Accelerating Genome Analysis

A Primer on an Ongoing Journey

Onur Mutlu & **Saugata Ghose**

https://safari.ethz.ch/

ARM Research Summit

September 16, 2019

SAFARI ETH zürich Carnegie Mellon

Overview

- System design for bioinformatics is a critical problem
  - It has large scientific, medical, societal, personal implications
- This talk is about accelerating a key step in bioinformatics: genome sequence analysis
  - In particular, read mapping
- Many bottlenecks exist in accessing and manipulating huge amounts of genomic data during analysis
- We will cover various recent ideas to accelerate read mapping
  - Our group's journey since September 2006

Agenda

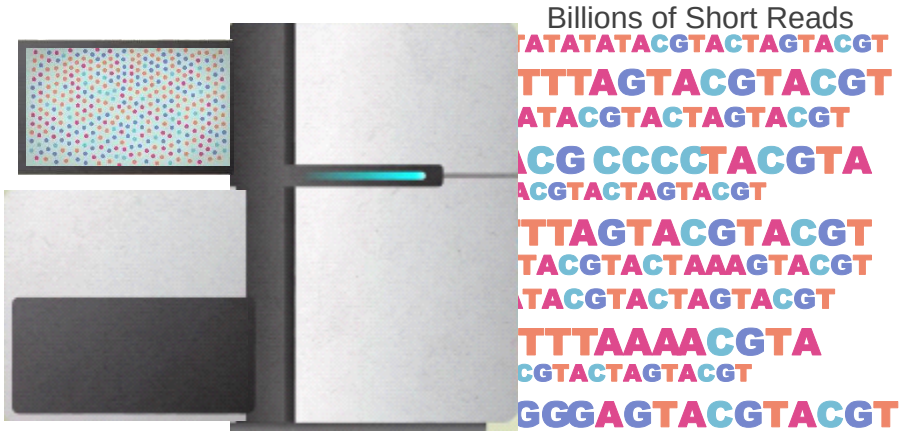
- **The Problem: DNA Read Mapping**
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
    FastHASH [Xin+, BMC Genomics 2013]
  - Exploiting SIMD Instructions
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- Hardware Acceleration
  - Specialized Architectures
    GateKeeper [Alser+, Bioinformatics 2017], MAGNET [Alser+, TIR 2017], Shouji [Alser+, Bioinformatics 2019]
  - Processing in Memory
    GRIM-Filter [Kim+, BMC Genomics 2018]
- Future Opportunities: New Sequencing Technologies
  [Senol Cali+, Briefings in Bioinformatics 2018]

What Is DNA Sequencing?

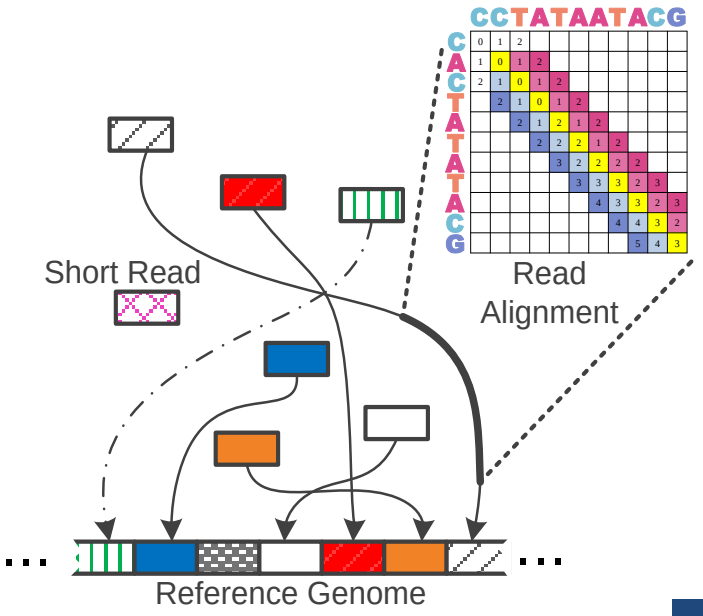
- Goal: **Find the complete sequence of A, C, G, T's in DNA**

- Challenge:
  - There is no machine that takes long DNA as an input, and gives the complete sequence as output
  - All sequencing machines **chop DNA into pieces** and identify relatively small pieces (**but not how they fit together**)

- Human Genome Project (1990 – 2003) assembled a full human reference genome
- **High Throughput Sequencing** is much faster, but needs to chop the genome into **short reads** (75-300 base pairs)



1 Sequencing



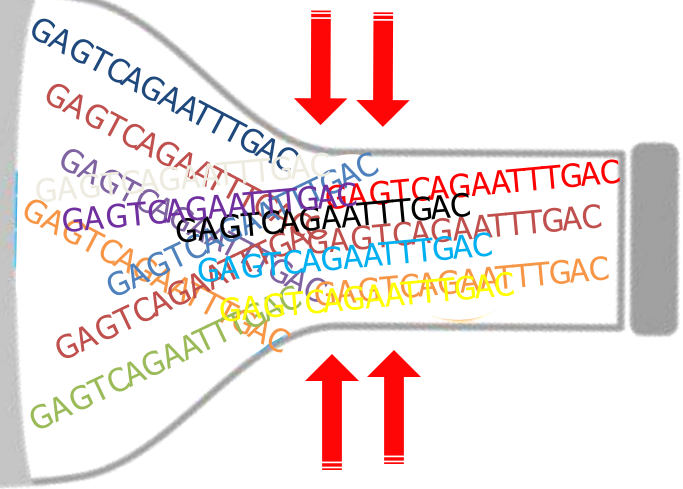
Read Mapping 2

Bottlenecked in Mapping!!

Illumina HiSeq4000

300 M

bases/min



on average

2 M

bases/min

(0.6%)

Challenges in Read Mapping

- Need to find many mappings of each read
  - A short read may map to many locations, especially with High-Throughput DNA Sequencing technologies
  - How can we find all mappings efficiently?

- Need to tolerate small variances/errors in each read
  - Each individual is different: Subject's DNA may slightly differ from the reference (mismatches, insertions, deletions)
  - How can we efficiently map each read with up to *e* errors present?

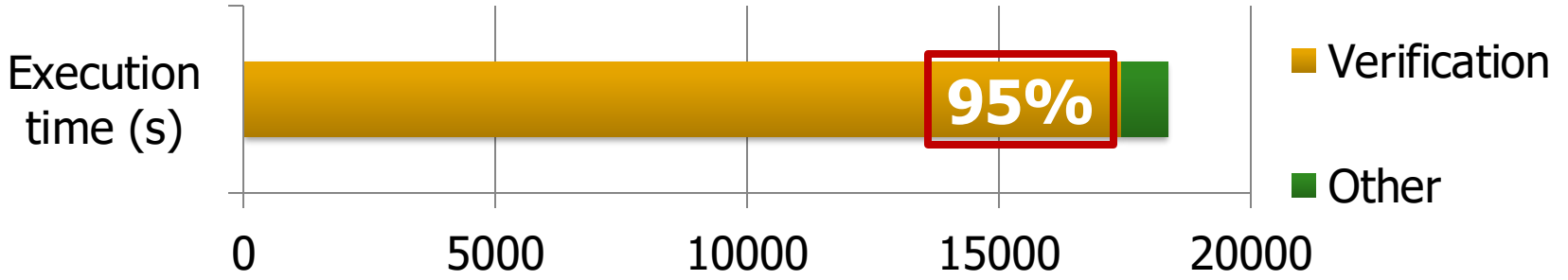
- Need to map each read very fast (i.e., performance is important)
  - Human DNA is 3.2 billion base pairs long → Millions to billions of reads (State-of-the-art mappers take weeks to map a human's DNA)
  - How can we design a much higher performance read mapper?

Agenda

- The Problem: DNA Read Mapping
- **Algorithmic Acceleration**
  - **Exploiting Structure of the Genome**
    FastHASH [Xin+, BMC Genomics 2013]
  - Exploiting SIMD Instructions
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- Hardware Acceleration
  - Specialized Architectures
    GateKeeper [Alser+, Bioinformatics 2017], MAGNET [Alser+, TIR 2017], Shouji [Alser+, Bioinformatics 2019]
  - Processing in Memory
    GRIM-Filter [Kim+, BMC Genomics 2018]
- Future Opportunities: New Sequencing Technologies
  [Senol Cali+, Briefings in Bioinformatics 2018]

Problem and Goal

- Poor performance of existing read mappers: Very slow
  - Verification/alignment takes too long to execute
  - Verification requires a memory access for reference genome + many base-pair-wise comparisons between the reference and the read (edit distance computation)



- Goal: Speed up the mapper by reducing the cost of verification

Reducing the Cost of Verification

- We observe that most verification (edit distance computation) calculations are unnecessary
  - 1 out of 1000 potential locations passes the verification process

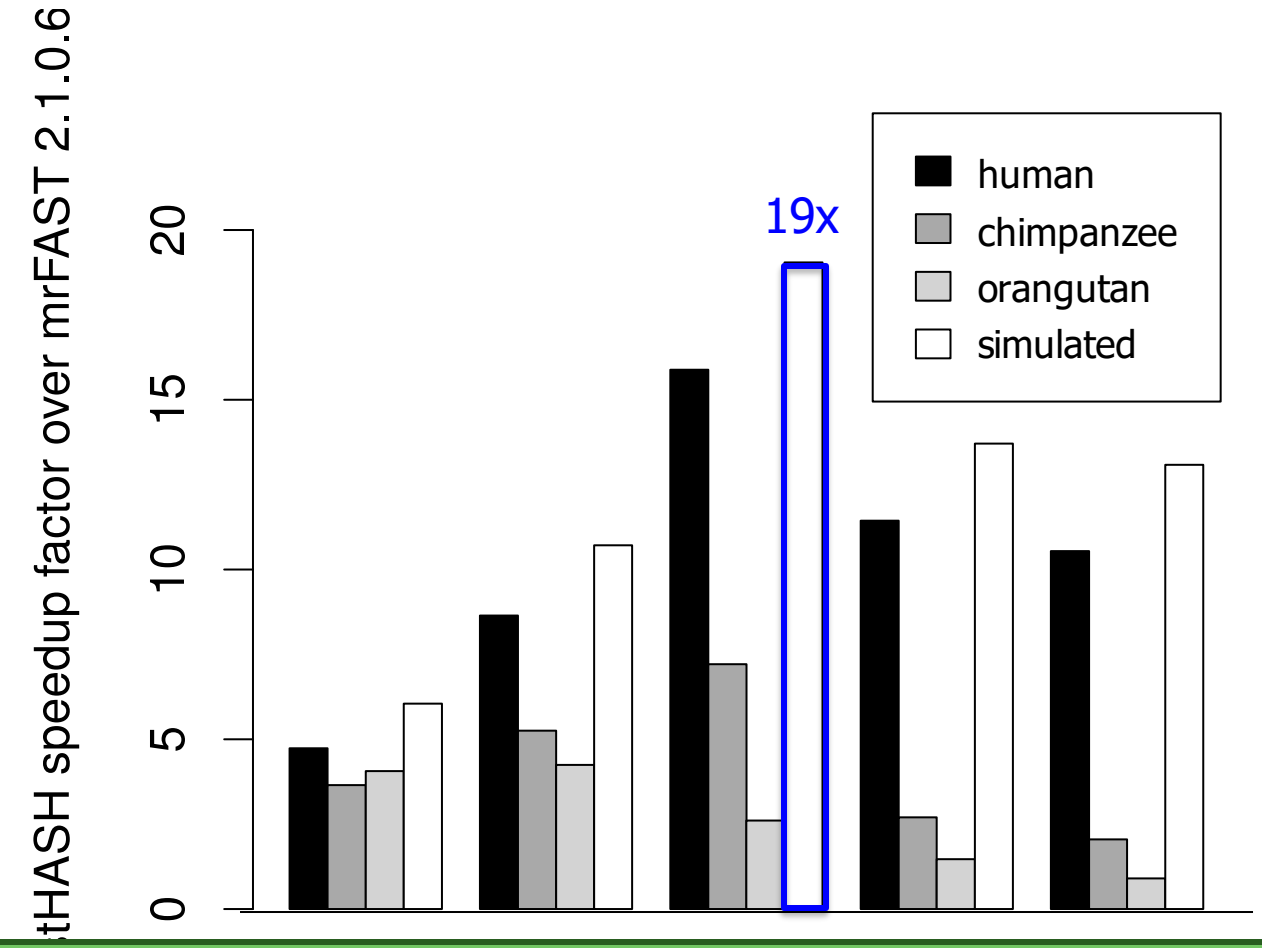
- We observe that we can get rid of unnecessary verification calculations by
  - *Detecting and rejecting* *early* invalid mappings:
    Filter fast before you align
  - *Reducing* the *number* of potential mappings to examine:
    Minimize costly edit distance computations

FastHASH: Efficient Read Filtering

1. **Adjacency Filtering (AF)**
   - Observation: Adjacent k-mers in the read should also be adjacent in the reference genome
   - Read mapper can quickly reject mappings that do **not** satisfy this property

2. **Cheap K-mer Selection (CKS)**
   - Observation: Some k-mers are cheaper to verify than others because they have shorter location lists (they occur less frequently in the reference genome)
     - Mapper needs to examine only *e+1* k-mers' locations to tolerate *e* errors
   - Read mapper can choose the cheapest *e+1* k-mers and verify their locations

FastHASH Speedup: Entire Read Mapper



With FastHASH, new mrFAST obtains up to 19x speedup over previous version, without losing valid mappings

Agenda

- The Problem: DNA Read Mapping
- **Algorithmic Acceleration**
  - Exploiting Structure of the Genome
    FastHASH [Xin+, BMC Genomics 2013]
  - **Exploiting SIMD Instructions**
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- Hardware Acceleration
  - Specialized Architectures
    GateKeeper [Alser+, Bioinformatics 2017], MAGNET [Alser+, TIR 2017], Shouji [Alser+, Bioinformatics 2019]
  - Processing in Memory
    GRIM-Filter [Kim+, BMC Genomics 2018]
- Future Opportunities: New Sequencing Technologies
  [Senol Cali+, Briefings in Bioinformatics 2018]

Shifted Hamming Distance

- Observation: If two strings differ by *E* edits, then every bp match can be aligned in at most *2E* shifts (of one of the strings).
  - Insight: Shifting a string by one "corrects" for one "error"

- Compute "Shifted Hamming Distance"
  - AND of 2E Hamming Distances of two strings, to filter out invalid mappings
  - Uses bit-parallel operations that nicely map to SIMD instructions

- Key result:
  - SHD is 3x faster than SeqAn (the best implementation of Gene Myers' bit-vector algorithm), with only a 7% false positive rate
  - The fastest CPU-based filtering (pre-alignment) mechanism

New Bottleneck: Filtering (Pre-Alignment)

Sequencing generates many reads, each of which potentially mapping to many locations

→

Filtering (Pre-alignment) eliminates the need to verify/align read to invalid mapping locations

→

Alignment/verification (costly edit distance computation) is performed **only** on reads that pass the filter

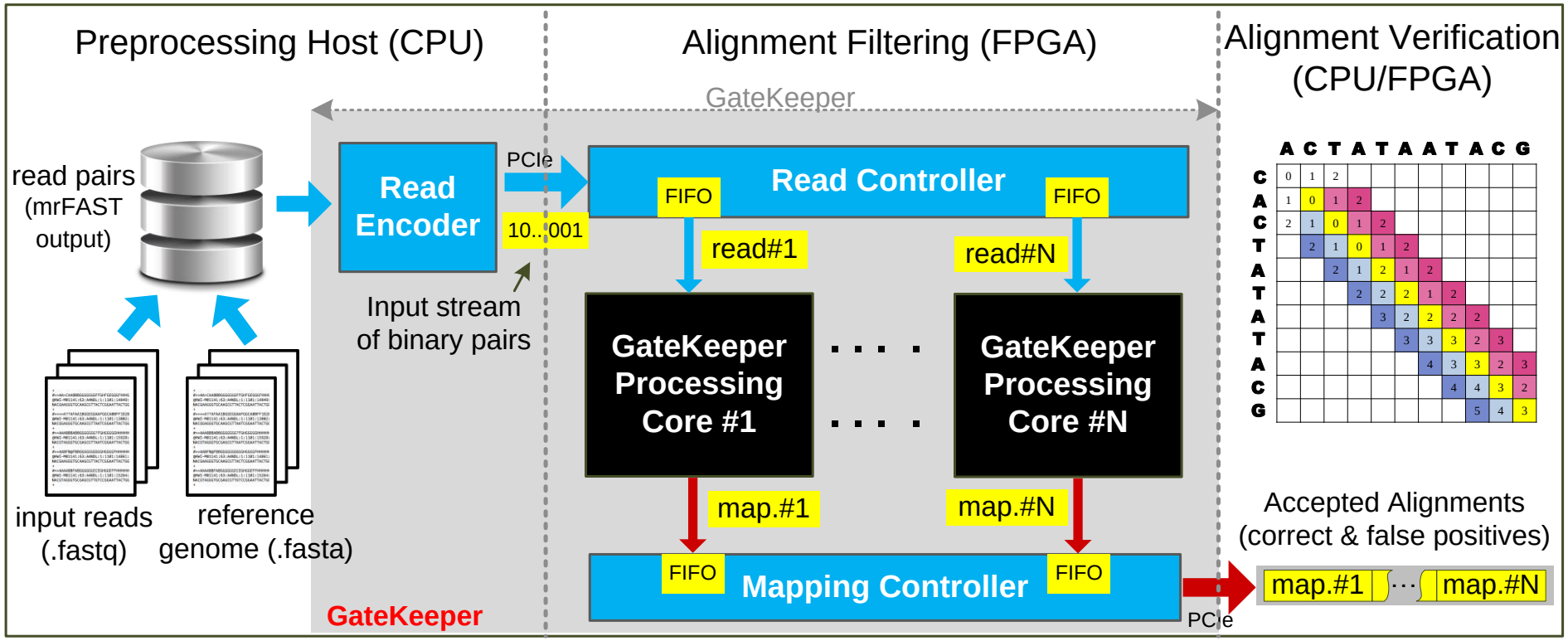
New bottleneck in read mapping becomes the "filtering (pre-alignment)" step

Agenda

- The Problem: DNA Read Mapping
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
    FastHASH [Xin+, BMC Genomics 2013]
  - Exploiting SIMD Instructions
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- **Hardware Acceleration**
  - **Specialized Architectures**
    GateKeeper [Alser+, Bioinformatics 2017], **MAGNET** [Alser+, TIR 2017], **Shouji** [Alser+, Bioinformatics 2019]
  - Processing in Memory
    GRIM-Filter [Kim+, BMC Genomics 2018]
- Future Opportunities: New Sequencing Technologies
  [Senol Cali+, Briefings in Bioinformatics 2018]

GateKeeper: FPGA-Accelerated Filtering

- **Maximum data throughput** =~13.3 billion bases/sec
- Can examine **8 (300 bp) or 16 (100 bp) mappings concurrently** at 250 MHz
- **Occupies 50%** (100 bp) to **91%** (300 bp) of the FPGA slice LUTs and registers



GateKeeper: Speed & Accuracy Results

90x-130x faster filter

than SHD (Xin et al., 2015) and the Adjacency Filter (Xin et al., 2013)

4x lower false accept rate

than the Adjacency Filter (Xin et al., 2013)

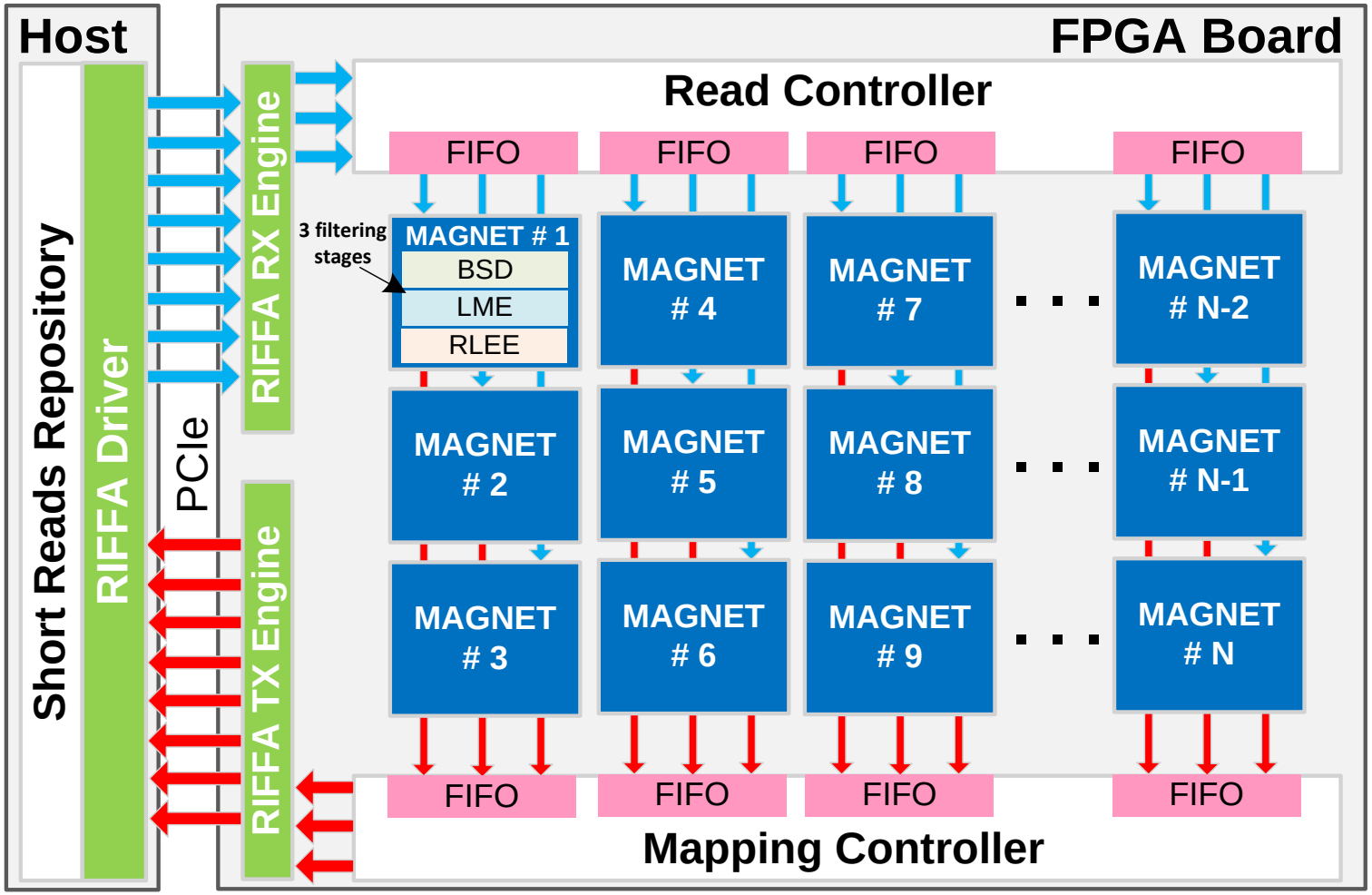
10x speedup in read mapping

with the addition of GateKeeper to the mrFAST mapper (Alkan et al., 2009)

Freely available online

github.com/BilkentCompGen/GateKeeper

MAGNET: Increased Accuracy Over GateKeeper



Shouji: FPGA-Based Sliding Window Filter

- Alser+, **"Shouji: A Fast and Efficient Pre-Alignment Filter for Sequence Alignment"** ***Bioinformatics***, 2019.

