GRAINS

Enabling High-Performance and Low-Cost Graph-Based Genome Analysis via Storage-Aware Algorithm-Architecture Co-Design

Nika Mansouri Ghiasi Harun Mustafa Talu Güloglu **Rakesh Nadig**

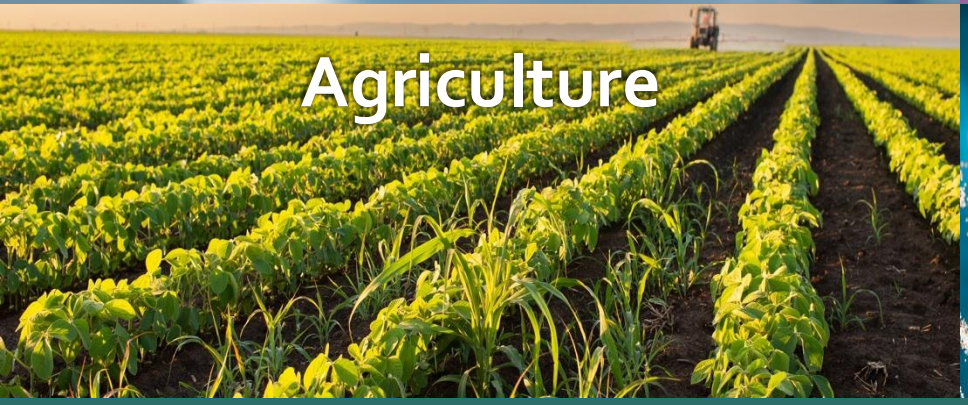
Konstantina Koliogeorgi Susana Rebolledo Ruiz Marc Rautmann Furkan Eris

Mohammad Sadrosadati Jisung Park Onur Mutlu

SAFARI

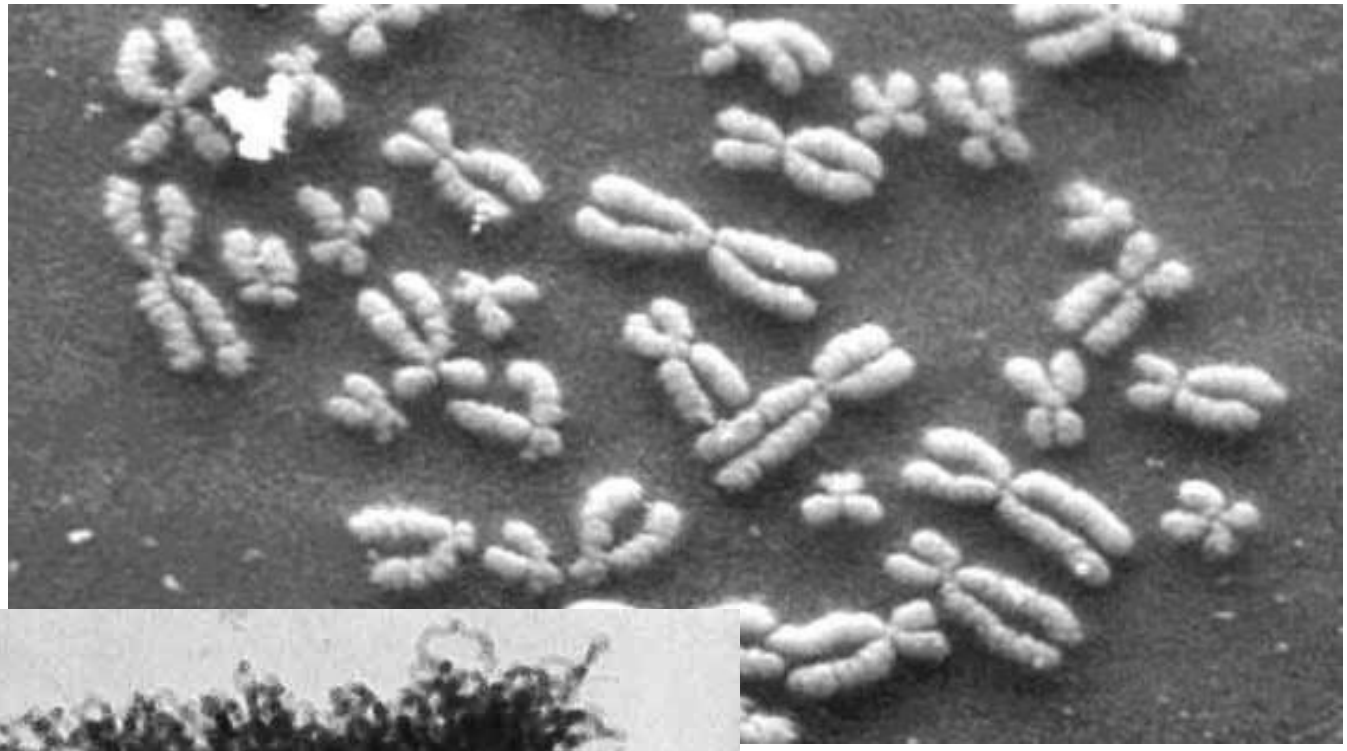
ETH zürich UC | Universidad de Cantabria POSTECH

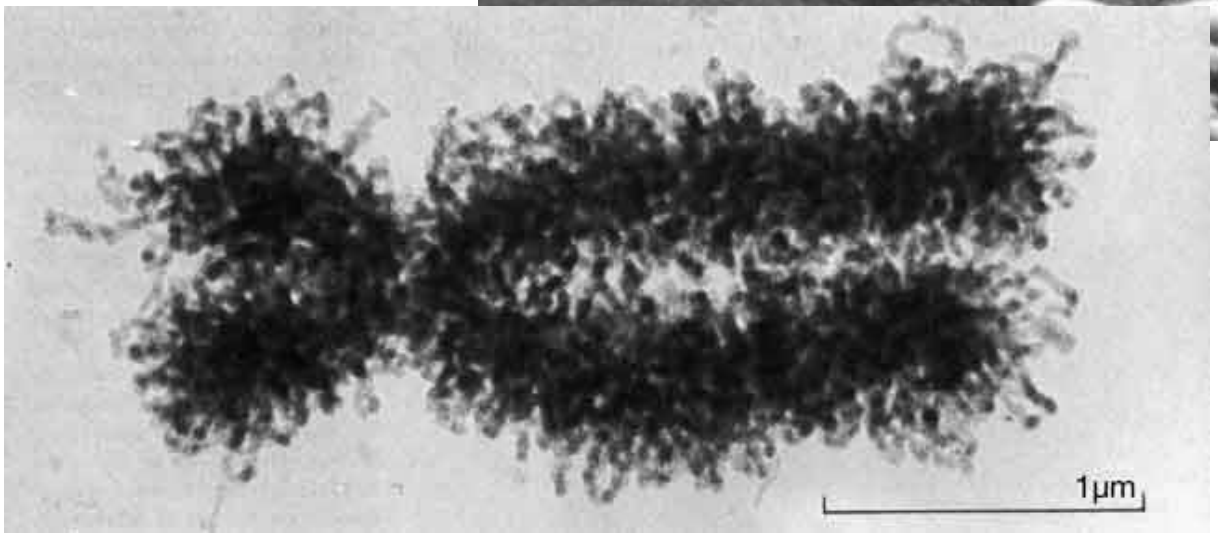
Applications of Genome Analysis



& Many More…

DNA Under Electron Microscope





human chromosome #12
from HeLa cell

High-Throughput Sequencers

Illumina MiSeq

Pacific Biosciences Sequel II

Oxford Nanopore PromethION

Oxford Nanopore MinION

Illumina NovaSeq 6000

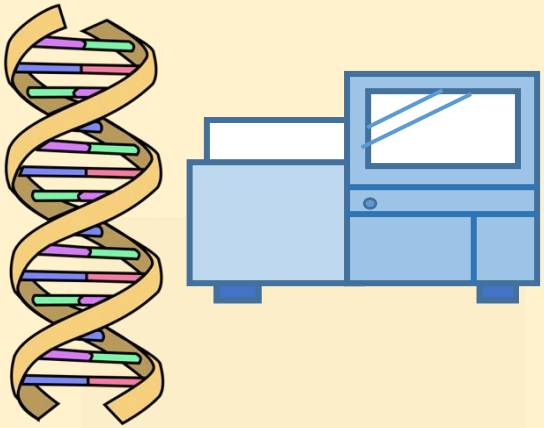
Pacific Biosciences RS II

Oxford Nanopore GridION

… and more! All produce data with different properties.

High-Throughput Sequencers

Sequencing technologies
extract smaller fragments of the original DNA sequence,
known as reads



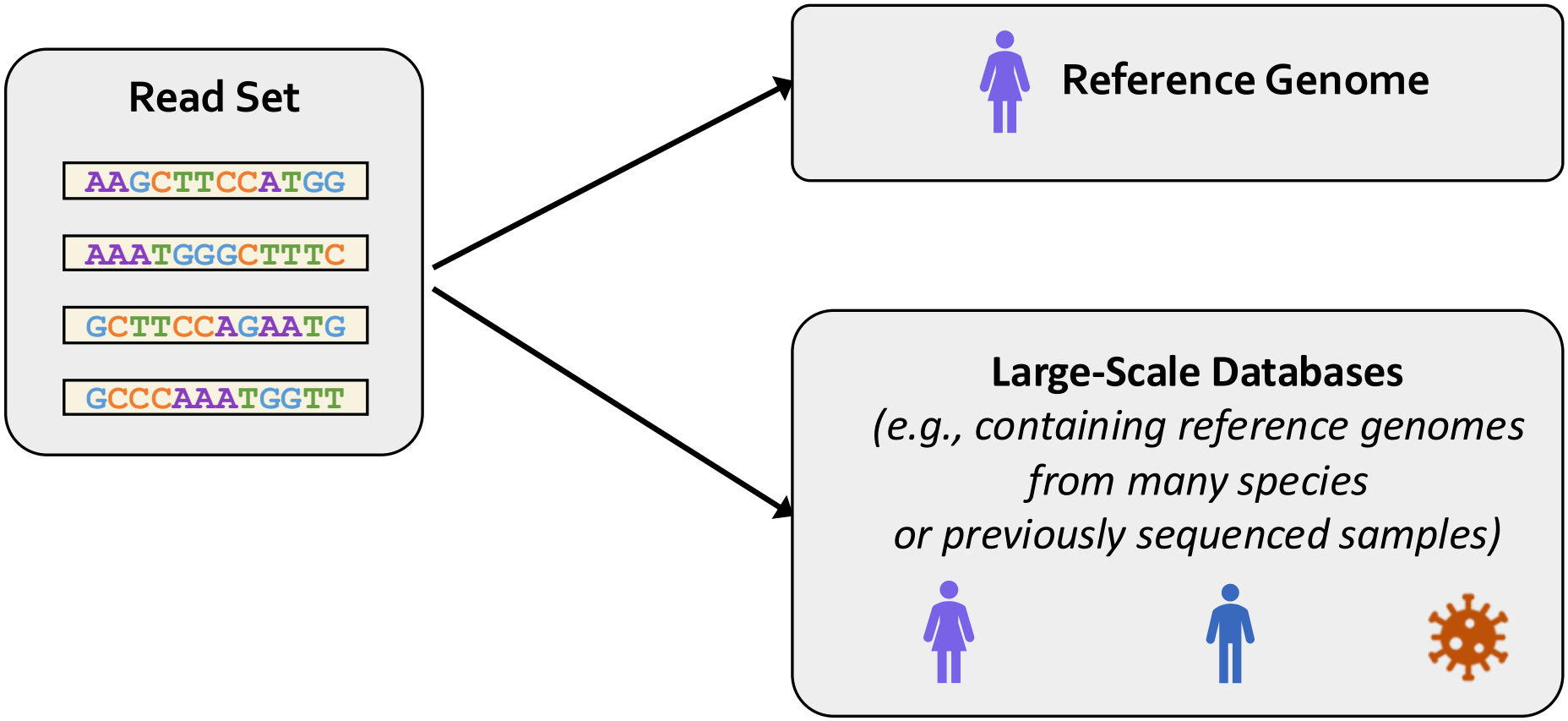
Oxford

Illumina NovaSeq 6000

Pacific Biosciences RS II

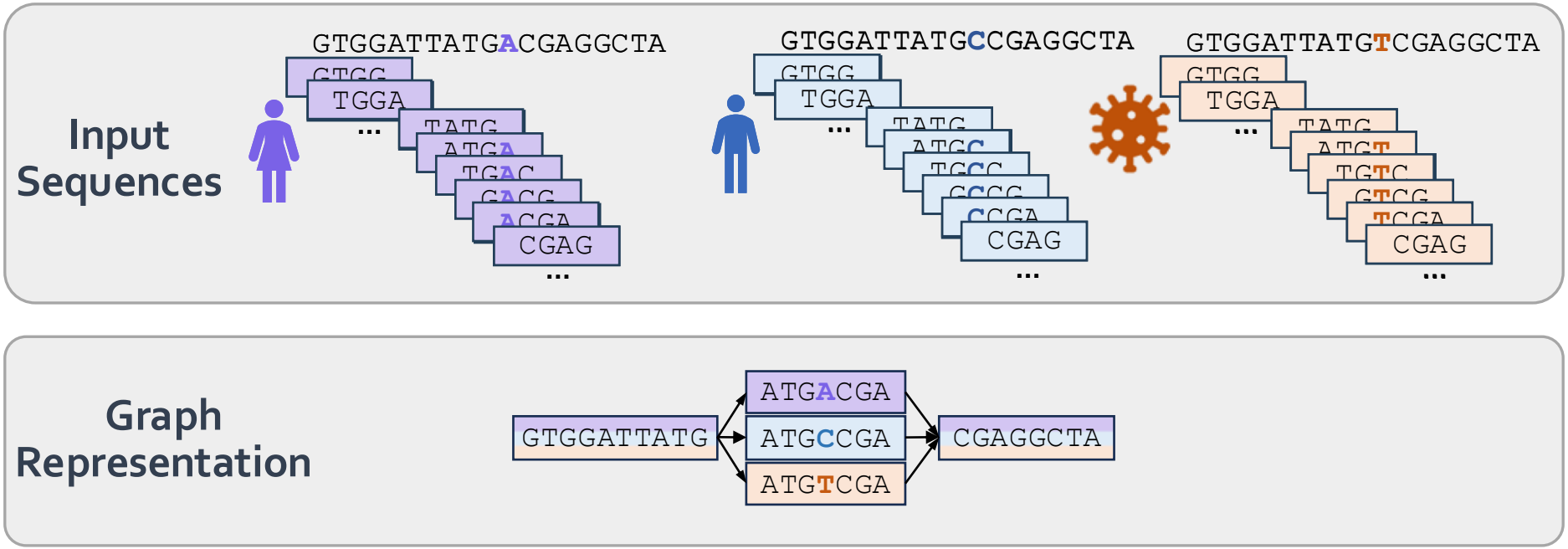
... and more! All produce data with different properties.

