Storage-Centric Computing for Genomics and Metagenomics

Nika Mansouri Ghiasi

n.mansorighiasi@gmail.com

ETH zürich

SAFARI

Brief Self Introduction

- A PhD student at the SAFARI Research Group @ ETH Zurich, advised by Professor Onur Mutlu

- **Research interests:**
  - Computer architecture
  - Large-scale bioinformatics applications
  - Storage systems
  - Near data processing
  - Emerging technologies such as ultra-dense 3D integrated systems

- **Contact information**
  - **Email:** n.mansorighiasi@gmail.com
  - **Personal website:** https://bit.ly/nikamgh

Outline

- Brief Intro to (Meta)Genomics

- Storage-Centric Designs for (Meta)Genomics
  - GenStore
  - MegIS

- Conclusion

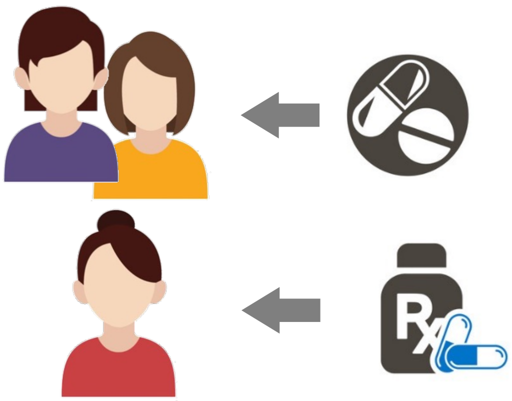
Outline

- *Brief Intro to (Meta)Genomics*

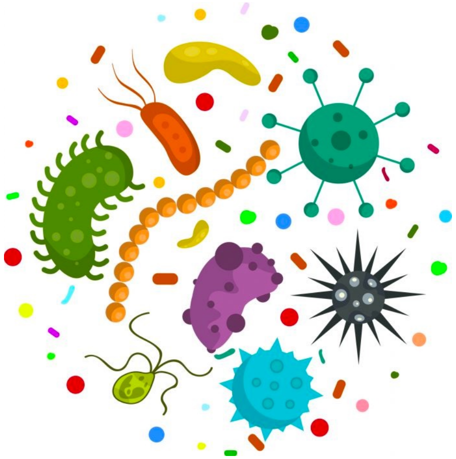
- Storage-Centric Designs for (Meta)Genomics
  - GenStore
  - MegIS

- Conclusion

Genomics and Metagenomics are Critical for Many Applications



Developing **personalized medicine**

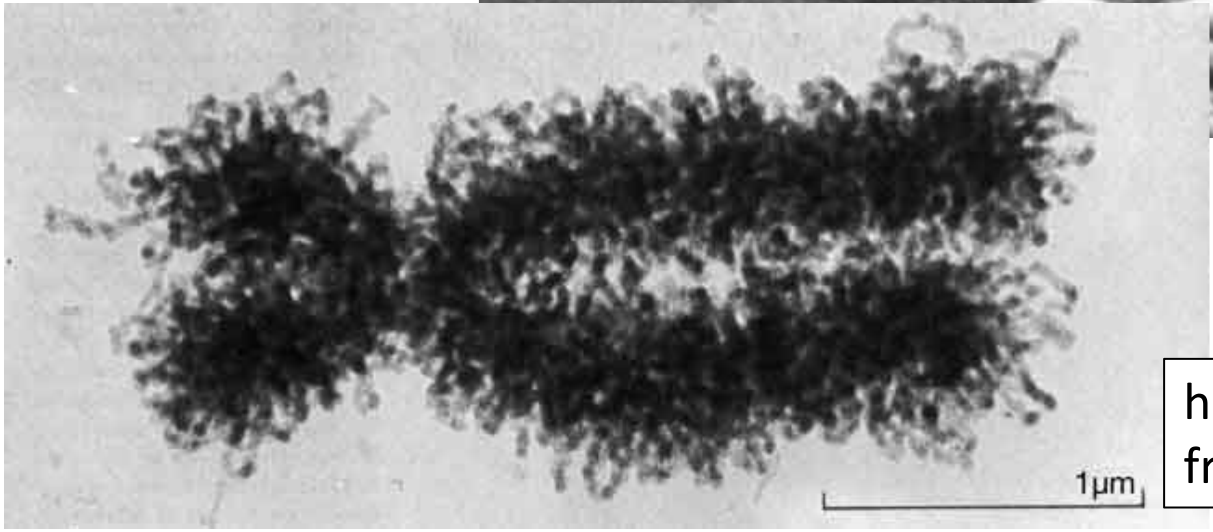


Predicting the presence and relative abundance of **microbes** in a sample

Rapid surveillance of **disease outbreaks**

Understanding **genetic variations, species, evolution, …**

DNA Under Electron Microscope

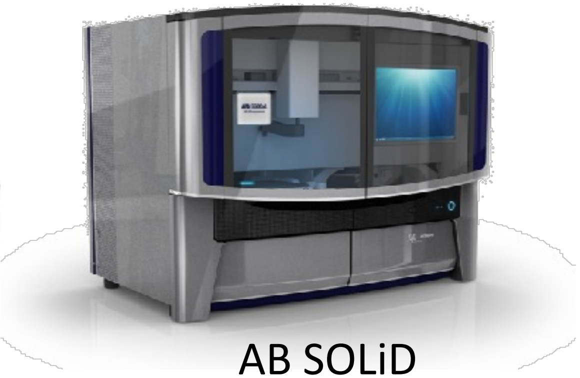


human chromosome #12
from HeLa's cell

CCTCCTCAGTGCCACCCAGCCCACTGGCAGCTCCCAAACA
GGCTCTTATTAAAACACCCTGTTCCCTGCCCCTTGGAGTG
AGGTGTCAAGGACCTAAACTAAAAAAAAAAAAAGAAAA
AGAAAAGAAAAAAGAATTTAAAATTTAAGTAATTCTTTGAA
AAAAACTAATTTCTAAGCTTCTTCATGTCAAGGACCTAATG
TGCTAAACAGCACTTTTT**TTGACCATTAT**TTTGGATCTGAAA
GAAATCAAGAATAAATGAAGGACTTGATACATTGGAAGA
GGAGAGTCAAGGACCTACAGAAAAAAAAAAAAAAGAAA
AAGAAAAGAAAAAAGA**A**TTTAAAATTTAAGTAATTCTTTGA
AAAAAACTAATTTCTAAGCTTCTT**C**ATGTCAAGGACCTAAT
GTCTGTGTTGCAGGTCTTCTTGCATTTCCCTGTCAAAAGA
AAAAGAATTTAAAATTTAAGTAATTCTTTGAAAAAAACTA
ATTTCTAAGCTTCTTCATGTCAAGGACCTAATGTCAGGCC
GGCTCTTATTAAAACACCCTGTTCCCTGCCCCTTGGAGTG

Genome Sequencers

Roche/454



AB SOLiD

Illumina MiSeq

Complete Genomics

Oxford Nanopore MinION

Illumina NovaSeq 6000

Illumina HiSeq2000

Pacific Biosciences RS

Oxford Nanopore GridION

Ion Torrent PGM

Ion Torrent Proton

… and more! All produce data with different properties.

High-Throughput Sequencers

Illumina MiSeq

Pacific
Biosciences
Sequel II

Oxford
Nanopore
PromethION

Illumina NovaSeq 6000

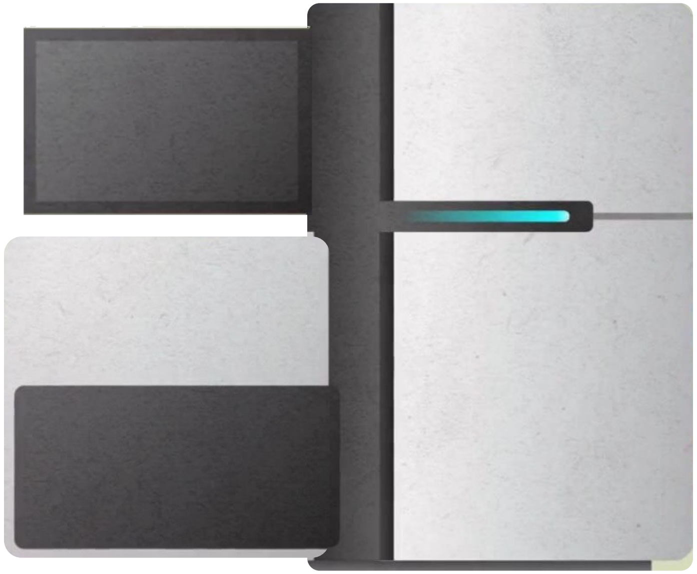
Pacific Biosciences RS II

Oxford Nanopore MinION

Oxford
Nanopore
SmidgION

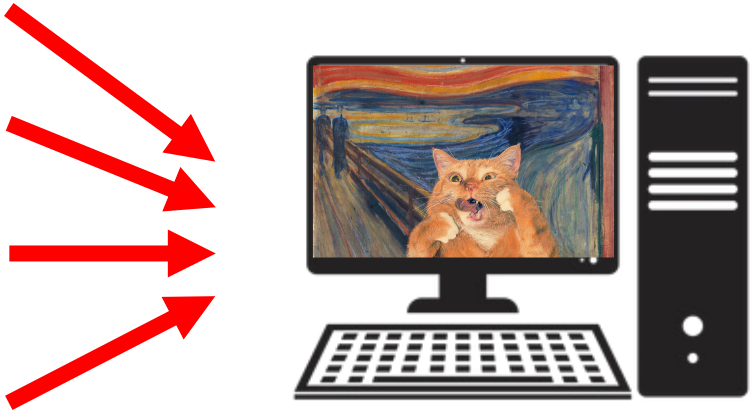
… and more! All produce data with different properties.

Problems with (Meta) Genome Analysis Today



Special-Purpose Machine
for **Data Generation**

FAST



General-Purpose Machine
for **Data Analysis**

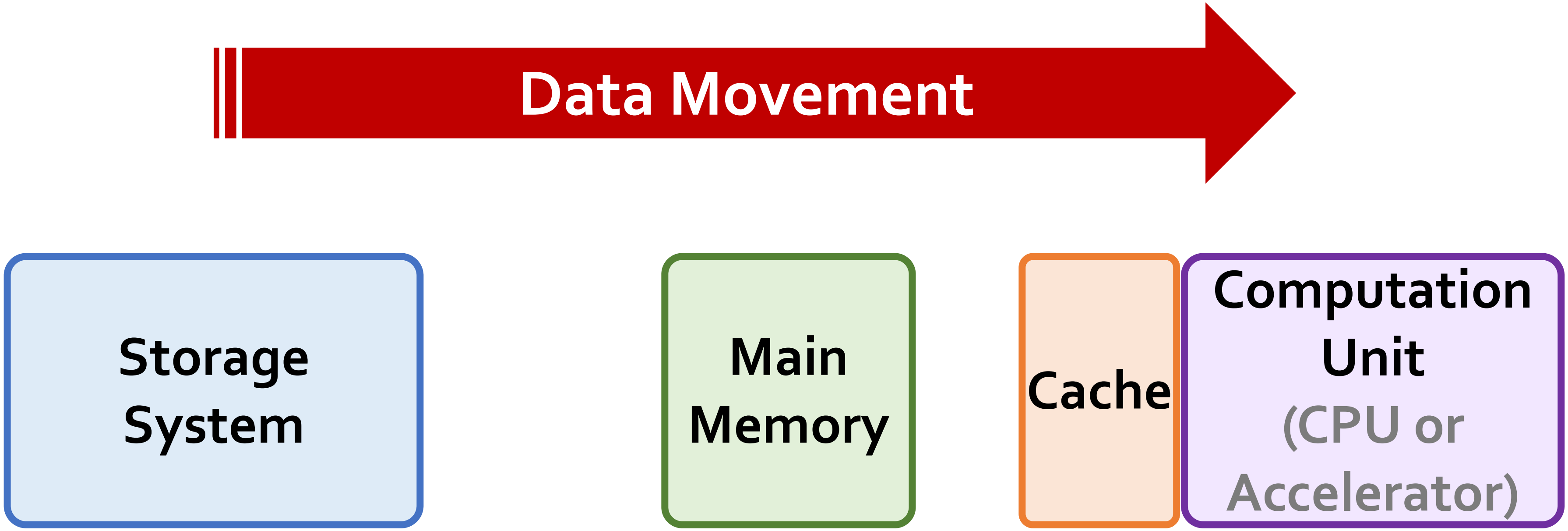
SLOW

Slow and inefficient processing capability

Large amounts of data movement

SAFARI This picture is similar for many "data generators & analyzers" today

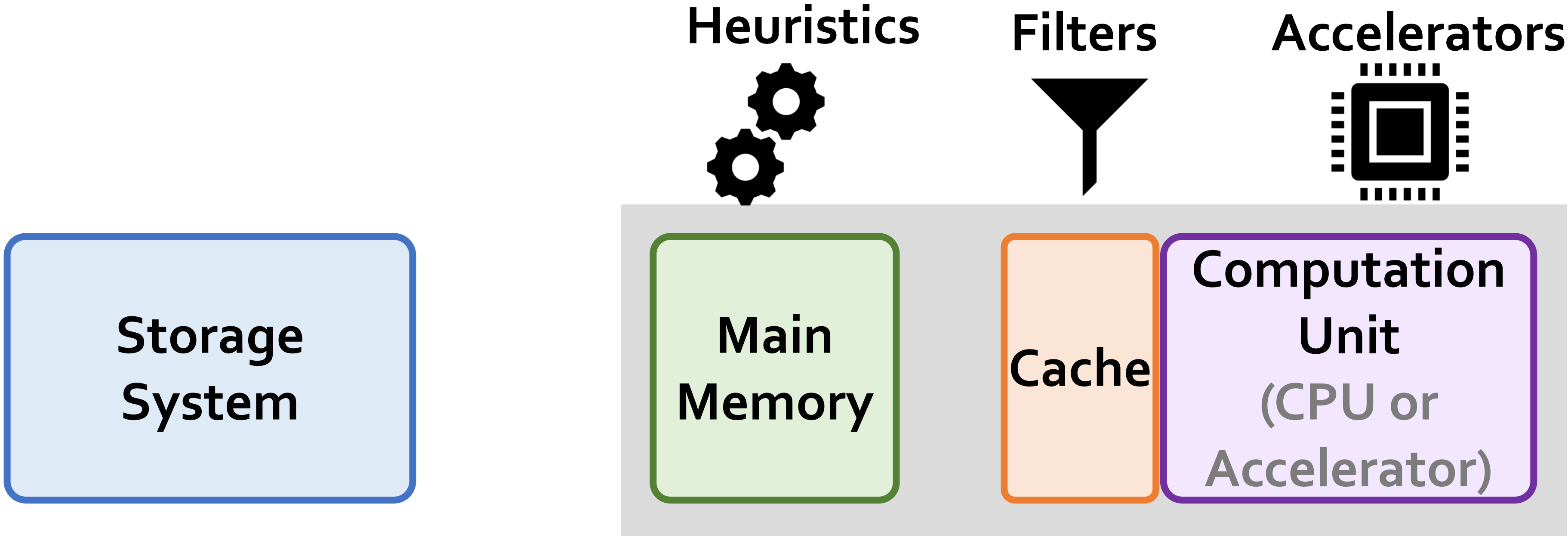
Genome Sequence Analysis



✗ Computation overhead

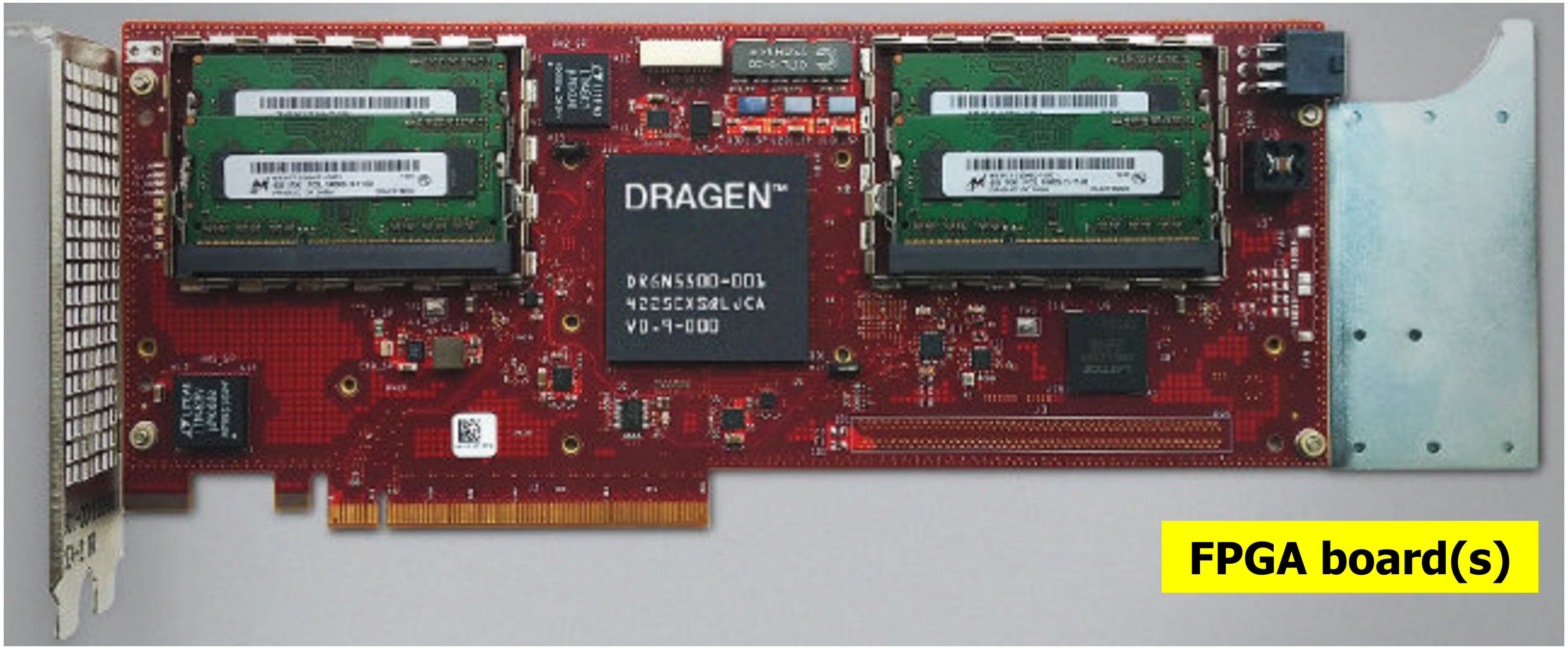
✗ Data movement overhead

Accelerating (Meta)Genome Analysis



Illumina DRAGEN Bio-IT Platform (2018)

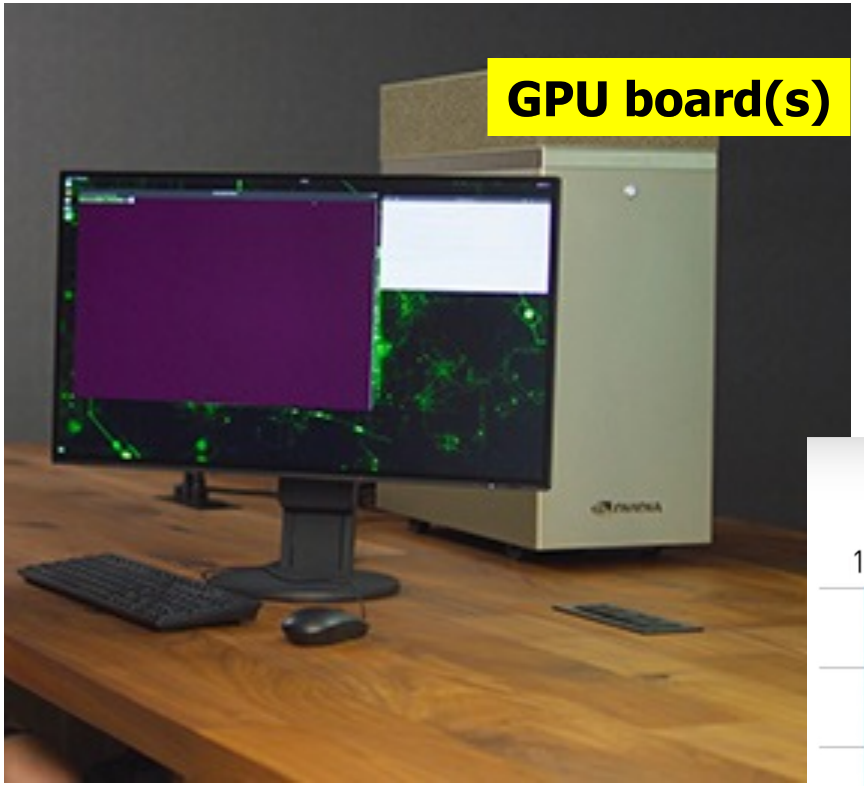
- Processes whole genome at 30x coverage in ~25 minutes with hardware support for data compression



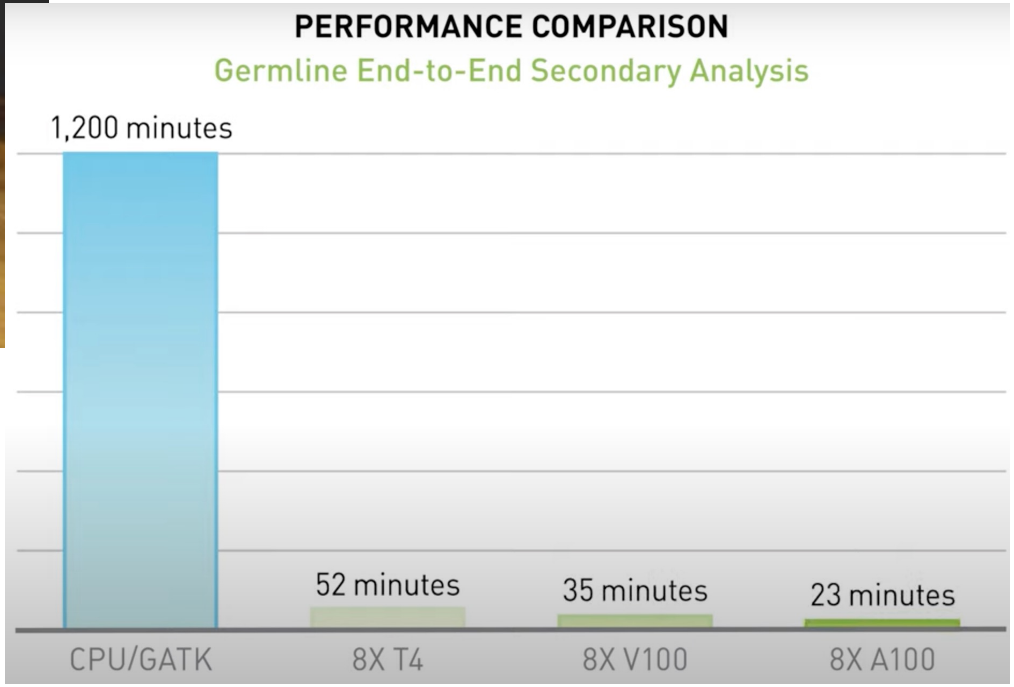
FPGA board(s)

emea.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html
emea.illumina.com/company/news-center/press-releases/2018/2349147.html

NVIDIA Clara Parabricks (2020)

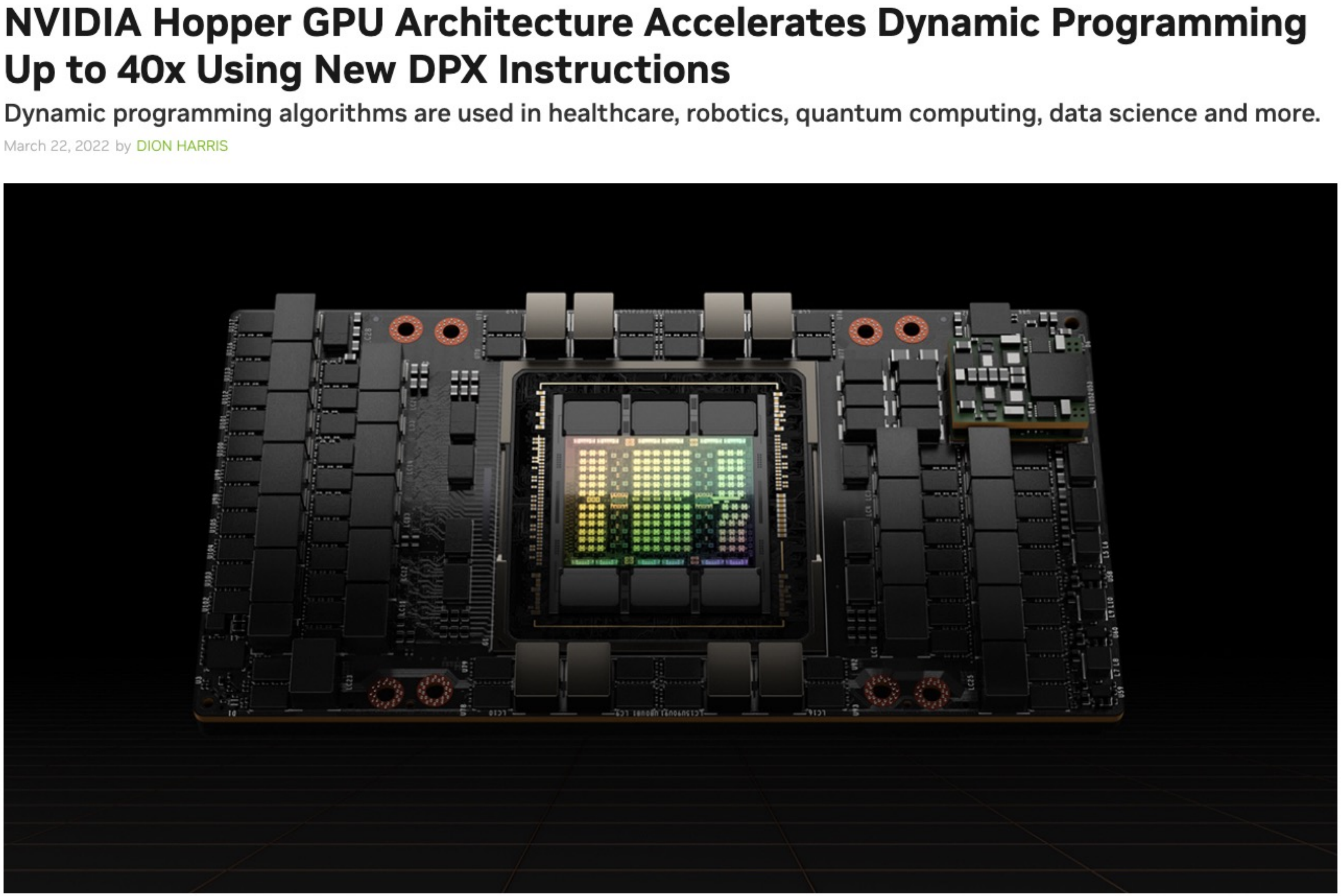


A University of Michigan startup in 2018 joined NVIDIA in 2020



https://developer.nvidia.com/clara-parabricks

NVIDIA Hopper DPX Instructions (2022)



NVIDIA Hopper GPU Architecture Accelerates Dynamic Programming Up to 40x Using New DPX Instructions

Dynamic programming algorithms are used in healthcare, robotics, quantum computing, data science and more.

March 22, 2022 by DION HARRIS

SAFARI https://blogs.nvidia.com/blog/2022/03/22/nvidia-hopper-accelerates-dynamic-programming-using-dpx-instructions/

- We are accelerating the transformation in how we analyze the human genome!

Bionano & NVIDIA:

Accelerating Analysis for Fast Time to Results

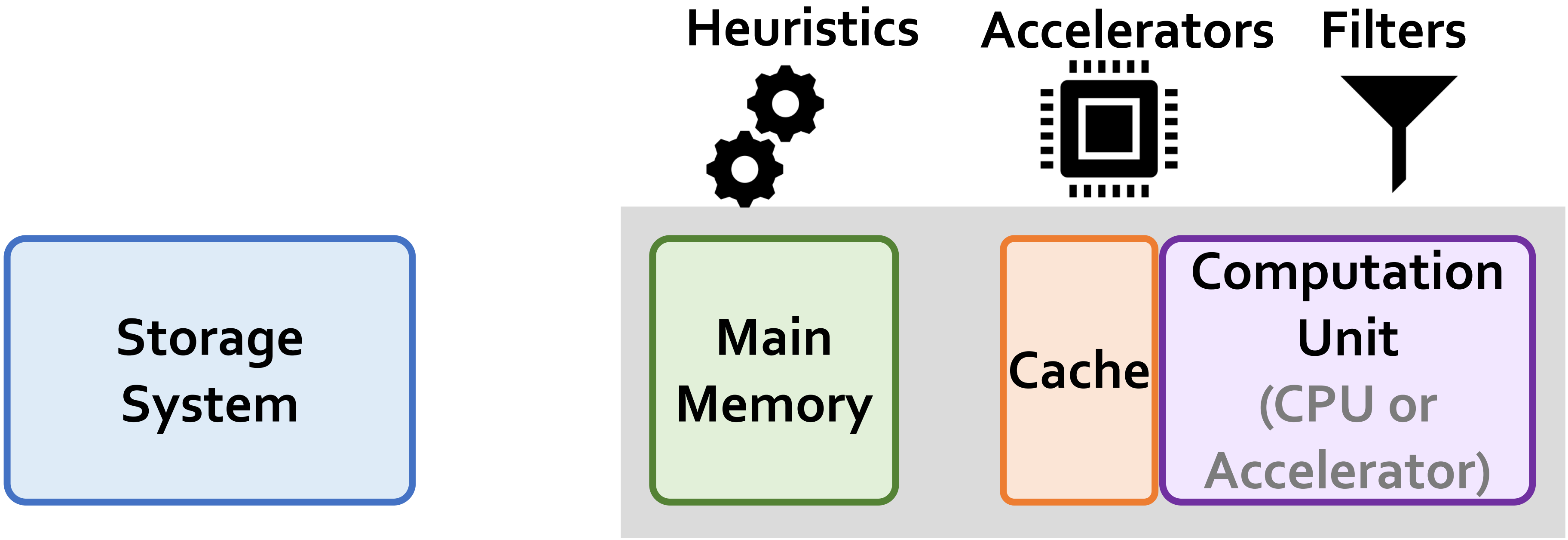
Technological solution to **support higher throughput**

New high-performance algorithms from Bionano

Powered by NVIDIA RTX™ 6000 Ada Generation GPUs

Workflow tailored for a **small lab and IT footprint**

Accelerating Genome Sequence Analysis

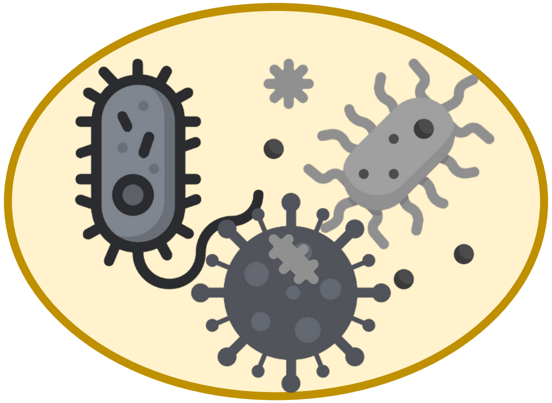


✓ Computation overhead

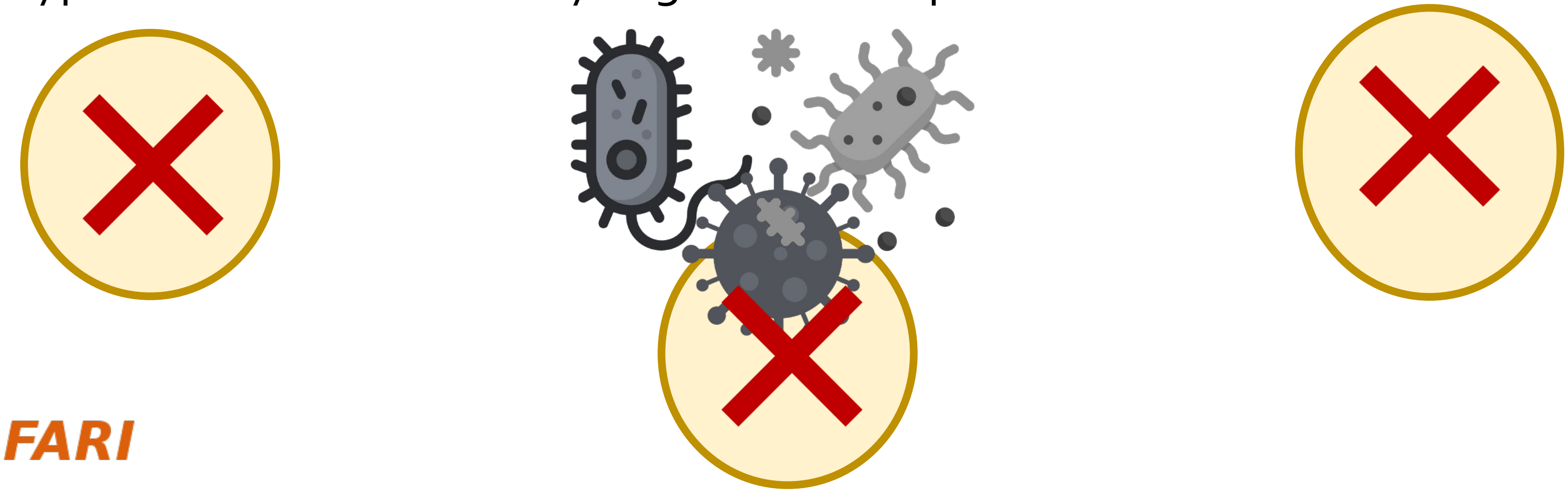
✗ Data movement overhead

What is Metagenomics?

