CMSC 829C: Algorithms, AI, and Hardware for Biological Data

Lecture 12: Example Paper Presentation – GRIM-Filter

Fall 2026

October 8, 2026

Can Firtina | canfirtina@gmail.com | cs.umd.edu/~firtina

Assistant Professor of Computer Science at UMD



UNIVERSITY OF MARYLAND

GRIM-Filter:

Fast seed location filtering in DNA read mapping using processing-in-memory technologies

BMC Genomics, 2018

Jeremie S. Kim,
Damla Senol Cali, Hongyi Xin, Donghyuk Lee,
Saugata Ghose, Mohammed Alser, Hasan Hassan,
Oguz Ergin, Can Alkan, and Onur Mutlu

Executive Summary

- **Genome Read Mapping** is a very important problem and is the first step in genome analysis

- Read Mapping is an **approximate string matching** problem
  - Find the best fit of 100 character strings into a 3 billion character dictionary
  - **Alignment** is currently the best method for determining the similarity between two strings, but is **very expensive**

- The paper proposes an algorithm called **GRIM-Filter**
  - Accelerates read mapping by reducing the number of required alignments
  - GRIM-Filter can be accelerated using **processing-in-memory**
    - Adds simple logic into **3D-Stacked memory**
    - Uses high internal memory bandwidth to perform parallel filtering

- GRIM-Filter with processing-in-memory delivers a **3.7x speedup**

GRIM-Filter Outline

1. Motivation and Goal
2. Background: Read Mappers
    a. Hash Table Based
    b. Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

GRIM-Filter Outline

1. **Motivation and Goal**
2. Background Read Mappers
    - **a.** Hash Table Based
    - **b.** Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Motivation and Goal

- **Sequencing**: determine the [A,C,G,T] series in DNA strand

- Today's machines sequence short strands of unknown origins (**reads**)
  - Reads are on the order of 100 – 20k base pairs (**bp**)
  - The human genome is approximately 3 billion bp

- **Read mapping** is the first step in analyzing someone's genome to detect predispositions to diseases, personalize medicine, etc.

- **Goal**: **accelerate** end-to-end performance of **read mapping**

GRIM-Filter Outline

1. Motivation and Goal
2. **Background: Read Mappers**
   - **a.** Hash Table Based
   - **b.** Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Background: Read Mappers

From **sequenced reads** to a **full genome**



Read mappers map **reads** to a known **reference genome** (>99.9% similarity across humans) with some minor errors allowed



Because of high similarity, long sequences in **reads**

mostly match in the **reference genome**



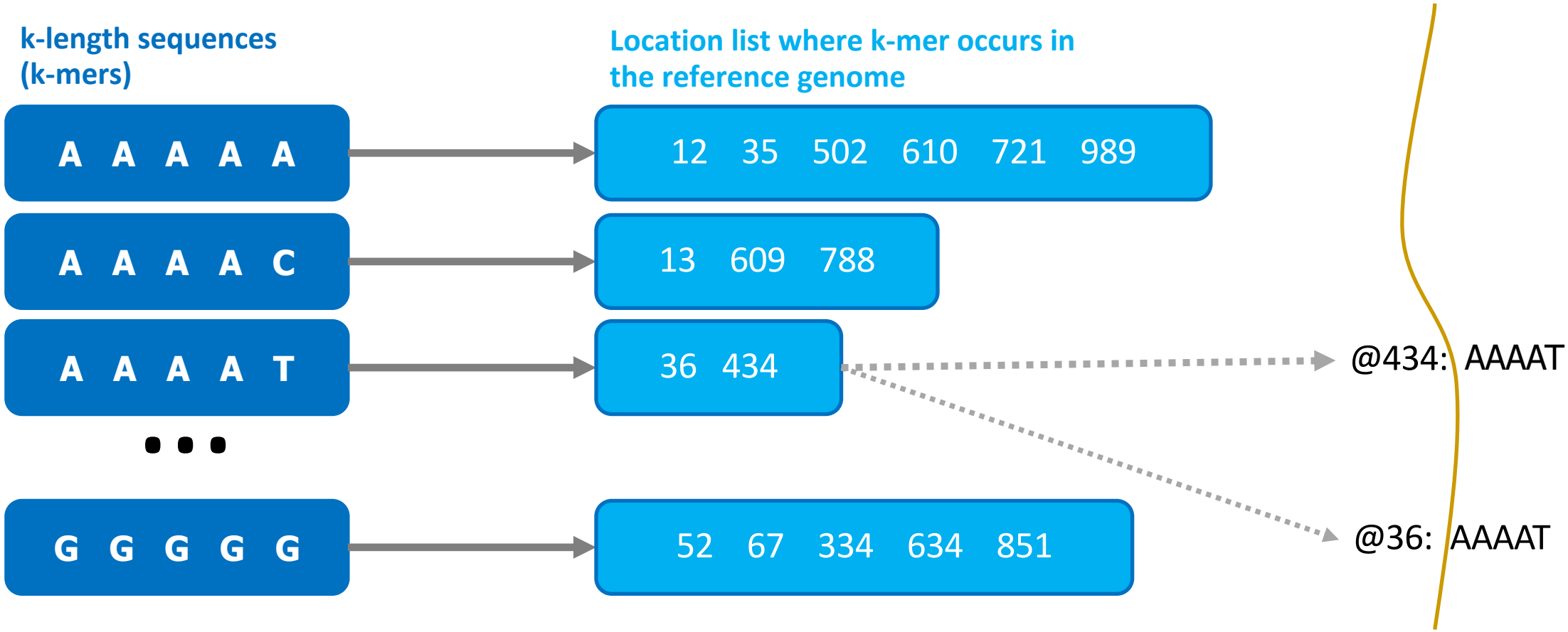
A hash table helps quickly map the reads!

GRIM-Filter Outline

1. Motivation and Goal
2. **Background: Read Mappers**
    - **a. Hash Table Based**
    - **b.** Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Generating Hash Tables

To map any reads, generate a **hash table** per **reference genome.**



Query the table with substrings from reads to quickly find a list of possible mapping locations

Hash Tables in Read Mapping

Read Sequence (100 bp)

99.9% of locations
result in a **mismatch**

Hash Table

Reference Genome

Goal: Filter these out
to avoid wasting time trying to align them

Location Filtering

- **Alignment** is expensive and requires the use of $O(n^2)$ dynamic programming algorithm
  - Millions to billions of reads must be aligned
  - Modern read mappers reduce the time spent on alignment for increased

1.
2.

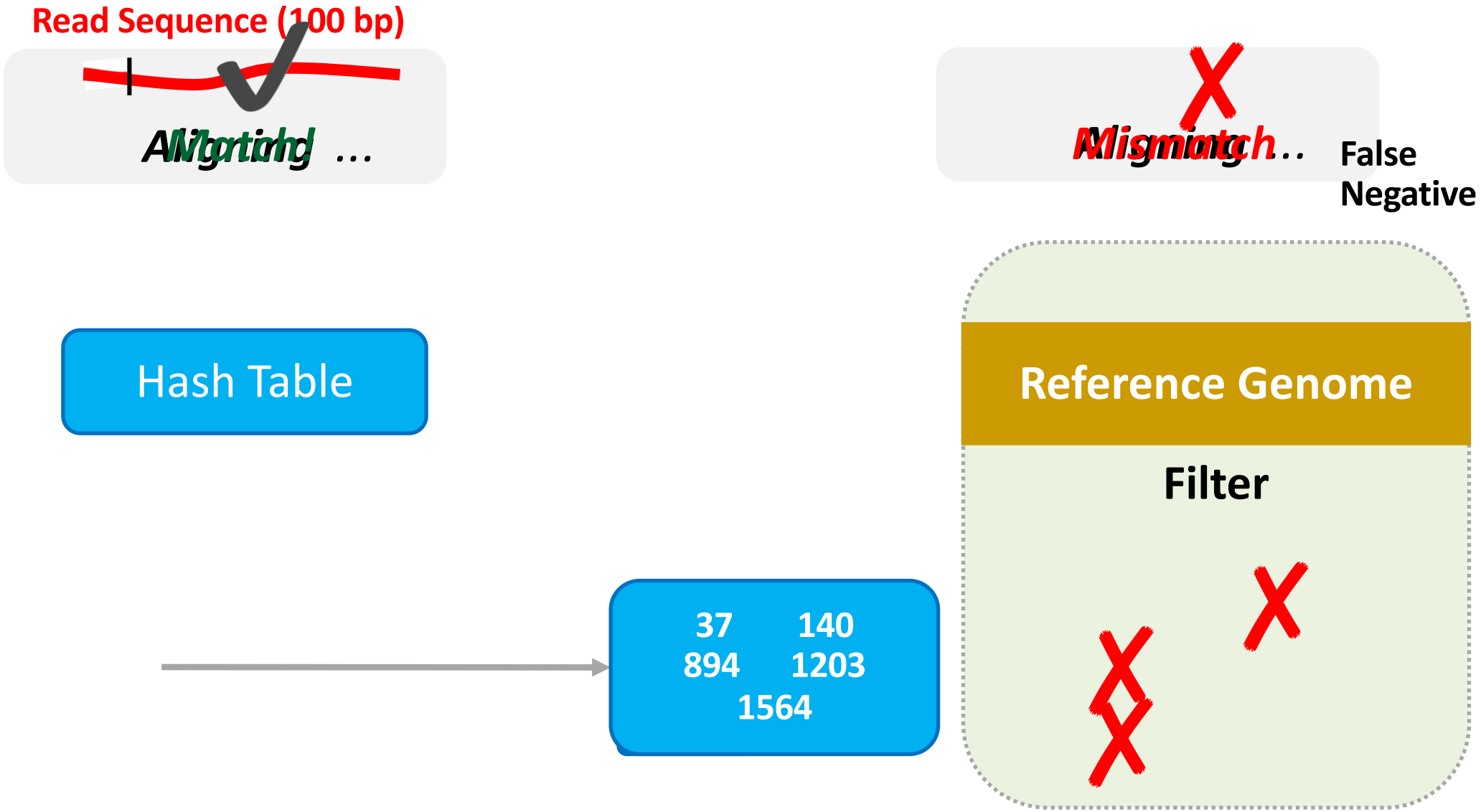
The goal is to accelerate read mapping by improving the filtering step

