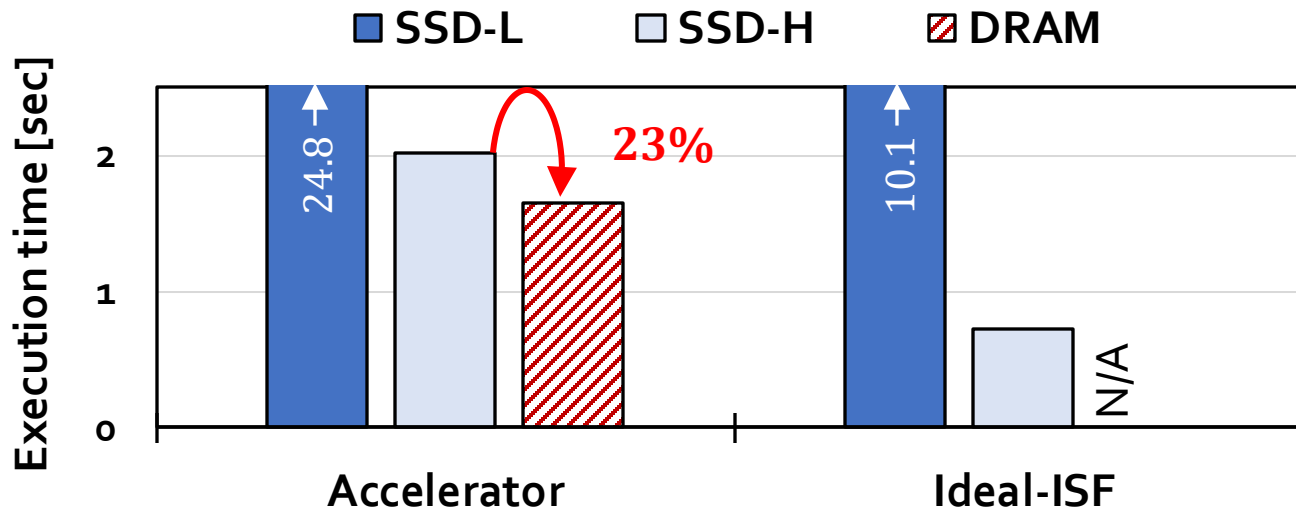
Overheads of Software Mappers



SW-filter provides limited benefits compared to Base

The filtering process **outside the SSD** must **compete** with the read mapping process for the resources in the system

Overheads of Hardware Mappers



Even the high-end SSD **does not fully alleviate** the storage bottleneck

The ideal in-storage filter significantly improves performance

Ideal-OSF

- Execution time of an **ideal in-storage filter**:

$$T_{\text{Ideal-ISF}} = T_{\text{I/O-Ref}} + \max \{ T_{\text{I/O-Unfiltered}}, T_{\text{RM-Unfiltered}} \}$$

- Execution time of an **ideal outside-storage filter**:
 - **60% slower** than Ideal-ISF in our analysis

$$T_{\text{Ideal-OSF}} = T_{\text{I/O-Ref}} + \max \{ T_{\text{I/O-All-Reads}}, T_{\text{RM-Unfiltered}} \}$$

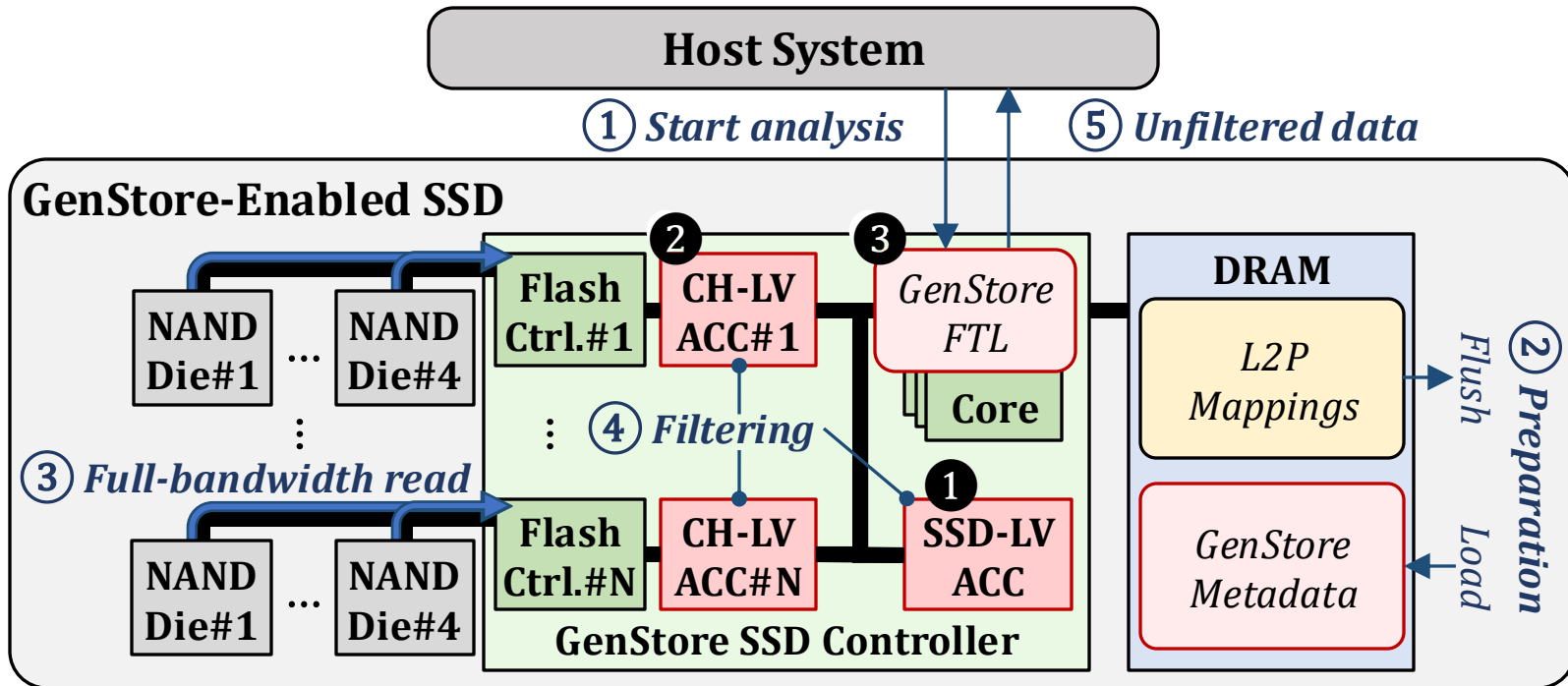
Comparison to PIM

- Even though read mapping applications could also benefit from other near-data, in-storage processing can fundamentally address the data movement problem by filtering **large, low-reuse data** where the data initially resides.
- Even if an ideal accelerator achieved a zero execution time, there would still exist the need to bring the data from storage to the accelerator.
 - **2.15x slower** than the execution time that Ideal-ISF+ACC provides in our motivational analysis

In-storage filter can be integrated with any read mapping accelerator, including PIM accelerators, to alleviate their data movement overhead.

Long Read Use Cases

Use case	Input read set (Short/Long)	Size [GB]	Reference	Align [%]
Sequencing errors	ERR3988483 (L) [157]	54	hg38 [144]	47.4
	HG002_ONT_20200204 (L) [158]	371		69.3
Rapidly evolving samples	SRR5413248 (L) [157]	1.69	NZ_NJEX02 [159]	60.0
	SRR12423642 (S) [157]	0.466	NC_045512.2 [160]	23.1
No reference	SRR6767727 (L) [157]	12.4	NZ_NJEX02 [159]	0.35
	SRR9953689 (L) [157]	15.9		37.0
Contamination	SRR9953689 (L) [157]	15.9	hg38 [144]	1.0



FTL: Metadata

- GenStore metadata includes the **mapping information** of the data structures necessary for read mapping acceleration
- In accelerator mode, GenStore also keeps in internal DRAM other metadata structures of the regular FTL
 - Examples include the **page status table and block read counts** which need to be updated during the filtering process
- We carefully design GenStore to only **sequentially access** the underlying NAND flash chips while operating as an accelerator
 - Requires **only a small amount of metadata** to access the stored data