Genome Analysis



- ***Determine species or genomic features*** *present in the sample*
- ***Find differences*** *from genomes in the reference genome or database*

Large-Scale Genome Graphs



Powerful and efficient representation of genomic data

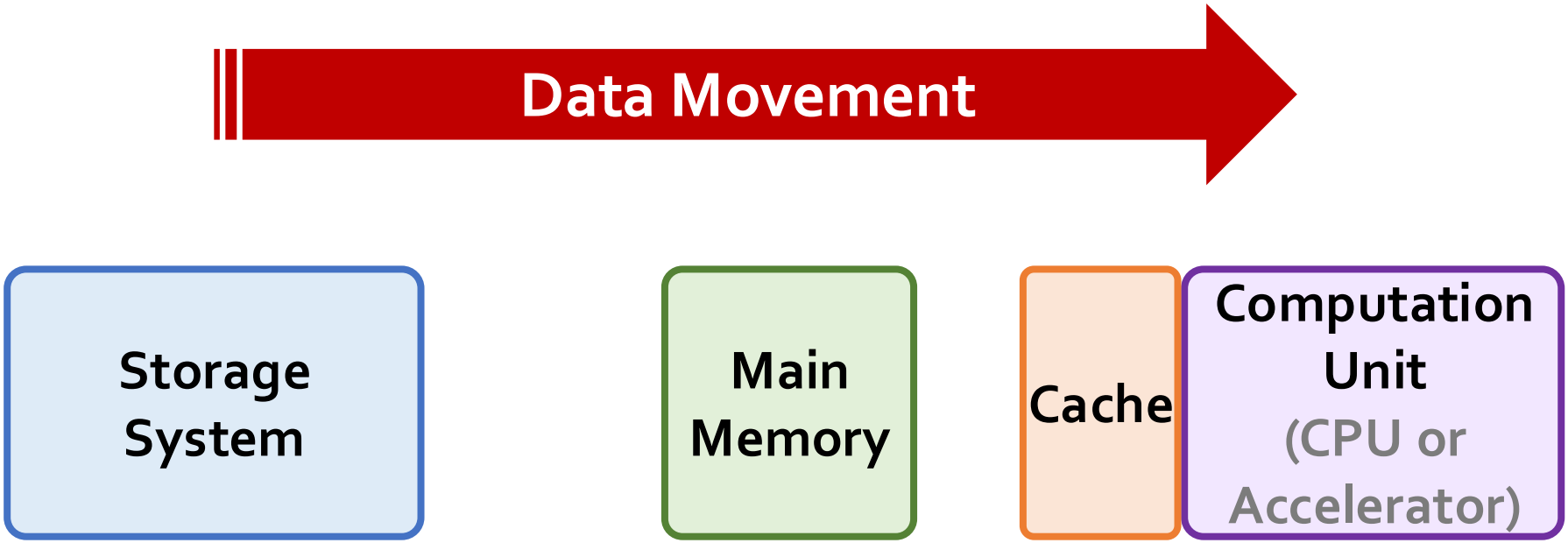
✓ Great expressive power

✓ Compact representation

Large-Scale Genome Graphs

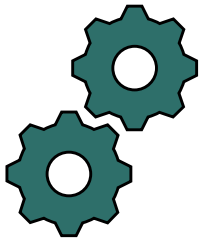
Critical for enabling massive, population-scale analyses

Data Movement Overheads



Data movement overhead

Storage-Centric Computing



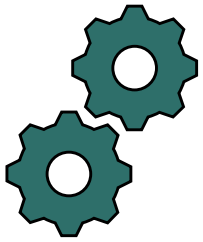
Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

Challenges



Storage System

Main Memory

Cache

Computation Unit (CPU or Accelerator)

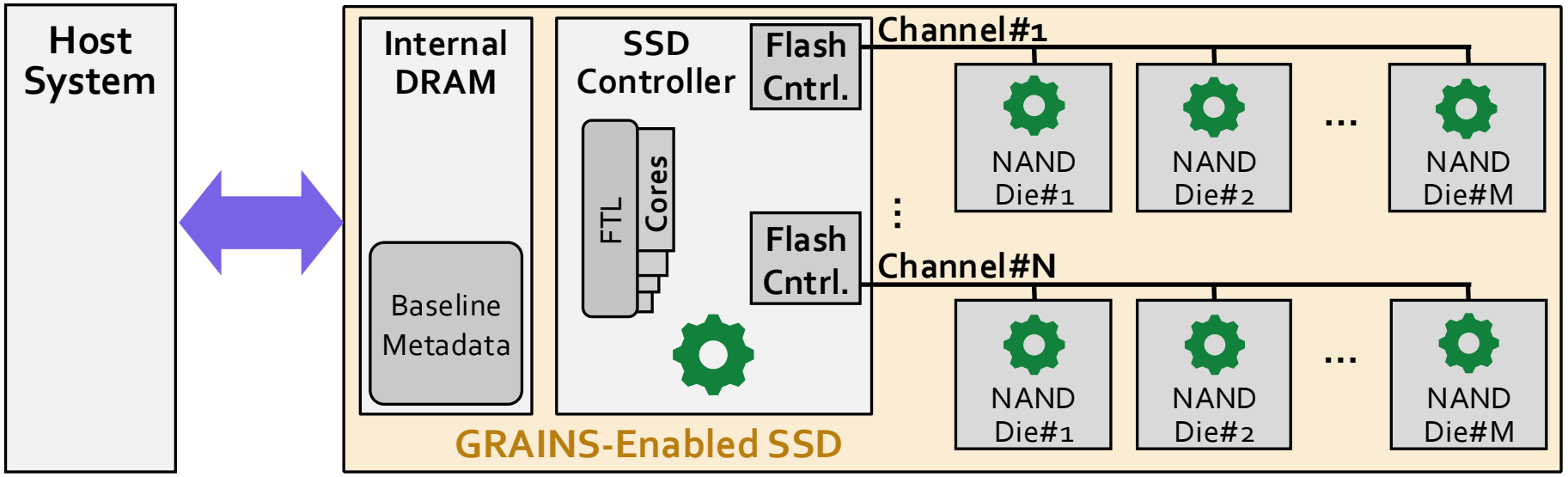
Graph-based sequence analysis tools cannot run in storage due to SSD limitations

Dependent and random accesses to the sequence graph

Different properties from conventional, non-genome graphs

GRAINS

- The *first* system for analysis with large-scale genome graphs in storage
- Enabled via **storage-aware algorithm-architecture co-design**, based on the **properties of large-scale genome graphs** to
  - Make the execution pipeline **more storage-friendly**
  - Further improve **performance**, **energy-efficiency**, and **cost-effectiveness** via **storage-centric computing**

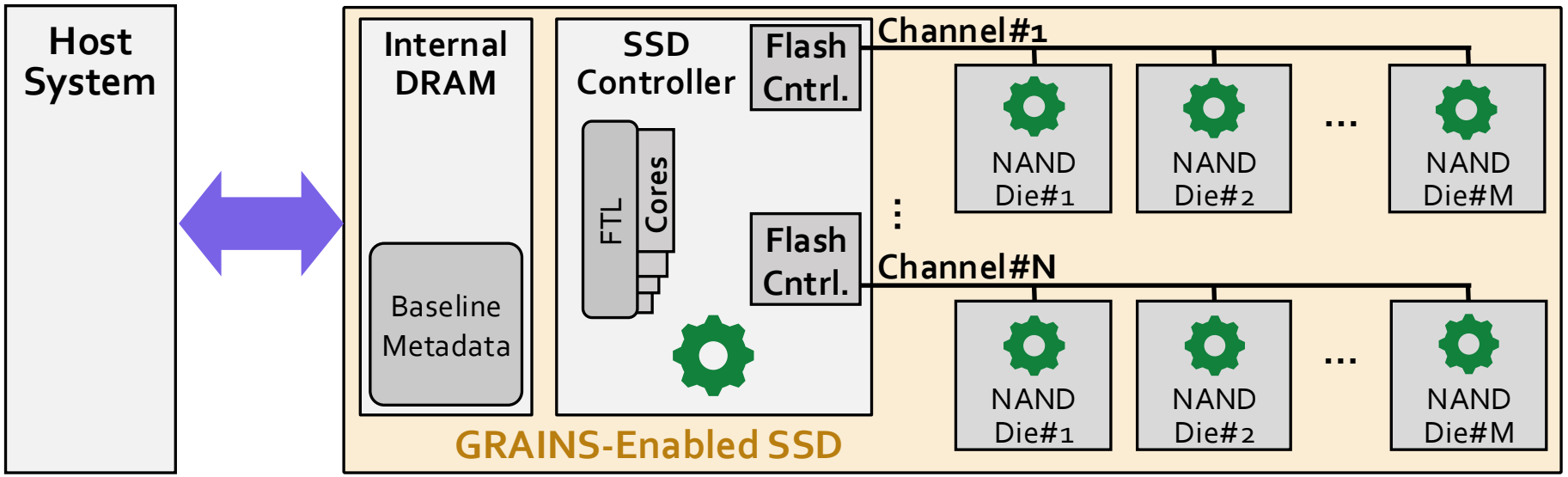


GRAINS

Higher performance (by 1.5×–47.8×)

Higher energy efficiency (by 3.1×–31.6×)

Higher cost effectiveness



Outline

Background

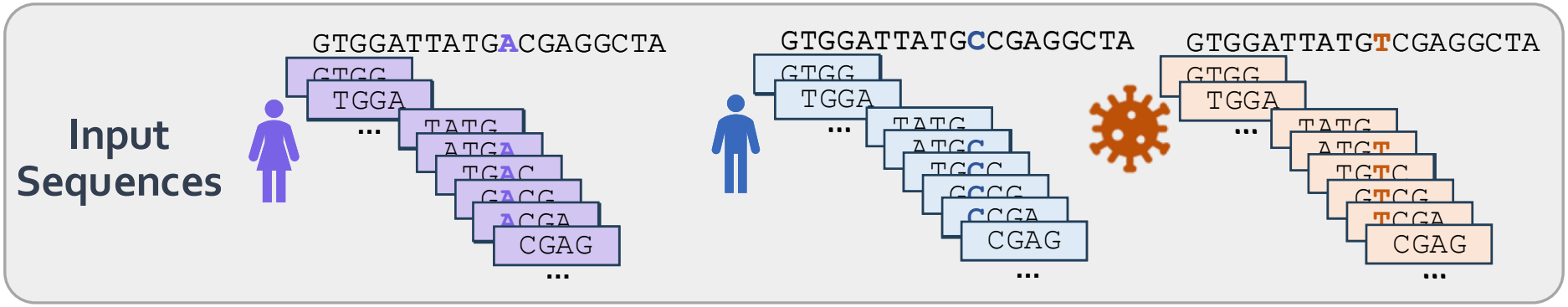
Motivation and Goal

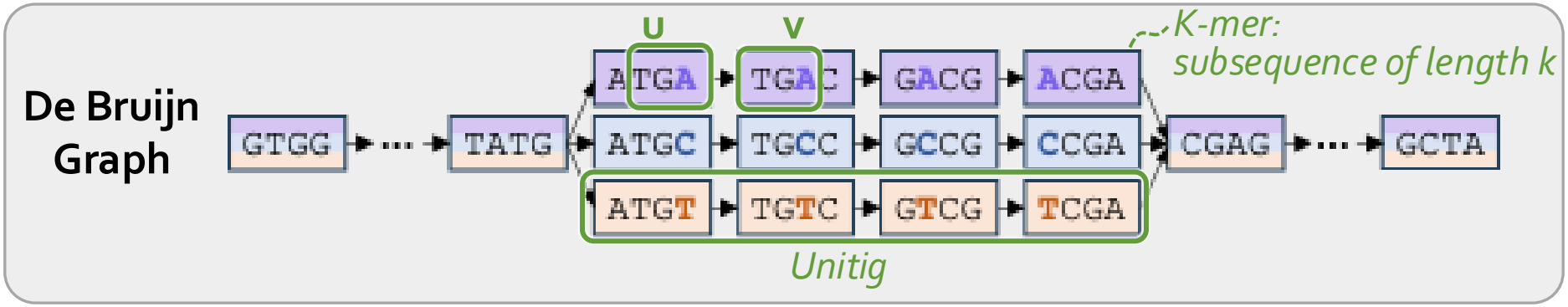
GRAINS

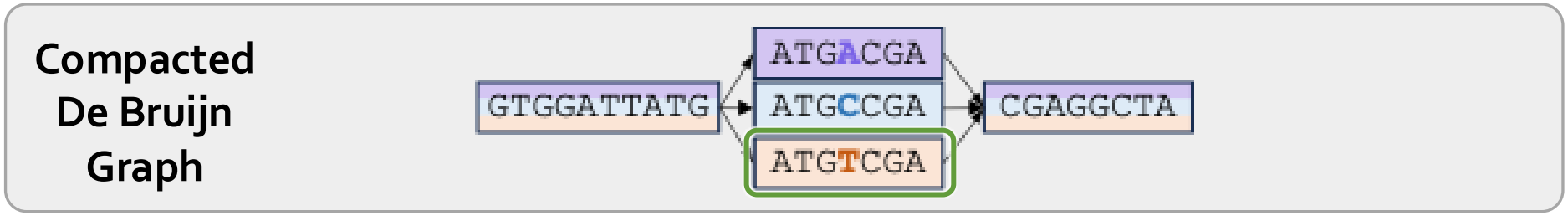
Evaluation

Conclusion

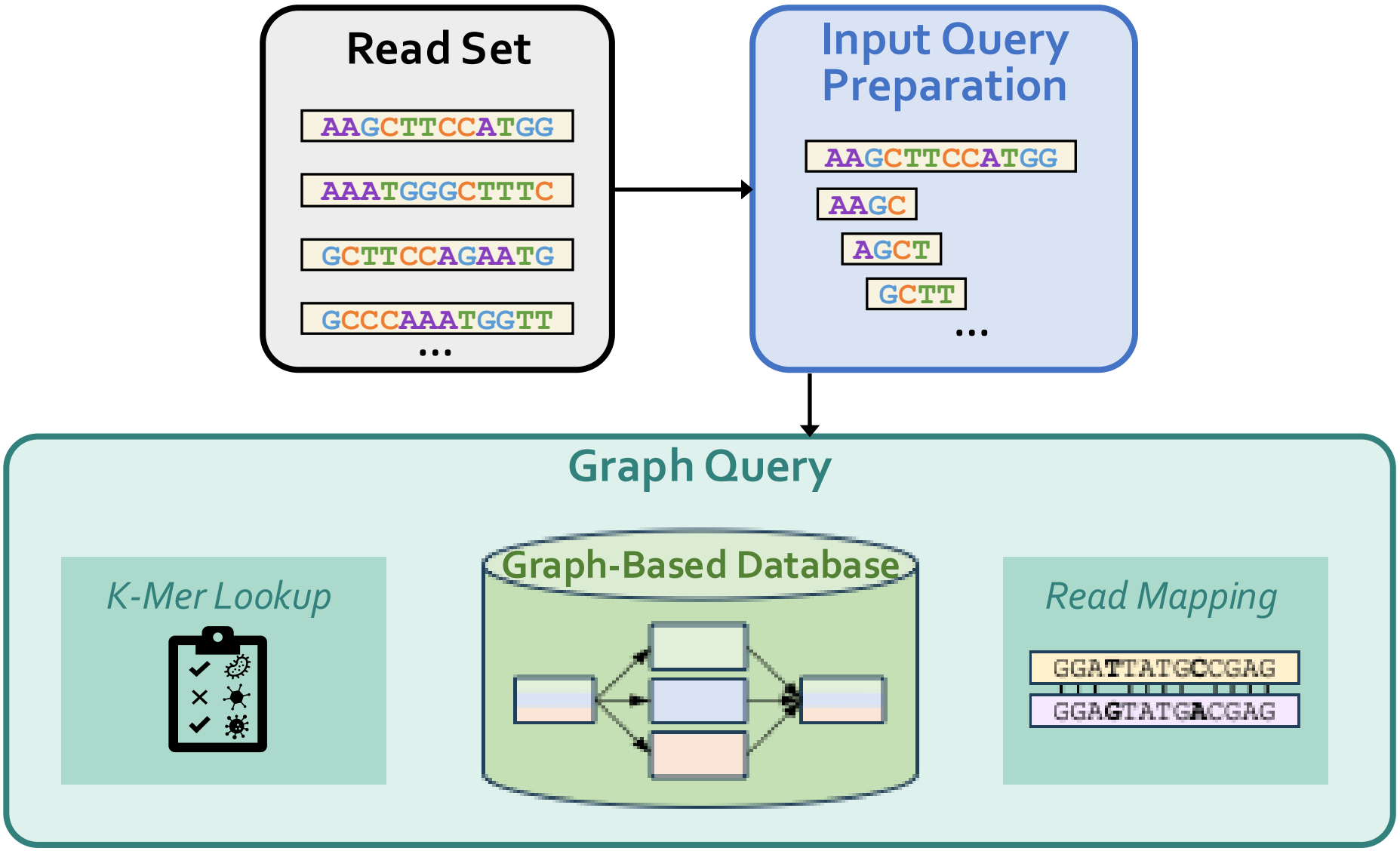
Genome Graphs







Graph-Based Genome Analysis



Outline

Background

Motivation and Goal

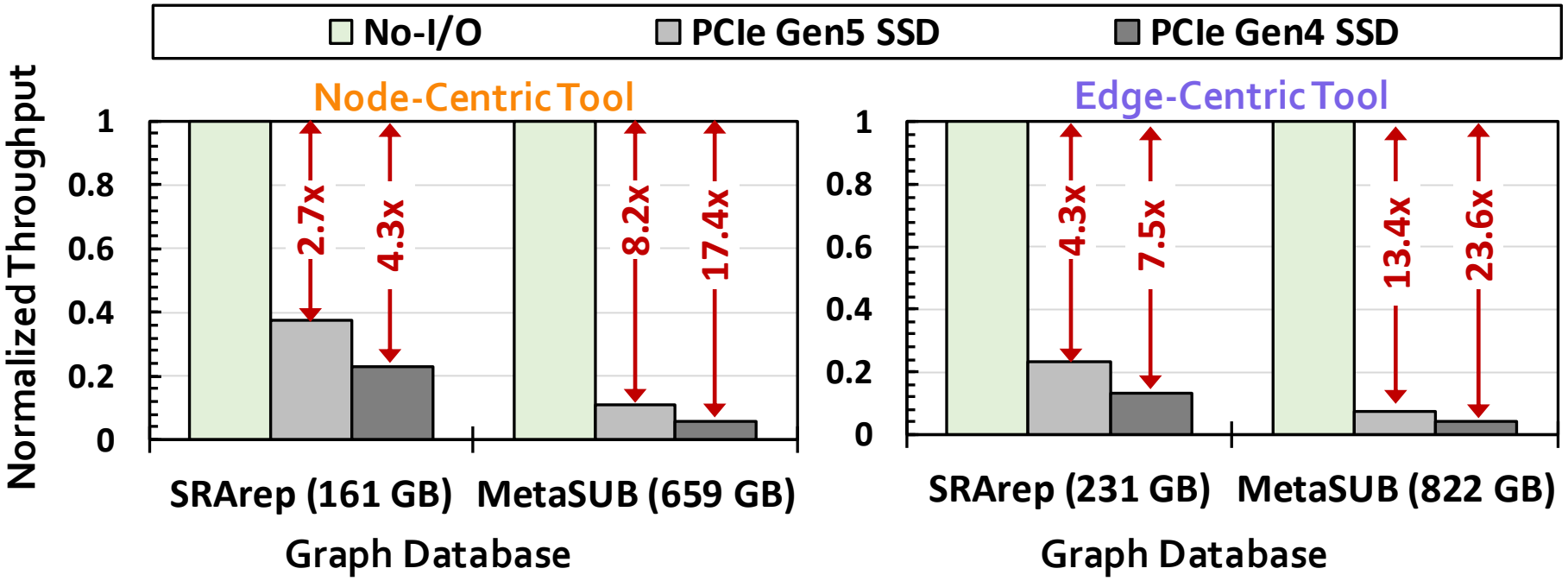
GRAINS

Evaluation

Conclusion

Motivation

- Case study of the performance of graph-based genome analysis tools
- With various state-of-the-art SSD configurations



