Enabling Accurate, Fast, and Memory-Efficient Genome Analysis via Efficient and Intelligent Algorithms

Can Firtina

canfirtina@gmail.com

10 January 2022

METU Graduate Seminar in Bioinformatics

SAFARI

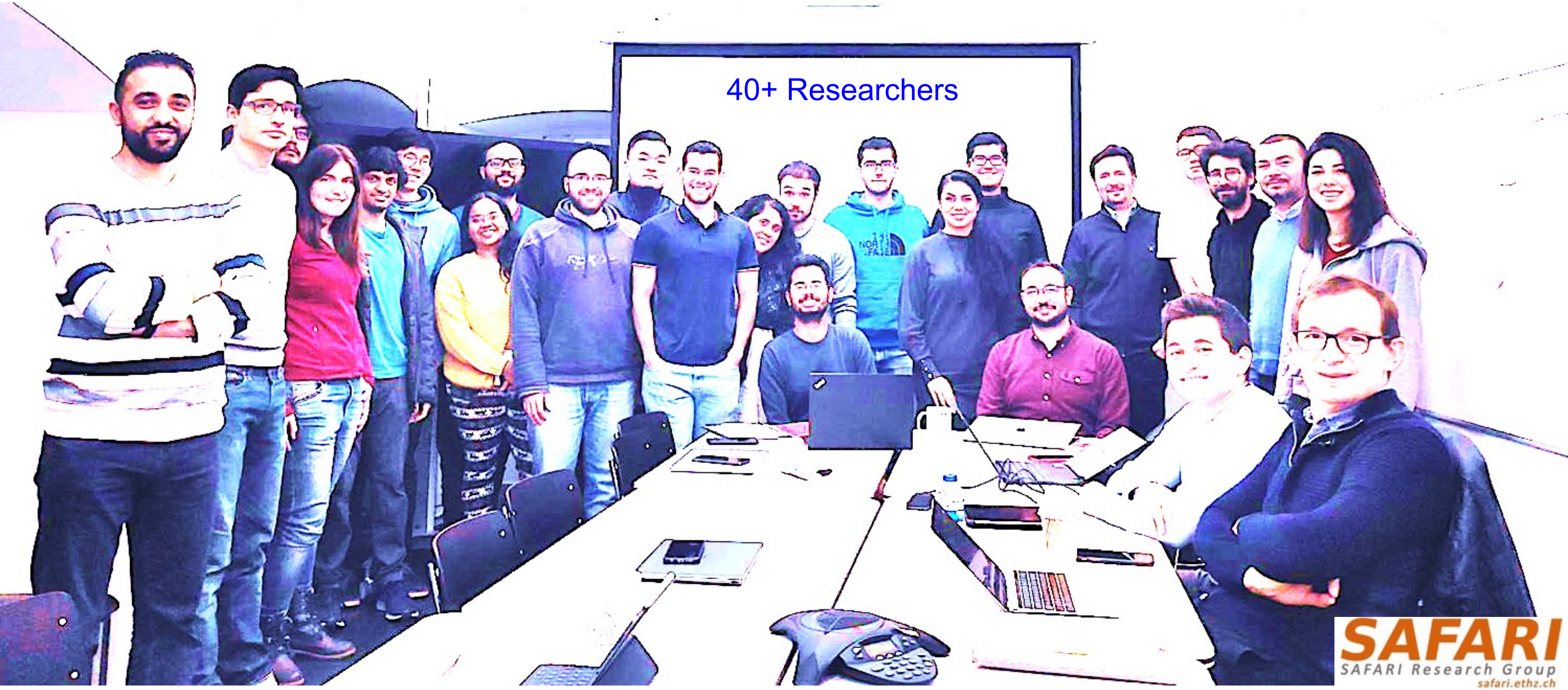
ETH zürich

Brief Self Introduction

- Ph.D. Student in the SAFARI Research Group
- Research interests:
  - Computational biology, accelerating genome analysis
- Some selected works:
  - Can Firtina+, "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches," *arXiv*, Dec. 2021.
  - Can Firtina+, "Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm," *Bioinformatics*, Jun. 2020.
  - Damla Senol Cali+ "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," in *MICRO 2020*.
  - Jeremie S. Kim+ "AirLift: A Fast and Comprehensive Technique for Translating Alignments between Reference Genomes," *arXiv*, 2019.
- Get to know us and our research
  - https://safari.ethz.ch/
  - Contact me: canfirtina@gmail.com

Onur Mutlu's SAFARI Research Group

Computer architecture, HW/SW, systems, bioinformatics, security, memory



Think BIG, Aim HIGH!

https://safari.ethz.ch

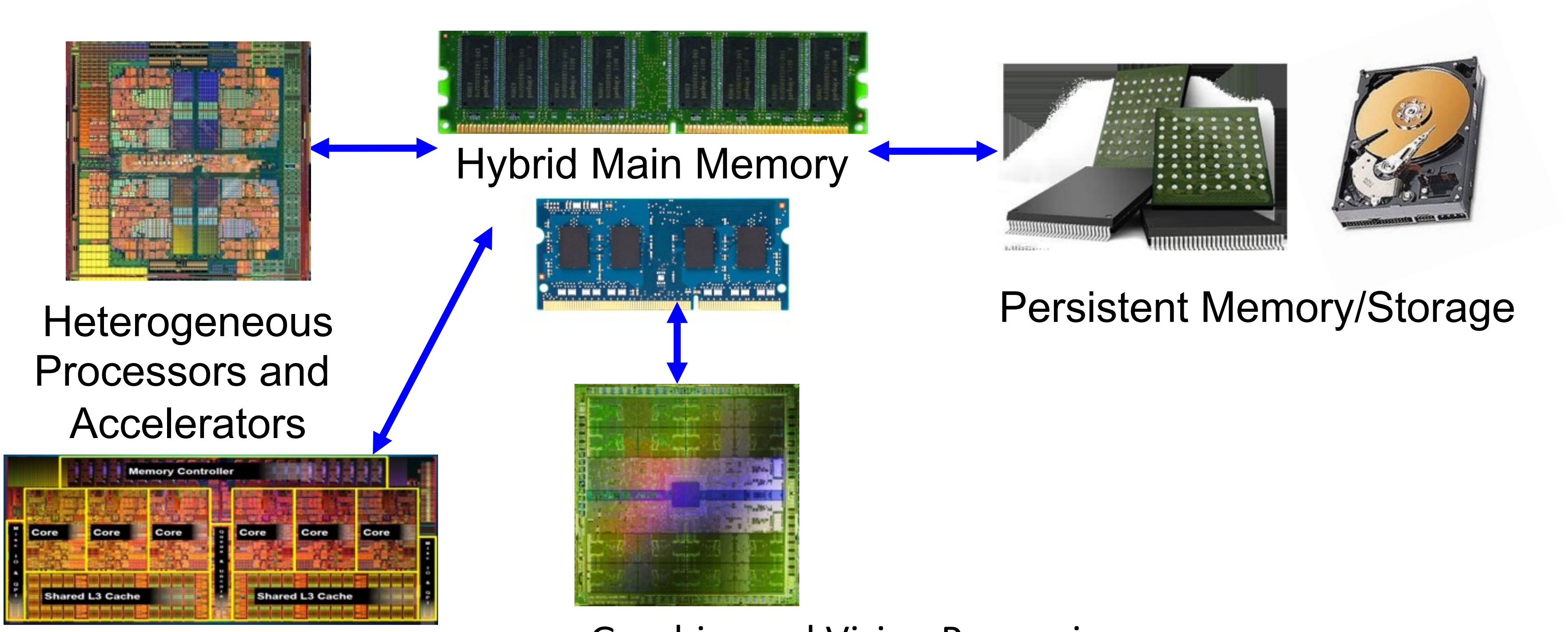
Professor Mutlu

- Onur Mutlu
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-…
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - https://people.inf.ethz.ch/omutlu/
  - omutlu@gmail.com (Best way to reach)
  - https://people.inf.ethz.ch/omutlu/projects.htm

- Research and Teaching in:
  - Computer architecture, computer systems, hardware security, bioinformatics
  - Memory and storage systems
  - Hardware security, safety, predictability
  - Fault tolerance
  - Hardware/software cooperation
  - Architectures for bioinformatics, health, medicine
  - …

Current Research Mission

Computer architecture, HW/SW, systems, bioinformatics, security



Hybrid Main Memory

Persistent Memory/Storage

Heterogeneous Processors and Accelerators

Graphics and Vision Processing

Build fundamentally better architectures

Four Key Current Directions

- Fundamentally Secure/Reliable/Safe Architectures

- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures

- Fundamentally Low-Latency and Predictable Architectures

- Architectures for AI/ML, Genomics, Medicine, Health

SAFARI Newsletter December 2021 Edition

- https://safari.ethz.ch/safari-newsletter-december-2021

Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZlSOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1
- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16
- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41lH-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41
- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3
- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LgzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25
- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39

Referenced Papers, Talks, Artifacts

- All are available at

  https://safari.ethz.ch

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

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/

Accelerating Genome Analysis

The Problem

Computing

is Bottlenecked by Data

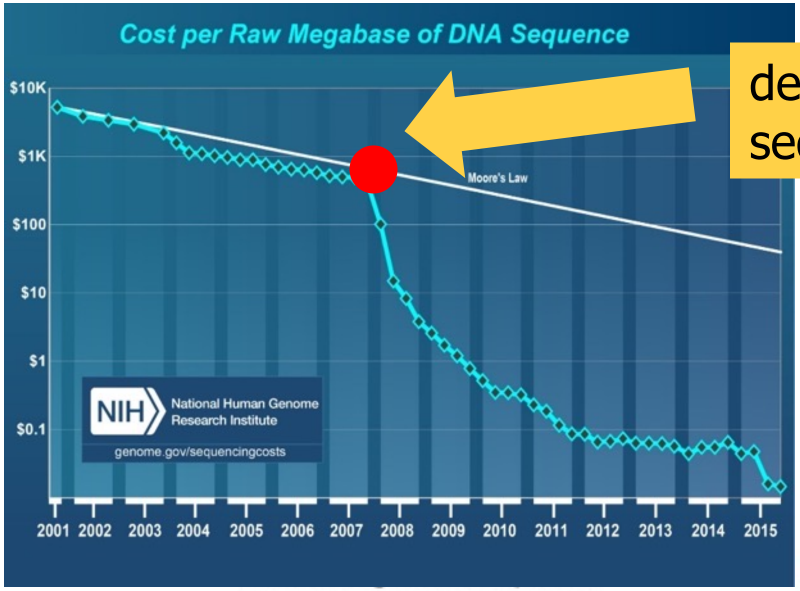
Data is Key for AI, ML, Genomics, …

- Important workloads are all data intensive

- They require rapid and efficient processing of large amounts of data

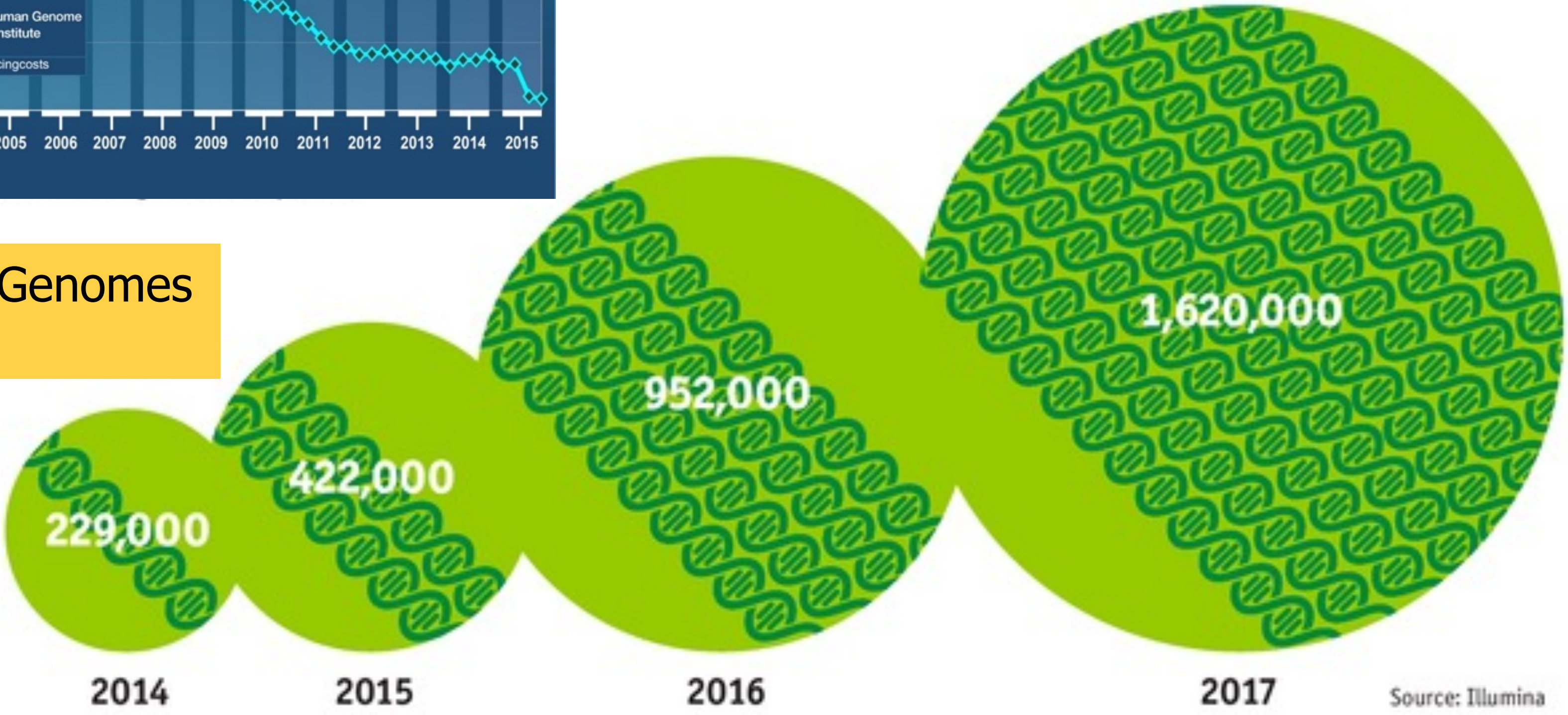
- Data is increasing
  - We can generate more than we can process

Data is Key for Future Workloads



development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

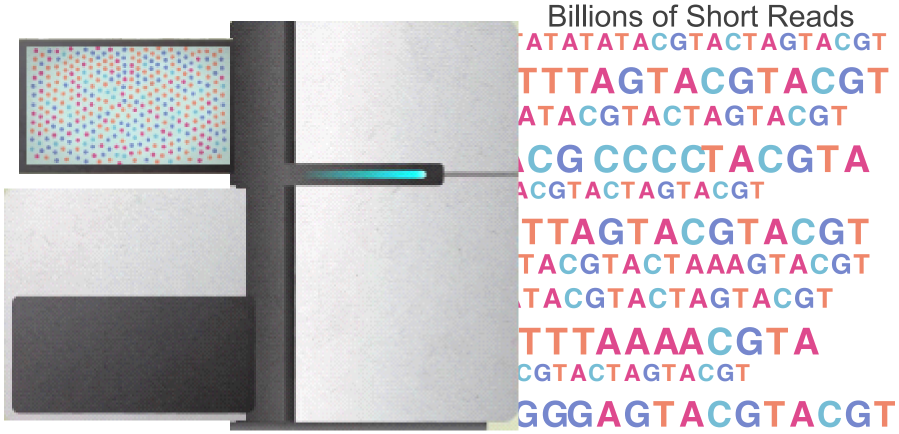


Source: Illumina

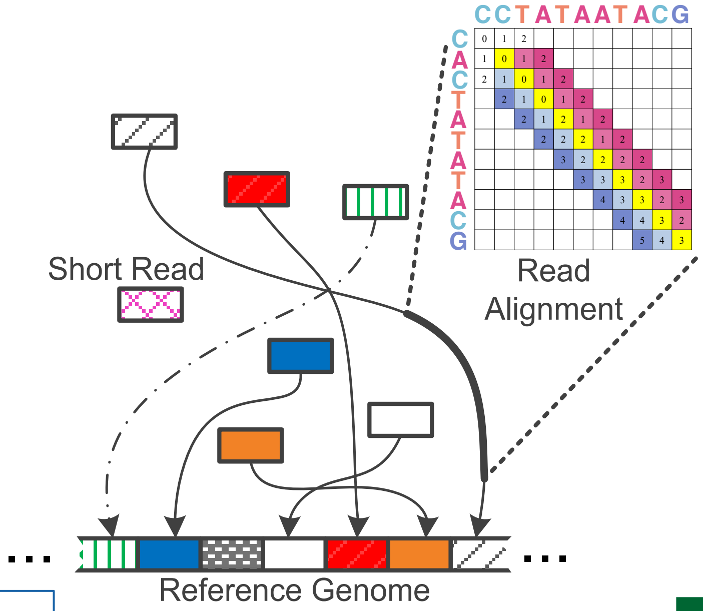
The Economist

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

Genome Analysis



1 Sequencing

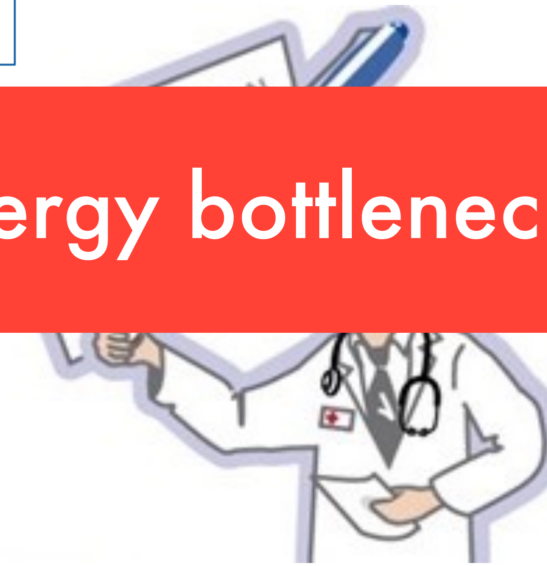


Read Mapping 2

Data → performance & energy bottleneck

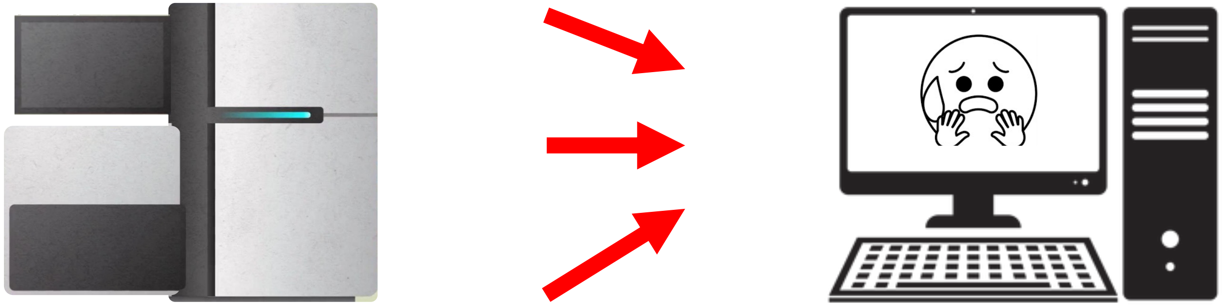
read4: CGCTTCCAT
read5: CCATGACGC
read6: TTCCATGAC

3 Variant Calling



Scientific Discovery 4

Lack of Specialized Compute Capability



Specialized Machine
for Sequencing

General-Purpose Machine
for Analysis

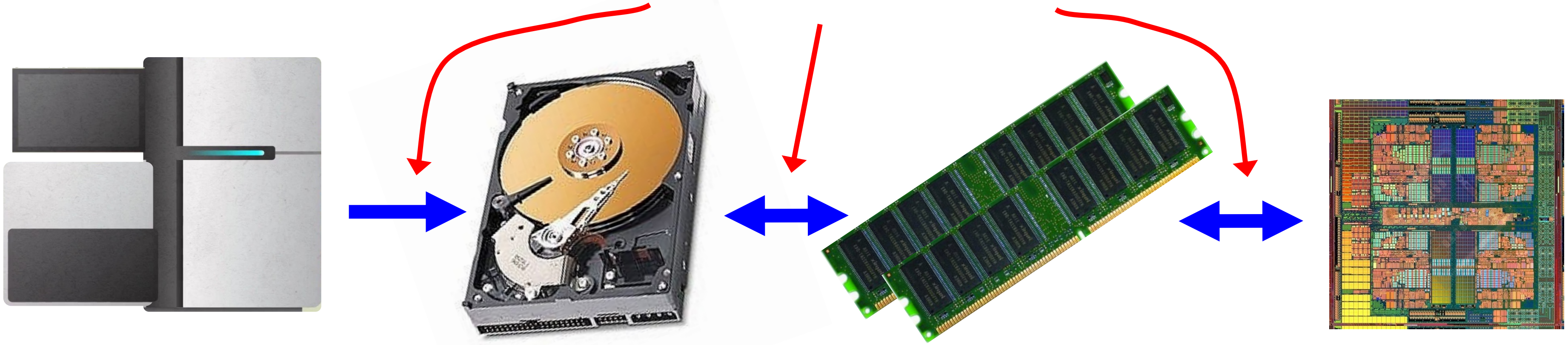
FAST

SLOW

Data Movement Dominates Performance

- **Data movement** dominates performance and is a **major** system **energy bottleneck** (accounting for 40%-62%)

Data Movement



Sequencing Machine

Storage (SSD/HDD)

Main Memory

Microprocessor

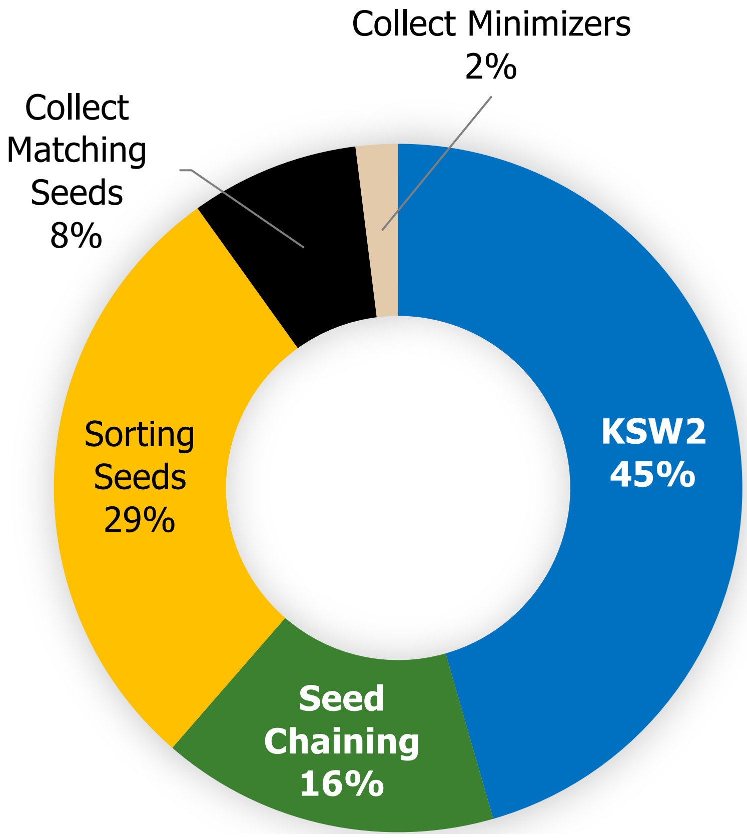
Single memory request consumes >160x-800x more energy compared to performing an addition operation

✻ Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018
★ Kestor et al., "Quantifying the Energy Cost of Data Movement in Scientific Applications," IISWC 2013
☆ Pandiyan and Wu, "Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms," IISWC 2014

Read Mapping Execution Time

>60%

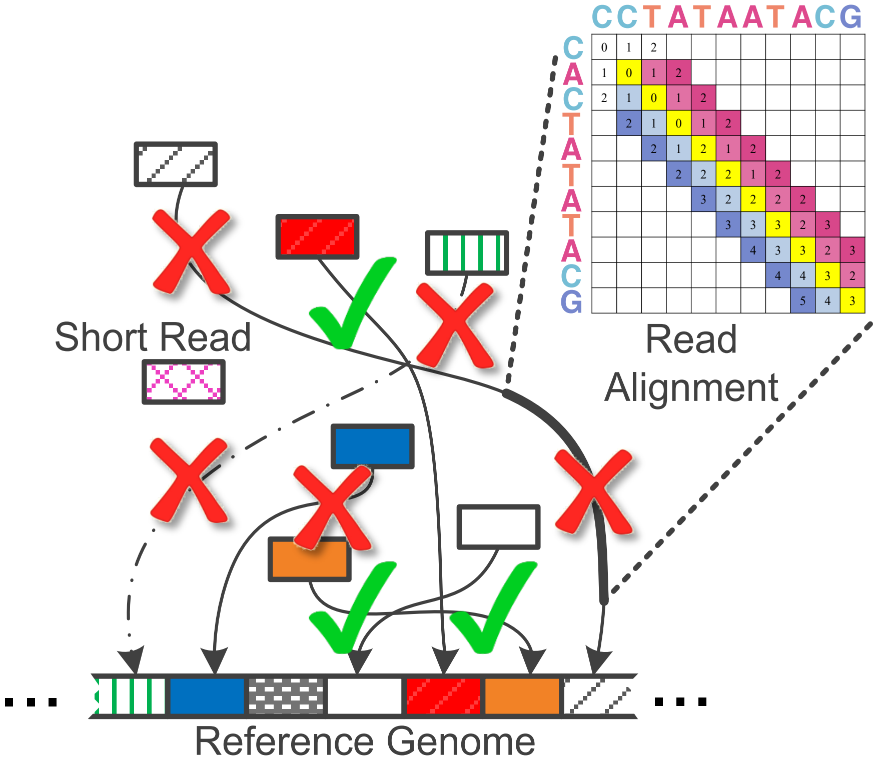
of the read mapper's execution time is spent in sequence alignment



minimap2

ONT FASTQ size: 103MB (151 reads), Mean length: 356,403 bp, std: 173,168 bp, longest length: 817,917 bp

Large Search Space for Mapping Location



	C	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

98%

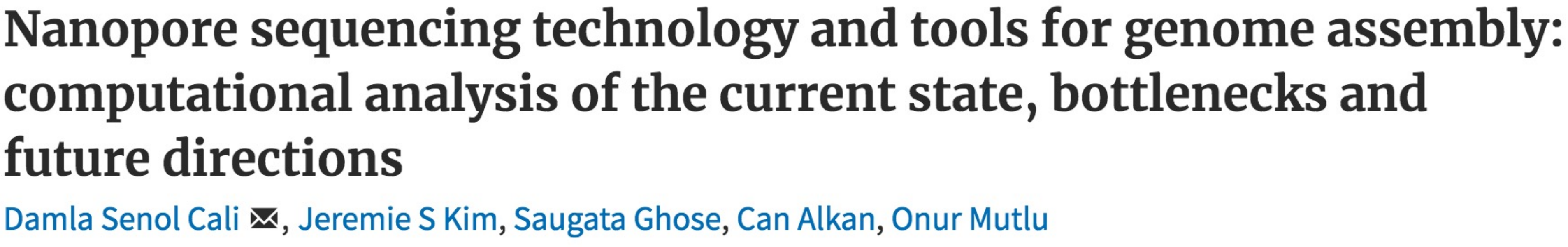
of candidate locations

have high dissimilarity

with a given read

Cheng *et al*, *BMC bioinformatics* (2015)
Xin *et al*, *BMC genomics* (2013)

New Genome Sequencing Technologies



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

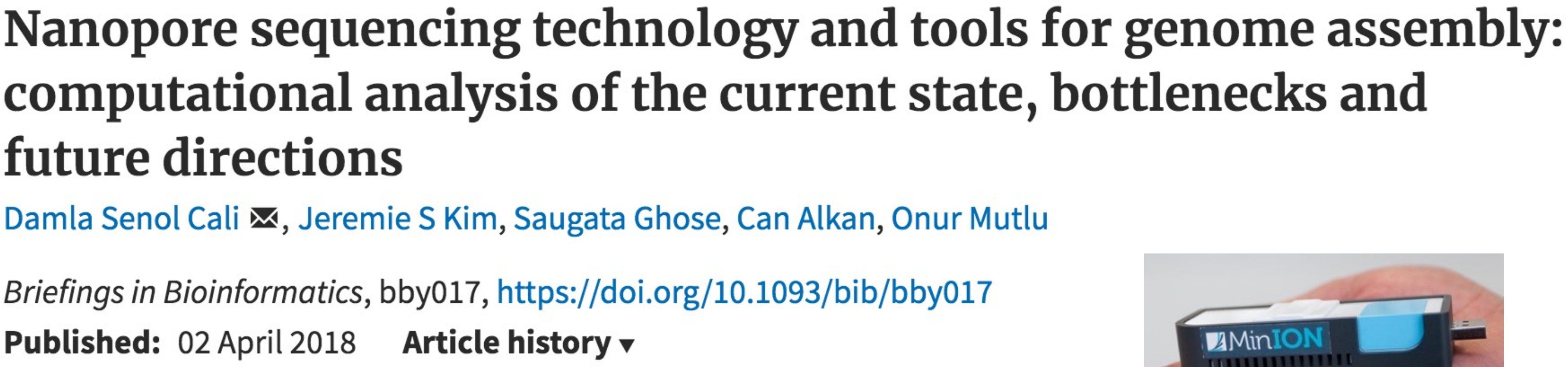
Briefings in Bioinformatics, bby017, https://doi.org/10.1093/bib/bby017

