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

Can Firtina
canfirtina@gmail.com

27 May 2022
Invited Seminar Talk at UC Berkeley

SAFARI ETH zürich

Brief Self Introduction

- Ph.D. Student in the SAFARI Research Group

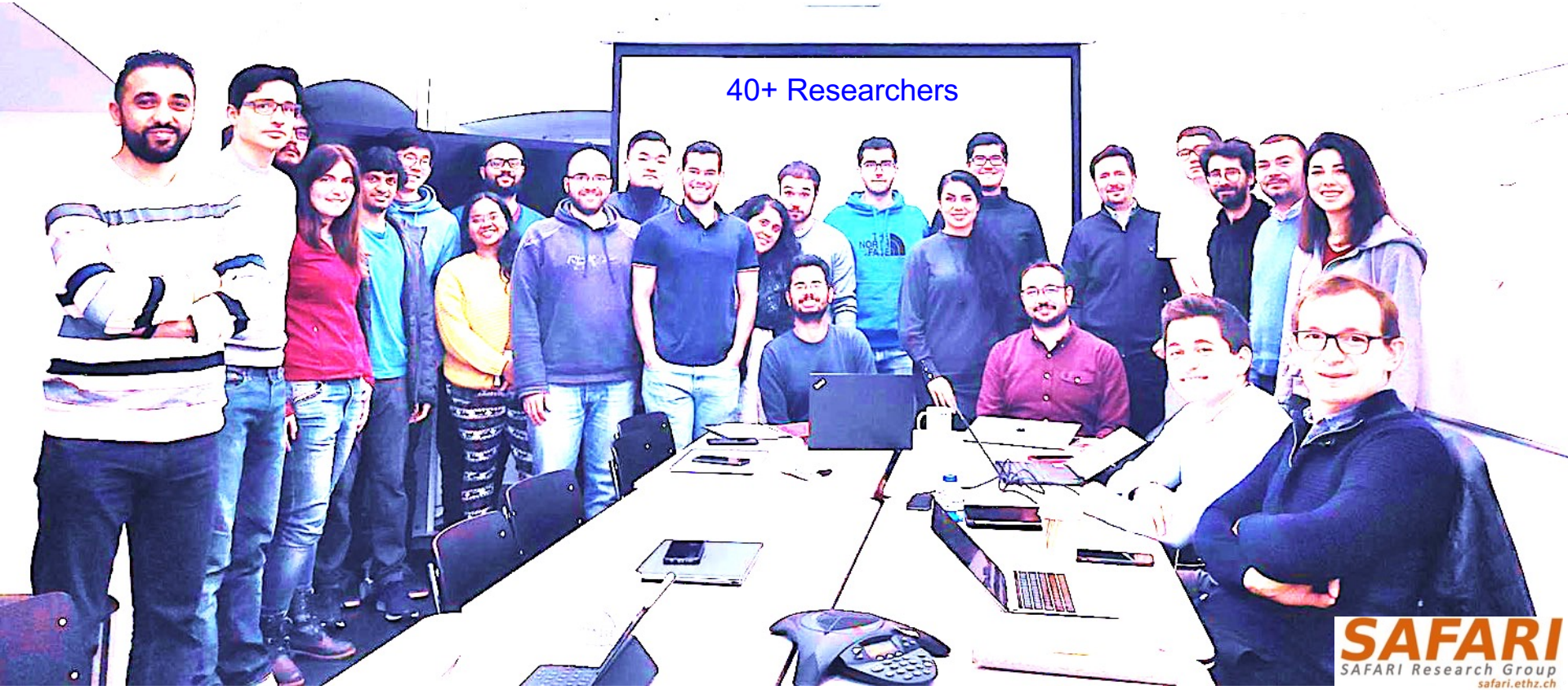
- Research interests:
  - Computational biology, accelerating genome analysis, genome editing

- Some selected works:
  - C. Firtina+, "ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis," *arXiv*, May 2022.
  - C. Firtina+, "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches," *arXiv*, December 2021.
  - Jeremie S. Kim, C. Firtina+ "AirLift: A Fast and Comprehensive Technique for Translating Alignments between Reference Genomes," *bioRxiv*, Feb. 2021.
  - C. Firtina+, "Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm," *Bioinformatics*, Jun. 2020.

- Get to know us and our research
  - https://safari.ethz.ch/
  - Contact me: canfirtina@gmail.com. Personal website: https://cfirtina.com

Onur Mutlu's SAFARI Research Group

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



Think BIG, Aim HIGH!

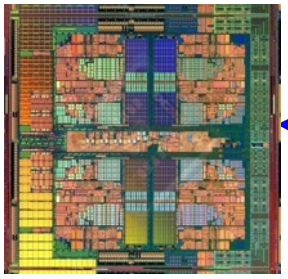
https://safari.ethz.ch

Professor Mutlu

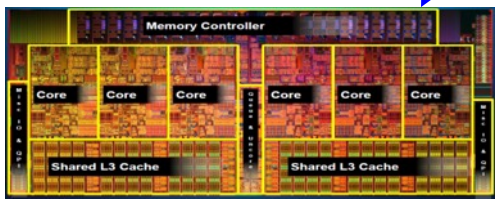
- Onur Mutlu
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-…
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - https://people.inf.ethz.ch/omutlu/
  - omutlu@gmail.com (Best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

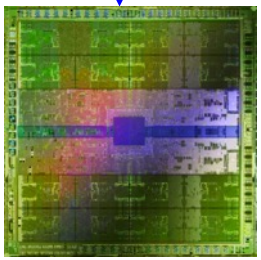
Computer architecture, HW/SW, systems, bioinformatics, security



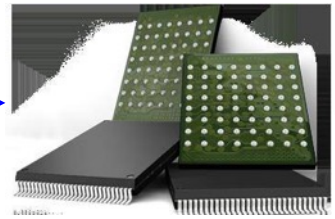
Heterogeneous Processors and Accelerators



Hybrid Main Memory



Graphics and Vision Processing



Persistent Memory/Storage

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

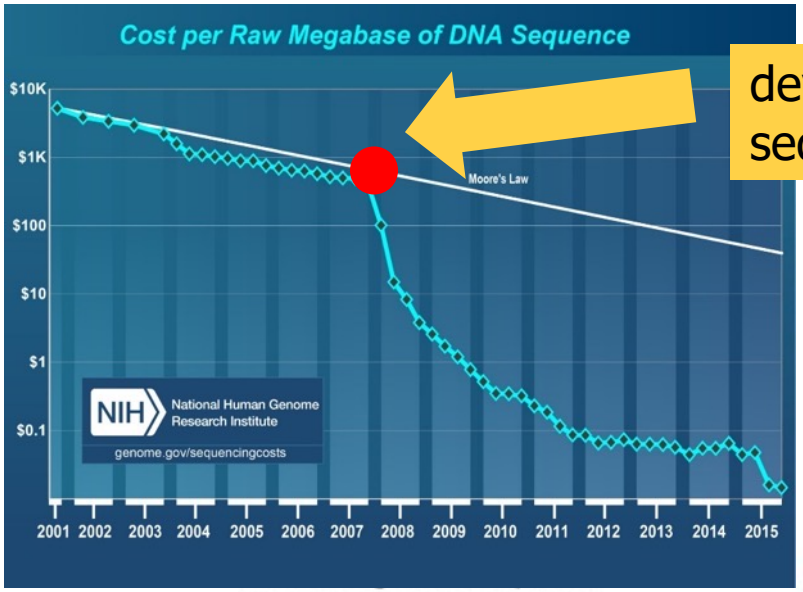
Computing

is Bottlenecked by Data

Data is Key for AI, ML, Genomics, …

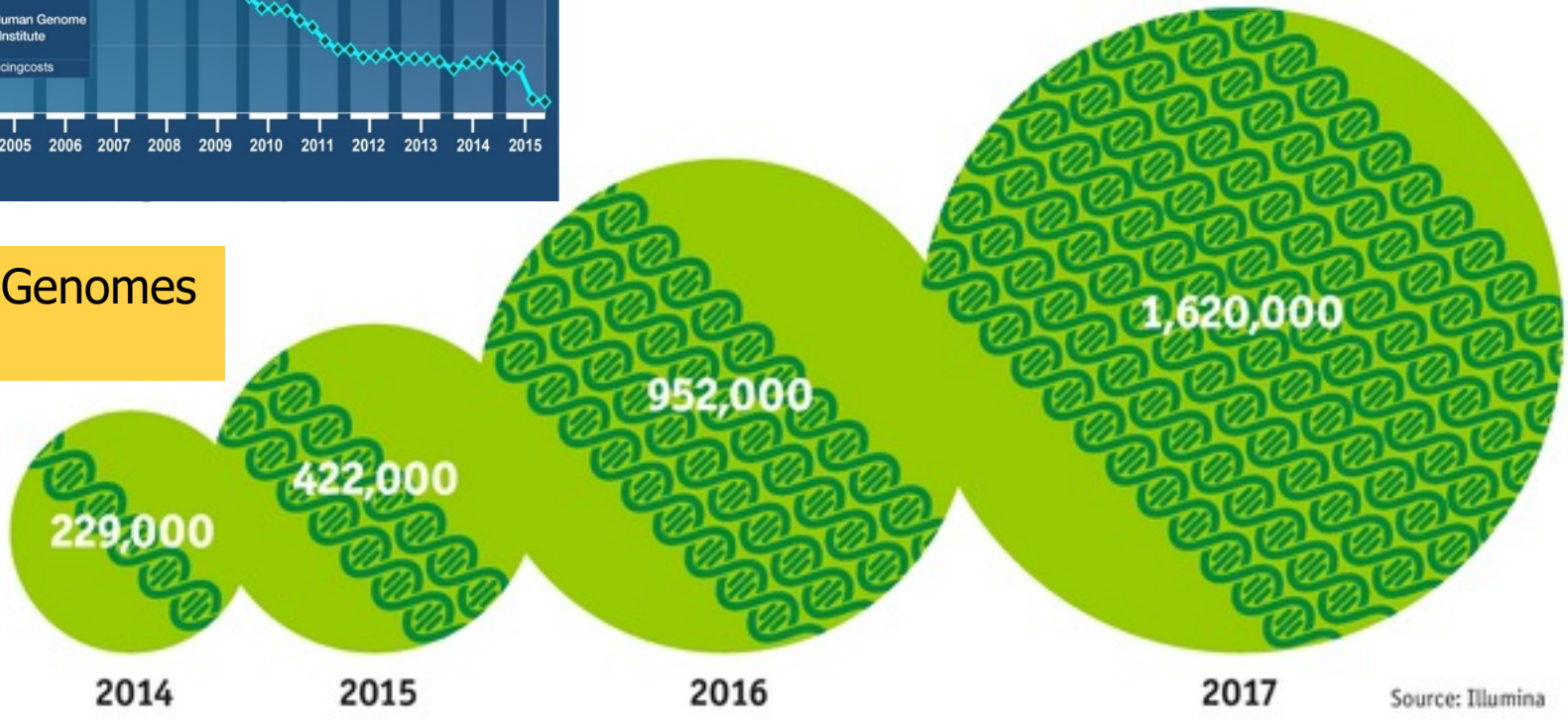
- Important workloads are all data intensive
- They require rapid and efficient processing of large amounts of data
- Data is increasing
  - We can generate more than we can process

Data is Key for Future Workloads



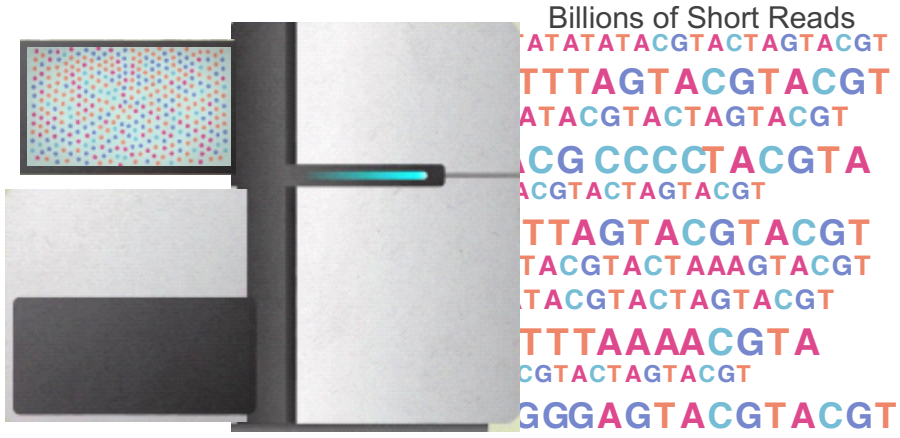
development of high-throughput sequencing (HTS) technologies

Number of Genomes Sequenced

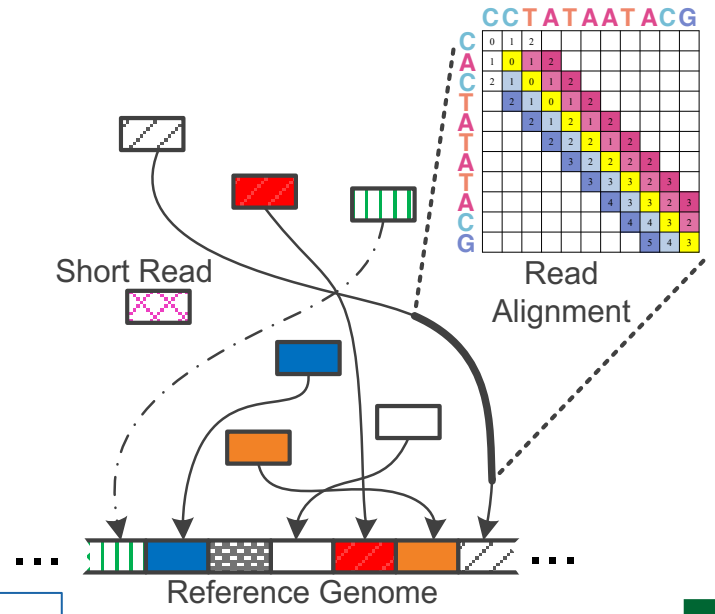


The Economist

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



1 Sequencing



Read Mapping 2

Genome Analysis

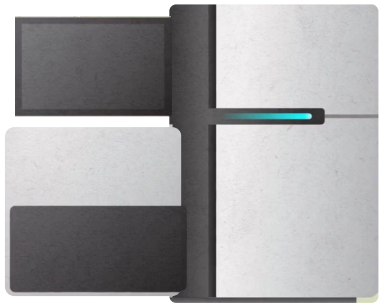
Data → performance & energy bottleneck

```
read4:          CGCTTCCAT
read5:            CCATGACGC
read6:          TTCCATGAC
```

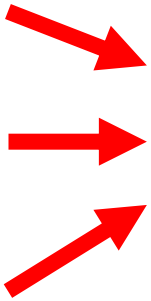
3 Variant Calling

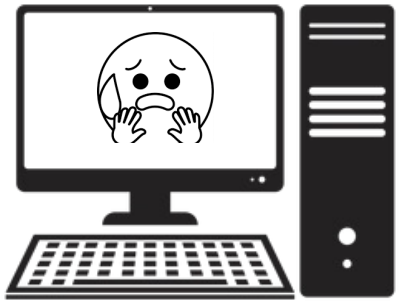
Scientific Discovery 4

Lack of Specialized Compute Capability



Specialized Machine
for Sequencing





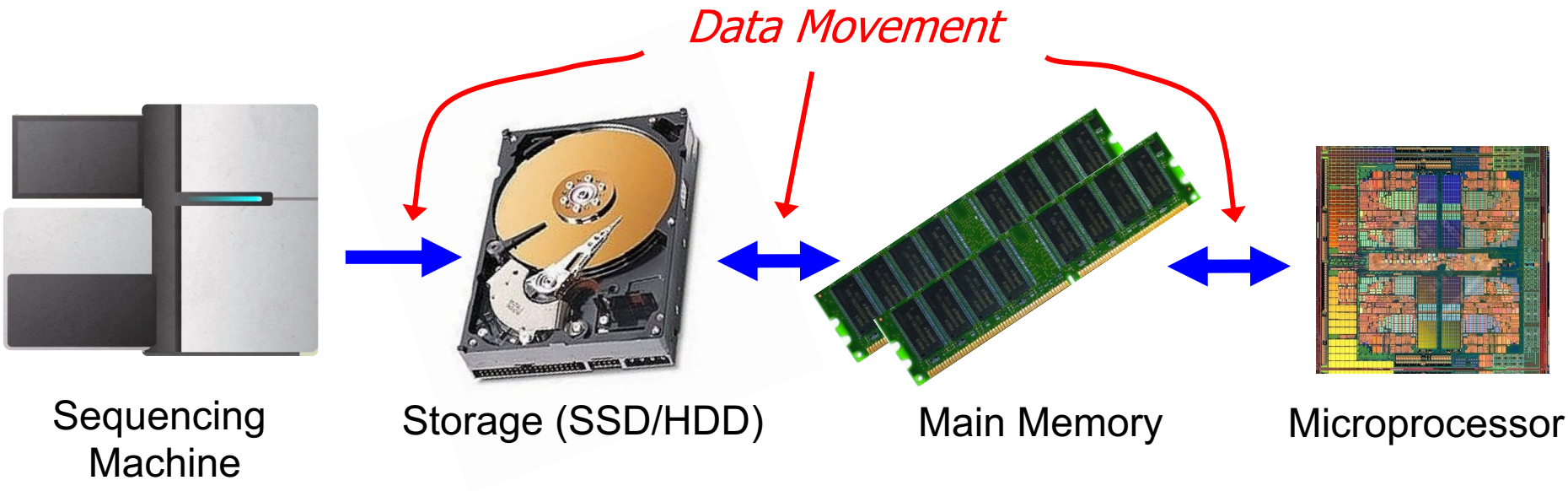
General-Purpose Machine
for Analysis

FAST

SLOW

Data Movement Dominates Performance

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



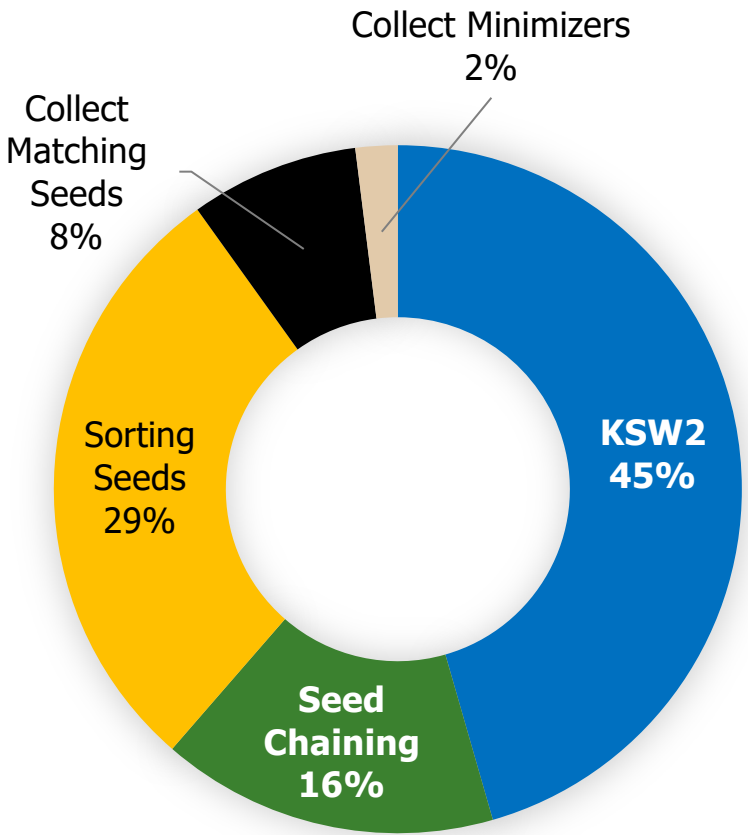
Single memory request consumes >160x-800x more energy compared to performing an addition operation