I/O data movement causes significant performance overhead

Overhead increases as databases grow

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 graph-based genome analysis completely and always resident in main memory

✗ *Energy inefficient, costly, unscalable, and unsustainable*

- Database sizes **increase rapidly**
- Different analyses need **different databases**

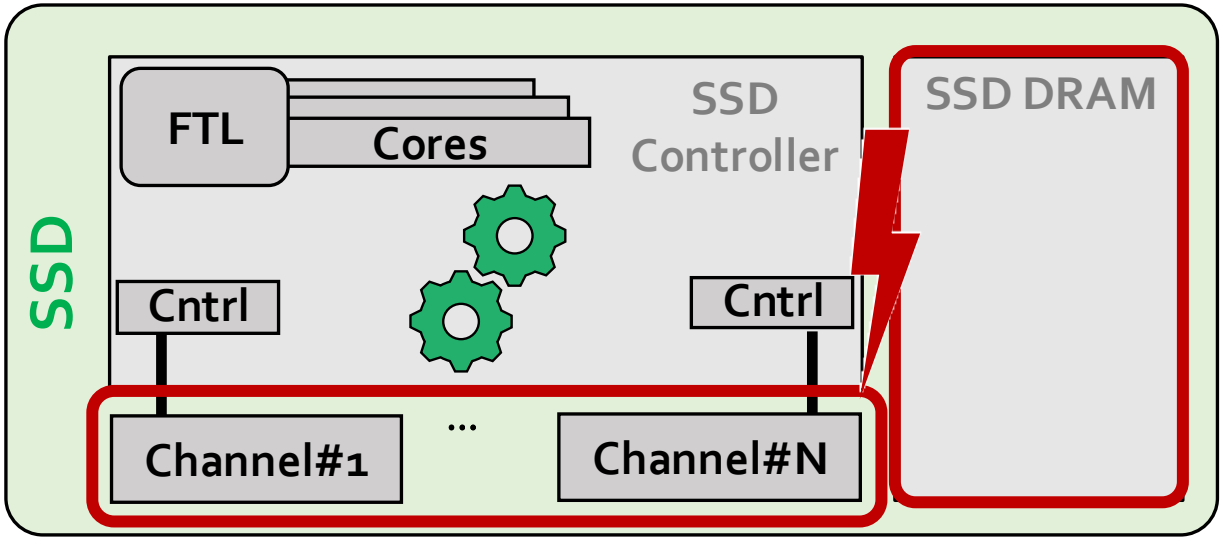
Our Goal

Improve graph-based genome 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 graph-based genome analysis tools can run in-storage due to SSD limitations

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



Challenges of In-Storage Processing

No graph-based genome analysis tools can run in-storage due to SSD limits

- Long latency of NAND flash chips

Dependent and random accesses to the sequence graph

- Limited DRAM bandwidth inside the SSD

Different properties from conventional, non-genome graphs

Outline

Background

Motivation and Goal

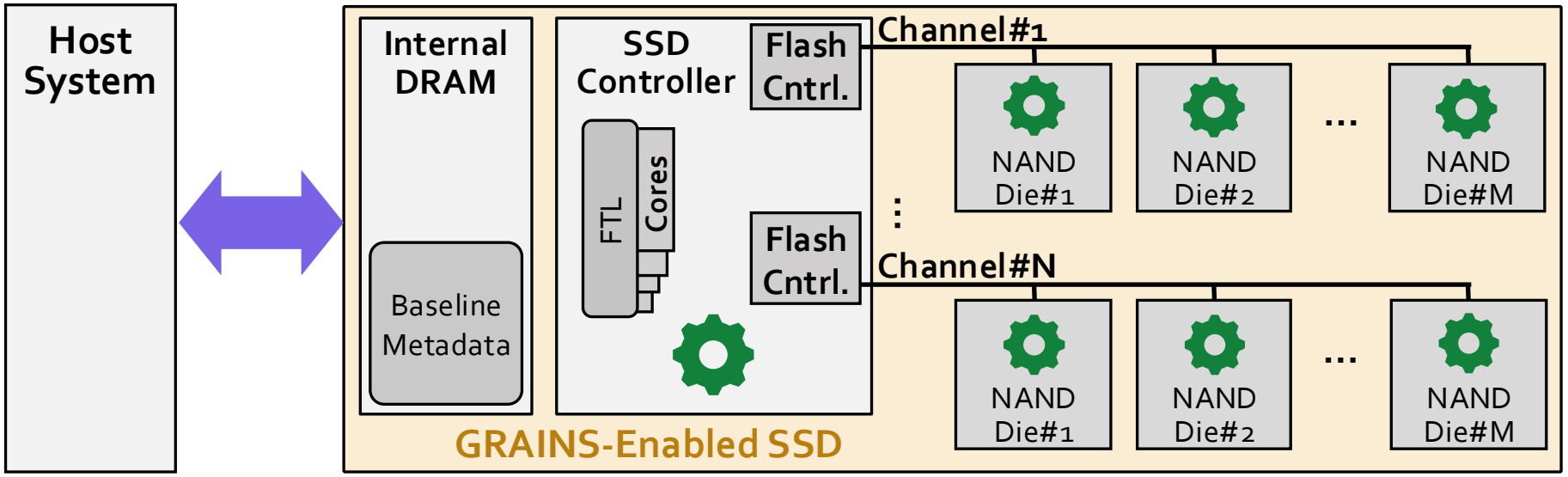
GRAINS

Evaluation

Conclusion

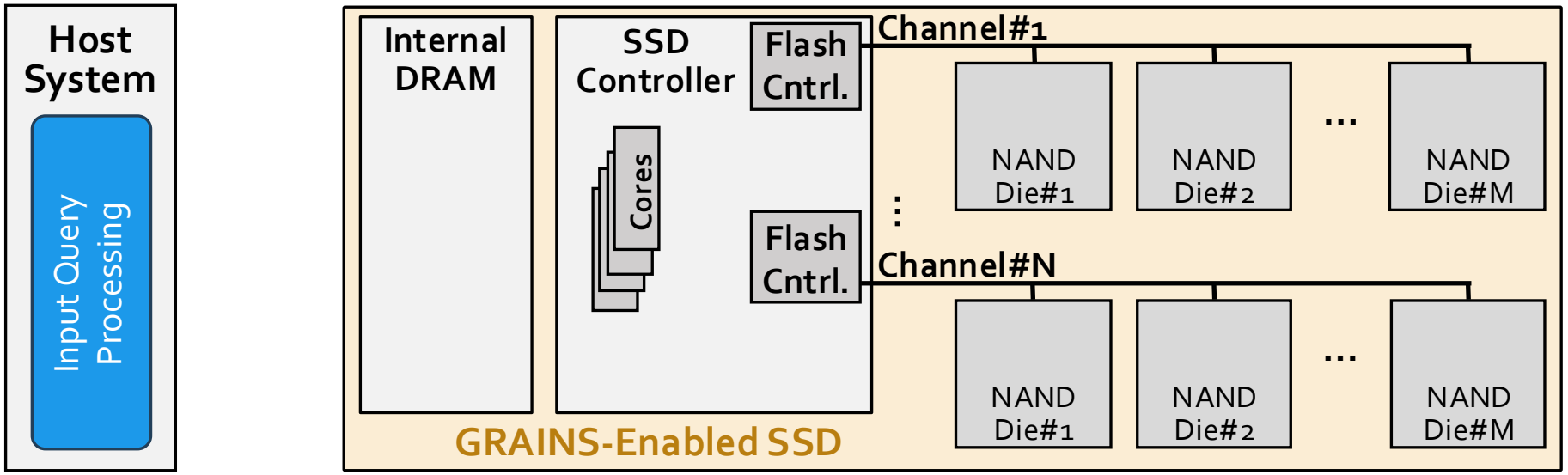
GRAINS

- The *first* system for analysis with large-scale genome graphs in storage
- Enabled via **storage-aware algorithm-architecture co-design**, based on the **properties of large-scale genome graphs** to
  - Make the execution pipeline **more storage-friendly**
  - Further improve **performance**, **energy-efficiency**, and **cost-effectiveness** via **storage-centric computing**



GRAINS's Algorithm-Architecture Co-Design

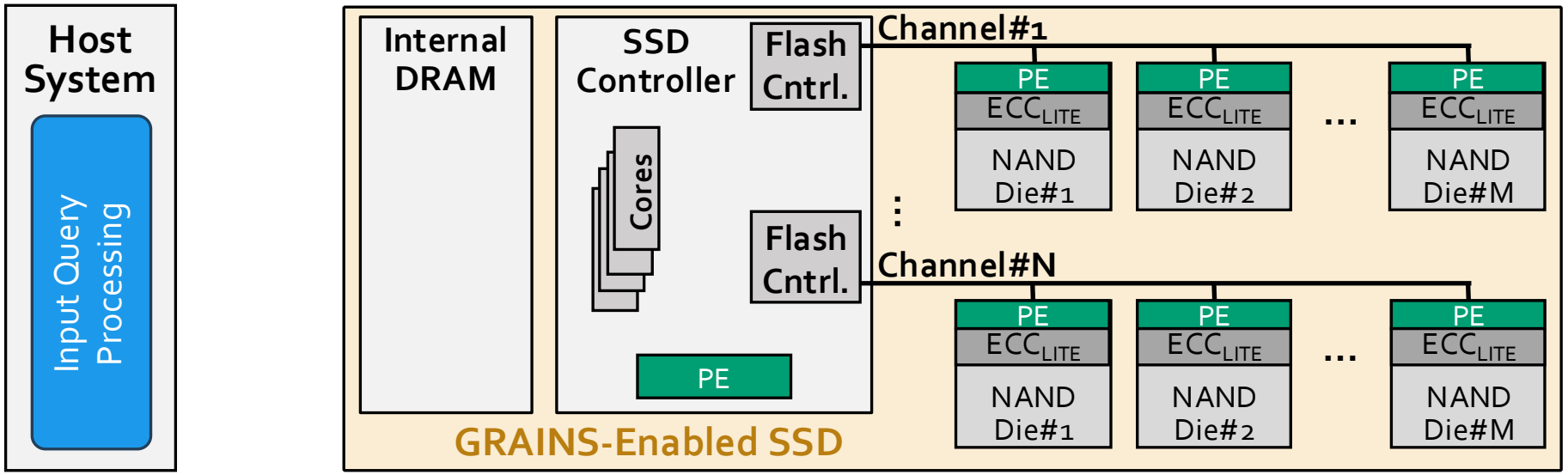
New batching technique and execution flow
based on the unique features of genome graphs that reduces #random accesses



GRAINS's Algorithm-Architecture Co-Design

New batching technique and execution flow
based on the unique features of genome graphs that reduces #random accesses

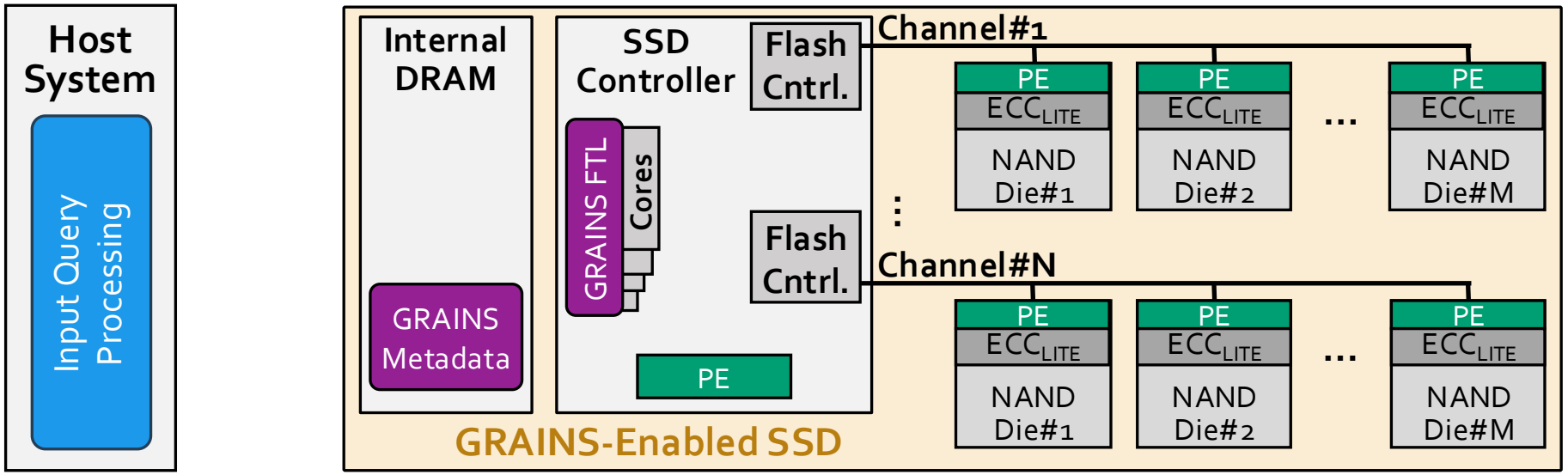
In-storage and In-flash processing
to avoid transferring low-reused pages and bandwidth waste



GRAINS's Algorithm-Architecture Co-Design

New batching technique and execution flow
based on the unique features of genome graphs that reduces #random accesses

In-storage and In-flash processing
to avoid transferring low-reused pages and bandwidth waste

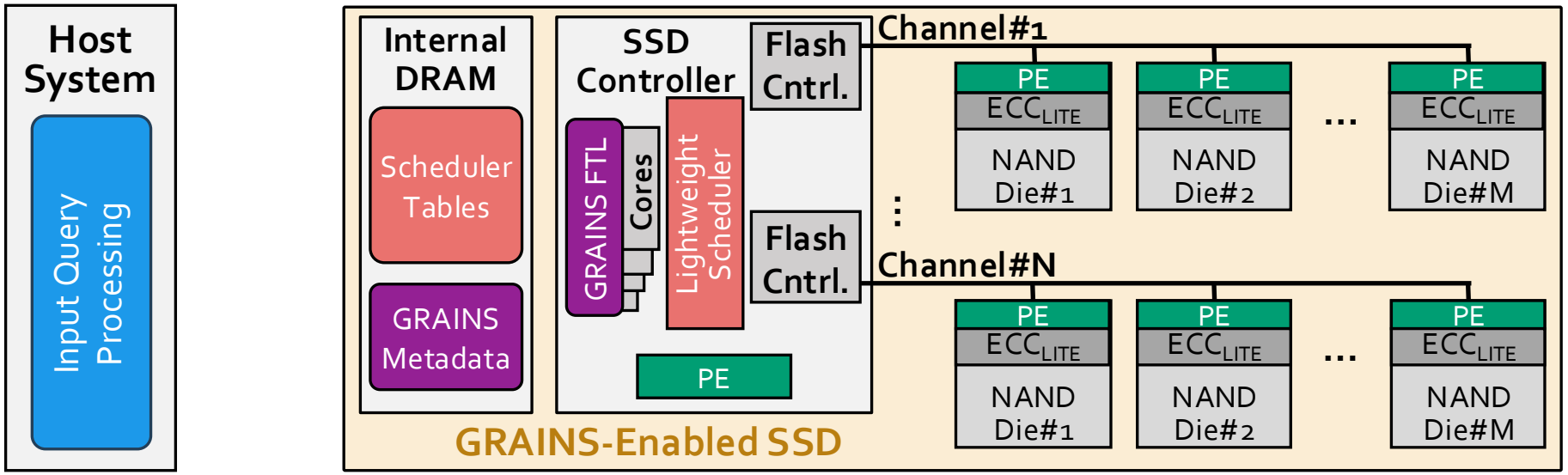


FTL and data placement
based on genome graph properties, to leverage SSD's full internal bandwidth