Published: 02 April 2018 **Article history ▾**

Oxford Nanopore MinION

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

New Genome Sequencing Technologies



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** ▾

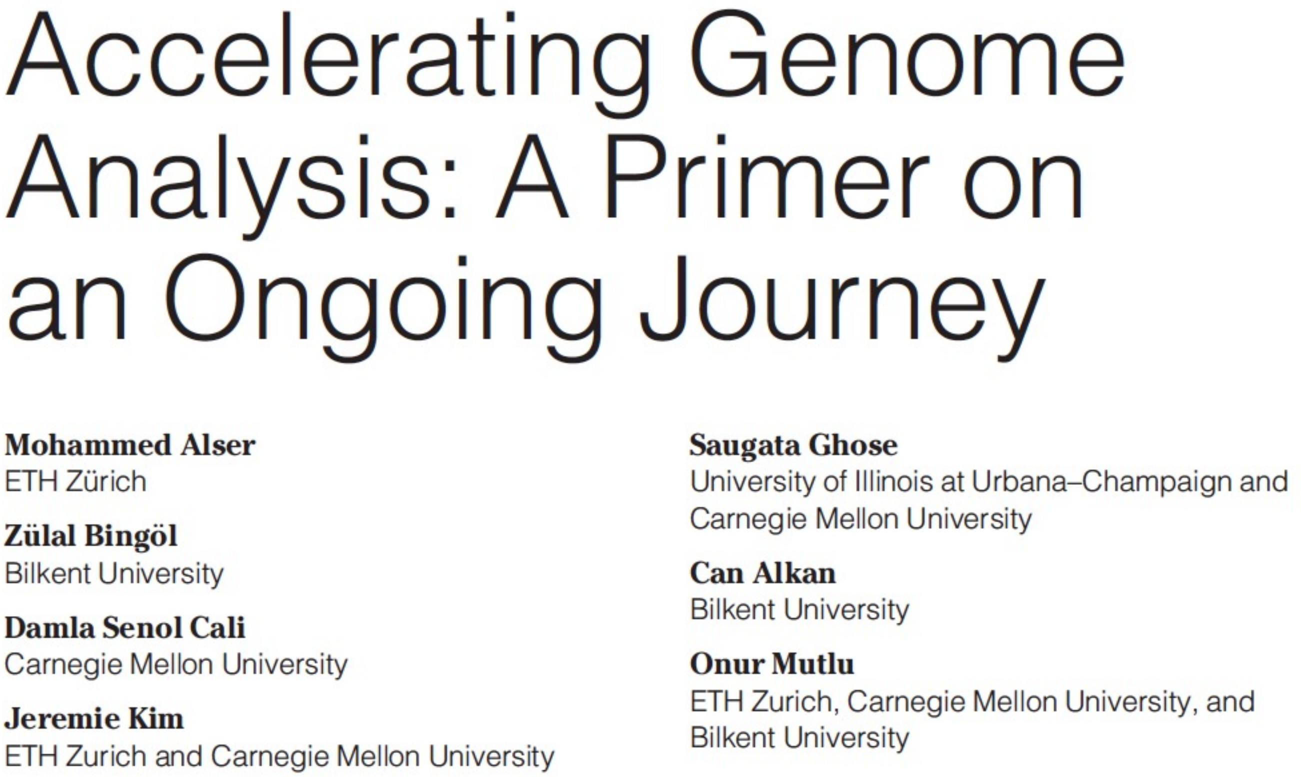
Oxford Nanopore MinION

Data → performance & energy bottleneck

We need intelligent algorithms and intelligent architectures that handle data well

Accelerating Genome Analysis [IEEE MICRO 2020]

- Mohammed Alser, Zulal Bingol, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, and Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  IEEE Micro (***IEEE MICRO***), Vol. 40, No. 5, pages 65-75, September/October 2020.
  [Slides (pptx)(pdf)]
  [Talk Video (1 hour 2 minutes)]



Accelerating Genome Analysis: A Primer on an Ongoing Journey

Mohammed Alser
ETH Zürich

Zülal Bingöl
Bilkent University

Damla Senol Cali
Carnegie Mellon University

Jeremie Kim
ETH Zurich and Carnegie Mellon University

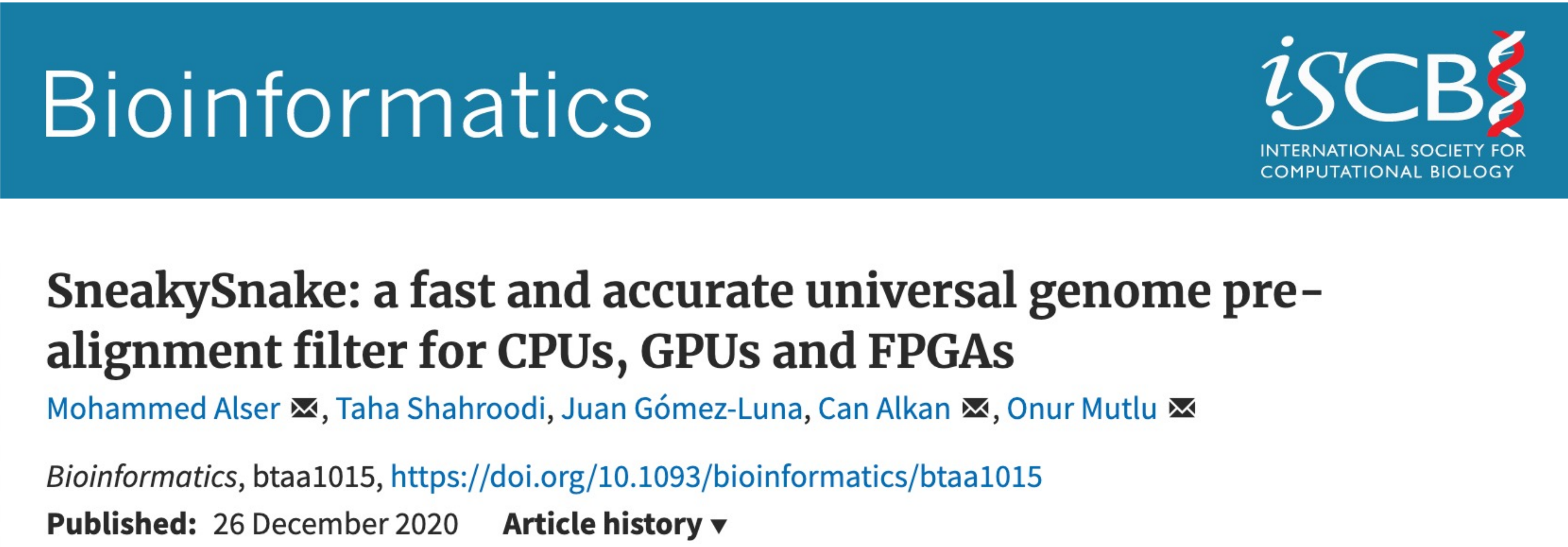
Saugata Ghose
University of Illinois at Urbana–Champaign and Carnegie Mellon University

Can Alkan
Bilkent University

Onur Mutlu
ETH Zurich, Carnegie Mellon University, and Bilkent University

Specialized Hardware for Pre-alignment Filtering

Mohammed Alser, Taha Shahroodi, Juan-Gomez Luna, Can Alkan, and Onur Mutlu,
"SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs"
Bioinformatics, 2020.
[Source Code]
[Online link at Bioinformatics Journal]



SneakySnake: a fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs

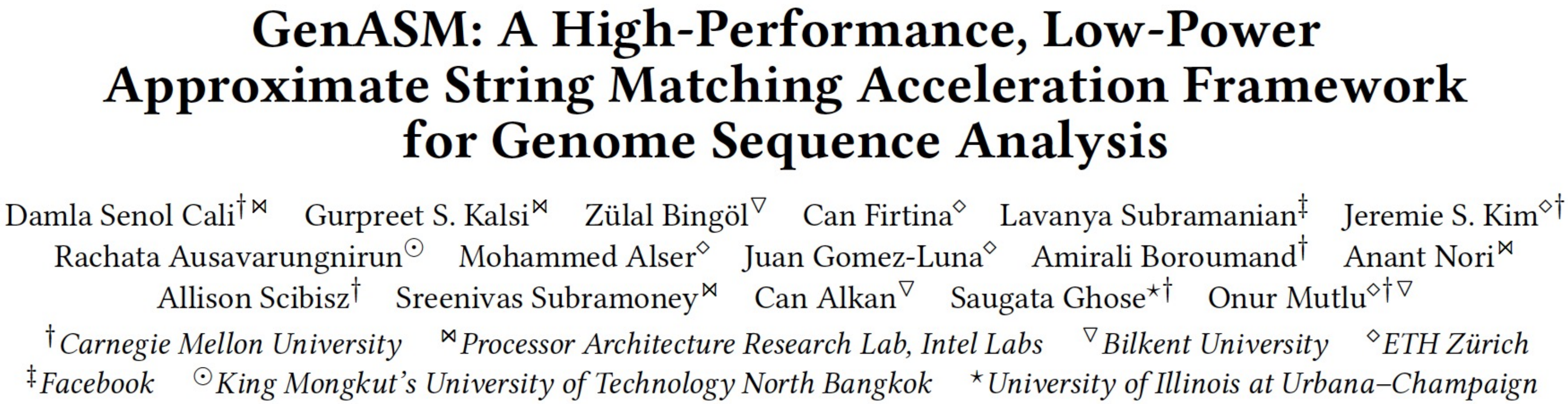
Mohammed Alser ✉, Taha Shahroodi, Juan Gómez-Luna, Can Alkan ✉, Onur Mutlu ✉

Bioinformatics, btaa1015, https://doi.org/10.1093/bioinformatics/btaa1015

Published: 26 December 2020 **Article history ▾**

GenASM Framework **[MICRO 2020]**

- Damla Senol Cali, Gurpreet S. Kalsi, Zulal Bingol, Can Firtina, Lavanya Subramanian, Jeremie S. Kim, Rachata Ausavarungnirun, Mohammed Alser, Juan Gomez-Luna, Amirali Boroumand, Anant Nori, Allison Scibisz, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis"
  Proceedings of the 53rd International Symposium on Microarchitecture (***MICRO***), Virtual, October 2020.
  [Lighting Talk Video (1.5 minutes)]
  [Lightning Talk Slides (pptx) (pdf)]
  [Talk Video (18 minutes)]
  [Slides (pptx) (pdf)]



GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis

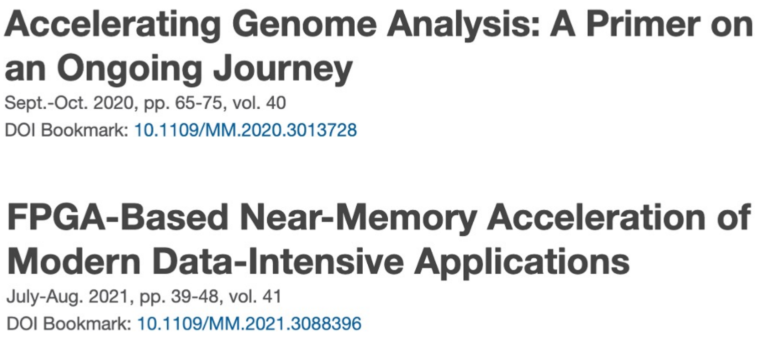
Damla Senol Cali[†⋈] Gurpreet S. Kalsi[⋈] Zülal Bingöl[▽] Can Firtina[◇] Lavanya Subramanian[‡] Jeremie S. Kim[◇†]
Rachata Ausavarungnirun[⊙] Mohammed Alser[◇] Juan Gomez-Luna[◇] Amirali Boroumand[†] Anant Nori[⋈]
Allison Scibisz[†] Sreenivas Subramoney[⋈] Can Alkan[▽] Saugata Ghose[⋆†] Onur Mutlu[◇†▽]

[†] *Carnegie Mellon University* [⋈] *Processor Architecture Research Lab, Intel Labs* [▽] *Bilkent University* [◇] *ETH Zürich*
[‡] *Facebook* [⊙] *King Mongkut's University of Technology North Bangkok* [⋆] *University of Illinois at Urbana–Champaign*

Future of Genome Sequencing & Analysis

Mohammed Alser, Zülal Bingöl, Damla Senol Cali, Jeremie Kim, Saugata Ghose, Can Alkan, Onur Mutlu
“Accelerating Genome Analysis: A Primer on an Ongoing Journey” IEEE Micro, August 2020.

MinION from ONT



Accelerating Genome Analysis: A Primer on an Ongoing Journey
Sept.-Oct. 2020, pp. 65-75, vol. 40
DOI Bookmark: 10.1109/MM.2020.3013728

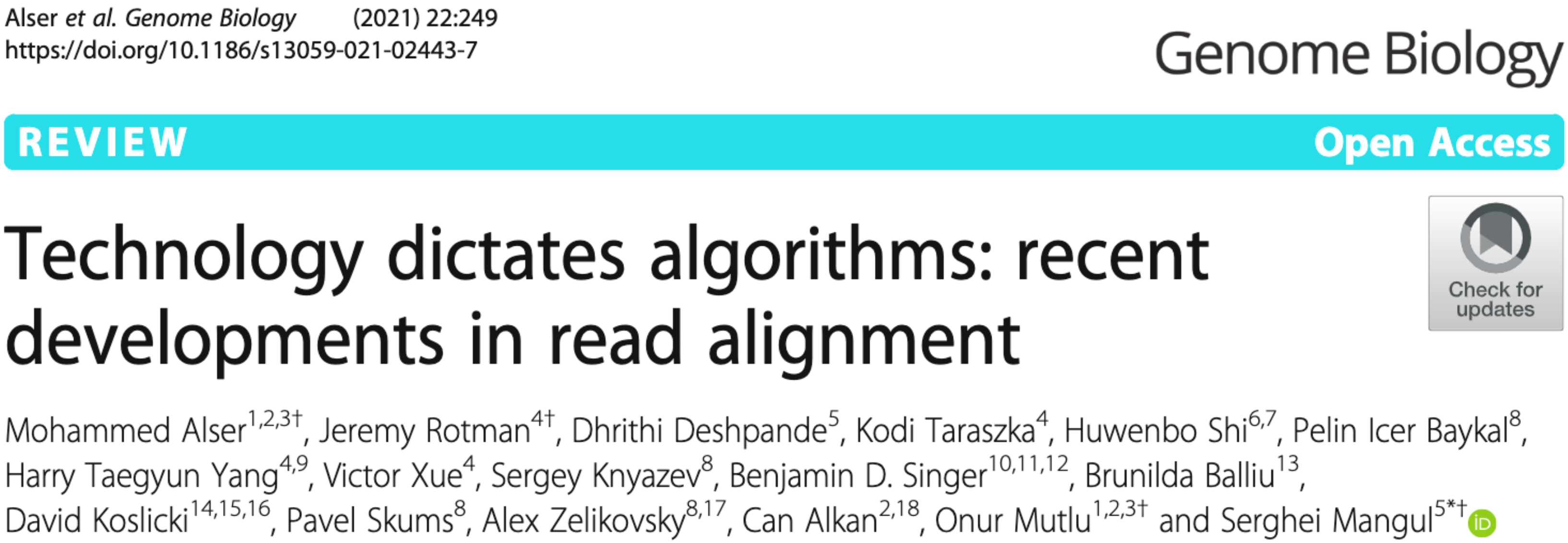
FPGA-Based Near-Memory Acceleration of Modern Data-Intensive Applications
July-Aug. 2021, pp. 39-48, vol. 41
DOI Bookmark: 10.1109/MM.2021.3088396

SmidgION from ONT

Read Mapping in 111 pages!

In-depth analysis of 107 read mappers (1988-2020)

Mohammed Alser, Jeremy Rotman, Dhrithi Deshpande, Kodi Taraszka, Huwenbo Shi, Pelin Icer Baykal, Harry Taegyun Yang, Victor Xue, Sergey Knyazev, Benjamin D. Singer, Brunilda Balliu, David Koslicki, Pavel Skums, Alex Zelikovsky, Can Alkan, Onur Mutlu, Serghei Mangul
"Technology dictates algorithms: Recent developments in read alignment"
Genome Biology, 2021
[Source code]



Alser *et al. Genome Biology* (2021) 22:249
https://doi.org/10.1186/s13059-021-02443-7

Genome Biology

REVIEW

Open Access

Technology dictates algorithms: recent developments in read alignment

Mohammed Alser[1,2,3†], Jeremy Rotman[4†], Dhrithi Deshpande[5], Kodi Taraszka[4], Huwenbo Shi[6,7], Pelin Icer Baykal[8], Harry Taegyun Yang[4,9], Victor Xue[4], Sergey Knyazev[8], Benjamin D. Singer[10,11,12], Brunilda Balliu[13], David Koslicki[14,15,16], Pavel Skums[8], Alex Zelikovsky[8,17], Can Alkan[2,18], Onur Mutlu[1,2,3†] and Serghei Mangul[5*†]

More on Fast & Efficient Genome Analysis …

- Onur Mutlu,
  "Accelerating Genome Analysis: A Primer on an Ongoing Journey"
  Invited Lecture at Technion, Virtual, 26 January 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 37 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]

Population-Scale Microbiome Profiling

SAFARI https://blog.wego.com/7-crowded-places-and-events-that-you-will-love/ 30

15:35 / 1:37:37

Onur Mutlu - Invited Lecture @Technion: Accelerating Genome Analysis: A Primer on an Ongoing Journey

740 views • Premiered Feb 6, 2021

Onur Mutlu Lectures
15.9K subscribers

https://www.youtube.com/watch?v=r7sn41lH-4A

Detailed Lectures on Genome Analysis

- Computer Architecture, Fall 2020, Lecture 3a
  - **Introduction to Genome Sequence Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=CrRb32v7SJc&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=5
- Computer Architecture, Fall 2020, Lecture 8
  - **Intelligent Genome Analysis** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=ygmQpdDTL7o&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=14
- Computer Architecture, Fall 2020, Lecture 9a
  - **GenASM: Approx. String Matching Accelerator** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=XoLpzmN-Pas&list=PL5Q2soXY2Zi9xidyIgBxUz7xRPS-wisBN&index=15
- Accelerating Genomics Project Course, Fall 2020, Lecture 1
  - **Accelerating Genomics** (ETH Zürich, Fall 2020)
  - https://www.youtube.com/watch?v=rgjl8ZyLsAg&list=PL5Q2soXY2Zi9E2bBVAgCqLgwiDRQDTyId

https://www.youtube.com/onurmutlulectures

A Blueprint for Fundamentally Better Architectures

- Onur Mutlu,
"Intelligent Architectures for Intelligent Computing Systems"
Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (***DATE***), Virtual, February 2021.
[Slides (pptx) (pdf)]
[IEDM Tutorial Slides (pptx) (pdf)]
[Short DATE Talk Video (11 minutes)]
[Longer IEDM Tutorial Video (1 hr 51 minutes)]

Intelligent Architectures for Intelligent Computing Systems

Onur Mutlu
ETH Zurich
omutlu@gmail.com

Analysis of first publicly available PIM Architecture

- Juan Gomez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu,
"Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware"
*Invited Paper at Workshop on Computing with Unconventional Technologies (**CUT**)*, Virtual, October 2021.
[arXiv version]
[PrIM Benchmarks Source Code]
[Slides (pptx) (pdf)]
[Talk Video (37 minutes)]
[Lightning Talk Video (3 minutes)]

Benchmarking Memory-Centric Computing Systems: Analysis of Real Processing-in-Memory Hardware

Juan Gómez-Luna, *ETH Zürich*
Izzat El Hajj, *American University of Beirut*
Ivan Fernandez, *University of Malaga*
Christina Giannoula, *National Technical University of Athens*
Geraldo F. Oliveira, *ETH Zürich*
Onur Mutlu, *ETH Zürich*

More on Processing in Memory

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, **"A Modern Primer on Processing in Memory"**
Invited Book Chapter in ***Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann***, Springer, to be published in 2021.

A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

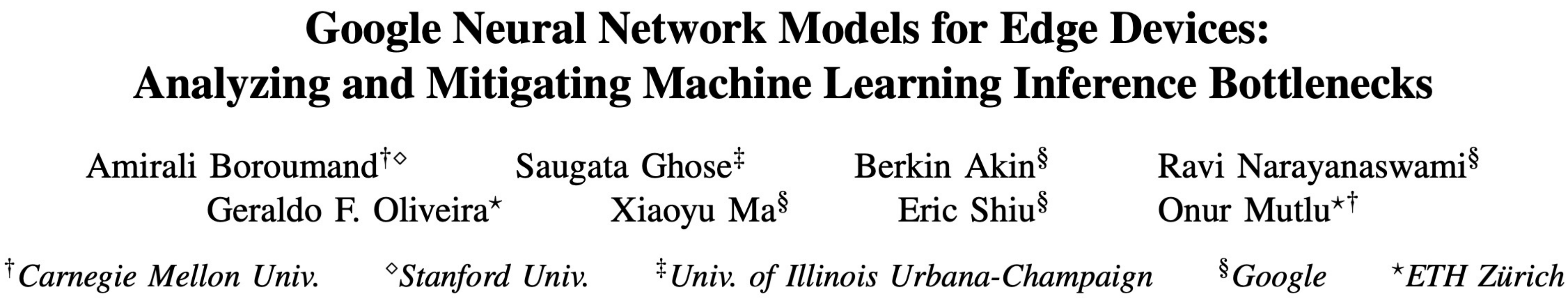
SAFARI Research Group

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

https://arxiv.org/pdf/1903.03988.pdf

Accelerating Neural Network Inference

- Amirali Boroumand, Saugata Ghose, Berkin Akin, Ravi Narayanaswami, Geraldo F. Oliveira, Xiaoyu Ma, Eric Shiu, and Onur Mutlu,
  "Google Neural Network Models for Edge Devices: Analyzing and Mitigating Machine Learning Inference Bottlenecks"
  *Proceedings of the 30th International Conference on Parallel Architectures and Compilation Techniques (**PACT**)*, Virtual, September 2021.
  [Slides (pptx) (pdf)]
  [Talk Video (14 minutes)]



Google Neural Network Models for Edge Devices:
Analyzing and Mitigating Machine Learning Inference Bottlenecks

Amirali Boroumand[†◇] Saugata Ghose[‡] Berkin Akin[§] Ravi Narayanaswami[§]
Geraldo F. Oliveira[⋆] Xiaoyu Ma[§] Eric Shiu[§] Onur Mutlu[⋆†]

[†]*Carnegie Mellon Univ.* [◇]*Stanford Univ.* [‡]*Univ. of Illinois Urbana-Champaign* [§]*Google* [⋆]*ETH Zürich*

We need intelligent algorithms and intelligent architectures that handle data well

Agenda for Today - Intelligent Algorithms

- Apollo
  - Correcting errors in genome assembly for more accurate genome analysis

- AirLift
  - Avoiding redundant computations when moving from one reference genome to another

- BLEND
  - Using fewer and high-quality seeds for faster, more memory-efficient, and more accurate sequence similarity identification

Agenda for Today - Intelligent Algorithms

- Apollo
  - Correcting errors in genome assembly for more accurate genome analysis
- AirLift
  - Avoiding redundant computations when moving from one reference genome to another
- BLEND
  - Using fewer and high-quality seeds for faster, more memory-efficient, and more accurate sequence similarity identification

Assembly Polishing using ML

- Can Firtina, Jeremie S. Kim, Mohammed Alser, Damla Senol Cali, A. Ercument Cicek, Can Alkan, and Onur Mutlu,
  "Apollo: A Sequencing-Technology-Independent, Scalable, and Accurate Assembly Polishing Algorithm"
  Bioinformatics, June 2020.
  [Source Code]
  [Online link at Bioinformatics Journal]

Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm

FREE

Can Firtina, Jeremie S Kim, Mohammed Alser, Damla Senol Cali, A Ercument Cicek, Can Alkan ✉, Onur Mutlu ✉

Bioinformatics, Volume 36, Issue 12, 15 June 2020, Pages 3669–3679, https://doi.org/10.1093/bioinformatics/btaa179

Published: 13 March 2020 **Article history ▾**

Executive Summary

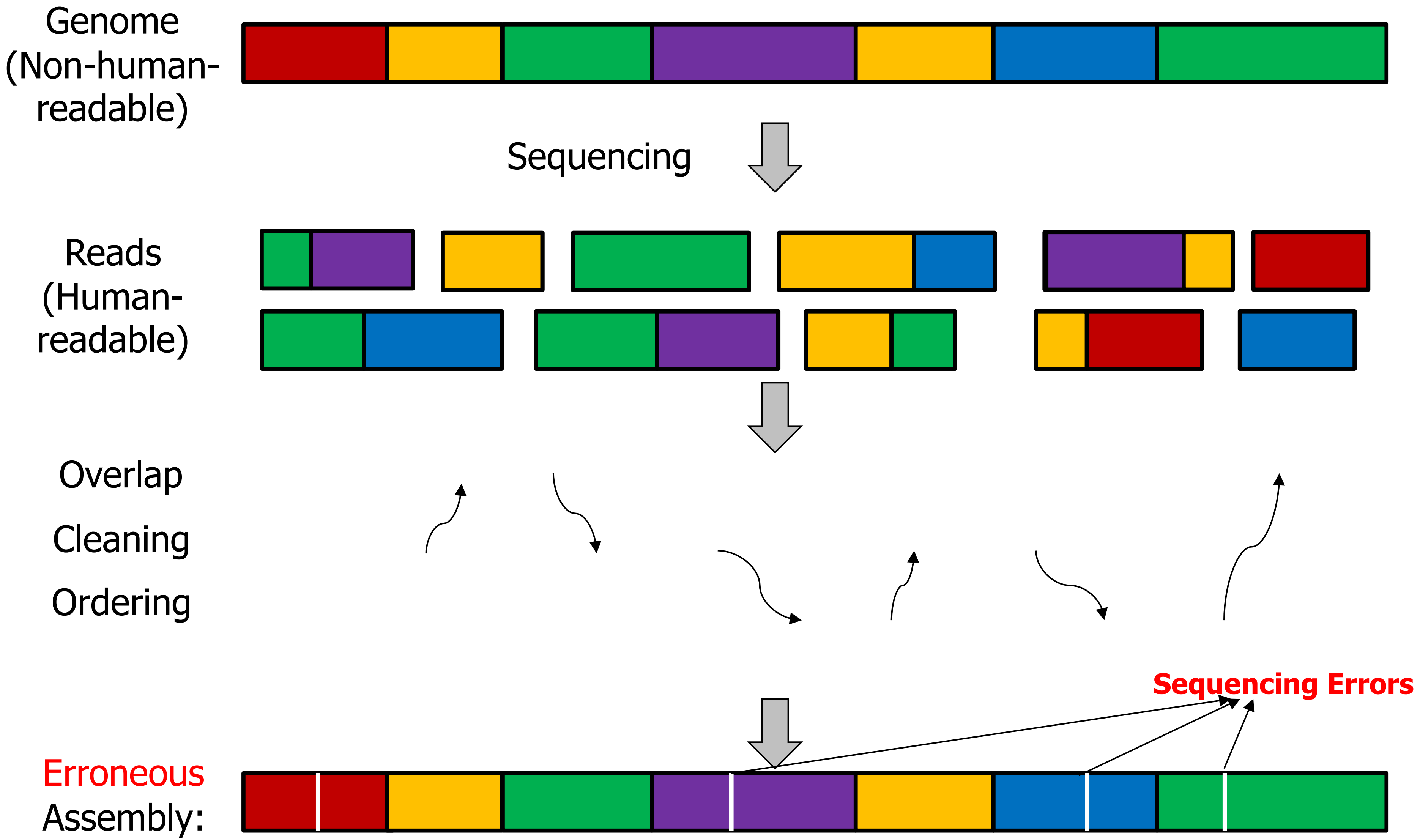
- **Problem:**
  - Long read de-novo assembly is inherently erroneous
  - Existing assembly polishing techniques cannot adapt to varying sequencing technologies and do not scale well for large genomes

- **Goal:** Propose a technology-independent and scalable assembly polishing algorithm – Apollo

- **Key Ideas:**
  - Use a probabilistic graph structure to identify the errors in assemblies accurately
    - Profile hidden Markov models (pHMMs)
  - Realign erroneous reads to the pHMM graph of an assembly to correct the errors by updating the probabilities based on a machine learning technique

- **Results**
  - Apollo is the only assembly polishing that is scalable to polish large genomes given the limited memory constraints (e.g., 192GB)
  - Apollo constructs the most reliable assemblies when hybrid set of reads (e.g., Illumina and PacBio) are used in a single run compared to other polishing tools
  - Apollo is ~25x slower on average (up to ~600x) than other polishers

Genome Assembly from Erroneous Reads



Genome (Non-human-readable)

Sequencing

Reads (Human-readable)