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

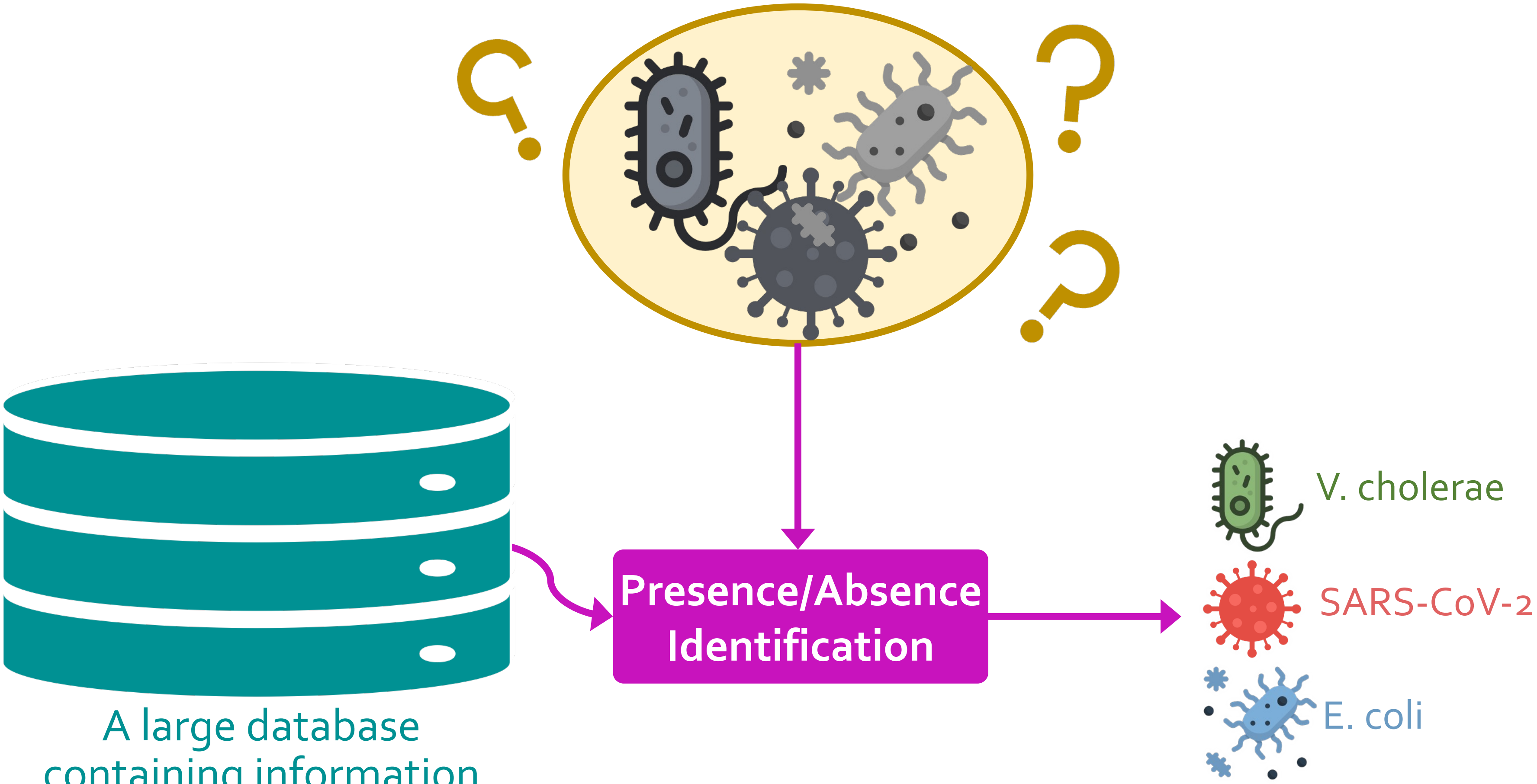


- **Overcomes the limitations of traditional genomics**
  - Bypasses the need for analyzing individual species in isolation



What is Metagenomics?

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



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

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

Our Goal

*Improve (meta)genomic analysis **performance***

*by reducing large **data movement overhead***

from the storage system

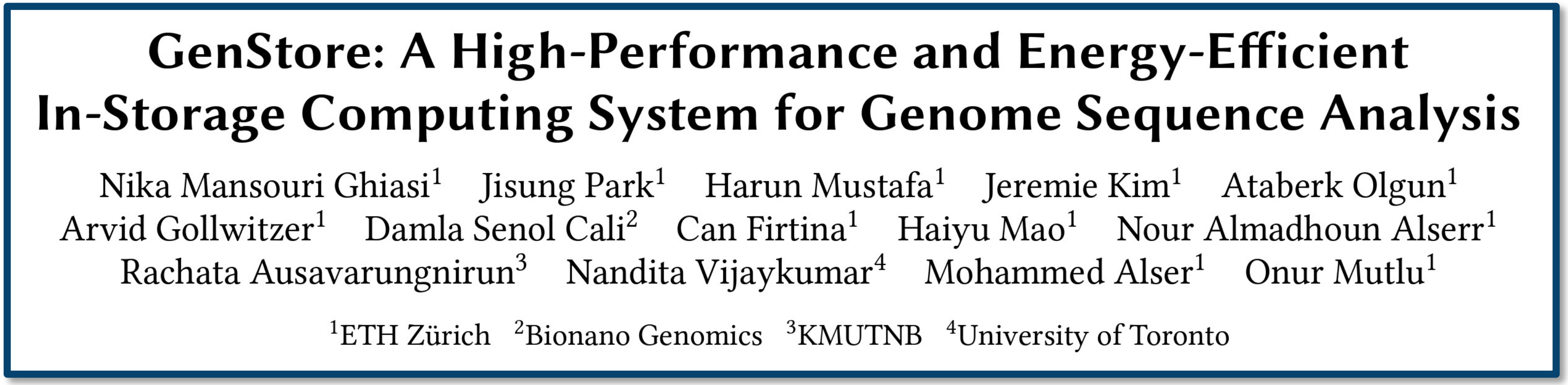
Outline

- Brief Intro to (Meta)Genomics

- *Storage-Centric Designs for (Meta)Genomics*
  - *GenStore*
  - MegIS

- Conclusion

GenStore *[ASPLOS'22]*



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

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

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

GenStore

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

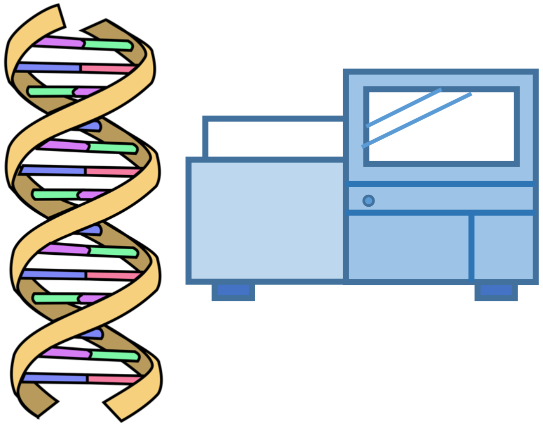
Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu

SAFARI

Genome Sequence Analysis

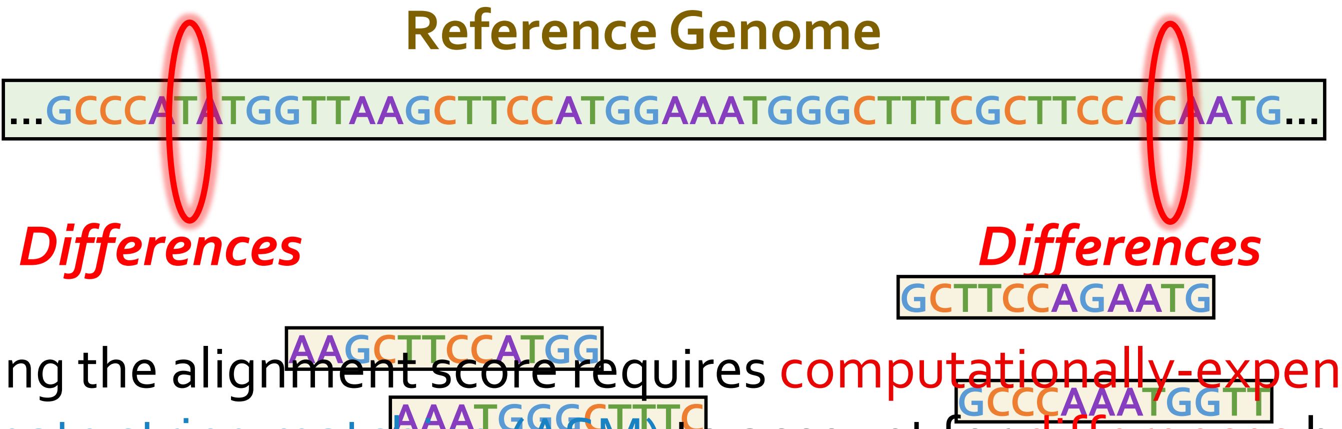
- Genome sequence analysis is critical for many applications
  - Personalized medicine
  - Outbreak tracing
  - Evolutionary studies

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



Genome Sequence Analysis

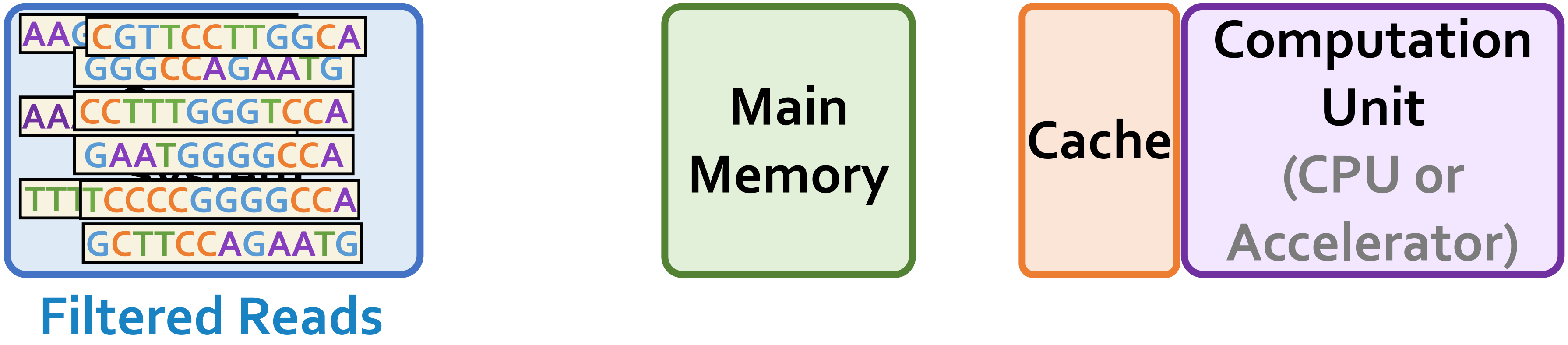
- **Read mapping:** first key step in genome sequence analysis
  - Aligns reads to potential matching locations in the reference genome
  - For each matching location, the alignment step finds the degree of similarity (alignment score)



- 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

Key Idea

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



Exactly-matching reads

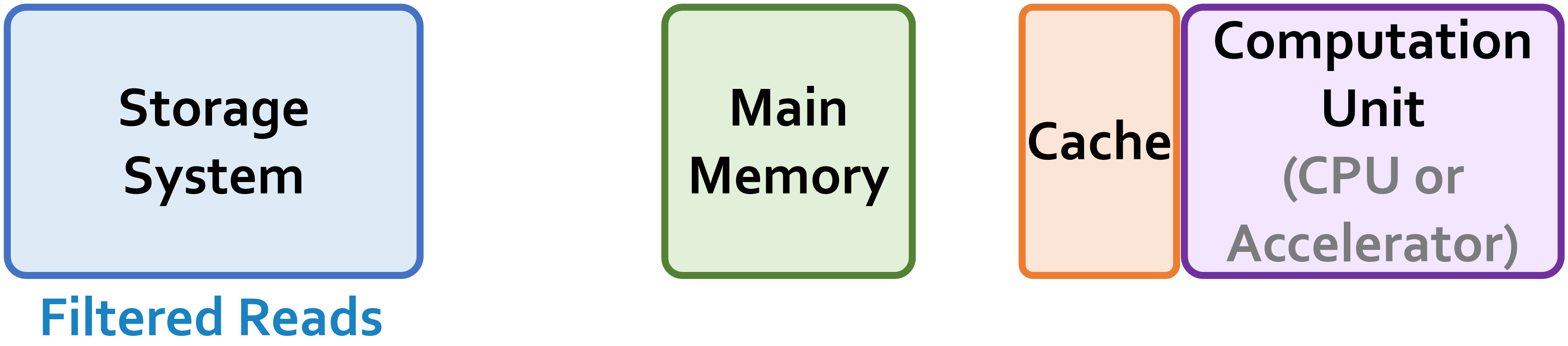
Do not need expensive approximate string matching during alignment

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

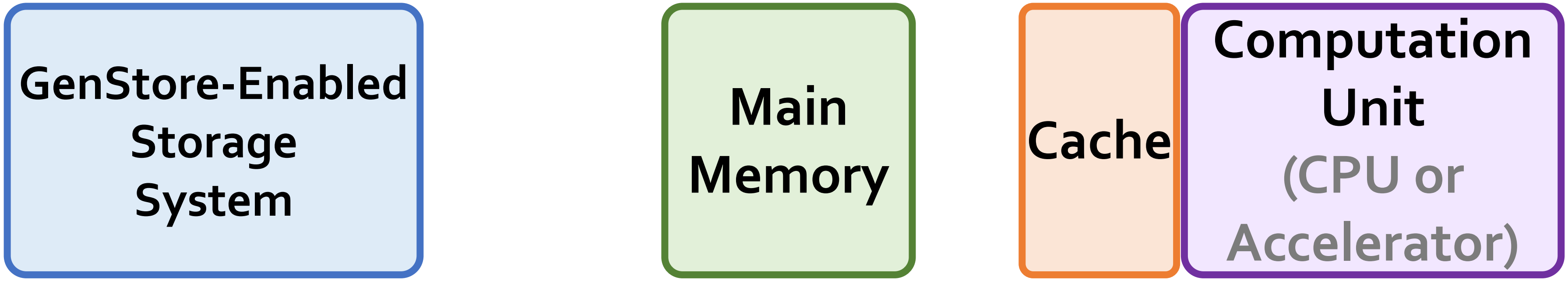


Read mapping workloads can exhibit different behavior

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

GenStore

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



GenStore-Enabled Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

✓ 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

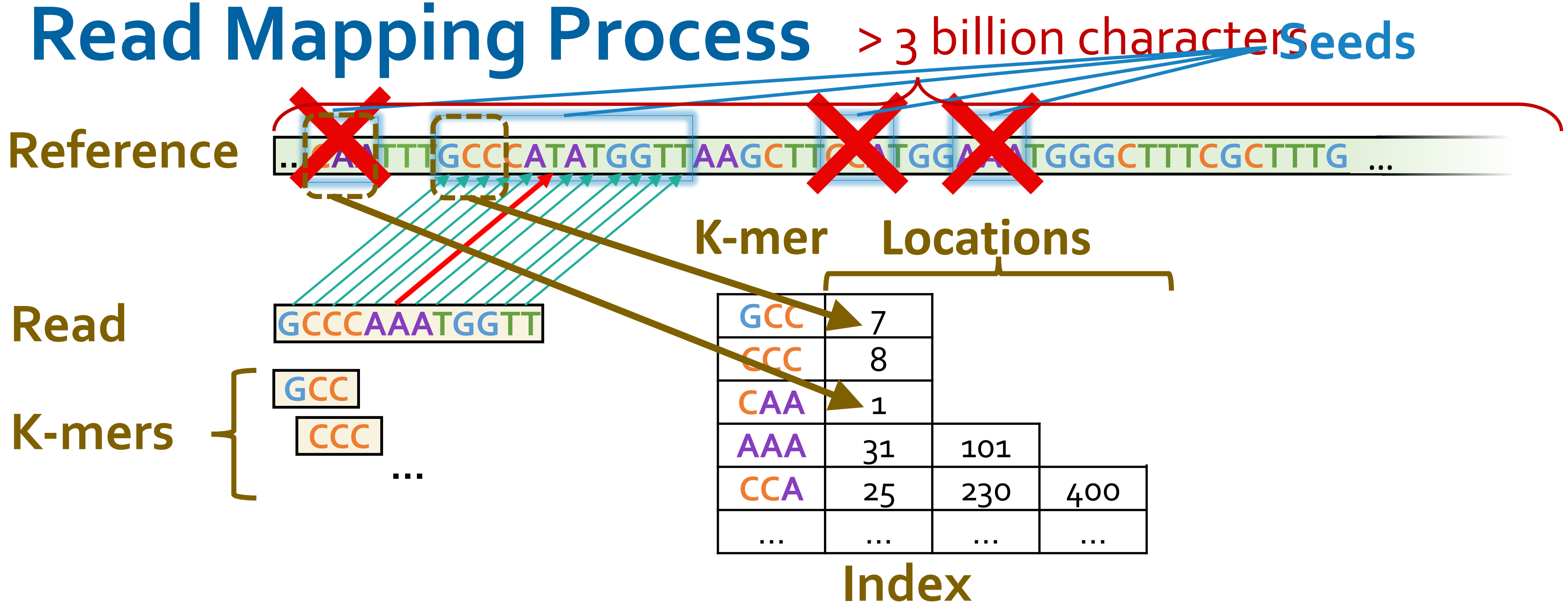
Motivation and Goal

GenStore

Evaluation

Conclusions

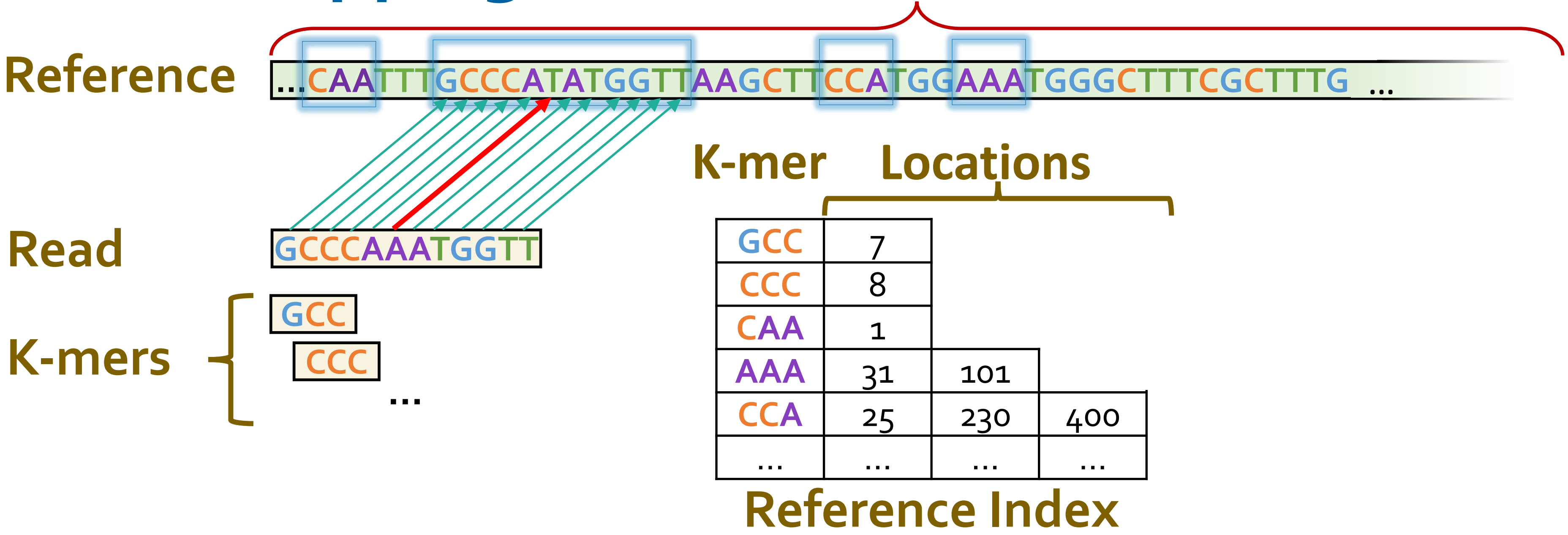
Read Mapping Process



K-mer	Locations		
GCC	7		
CCC	8		
CAA	1		
AAA	31	101	
CCA	25	230	400
...	...	...	...

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

Read Mapping Process



K-mer	Locations		
GCC	7		
CCC	8		
CAA	1		
AAA	31	101	
CCA	25	230	400
...	...	...	...

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

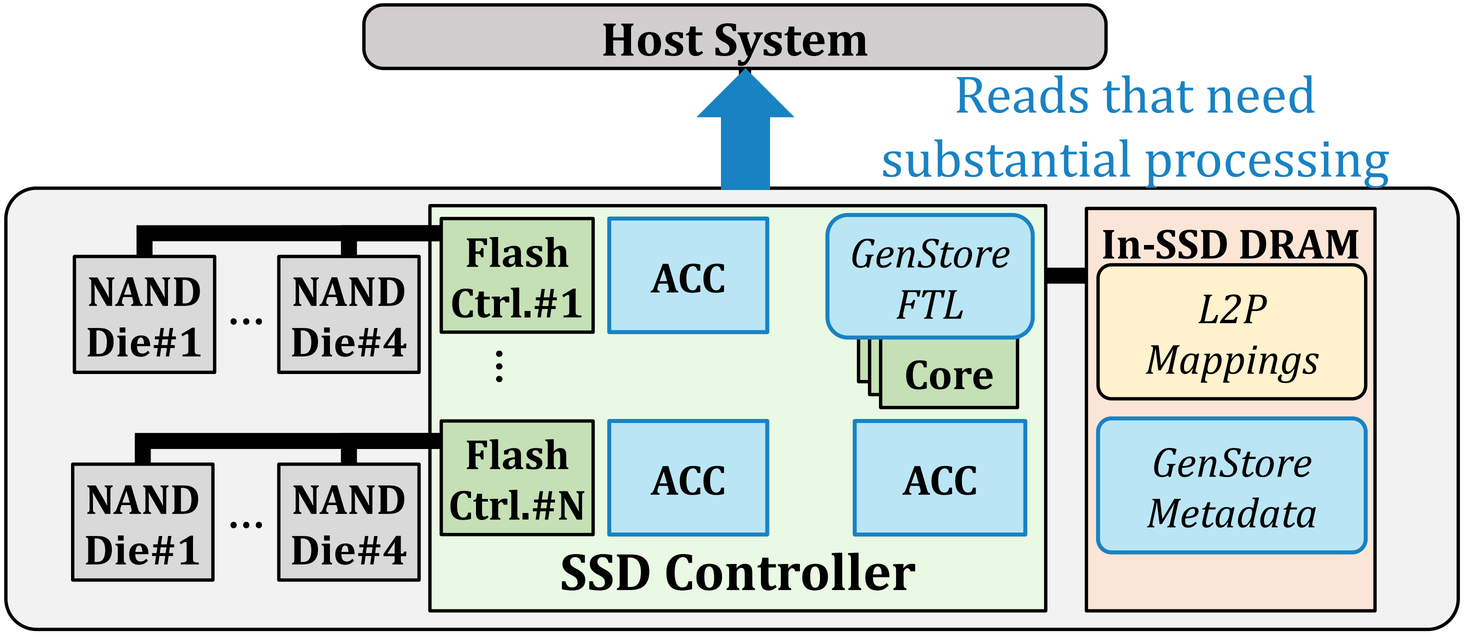
Evaluation

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

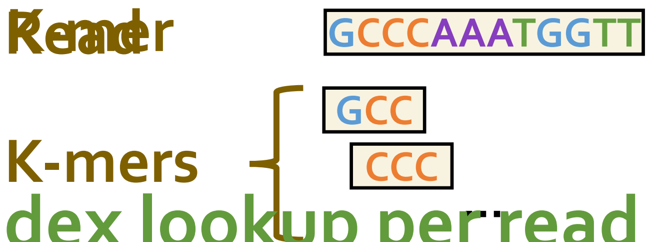
- **Challenge:** large number of random accesses per read to the reference genome and its index

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



✓ **Only one index lookup per read**

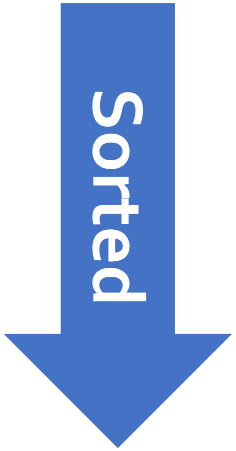
- **Sorted read-sized k-mers:** to avoid random accesses to the index

✓ **Sequential scan of the read set and the index**

GenStore-EM: Data Structures

Sorted Read Table

	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	…

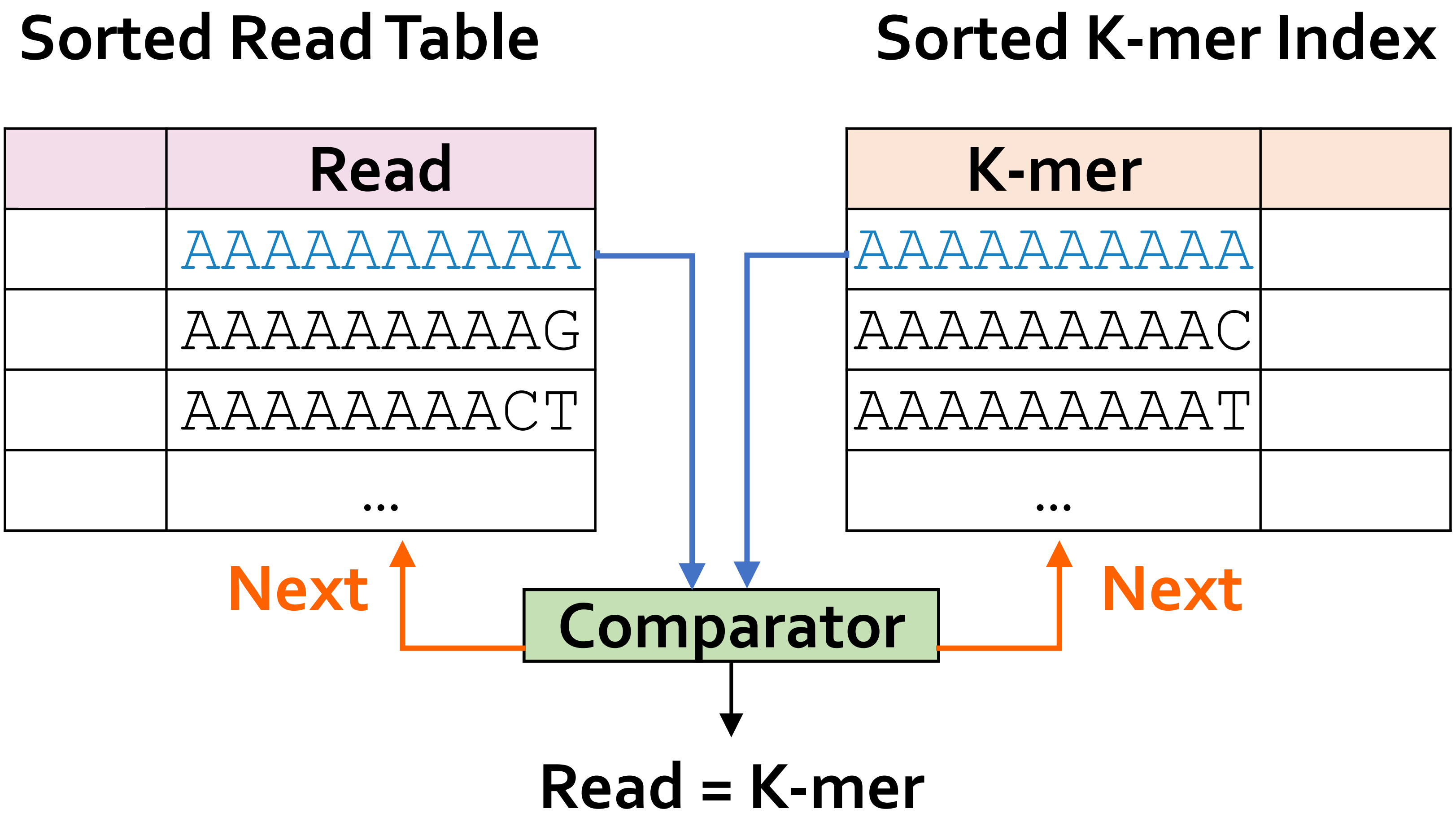


Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
…	

Read-sized K-mers

GenStore-EM: Finding a Match



Sorted Read Table

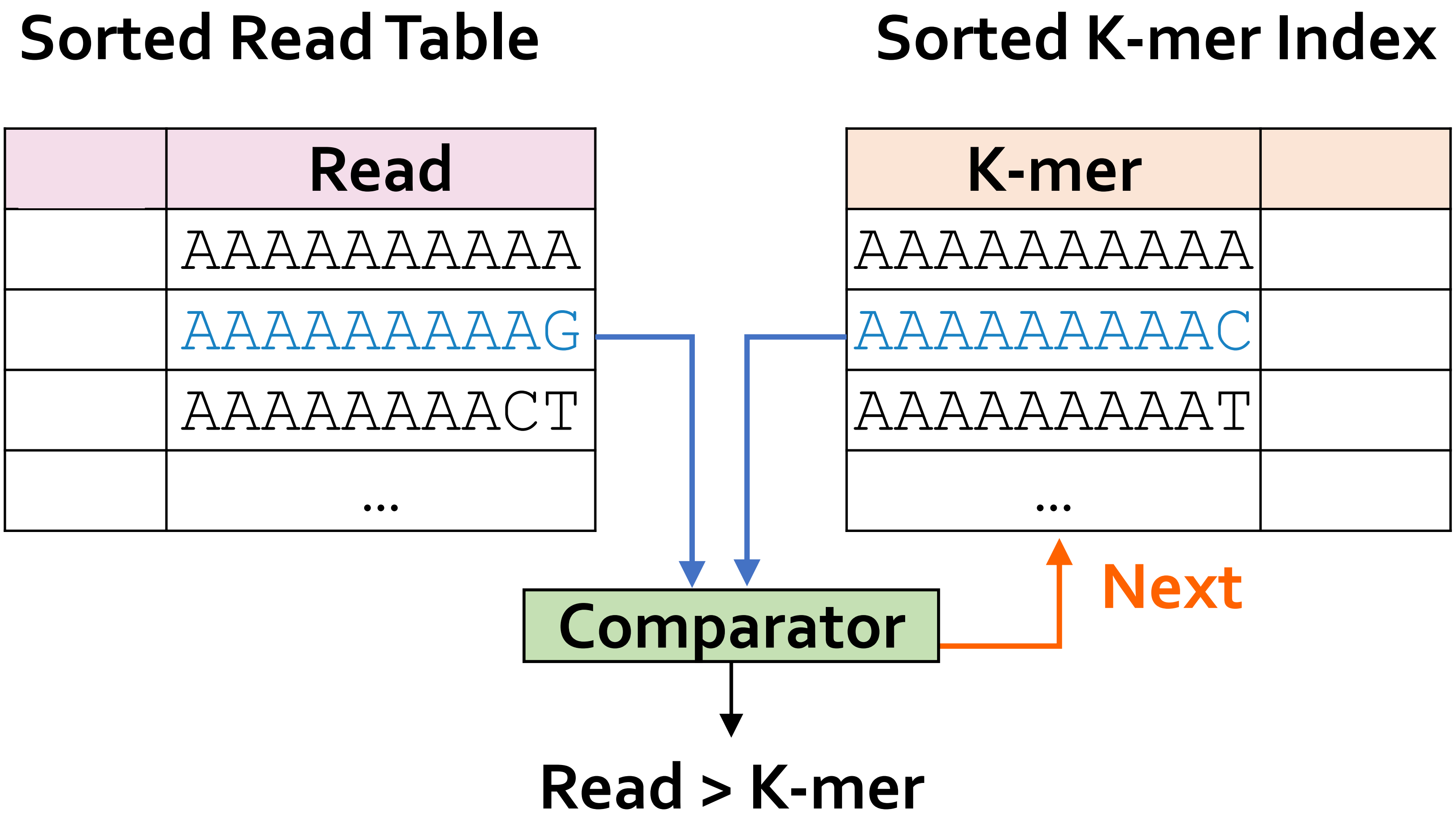
	Read
	AAAAAAAAAAA
	AAAAAAAAAAG
	AAAAAAAAACT
	…