  - Both methods are used by mappers today, but filtering has replaced alignment as the bottleneck [Xin+, BMC Genomics 2013]

GRIM-Filter Outline

1. Motivation and Goal
2. **Background: Read Mappers**
    a. Hash Table Based
    b. Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Hash Tables in Read Mapping



GRIM-Filter Outline

1. Motivation and Goal
2. Background: Read Mappers
    a. Hash Table Based
    b. Hash Table Based with Filter
3. **Proposal: GRIM-Filter**
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Proposal: GRIM-Filter

1. **Data Structures: Bins & Bitvectors**
2. Checking a Bin
3. Integrating GRIM-Filter into a Mapper

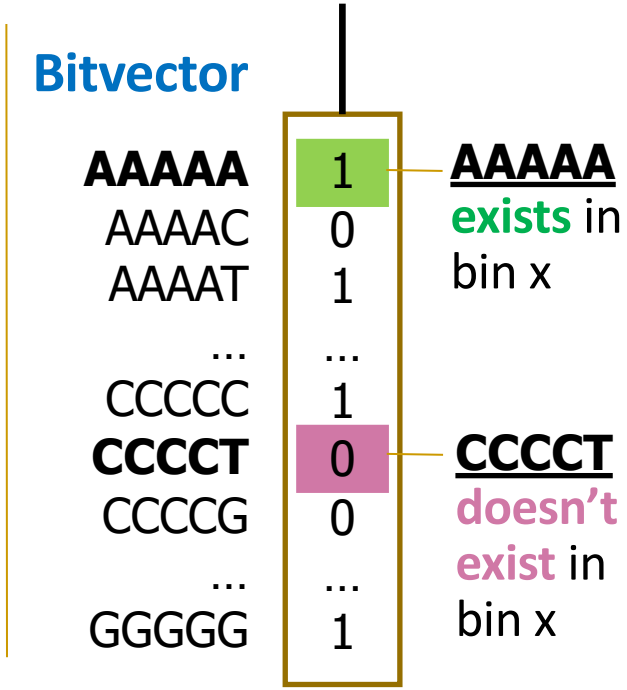
GRIM-Filter: Bins

- The genome is partitioned into large sequences (**bins**).

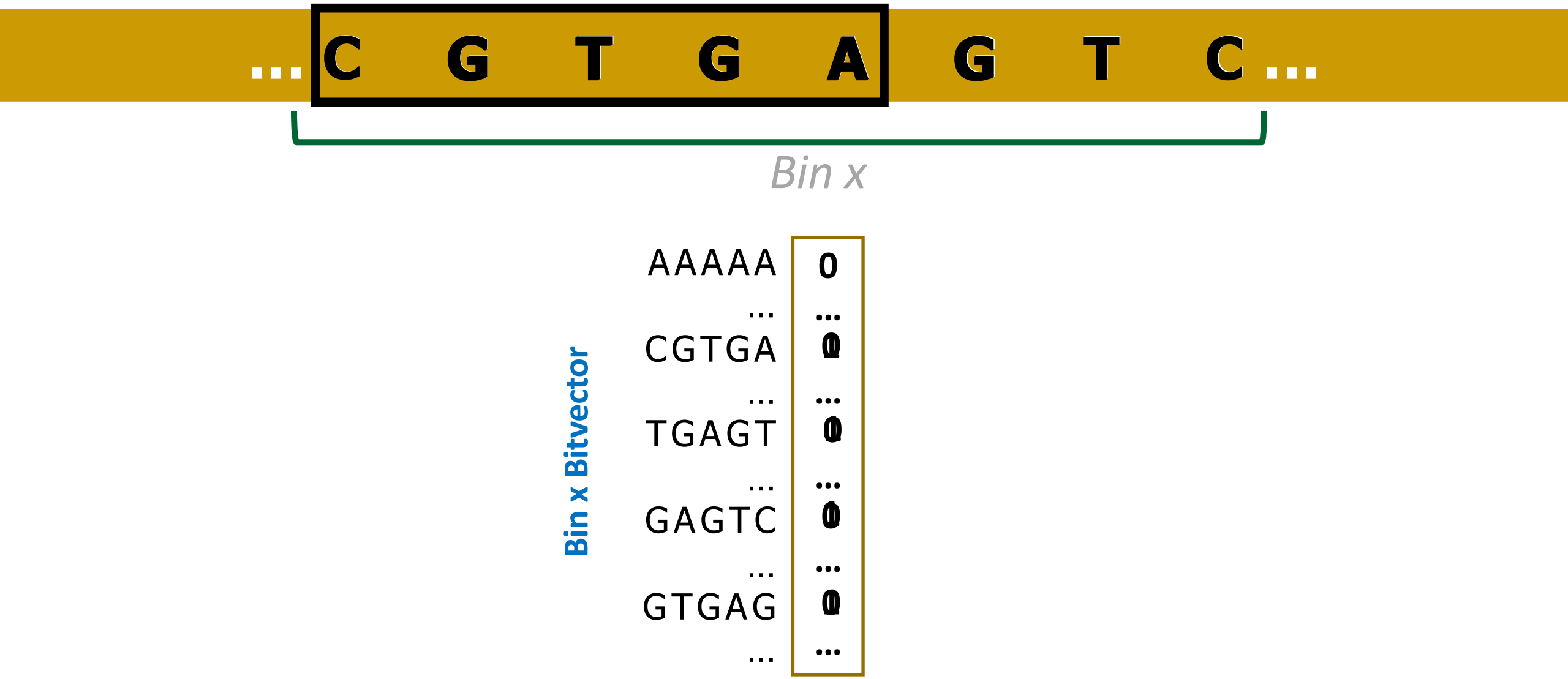


- Represent each bin with a **bitvector** that holds the occurrence of all permutations of a small string (**token**) in the bin

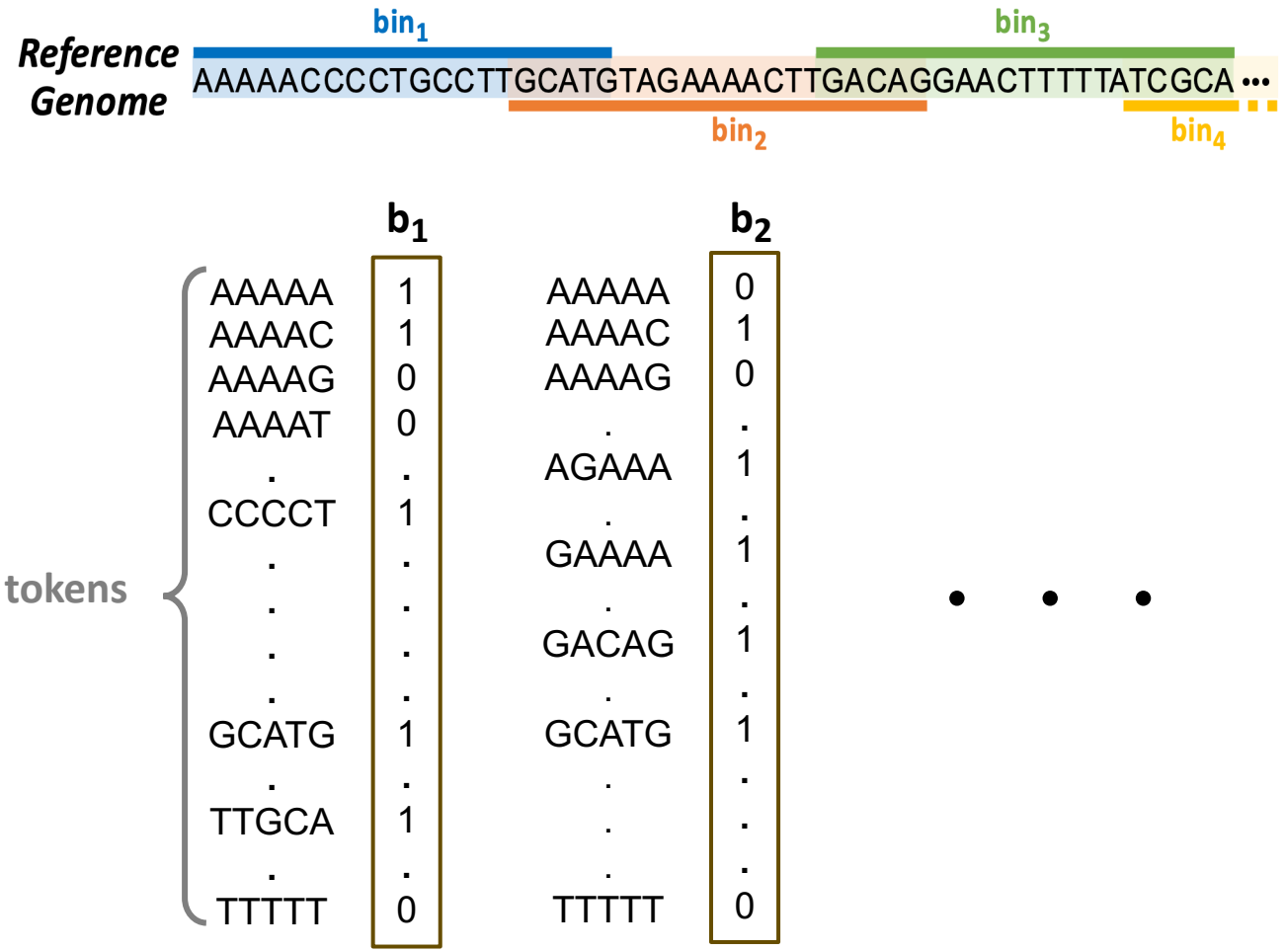
- To account for matches that straddle bins, the bins overlap
  - A read will now always completely fall within a single bin



GRIM-Filter: Bitvectors



GRIM-Filter: Bitvectors



Storing all bitvectors requires $4^n * t$ bits in memory, where t = number of bins.