GRAINS's Algorithm-Architecture Co-Design

New batching technique and execution flow
based on the unique features of genome graphs that reduces #random accesses

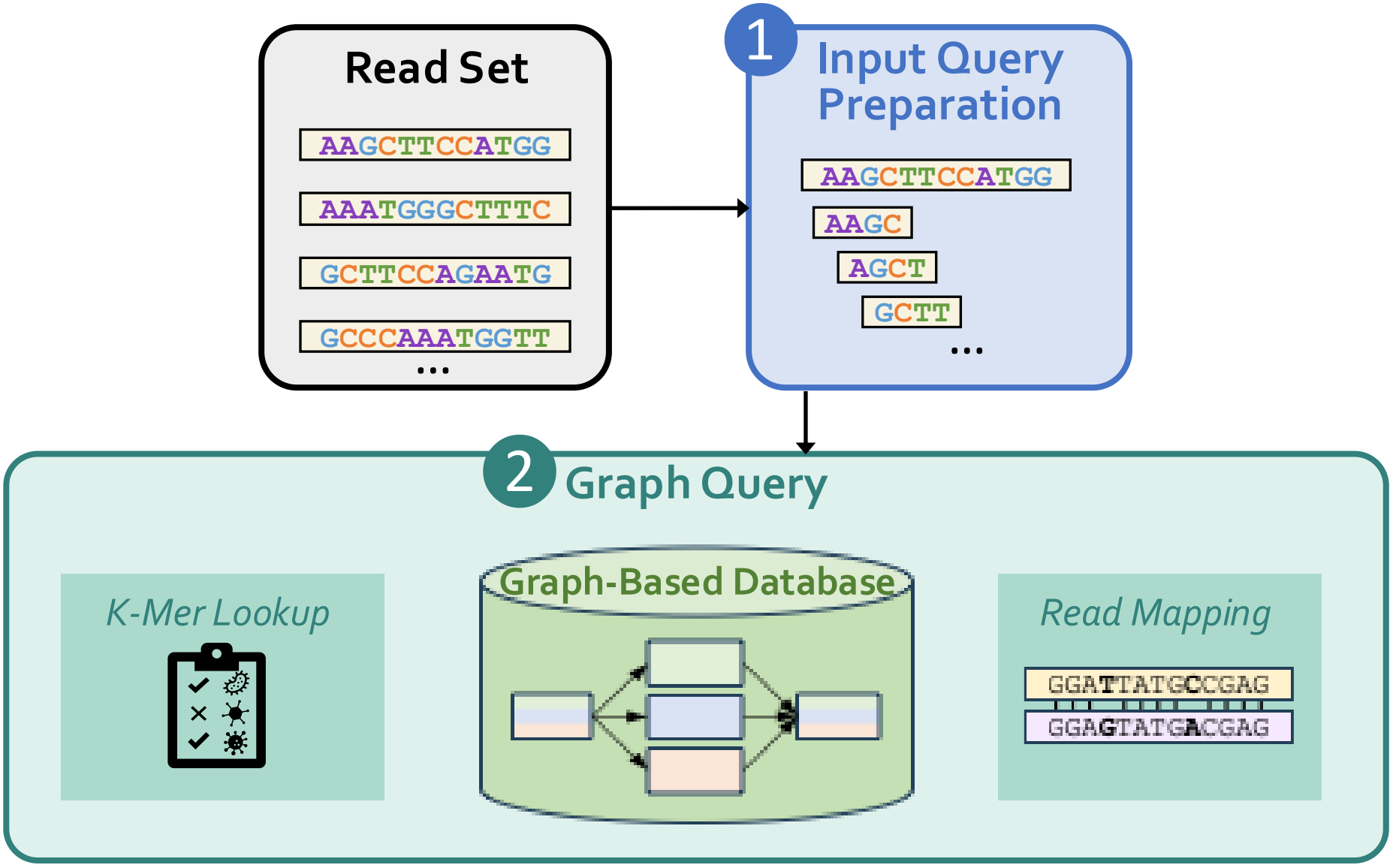
In-storage and In-flash processing
to avoid transferring low-reused pages and bandwidth waste



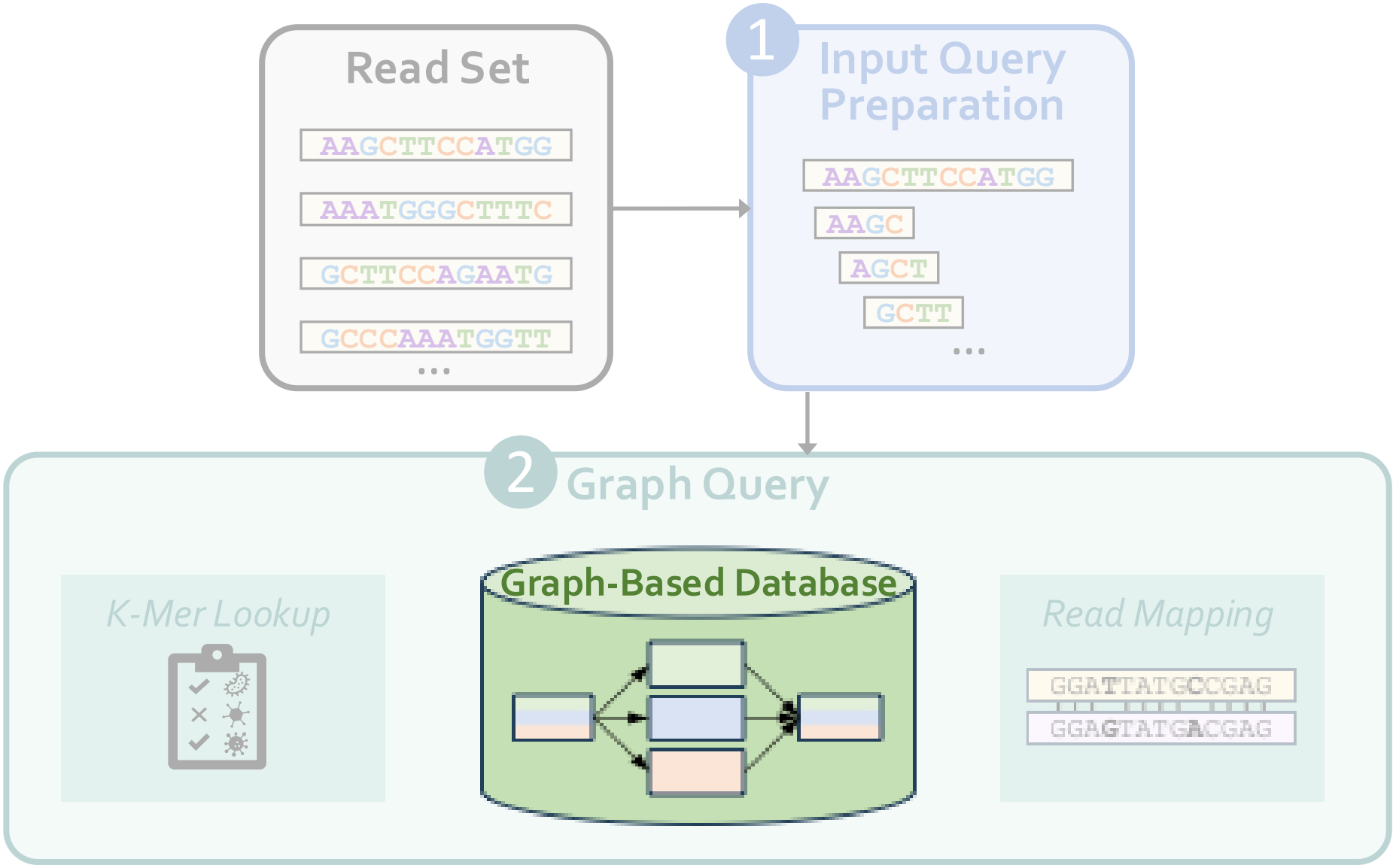
FTL and data placement
based on genome graph properties, to leverage SSD's full internal bandwidth

Lightweight scheduling technique
enabled by repurposing the existing SSD structures to leverage the full parallelism of all flash dies

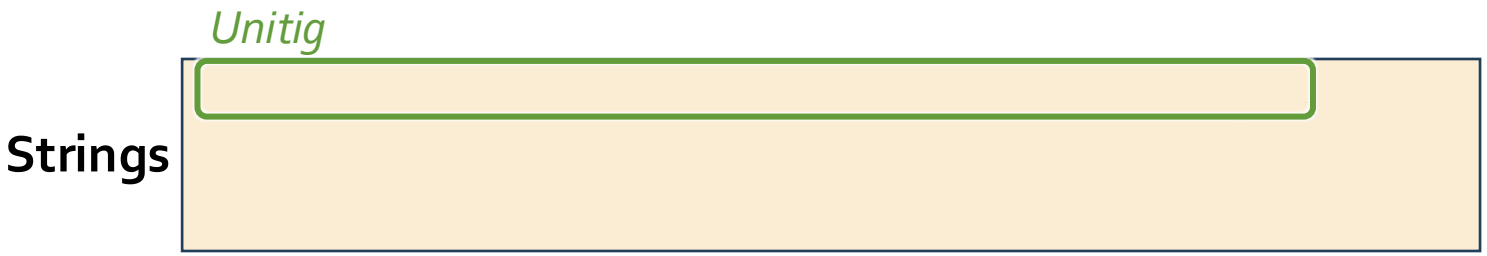
GRAINS


Read Set
AAGCTTCCATGG
AAATGGGCTTTC
GCTTCCAGAATG
GCCCAAATGGTT
...
1
Input Query Preparation
AAGCTTCCATGG
AAGC
AGCT
GCTT
...
2
Graph Query
K-Mer Lookup
Graph-Based Database
Read Mapping
GGATTATGCCGAG
GGAGTATGACGAG

Data Structures

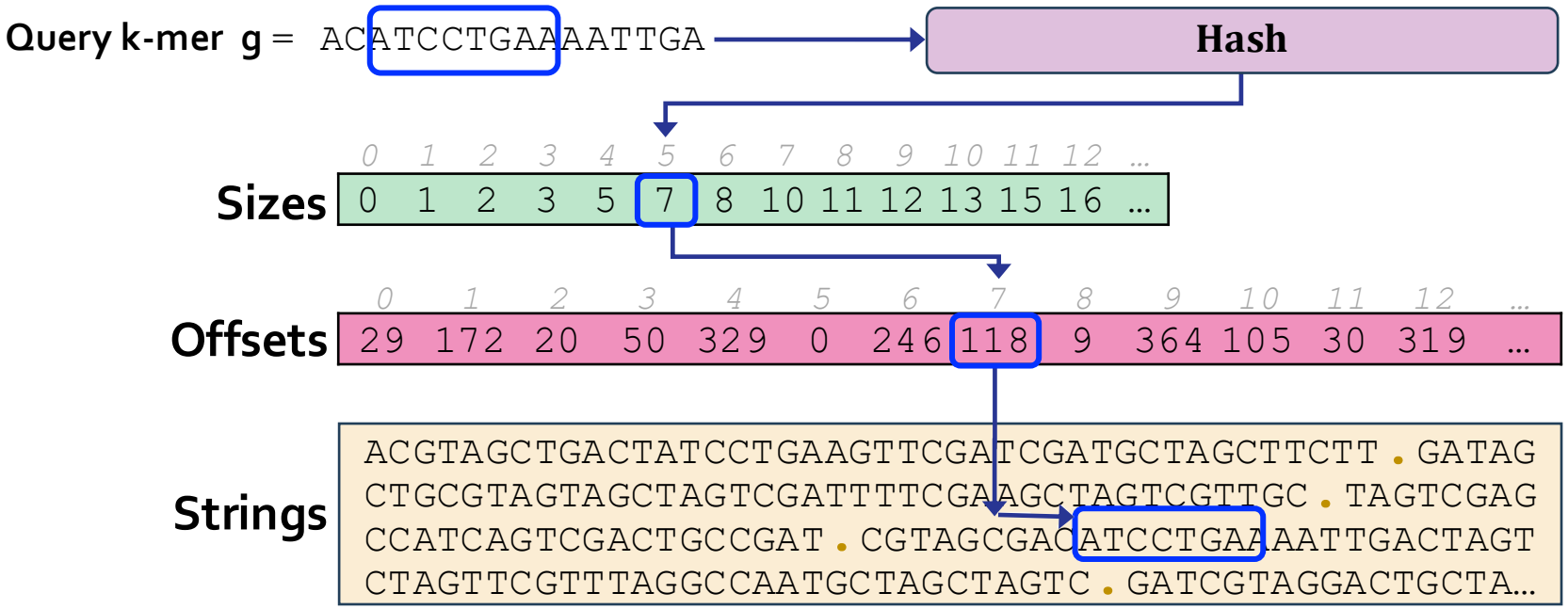


Data Structures

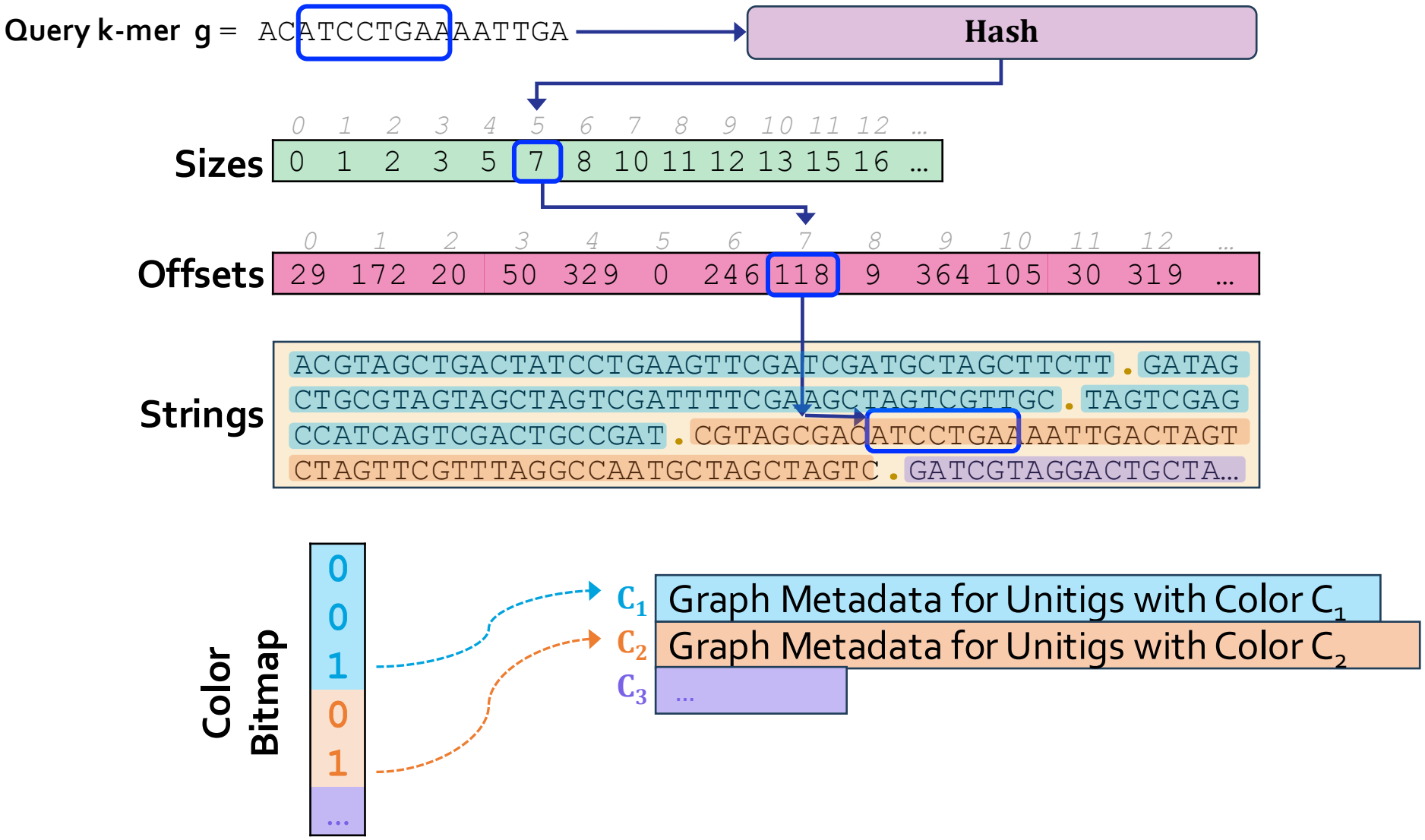


[SSHash, Bioinformatics'22]

