Accelerating Genome Analysis

A Primer on an Ongoing Journey

Onur Mutlu
omutlu@gmail.com
https://people.inf.ethz.ch/omutlu

16 February 2019
AACBB Keynote Talk

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
  - My personal journey since September 2006

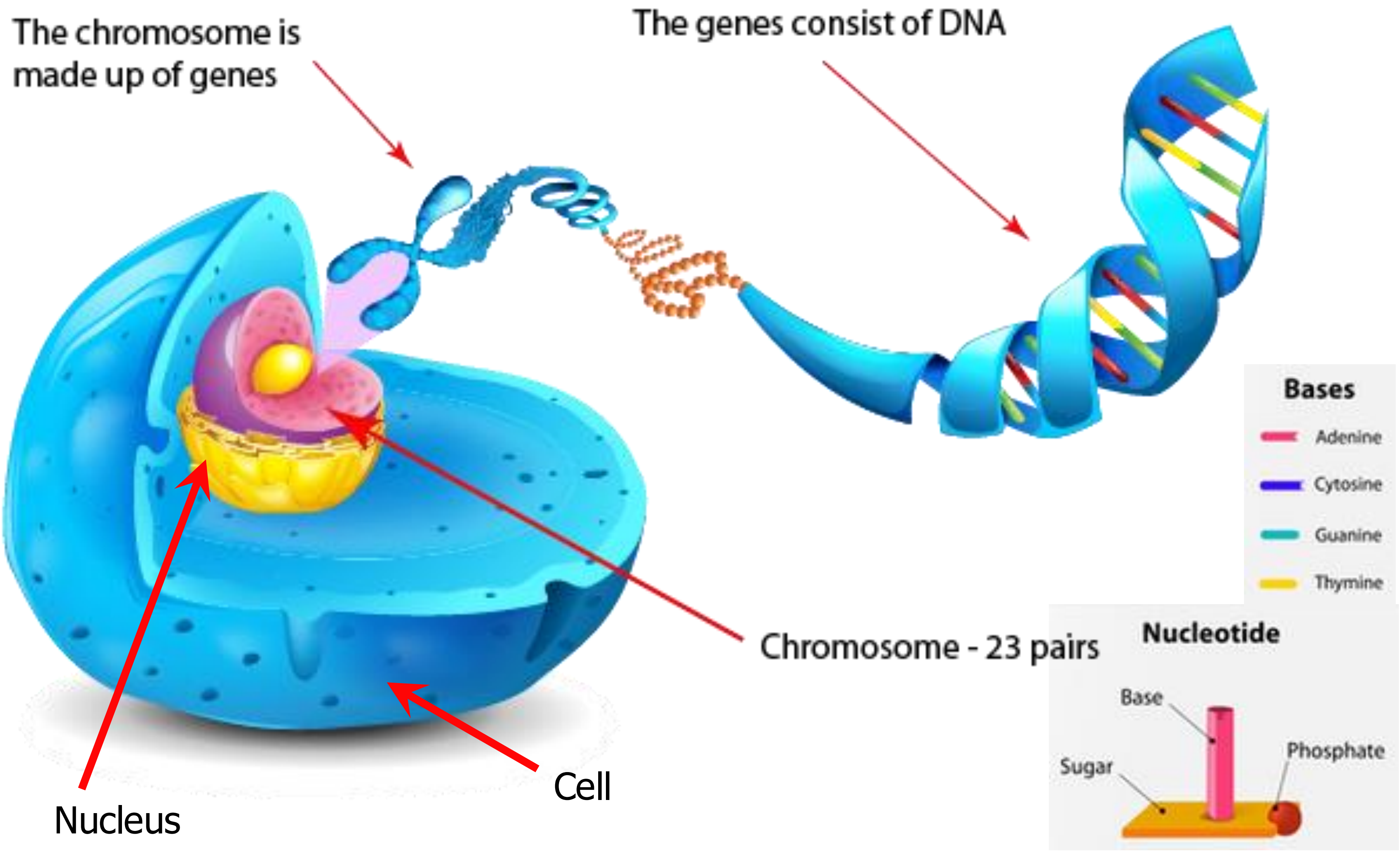
Our Dream (in 2007)

- An embedded device that can perform comprehensive genome analysis in real time (within a minute)
  - Which of these DNAs does this DNA segment match with?
  - What is the likely genetic disposition of this patient to this drug?
  - . . .

Agenda

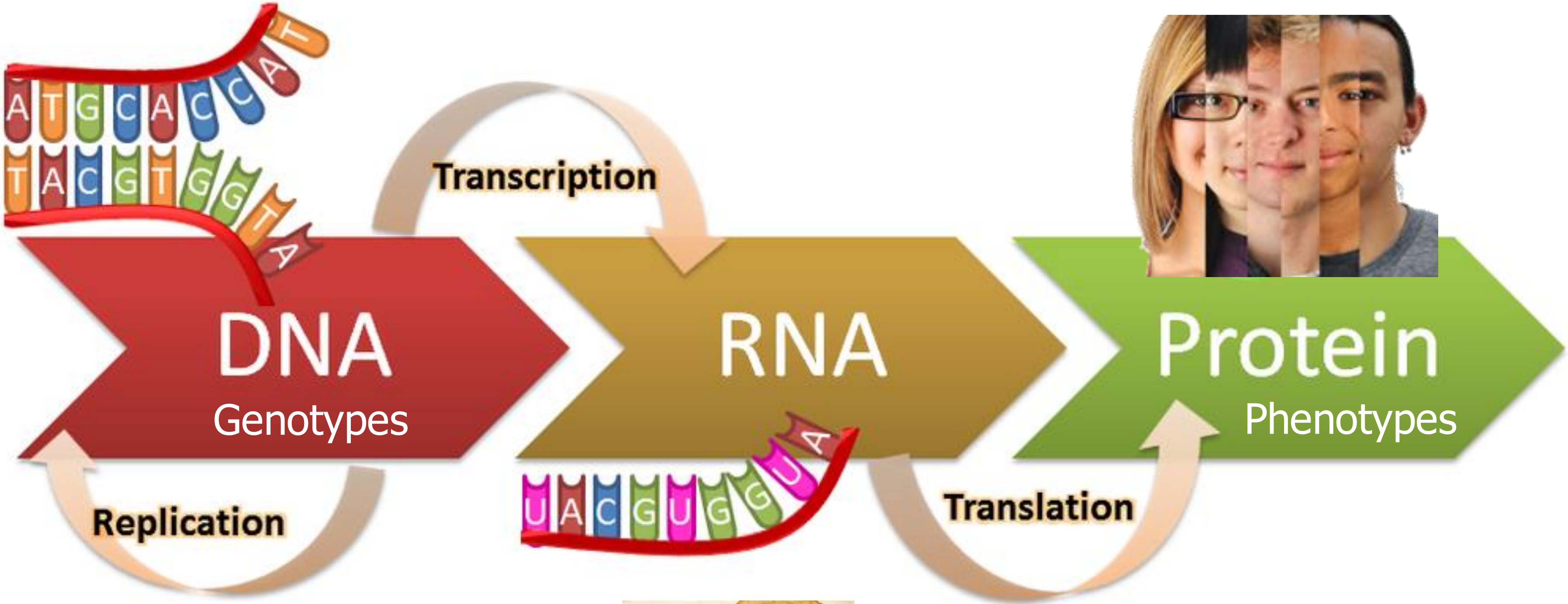
- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions
- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory
- Future Opportunities: New Sequencing Technologies

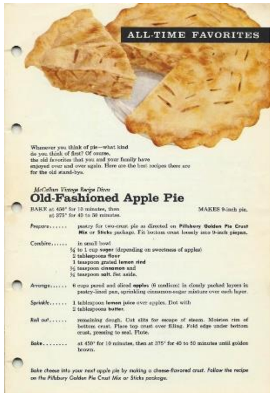
What Is a Genome Made Of?



The discovery of DNA’s double-helical structure (Watson+, 1953)

The Central Dogma of Molecular Biology




ALL-TIME FAVORITES

Whenever you think of pie—what kind do you think of first? Of course, the old favorites that you and your family have enjoyed over and over again. Here are the best recipes there are for the old stand-bys.

McCollum Vintage Recipe Diver

Old-Fashioned Apple Pie

BAKE at 450° for 10 minutes, then
at 375° for 40 to 50 minutes.

MAKES 9-inch pie.

Prepare...... pastry for two-crust pie as directed on **Pillsbury Golden Pie Crust Mix** or **Sticks** package. Fit bottom crust loosely into 9-inch piepan.

Combine...... in small bowl
¾ to 1 cup **sugar** (depending on sweetness of apples)
2 tablespoons **flour**
1 teaspoon grated **lemon rind**
½ teaspoon **cinnamon** and
¼ teaspoon **salt**. Set aside.

Arrange...... 6 cups pared and sliced **apples** (6 medium) in closely packed layers in pastry-lined pan, sprinkling cinnamon-sugar mixture over each layer.

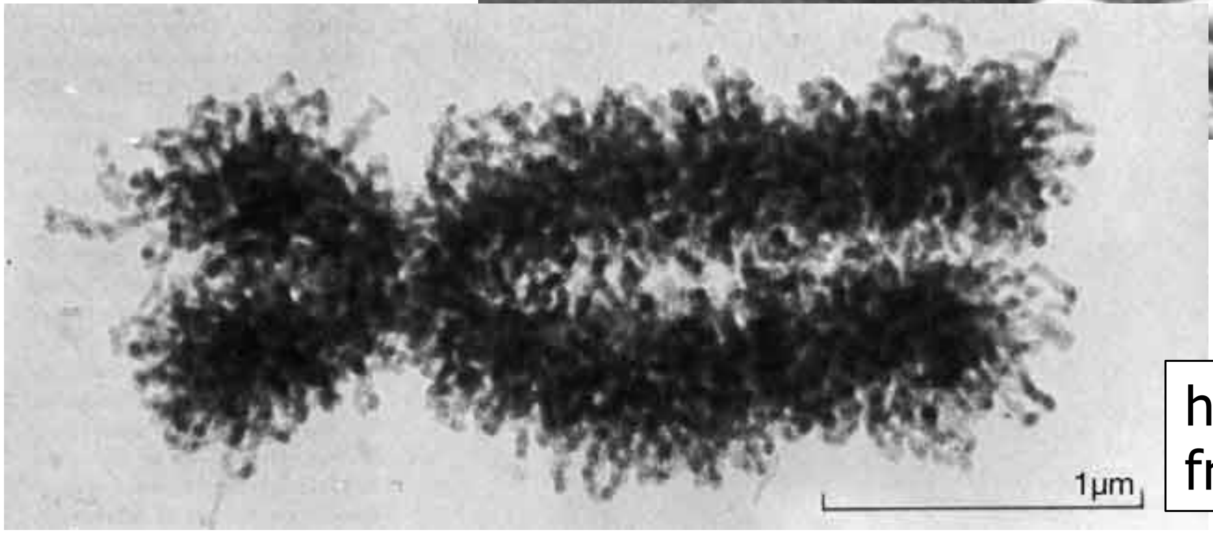
Sprinkle...... 1 tablespoon **lemon juice** over apples. Dot with
2 tablespoons **butter**.

Roll out...... remaining dough. Cut slits for escape of steam. Moisten rim of bottom crust. Place top crust over filling. Fold edge under bottom crust, pressing to seal. Flute.

Bake......... at 450° for 10 minutes, then at 375° for 40 to 50 minutes until golden brown.

Bake cheese into your next apple pie by making a cheese-flavored crust. Follow the recipe on the Pillsbury Golden Pie Crust Mix or Sticks package.

DNA Under Electron Microscope



human chromosome #12
from HeLa’s cell

DNA Sequencing

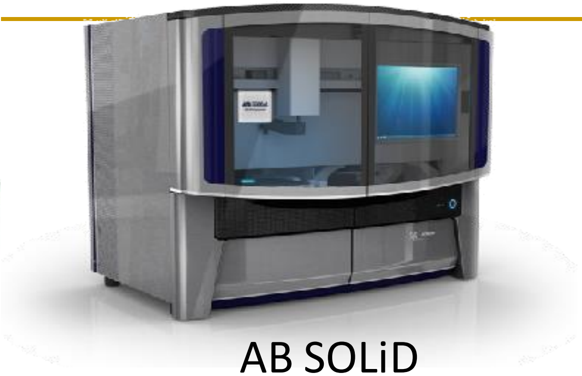
- Goal:
  - Find the complete sequence of A, C, G, T’s in DNA.

- 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)

Untangling Yarn Balls & DNA Sequencing

Genome Sequencers

Roche/454



AB SOLiD

Illumina MiSeq

Complete Genomics

Illumina HiSeq2000

Pacific Biosciences RS

Oxford Nanopore MinION

Illumina NovaSeq 6000

Oxford Nanopore GridION

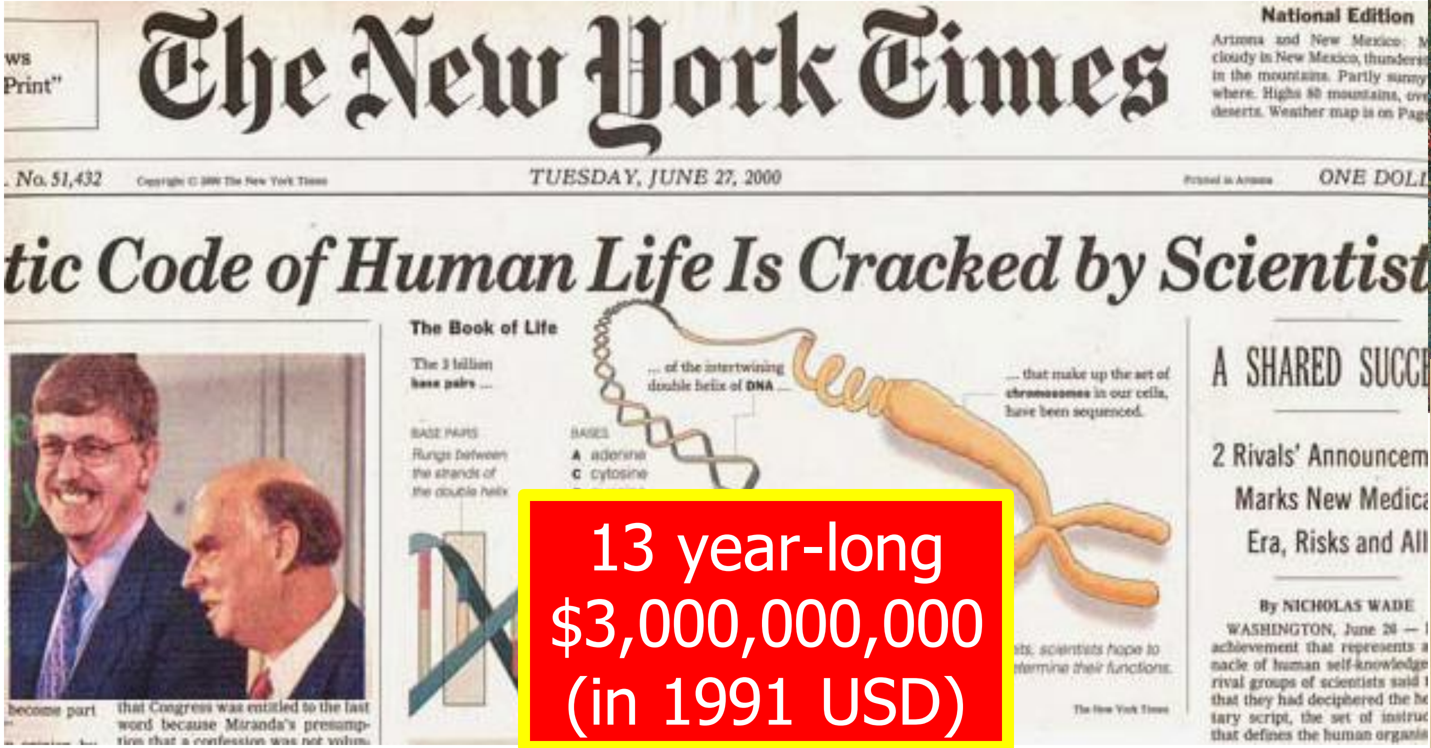
Ion Torrent PGM

Ion Torrent Proton

… and more! All produce data with different properties.

The Genomic Era

- 1990-2003: The Human Genome Project (HGP) provides a complete and accurate sequence of all **DNA base pairs** that make up the human genome and finds 20,000 to 25,000 human genes.



ws
Print"

The New York Times

National Edition

Arizona and New Mexico: M
cloudy in New Mexico, thunderst
in the mountains. Partly sunny
where. Highs 80 mountains, ove
deserts. Weather map is on Page

No. 51,432 Copyright © 2000 The New York Times TUESDAY, JUNE 27, 2000 Printed in Arizona ONE DOLL

tic Code of Human Life Is Cracked by Scientist

The Book of Life

The 3 billion **base pairs** ...

... of the intertwining double helix of **DNA** ...

... that make up the set of **chromosomes** in our cells, have been sequenced.

BASE PAIRS
Rungs between the strands of the double helix

BASES
A adenine
C cytosine

ts; scientists hope to
etermine their functions.

The New York Times

A SHARED SUCCE

2 Rivals' Announcem
Marks New Medica
Era, Risks and All

By NICHOLAS WADE

WASHINGTON, June 26 — achievement that represents a nacle of human self-knowledge rival groups of scientists said that they had deciphered the he tary script, the set of instruc that defines the human organis

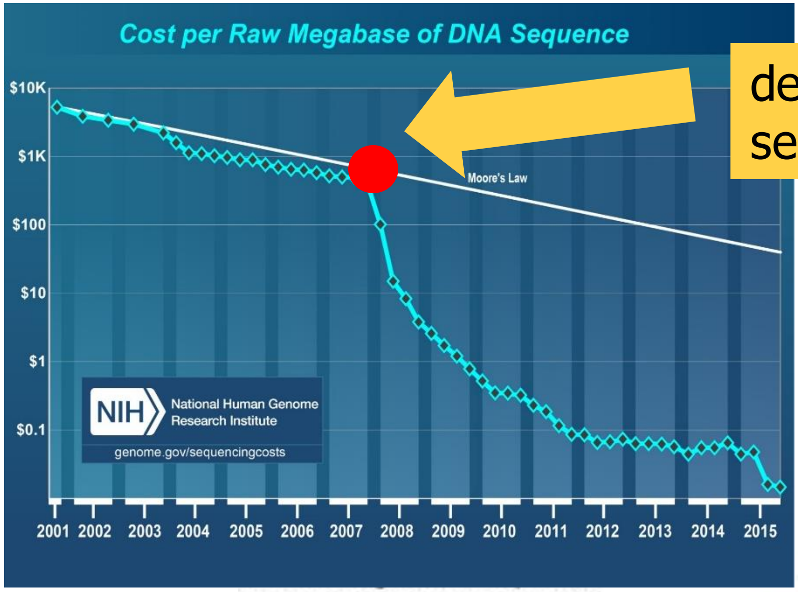
become part

that Congress was entitled to the last word because Miranda's presumption that a confession was not volun-

13 year-long
$3,000,000,000
(in 1991 USD)

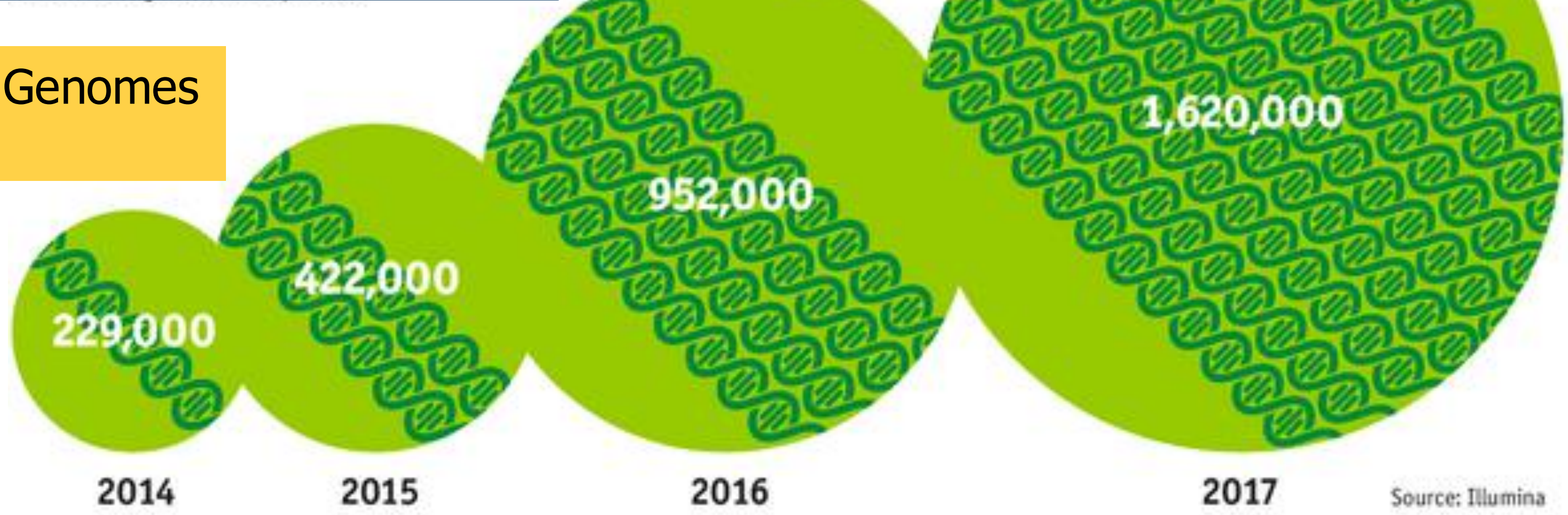
15 February 2001
nature
www.nature.com
the human genome
Nuclear fission
Five-dimensional energy landscapes
Seafloor spreading
The view from under the Arctic ice
Career prospects
Sequence creates new opportunities

The Genomic Era (continued)



development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

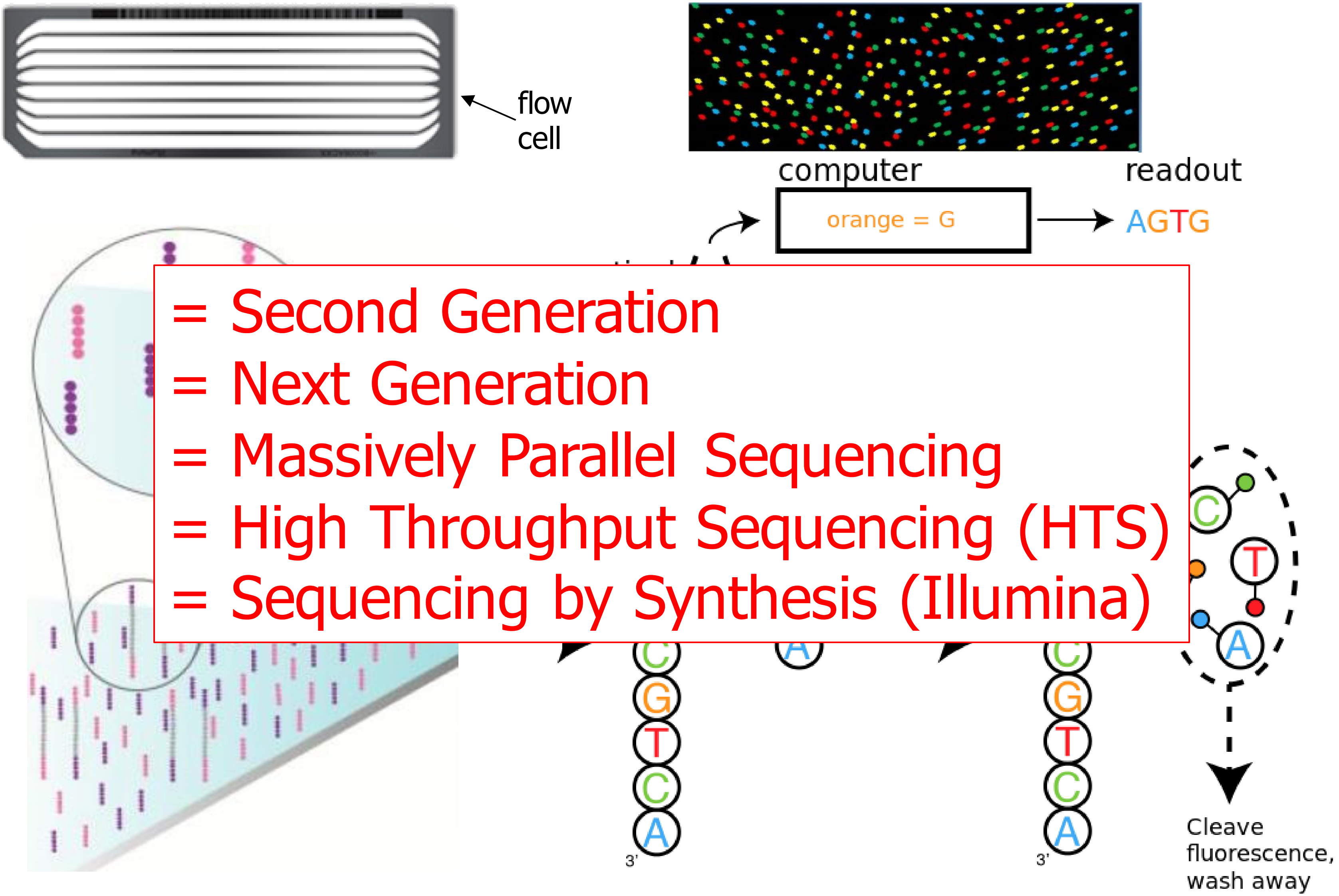


Source: Illumina

The Economist

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

High-Throughput Sequencing (HTS)



= Second Generation

= Next Generation

= Massively Parallel Sequencing

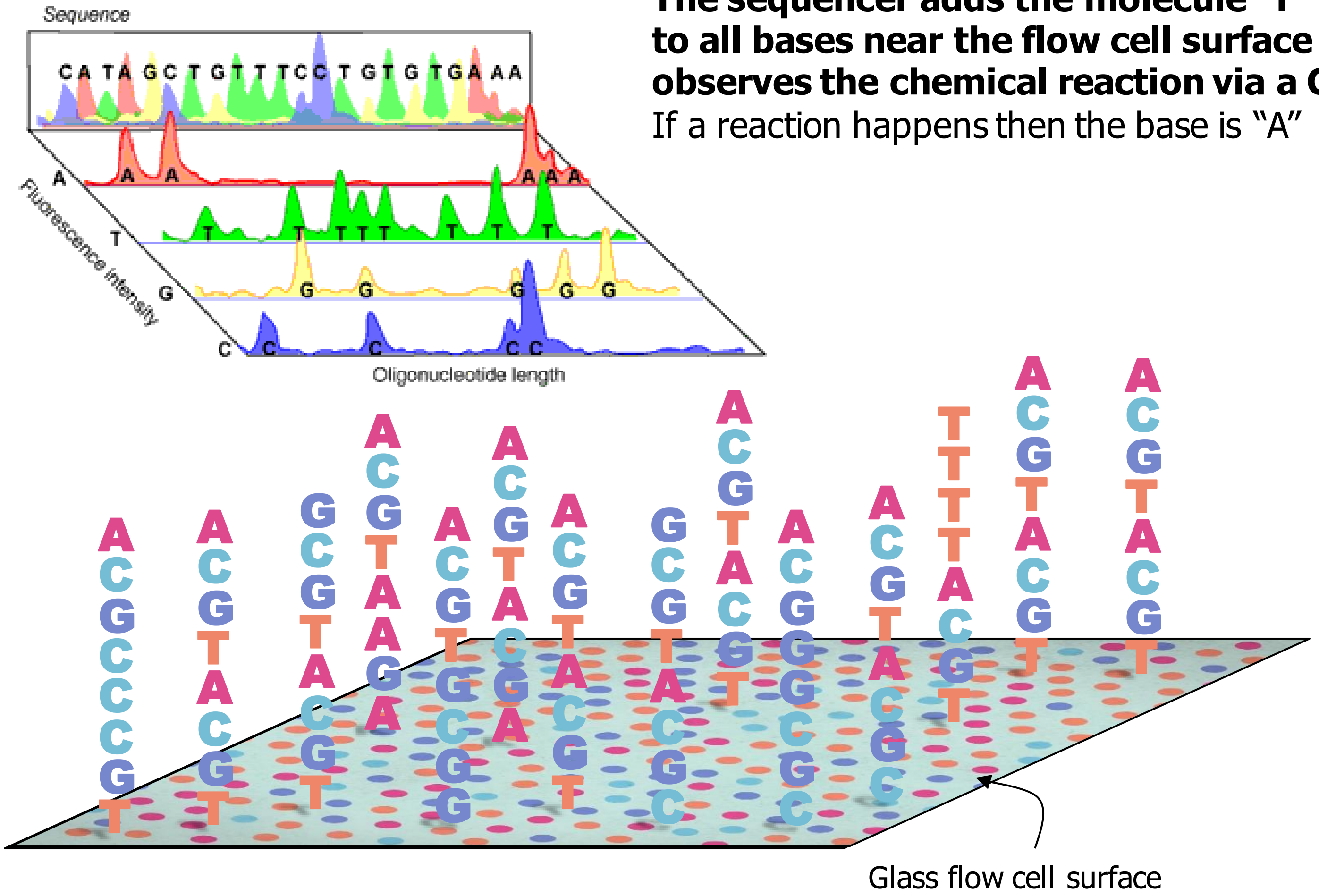
= High Throughput Sequencing (HTS)

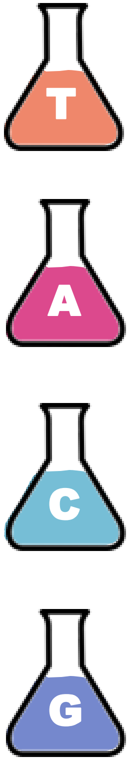
= Sequencing by Synthesis (Illumina)

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.