Sorted K-mer Index

K-mer	
AAAAAAAAAAA	
AAAAAAAAAAC	
AAAAAAAAAAT	
…	

Exact match → Filter the read

GenStore-EM: Not Finding a Match



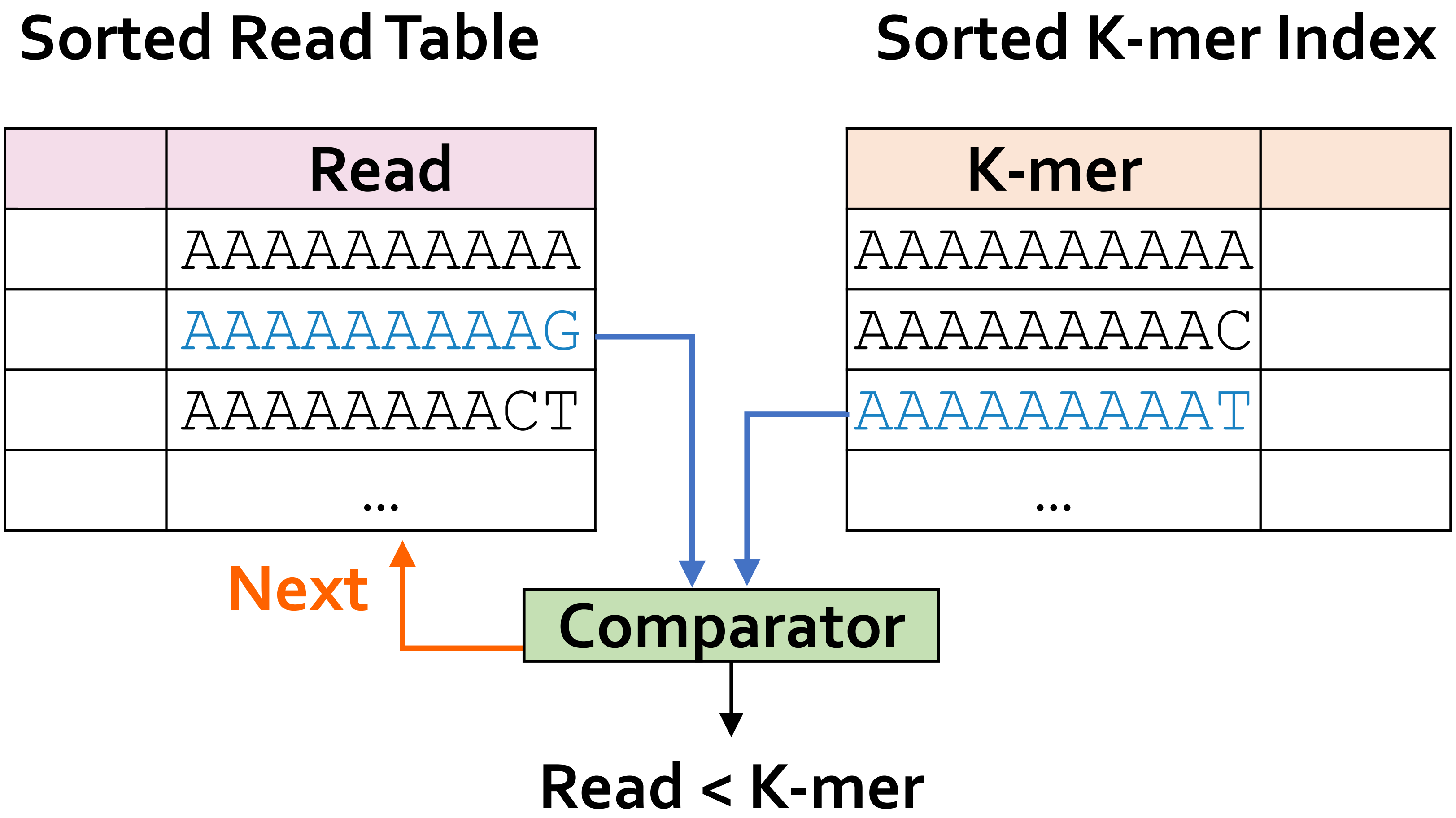
Sorted Read Table

	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
...	

GenStore-EM: Not Finding a Match



Sorted Read Table

	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	…

Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
…	

Read < K-mer

Not an exact match → Send to read mapper

GenStore-EM: Not Finding a Match

Sorted Read Table

	Read
	AAAAAAAAAA

Sorted K-mer Index

K-mer	
AAAAAAAAAA	

✓ Avoids random accesses

✓ Simple low-cost logic

Comparator

Read < K-mer

Not an exact match → Send to read mapper

SAFARI

GenStore-EM: Optimization

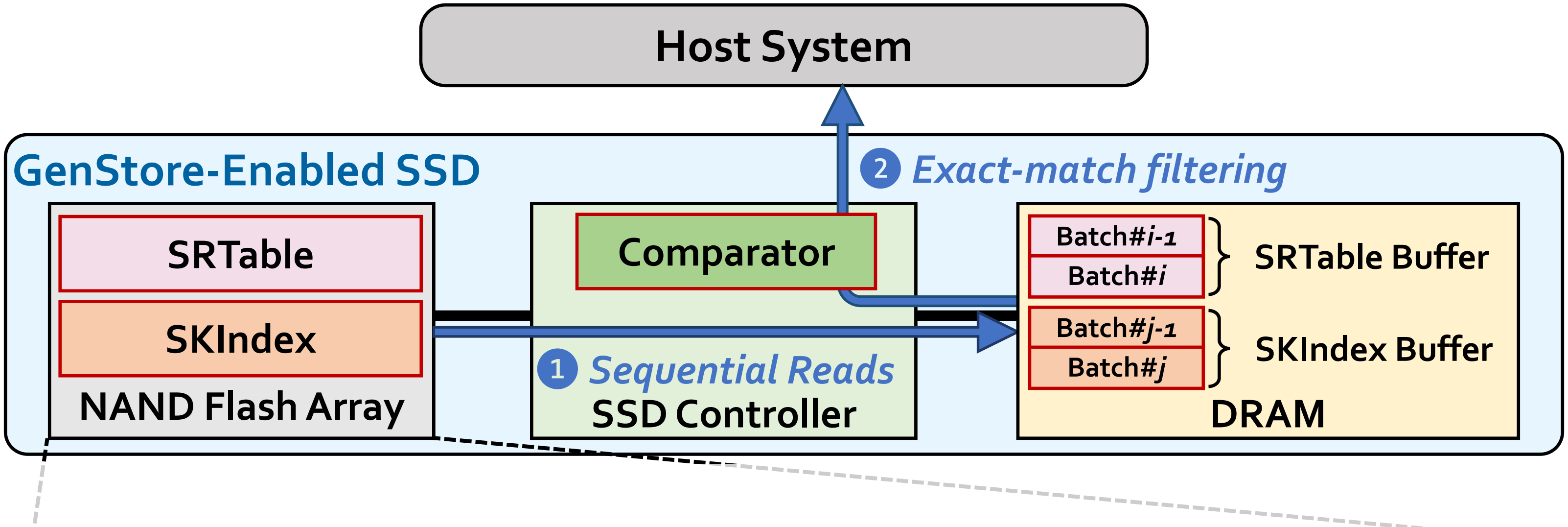
- Read-sized k-mer index takes up a large amount of space (126 GB for human index) due to the larger number of unique k-mers

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

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

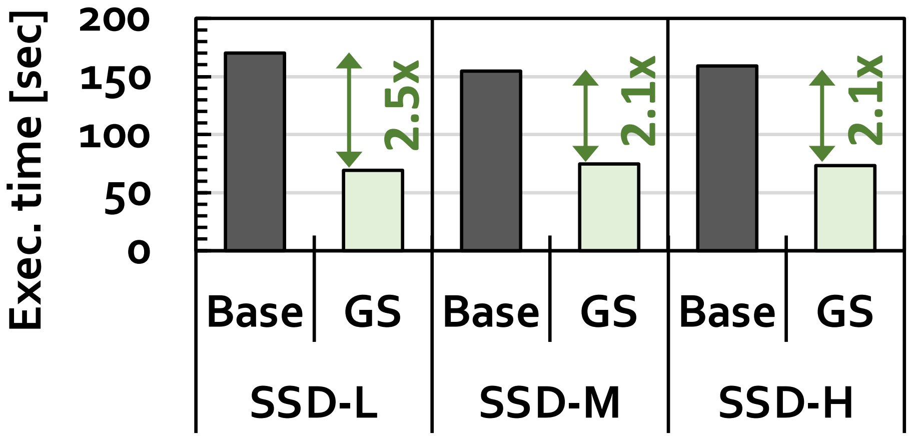
SSD Configurations

- **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)

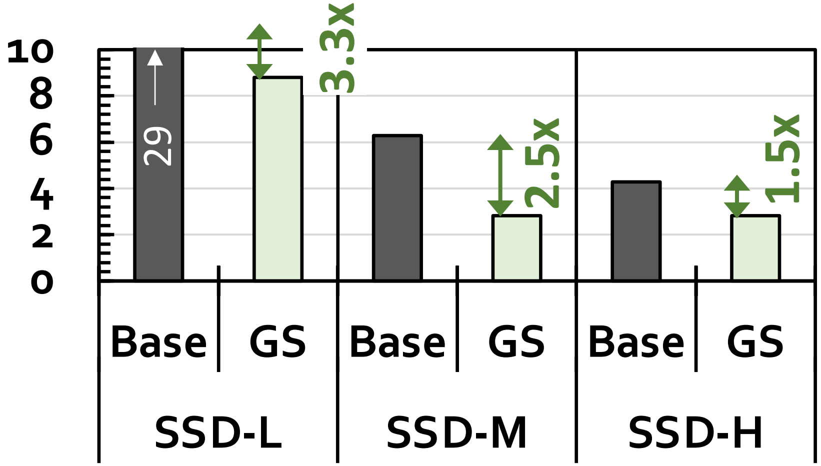
Performance – GenStore-EM

For a read set with 80% exactly-matching reads

With the Software Mapper



With the Hardware Mapper



2.1× - 2.5× speedup compared to the software Base

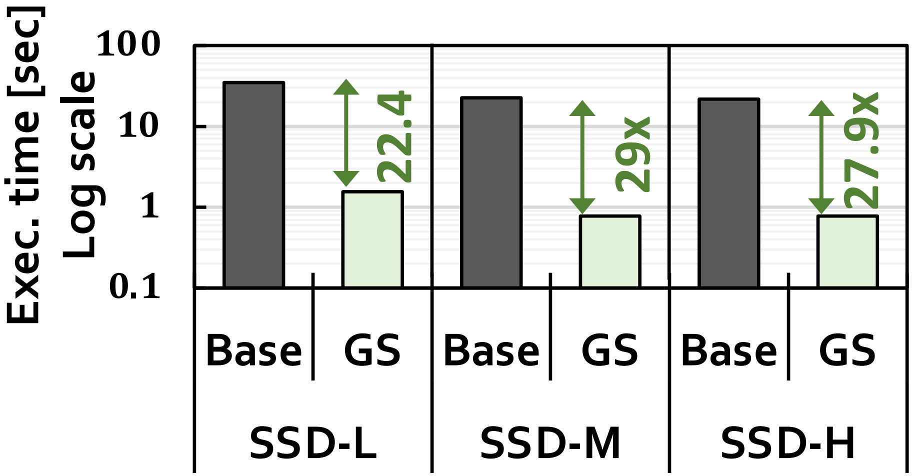
1.5× – 3.3× speedup compared to the hardware Base

On average 3.92× energy reduction

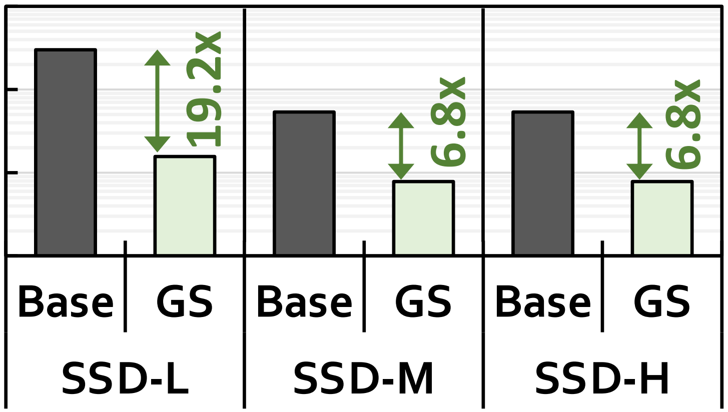
Performance – GenStore-NM

For a read set with 99.7% non-matching reads

With the Software Mapper



With the Hardware Mapper



22.4× – 27.9× speedup compared to the software Base

6.8× – 19.2× speedup compared to the hardware Base

On average 27.2× energy reduction

Area and Power

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

Logic unit	# of instances	Area [mm²]	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
Total for an 8-channel SSD	-	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

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

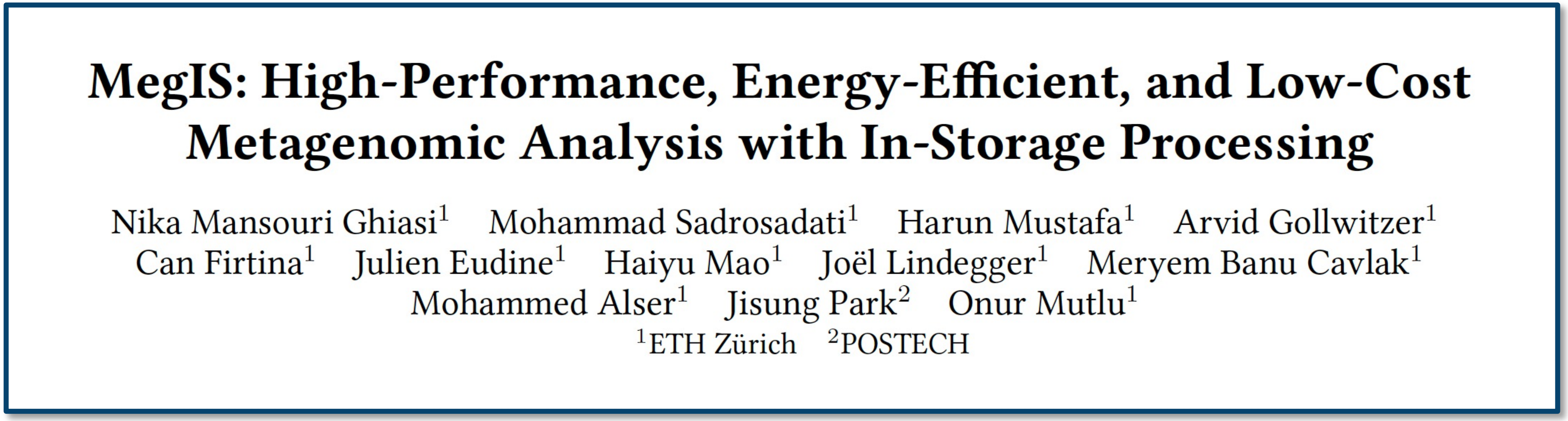
Outline

- Brief Intro to (Meta)Genomics

- *Storage-Centric Designs for (Meta)Genomics*
  - GenStore
  - *MegIS*

- Conclusion

MegIS *[ISCA'24]*



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

Nika Mansouri Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

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

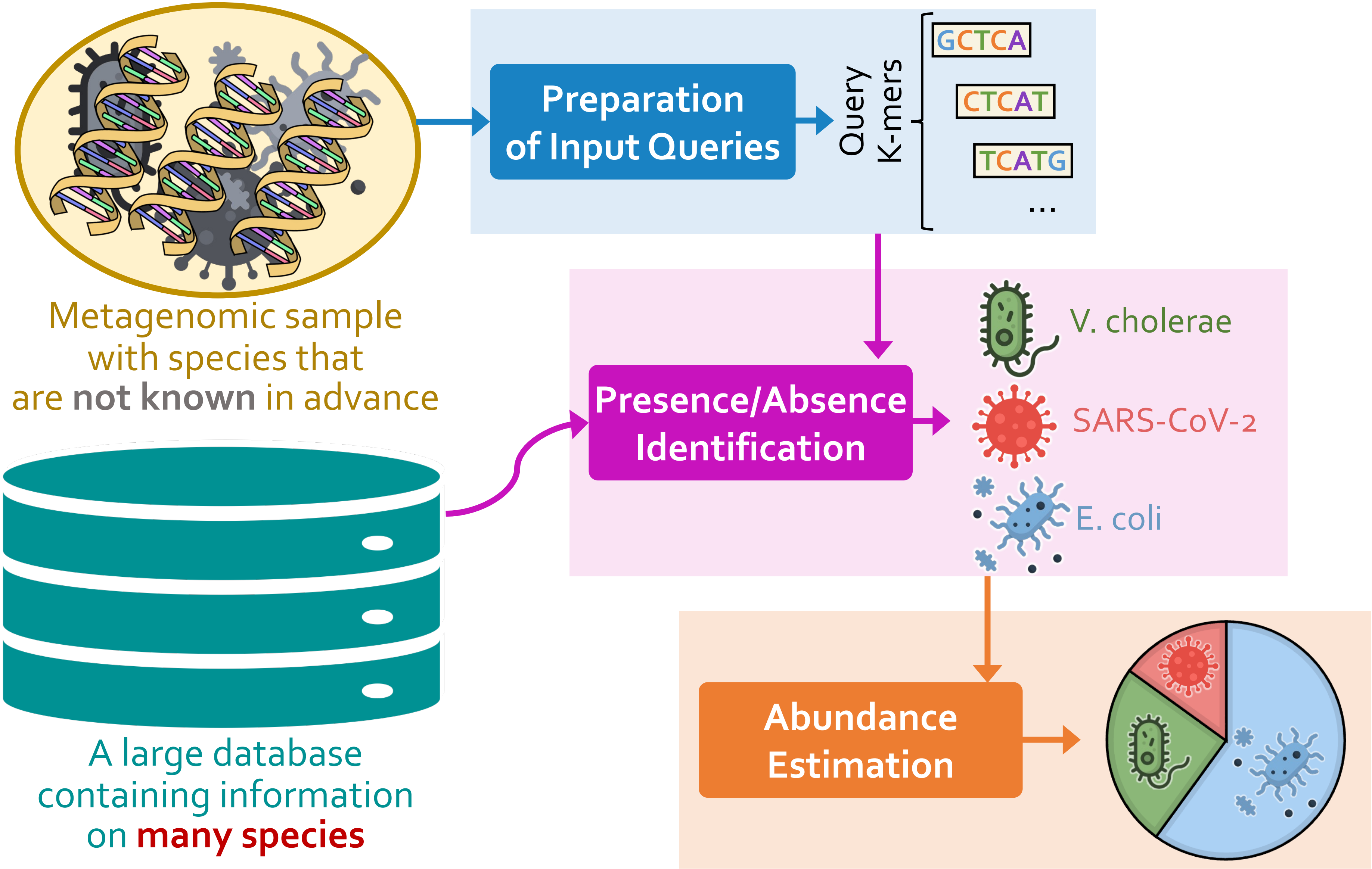
Motivation and Goal

MegIS

Evaluation

Conclusion

Metagenomic Analysis



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

Preparation of Input Queries

Presence/Absence Identification

Abundance Estimation

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

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

Outline

Background

Motivation and Goal

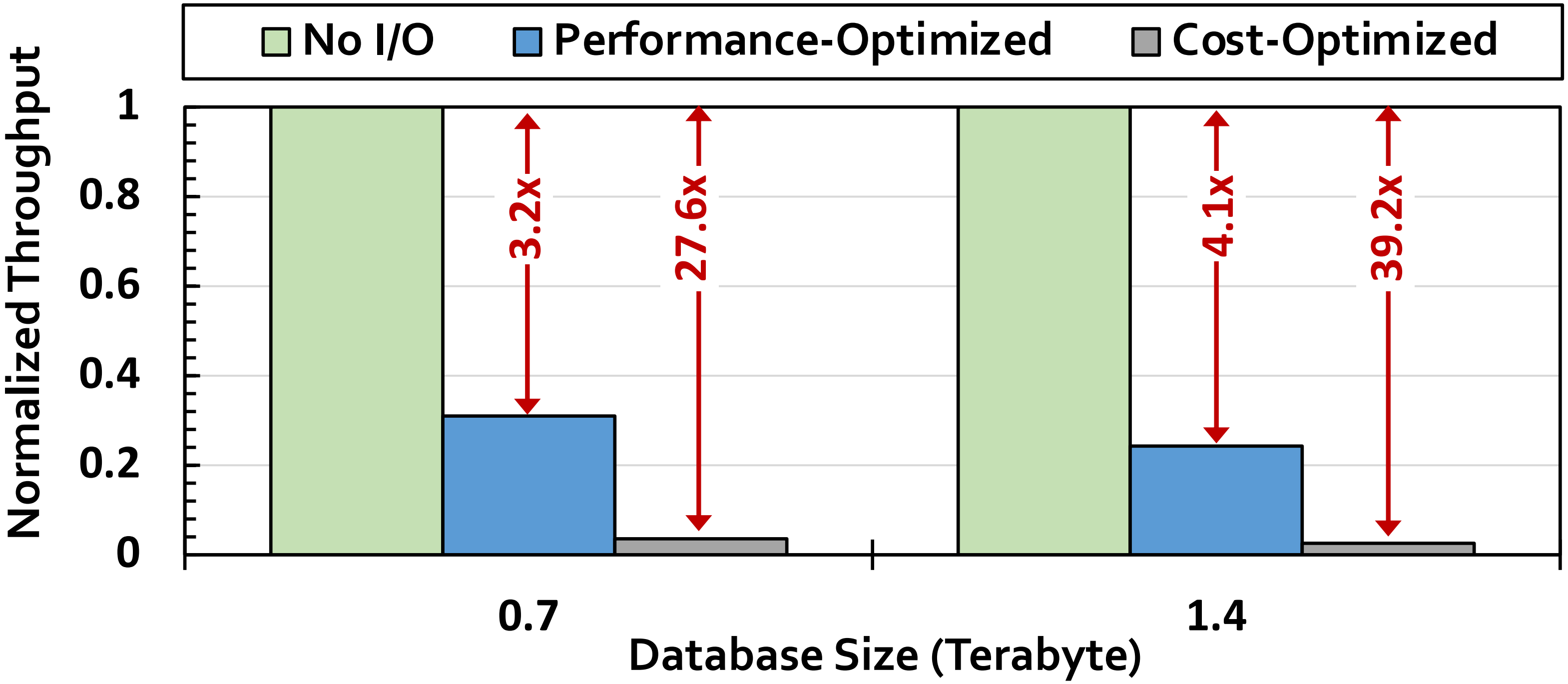
MegIS

Evaluation

Conclusion

Motivation

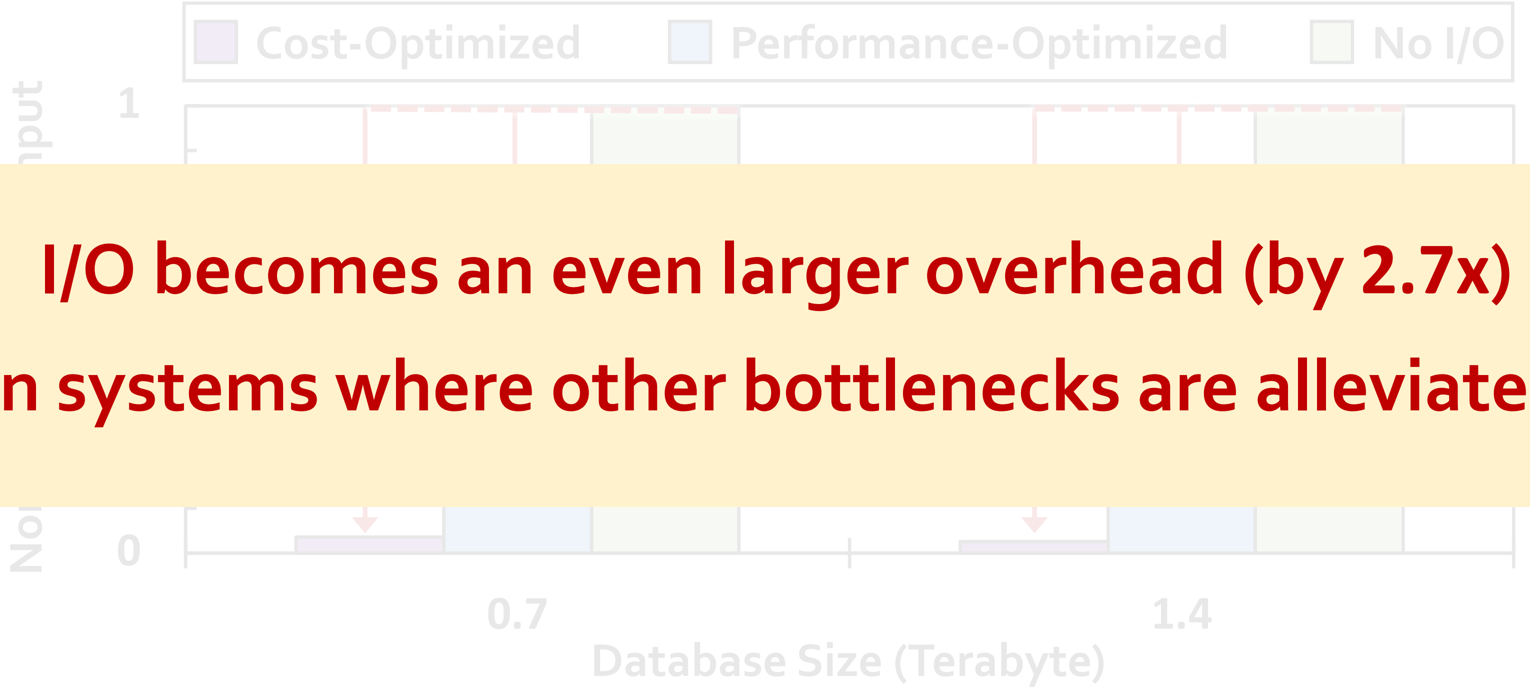
- Case study of the performance of metagenomic analysis tools
- With various state-of-the-art SSD configurations



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**

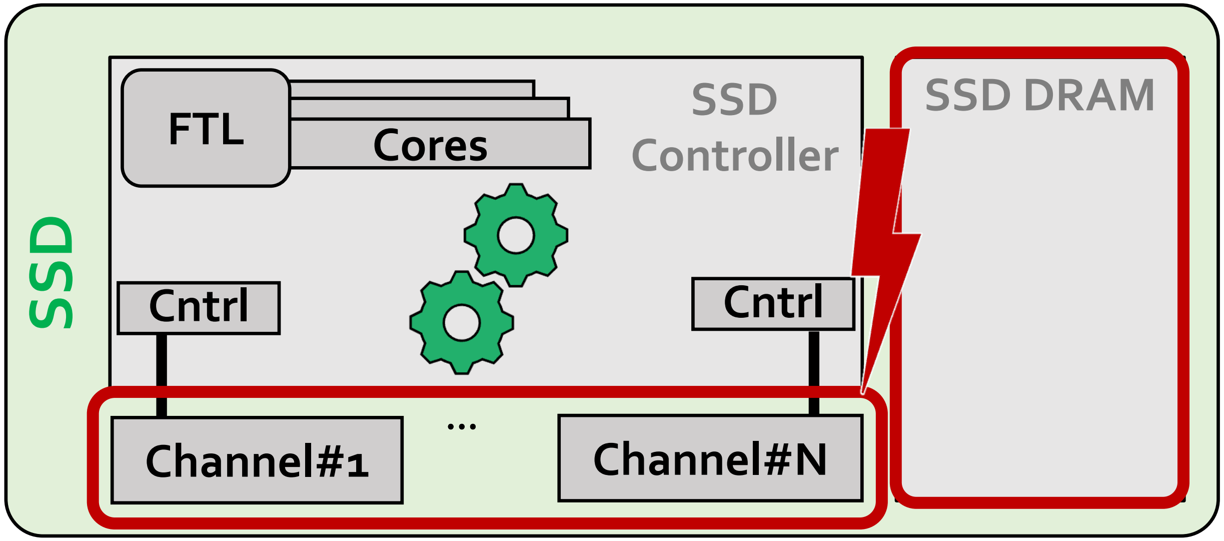
Our Goal

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 tools 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