Overlap

Cleaning

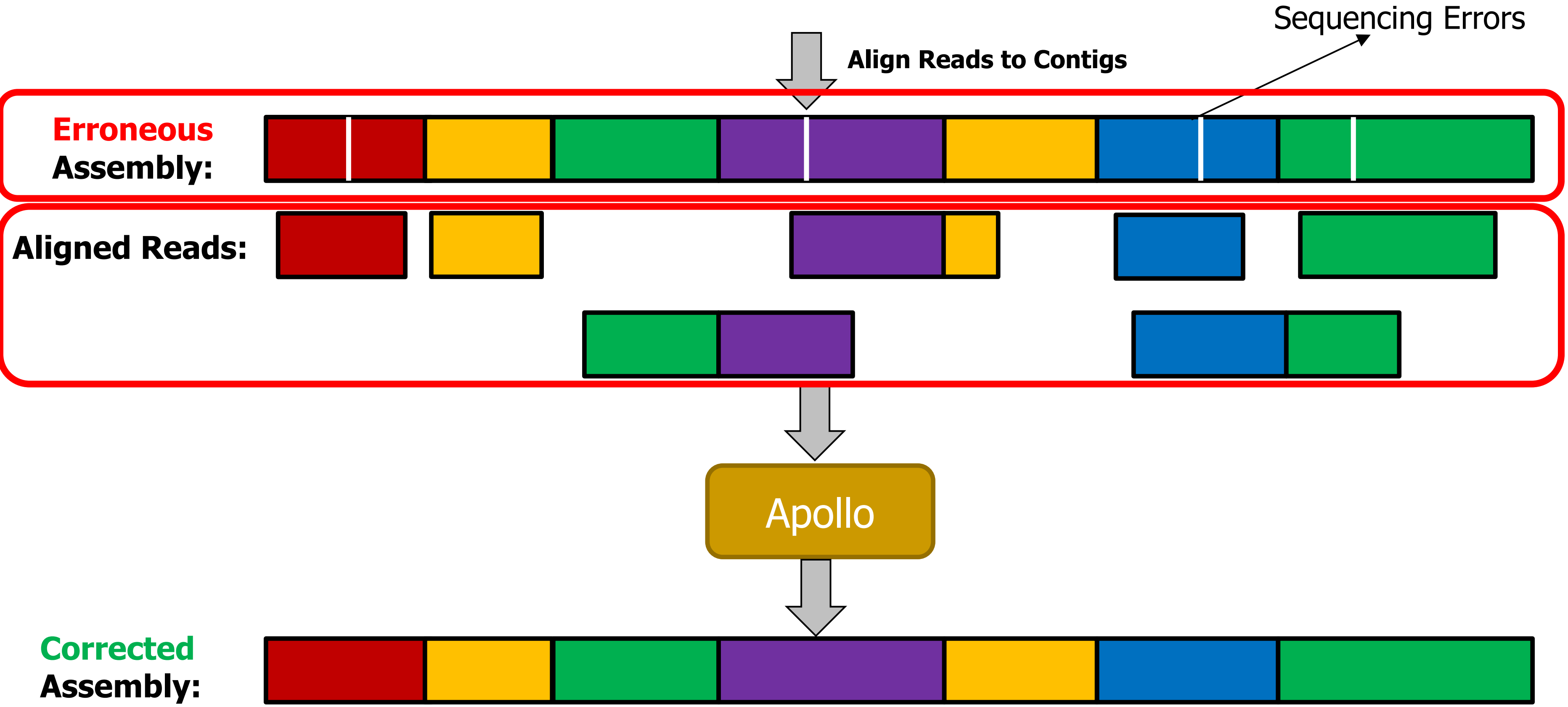
Ordering

Sequencing Errors

Erroneous Assembly:

Assembly Polishing with Apollo

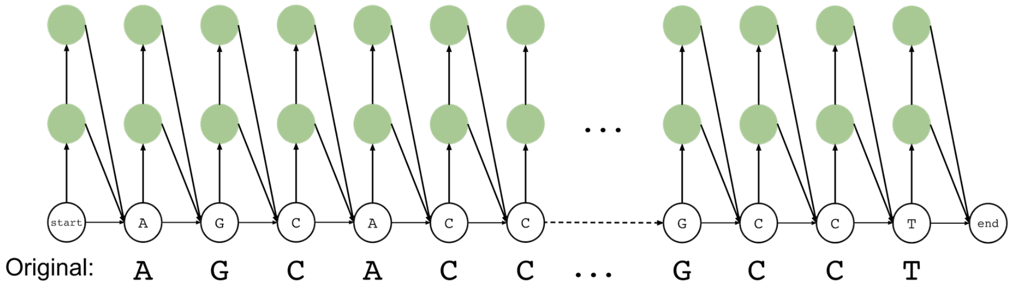
- Sequencing errors on reads may propagate to contigs
  - Leading to inaccurate analysis on the assembly we just generated
- Idea: Align reads back to contigs again to generate a more accurate consensus with Apollo



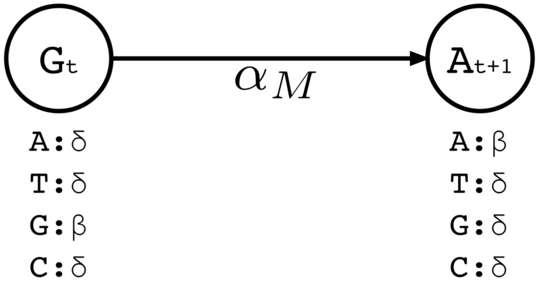
Profile Hidden Markov Models (pHMMs)

A profile hidden Markov model of the assembly:

AGCACC…GCCT



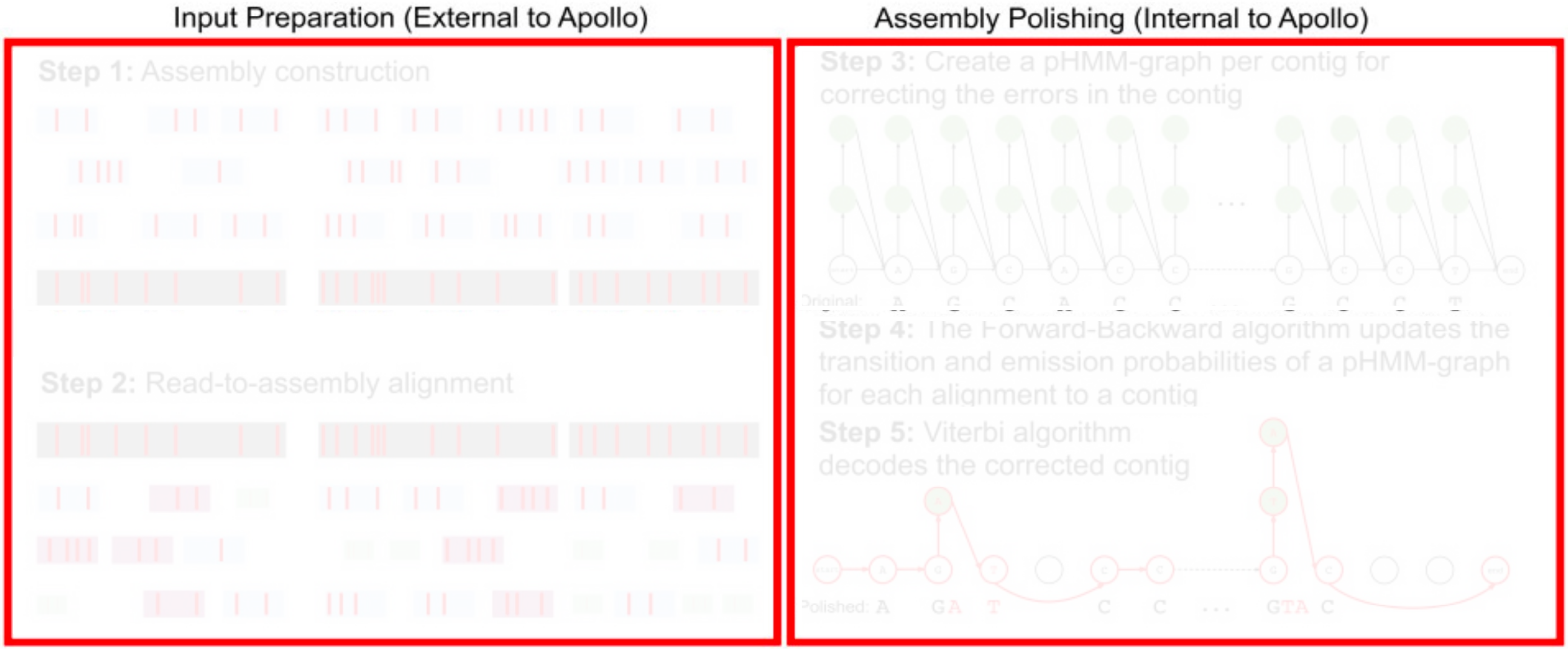
- Three components:
  - States
  - Emissions
  - Transitions (directed edges)



- Modification roles and probabilities are assigned to states, transitions, and emissions
  - Substitution, insertion, deletion, or match (no modification)

Apollo Workflow

- Steps external to Apollo
  - Step 1: Generate de novo assembly using any assembler
  - Step 2: Align reads to the de novo assembly of a genome
- Steps internal to Apollo
  - Step 3: Generate a pHMM graph for a genome assembly
  - Step 4: Update the probabilities of the pHMM graph using the alignment information from step 3
  - Step 5: Use the updated probabilities in pHMM to decode the corrected version of the genome assembly


Input Preparation (External to Apollo)
Assembly Polishing (Internal to Apollo)
Step 1: Assembly construction
Step 2: Read-to-assembly alignment
Step 3: Create a pHMM-graph per contig for correcting the errors in the contig
Original: A G C A C C ... G C C T
Step 4: The Forward-Backward algorithm updates the transition and emission probabilities of a pHMM-graph for each alignment to a contig
Step 5: Viterbi algorithm decodes the corrected contig
Polished: A GA T C C ... GTA C

Key Results

- State-of-the-art polishing tools: Racon, Pilon, Quiver, Nanopolish

- Apollo is the only tool that can scale to polish large genomes
  - Read set: PacBio (35x and 8.9x) and Illumina (22x)
  - Racon, Pilon and Quiver exceeds memory requirements (192GB) when using high/medium coverage PacBio/Illumina reads
  - *Apollo is the only algorithm* that is scalable to polish large contigs given the memory constraints

- Apollo generates the most reliable assemblies
  - Canu assembler rather than Miniasm
  - Polish using both long and Illumina reads (i.e., hybrid reads)
  - Apollo to use hybrid reads
    - It can use multiple read sets in a single run

- Apollo performs better than Nanopolish (~2-5x) but worse than Racon, Pilon, and Quiver (up to 600x, on average ~20-25x)

Future Work

- Apollo performs worse due to its computationally expensive parameter update (training) and decoding (inference) steps
  - Both training and inference steps are based on **embarrassingly parallel algorithms**
  - CPU *cannot* utilize all available parallelism
  - We implemented the training step in **GPUs** and observe that we can achieve around 45x performance improvement compared to the CPU
    - Can we do better? **Hardware acceleration** for training?
  - **Combining training and inference steps** in an accelerator would potentially provide even better performance improvements

- **Parameter optimizations** for different sequencing technologies to improve sensitivity

Executive Summary

- **Problem:**
  - Long read de-novo assembly is inherently erroneous
  - Existing assembly polishing techniques cannot adapt to varying sequencing technologies and do not scale well for large genomes

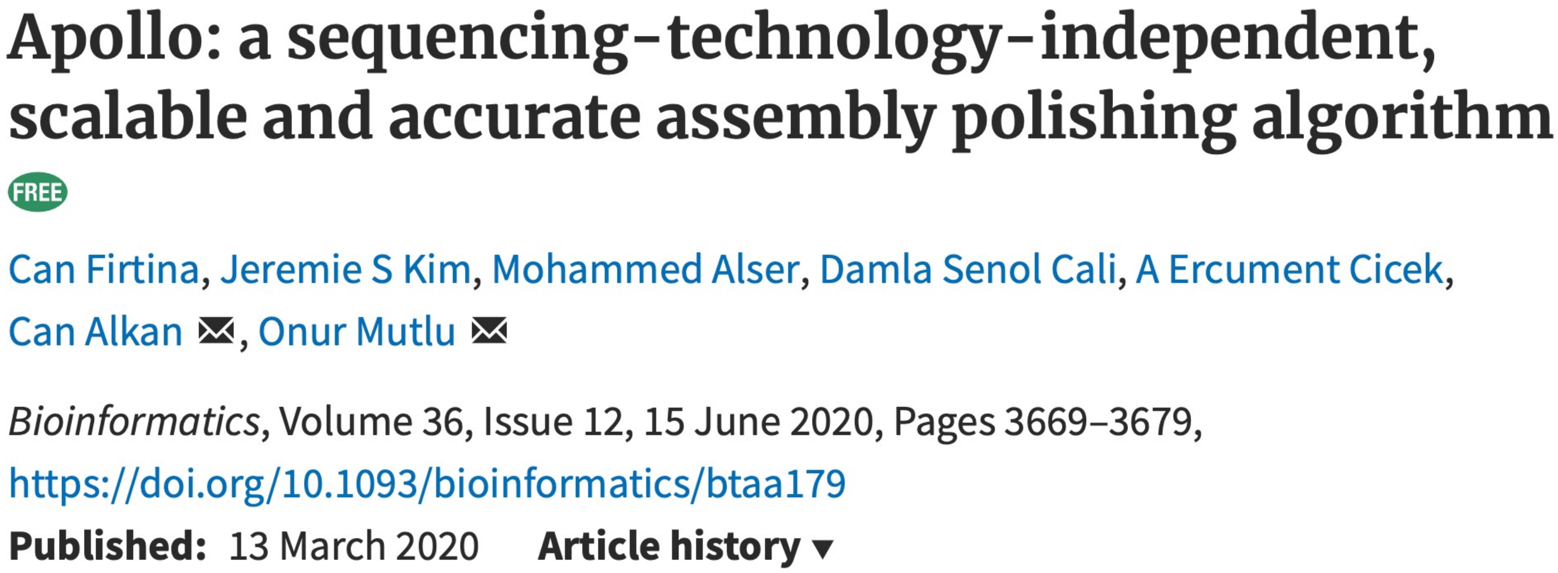
- **Goal:** Propose a technology-independent and scalable assembly polishing algorithm – Apollo

- **Key Ideas:**
  - Use a probabilistic graph structure to identify the errors in assemblies accurately
    - Profile hidden Markov models (pHMMs)
  - Realign erroneous reads to the pHMM graph of an assembly to correct the errors by updating the probabilities based on a machine learning technique

- **Results**
  - Apollo is the only assembly polishing that is scalable to polish large genomes given the limited memory constraints (e.g., 192GB)
  - Apollo constructs the most reliable assemblies when hybrid set of reads (e.g., Illumina and PacBio) are used in a single run compared to other polishing tools
  - Apollo is ~25x slower on average (up to ~600x) than other polishers

Assembly Polishing using ML

- Can Firtina, Jeremie S. Kim, Mohammed Alser, Damla Senol Cali, A. Ercument Cicek, Can Alkan, and Onur Mutlu,
  "Apollo: A Sequencing-Technology-Independent, Scalable, and Accurate Assembly Polishing Algorithm"
  Bioinformatics, June 2020.
  [Source Code]
  [Online link at Bioinformatics Journal]



Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm

Can Firtina, Jeremie S Kim, Mohammed Alser, Damla Senol Cali, A Ercument Cicek, Can Alkan ✉, Onur Mutlu ✉

Bioinformatics, Volume 36, Issue 12, 15 June 2020, Pages 3669–3679, https://doi.org/10.1093/bioinformatics/btaa179

Published: 13 March 2020 **Article history ▾**

Agenda for Today - Intelligent Algorithms

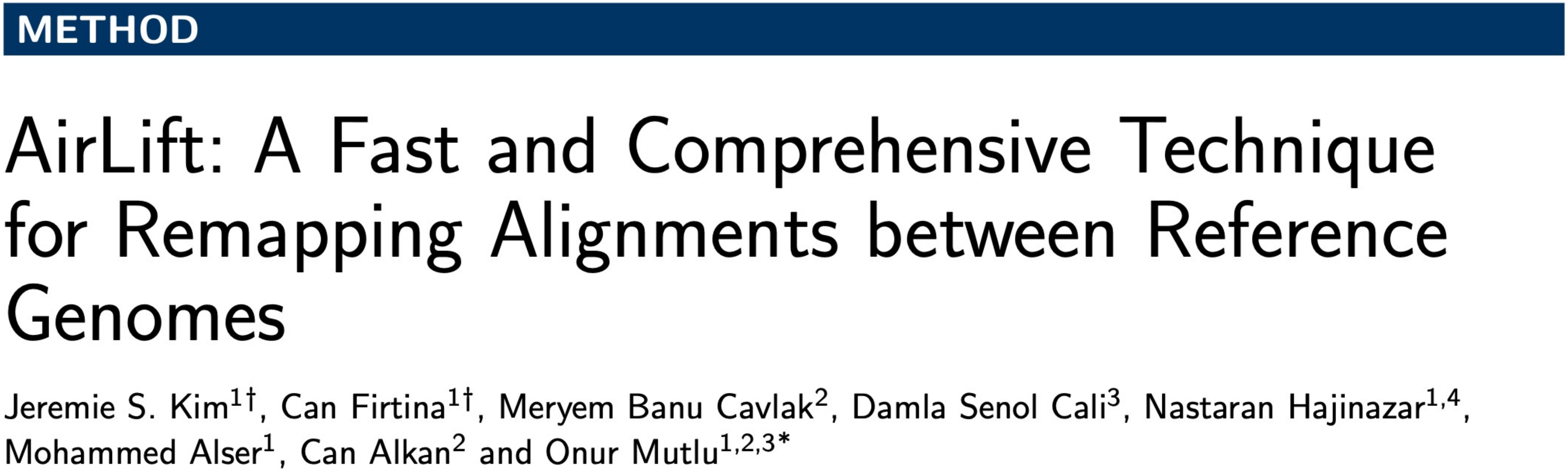
- Apollo
  - Correcting errors in genome assembly for more accurate genome analysis

- AirLift
  - Avoiding redundant computations when moving from one reference genome to another

- BLEND
  - Using fewer and high-quality seeds for faster, more memory-efficient, and more accurate sequence similarity identification

Remapping Alignments Between References

- Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Nastaran Hajinazar, Mohammed Alser, Can Alkan, and Onur Mutlu,
  "AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes"
  Preprint in ***arXiv*** *and* ***bioRxiv***, 17 February 2021.
  [bioRxiv preprint]
  [arXiv preprint]
  [AirLift Source Code and Data]



METHOD

AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes

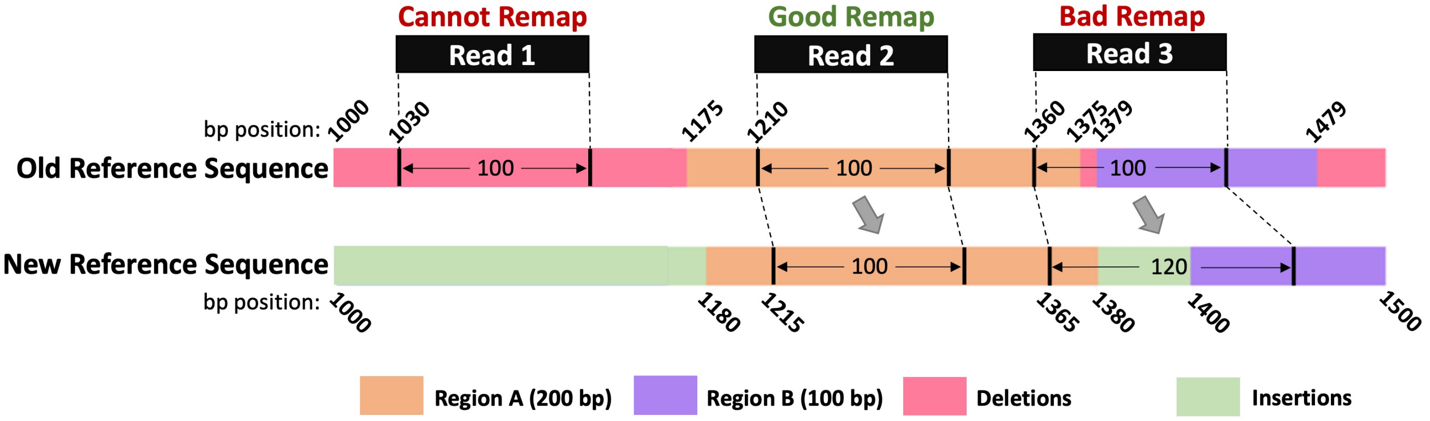
Jeremie S. Kim[1†], Can Firtina[1†], Meryem Banu Cavlak[2], Damla Senol Cali[3], Nastaran Hajinazar[1,4], Mohammed Alser[1], Can Alkan[2] and Onur Mutlu[1,2,3*]

Background

- Reference genomes are updated frequently (~quarterly)

- For accurate annotations for an updated genome, all reads of a given sample are completely remapped to the new, updated reference genome
  - This becomes unreasonable with the number of sequenced genomes available now

- Currently, existing works speed up this process by only moving annotations (e.g., read mapping positions) given known matching regions



Background (cont'd)

- A number of tools exist for remapping annotations
  - To remap across samples of the same species or across very similar reference genomes

- Tools:
  1. UCSC Liftover
  2. CrossMap
  3. RECOT
  4. Segment_liftover
  5. NCBI Genome Remapping Service
  6. Assembly Converter (Ensembl)
  7. Galaxy (based on UCSC liftover)
  8. Pyliftover

Generate *Chain* files, which indicates similar blocks of regions across two genomes (via global alignment) and move annotations using the chain file

Executive Summary

- **Problem:**
  - Remapping full set of reads to newer reference genome is accurate, but very computationally costly
  - Existing tools that move the annotations from one reference to another miss moving large portion of annotations to a new reference genome

- **Goal:** Maintain baseline accuracy of remapping full set of reads, while significantly accelerating the process of moving from one reference to another

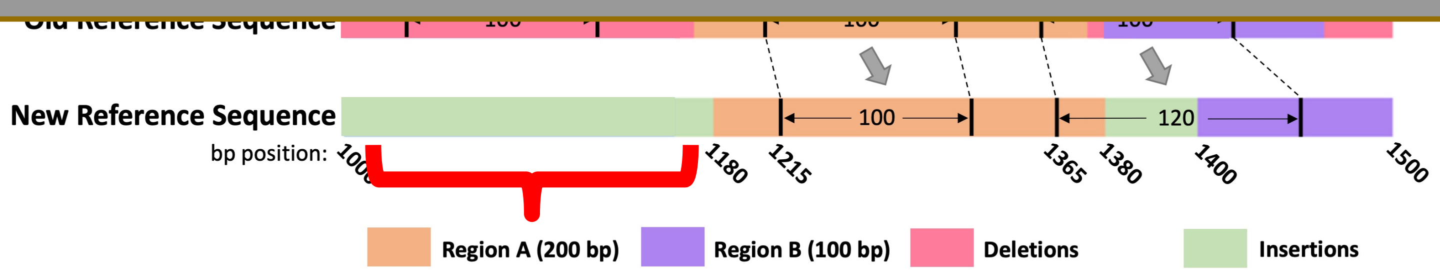
- **Key Ideas:**
  - Categorize and label each region in the old reference genome compared to the new reference genome
  - Either 1) Remap a read to a new reference genome or 2) simply move its annotation based on the label of the region that the read falls into in the old reference genome

- **Results**
  - 2.6x – 6.7x speedup compared to fully mapping reads, while identifying SNPs and Indels with precision and recall similar to full mapping

Problem

- Existing tools move annotations (e.g., read mapping positions) only if the same region appears in the new reference genome
  - Same regions between two reference genomes are identified using **global alignment – chain files** contain the aligning blocks

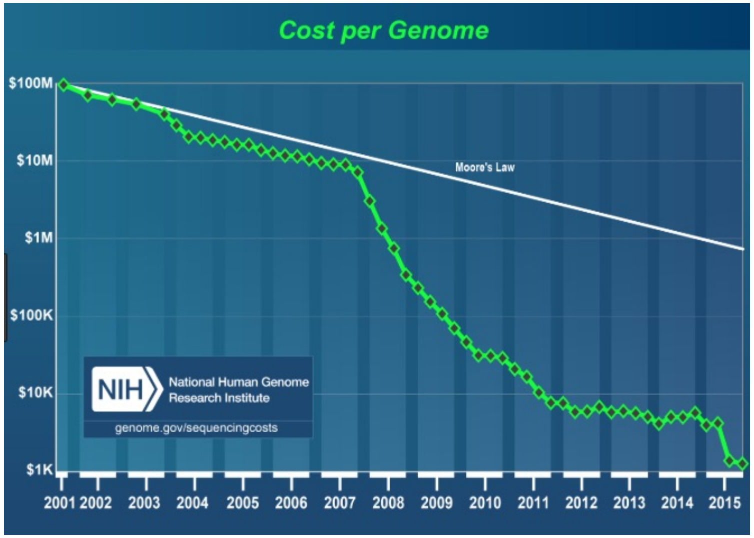
Simply moving annotations is not accurate and it loses a **significant amount** of information



Potentially important region where annotations are lost

Problem (cont'd)

- To reach baseline accuracy of completely read mapping reads to the new reference genome:
  - Could re-run read mapping to the entire new reference genome
    - This is **expensive** especially if we want to map every existing sequenced genome (**>500,000** as of 2017) to every reference genome update (quarterly)



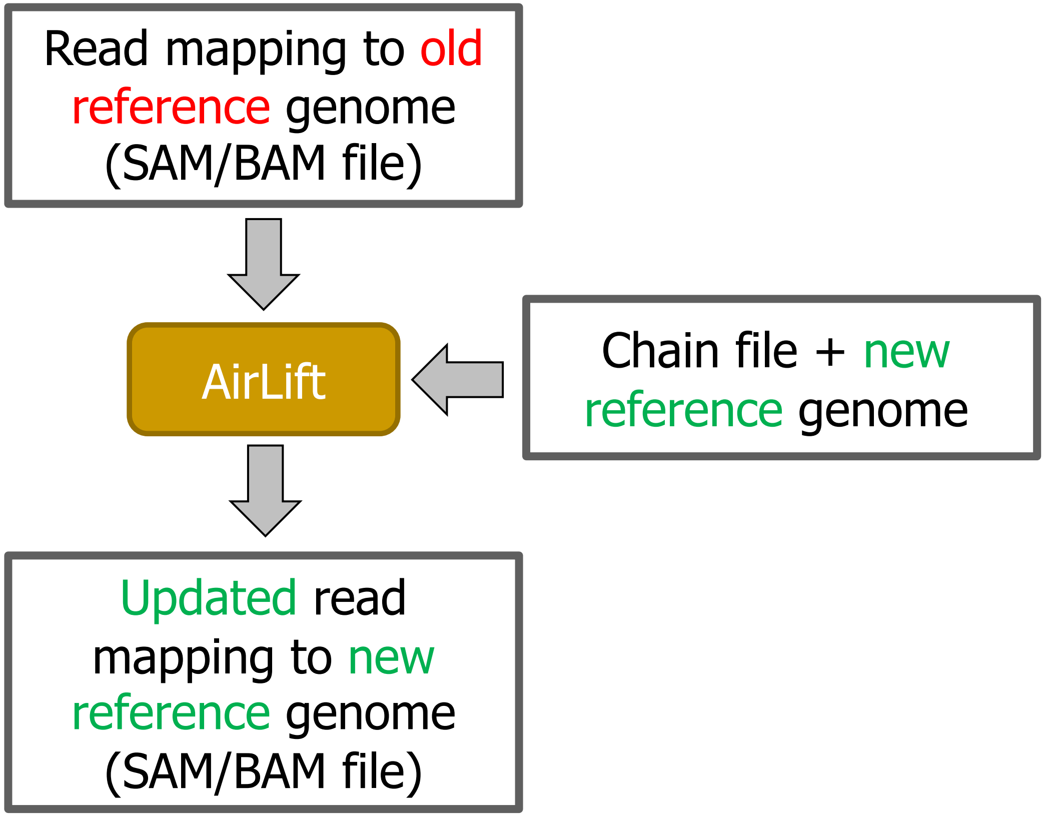
Goal

- Maintain baseline accuracy of mapping reads to an updated reference genome, while significantly accelerating the process
  - Baseline is using a full mapper

- Enable quick and accurate mapping of every sample to every updated reference genome no matter how small the changes
  - This will quickly give us the most up to date information about a sample's genome

Mechanism

- We provide a method for quickly remapping **all the reads** of a sample from one reference genome to an updated version of the reference genome



Read mapping to old reference genome (SAM/BAM file)