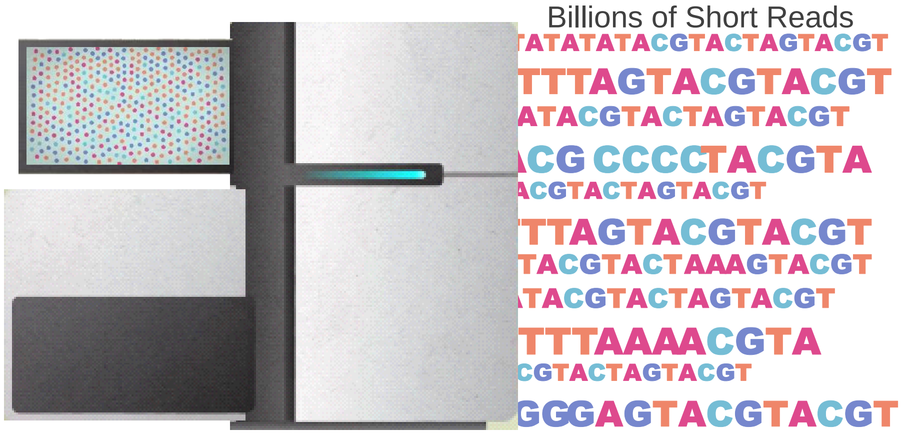
High-Throughput Sequencing

- Massively parallel sequencing technology
  - Illumina, Roche 454, Ion Torrent, SOLID…

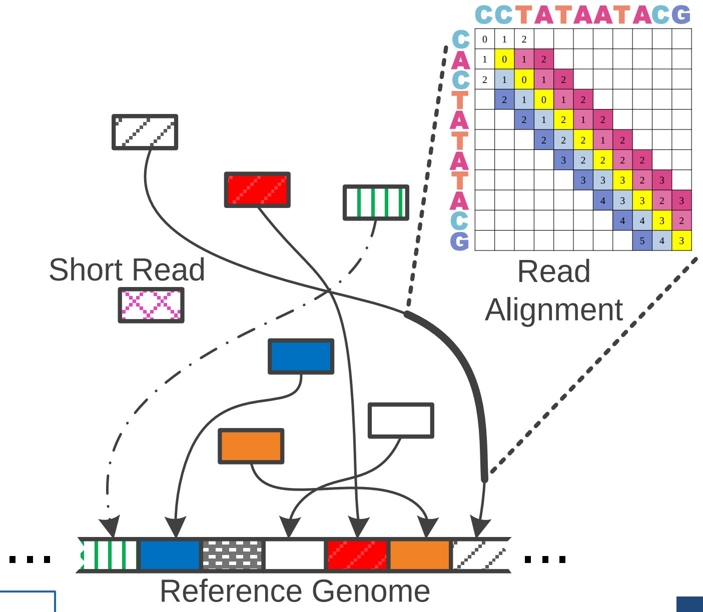
- Small DNA fragments are first amplified and then sequenced in parallel, leading to
  - High throughput
  - High speed
  - Low cost
  - Short reads

- 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

Genome Analysis



1 Sequencing



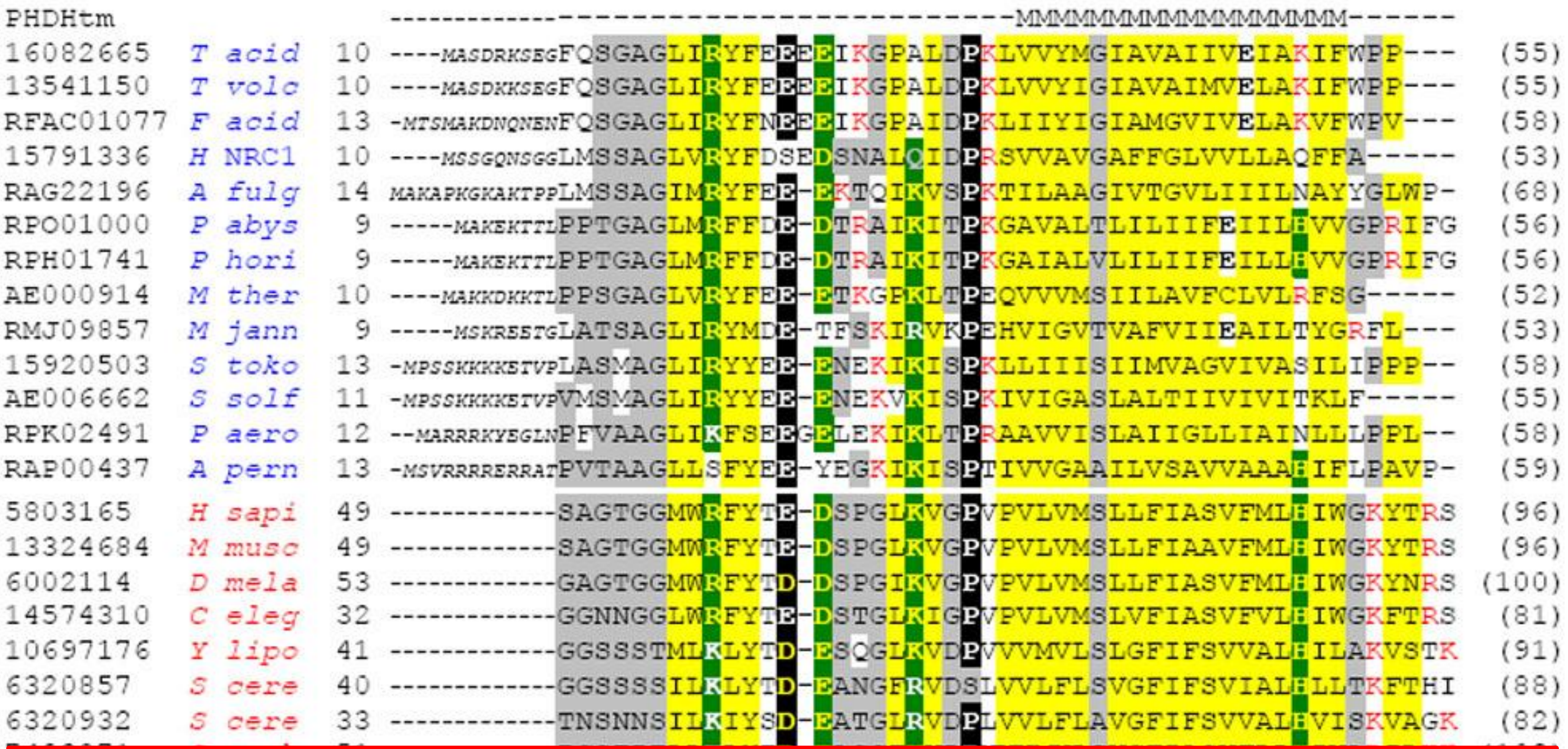
Read Mapping 2

```
reference: TTTATCGCTTCCATGACGCAG
read1:        ATCGCATCC
read2:       TATCGCATC
read3:            CATCCATGA
read4:          CGCTTCCAT
read5:               CCATGACGC
read6:             TTCCATGAC
```

3 Variant Calling

Scientific Discovery 4

Multiple sequence alignment



```
PHDHtm                               ---------------------------------MMMMMMMMMMMMMMMMMMM------
16082665  T acid  10 ----MASDRKSEGFQSGAGLIRYFEEEEIKGPALDPKLVVYMGIAVAIIVEIAKIFWPP---   (55)
13541150  T volc  10 ----MASDKKSEGFQSGAGLIRYFEEEEIKGPALDPKLVVYIGIAVAIMVELAKIFWPP---   (55)
RFAC01077 F acid  13 -MTSMAKDNQNENFQSGAGLIRYFNEEEIKGPAIDPKLIIYIGIAMGVIVELAKVFWPV---   (58)
15791336  H NRC1  10 ----MSSGQNSGGLMSSAGLVRYFDSEDSNALQIDPRSVVAVGAFFGLVVLLAQFFA-----   (53)
RAG22196  A fulg  14 MAKAPKGKAKTPPLMSSAGIMRYFEE-EKTQIKVSPKTILAAGIVTGVLIIIILNAYYGLWP-   (68)
RPO01000  P abys   9 -----MAKEKTTLPPTGAGLMRFFDE-DTRAIKITPKGAVALTLILIIFEIILHVVGPRIFG   (56)
RPH01741  P hori   9 -----MAKEKTTLPPTGAGLMRFFDE-DTRAIKITPKGAIALVLILIIFEILLHVVGPRIFG   (56)
AE000914  M ther  10 ----MAKKDKKTLPPSGAGLVRYFEE-ETKGPKLTPEQVVVMSIILAVFCLVLRFSG-----   (52)
RMJ09857  M jann   9 -----MSKREETGLATSAGLIRYMDE-TFSKIRVKPEHVIGVTVAFVIIEAILTYGRFL---   (53)
15920503  S toko  13 -MPSSKKKKETVPLASMAGLIRYYEE-ENEKIKISPKLLIIISIIMVAGVIVASILIPPP--   (58)
AE006662  S solf  11 -MPSSKKKKETVPVMSMAGLIRYYEE-ENEKVKISPKIVIGASLALTIIVIVITKLF-----   (55)
RPK02491  P aero  12 --MARRRKYEGLNPFVAAGLIKFSEEGELEKIKLTPRAAVVISLAIIGLLIAINLLLPPL--   (58)
RAP00437  A pern  13 -MSVRRRRERRATPVTAAGLLSFYEE-YEGKIKISPTIVVGAAILVSAVVAAAHIFLPAVP-   (59)
5803165   H sapi  49 ------------SAGTGGMWRFYTE-DSPGLKVGPVPVLVMSLLFIASVFMLHIWGKYTRS   (96)
13324684  M musc  49 ------------SAGTGGMWRFYTE-DSPGLKVGPVPVLVMSLLFIAAVFMLHIWGKYTRS   (96)
6002114   D mela  53 ------------GAGTGGMWRFYTD-DSPGIKVGPVPVLVMSLLFIASVFMLHIWGKYNRS  (100)
14574310  C eleg  32 ------------GGNNGGLWRFYTE-DSTGLKIGPVPVLVMSLVFIASVFVLHIWGKFTRS   (81)
10697176  Y lipo  41 ------------GGSSSTMLKLYTD-ESQGLKVDPVVVMVLSLGFIFSVVALHILAKVSTK   (91)
6320857   S cere  40 ------------GGSSSSILKLYTD-EANGFRVDSLVVLFLSVGFIFSVIALHLLTKFTHI   (88)
6320932   S cere  33 ------------TNSNNSILKIYSD-EATGLRVDPLVVLFLAVGFIFSVVALHVISKVAGK   (82)
```

Example Question: If I give you a bunch of sequences, tell me where they are the same and where they are different.

The Genetic Similarity Between Species

Human ~ Human

99.9%

Human ~ Chimpanzee

96%

Human ~ Cat

90%

Human ~ Cow

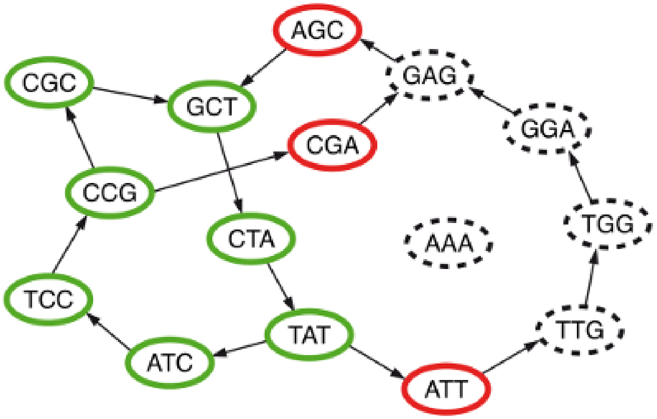
80%

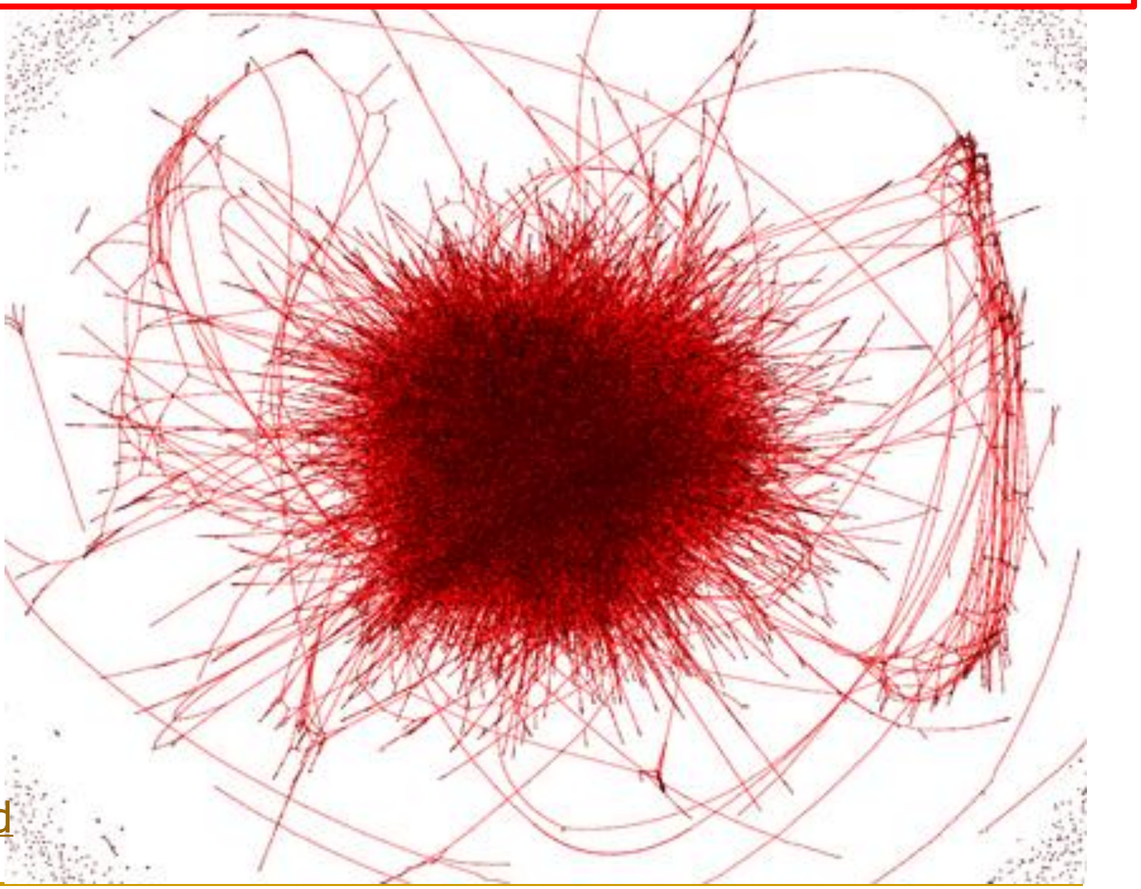
Human ~ Banana

50-60%

Metagenomics, genome assembly, de novo sequencing

Question 2: Given a bunch of short sequences, Can you identify the approximate species cluster for genomically unknown organisms (bacteria)?



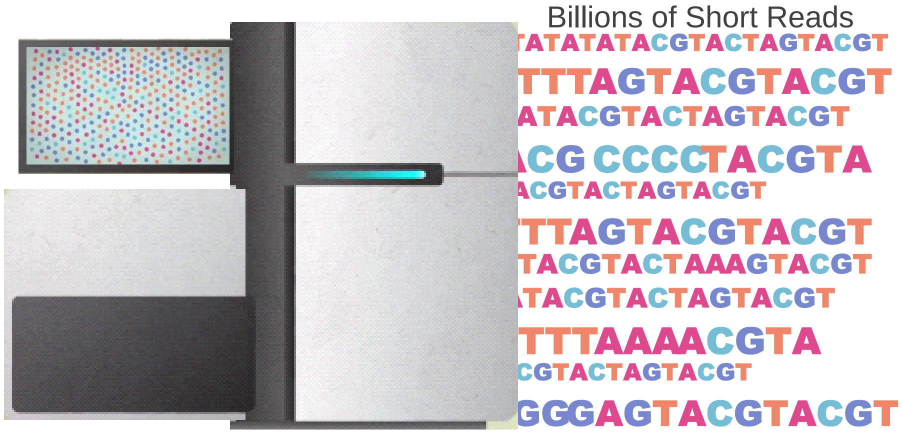


uncleaned de Bruijn graph

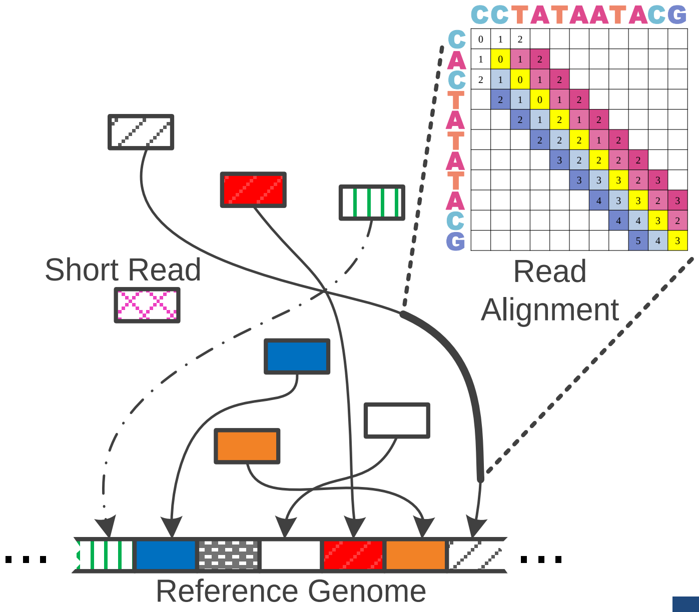
http://math.oregonstate.edu/~koslickd

Problem

Need to construct the entire genome from many reads

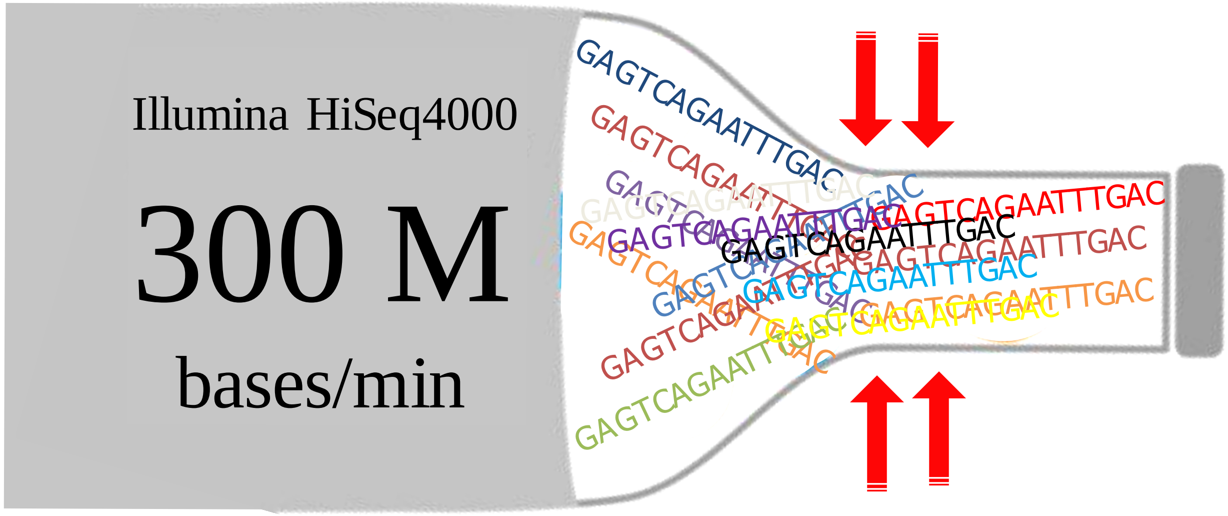


1 Sequencing



Read Mapping 2

Bottlenecked in Mapping!!



Illumina HiSeq4000

300 M

bases/min

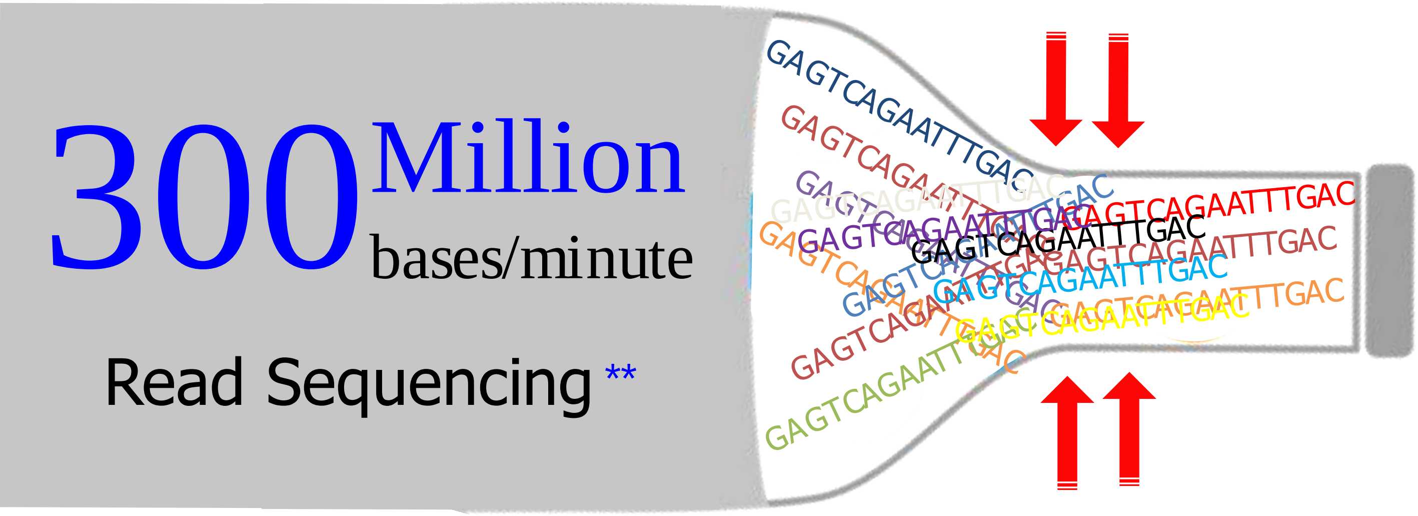
on average

2 M

bases/min

(0.6%)

The Read Mapping Bottleneck



300 Million bases/minute

Read Sequencing **

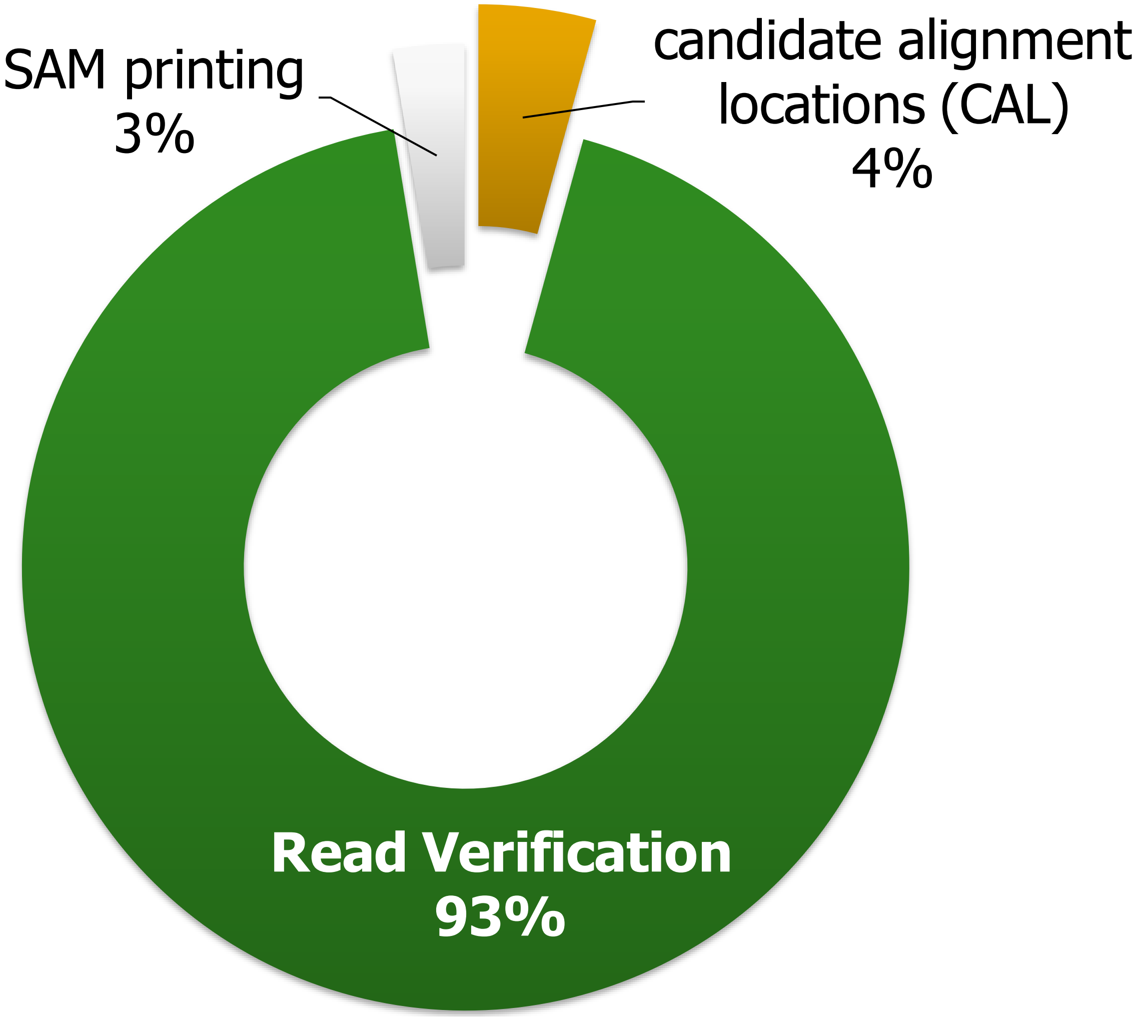
2 Million bases/minute

Read Mapping *

150x slower

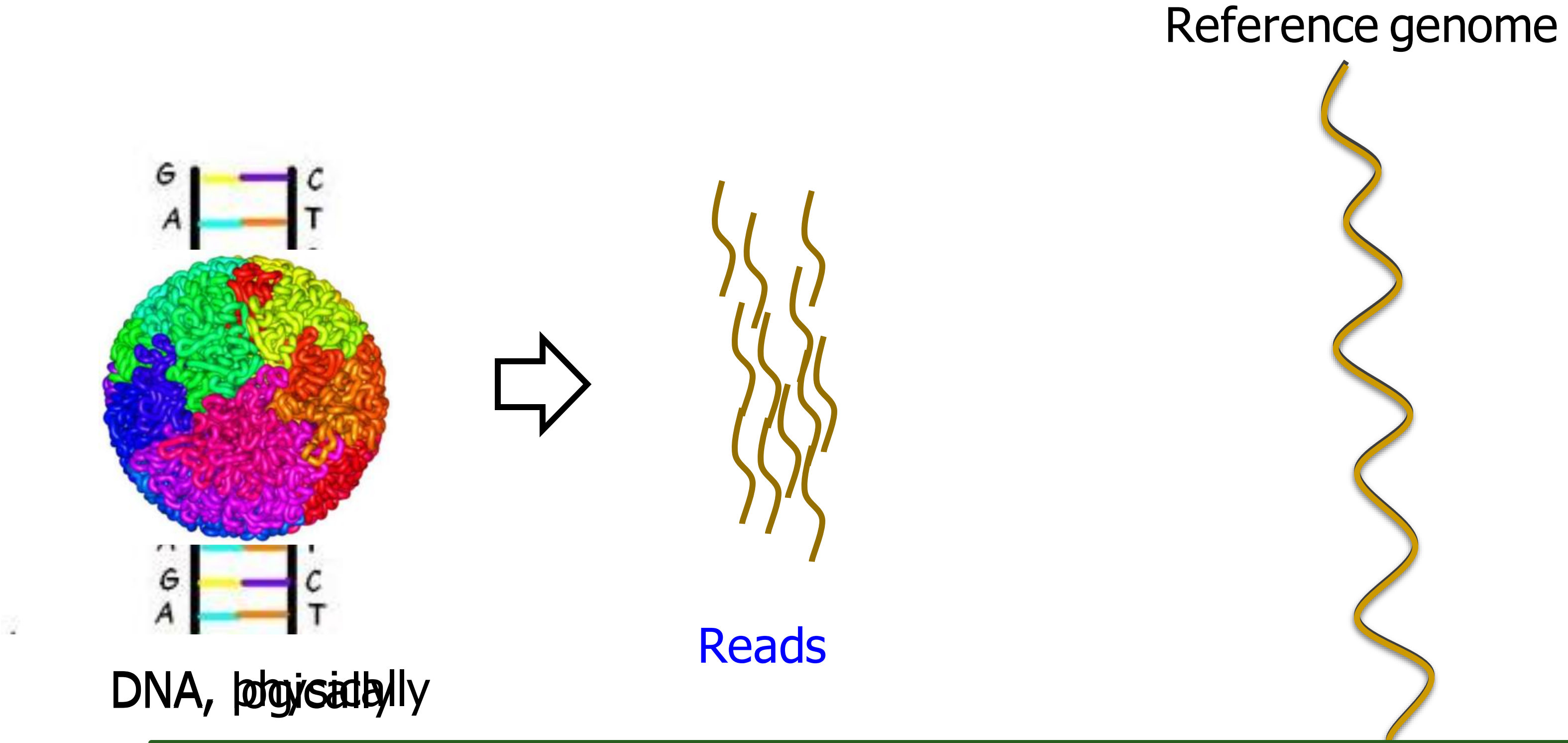
* BWA-MEM
** HiSeqX10, MinION

Read Mapping Execution Time Breakdown



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)

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?

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.

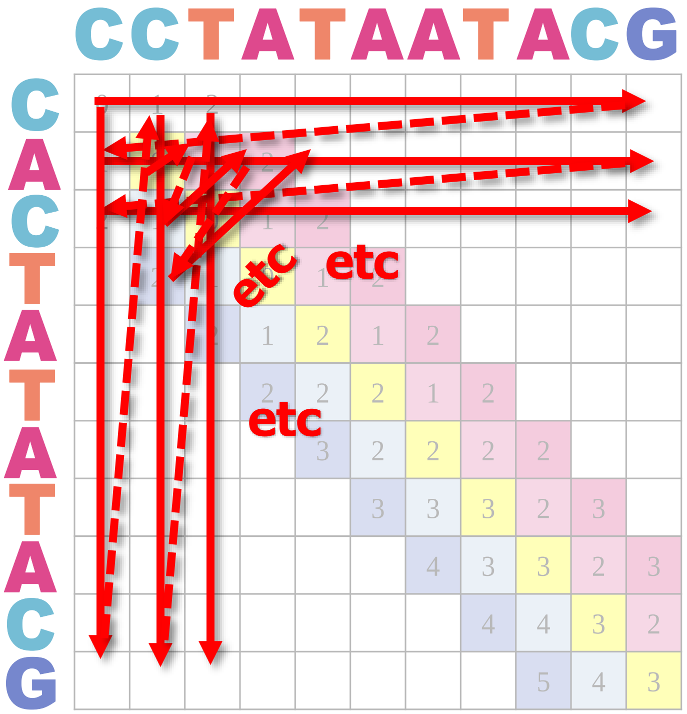
NETHERLANDS x SWITZERLAND

🟩 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.

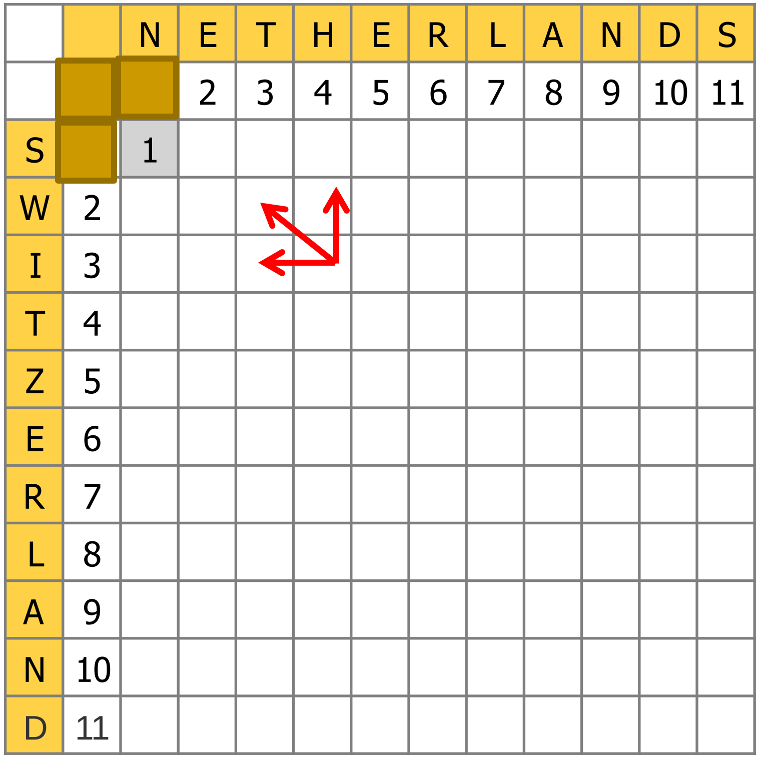


Read Alignment

Example: Dynamic Programming Table

immediate left,
upper left,
upper entries of its own

NETHERLANDS x SWITZERLAND



		N	E	T	H	E	R	L	A	N	D	S
			2	3	4	5	6	7	8	9	10	11
S		1										
W	2											
I	3											
T	4											
Z	5											
E	6											
R	7											
L	8											
A	9											
N	10											
D	11											

Example: Dynamic Programming Table

NETHERLANDS x SWITZERLAND

		N	E	T	H	E	R	L	A	N	D	S
	0	1	2	3	4	5	6	7	8	9	10	11
S	1	1	2	3	4	5	6	7	8	9	10	10
W	2	2	2	3	4	5	6	7	8	9	10	11
I	3	3	3	3	4	5	6	7	8	9	10	11
T	4	4	4	3	4	5	6	7	8	9	10	11
Z	5	5	5	4	4	5	6	7	8	9	10	11
E	6	6	5	5	5	4	5	6	7	8	9	10
R	7	7	6	6	6	5	4	5	6	7	8	9
L	8	8	7	7	7	6	5	4	5	6	7	8
A	9	9	8	8	8	7	6	5	4	5	6	7
N	10	9	9	9	9	8	7	6	5	4	5	6
D	11	10	10	10	10	9	8	7	6	5	4	5

- Matrix-filling is O(mn) time and space.
- Backtrace is O(m + n) time.

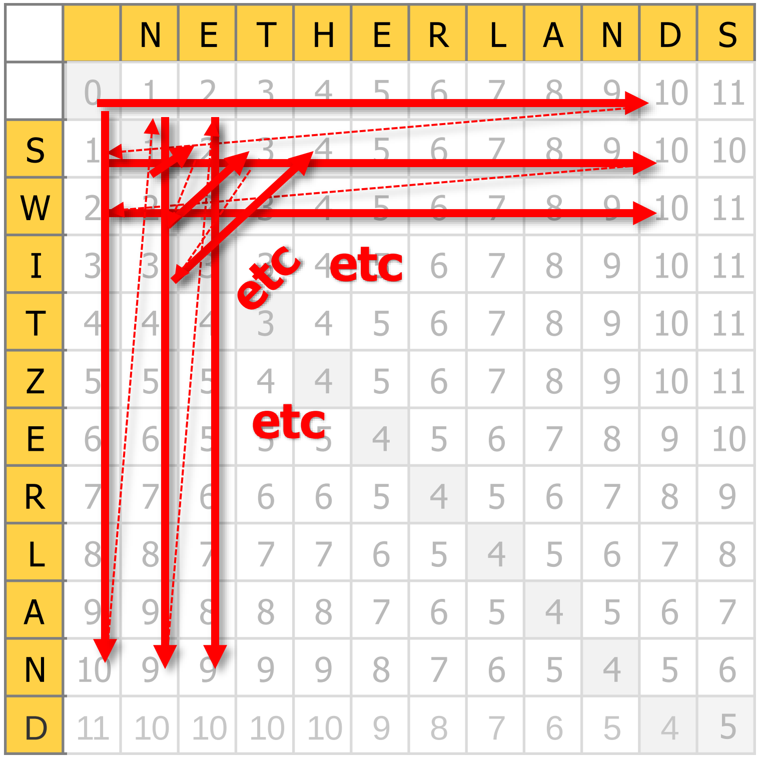
Example: Dynamic Programming

- **Quadratic-time** dynamic-programming algorithm

WHY?!

