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

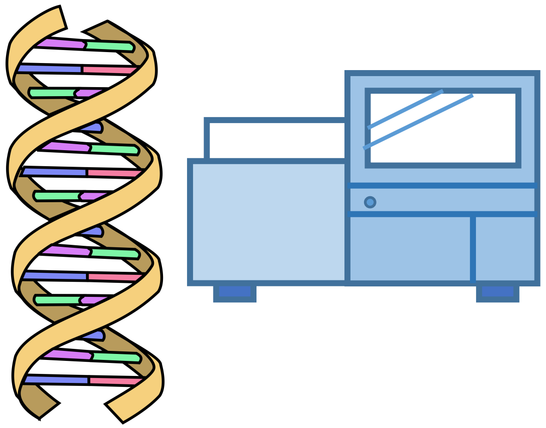
Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu

SAFARI

Genome Sequence Analysis

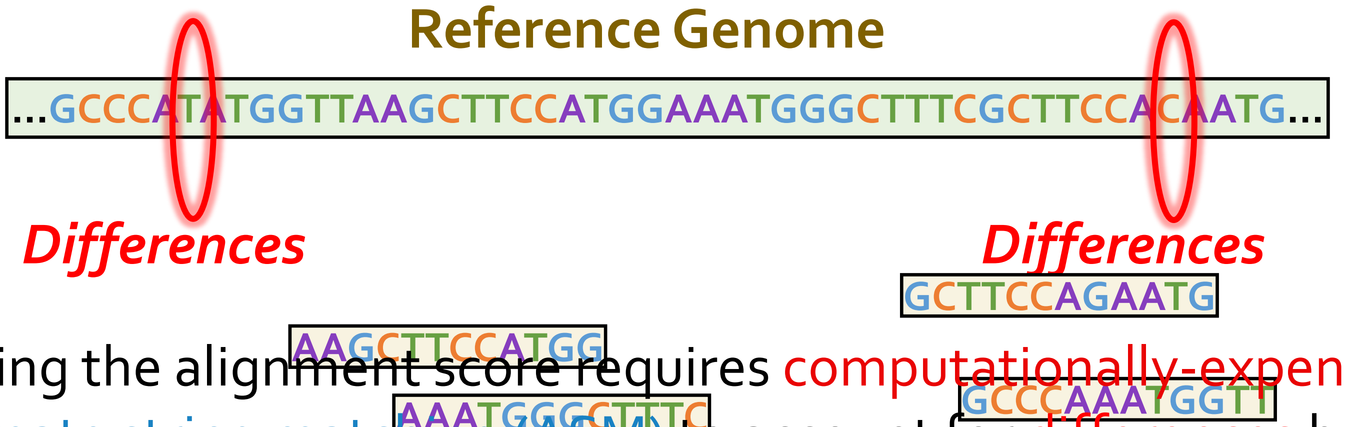
- Genome sequence analysis is critical for many applications
  - Personalized medicine
  - Outbreak tracing
  - Evolutionary studies