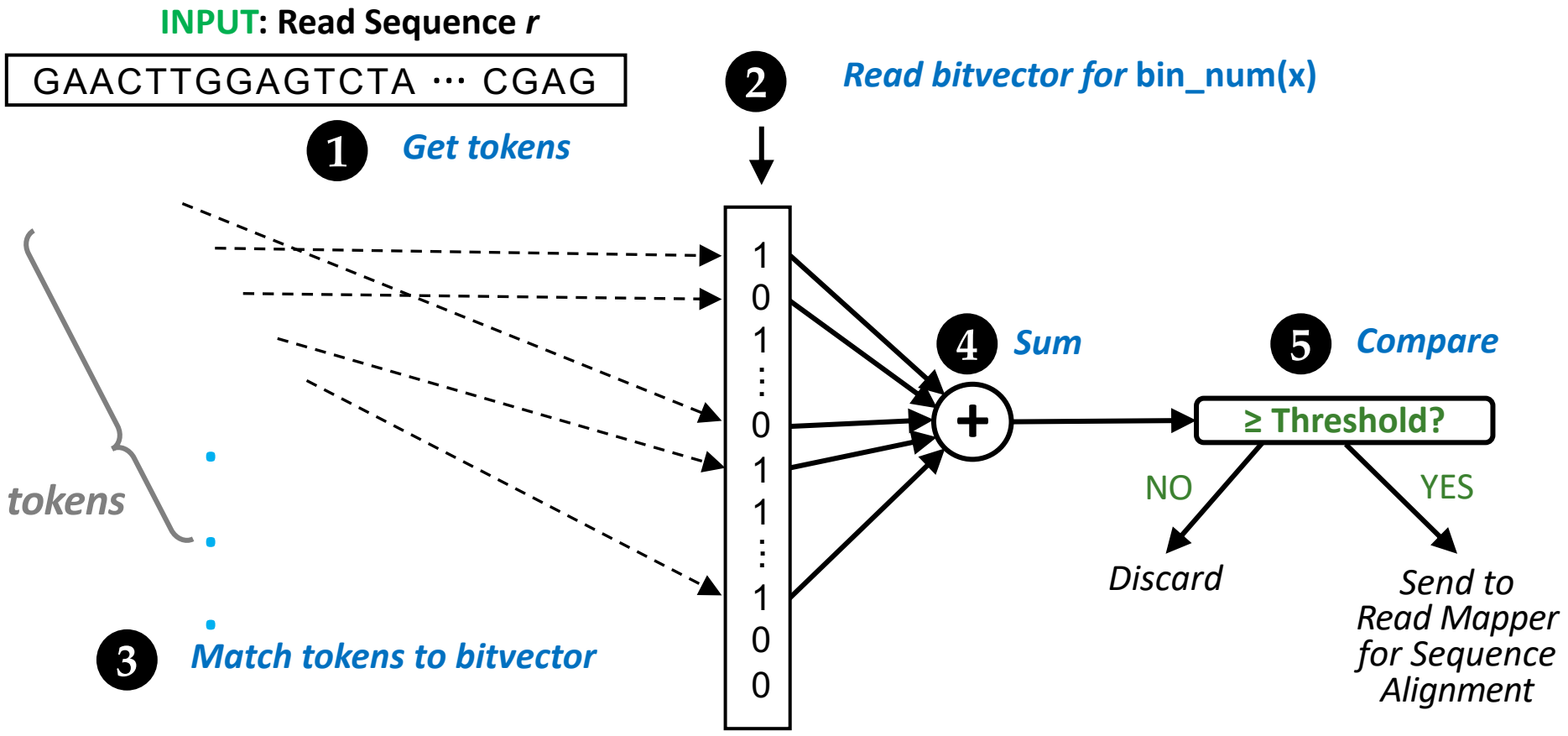
For **bin size** ~200, and **n** = 5, **memory footprint** ~3.8 GB

Proposal: GRIM-Filter

1. Data Structures: Bins & Bitvectors
2. **Checking a Bin**
3. Integrating GRIM-Filter into a Mapper

GRIM-Filter: Checking a Bin

How GRIM-Filter determines whether to **discard** potential match locations in a given bin **prior** to alignment



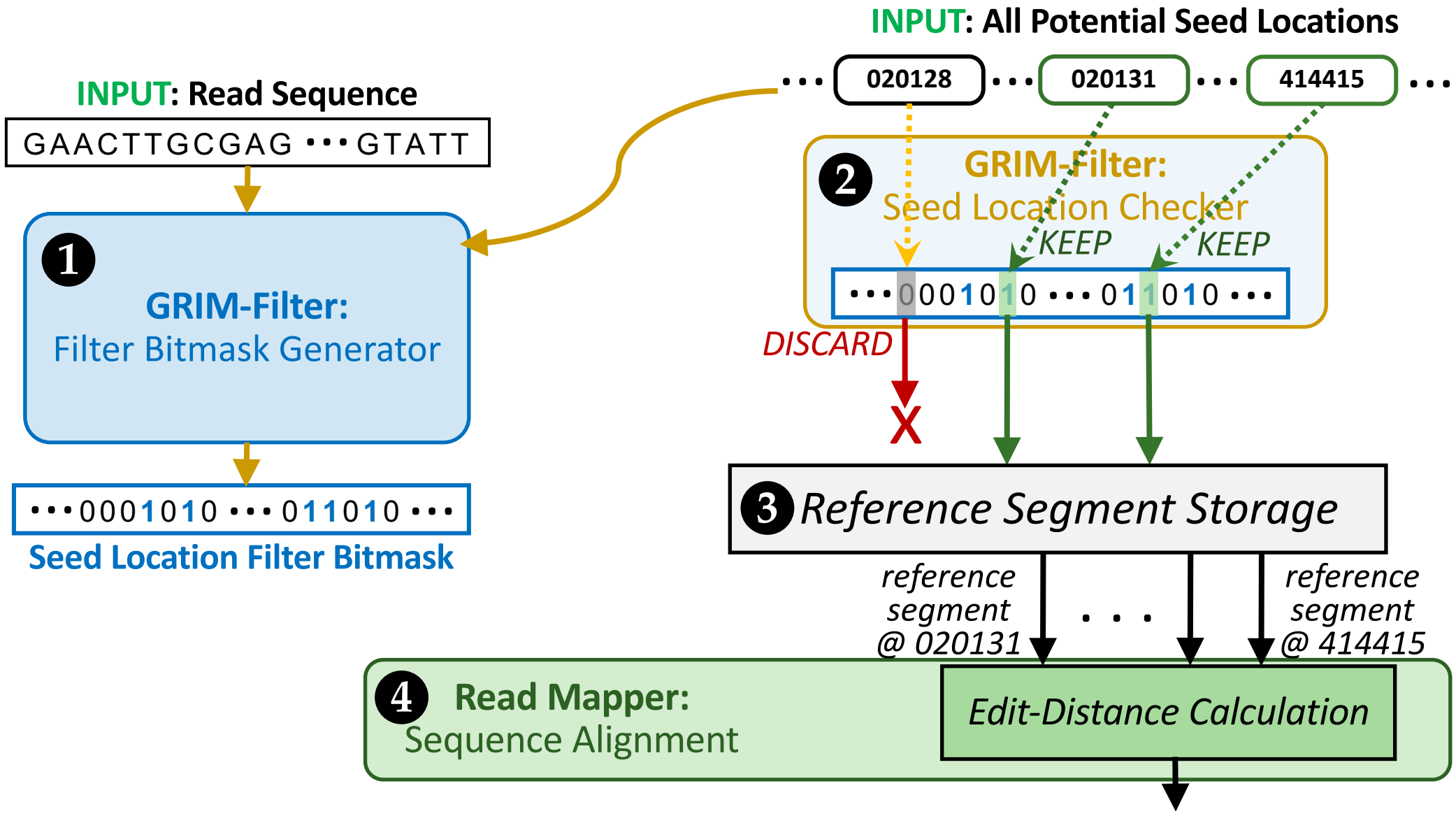
Proposal: GRIM-Filter

1. Data Structures: Bins & Bitvectors
2. Checking a Bin
3. Integrating GRIM-Filter into a Mapper

Proposal: GRIM-Filter

1. Data Structures: Bins & Bitvectors
2. Checking a Bin
3. **Integrating GRIM-Filter into a Mapper**

Integrating GRIM-Filter into a Read Mapper



GRIM-Filter Outline

1. Motivation and Goal
2. Background: Read Mappers
    a. Hash Table Based
    b. Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. Conclusion

Key Properties of GRIM-Filter

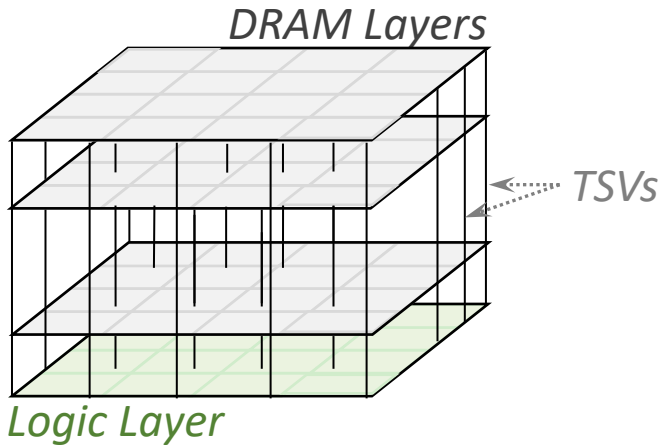
1. **Simple Operations**:
   - To check a given bin, find the **sum** of all bits corresponding to each token in the read
   - **Compare** against threshold to determine whether to align

2. **Highly Parallel**: Each bin is operated on independently and there are many many bins

3. **Memory Bound**: Given the frequent accesses to the large bitvectors, GRIM-Filter is memory bound

These properties together make GRIM-Filter a good algorithm to be run in 3D-Stacked DRAM