Data Structures

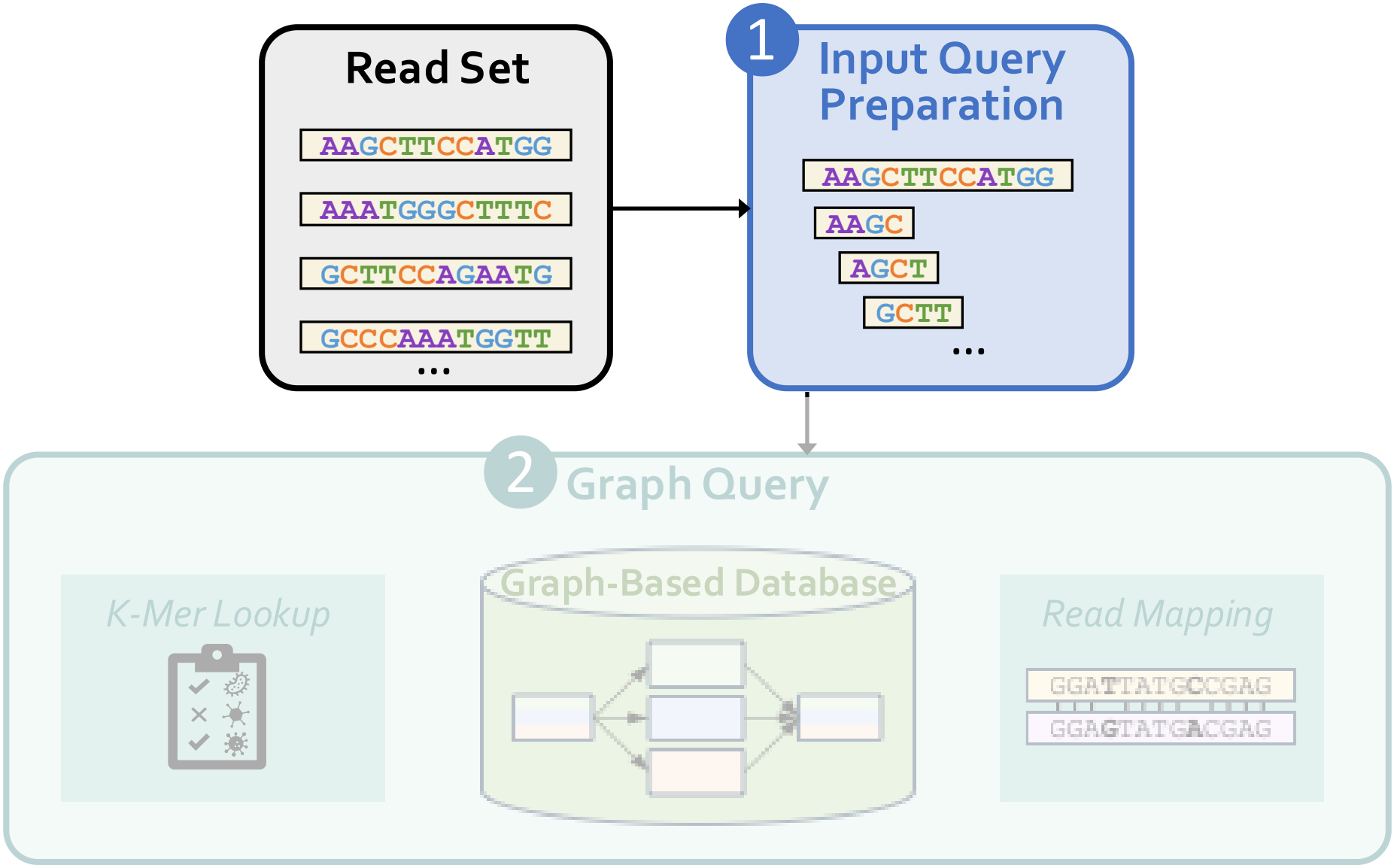


Data Structures



[SSHash, Bioinformatics'22][Fulgor, WABI'23]

Input Query Processing

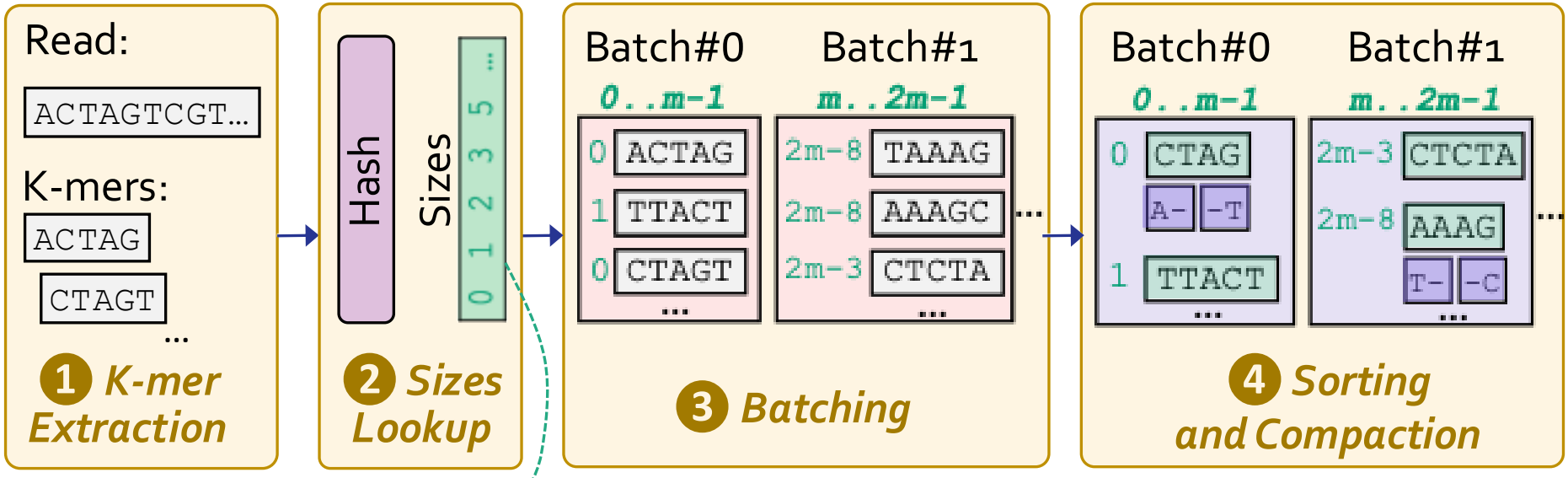


Input Query Processing

- **GRAINS prepares reads for efficient queries in the next step**
  - *Performed in the **host system** due to **large #writes***

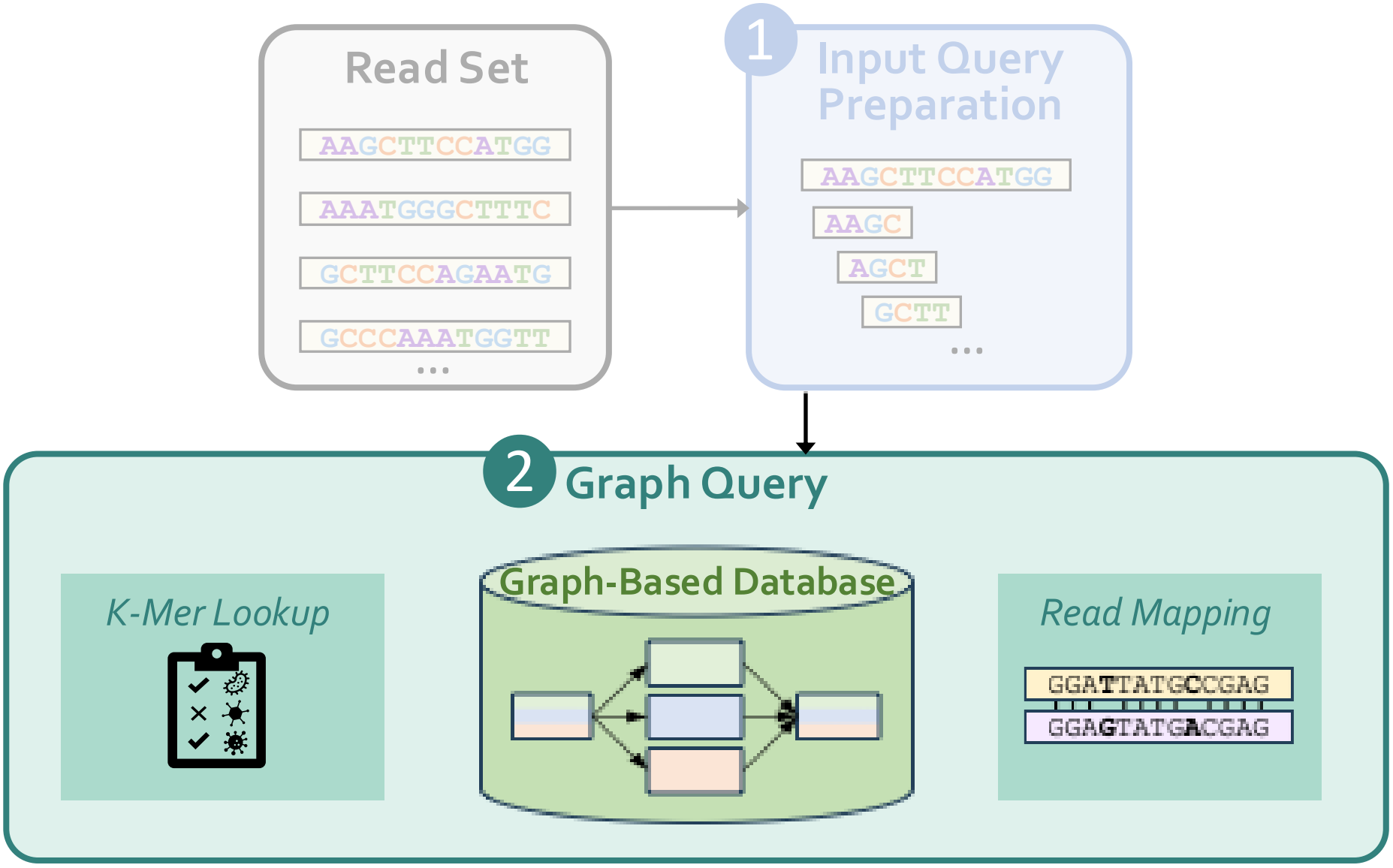
Cross-read k-mer batching
to reduce #random accesses

Genome-graph-aware query reordering
to improve access patterns



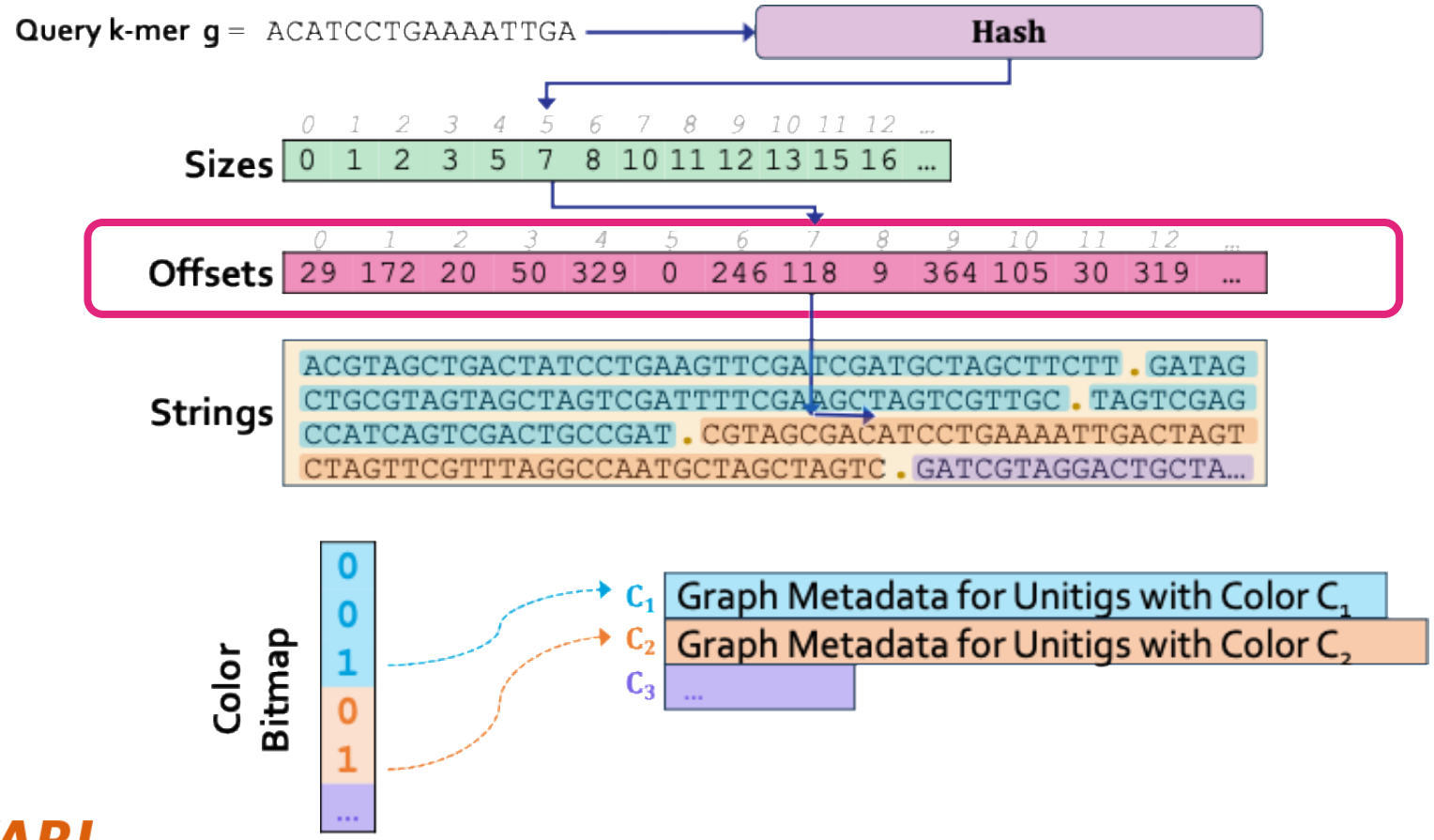
Significantly smaller than other sub-structures + *Enables efficient batching and compaction*

Graph Query


Read Set
AAGCTTCCATGG
AAATGGGCTTTC
GCTTCCAGAATG
GCCCAAATGGTT
...
1
Input Query Preparation
AAGCTTCCATGG
AAGC
AGCT
GCTT
...
2
Graph Query
K-Mer Lookup
Graph-Based Database
Read Mapping
GGATTATGCCGAG
GGAGTATGACGAG