FTL: Data Placement

- GenStore needs to properly place its data structures to enable the **full utilization of the internal SSD bandwidth**
- When each data structure is initially written to the SSD, GenStore **sequentially and evenly** distributes it across NAND flash chips
- GenStore can specify the physical location of a 30-GB data structure by maintaining only the list of 1,250 (30 GB/24 MB) physical block addresses
- It significantly reduces the size of the necessary mapping information from **300 MB** (with conventional 4-KiB page mapping) to only **5 KB** (1,250 \times 4 bytes)

FTL: SSD Management Tasks

- In accelerator mode, GenStore only reads data structures to perform filtering, and does not write any new data
 - GenStore does not require any write-related SSD-management tasks such as **garbage collection** and **wear-leveling**
- The other tasks necessary for ensuring data reliability can be done before or after the filtering process
 - GenStore significantly limits the amount of data whose **retention age** would exceed the manufacturer-specified threshold since GenStore's filtering process takes a short time.
 - GenStore-FTL can easily **avoid read disturbance errors** for data with high read counts since GenStore sequentially reads NAND flash blocks only once during filtering

Data Sizes

- Conventional k-mer index in Minimap2 + reference genome: 7 GB (k = 15)
- Read-sized k-mer index before optimization: 126 GB (k= 150)
- Read-sized k-mer index after optimization: 32 GB (k = 150)

SSD Specs

- **SSD-L:** SATA3 interface (0.5 GB/s sequential read)
 - 1.2 GB/s per channel bandwidth
 - 8 channels
- **SSD-L:** PCIe Gen3 M.2 interface (3.5 GB/s sequential read)
 - 1.2 GB/s per channel bandwidth
 - 16 channels
- **SSD-L:** PCIe Gen4 interface (7 GB/s sequential read)
 - 1.2 GB/s per channel bandwidth
 - 16 channels

Evaluation Methodology

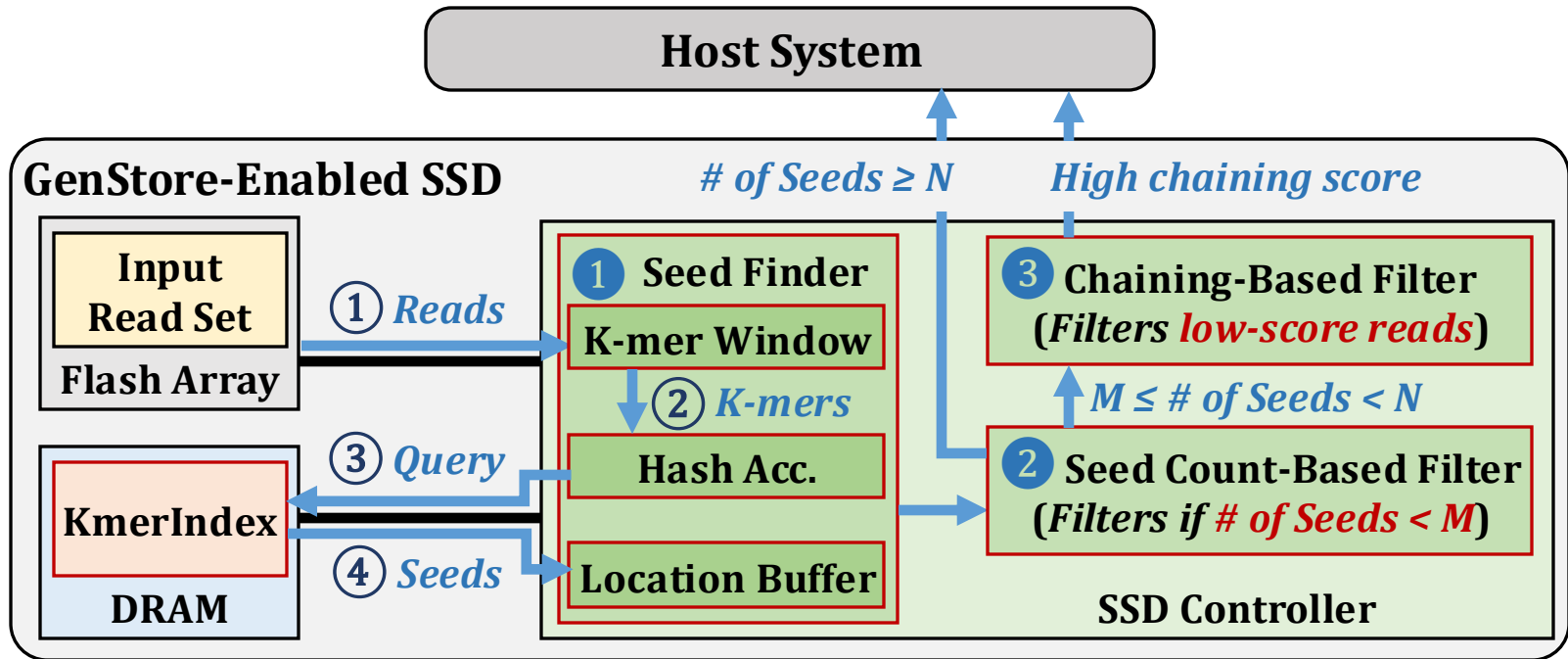
- **Performance modeling**

- Ramulator for DRAM timing
- MQSim for SSD timing
- We model the end-to-end throughput of GenStore based on the throughput of each GenStore pipeline stage
 - Accessing NAND flash chips
 - Accessing internal DRAM
 - Accelerator computation
 - Transferring unfiltered data to the host