SAFARI

Outline

Background

Motivation and Goal

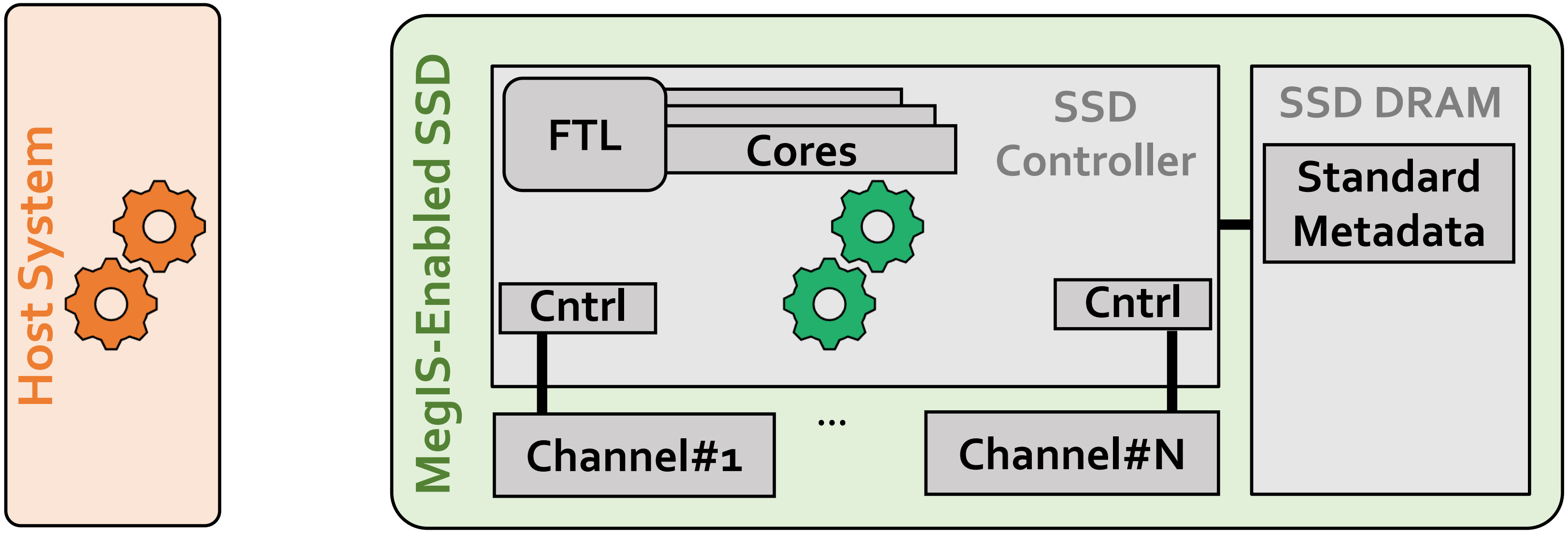
MegIS

Evaluation

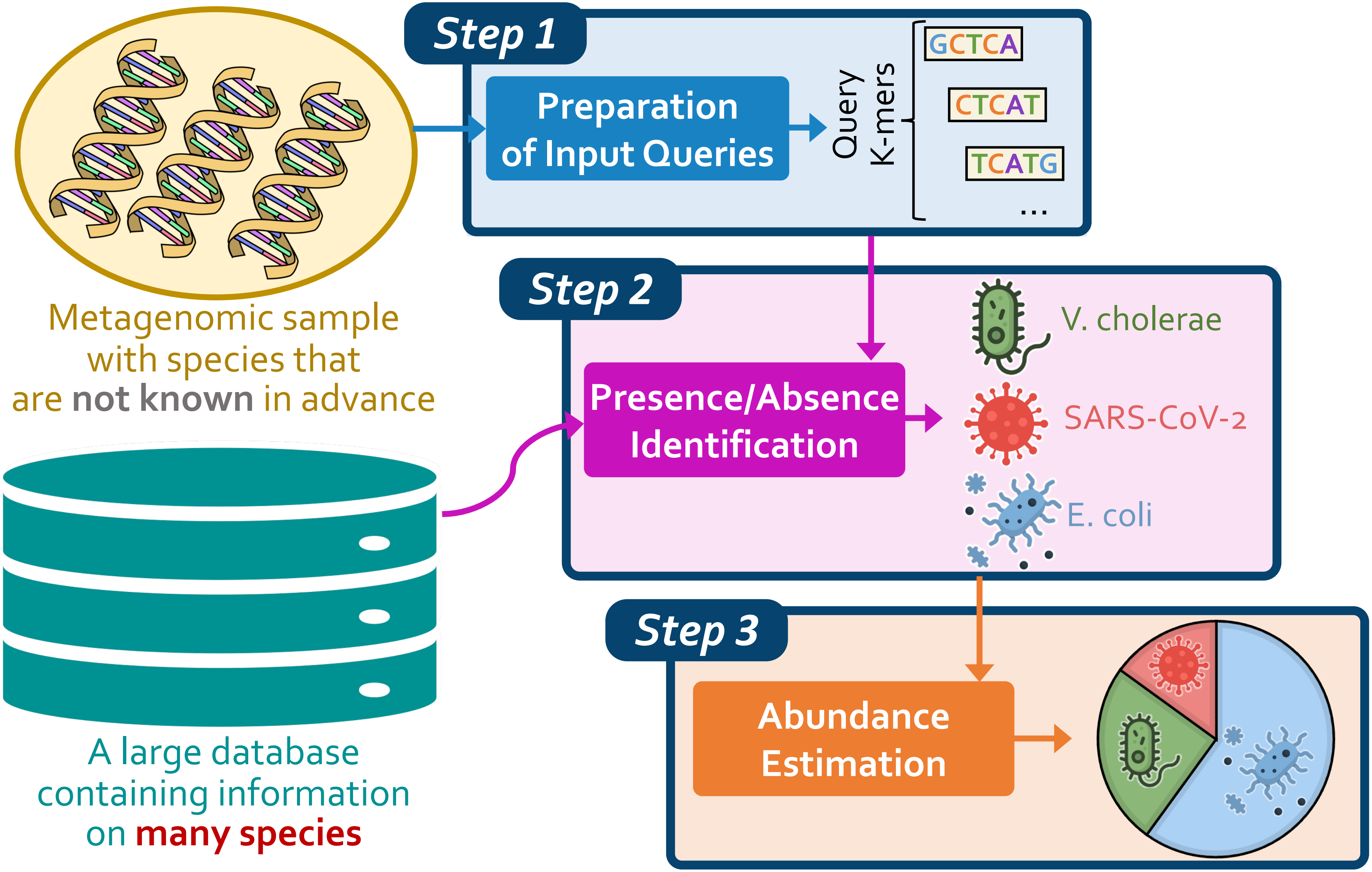
Conclusion

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**

Step 1

Preparation of Input Queries

Query K-mers

GCTCA

CTCAT

TCATG

...

Step 2

Presence/Absence Identification

V. cholerae

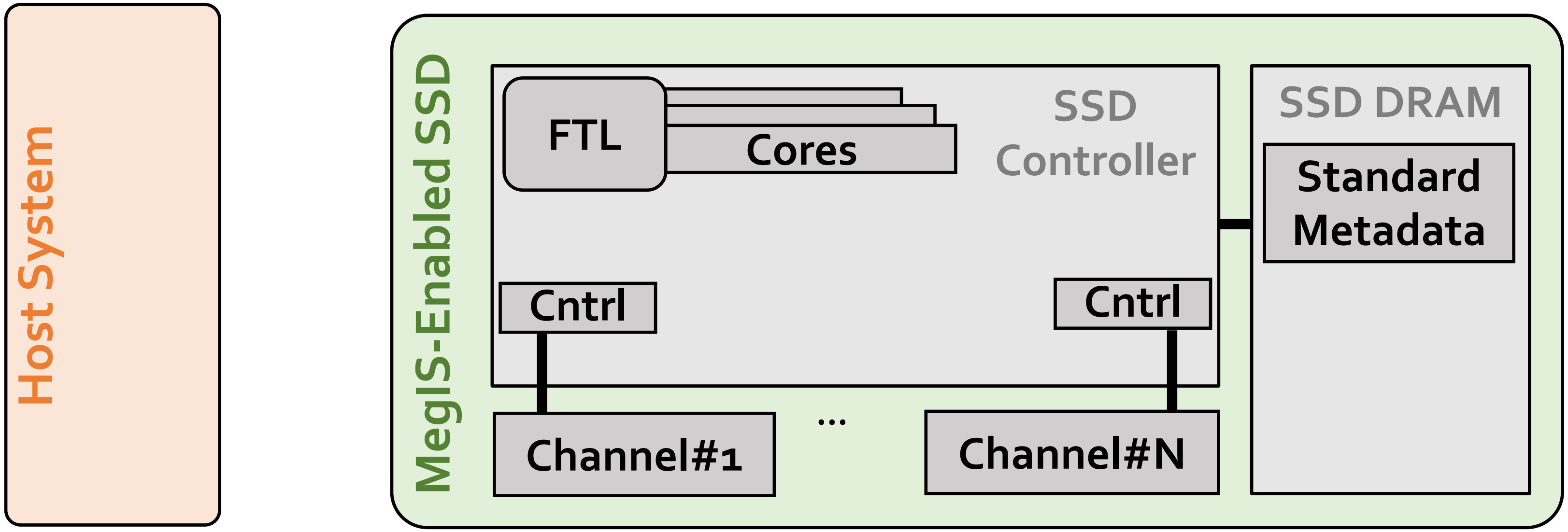
SARS-CoV-2

E. coli

Step 3

Abundance Estimation

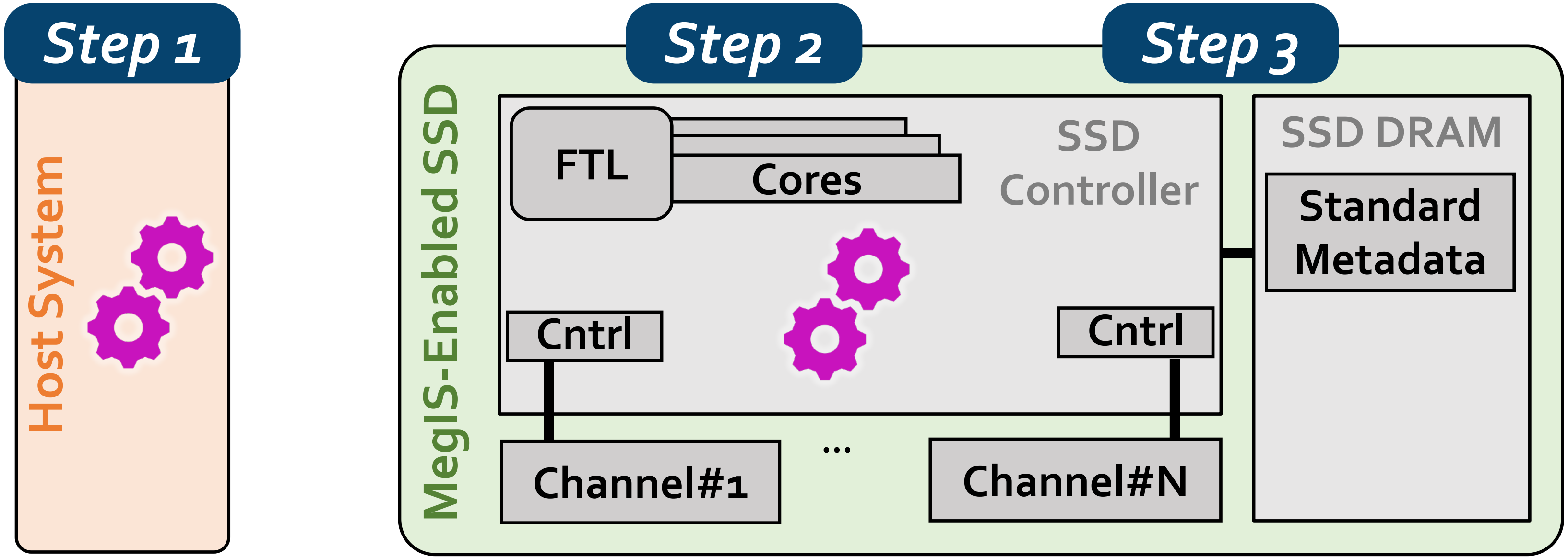
MegIS Hardware-Software Co-Design



MegIS Hardware-Software Co-Design

Task partitioning and mapping

- *Each step executes in its most suitable system*



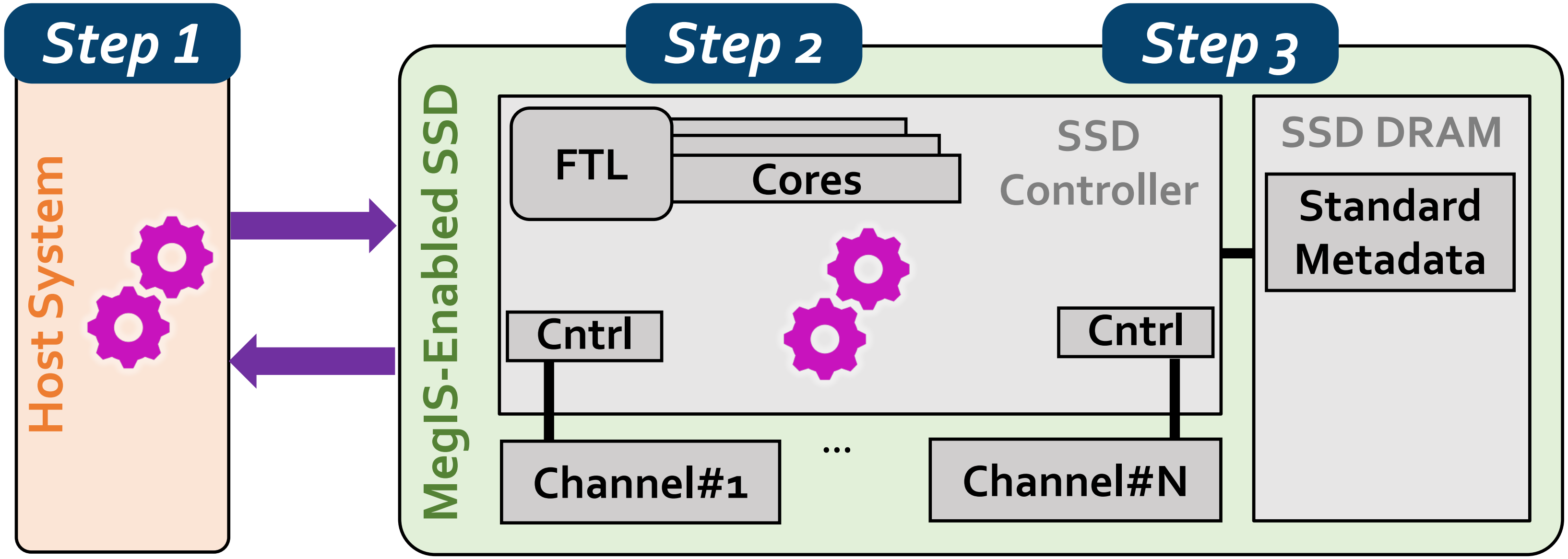
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- *Each step executes in its most suitable system*

Data/computation flow coordination

- *Reduce communication overhead*
- *Reduce #writes to flash chips*



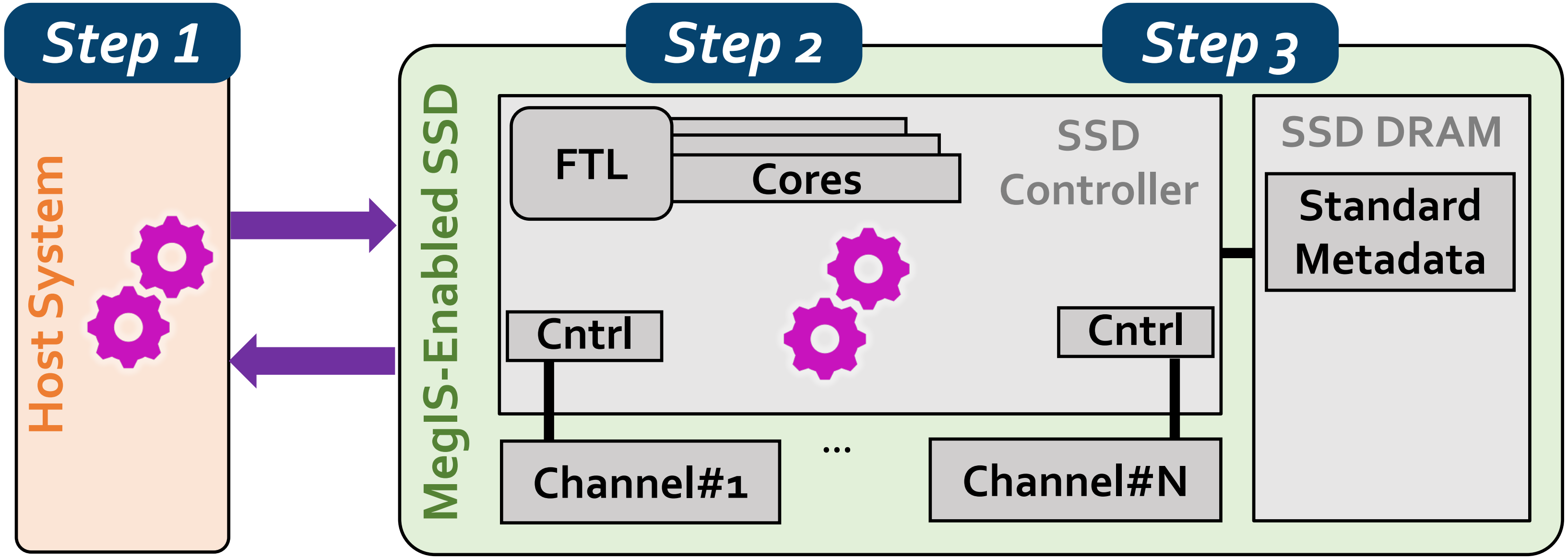
MegIS Hardware-Software Co-Design

Task partitioning and mapping

- *Each step executes in its most suitable system*

Data/computation flow coordination

- *Reduce communication overhead*
- *Reduce #writes to flash chips*



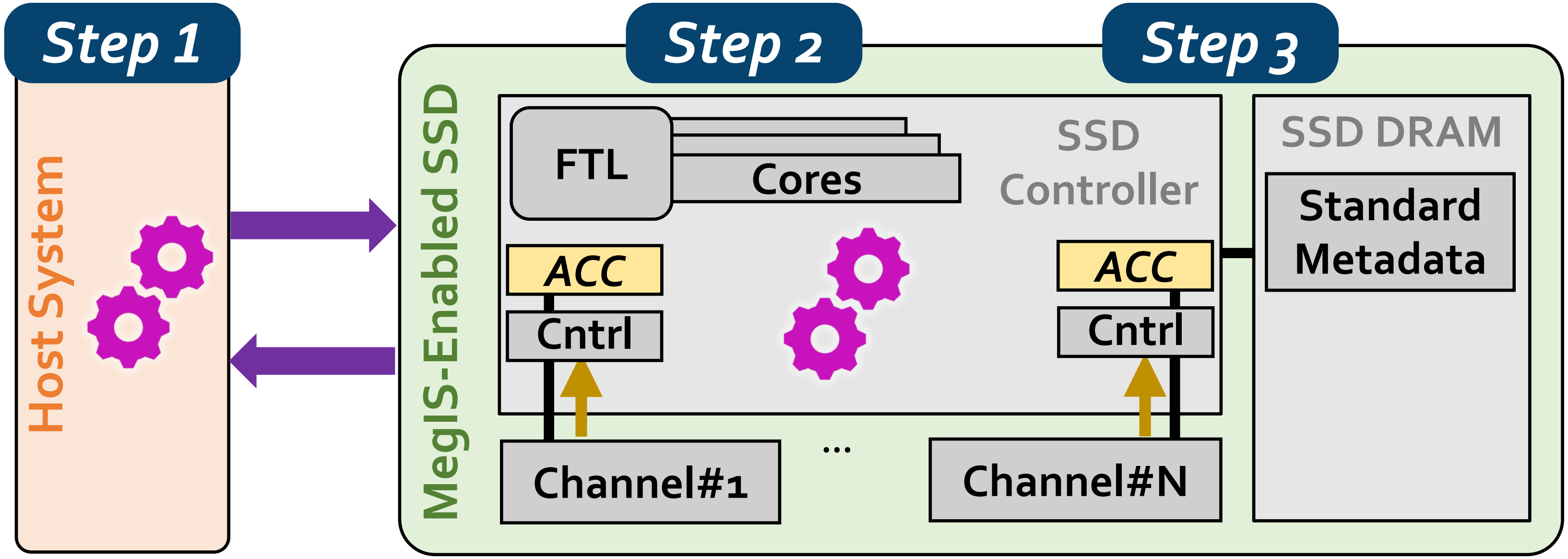
Storage-aware algorithms

- *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*

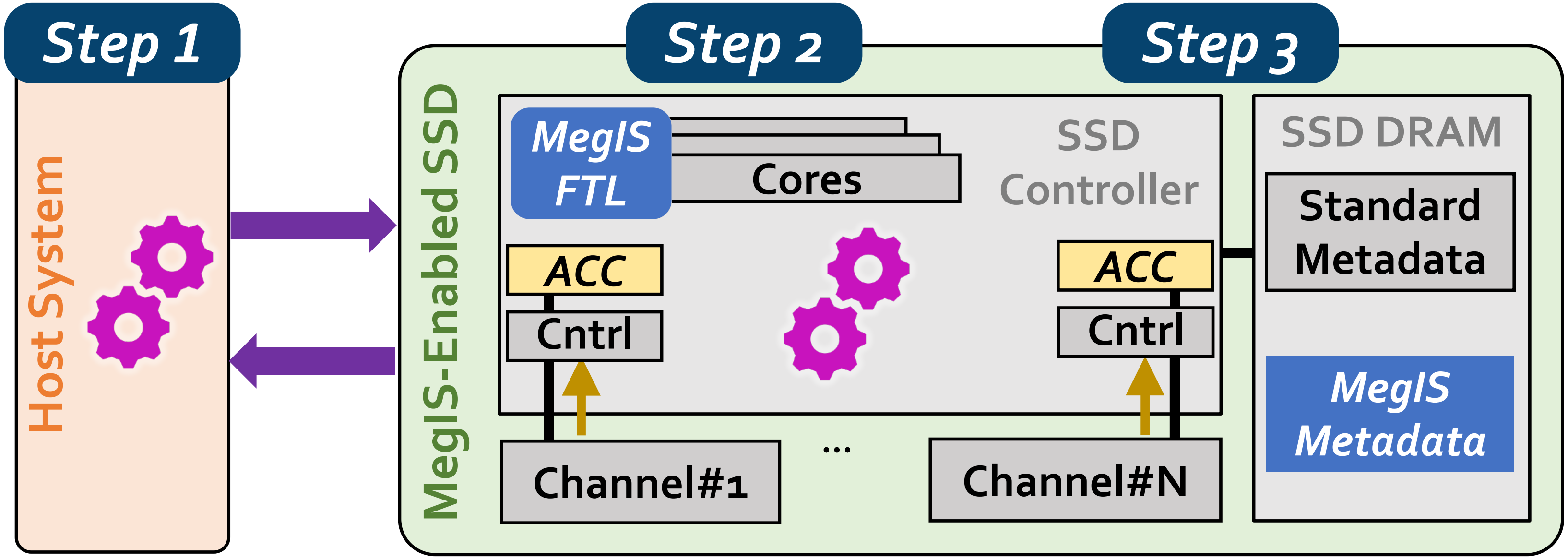
MegIS Hardware-Software Co-Design

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*

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)

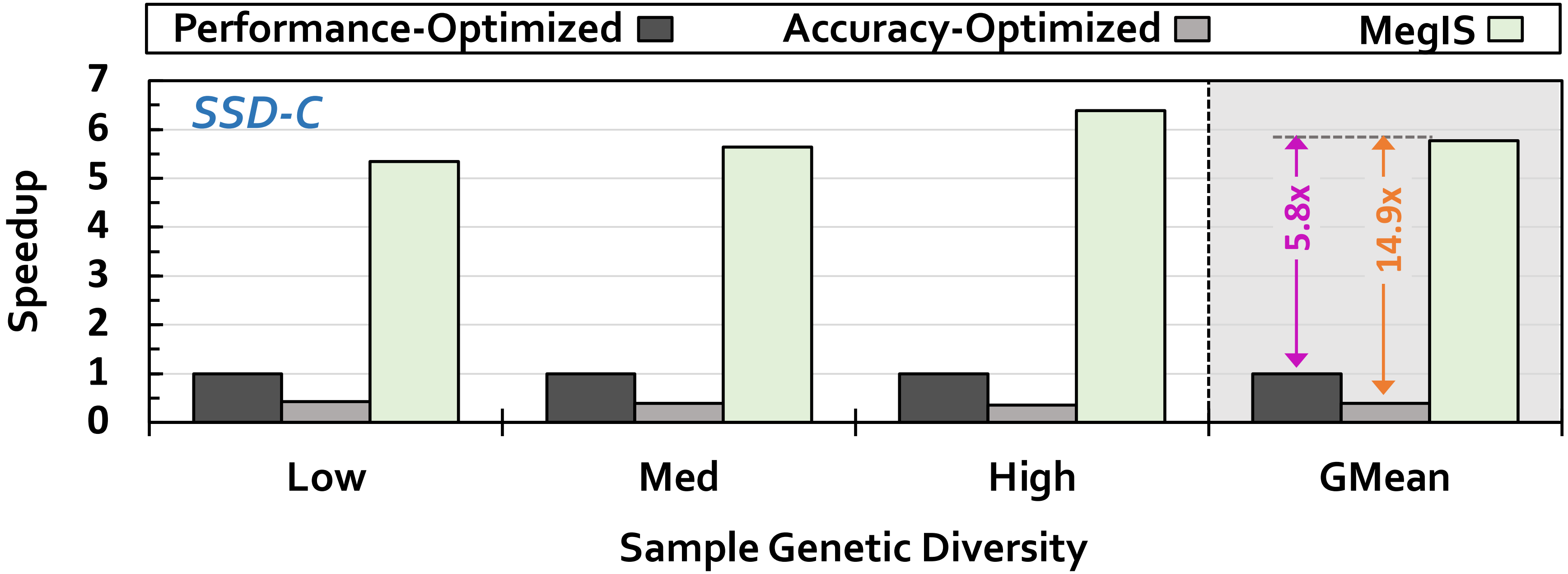
Metagenomic Analysis Task

- 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

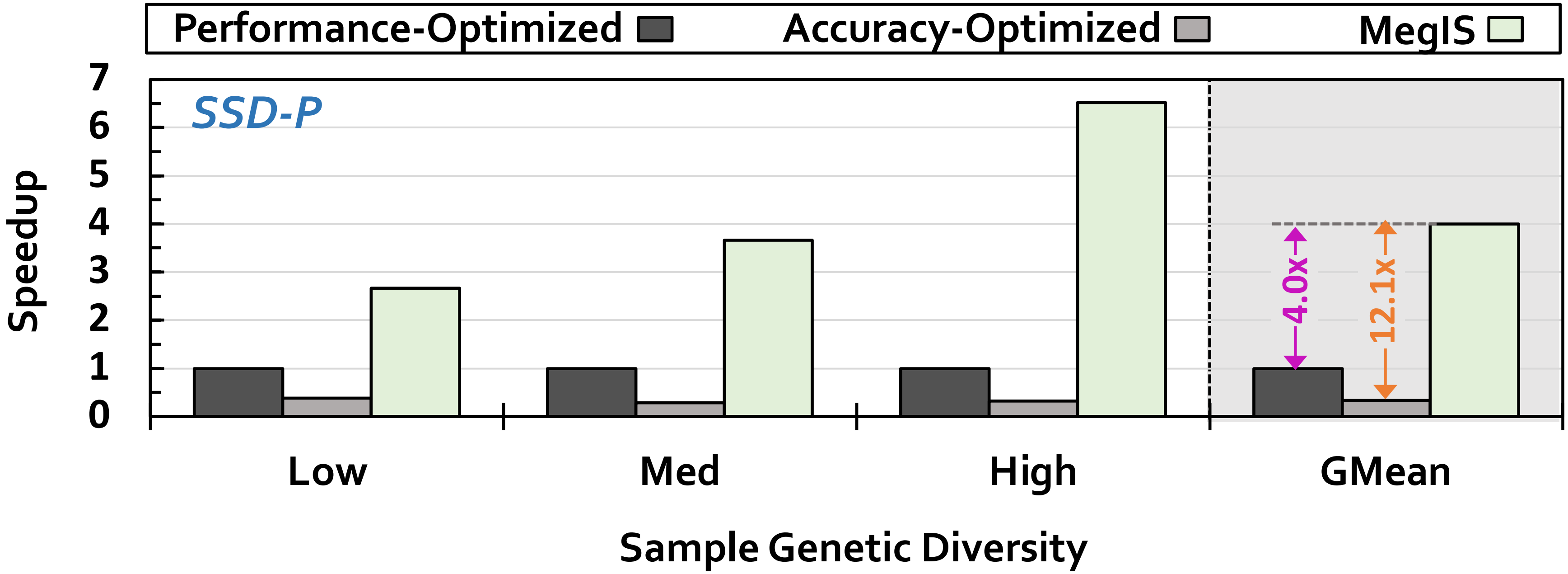
Speedup over Software (with Cost-Optimized SSD)



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

SAFARI

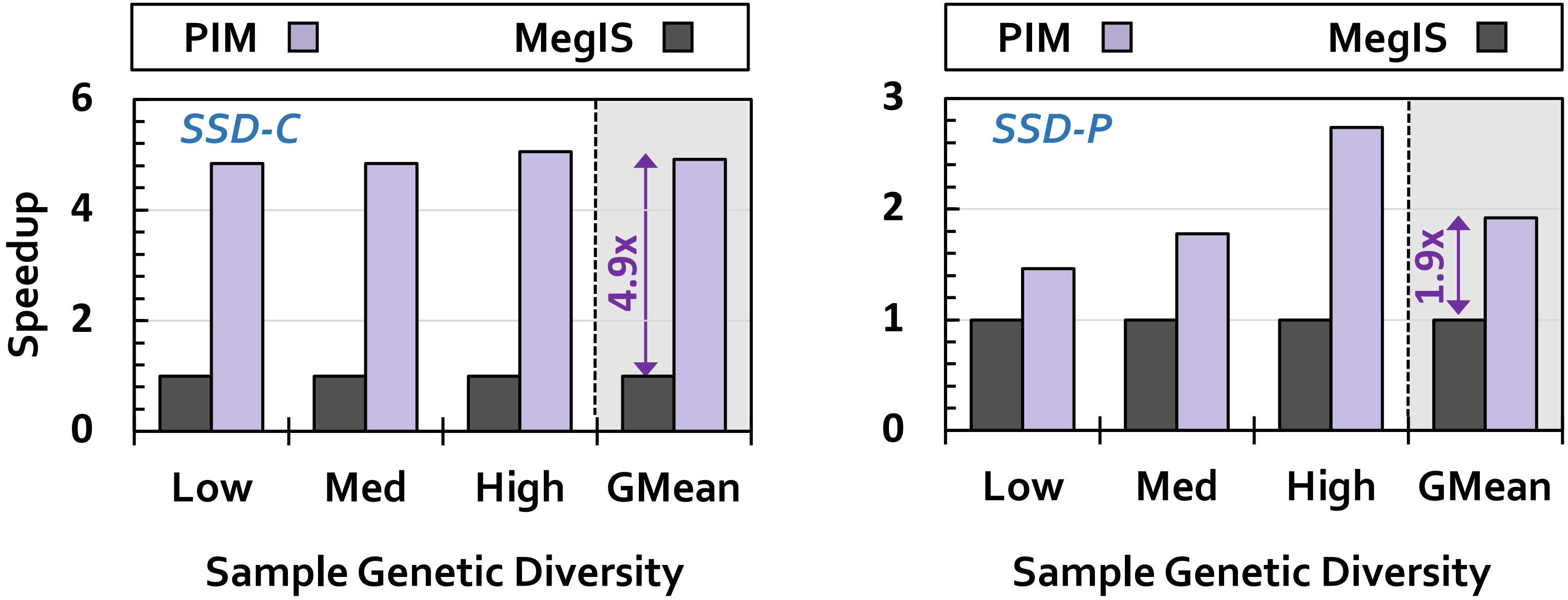
Speedup over Software (with Performance-Optimized SSD)