- Open-source: https://github.com/CMU-SAFARI/Shouji

Sequence alignment

Shouji: a fast and efficient pre-alignment filter for sequence alignment

Mohammed Alser[1,2,3,*], Hasan Hassan[1], Akash Kumar[2], Onur Mutlu[1,3,*] and Can Alkan[3,*]

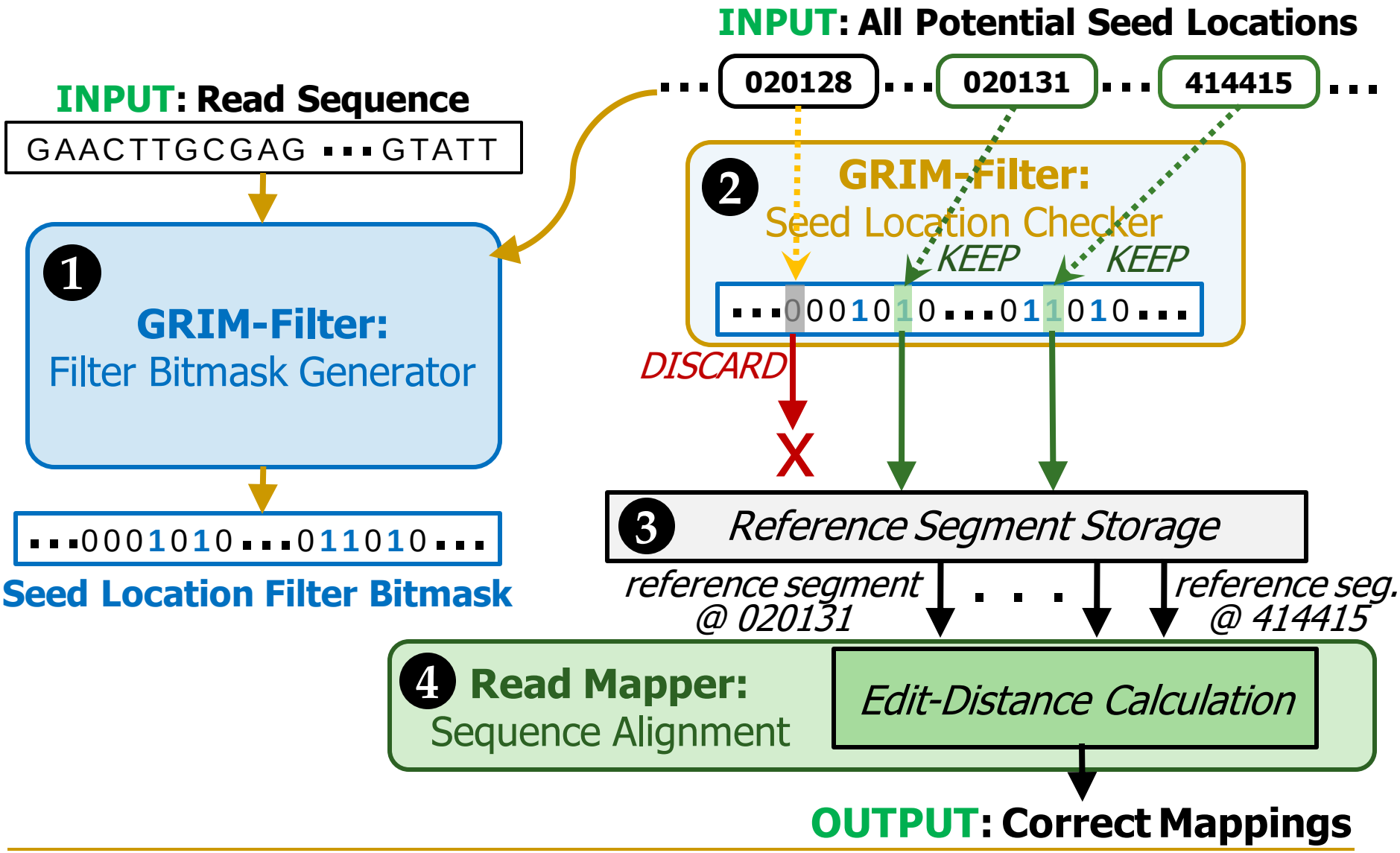
Agenda

- The Problem: DNA Read Mapping
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
    FastHASH [Xin+, BMC Genomics 2013]
  - Exploiting SIMD Instructions
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- **Hardware Acceleration**
  - Specialized Architectures
    GateKeeper [Alser+, Bioinformatics 2017], MAGNET [Alser+, TIR 2017], Shouji [Alser+, Bioinformatics 2019]
  - **Processing in Memory**
    GRIM-Filter [Kim+, BMC Genomics 2018]
- Future Opportunities: New Sequencing Technologies
  [Senol Cali+, Briefings in Bioinformatics 2018]

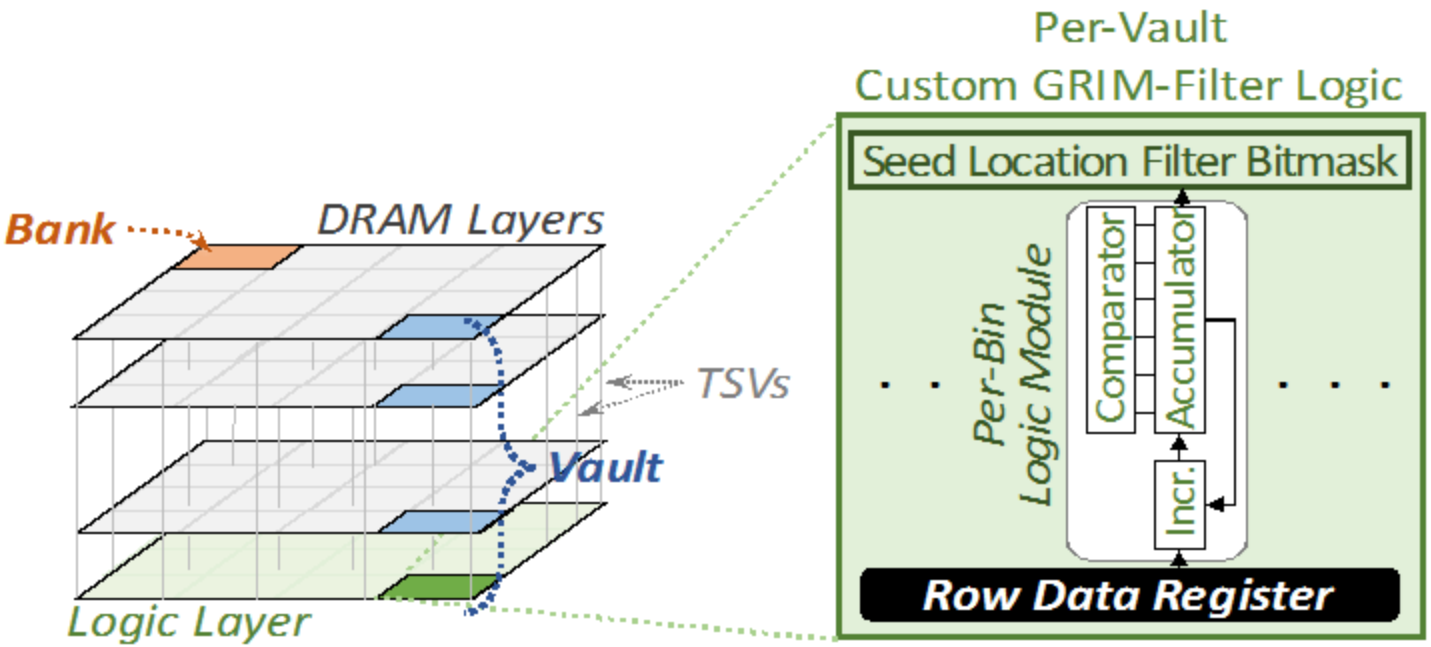
Read Mapping & Filtering

- Problem: Heavily bottlenecked by Data Movement
  - GateKeeper performance limited by DRAM bandwidth [Alser+, Bioinformatics 2017]
  - Ditto for SHD [Xin+, Bioinformatics 2015]
- Solution: Processing-in-memory can alleviate the bottleneck
- However, we need to design mapping & filtering algorithms to fit processing-in-memory

GRIM-Filter: Fast Seed Location Filtering



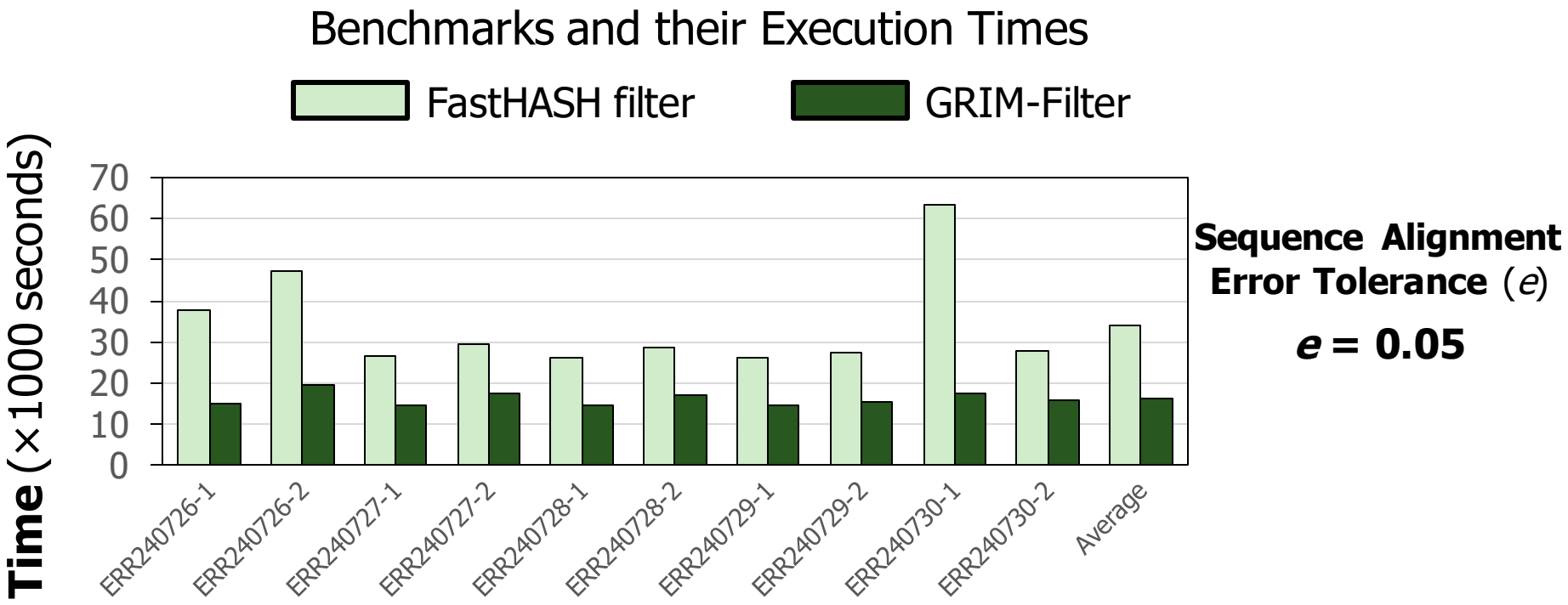
Adding GRIM-Filter to 3D-Stacked DRAM



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

Details are in [Kim+, BMC Genomics 2018]

GRIM-Filter Outperforms Other Filters



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

Agenda

- The Problem: DNA Read Mapping
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
    FastHASH [Xin+, BMC Genomics 2013]
  - Exploiting SIMD Instructions
    Shifted Hamming Distance [Xin+, Bioinformatics 2015]
- Hardware Acceleration
  - Specialized Architectures
    GateKeeper [Alser+, Bioinformatics 2017], MAGNET [Alser+, TIR 2017], Shouji [Alser+, Bioinformatics 2019]
  - Processing in Memory
    GRIM-Filter [Kim+, BMC Genomics 2018]
- **Future Opportunities: New Sequencing Technologies**
  [Senol Cali+, Briefings in Bioinformatics 2018]

Nanopore Sequencing Technology

- **Nanopore sequencing** is an emerging and a promising single-molecule DNA sequencing technology
  - No amplification → Less limit on read length → Longer read length

- First nanopore sequencing device, **MinION**, made commercially available by **Oxford Nanopore Technologies** (ONT) in **May 2014**
  - Inexpensive
  - Long read length (> 882K bp)
  - Portable: Pocket-sized
  - Produces data in real-time

Challenges of Nanopore Sequencing

- One major drawback: high error rates
- Nanopore sequence analysis tools have a critical role to:
  - overcome high error rates
  - take better advantage of the technology
- Faster tools are critically needed to:
  - Take better advantage of the real-time data production capability of MinION
  - Enable fast, real-time data analysis

More on Nanopore Sequencing & Tools

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

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

Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017
Published: 02 April 2018 **Article history** ▾

BiB arXiv

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

Conclusion

- System design for bioinformatics is a critical problem
  - It has large scientific, medical, societal, personal implications
- This talk is about accelerating a key step in bioinformatics: genome sequence analysis
  - In particular, read mapping
- We covered various recent ideas to accelerate read mapping
  - Our group's journey since September 2006
- **Many future opportunities exist**
  - **Especially with new sequencing technologies**
  - **Especially with new applications and use cases**

Acknowledgments

- Can Alkan, Bilkent University

- Many students at ETH, CMU, Bilkent
  - Mohammed Alser, Damla Senol Cali, Jeremie Kim, Hasan Hassan, Donghyuk Lee, Hongyi Xin, …

- Funders:
  - NIH and Industrial Partners (Alibaba, AMD, Google, Facebook, HP Labs, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, Qualcomm, Rambus, Samsung, Seagate, VMware)

- All papers, source code, and more are at:
  - https://people.inf.ethz.ch/omutlu/projects.htm

Accelerating Genome Analysis

A Primer on an Ongoing Journey

Onur Mutlu & **Saugata Ghose**

https://safari.ethz.ch/

ARM Research Summit

September 16, 2019

SAFARI ETH zürich Carnegie Mellon

OUR BIOINFORMATICS PUBLICATIONS

Our Bioinformatics Works (I)

- Algorithmic Acceleration (I)
  - C. Alkan, J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, S. C. Sahinalp, R. A. Gibbs, E. E. Eichler, **"Personalized copy number and segmental duplication maps using next-generation sequencing"**, ***Nature Genetics***, 2009. [Source Code]
  - H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, C. Alkan, **"Accelerating Read Mapping with FastHASH"**, ***BMC Genomics***, 2013. Also appears in ***APBC***, 2013. [Source Code]
  - D. Lee, F. Hormozdiari, H. Xin, F. Hach, O. Mutlu, C. Alkan, **"Fast and Accurate Mapping of Complete Genomics Reads"**, ***Methods***, 2014. [Source Code]

Our Bioinformatics Works (II)

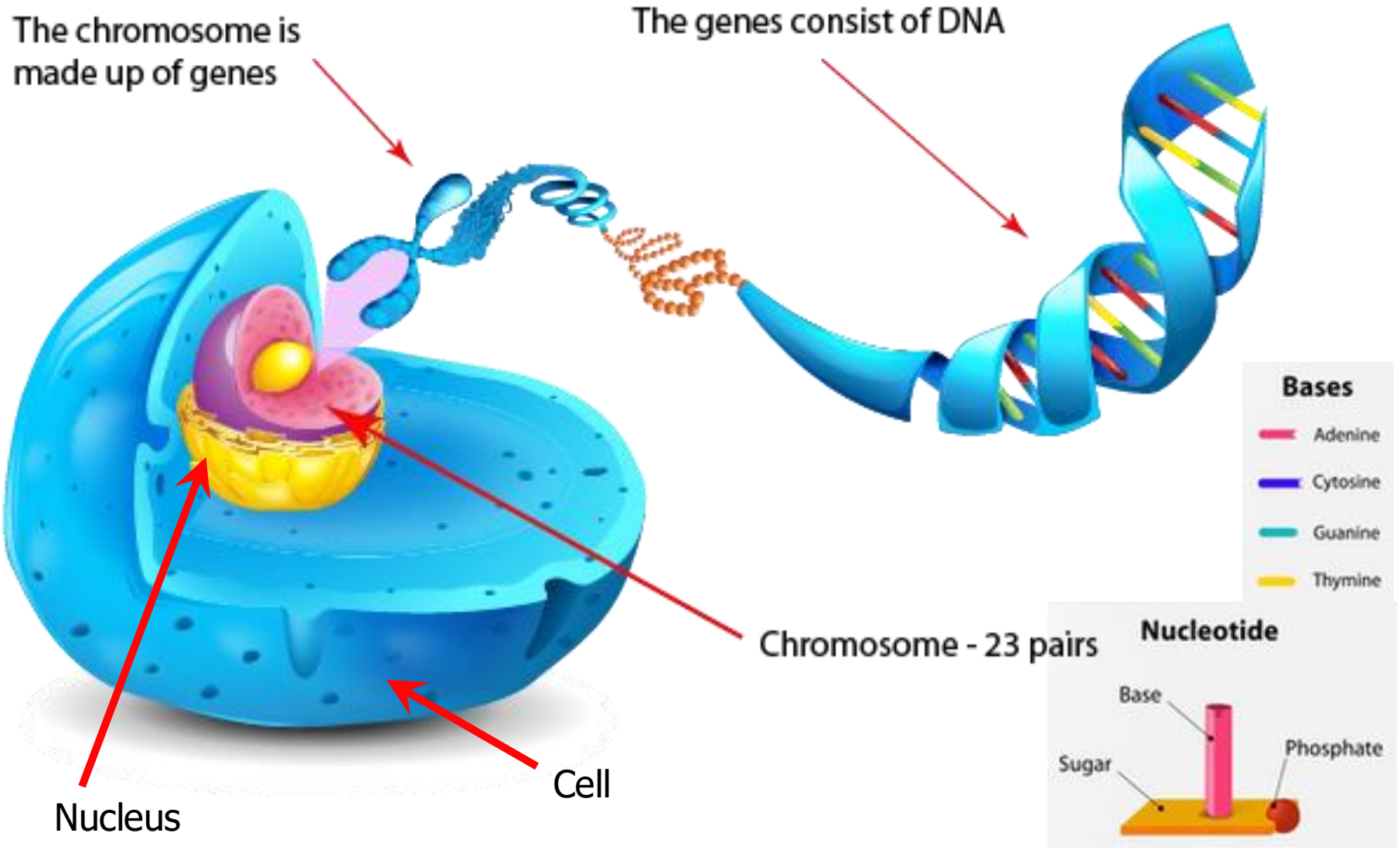
- Algorithmic Acceleration (II)
  - H. Xin, J. Greth, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, O. Mutlu, **"Shifted Hamming Distance: A Fast and Accurate SIMD-friendly Filter to Accelerate Alignment Verification in Read Mapping"**, ***Bioinformatics***, 2015. [Source Code]
  - H. Xin, S. Nahar, R. Zhu, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, O. Mutlu, **"Optimal Seed Solver: Optimizing Seed Selection in Read Mapping"**, ***Bioinformatics***, 2015. [Source Code]
- Genome Sequencing Pipeline Analysis
  - D. Senol Cali, J. S. Kim, S. Ghose, C. Alkan, O. Mutlu, **"Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions"**, ***Briefings in Bioinformatics (BIB)***, 2018.

Our Bioinformatics Works (III)

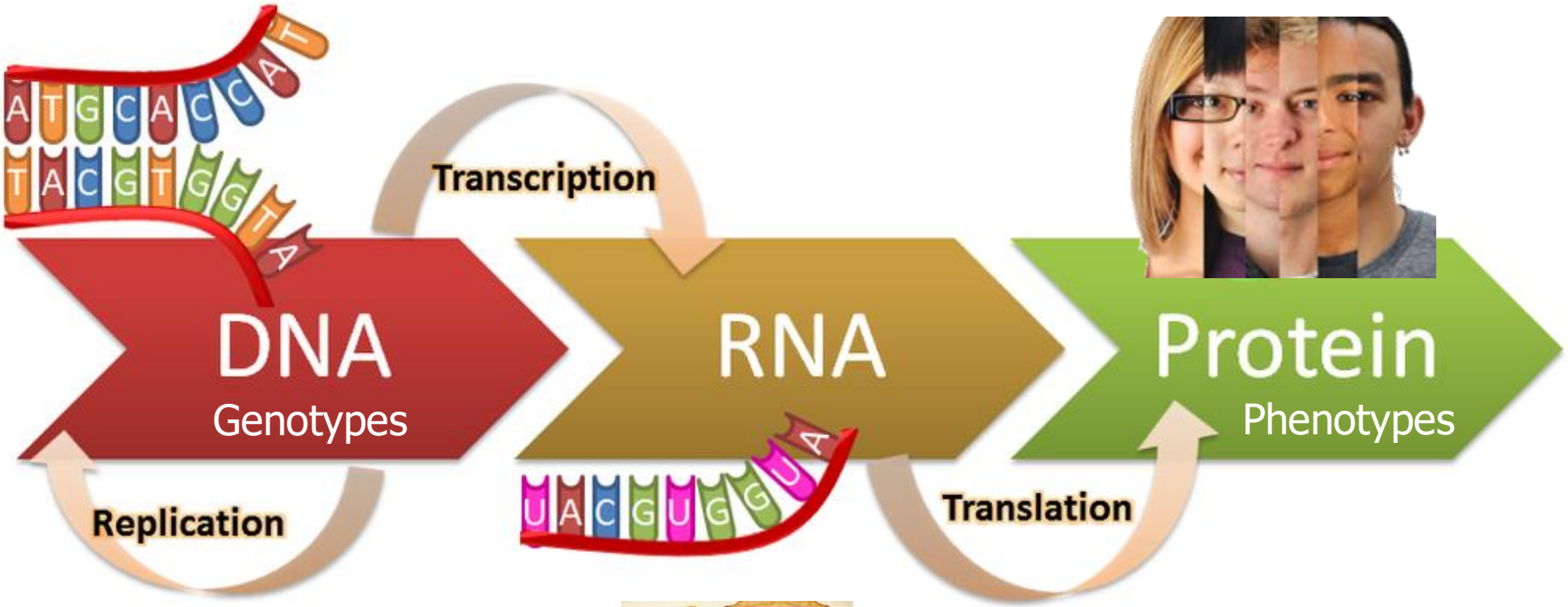
- Hardware Acceleration
  - M. Alser, H. Hassan, H. Xin, O. Ergin, O. Mutlu, C. Alkan, **"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"**, ***Bioinformatics***, 2017. [Source Code]
  - M. Alser, O. Mutlu, C. Alkan, **"MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering"**, ***IPSI Transactions on Internet Research***, 2017. [Source Code]
  - J. S. Kim, D. Senol Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, O. Mutlu, **"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"**, ***BMC Genomics***, 2018. Also appears in ***APBC***, 2018. [Source Code]
  - M. Alser, H. Hassan, A. Kumar, O. Mutlu, C. Alkan, **"Shouji: A Fast and Efficient Pre-Alignment Filter for Sequence Alignment"** ***Bioinformatics***, 2019. [Source Code]

BACKUP SLIDES

What Is a Genome Made Of?