* 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%

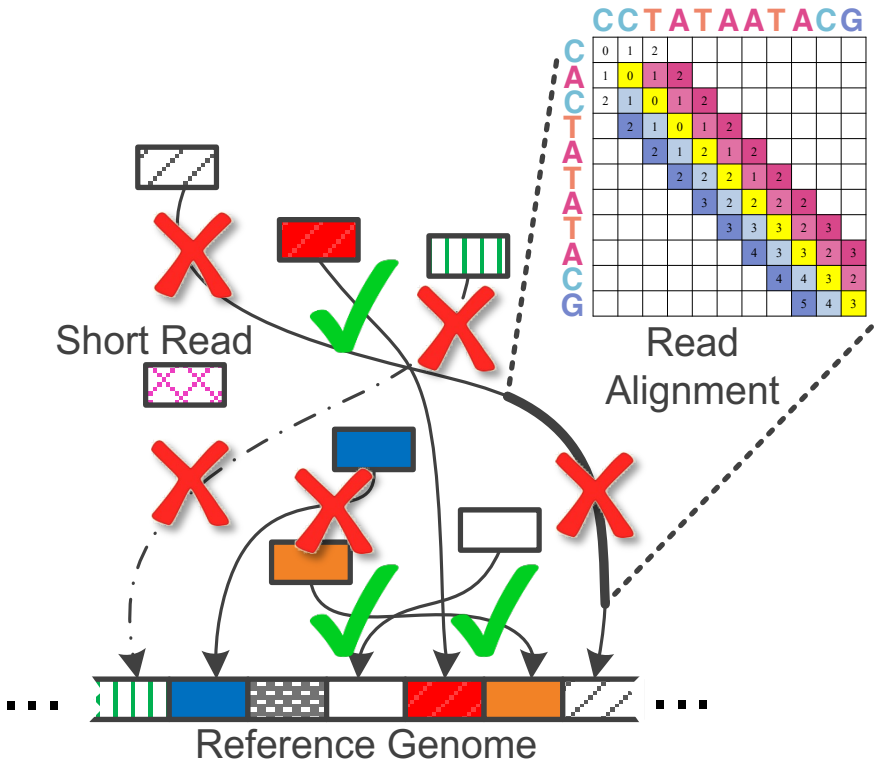
of the read mapper's execution time is spent in sequence alignment



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



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** ▾

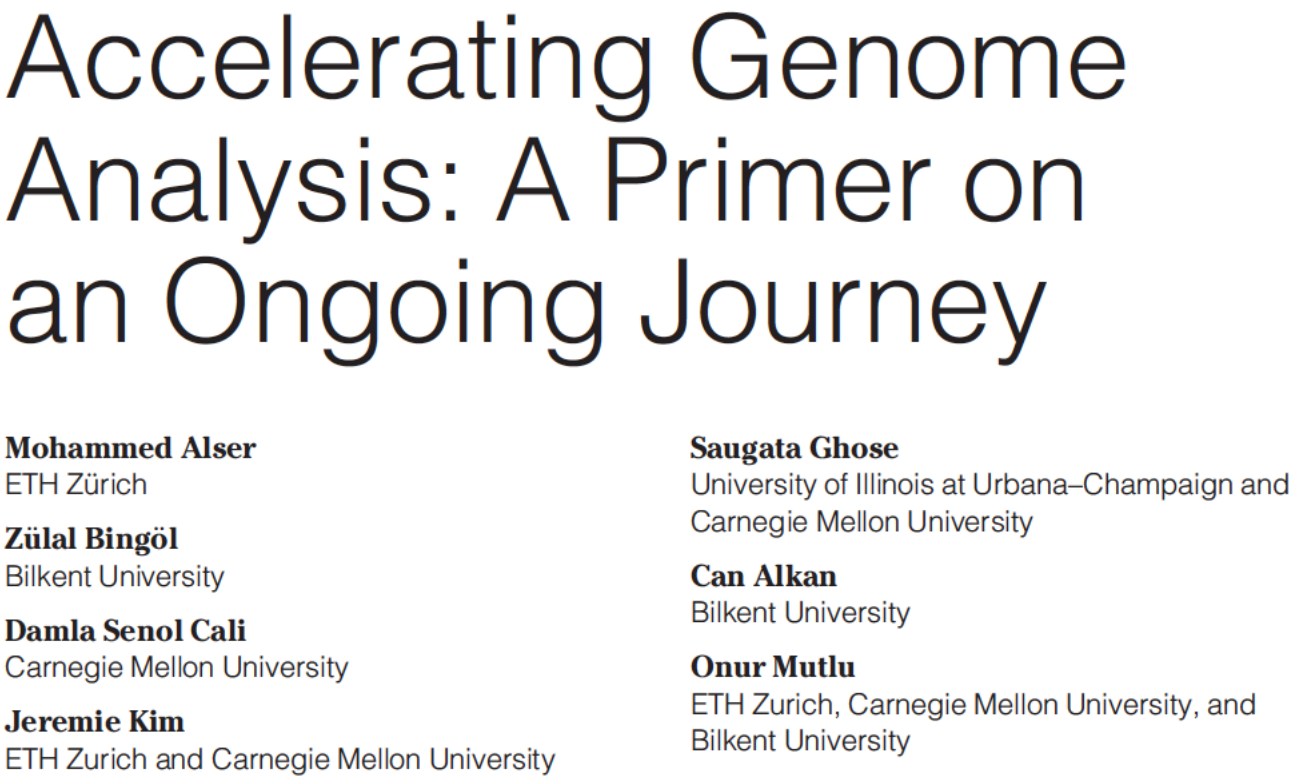
Oxford Nanopore MinION

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)]



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*

New Applications: Genome Graphs

SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping

Damla Senol Cali[1] Konstantinos Kanellopoulos[2] Joël Lindegger[2] Zülal Bingöl[3]
Gurpreet S. Kalsi[4] Ziyi Zuo[5] Can Firtina[2] Meryem Banu Cavlak[2] Jeremie Kim[2]
Nika Mansouri Ghiasi[2] Gagandeep Singh[2] Juan Gómez-Luna[2] Nour Almadhoun Alserr[2]
Mohammed Alser[2] Sreenivas Subramoney[4] Can Alkan[3] Saugata Ghose[6] Onur Mutlu[2]

[1]Bionano Genomics [2]ETH Zürich [3]Bilkent University [4]Intel Labs
[5]Carnegie Mellon University [6]University of Illinois Urbana-Champaign

In-Storage Genome Filtering [ASPLOS 2022]

- Nika Mansouri Ghiasi, Jisung Park, Harun Mustafa, Jeremie Kim, Ataberk Olgun, Arvid Gollwitzer, Damla Senol Cali, Can Firtina, Haiyu Mao, Nour Almadhoun Alserr, Rachata Ausavarungnirun, Nandita Vijaykumar, Mohammed Alser, and Onur Mutlu,
  "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis"
  *Proceedings of the 27th International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Virtual, February-March 2022.
  [Lightning Talk Slides (pptx) (pdf)]
  [Lightning Talk Video (90 seconds)]

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

Nika Mansouri Ghiasi[1] Jisung Park[1] Harun Mustafa[1] Jeremie Kim[1] Ataberk Olgun[1]
Arvid Gollwitzer[1] Damla Senol Cali[2] Can Firtina[1] Haiyu Mao[1] Nour Almadhoun Alserr[1]
Rachata Ausavarungnirun[3] Nandita Vijaykumar[4] Mohammed Alser[1] Onur Mutlu[1]

[1]ETH Zürich [2]Bionano Genomics [3]KMUTNB [4]University of Toronto

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)]

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

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]

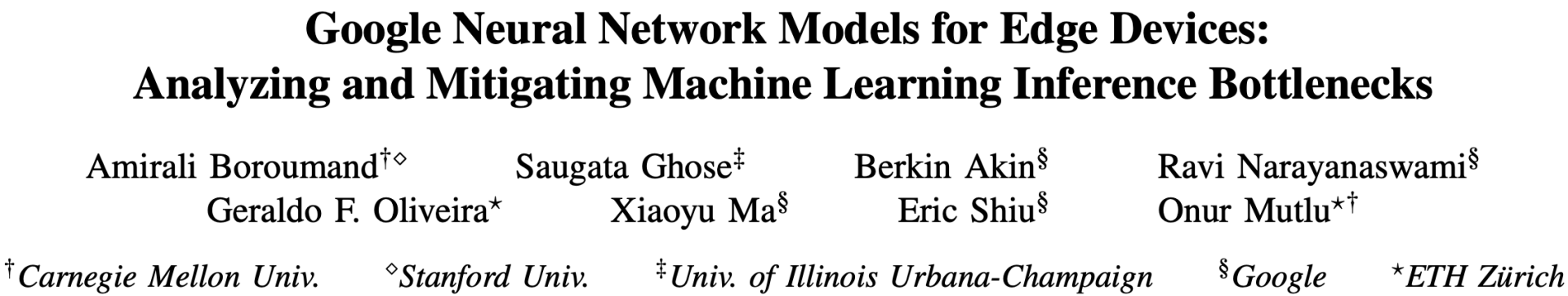
SAFARI Research Group

[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

- BLEND
  - Finding approximate (fuzzy) seed matches with a single lookup
- AirLift
  - Avoiding redundant computations when moving from one reference genome to another
- Apollo & ApHMM
  - Error correction using profile Hidden Markov Models (pHMMs) and accelerating pHMMs

Agenda for Today - Intelligent Algorithms

- BLEND
  - Finding approximate (fuzzy) seed matches with a single lookup
- AirLift
  - Avoiding redundant computations when moving from one reference genome to another
- Apollo & ApHMM
  - Error correction using profile Hidden Markov Models (pHMMs) and accelerating pHMMs

Finding Approximate Seed Matches

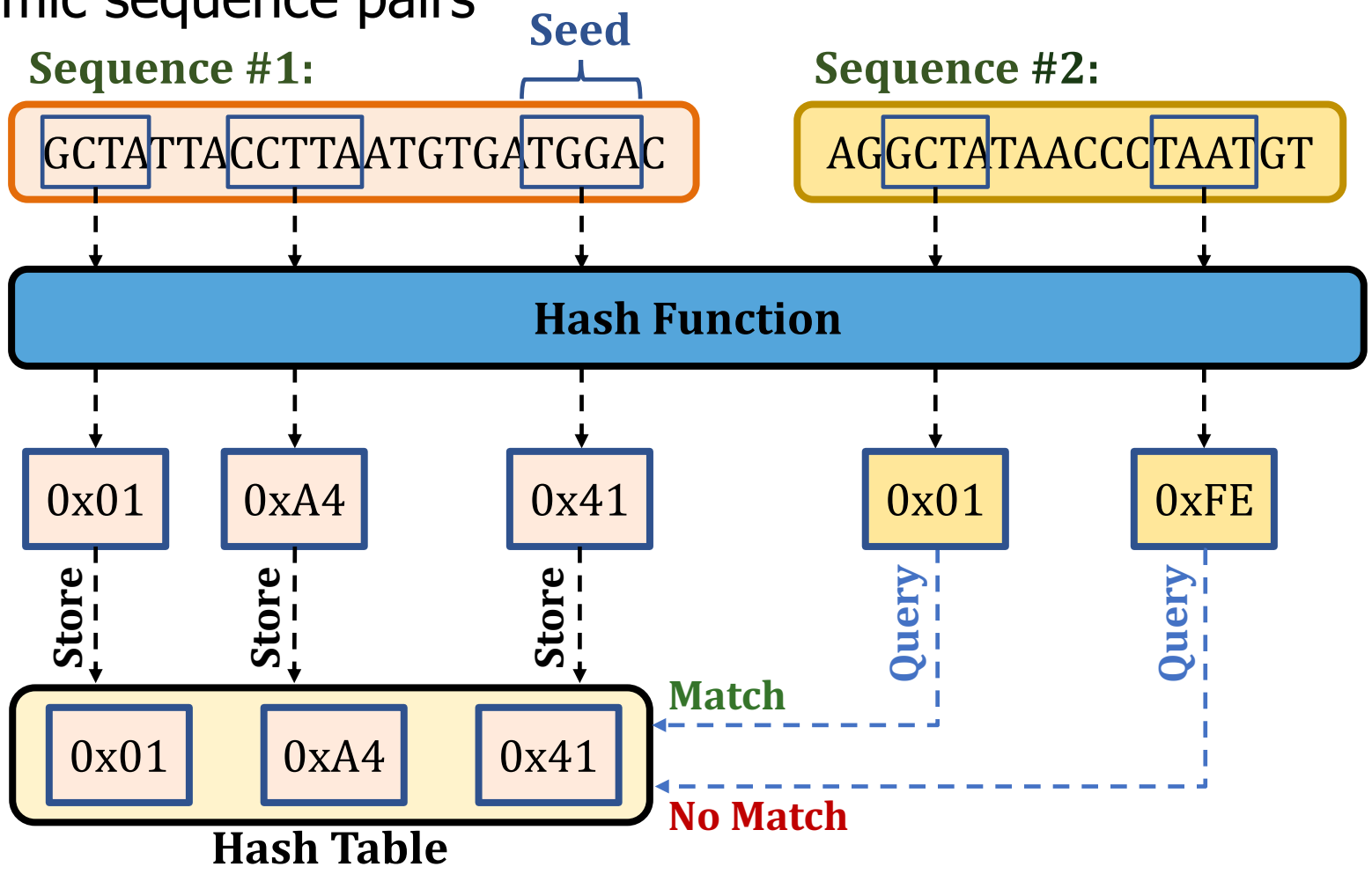
- Can Firtina, Jisung Park, Mohammed Alser, Jeremie S. Kim, Damla Senol Cali, Taha Shahroodi, Nika Mansouri-Ghiasi, Gagandeep Singh, Konstantinos Kanellopoulos, Can Alkan, and Onur Mutlu,
  "BLEND: A Fast, Memory-Efficient, and Accurate Mechanism to Find Fuzzy Seed Matches"
  *Preprint in **arXiv***, 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] Mohammed Alser[1] Jeremie S. Kim[1] Damla Senol Cali[2]
Taha Shahroodi[3] Nika Mansouri-Ghiasi[1] Gagandeep Singh[1] Konstantinos Kanellopoulos[1]
Can Alkan[4] Onur Mutlu[1]
[1]*ETH Zurich* [2]*Bionano Genomics* [3]*TU Delft* [4]*Bilkent University*

Background – Seeds

Goal: Quickly identify matching subsequences between genomic sequence pairs



Background – Matching Seeds

- **Hash tables** enable finding seed matches with a single lookup of hash values
  - **Finds exact-matching seeds** based on their hash values

- There are many seeding techniques
  - **Minimizers:** Sample k-mers based on their hash values
  - **Spaced seeds:** Ignore characters at certain positions of k-mers to tolerate substitutions between seeds
  - **Linked k-mers (e.g., strobemers):** Ignore gaps between sampled k-mers to tolerate both indels and substitutions

Problem

- All of the seeding approaches use hash functions with **low collision rates**
  - **Seeds must exactly match** to find seed matches using their hash values
  - **Dissimilar** seeds have **different hash values**
  - Highly **similar** seeds have **different hash values**

- The **length** and **number of seed matches → sensitivity**, **performance**, and **memory footprint**
  - **Exact-matching requirements** introduce challenges for determining these parameters

- Existing attempts optimizing seed matches **suffer from** either
  - **Increasing the use of the costly sequence alignment step** due to many seed matches
  - **Limited sensitivity** when finding seed matches

Goal

- **Enable finding fuzzy (i.e., approximate) matches** of seeds with a **single lookup of hash values** of these seeds

- Provide a mechanism for any seeding technique to **generate the same hash values for highly similar seeds**

- To this end, we propose **BLEND, a fast, memory-efficient, and accurate mechanism that can find fuzzy seed matches**

Key Ideas

- Exploit the key characteristics of the **SimHash technique**
  - **Same hash value for highly similar seeds** without imposing exact matches of seeds
  - Fuzzy seed matches with a **single lookup** while providing **window guarantees for minimizers**
  - **Increased the collision rate for similar seeds** while **preserving the low collision rate for dissimilar seeds**

- A **mechanism for converting seeds into sets of k-mers** to use these sets with the SimHash technique

The SimHash Technique

- Used for detecting near-duplicates mainly for web crawling by Google

- **Insight:** Two sets are similar if their content is highly similar
- **Goal:** Use the insight to enable generating the same hash value for highly similar sets
- **Input:** Hash values of a set of items (e.g., words in a sentence)
- **Output:** A hash value for the entire set (e.g., hash value of the sentence)

The SimHash Technique - Example

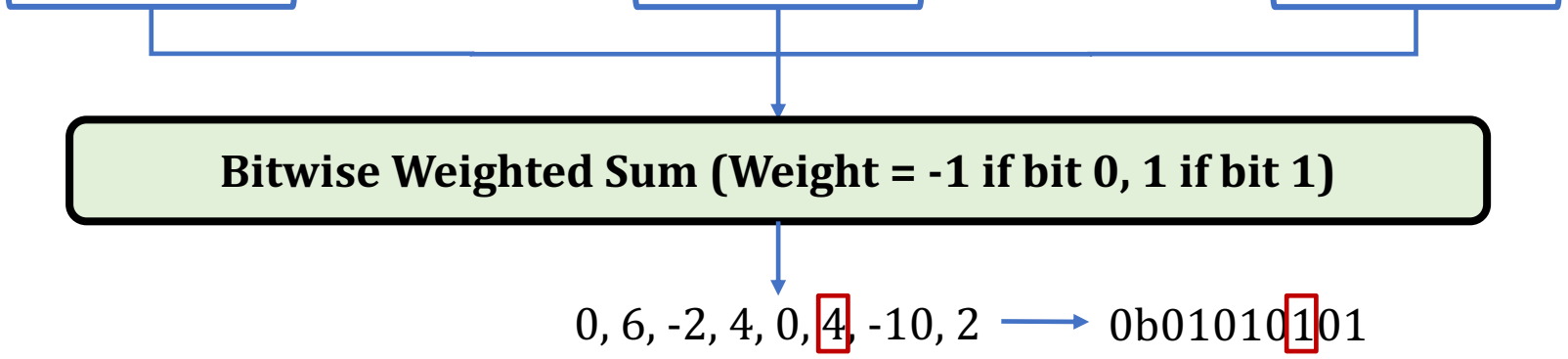
- **Input set:** A sentence
  - **Items**: Words

This is an example sentence to generate a hash value with SimHash

Word	Hash Value
This	0b01110100
is	0b11011101
an	0b00011100
example	0b01000001

Word	Hash Value
sentence	0b11110000
to	0b01111101
generate	0b10011100
a	0b11000001

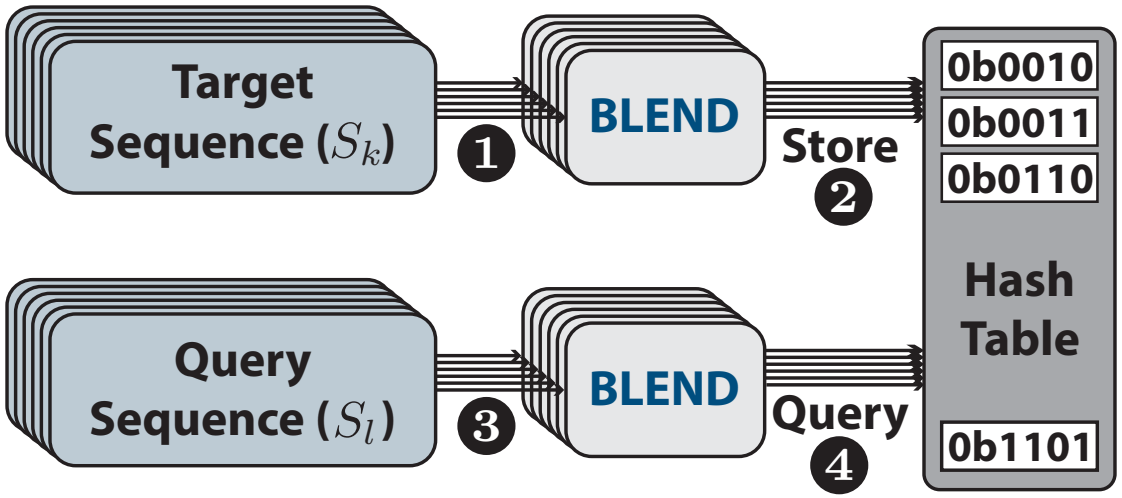
Word	Hash Value
hash	0b11111101
value	0b00100001
with	0b01010101
SimHash	0b11001110



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)