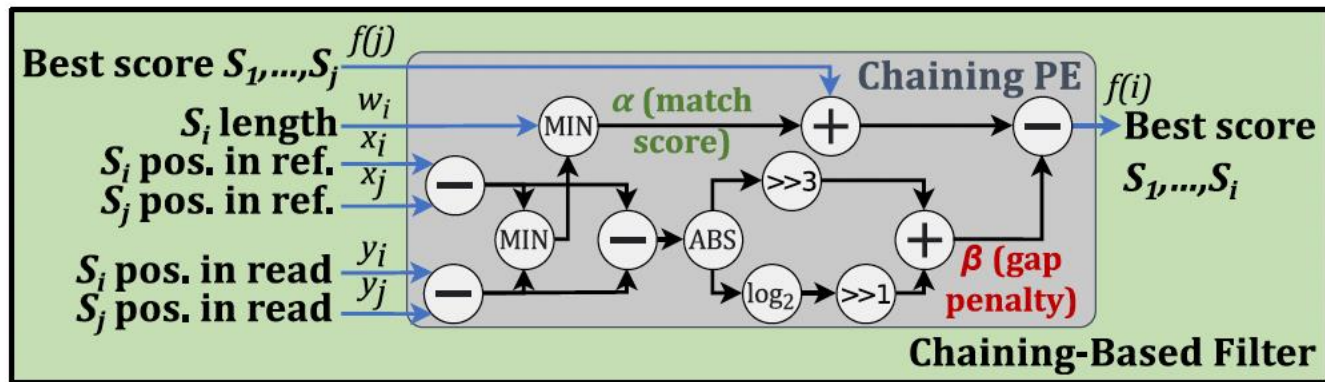
- **Real system results**

- AMD EPYC 7742 CPU
- 1TB DDR4 DRAM
- AMD μ Prof

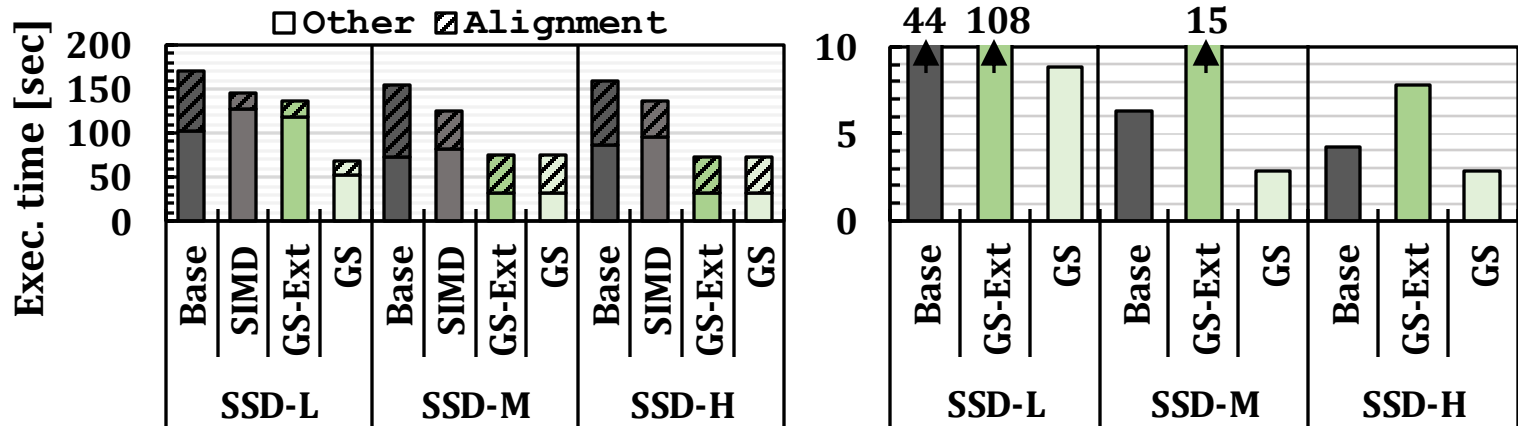
GenStore-NM



Chaining Processing Element



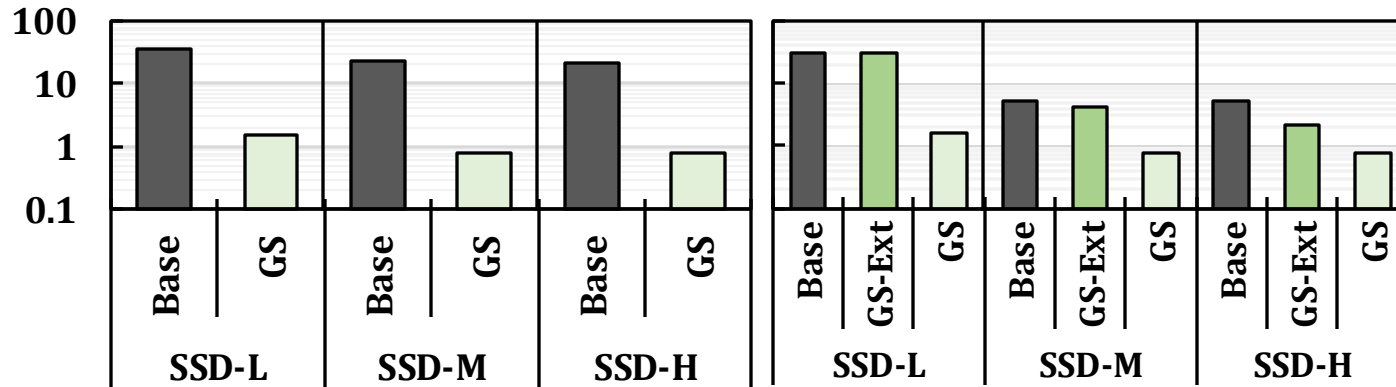
GenStore-EM



GS-Ext provides significant performance improvements over both Base and SIMD in SSD-M and SSD-H.

GS-Ext provides limited benefits over SIMD in SSD-L due to low external I/O bandwidth.

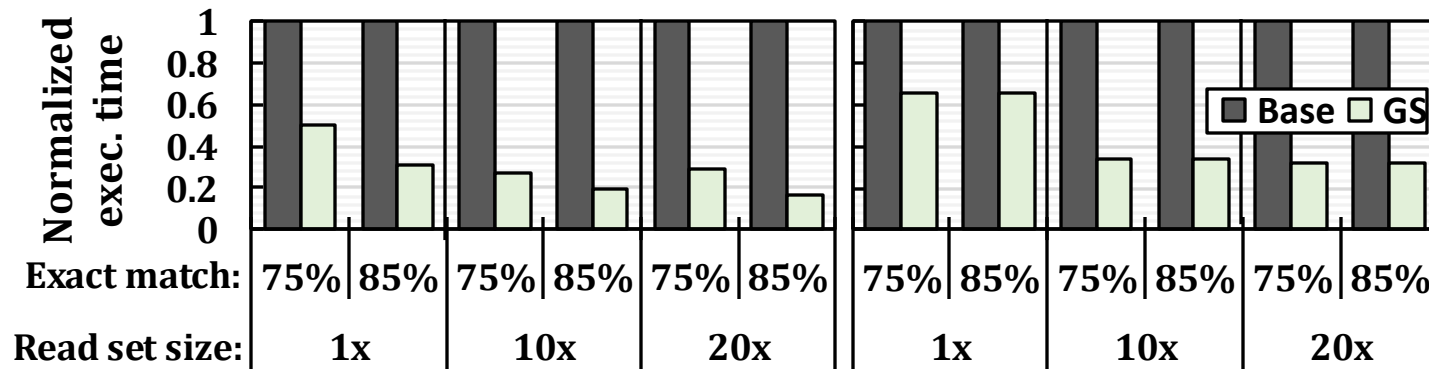
GenStore-NM



**GS-Ext performs significantly slower than Base (2.28x - 1.91x)
on all systems.**

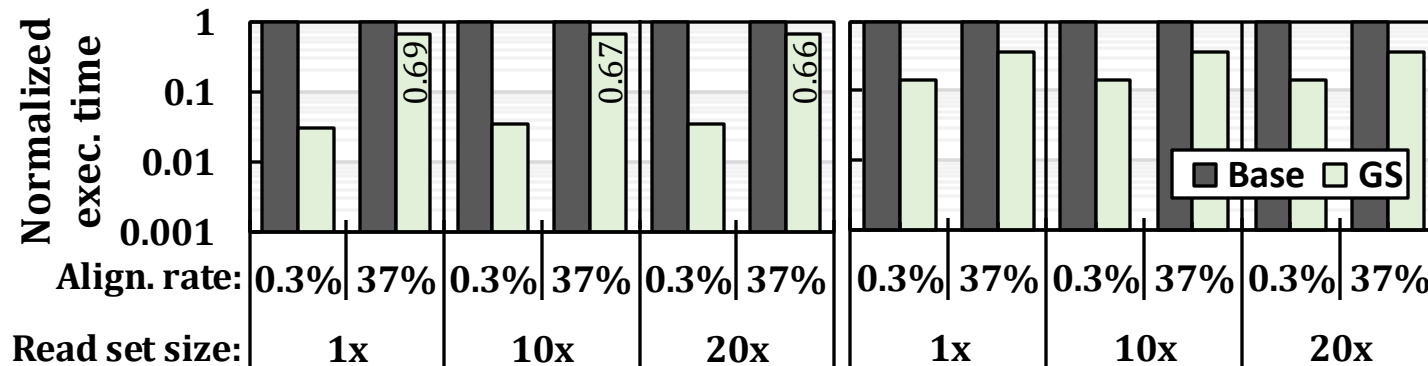
Effect of Inputs on GenStore-EM

$$DM_Saving = \frac{Size_{Ref} + Size_{ReadSet}}{Size_{Ref} + Size_{ReadSet} \times (1 - Ratio_{Filter})}$$



Effect of Inputs on GenStore-NM

$$DM_Saving = \frac{Size_{Ref} + Size_{ReadSet}}{Size_{Ref} + Size_{ReadSet} \times (1 - Ratio_{Filter})}$$