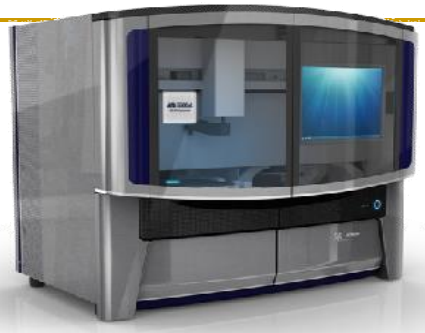


The Central Dogma of Molecular Biology



Genome Sequencers

Roche/454



AB SOLiD

Illumina MiSeq

Complete Genomics

Oxford Nanopore MinION

Illumina HiSeq2000

Pacific Biosciences RS

Oxford Nanopore GridION

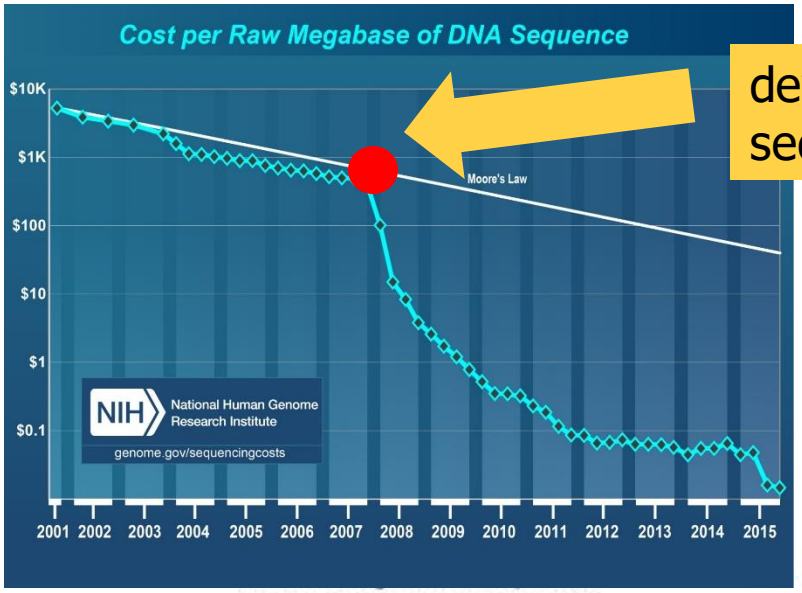
Illumina NovaSeq 6000

Ion Torrent PGM

Ion Torrent Proton

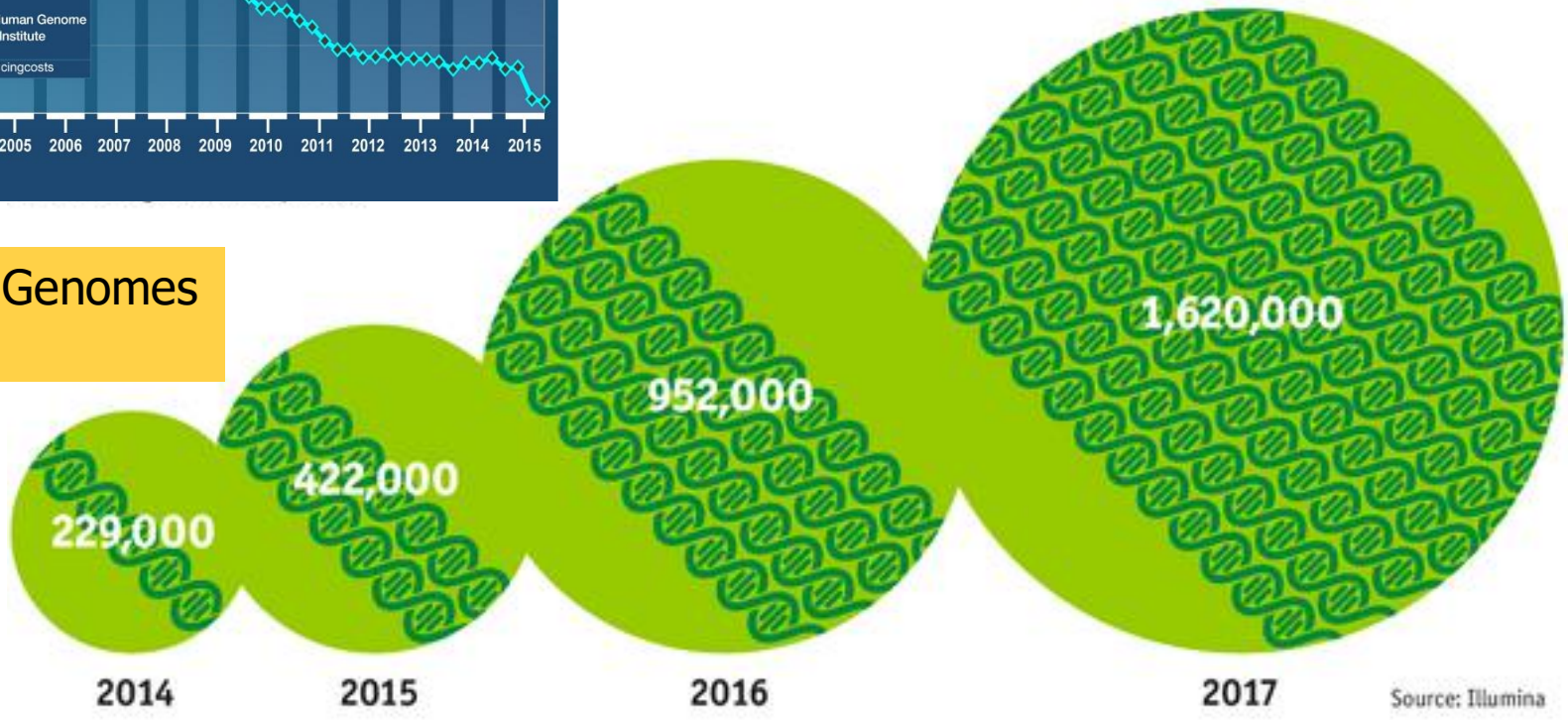
… and more! All produce data with different properties.

The Genomic Era



development of high-throughput sequencing (HTS) technologies

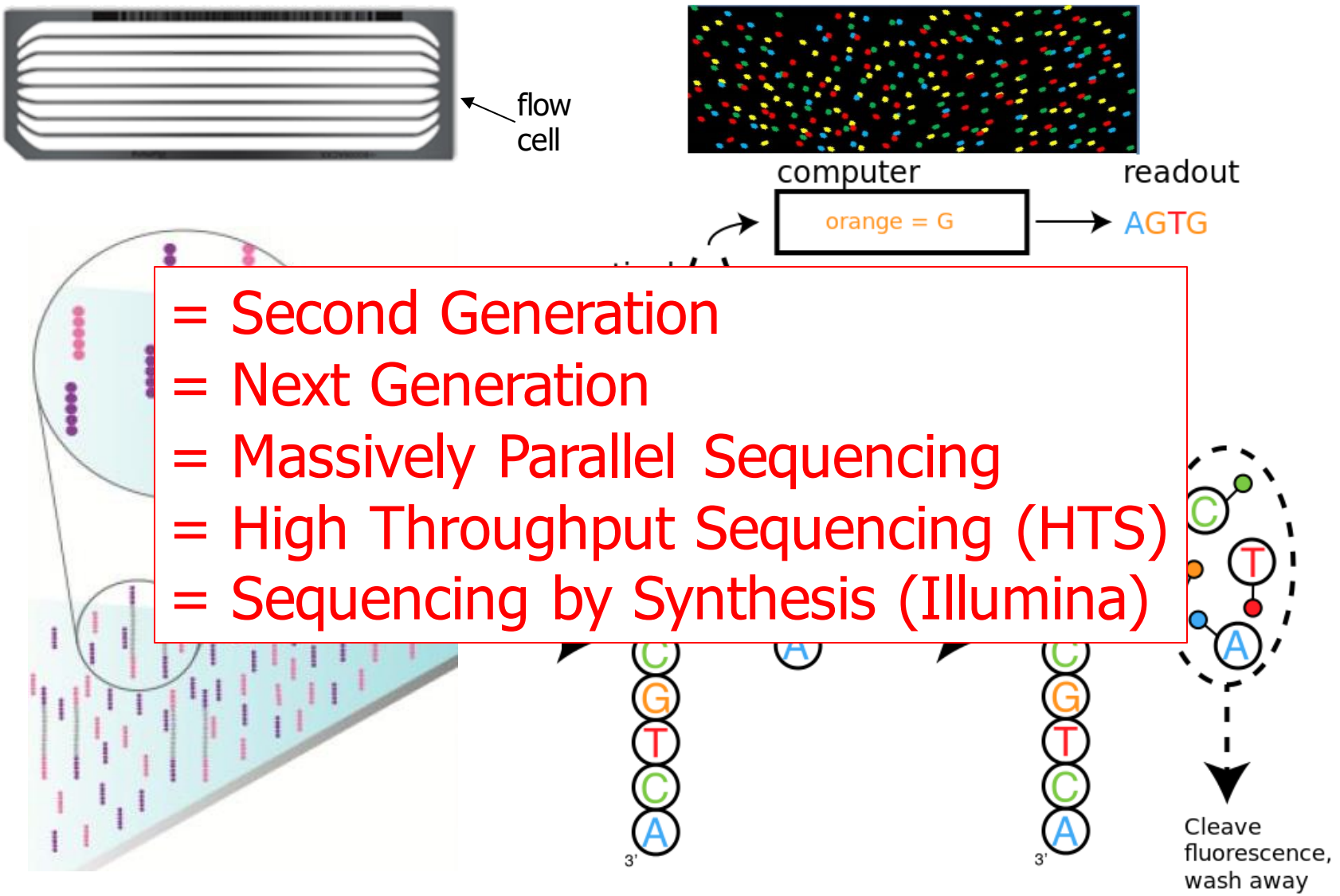
Number of Genomes Sequenced



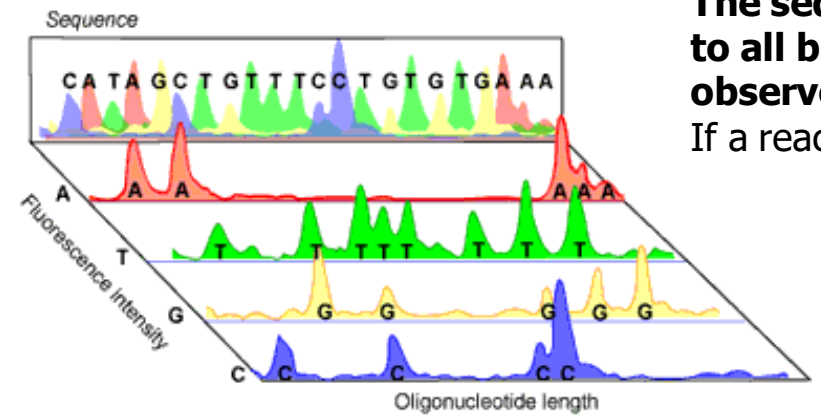
The Economist

http://www.economist.com/news/21631808-so-much-genetic-data-so-many-uses-genes-unzipped

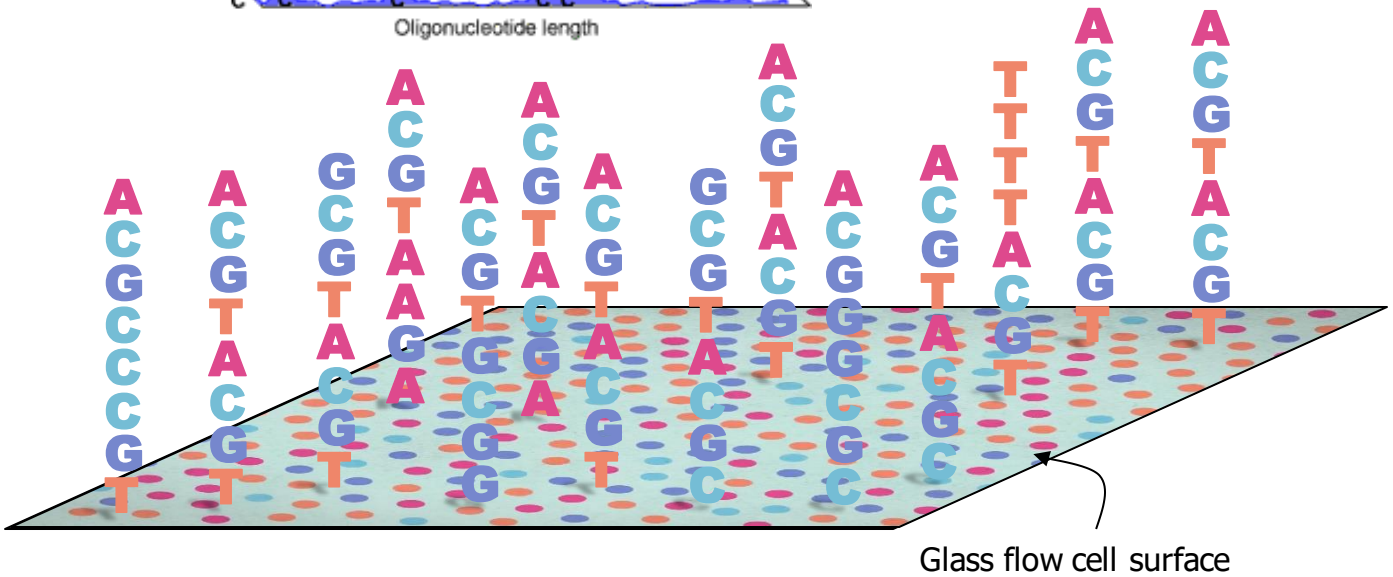
High-Throughput Sequencing (HTS)



High-Throughput Sequencing (HTS)



The sequencer adds the molecule "T" to all bases near the flow cell surface and observes the chemical reaction via a CMOS sensor.
If a reaction happens then the base is "A"



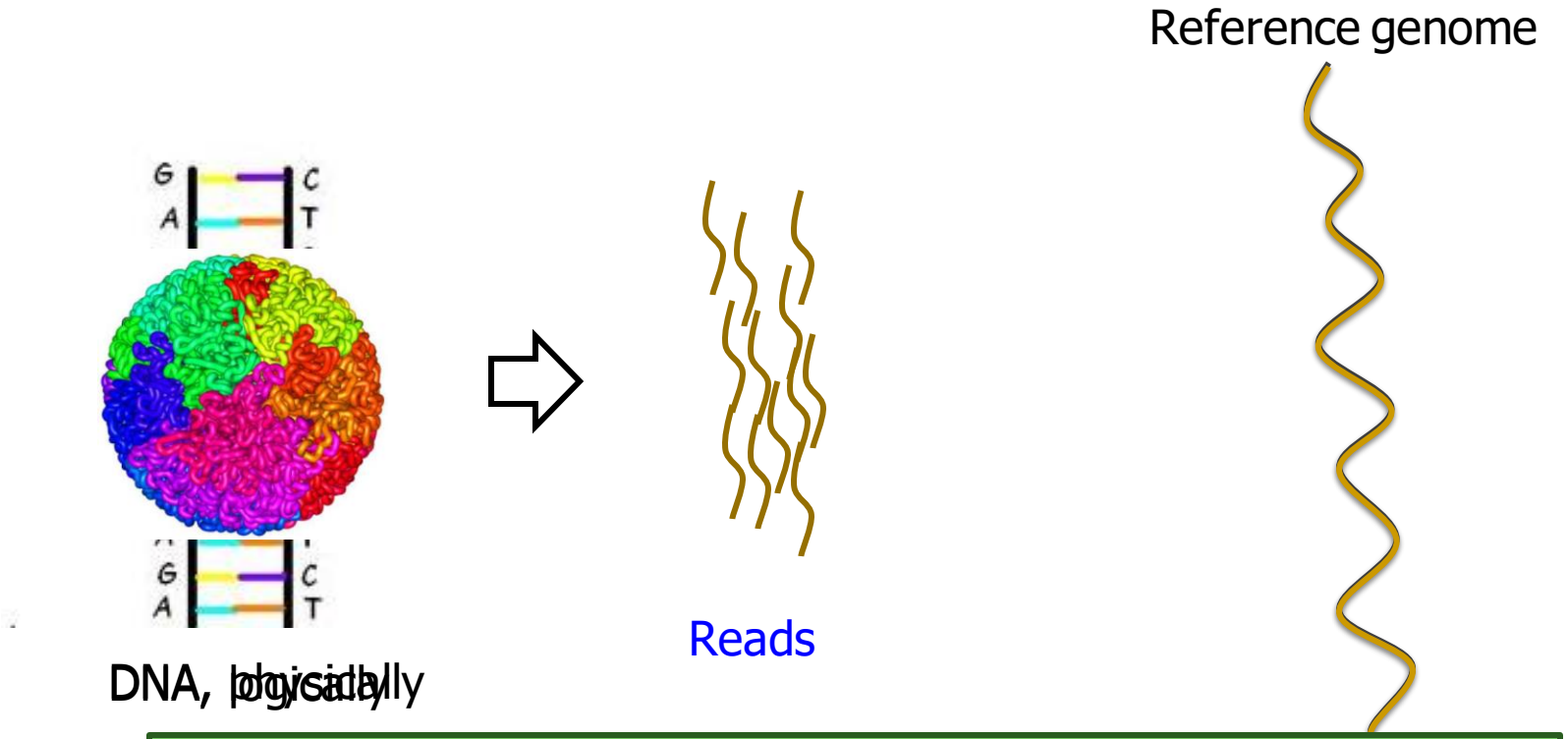
As a workaround, HTS technologies sequence random short DNA fragments (75-300 basepairs long) of copies of the original molecule.

High-Throughput Sequencing

- Massively parallel sequencing technology
  - Illumina, Roche 454, Ion Torrent, SOLID…
- Small DNA fragments are first amplified and then sequenced in parallel, leading to
  - High throughput
  - High speed
  - Low cost
  - Short reads
- Sequencing is done by either reading optical signals as each base is added, or by detecting hydrogen ions instead of light, leading to:
  - Low error rates (relatively)
  - Reads lack information about their order and which part of genome they are originated from

Read Mapping

- Map many short DNA fragments (reads) to a known reference genome with some differences allowed



Mapping short reads to reference genome is challenging (billions of 50-300 base pair reads)

Read Alignment/Verification

- **Edit distance** is defined as the minimum number of edits (i.e. insertions, deletions, or substitutions) needed to make the read exactly match the reference segment.

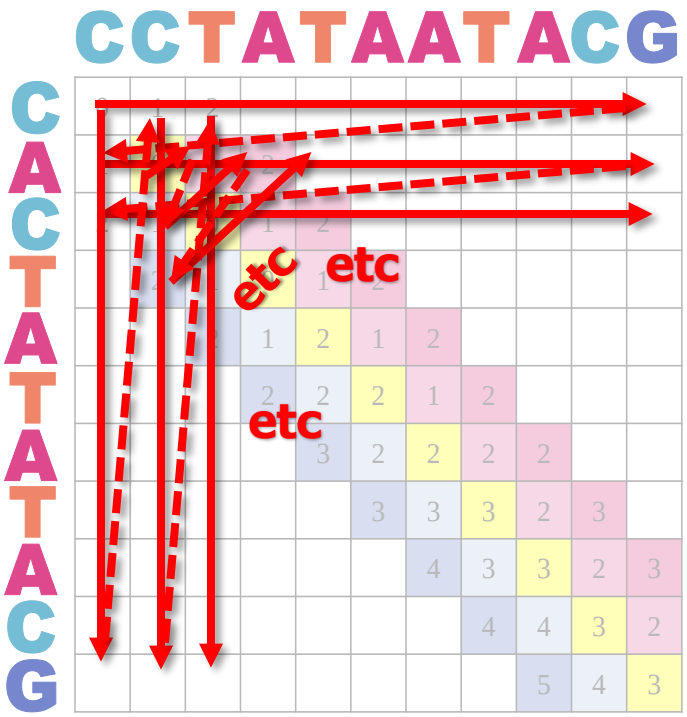
NETHERLANDS x SWITZERLAND

N	E	-	T	H	E	R	L	A	N	D	S
S	W	I	T	Z	E	R	L	A	N	D	-

match
deletion
insertion
mismatch

Why Is Read Alignment Slow?

- Quadratic-time dynamic-programming algorithm(s)
- Data dependencies limit the computation parallelism
- Entire matrix computed even though strings may be dissimilar.