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

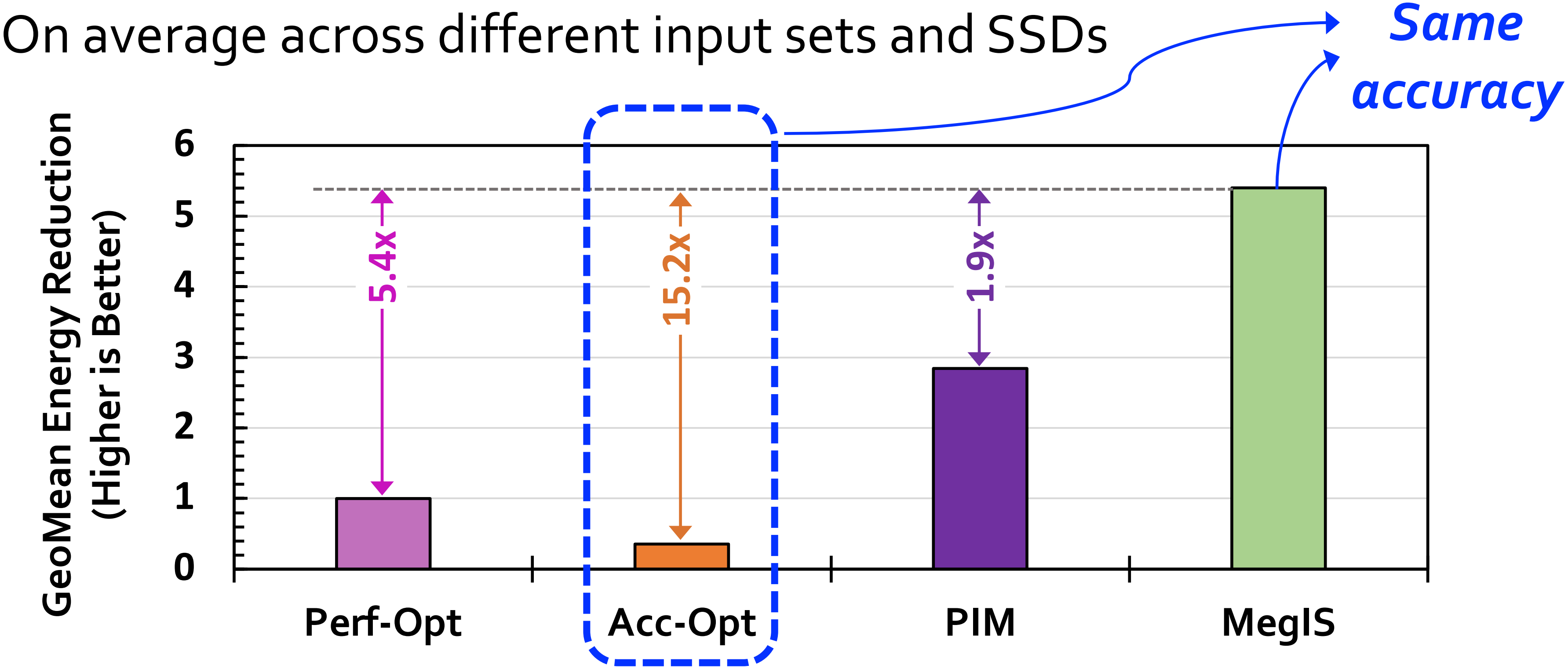


MegIS provides significant speedup over the PIM baseline

SAFARI

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:

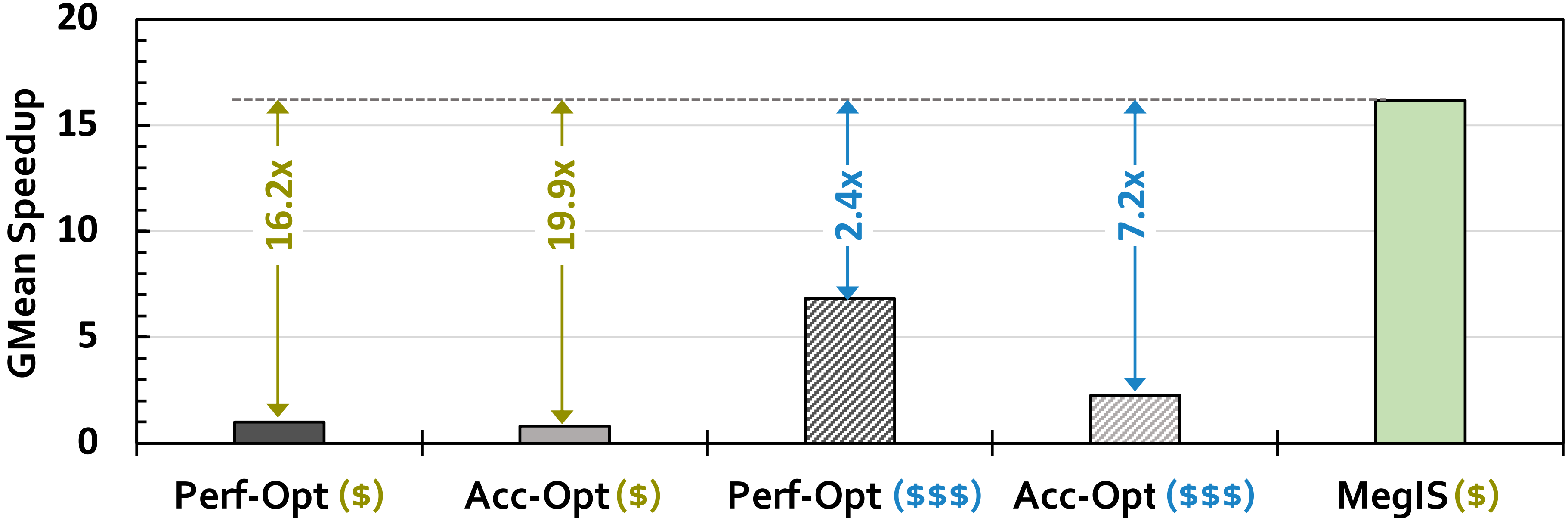
- **Area:** 0.04 mm^2
- **Power:** 7.658 mW

*(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**

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**

MegIS

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

https://arxiv.org/abs/2406.19113

SAFARI

ETH zürich

POSTECH

Outline

- **Brief Intro to (Meta)Genomics**

- **Storage-Centric Designs for (Meta)Genomics**
  - **GenStore**
  - **MegIS**

- ***Conclusion***

Specializing the Storage System for Genomics & Metagenomics Can Provide Large Benefits

Specializing the Storage System for Genomics & Metagenomics

Storage-centric designs

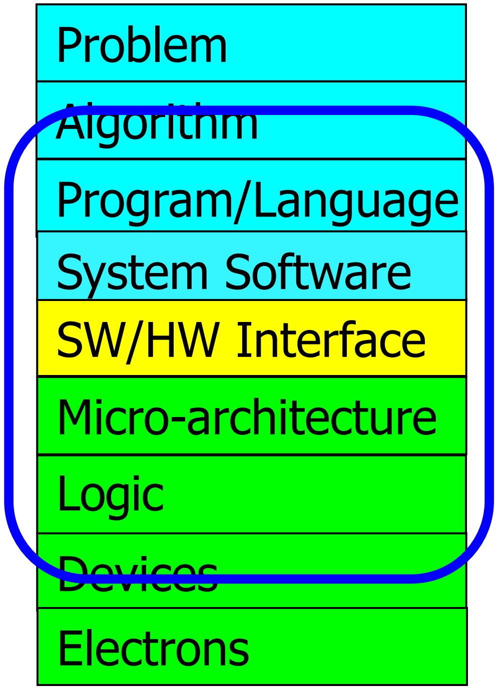
improve system cost-efficiency

and makes accurate (meta)genomics

more accessible for wider adoption

(Co-)Optimizing Algorithm-Architecture-Device is Critical

Computer Architecture (Expanded View)



Problem

Algorithm

Program/Language

System Software

SW/HW Interface

Micro-architecture

Logic

Devices

Electrons

More About My Research

My Website:

https://bit.ly/nikamgh

Works Described in This Talk	**GenStore** ASPLOS'22	**MegIS** ISCA'24			
Near-Data Processing (Other Works)	**ALP** IEEE TETC'22	**CODIC** ISCA'21	**SIMDRAM** ASPLOS'21		
Optimizing Memory and Storage Systems	**Venice** ISCA'23	**FIGARO** MICRO'20	**CROW** ISCA'19	**CAL** MICRO'18	**FLIN** ISCA'18
Algorithms	**MLA** ISMB'24	**RawHash** ISMB'23	**BLEND** Bioinformatics'23	**TargetCall** APBC'23	
Algorithm-Architecture Co-Design	**Scrooge** Bioinformatics'23	**SeGraM** ISCA'22	**SMASH** MICRO'19		
Device-Architecture Co-Design	**Understanding and Modeling Ultra-Dense 3D Memory Systems** PACT SRC'24				

Storage-Centric Computing for Genomics and Metagenomics

Nika Mansouri Ghiasi

n.mansorighiasi@gmail.com

ETH zürich

SAFARI

Backup Slides

End-to-End Workflow of Genome Sequence Analysis

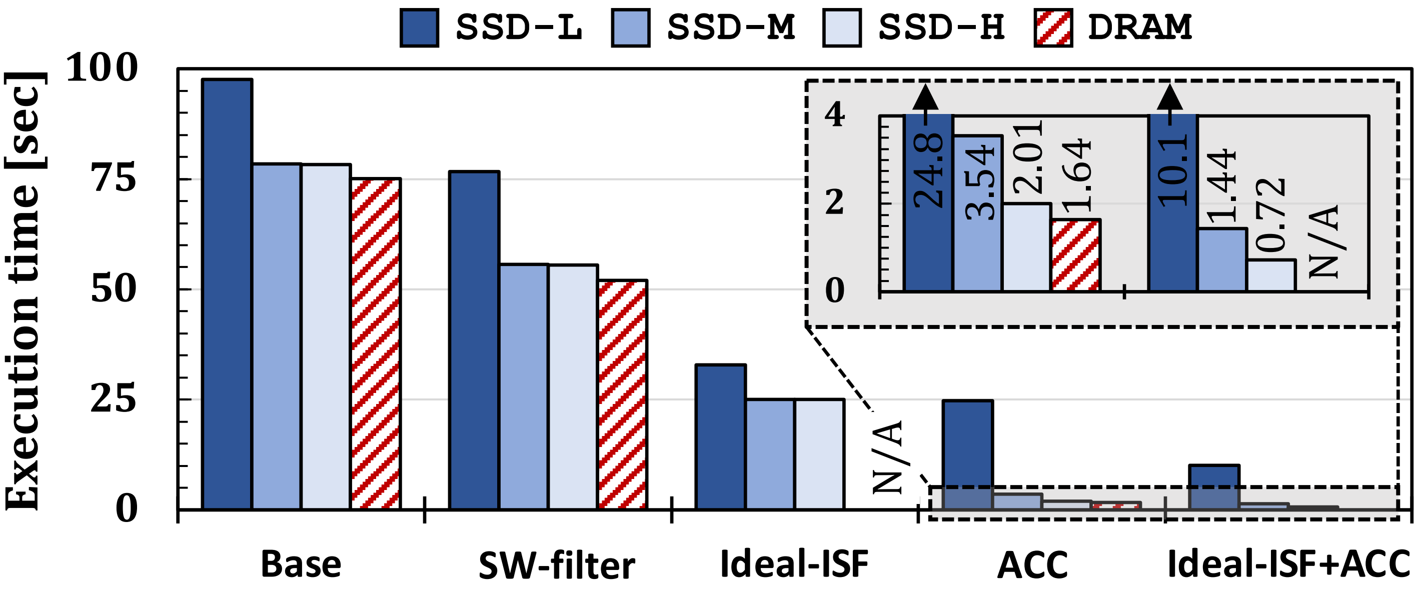
- There are three key initial steps in a standard genome sequencing and analysis workflow
  - Collection, preparation, and sequencing of a DNA sample in the laboratory
  - Basecalling
  - Read mapping

- 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

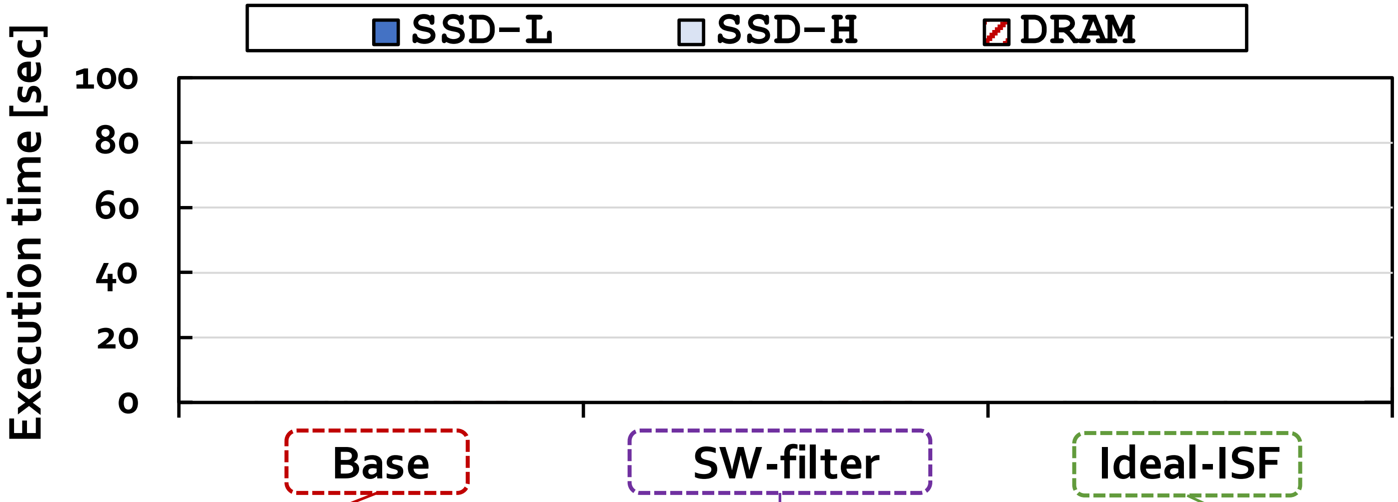
- We focus on optimizing the performance of read mapping because sequencing and basecalling are performed only once per read set, whereas read mapping can be performed many times
  - Analyzing the differences between a reads from an individual and many reference genomes of other individuals
  - Repeating 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


SSD-L
SSD-M
SSD-H
DRAM
Execution time [sec]
100
75
50
25
0
Base
SW-filter
Ideal-ISF
ACC
Ideal-ISF+ACC
N/A
4
2
0
24.8
3.54
2.01
1.64
10.1
1.44
0.72
N/A

Motivation

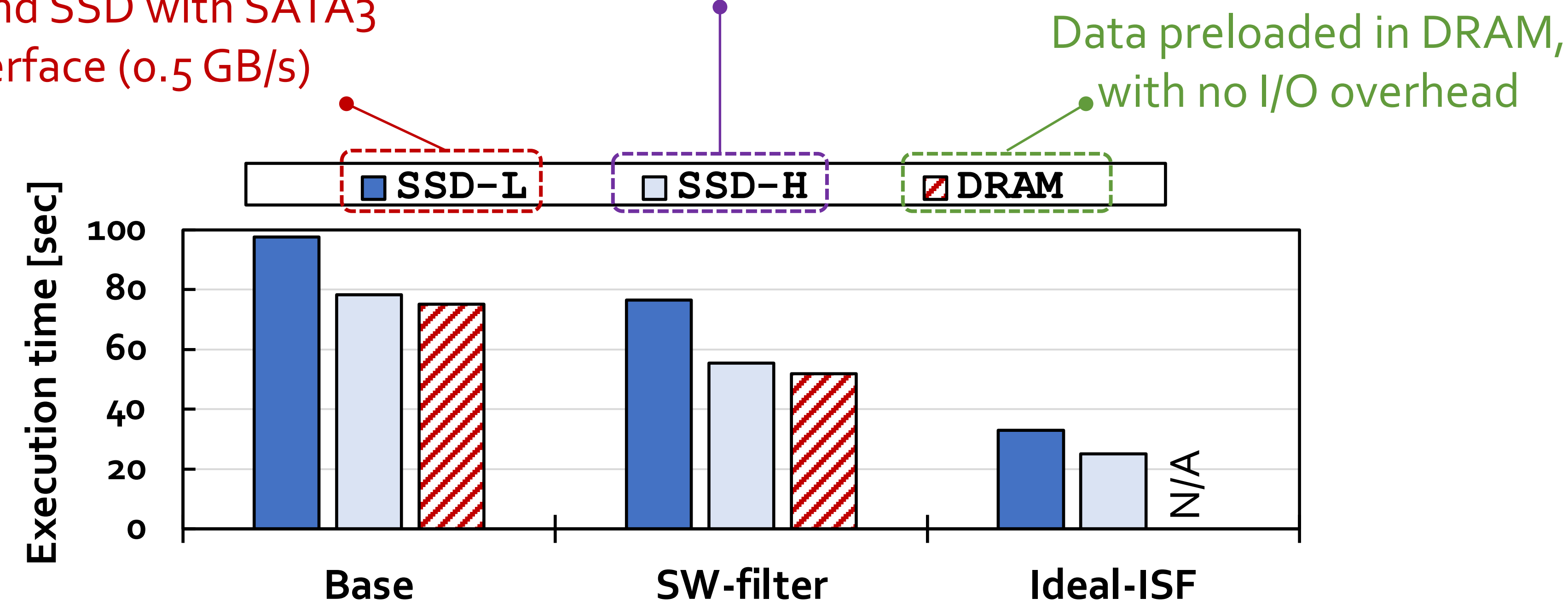


State-of-the-art software read mapper, Minimap2

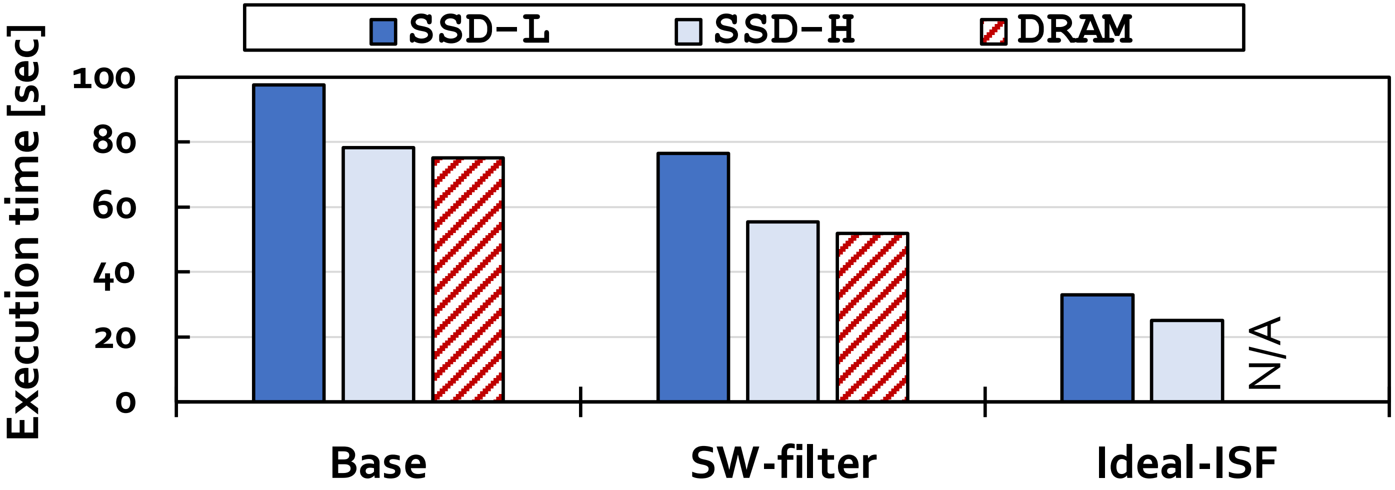
Base integrated with an ideal in-storage filter

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

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
SSD-L
SSD-H
DRAM
Execution time [sec]
100
80
60
40
20
0
Base
SW-filter
Ideal-ISF
N/A

Benefits of Ideal In-Storage Filter

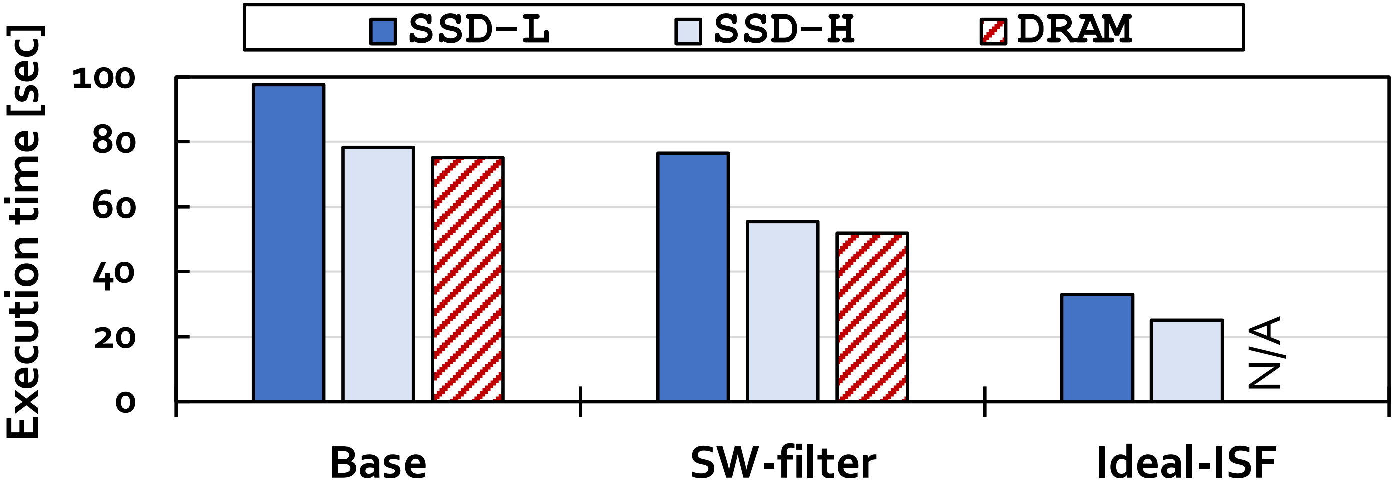


The ideal in-storage filter significantly improves performance by

1) Reducing computation overhead

2) Reducing data movement overhead

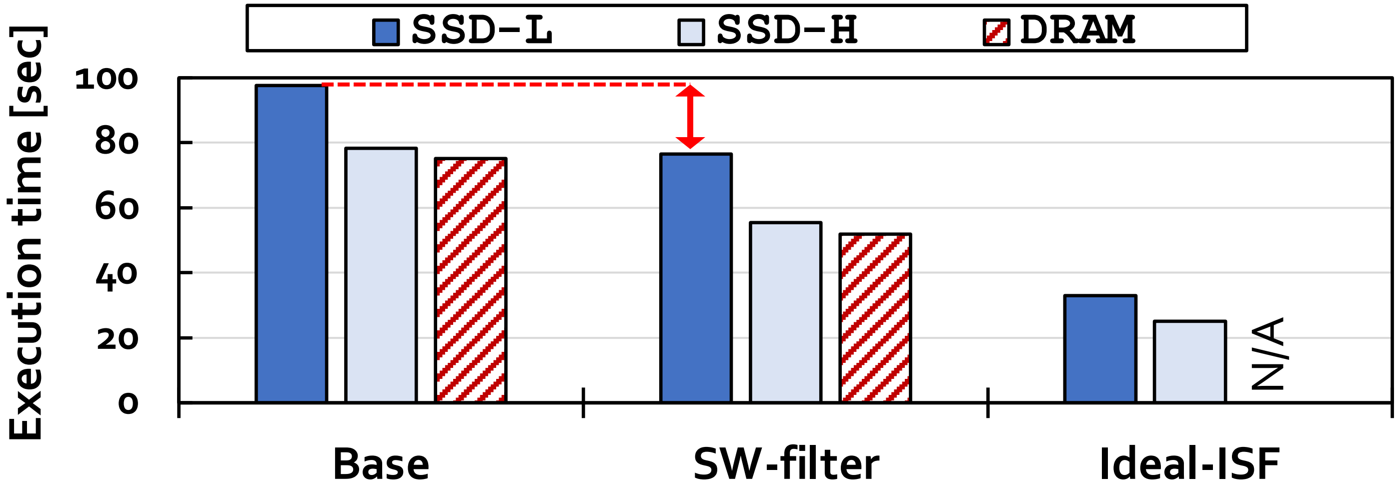
Overheads of Software Mappers



I/O has a significant impact on application performance

which can be alleviated at the cost of
expensive storage devices and interfaces

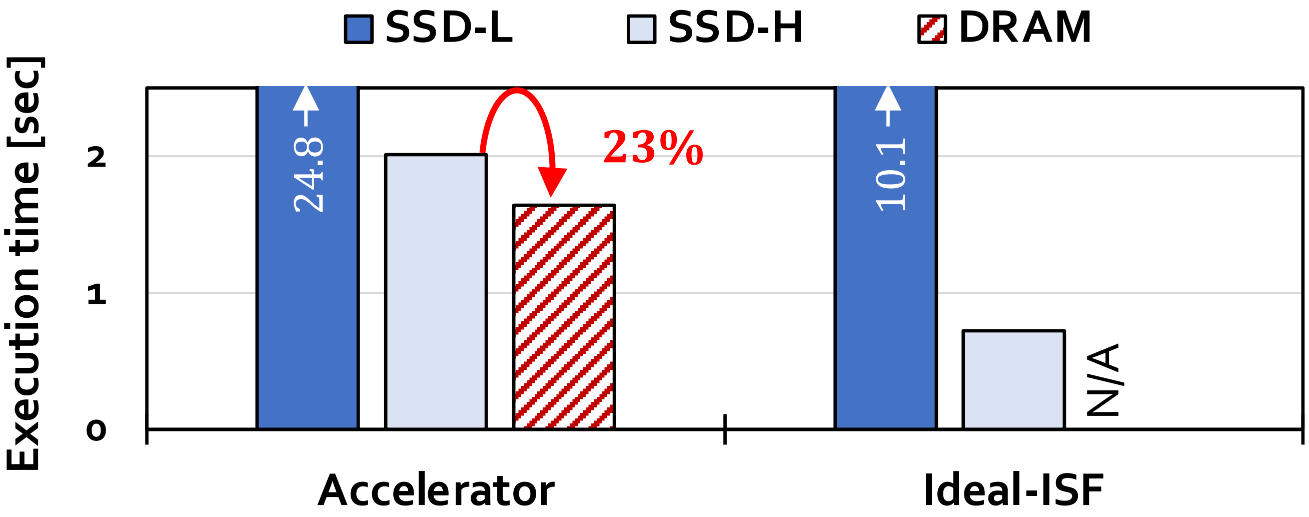
Overheads of Software Mappers



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:

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

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

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

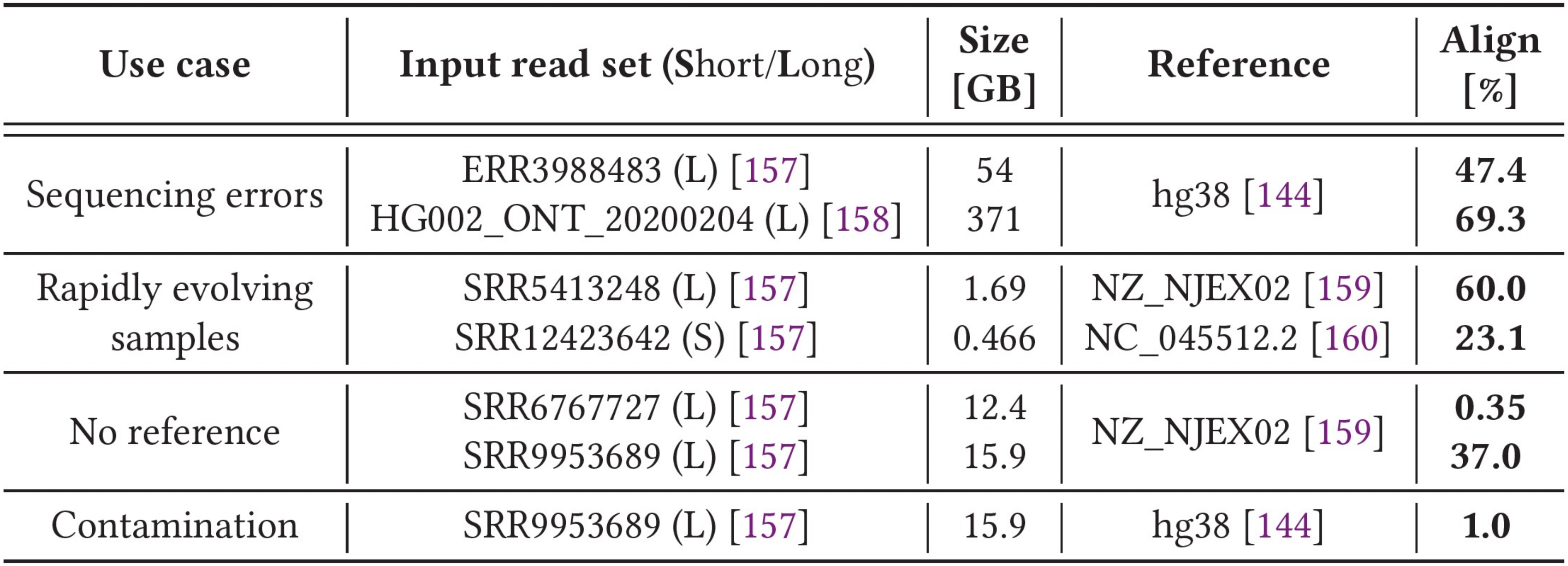
Comparison to PIM

- 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

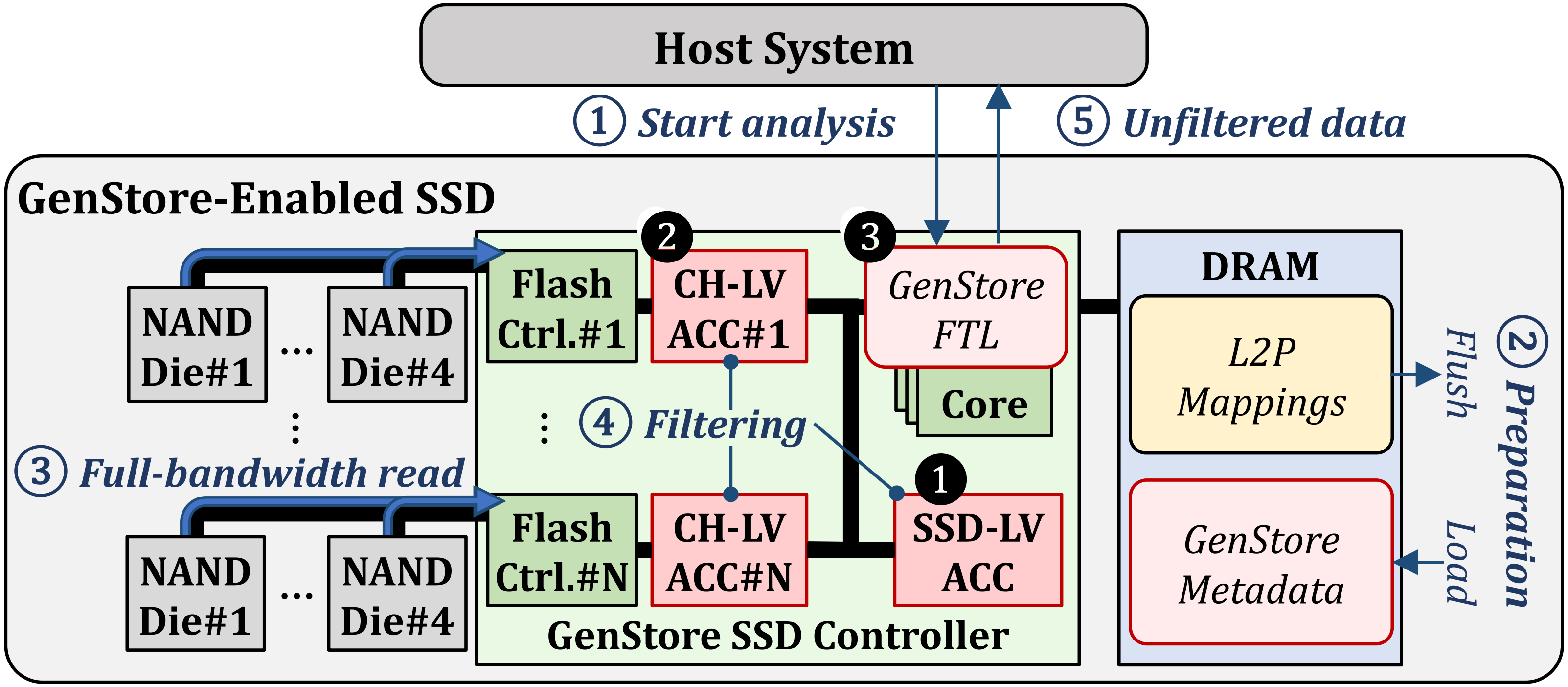
In-storage filter can be integrated with any read mapping accelerator, including PIM accelerators, to alleviate their data movement overhead.