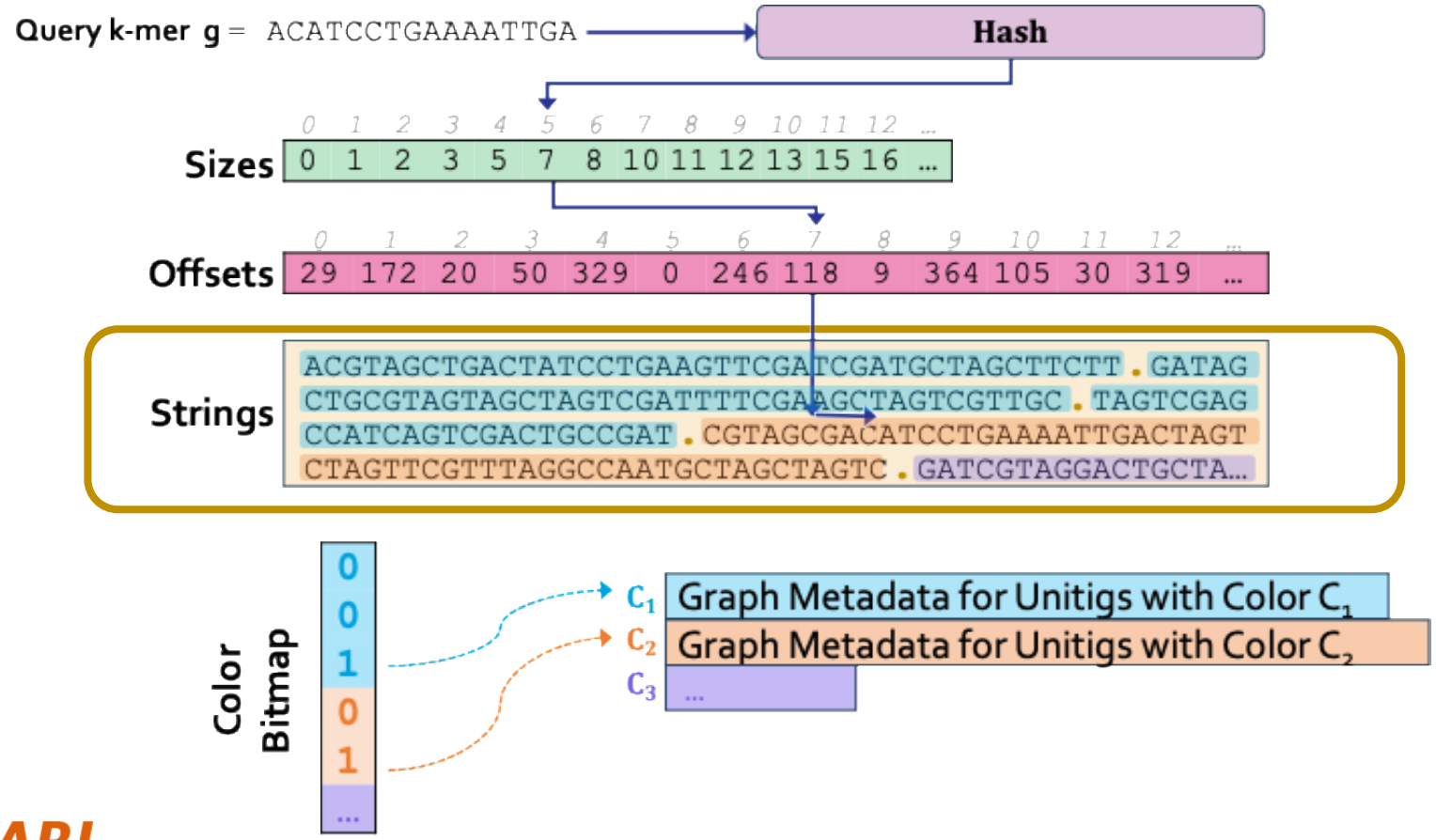
Graph Query: Overview

- **GRAINS looks up the query k-mers in the graph and retrieves their associated IDs**
  - By accessing **Offsets**



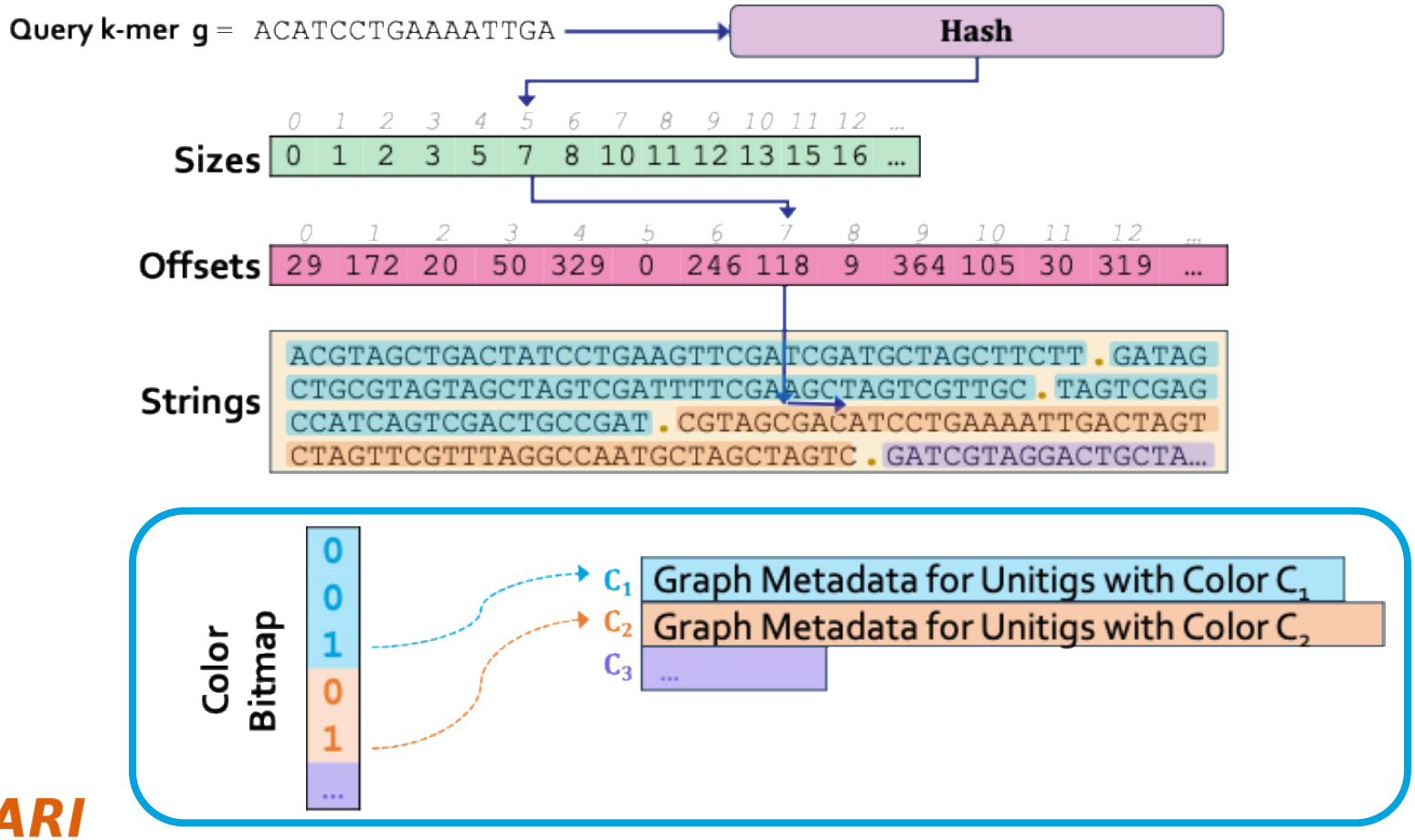
Graph Query: Overview

- **GRAINS looks up the query k-mers in the graph and retrieves their associated IDs**
  - By accessing **Offsets**, **Strings**



Graph Query: Overview

- **GRAINS looks up the query k-mers in the graph and retrieves their associated IDs**
  - By accessing **Offsets**, **Strings**, and **Colors**



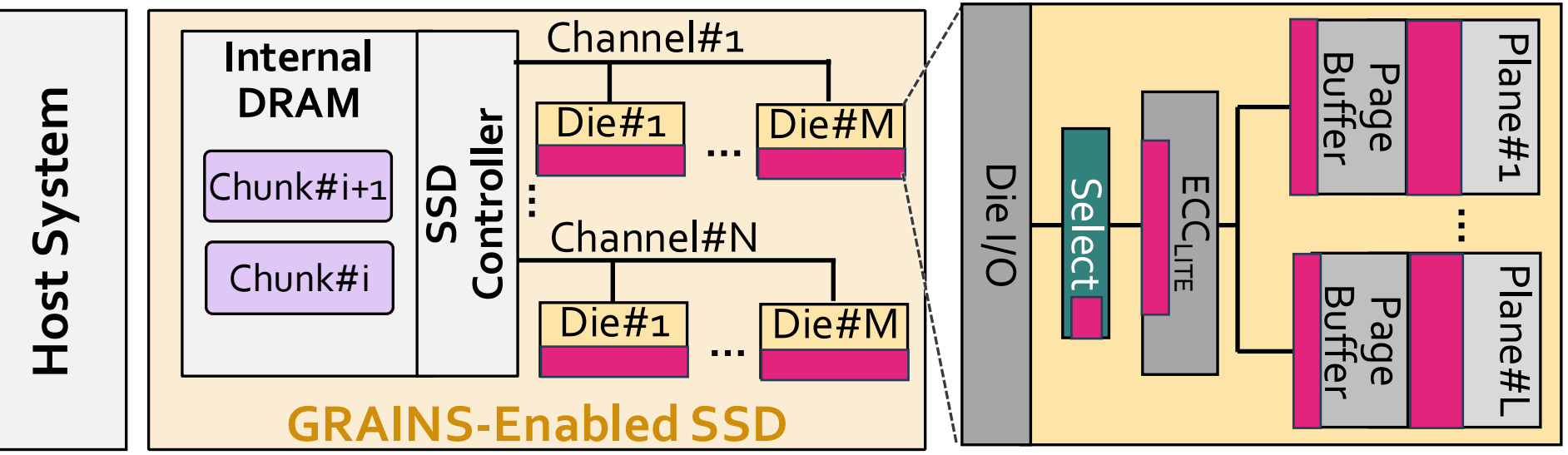
Graph Query: Overview

- **GRAINS looks up the query k-mers in the graph and retrieves their associated IDs**
  - By accessing **Offsets**, **Strings**, and **Colors**

- **GRAINS executes this step inside the SSD since it**
  - Involves accessing **large graph structures** with **low reuse**
  - Requires only **lightweight computation**

- **To efficiently execute this step, GRAINS needs to**
  - Leverage the **large internal bandwidth**
  - Not require **expensive hardware resources in the SSD** *(e.g., costly logic units or large DRAM capacity/bandwidth)*

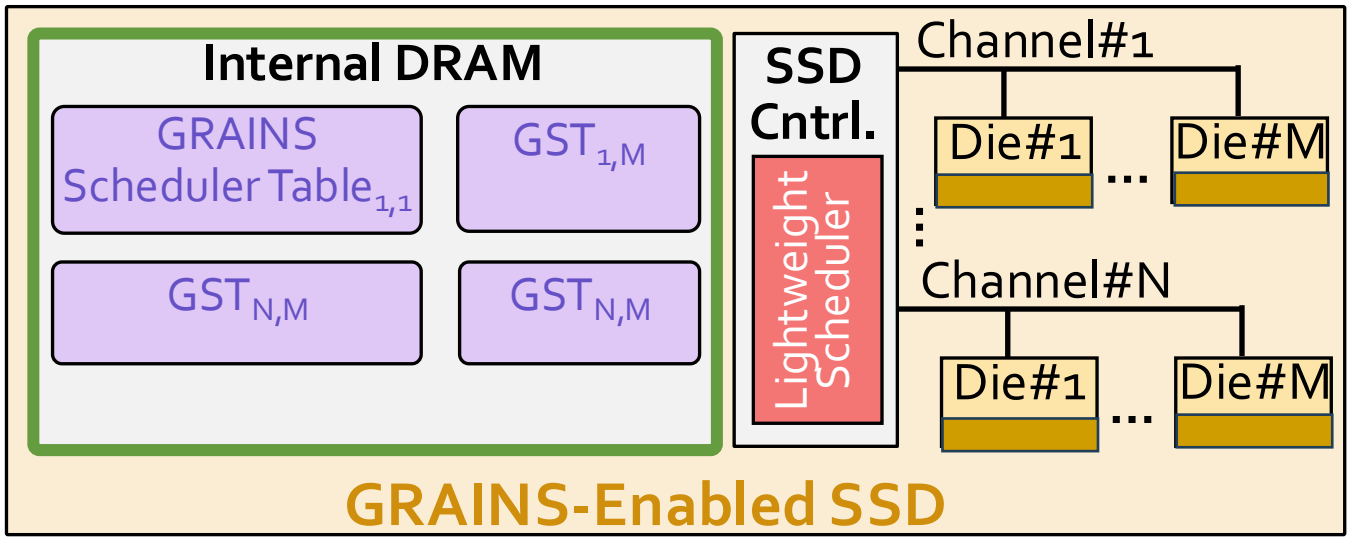
Graph Query: Offsets



- *Efficient **pipeline** between host and SSD*
- *Sequential accesses to fully **exploit internal bandwidth***
- *Only a **small part of each page moves** outside the dies*

Graph Query: Strings

- We aim for Strings accesses to fully exploit internal bandwidth as well
  - However, unlike Offsets, the indices into Strings arrive unsorted



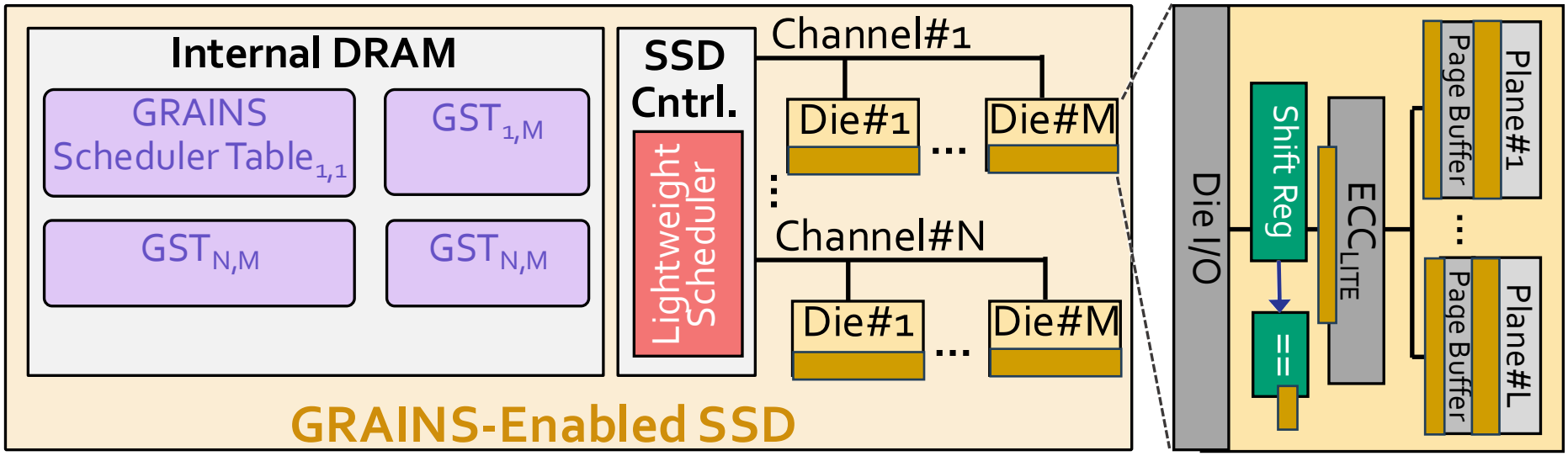
One row per Strings page, each row storing
1) *bit address in the page*

✓ ***Efficient address mapping*** *frees most of the DRAM*

✓ ***Compact k-mer representation*** *keeps GST entries small*

Graph Query: Strings

- We aim for Strings accesses to fully exploit internal bandwidth as well
  - However, unlike Offsets, the indices into Strings arrive unsorted