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



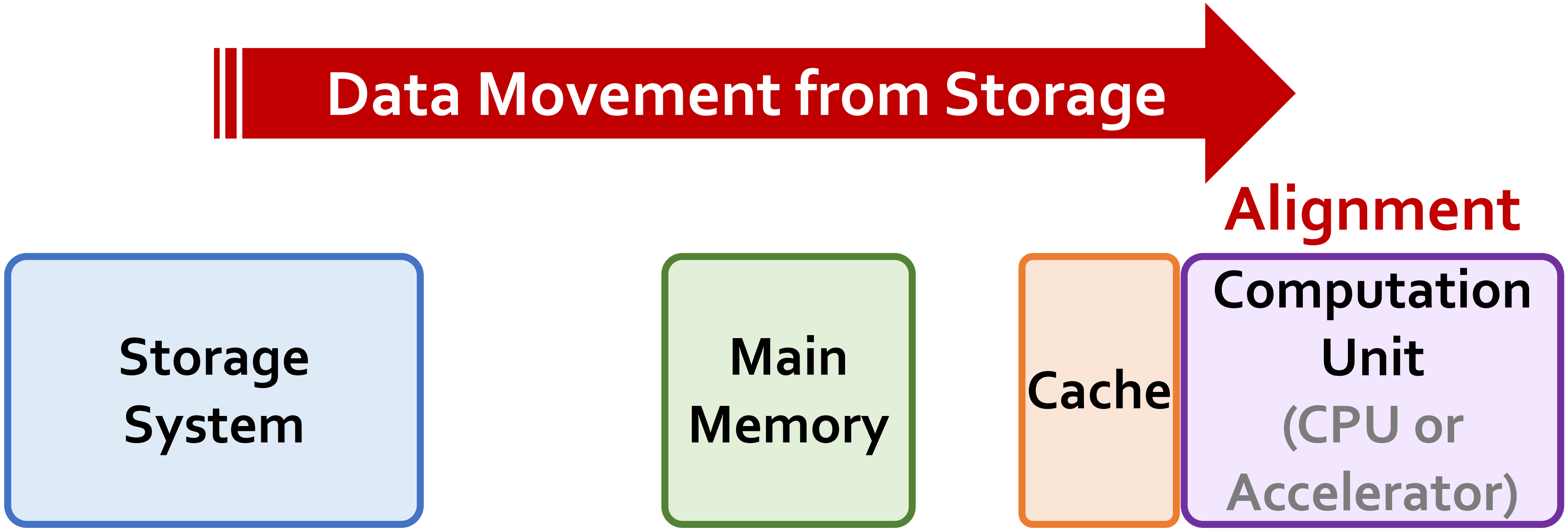
Genome Sequence Analysis

- **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

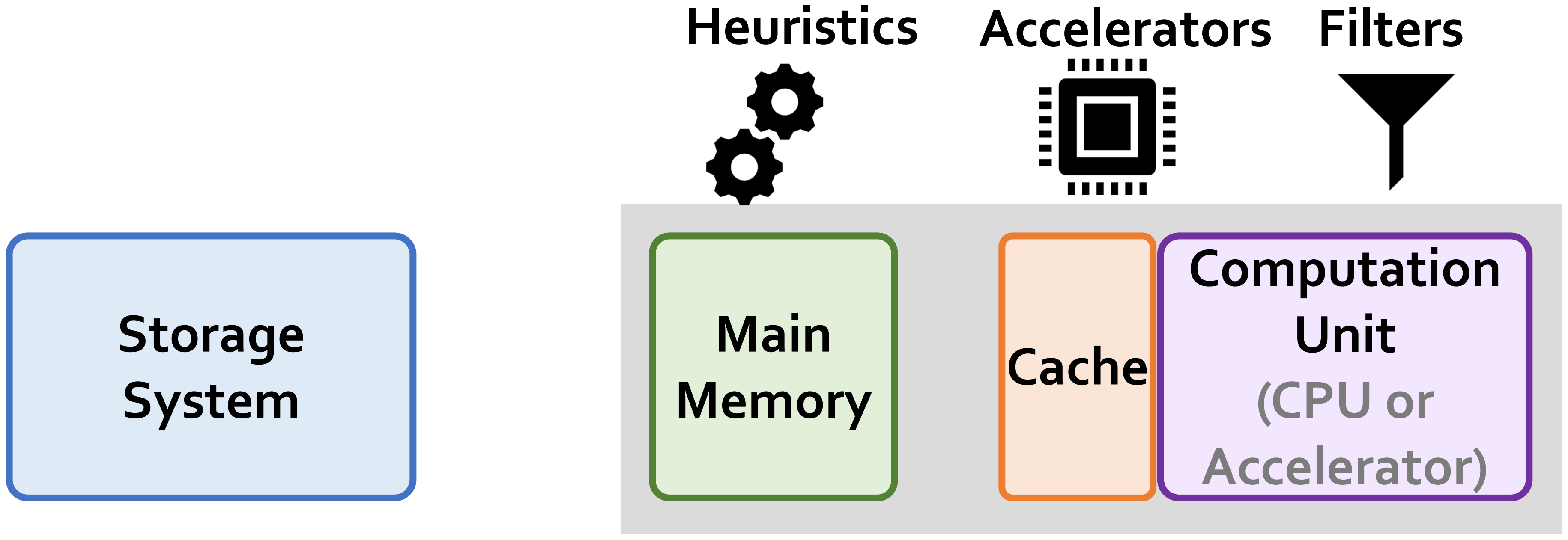
Genome Sequence Analysis



✗ Computation overhead

✗ Data movement overhead