Long Read Use Cases



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

FTL


Host System
① Start analysis
⑤ Unfiltered data
GenStore-Enabled SSD
NAND Die#1
...
NAND Die#4
NAND Die#1
...
NAND Die#4
③ Full-bandwidth read
Flash Ctrl.#1
Flash Ctrl.#N
2
CH-LV ACC#1
CH-LV ACC#N
④ Filtering
3
GenStore FTL
Core
1
SSD-LV ACC
GenStore SSD Controller
DRAM
L2P Mappings
GenStore Metadata
Flush
Load
② Preparation

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 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

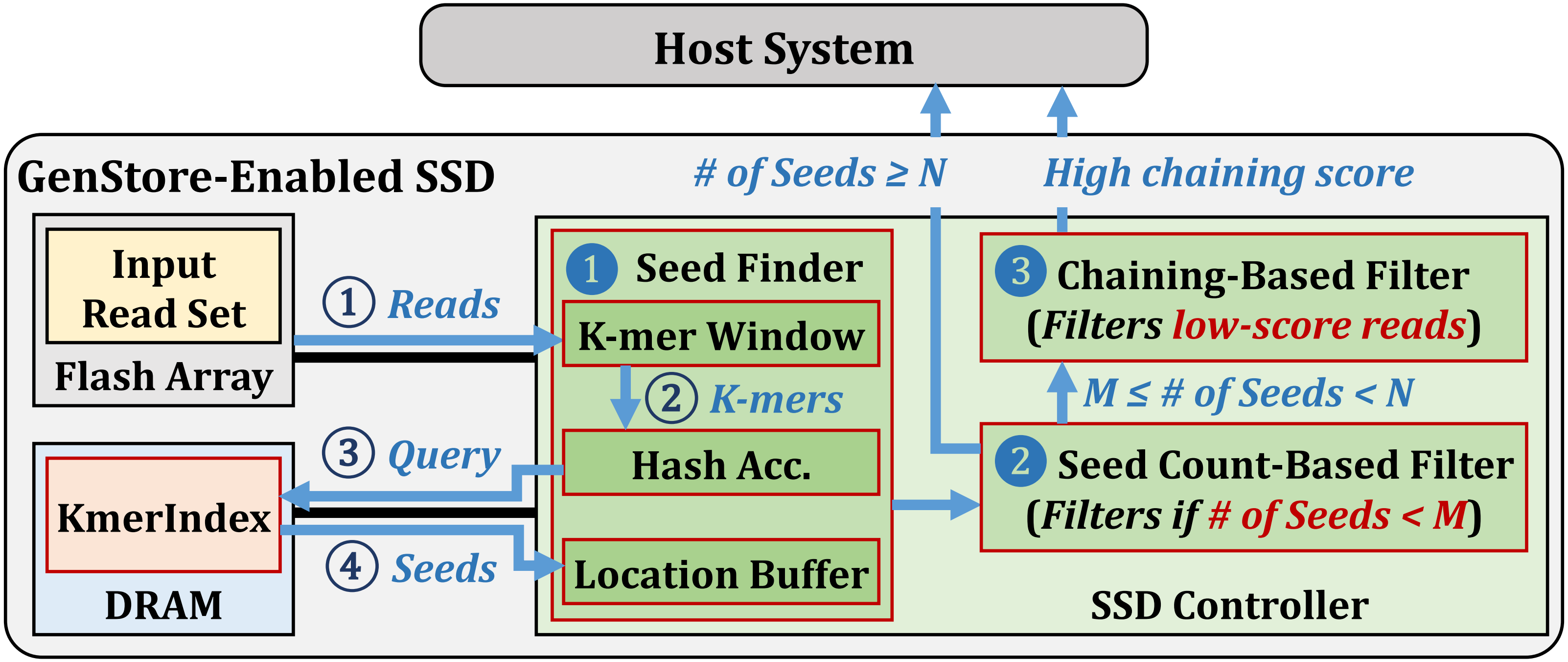
- **SSD-L:** PCIe Gen3 M.2 interface (3.5 GB/s sequential read)
  - 1.2 GB/s per channel bandwidth
  - 16 channels

- **SSD-L:** PCIe Gen4 interface (7 GB/s sequential read)
  - 1.2 GB/s per channel bandwidth
  - 16 channels

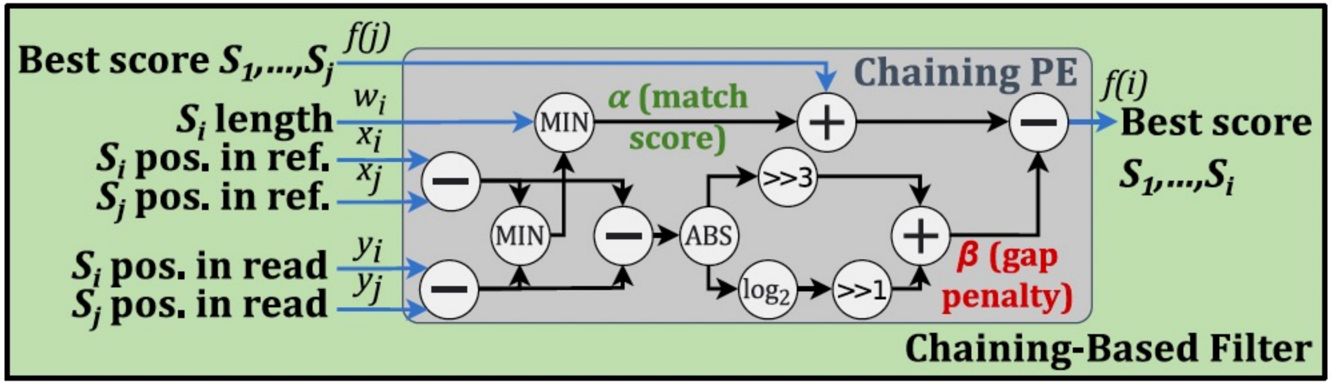
Evaluation Methodology

- **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 μProf

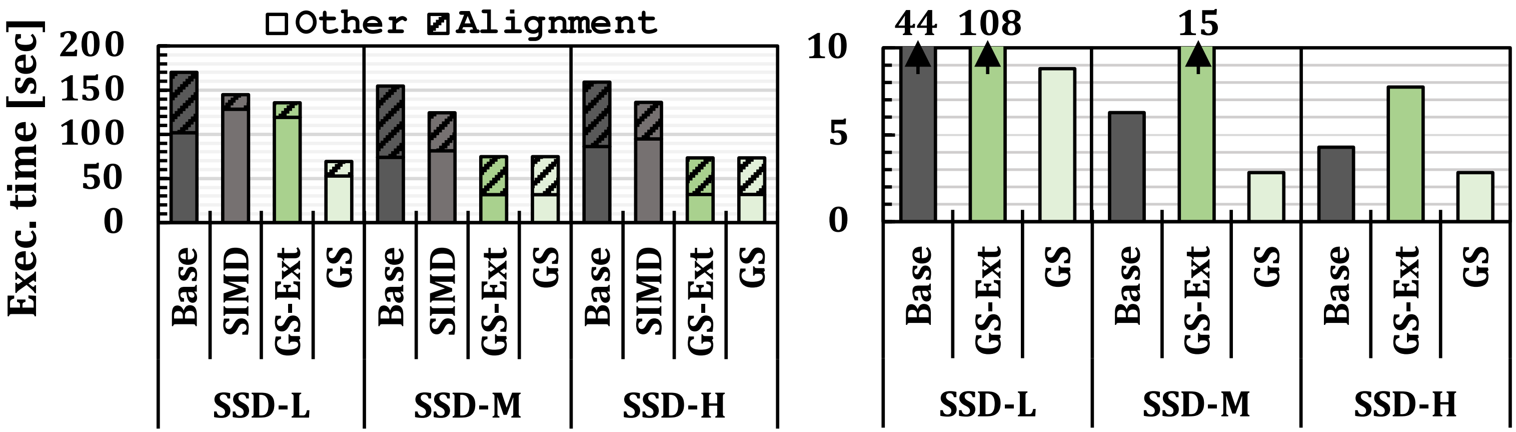
GenStore-NM


Host System
GenStore-Enabled SSD
# of Seeds ≥ N
High chaining score
Input Read Set
Flash Array
1 Reads
1 Seed Finder
K-mer Window
2 K-mers
Hash Acc.
Location Buffer
3 Query
4 Seeds
KmerIndex
DRAM
3 Chaining-Based Filter
(Filters low-score reads)
M ≤ # of Seeds < N
2 Seed Count-Based Filter
(Filters if # of Seeds < M)
SSD Controller

Chaining Processing Element



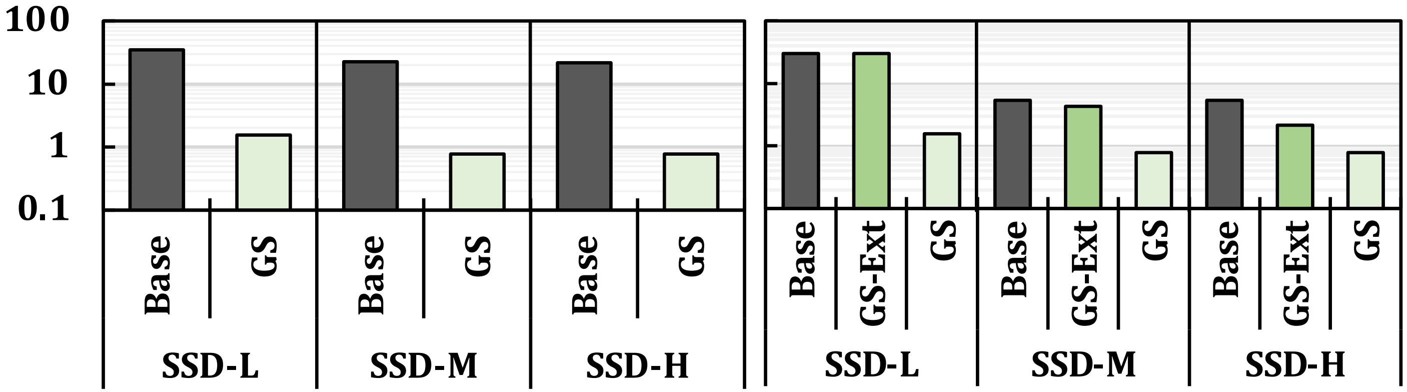
GenStore-EM



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

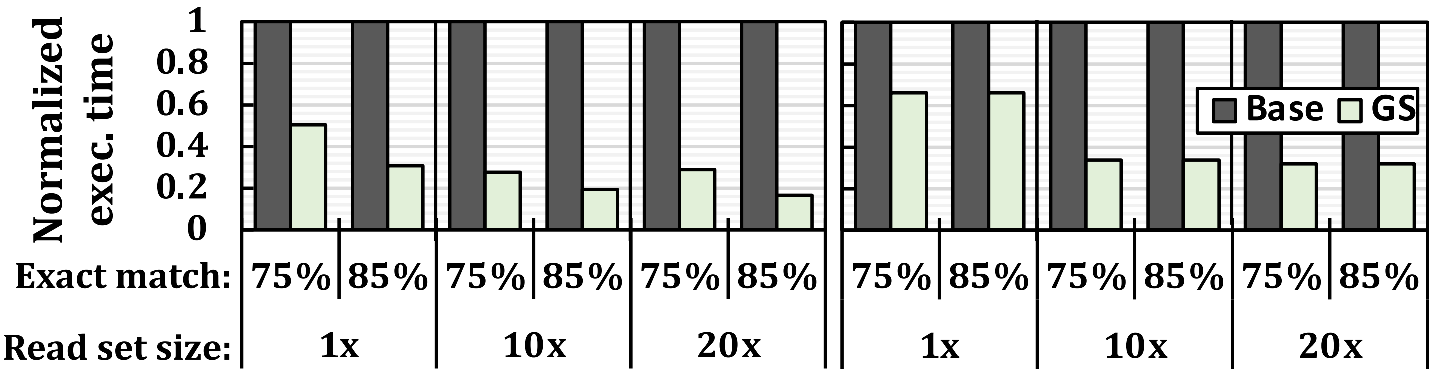


GS-Ext performs significantly slower than Base (2.28x - 1.91x) on all systems.

SAFARI

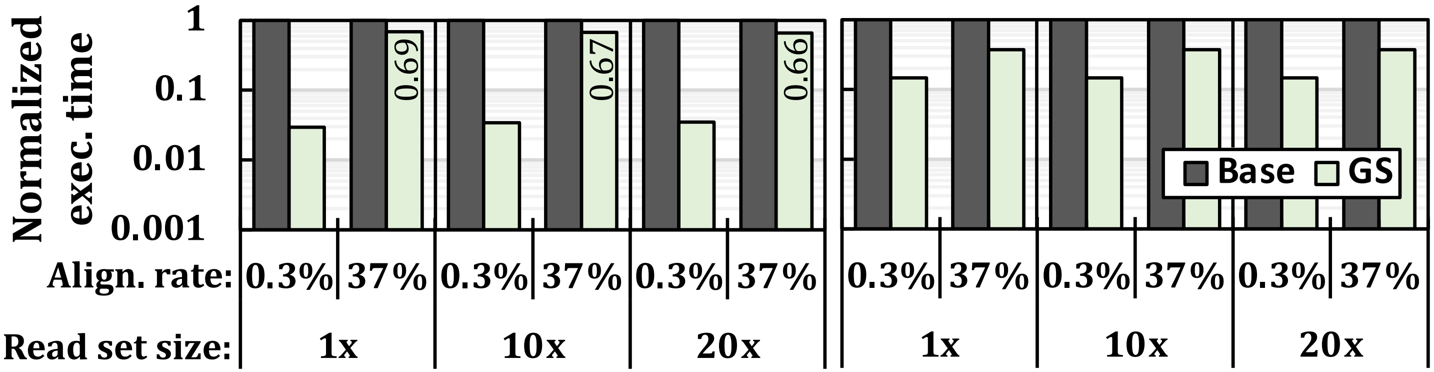
Effect of Inputs on GenStore-EM

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



Effect of Inputs on GenStore-NM

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



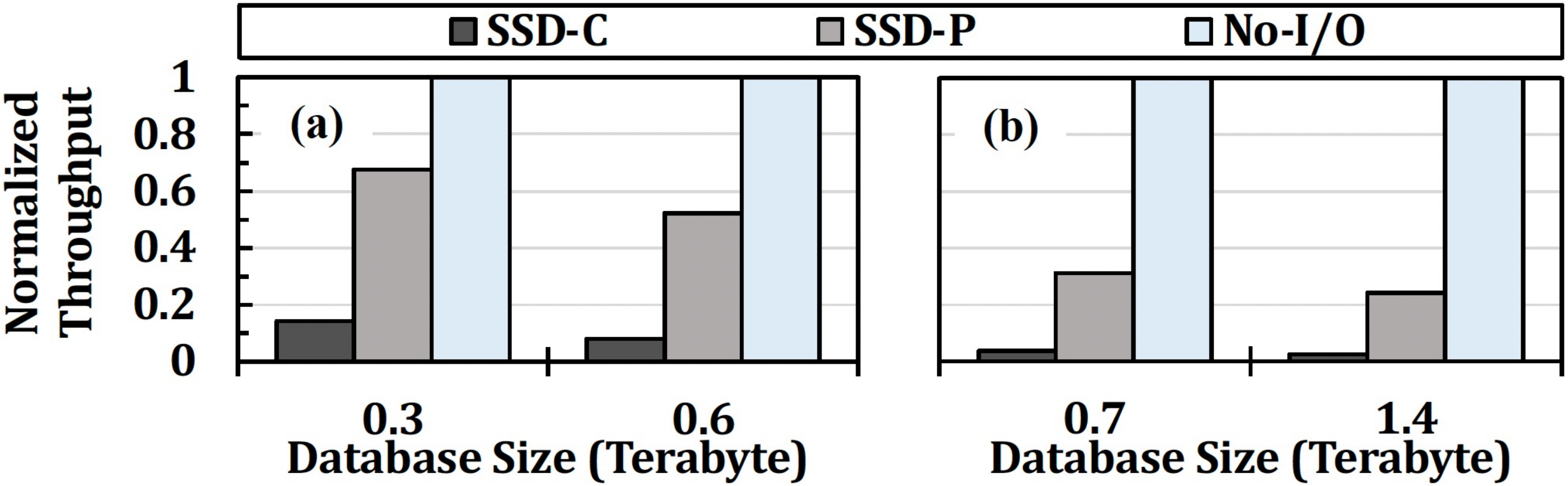
MegIS Backup Slides

Motivational Analysis

Database access patterns

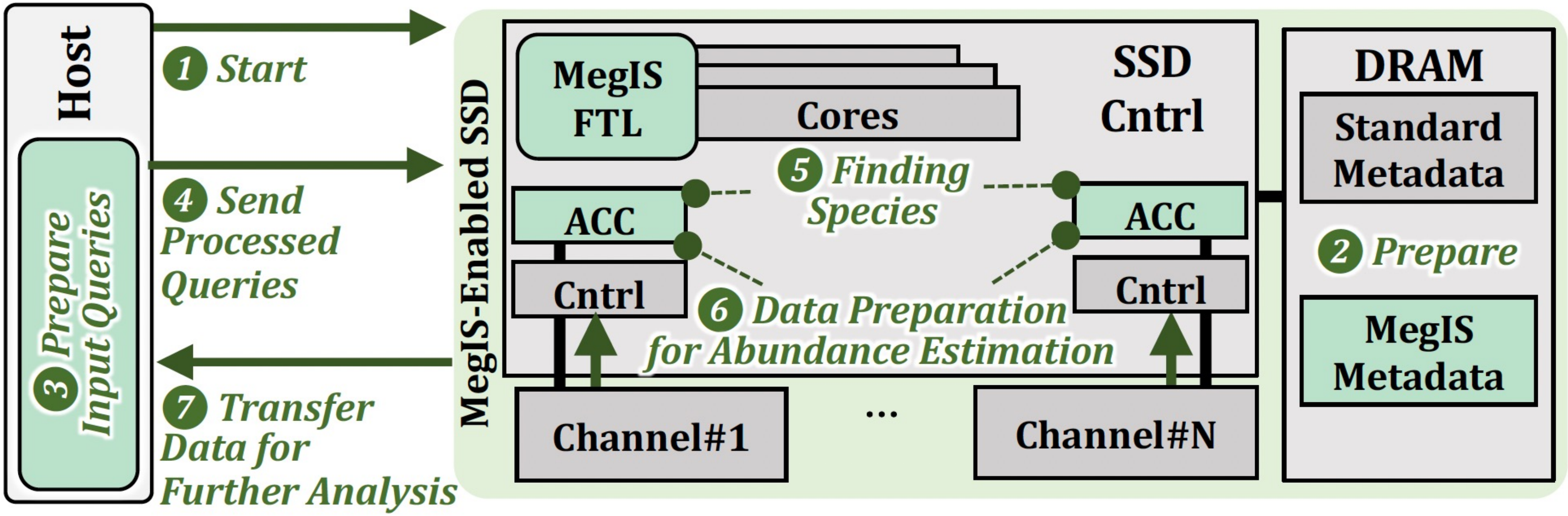
(a) Random Query

(b) Streaming Query

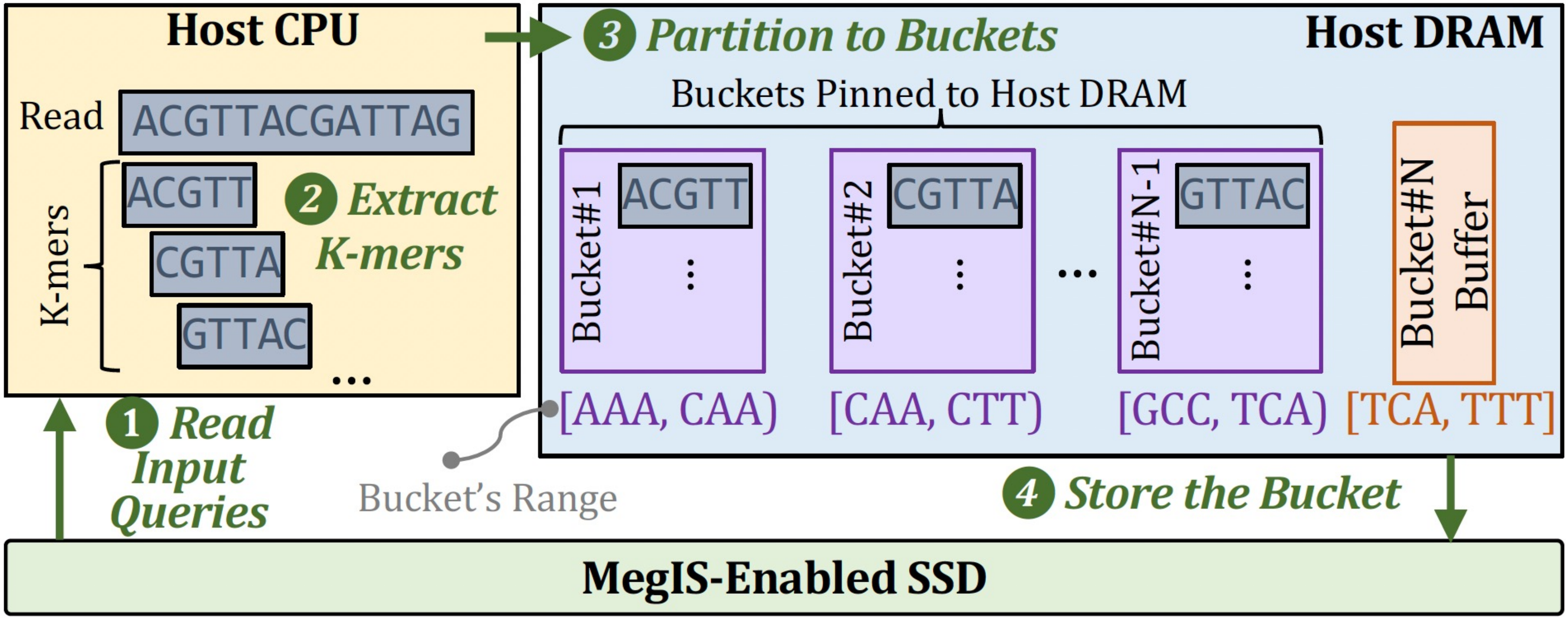


SAFARI

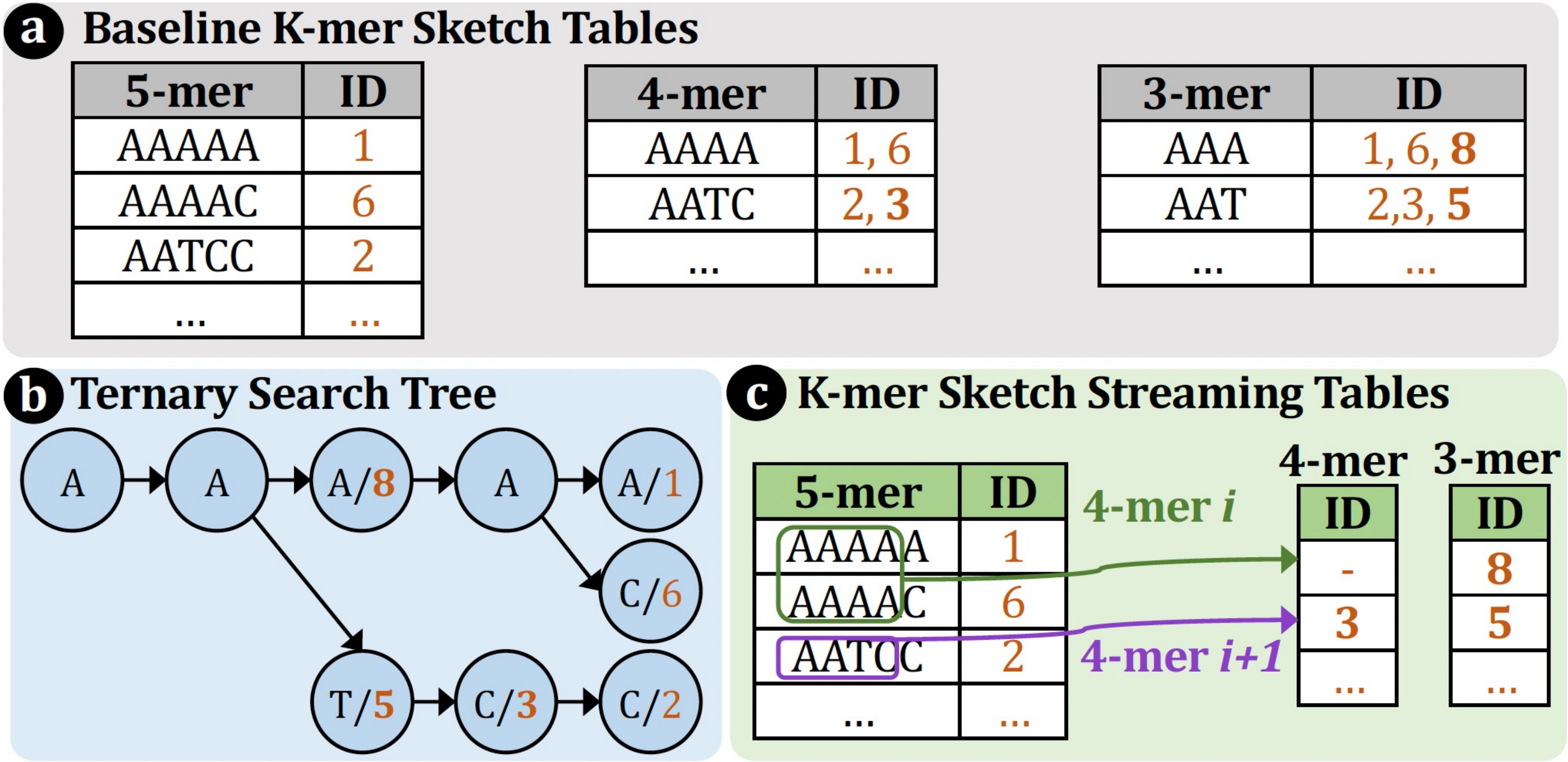
Overview of MegIS's Steps



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

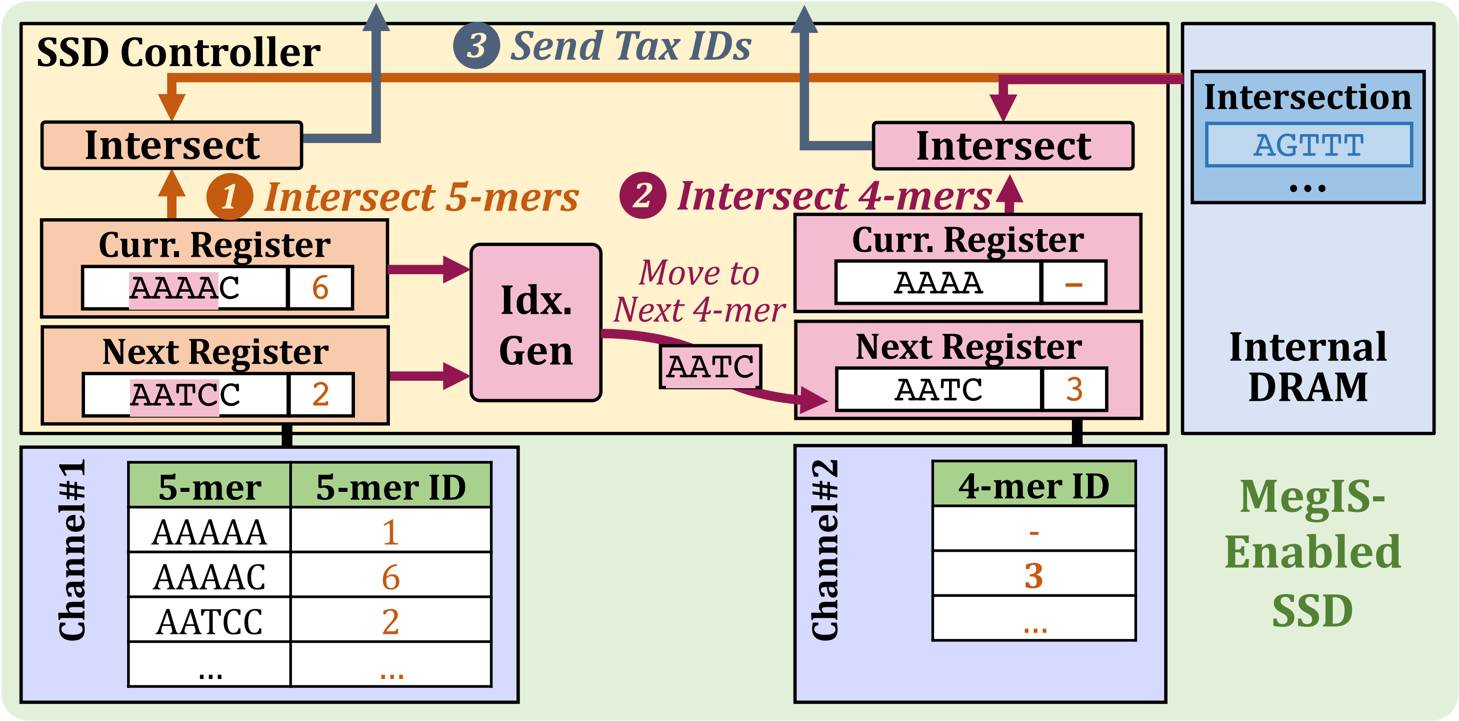
c K-mer Sketch Streaming Tables

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

4-mer ID
-
3
...

3-mer ID
8
5
...