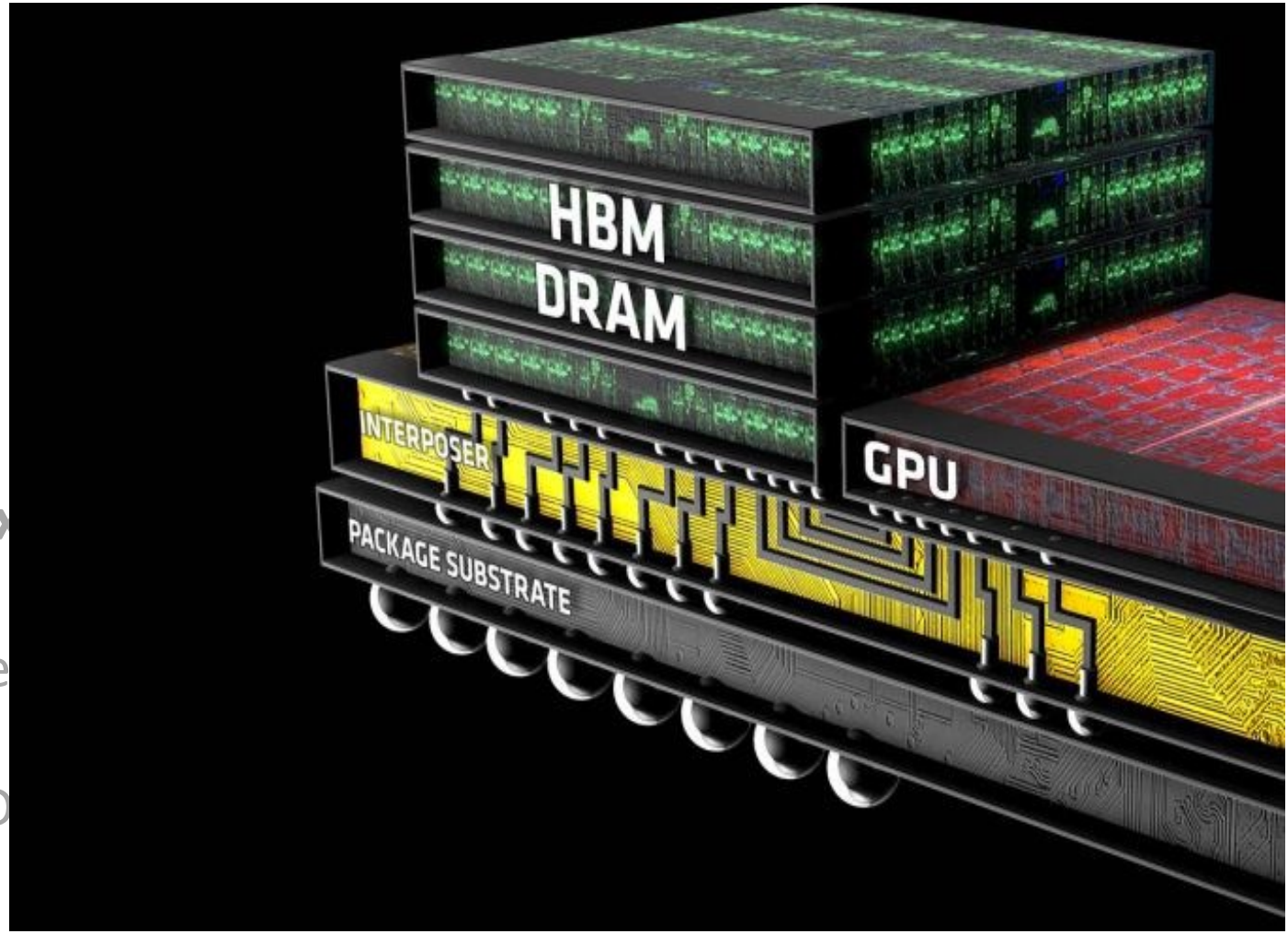
3D-Stacked Memory



- 3D-Stacked DRAM architecture has **extremely high bandwidth** as well as a stacked customizable logic layer
  - Logic Layer enables **Processing-in-Memory**, offloading computation to this layer and alleviating the memory bus
  - Embed GRIM-Filter operations into **DRAM logic layer** and appropriately distribute bitvectors throughout memory

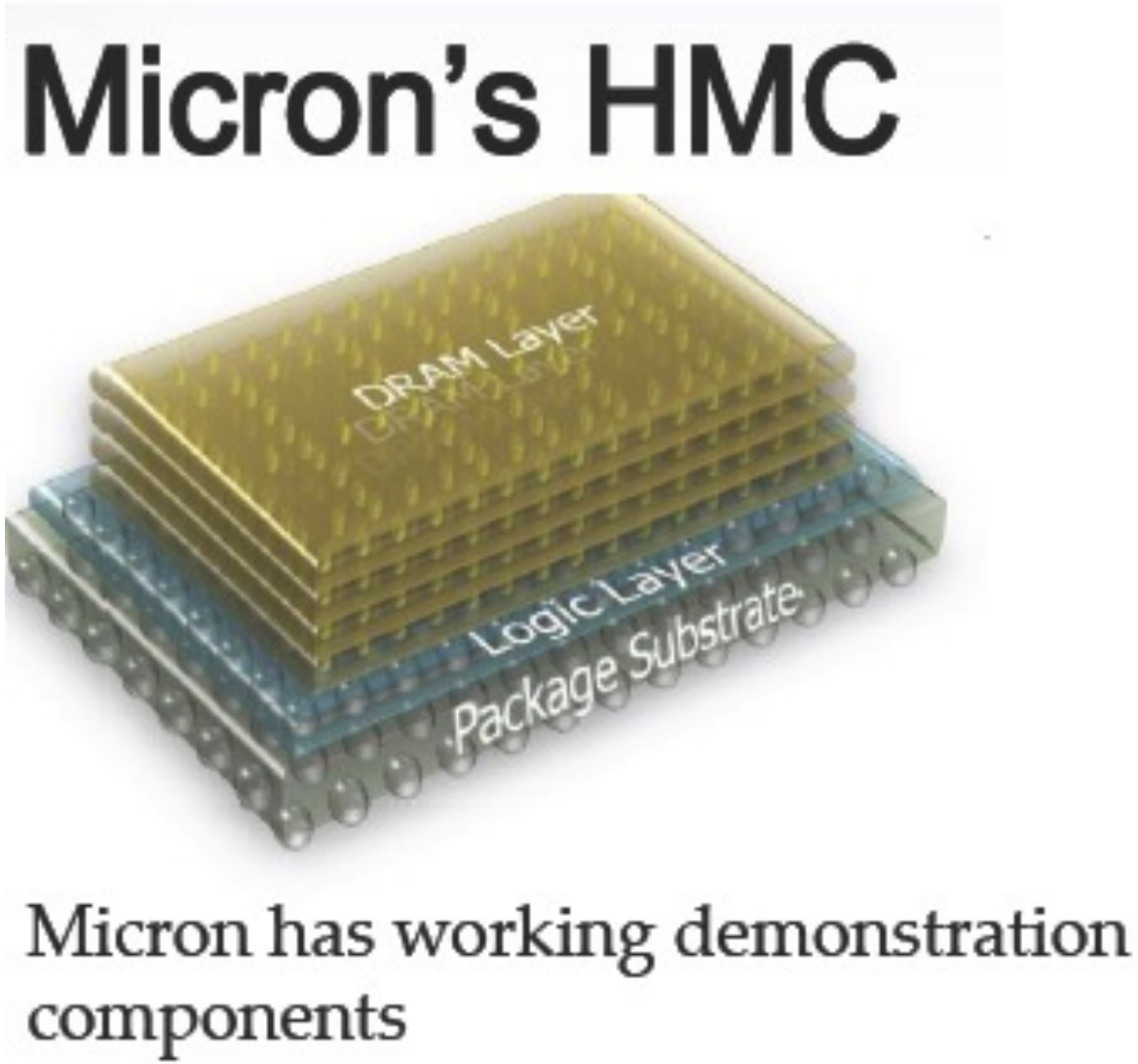
3D-Stacked Memory

- 3D-Stacked DRAM architecture has **ex** customizable logic layer
  - Logic Layer enables Processing in Me alleviating the memory bus
  - Embed GRIM-Filter operations into D bitvectors throughout memory

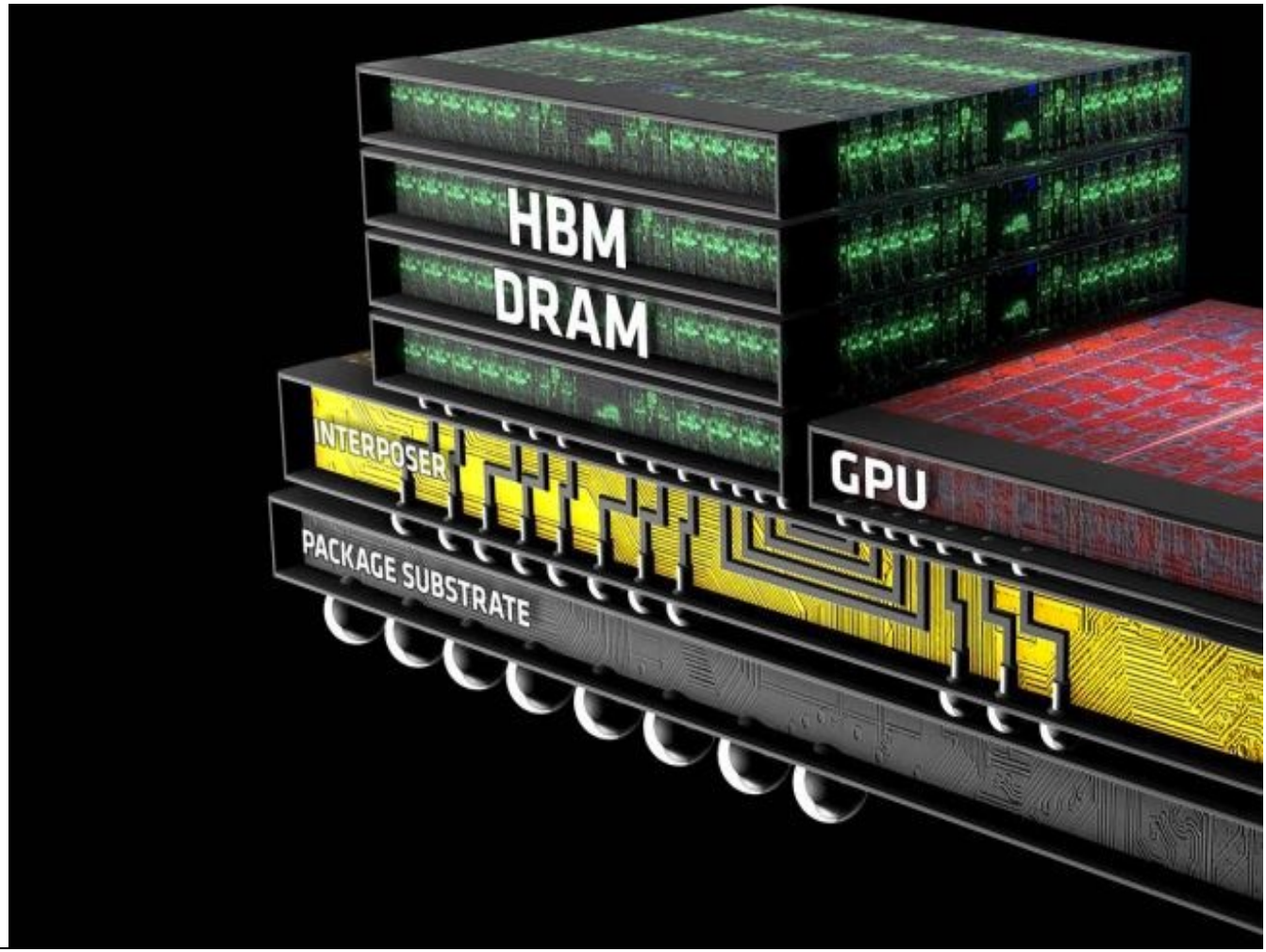


http://i1-news.softpedia-static.com/images/news2/Micron-and-Samsung-Join-Force-to-Create-Next-Gen-Hybrid-Memory-2.png