Accelerating Genome Sequence Analysis

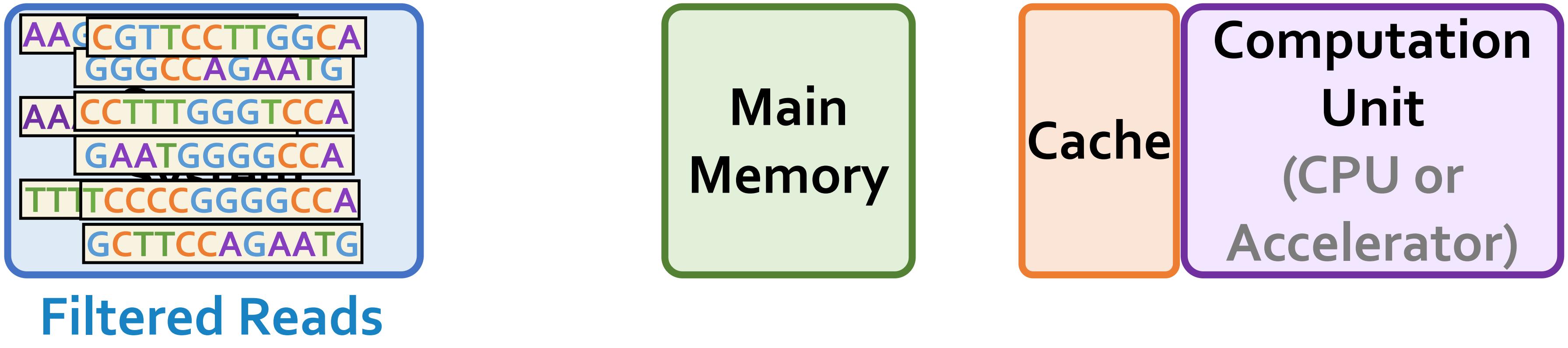


✓ Computation overhead

✗ Data movement overhead

Key Idea

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



Exactly-matching reads

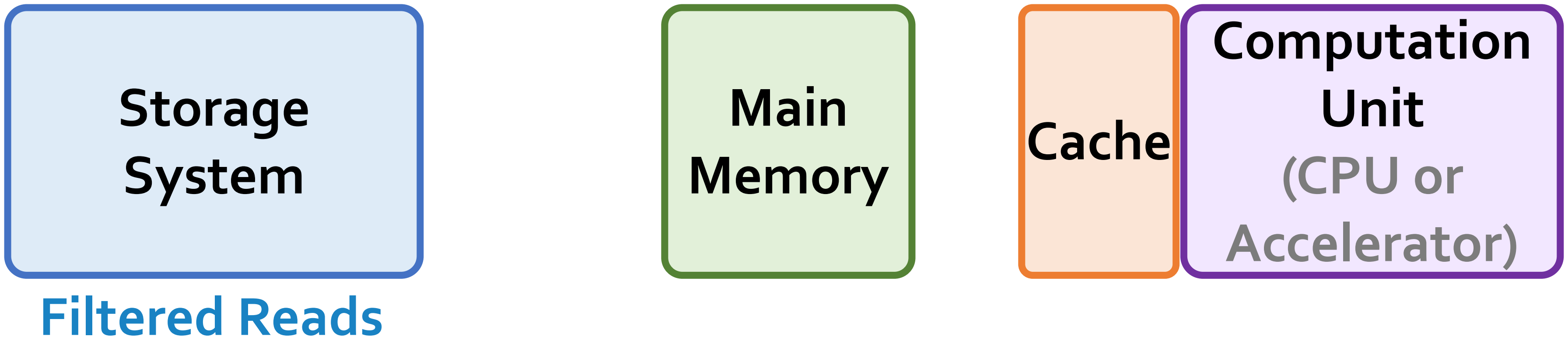
Do not need expensive approximate string matching during alignment