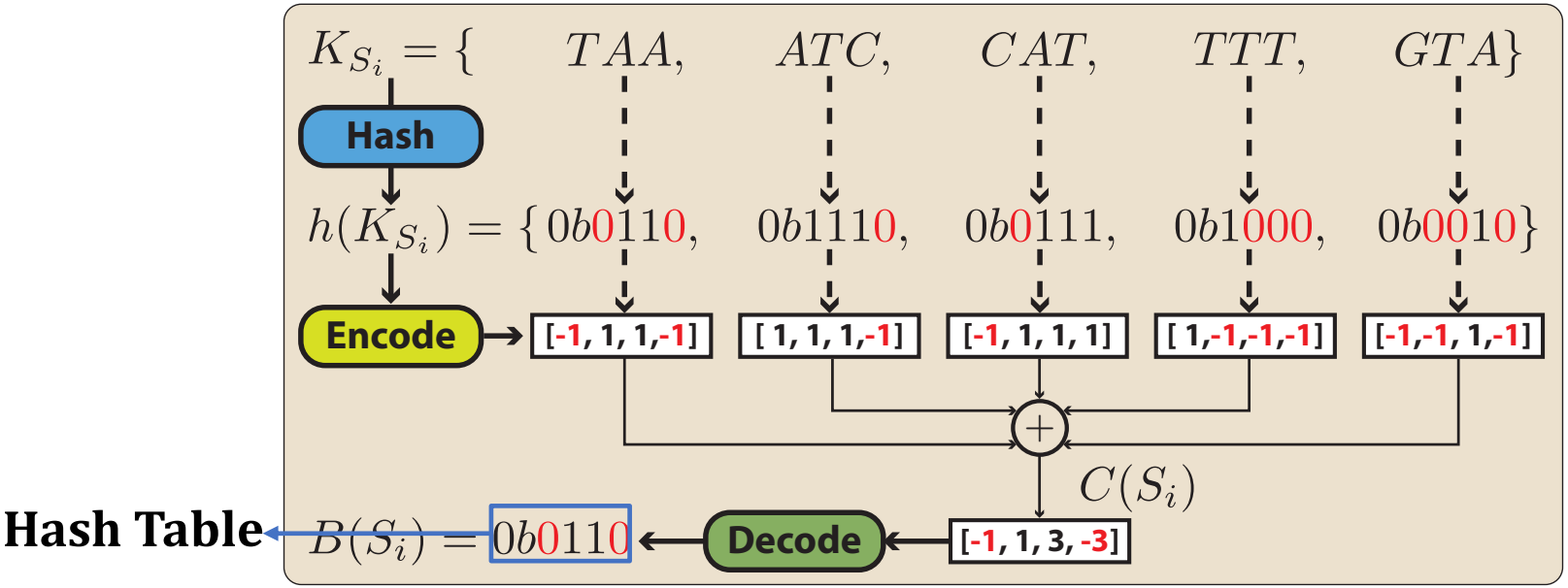
- BLEND can generate the **same hash value** for highly similar seeds



Walkthrough – Generating Hash Values

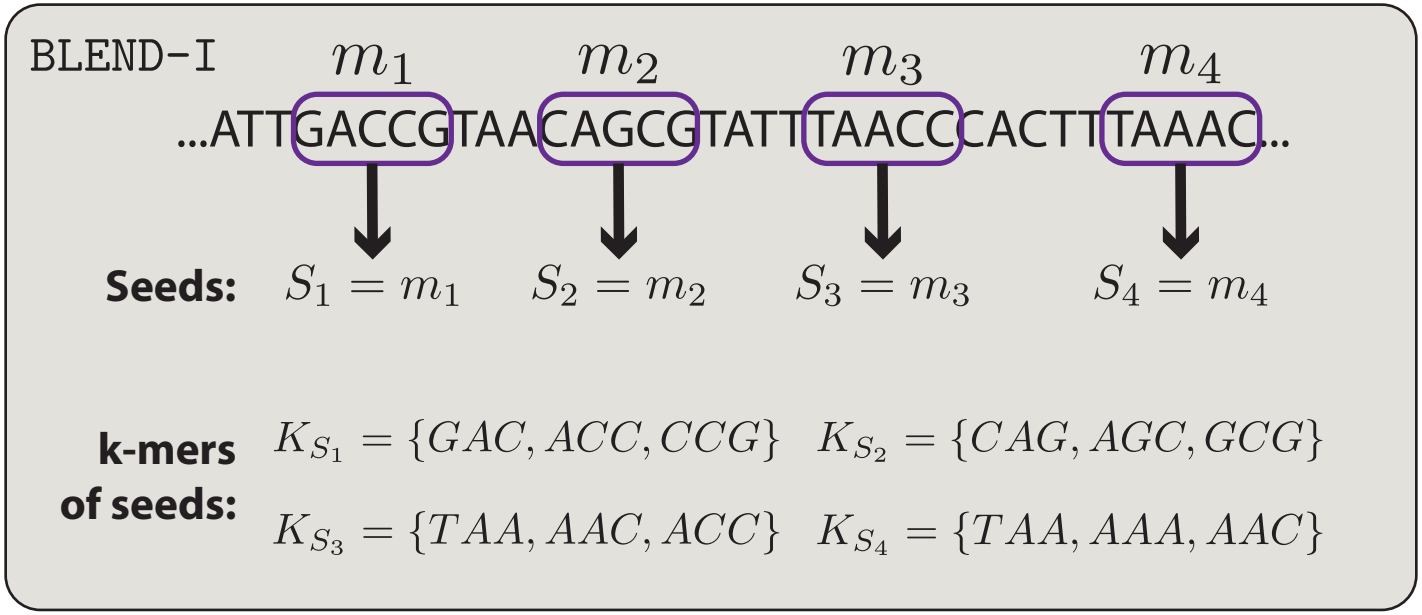
- **Input set:** A seed
  - **Items:** Selected substrings (k-mers) of the seed

- **Goal:** Generate the hash value of the seed using the SimHash technique



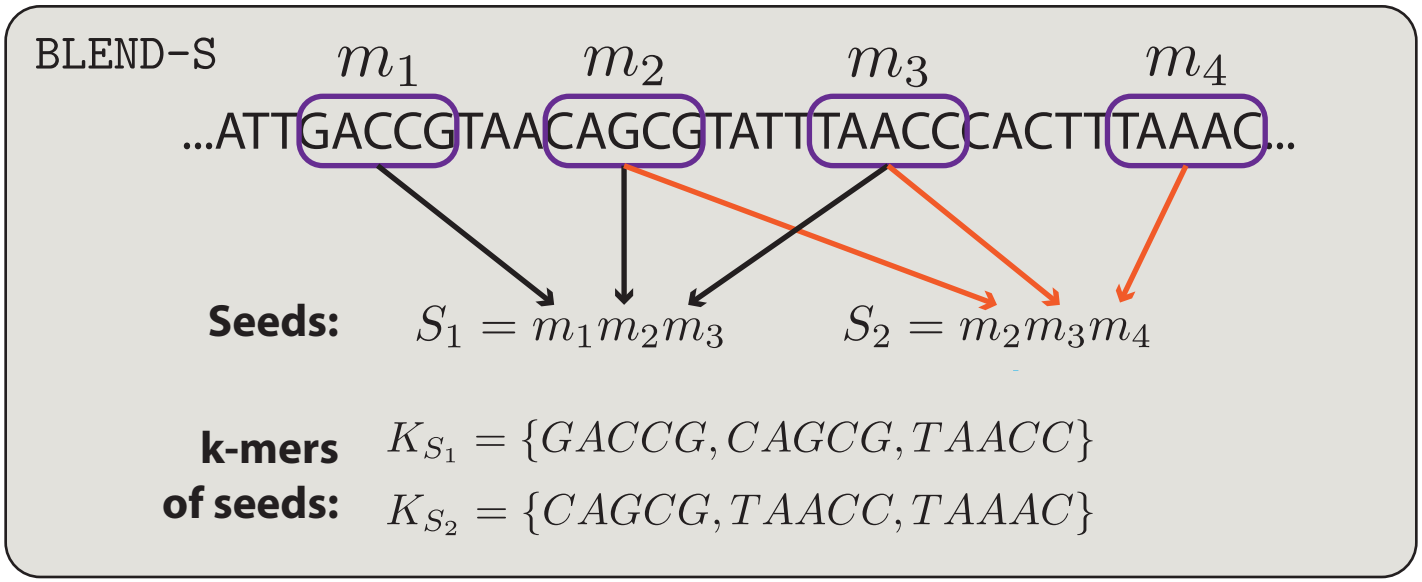
Walkthrough – Selecting k-mers from Seeds

- ***BLEND-I***
  - Include **all overlapping k-mers** of a seed in the set for hashing
- More sensitive to substitutions



Walkthrough – Selecting k-mers from Seeds

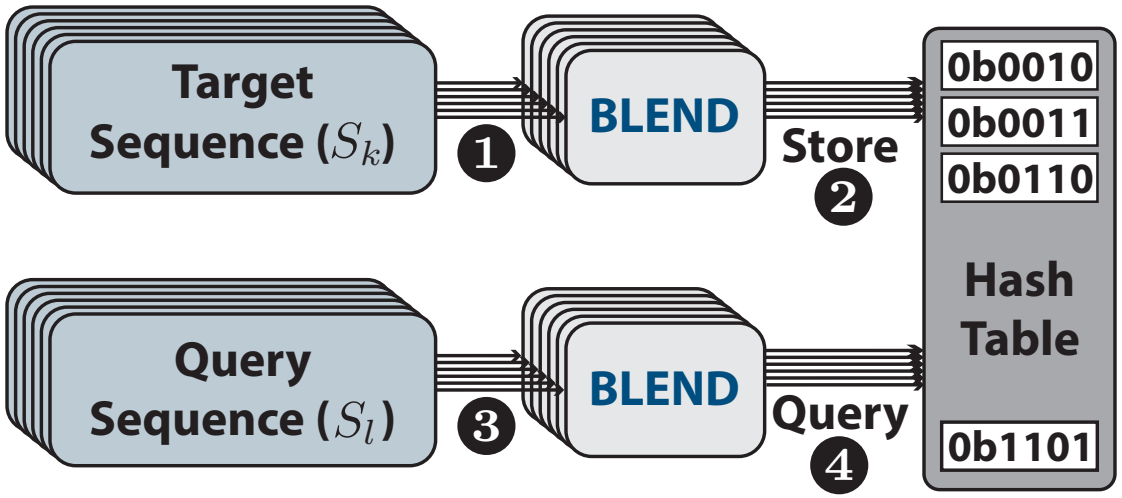
- ***BLEND-S***
  1. Find minimizers using regular hash functions
  2. Link every n consecutive minimizers to generate a seed
  3. Include **all linked k-mers** in the set for hashing
- Higher tolerance for indels and substitutions



Finding Fuzzy Seed Matches

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

- Single lookup of hash values from hash table return both exact and fuzzy seed matches



Evaluation Methodology

- We integrate the BLEND mechanism into Minimap2
- Real and simulated datasets

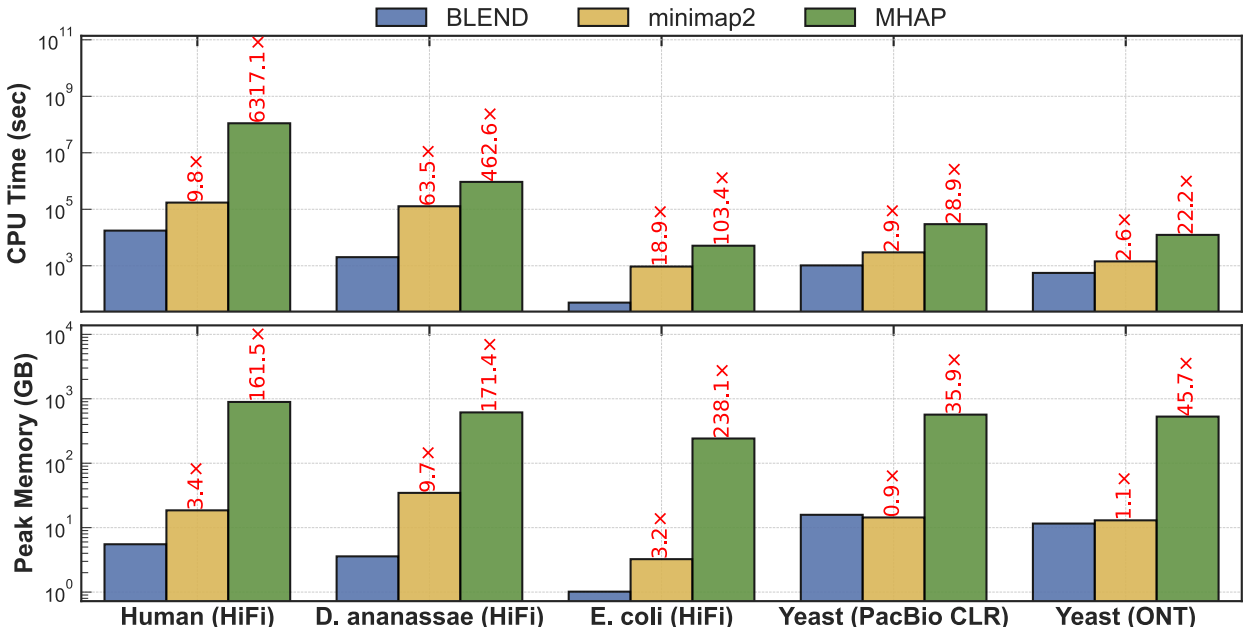
Organism	Library	Reads (#)	Seq. Depth	SRA Accession	Reference Genome
Human CHM13	PacBio HiFi	3,167,477	16	SRR11292122-3	GCA_009914755.3
D. ananassae	PacBio HiFi	1,195,370	50	SRR11442117	[60]
E. coli	PacBio HiFi	38,703	100	SRR11434954	[60]
Yeast	PacBio CLR*	270,849	200	Simulated P6-C4	GCA_000146045.2
	Oxford Nanopore Tech.*	135,296	100	Simulated R9.5	GCA_000146045.2
	Illumina MiSeq	3,318,467	80	ERR1938683	

* We use PBSIM2 to generate the simulated PacBio and ONT reads from the Yeast genome.
We include the simulated chemistry under SRA Accession.

- Use Cases
  - Read overlapping
  - Read mapping

Read Overlapping – Computational Resources

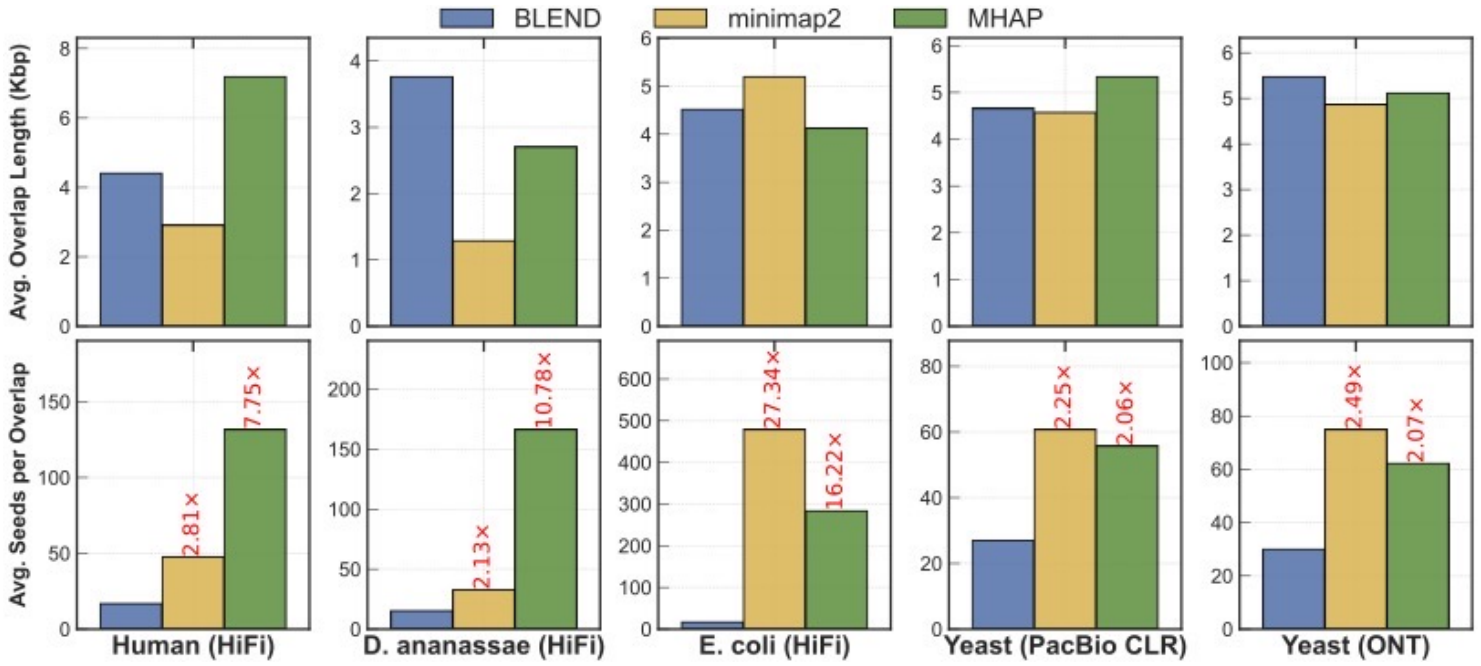
- **Significant improvements** in terms of both **performance** and **memory**
  - **Speedup** by 2.6×-63.5× (on average **19.5×**) and 22.2×-6317.1× (on average **1386.8×**)
  - Reducing the **memory** footprint by 0.9×-9.7× (on average **3.6×**) and 35.9×-238.1× (on average **130.5×**) compared to **minimap2 and MHAP**, respectively



Read Overlapping – Overlap Statistics

- BLEND finds **overlaps longer than minimap2 and MHAP** can find in most cases

- BLEND uses **significantly fewer seed matches** than other tools **by up to 27.34×** to find these longer overlaps.



Read Overlapping – Accuracy

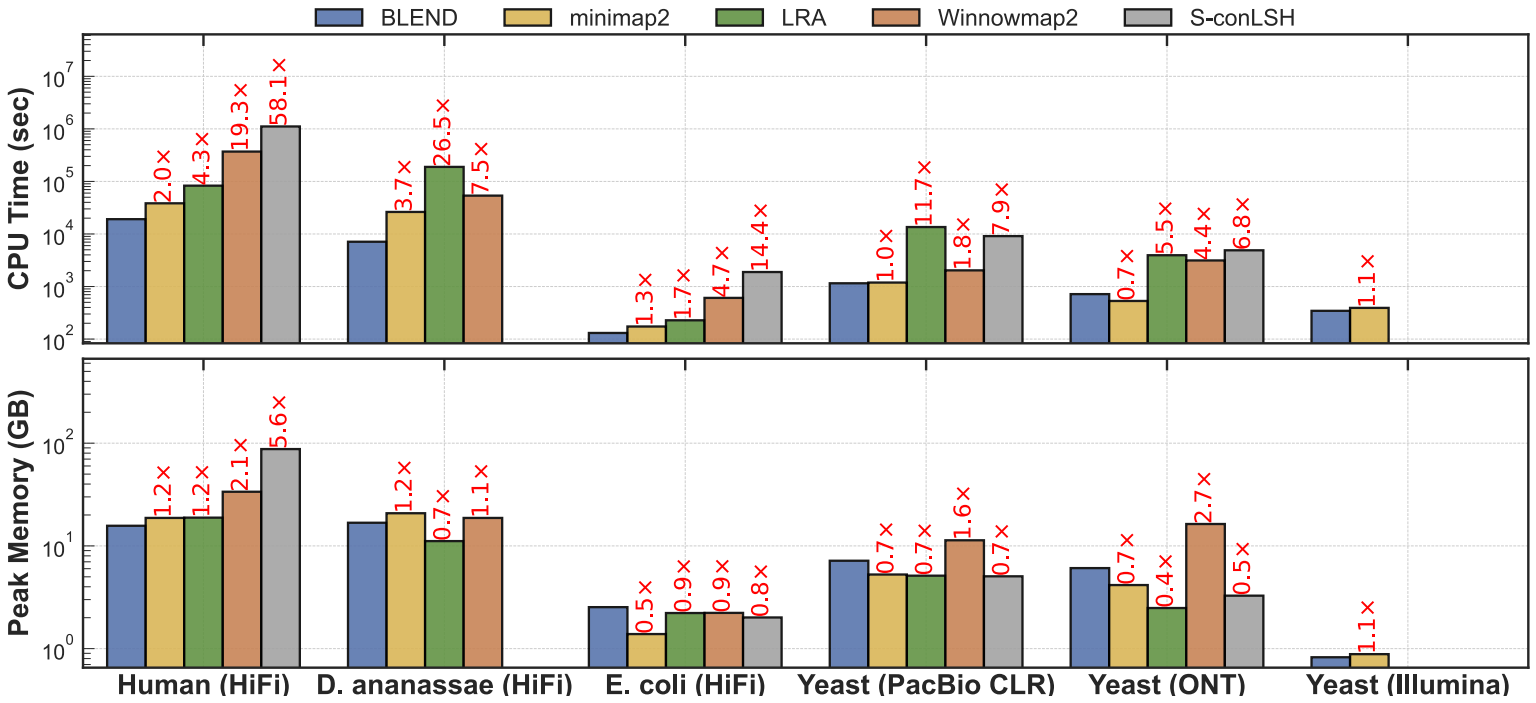
- **More accurate de novo assemblies** in most cases

- Contiguity results are similar to minimap2

Dataset	Tool	Average Identity (%)	Genome Fraction (%)	K-mer Compl. (%)	Aligned Length (Mbp)	Misassembly Ratio (%)	NGA50 (Kbp)	Average GC (%)	Assembly Length (Mbp)	Largest Contig (Mbp)	NG50 (Kbp)
Human CHM13	BLEND	**99.8526**	**98.4847**	**90.15**	3,092.59	**0.0108**	5,442	**40.78**	**3,095.210**	22.840	5,442
	minimap2	99.7421	97.1493	83.05	**3,095.49**	0.0503	**7,133**	40.71	3,100.974	**47.139**	**7,134**
	MHAP	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
	Reference	100	100	100	3,054.83	0	154,260	40.85	3,054.832	248.387	154,260
D. ananassae	BLEND	**99.7856**	**97.2308**	**86.43**	240.39	0.1230	**792**	**41.75**	**247.153**	**6.233**	**799**
	minimap2	99.7044	96.3190	72.33	**289.45**	**0.1013**	273	41.68	298.280	4.434	273
	MHAP	99.5551	0.7276	0.21	2.29	0.2197	N/A	42.07	2.350	0.286	N/A
	Reference	100	100	100	213.81	0	26,427	41.81	213.818	30.673	26,427
E. coli	BLEND	**99.8320**	**99.8801**	**87.91**	**5.12**	**0.0344**	**3,417**	**50.53**	5.122	**3.417**	**3,417**
	minimap2	99.7064	99.8748	79.27	5.09	0.3068	3,087	50.47	**5.042**	3.089	3,089
	MHAP	N/A	N/A	N/A	N/A	N/A	N/A	N/A	5.094	N/A	N/A
	Reference	100	100	100	5.05	0	4,945	50.52	5.046	4.946	4,945
Yeast (PacBio)	BLEND	89.1582	**97.6297**	**11.13**	**0.227**	N/A	N/A	**38.80**	13.679	1.105	551
	minimap2	88.9002	96.9709	9.74	0.195	N/A	N/A	38.85	**12.333**	**1.561**	**828**
	MHAP	**89.2182**	88.5928	9.5	0.186	N/A	N/A	38.81	10.990	1.024	436
	Reference	100	100	100	12.16	0	924	38.15	12.157	1.532	924
Yeast (ONT)	BLEND	**89.7622**	99.2982	**13.68**	**0.377**	N/A	N/A	**38.66**	**12.164**	1.554	825
	minimap2	88.9393	**99.6878**	12.06	0.328	N/A	N/A	38.74	12.373	**1.560**	**942**
	MHAP	89.1970	89.2785	11.35	0.297	N/A	N/A	38.84	10.920	1.443	619
	Reference	100	100	100	12.16	0	924	38.15	12.157	1.532	924

Read Mapping – Computational Resources

- **Speedup** by, on average, **1.7×, 12.5×, 7.5×, and 21.8×** compared to minimap2, LRA, Winnowmap2, and S-conLSH, respectively.
- Slightly more memory overhead than minimap2 and LRA (on average 0.9× and 0.8×), and lower than Winnowmap2 and S-conLSH (on average by 1.7× and 1.9×).