3D-Stacked Memory

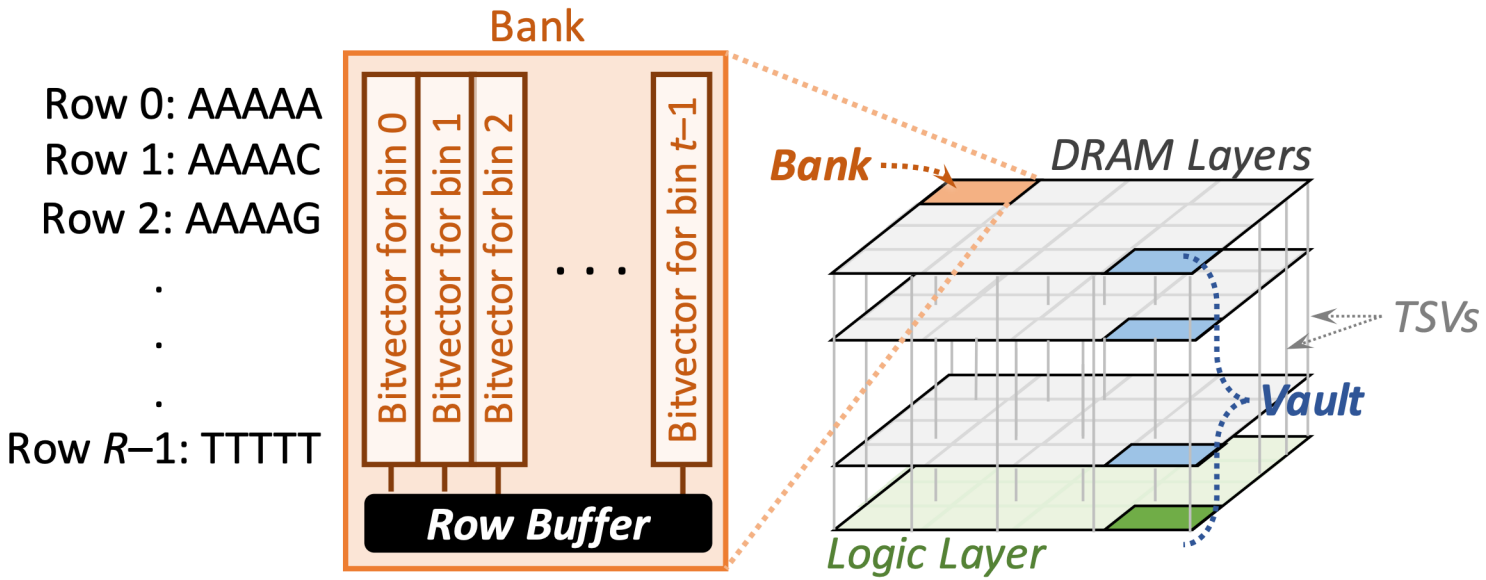


http://images.anandtech.com/doci/9266/HBMCar_678x452.jpg



http://i1-news.softpedia-static.com/images/news2/Micron-and-Samsung-Join-Force-to-Create-Next-Gen-Hybrid-Memory-2.png

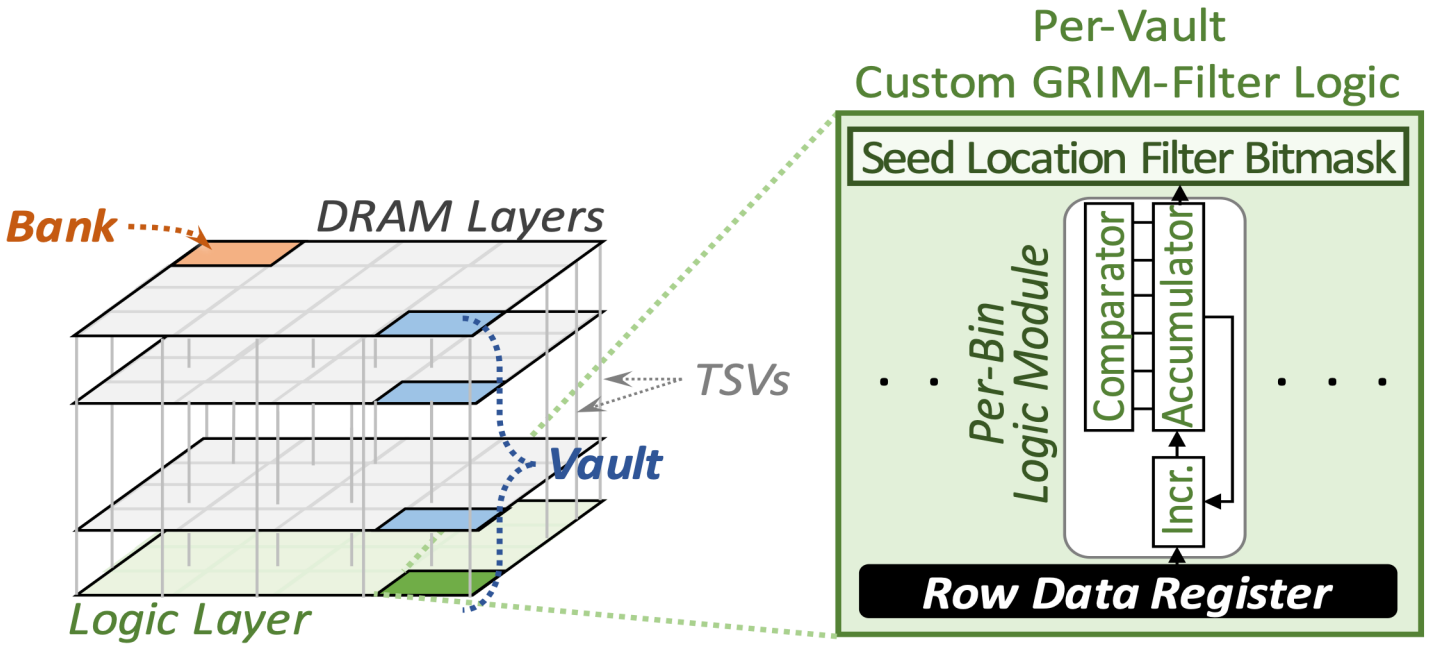
GRIM-Filter in 3D-Stacked DRAM



- Each DRAM layer is organized as an array of **banks**
  - A **bank** is an array of cells with a row buffer to transfer data

- The layout of bitvectors in a bank enables filtering many bins in parallel

GRIM-Filter in 3D-Stacked DRAM



- Customized logic for accumulation and comparison per genome segment
  - Low area overhead, simple implementation
  - For HBM2: 4096 incrementer LUTs, 7-bit counters, and comparators in logic layer

Details are in the paper

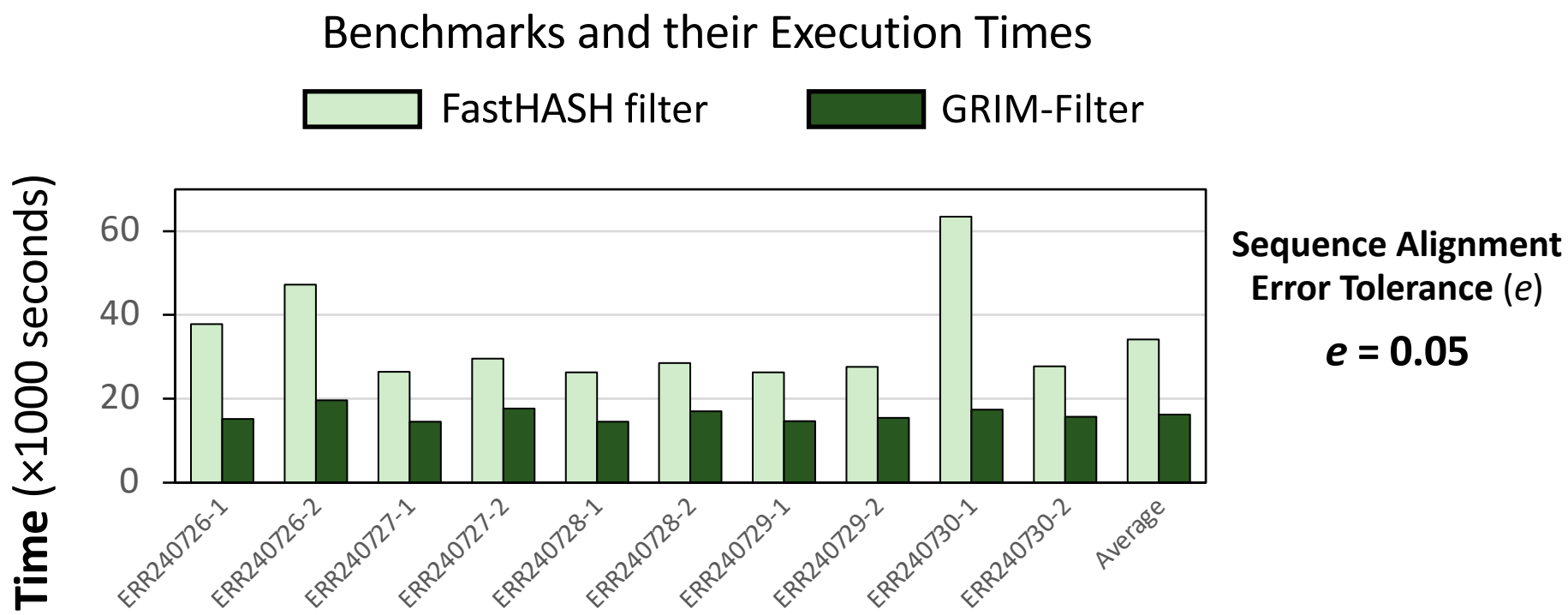
GRIM-Filter Outline

1. Motivation and Goal
2. Background: Read Mappers
    a. Hash Table Based
    b. Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. **Results**
6. Conclusion