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*

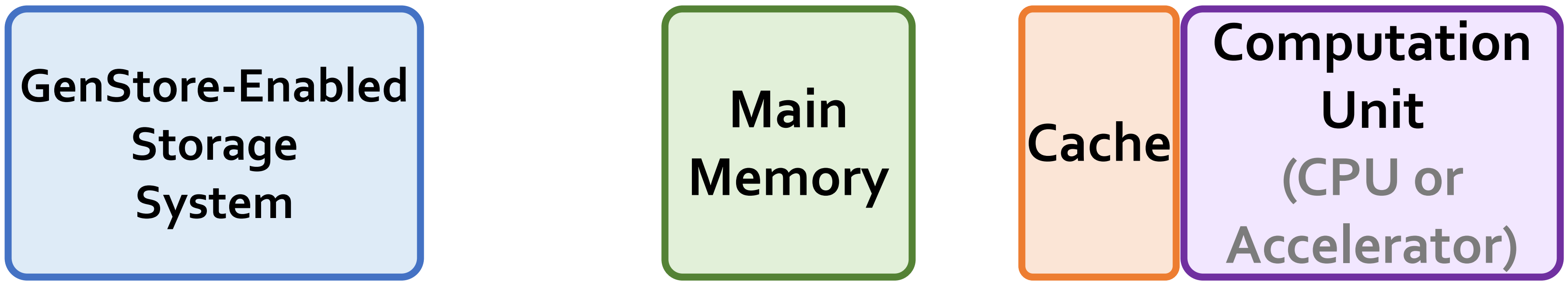


Read mapping workloads can exhibit different behavior

There are limited hardware resources in the storage system

GenStore

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



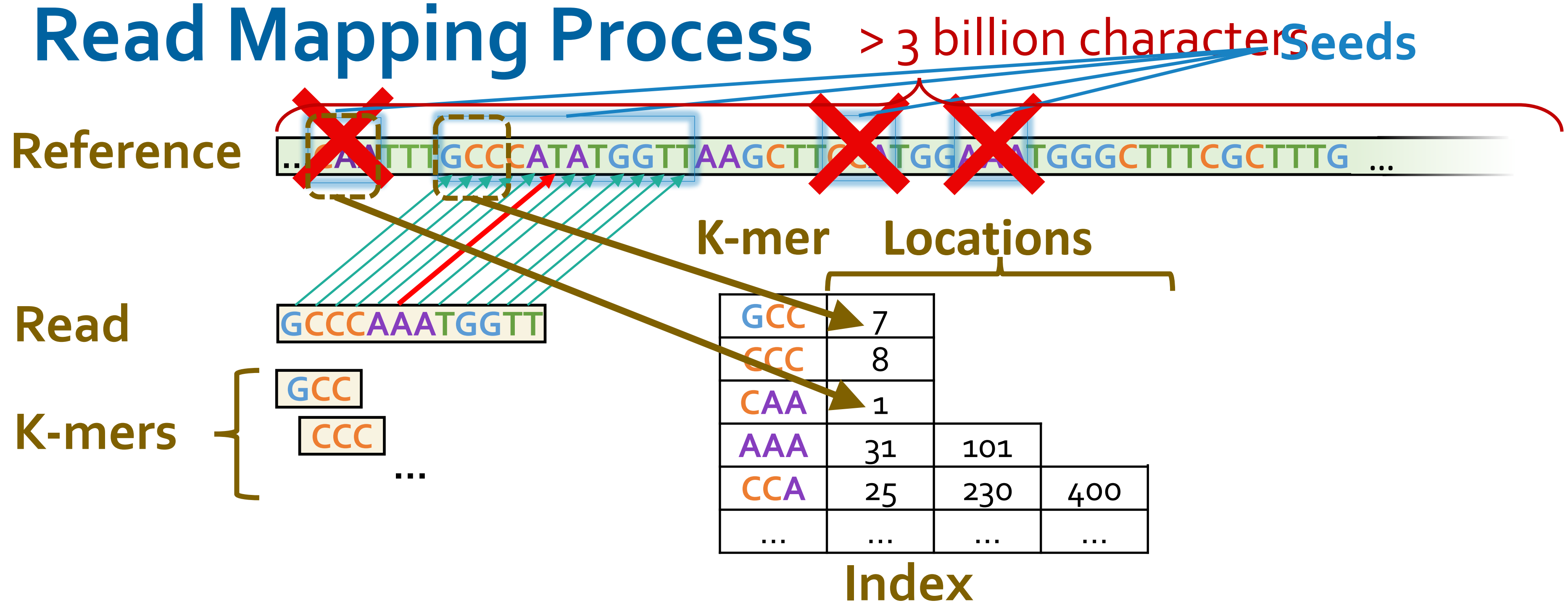
✓ Computation overhead

✓ Data movement overhead

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

Outline

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

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

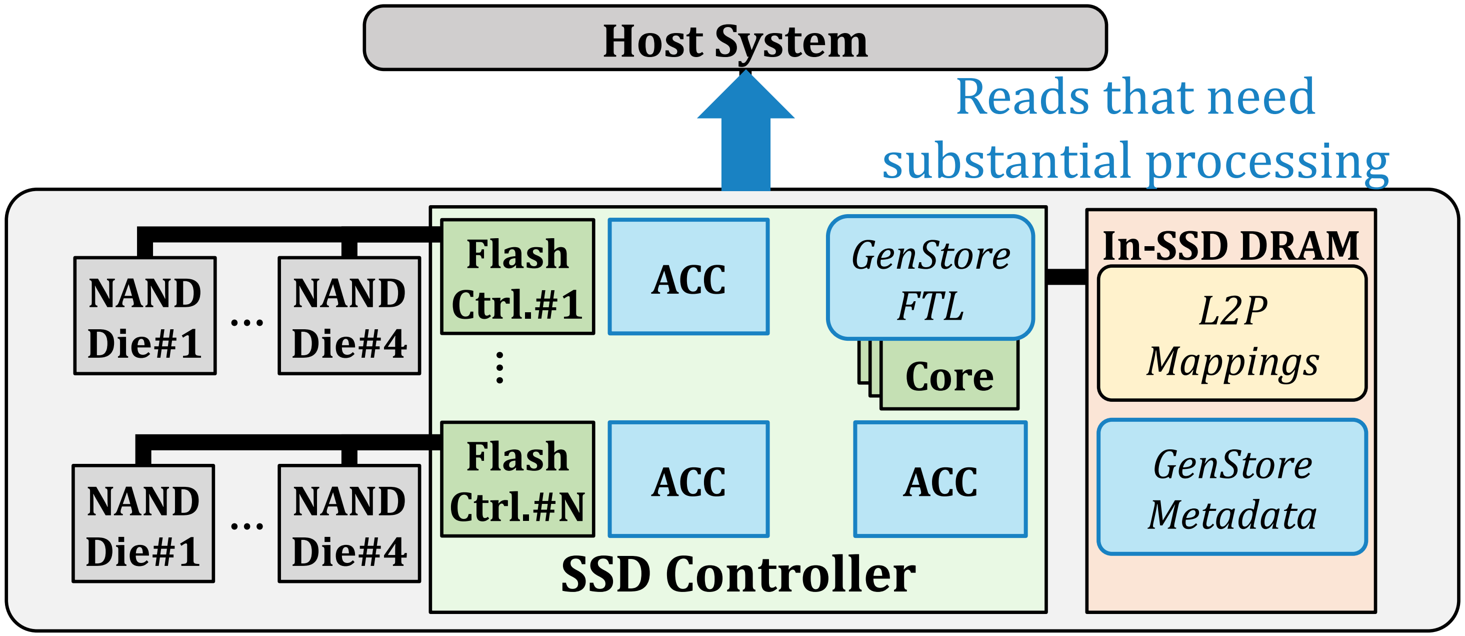
Do not require significant hardware overhead

Outline

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

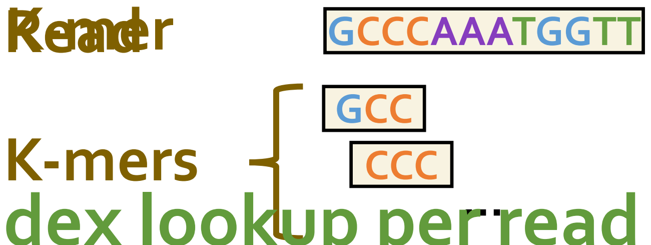
- Efficient in-storage filter for reads with at least one exact match in the reference genome
- Uses simple operations, without requiring alignment
- **Challenge:** large number of random accesses per read to the reference genome and its index