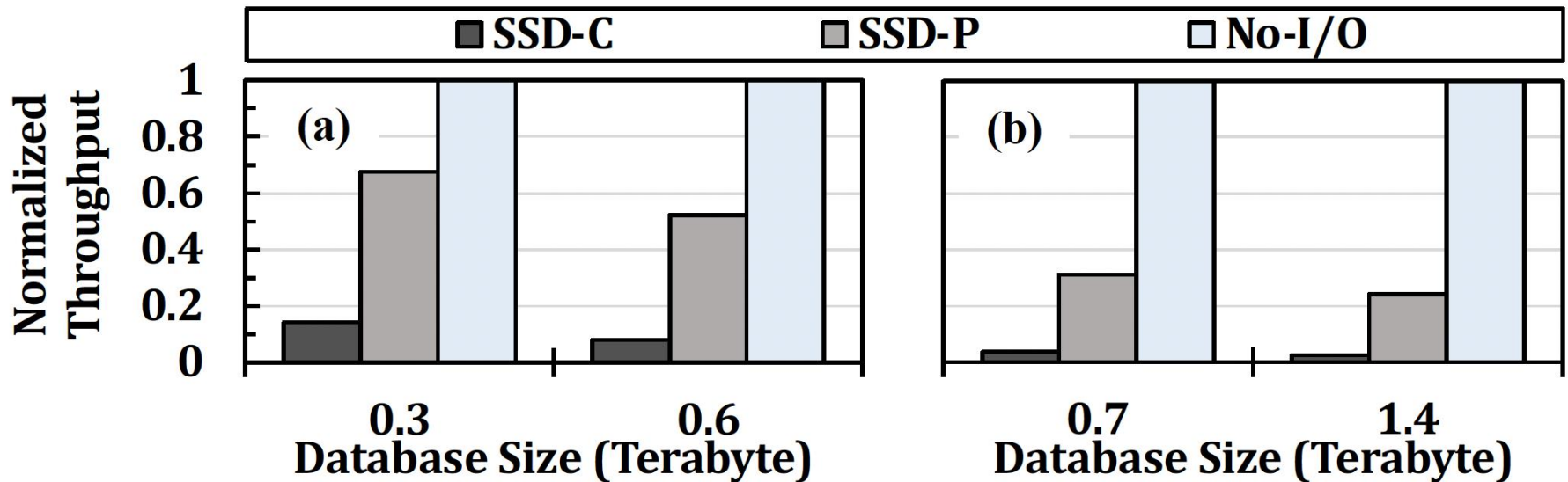
MegIS Backup Slides

Motivational Analysis

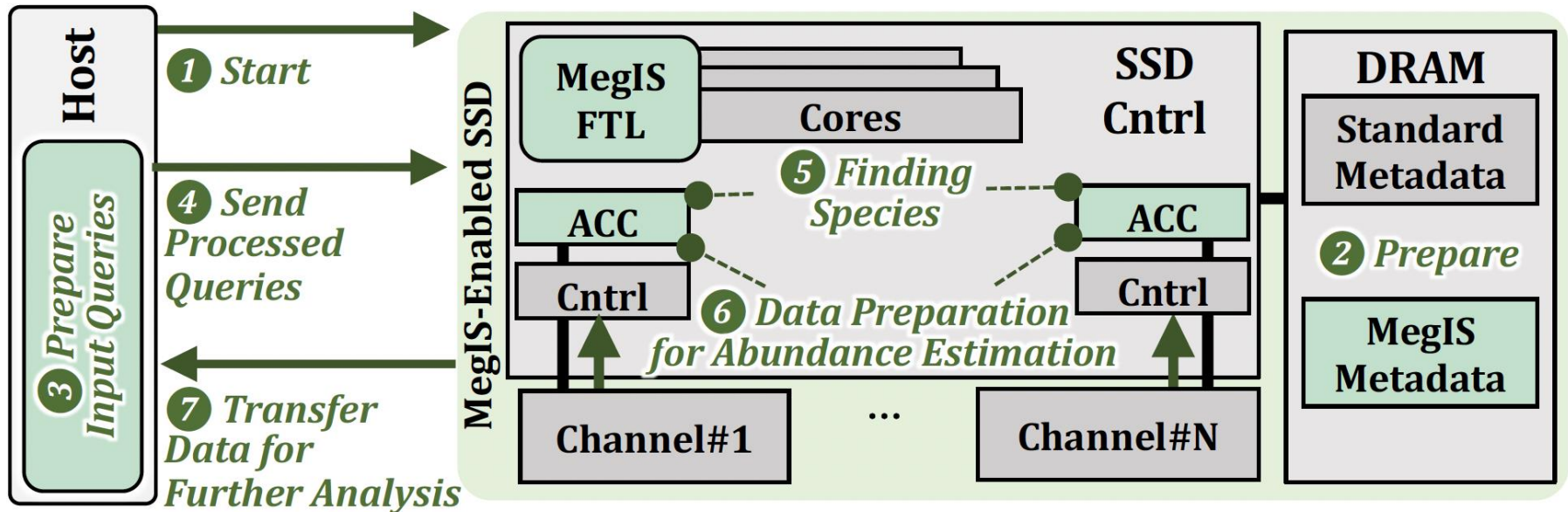
Database access patterns

(a) Random Query

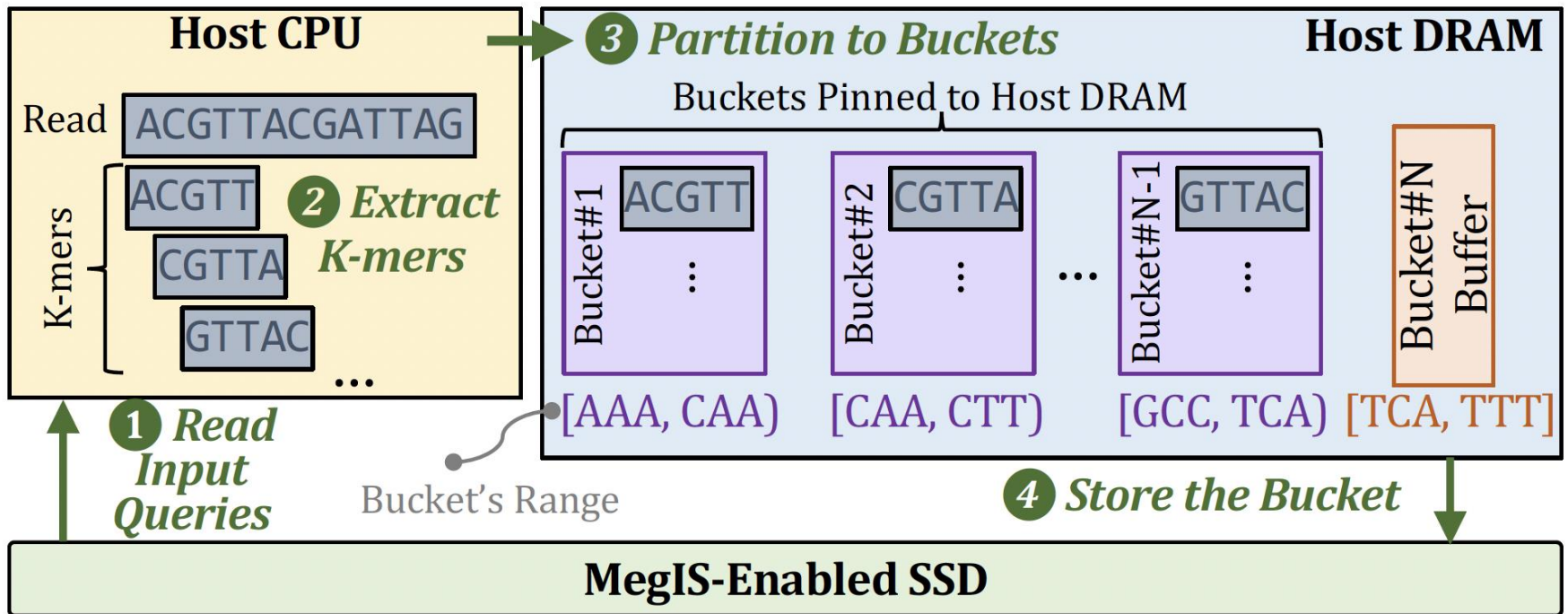
(b) Streaming Query



Overview of MegIS's Steps



More Details on Step 1



K-mer Sketch Data Structures

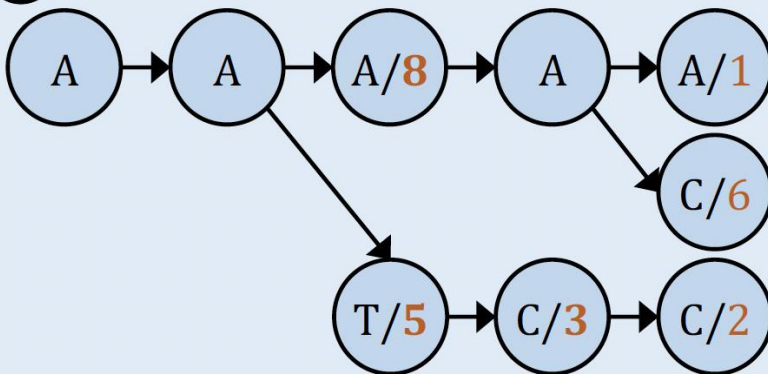
a Baseline K-mer Sketch Tables

5-mer	ID
AAAAA	1
AAAAC	6
AATCC	2
...	...