Read Alignment

Read Mapping Algorithms: Two Styles

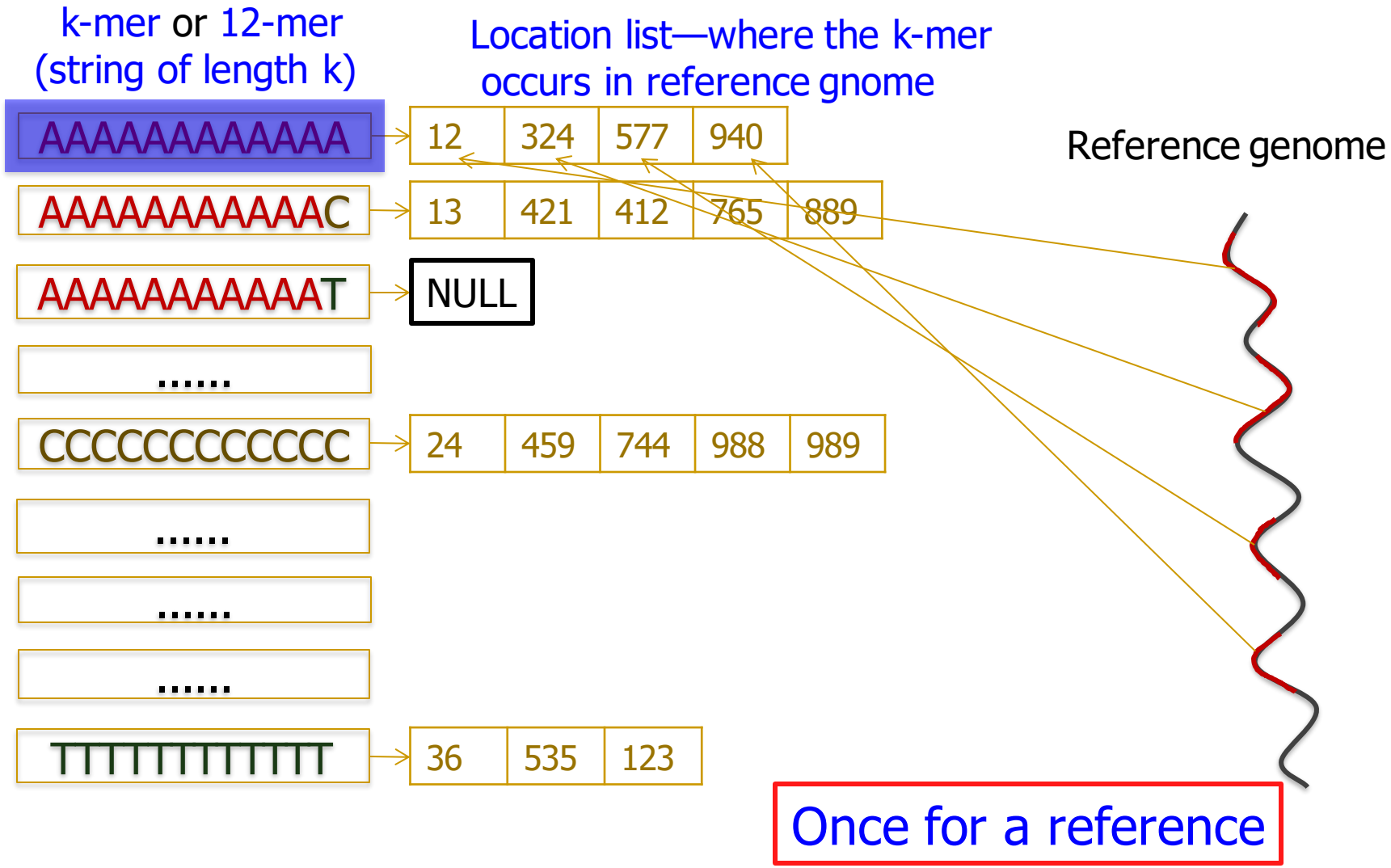
- Hash based seed-and-extend (hash table, suffix array, suffix tree)
  - Index the "k-mers" in the genome into a hash table (pre-processing)
  - When searching a read, find the location of a k-mer in the read; then extend through alignment
  - More sensitive (can find all mapping locations), but slow
  - Requires large memory; this can be reduced with cost to run time

- Burrows-Wheeler Transform & Ferragina-Manzini Index based aligners
  - BWT is a compression method used to compress the genome index
  - Perfect matches can be found very quickly, memory lookup costs increase for imperfect matches
  - Reduced sensitivity

Hash Table Based Read Mappers

- Key Idea
  - Preprocess the reference into a *Hash Table*
  - Use *Hash Table* to map reads

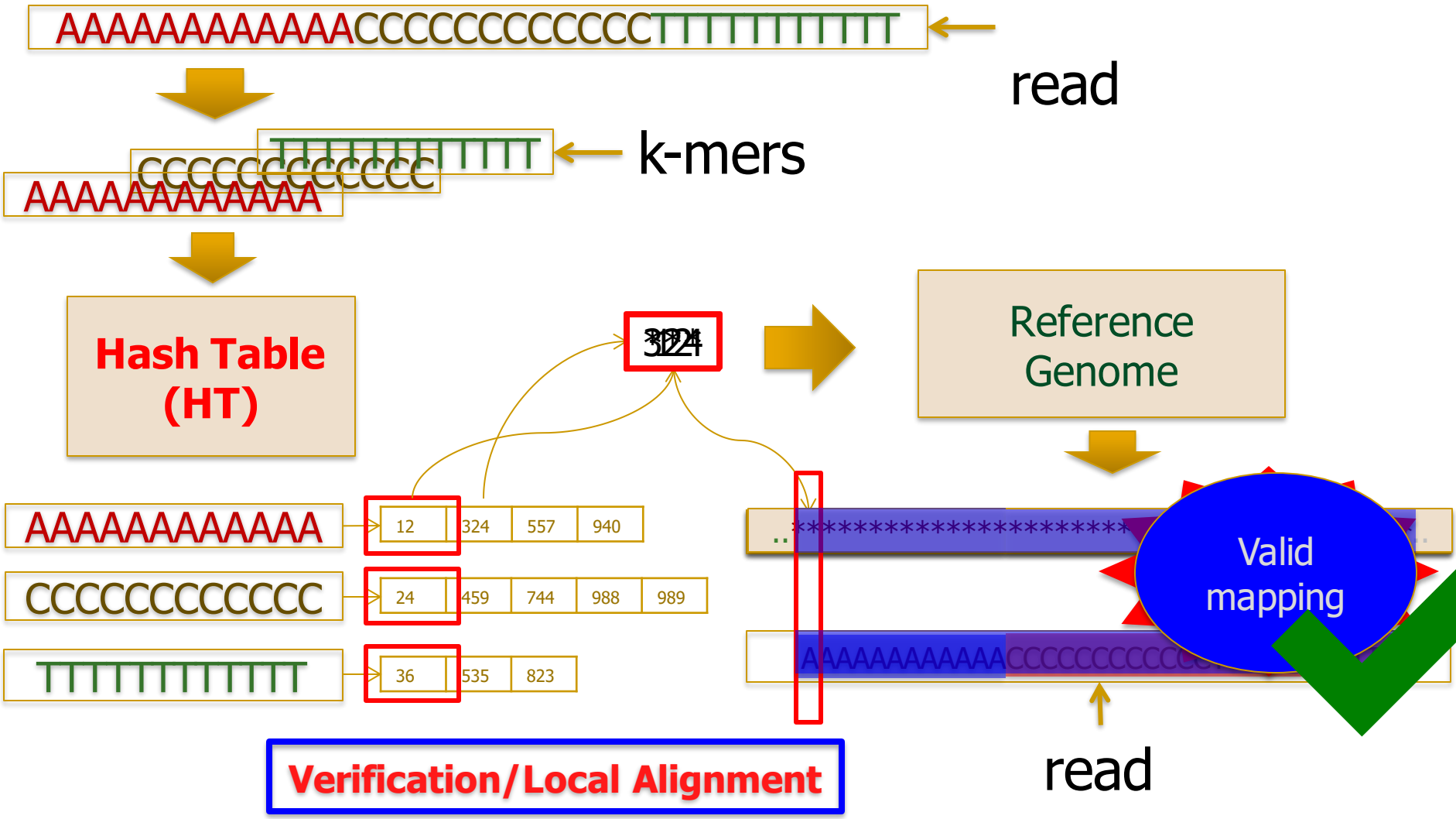
Hash Table-Based Mappers [Alkan+ Nature Gen'09]



Hash Table Based Read Mappers

- Key Idea
  - Preprocess the reference into a *Hash Table*
  - Use *Hash Table* to map reads

Hash Table-Based Mappers [Alkan+ Nature Gen'09]



Advantages of Hash Table Based Mappers

- + Guaranteed to find *all* mappings → very sensitive
- + Can tolerate up to *e* errors

nature genetics

http://mrfast.sourceforge.net/

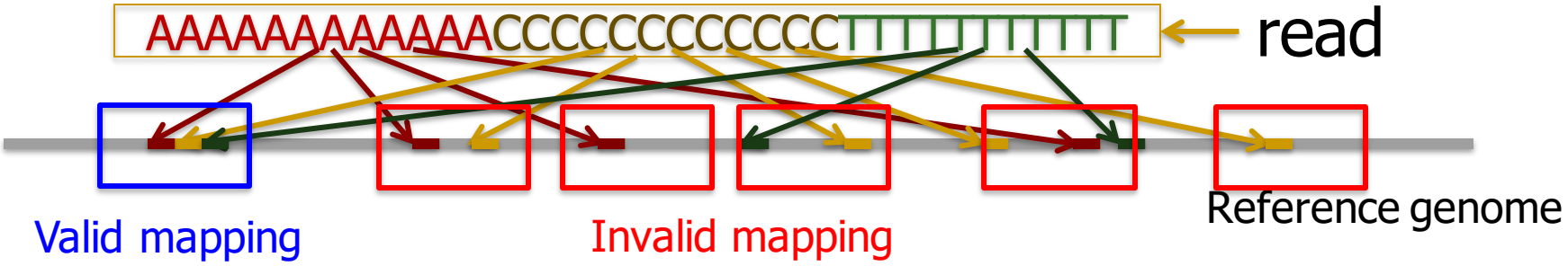
Personalized copy number and segmental duplication maps using next-generation sequencing

Can Alkan[1,2], Jeffrey M Kidd[1], Tomas Marques-Bonet[1,3], Gozde Aksay[1], Francesca Antonacci[1], Fereydoun Hormozdiari[4], Jacob O Kitzman[1], Carl Baker[1], Maika Malig[1], Onur Mutlu[5], S Cenk Sahinalp[4], Richard A Gibbs[6] & Evan E Eichler[1,2]

Alkan+, **"Personalized copy number and segmental duplication maps using next-generation sequencing"**, Nature Genetics 2009.

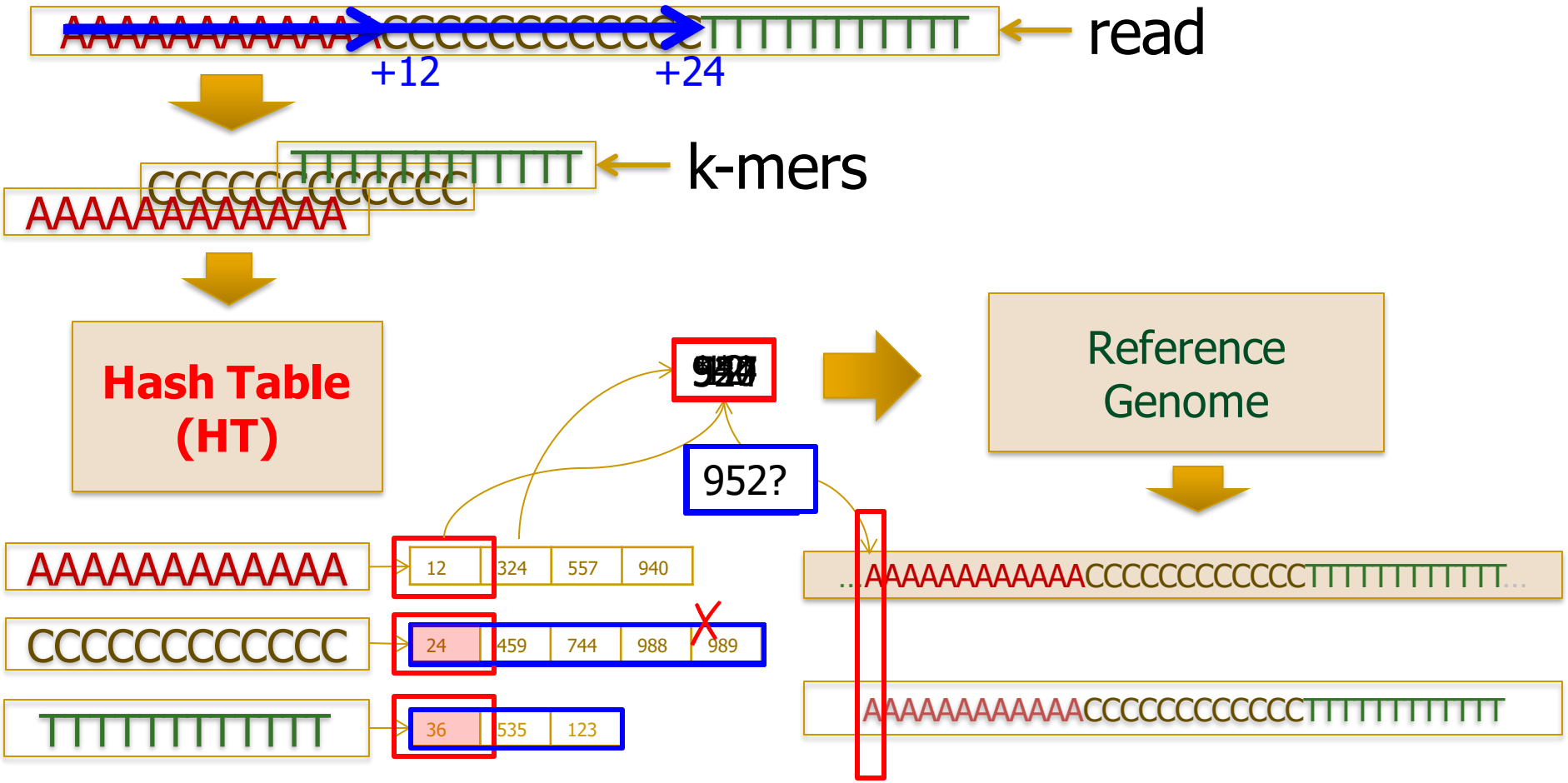
Adjacency Filtering (AF)

- **Goal:** detect and filter out invalid mappings at early stage
- **Key Insight:** For a valid mapping, adjacent k-mers in the read are also adjacent in the reference genome



- **Key Idea:** search for adjacent locations in the k-mers' location lists
  - If more than *e* k-mers fail → there must be more than e errors → invalid mapping

Adjacency Filtering (AF)



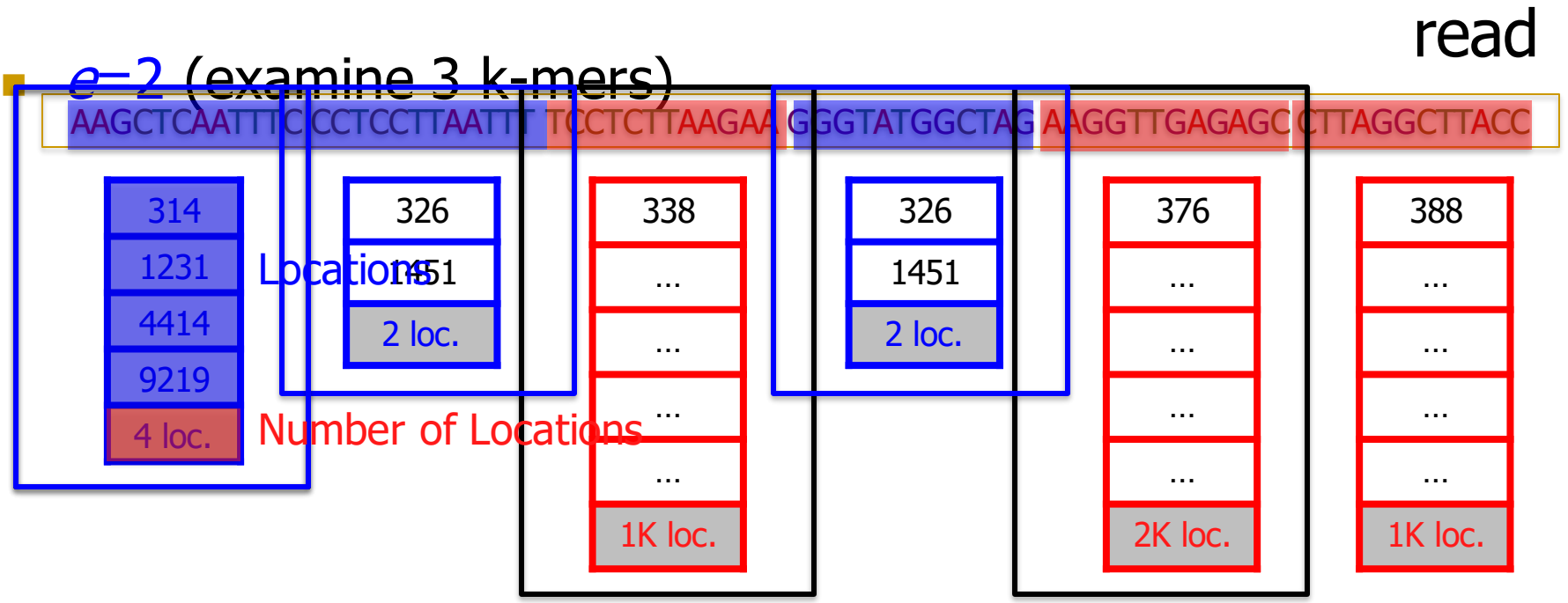
Cheap K-mer Selection (CKS)

- **Goal:** Reduce the number of potential mappings to examine

- **Key insight:**
  - K-mers have different cost to examine: Some k-mers are *cheaper* as they have fewer locations than others (occur less frequently in reference genome)

- **Key idea:**
  - Sort the k-mers based on their number of locations
  - Select the k-mers with the fewest number locations to verify

Cheap K-mer Selection



Expensive 3 k-mers

Previous work needs to verify:

3004 locations

FastHASH verifies only:

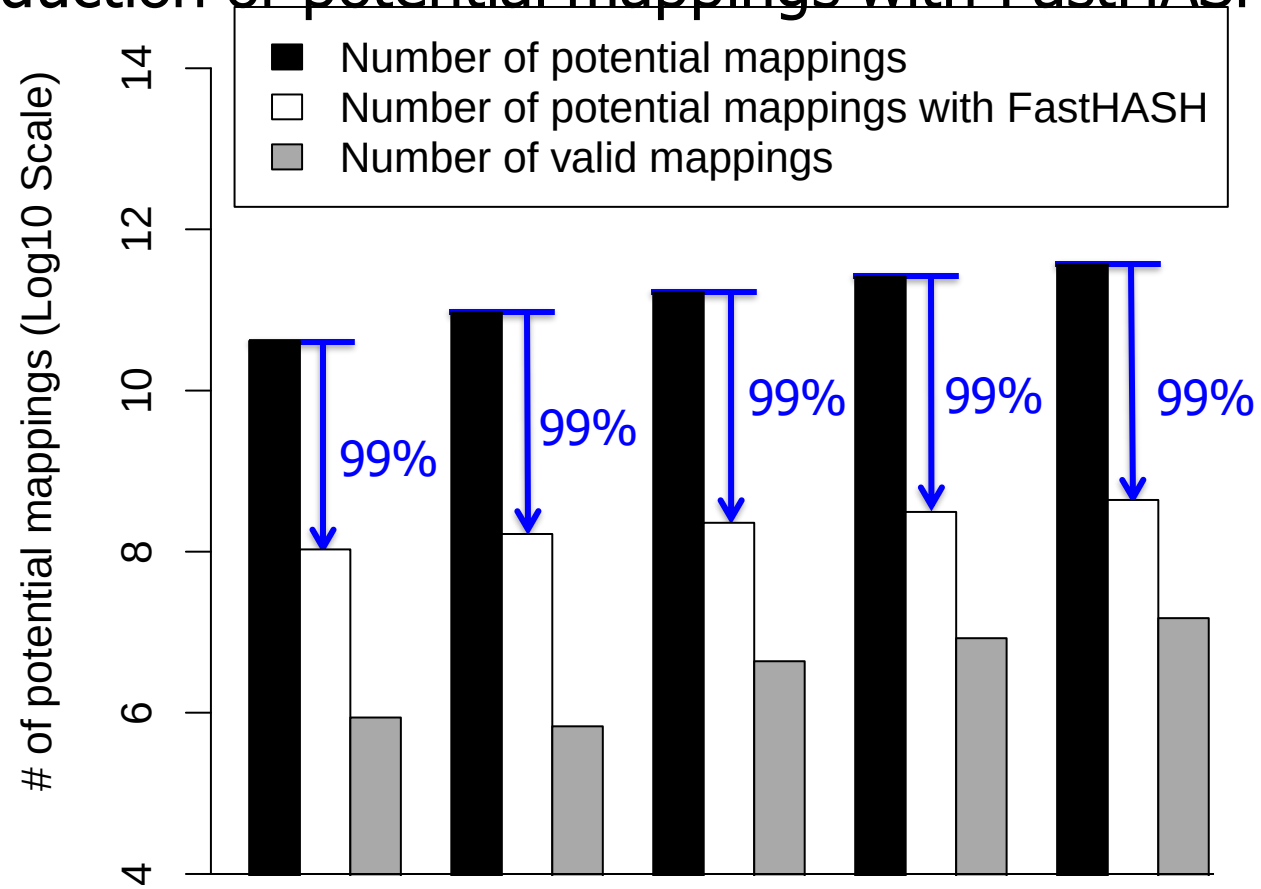
8 locations

Methodology

- Implemented FastHASH on top of state-of-the-art mapper: mrFAST
  - New version mrFAST-2.5.0.0 over mrFAST-2.1.0.6
- Tested with real read sets generated from Illumina platform
  - 1M reads of a human (160 base pairs)
  - 500K reads of a chimpanzee (101 base pairs)
  - 500K reads of a orangutan (70 base pairs)
- Tested with simulated reads generated from reference genome
  - 1M simulated reads of human (180 base pairs)
- Evaluation system
  - Intel Core i7 Sandy Bridge machine
  - 16 GB of main memory

Analysis

- Reduction of potential mappings with FastHASH



FastHASH filters out over 99% of the potential mappings without sacrificing any valid mappings

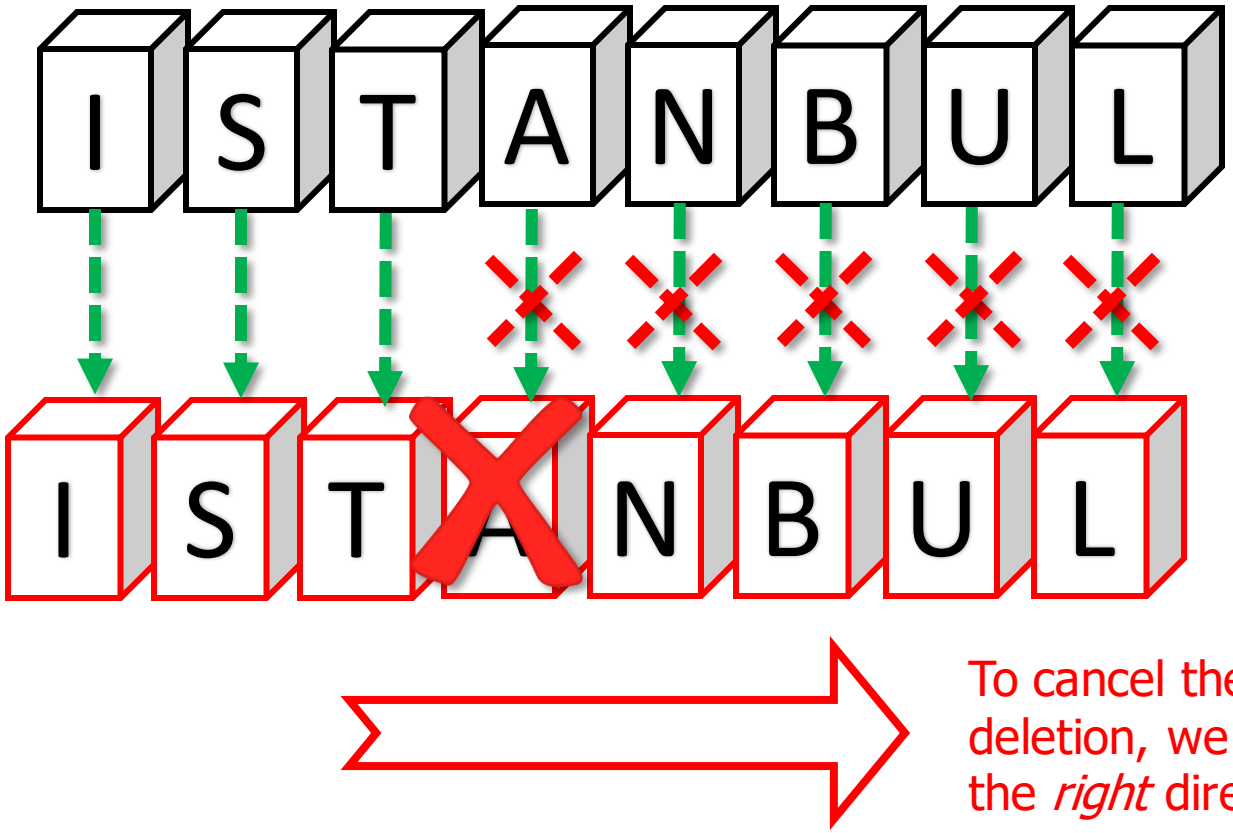
FastHASH Conclusion

- Problem: Existing read mappers perform poorly in mapping millions of short reads to the reference genome, in the presence of errors
- Observation: Most of the verification calculations are unnecessary → filter them out
- Key Idea: Exploit the structure of the genome to
  - Reject invalid mappings early (Adjacency Filtering)
  - Reduce the number of possible mappings to examine (Cheap K-mer Selection)
- Key Result: FastHASH obtains up to 19x speedup over the state-of-the-art mapper without losing valid mappings