More from the paper…

- Read mapping accuracy of BLEND is similar to minimap2
  - Winnowmap2 generates the most accurate read mapping

- When the same parameters are used (e.g., window length)
  - Minimap2 also provides significant improvements in terms of performance and memory footprint (similar to BLEND) but with the cost of accuracy
  - This is because BLEND can find more seed matches (exact and approximate matches) than minimap2

- *BLEND-S* is preferred for HiFi data whereas *BLEND-I* is more suitable for erroneous reads

Conclusion

- BLEND can find fuzzy seed matches with a single lookup to improve the performance, memory efficiency, and accuracy
- Efficient fuzzy seed matches provide **higher sensitivity** and **extra room for adjusting the parameters** designed for exact-matching seeds
- **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 overlaps** while using **fewer seed matches** to find these overlaps

Executive Summary

- **Problem:** Existing attempts optimizing seed matches **suffer from** either
  - **Increasing the use of the costly sequence alignment step** due to many seed matches
  - **Limited sensitivity** when finding seed matches
- **Goal: Enable finding fuzzy (i.e., approximate) seed matches** with a **single lookup**
- **Key Ideas:**
  - Generate the same hash value for highly similar seeds using the SimHash technique
    - Provide mechanisms to convert seeds into sets of items for SimHash
  - Use hash tables to find fuzzy seed matches with a single lookup
- **Key Results**
  - For read overlapping, **significant improvements** in terms of both **performance** and **memory**
  - We can construct more accurate assemblies using the overlaps that BLEND finds
  - For read mapping, **performance improvements** compared to all tools

Finding Approximate Seed Matches

- Can Firtina, Jisung Park, Mohammed Alser, Jeremie S. Kim, 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***, 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] Mohammed Alser[1] Jeremie S. Kim[1] Damla Senol Cali[2]
Taha Shahroodi[3] Nika Mansouri-Ghiasi[1] Gagandeep Singh[1] Konstantinos Kanellopoulos[1]
Can Alkan[4] Onur Mutlu[1]
[1]*ETH Zurich* [2]*Bionano Genomics* [3]*TU Delft* [4]*Bilkent University*

Agenda for Today - Intelligent Algorithms

- BLEND
  - Finding approximate (fuzzy) seed matches with a single lookup
- AirLift
  - Avoiding redundant computations when moving from one reference genome to another
- Apollo & ApHMM
  - Error correction using profile Hidden Markov Models (pHMMs) and accelerating pHMMs

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]

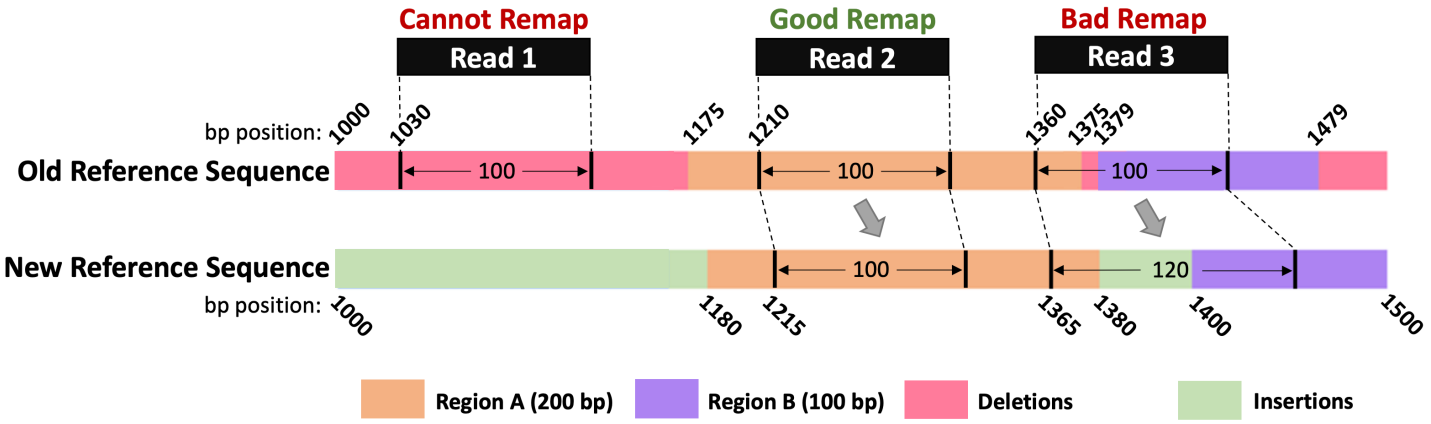
METHOD

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)
  - Recent high quality reference genomes (CHM13-T2T)
- 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
- Many 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

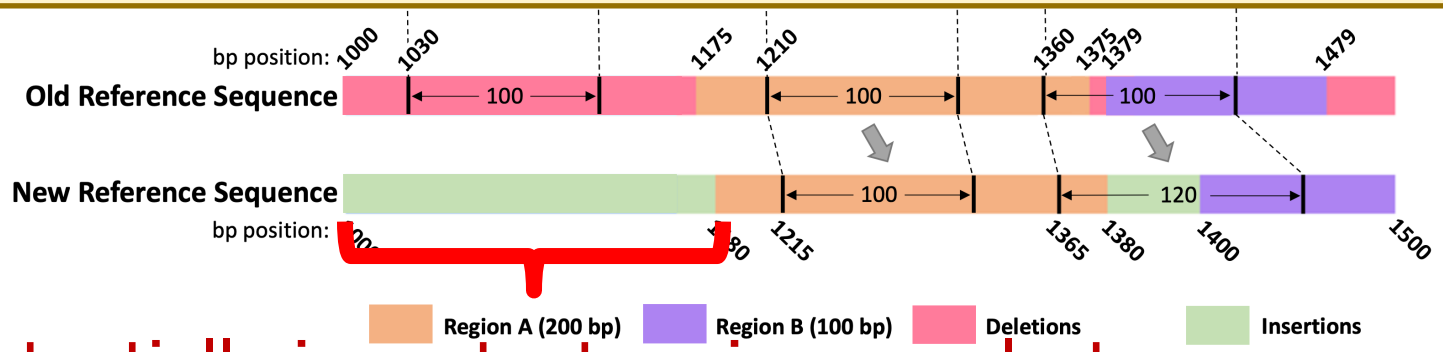
Generate **chain files**, which indicates similar blocks of regions across two genomes (via global alignment) and move annotations using the chain file

Problem

- Existing tools move annotations (e.g., read mapping positions) **only when** 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, based on a global alignment does not account for a **large portion of genes**

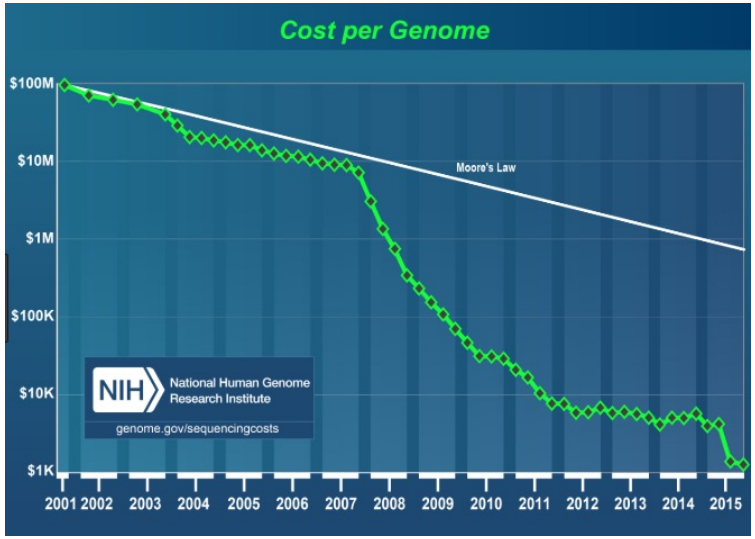
Simply moving annotations is **not accurate** as it **loses** a **significant amount of information**



Potentially important regions 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)



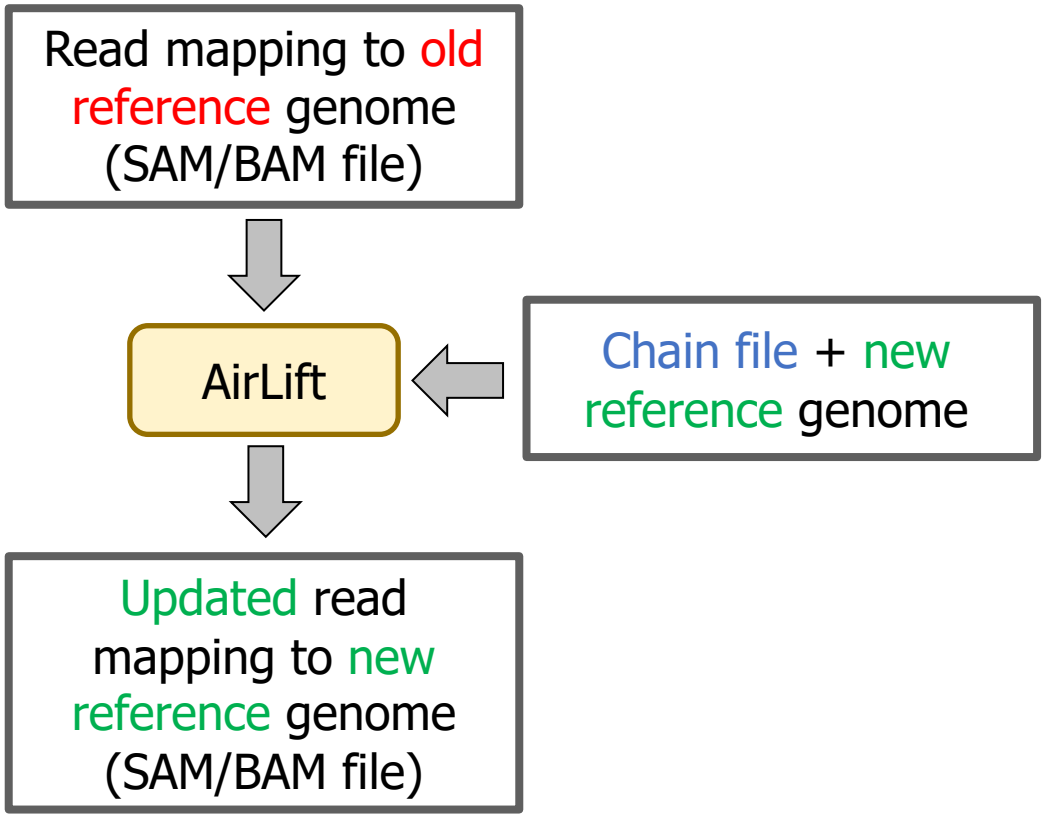
Goal

- 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

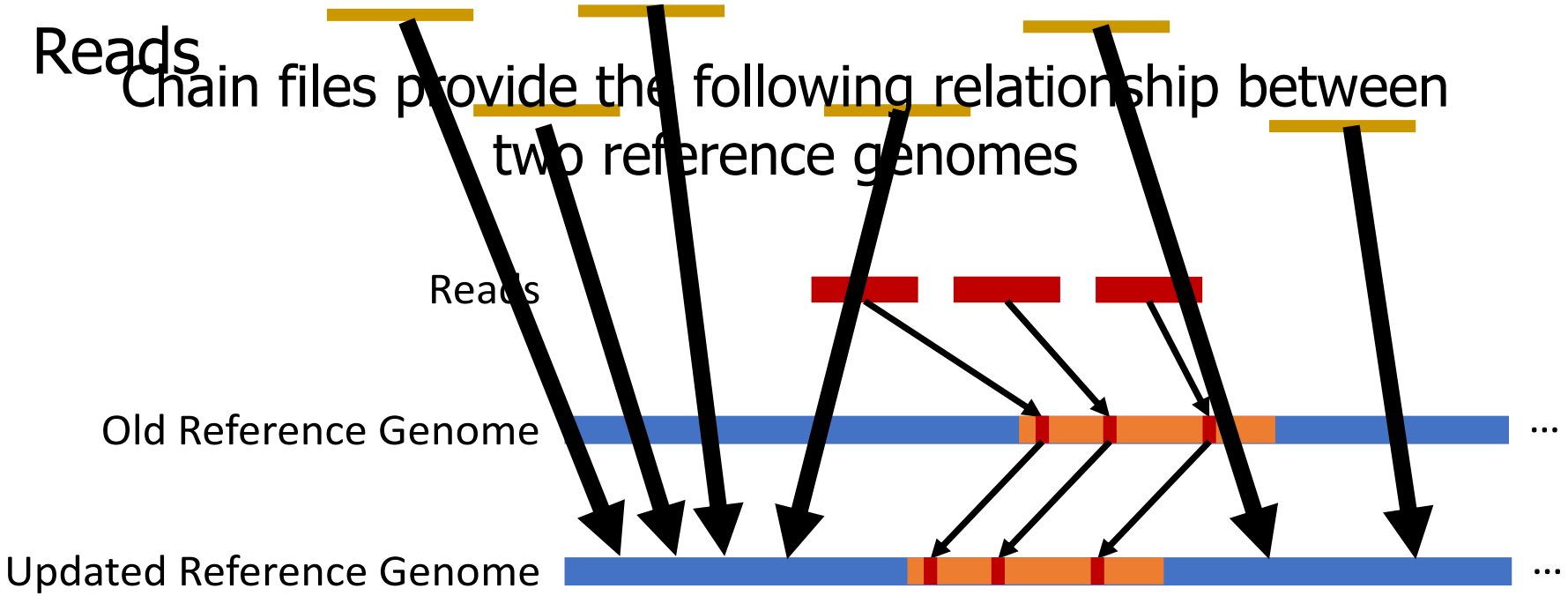
Mechanism

- 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



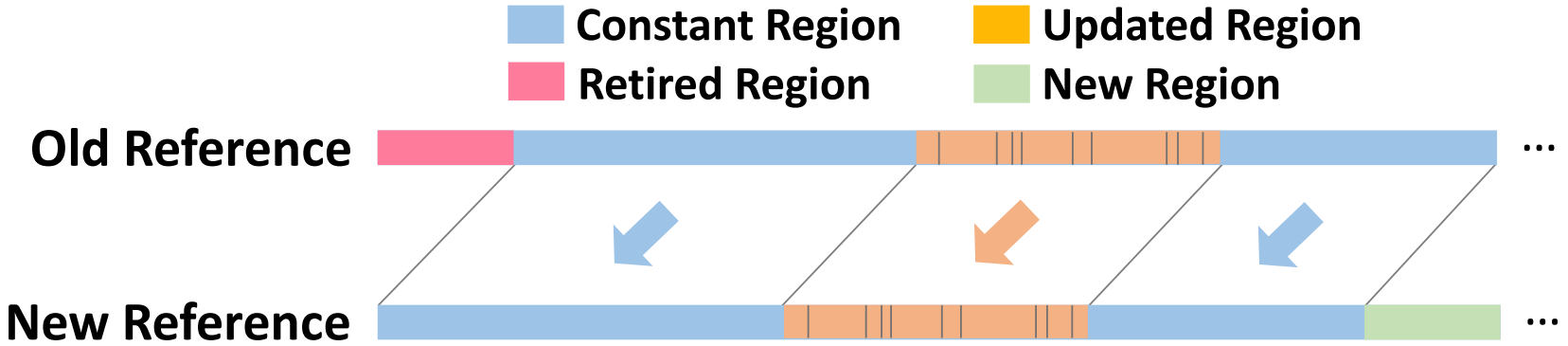
Mechanism (cont'd)