Methodology

- Performance simulated using a custom 3D-Stacked DRAM simulator
- Evaluate 10 real read data sets (From the 1000 Genomes Project)
  - Each data set consists of 4 million reads of length 100
- Evaluate two key metrics
  - Performance
  - False negative rate
    - The fraction of locations that pass the filter but result in a mismatch
  - Compare against a state-of-the-art filter, FastHASH [Xin+, BMC Genomics 2013] when using mrFAST, but **GRIM-Filter can be used with ANY read mapper**

GRIM-Filter Performance

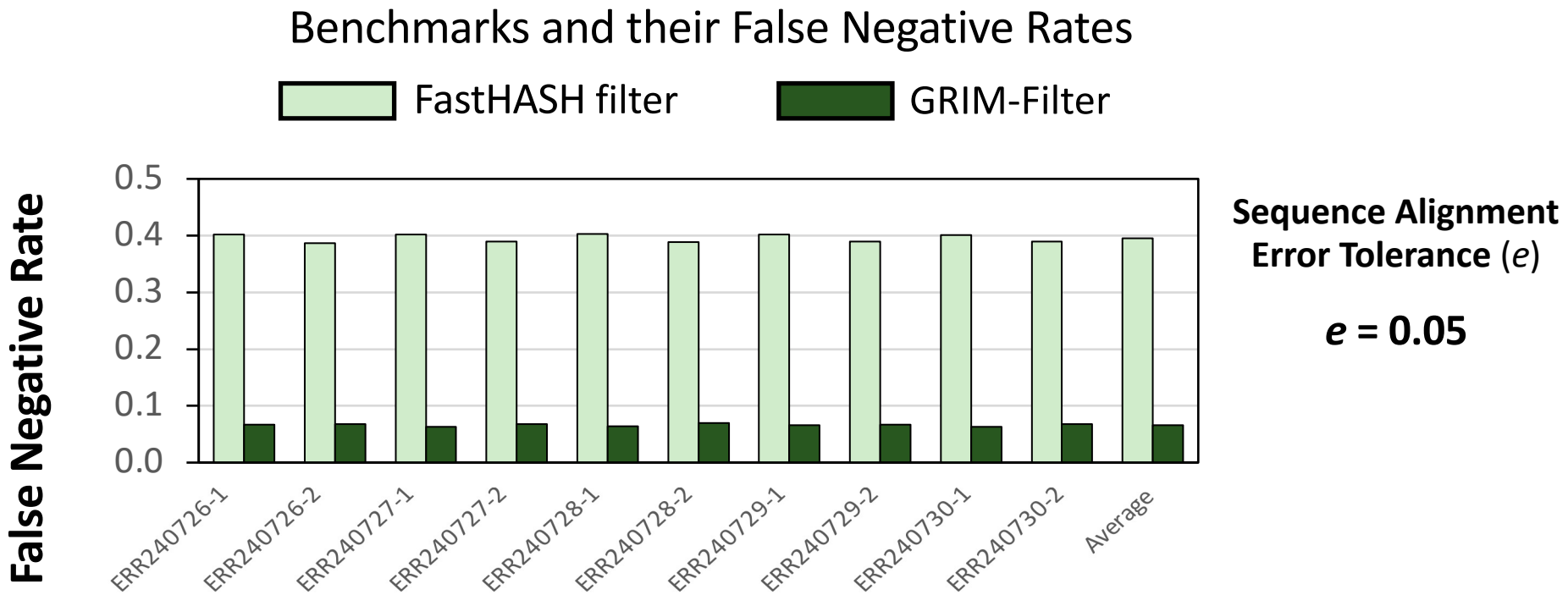


1.8x-3.7x performance benefit across real data sets

2.1x average performance benefit

GRIM-Filter gets performance due to its hardware-software co-design

GRIM-Filter False Negative Rate



5.6x-6.4x False Negative reduction across real data sets

6.0x average reduction in False Negative Rate

GRIM-Filter utilizes more information available in the read to filter

Other Results in the Paper

- Sensitivity of execution time and false negative rates to error tolerance of string matching
- Read mapper execution time breakdown
- Sensitivity studies on the filter
  - Token Size
  - Bin Size
  - Error Tolerance

GRIM-Filter Outline

1. Motivation and Goal
2. Background: Read Mappers
    a. Hash Table Based
    b. Hash Table Based with Filter
3. Proposal: GRIM-Filter
4. Mapping GRIM-Filter to 3D-Stacked Memory
5. Results
6. **Conclusion**

Conclusion

GRIM-Filter is an in-memory filtering algorithm to accelerate end-to-end read mapping by reducing the number of required alignments

Key ideas:

- Introduce a **new representation** of coarse-grained segments of the reference genome
- Use **massively-parallel in-memory operations** to identify read presence within each coarse-grained segment

Key contributions and results:

- Customized filtering algorithm for 3D-Stacked DRAM
- Compared to the previous best filter
  - The paper shows 1.8x-3.7x read mapping speedup
  - The paper shows 5.6x-6.4x fewer false negatives

GRIM-Filter is a universal filter that can be applied to any read mapper

GRIM-Filter:

Fast seed location filtering in DNA read mapping using processing-in-memory technologies

BMC Genomics, 2018

Jeremie S. Kim,
Damla Senol Cali, Hongyi Xin, Donghyuk Lee,
Saugata Ghose, Mohammed Alser, Hasan Hassan,
Oguz Ergin, Can Alkan, and Onur Mutlu

STORM

Strengths

- First seed location filter designed for processing-in-memory
  - Simple, parallel, memory-bound operations fit the logic layer of 3D-stacked memory
- Never discards a correct mapping location
- 1.8–3.7× faster read mapping and 5.6–6.4× lower false negative rate than mrFAST with FastHASH
- Careful parameter study (token size, number of bins, error tolerance)
- Freely available code, including the 3D-stacked memory simulator

Weaknesses

- Scalability to long reads is unclear

- Scaling to longer seeds will be impractical
  - Fundamental limits on the k-mer length

- GRIM-Filter ignores the positions of k-mers (tokens)

Takeaways

- Most alignment time in read mapping goes to locations that do not match
  - mrFAST with FastHASH spends 59% of its time aligning such locations

- Design the algorithm for the hardware
  - Simple counting over many bins better suits PIM

- A filter can be fast and still accurate when its threshold is conservative

Q&A

- Audience: Here, we will discuss the questions you submitted through Canvas

- Presenters: Try to guess what questions you may receive and have backup slides to answer them nicely

Discussion & Ideas

- Can GRIM-Filter work for long and error-prone reads?
  - Bins hold about 200 bases, but a long read spans thousands
  - Is there already published work on this?

- Can GRIM-Filter run on real processing-in-memory hardware today?
  - UPMEM? The logic layer of HBM? What is lost without custom logic?

- …

CMSC 829C: Algorithms, AI, and Hardware for Biological Data

Lecture 12: Example Paper Presentation – GRIM-Filter

Fall 2026

October 8, 2026

Can Firtina | canfirtina@gmail.com | cs.umd.edu/~firtina

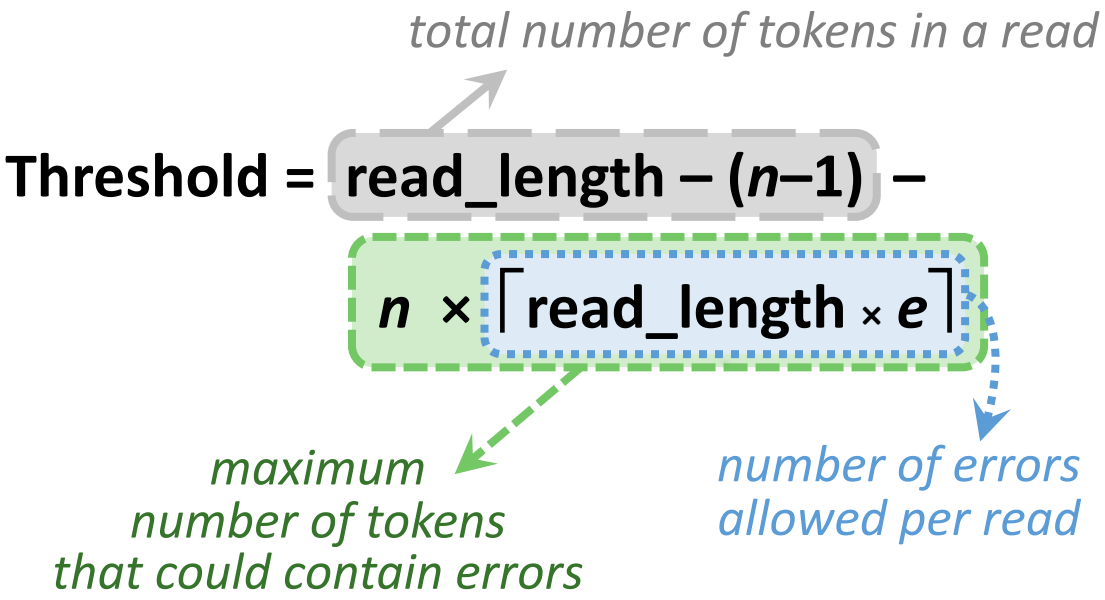
Assistant Professor of Computer Science at UMD