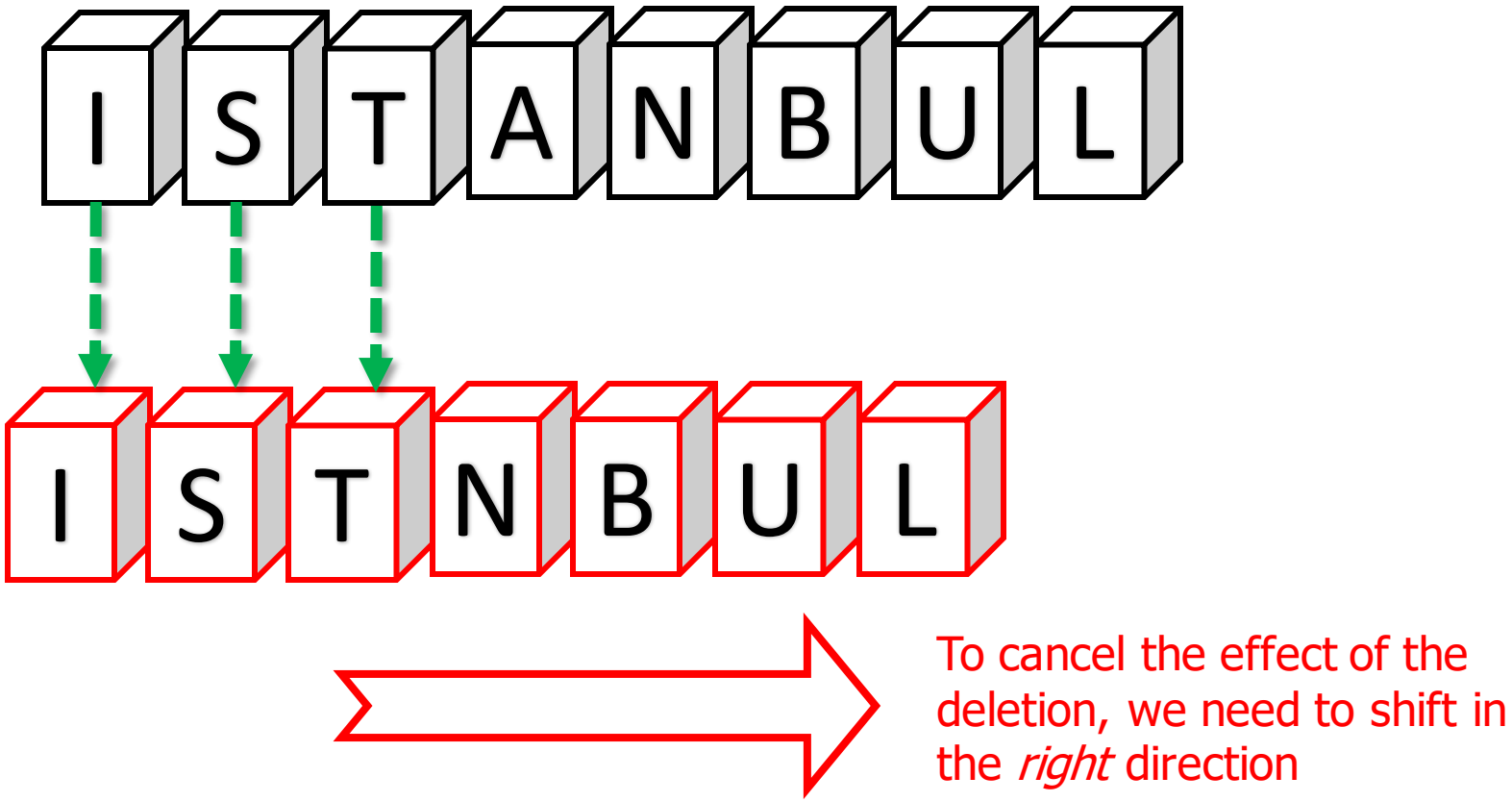
Hamming Distance (∑⊕)

3 matches 5 mismatches

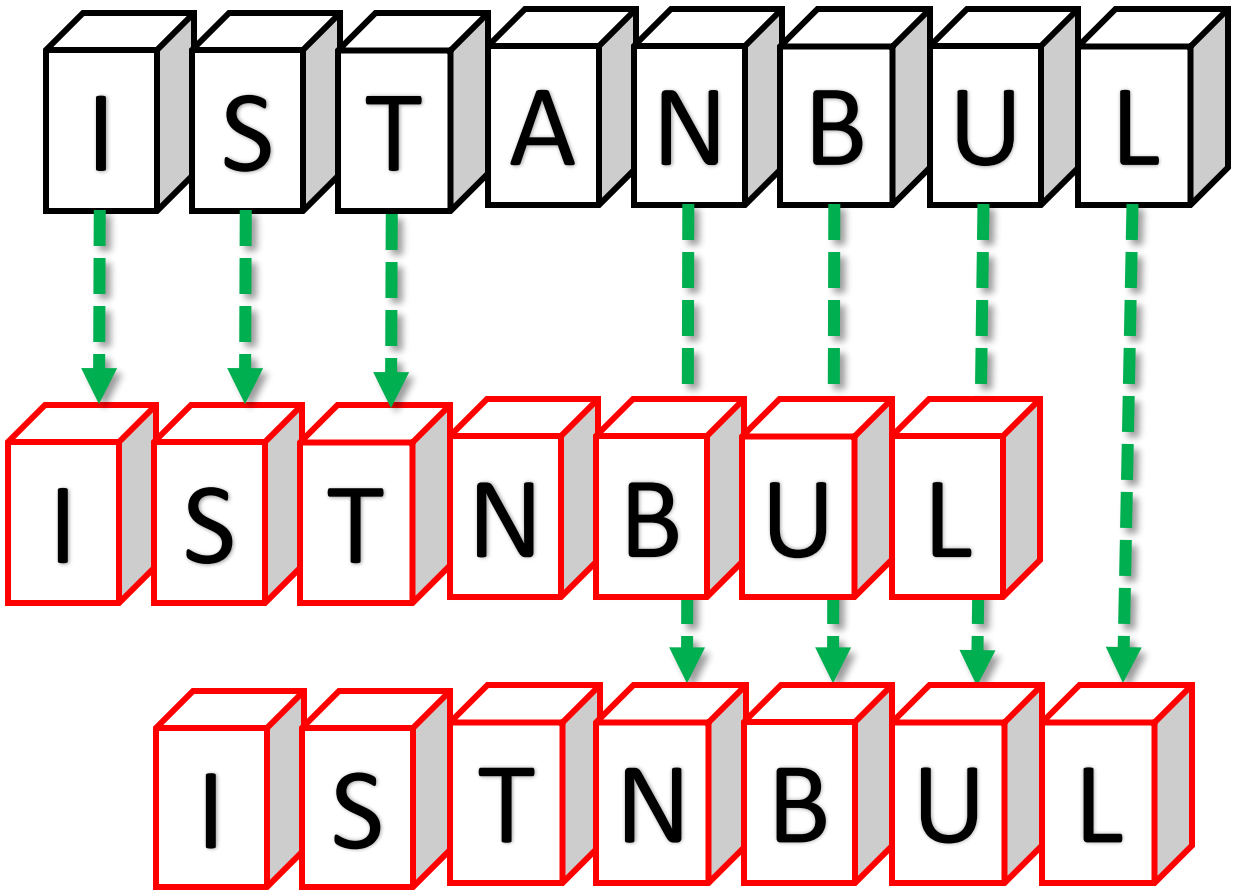
Edit = 1 Deletion



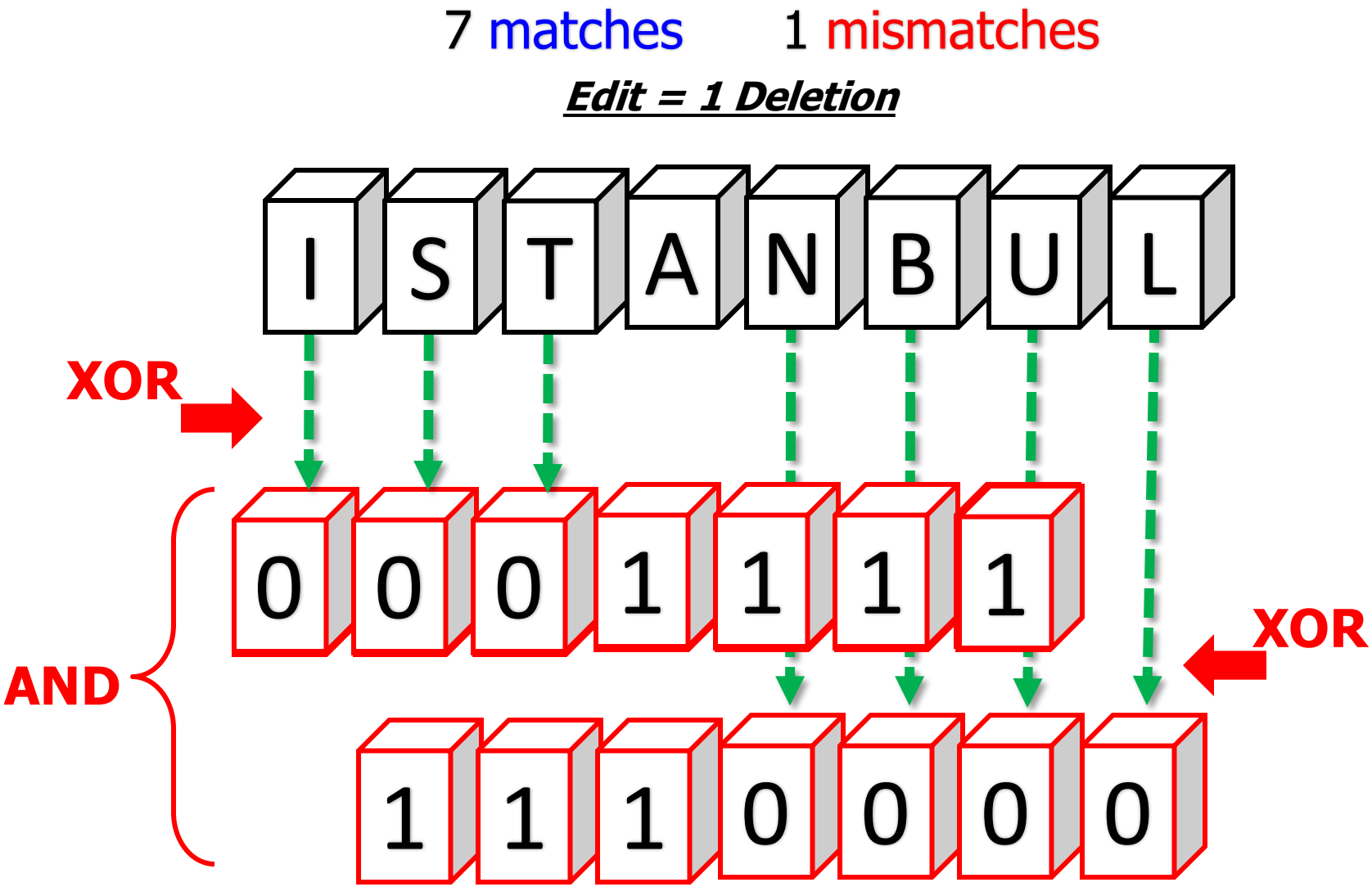
Insight: Shifting a String Helps Similarity Search



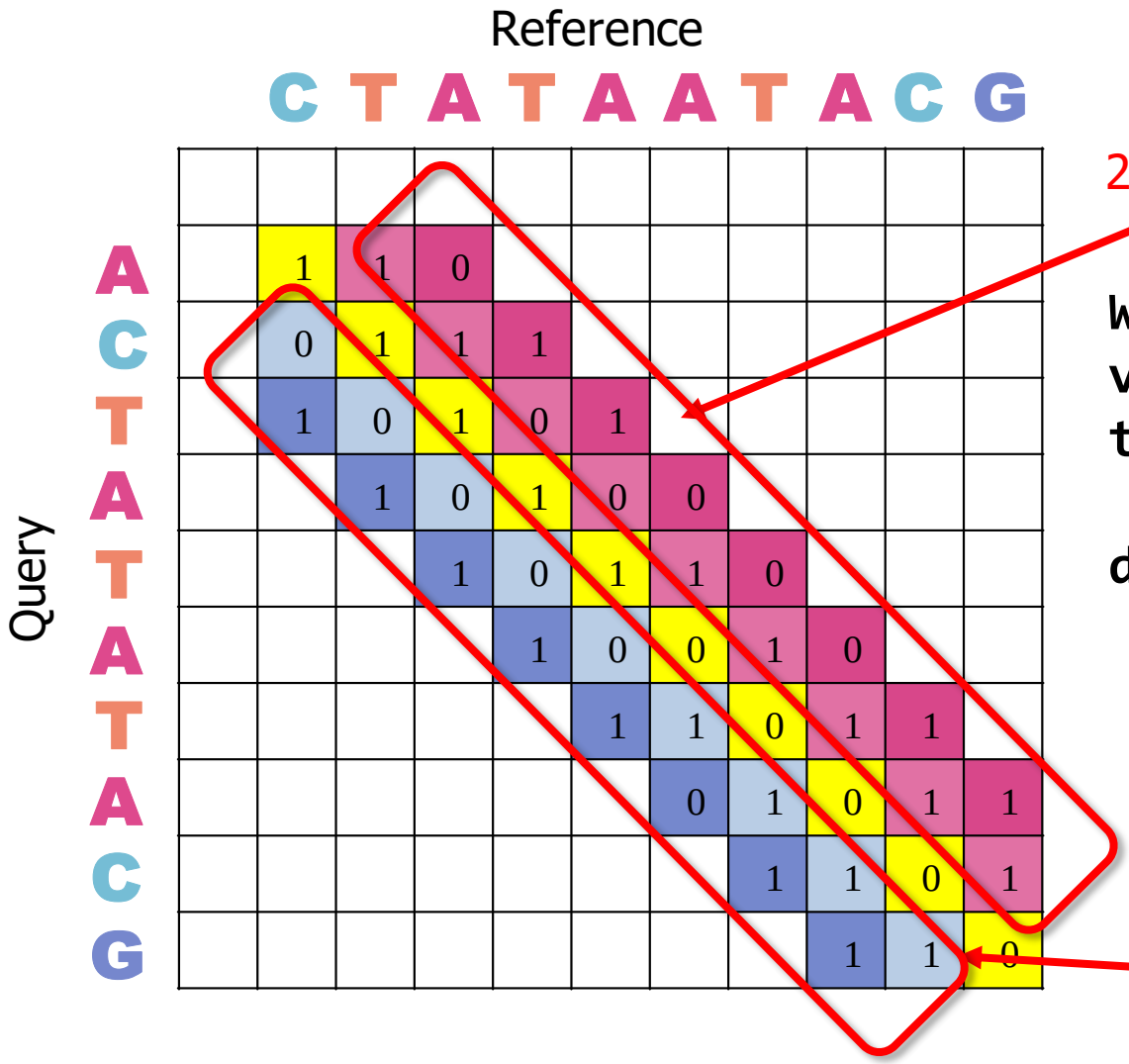
Insight: Shifting a String Helps Similarity Search



Shifted Hamming Distance

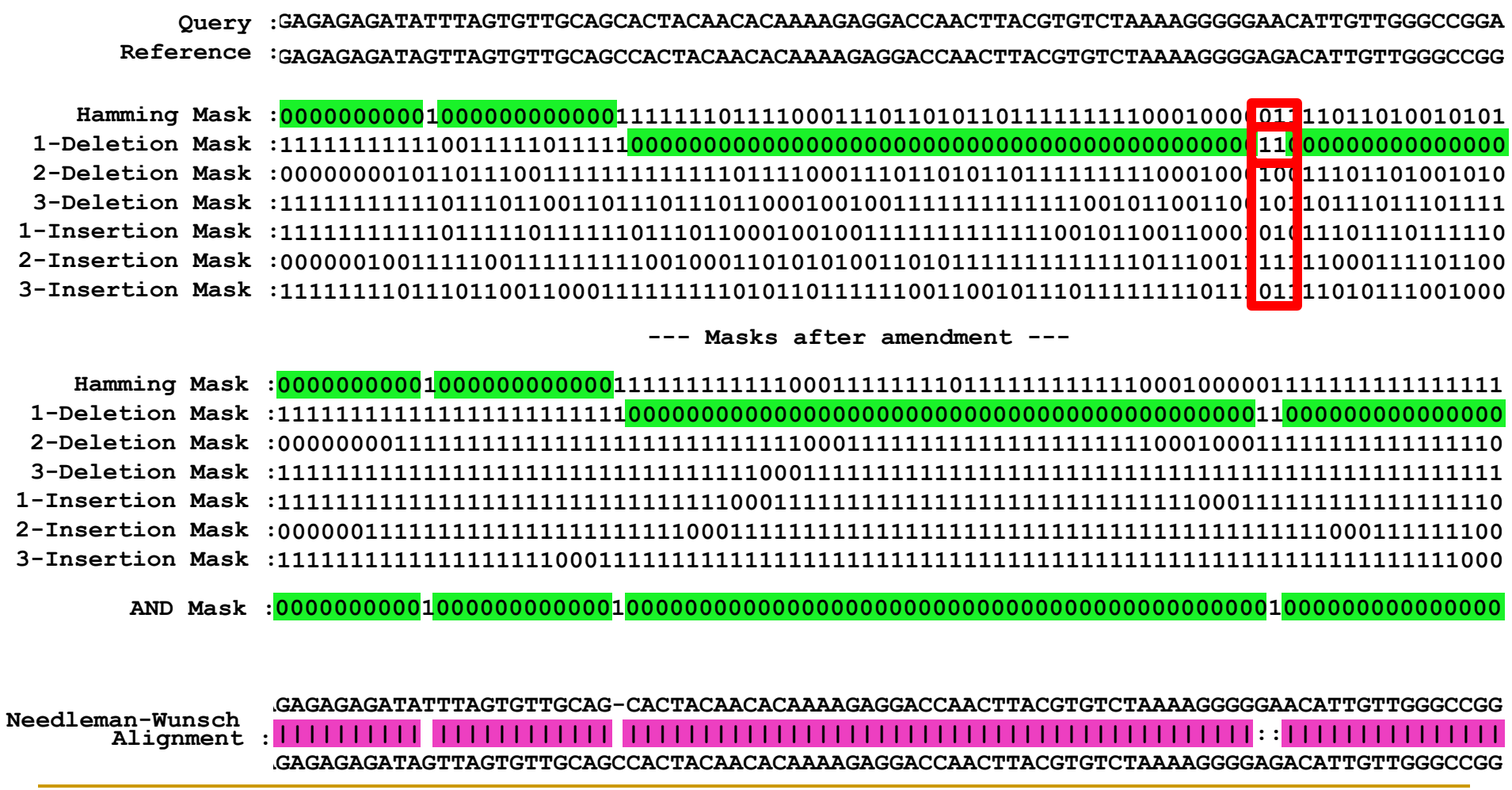


Highly Parallel Matrix Computation



Key Idea of SHD Filtering

Generate 2E+1 masks → Amend random zeros: 101 → 111 & 1001 → 1111 → AND all masks, ACCEPT iff number of '1' ≤ Threshold



Alignment vs. Pre-alignment (Filtering)

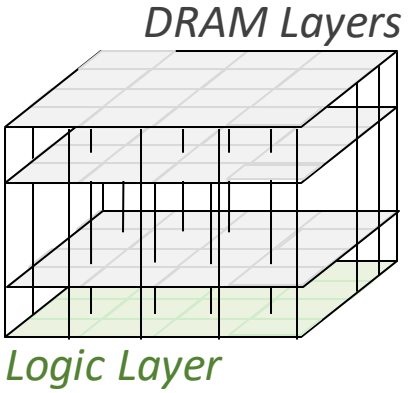
Needleman-Wunsch

		C	T	A	T	A	A	T	A	C	G
	0	1	2								
A	1	0	1	2							
C	2	1	0	1	2						
T		2	1	0	1	2					
A			2	1	2	1	2				
T				2	2	2	1	2			
A					3	2	2	2	2		

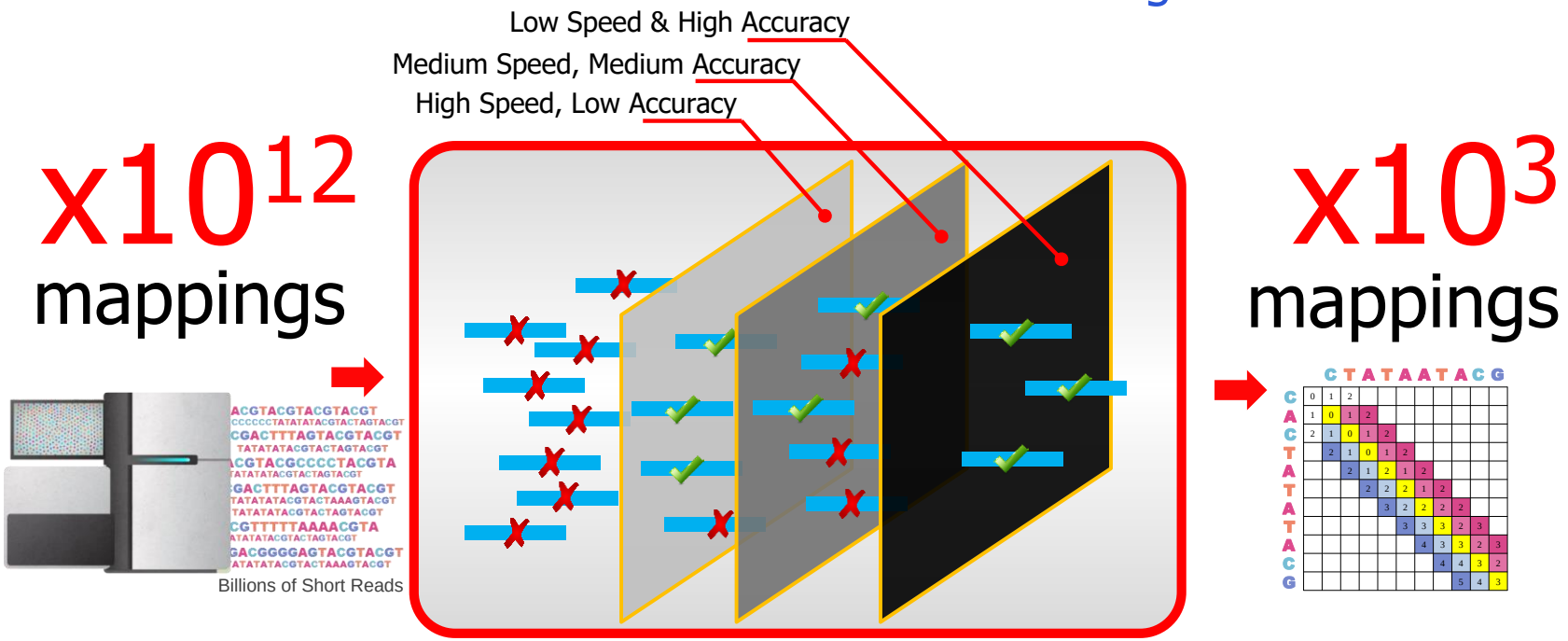
SHD

		C	T	A	T	A	A	T	A	C	G
A		1	1	0							
C		0	1	1	1						
T		1	0	1	0	1					
A			1	0	1	0	0				
T				1	0	1	1	0			
A					1	0	0	1	0		