AirLift

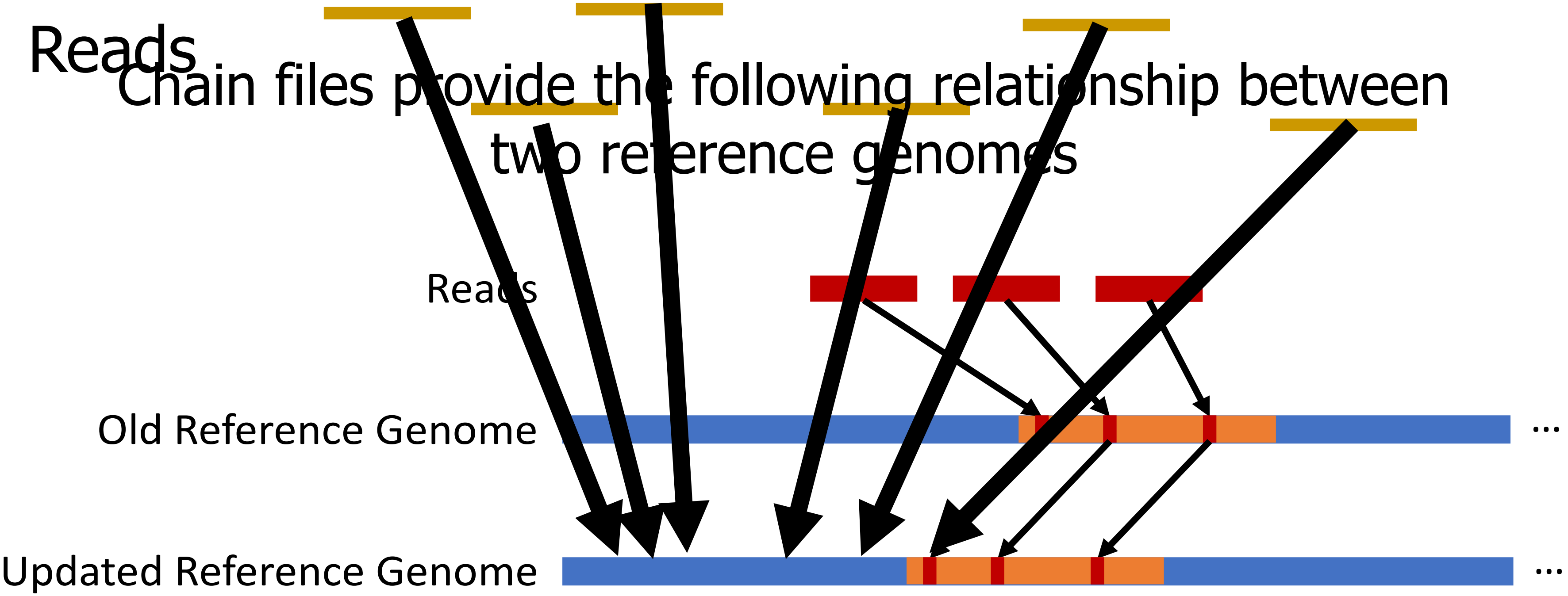
Chain file + new reference genome

Updated read mapping to new reference genome (SAM/BAM file)

Mechanism (cont'd)

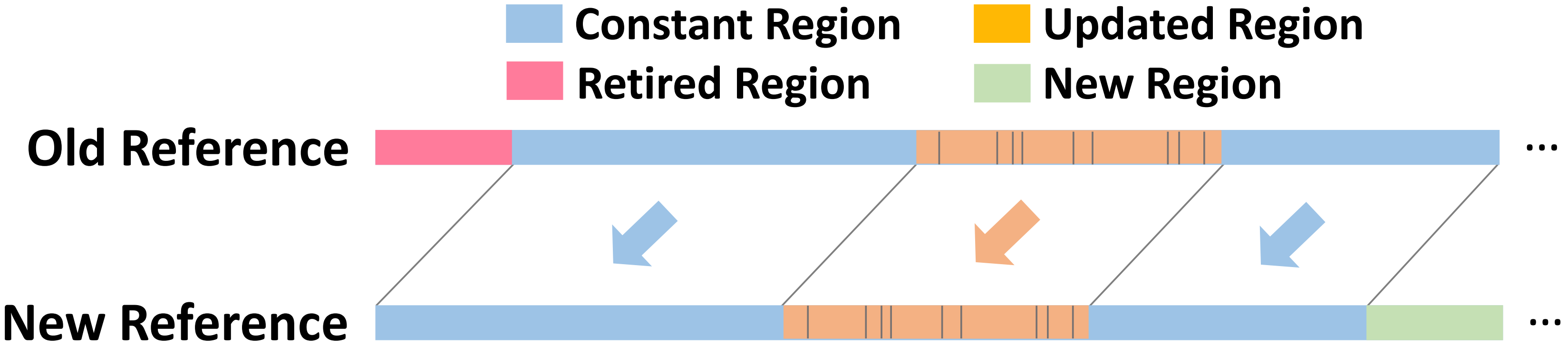
- We provide a method for quickly remapping **all the reads** of a sample from one reference genome to an updated version of the reference genome

Chain files provide the following relationship between two reference genomes

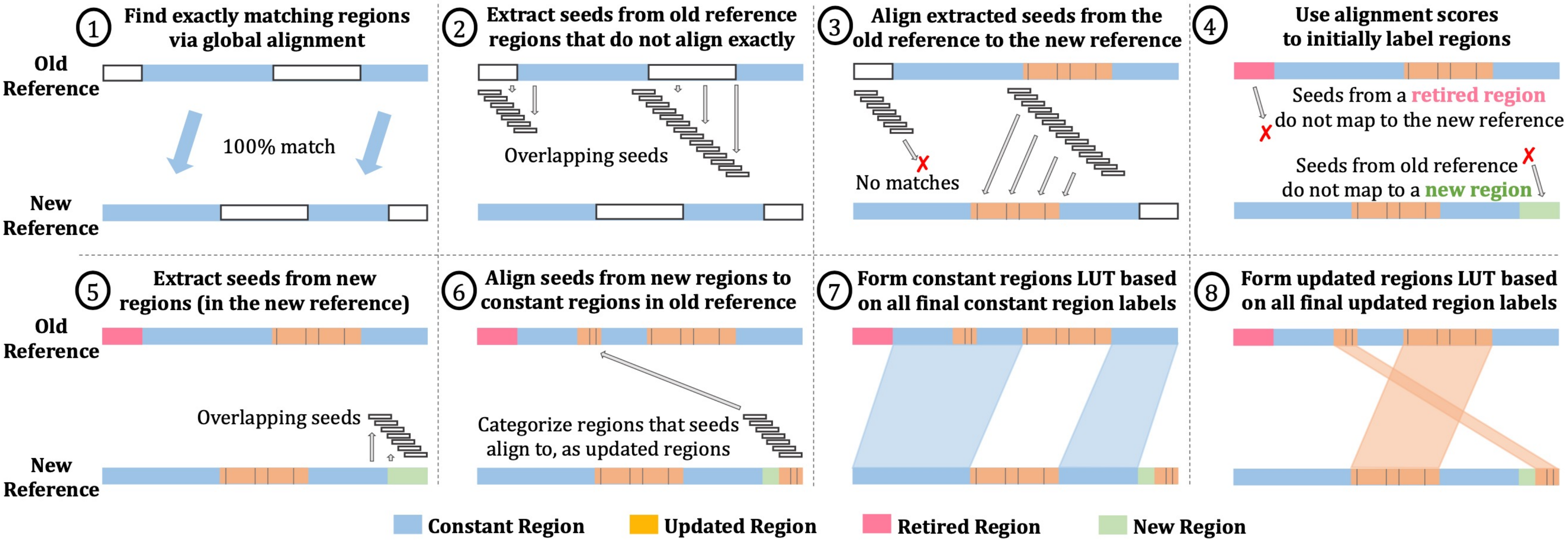


Labeling Regions of Reference Genomes

- We quickly label the regions between two reference genomes to decide if we should
  - Move the read mapping position without performing costly alignment or
  - Remap reads that maps in the regions that are missing in the new reference genome
- Finding labels between a pair of reference genomes is a one-time task: **AirLift Index**
- Labels: Constant, updated, retired, and new regions



AirLift Index: Labeling Regions in 8 steps



1 Find exactly matching regions via global alignment

Old Reference

100% match

New Reference

2 Extract seeds from old reference regions that do not align exactly

Overlapping seeds

3 Align extracted seeds from the old reference to the new reference

No matches

4 Use alignment scores to initially label regions

Seeds from a retired region do not map to the new reference

Seeds from old reference do not map to a new region

5 Extract seeds from new regions (in the new reference)

Old Reference

Overlapping seeds

New Reference

6 Align seeds from new regions to constant regions in old reference

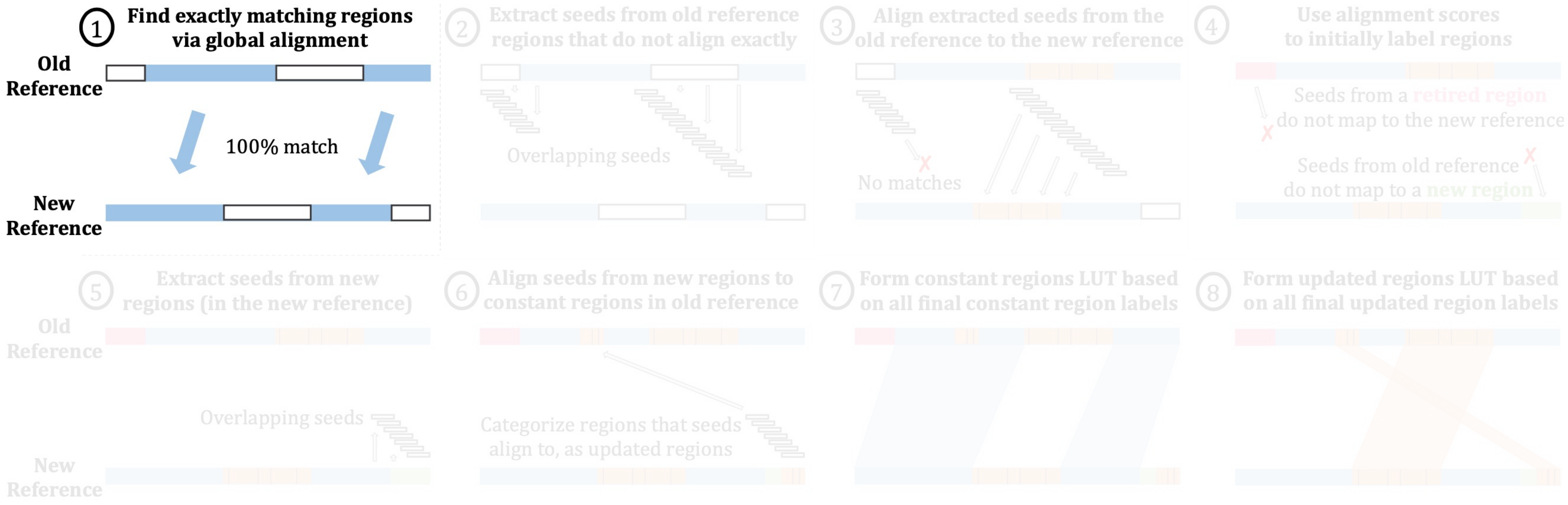
Categorize regions that seeds align to, as updated regions

7 Form constant regions LUT based on all final constant region labels

8 Form updated regions LUT based on all final updated region labels

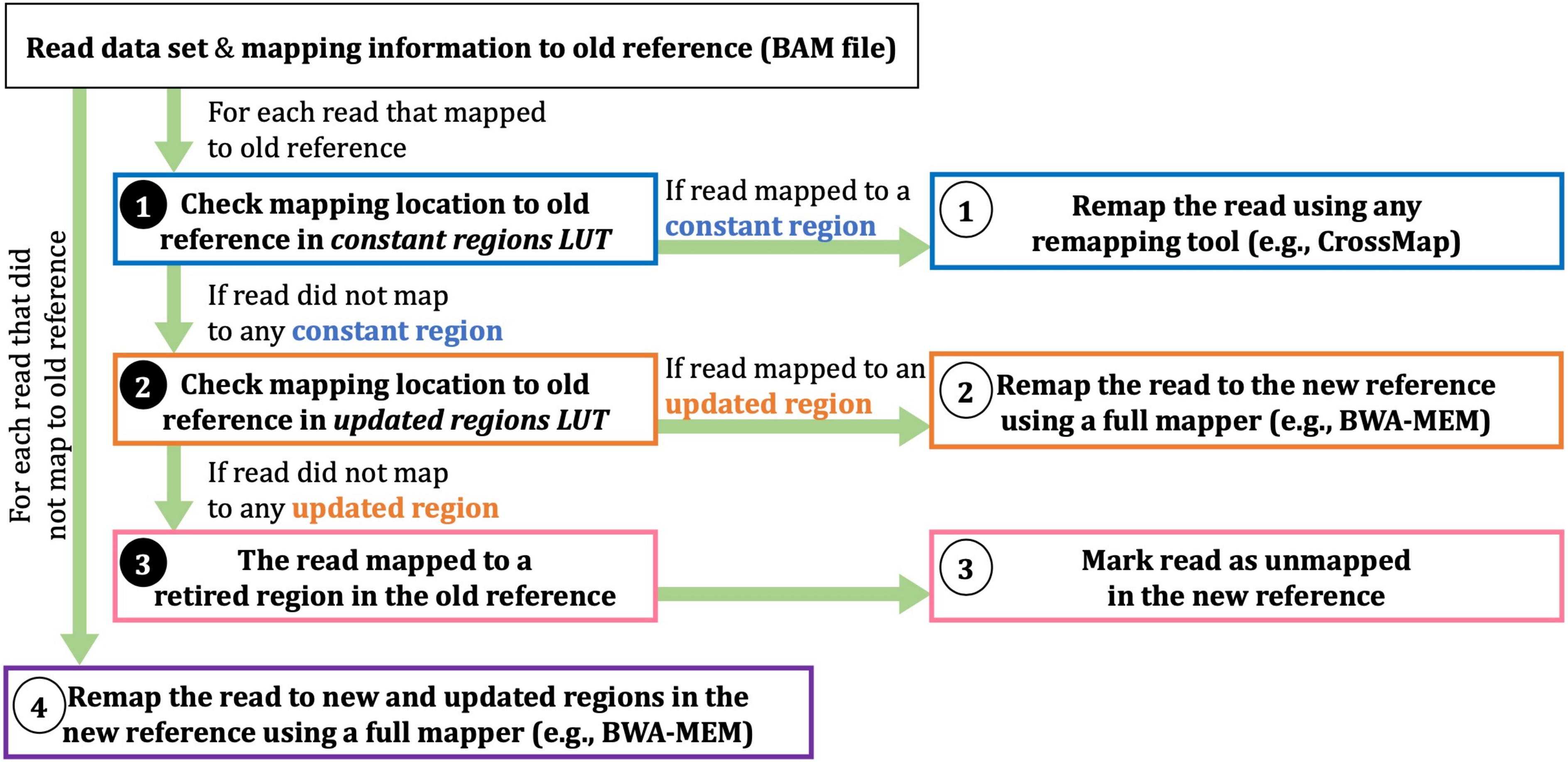
🟦 **Constant Region** 🟧 **Updated Region** 🟥 **Retired Region** 🟩 **New Region**

AirLift Index: Labeling Regions in 8 steps


1 Find exactly matching regions via global alignment
Old Reference
100% match
New Reference
2 Extract seeds from old reference regions that do not align exactly
Overlapping seeds
3 Align extracted seeds from the old reference to the new reference
No matches
4 Use alignment scores to initially label regions
Seeds from a retired region do not map to the new reference
Seeds from old reference do not map to a new region
5 Extract seeds from new regions (in the new reference)
Old Reference
Overlapping seeds
New Reference
6 Align seeds from new regions to constant regions in old reference
Categorize regions that seeds align to, as updated regions
7 Form constant regions LUT based on all final constant region labels
8 Form updated regions LUT based on all final updated region labels

🟦 **Constant Region** 🟨 **Updated Region** 🟥 **Retired Region** 🟩 **New Region**

Updating Read Mapping using AirLift Index



Read data set & mapping information to old reference (BAM file)

For each read that mapped to old reference

1 Check mapping location to old reference in *constant regions LUT*

If read mapped to a **constant region**

1 Remap the read using any remapping tool (e.g., CrossMap)

If read did not map to any **constant region**

2 Check mapping location to old reference in *updated regions LUT*

If read mapped to an **updated region**

2 Remap the read to the new reference using a full mapper (e.g., BWA-MEM)

If read did not map to any **updated region**

3 The read mapped to a retired region in the old reference

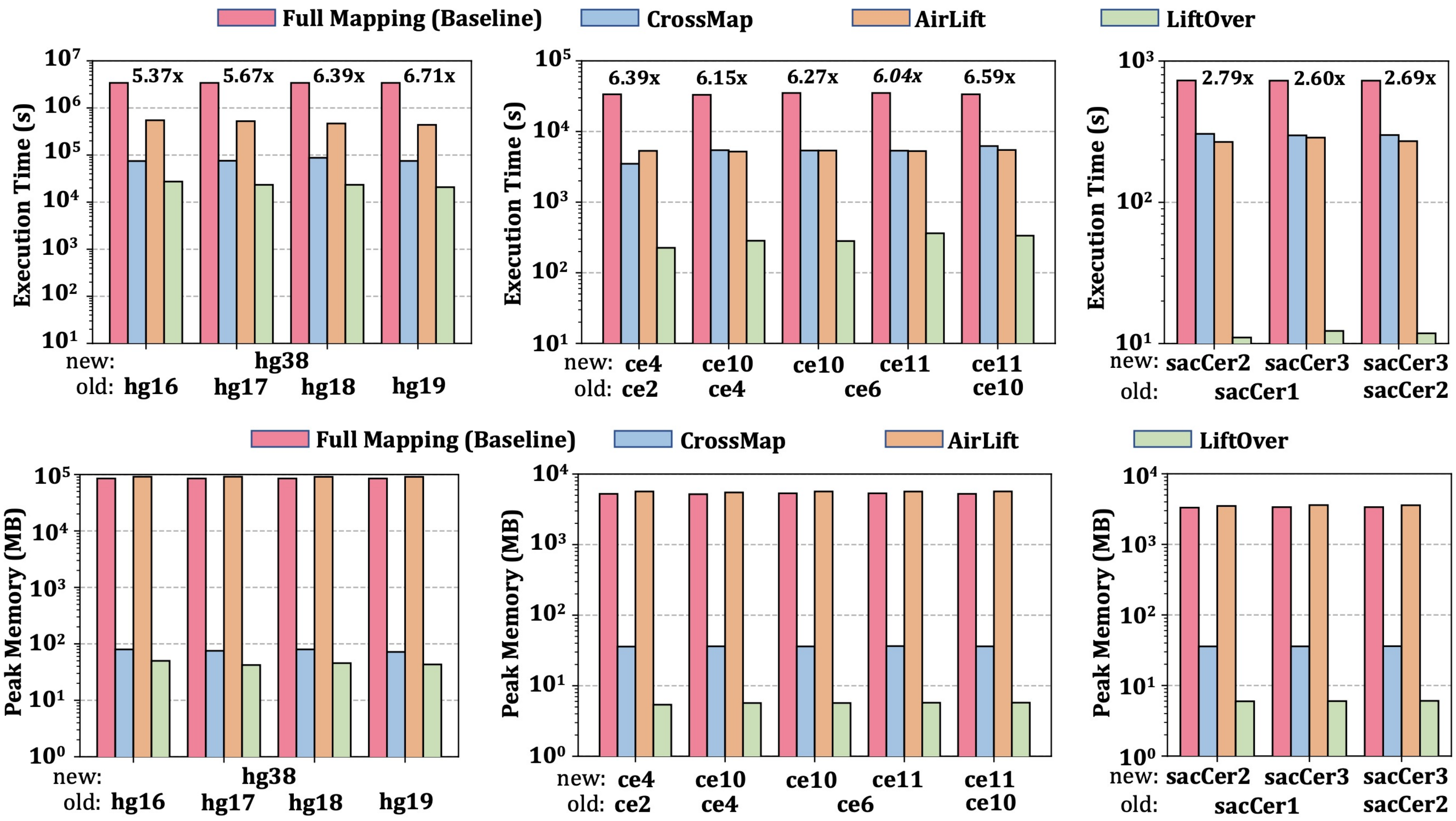
3 Mark read as unmapped in the new reference

For each read that did not map to old reference

4 Remap the read to new and updated regions in the new reference using a full mapper (e.g., BWA-MEM)

Key Results – Performance and Memory

- AirLift is 2.6x-6.71x faster than full mapping while requiring similar peak memory usage



Key Results – Variant Calling

- We use GATK to call the variants in read mappings that full mapping (to new reference genome) and AirLift generate

- Variant calling ground truth from Platinum Genomes and GIAB for the human NA12878 sample

- Variant calling accuracy is comparable to full mapping

Remap	Read Sets		vs. Full Mapping		vs. Ground Truth	
Technique	from	to	SNP (%)	Indel (%)	SNP (%)	Indel (%)
Full Mapping	-	hg38	-	-	99.54/88.00	81.31/92.38
AirLift	hg16	hg38	96.81/97.01	85.03/87.30	98.77/87.32	75.76/90.36
	hg17		96.69/97.34	84.65/88.59	98.78/87.68	75.51/90.97
	hg18		96.66/97.81	84.94/88.68	98.80/88.16	75.73/91.03
	hg19		97.03/97.69	85.41/88.98	98.88/87.84	76.01/91.23
AirLift	ce2	ce4	90.82/97.29	96.97/97.66	-	-
	ce4	ce10	91.06/96.96	96.81/97.30	-	-
	ce6		91.11/97.00	96.81/97.33	-	-
	ce6	ce11	90.01/96.12	95.86/96.18	-	-
	ce10		90.03/96.48	95.90/96.44	-	-
AirLift	sacCer1	sacCer2	95.30/98.82	95.83/94.74	-	-
	sacCer1	sacCer3	86.35/94.27	90.38/88.65	-	-
	sacCer2		87.03/91.19	91.14/88.65	-	-

Executive Summary

- **Problem:**
  - Remapping full set of reads to newer reference genome is accurate, but very computationally costly
  - Existing tools that move the annotations from one reference to another miss moving large portion of annotations to a new reference genome

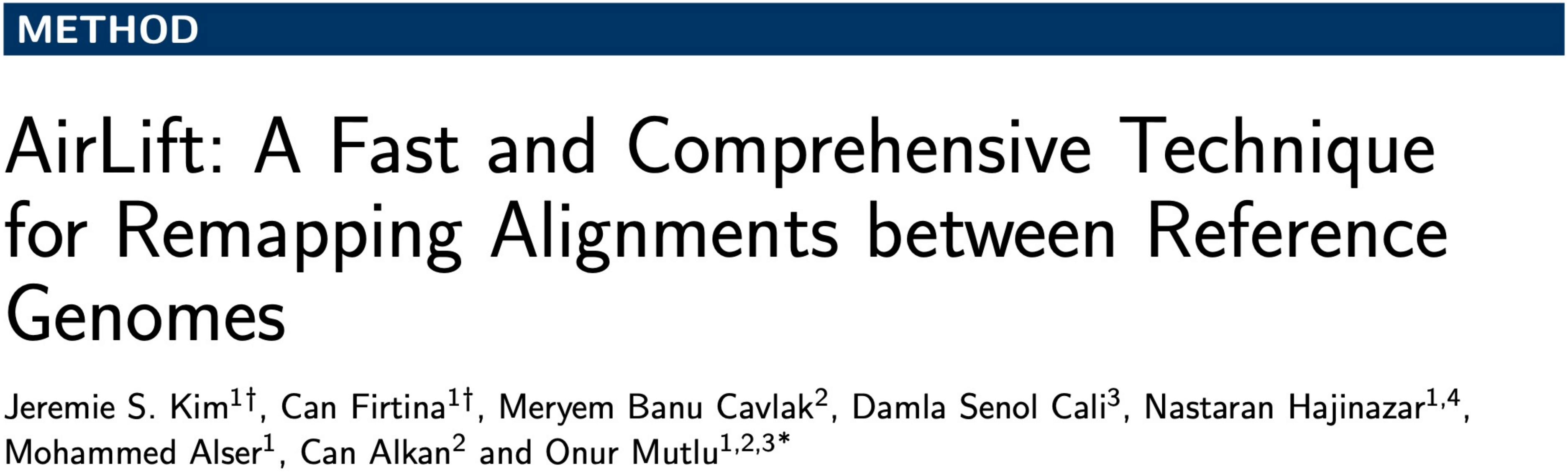
- **Goal:** Maintain baseline accuracy of remapping full set of reads, while significantly accelerating the process of moving from one reference to another

- **Key Ideas:**
  - Categorize and label each region in the old reference genome compared to the new reference genome
  - Either 1) Remap a read to a new reference genome or 2) simply move its annotation based on the label of the region that the read falls into in the old reference genome

- **Results**
  - 2.6x – 6.7x speedup compared to fully mapping reads, while identifying SNPs and Indels with precision and recall similar to full mapping

Remapping Alignments Between References

- Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Nastaran Hajinazar, Mohammed Alser, Can Alkan, and Onur Mutlu,
  "AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes"
  Preprint in ***arXiv*** *and* ***bioRxiv***, 17 February 2021.
  [bioRxiv preprint]
  [arXiv preprint]
  [AirLift Source Code and Data]



METHOD

AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes

Jeremie S. Kim[1†], Can Firtina[1†], Meryem Banu Cavlak[2], Damla Senol Cali[3], Nastaran Hajinazar[1,4], Mohammed Alser[1], Can Alkan[2] and Onur Mutlu[1,2,3*]

Agenda for Today - Intelligent Algorithms

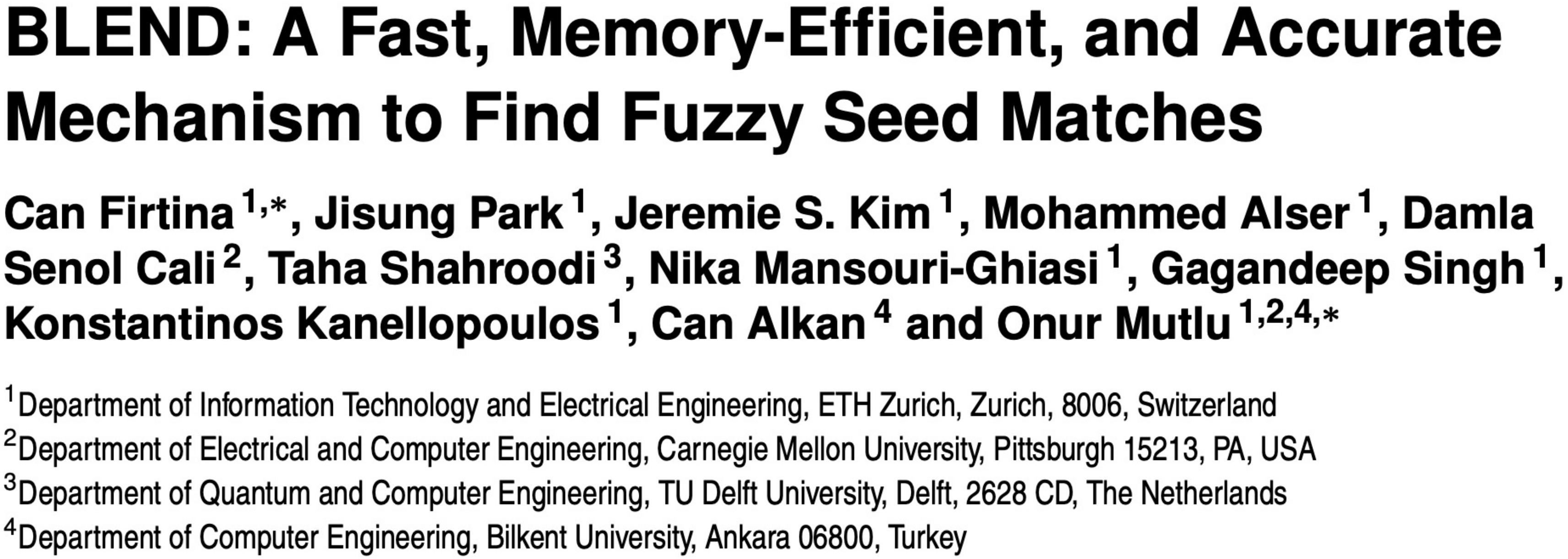
- Apollo
  - Correcting errors in genome assembly for more accurate genome analysis

- AirLift
  - Avoiding redundant computations when moving from one reference genome to another

- BLEND
  - Using fewer and high-quality seeds for faster, more memory-efficient, and more accurate sequence similarity identification

Finding Efficient and Accurate Seed Matches

- Can Firtina, Jisung Park, Jeremie S. Kim, Mohammed Alser, Damla Senol Cali, Taha Shahroodi, Nika Mansouri-Ghiasi, Gagandeep Singh, Konstantinos Kanellopoulos, Can Alkan, and Onur Mutlu,
  "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches"
  Preprint in ***arXiv***, 16 December 2021.
  [arXiv preprint]
  [BLEND Source Code and Data]



BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches

Can Firtina [1,*], Jisung Park [1], Jeremie S. Kim [1], Mohammed Alser [1], Damla Senol Cali [2], Taha Shahroodi [3], Nika Mansouri-Ghiasi [1], Gagandeep Singh [1], Konstantinos Kanellopoulos [1], Can Alkan [4] and Onur Mutlu [1,2,4,*]