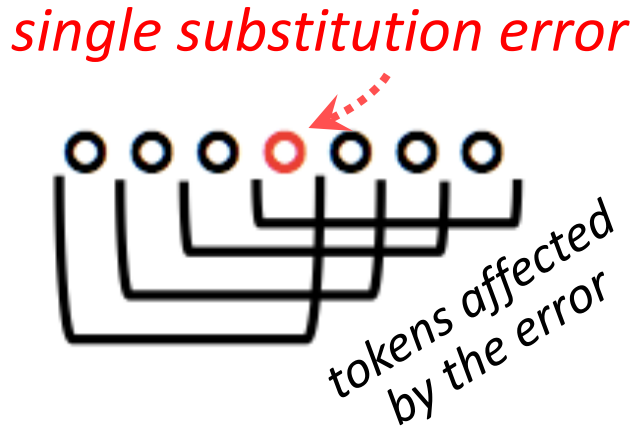


UNIVERSITY OF MARYLAND

Backup Slides

GRIM-Filter: Error Tolerance

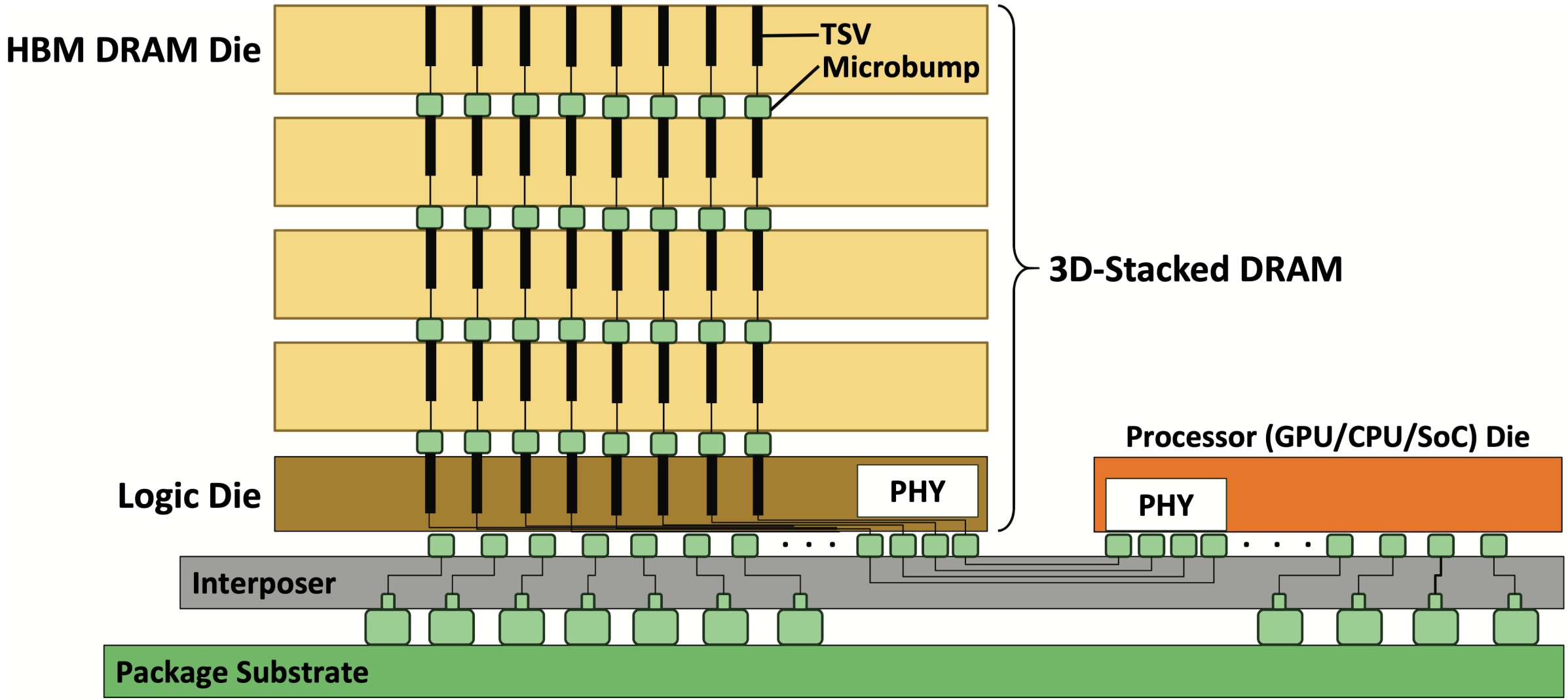


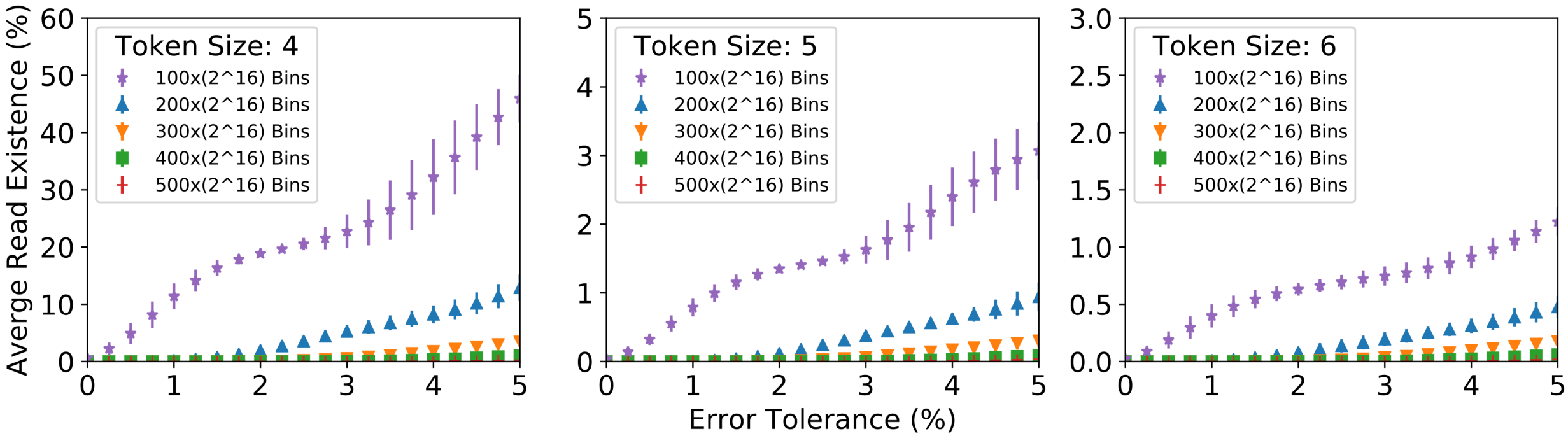


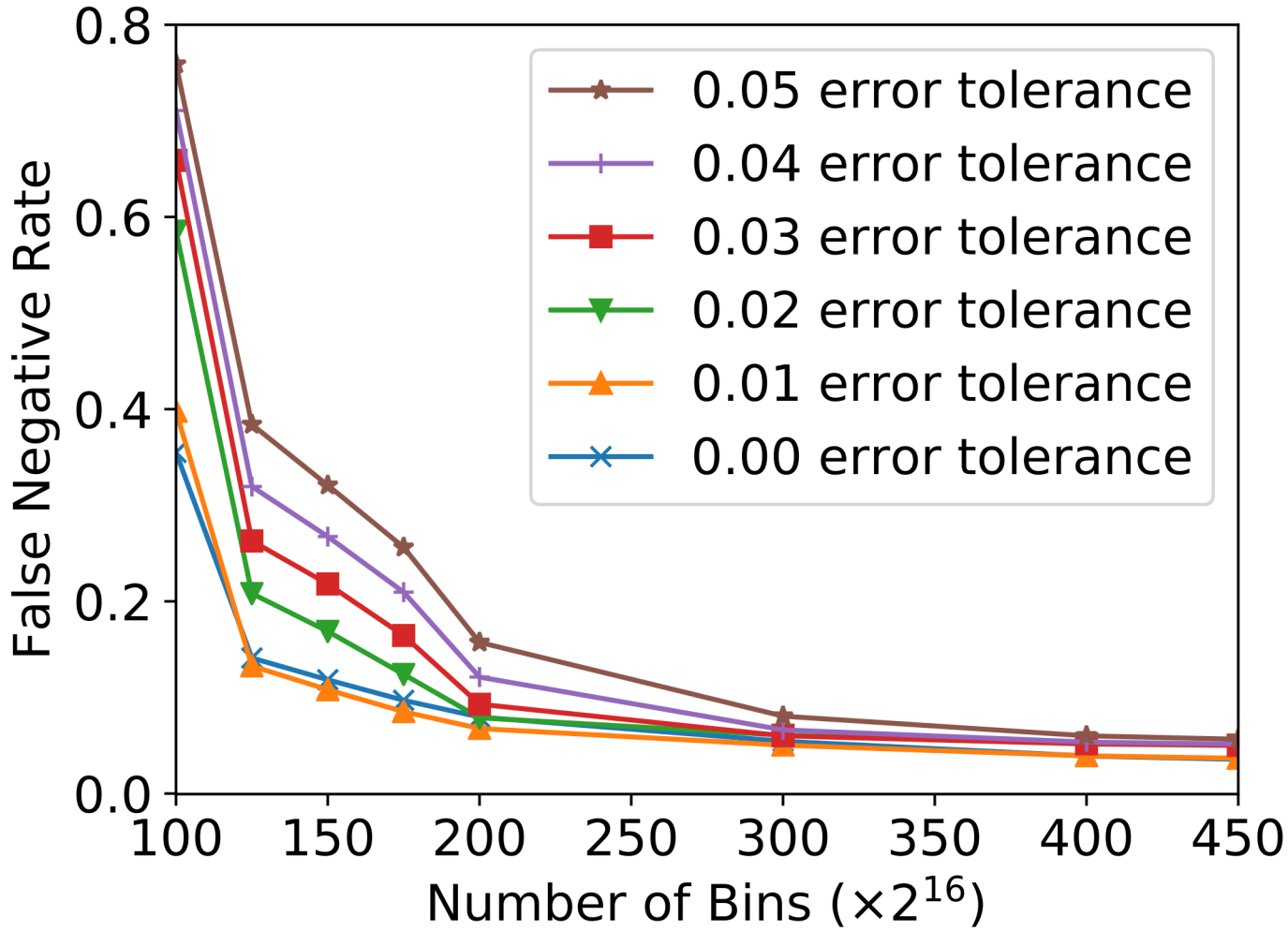
one substitution error affects four tokens when n = 4