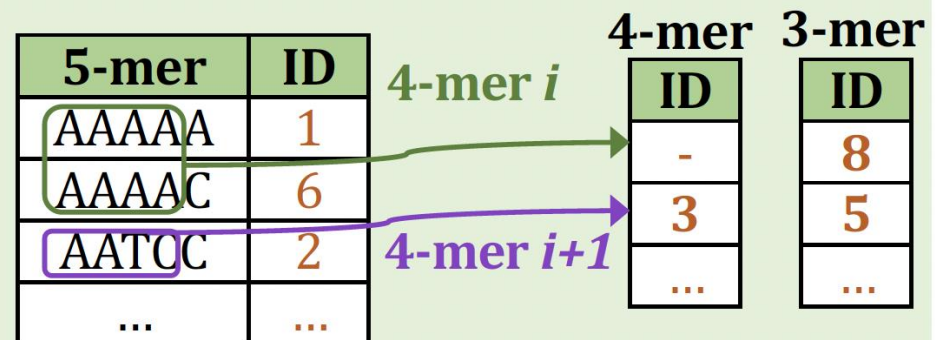
4-mer	ID
AAAA	1, 6
AATC	2, 3
...	...

3-mer	ID
AAA	1, 6, 8
AAT	2, 3, 5
...	...

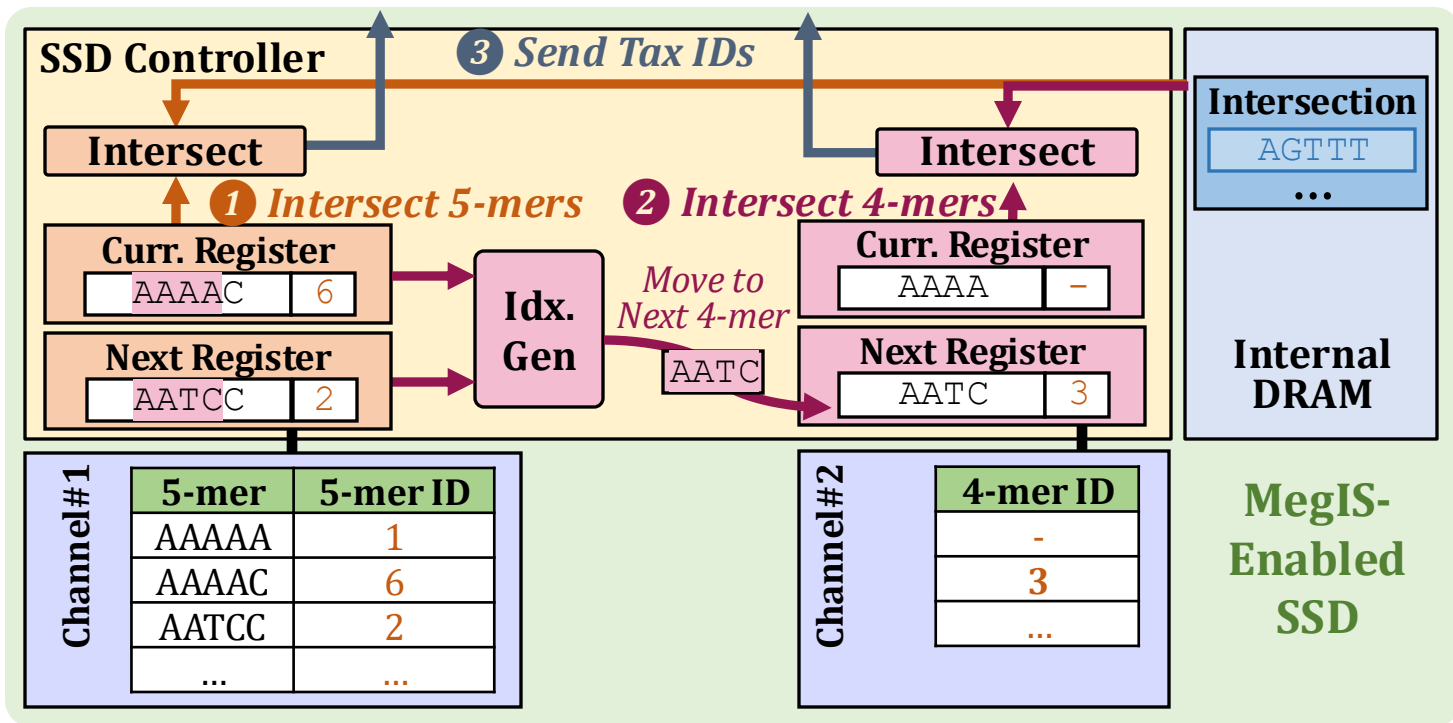
b Ternary Search Tree



c K-mer Sketch Streaming Tables



K-mer Sketch Streaming Hardware Design



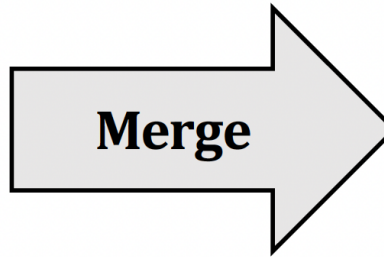
Index Generation in Step 3

K-mer	Loc.
ATT	14
CCA	9
GCT	5
...	...

Reference Index
Organism A

K-mer	Loc.
AAG	2
CCA	21
TGC	4
...	...