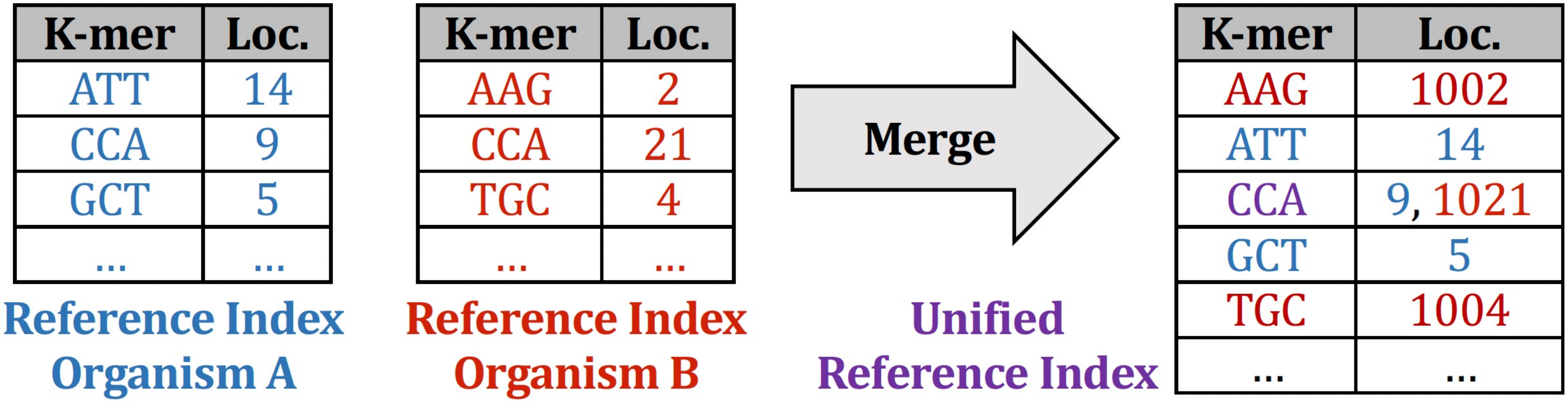
K-mer Sketch Streaming Hardware Design



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

4-mer ID
-
3
...

Index Generation in Step 3



K-mer	Loc.
ATT	14
CCA	9
GCT	5
…	…

Reference Index Organism A

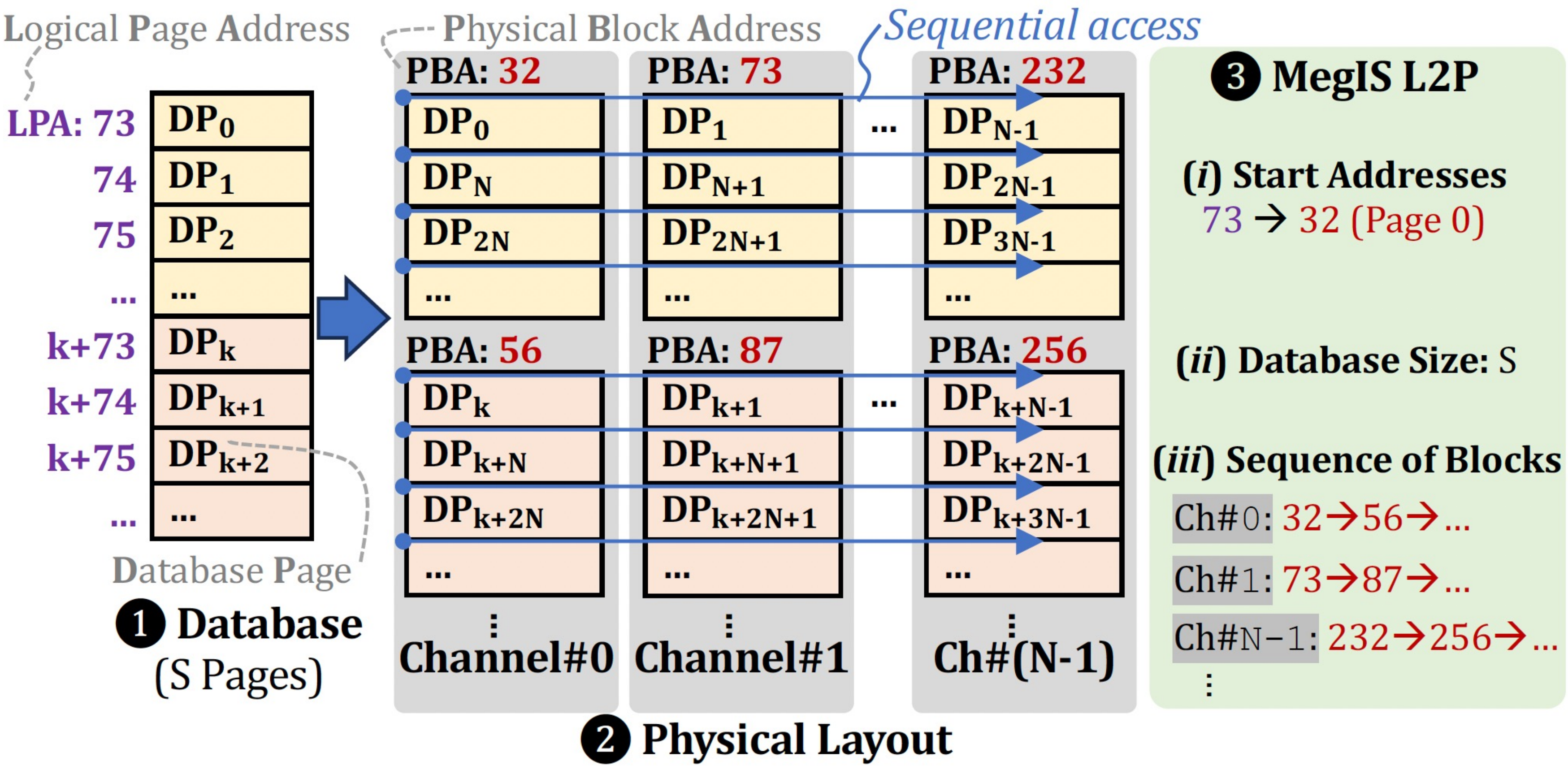
K-mer	Loc.
AAG	2
CCA	21
TGC	4
…	…

Reference Index Organism B

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

Unified Reference Index

MegIS FTL



3 **MegIS L2P**

(*i*) Start Addresses
73 → 32 (Page 0)

(*ii*) Database Size: S

(*iii*) Sequence of Blocks

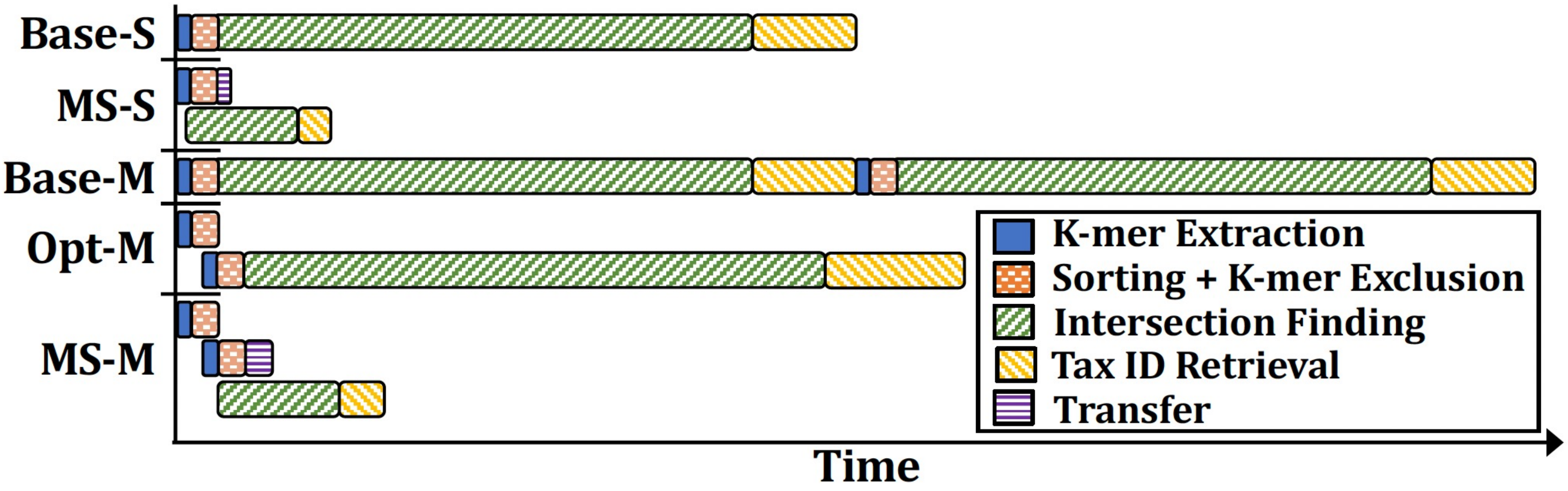
Ch#0: 32→56→...

Ch#1: 73→87→...

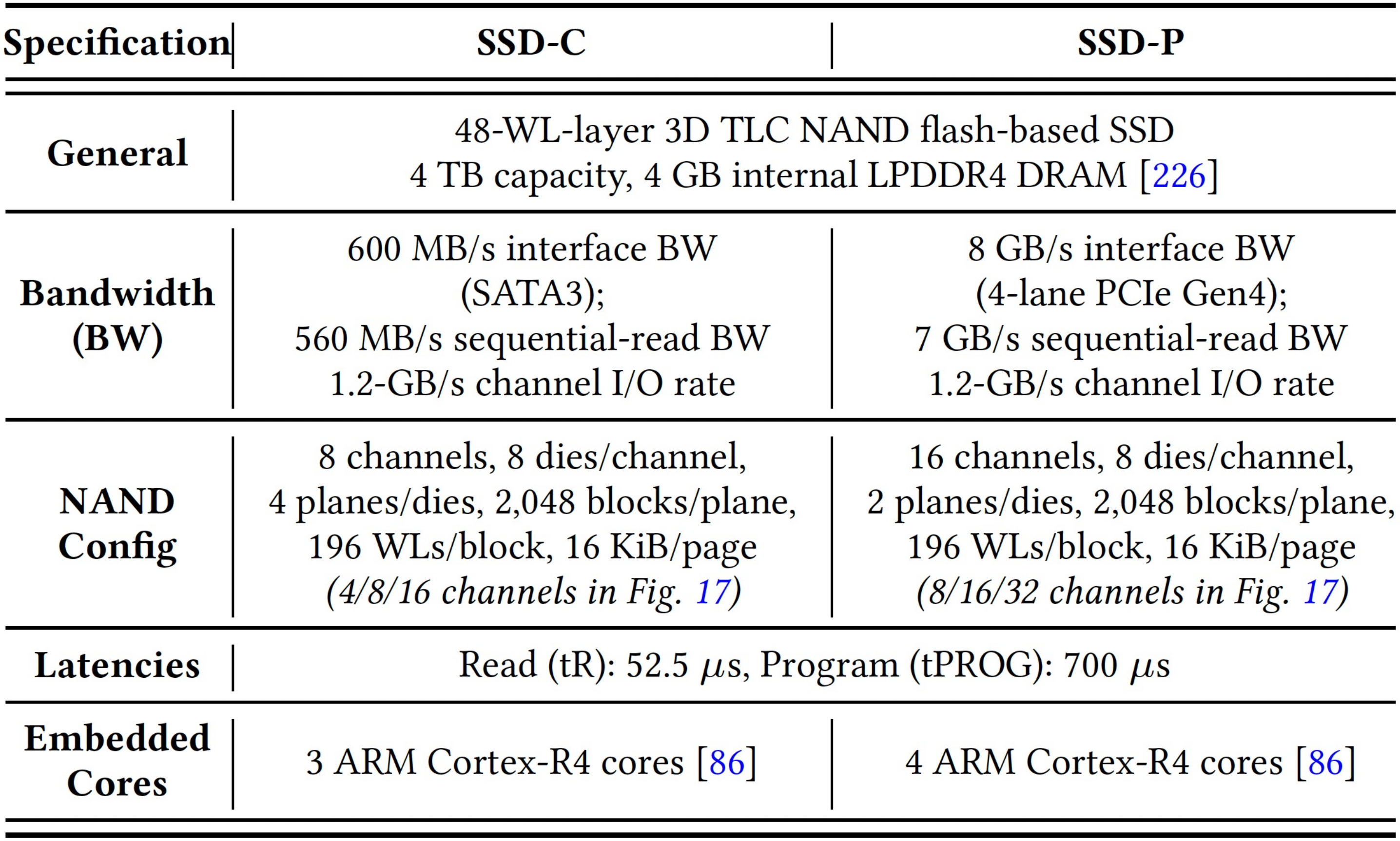
Ch#N-1: 232→256→...

⋮

Multi-Sample Analysis

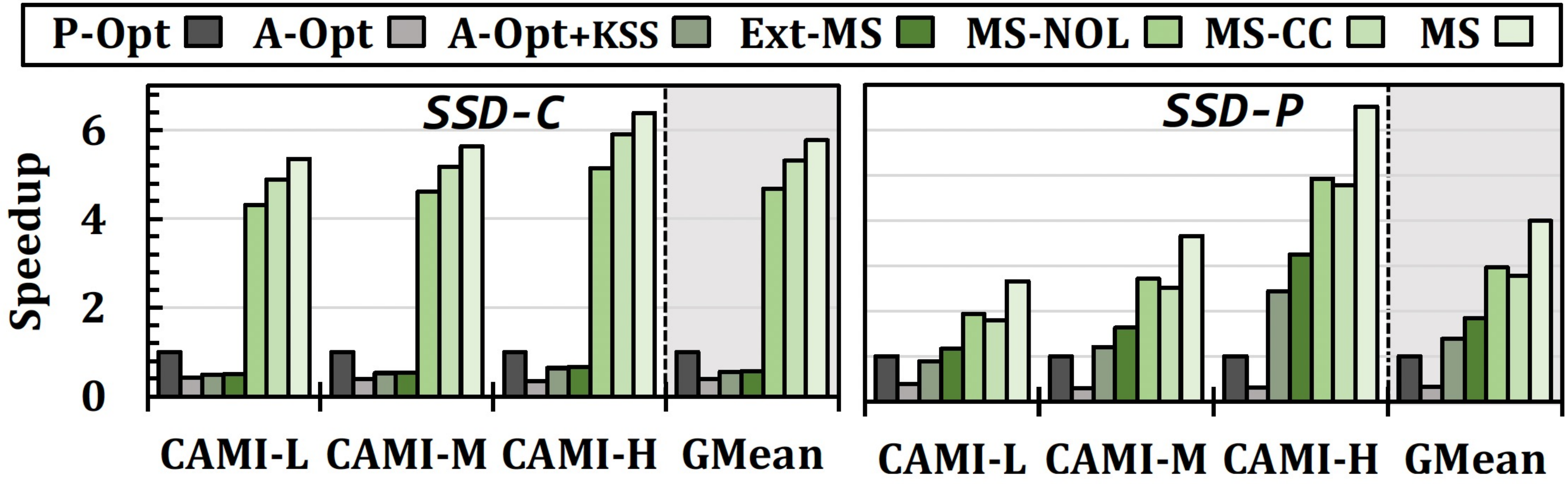


SSD Configurations

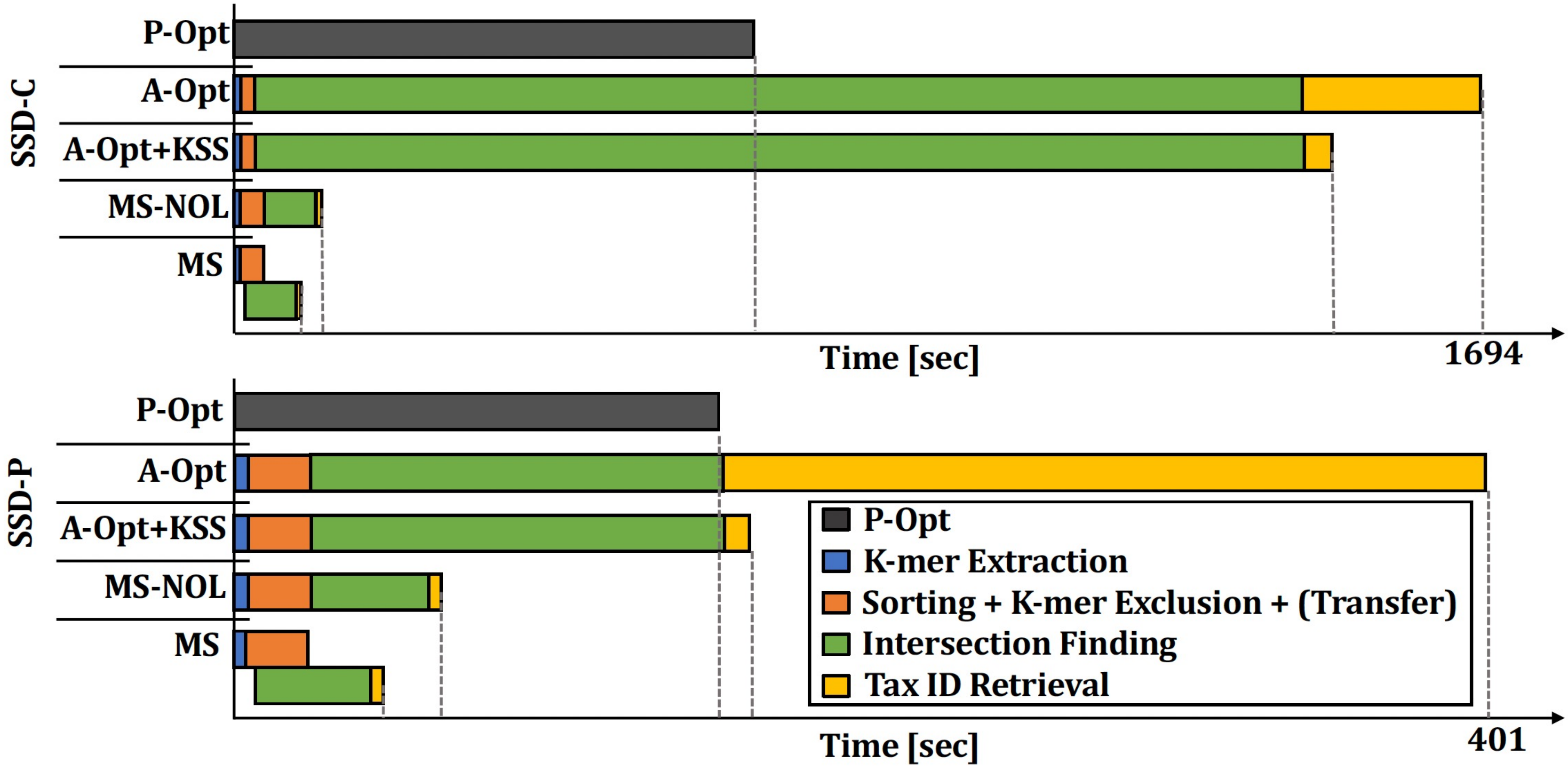


Specification	SSD-C	SSD-P
General	48-WL-layer 3D TLC NAND flash-based SSD 4 TB capacity, 4 GB internal LPDDR4 DRAM [226]	
Bandwidth (BW)	600 MB/s interface BW (SATA3); 560 MB/s sequential-read BW 1.2-GB/s channel I/O rate	8 GB/s interface BW (4-lane PCIe Gen4); 7 GB/s sequential-read BW 1.2-GB/s channel 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 μs, Program (tPROG): 700 μs	
Embedded Cores	3 ARM Cortex-R4 cores [86]	4 ARM Cortex-R4 cores [86]

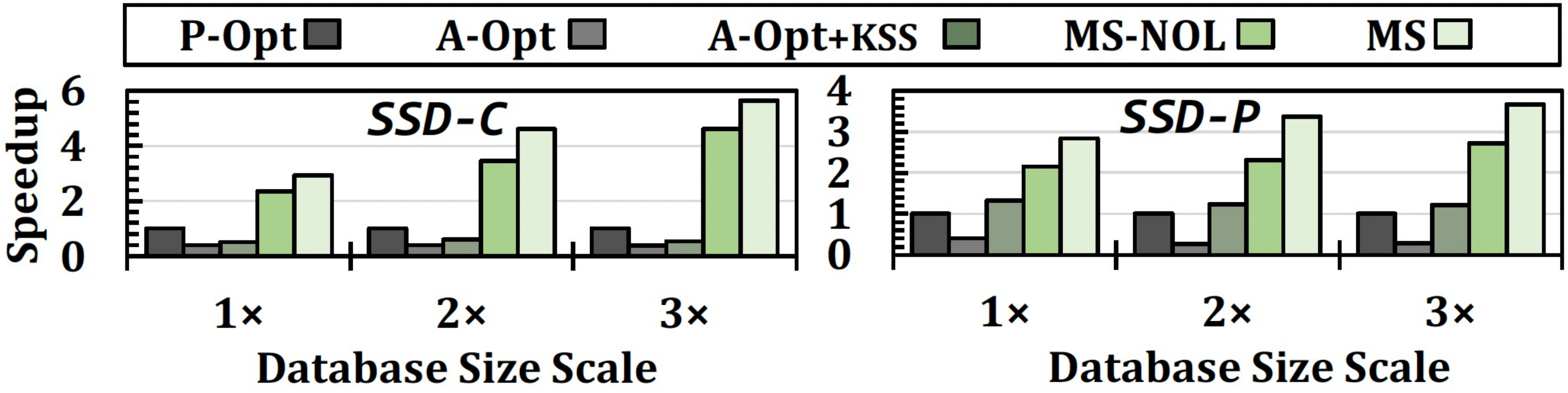
Impact of Different Optimizations



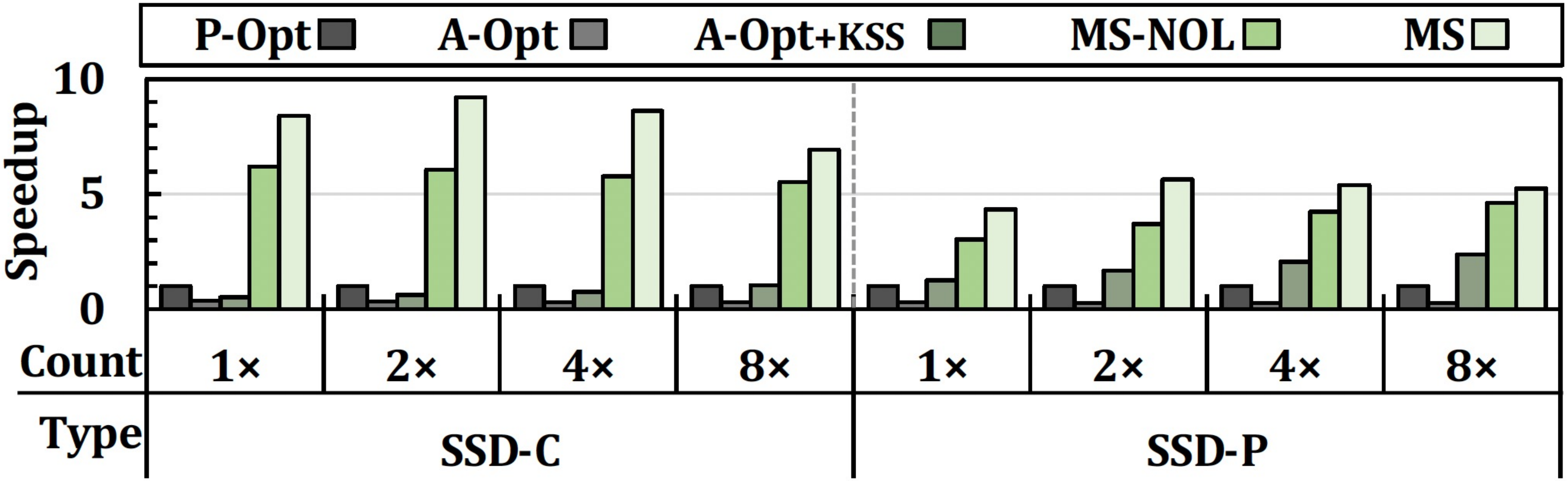
Impact of Different Optimizations



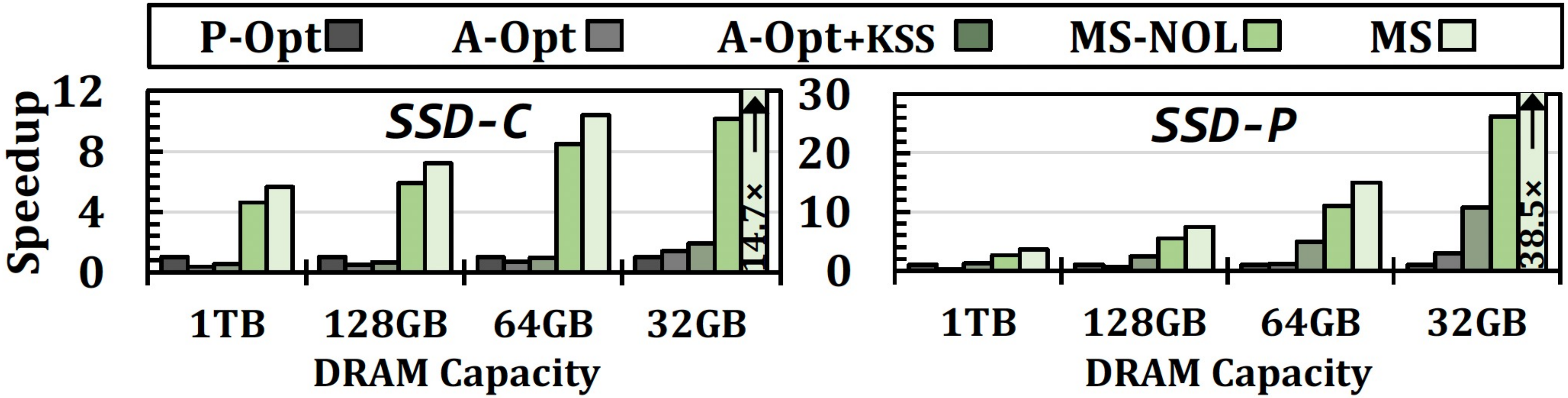
Speedup with Different Database Sizes



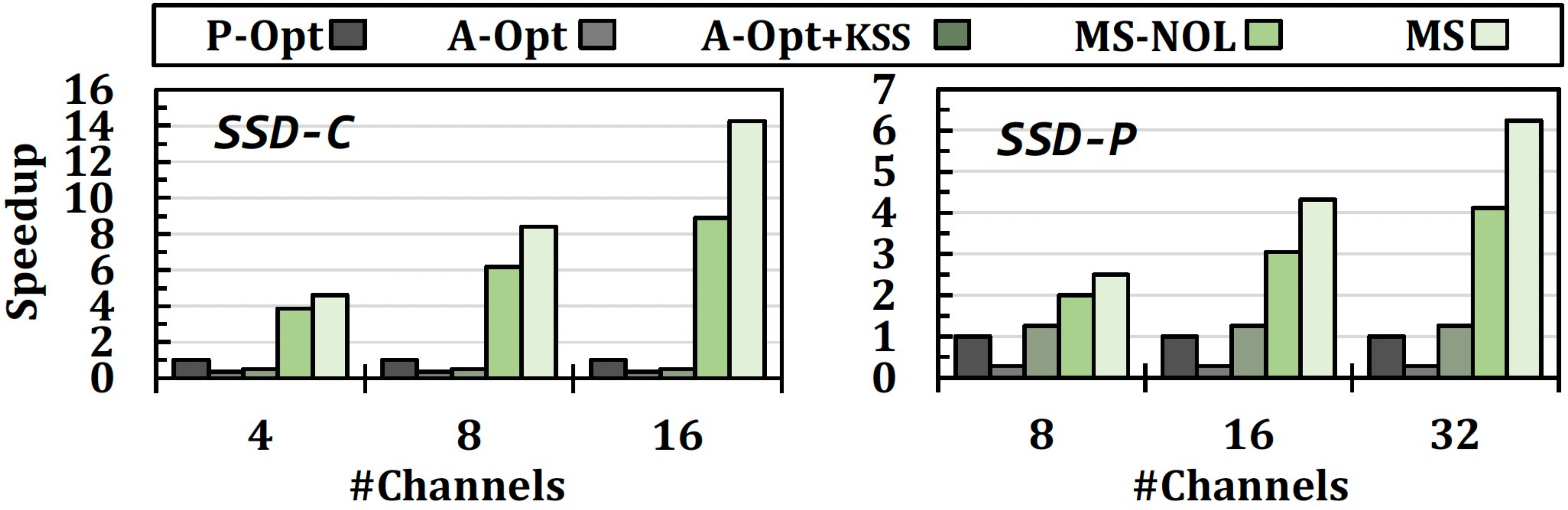
Speedup with Different #SSDs



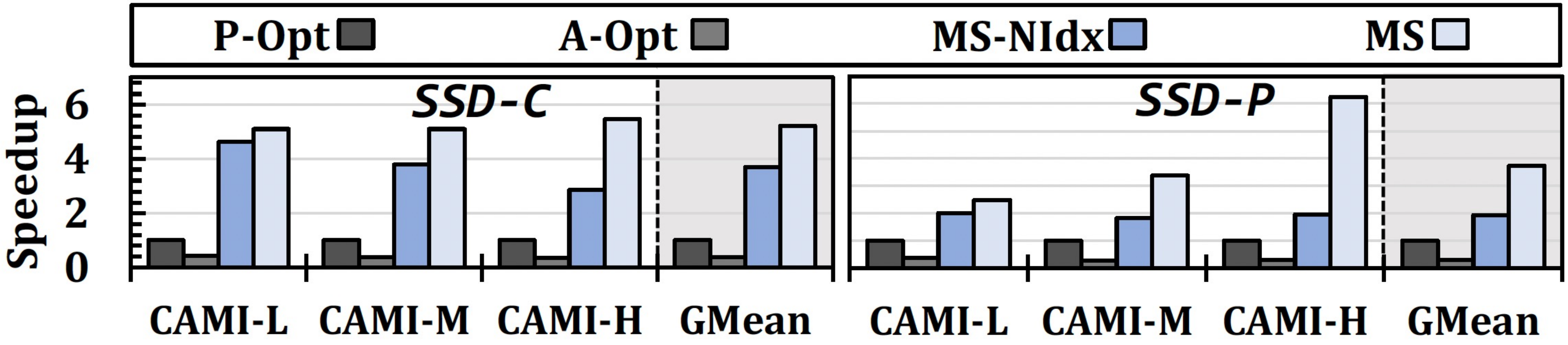
Speedup with Different Main Memory Capacities