NETHERLANDS x SWITZERLAND

NETHERLANDS x S
NETHERLANDS x SW
NETHERLANDS x SWI
NETERLANDS x SWIT
NETHERLANDS x SWITZ
NETHERLANDS x SWITZE
NETHERLANDS x SWITZER
NETHERLANDS x SWITZERL
NETHERLANDS x SWITZERLA
NETHERLANDS x SWITZERLAN
NETHERLANDS x SWITZERLAND



		N	E	T	H	E	R	L	A	N	D	S
	0	1	2	3	4	5	6	7	8	9	10	11
S	1	1	2	3	4	5	6	7	8	9	10	10
W	2	2	2	3	4	5	6	7	8	9	10	11
I	3	3	3	3	4	[illegible]	6	7	8	9	10	11
T	4	4	4	3	4	5	6	7	8	9	10	11
Z	5	5	5	4	4	5	6	7	8	9	10	11
E	6	6	5	5	5	4	5	6	7	8	9	10
R	7	7	6	6	6	5	4	5	6	7	8	9
L	8	8	7	7	7	6	5	4	5	6	7	8
A	9	9	8	8	8	7	6	5	4	5	6	7
N	10	9	9	9	9	8	7	6	5	4	5	6
D	11	10	10	10	10	9	8	7	6	5	4	5

Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design

- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions

- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory

- Future Opportunities: New Sequencing Technologies

Read Mapping Algorithms: Two Styles

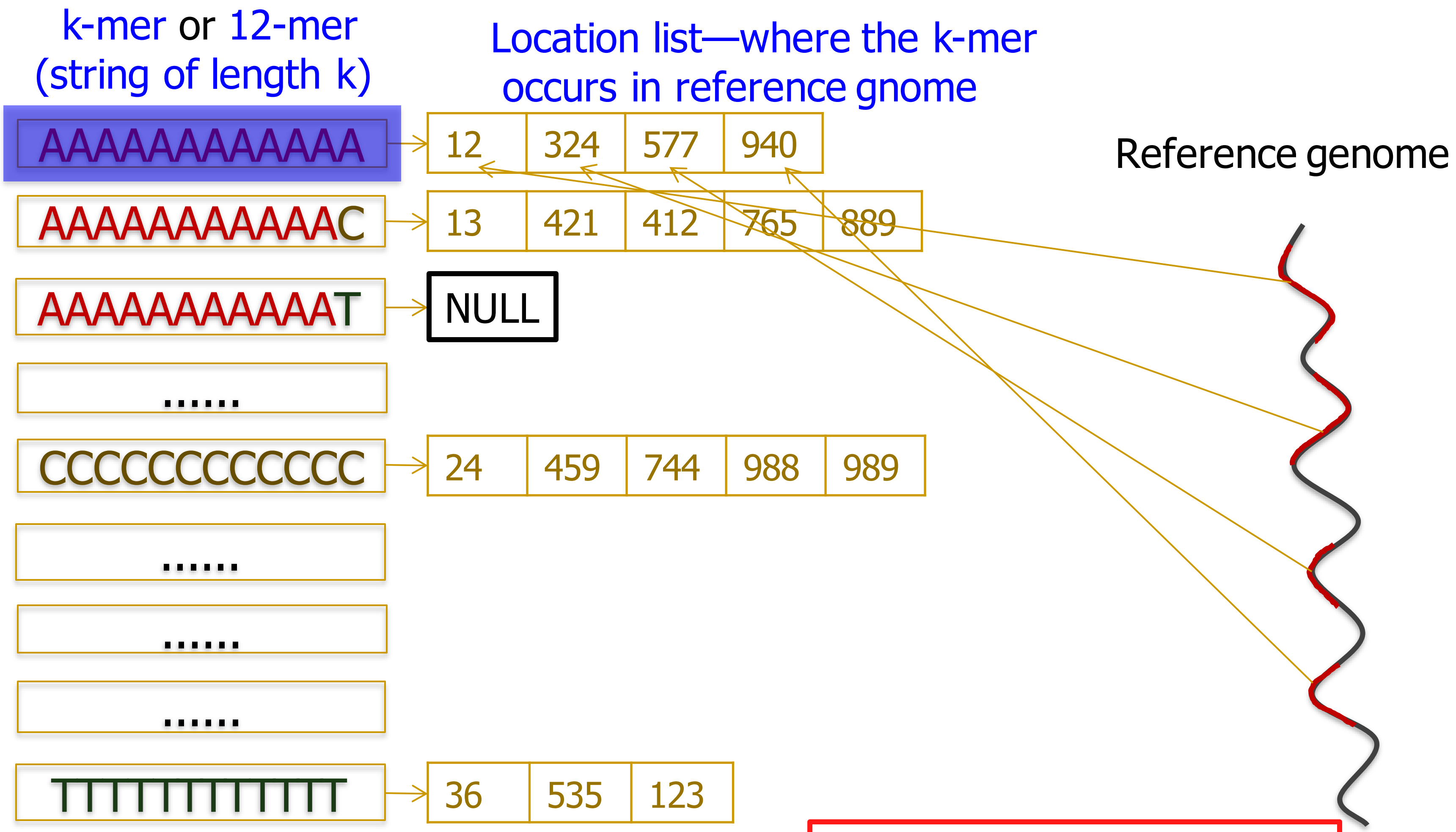
- Hash based seed-and-extend (hash table, suffix array, suffix tree)
  - Index the “k-mers” in the genome into a hash table (pre-processing)
  - When searching a read, find the location of a k-mer in the read; then extend through alignment
  - More sensitive (can find all mapping locations), but slow
  - 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]



k-mer or 12-mer (string of length k)

Location list—where the k-mer occurs in reference gnome

AAAAAAAAAAAA	12	324	577	940	
AAAAAAAAAAAC	13	421	412	765	889
AAAAAAAAAAAT	NULL				
......					
CCCCCCCCCCCC	24	459	744	988	989
......					
......					
......					
TTTTTTTTTTTT	36	535	123		

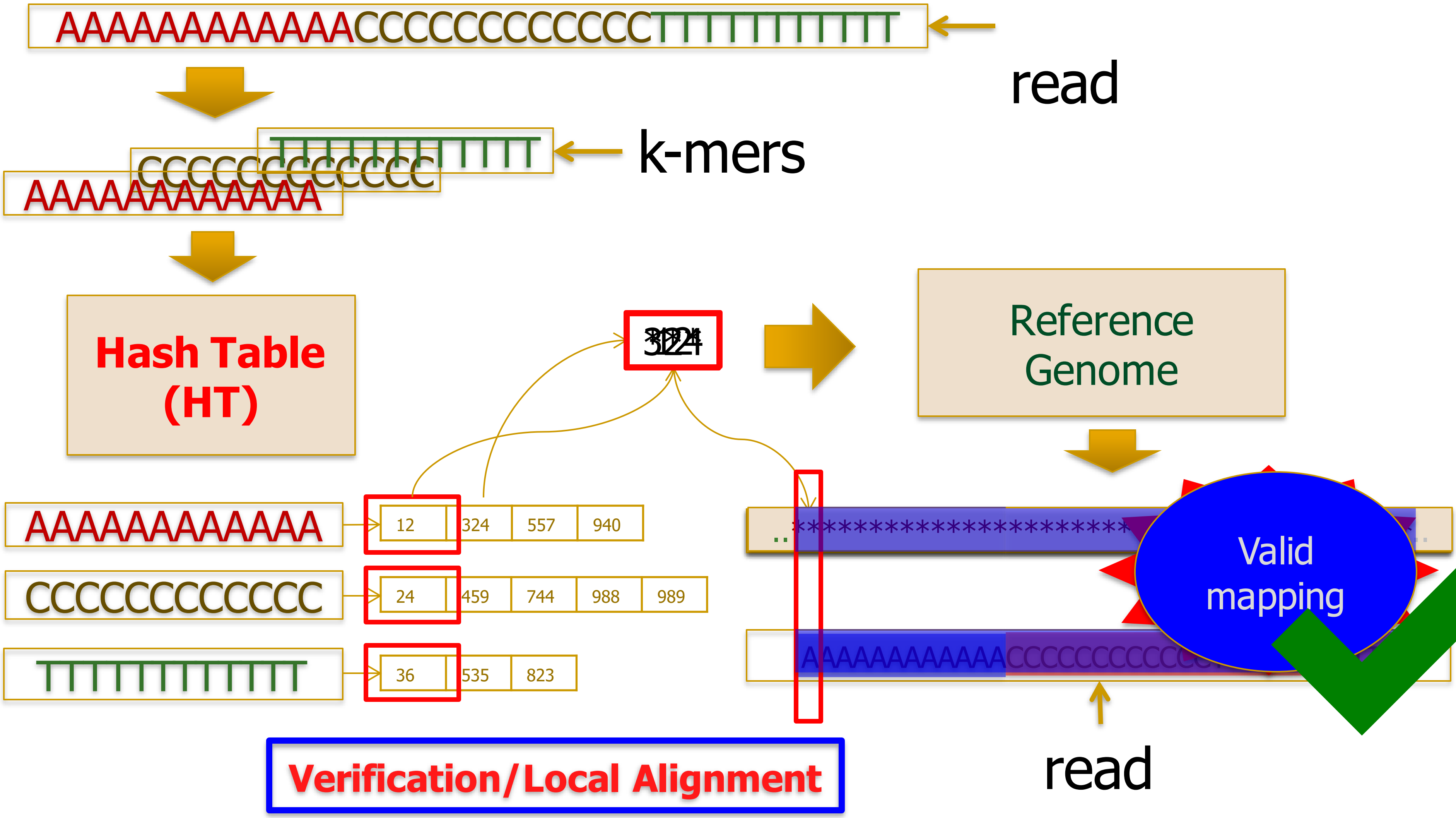
Reference genome

Once for a reference

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]

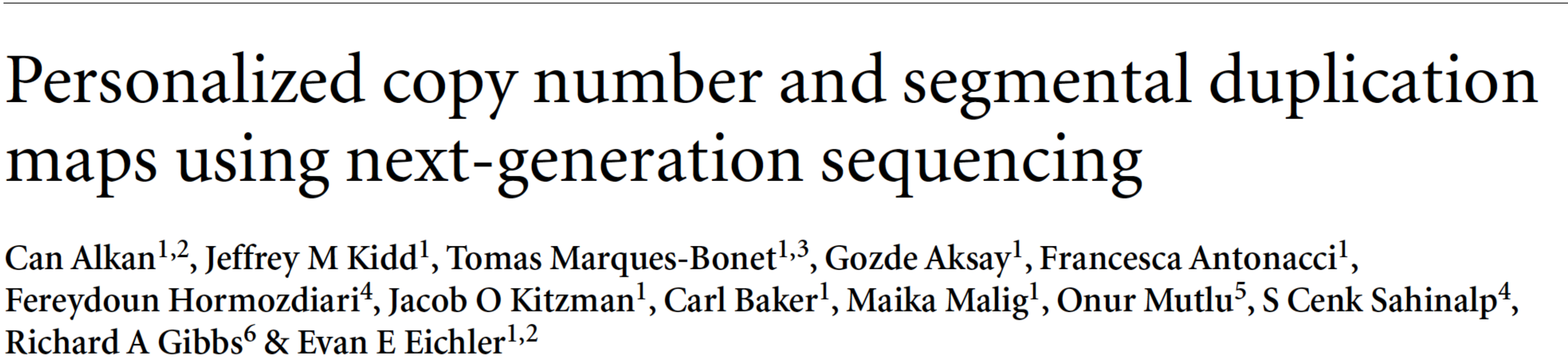


Advantages of Hash Table Based Mappers

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

nature genetics

http://mrfast.sourceforge.net/



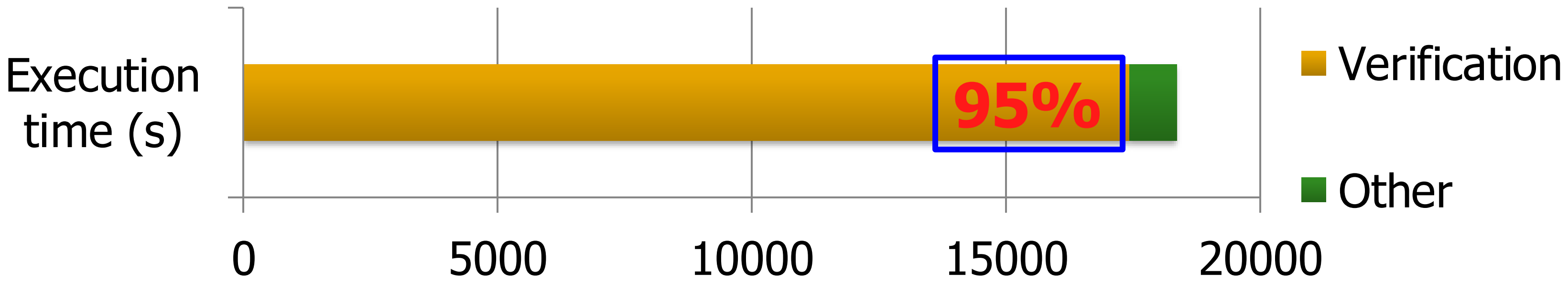
Personalized copy number and segmental duplication maps using next-generation sequencing

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.

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

Overarching Key Idea

Filter fast before you align

Minimize costly
edit distance computations

Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions
- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory
- Future Opportunities: New Sequencing Technologies

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 (filtering)
  - *Reducing* the *number* of potential mappings to examine

Key Observations [Xin+, BMC Genomics 2013]

- Observation 1
  - 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

- Observation 2
  - 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

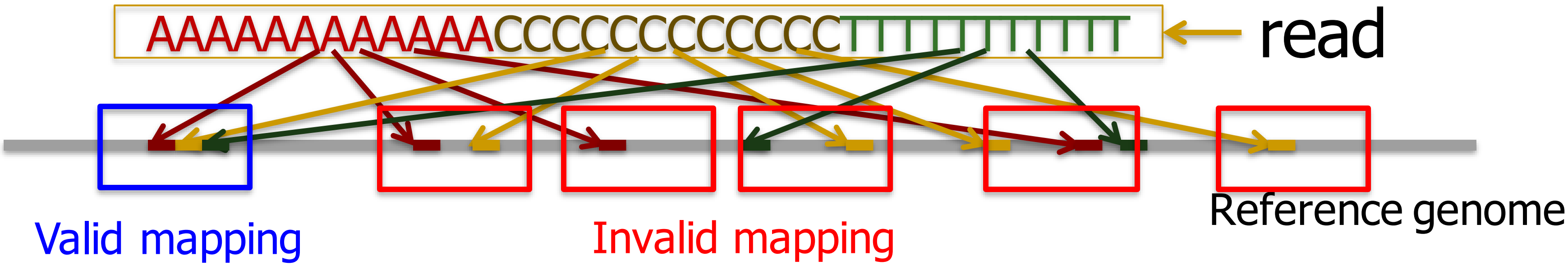
FastHASH Mechanisms [Xin+, BMC Genomics 2013]

- **Adjacency Filtering (AF)**: Rejects obviously invalid mapping locations at early stage to avoid unnecessary verifications

- **Cheap K-mer Selection (CKS):** Reduces the absolute number of potential mapping locations to verify

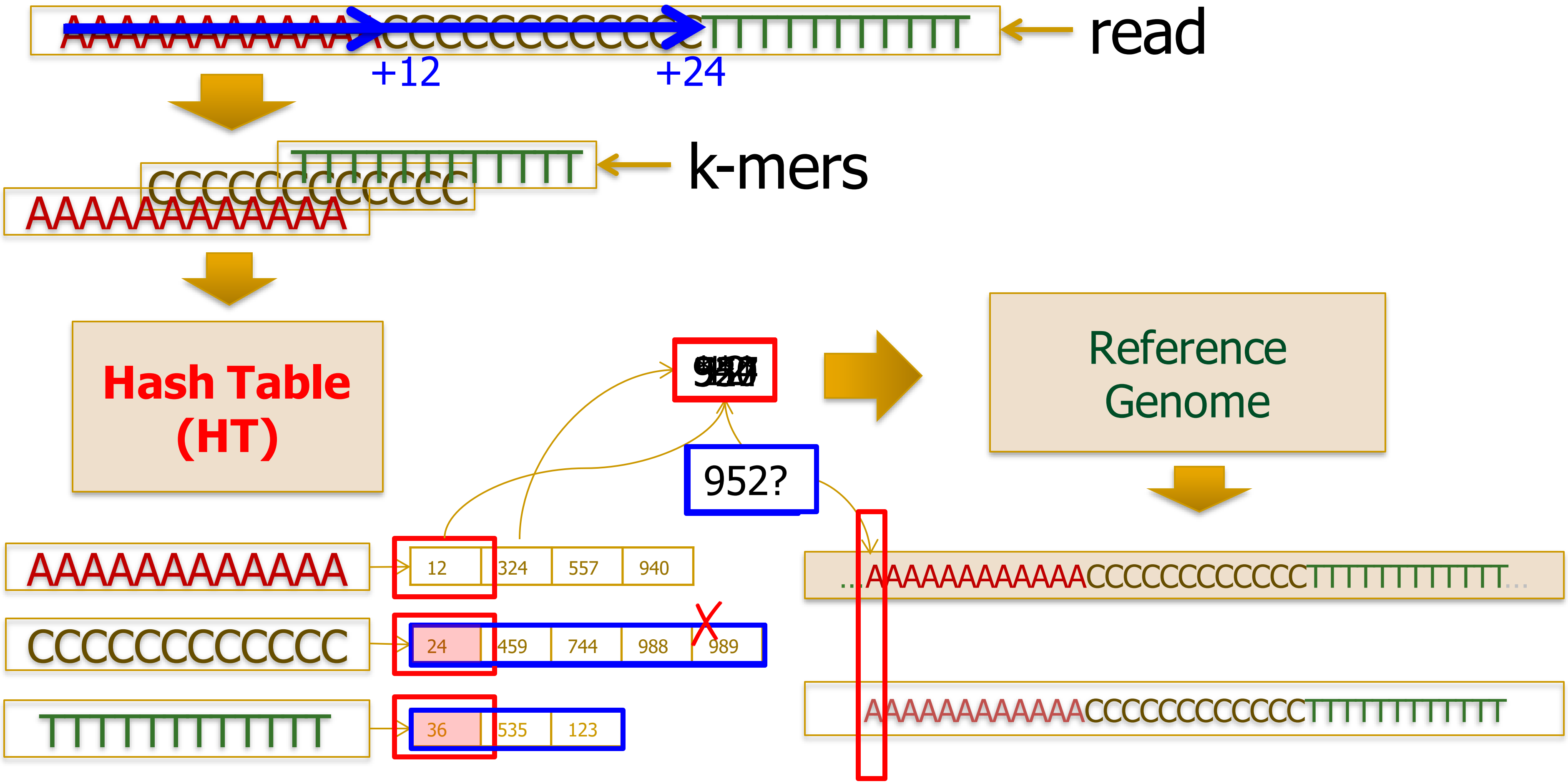
Adjacency Filtering (AF)

- **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)



FastHASH Mechanisms [Xin+, BMC Genomics 2013]

- **Adjacency Filtering (AF)**: Rejects obviously invalid mapping locations at early stage to avoid unnecessary verifications

- **Cheap K-mer Selection (CKS):** Reduces the absolute number of potential mapping locations to verify

Cheap K-mer Selection (CKS)

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

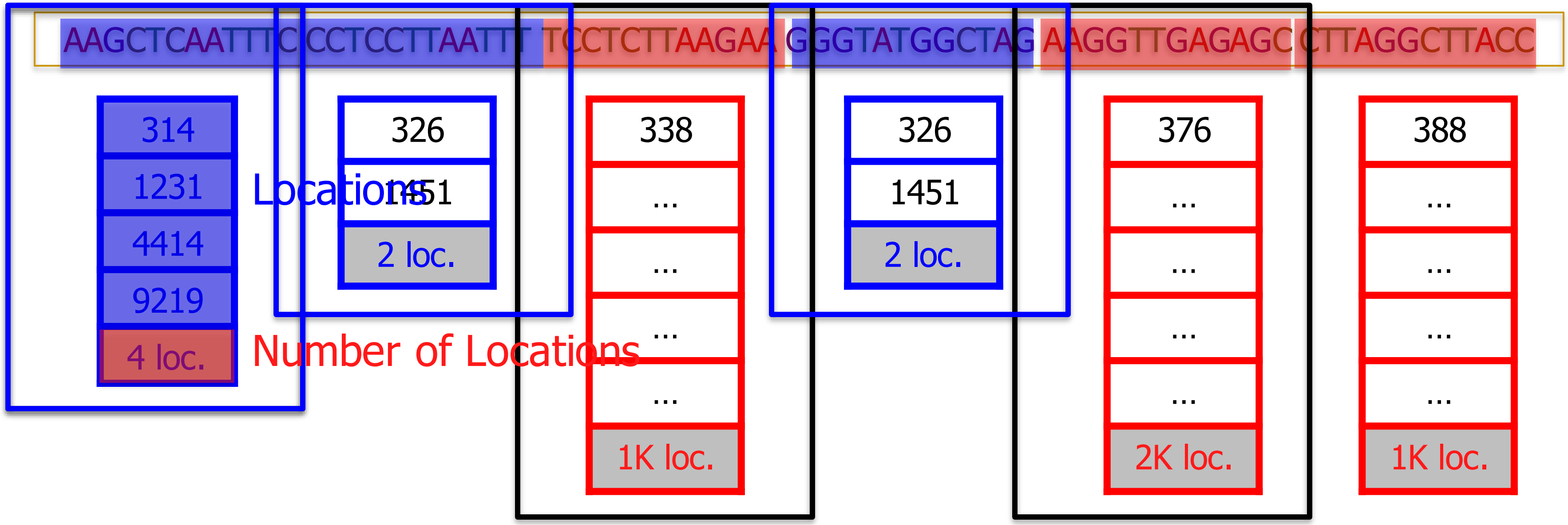
- **Key insight:**
  - K-mers have different cost to examine: Some k-mers are *cheaper* as they have fewer locations than others (occur less frequently in reference genome)

- **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

- *e*=2 (examine 3 k-mers)

read



AAGCTCAATTTC CCTCCTTAATTT TCCTCTTAAGAA GGGTATGGCTAG AAGGTTGAGAGC CTTAGGCTTACC

314	326	338	326	376	388
1231	1451	…	1451	…	…
4414	2 loc.	…	2 loc.	…	…
9219		…		…	…
4 loc.		…		…	…
		1K loc.		2K loc.	1K loc.

Locations

Number of Locations

Expensive 3 k-mers

Cheapest 3 k-mers

Previous work needs to verify:

3004 locations

FastHASH verifies only:

8 locations

Methodology

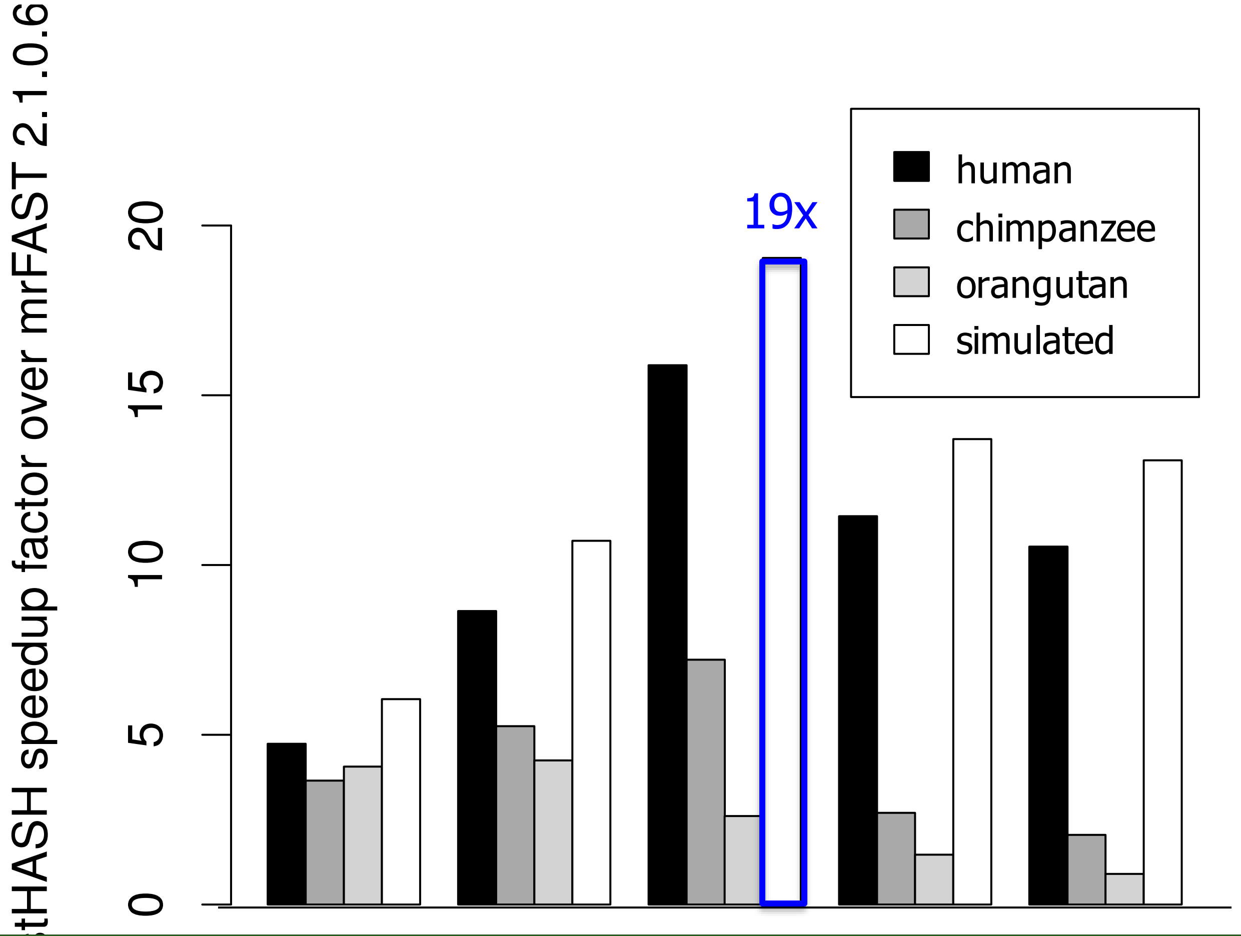
- Implemented FastHASH on top of state-of-the-art mapper: mrFAST
  - New version mrFAST-2.5.0.0 over mrFAST-2.1.0.6

- 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

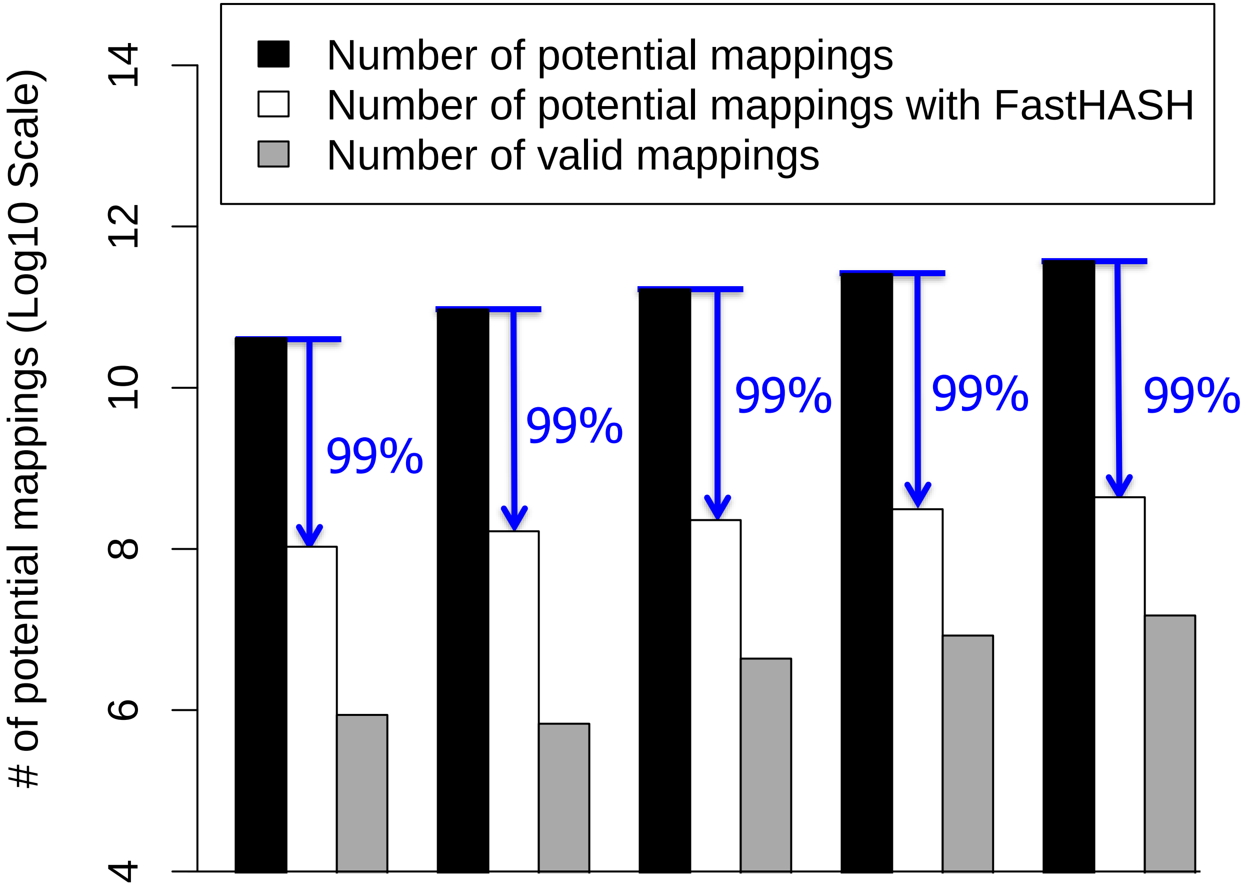
FastHASH Speedup: Entire Read Mapper



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

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

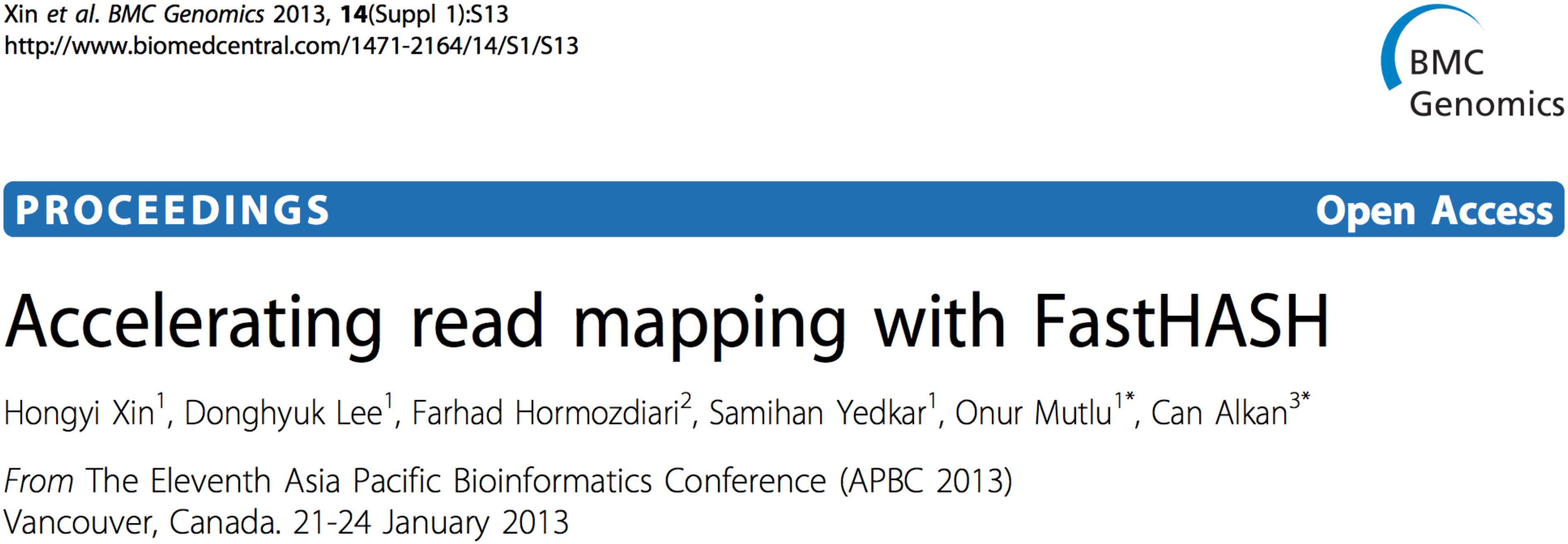
- Observation: Most of the verification calculations are unnecessary → filter them out

- 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

More on FastHASH

- Download source code and try for yourself
  - Download link to FastHASH



Xin *et al. BMC Genomics* 2013, **14**(Suppl 1):S13
http://www.biomedcentral.com/1471-2164/14/S1/S13

BMC Genomics

PROCEEDINGS Open Access

Accelerating read mapping with FastHASH

Hongyi Xin[1], Donghyuk Lee[1], Farhad Hormozdiari[2], Samihan Yedkar[1], Onur Mutlu[1*], Can Alkan[3*]

From The Eleventh Asia Pacific Bioinformatics Conference (APBC 2013)
Vancouver, Canada. 21-24 January 2013

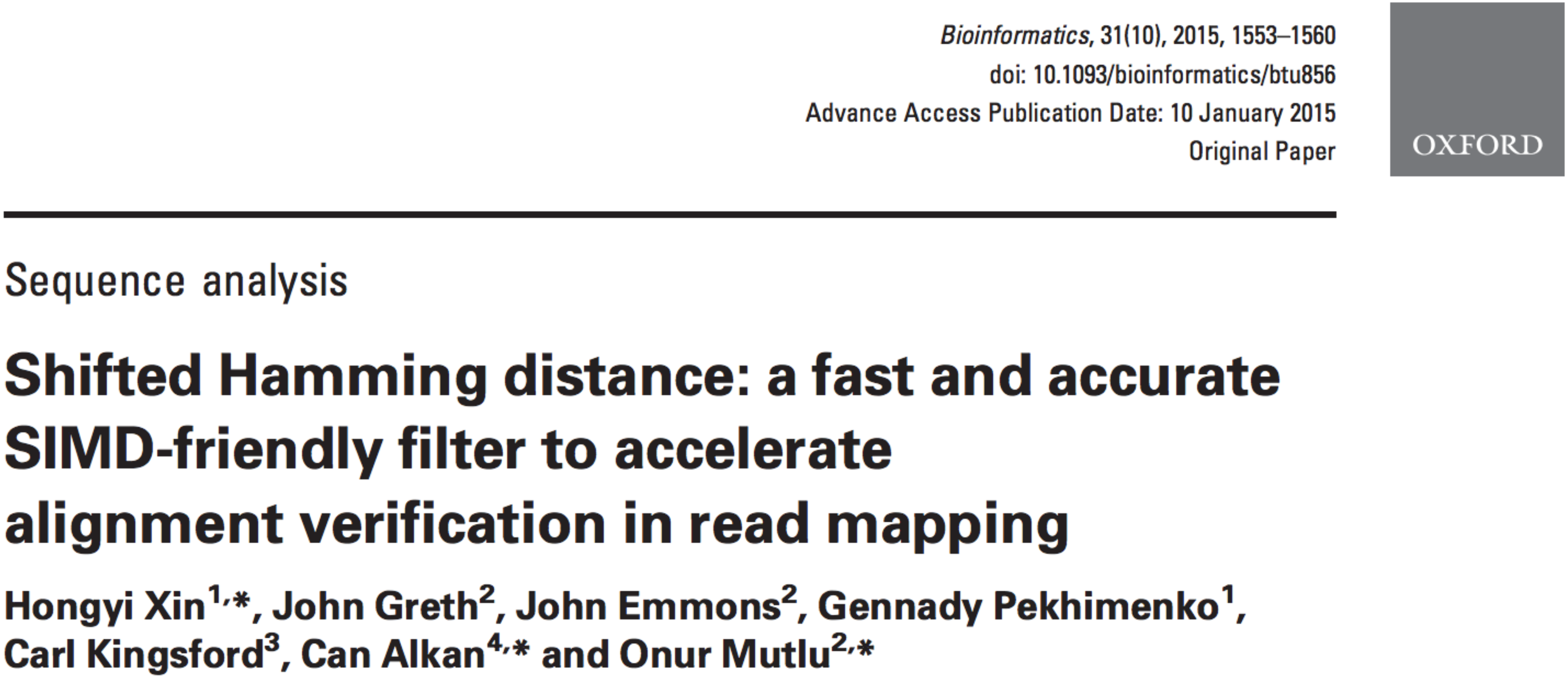
Xin+, **"Accelerating Read Mapping with FastHASH"**, BMC Genomics 2018.

Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions
- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory
- Future Opportunities: New Sequencing Technologies

An Example: Shifted Hamming Distance

https://github.com/CMU-SAFARI/Shifted-Hamming-Distance



Bioinformatics, 31(10), 2015, 1553–1560
doi: 10.1093/bioinformatics/btu856
Advance Access Publication Date: 10 January 2015
Original Paper

OXFORD

Sequence analysis

Shifted Hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping

Hongyi Xin[1,*], John Greth[2], John Emmons[2], Gennady Pekhimenko[1], Carl Kingsford[3], Can Alkan[4,*] and Onur Mutlu[2,*]

Xin+, **"Shifted Hamming Distance: A Fast and Accurate SIMD-friendly Filter to Accelerate Alignment Verification in Read Mapping", Bioinformatics 2015.**

Shifted Hamming Distance

- Key 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"

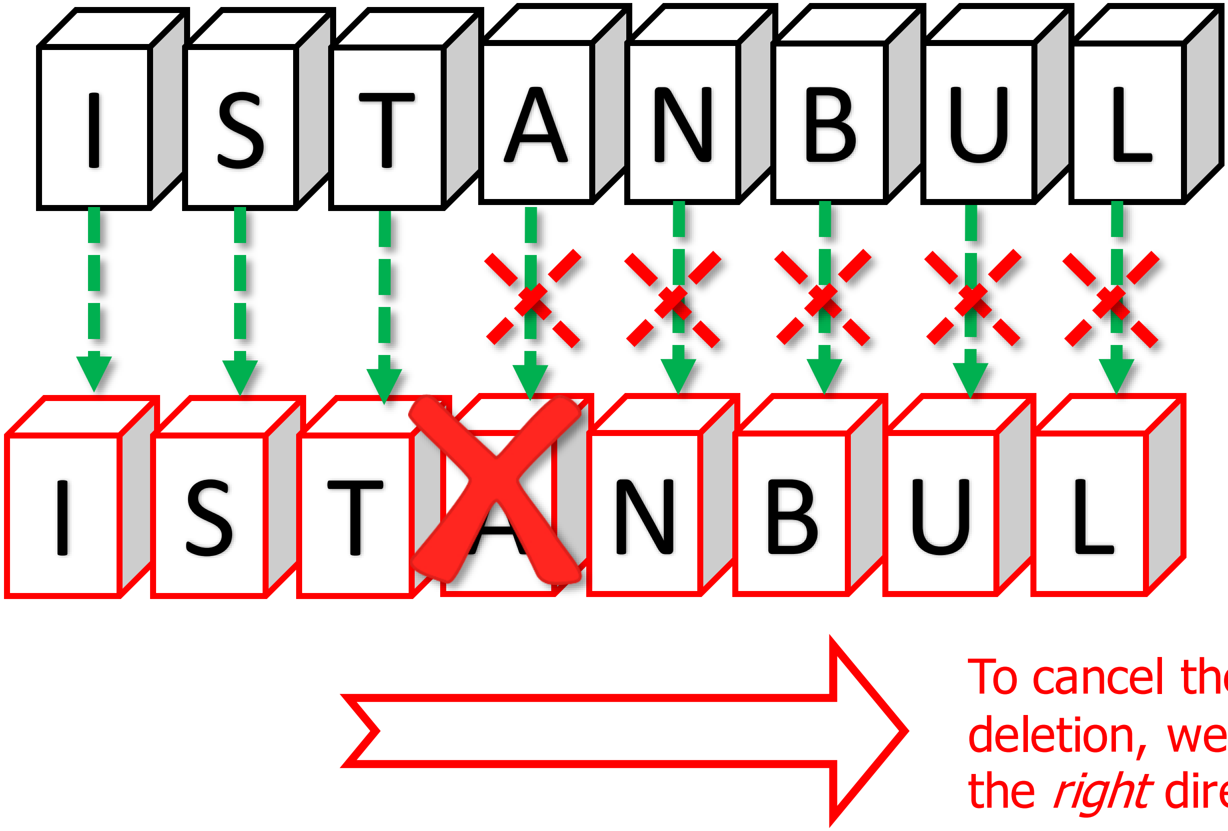
- Key idea:
  - Compute "Shifted Hamming Distance": AND of 2E Hamming Distances of two strings, to filter out invalid mappings
    - Uses bit-parallel operations that nicely map to SIMD instructions

- 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

Hamming Distance (∑⊕)

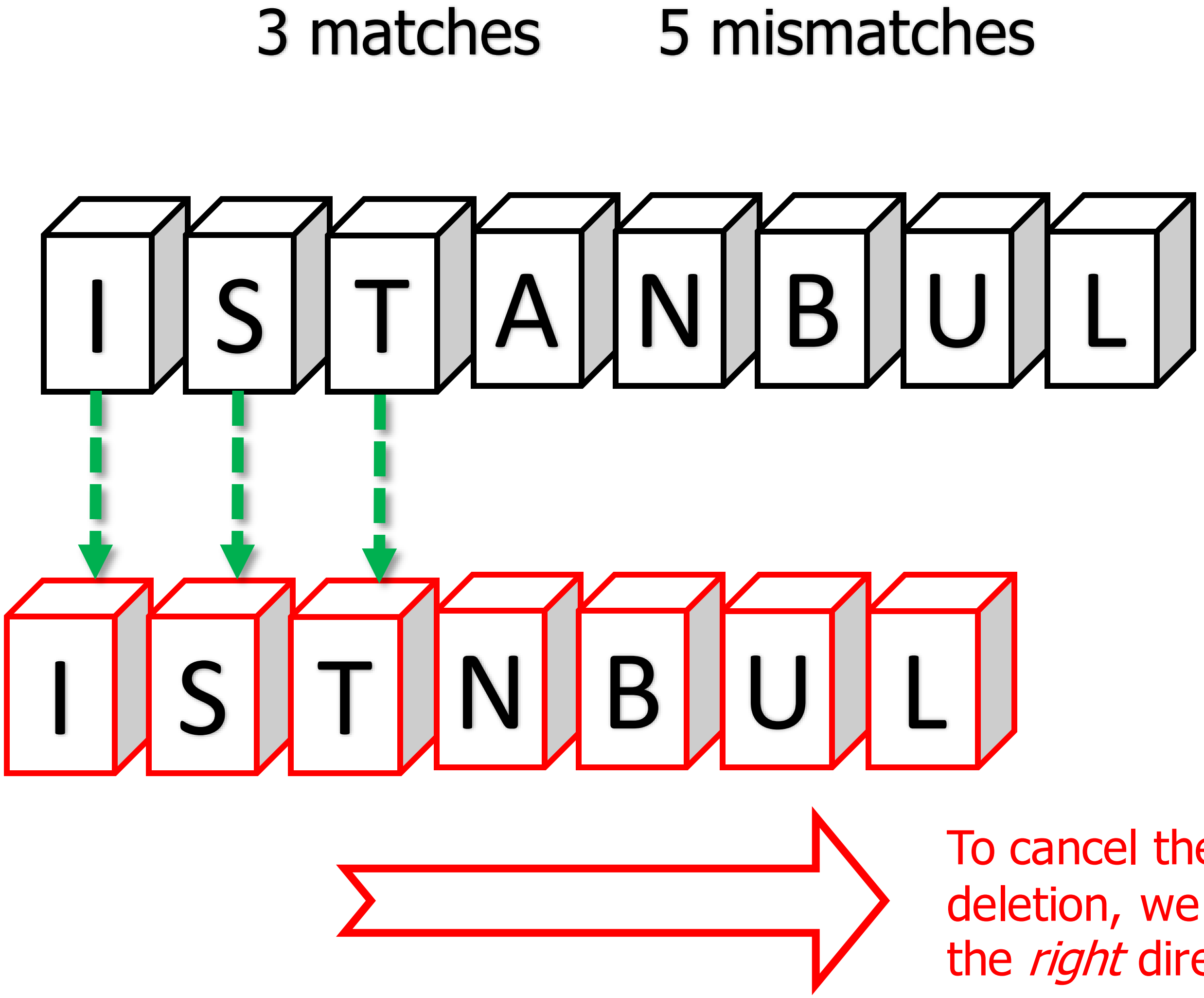
3 matches 5 mismatches

Edit = 1 Deletion



To cancel the effect of a deletion, we need to shift in the *right* direction

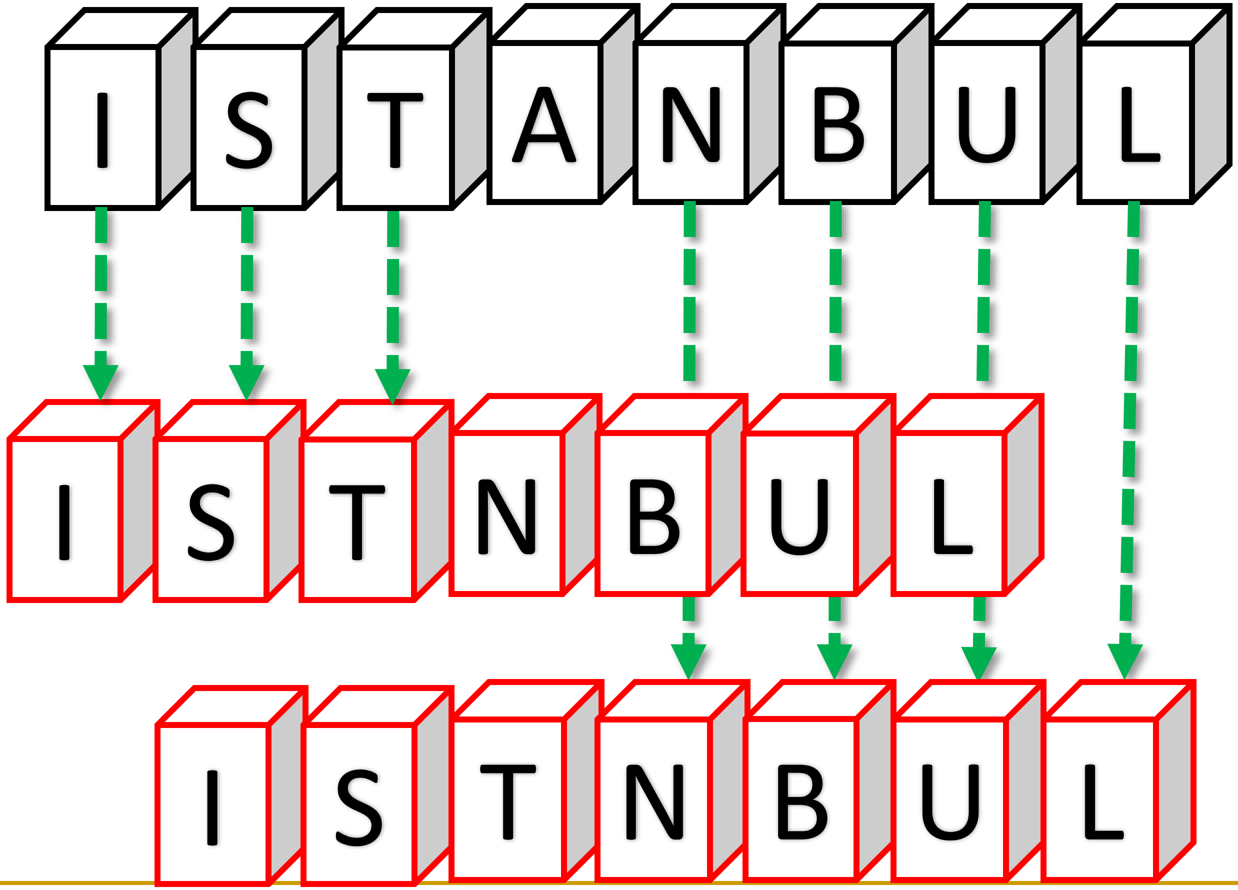
Insight: Shifting a String Helps Similarity Search



To cancel the effect of the deletion, we need to shift in the *right* direction

Insight: Shifting a String Helps Similarity Search

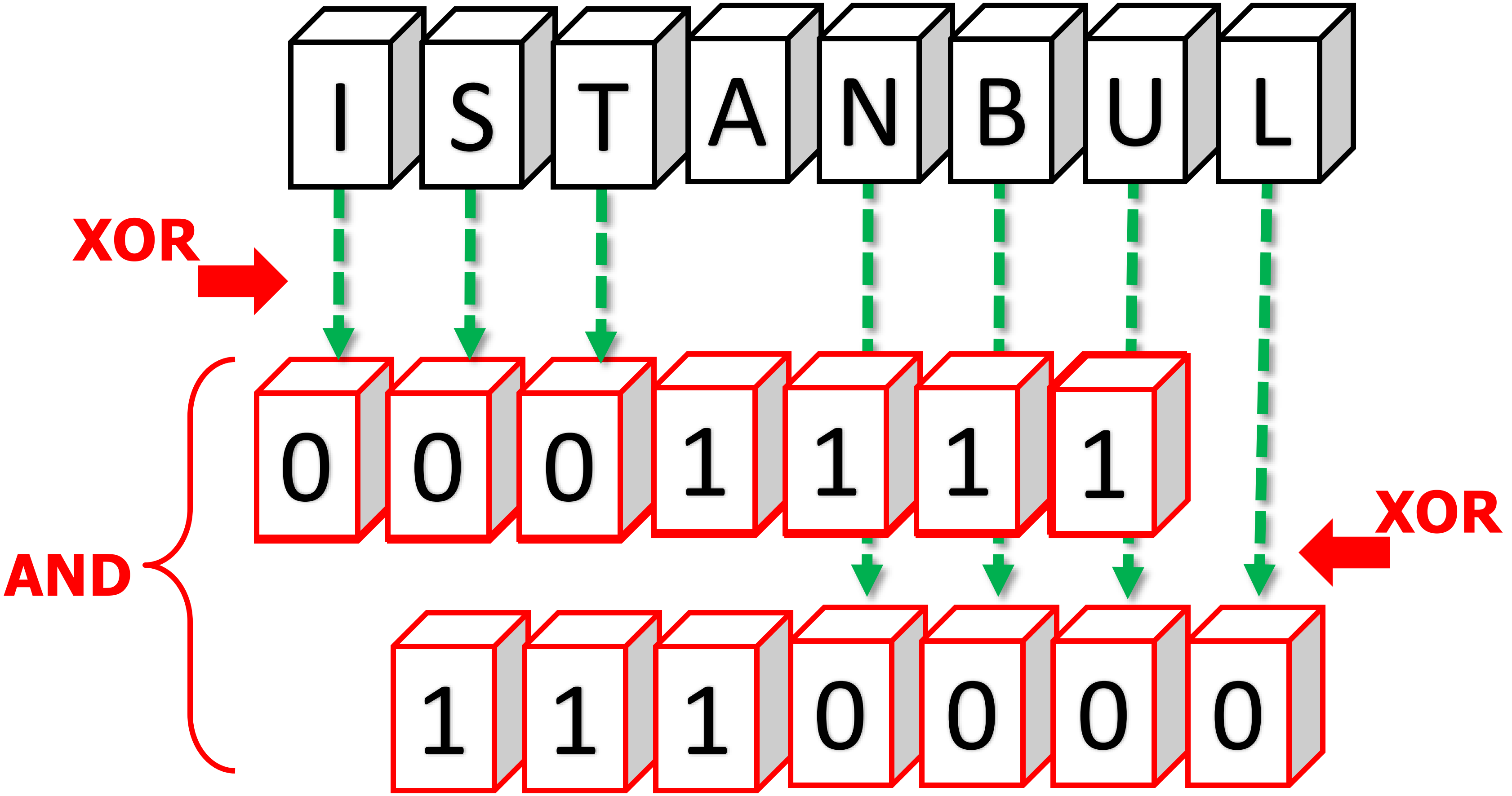
7 matches 1 mismatches



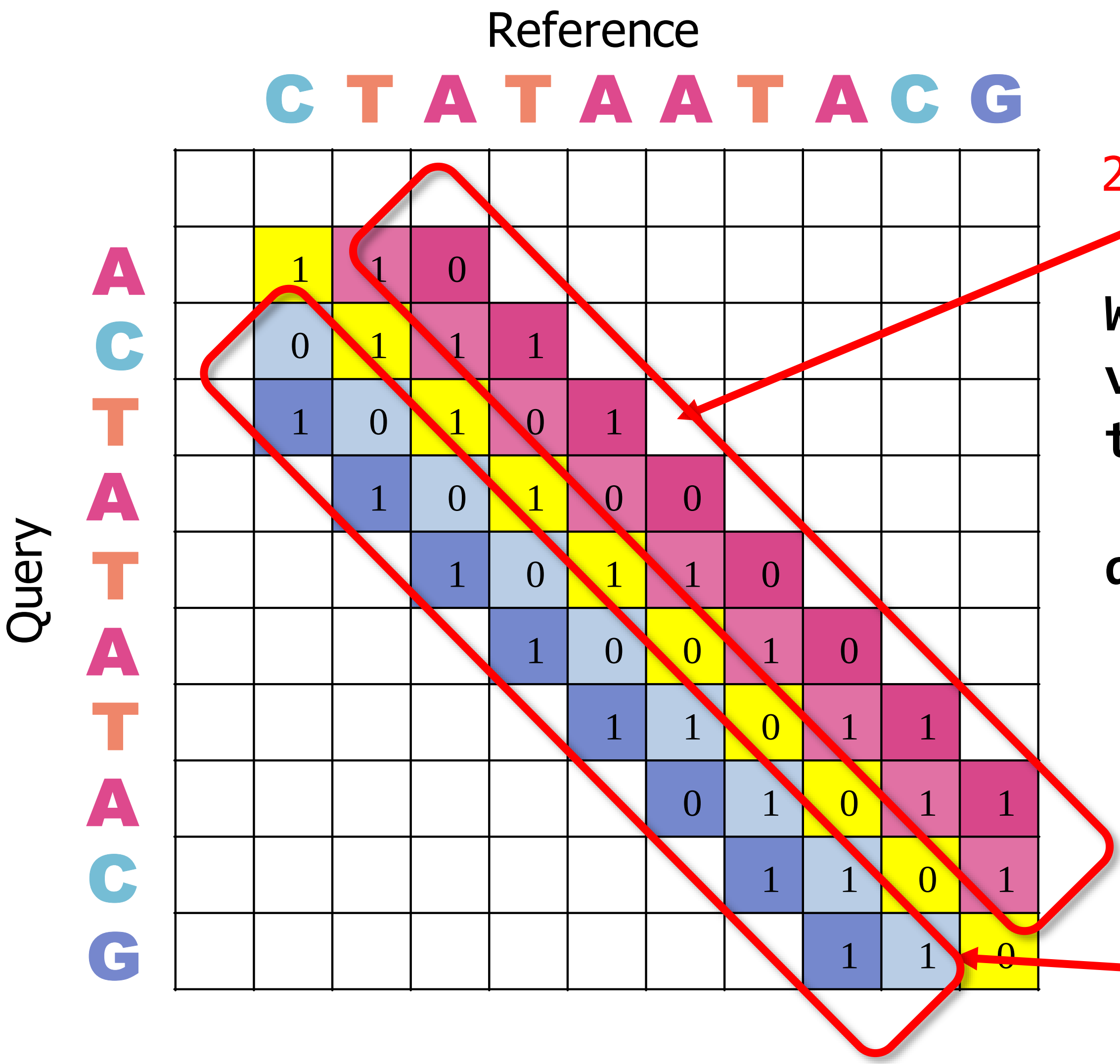
Shifted Hamming Distance

7 matches   1 mismatches

Edit = 1 Deletion



Highly Parallel Matrix Computation



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

2 Deletion Hamming masks

We need to compute 2E+1 vectors, E=edit distance threshold

```
dp[i][j]= 0 if X[i]=Y[j]
          1 if X[i]≠Y[j]
```

No data dependencies!

2 Insertion Hamming masks

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

```
            Query :GAGAGAGATATTTAGTGTTGCAGCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAACATTGTTGGGCCGGA
        Reference :GAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG

     Hamming Mask :00000000001000000000000111111101111100011101101011011111111110001000001111011010010101
  1-Deletion Mask :11111111111001111101111100000000000000000000000000000000000000000000011000000000000000
  2-Deletion Mask :00000000101101110011111111111110111110001110110101101111111111000100010011101101001010
  3-Deletion Mask :11111111111011101100110111011101100010010011111111111111110010110011010110111011101111
  1-Insertion Mask :11111111111011111011111101110110001001001111111111111110010110011000101011101110111110
  2-Insertion Mask :00000010011111001111111110010001101010100110101111111111111110111001111111000111101100
  3-Insertion Mask :11111111011101100110001111111110101101111111100110010111011111111011101111010111001000

                         --- Masks after amendment ---

     Hamming Mask :00000000001000000000000111111111111100011111110111111111111110001000001111111111111111
  1-Deletion Mask :11111111111111111111111100000000000000000000000000000000000000000000011000000000000000
  2-Deletion Mask :00000000111111111111111111111111111110001111111111111111111111000100011111111111111110
  3-Deletion Mask :11111111111111111111111111111111100011111111111111111111111111111111111111111111111111
  1-Insertion Mask :11111111111111111111111111111110001111111111111111111111111111111000111111111111111110
  2-Insertion Mask :00000011111111111111111111110001111111111111111111111111111111111111111100011111110 0
  3-Insertion Mask :11111111111111111110001111111111111111111111111111111111111111111111111111111111111000

         AND Mask :00000000001000000000000100000000000000000000000000000000000000000000000100000000000000 0

Needleman-Wunsch   GAGAGAGATATTTAGTGTTGCAG-CACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAACATTGTTGGGCCGG
        Alignment :|||||||||| |||||||||||| |||||||||||||||||||||||||||||||||||||||||||||::|||||||||||||||
                   GAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG
```

Alignment vs. Pre-alignment (Filtering)

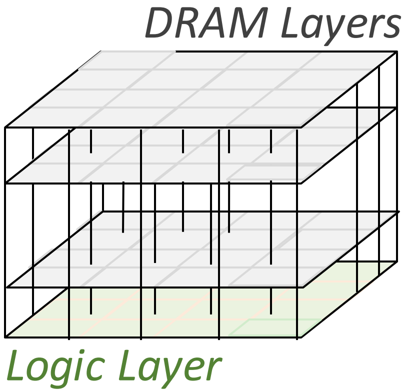
Needleman-Wunsch

		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



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
  - State-of-the-art Read Mapper Design
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions
- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory
- Future Opportunities: New Sequencing Technologies

Location Filtering

- **Alignment** is expensive
  - We need to align millions to billions of reads

- M … t
  f
  1
  1
  2
  out mismatches quickly

Our 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]**

Ideal Filtering Algorithm

Minimal False Accept Rate

Maximal True Reject Rate

Filter out all incorrect mappings

Zero False Reject Rate

Do not filter out any correct mappings

Faster Than Mapper

Alignment vs. Pre-alignment (Filtering)

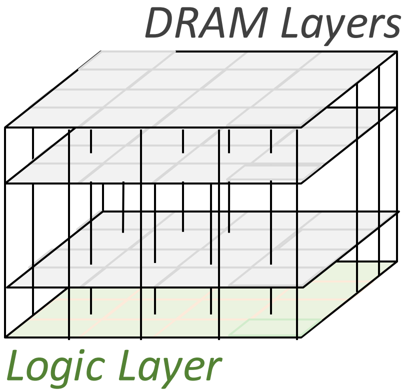
Needleman-Wunsch

		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

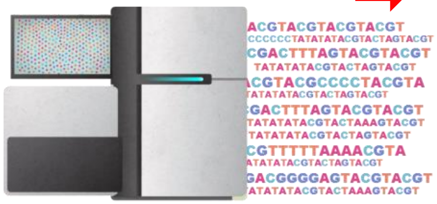


Our Solution: GateKeeper

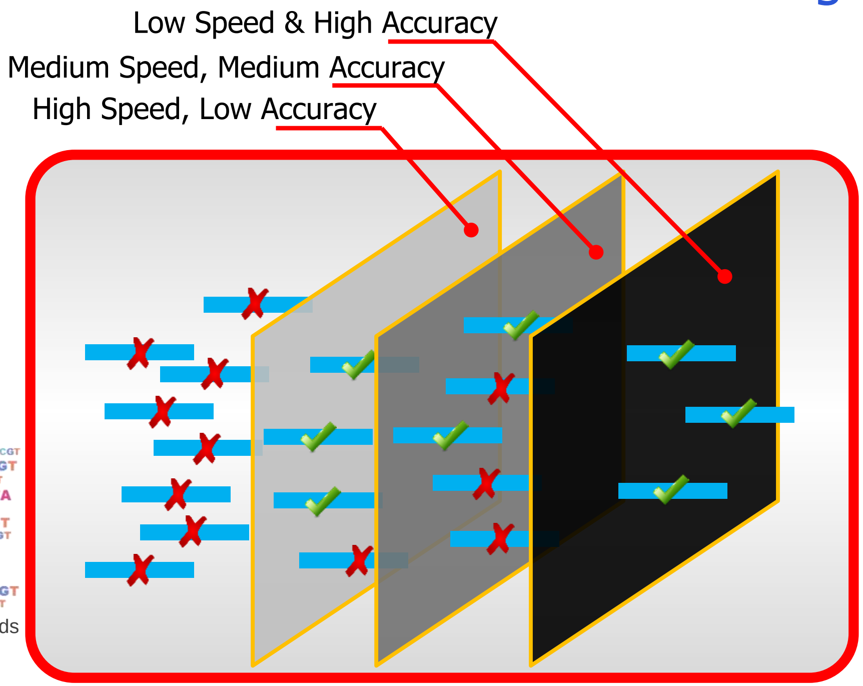
Alignment Filter +

= 1st FPGA-based Alignment Filter.

x10^{12} mappings



Billions of Short Reads



x10^3 mappings

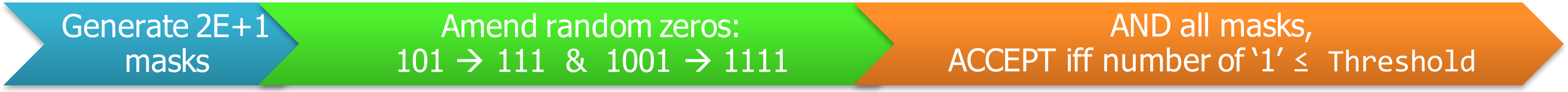
		C	T	A	T	A	A	T	A	C	G
C	0	1	2								
A	1	0	1	2							
C	2	1	0	1	2						
T		2	1	0	1	2					
A			2	1	2	1	2				
T				2	2	2	1	2			
A					3	2	2	2	2		
T						3	3	3	2	3	
A							4	3	3	2	3
C								4	4	3	2
G									5	4	3

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

```
         Query :GAGAGAGATATTTAGTGTTGCAGCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGGAACATTGTTGGGCCGGA
     Reference :GAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG

  Hamming Mask :000000000010000000000001111111011110001110110101101111111110001000001111011010010101
1-Deletion Mask :111111111110011111011111000000000000000000000000000000000000000000011000000000000000
2-Deletion Mask :000000001011011100111111111111101111000111011010110111111111000100010011101101001010
3-Deletion Mask :111111111110111011001101110111011000100100111111111111100101100110010110111011101111
1-Insertion Mask :111111111110111110111111011101100010010011111111111111001011001100010101110111011111 0
2-Insertion Mask :000000100111110011111111100100011010101001101011111111111111011100111111100011101100
3-Insertion Mask :111111110111011001100011111111101011011111100110010111011111111011101111010111001000

                    --- Masks after amendment ---

  Hamming Mask :000000000010000000000001111111111110001111111101111111111110001000001111111111111111
1-Deletion Mask :111111111111111111111111000000000000000000000000000000000000000000011000000000000000
2-Deletion Mask :000000001111111111111111111111111111000111111111111111111111000100011111111111111110
3-Deletion Mask :111111111111111111111111111111111000111111111111111111111111111111111111111111111111
1-Insertion Mask :111111111111111111111111111111100011111111111111111111111111100011111111111111111 0
2-Insertion Mask :000000111111111111111111111100011111111111111111111111111111111111111111100011111100
3-Insertion Mask :111111111111111111100011111111111111111111111111111111111111111111111111111111111000

      AND Mask :000000000010000000000001000000000000000000000000000000000000000000001000000000000000

                 GAGAGAGATATTTAGTGTTGCAG-CACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGGAACATTGTTGGGCCGG
Needleman-Wunsch
      Alignment :|||||||||| |||||||||||| |||||||||||||||||||||||||||||||||||||||||||::|||||||||||||||
                 GAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCCGG
```

GateKeeper Walkthrough (cont'd)

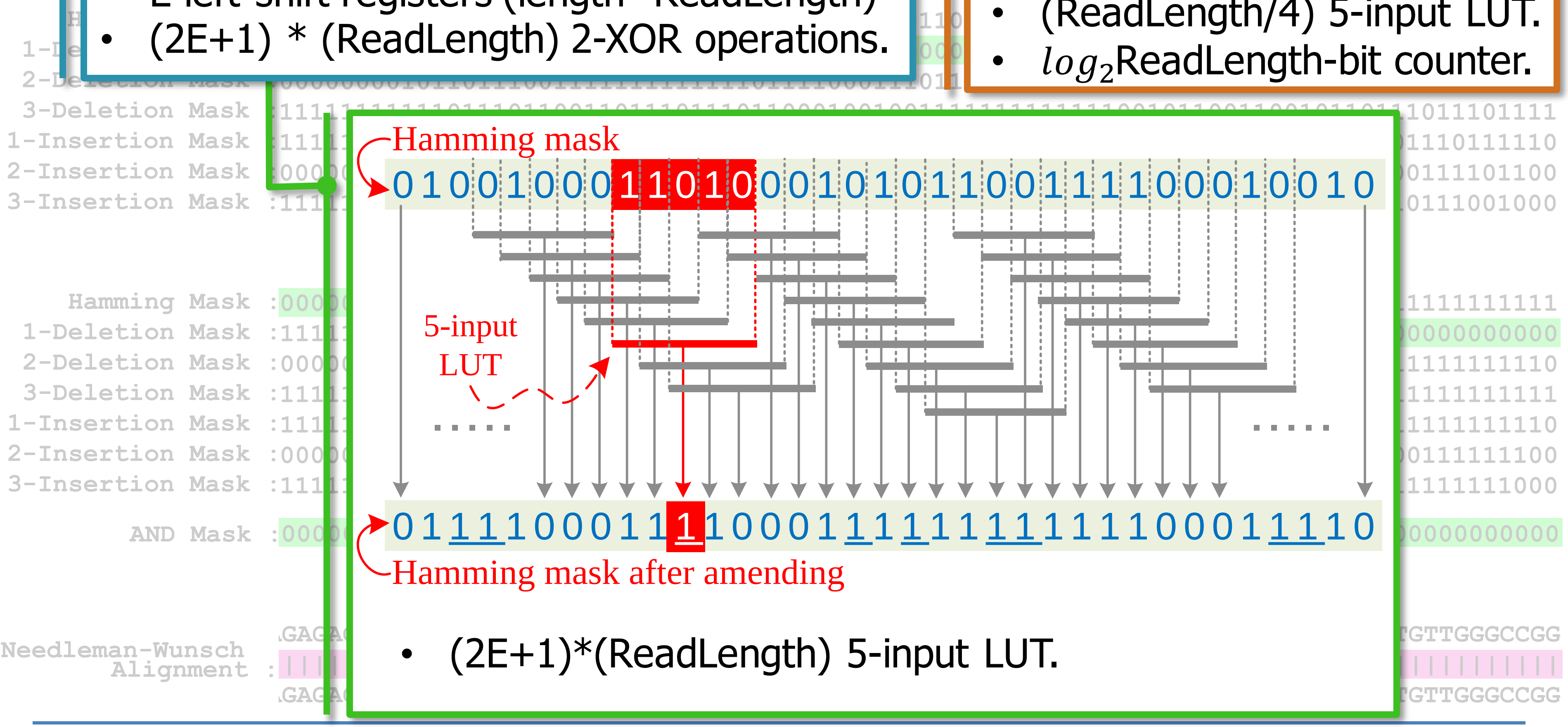
Generate 2E+1 masks

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.

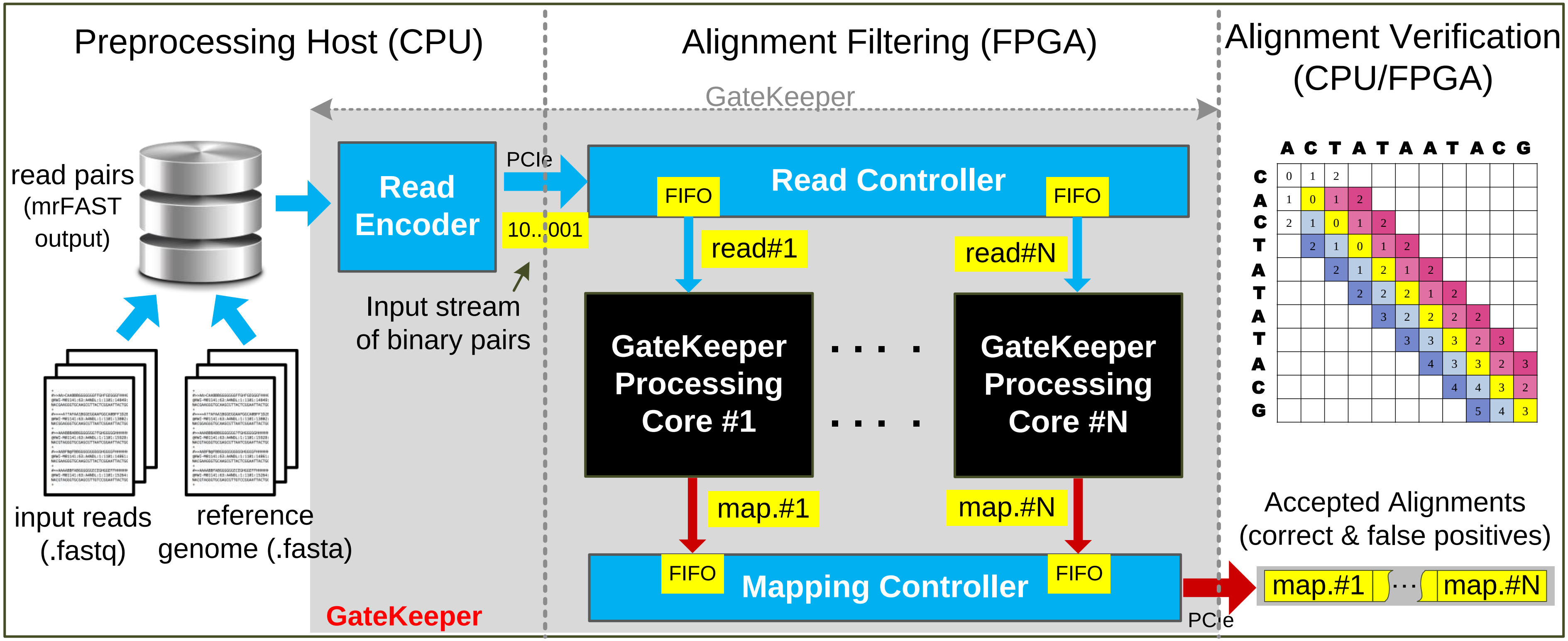
- (2E)*(ReadLength) 2-AND operations.
- (ReadLength/4) 5-input LUT.
- log_2ReadLength-bit counter.