Independent vectors can be processed in parallel using hardware technologies



GateKeeper: Using FPGAs for Filtering



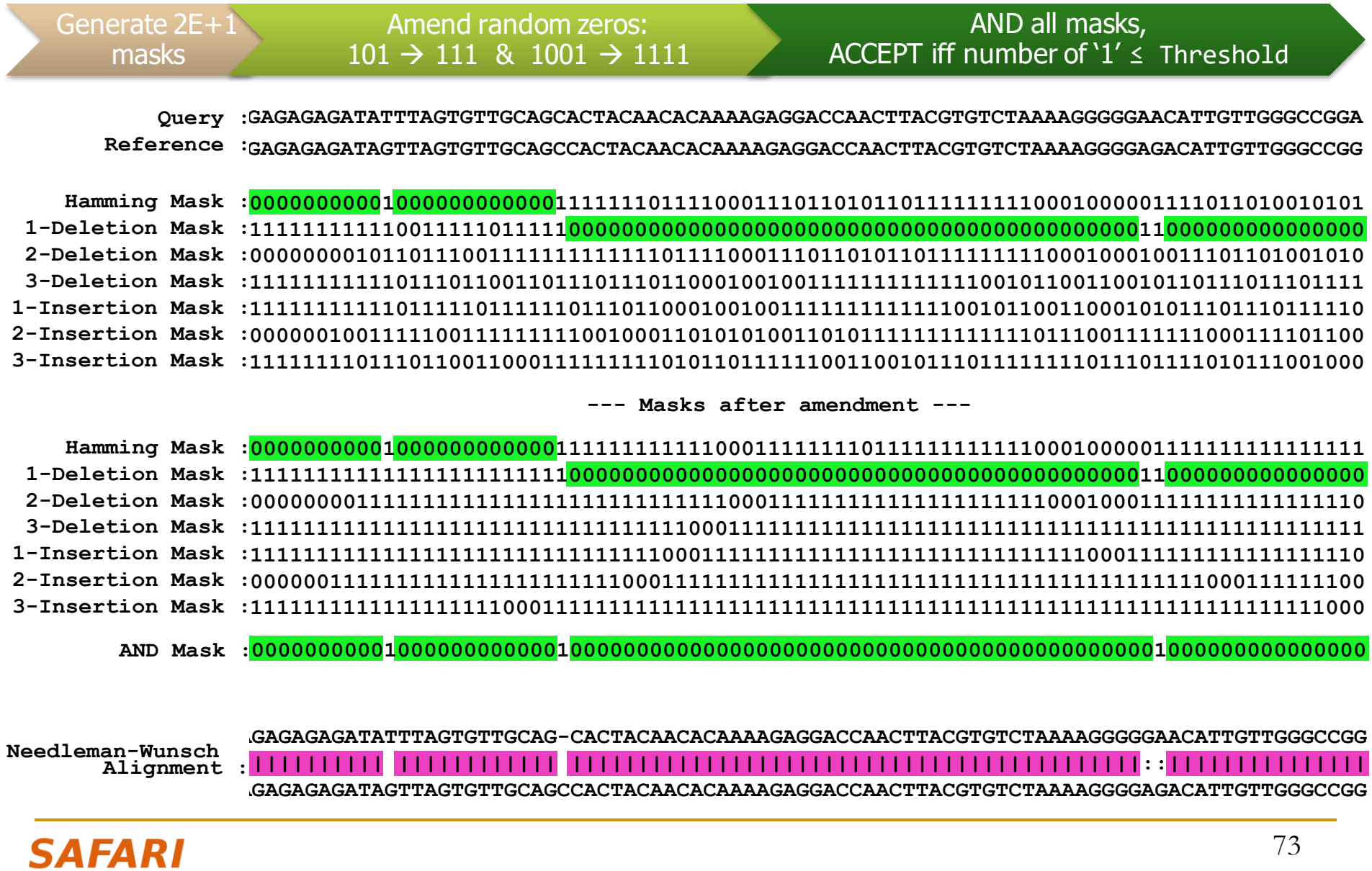
1 **High throughput DNA sequencing (HTS) technologies**

2 **Read Pre-Alignment Filtering**
Fast & Low False Positive Rate

3 **Read Alignment**
Slow & Zero False Positives

GateKeeper Walkthrough

Generate 2E+1 masks → Amend random zeros: 101 → 111 & 1001 → 1111 → AND all masks, ACCEPT iff number of '1' ≤ Threshold

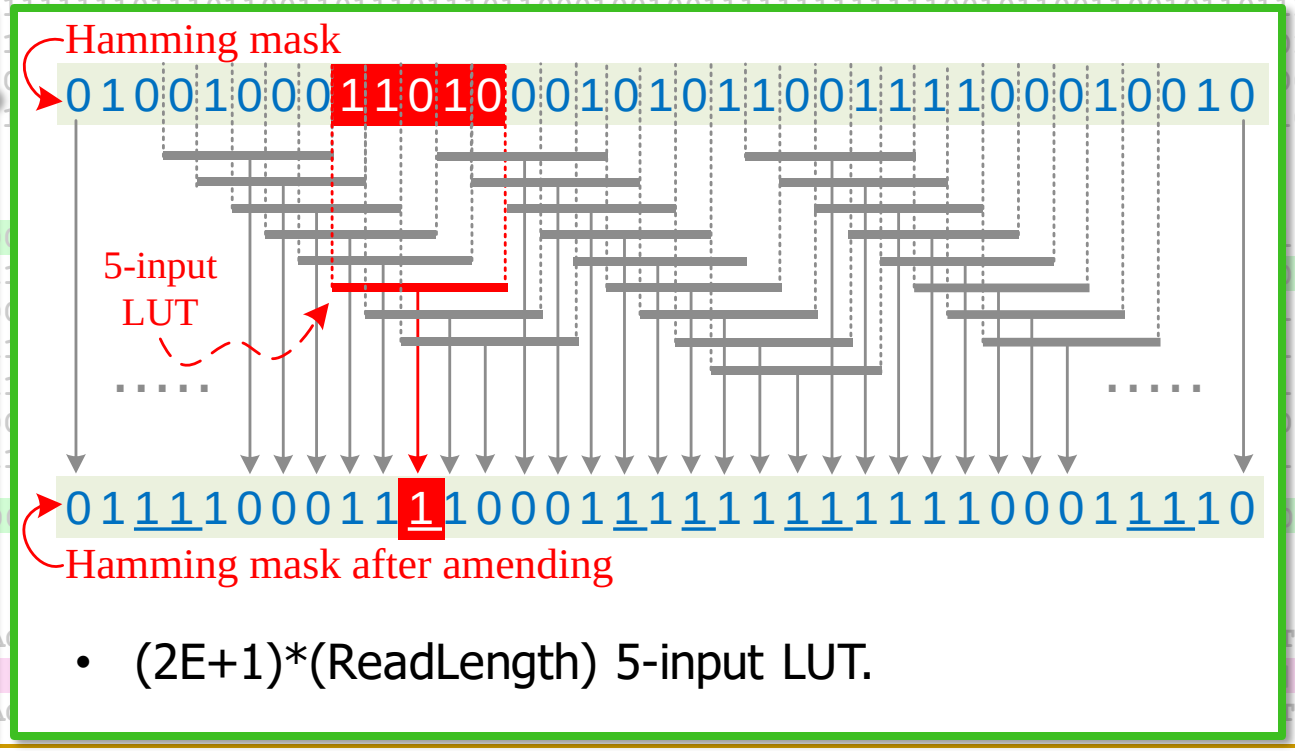


GateKeeper Walkthrough (cont'd)

Generate 2E+1 masks → Amend random zeros: 101 → 111 & 1001 → 1111 → AND all masks, ACCEPT iff number of '1' ≤ Threshold

- E right-shift registers (length=ReadLength)
- E left-shift registers (length=ReadLength)
- (2E+1) * (ReadLength) 2-XOR operations.

- (2E)*(ReadLength) 2-AND operations.
- (ReadLength/4) 5-input LUT.
- log_2ReadLength-bit counter.



- (2E+1)*(ReadLength) 5-input LUT.

GateKeeper vs. SHD

GateKeeper	SHD
FPGA (Xilinx VC709)	Intel SIMD
Multi-core (parallel)	Single-core (sequential)
Examines a single mapping @ 125 MHz	Examines a single mapping @ ~2MHz
Limited to PCIe Gen3(4x) transfer rate (128 bits @ 250MHz)	Limited to a read length of 128 bp (SSE register size)
Amending requires:	Amending requires:
(2E+1) 5-input LUT.	4(2E+1) bitwise OR.
	4(2E+1) packed shuffle.
	3(2E+1) shift.

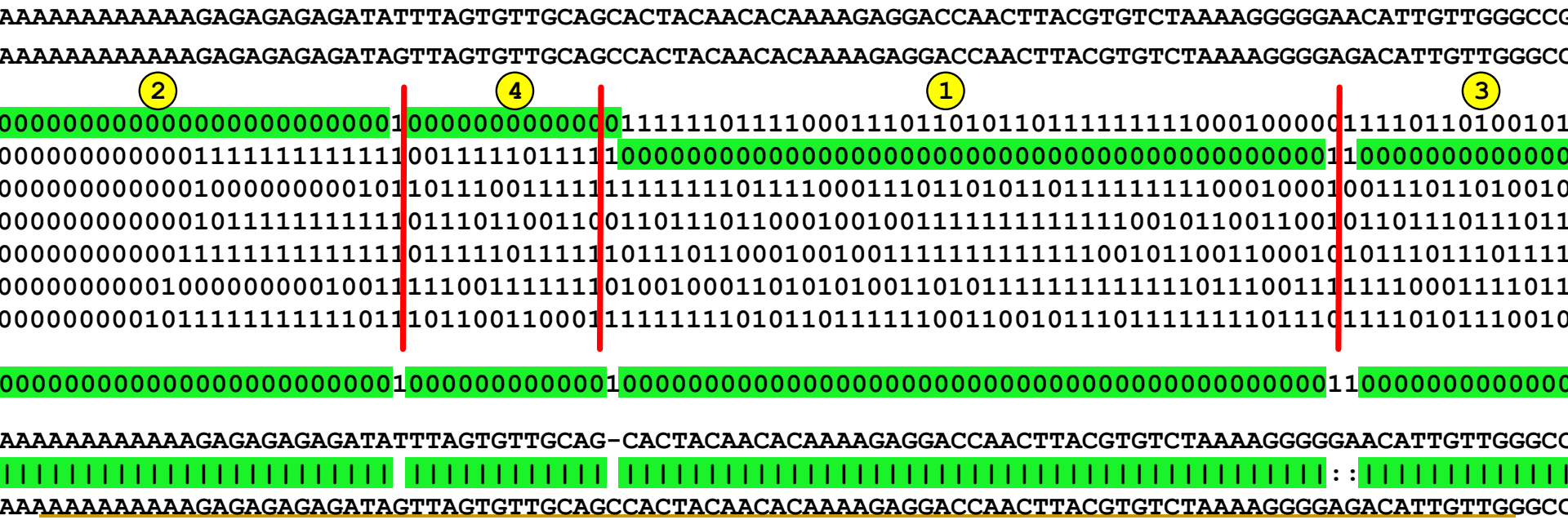
Conclusions

- FPGA-based pre-alignment greatly speeds up read mapping
  - 10x speedup of a state-of-the-art mapper (mrFAST)

- FPGA-based pre-alignment can be integrated with the sequencer
  - It can help to hide the complexity and details of the FPGA
  - Enables real-time filtering while sequencing

MAGNET (AACBB 2018, TIR 2017)

- Key observation: the use of **AND operation** to check if a zero (match) exists in a column introduces filtering inaccuracy.
- Key Idea: count the **consecutive zeros** in each mask and select the longest in a divide-and-conquer approach.
- **MAGNET** is **17x to 105x more accurate** than GateKeeper and SHD.



MAGNET Walkthrough

Build Neighborhood Map → Track the Diagonally Consecutive Matches → ACCEPT iff number of '1' ≤ Threshold

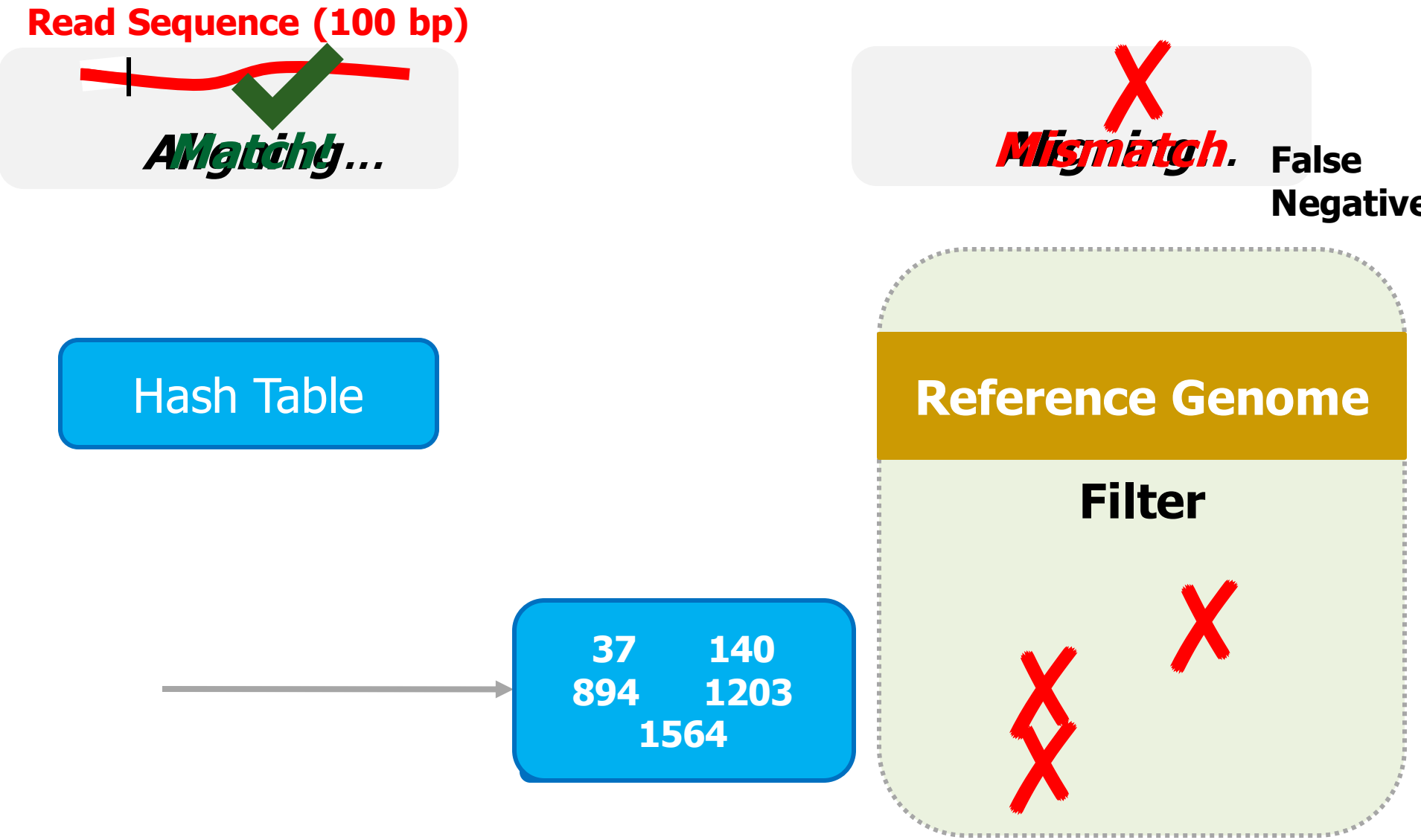
Find the longest segment of consecutive zeros

Exclude the errors from the search space

Divide the problem into two subproblems and repeat

"MAGNET: understanding and improving the accuracy of genome pre-alignment filtering", *arXiv preprint* 2017

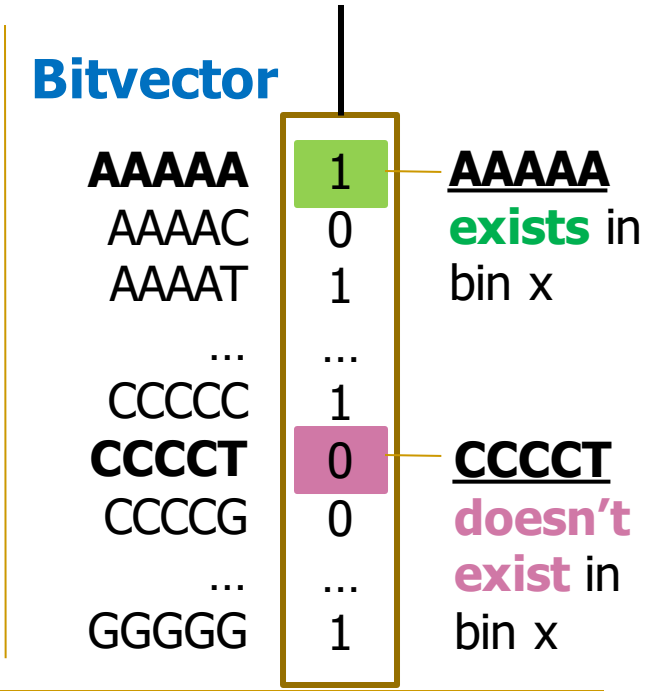
Hash Tables in Read Mapping



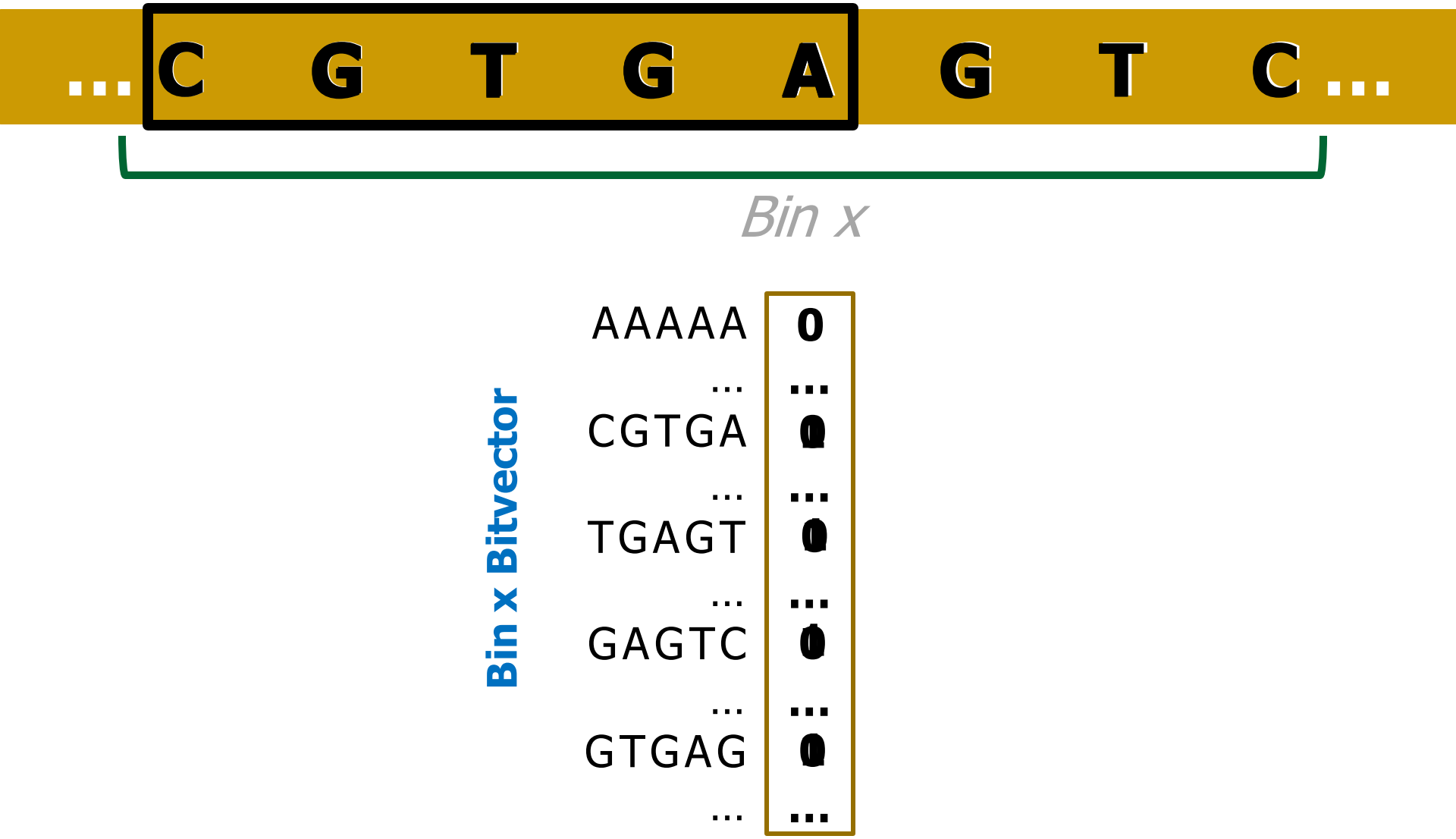
GRIM-Filter: Bins

- We partition the genome into large sequences (**bins**).