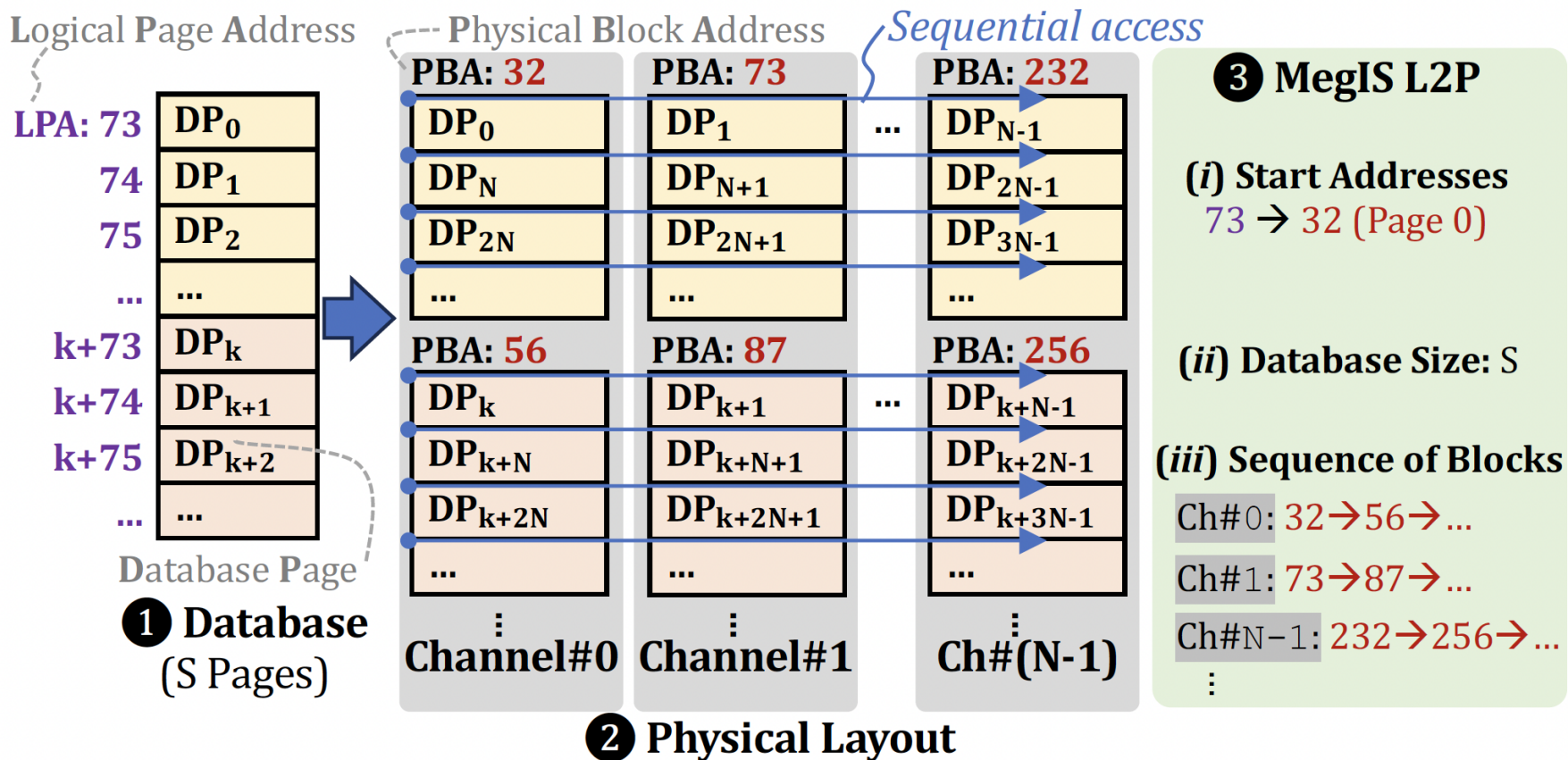
Reference Index
Organism B



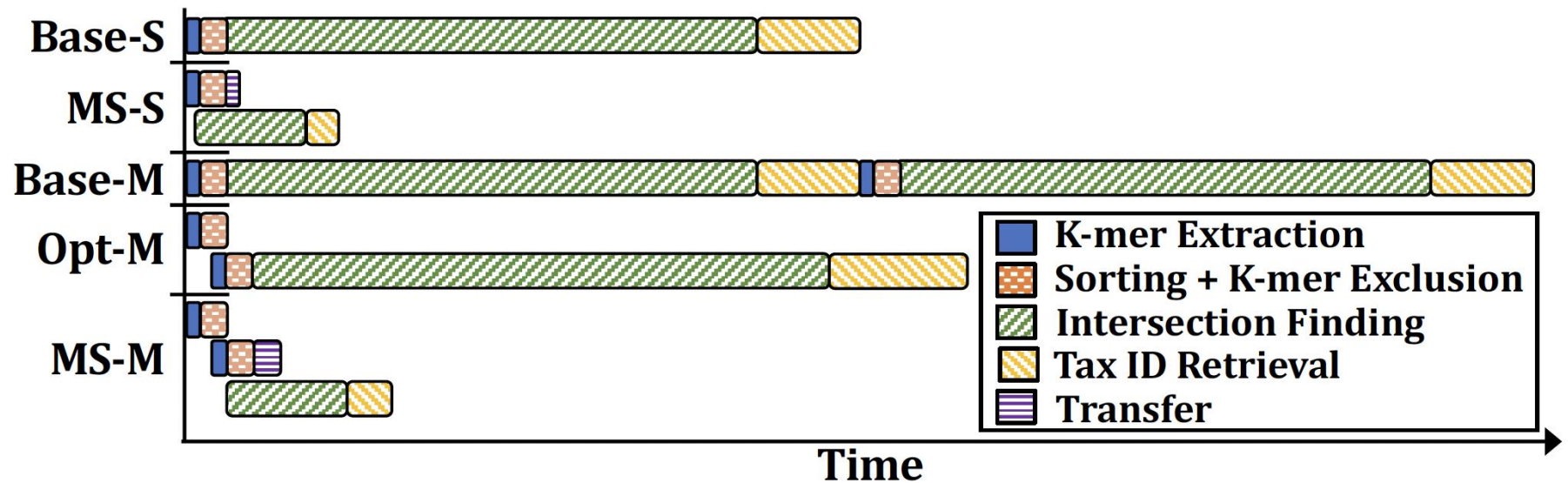
K-mer	Loc.
AAG	1002
ATT	14
CCA	9, 1021
GCT	5
TGC	1004
...	...