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

GateKeeper Accelerator Architecture

- **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



Preprocessing Host (CPU)
Alignment Filtering (FPGA)
Alignment Verification (CPU/FPGA)
GateKeeper
read pairs (mrFAST output)
input reads (.fastq)
reference genome (.fasta)
Read Encoder
PCIe
10..001
Input stream of binary pairs
Read Controller
FIFO
FIFO
read#1
read#N
GateKeeper Processing Core #1
. . . .
. . . .
GateKeeper Processing Core #N
map.#1
map.#N
FIFO
FIFO
Mapping Controller
PCIe
GateKeeper
Accepted Alignments (correct & false positives)
map.#1
…
map.#N

	A	C	T	A	T	A	A	T	A	C	G
C	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		
T						3	3	3	2	3	
A							4	3	3	2	3
C								4	4	3	2
G									5	4	3

GateKeeper vs. SHD

GateKeeper

- FPGA (Xilinx VC709)
- Multi-core (parallel)
- Examines a single mapping @ 125 MHz
- Limited to PCIe Gen3(4x) transfer rate (128 bits @ 250MHz)
- Amending requires:
  - (2E+1) 5-input LUT.

SHD

- Intel SIMD
- Single-core (sequential)
- Examines a single mapping @ ~2MHz
- Limited to a read length of 128 bp (SSE register size)
- Amending requires:
  - 4(2E+1) bitwise OR.
  - 4(2E+1) packed shuffle.
  - 3(2E+1) shift.

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

Conclusions

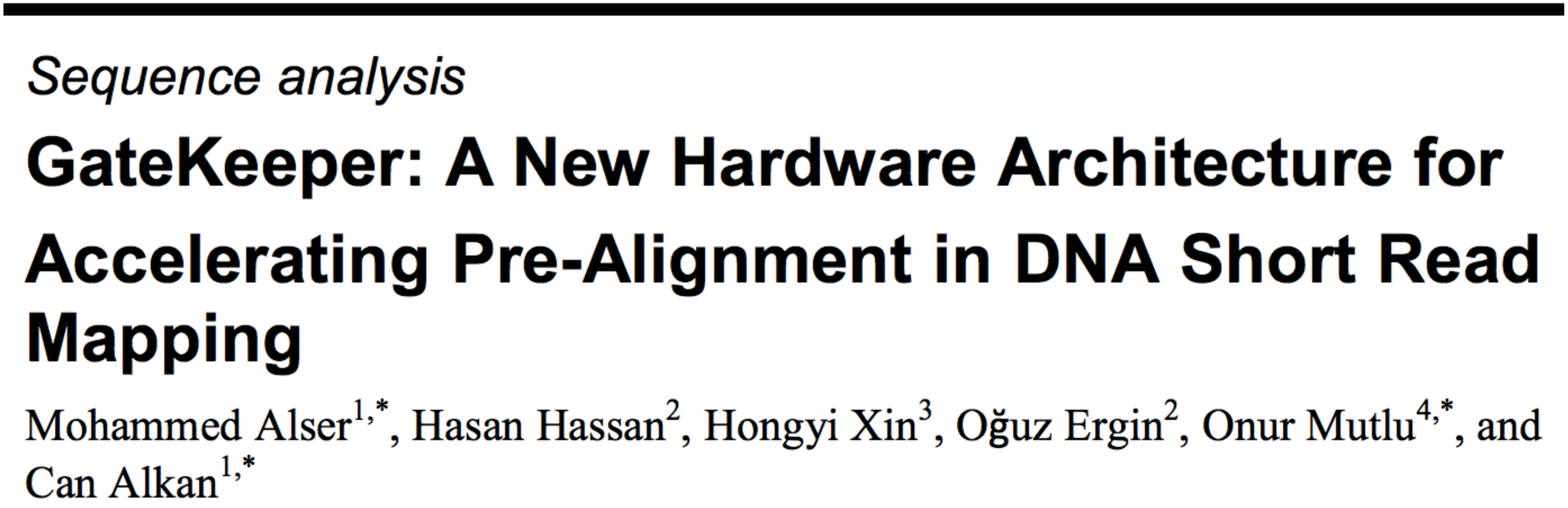
- FPGA-based pre-alignment greatly speeds up read mapping
  - 10x speedup of a state-of-the-art mapper (mrFAST)

- 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

More on GateKeeper

- Download and test for yourself
  https://github.com/BilkentCompGen/GateKeeper

Alser+, **"GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping"**, Bioinformatics, 2017.



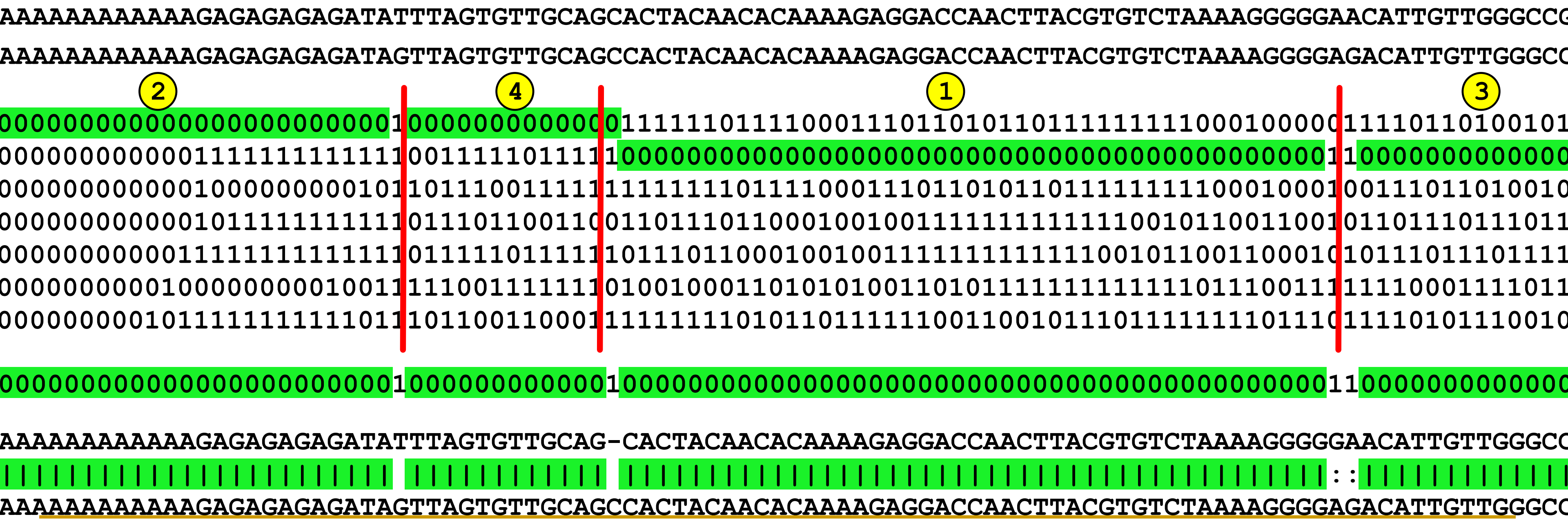
Sequence analysis

GateKeeper: A New Hardware Architecture for Accelerating Pre-Alignment in DNA Short Read Mapping

Mohammed Alser[1,*], Hasan Hassan[2], Hongyi Xin[3], Oğuz Ergin[2], Onur Mutlu[4,*], and Can Alkan[1,*]

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.

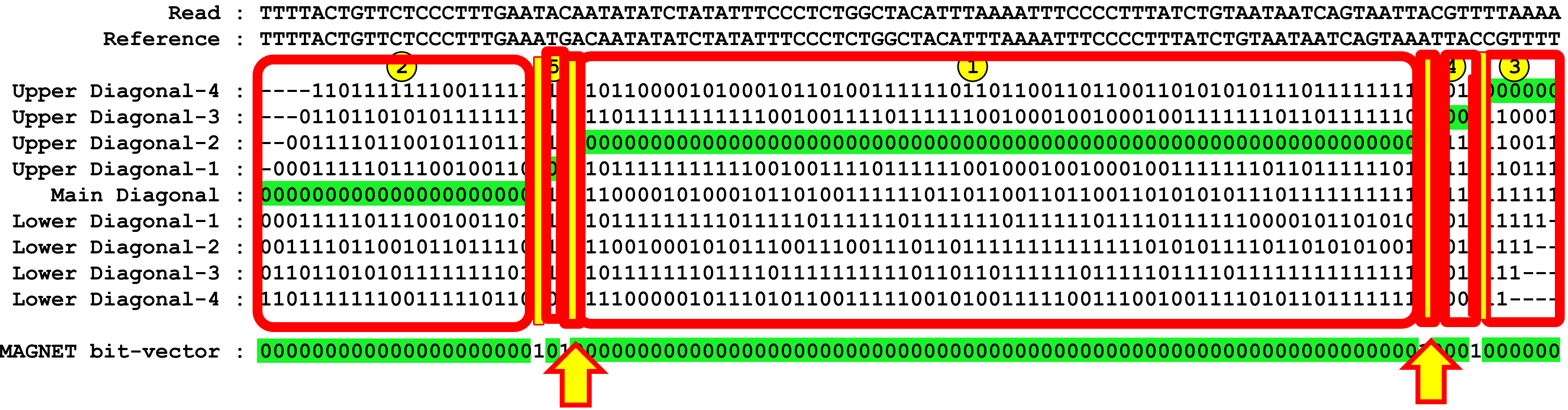


```
AAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTGCAGCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAACATTGTTGGCCG
AAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCC
          2                     4                        1                            3
00000000000000000000000001000000000000001111110111100011101101011011111111100010000011110110100101
00000000000001111111111110011111011111100000000000000000000000000000000000000000000011000000000000
000000000000100000000010110111001111111111111011110001110110101101111111110001000100111011010010
000000000000101111111111101110110011001101110110001001001111111111111001011001100101101110111011
000000000000111111111111101111101111110111011000100100111111111111100101100110001010111011101111
00000000001000000000010011111100111111101001000110101010011010111111111111110111001111111000111011
00000000010111111111111011101100110001111111110101101111110011001011101111111110111011110101110010

00000000000000000000000001000000000000001000000000000000000000000000000000000000000000011000000000000

AAAAAAAAAAAAGAGAGAGAGATATTTAGTGTTGCAG-CACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAACATTGTTGGGCC
|||||||||||||||||||||||| ||||||||||||| |||||||||||||||||||||||||||||||||||||||||||||::|||||||||||||
AAAAAAAAAAAAGAGAGAGAGATAGTTAGTGTTGCAGCCACTACAACACAAAAGAGGACCAACTTACGTGTCTAAAAGGGGAGACATTGTTGGGCC
```

SAFARI

MAGNET Walkthrough

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



```
            Read : TTTTACTGTTCTCCCTTTGAATACAATATATCTATATTTCCCTCTGGCTACATTTAAAATTTCCCCTTTATCTGTAATAATCAGTAATTACGTTTTAAAA
       Reference : TTTTACTGTTCTCCCTTTGAAATGACAATATATCTATATTTCCCTCTGGCTACATTTAAAATTTCCCCTTTATCTGTAATAATCAGTAAATTACCGTTTT
Upper Diagonal-4 : ----11011111111001111 1 1011000010100010110100111111011011001101100110101010111011111111 01 000000
Upper Diagonal-3 : ---011011010101111111 1 1101111111111100100111011111110010001001000100111111101101111110 00 10001
Upper Diagonal-2 : --0011110110010110111 1 0000000000000000000000000000000000000000000000000000000000000000 11 10011
Upper Diagonal-1 : -00011111011100100110 0 1011111111111001001110111111100100010010001001111111011011111101 11 10111
   Main Diagonal : 00000000000000000000 1 1100001010001011010011111101101100110110011010101011101111111111 11 11111
Lower Diagonal-1 : 00011111011100100110 1 1011111111110111110111111011111110111111011110111111000010110101 01 1111-
Lower Diagonal-2 : 00111101100101101111 1 1100100010101110011100111011011111111111111110101011110110101001 01 111--
Lower Diagonal-3 : 01101101010111111110 1 1011111110111101111111111011011011111101111101111011111111111111 01 11---
Lower Diagonal-4 : 11011111111001111101 0 1110000010111010110011111001010011111001110010011110101101111111 00 1----

MAGNET bit-vector : 000000000000000000000101000000000000000000000000000000000000000000000000000000000000000000001000000
```

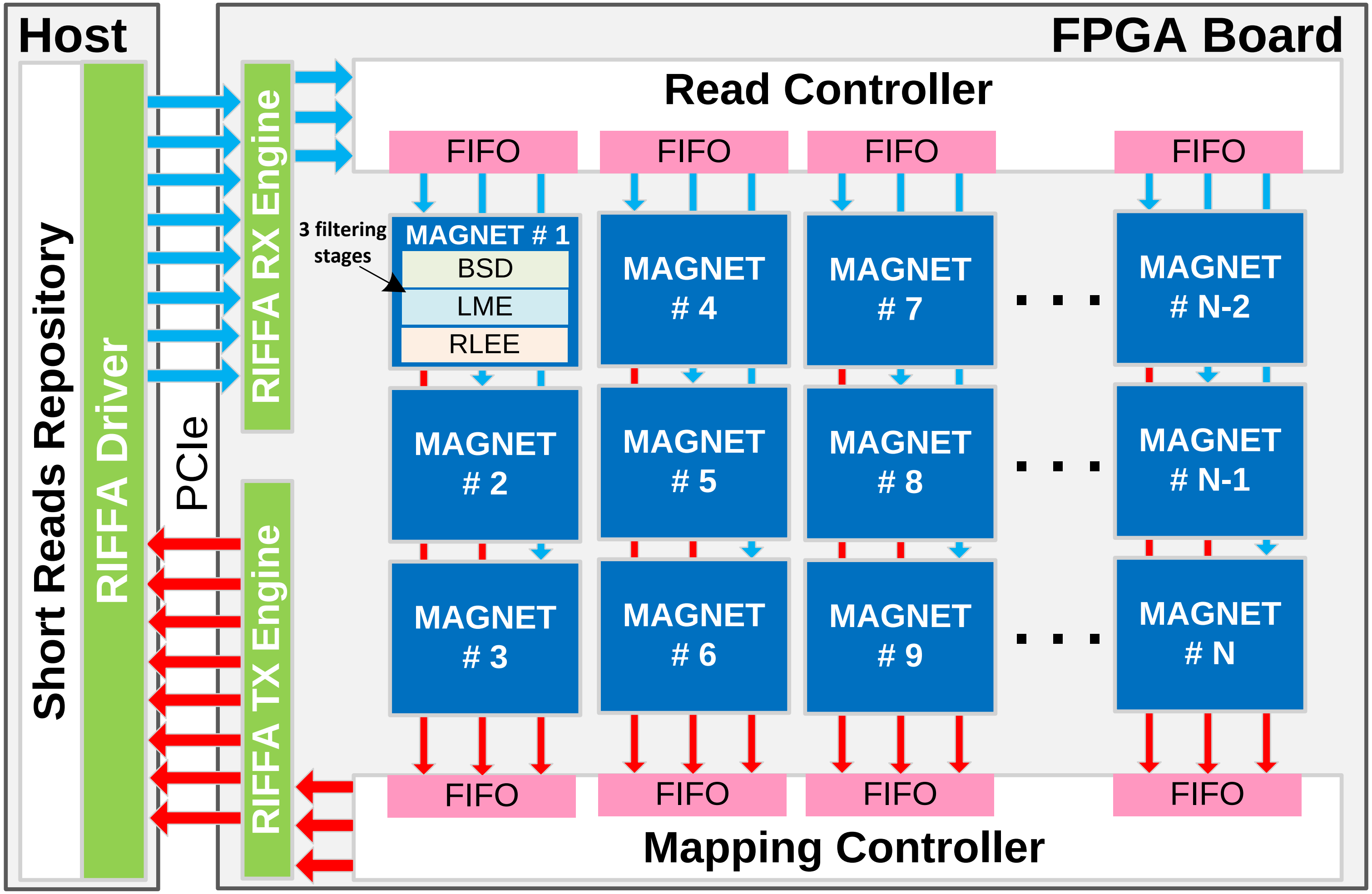
Find the longest segment of consecutive zeros

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

MAGNET Accelerator



Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design

- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions

- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory

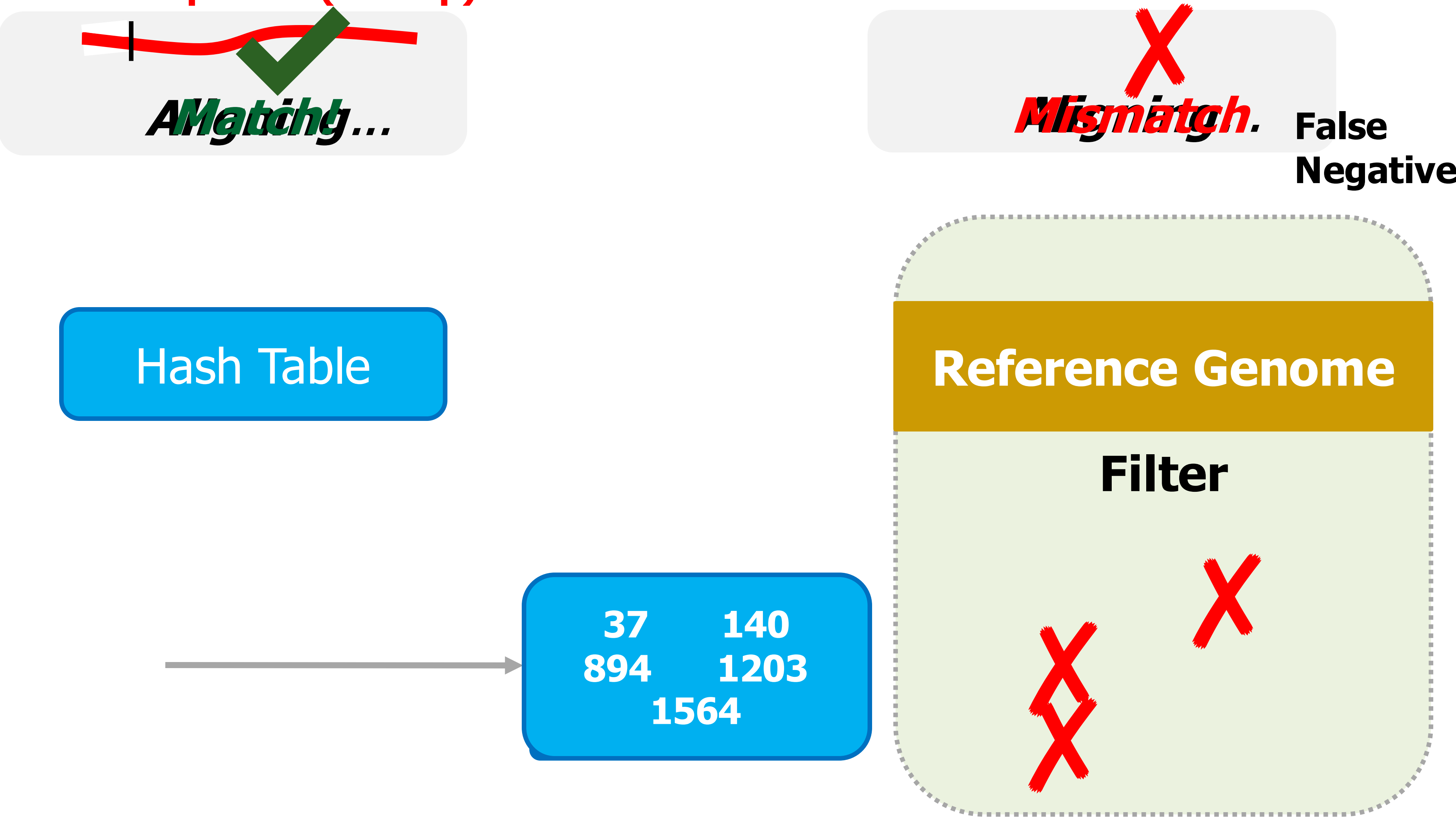
- Future Opportunities: New Sequencing Technologies

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

Hash Tables in Read Mapping

Read Sequence (100 bp)



Aligning…

Match!

Aligning.

Mismatch

False Negative

Hash Table

37 140
894 1203
1564

Reference Genome

Filter

Read Mapping & Filtering in Memory

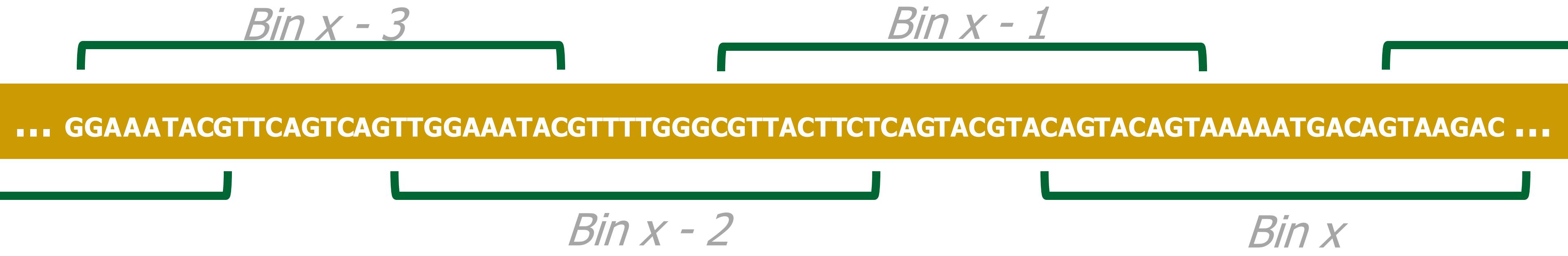
We need to design
mapping & filtering algorithms
that fit processing-in-memory

Our Proposal: GRIM-Filter

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

GRIM-Filter: Bins

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


Bin x - 3
Bin x - 1
... GGAAATACGTTCAGTCAGTTGGAAATACGTTTTGGGCGTTACTTCTCAGTACGTACAGTACAGTAAAAATGACAGTAAGAC ...
Bin x - 2
Bin x

- 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

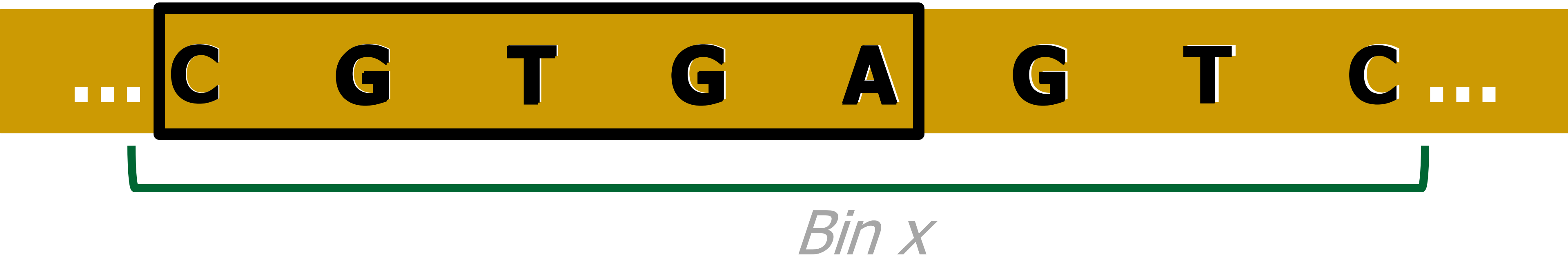
Bitvector

AAAAA	1
AAAAC	0
AAAAT	1
...	...
CCCCC	1
CCCCT	0
CCCCG	0
...	...
GGGGG	1

AAAAA **exists** in bin x

CCCCT **doesn't exist** in bin x

GRIM-Filter: Bitvectors

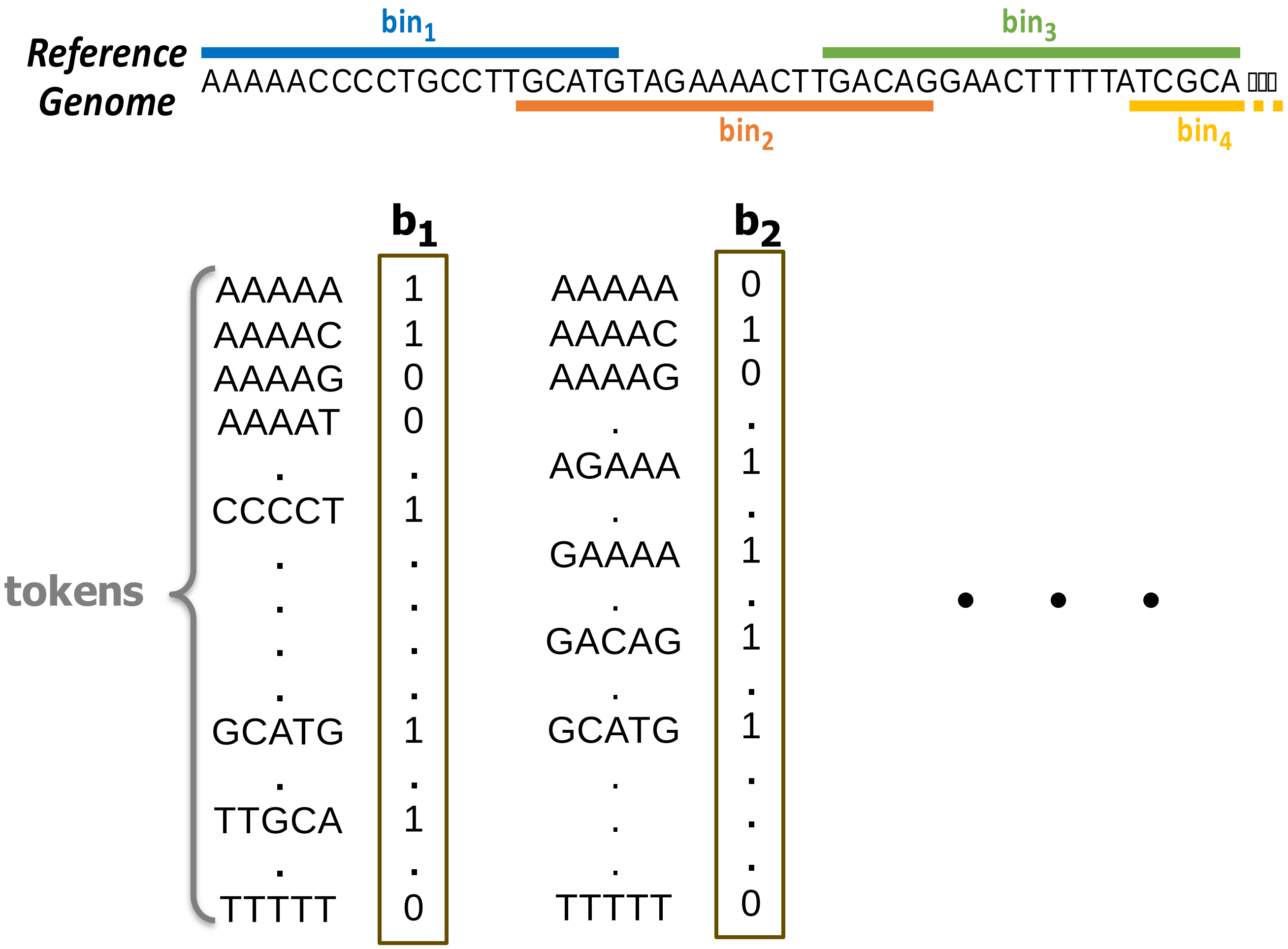


Bin x Bitvector

AAAAA	**0**
…	**…**
CGTGA	**0**
…	**…**
TGAGT	**0**
…	**…**
GAGTC	**0**
…	**…**
GTGAG	**0**
…	**…**

SAFARI

GRIM-Filter: Bitvectors



Reference Genome

bin$_1$ bin$_3$

AAAAACCCCTGCCTTGCATGTAGAAAACTTGACAGGAACTTTTATCGCA ▯▯▯

bin$_2$ bin$_4$

tokens

	b$_1$
AAAAA	1
AAAAC	1
AAAAG	0
AAAAT	0
.	.
CCCCT	1
.	.
.	.
.	.
.	.
GCATG	1
.	.
TTGCA	1
.	.
TTTTT	0

	b$_2$
AAAAA	0
AAAAC	1
AAAAG	0
.	.
AGAAA	1
.	.
GAAAA	1
.	.
GACAG	1
.	.
GCATG	1
.	.
.	.
.	.
TTTTT	0

• • •

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

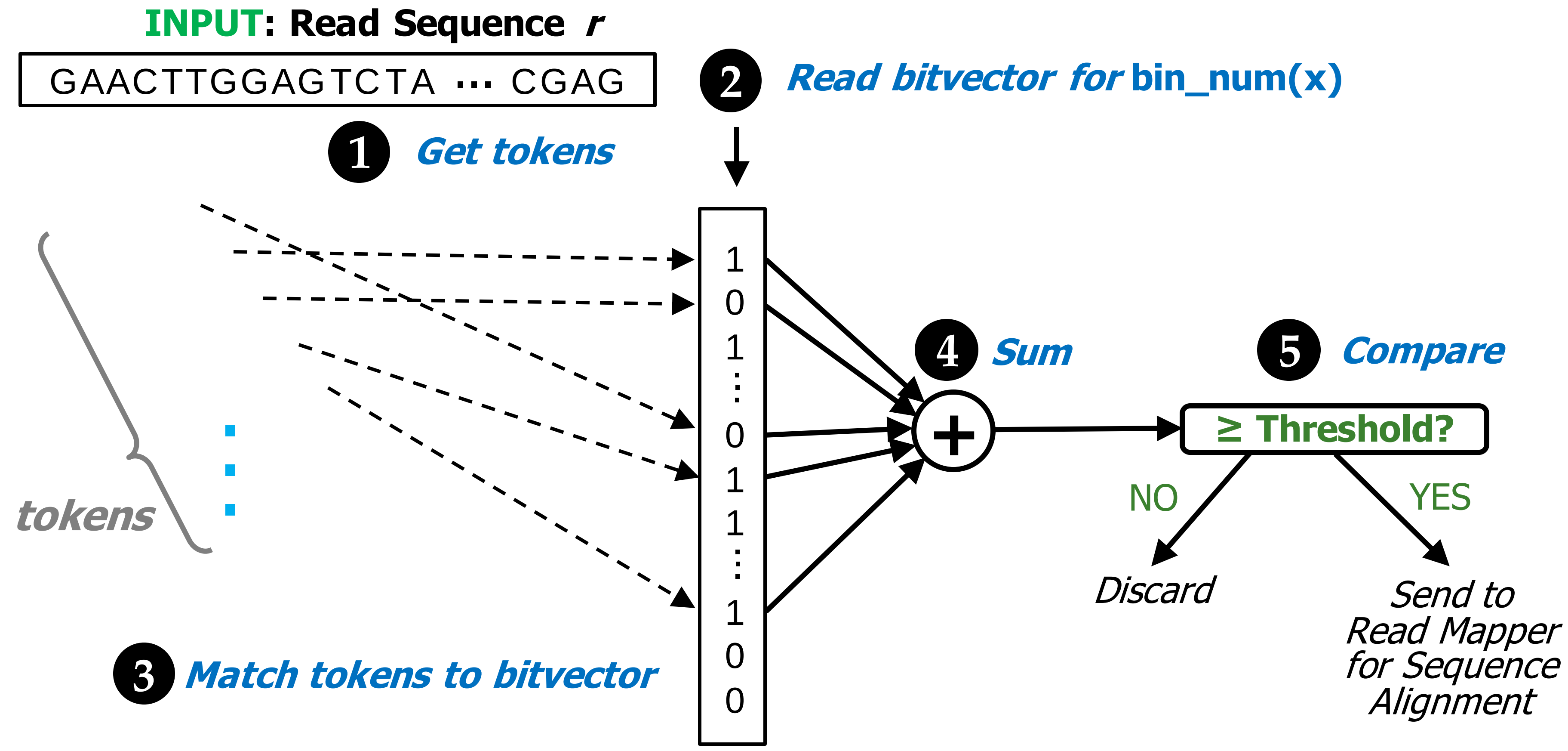
For **bin size** ~200, and **n** = 5, **memory footprint** ~3.8 GB

Our 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



INPUT: Read Sequence *r*

GAACTTGGAGTCTA ... CGAG

1 *Get tokens*

2 *Read bitvector for* **bin_num(x)**

3 *Match tokens to bitvector*

4 *Sum*

5 *Compare*

≥ Threshold?