- 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

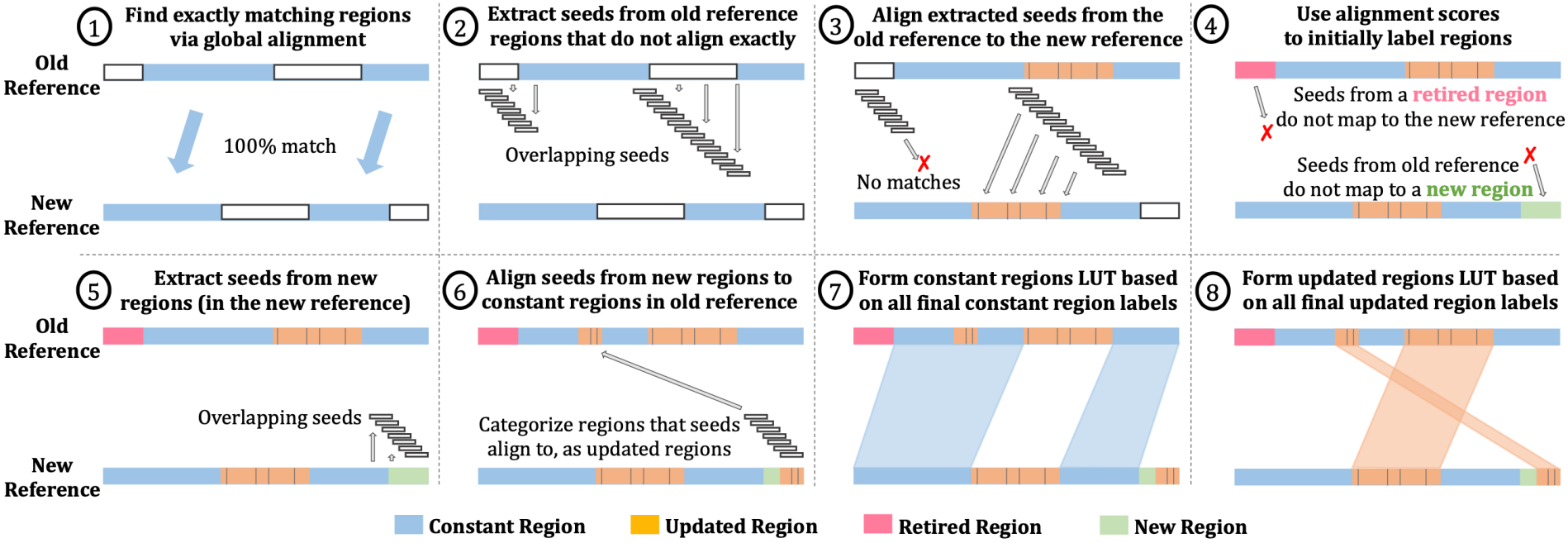


Labeling Regions of Reference Genomes

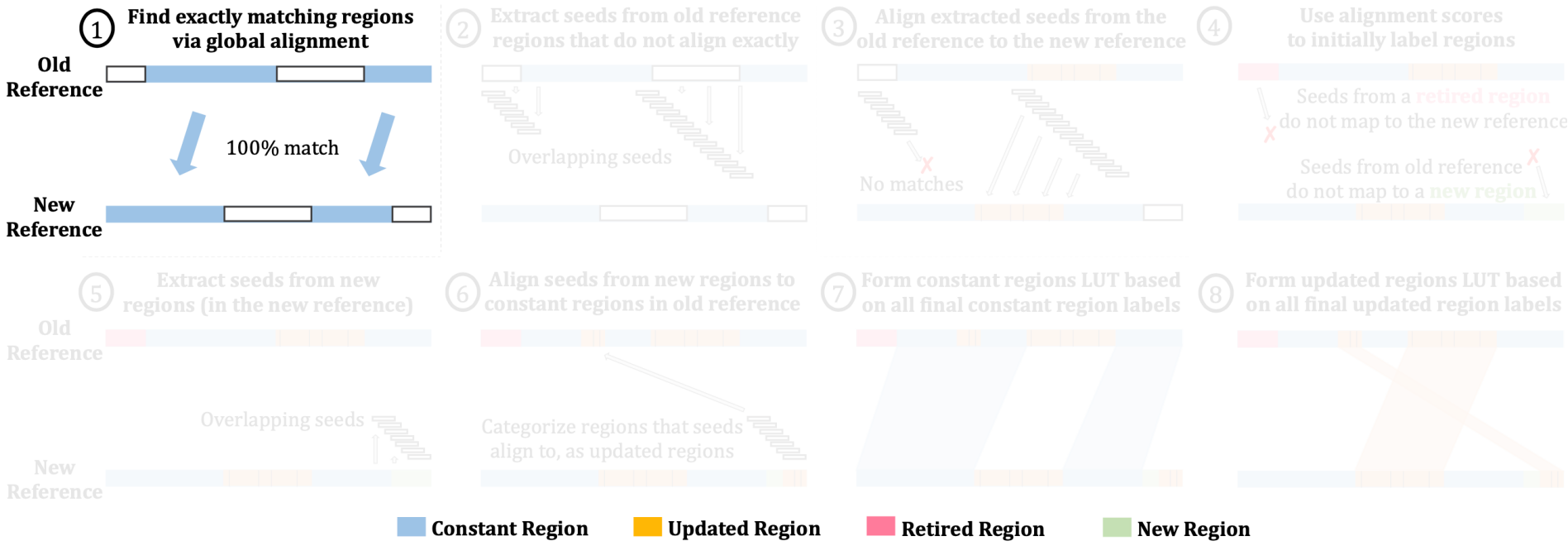
- We quickly label the regions between two reference genomes to decide if we should
  - Move the read mapping position without performing costly alignment or
  - Remap reads that maps in the regions that are missing in the new reference genome
- Finding labels between a pair of reference genomes is a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



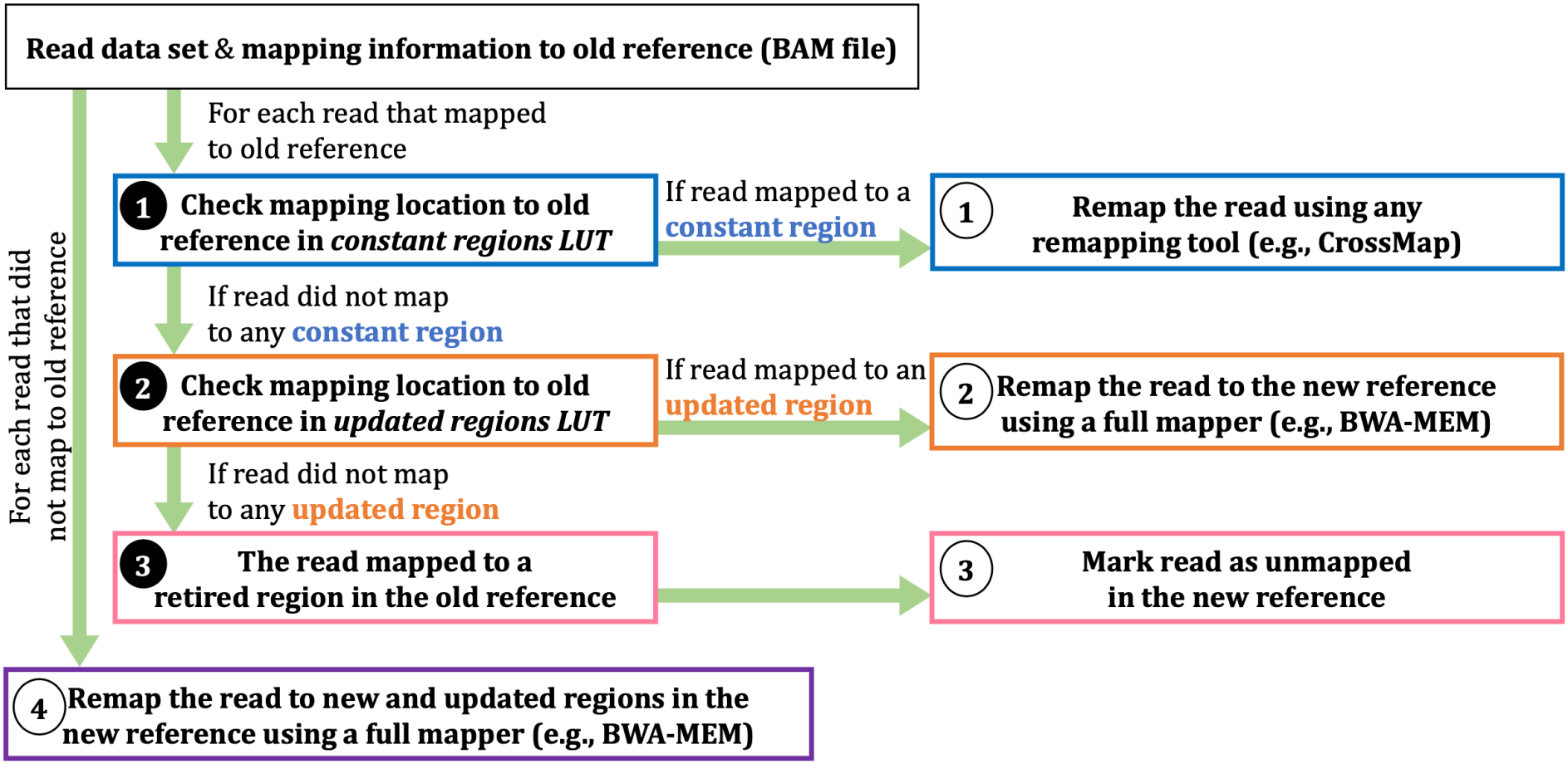
AirLift Index: Labeling Regions in 8 steps



AirLift Index: Labeling Regions in 8 steps

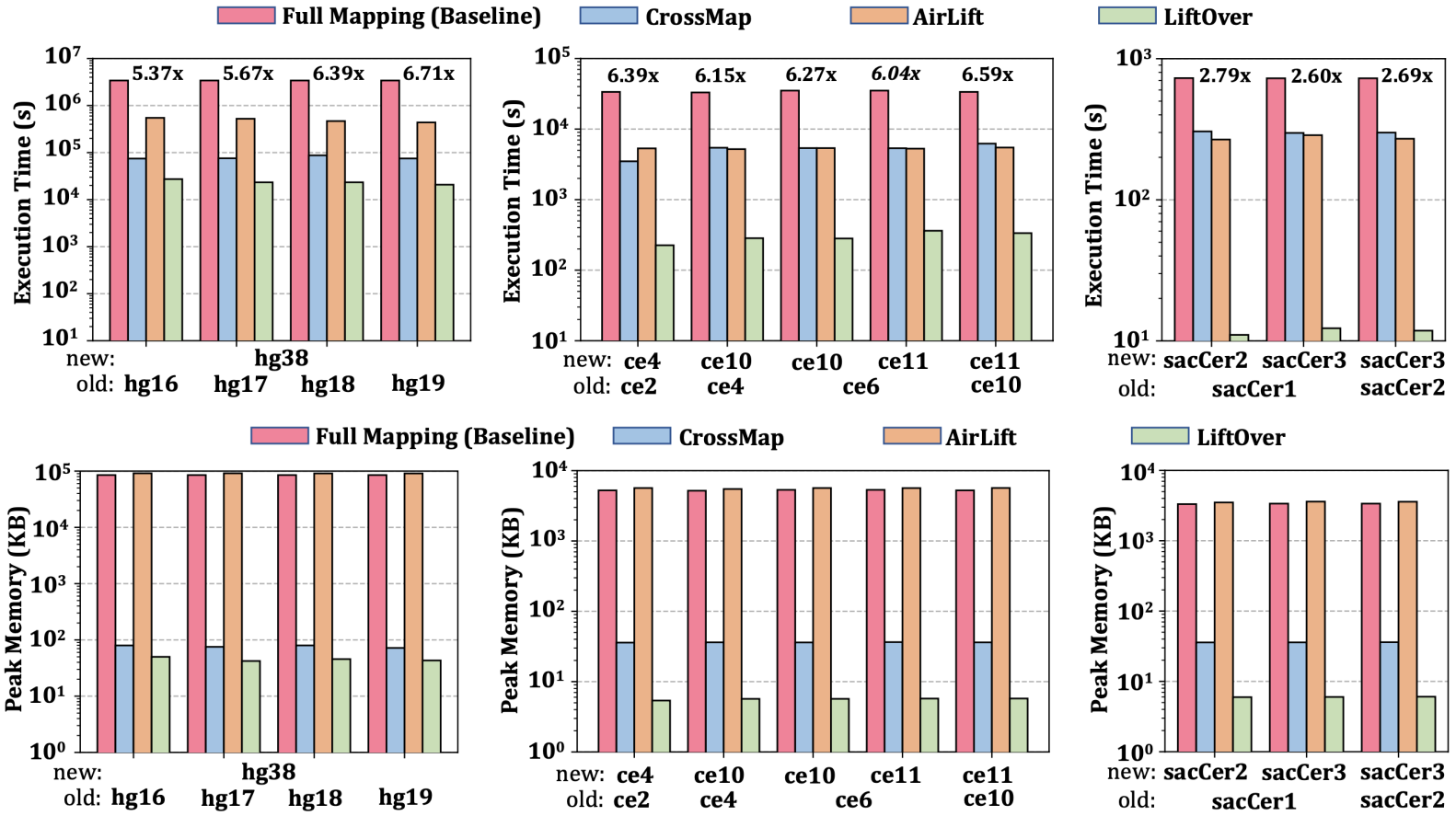


Updating Read Mapping using AirLift Index



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 Technique	Read Sets from	Read Sets to	vs. Full Mapping SNP (%)	vs. Full Mapping Indel (%)	vs. Ground Truth SNP (%)	vs. Ground Truth 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 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*]

Mapping Constant Regions Between References

- Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Can Alkan, and Onur Mutlu,
  "FastRemap: A Tool for Quickly Remapping Reads between Genome Assemblies"
  *Preprint in **arXiv***, 26 January 2022.
  [arXiv preprint]
  [FastRemap Source Code]

FastRemap: A Tool for Quickly Remapping Reads between Genome Assemblies

Jeremie S. Kim[1] Can Firtina[1] Meryem Banu Cavlak[1] Damla Senol Cali[2,3]
Can Alkan[4] Onur Mutlu[1,2,4]

[1]*ETH Zürich* [2]*Carnegie Mellon University* [3]*Bionano Genomics* [4]*Bilkent University*

Agenda for Today - Intelligent Algorithms

- BLEND
  - Finding approximate (fuzzy) seed matches with a single lookup
- AirLift
  - Avoiding redundant computations when moving from one reference genome to another
- Apollo & ApHMM
  - Error correction using profile Hidden Markov Models (pHMMs) and accelerating pHMMs

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 ✉

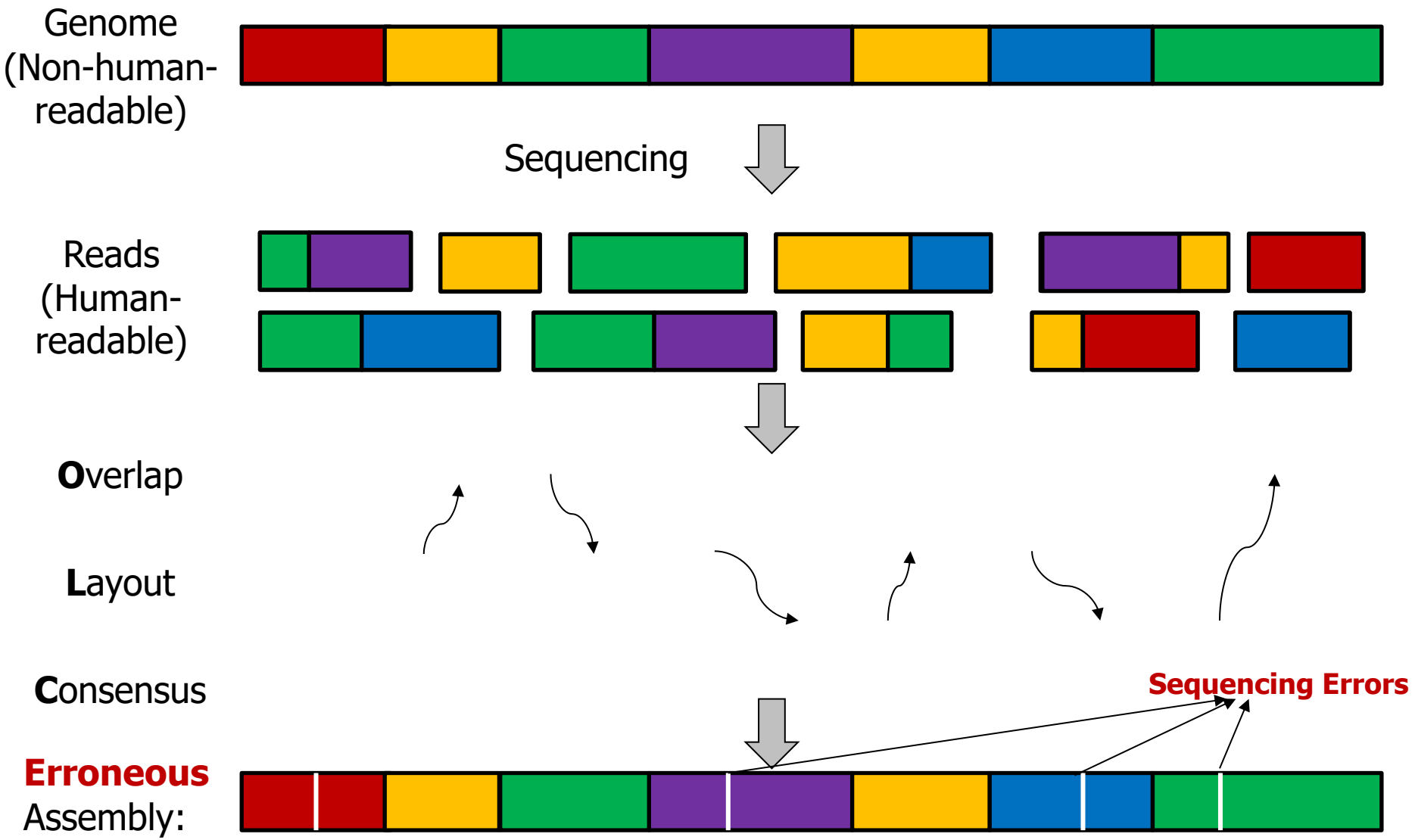
Bioinformatics, Volume 36, Issue 12, 15 June 2020, Pages 3669–3679,
https://doi.org/10.1093/bioinformatics/btaa179

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

Executive Summary

- **Problem:**
  - Long read *de novo* assembly is inherently erroneous – needs error correction
  - Existing assembly error correction (polishing) techniques cannot adapt to varying sequencing technologies and **do not scale** well for large genomes
- **Goal:** Propose a technology-independent, accurate, and scalable assembly polishing algorithm
- **Key Ideas:**
  - Use a probabilistic graph structure to represent assemblies with potential errors
    - Profile hidden Markov models (pHMMs)
  - Update (train) the probabilities using read alignments to the assembly
  - Decode the trained pHMM to identify and correct the errors in the assembly
- **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 accurate assemblies** when reads from multiple technologies are used within a single run compared to other polishing tools
  - Apollo is ~25x **slower** on average (up to ~600x) than other polishers

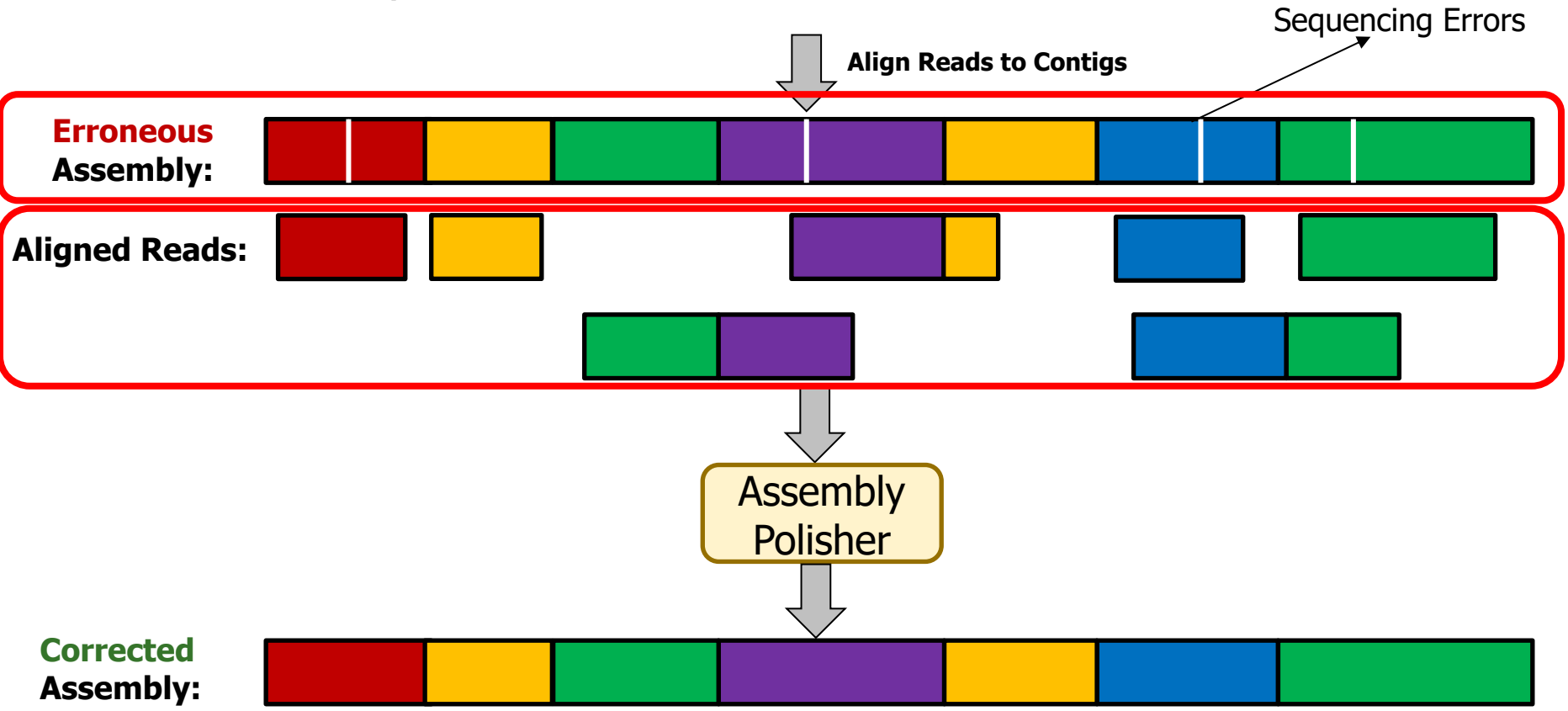
De novo Assembly from Erroneous Reads



Assembly Polishing

- Sequencing errors on reads may propagate to contigs
  - Leading to inaccurate downstream analysis