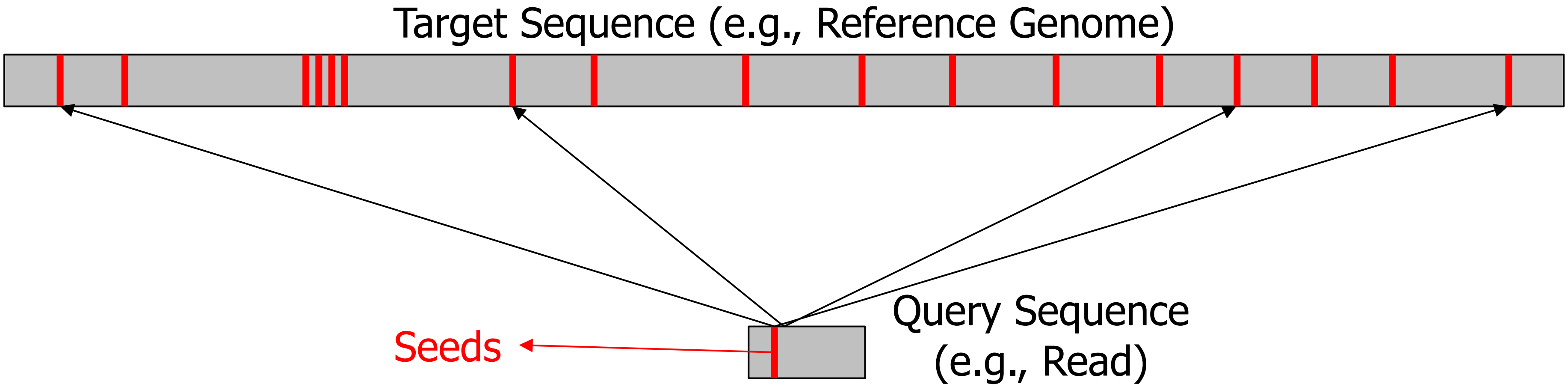
[1] Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, 8006, Switzerland
[2] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA
[3] Department of Quantum and Computer Engineering, TU Delft University, Delft, 2628 CD, The Netherlands
[4] Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey

Background – Seeds

- **Seeds:** Short subsequences used for matching between sequences (e.g., read and a reference genome)

- Seeds enable filtering out large portion of sequence pairs that are unlikely to be similar
  - Distance calculation (e.g., approximate string matching) is costly

- **Heuristic:** Calculate the distance between a pair of sequences only if they share some amount of seeds



Background – Matching Seeds

- Several ways for finding seed matches:
  - Building up an index: FM-index, Hash tables…
  - Calculating similarities between seeds using their k-mer content

- Types of seed matches:
  - Exact-matching seeds:
    - Faster to find – Relatively low computational cost
    - Usually very short – many matches
  - Approximate (fuzzy) seed matches:
    - More costly to find
    - Can be more selective in seed matches

Executive Summary

- **Problem:** Existing seed matching techniques suffer from performing either
  - Too many costly approximate string matching due to finding a large number of exact-matching seeds
  - Costly similarity calculations to find fewer fuzzy seed matches

- **Goal:** Quickly and efficiently find fuzzy seed matches between sequences to improve the performance, memory efficiency, and accuracy of the applications that use seeds

- **Key Ideas:**
  - Generate the same hash value for highly similar seeds using a hashing technique called SimHash
    - A single hash value for a seed to find fuzzy seed matches, unlike other works that use many hash values
    - The same hash value for highly similar seeds even though these seeds may mismatch at any arbitrary position, unlike spaced seeding
  - Build and use the hash table using these hash values that BLEND generates

- **Key Results**
  - For finding overlapping reads, on average, 8.6x speedup and 5.43x lower memory footprint compared to the state-of-the-art, Minimap2
  - We can construct more accurate and contiguous assemblies using the overlapping reads that BLEND finds compared to using the ones Minimap2 finds

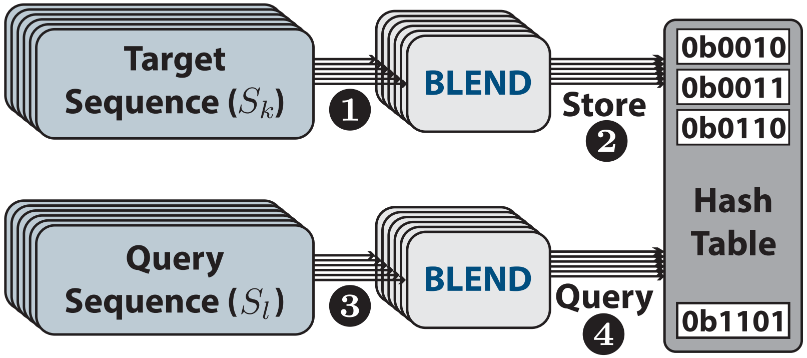
High-Level Overview of BLEND

- We find fuzzy seed matches between two sets of sequences
  - **Target** sequences (e.g., a reference genome)
  - **Query** sequences (e.g., read sequences)

- BLEND can generate the same **single** hash value for similar seeds

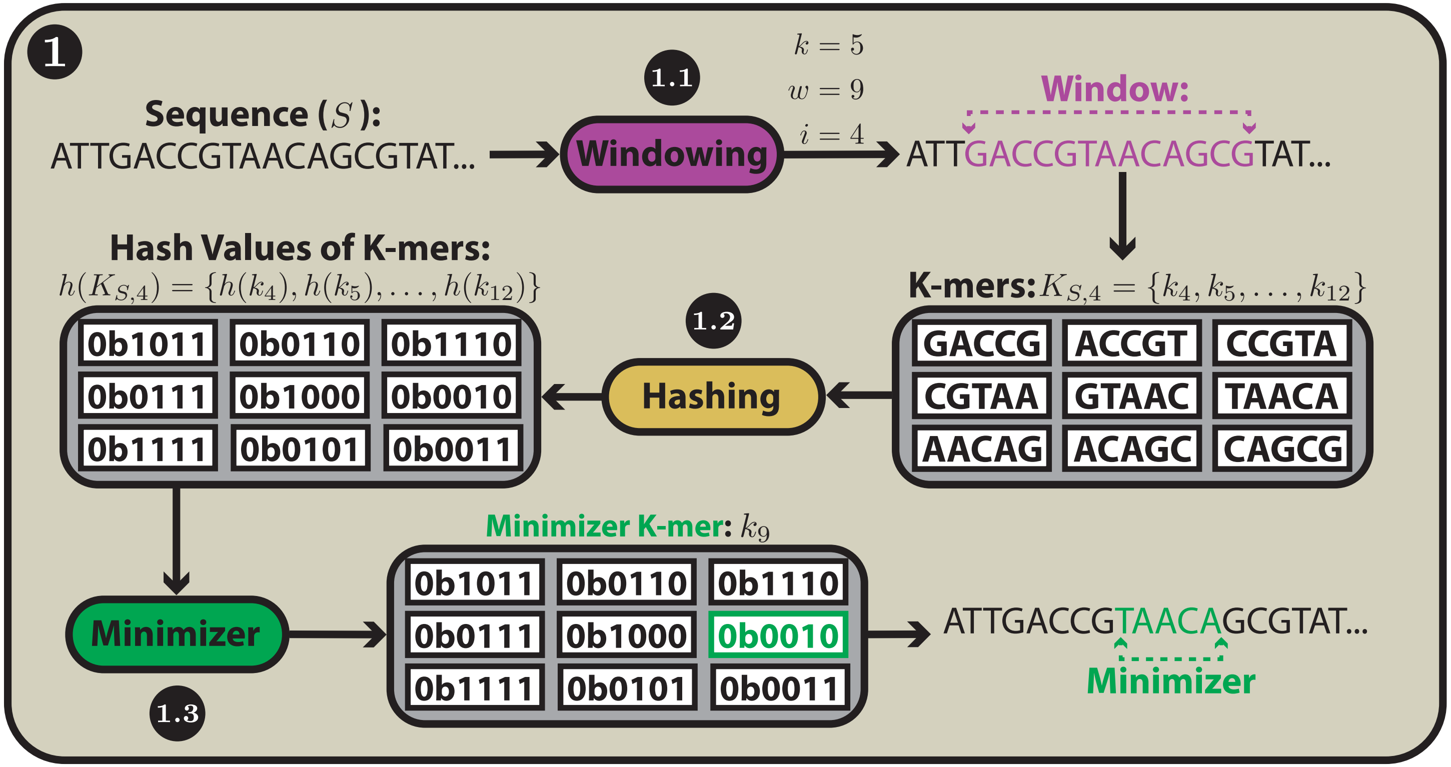
- Using BLEND, we generate the hash values of all the seeds of all **target sequences** and **store** them in a hash table
  - Enables efficient lookup of these hash values

- BLEND generates the hash value for each seed in **query sequences** and **queries** the hash table with these hash values
  - Enables finding fuzzy seed matches if the same hash value exist in the hash table



Building the Hash Table – Step 1

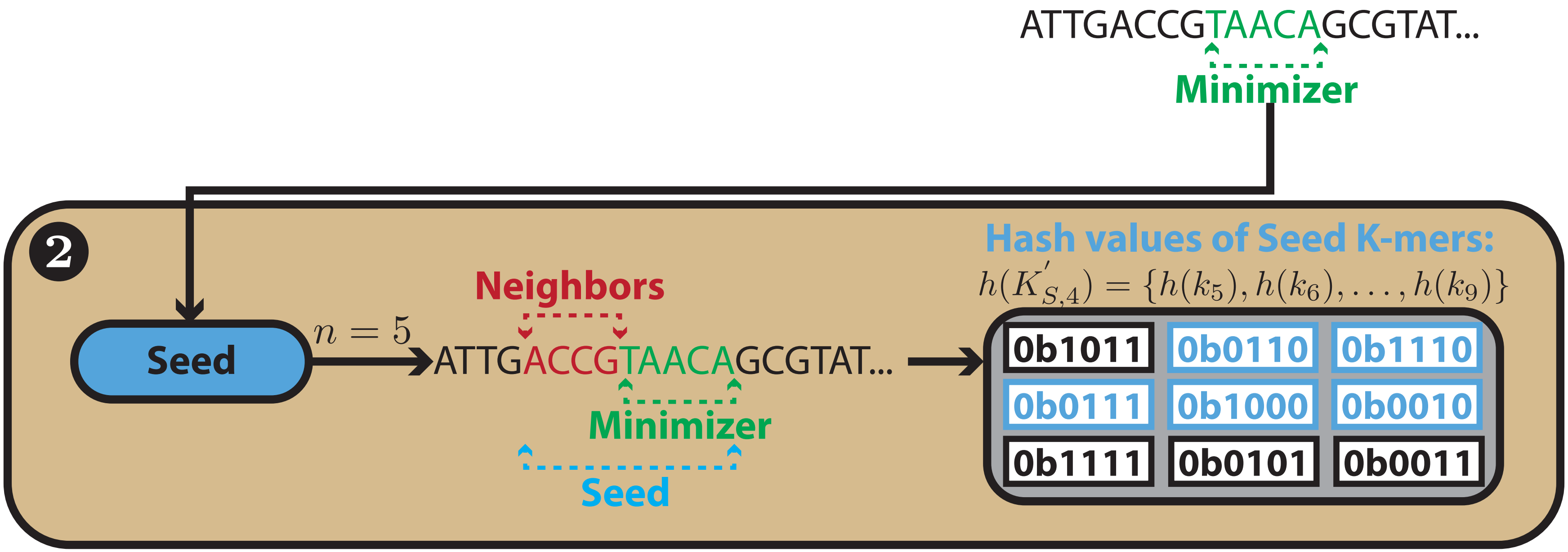
1. Select (some) k-mers of a target sequence
   - We can reduce the storage requirements per sequence by just storing the few k-mers, not all, with minimal information loss (i.e., minimizer k-mers)



Roberts, Michael +, "Reducing Storage Requirements for Biological Sequence Comparison," *Bioinformatics*, 2004.

Building the Hash Table – Step 2

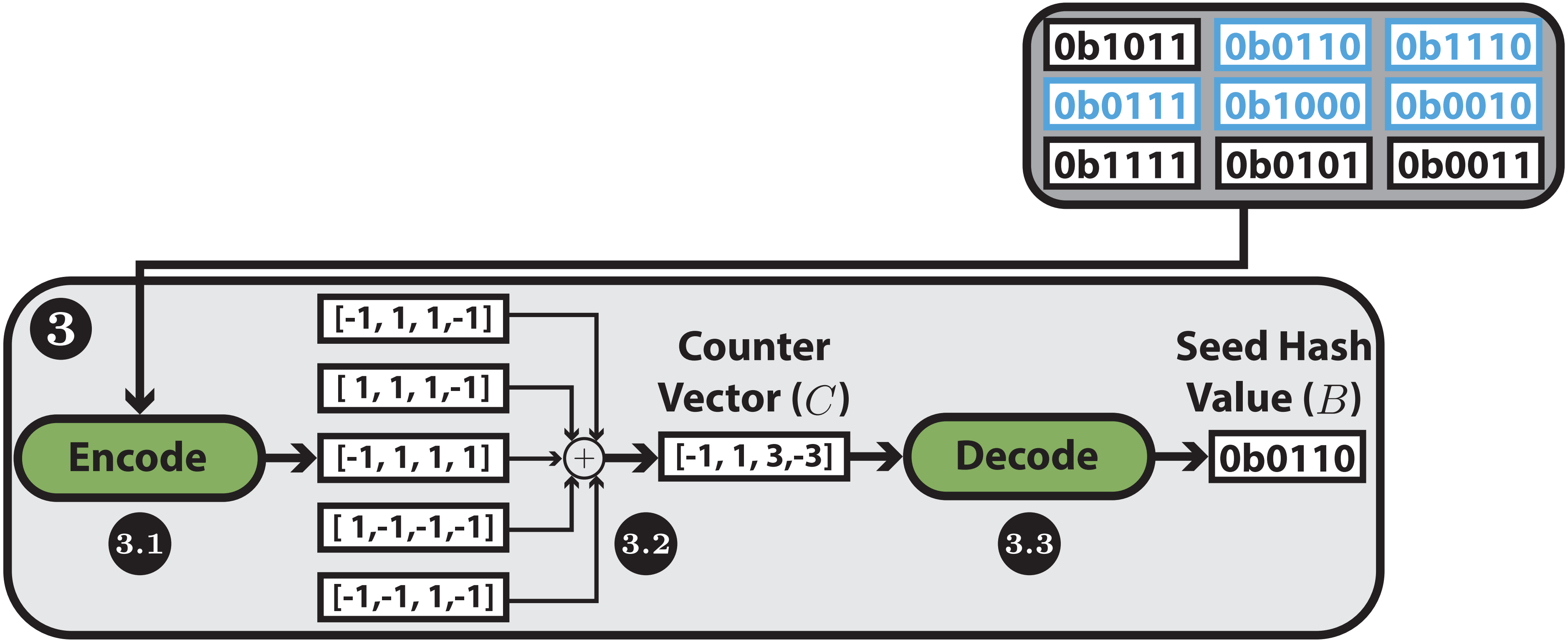
2. Generate seeds around each selected k-mer
   - ❑ We use seeds because we want to avoid searching the entire sequence
   - ❑ We identify the sites to use as seeds around the selected k-mers because we want to use as few seeds as possible for a sequence (low computational and space overhead)



SAFARI

Building the Hash Table – Step 3

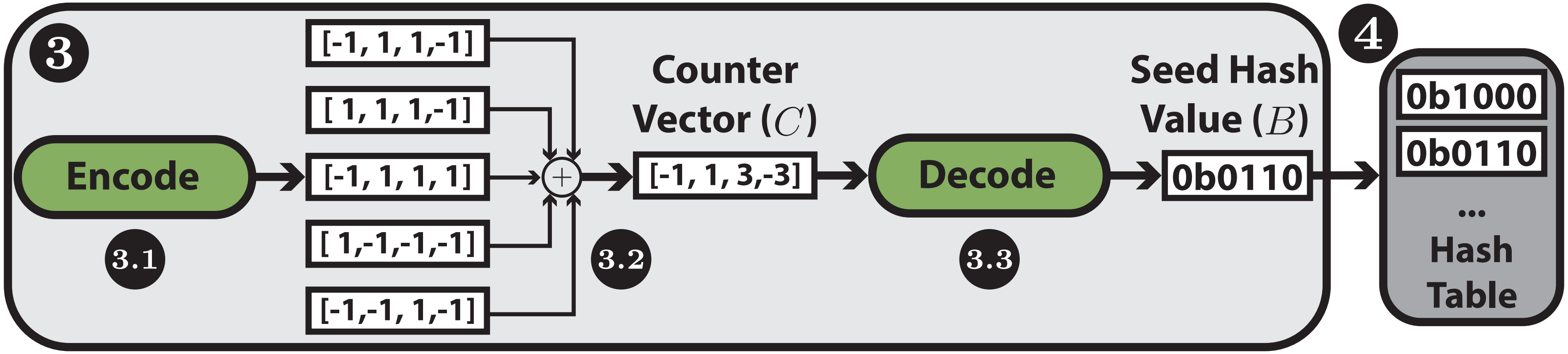
3. Generate the hash values of seeds using SimHash
   - ❑ To enable efficient comparisons of equivalence or high similarity between seeds



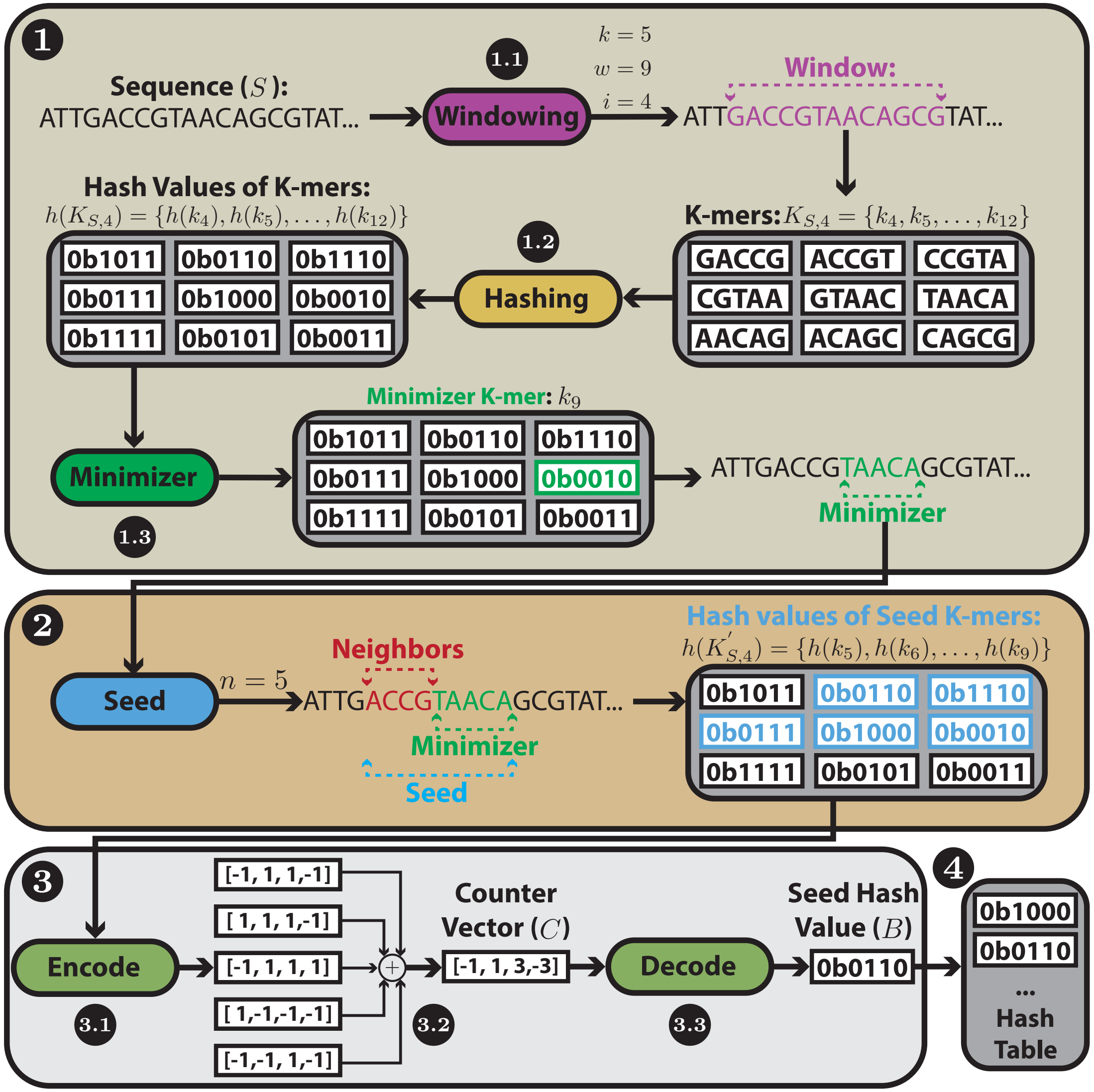
Charikar, Moses S. "Similarity Estimation Techniques from Rounding Algorithms." STOC, 2002. New York, NY, USA

Building the Hash Table – Step 4

1. Use the hash table to store
   - Hash values as keys
   - List of seeds that have the same hash value in the target sequence



Overview of all the Steps



1

Sequence (S):
ATTGACCGTAACAGCGTAT...

1.1

Windowing

$k=5$
$w=9$
$i=4$

Window:
ATTGACCGTAACAGCGTAT...

K-mers: $K_{S,4} = \{k_4, k_5, \ldots, k_{12}\}$

GACCG	ACCGT	CCGTA
CGTAA	GTAAC	TAACA
AACAG	ACAGC	CAGCG

1.2

Hashing

Hash Values of K-mers:
$h(K_{S,4}) = \{h(k_4), h(k_5), \ldots, h(k_{12})\}$

0b1011	0b0110	0b1110
0b0111	0b1000	0b0010
0b1111	0b0101	0b0011

Minimizer

1.3

Minimizer K-mer: k_9

0b1011	0b0110	0b1110
0b0111	0b1000	0b0010
0b1111	0b0101	0b0011

ATTGACCGTAACAGCGTAT...
Minimizer

2

Seed

$n=5$

Neighbors
ATTGACCGTAACAGCGTAT...
Minimizer
Seed

Hash values of Seed K-mers:
$h(K'_{S,4}) = \{h(k_5), h(k_6), \ldots, h(k_9)\}$

0b1011	0b0110	0b1110
0b0111	0b1000	0b0010
0b1111	0b0101	0b0011

3

Encode

3.1

[-1, 1, 1,-1]
[1, 1, 1,-1]
[-1, 1, 1, 1]
[1,-1,-1,-1]
[-1,-1, 1,-1]

3.2

Counter Vector (C)
[-1, 1, 3,-3]

Decode

3.3

Seed Hash Value (B)
0b0110

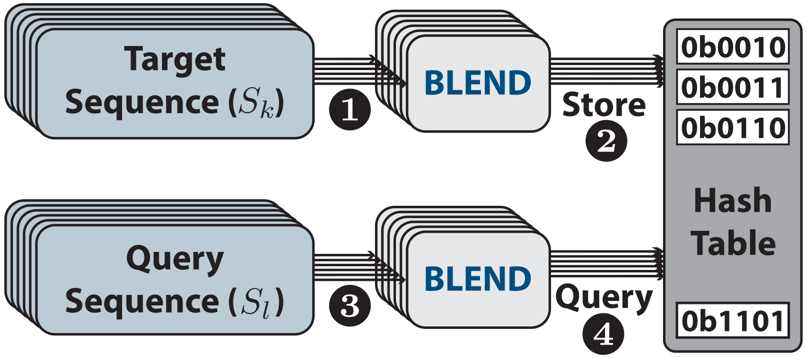
4

0b1000
0b0110
...
Hash Table

Querying the Hash Table

- For each query sequence (e.g., a read sequence), apply the same steps 1-3 to generate the seeds and their hash values

- Use the hash table that BLEND builds for the target sequence to find matching hash values
  - Between target and query sequences
  - Matching hash values: List of fuzzy seed matches
  - BLEND can assign the same hash value for similar seeds



Evaluation Methodology

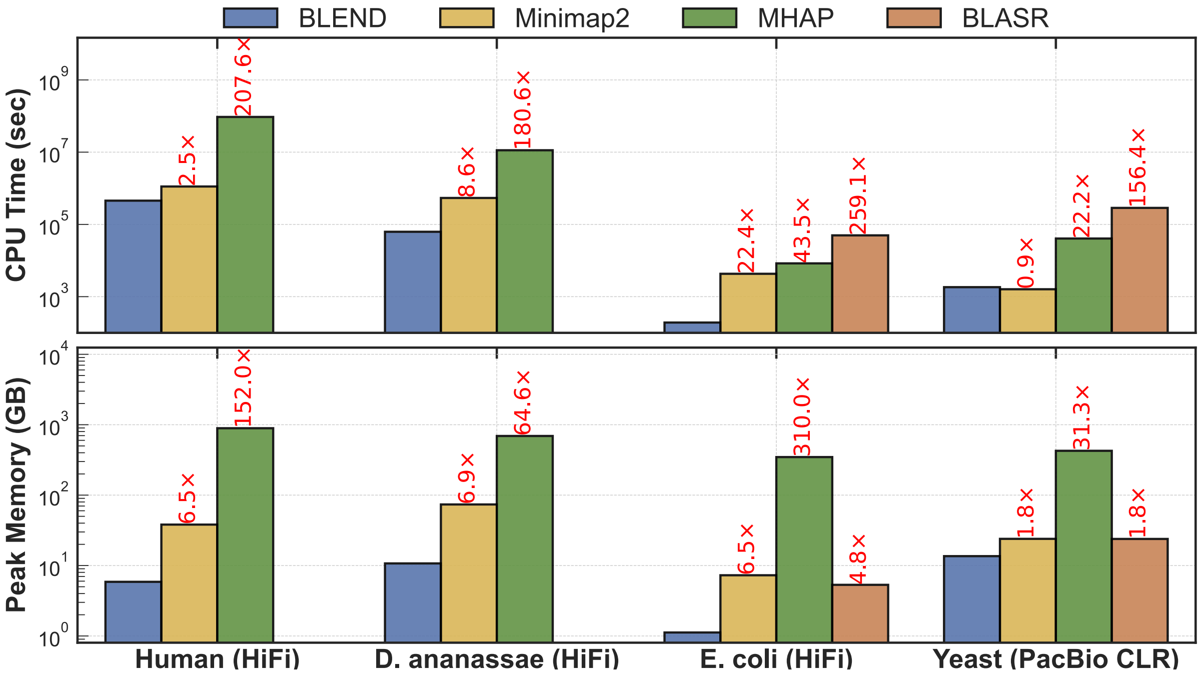
- We integrate the BLEND mechanism into Minimap2 so that it can find fuzzy seed matches when performing end-to-end 1) finding overlapping reads and 2) read mapping

- Real and simulated datasets
  - Real accurate and long reads (HiFi): Human, D. ananassae (fruit fly), and E. coli (bacteria) genomes
  - Simulated erroneous long reads (PacBio CLR): Yeast genome
  - Real Illumina paired-end short reads: Yeast genome

- Use Cases
  - Finding overlapping reads
  - Read mapping

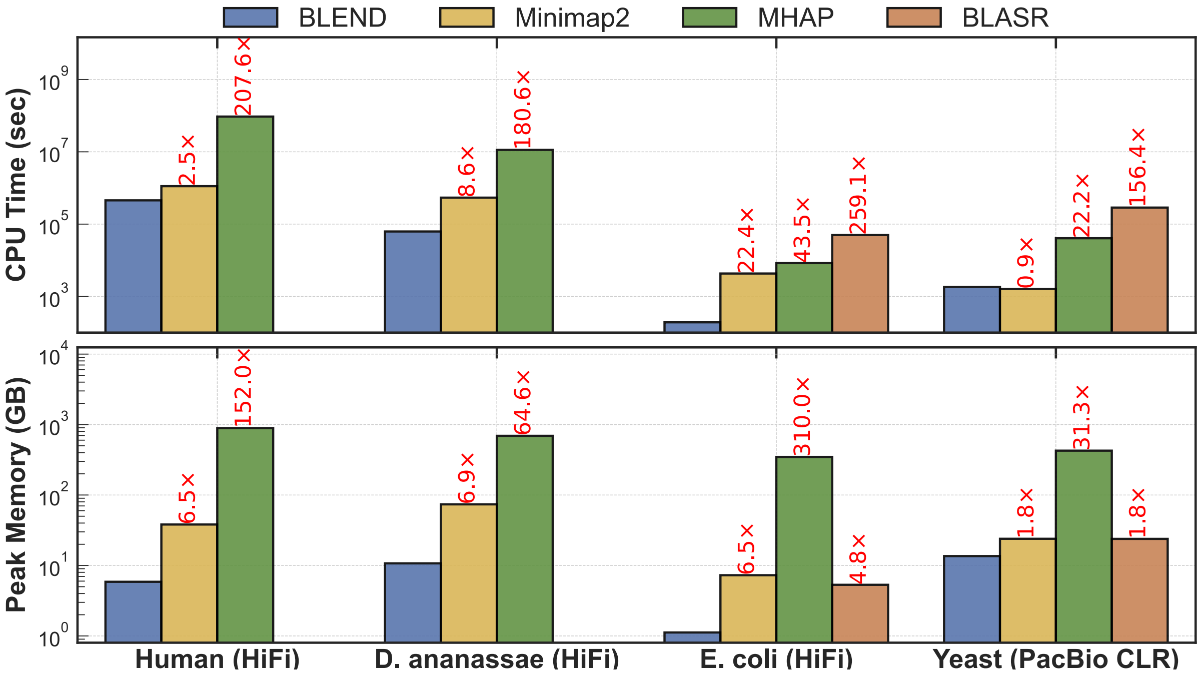
Finding Overlapping Reads -- Performance