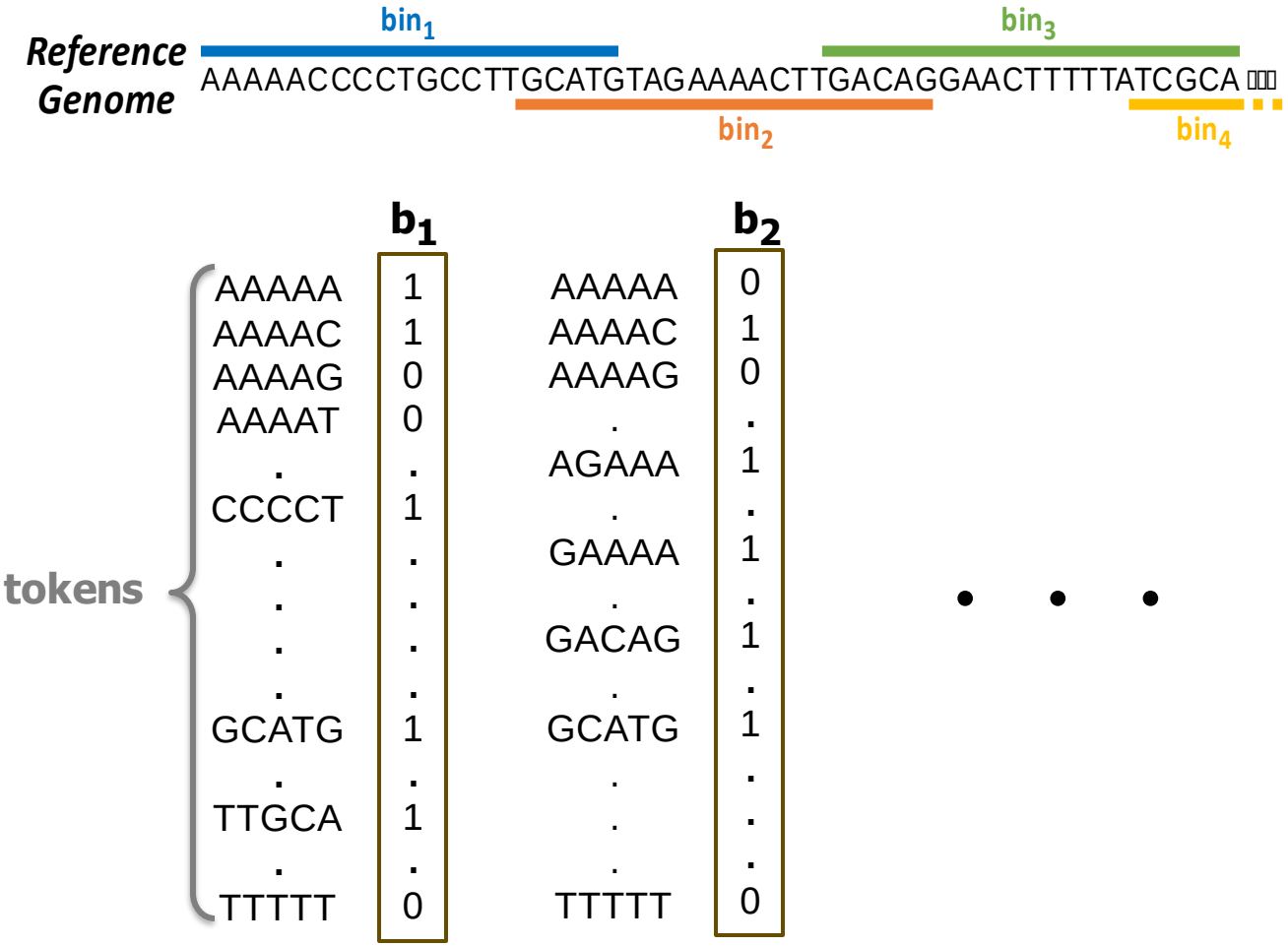
- Represent each bin with a **bitvector** that holds the occurrence of all permutations of a small string (**token**) in the bin
- To account for matches that straddle bins, we employ overlapping bins
  - A read will now always completely fall within a single bin



GRIM-Filter: Bitvectors



GRIM-Filter: Bitvectors

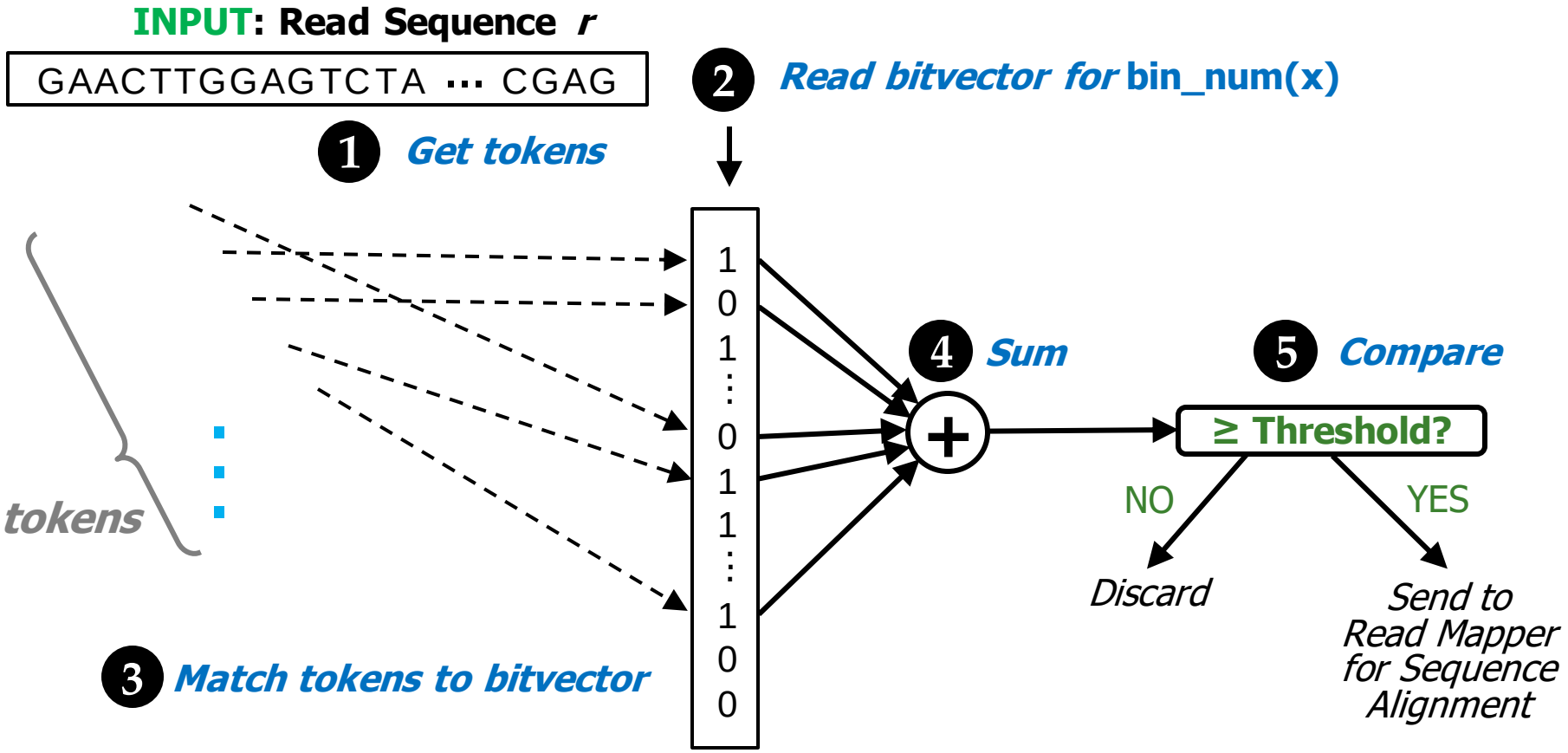


Storing all bitvectors requires $4^n * t$ bits in memory, where t = number of bins.

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

GRIM-Filter: Checking a Bin

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



Key Properties of GRIM-Filter

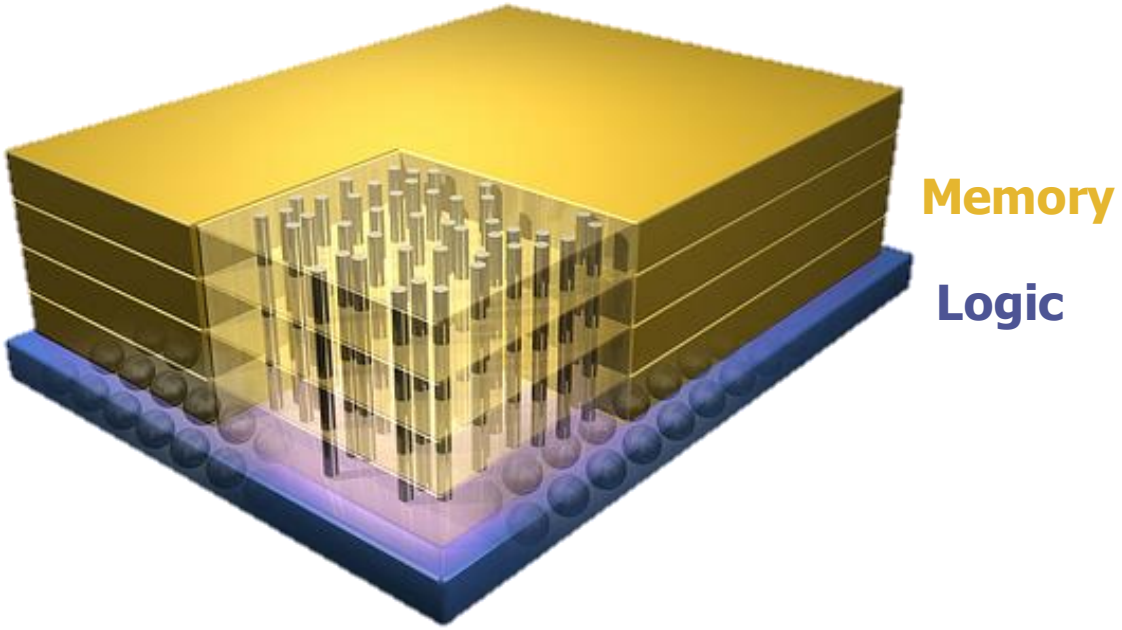
1. **Simple Operations**:
   - To check a given bin, find the **sum** of all bits corresponding to each token in the read
   - **Compare** against threshold to determine whether to align

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, we find that GRIM-Filter is memory bound

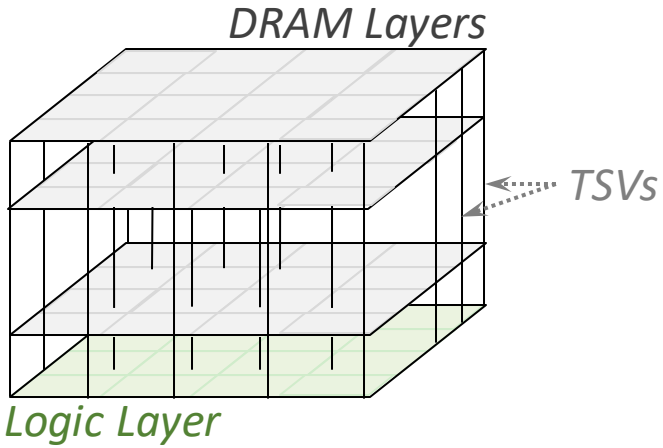
These properties together make GRIM-Filter a good algorithm to be run in 3D-Stacked DRAM

Opportunity: 3D-Stacked Logic+Memory



Other "True 3D" technologies under development

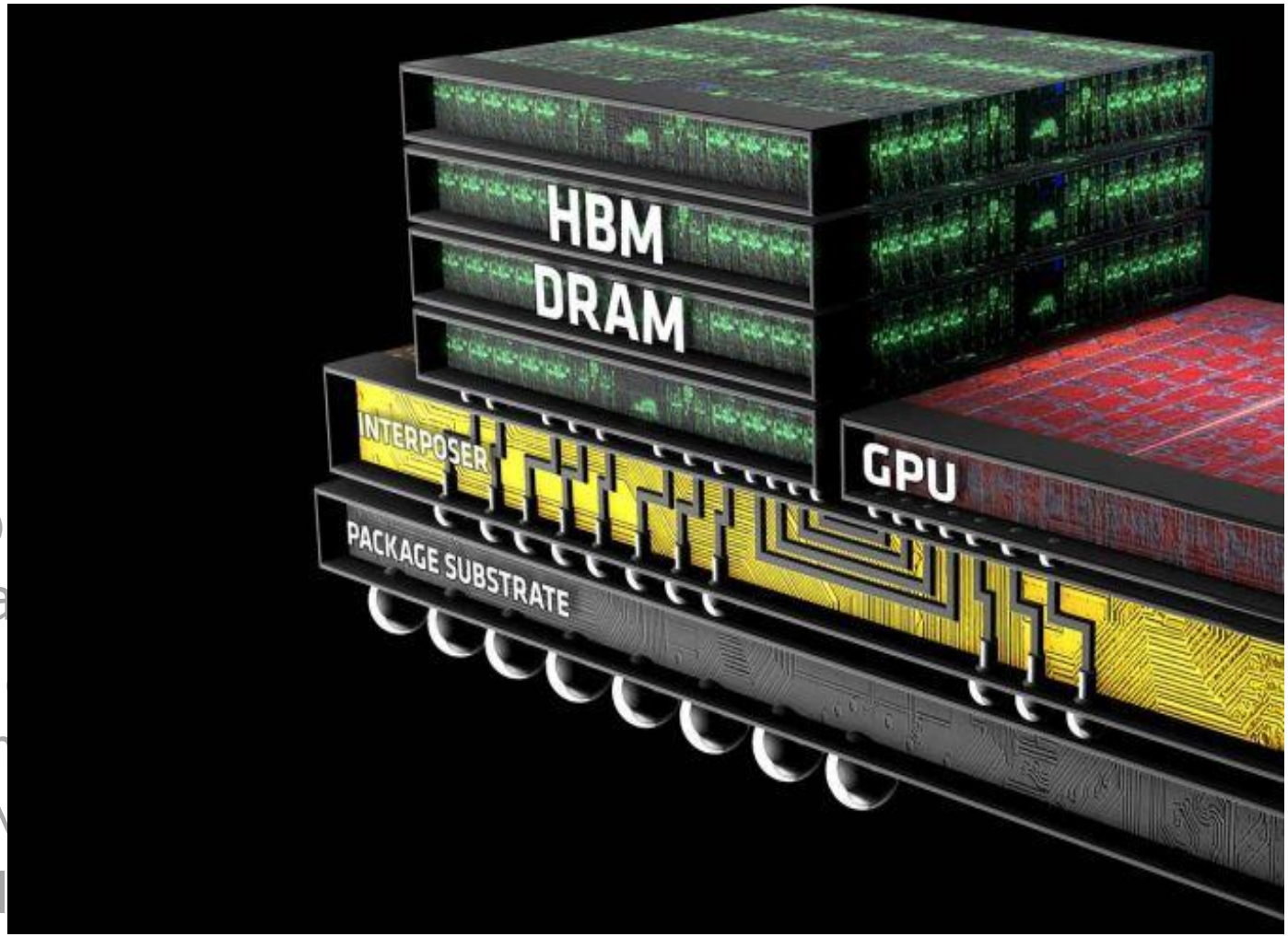
3D-Stacked Memory



- 3D-Stacked DRAM architecture has **extremely high bandwidth** as well as a stacked customizable logic layer
  - Logic Layer enables **Processing-in-Memory**, via high-bandwidth low-latency access to DRAM layers
  - Embed GRIM-Filter operations into **DRAM logic layer** and appropriately distribute bitvectors throughout memory

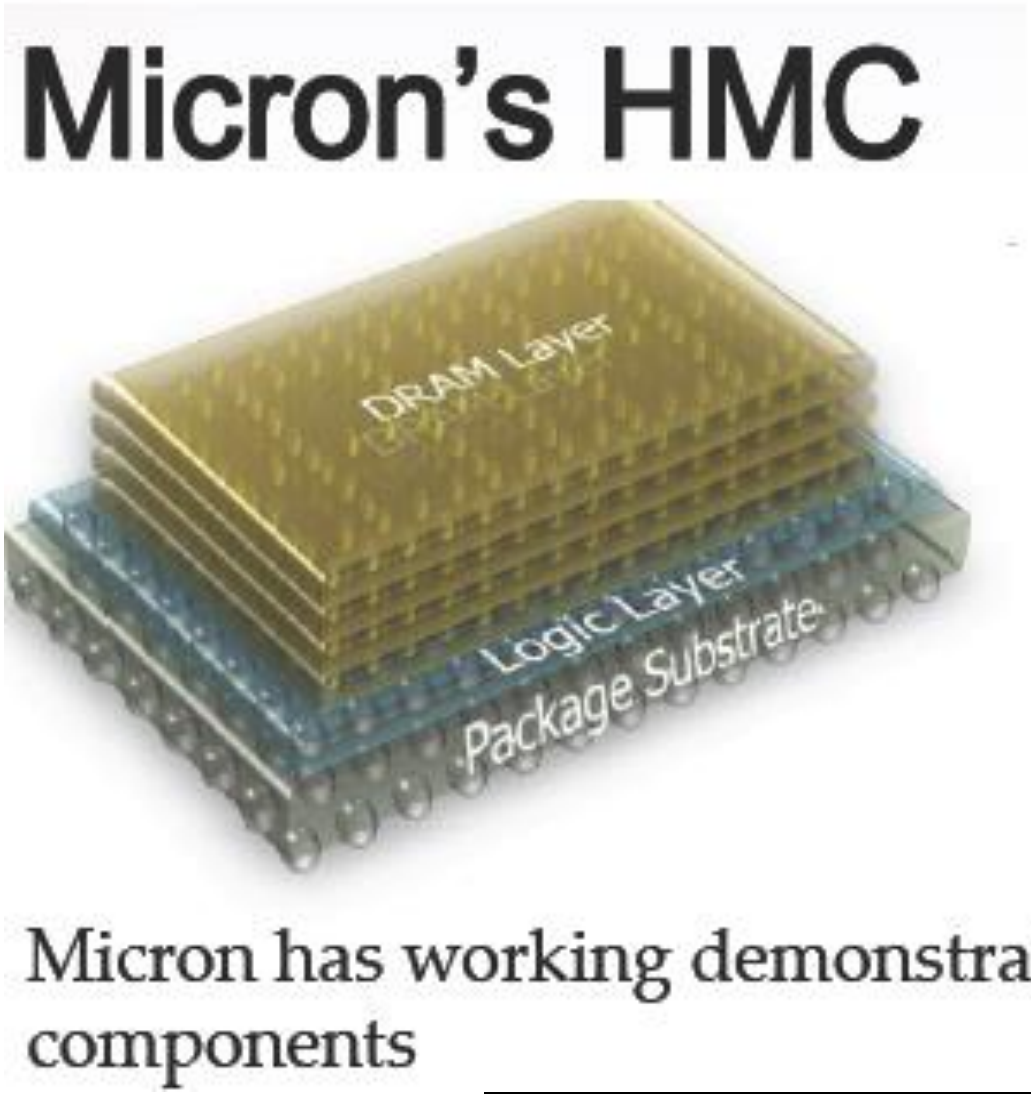
3D-Stacked Memory

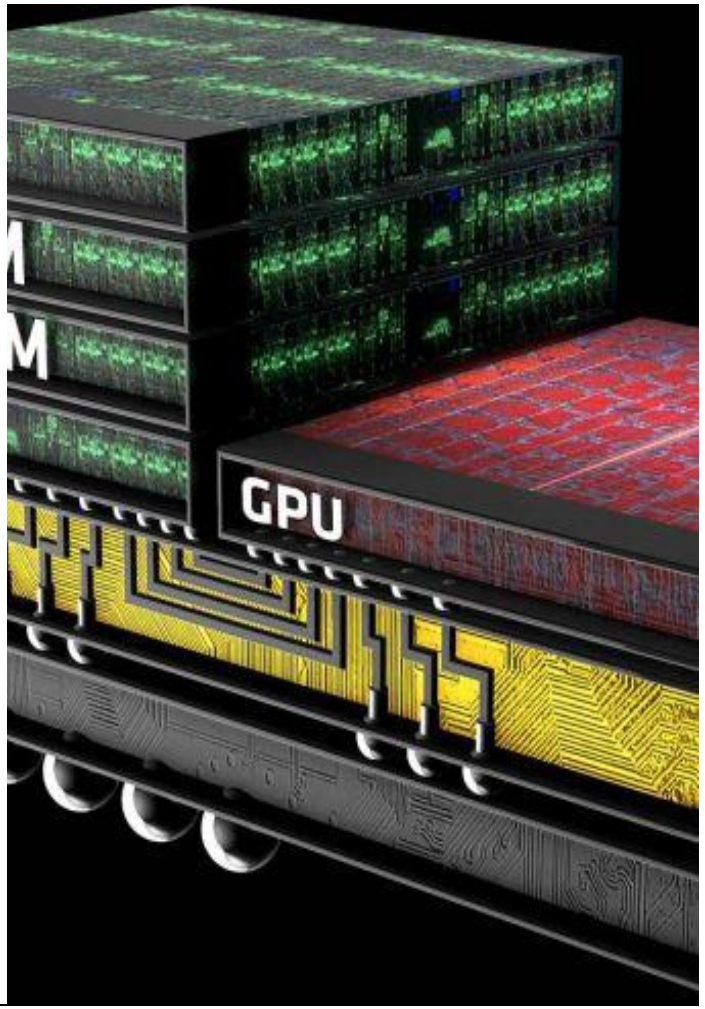
- 3D-Stacked D
  bandwidth a
  - Logic Layer
    computation
  - Embed GRI
    appropriatel