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

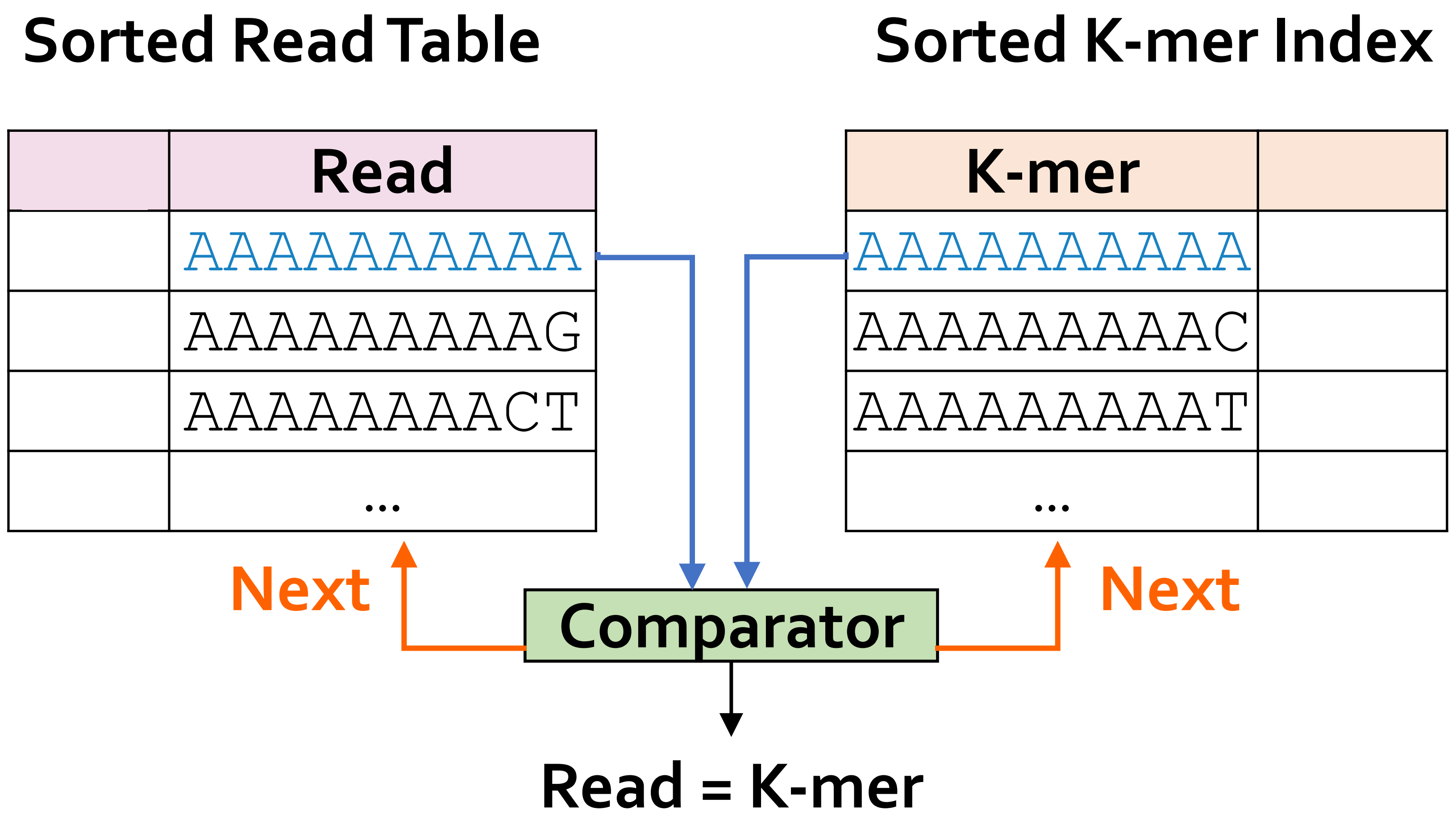
	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	…

Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
…	

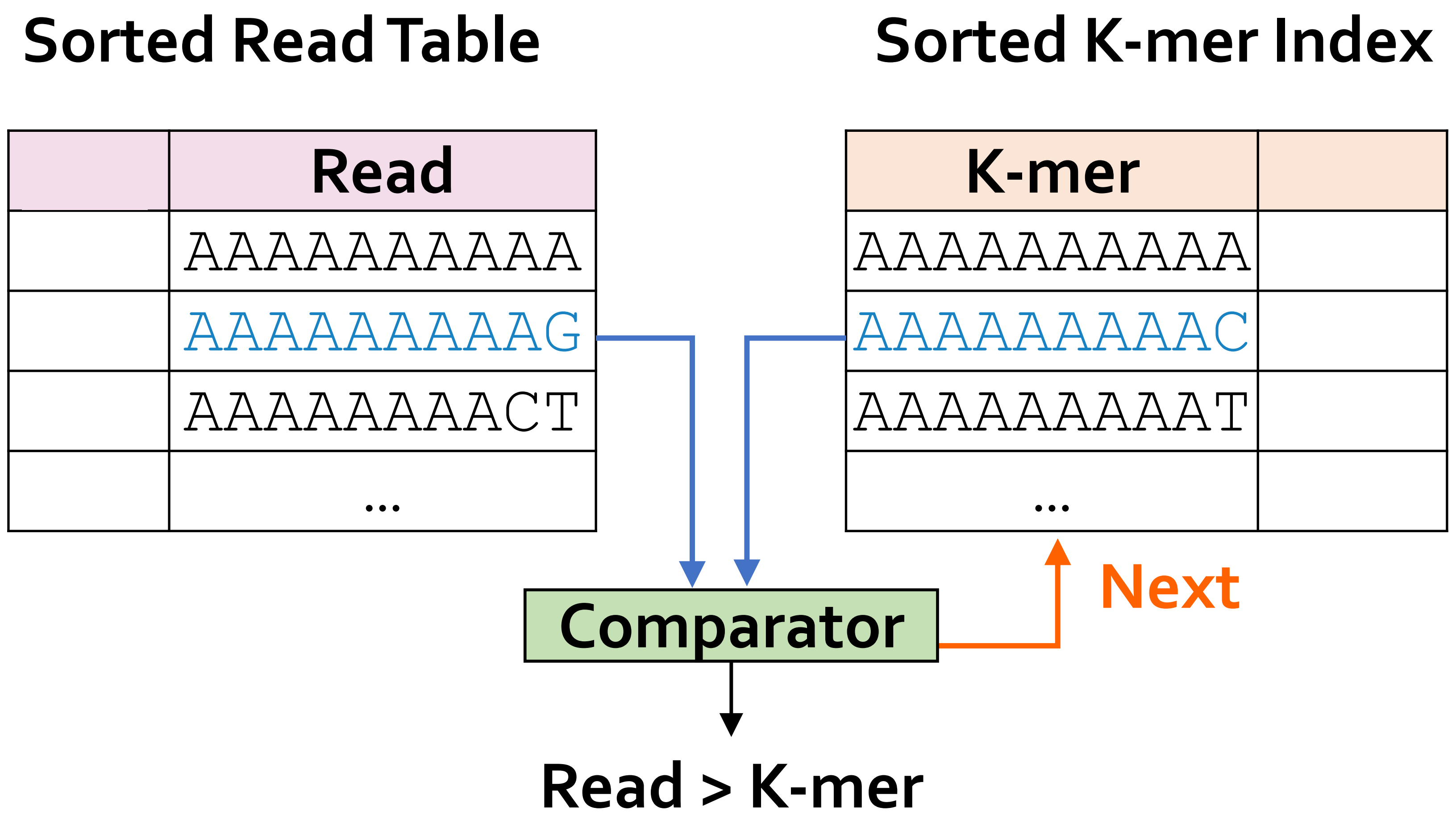
Read-sized K-mers

GenStore-EM: Finding a Match



Exact match → Filter the read

GenStore-EM: Not Finding a Match



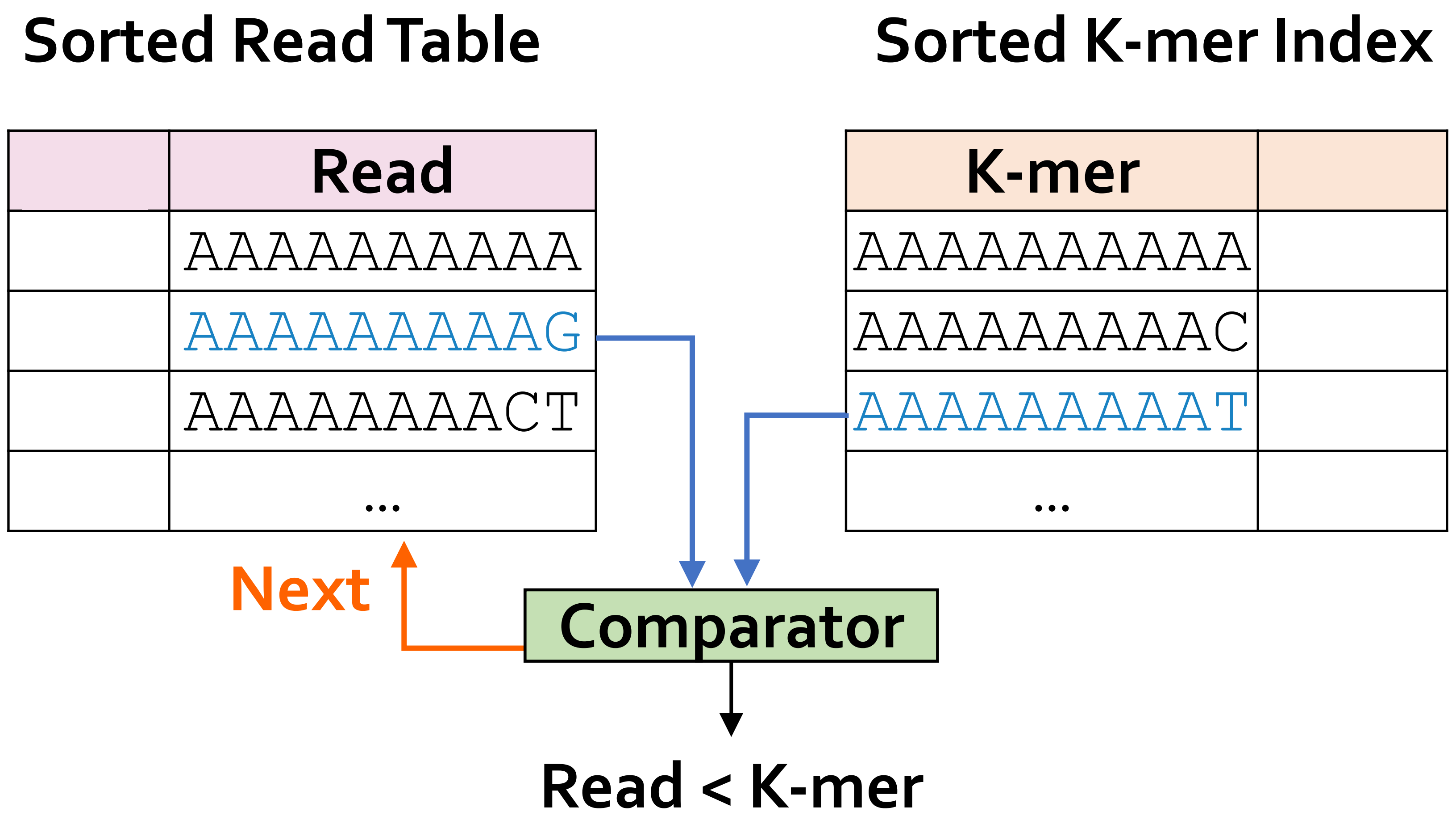
Sorted Read Table

	Read
	AAAAAAAAAA
	AAAAAAAAAG
	AAAAAAAACT
	...

Sorted K-mer Index

K-mer	
AAAAAAAAAA	
AAAAAAAAAC	
AAAAAAAAAT	
...	