- **Observation 1:**
  - Speedup by 8.6x, 113.48x, and 207.75x, on average
  - Lower memory footprint by 5.43, 139.48x, and 3.3x
  - Compared to Minimap2, MHAP, and BLASR, respectively



Finding Overlapping Reads -- Performance

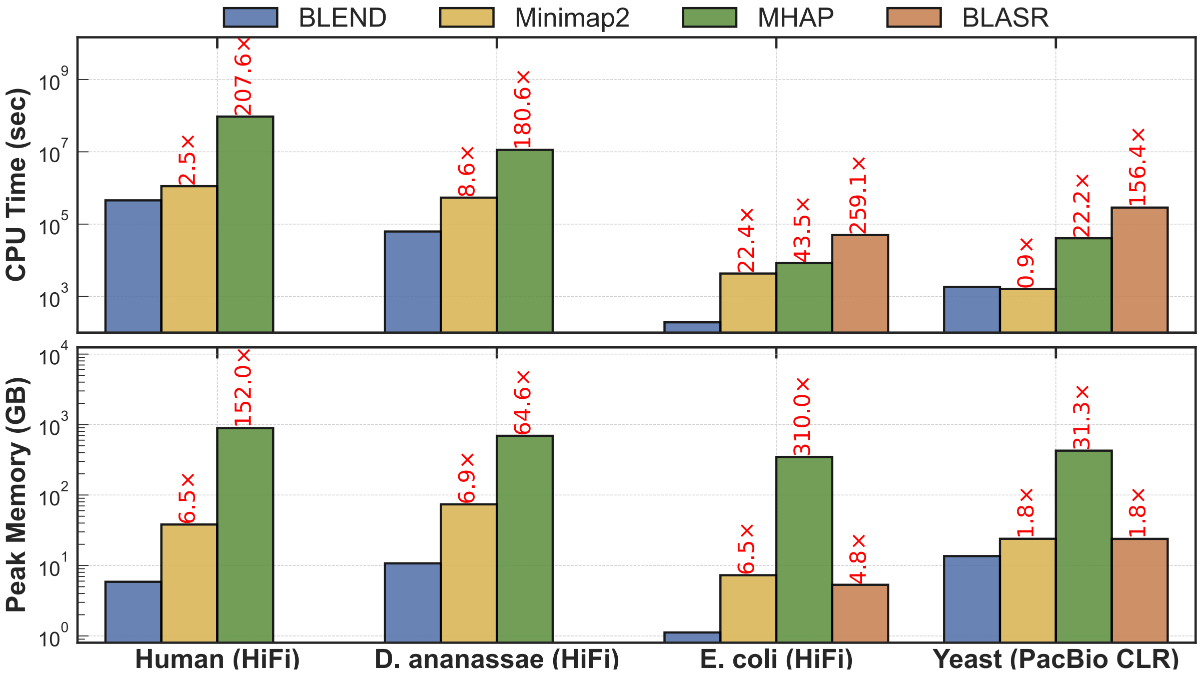
- **Observation 2:** when considering only HiFi reads
  - Speedup by 11.16x, 143.9x, and 259x, on average
  - Lower memory footprint by 6.63x, 175.53x, 4.8x, on average
  - Compared to Minimap2, MHAP, and BLASR, respectively



Finding Overlapping Reads -- Performance

- **Observation 3:** BLEND requires less than 16GB of memory in all cases, making it largely possible to perform finding overlapping reads even in personal computers

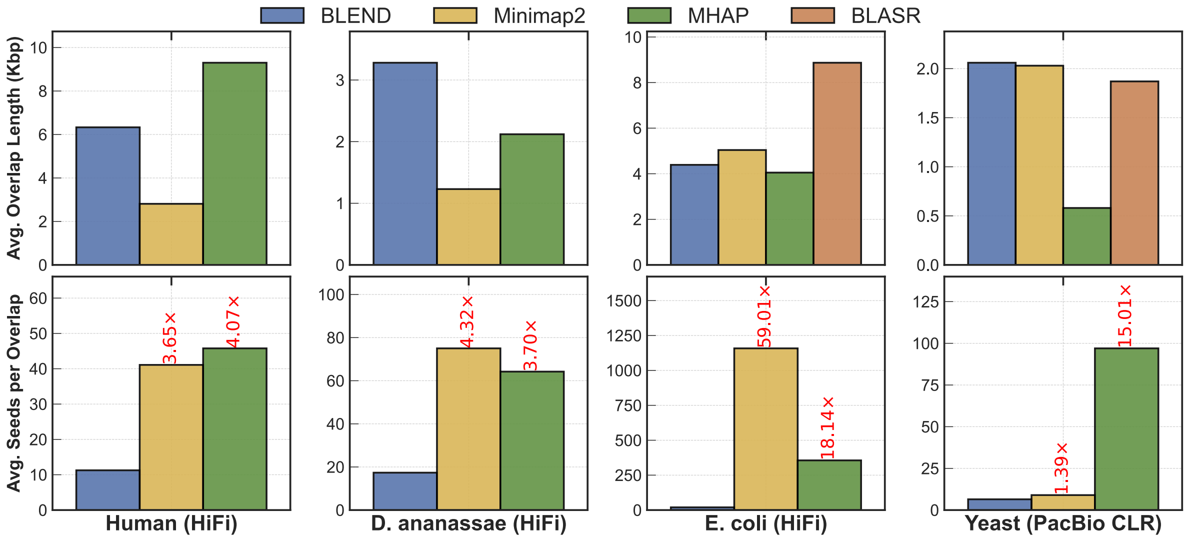
- **Observation 4:** BLEND is not as efficient when using erroneous PacBio CLR reads



Finding Overlapping Reads – Overlap Statistics

- **Observation 1:** BLEND usually finds longer overlaps between reads than most tools in all cases

- **Observation 2:** BLEND uses much fewer seeds to find these longer overlaps



Finding Overlapping Reads – Quality

- More contiguous assemblies as BLEND can find long overlaps

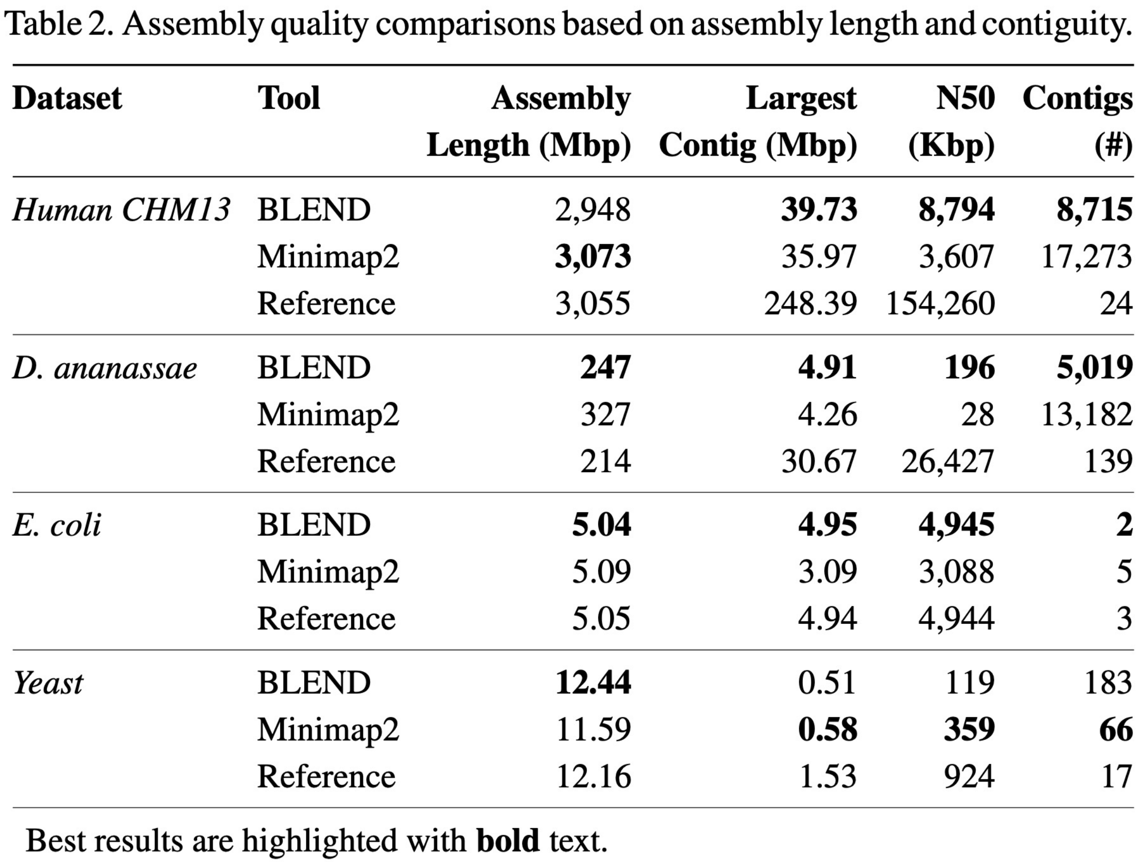


Table 2. Assembly quality comparisons based on assembly length and contiguity.

Dataset	**Tool**	**Assembly Length (Mbp)**	**Largest Contig (Mbp)**	**N50 (Kbp)**	**Contigs (#)**
Human CHM13	BLEND	2,948	**39.73**	**8,794**	**8,715**
	Minimap2	**3,073**	35.97	3,607	17,273
	Reference	3,055	248.39	154,260	24
D. ananassae	BLEND	**247**	**4.91**	**196**	**5,019**
	Minimap2	327	4.26	28	13,182
	Reference	214	30.67	26,427	139
E. coli	BLEND	**5.04**	**4.95**	**4,945**	**2**
	Minimap2	5.09	3.09	3,088	5
	Reference	5.05	4.94	4,944	3
Yeast	BLEND	**12.44**	0.51	119	183
	Minimap2	11.59	**0.58**	**359**	**66**
	Reference	12.16	1.53	924	17

Best results are highlighted with **bold** text.

Finding Overlapping Reads – Accuracy

- More accurate assemblies as BLEND can fill the missing information by using unique fuzzy seed matches

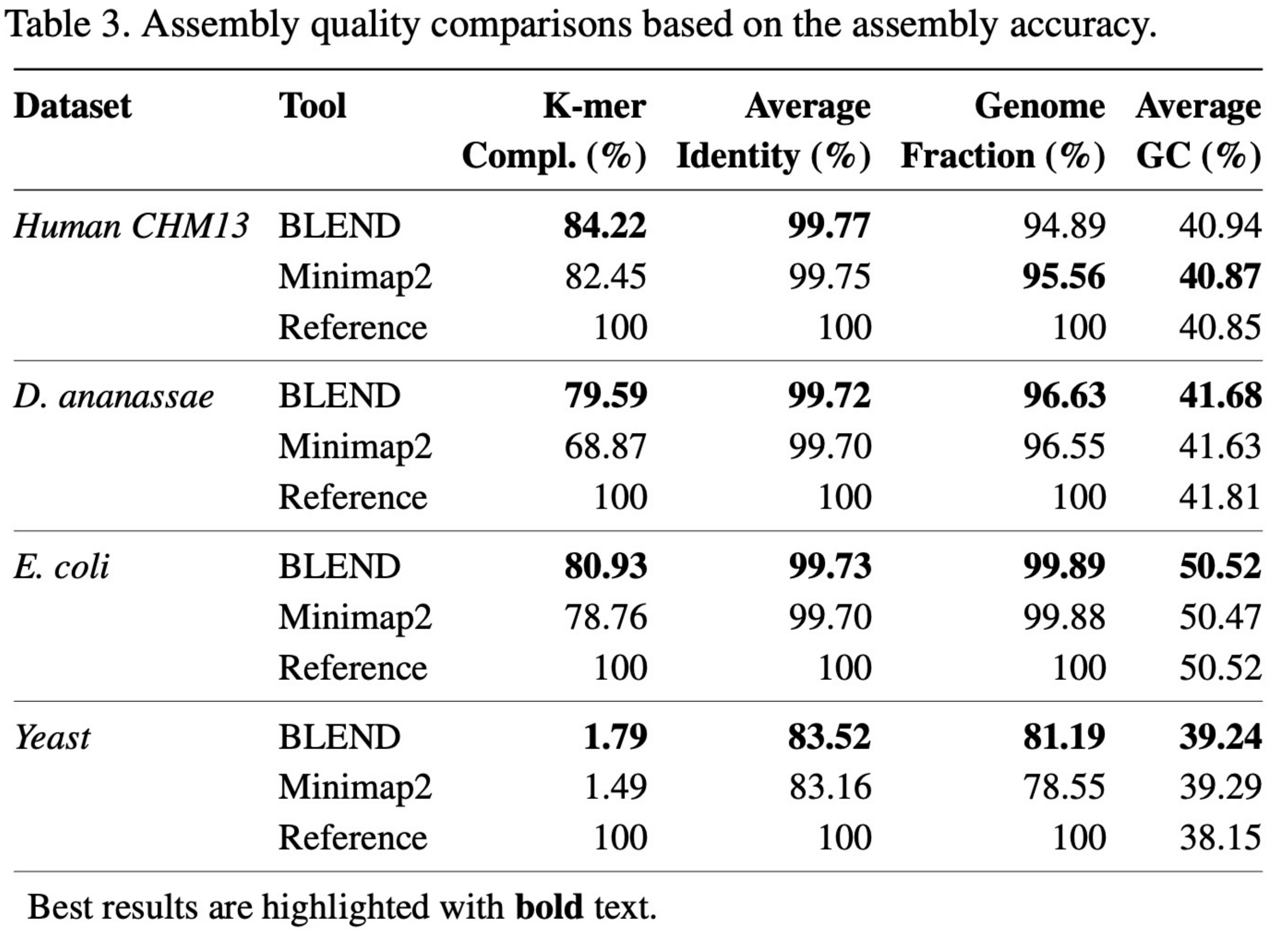


Table 3. Assembly quality comparisons based on the assembly accuracy.

Dataset	**Tool**	**K-mer Compl. (%)**	**Average Identity (%)**	**Genome Fraction (%)**	**Average GC (%)**
Human CHM13	BLEND	**84.22**	**99.77**	94.89	40.94
	Minimap2	82.45	99.75	**95.56**	**40.87**
	Reference	100	100	100	40.85
D. ananassae	BLEND	**79.59**	**99.72**	**96.63**	**41.68**
	Minimap2	68.87	99.70	96.55	41.63
	Reference	100	100	100	41.81
E. coli	BLEND	**80.93**	**99.73**	**99.89**	**50.52**
	Minimap2	78.76	99.70	99.88	50.47
	Reference	100	100	100	50.52
Yeast	BLEND	**1.79**	**83.52**	**81.19**	**39.24**
	Minimap2	1.49	83.16	78.55	39.29
	Reference	100	100	100	38.15

Best results are highlighted with **bold** text.

Conclusion

- BLEND can find fuzzy seed matches efficiently to improve the performance, memory efficiency, and accuracy

- Significant speedups and lower memory footprint especially when using HiFi reads
  - We can increase the window length without reducing the accuracy thanks to finding unique fuzzy seed matches that Minimap2 cannot find

- BLEND finds longer overlapping regions between sequences while using fewer seeds to find these overlaps

Executive Summary

- **Problem:** Using seed matches between sequences lead to
  - Too many costly chaining and alignment operations due to many short exact-matching seeds
  - Finding fuzzy seed matches impose high performance and memory space overheads

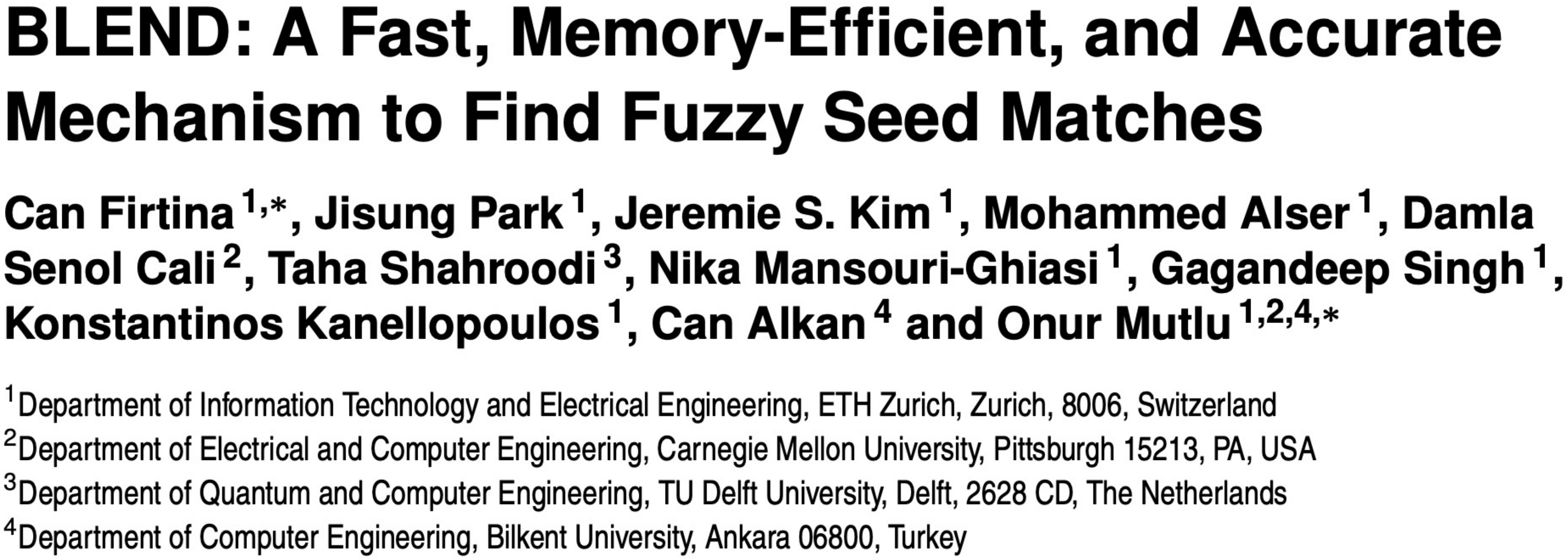
- **Goal:** Quickly and efficiently find fuzzy seed matches between sequences to improve the performance, memory efficiency, and accuracy of the applications that use seeds

- **Key Ideas:**
  - Generate the same hash value for highly similar seeds using a hashing technique called SimHash
    - A single hash value for a seed to find fuzzy seed matches, unlike other works that use many hash values
    - The same hash value for highly similar seeds even though these seeds may mismatch at any arbitrary position, unlike spaced seeding
  - Build and use the hash table using these hash values that BLEND generates

- **Key Results**
  - For finding overlapping reads, on average, 8.6x speedup and 5.43x lower memory footprint compared to the state-of-the-art, Minimap2
  - We can construct more accurate and contiguous assemblies using the overlapping reads that BLEND finds compared to using the ones Minimap2 finds

Finding Efficient and Accurate Seed Matches

- Can Firtina, Jisung Park, Jeremie S. Kim, Mohammed Alser, Damla Senol Cali, Taha Shahroodi, Nika Mansouri-Ghiasi, Gagandeep Singh, Konstantinos Kanellopoulos, Can Alkan, and Onur Mutlu,
  "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches"
  Preprint in ***arXiv***, 16 December 2021.
  [arXiv preprint]
  [BLEND Source Code and Data]



BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches

Can Firtina [1,*], Jisung Park [1], Jeremie S. Kim [1], Mohammed Alser [1], Damla Senol Cali [2], Taha Shahroodi [3], Nika Mansouri-Ghiasi [1], Gagandeep Singh [1], Konstantinos Kanellopoulos [1], Can Alkan [4] and Onur Mutlu [1,2,4,*]

[1] Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, 8006, Switzerland
[2] Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh 15213, PA, USA
[3] Department of Quantum and Computer Engineering, TU Delft University, Delft, 2628 CD, The Netherlands
[4] Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey

Prior Research on Genome Analysis (1/2)

- Firtina +, "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches", arXiv, 2021.
- Alser +, "SneakySnake: A Fast and Accurate Universal Genome Pre-Alignment Filter for CPUs, GPUs, and FPGAs." to appear in *Bioinformatics*, 2020.
- Senol Cali+, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis", *MICRO* 2020.
- Alser+, "Technology dictates algorithms: Recent developments in read alignment", to appear in *Genome Biology*, 2021.
- Kim+, "AirLift: A Fast and Comprehensive Technique for Translating Alignments between Reference Genomes", *arXiv*, 2020
- Alser+, "Accelerating Genome Analysis: A Primer on an Ongoing Journey", *IEEE Micro*, 2020.

Prior Research on Genome Analysis (2/2)

- Firtina+, "Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm", *Bioinformatics*, 2020.

- Alser+, "Shouji: a fast and efficient pre-alignment filter for sequence alignment", *Bioinformatics* 2019.

- Kim+, "GRIM-Filter: Fast Seed Location Filtering in DNA Read Mapping Using Processing-in-Memory Technologies", *BMC Genomics*, 2018.

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

- Alser+, "MAGNET: understanding and improving the accuracy of genome pre-alignment filtering", *IPSI Transaction*, 2017.

Enabling Accurate, Fast, and Memory-Efficient Genome Analysis via Efficient and Intelligent Algorithms

Can Firtina
canfirtina@gmail.com

10 January 2022

METU Graduate Seminar in Bioinformatics

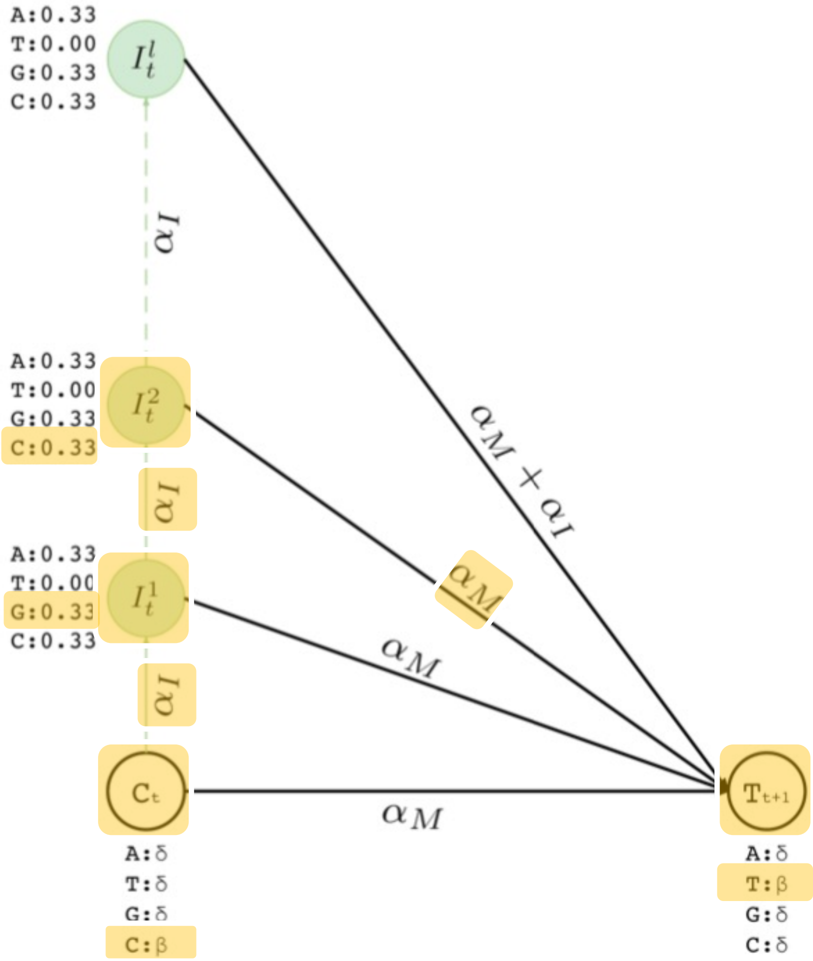
SAFARI

ETH zürich

Backup Slides

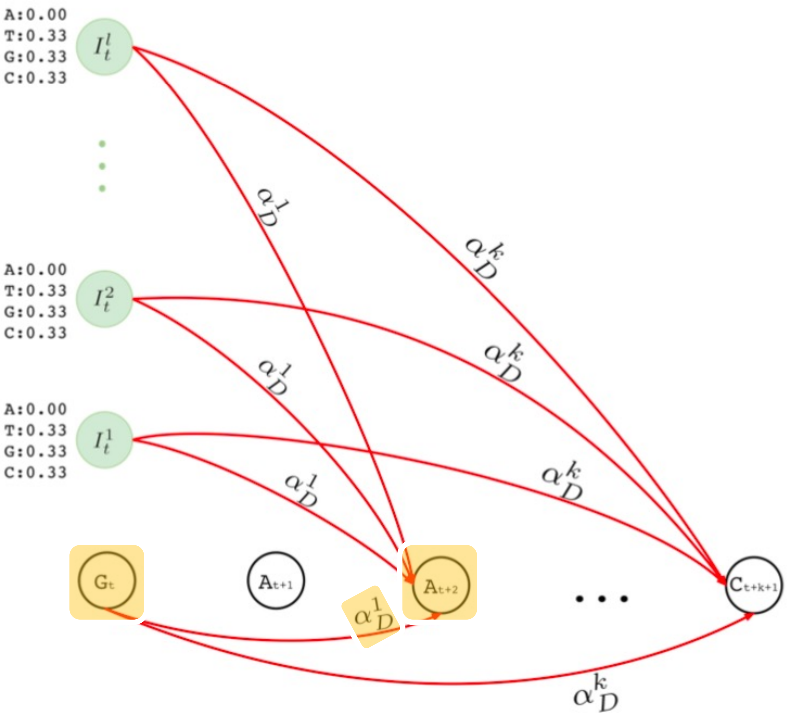
Resolving deletion errors

- Insertion states to insert *at most* / many bases *between two bases in a contig*
- To insert "GC" between "CT"
  - Visit match state at position t and *emit C*
  - Visit first *insertion state* after position t and *emit G with* ***deletion error*** *probability*
  - Visit second insertion state and *emit C with deletion error probability*
  - From second insertion state visit match state at position t+1 and *emit T*
  - Resulting sequence "CGCT"
- Maximum number of insertions is a parameter to Apollo



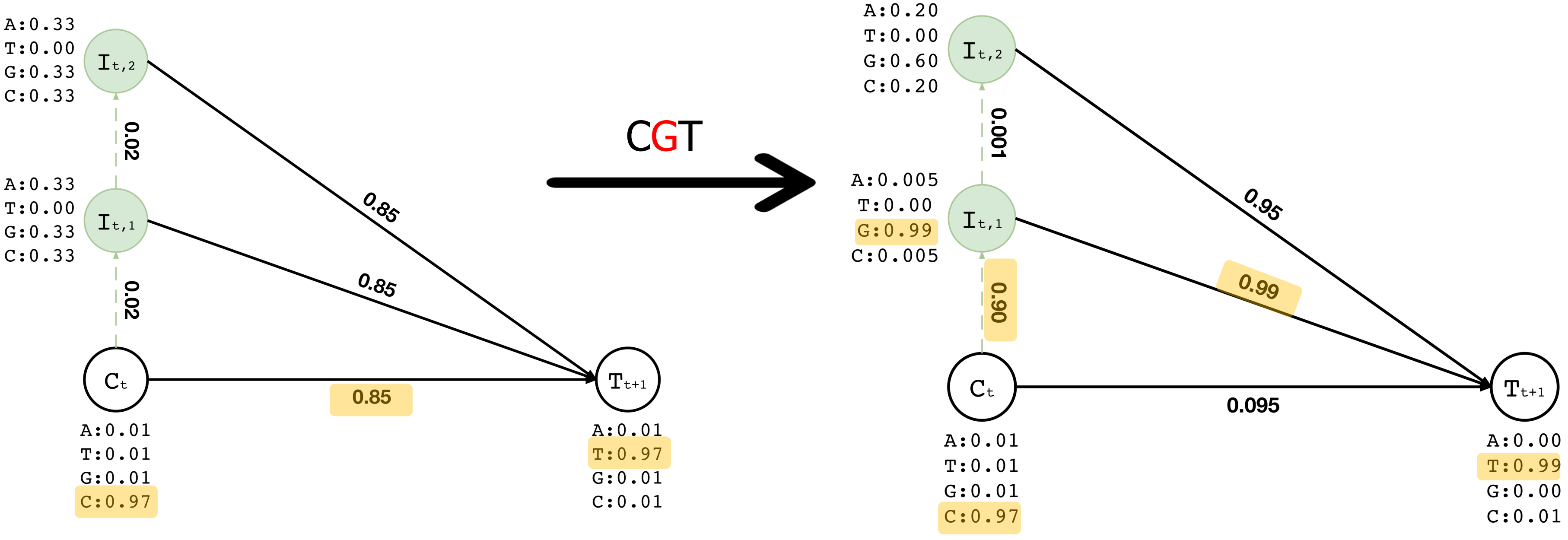
Resolving insertion errors

- Deletion transitions to delete one or many bases in a row
- To delete the first A in “GAA”
  - Visit match state at position t and *emit G*
  - Visit match state at position t+2 and emit A with *single* **insertion error** probability
  - Resulting sequence: “GA”
- Having single or more deletions in a row may not be necessarily equally likely
- Maximum number of deletions in a row is a parameter to Apollo