NO → *Discard*

YES → *Send to Read Mapper for Sequence Alignment*

SAFARI

Our Proposal: GRIM-Filter

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

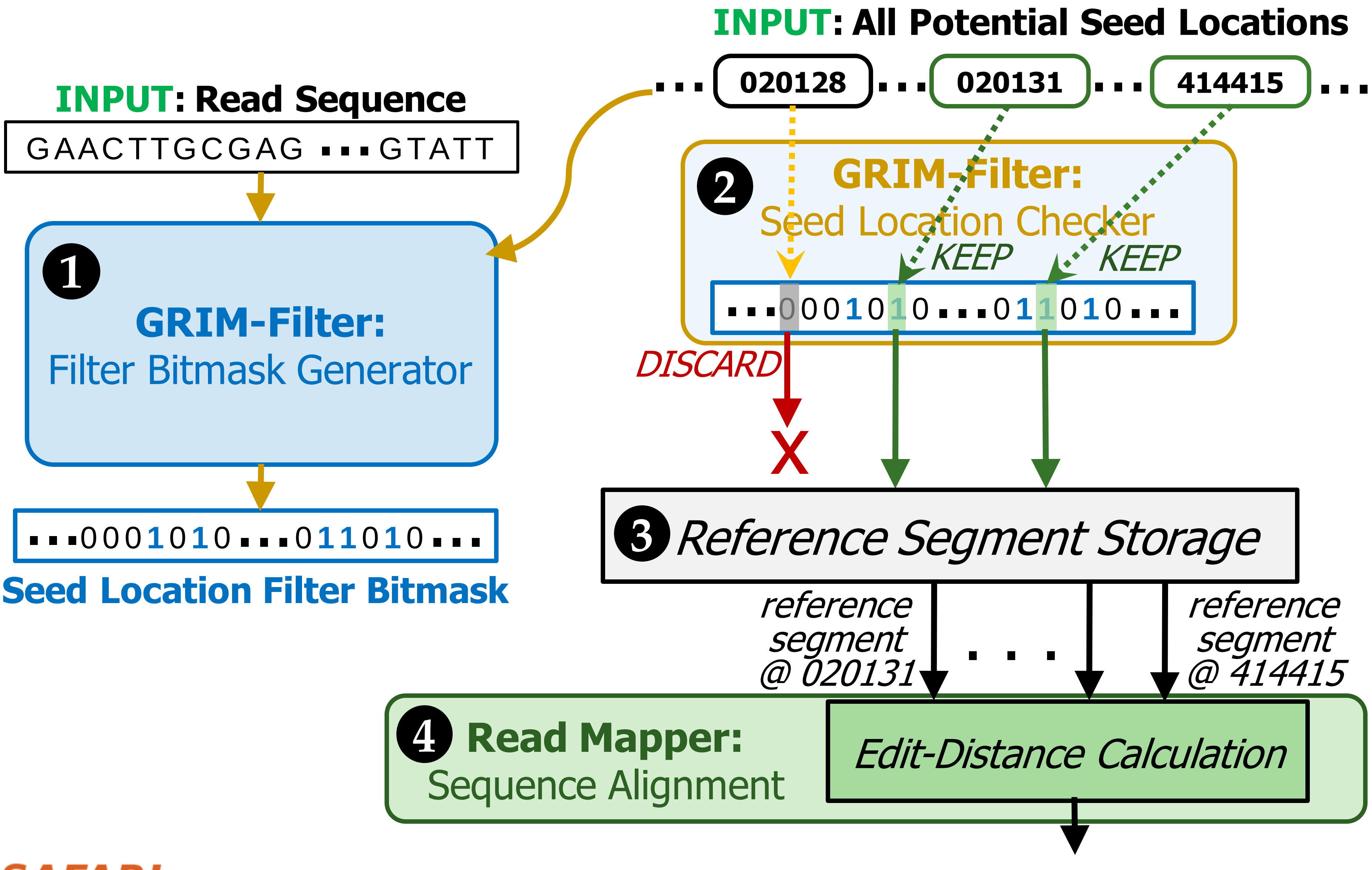
Our Proposal: GRIM-Filter

1. Data Structures: Bins & Bitvectors

2. Checking a Bin

3. **Integrating GRIM-Filter into a Mapper**

Integrating GRIM-Filter into a Read Mapper



OUTPUT: Correct Mappings

Key Properties of GRIM-Filter

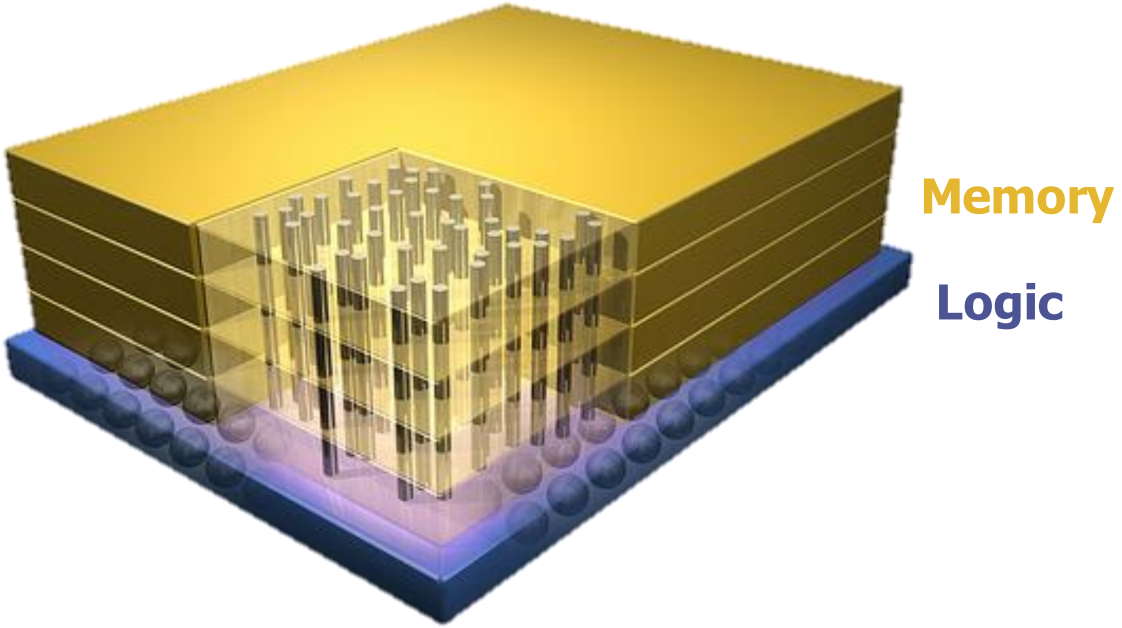
1. **Simple Operations**:
   - To check a given bin, find the **sum** of all bits corresponding to each token in the read
   - **Compare** against threshold to determine whether to align

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

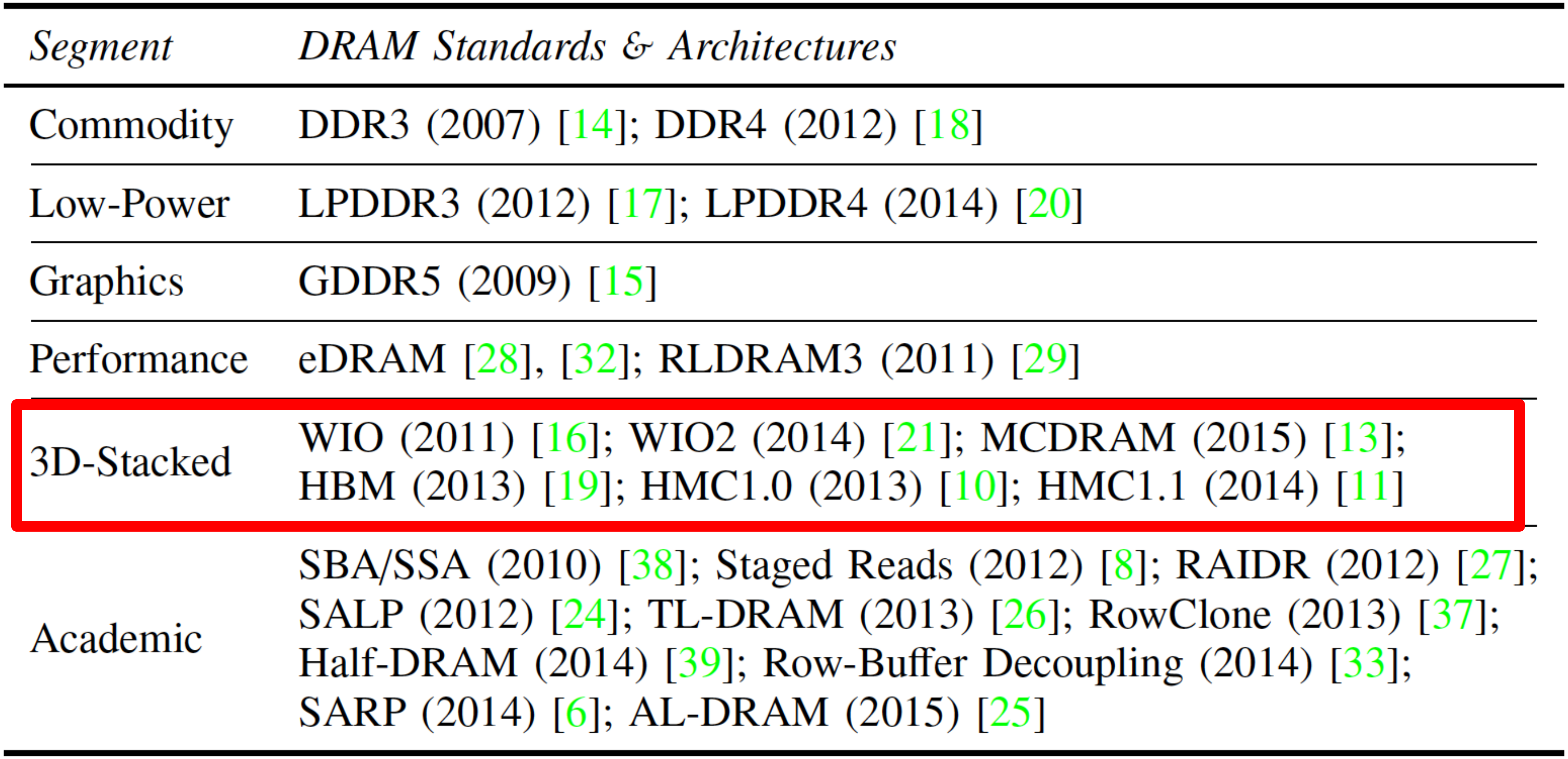
These properties together make GRIM-Filter a good algorithm to be run in 3D-Stacked DRAM

Opportunity: 3D-Stacked Logic+Memory



Other "True 3D" technologies under development

DRAM Landscape (circa 2015)

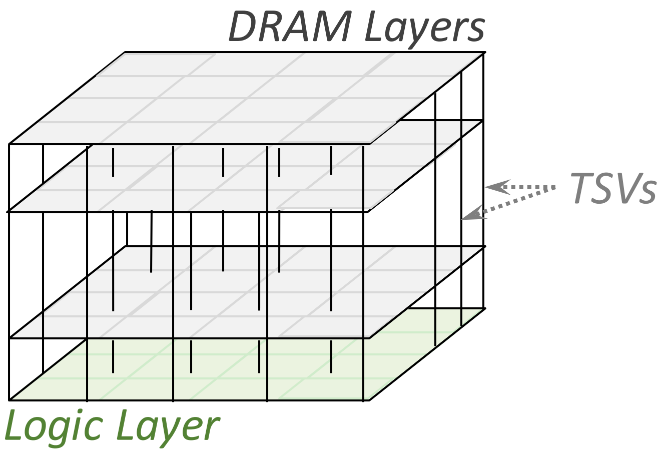


Segment	*DRAM Standards & Architectures*
Commodity	DDR3 (2007) [14]; DDR4 (2012) [18]
Low-Power	LPDDR3 (2012) [17]; LPDDR4 (2014) [20]
Graphics	GDDR5 (2009) [15]
Performance	eDRAM [28], [32]; RLDRAM3 (2011) [29]
3D-Stacked	WIO (2011) [16]; WIO2 (2014) [21]; MCDRAM (2015) [13]; HBM (2013) [19]; HMC1.0 (2013) [10]; HMC1.1 (2014) [11]
Academic	SBA/SSA (2010) [38]; Staged Reads (2012) [8]; RAIDR (2012) [27]; SALP (2012) [24]; TL-DRAM (2013) [26]; RowClone (2013) [37]; Half-DRAM (2014) [39]; Row-Buffer Decoupling (2014) [33]; SARP (2014) [6]; AL-DRAM (2015) [25]

Table 1. Landscape of DRAM-based memory

Kim+, "Ramulator: A Flexible and Extensible DRAM Simulator", IEEE CAL 2015.

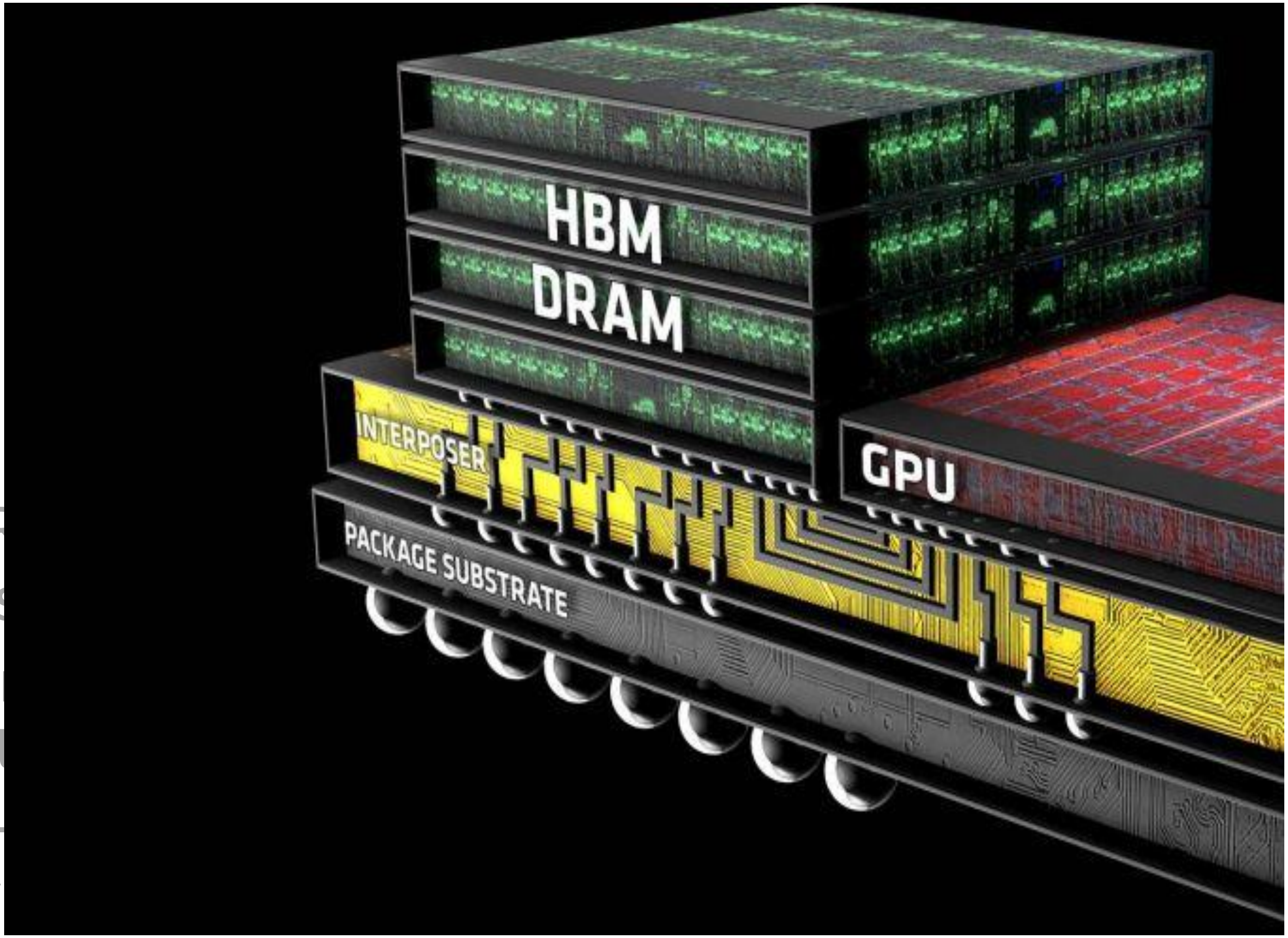
3D-Stacked Memory



- 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

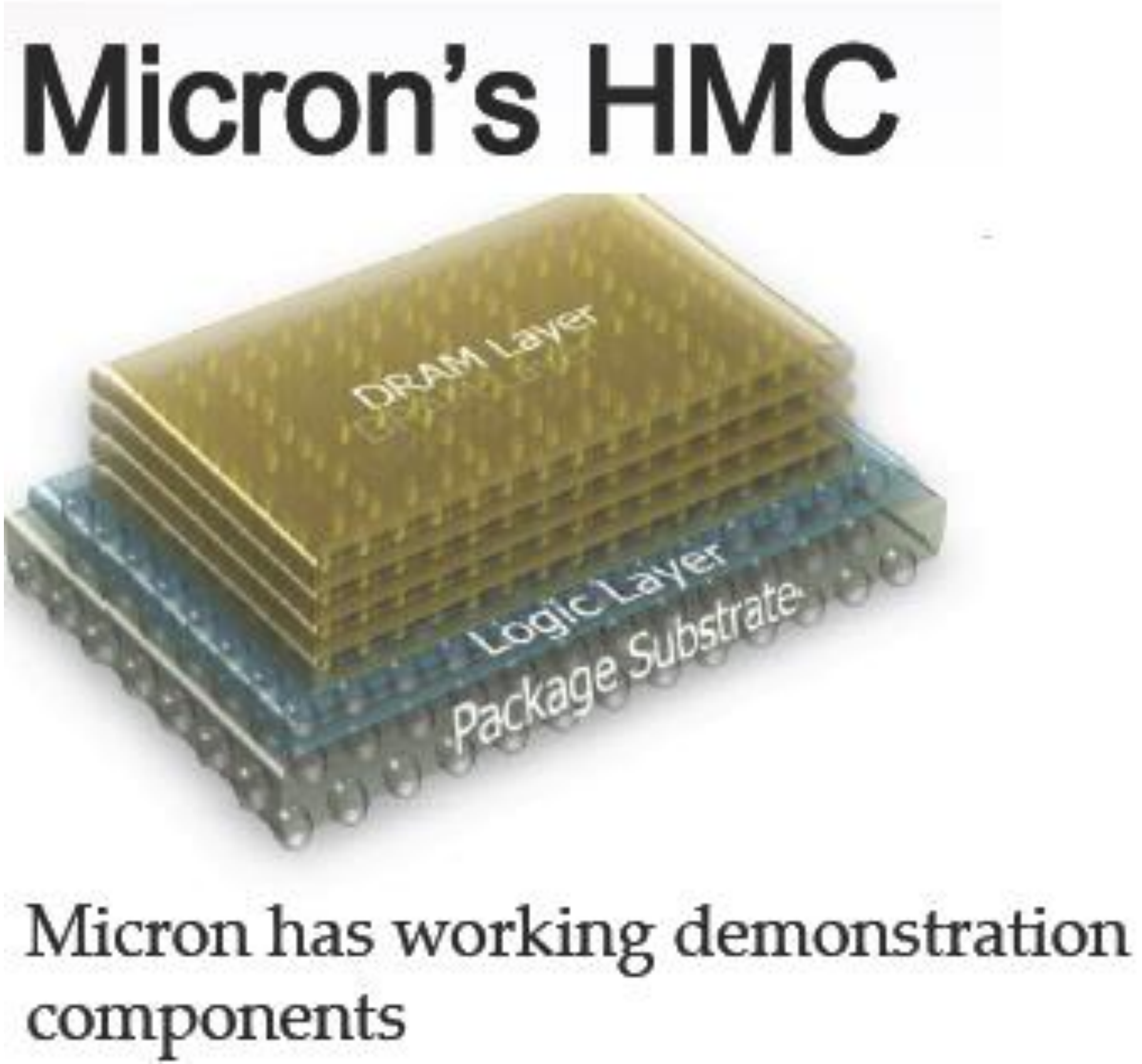
3D-Stacked Memory

- 3D-Stacked DR
  bandwidth as
  - Logic Layer e
    computation t
  - Embed GRIM-
    appropriately



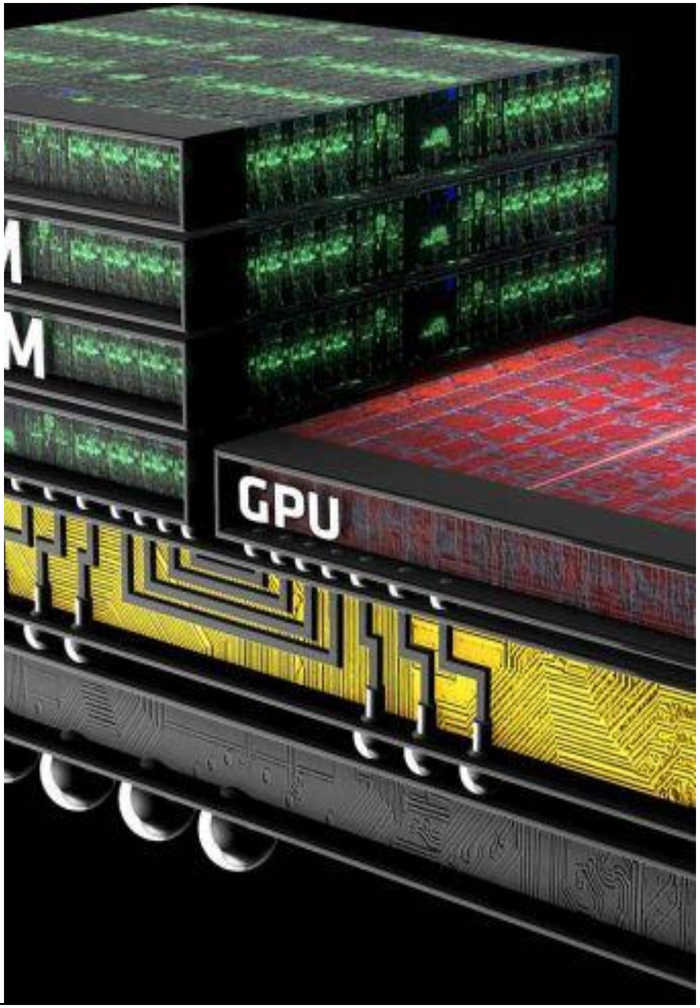
http://i1-news.softpedia-static.com/images/news2/Micron-and-Samsung-Join-Force-to-Create-Next-Gen-Hybrid-Memory-2.png

3D-Stacked Memory



Micron's HMC

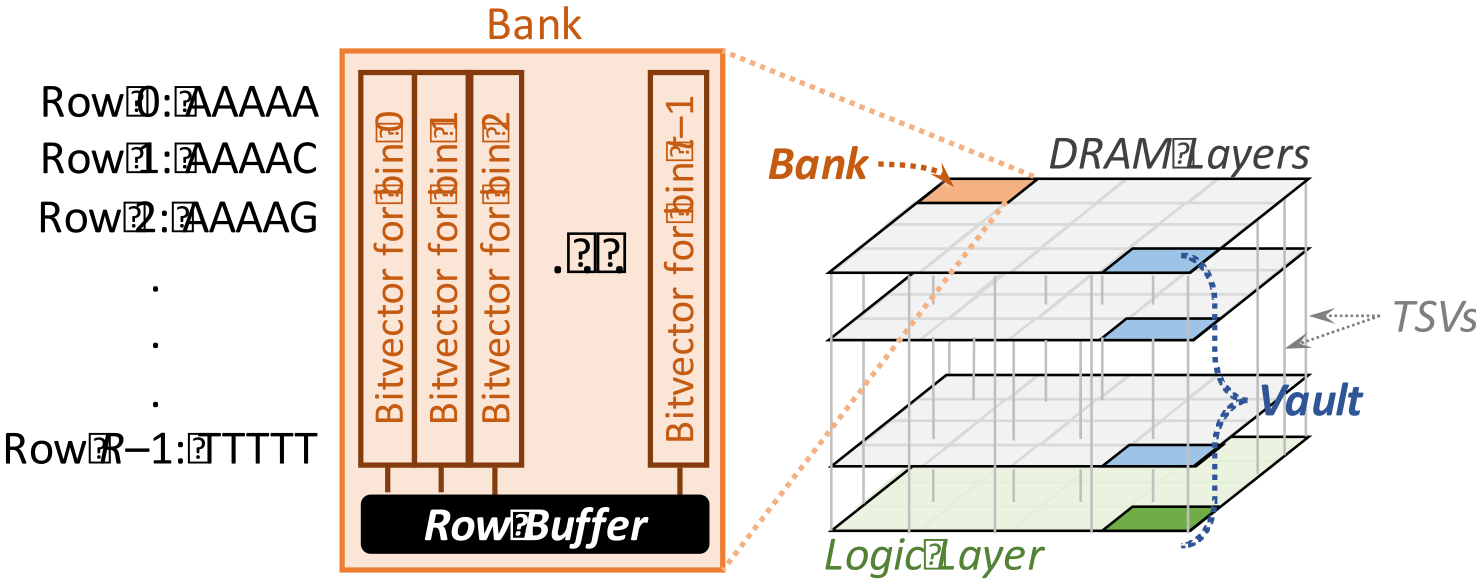
Micron has working demonstration components



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

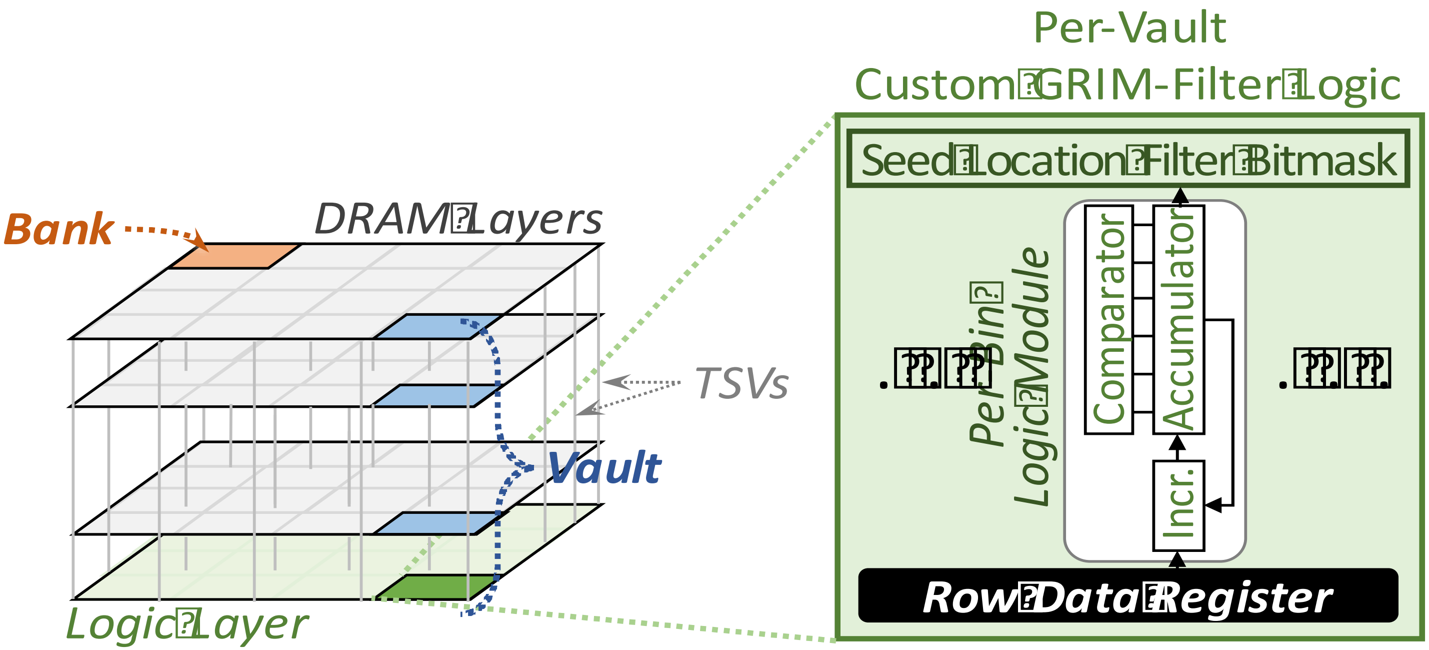
GRIM-Filter in 3D-Stacked DRAM



- 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, we use 4096 incrementer LUTs, 7-bit counters, and comparators in logic layer

Details are in [Kim+, BMC Genomics 2018]

Methodology

- Performance simulated using an in-house 3D-Stacked DRAM simulator

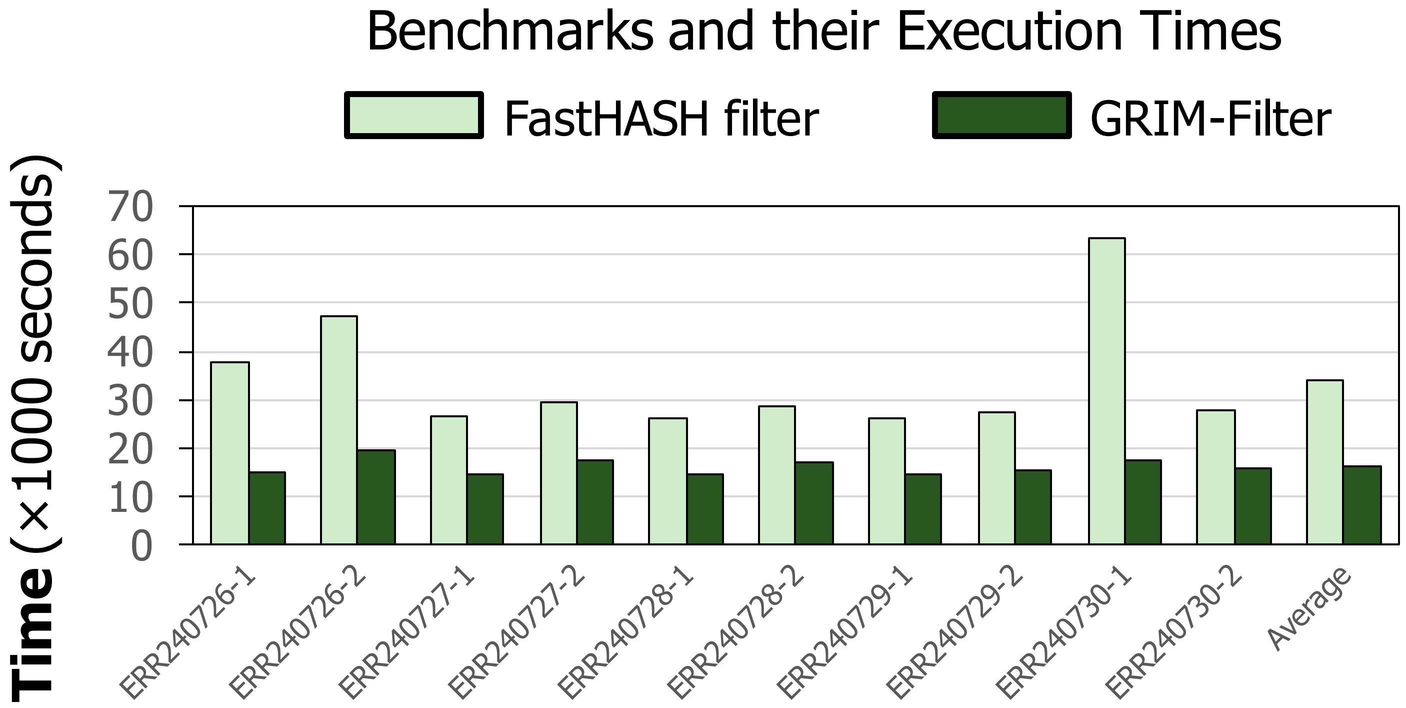
- Evaluate 10 real read data sets (From the 1000 Genomes Project)
  - Each data set consists of 4 million reads of length 100

- 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**

SAFARI

GRIM-Filter Performance



Sequence Alignment Error Tolerance (*e*)

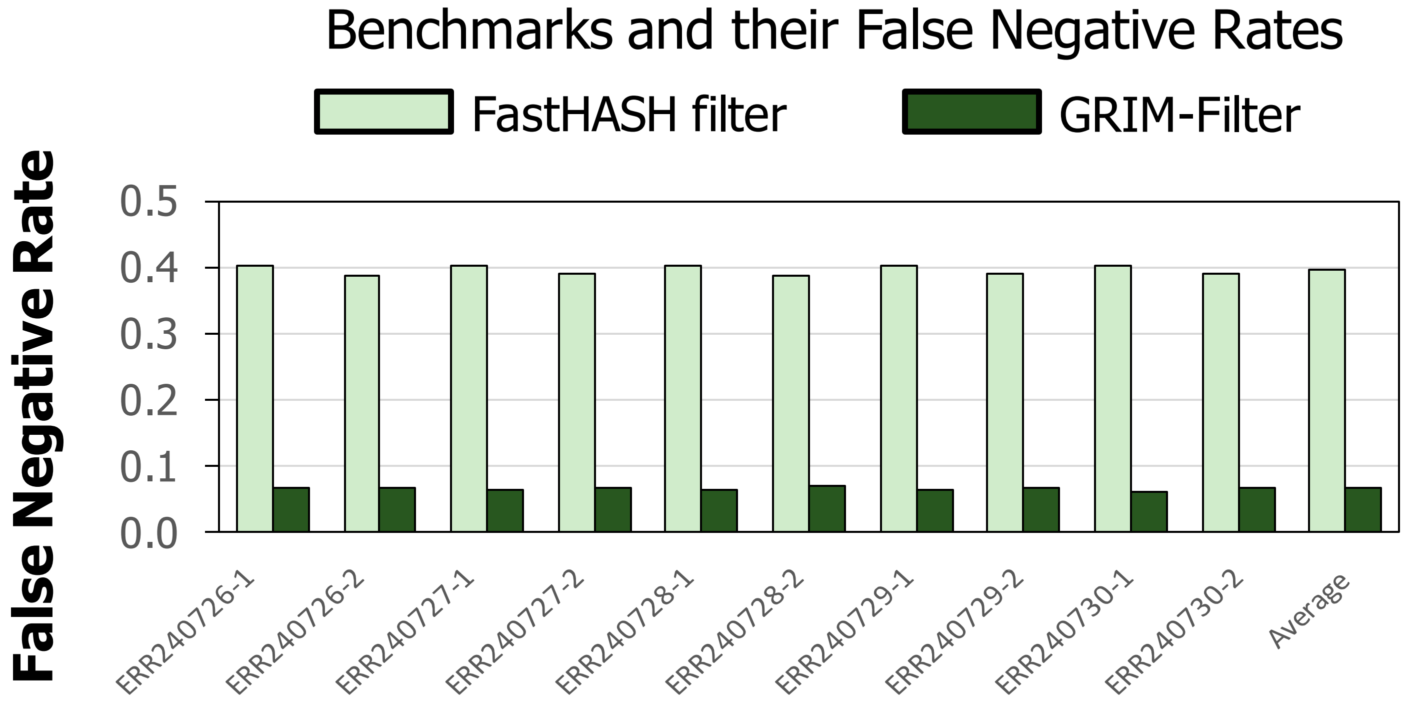
***e* = 0.05**

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



Sequence Alignment Error Tolerance (*e*)