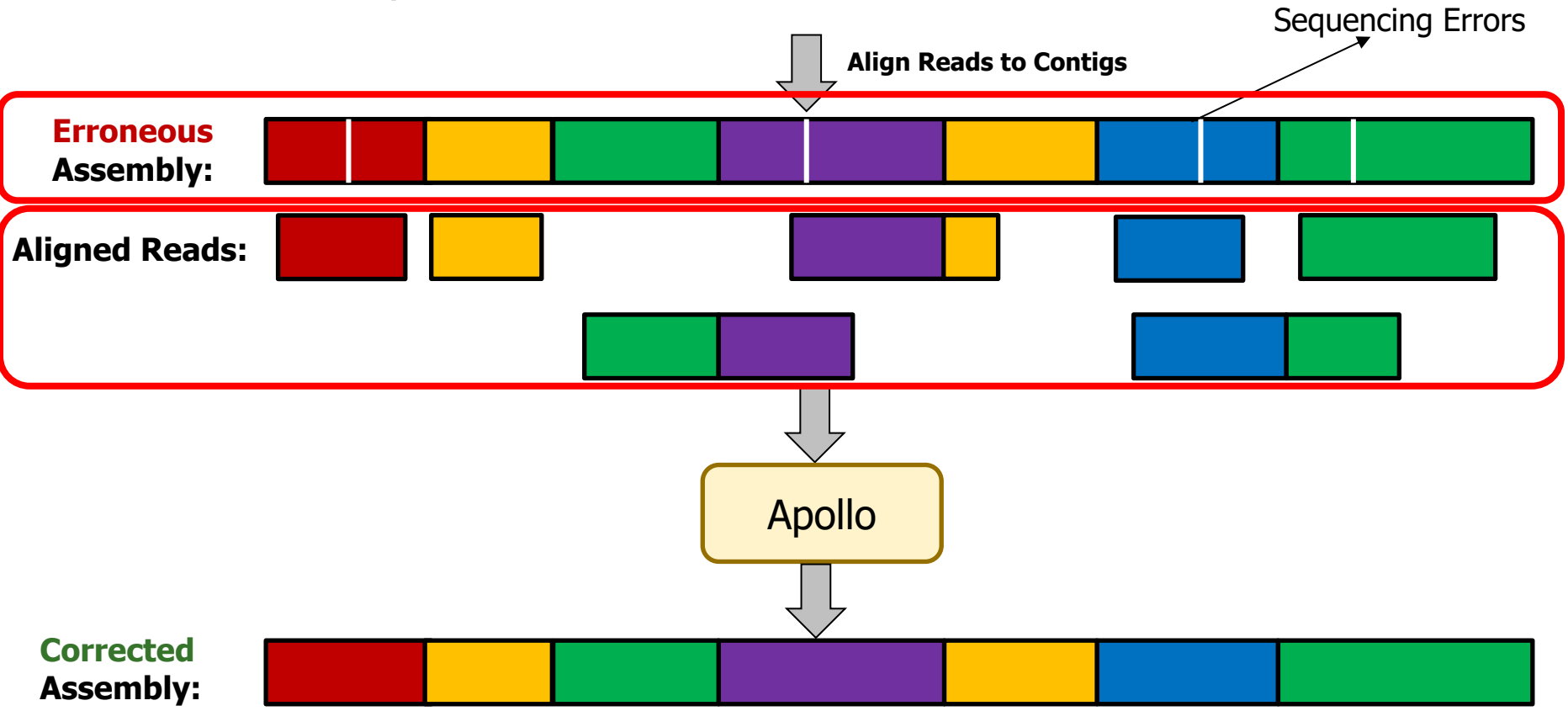
- **Idea:** Align the reads to contigs to re-adjust the consensus step of *de novo* assembly



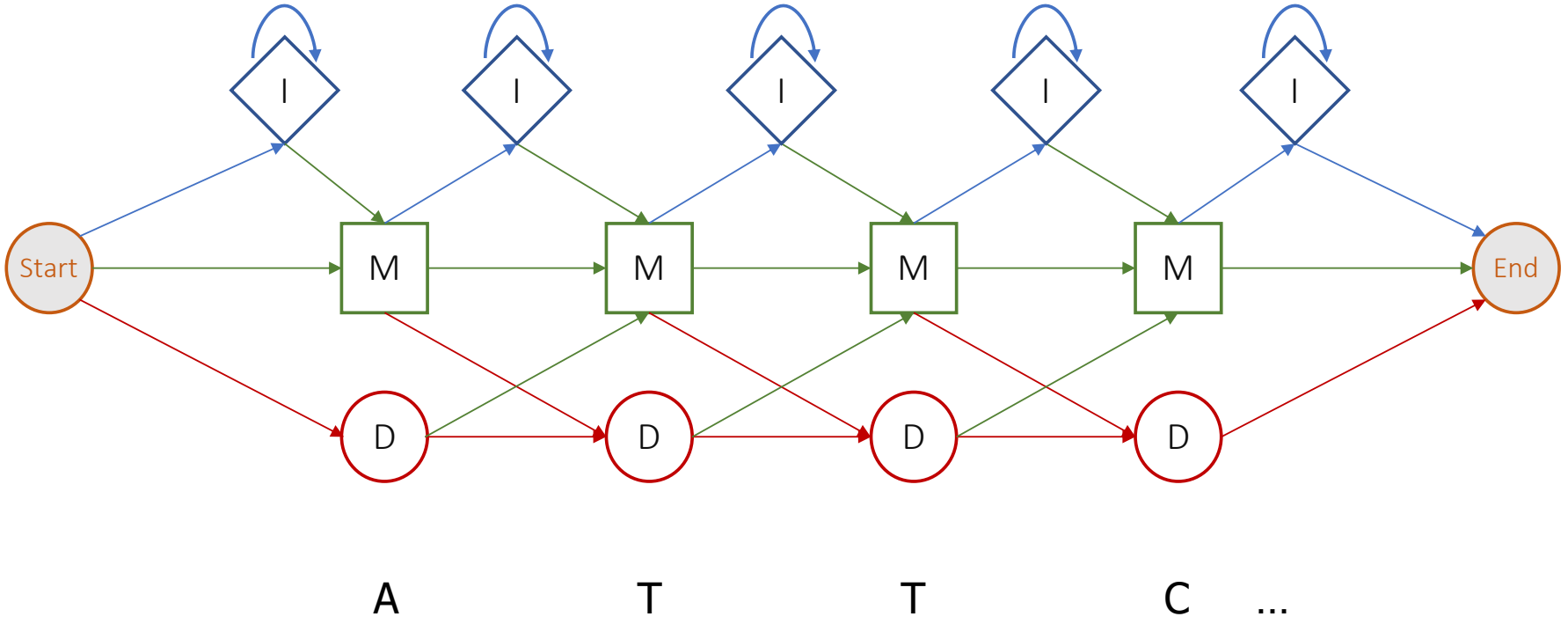
Assembly Polishing

- Sequencing errors on reads may propagate to contigs
  - Leading to inaccurate downstream analysis

- **Idea:** Align the reads to contigs to re-adjust the consensus step of *de novo* assembly

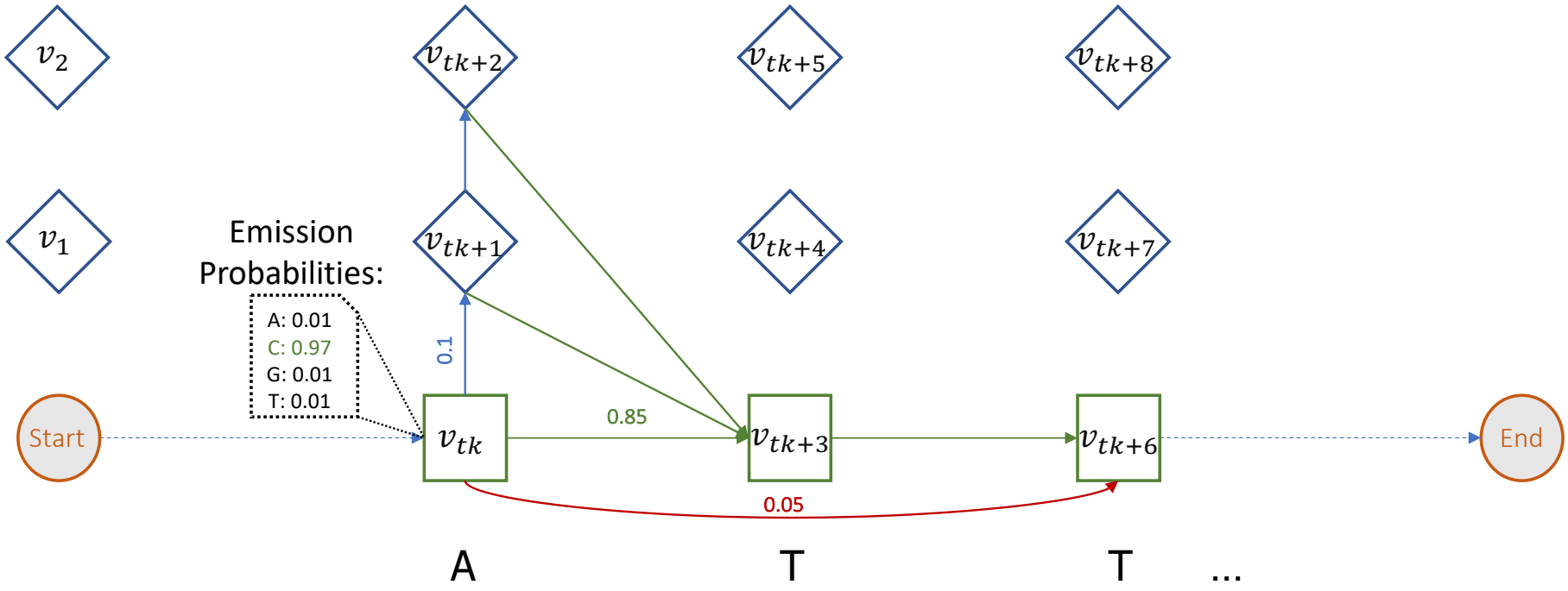


Profile Hidden Markov Models (pHMMs)



An alternative design of pHMMs

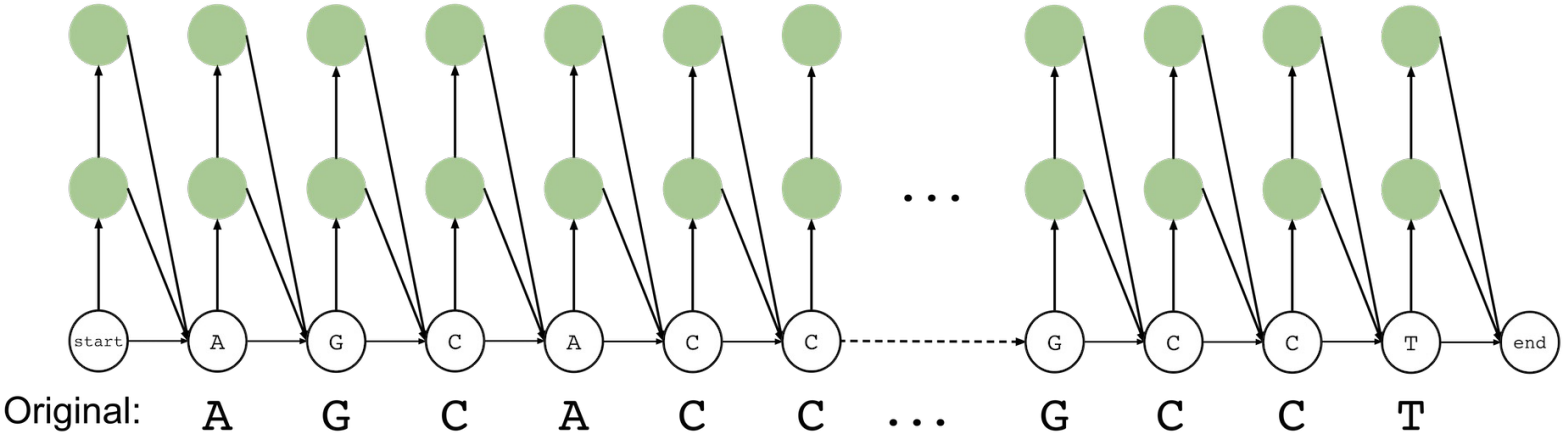
- No self-loops
  - To precisely determine number of insertions and inserted characters
- No deletion states
  - Each state emits a character – an accurate heuristic for better memory management



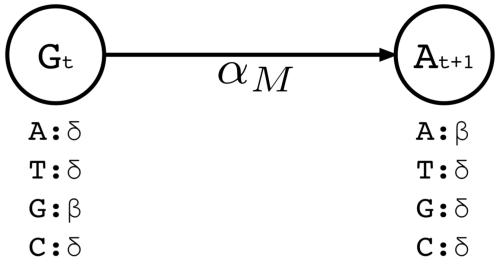
Assembly Polishing using pHMMs

A profile hidden Markov model of the assembly:

AGCACC…GCCT

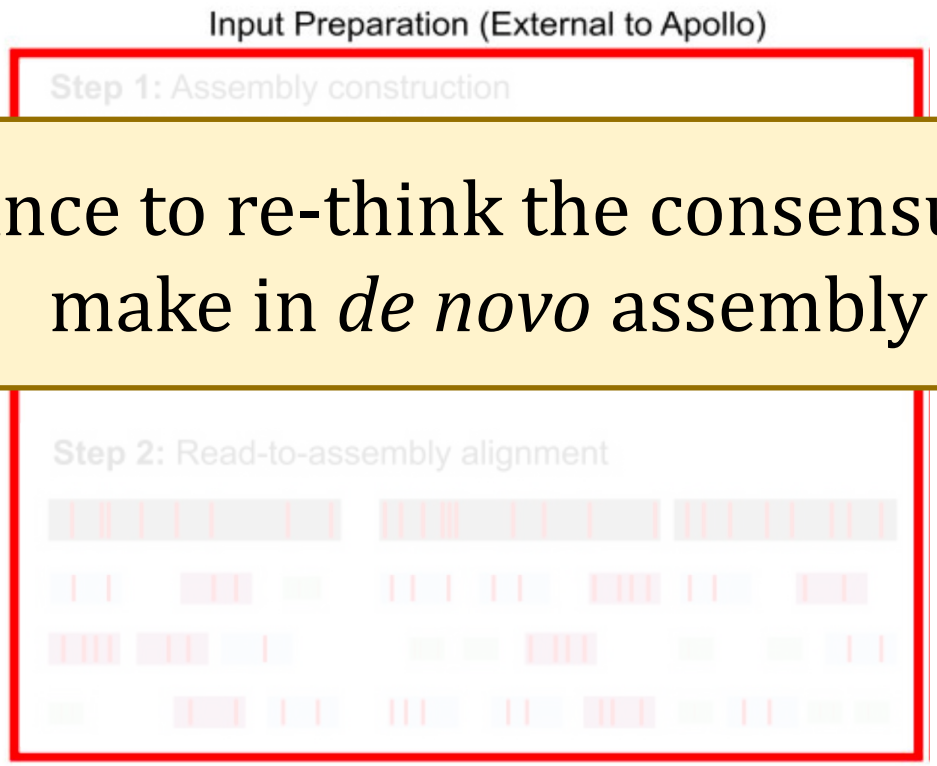


- Parameters to update:
  - Emission probabilities
  - Transitions probabilities



Constructing Accurate *de novo* Assemblies

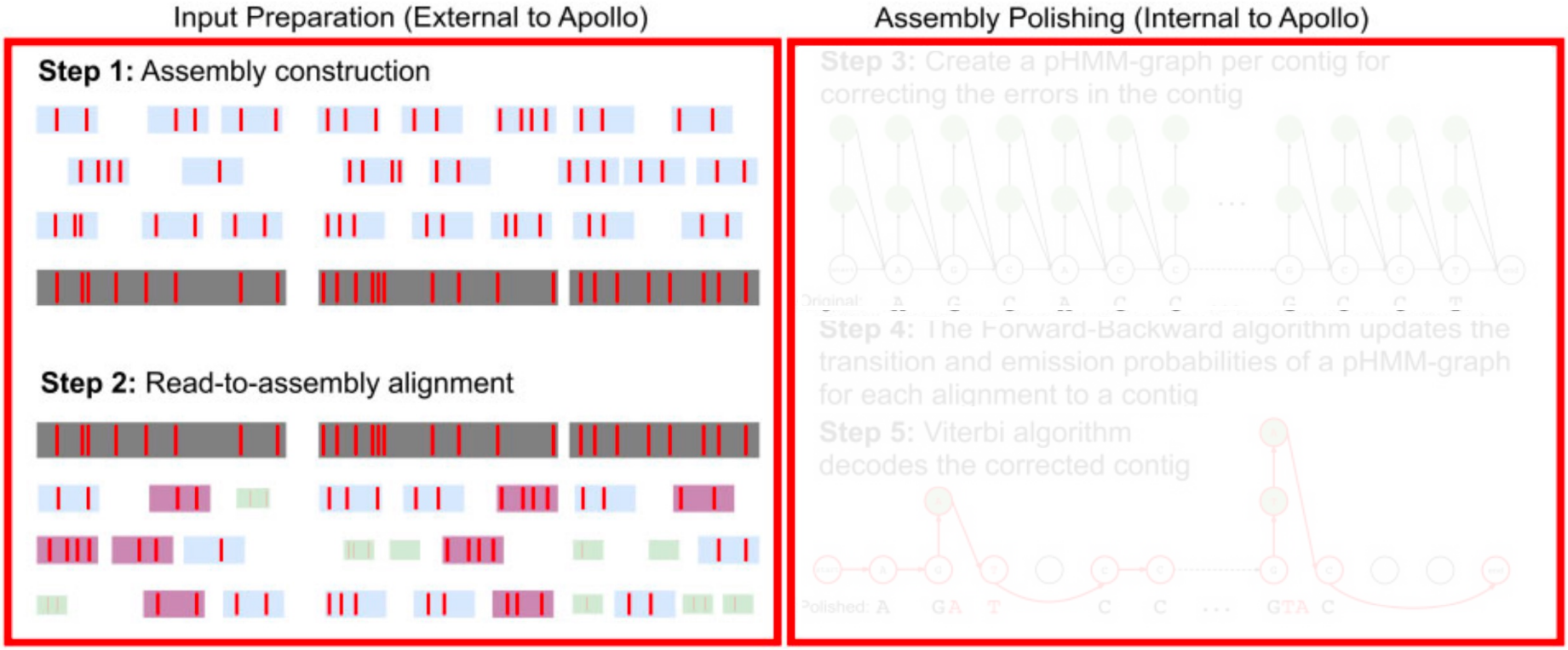
- Steps **external** to Apollo
  - **Step 1:** Construct *de novo* assembly using any assembler
  - **Step 2:** Align reads to *de novo* assembly of a genome



Another chance to re-think the consensus choices we make in *de novo* assembly

Apollo Workflow – Polishing

- **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



Evaluation Methodology and 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 tool* 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 & Limitations

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

- **Apollo performs worse** due to its computationally expensive **training** and **decoding** (inference) steps
  - Both training and inference algorithms provide **massive parallelism opportunities**
  - CPUs are not well suited for efficiently utilizing all available parallelism
  - Can we do better? **Hardware acceleration**?

Hardware Acceleration for pHMMs

- Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joel Lindegger, Mohammed Alser, Juan Gómez-Luna, Sreenivas Subramoney, and Onur Mutlu, **"ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis"**
  arXiv, 2022.
  [Source Code – will be public soon at http://github.com/CMU-SAFARI]

ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis

Can Firtina[1] Kamlesh Pillai[2] Gurpreet S. Kalsi[2] Bharathwaj Suresh[2] Damla Senol Cali[3]
Jeremie S. Kim[1] Taha Shahroodi[4] Meryem Banu Cavlak[1] Joel Lindegger[1] Mohammed Alser[1]
Juan Gómez Luna[1] Sreenivas Subramoney[2] Onur Mutlu[1]
[1]*ETH Zurich* [2]*Intel Labs* [3]*Bionano Genomics* [4]*TU Delft*

PHMMs in Many Applications

- Represent sequence of events with **potential** modifications to these events
  - To compare one sequence to many other
  - To construct the consensus sequence of many sequences

- **Genomics**
  - Multiple sequence alignment
  - Protein family search
  - Error correction

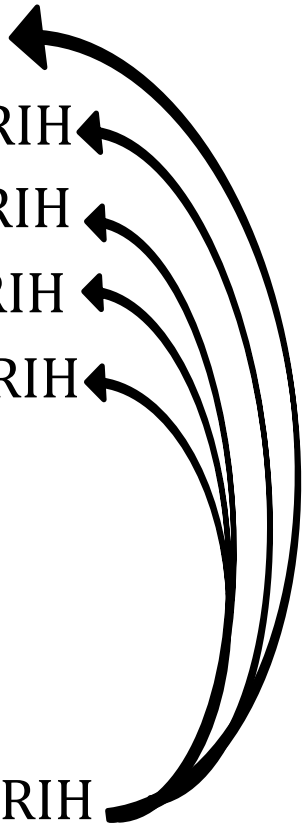
- **Malware detection**

- **Pattern matching**

Comparing a Sequence to Many Sequences

Protein Family:

CSQCGKSFSQSSSLRTHQRIH
YHCSQCGKSFSQSSSLRTHQRIH
YHCSQCGKGFSRSSSLSTHQRIH
YHCSQCGKSFSDSSSLQSHERIH
YYCSQCEKSFNQLSTLQSHQRIH



A Protein Sequence:

YQCDICGKSFSHNGHLVTHNRIH

Too many costly sequence alignment!