Unified
Reference Index

MegIS FTL



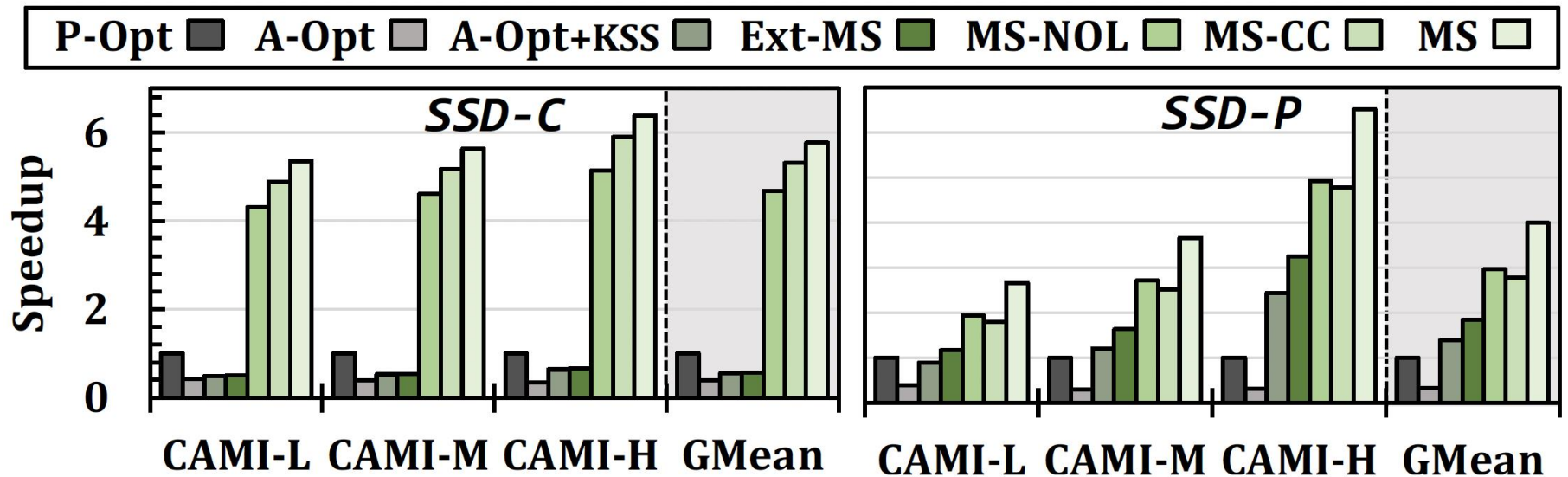
Multi-Sample Analysis



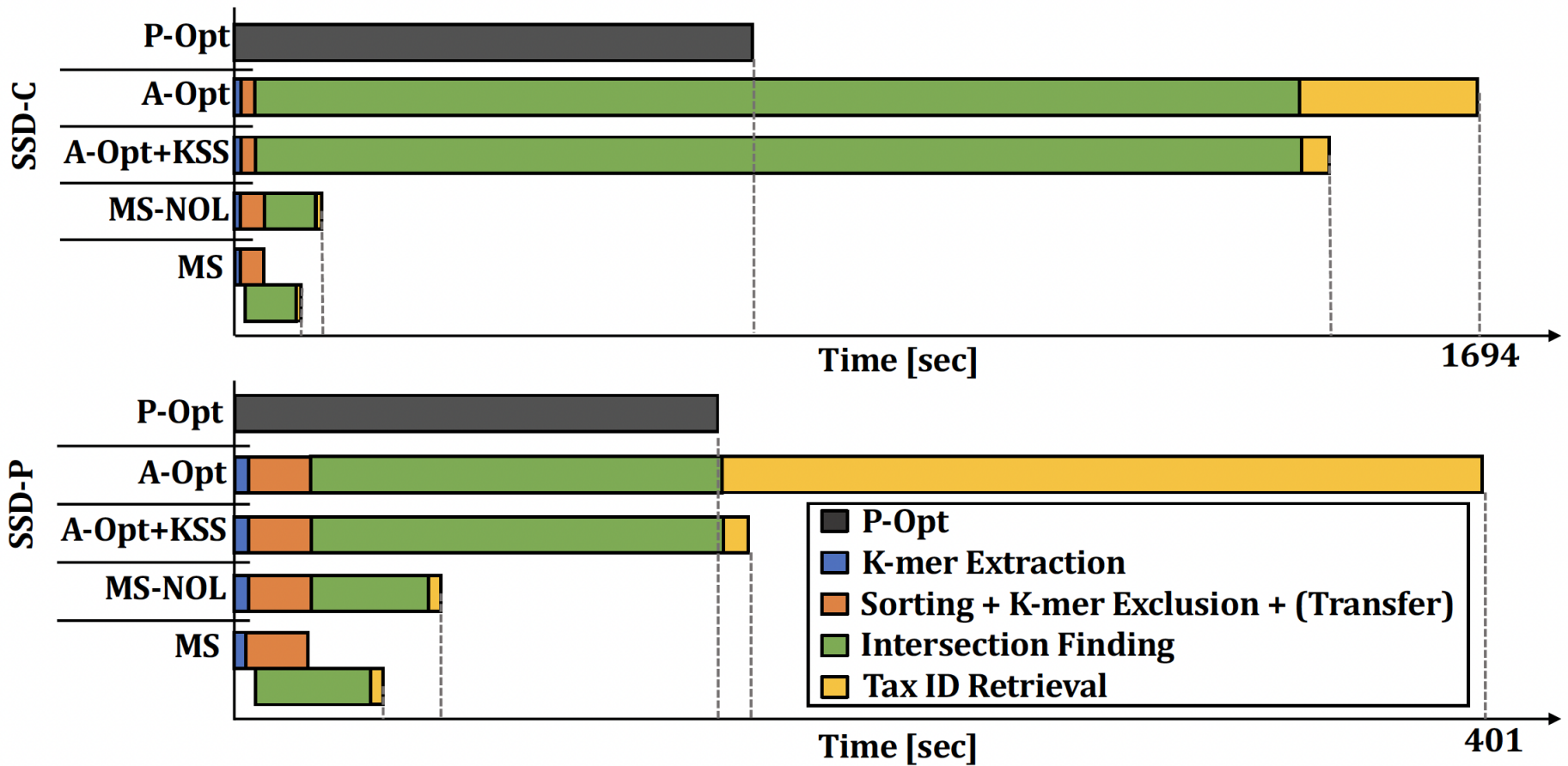
SSD Configurations

Specification	SSD-C	SSD-P
General	48-WL-layer 3D TLC NAND flash-based SSD 4 TB capacity, 4 GB internal LPDDR4 DRAM [226]	
Bandwidth (BW)	600 MB/s interface BW (SATA3); 560 MB/s sequential-read BW 1.2-GB/s channel I/O rate	8 GB/s interface BW (4-lane PCIe Gen4); 7 GB/s sequential-read BW 1.2-GB/s channel I/O rate
NAND Config	8 channels, 8 dies/channel, 4 planes/dies, 2,048 blocks/plane, 196 WLS/block, 16 KiB/page (4/8/16 channels in Fig. 17)	16 channels, 8 dies/channel, 2 planes/dies, 2,048 blocks/plane, 196 WLS/block, 16 KiB/page (8/16/32 channels in Fig. 17)
Latencies	Read (tR): 52.5 μ s, Program (tPROG): 700 μ s	
Embedded Cores	3 ARM Cortex-R4 cores [86]	4 ARM Cortex-R4 cores [86]