Training

- Training data:
  - Read aligned to the location t of a contig
- Assume we have the read "CGT" aligned to location t
- After training the corresponding region of the graph we would expect change in the probabilities so that it will be likely to emit "CGT" **somehow**



The Forward-Backward algorithm

- Calculating the likelihood of visiting a state to emit a certain character of a given sequence (i.e., aligned read)
  - Forward calculation (F)

$$F_1(j) = \alpha_{0j} e_j(r[1]) \quad s.t. \quad j \in V_s, \quad E_{0j} \in E_s$$

$$F_t(j) = \sum_{i \in V_s} F_{t-1}(i) \alpha_{ij} e_j(r[t]) \quad j \in V_s, \quad 1 < t \leq m$$

  - Backward calculation (B)

$$B_m(i) = \alpha_{i(m+1)} \quad i \in V_s, \quad E_{i(m+1)} \in E_s$$

$$B_t(i) = \sum_{j \in V_s} \alpha_{ij} e_j(r[t+1]) B_{t+1}(j) \quad j \in V_s, \; 1 \leq t < m$$

- Backward calculation needs a starting point

Training: The Baum-Welch algorithm

- Expectation maximization step using the Baum-Welch algorithm

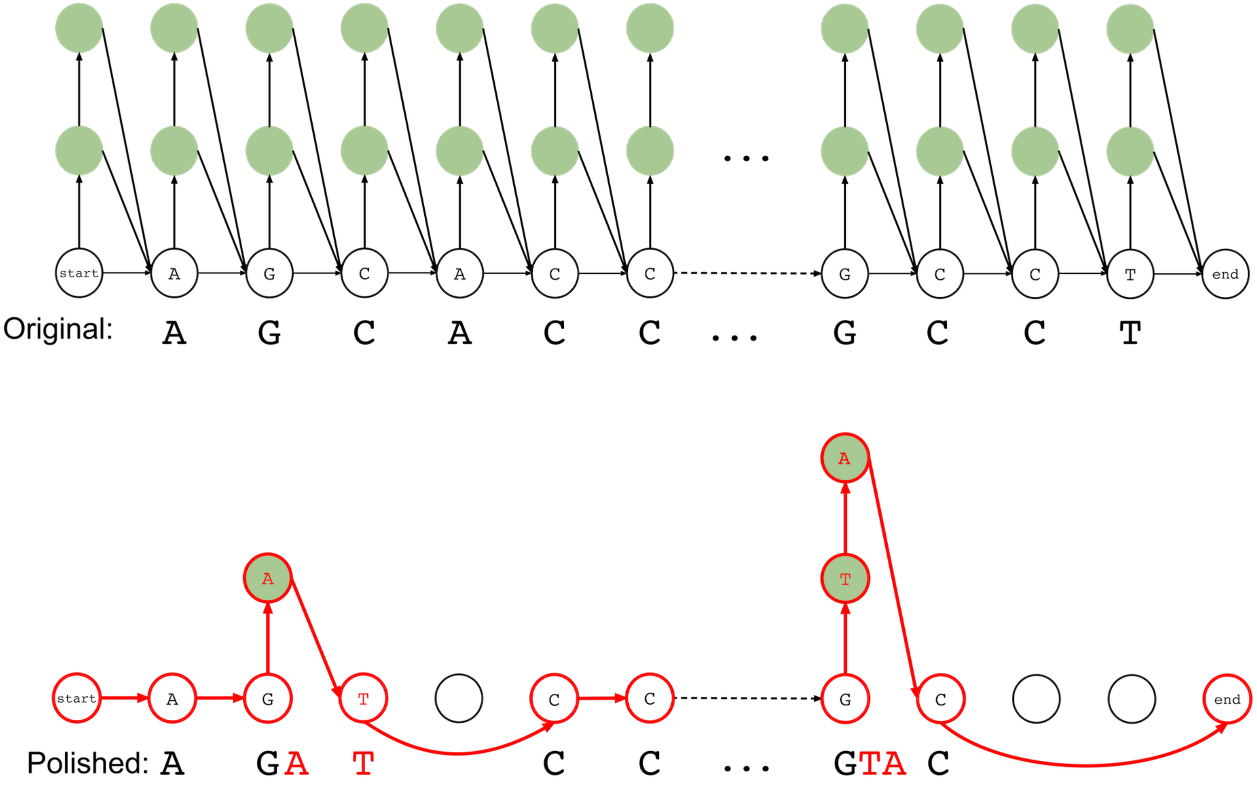
$$e_i^*(X) = \frac{\sum_{t=1}^{m} F_t(i) B_t(i)(r[t] == X)}{\sum_{t=1}^{m} F_t(i) B_t(i)} \quad \forall X \in \Sigma, \forall i \in V_s$$

$$\alpha_{ij}^* = \frac{\sum_{t=1}^{m-1} \alpha_{ij} e_j(r[t+1]) F_t(i) B_{t+1}(j)}{\sum_{t=1}^{m-1} \sum_{x \in V_s} \alpha_{ix} e_x(r[t+1]) F_t(i) B_{t+1}(x)} \quad \forall E_{ij} \in E_s$$

- If there are multiple reads aligning to same region, we have multiple F(i) for a position t
  - Take the average and use it as F(i) for position t

Inference: The Viterbi algorithm

- Our original contig before polishing was: “AGCACC…GCCT”
- After updating the probabilities, the most likely path from start to end reveals the corrected contig: “AGATCC…GTAC”



start
A
G
C
A
C
C
...
G
C
C
T
end
Original: A G C A C C ... G C C T
A
T
A
start
A
G
T
C
C
G
C
end
Polished: A G A T C C ... G T A C

Data Sets

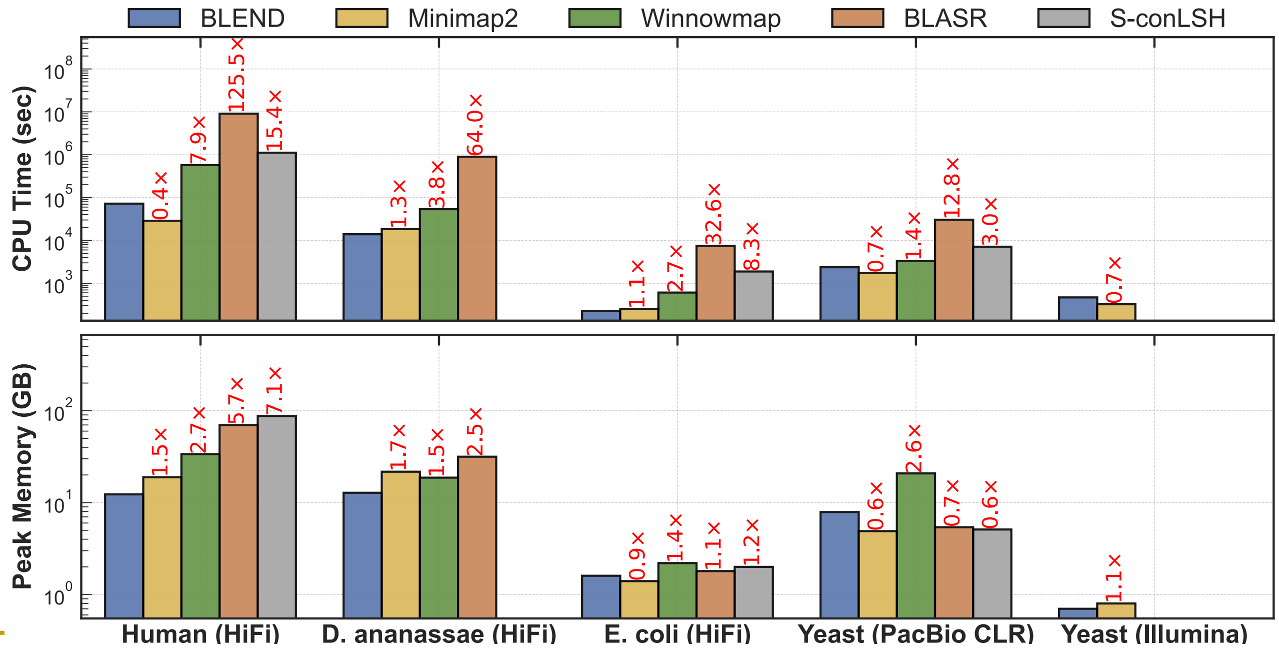
Data Set	Accession Number	Details
E.coli K-12 - ONT	Loman Lab*	164,472 reads (avg. 9,010bps, 319X coverage) via Metrichor
E.coli K-12 - Ground Truth	GenBank NC_000913	Strain MG1655 (4,641Kbps)
E.coli O157 - PacBio	SRA SRR5413248	177,458 reads (avg. 4,724bps, 151X coverage)
E.coli O157 - Illumina	SRA SRR5413247	11,856,506 paired-end reads (150bps each, 643X coverage)
E.coli O157 - Ground Truth	GenBank NJEX02000001	Strain FDAARGOS_292 (5,566Kbps)
E.coli O157:H7 - PacBio	SRA SRR1509640	76,279 reads (avg. 8,270bps, 112X coverage)
E.coli O157:H7 - Illumina	SRA SRR1509643	2,978,835 paired-end reads (250bps each, 265X coverage)
E.coli O157:H7 - Ground Truth	GCA_000732965	Strain EDL933 (5,639Kbps)
Yeast S288C - PacBio	SRA ERR165511(8-9), ERR1655125	296,485 reads (avg. 5,735bps, 140X coverage)
Yeast S288C - Illumina	SRA ERR1938683	3,318,467 paired-end reads (150bps each, 82X coverage)
Yeast S288C - Ground Truth	GCA_000146055.2	Strain S288C (12,157Kbps)
Human CHM1 - PacBio	SRA SRR130433(1-5)	912,421 reads (avg. 8,646bps, 2.6X coverage)
Human CHM1 - Ground Truth	GCA_000306695.2	3.04Gbps
Human HG002 - PacBio	SRA SRR2036(394-471), SRR203665(4-9)	15,892,517 reads (avg. 6,550bps, 35X coverage)
Human HG002 - Illumina	SRA SRR17664(42-59)	222,925,733 paired-end reads (148bps each, 22X coverage)
Human HG002 - Ground Truth	GCA_001542345.1	Ashkenazim trio - Son (2.99Gbps)

Experimental Setup

- CPU: Intel®Xeon®Gold 5118 CPU @ 2.30GHz
  - 24 cores (2 threads per core)
- Max memory: 192GB
- Assigned 45 threads to all tools
- Apollo was compared with the state-of-the-art polishing tools
  - Racon, Pilon, Quiver, Nanopolish

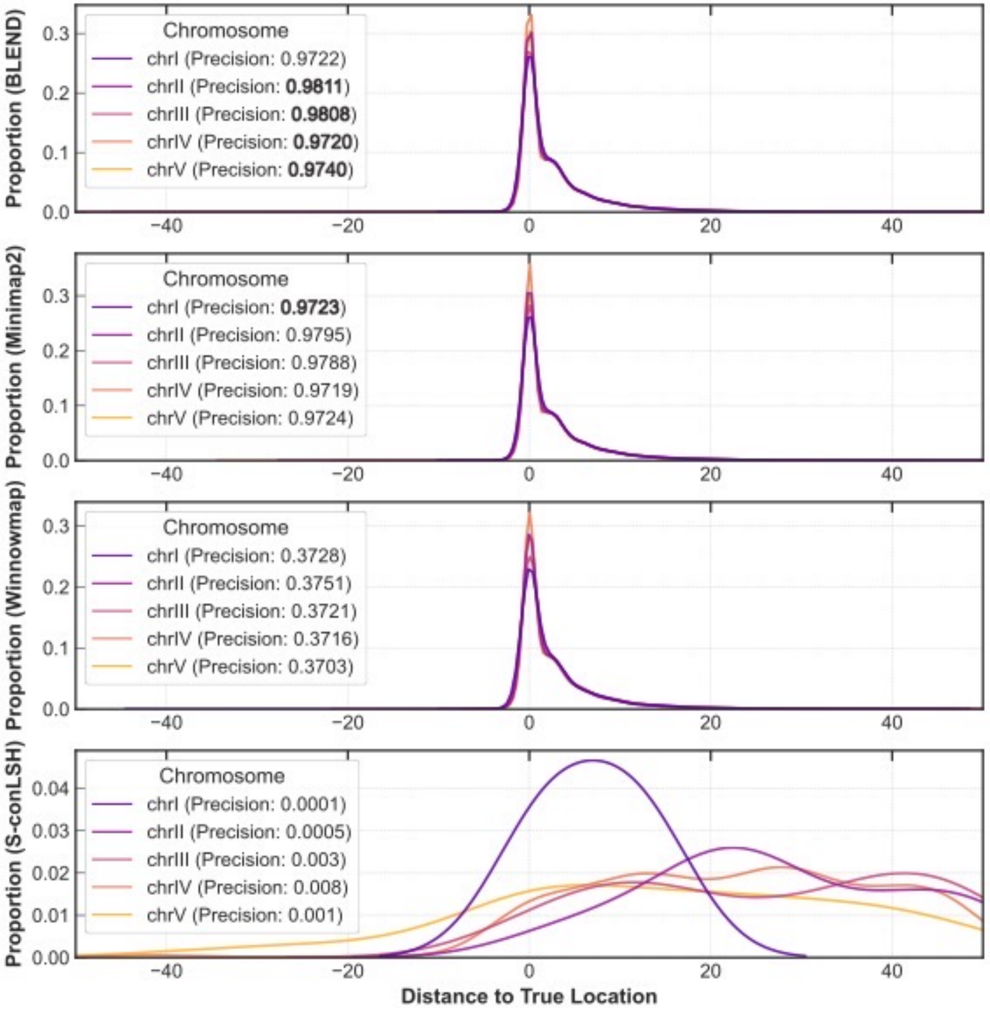
Read Mapping -- Performance

- BLEND is mainly effective when using HiFi reads
- Speedup and memory efficiency is usually lower than what we observe from finding overlapping reads
  - We take an additional computationally costly step in read mapping that hinders the benefits of BLEND
    - Read alignment



Read Mapping – Accuracy

- BLEND maps more reads to their correct chromosomes (better precision)
- S-conLSH provides the worst accuracy as the precision and the average distance to true position is always the lowest compared to others



Read Mapping – Quality

- BLEND usually provides similar quality to Minimap2
- Winnowmap usually provides the best quality when using HiFi reads

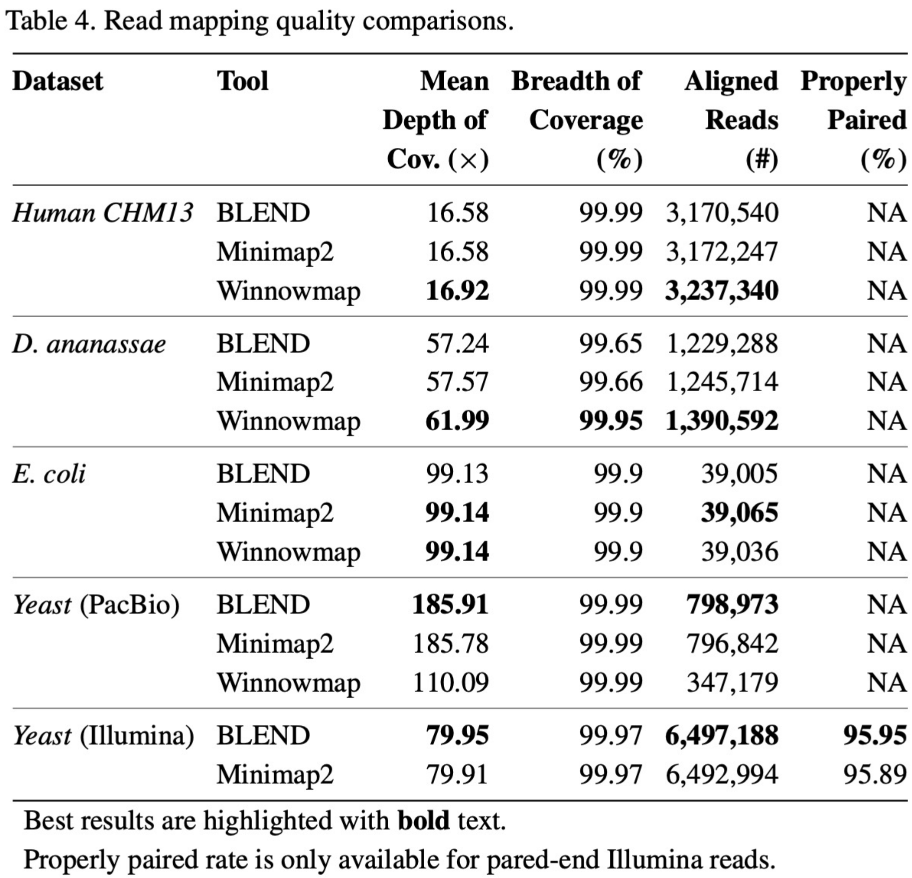


Table 4. Read mapping quality comparisons.

Dataset	Tool	Mean Depth of Cov. (×)	Breadth of Coverage (%)	Aligned Reads (#)	Properly Paired (%)
Human CHM13	BLEND	16.58	99.99	3,170,540	NA
	Minimap2	16.58	99.99	3,172,247	NA
	Winnowmap	**16.92**	99.99	**3,237,340**	NA
D. ananassae	BLEND	57.24	99.65	1,229,288	NA
	Minimap2	57.57	99.66	1,245,714	NA
	Winnowmap	**61.99**	**99.95**	**1,390,592**	NA
E. coli	BLEND	99.13	99.9	39,005	NA
	Minimap2	**99.14**	99.9	**39,065**	NA
	Winnowmap	**99.14**	99.9	39,036	NA
Yeast (PacBio)	BLEND	**185.91**	99.99	**798,973**	NA
	Minimap2	185.78	99.99	796,842	NA
	Winnowmap	110.09	99.99	347,179	NA
Yeast (Illumina)	BLEND	**79.95**	99.97	**6,497,188**	**95.95**
	Minimap2	79.91	99.97	6,492,994	95.89

Best results are highlighted with **bold** text.
Properly paired rate is only available for pared-end Illumina reads.

Readings, Videos, Reference Materials

Research & Teaching: Some Overview Talks

https://www.youtube.com/onurmutlulectures

- Future Computing Architectures
  - https://www.youtube.com/watch?v=kgiZlSOcGFM&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=1
- Enabling In-Memory Computation
  - https://www.youtube.com/watch?v=njX_14584Jw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=16
- Accelerating Genome Analysis
  - https://www.youtube.com/watch?v=r7sn41lH-4A&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=41
- Rethinking Memory System Design
  - https://www.youtube.com/watch?v=F7xZLNMIY1E&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=3
- Intelligent Architectures for Intelligent Machines
  - https://www.youtube.com/watch?v=c6_LgzuNdkw&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=25
- The Story of RowHammer
  - https://www.youtube.com/watch?v=sgd7PHQQ1AI&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=39

Accelerated Memory Course (~6.5 hours)

- ACACES 2018
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu July 9-13, 2018
  - ~6.5 hours of lectures

- Website for the Course including Videos, Slides, Papers
  - https://people.inf.ethz.ch/omutlu/acaces2018.html
  - https://www.youtube.com/playlist?list=PL5Q2soXY2Zi-HXxomthrpDpMJm05P6J9x

- All Papers are at:
  - https://people.inf.ethz.ch/omutlu/projects.htm
  - Final lecture notes and readings (for all topics)

Longer Memory Course (~18 hours)

- TU Wien 2019
  - Memory Systems and Memory-Centric Computing Systems
  - Taught by Onur Mutlu June 12-19, 2019
  - ~18 hours of lectures

- Website for the Course including Videos, Slides, Papers
  - https://safari.ethz.ch/memory_systems/TUWien2019
  - https://www.youtube.com/playlist?list=PL5Q2soXY2Zi_gntM55VoMlKlw7YrXOhbl

- All Papers are at:
  - https://people.inf.ethz.ch/omutlu/projects.htm
  - Final lecture notes and readings (for all topics)

An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
  - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

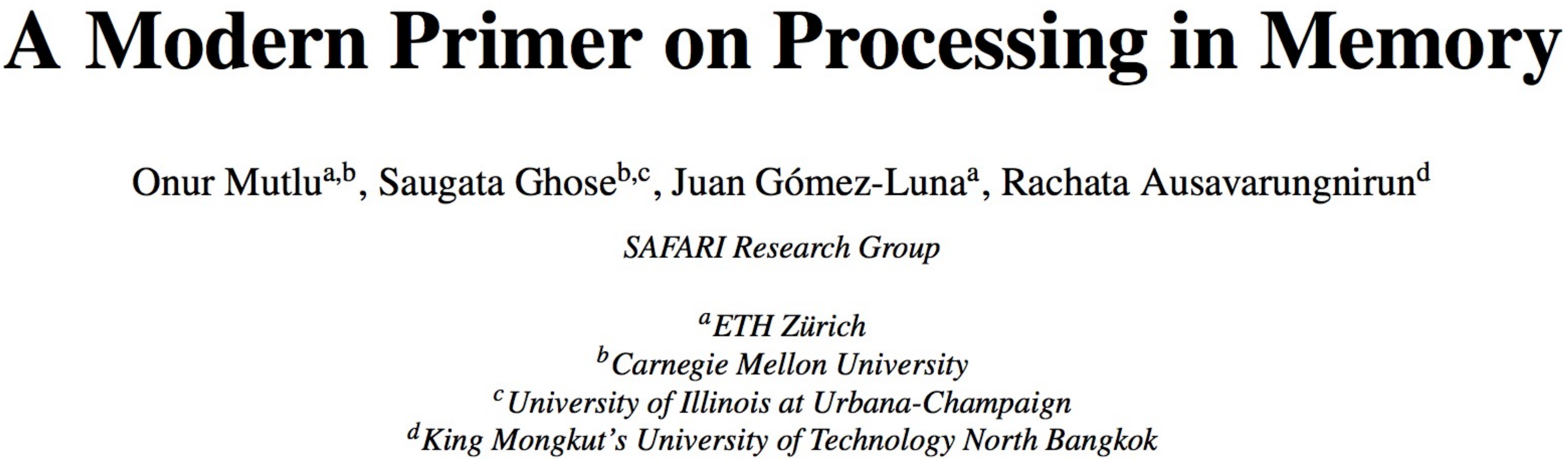
- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15

More Thoughts and Suggestions

- Onur Mutlu,
  "Some Reflections (on DRAM)"
  *Award Speech for ACM SIGARCH Maurice Wilkes Award, at the **ISCA** Awards Ceremony*, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  *57th Design Automation Conference Early Career Workshop (**DAC**)*, Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]

Reference Overview Paper



A Modern Primer on Processing in Memory

Onur Mutlu[a,b], Saugata Ghose[b,c], Juan Gómez-Luna[a], Rachata Ausavarungnirun[d]

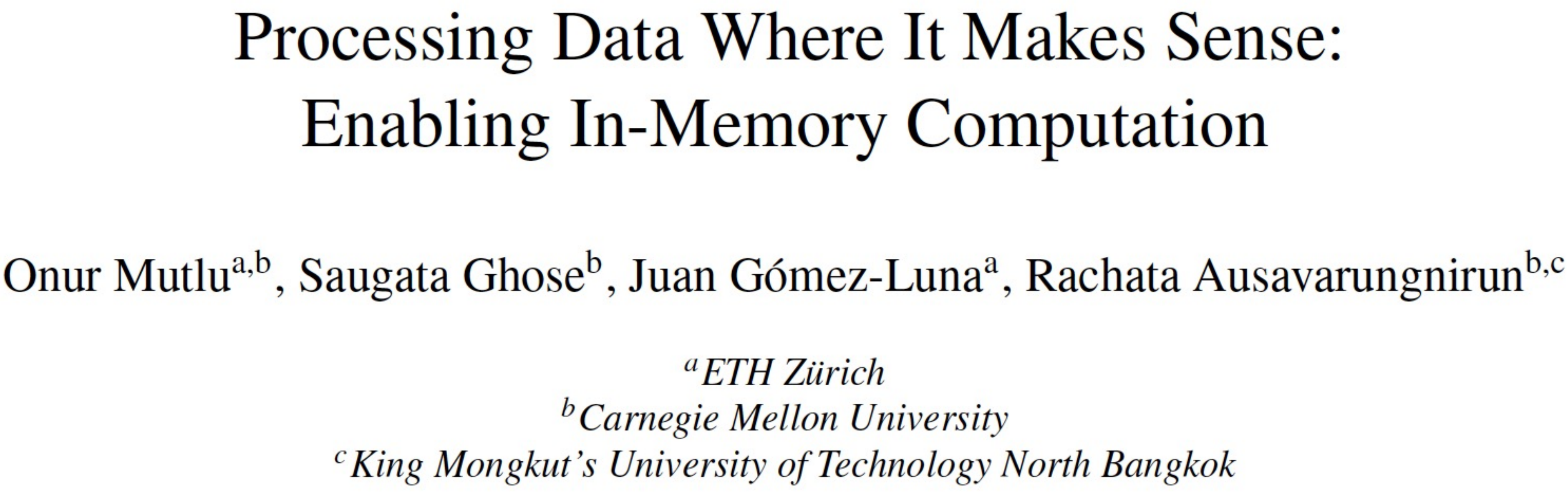
SAFARI Research Group

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *University of Illinois at Urbana-Champaign*
[d] *King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, **"A Modern Primer on Processing in Memory"**
Invited Book Chapter in ***Emerging Computing: From Devices to Systems - Looking Beyond Moore and Von Neumann***, Springer, to be published in 2021.

https://arxiv.org/pdf/1903.03988.pdf

Reference Overview Paper I



Processing Data Where It Makes Sense: Enabling In-Memory Computation

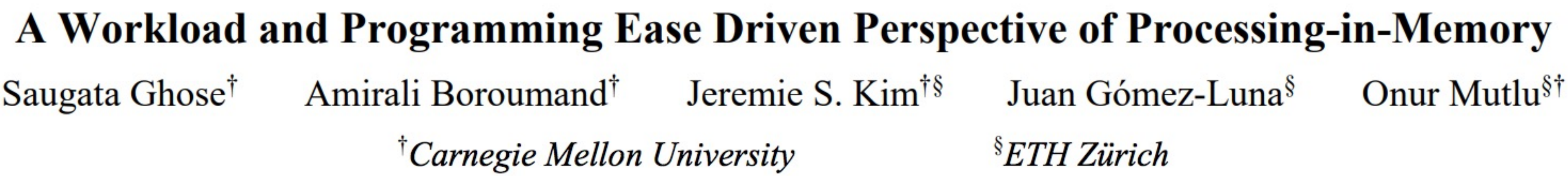
Onur Mutlu[a,b], Saugata Ghose[b], Juan Gómez-Luna[a], Rachata Ausavarungnirun[b,c]

[a] *ETH Zürich*
[b] *Carnegie Mellon University*
[c] *King Mongkut's University of Technology North Bangkok*

Onur Mutlu, Saugata Ghose, Juan Gomez-Luna, and Rachata Ausavarungnirun, **"Processing Data Where It Makes Sense: Enabling In-Memory Computation"**
Invited paper in Microprocessors and Microsystems (***MICPRO***), June 2019.
[arXiv version]

https://arxiv.org/pdf/1903.03988.pdf

Reference Overview Paper II



A Workload and Programming Ease Driven Perspective of Processing-in-Memory

Saugata Ghose[†] Amirali Boroumand[†] Jeremie S. Kim[†§] Juan Gómez-Luna[§] Onur Mutlu[§†]

[†] *Carnegie Mellon University* [§] *ETH Zürich*

Saugata Ghose, Amirali Boroumand, Jeremie S. Kim, Juan Gomez-Luna, and Onur Mutlu, **"Processing-in-Memory: A Workload-Driven Perspective"** *Invited Article in IBM Journal of Research & Development, Special Issue on Hardware for Artificial Intelligence*, to appear in November 2019. [Preliminary arXiv version]

https://arxiv.org/pdf/1907.12947.pdf

Reference Overview Paper III

Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions

SAUGATA GHOSE, KEVIN HSIEH, AMIRALI BOROUMAND, RACHATA AUSAVARUNGNIRUN
Carnegie Mellon University

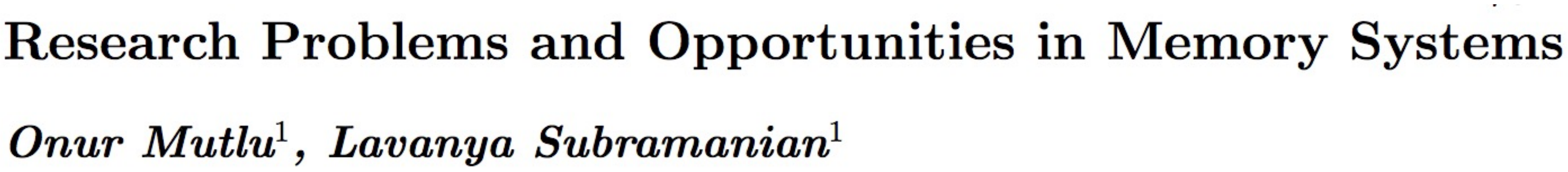
ONUR MUTLU
ETH Zürich and Carnegie Mellon University

Saugata Ghose, Kevin Hsieh, Amirali Boroumand, Rachata Ausavarungnirun, Onur Mutlu, **"Enabling the Adoption of Processing-in-Memory: Challenges, Mechanisms, Future Research Directions"**
Invited Book Chapter, to appear in 2018.
[Preliminary arxiv.org version]

https://arxiv.org/pdf/1802.00320.pdf

Reference Overview Paper IV

- Onur Mutlu and Lavanya Subramanian,
  "Research Problems and Opportunities in Memory Systems"
  *Invited Article in Supercomputing Frontiers and Innovations (**SUPERFRI**)*, 2014/2015.