Can we do better?

Comparing a Sequence to Many Sequences

Protein Family:

CSQCGKSFSQSSSLRTHQRIH

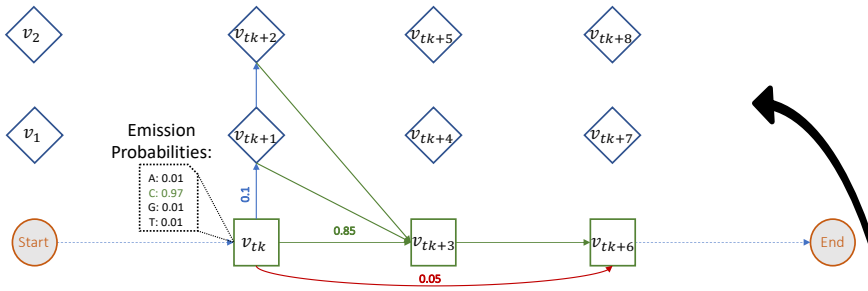
YHCSQCGKSFSQSSSLRTHQRIH

YHCSQCGKGFSRSSSLSTHQRIH

YHCSQCGKSFSDSSSLQSHERIH

YYCSQCEKSFNQLSTLQSHQRIH

Protein Family:



Single comparison instead of many

A Protein Sequence:

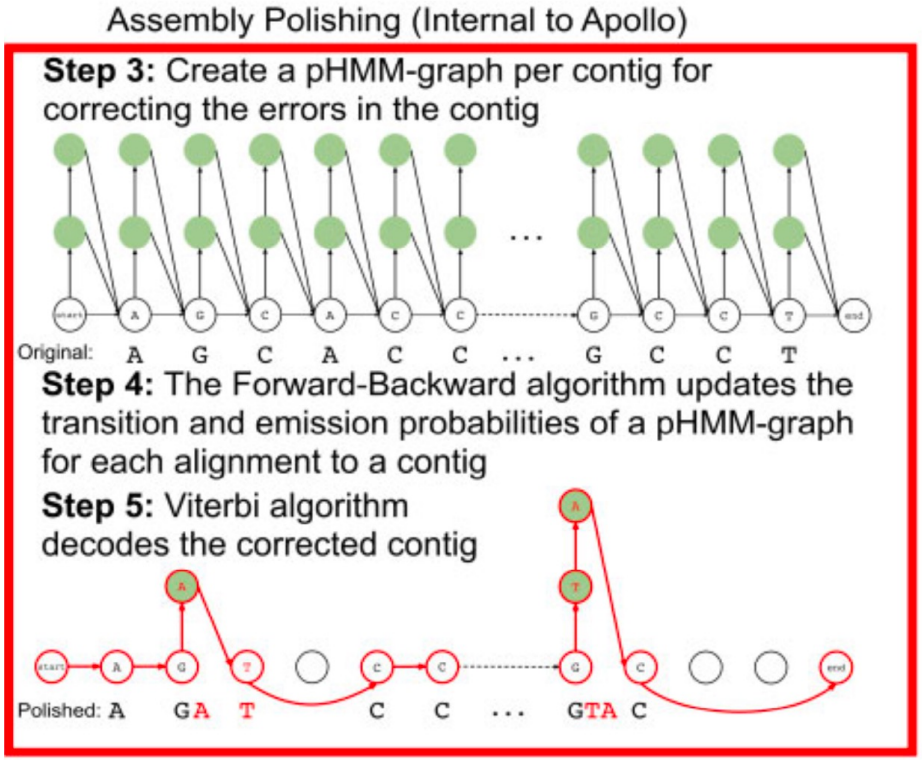
YQCDICGKSFSHNGHLVTHNRIH

A Protein Sequence:

YQCDICGKSFSHNGHLVTHNRIH

Constructing the Consensus Sequence

- Hercules [Firtina+, 2018]
- Apollo [Firtina+, 2020]
- Lanyue+, 2020



Training pHMMs

- The Baum-Welch algorithm

Massive Parallelism: Many states at time t

- Backward Calculation

Parallelism: **Simple** arithmetic operations

- Parameter Updates

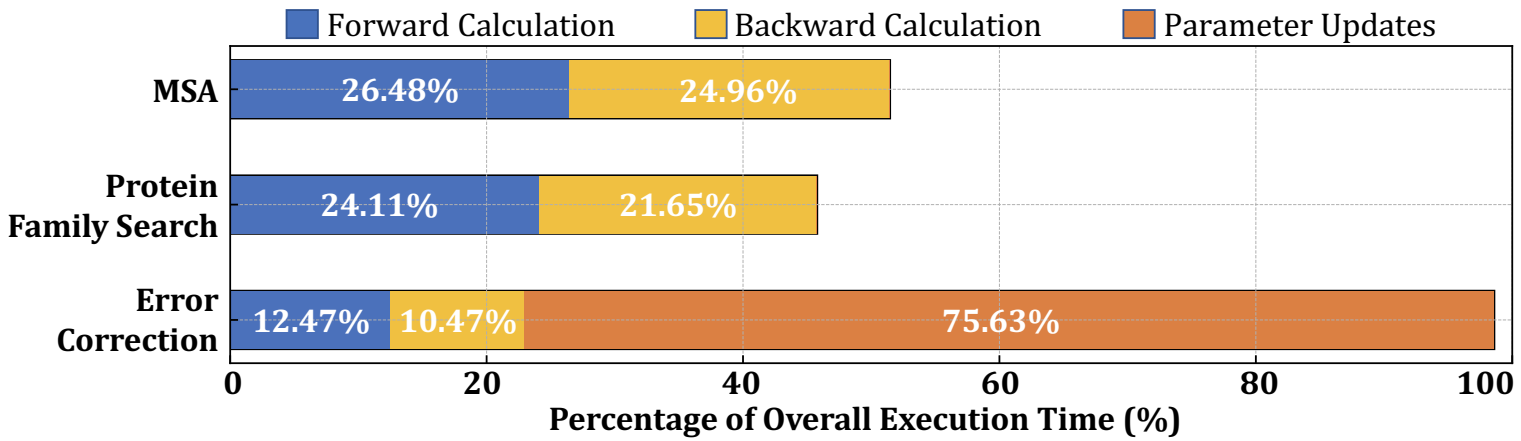
Data dependency

- **Multiple iterations** of the Baum-Welch algorithm may be needed for high accuracy

Motivation

- The Baum-Welch algorithm is the common approach for **accurate training**

- The Baum-Welch algorithm is the main source of the **performance bottleneck** in pHMM-based applications

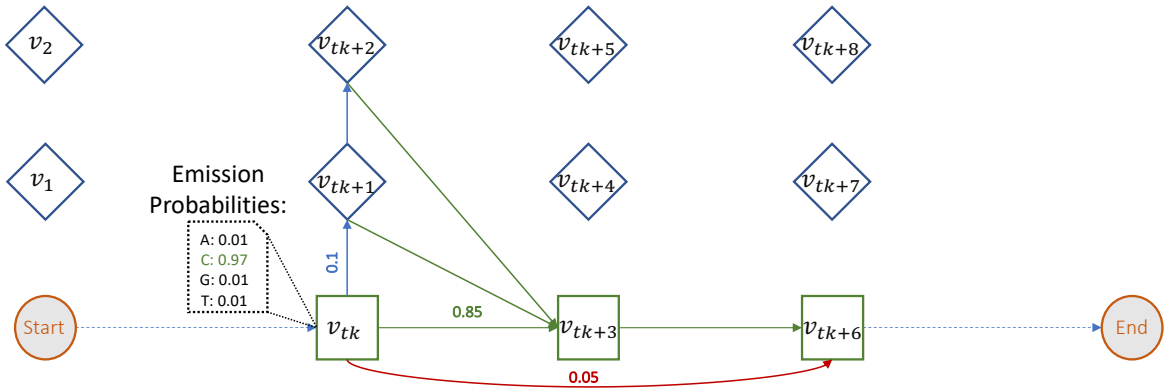


Goal

- **Accelerate** the Baum-Welch algorithm for pHMMs
- Provide an **accurate acceleration framework** for a wide range of pHMM-based applications
- To this end, we propose **ApHMM**, the **first** hardware-software co-designed **acceleration framework** for a wide range of bioinformatics applications that use pHMMs

Key Ideas – PHMM design

- Adopt the modified pHMM design to **support a wide range of applications**
  - Resolves the complexity and ambiguity issues associated with the commonly-used pHMM design

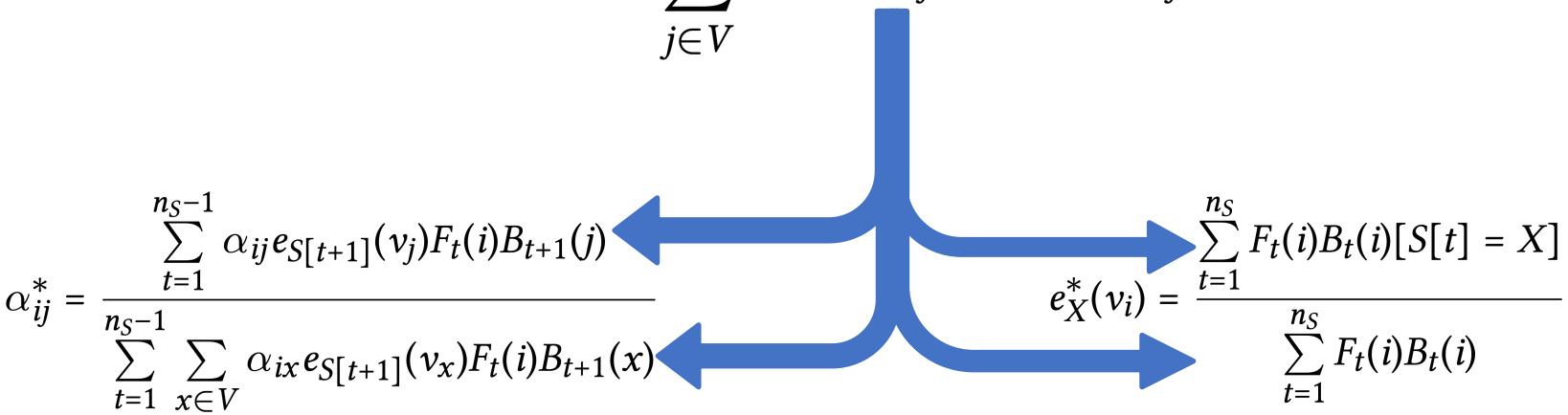


Key Ideas - Pipelining

- Identify and exploit the **pipelining opportunities**
  - To minimize the memory overhead

- The Backward calculation **does not need to be stored**
  - Pipeline for updating transition and emission probabilities

$$B_t(i) = \sum_{j \in V} B_{t+1}(j)\alpha_{ij}e(S[t+1], v_j)$$

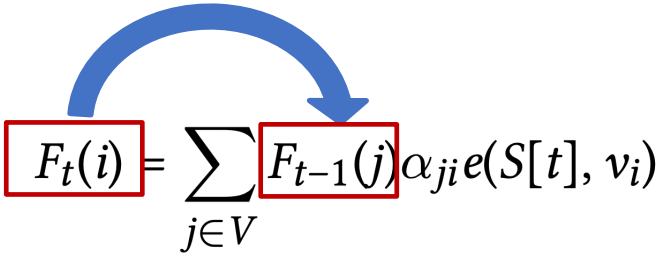


$$\alpha^*_{ij} = \frac{\sum_{t=1}^{n_S-1} \alpha_{ij} e_{S[t+1]}(v_j) F_t(i) B_{t+1}(j)}{\sum_{t=1}^{n_S-1} \sum_{x \in V} \alpha_{ix} e_{S[t+1]}(v_x) F_t(i) B_{t+1}(x)}$$

$$e^*_X(v_i) = \frac{\sum_{t=1}^{n_S} F_t(i) B_t(i) [S[t] = X]}{\sum_{t=1}^{n_S} F_t(i) B_t(i)}$$

Key Ideas – Cache Reuse

- Exploit the iterative nature of the Baum-Welch computations (e.g., Forward)
  - To minimize the data movement overhead
  - Improve the cache reuse with loop tiling



Key Ideas – Lookup Tables (LUTs)

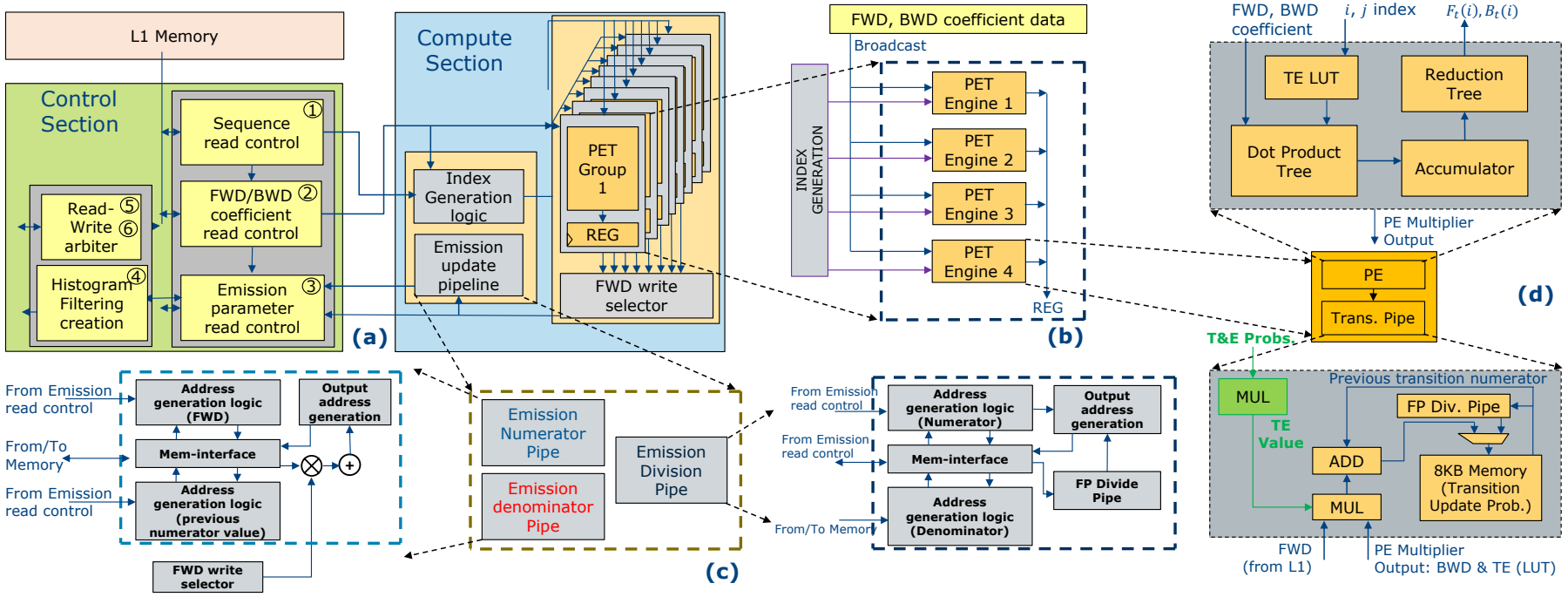
- Avoid performing the same calculations
  - Limited combinations of emission and transition multiplications
  - Use Lookup Tables (LUTs) to minimize the computational overhead

$$F_t(i) = \sum_{j \in V} F_{t-1}(j) \alpha_{ji} e(S[t], v_i)$$

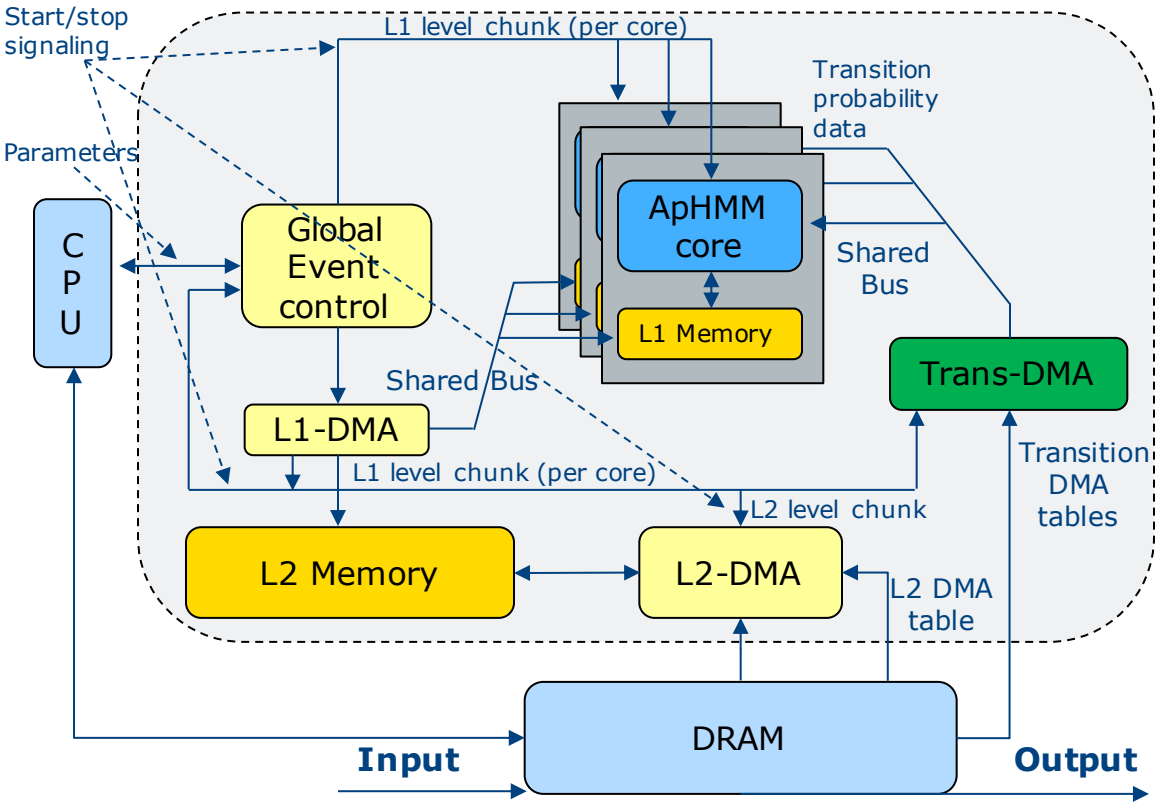
$$B_t(i) = \sum_{j \in V} B_{t+1}(j) \alpha_{ij} e(S[t+1], v_j)$$

$$\alpha^*_{ij} = \frac{\sum_{t=1}^{n_S-1} \alpha_{ij} e_{S[t+1]}(v_j) F_t(i) B_{t+1}(j)}{\sum_{t=1}^{n_S-1} \sum_{x \in V} \alpha_{ix} e_{S[t+1]}(v_x) F_t(i) B_{t+1}(x)}$$

Putting it all together – Hardware Design



System Integration – Many cores



Evaluation Methodology

- Hardware accelerator: ApHMM
  - Implemented in SystemVerilog
  - Synthesis in a typical 28nm process technology @1GHz
  - GPU implementation: ApHMM-GPU (CUDA 11.6)

- Use cases:
  - Error correction
  - Protein Family Search
  - Multiple Sequence Alignment