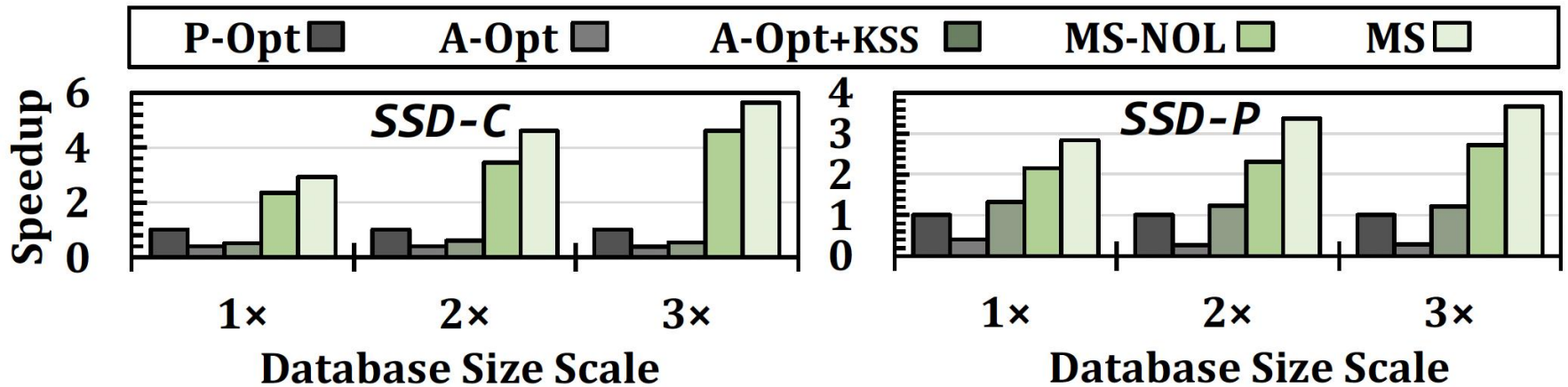
Impact of Different Optimizations



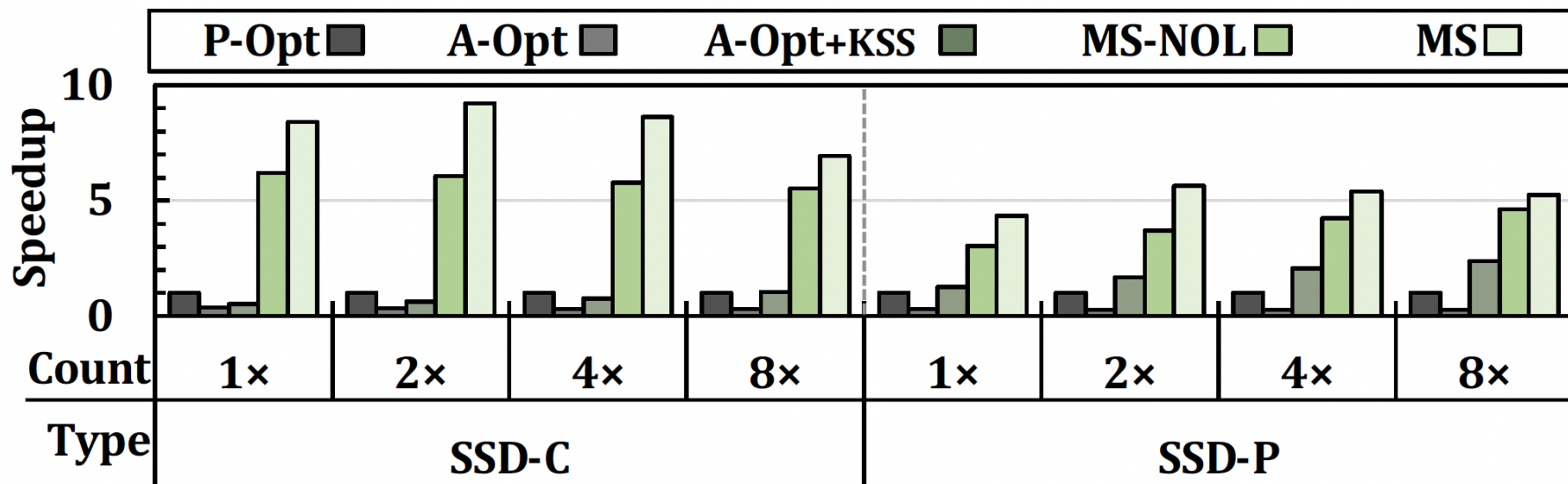
Impact of Different Optimizations



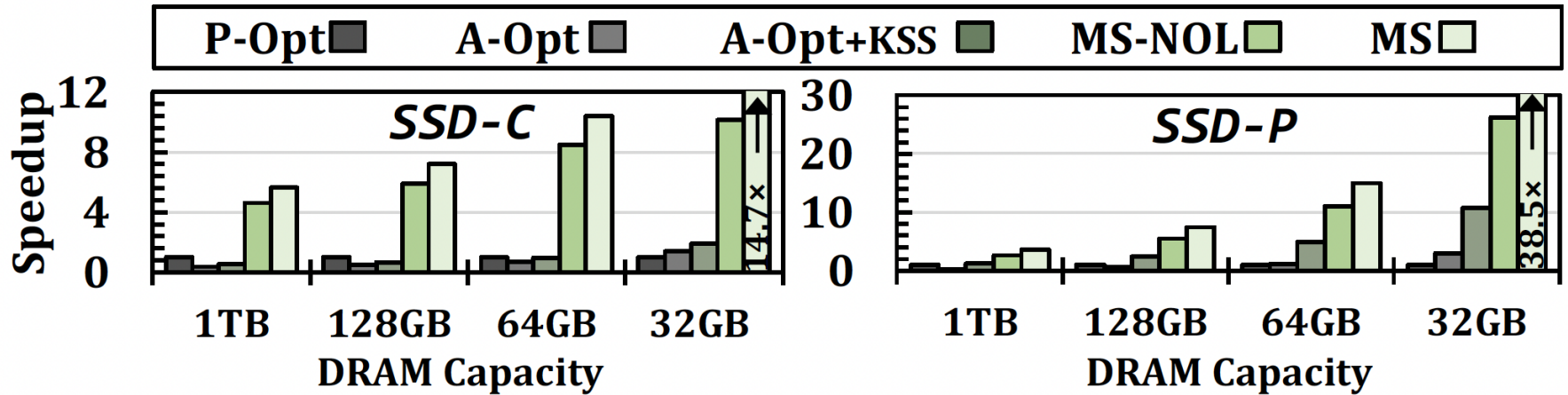
Speedup with Different Database Sizes



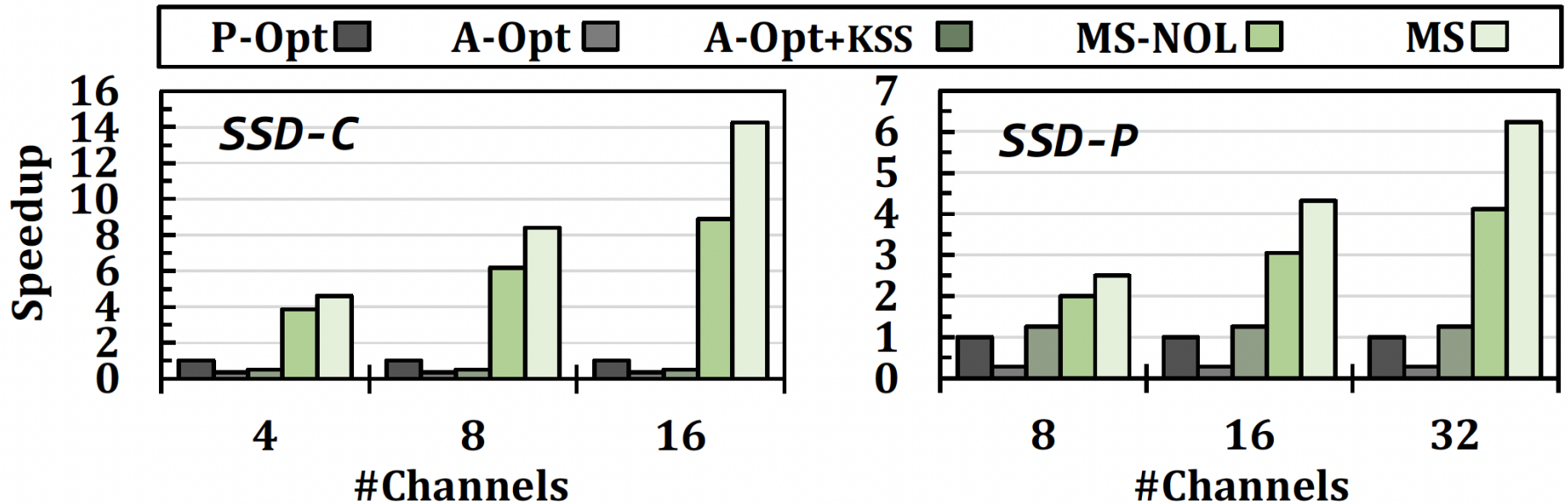
Speedup with Different #SSDs



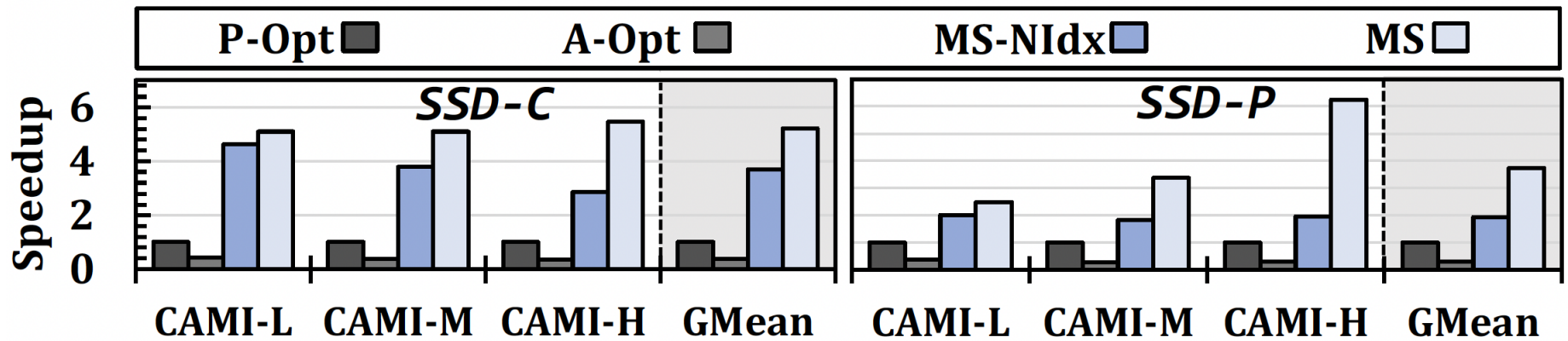
Speedup with Different Main Memory Capacities



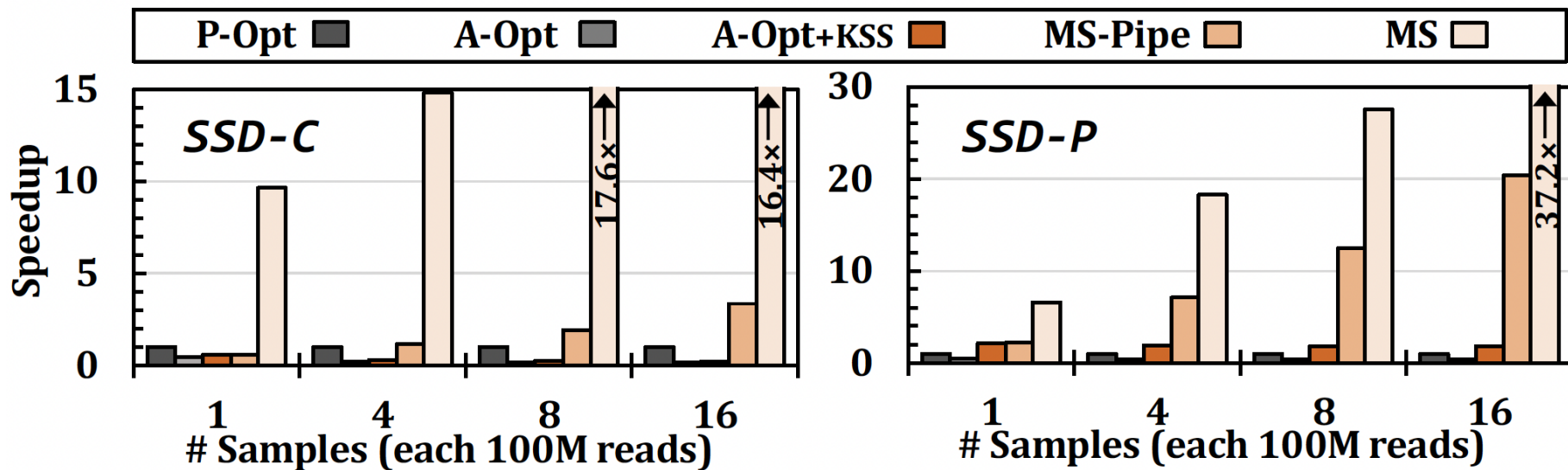
Speedup with Varying SSD Internal Bandwidth



Speedup of Abundance Estimation



Multi-Sample Use Case



Area and Power

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

Logic Unit	# of instances	Area [mm ²]	Power [mW]
Intersect (120-bit)	1 per channel	0.001361	0.284
k-mer Registers (2 x 120-bit)	1 per channel	0.002821	0.645
Index Generator (64-bit)	1 per channel	0.000272	0.025
Control Unit	1 per SSD	0.000188	0.026
<i>Total for an 8-channel SSD</i>	-	<i>0.04</i>	<i>7.658</i>

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