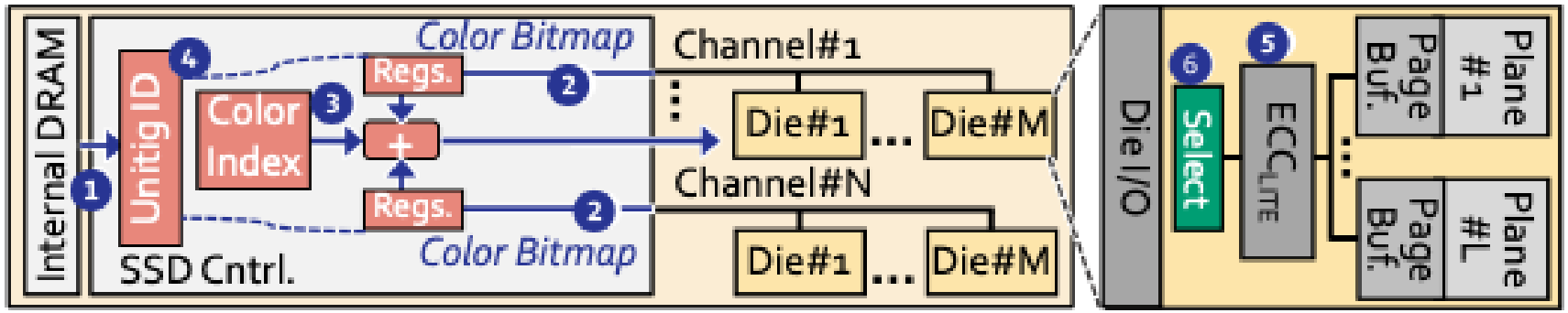


✓ *Avoids redundant page accesses*

✓ *Maximizes die-level parallelism,*
without relying on sorting accelerators

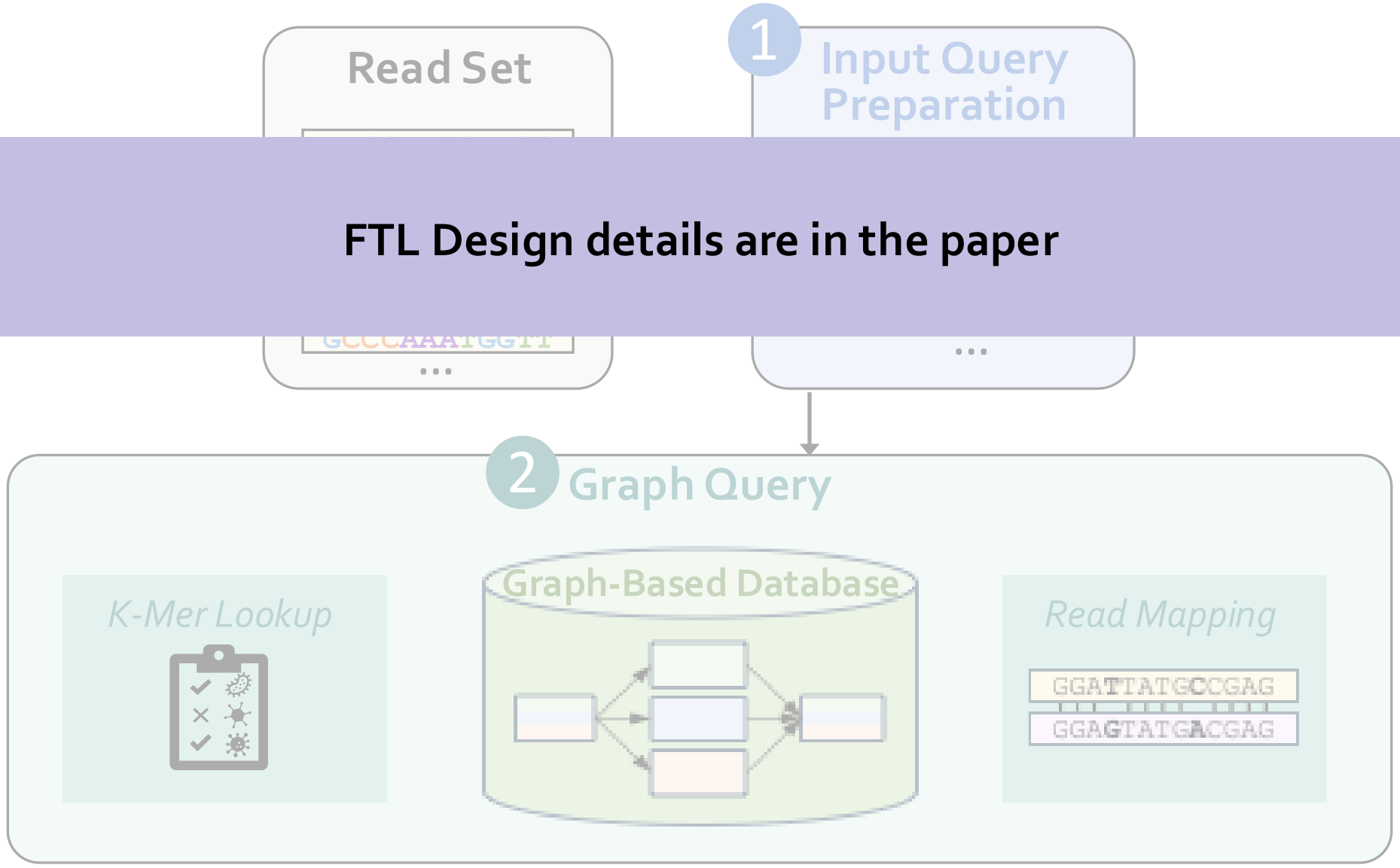
Graph Query: Colors

- For each k-mer matching a **unitig in Strings**, GRAINS retrieves the **color** of that unitig
- GRAINS returns the colors to the host, with **colors identifying database entries (e.g., species or samples)** containing the matched sequence



Design details are in the paper

Graph Query


Read Set
1
Input Query Preparation
FTL Design details are in the paper
2
Graph Query
K-Mer Lookup
Graph-Based Database
Read Mapping

More in the Paper

GRAINS: Enabling High-Performance and Low-Cost Graph-Based Genome Analysis via Storage-Aware Algorithm-Architecture Co-Design

Nika Mansouri Ghiasi[1] Harun Mustafa[2,1] Talu Güloglu[1] Rakesh Nadig[1]
Konstantina Koliogeorgi[1] Susana Rebolledo Ruiz[1,3] Marc Rautmann[1] Furkan Eris[1]
Mohammad Sadrosadati[1] Jisung Park[4] Onur Mutlu[1]

[1]ETH Zürich [2]Johns Hopkins University [3]University of Cantabria [4]POSTECH

https://arxiv.org/abs/2606.26468

Outline

Background

Motivation and Gaol

GRAINS

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® host CPU with 128 physical cores
- 1.5 TB host DRAM

Baseline Comparison Points

- **Fulgor** *[WABI'23]*: A state-of-the-art node-centric tool
- **MetaGraph** *[Nature'25]*: A state-of-the-art edge-centric tool
- **AccIdealMem**: An idealized accelerator for graph queries with no main memory overheads, but still suffers from the I/O overhead of moving a large amount of low-reuse data

SSD Configurations

- **SSD-G4:** With PCIe Gen4 interface (7 GB/s sequential read bandwidth)
- **SSD-G5:** With PCIe Gen5 interface (14.8 GB/s sequential read bandwidth)

Evaluation Methodology Overview (II)

Graph-Based Genome Analysis Task

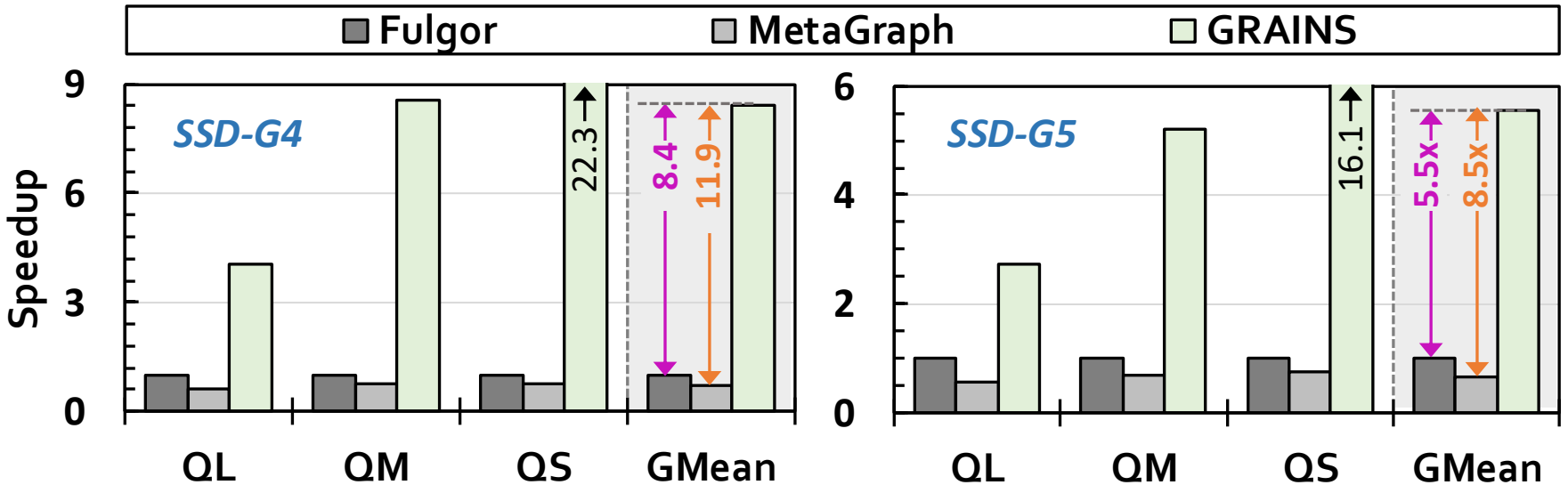
- K-mer set lookup
- Alignment-based read mapping
- Alignment-free read mapping

Query Samples

- With varying sizes
  - Small
  - Medium
  - Large

Benefits over Software

- When performing k-mer set lookups

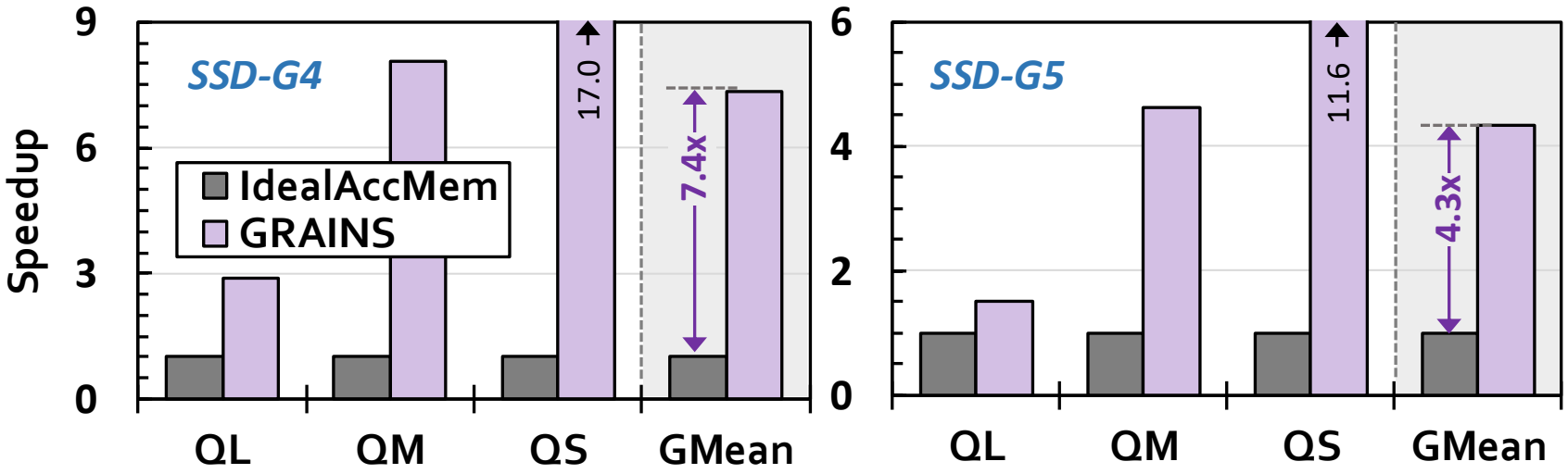


GRAINS provides significant speedup over Fulgor and Metagraph baselines, with both SSD-G4 and SSD-G5

GRAINS improves average energy-efficiency by 11.3x and 21.5x over Fulgor and Metagraph, respectively

Benefits over Hardware

- When performing k-mer set lookups

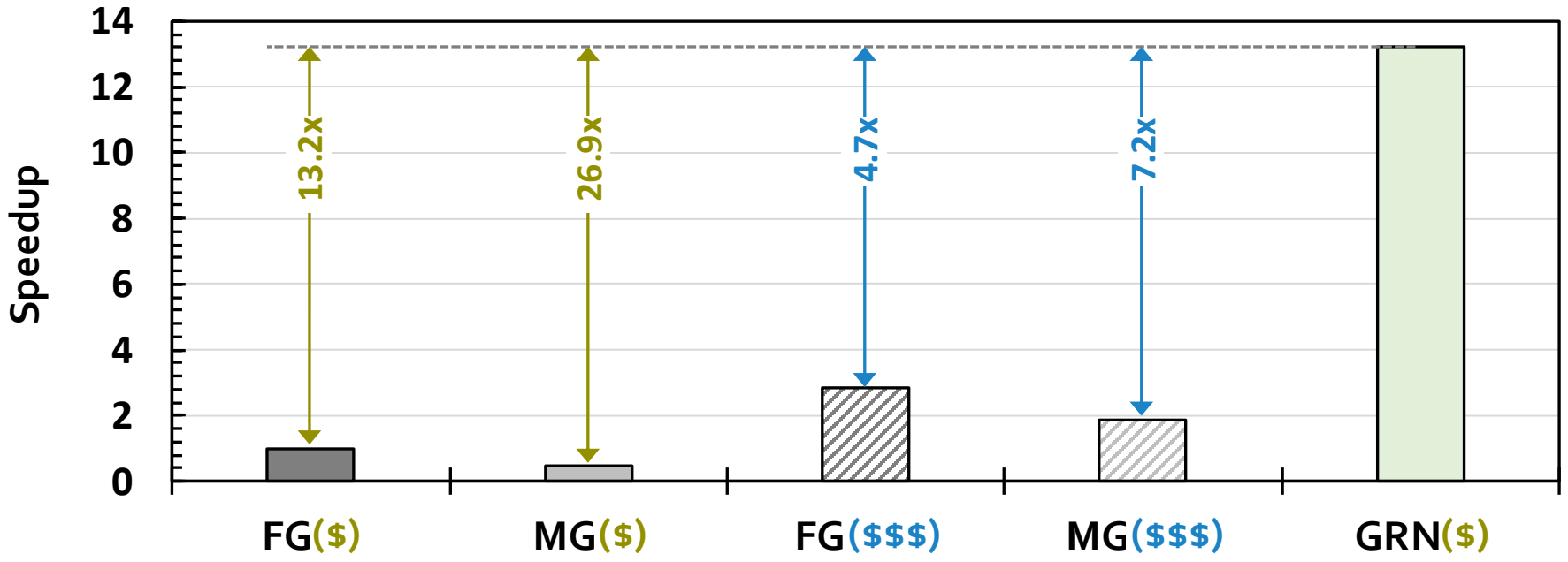


GRAINS provides significant speedups over AccIdealMem baseline

GRAINS improves average energy-efficiency by 3.1—20.7x over AccIdealMem

System Cost-Efficiency

- **Cost-optimized system ($):** With SSD-G4 and 64-GB Host DRAM
- **Performance-optimized system ($$$):** With SSD-G5 and 1-TB Host DRAM

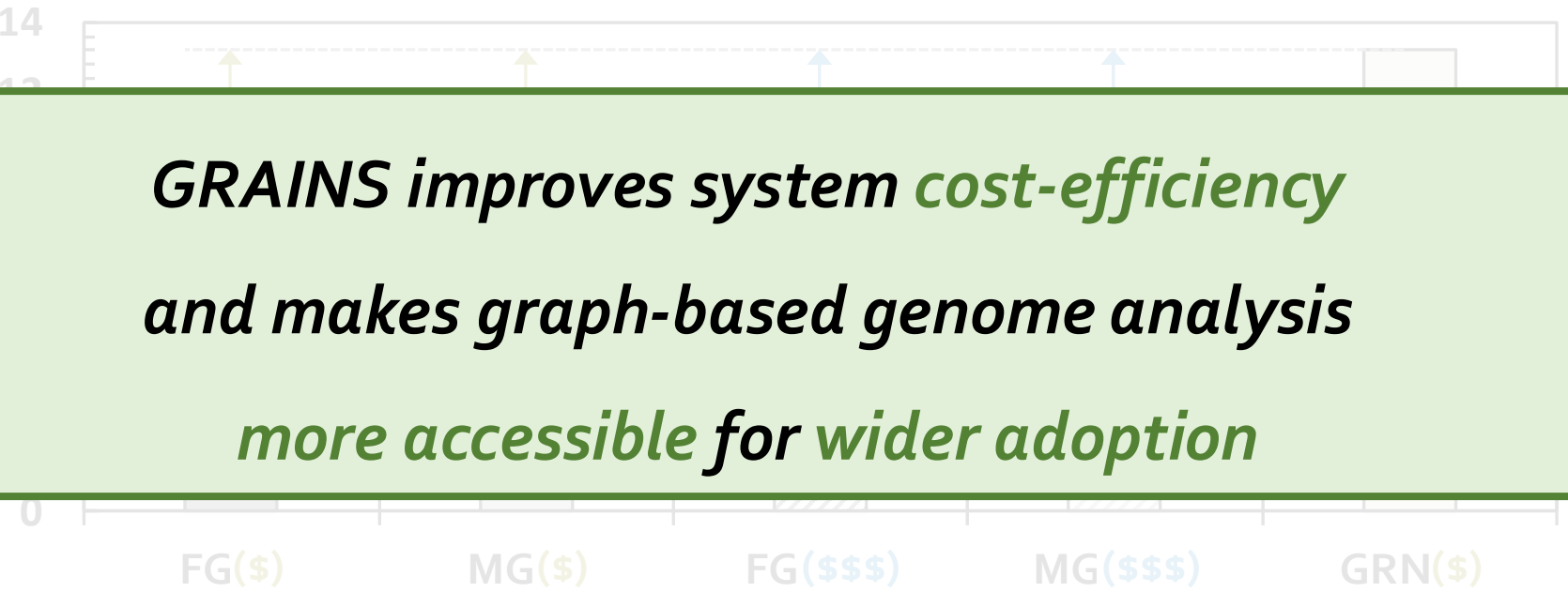


GRAINS outperforms the baselines

***even* when running on a much less costly system**

System Cost-Efficiency

- **Cost-optimized system ($):** With SSD-G4 and 64-GB Host DRAM
- **Performance-optimized system ($$$):** With SSD-G5 and 1-TB Host DRAM