GenStore-EM: Not Finding a Match



Not an exact match → Send to read mapper

GenStore-EM: Not Finding a Match

Sorted Read Table
Sorted K-mer Index
Read
AAAAAAAAAA
K-mer
AAAAAAAAAAA
Comparator
Read < K-mer

✓ Avoids random accesses

✓ Simple low-cost logic

Not an exact match → Send to read mapper

GenStore-EM: Optimization

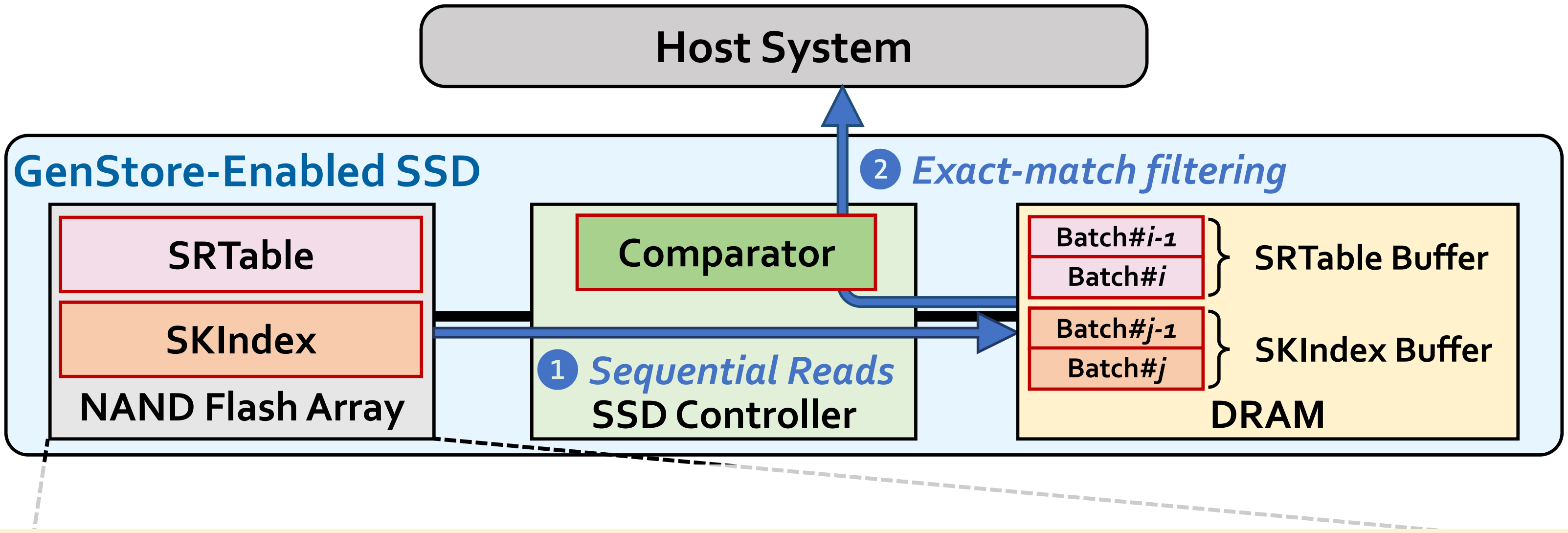
- Read-sized k-mer index takes up a large amount of space (126 GB for human index) due to the larger number of unique k-mers

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

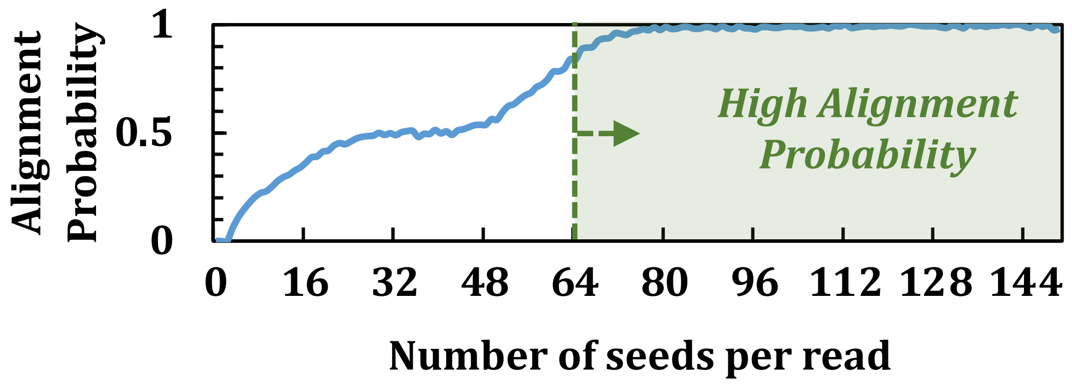
- **Challenge:** how to perform chaining inside the SSD

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



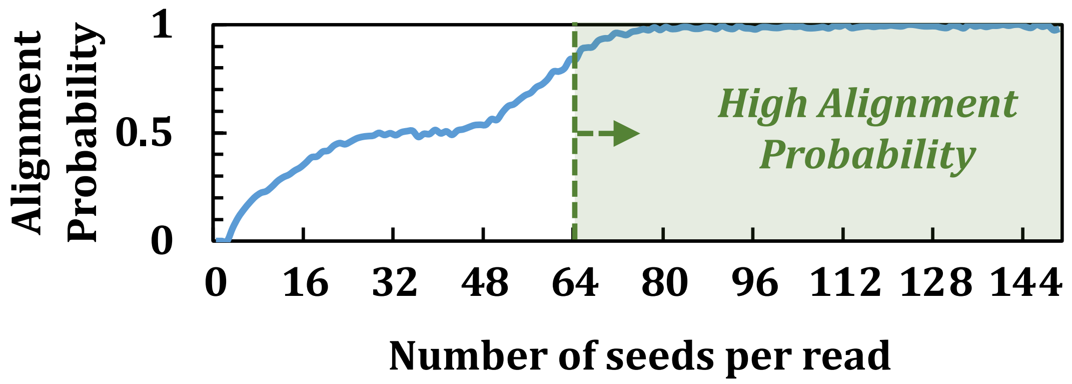
Reads with a sufficiently large number of seeds are very likely to align to the reference genome