***e* = 0.05**

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

More on GRIM-Filter

- Jeremie S. Kim, Damla Senol Cali, Hongyi Xin, Donghyuk Lee, Saugata Ghose, Mohammed Alser, Hasan Hassan, Oguz Ergin, Can Alkan, and Onur Mutlu, **"GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies"**
  BMC Genomics, 2018.
  *Proceedings of the 16th Asia Pacific Bioinformatics Conference (**APBC**)*, Yokohama, Japan, January 2018.
  arxiv.org Version (pdf)

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

Jeremie S. Kim[1,6*], Damla Senol Cali[1], Hongyi Xin[2], Donghyuk Lee[3], Saugata Ghose[1], Mohammed Alser[4], Hasan Hassan[6], Oguz Ergin[5], Can Alkan[4*] and Onur Mutlu[6,1*]

From The Sixteenth Asia Pacific Bioinformatics Conference 2018
Yokohama, Japan. 15-17 January 2018

Aside: In-Memory Graph Processing

- Large graphs are everywhere (circa 2015)

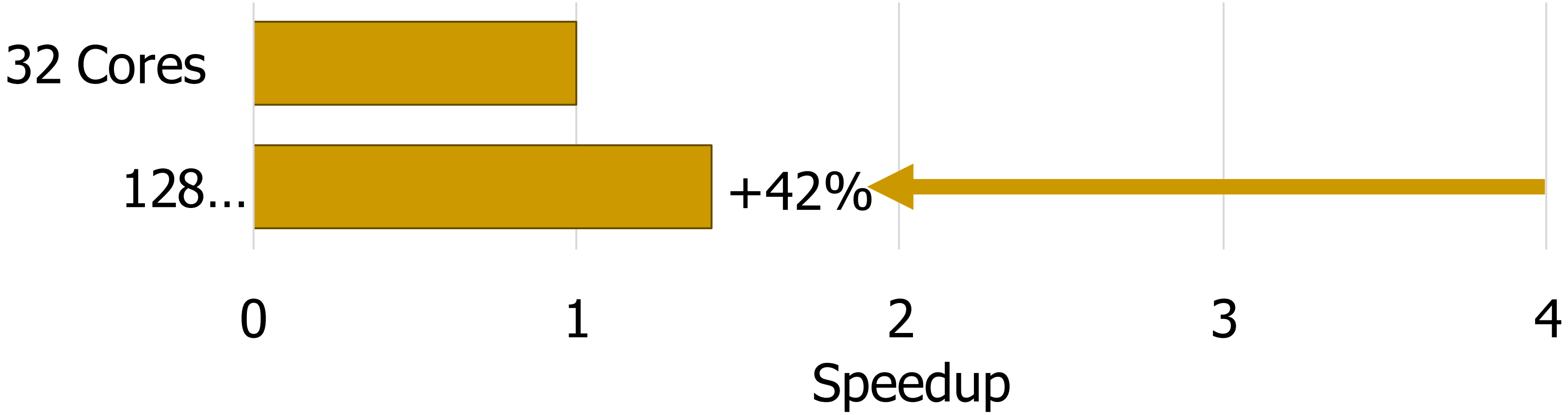
36 Million
Wikipedia Pages

1.4 Billion
Facebook Users

300 Million
Twitter Users

30 Billion
Instagram Photos

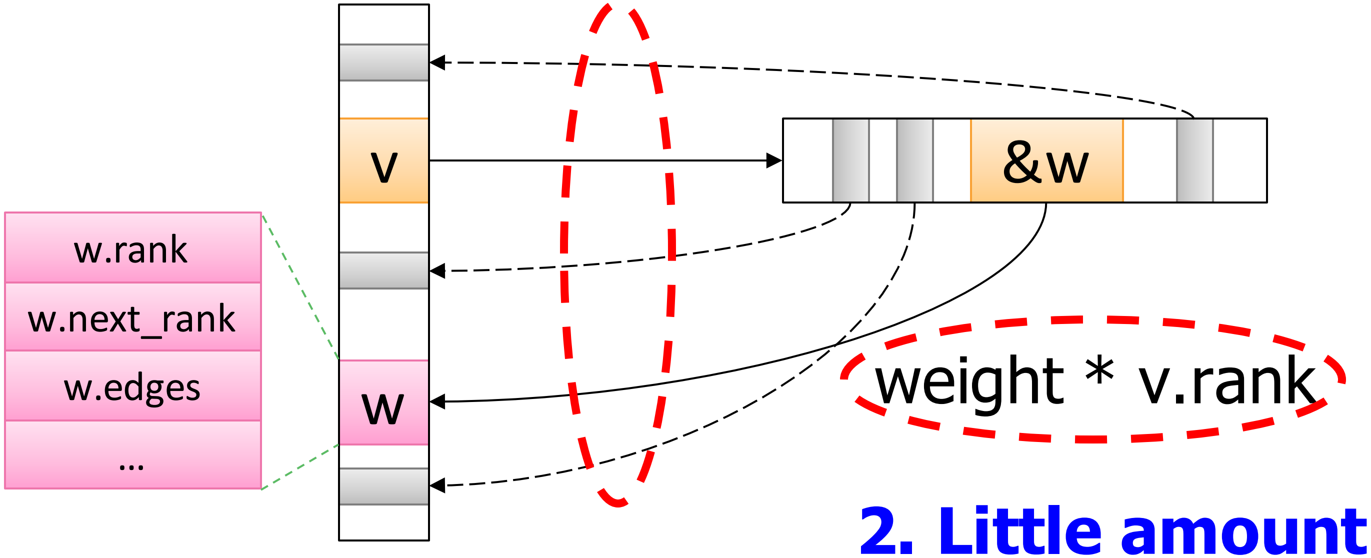
- Scalable large-scale graph processing is challenging



Key Bottlenecks in Graph Processing

```
for (v: graph.vertices) {
  for (w: v.successors) {
    w.next_rank += weight * v.rank;
  }
}
```

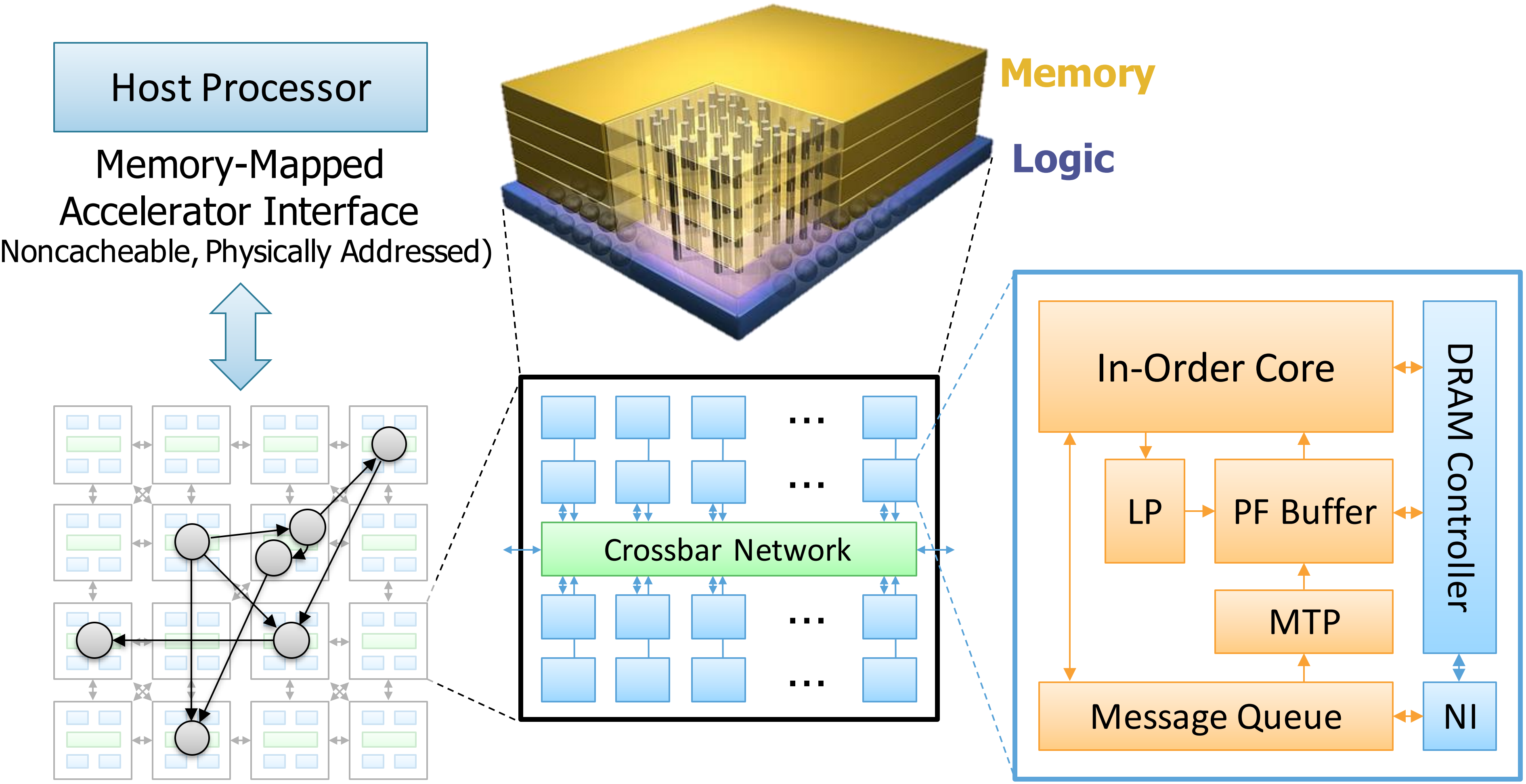
1. Frequent random memory accesses



2. Little amount of computation

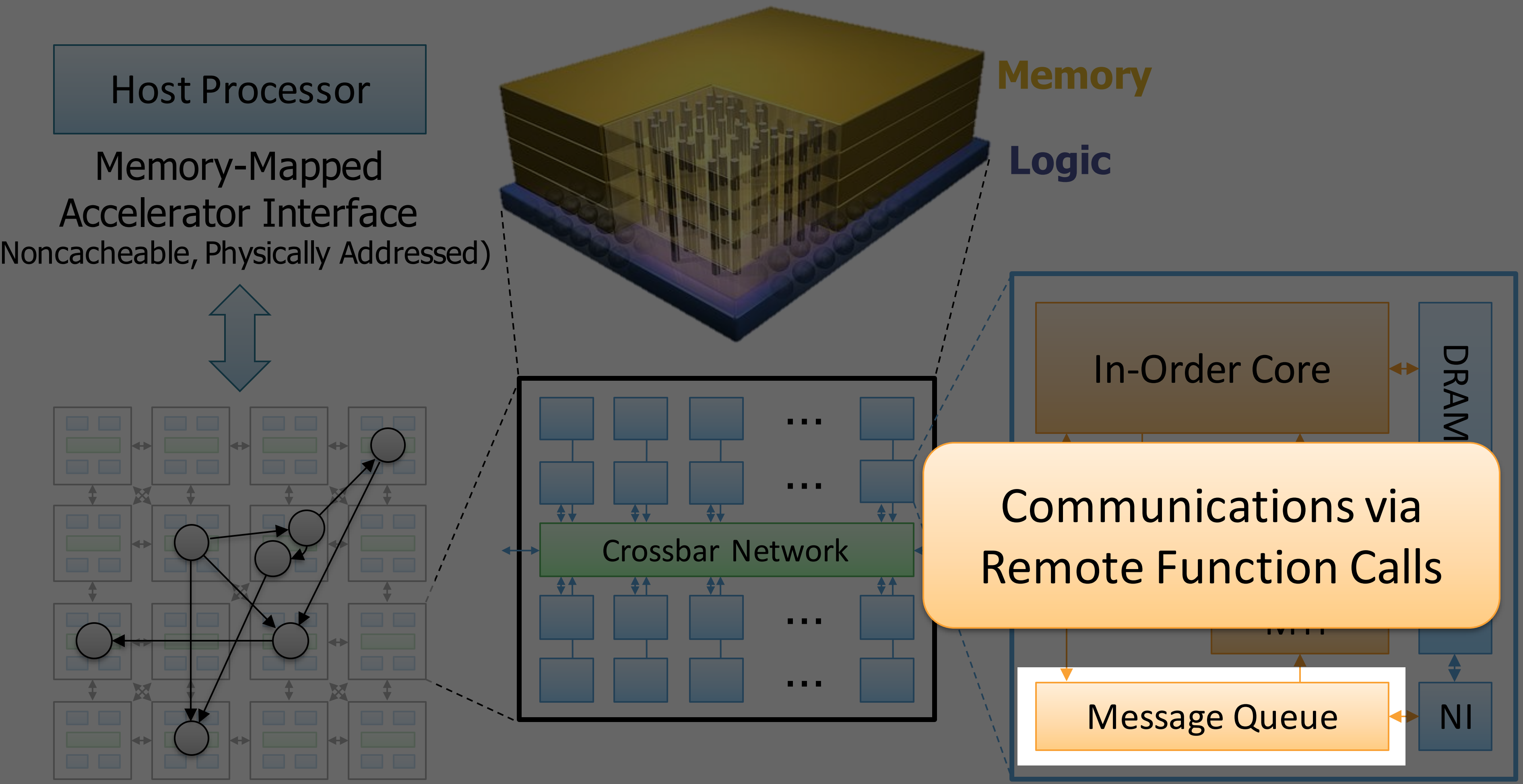
Tesseract System for Graph Processing

Interconnected set of 3D-stacked memory+logic chips with simple cores



Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract System for Graph Processing



Host Processor

Memory-Mapped
Accelerator Interface
(Noncacheable, Physically Addressed)

Memory

Logic

Crossbar Network

In-Order Core

DRAM

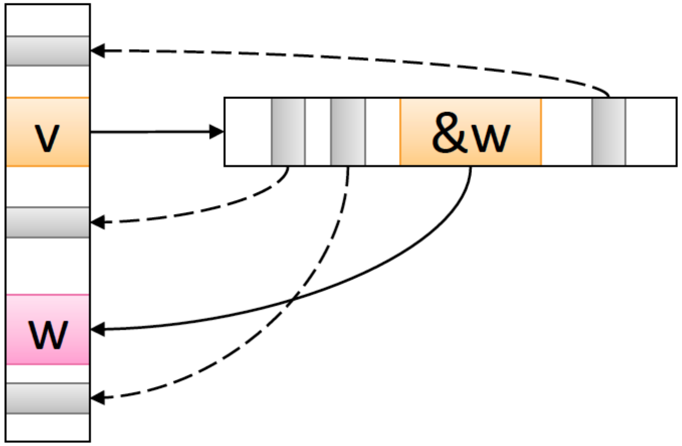
Message Queue

NI

Communications via
Remote Function Calls

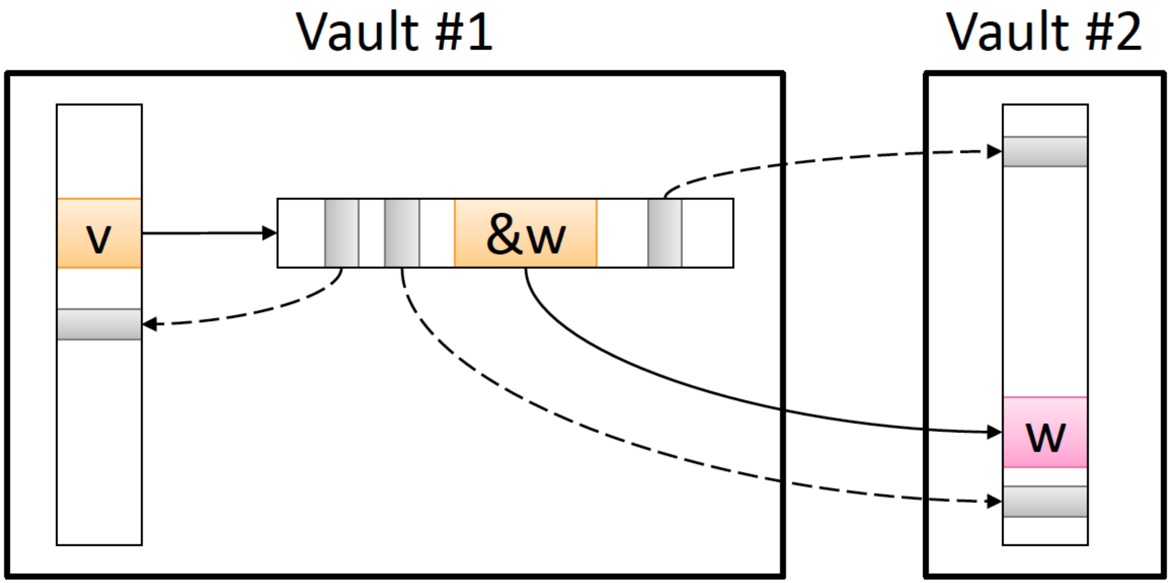
Communications In Tesseract (I)

```
for (v: graph.vertices) {
  for (w: v.successors) {
    w.next_rank += weight * v.rank;
  }
}
```



Communications In Tesseract (II)

```
for (v: graph.vertices) {
    for (w: v.successors) {
        w.next_rank += weight * v.rank;
    }
}
```

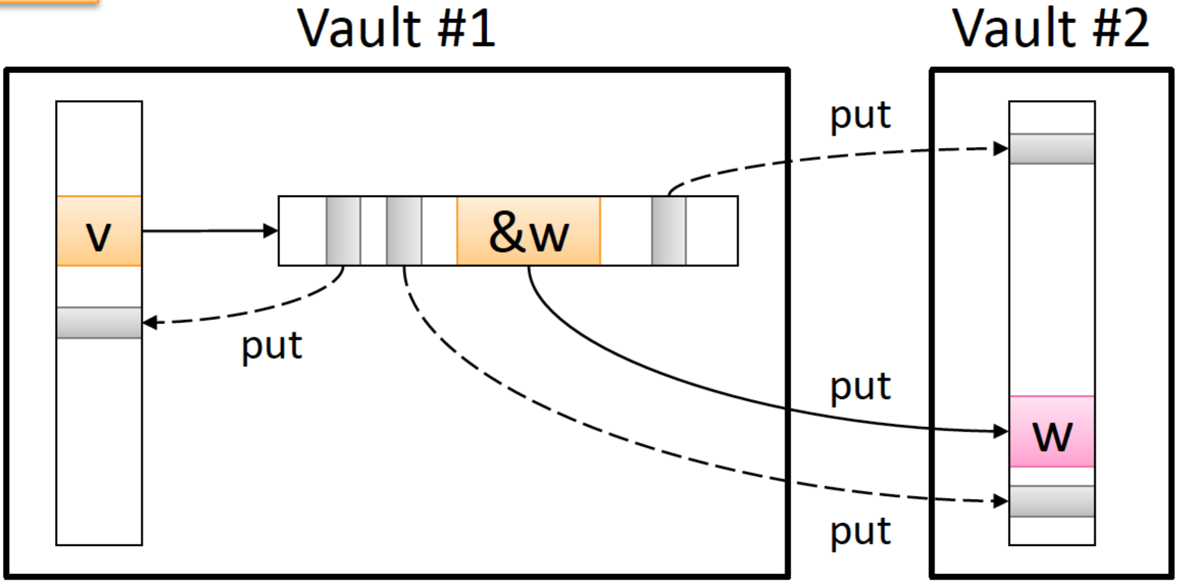


Communications In Tesseract (III)

```
for (v: graph.vertices) {
  for (w: v.successors) {
    put(w.id, function() { w.next_rank += weight * v.rank; });
  }
}
barrier();
```

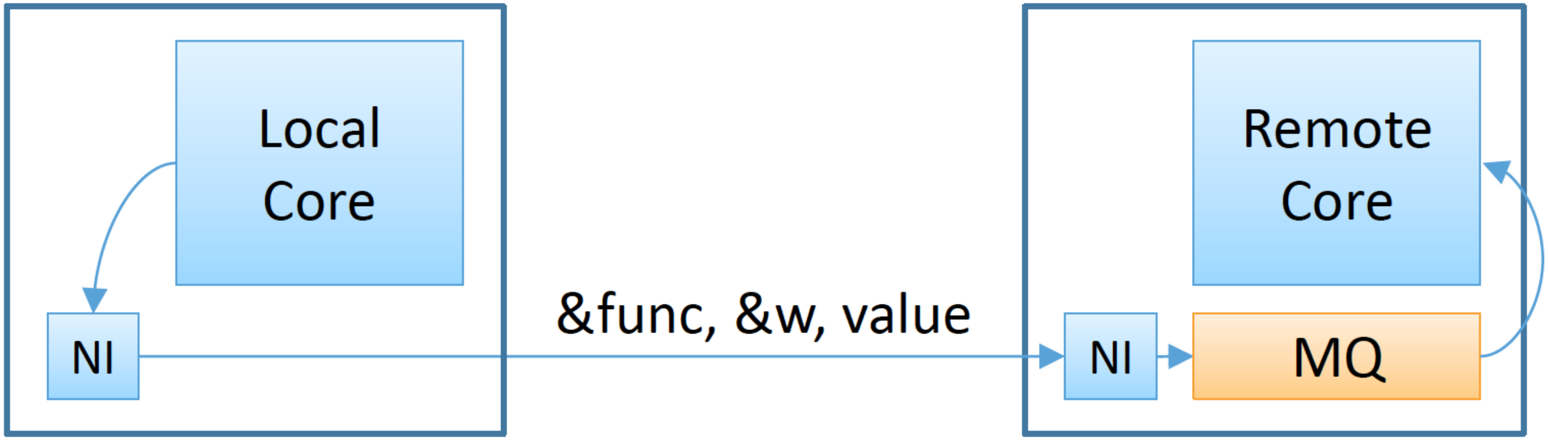
Non-blocking Remote Function Call

Can be **delayed** until the nearest barrier



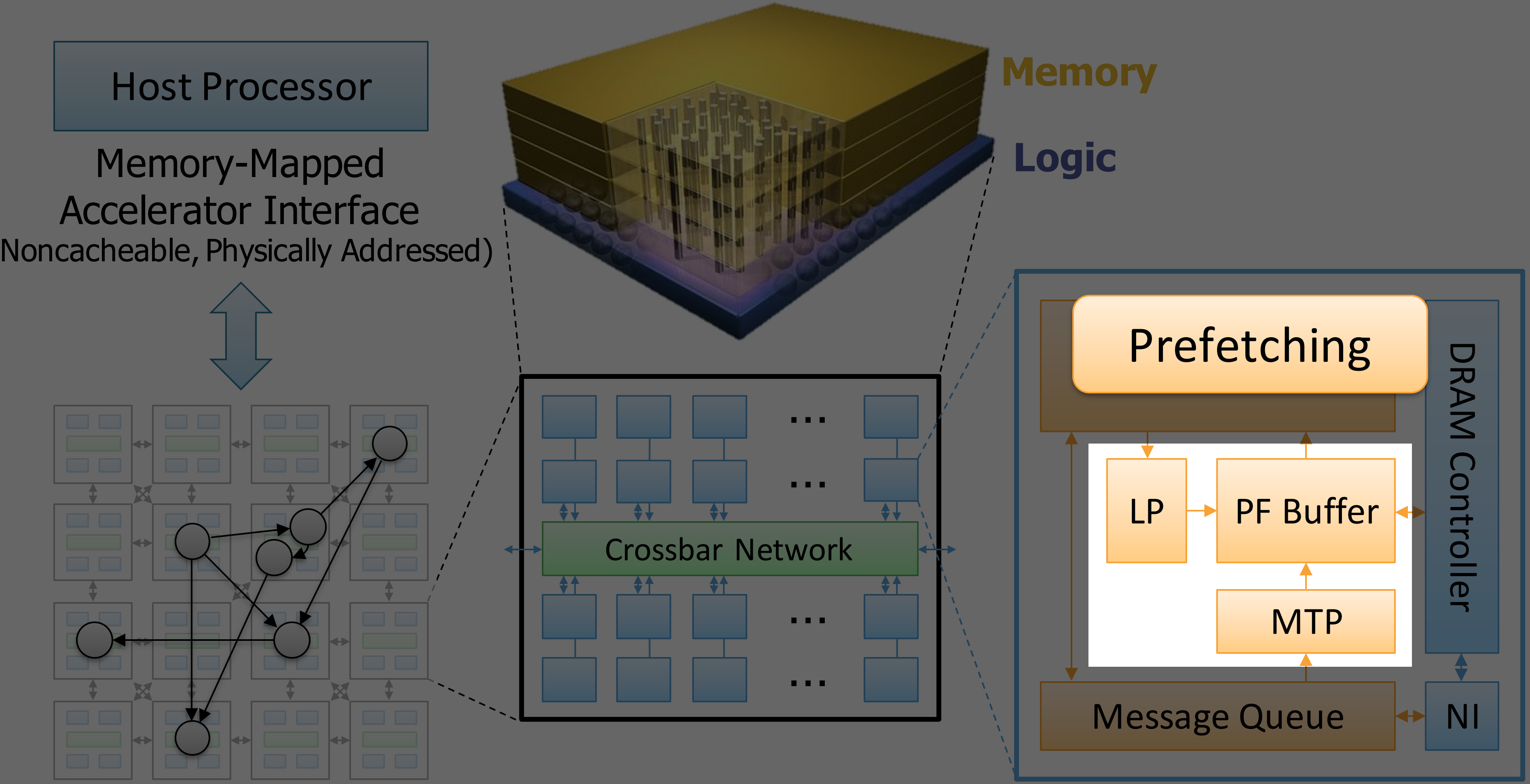
Remote Function Call (Non-Blocking)

1. Send function address & args to the remote core
2. Store the incoming message to the message queue
3. Flush the message queue when it is full or a synchronization barrier is reached

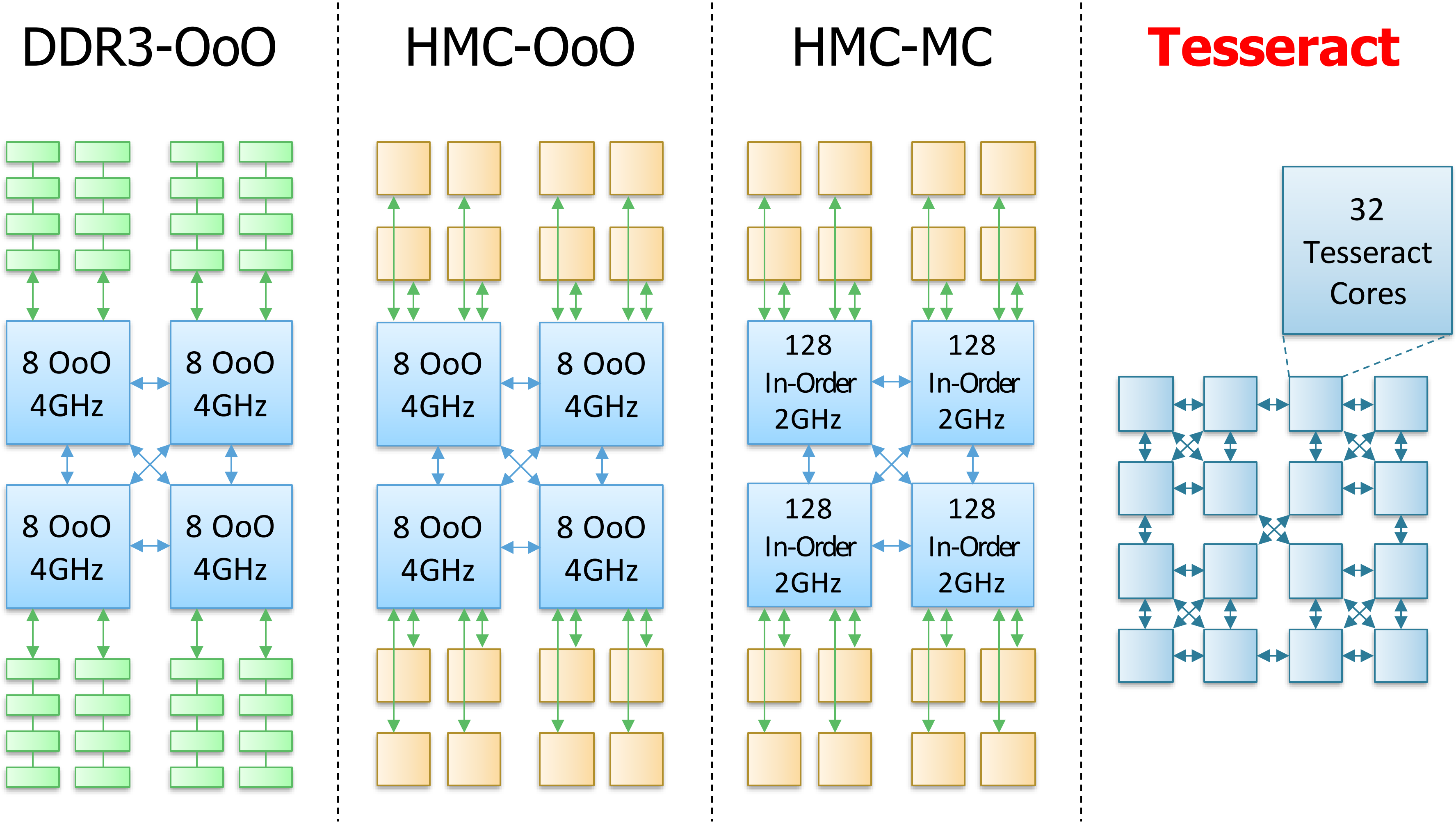


```
put(w.id, function() { w.next_rank += value; })
```