GRAINS improves system cost-efficiency

and makes graph-based genome analysis

more accessible for wider adoption

GRAINS outperforms the baselines

even when running on a much less costly system

Area and Power

- Based on **Synthesis** of **GRAINS's hardware units** using the Synopsys Design Compiler @ 22nm technology node

- ECC_{LITE} area based on the original paper *[AiF, ISCA'25]*

Logic unit	Area [mm²]	Power [mW]
On SSD Controller	0.0025	0.21
On Die (ECC_LITE)	0.036	18.04
On Die (Others)	0.000093	0.01

On-controller logic units are 0.7% of the four cores on an SSD controller

On-die logic units take only 0.2% of the flash die's area and fit well within its power budget

More in the Paper

- GRAINS's speedups as an accelerator **outside the SSD**
- Impact of different GRAINS **optimizations in isolation**
  - Batching
  - Batching + Scheduling
  - Batching + Scheduling + SCC
- GRAINS's speedups with **varying #SSDs**
- **Execution time breakdown** of different steps
- GRAINS's integration with state-of-the-art **graph alignment accelerator**
- GRAINS's speedup when performing **Alignment-Free Read Mapping**

More in the Paper

GRAINS: Enabling High-Performance and Low-Cost Graph-Based Genome Analysis via Storage-Aware Algorithm-Architecture Co-Design

Nika Mansouri Ghiasi[1] Harun Mustafa[2,1] Talu Güloglu[1] Rakesh Nadig[1]
Konstantina Koliogeorgi[1] Susana Rebolledo Ruiz[1,3] Marc Rautmann[1] Furkan Eris[1]
Mohammad Sadrosadati[1] Jisung Park[4] Onur Mutlu[1]

[1]ETH Zürich [2]Johns Hopkins University [3]University of Cantabria [4]POSTECH

https://arxiv.org/abs/2606.26468

Outline

Background

Motivation and Goal

GRAINS

Evaluation

Conclusion

Conclusion

Graph-based genome analysis analysis suffers from
significant storage I/O data movement overhead

GRAINS

The *first system* for *large-scale genome graphs analysis* in storage
Enabled via *storage-aware algorithm-architecture co-design*,
exploiting the *properties of large-scale genome graphs*

Improves performance

2.7×–47.8× over **software** baselines

1.5×–17.0× over **hardware** software

Reduces energy consumption

4.4×–31.6× over **software** baseline

3.1×–20.7× over **hardware** baseline

Improves system cost-effectiveness

Makes graph-based genome analysis
more accessible for wider adoption

GRAINS:
Enabling High-Performance and Low-Cost Graph-Based Genome Analysis via Storage-Aware Algorithm-Architecture Co-Design

Nika Mansouri Ghiasi Harun Mustafa Talu Güloglu **Rakesh Nadig**

Konstantina Koliogeorgi Susana Rebolledo Ruiz Marc Rautmann Furkan Eris

Mohammad Sadrosadati Jisung Park Onur Mutlu

SAFARI

ETH zürich UC | Universidad de Cantabria POSTECH

Backup Slides

Unique Characteristics of Genome Graphs

- **Different Node and Edge Encodings**
  - Traditional graphs must explicitly store O(N) to O(N^2) edges
  - DBGs store only nodes or edges and the other is derived implicitly
  - Enabled by the inherent overlap structure of encoded sequences

- **Different Structural Characteristics**
  - In/out-degrees capped at 4 (A, C, G, T), regardless of graph size
  - Power-law hub optimizations from conventional graphs are inapplicable
  - Necessitates fundamentally different optimization approaches

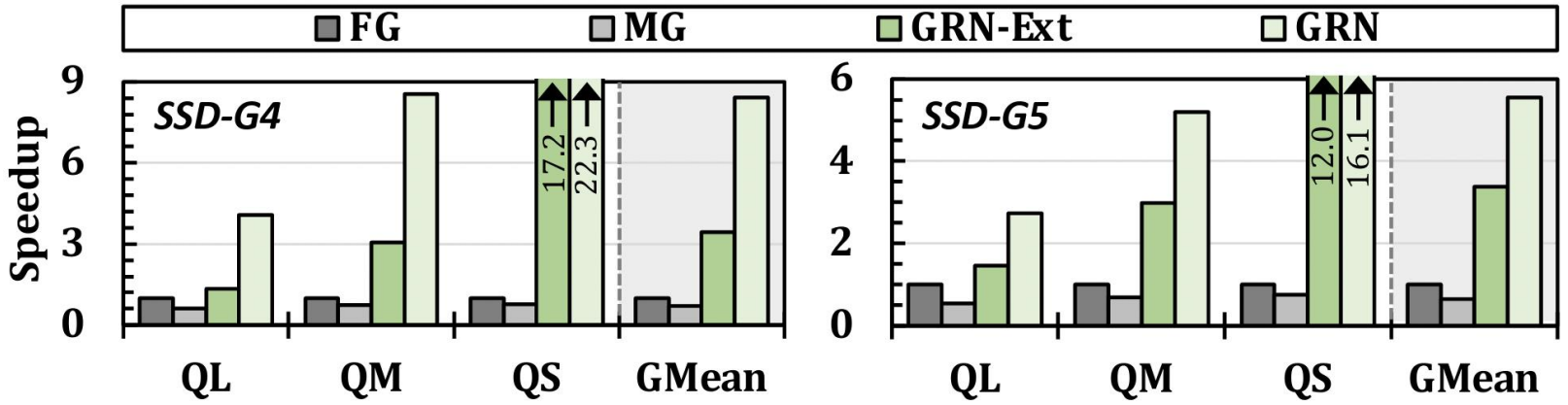
- **Different Access Pattern Optimizations**
  - Compressed data structures enforce node orderings ➔ no reordering possible
  - Structural coupling between queries and graph enables unique locality optimizations
  - Query reads sharing k-mer substrings map to nearby index regions

Benefits Outside the SSD (I)

- **Key benefits of storage-centric designs outside storage**
  - Alleviating the burden of moving and analyzing low-reuse data
  - More storage-friendly execution pipelines: Efficient access patterns, and fewer random accesses
    - Efficient use of the available I/O bandwidth
    - Less reliance on large DRAM capacity

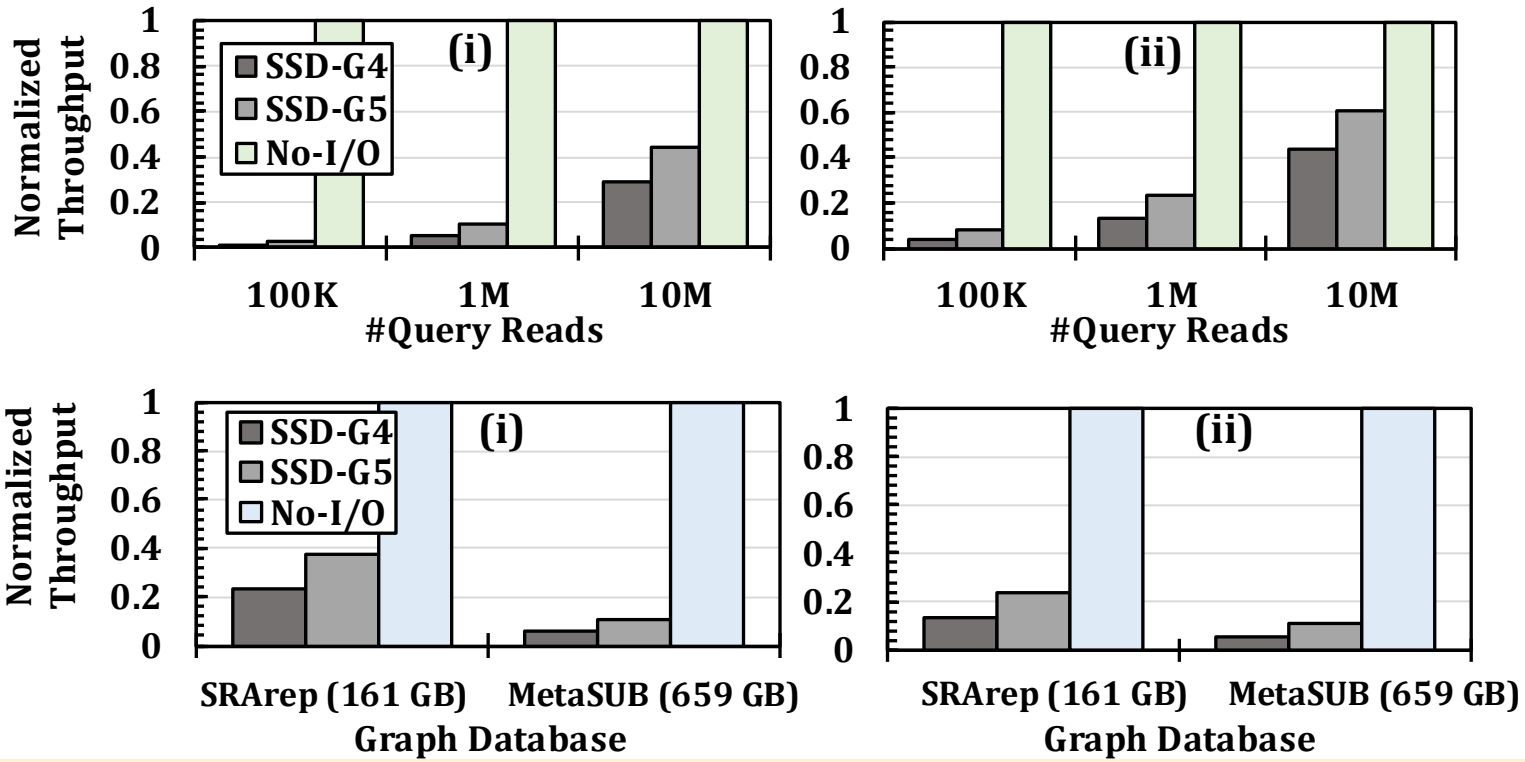
- **However, there are more benefits from analysis inside the storage**
  - Higher IFP/ISP bandwidth
  - Avoid moving low-reuse data
  - Does not rely on wider external interfaces

Benefits Outside the SSD (II)



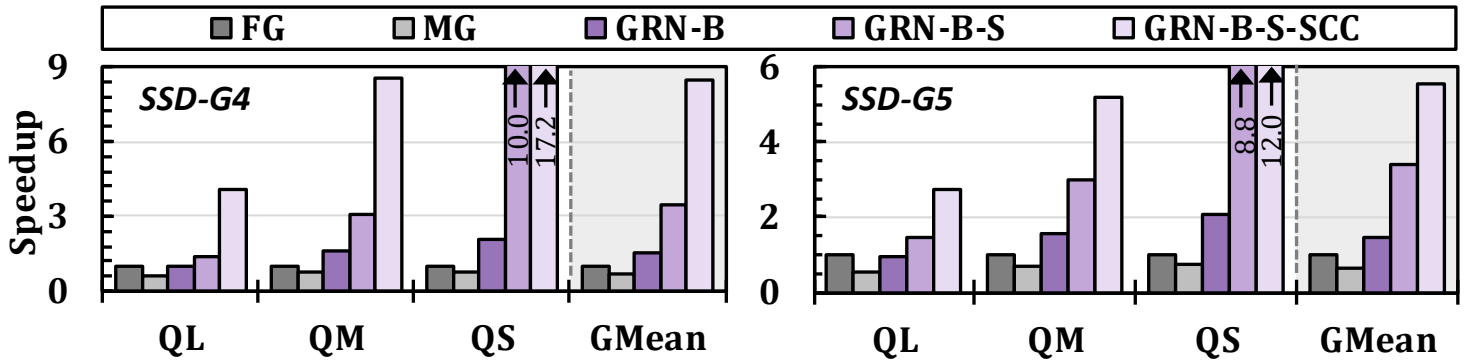
Motivation

- Case study of the performance of genome graph analysis tools
- With various state-of-the-art SSD configurations and data sizes

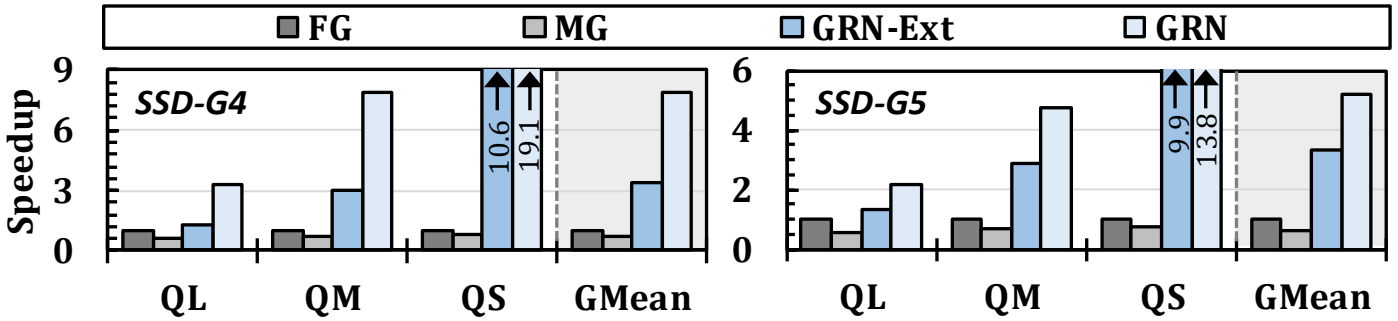


I/O data movement causes significant performance overhead

Effect of Different GRAINS Optimizations

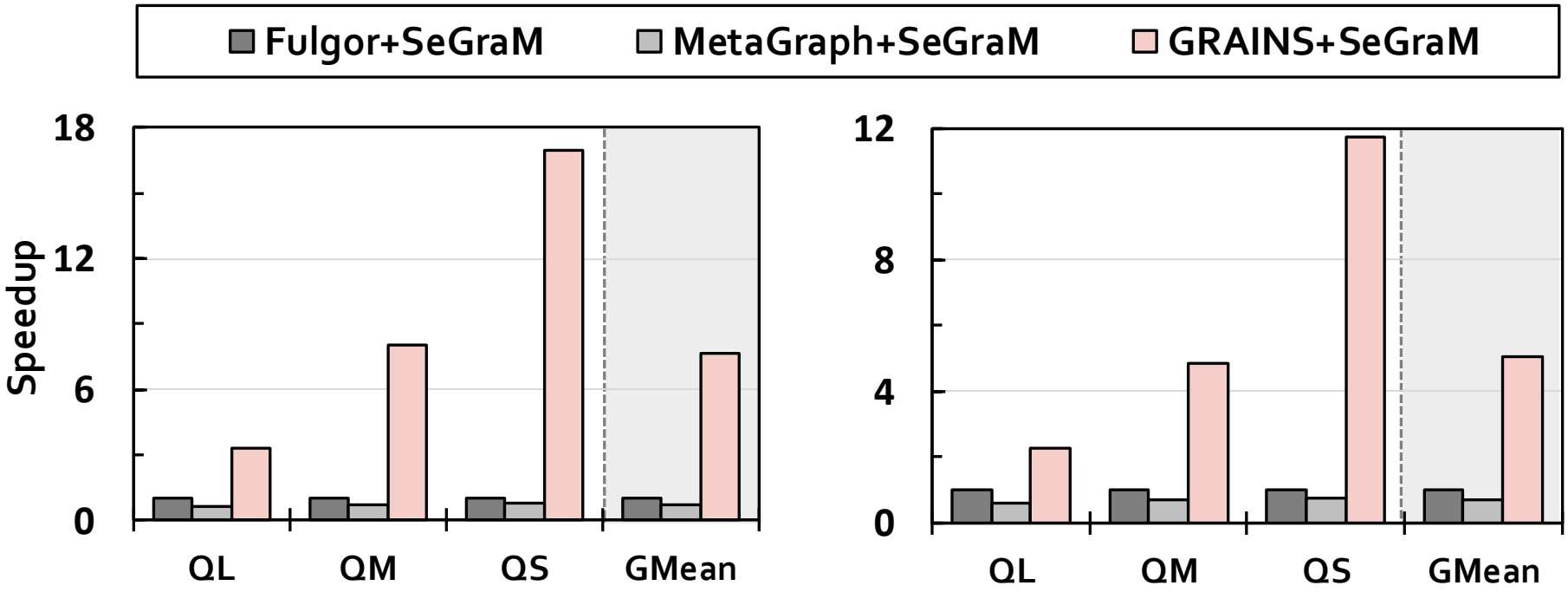


Speedup for Alignment-Free Read Mapping



Alignment-Based Read Mapping

- Integrating GRAINS with a state-of-the-art graph alignment accelerator, SeGraM *[ISCA'22]*



When integrated with SeGraM,
GRAINS provides significant speedup over
Fulgor and Metagraph baselines, with both SSD-G4 and SSD-G5

Speedup with Different Numbers of SSDs