✓ **Filters many non-aligning reads without costly hardware resources in the SSD**

SAFARI

GenStore-NM: Mechanism

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



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

SAFARI

Evaluation Methodology

Read Mappers

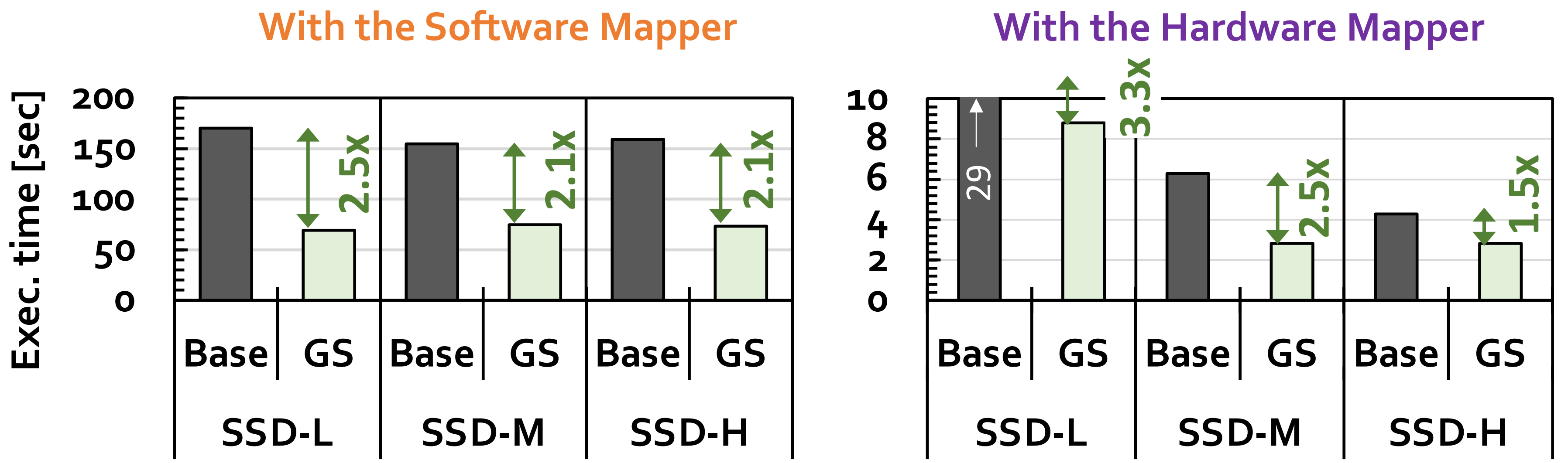
- **Base:** state-of-the-art software or hardware read mappers
  - Minimap2 [Bioinformatics’18]: software mapper for short and long reads
  - GenCache [MICRO’19]: hardware mapper for short reads
  - Darwin [ASPLOS’18]: hardware mapper for long reads
- **GS:** Base integrated with GenStore

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



2.1× - 2.5× speedup compared to the software Base

1.5× – 3.3× speedup compared to the hardware Base

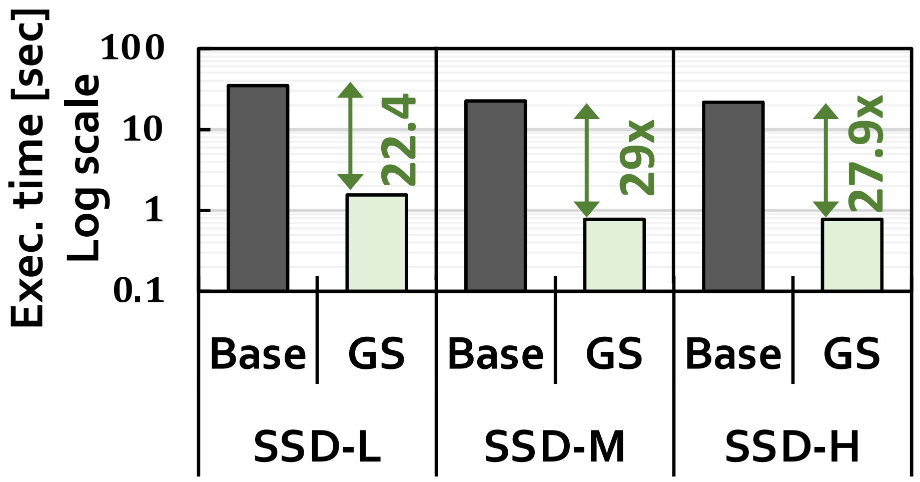
On average 3.92× energy reduction

SAFARI

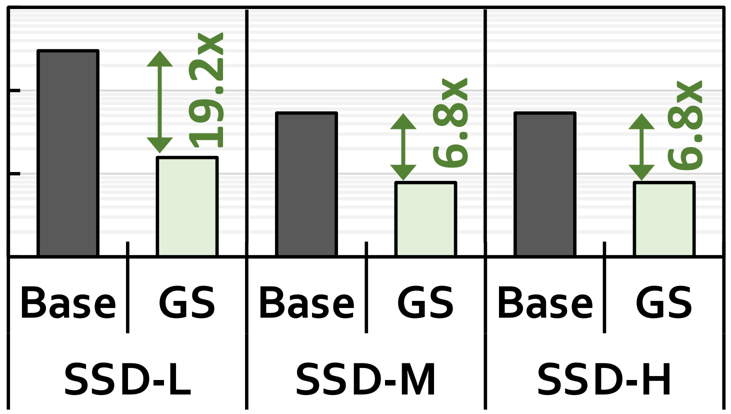
Performance – GenStore-NM

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

SAFARI

Other Results in the Paper

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

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

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

Outline

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