Tesseract System for Graph Processing



Evaluated Systems



DDR3-OoO

8 OoO 4GHz

8 OoO 4GHz

8 OoO 4GHz

8 OoO 4GHz

102.4GB/s

HMC-OoO

8 OoO 4GHz

8 OoO 4GHz

8 OoO 4GHz

8 OoO 4GHz

640GB/s

HMC-MC

128 In-Order 2GHz

128 In-Order 2GHz

128 In-Order 2GHz

128 In-Order 2GHz

640GB/s

Tesseract

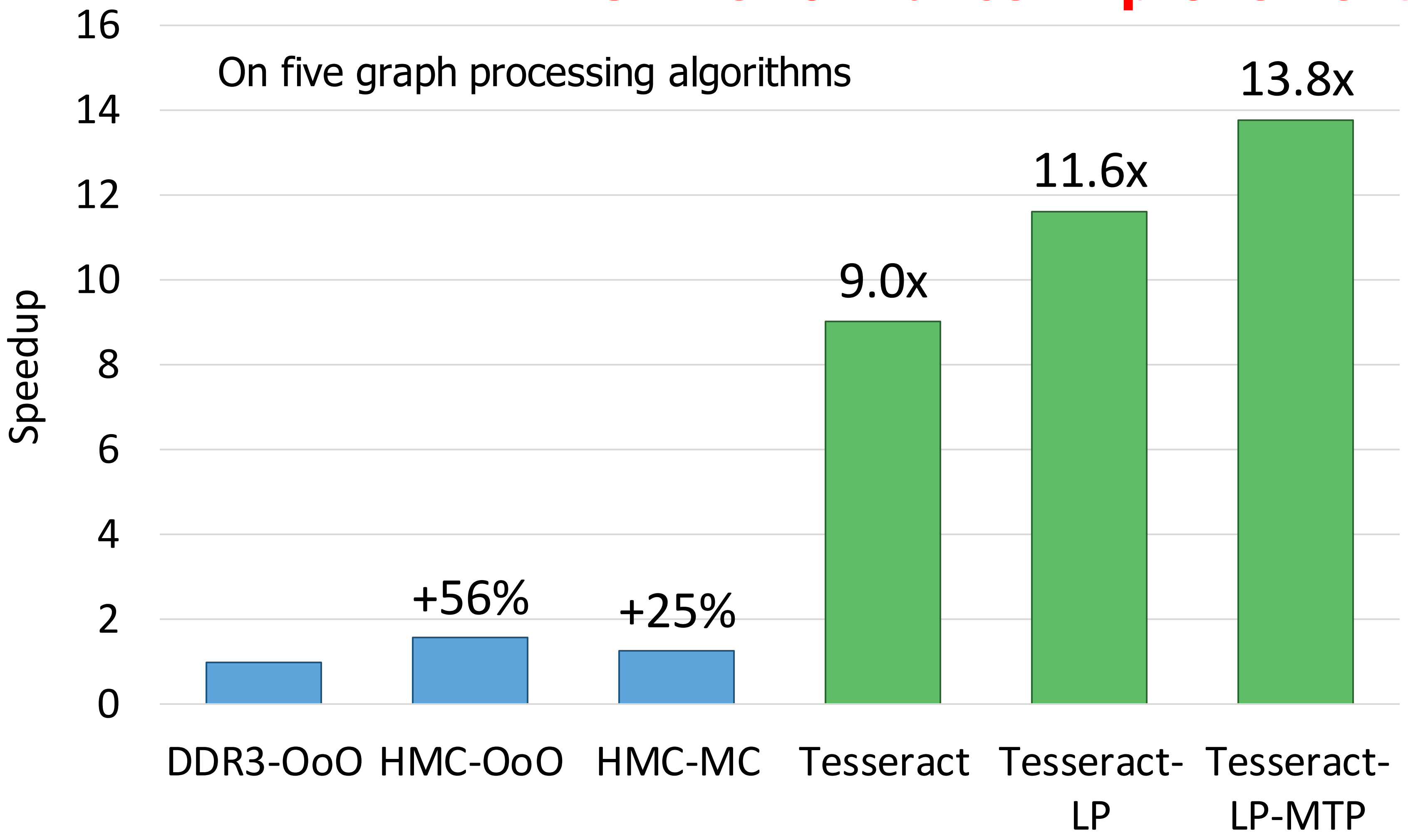
32 Tesseract Cores

8TB/s

SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

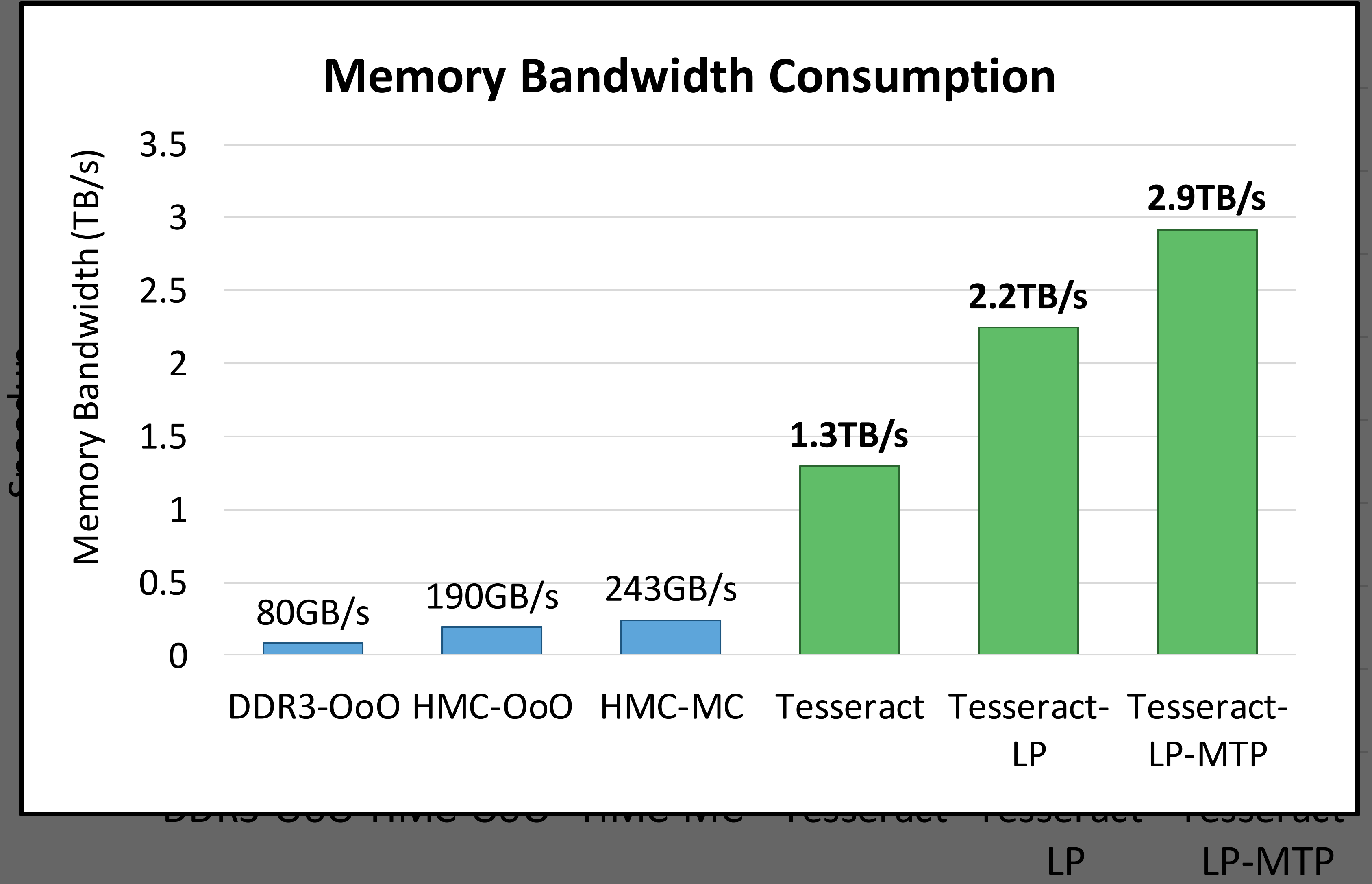
Tesseract Graph Processing Performance

>13X Performance Improvement

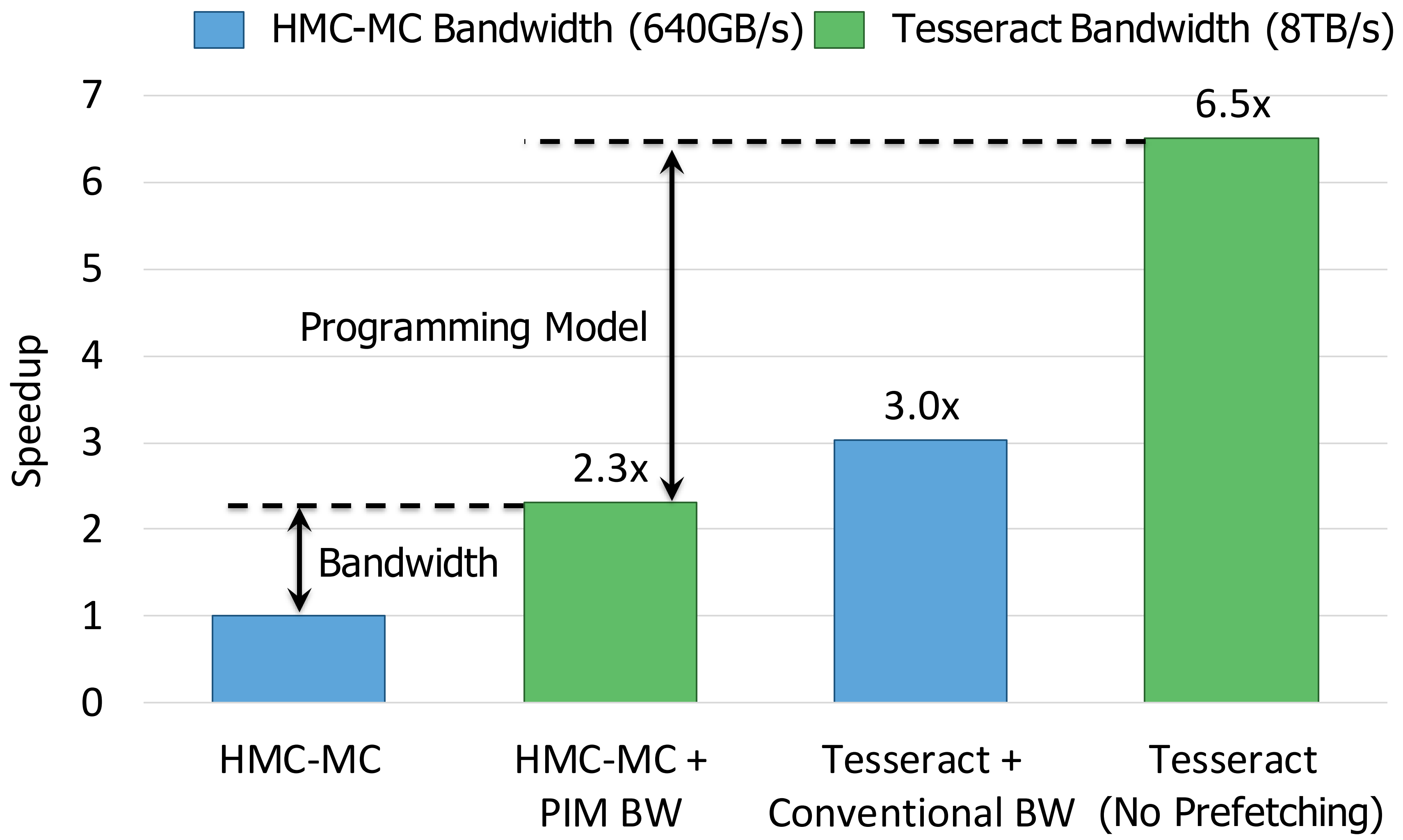


Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

Tesseract Graph Processing Performance

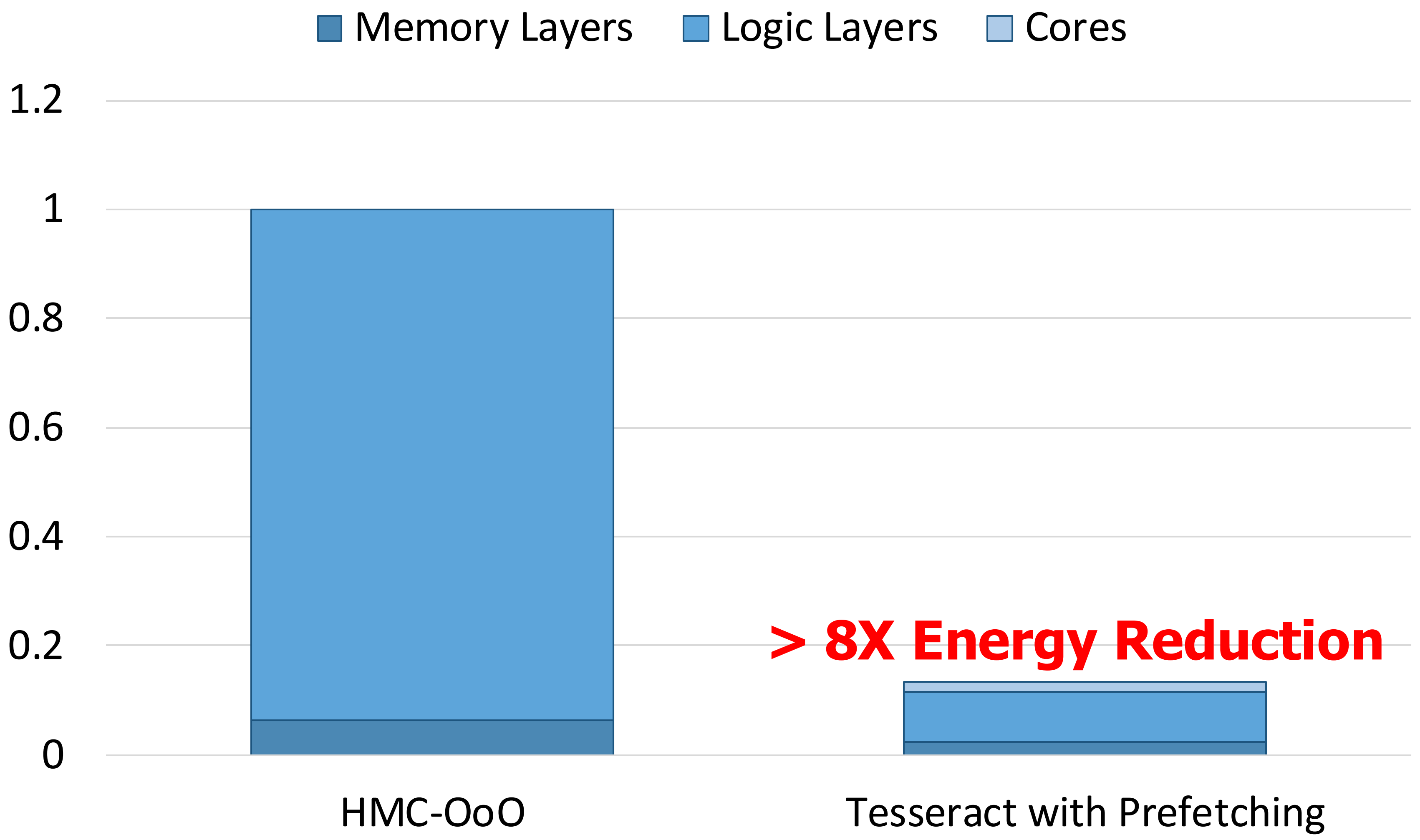


Effect of Bandwidth & Programming Model



SAFARI

Tesseract Graph Processing System Energy

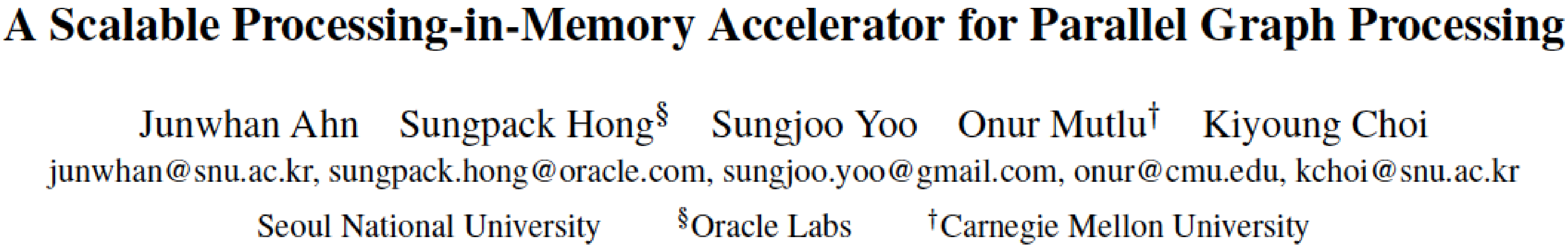


> 8X Energy Reduction

SAFARI Ahn+, "A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing" ISCA 2015.

More on Tesseract

- Junwhan Ahn, Sungpack Hong, Sungjoo Yoo, Onur Mutlu, and Kiyoung Choi,
"A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing"
Proceedings of the 42nd International Symposium on Computer Architecture (***ISCA***), Portland, OR, June 2015.
[Slides (pdf)] [Lightning Session Slides (pdf)]



A Scalable Processing-in-Memory Accelerator for Parallel Graph Processing

Junwhan Ahn Sungpack Hong§ Sungjoo Yoo Onur Mutlu† Kiyoung Choi
junwhan@snu.ac.kr, sungpack.hong@oracle.com, sungjoo.yoo@gmail.com, onur@cmu.edu, kchoi@snu.ac.kr
Seoul National University §Oracle Labs †Carnegie Mellon University

Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design

- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions

- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory

- Future Opportunities: New Sequencing Technologies

Recall: 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
    - Amplification step limits the read length since too short or too long fragments are not amplified well.

- 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

Nanopore Sequencing Technology

- **Nanopore sequencing** is an emerging and a promising single-molecule DNA sequencing technology

- 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

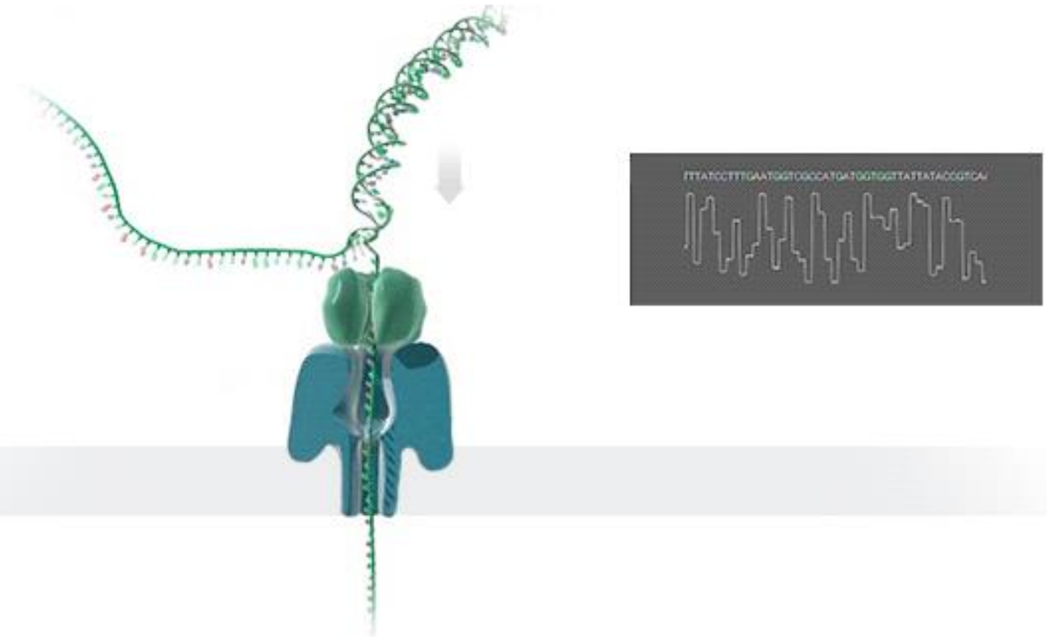
Nanopore Sequencing Technology

an emerging and a promising
ncing technology

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

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.

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

Nanopore Genome Assembly Pipeline

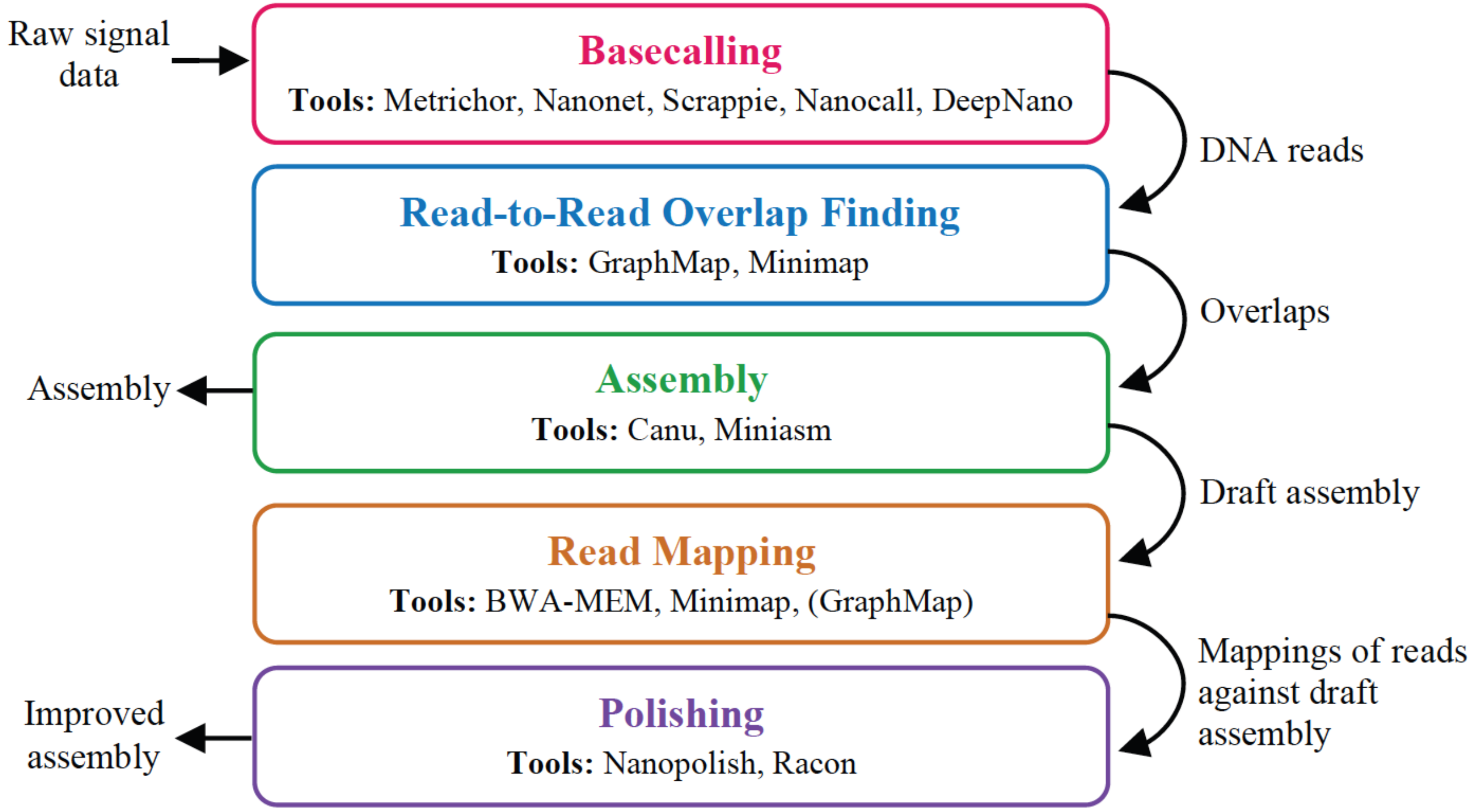


Figure 1. The analyzed genome assembly pipeline using nanopore sequence data, with its five steps and the associated tools for each step.

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

Nanopore Genome Assembly Tools (I)

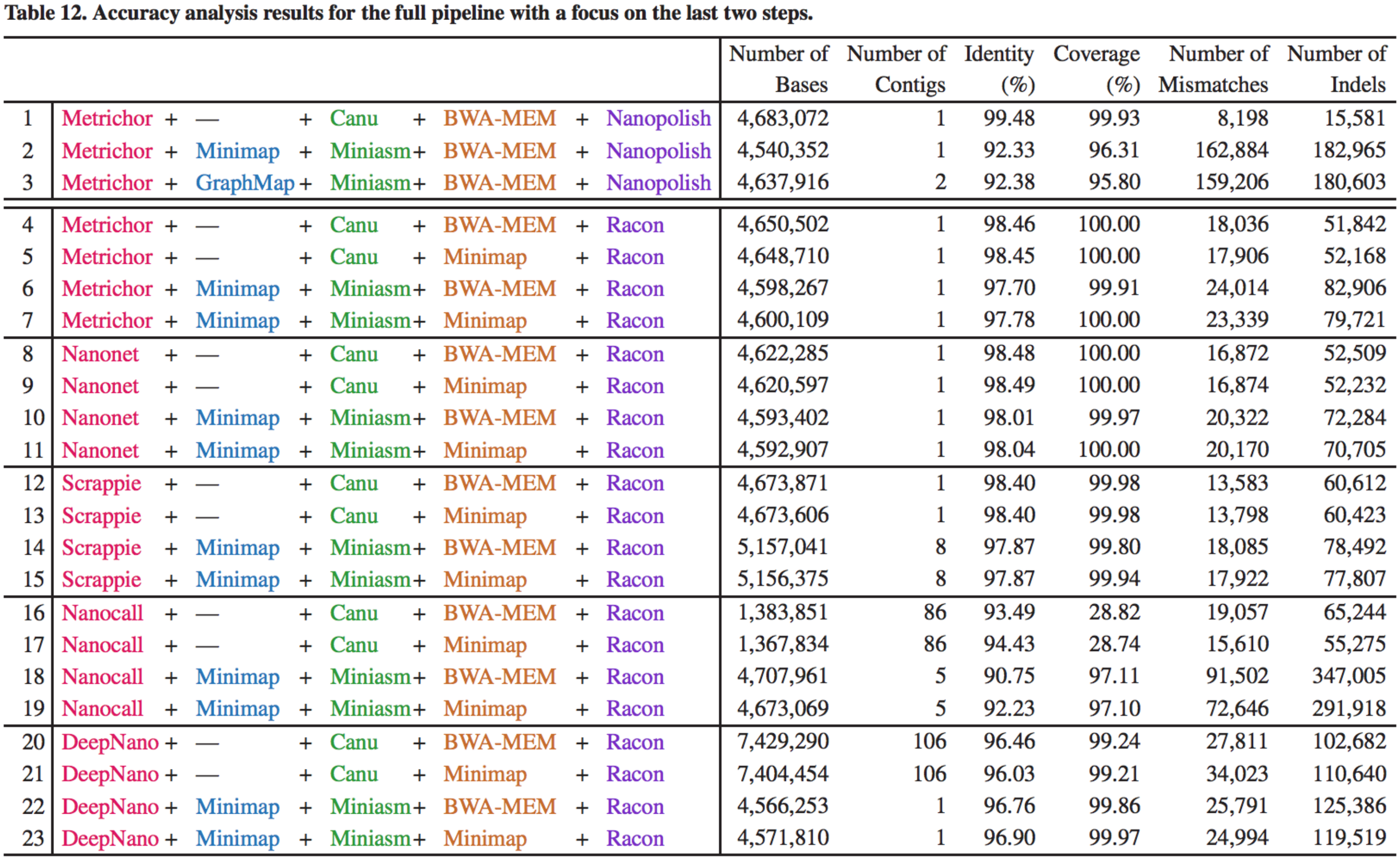


Table 12. Accuracy analysis results for the full pipeline with a focus on the last two steps.

										Number of	Number of	Identity	Coverage	Number of	Number of
										Bases	Contigs	(%)	(%)	Mismatches	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)

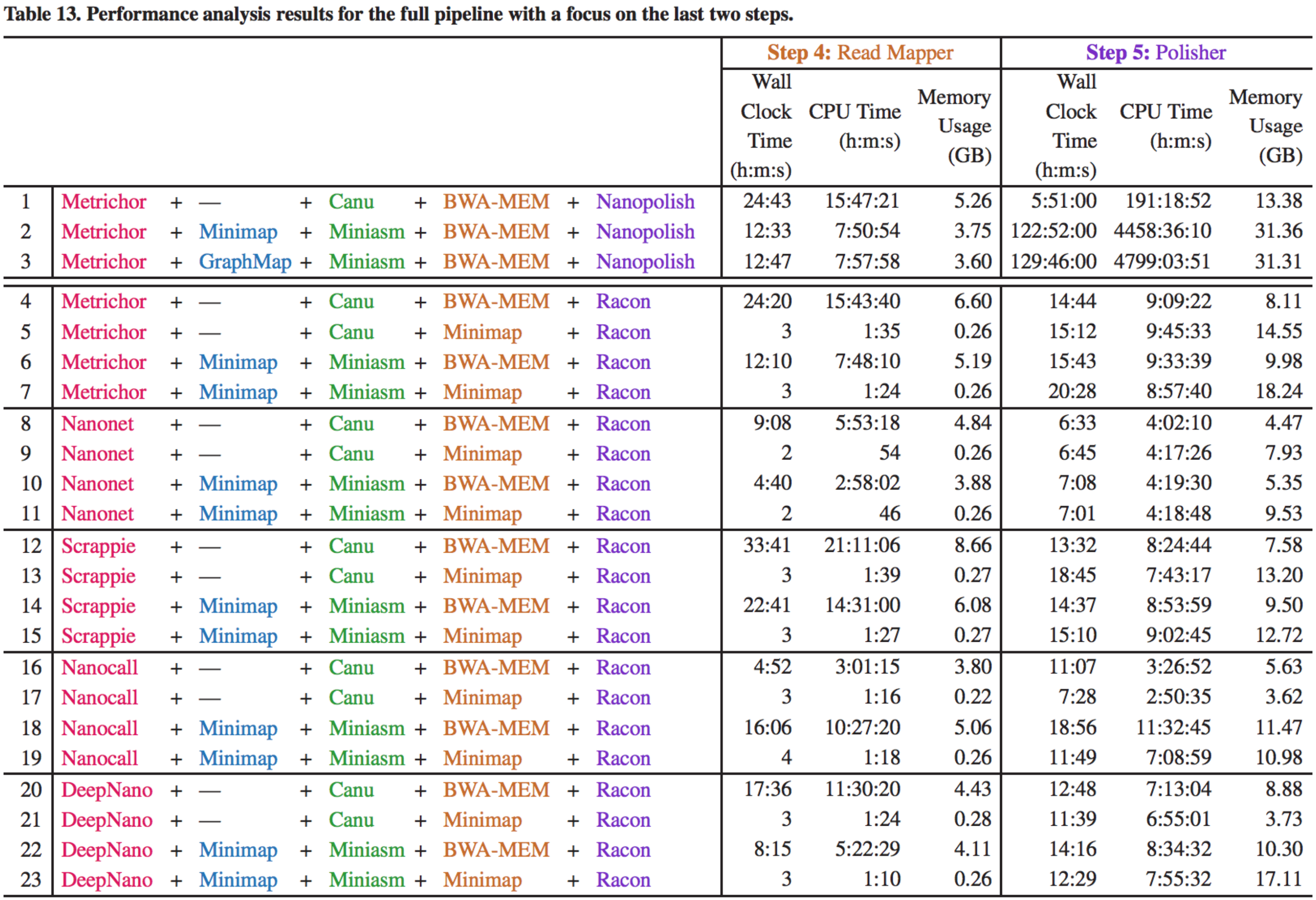
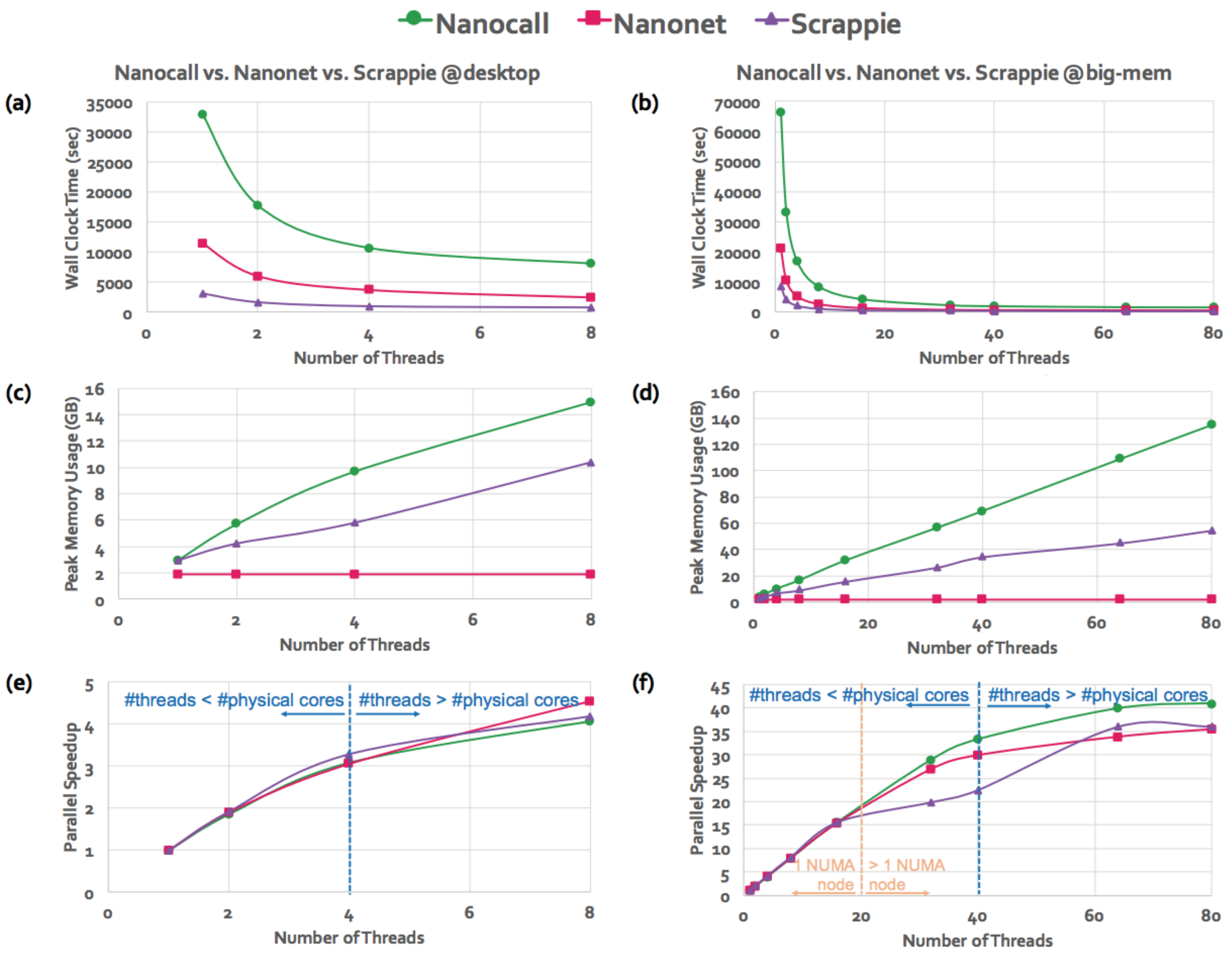


Table 13. Performance analysis results for the full pipeline with a focus on the last two steps.

						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

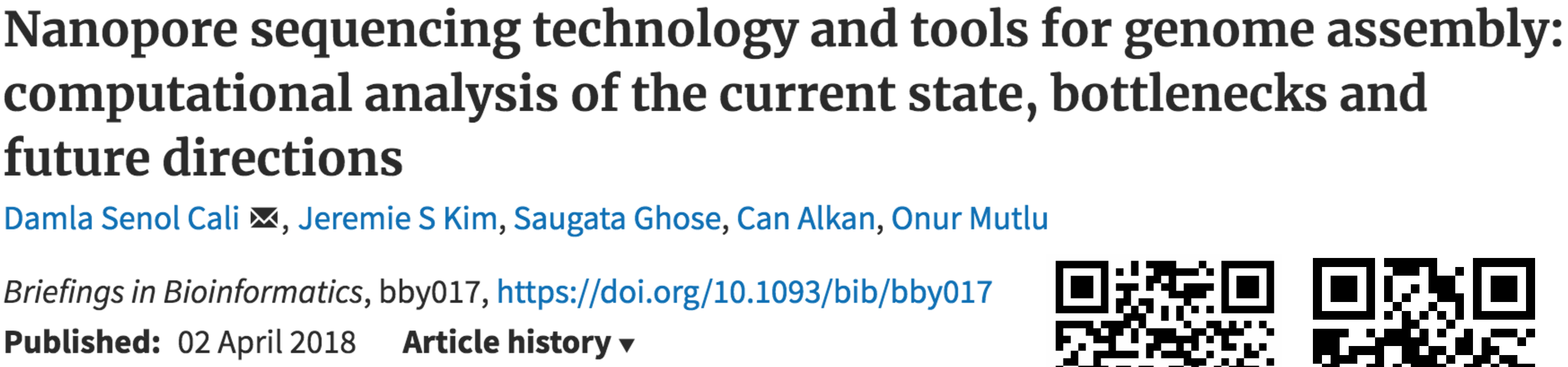
Senol Cali+, "**Nanopore Sequencing Technology and Tools for Genome Assembly**" Briefings in Bioinformatics, 2018.

Nanopore Genome Assembly Tools (III)



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

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]

Agenda

- The Problem: DNA Read Mapping
  - State-of-the-art Read Mapper Design
- Algorithmic Acceleration
  - Exploiting Structure of the Genome
  - Exploiting SIMD Instructions
- Hardware Acceleration
  - Specialized Architectures
  - Processing in Memory
- Future Opportunities: New Sequencing Technologies

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
  - My personal 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

omutlu@gmail.com

https://people.inf.ethz.ch/omutlu

16 February 2019

AACBB Keynote Talk

SAFARI ETH zürich Carnegie Mellon