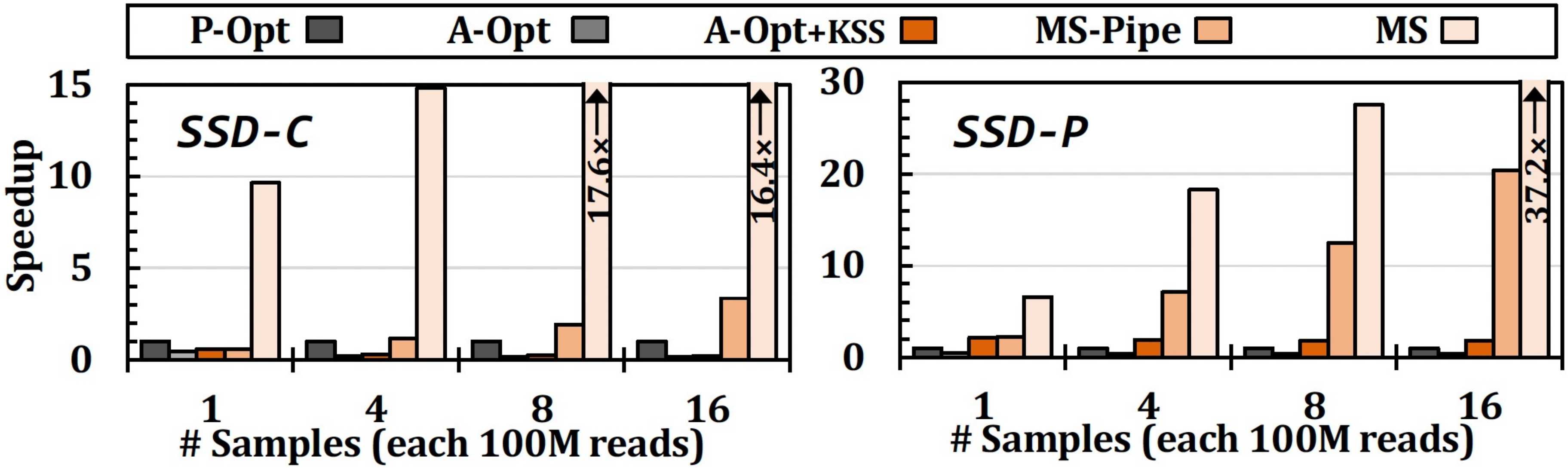
Speedup with Varying SSD Internal Bandwidth



Speedup of Abundance Estimation



Multi-Sample Use Case



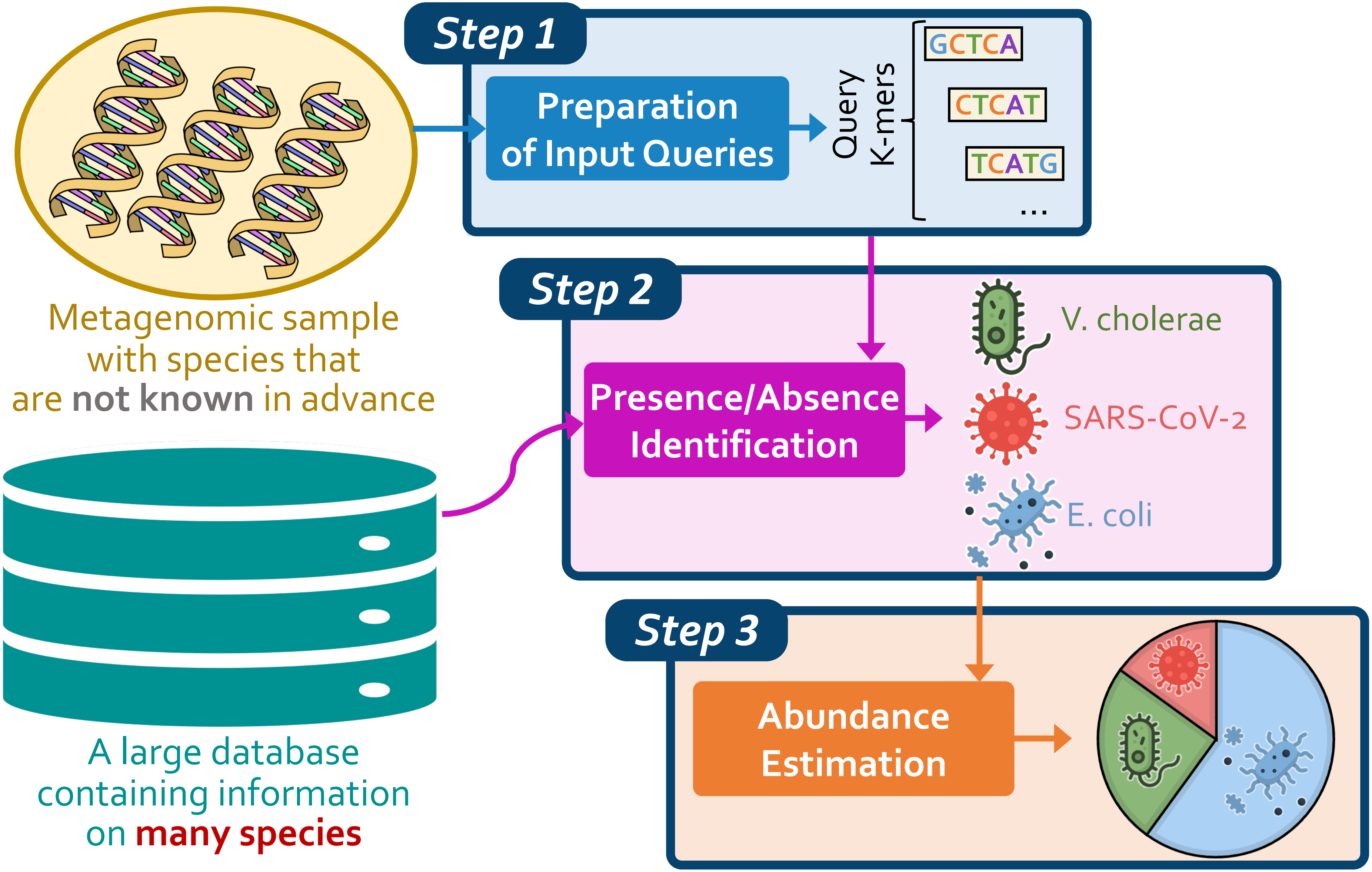
Area and Power

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

Logic Unit	# of instances	Area [mm²]	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
Total for an 8-channel SSD	-	*0.04*	*7.658*

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

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

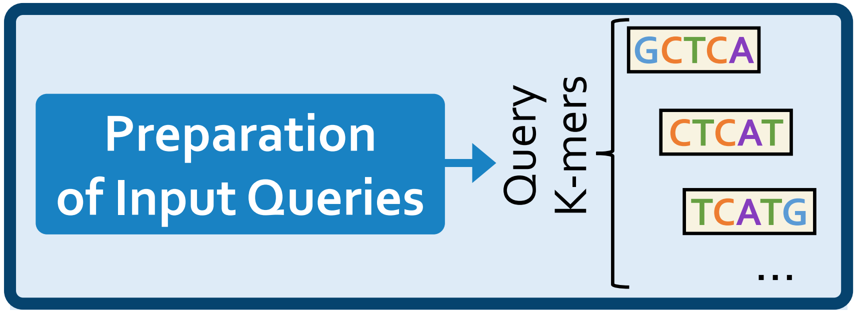
SARS-CoV-2

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

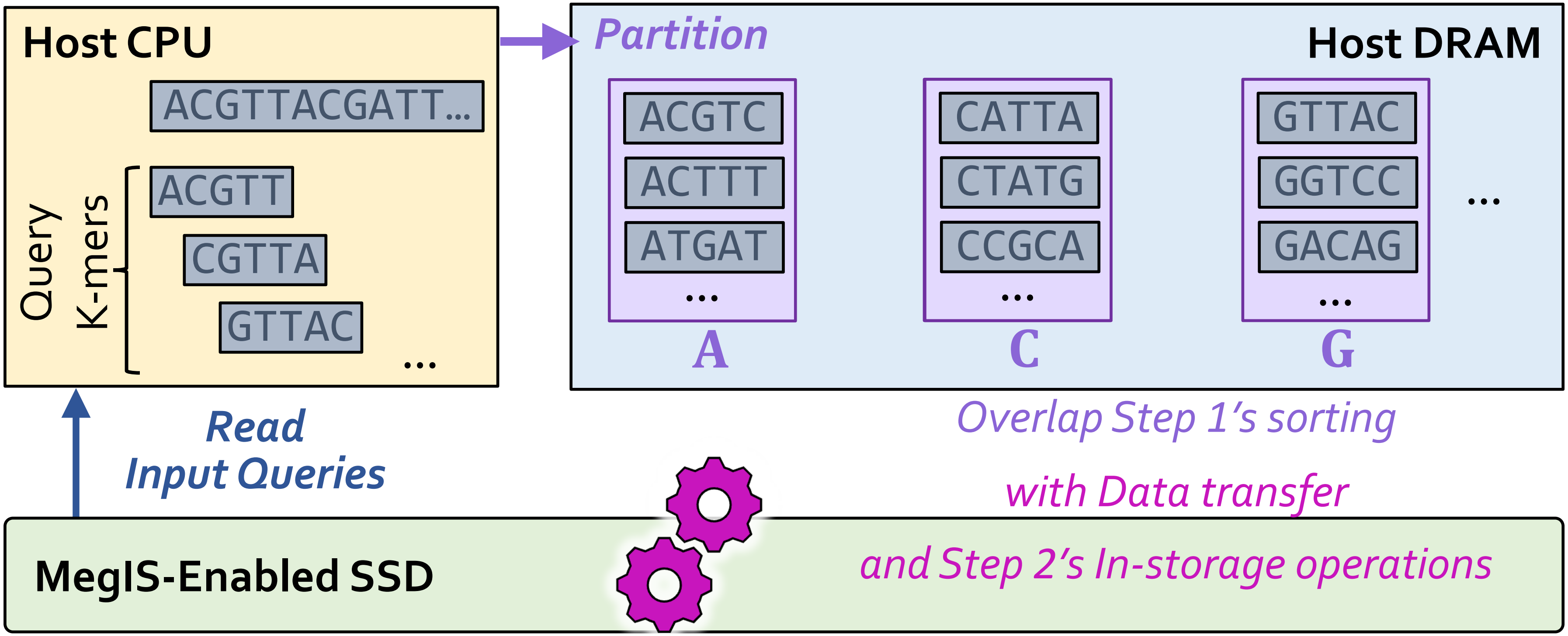
To execute Step 1 efficiently in the host system, MegIS needs to:

- 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**

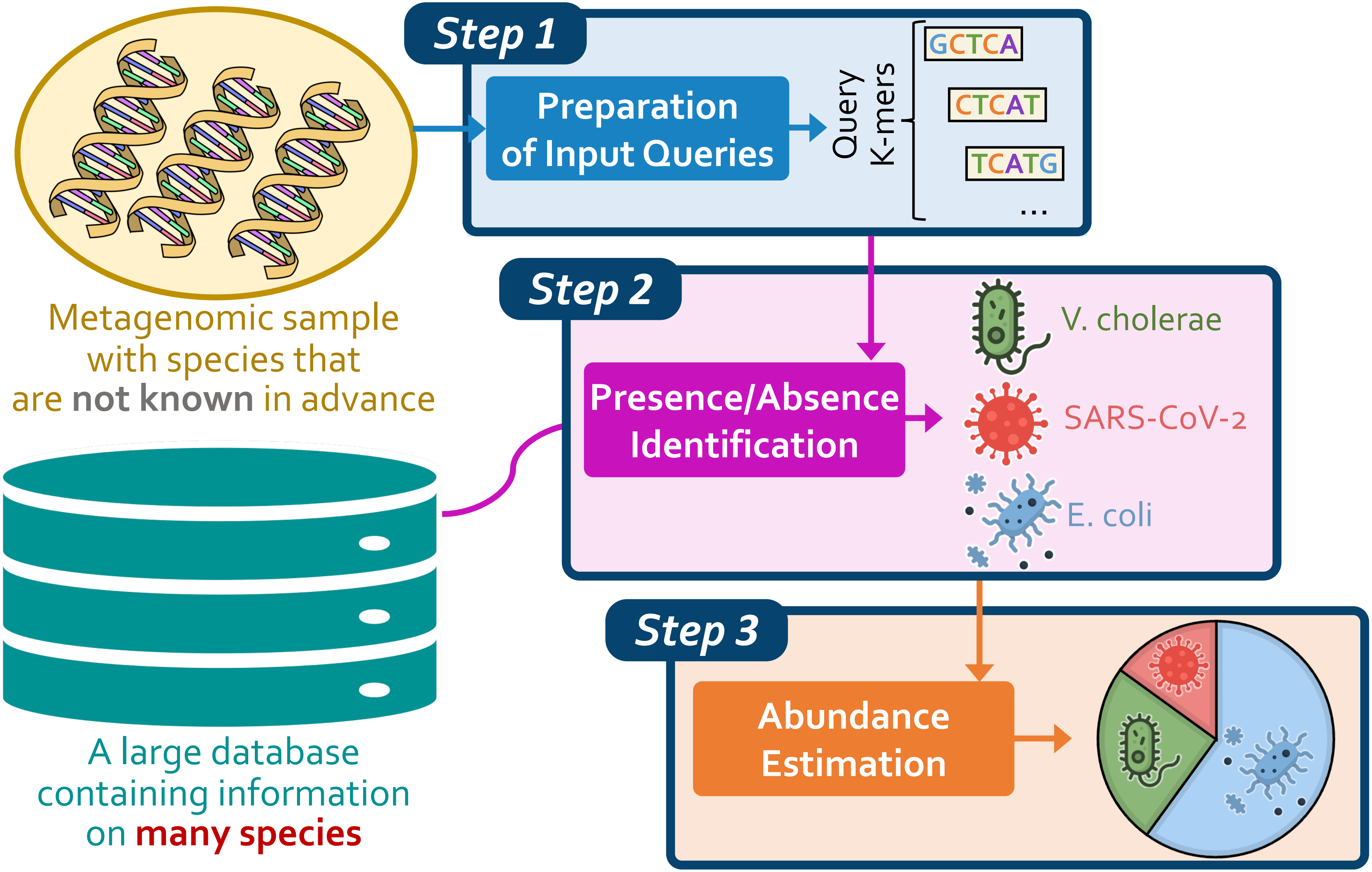


Overlap Step 1's sorting

with Data transfer

and Step 2's In-storage operations

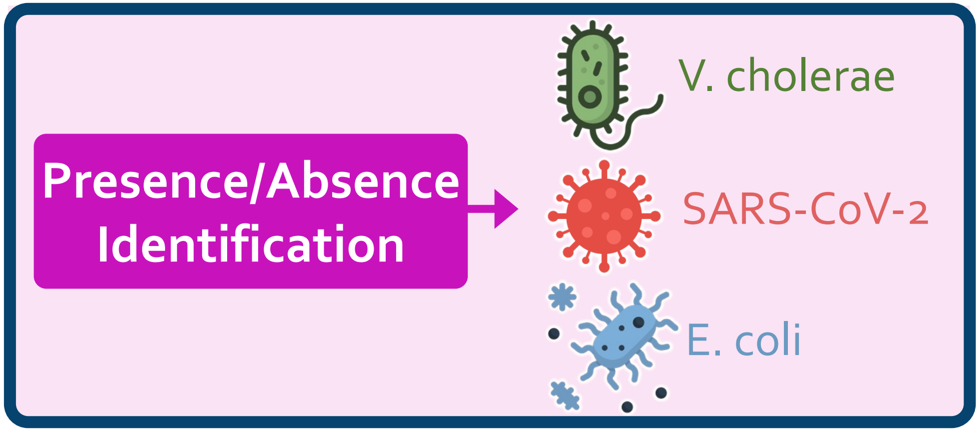
Step 2 Overview



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

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

Step 2 Overview



Presence/Absence Identification

V. cholerae

SARS-CoV-2

E. coli

- **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

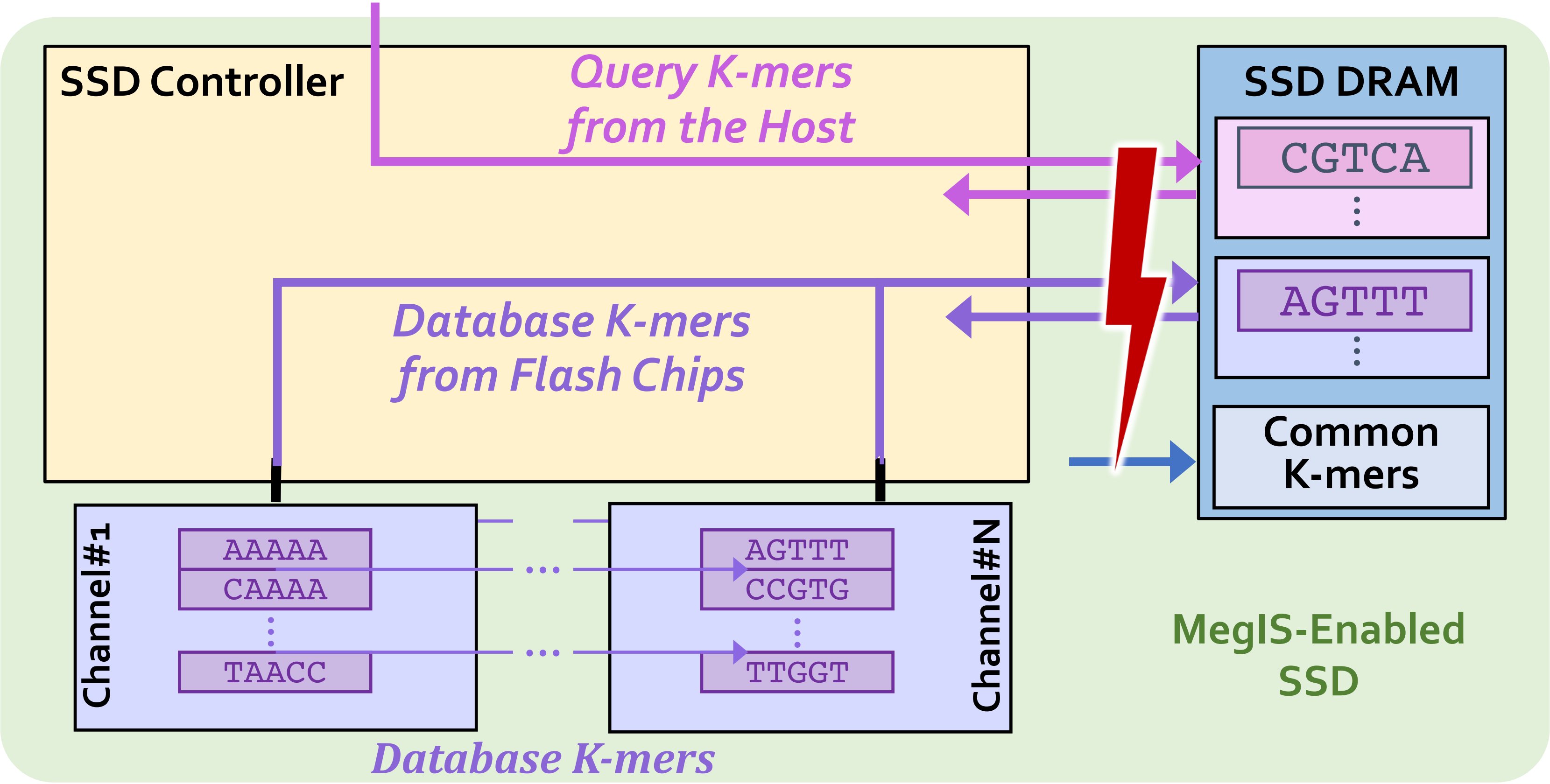
- Accesses **large data** with **low reuse**
- Involves **lightweight computation**

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

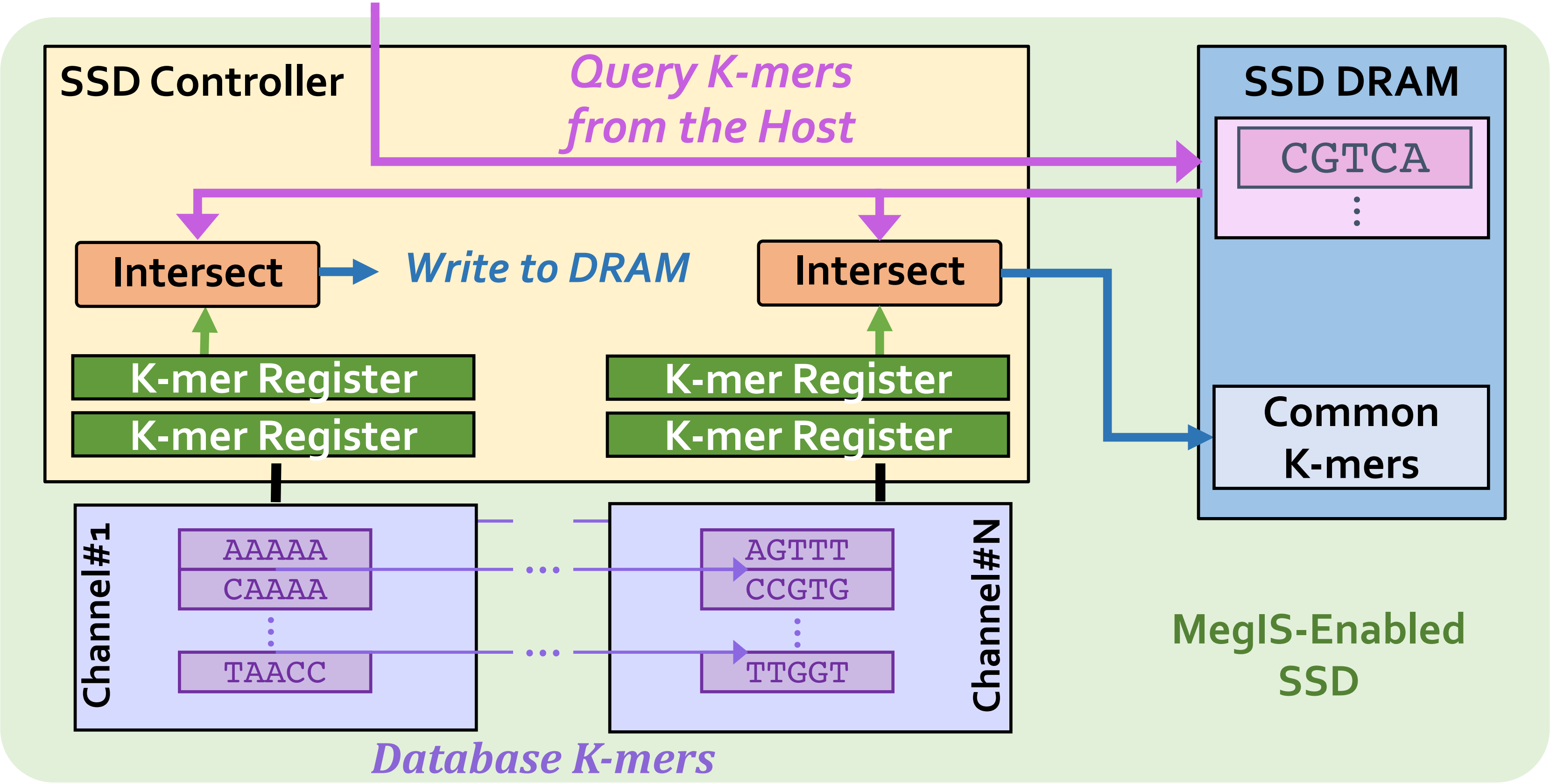


Step 2 Design: Identifying the Common K-mers

- **Challenge:** Limited internal DRAM bandwidth

✓ *Compute directly on the flash data streams* *[Zou+, MICRO'22]*

✓ *Reduce buffer size based on application features*

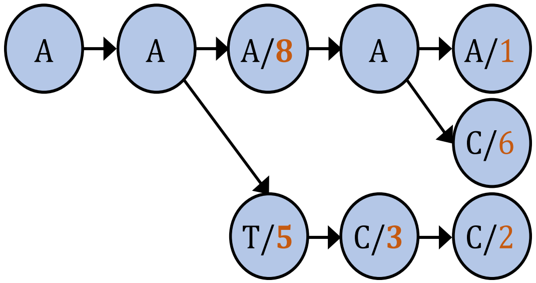


Step 2 Design: Retrieving the Species ID

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

K-mer	
AAAAA	
AAAAC	
AATCC	
…	

Space-Inefficient



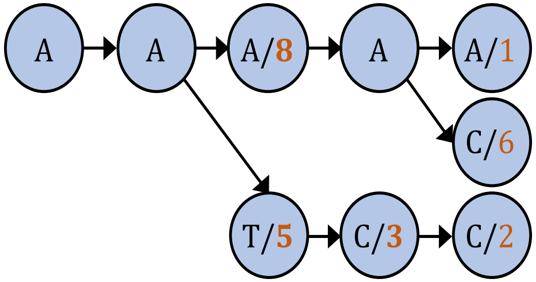
Space-Efficient

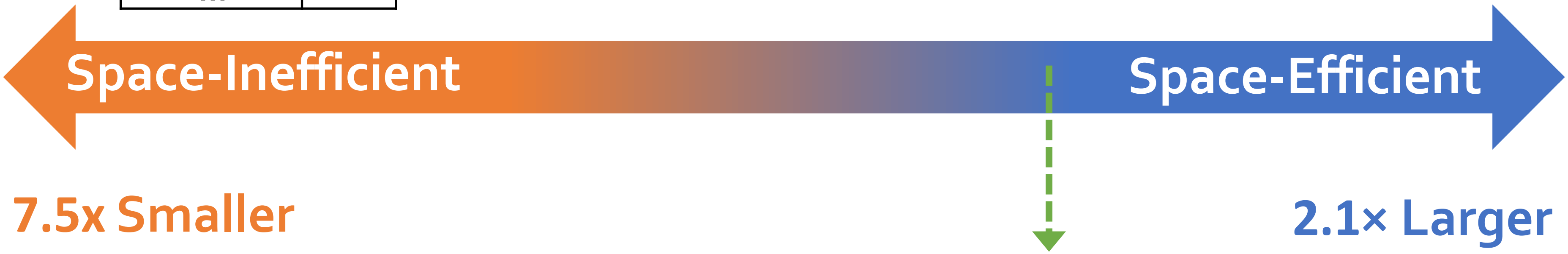
✗ ***Slow inside the SSD due to long NAND flash latency***

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

2.1× Larger

K-mer Sketch Streaming

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

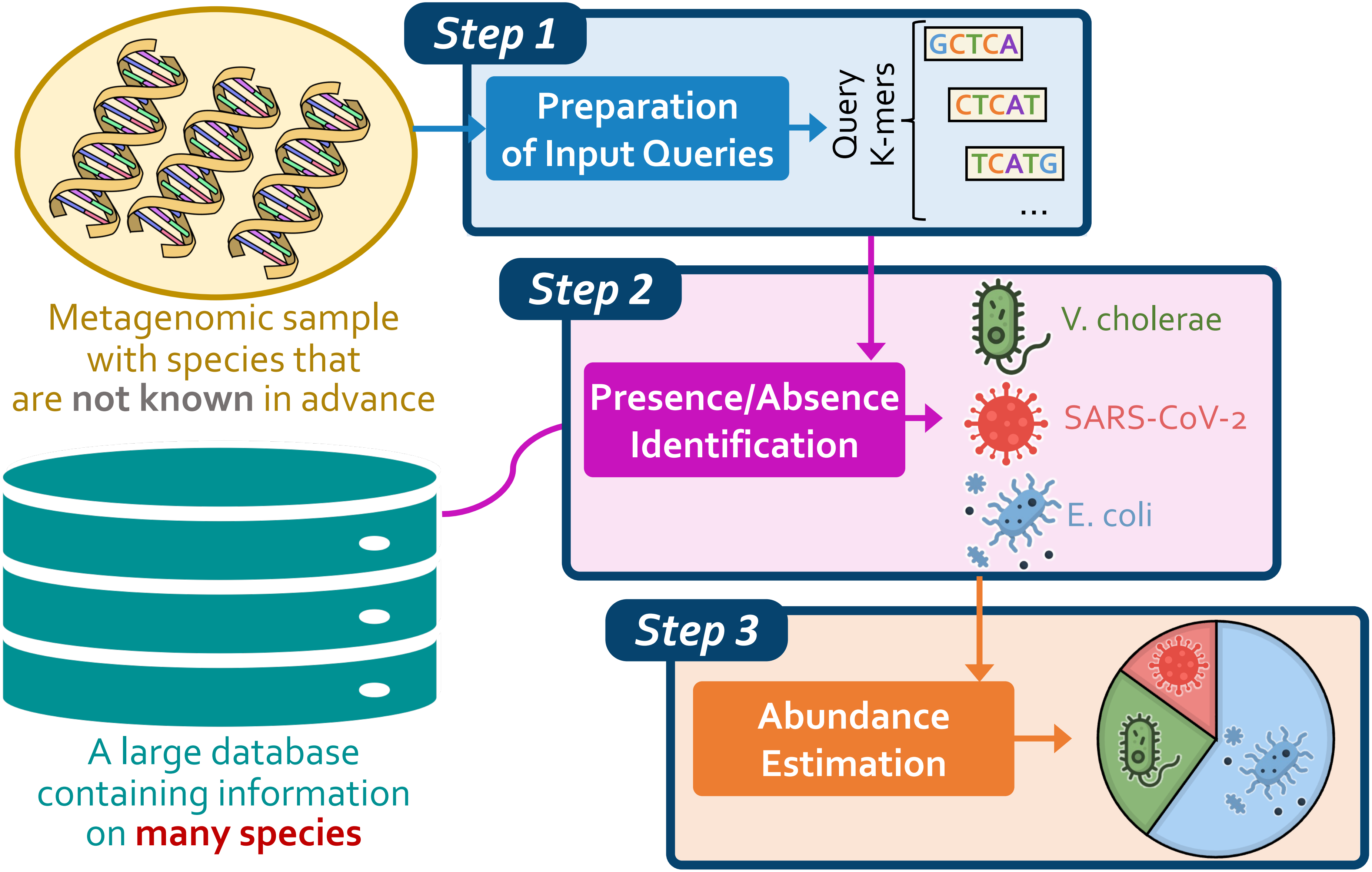
Step 2 Design: Retrieving the Species ID

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

Design details are in the paper

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

Step 3



Step 1

Preparation of Input Queries

Query K-mers

GCTCA

CTCAT

TCATG

...

Step 2

Presence/Absence Identification

V. cholerae

SARS-CoV-2

E. coli

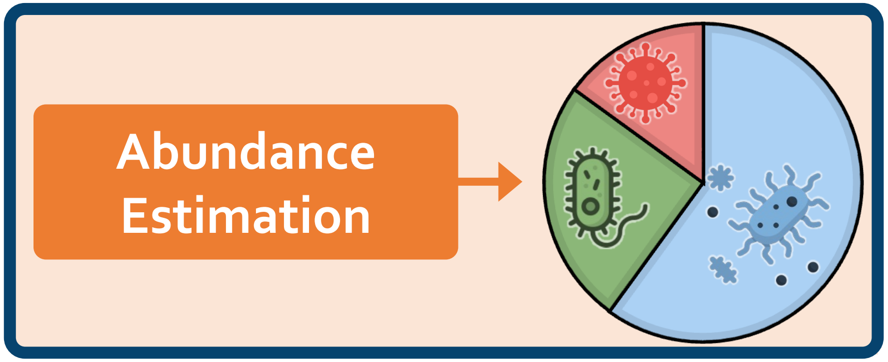
Step 3

Abundance Estimation

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

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

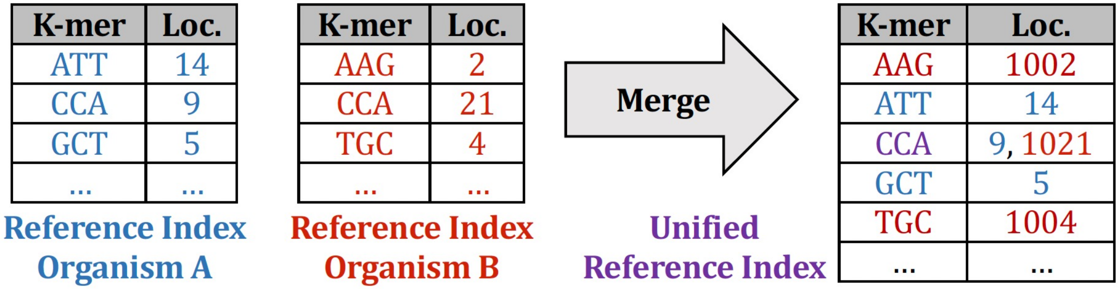
Step 3



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



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

Step 3 and MegIS FTL are in the paper