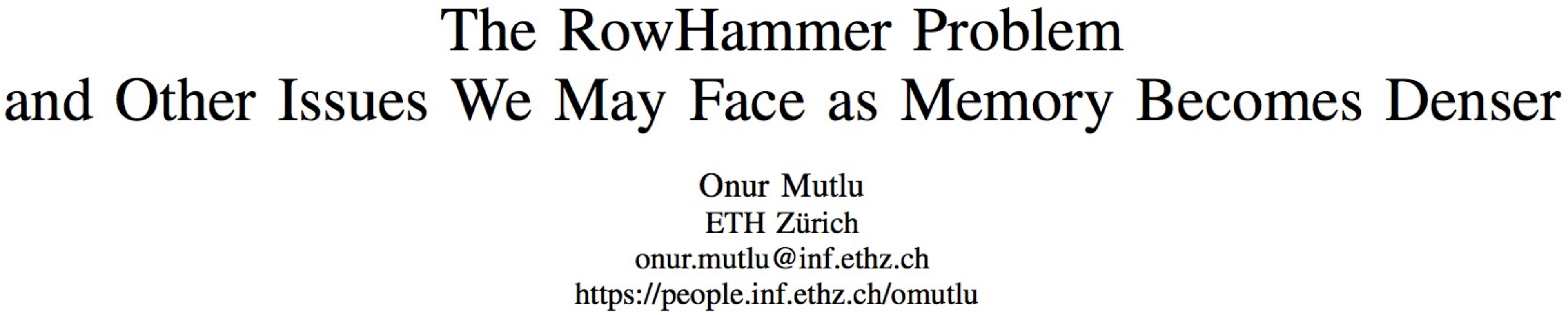
Research Problems and Opportunities in Memory Systems

Onur Mutlu[1], *Lavanya Subramanian*[1]

https://people.inf.ethz.ch/omutlu/pub/memory-systems-research_superfri14.pdf

Reference Overview Paper V

- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Lausanne, Switzerland, March 2017.
  [Slides (pptx) (pdf)]



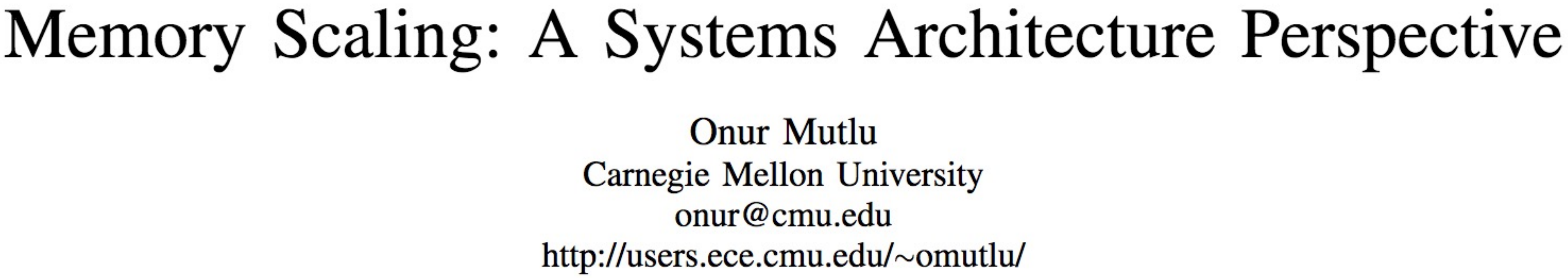
The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser

Onur Mutlu
ETH Zürich
onur.mutlu@inf.ethz.ch
https://people.inf.ethz.ch/omutlu

https://people.inf.ethz.ch/omutlu/pub/rowhammer-and-other-memory-issues_date17.pdf

Reference Overview Paper VI

- Onur Mutlu,
  "Memory Scaling: A Systems Architecture Perspective"
  Technical talk at MemCon 2013 (***MEMCON***), Santa Clara, CA, August 2013. [Slides (pptx) (pdf)]
  [Video] [Coverage on StorageSearch]

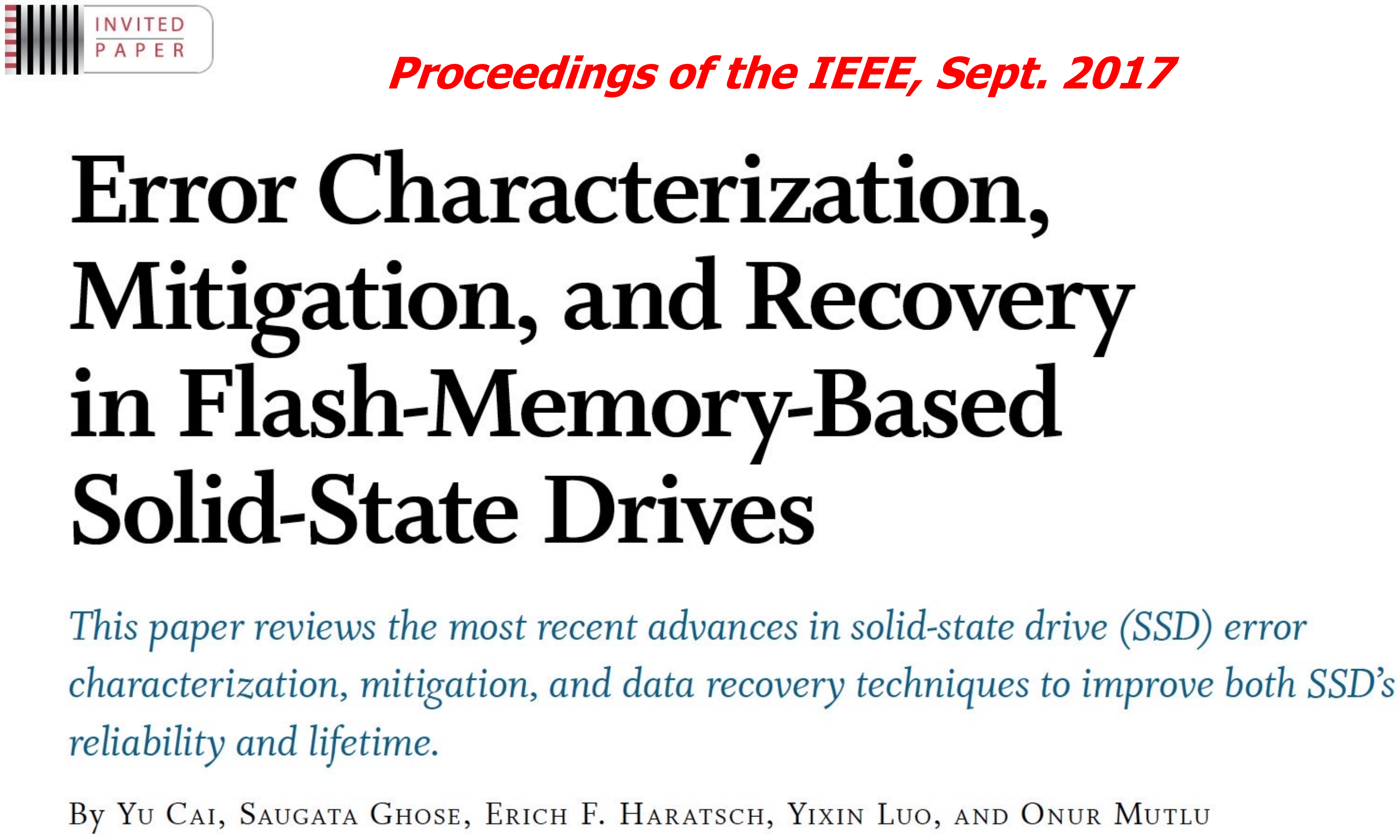


Memory Scaling: A Systems Architecture Perspective

Onur Mutlu
Carnegie Mellon University
onur@cmu.edu
http://users.ece.cmu.edu/~omutlu/

https://people.inf.ethz.ch/omutlu/pub/memory-scaling_memcon13.pdf

Reference Overview Paper VII



INVITED PAPER

Proceedings of the IEEE, Sept. 2017

Error Characterization, Mitigation, and Recovery in Flash-Memory-Based Solid-State Drives

This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

By Yu Cai, Saugata Ghose, Erich F. Haratsch, Yixin Luo, and Onur Mutlu

https://arxiv.org/pdf/1706.08642

Reference Overview Paper VIII

- Onur Mutlu and Jeremie Kim,
  "RowHammer: A Retrospective"
  *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (**TCAD**) Special Issue on Top Picks in Hardware and Embedded Security*, 2019.
  [Preliminary arXiv version]
  [Slides from COSADE 2019 (pptx)]
  [Slides from VLSI-SOC 2020 (pptx) (pdf)]
  [Talk Video (30 minutes)]

RowHammer: A Retrospective

Onur Mutlu[§‡] Jeremie S. Kim[‡§]
[§]ETH Zürich [‡]Carnegie Mellon University

Related Videos and Course Materials (I)

- **Undergraduate Digital Design & Computer Architecture Course Lecture Videos (2020, 2019, 2018, 2017, 2015, 2014, 2013)**
- **Undergraduate Digital Design & Computer Architecture Course Materials (2020, 2019, 2018, 2015, 2014, 2013)**

- **Graduate Computer Architecture Course Lecture Videos (2019, 2018, 2017, 2015, 2013)**
- **Graduate Computer Architecture Course Materials (2019, 2018, 2017, 2015, 2013)**

- **Parallel Computer Architecture Course Materials (Lecture Videos)**

Related Videos and Course Materials (II)

- Seminar in Computer Architecture Course Lecture Videos (Spring 2020, Fall 2019, Spring 2019, 2018)
- Seminar in Computer Architecture Course Materials (Spring 2020, Fall 2019, Spring 2019, 2018)

- Memory Systems Course Lecture Videos (Sept 2019, July 2019, June 2019, October 2018)
- Memory Systems Short Course Lecture Materials (Sept 2019, July 2019, June 2019, October 2018)
- ACACES Summer School Memory Systems Course Lecture Videos (2018, 2013)
- ACACES Summer School Memory Systems Course Materials (2018, 2013)

Some Open Source Tools (I)

- Rowhammer – Program to Induce RowHammer Errors
  - https://github.com/CMU-SAFARI/rowhammer
- Ramulator – Fast and Extensible DRAM Simulator
  - https://github.com/CMU-SAFARI/ramulator
- MemSim – Simple Memory Simulator
  - https://github.com/CMU-SAFARI/memsim
- NOCulator – Flexible Network-on-Chip Simulator
  - https://github.com/CMU-SAFARI/NOCulator
- SoftMC – FPGA-Based DRAM Testing Infrastructure
  - https://github.com/CMU-SAFARI/SoftMC

- Other open-source software from SAFARI
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

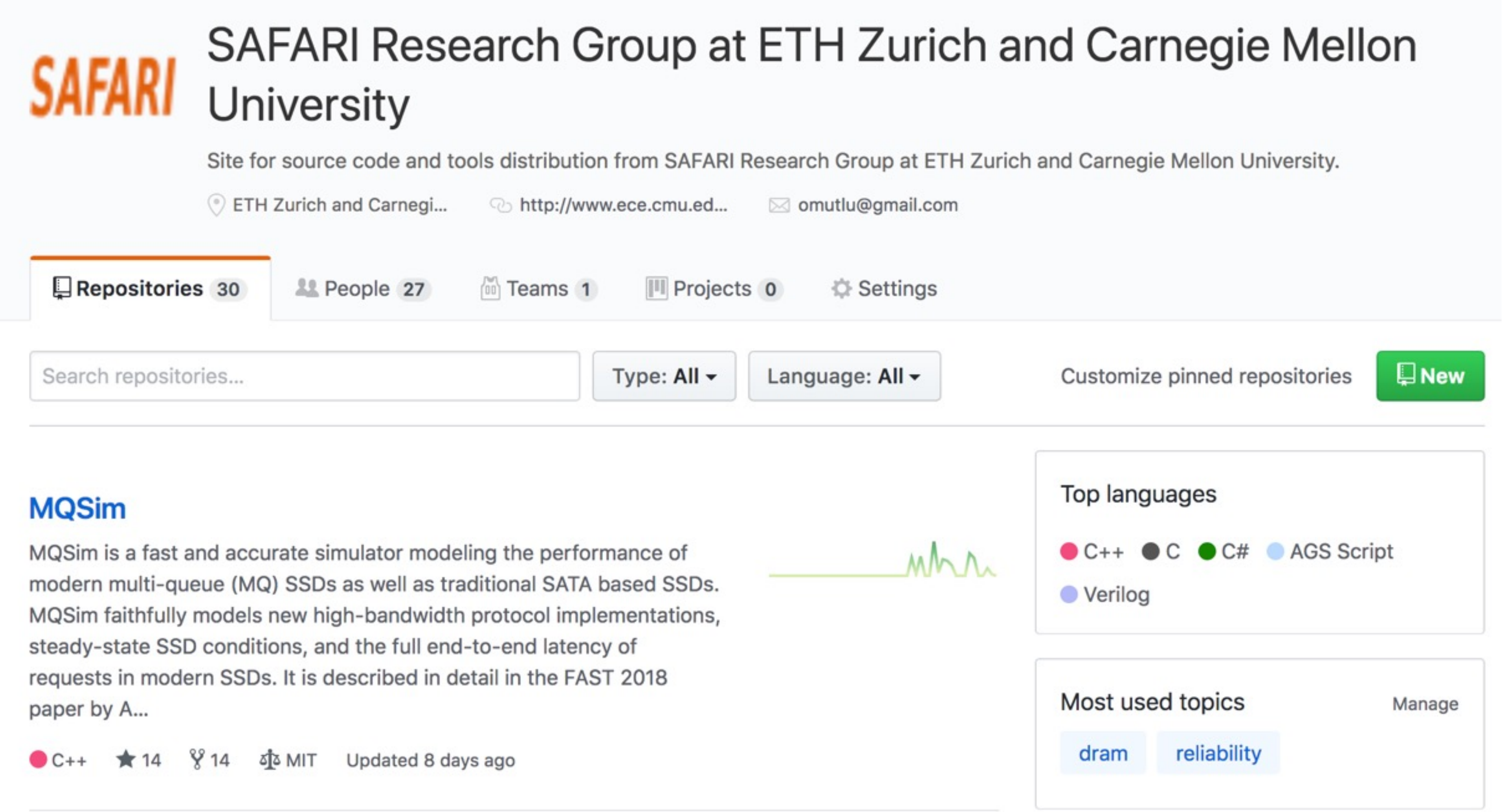
Some Open Source Tools (II)

- MQSim – A Fast Modern SSD Simulator
  - https://github.com/CMU-SAFARI/MQSim
- Mosaic – GPU Simulator Supporting Concurrent Applications
  - https://github.com/CMU-SAFARI/Mosaic
- IMPICA – Processing in 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/IMPICA
- SMLA – Detailed 3D-Stacked Memory Simulator
  - https://github.com/CMU-SAFARI/SMLA
- HWASim – Simulator for Heterogeneous CPU-HWA Systems
  - https://github.com/CMU-SAFARI/HWASim

- Other open-source software from SAFARI
  - **https://github.com/CMU-SAFARI/**
  - **http://www.ece.cmu.edu/~safari/tools.html**

More Open Source Tools (III)

- A lot more open-source software from SAFARI
  - https://github.com/CMU-SAFARI/
  - http://www.ece.cmu.edu/~safari/tools.html



SAFARI Research Group at ETH Zurich and Carnegie Mellon University

Site for source code and tools distribution from SAFARI Research Group at ETH Zurich and Carnegie Mellon University.

ETH Zurich and Carnegi... http://www.ece.cmu.ed... omutlu@gmail.com

MQSim

MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end latency of requests in modern SSDs. It is described in detail in the FAST 2018 paper by A...

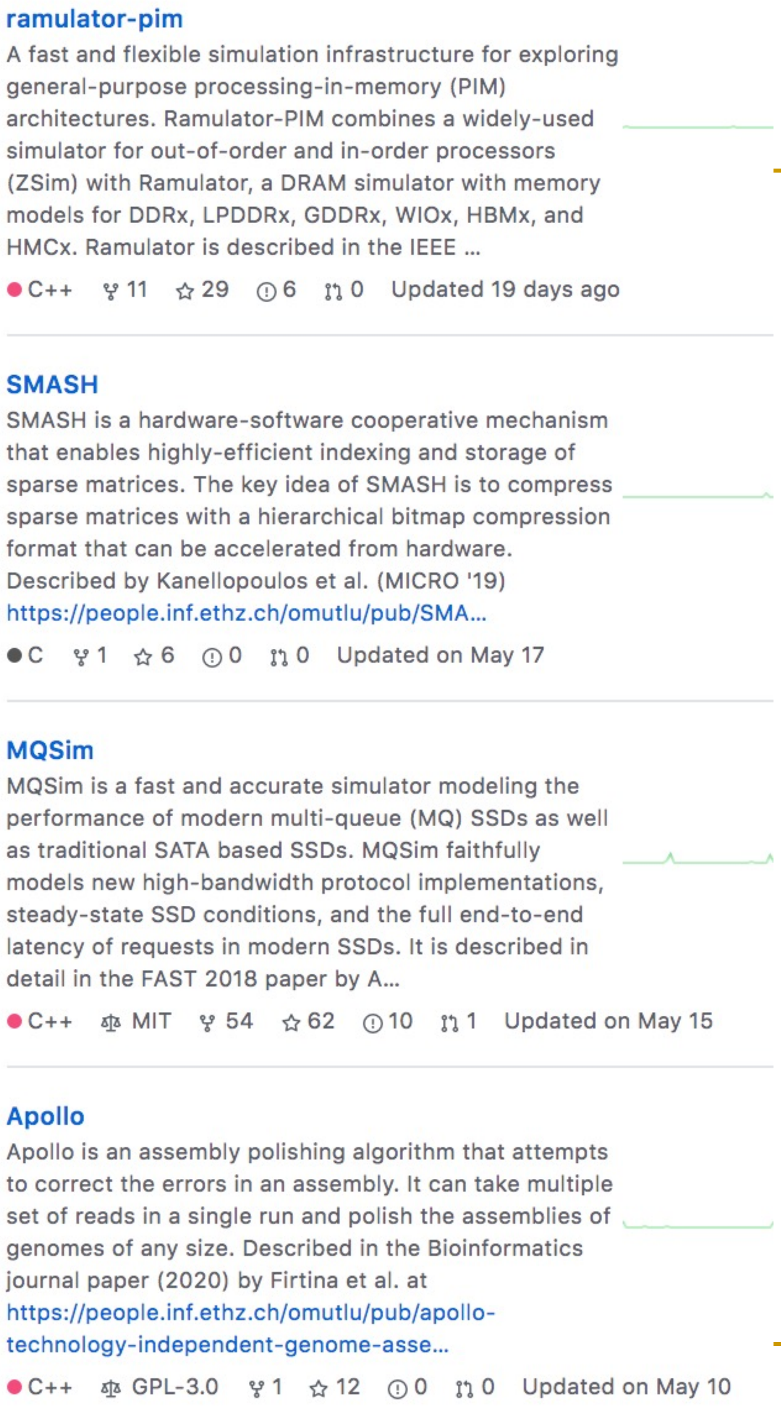
● C++ ★ 14 14 MIT Updated 8 days ago

Top languages

● C++ ● C ● C# ● AGS Script ● Verilog

Most used topics Manage

dram reliability



ramulator-pim

A fast and flexible simulation infrastructure for exploring general-purpose processing-in-memory (PIM) architectures. Ramulator-PIM combines a widely-used simulator for out-of-order and in-order processors (ZSim) with Ramulator, a DRAM simulator with memory models for DDRx, LPDDRx, GDDRx, WIOx, HBMx, and HMCx. Ramulator is described in the IEEE ...

● C++ ⑂ 11 ☆ 29 ⓘ 6 ⇅ 0 Updated 19 days ago

SMASH

SMASH is a hardware-software cooperative mechanism that enables highly-efficient indexing and storage of sparse matrices. The key idea of SMASH is to compress sparse matrices with a hierarchical bitmap compression format that can be accelerated from hardware. Described by Kanellopoulos et al. (MICRO '19) https://people.inf.ethz.ch/omutlu/pub/SMA...

● C ⑂ 1 ☆ 6 ⓘ 0 ⇅ 0 Updated on May 17

MQSim

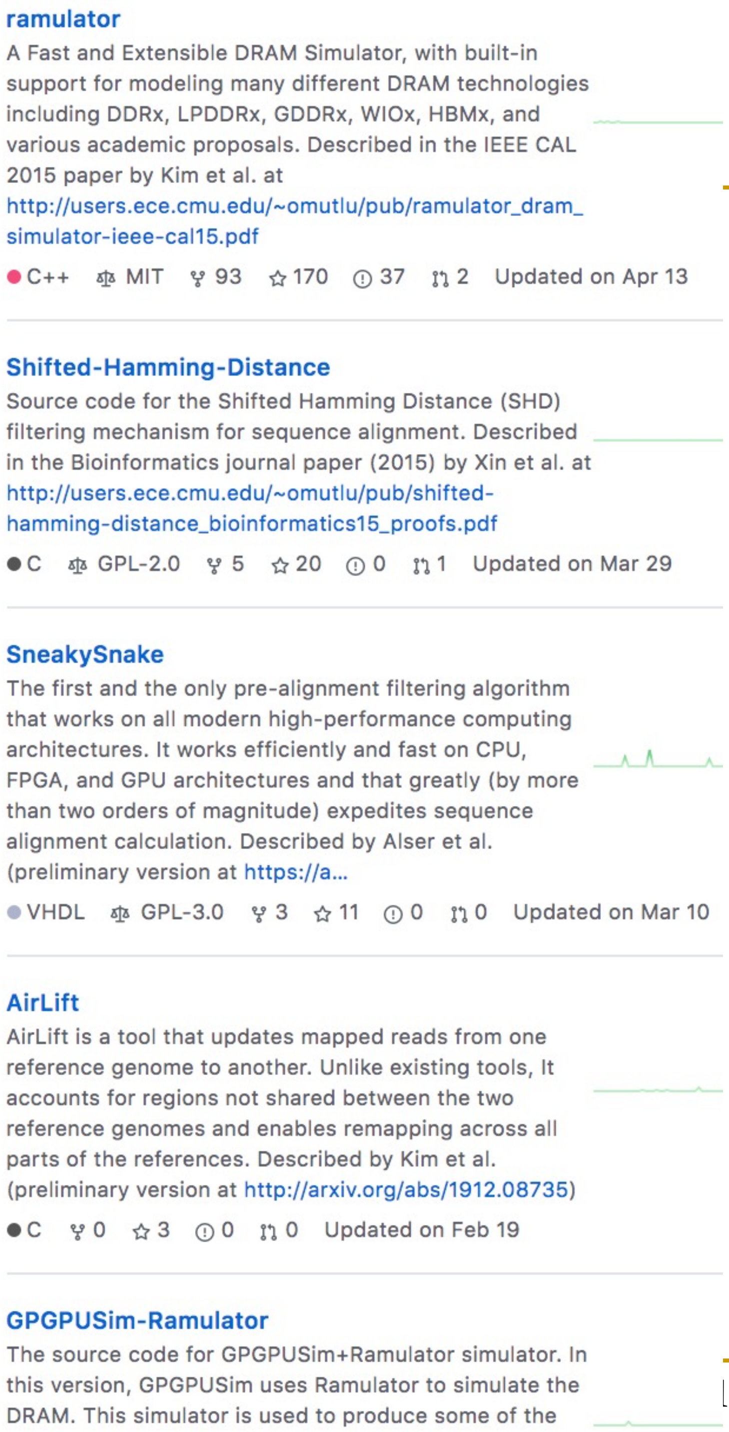
MQSim is a fast and accurate simulator modeling the performance of modern multi-queue (MQ) SSDs as well as traditional SATA based SSDs. MQSim faithfully models new high-bandwidth protocol implementations, steady-state SSD conditions, and the full end-to-end latency of requests in modern SSDs. It is described in detail in the FAST 2018 paper by A...

● C++ ⚖ MIT ⑂ 54 ☆ 62 ⓘ 10 ⇅ 1 Updated on May 15

Apollo

Apollo is an assembly polishing algorithm that attempts to correct the errors in an assembly. It can take multiple set of reads in a single run and polish the assemblies of genomes of any size. Described in the Bioinformatics journal paper (2020) by Firtina et al. at https://people.inf.ethz.ch/omutlu/pub/apollo-technology-independent-genome-asse...

● C++ ⚖ GPL-3.0 ⑂ 1 ☆ 12 ⓘ 0 ⇅ 0 Updated on May 10



ramulator

A Fast and Extensible DRAM Simulator, with built-in support for modeling many different DRAM technologies including DDRx, LPDDRx, GDDRx, WIOx, HBMx, and various academic proposals. Described in the IEEE CAL 2015 paper by Kim et al. at http://users.ece.cmu.edu/~omutlu/pub/ramulator_dram_simulator-ieee-cal15.pdf

● C++ ⚖ MIT ⑂ 93 ☆ 170 ⓘ 37 ⇅ 2 Updated on Apr 13

Shifted-Hamming-Distance

Source code for the Shifted Hamming Distance (SHD) filtering mechanism for sequence alignment. Described in the Bioinformatics journal paper (2015) by Xin et al. at http://users.ece.cmu.edu/~omutlu/pub/shifted-hamming-distance_bioinformatics15_proofs.pdf

● C ⚖ GPL-2.0 ⑂ 5 ☆ 20 ⓘ 0 ⇅ 1 Updated on Mar 29

SneakySnake

The first and the only pre-alignment filtering algorithm that works on all modern high-performance computing architectures. It works efficiently and fast on CPU, FPGA, and GPU architectures and that greatly (by more than two orders of magnitude) expedites sequence alignment calculation. Described by Alser et al. (preliminary version at https://a...

● VHDL ⚖ GPL-3.0 ⑂ 3 ☆ 11 ⓘ 0 ⇅ 0 Updated on Mar 10

AirLift

AirLift is a tool that updates mapped reads from one reference genome to another. Unlike existing tools, It accounts for regions not shared between the two reference genomes and enables remapping across all parts of the references. Described by Kim et al. (preliminary version at http://arxiv.org/abs/1912.08735)

● C ⑂ 0 ☆ 3 ⓘ 0 ⇅ 0 Updated on Feb 19

GPGPUSim-Ramulator

The source code for GPGPUSim+Ramulator simulator. In this version, GPGPUSim uses Ramulator to simulate the DRAM. This simulator is used to produce some of the

Referenced Papers, Talks, Artifacts

- All are available at

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

  http://scholar.google.com/citations?user=7XyGUGkAAAAJ&hl=en

  https://www.youtube.com/onurmutlulectures

  https://github.com/CMU-SAFARI/

An Interview on Research and Education

- Computing Research and Education (@ ISCA 2019)
  - https://www.youtube.com/watch?v=8ffSEKZhmvo&list=PL5Q2soXY2Zi_4oP9LdL3cc8G6NIjD2Ydz

- Maurice Wilkes Award Speech (10 minutes)
  - https://www.youtube.com/watch?v=tcQ3zZ3JpuA&list=PL5Q2soXY2Zi8D_5MGV6EnXEJHnV2YFBJl&index=15

More Thoughts and Suggestions

- Onur Mutlu,
  "Some Reflections (on DRAM)"
  *Award Speech for ACM SIGARCH Maurice Wilkes Award, at the **ISCA** Awards Ceremony*, Phoenix, AZ, USA, 25 June 2019.
  [Slides (pptx) (pdf)]
  [Video of Award Acceptance Speech (Youtube; 10 minutes) (Youku; 13 minutes)]
  [Video of Interview after Award Acceptance (Youtube; 1 hour 6 minutes) (Youku; 1 hour 6 minutes)]
  [News Article on "ACM SIGARCH Maurice Wilkes Award goes to Prof. Onur Mutlu"]

- Onur Mutlu,
  "How to Build an Impactful Research Group"
  *57th Design Automation Conference Early Career Workshop (**DAC**)*, Virtual, 19 July 2020.
  [Slides (pptx) (pdf)]

End of Backup Slides