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

3D-Stacked Memory

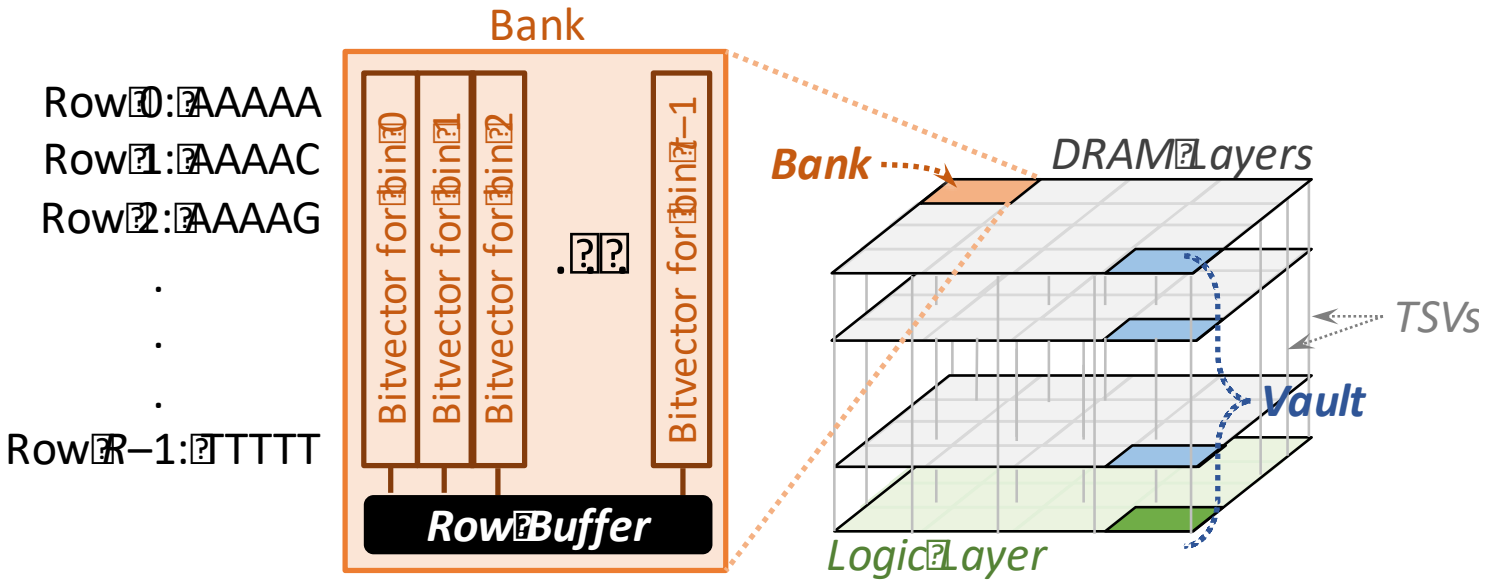




http://images.anandtech.com/doci/9266/HBMCar_678x452.jpg

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



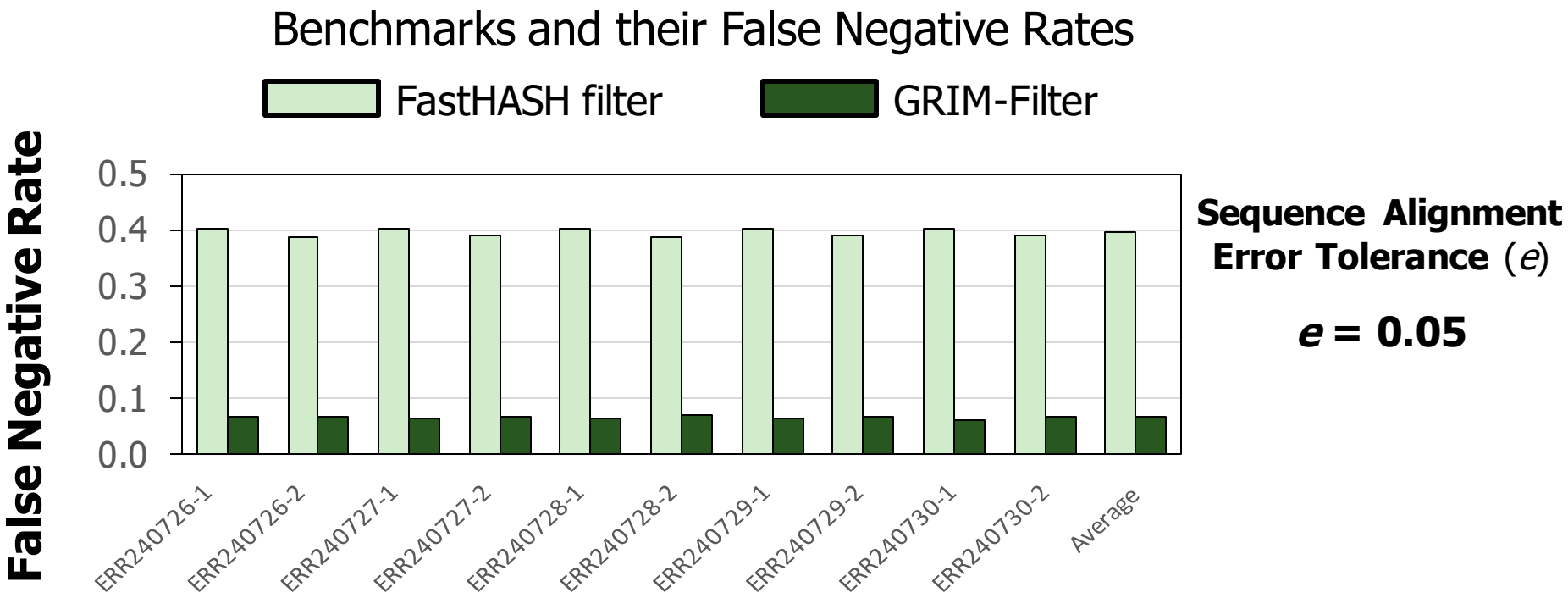
- Each DRAM layer is organized as an array of **banks**
  - A **bank** is an array of cells with a row buffer to transfer data

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

Methodology

- Performance simulated using an in-house 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 False Negative Rate

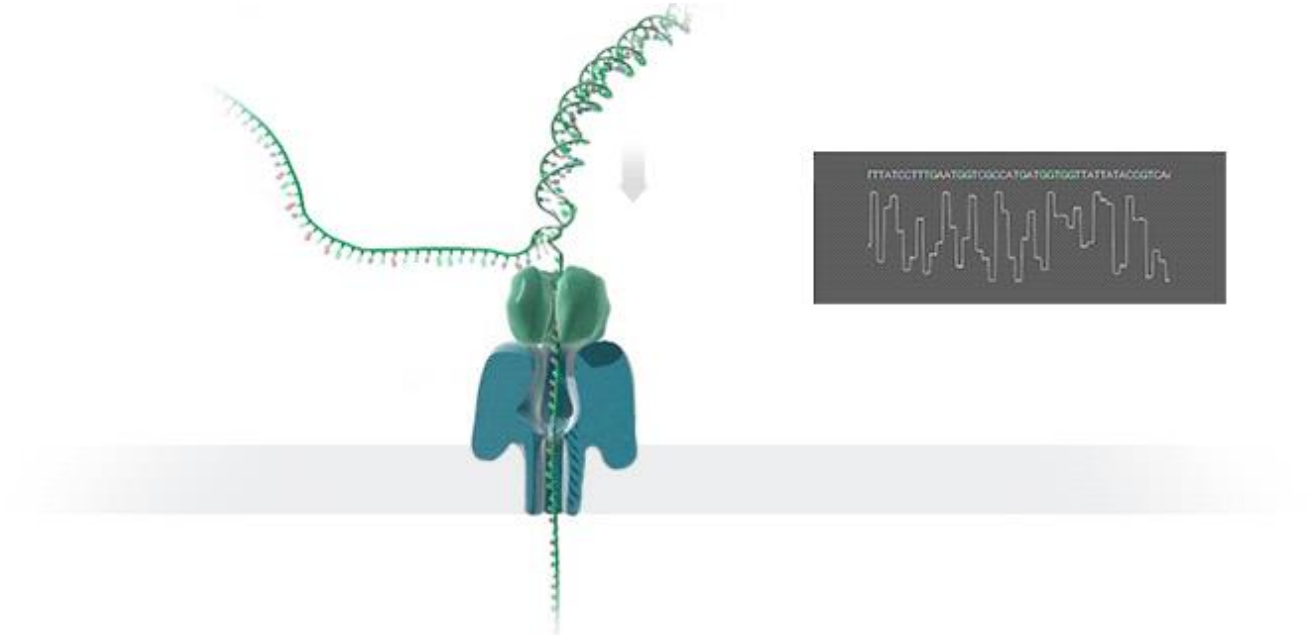


5.6x-6.4x False Negative reduction across real data sets

6.0x average reduction in False Negative Rate

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

Nanopore Sequencing



- **Nanopore** is a nano-scale hole
- In nanopore sequencers, an **ionic current** passes through the nanopores
- When the DNA strand passes through the nanopore, the sequencer measures the the **change in current**
- This change is used to identify the bases in the strand with the help of **different electrochemical structures** of the different bases

Advantages of Nanopore Sequencing

Nanopores:

- Do *not* require any labeling of the DNA or nucleotide for detection during sequencing
- Rely on the electronic or chemical structure of the different nucleotides for identification
- Allow sequencing very long reads, and
- Provide portability, low cost, and high throughput.

Nanopore Genome Assembly Pipeline

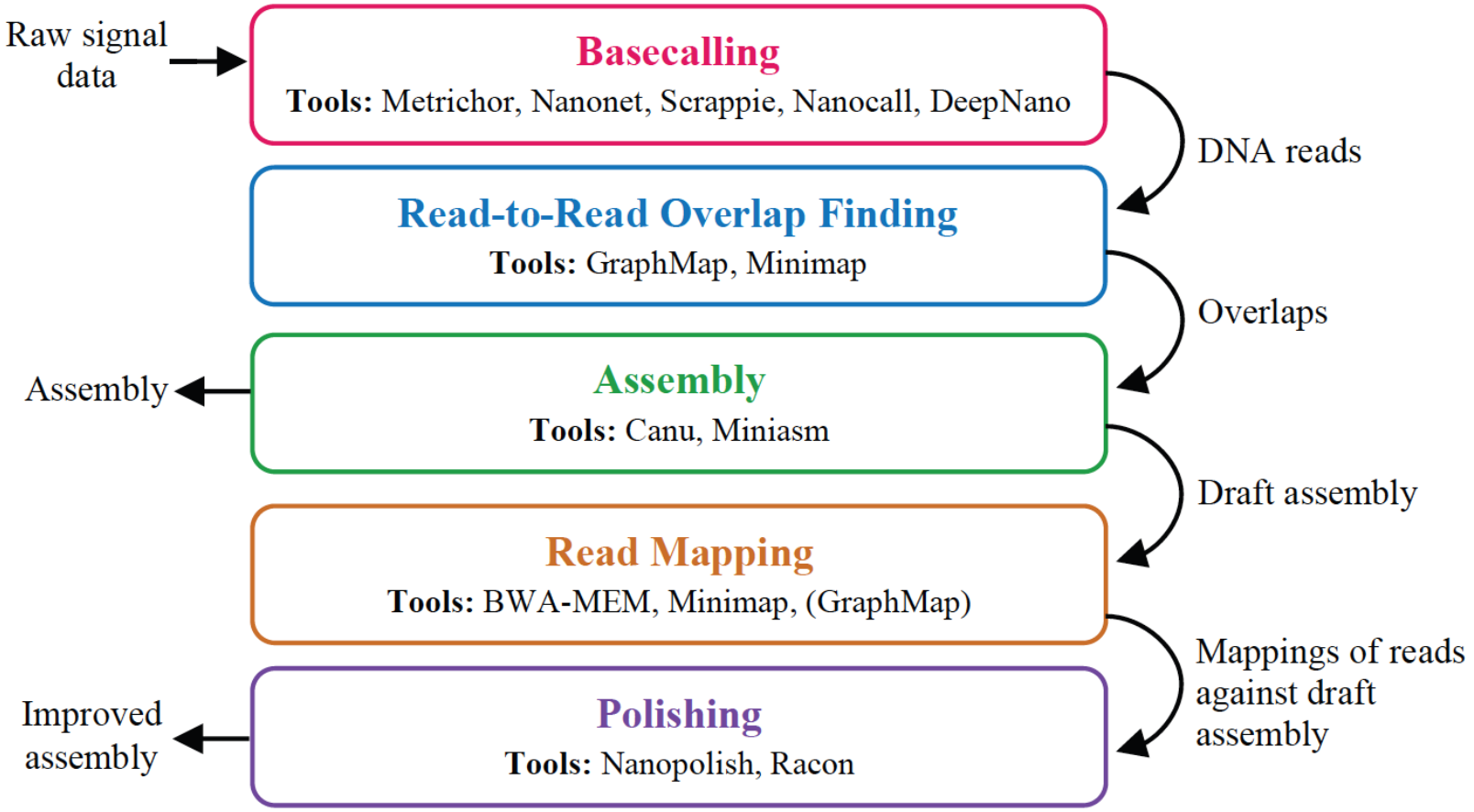


Figure 1. The analyzed genome assembly pipeline using nanopore sequence data, with its five steps and the associated tools for each step.

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly**" Briefings in Bioinformatics, 2018.

Nanopore Genome Assembly Tools (I)

Table 12. Accuracy analysis results for the full pipeline with a focus on the last two steps.

						Number of Bases	Number of Contigs	Identity (%)	Coverage (%)	Number of Mismatches	Number of Indels
1	Metrichor	+ —	+ Canu	+ BWA-MEM	+ Nanopolish	4,683,072	1	99.48	99.93	8,198	15,581
2	Metrichor	+ Minimap	+ Miniasm	+ BWA-MEM	+ Nanopolish	4,540,352	1	92.33	96.31	162,884	182,965
3	Metrichor	+ GraphMap	+ Miniasm	+ BWA-MEM	+ Nanopolish	4,637,916	2	92.38	95.80	159,206	180,603
4	Metrichor	+ —	+ Canu	+ BWA-MEM	+ Racon	4,650,502	1	98.46	100.00	18,036	51,842
5	Metrichor	+ —	+ Canu	+ Minimap	+ Racon	4,648,710	1	98.45	100.00	17,906	52,168
6	Metrichor	+ Minimap	+ Miniasm	+ BWA-MEM	+ Racon	4,598,267	1	97.70	99.91	24,014	82,906
7	Metrichor	+ Minimap	+ Miniasm	+ Minimap	+ Racon	4,600,109	1	97.78	100.00	23,339	79,721
8	Nanonet	+ —	+ Canu	+ BWA-MEM	+ Racon	4,622,285	1	98.48	100.00	16,872	52,509
9	Nanonet	+ —	+ Canu	+ Minimap	+ Racon	4,620,597	1	98.49	100.00	16,874	52,232
10	Nanonet	+ Minimap	+ Miniasm	+ BWA-MEM	+ Racon	4,593,402	1	98.01	99.97	20,322	72,284
11	Nanonet	+ Minimap	+ Miniasm	+ Minimap	+ Racon	4,592,907	1	98.04	100.00	20,170	70,705
12	Scrappie	+ —	+ Canu	+ BWA-MEM	+ Racon	4,673,871	1	98.40	99.98	13,583	60,612
13	Scrappie	+ —	+ Canu	+ Minimap	+ Racon	4,673,606	1	98.40	99.98	13,798	60,423
14	Scrappie	+ Minimap	+ Miniasm	+ BWA-MEM	+ Racon	5,157,041	8	97.87	99.80	18,085	78,492
15	Scrappie	+ Minimap	+ Miniasm	+ Minimap	+ Racon	5,156,375	8	97.87	99.94	17,922	77,807
16	Nanocall	+ —	+ Canu	+ BWA-MEM	+ Racon	1,383,851	86	93.49	28.82	19,057	65,244
17	Nanocall	+ —	+ Canu	+ Minimap	+ Racon	1,367,834	86	94.43	28.74	15,610	55,275
18	Nanocall	+ Minimap	+ Miniasm	+ BWA-MEM	+ Racon	4,707,961	5	90.75	97.11	91,502	347,005
19	Nanocall	+ Minimap	+ Miniasm	+ Minimap	+ Racon	4,673,069	5	92.23	97.10	72,646	291,918
20	DeepNano	+ —	+ Canu	+ BWA-MEM	+ Racon	7,429,290	106	96.46	99.24	27,811	102,682
21	DeepNano	+ —	+ Canu	+ Minimap	+ Racon	7,404,454	106	96.03	99.21	34,023	110,640
22	DeepNano	+ Minimap	+ Miniasm	+ BWA-MEM	+ Racon	4,566,253	1	96.76	99.86	25,791	125,386
23	DeepNano	+ Minimap	+ Miniasm	+ Minimap	+ Racon	4,571,810	1	96.90	99.97	24,994	119,519

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly**" Briefings in Bioinformatics, 2018.

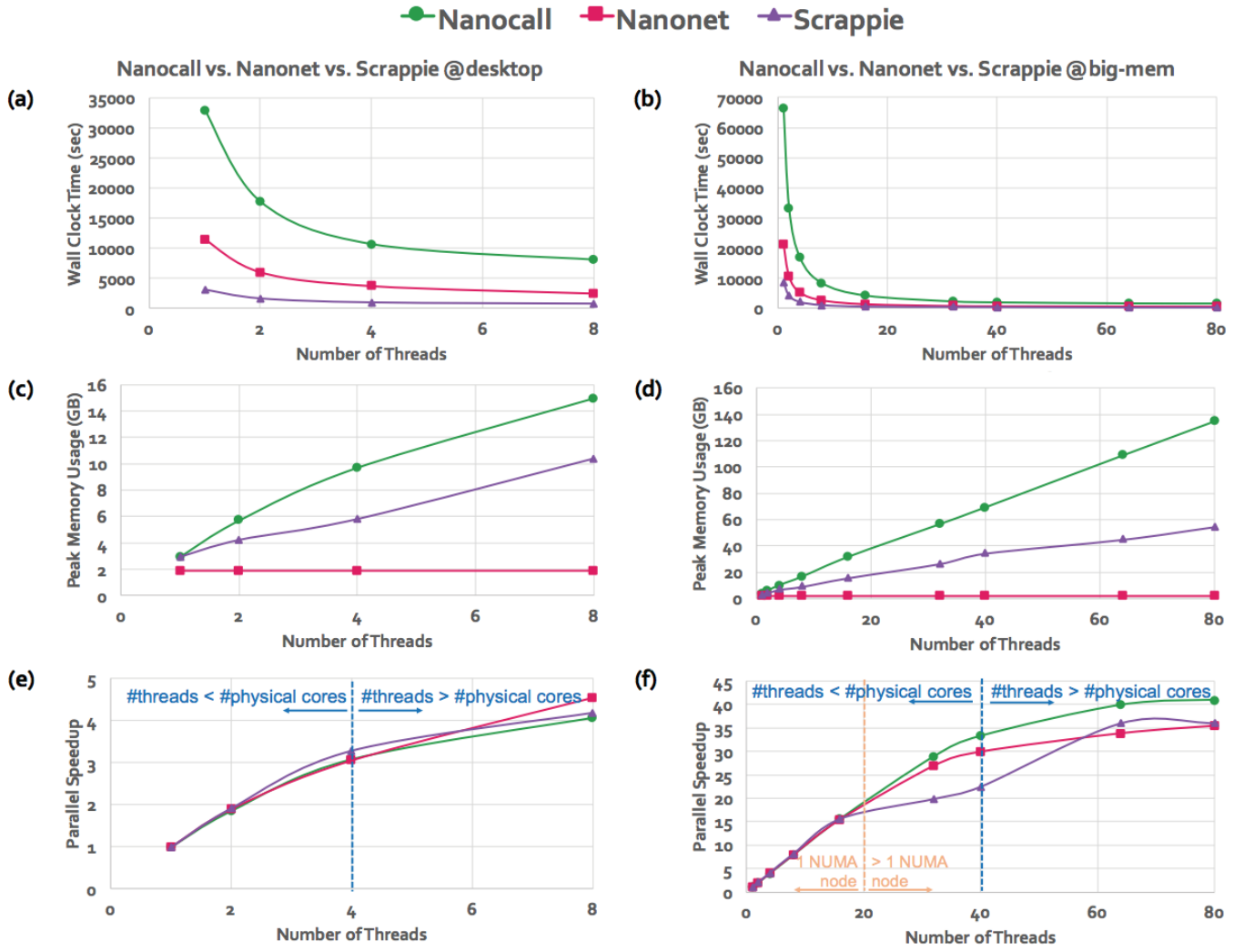
Nanopore Genome Assembly Tools (II)

Table 13. Performance analysis results for the full pipeline with a focus on the last two steps.

									Step 4: Read Mapper			Step 5: Polisher		
									Wall Clock Time (h:m:s)	CPU Time (h:m:s)	Memory Usage (GB)	Wall Clock Time (h:m:s)	CPU Time (h:m:s)	Memory Usage (GB)
1	Metrichor	+	—	+	Canu	+	BWA-MEM	+	Nanopolish	24:43	15:47:21	5.26	5:51:00	191:18:52	13.38
2	Metrichor	+	Minimap	+	Miniasm	+	BWA-MEM	+	Nanopolish	12:33	7:50:54	3.75	122:52:00	4458:36:10	31.36
3	Metrichor	+	GraphMap	+	Miniasm	+	BWA-MEM	+	Nanopolish	12:47	7:57:58	3.60	129:46:00	4799:03:51	31.31
4	Metrichor	+	—	+	Canu	+	BWA-MEM	+	Racon	24:20	15:43:40	6.60	14:44	9:09:22	8.11
5	Metrichor	+	—	+	Canu	+	Minimap	+	Racon	3	1:35	0.26	15:12	9:45:33	14.55
6	Metrichor	+	Minimap	+	Miniasm	+	BWA-MEM	+	Racon	12:10	7:48:10	5.19	15:43	9:33:39	9.98
7	Metrichor	+	Minimap	+	Miniasm	+	Minimap	+	Racon	3	1:24	0.26	20:28	8:57:40	18.24
8	Nanonet	+	—	+	Canu	+	BWA-MEM	+	Racon	9:08	5:53:18	4.84	6:33	4:02:10	4.47
9	Nanonet	+	—	+	Canu	+	Minimap	+	Racon	2	54	0.26	6:45	4:17:26	7.93
10	Nanonet	+	Minimap	+	Miniasm	+	BWA-MEM	+	Racon	4:40	2:58:02	3.88	7:08	4:19:30	5.35
11	Nanonet	+	Minimap	+	Miniasm	+	Minimap	+	Racon	2	46	0.26	7:01	4:18:48	9.53
12	Scrappie	+	—	+	Canu	+	BWA-MEM	+	Racon	33:41	21:11:06	8.66	13:32	8:24:44	7.58
13	Scrappie	+	—	+	Canu	+	Minimap	+	Racon	3	1:39	0.27	18:45	7:43:17	13.20
14	Scrappie	+	Minimap	+	Miniasm	+	BWA-MEM	+	Racon	22:41	14:31:00	6.08	14:37	8:53:59	9.50
15	Scrappie	+	Minimap	+	Miniasm	+	Minimap	+	Racon	3	1:27	0.27	15:10	9:02:45	12.72
16	Nanocall	+	—	+	Canu	+	BWA-MEM	+	Racon	4:52	3:01:15	3.80	11:07	3:26:52	5.63
17	Nanocall	+	—	+	Canu	+	Minimap	+	Racon	3	1:16	0.22	7:28	2:50:35	3.62
18	Nanocall	+	Minimap	+	Miniasm	+	BWA-MEM	+	Racon	16:06	10:27:20	5.06	18:56	11:32:45	11.47
19	Nanocall	+	Minimap	+	Miniasm	+	Minimap	+	Racon	4	1:18	0.26	11:49	7:08:59	10.98
20	DeepNano	+	—	+	Canu	+	BWA-MEM	+	Racon	17:36	11:30:20	4.43	12:48	7:13:04	8.88
21	DeepNano	+	—	+	Canu	+	Minimap	+	Racon	3	1:24	0.28	11:39	6:55:01	3.73
22	DeepNano	+	Minimap	+	Miniasm	+	BWA-MEM	+	Racon	8:15	5:22:29	4.11	14:16	8:34:32	10.30
23	DeepNano	+	Minimap	+	Miniasm	+	Minimap	+	Racon	3	1:10	0.26	12:29	7:55:32	17.11

Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly**" Briefings in Bioinformatics, 2018.

Nanopore Genome Assembly Tools (III)



Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly**" to appear in Briefings in Bioinformatics, 2018.