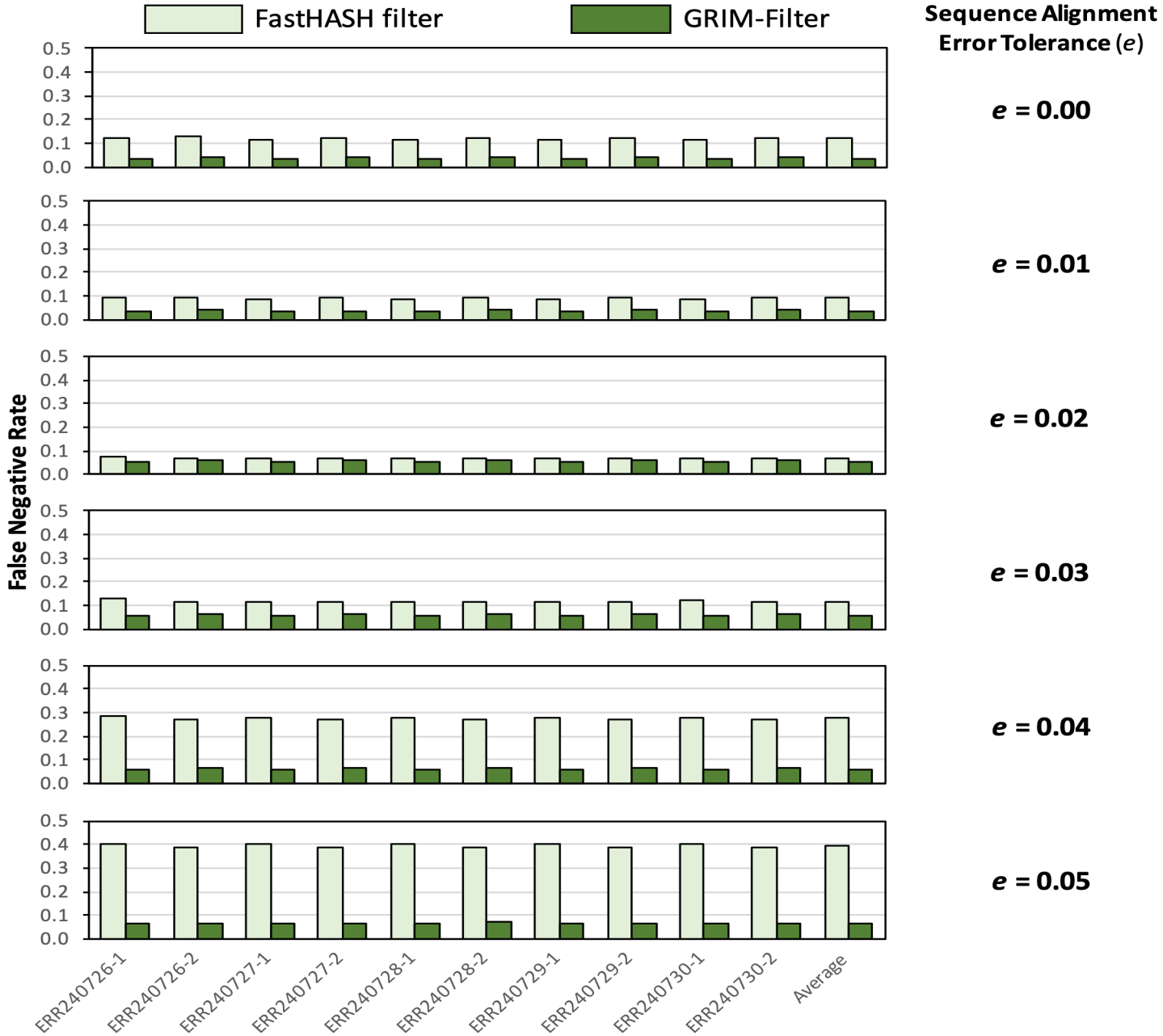
GRIM-Filter can support different error tolerances by simply changing the threshold value

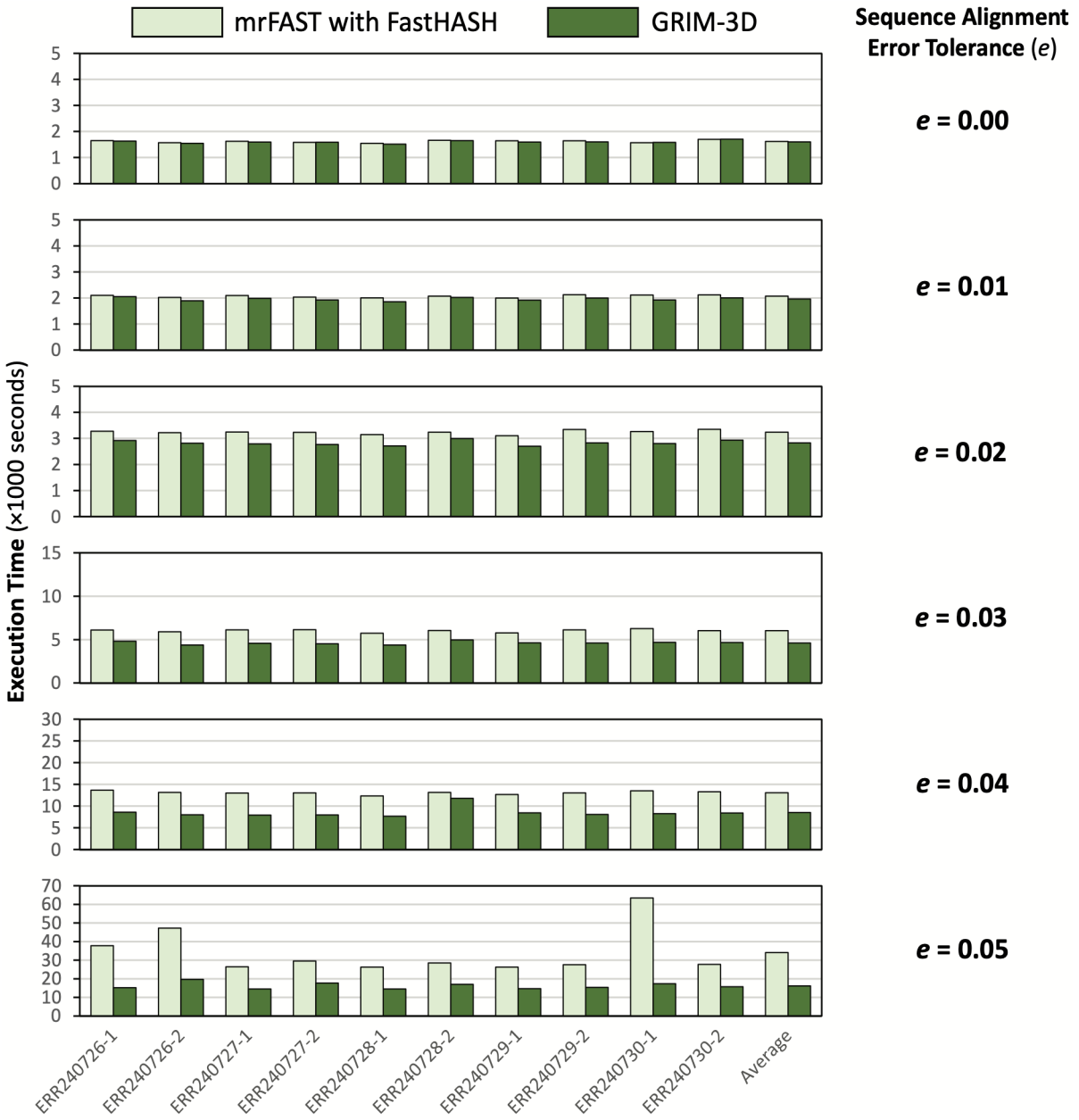
More details in the paper


HBM DRAM Die
TSV
Microbump
3D-Stacked DRAM
Logic Die
PHY
Processor (GPU/CPU/SoC) Die
PHY
Interposer
Package Substrate


Averge Read Existence (%)
Token Size: 4
100x(2^16) Bins
200x(2^16) Bins
300x(2^16) Bins
400x(2^16) Bins
500x(2^16) Bins
Token Size: 5
Token Size: 6
Error Tolerance (%)


0.8
0.6
0.4
0.2
0.0
False Negative Rate
100
150
200
250
300
350
400
450
Number of Bins (×2^16)
0.05 error tolerance
0.04 error tolerance
0.03 error tolerance
0.02 error tolerance
0.01 error tolerance
0.00 error tolerance


FastHASH filter
GRIM-Filter
Sequence Alignment Error Tolerance (e)
False Negative Rate
0.5
0.4
0.3
0.2
0.1
0.0
e = 0.00
e = 0.01
e = 0.02
e = 0.03
e = 0.04
e = 0.05
ERR240726-1
ERR240726-2
ERR240727-1
ERR240727-2
ERR240728-1
ERR240728-2
ERR240729-1
ERR240729-2
ERR240730-1
ERR240730-2
Average


mrFAST with FastHASH
GRIM-3D
Sequence Alignment Error Tolerance (e)
Execution Time (×1000 seconds)
e = 0.00
e = 0.01
e = 0.02
e = 0.03
e = 0.04
e = 0.05
ERR240726-1
ERR240726-2
ERR240727-1
ERR240727-2
ERR240728-1
ERR240728-2
ERR240729-1
ERR240729-2
ERR240730-1
ERR240730-2
Average

CMSC 829C: Algorithms, AI, and Hardware for Biological Data

Lecture 12: Example Paper Presentation – GRIM-Filter

Fall 2026

October 8, 2026

Can Firtina | canfirtina@gmail.com | cs.umd.edu/~firtina

Assistant Professor of Computer Science at UMD



UNIVERSITY OF MARYLAND