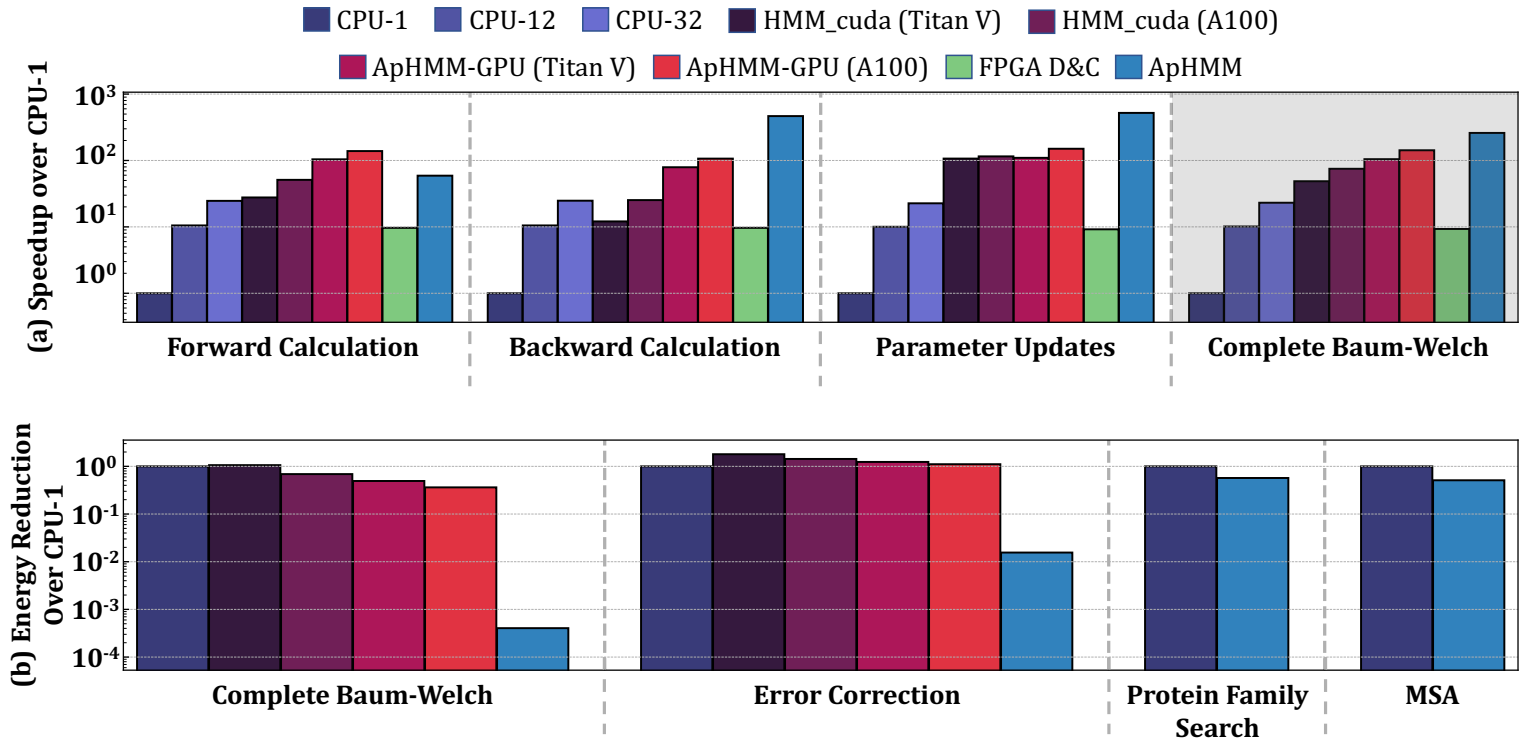
- Compared to the CPU, GPU, and FPGA implementations

- Baseline CPU: AMD EPYC 7742 (7nm) @2.26GHz
- Baseline GPUs: NVIDIA A100 and NVIDIA Titan V

Results: Executing the Baum-Welch Algorithm

HW-SW optimizations provide **significant improvements** in terms of **performance** and **energy consumption**

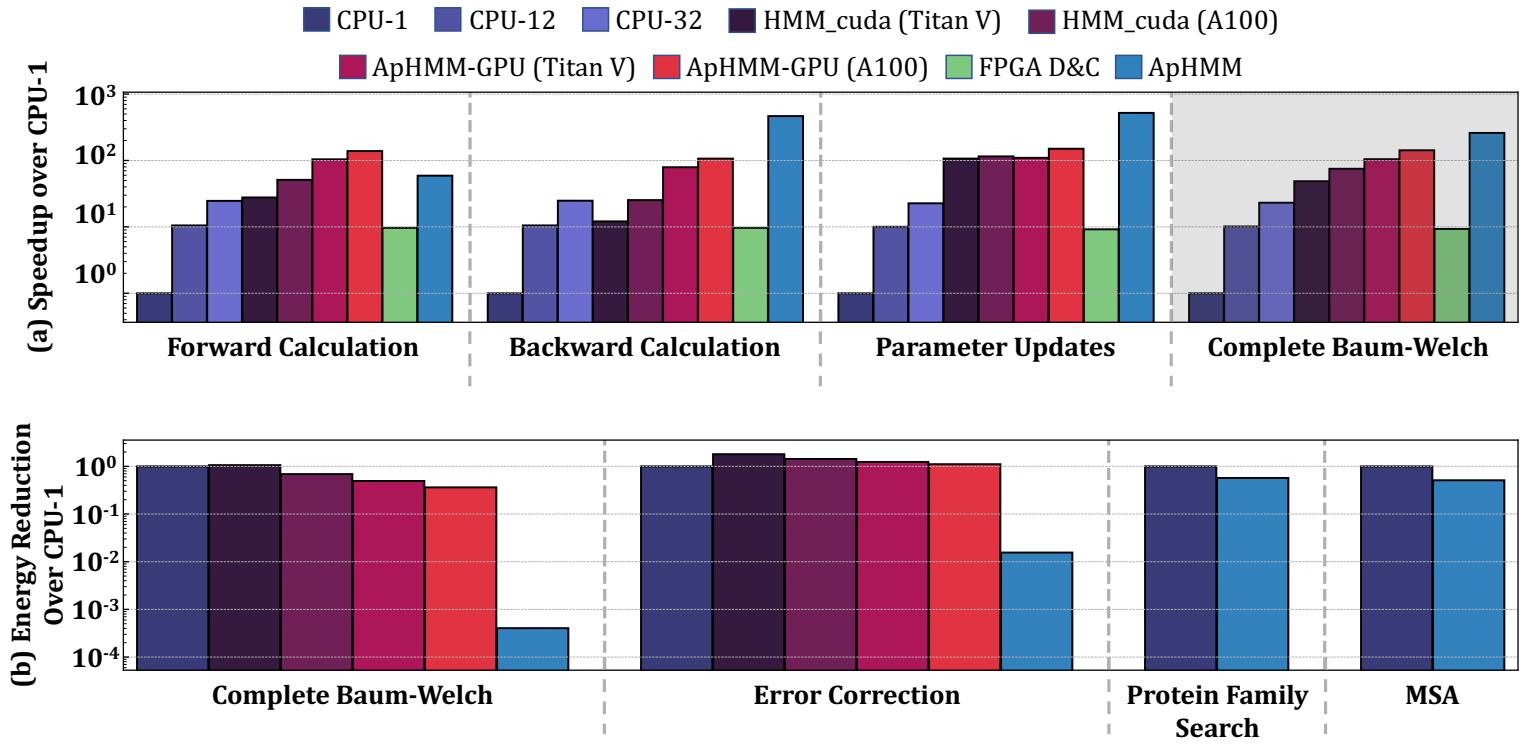
Data movement becomes the **bottleneck** for the Forward Calculation (fully stored in the L2 cache and DRAM)



Results: Executing the Baum-Welch Algorithm

ApHMM-GPU takes **advantage of pHMM-based optimizations** when compared to HMM_cuda

Limited speedup for the GPU implementations due to **frequent accesses to host** for synchronization and sorting (hint for specialized HW design)

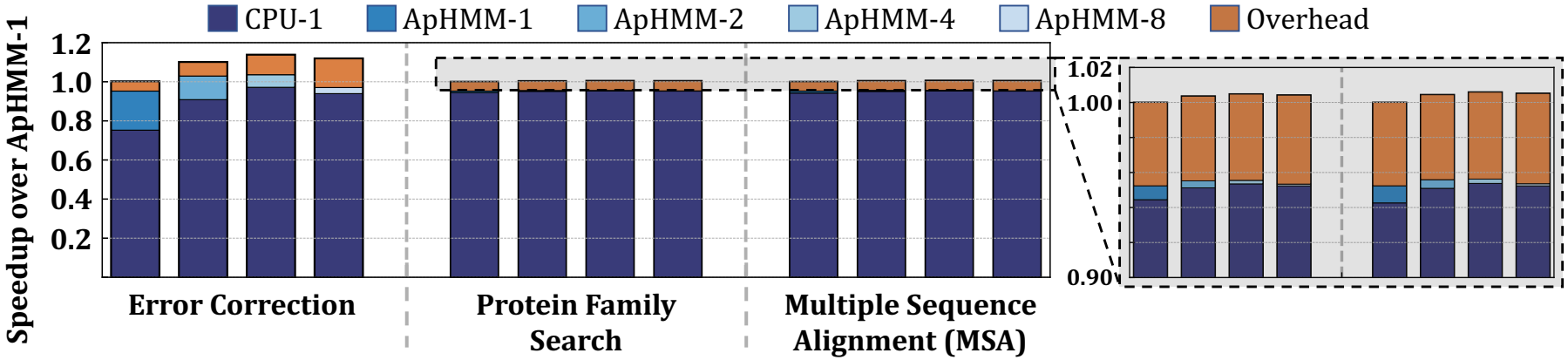


Results: Number of ApHMM Cores

4 ApHMM-cores provides the best overall speedup

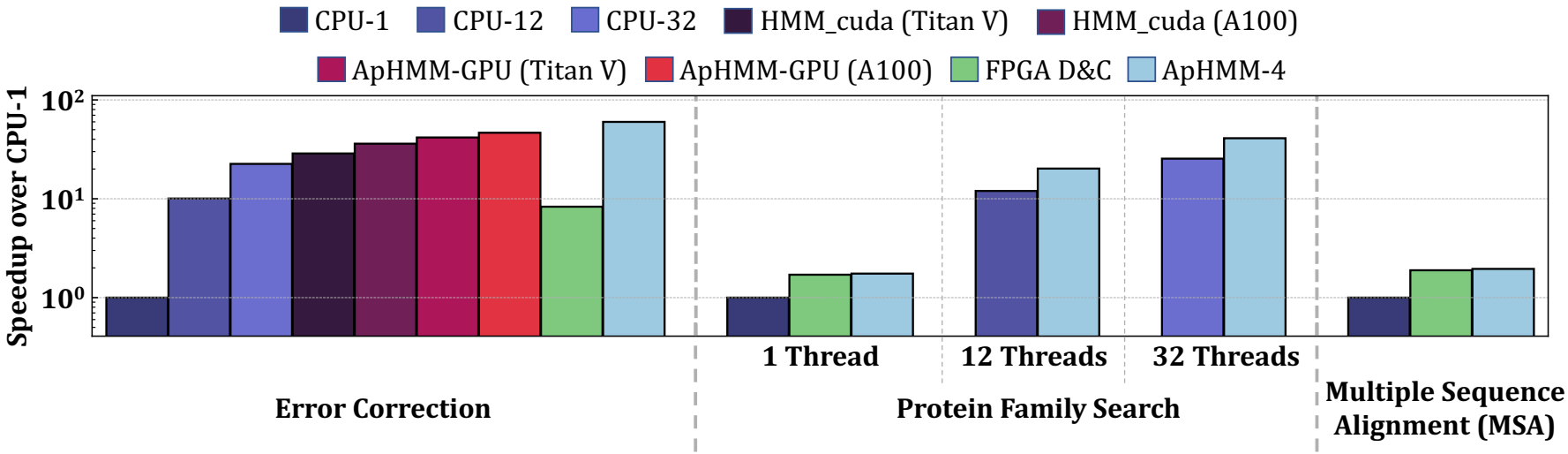
Data movement overhead starts becoming the bottleneck as we increase the number of cores

Room for the performance improvement by placing ApHMM **inside or near the memory**



Results: Error Correction

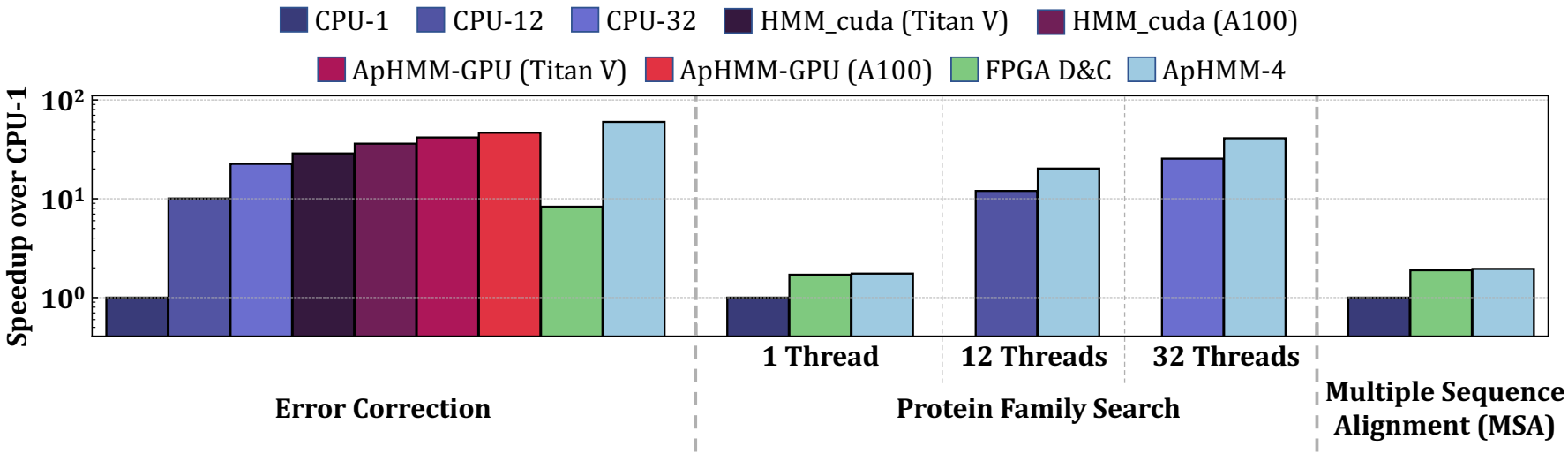
Acceleration where it makes sense: Significant performance and energy improvements for **error correction**



Results: Protein Family Search

Speedup by 1.61×- 1.75× (CPU) and 1.03× (FPGA)

Lower speedup ratio than error correction due to:
1) Smaller portion is accelerated (45.76%)
2) Increased DRAM access due to larger alphabet size

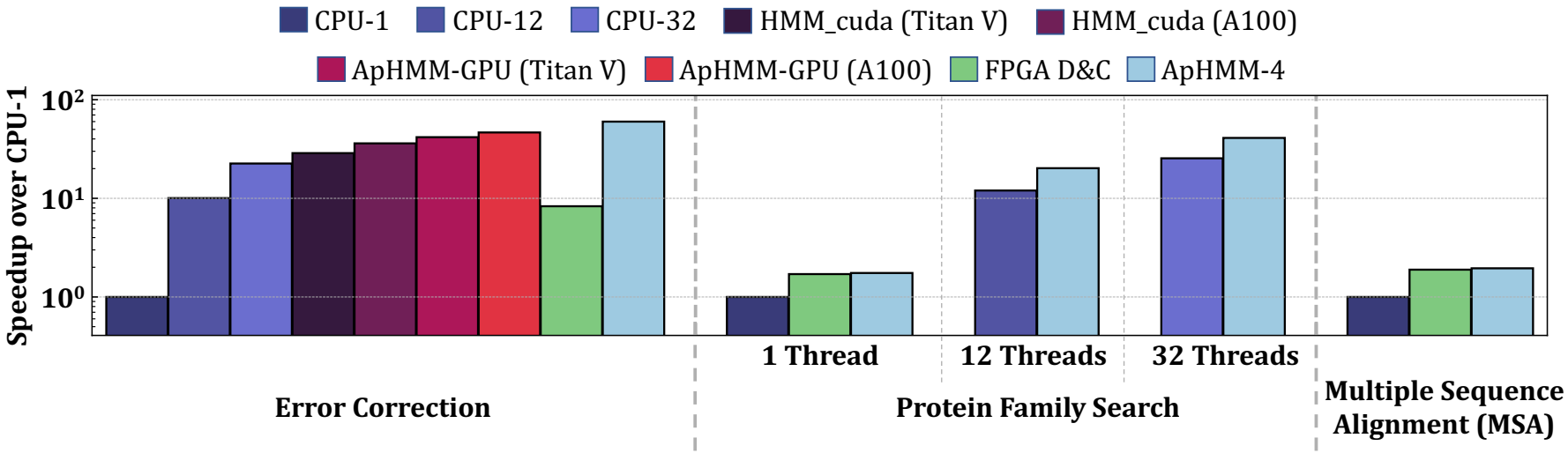


Results: Multiple Sequence Alignment

Speedup by 1.95× (CPU) and 1.03× (FPGA)

Better speedup ratio than protein family search:
Larger portion is accelerated (51.44%)

ApHMM provides better implementation than FPGA D&C even if we ignore its data movement overheads



Conclusion

- Accelerating the Baum-Welch algorithm provides **substantial performance improvements and energy reductions**
- **HW-SW co-design** provides the best overall performance
- Data movement becomes the bottleneck especially for Forward calculation
  - In-memory or near-memory solutions
- Our GPU implementation is the **first GPU implementation** of the Baum-Welch algorithm designed **for pHMMs**

Hardware Acceleration for pHMMs

- Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joel Lindegger, Mohammed Alser, Juan Gómez-Luna, Sreenivas Subramoney, and Onur Mutlu, **"ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis"**
  arXiv, 2022.
  [Source Code – will be public soon at http://github.com/CMU-SAFARI]

ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis

Can Firtina[1] Kamlesh Pillai[2] Gurpreet S. Kalsi[2] Bharathwaj Suresh[2] Damla Senol Cali[3]
Jeremie S. Kim[1] Taha Shahroodi[4] Meryem Banu Cavlak[1] Joel Lindegger[1] Mohammed Alser[1]
Juan Gómez Luna[1] Sreenivas Subramoney[2] Onur Mutlu[1]
[1]*ETH Zurich* [2]*Intel Labs* [3]*Bionano Genomics* [4]*TU Delft*

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 ▾**

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

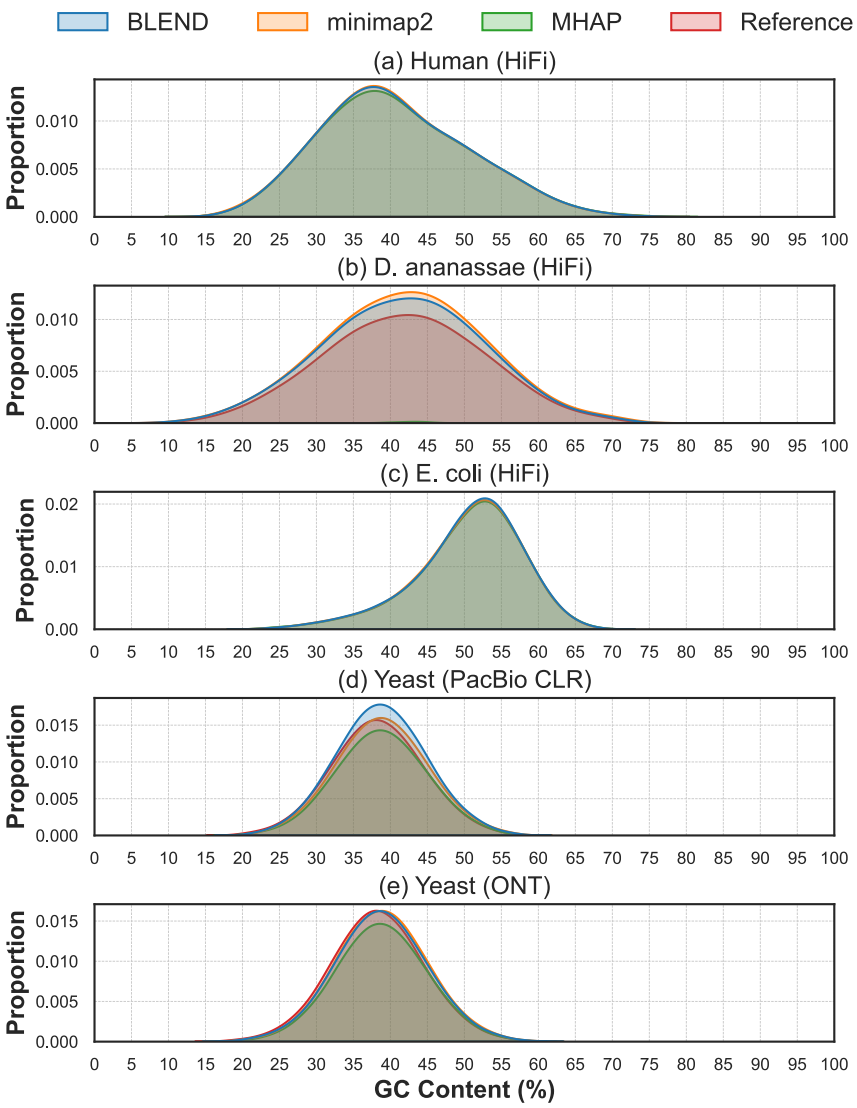
Can Firtina
canfirtina@gmail.com

27 May 2022
Invited Seminar Talk at UC Berkeley

SAFARI ETH zürich

Backup Slides

BLEND – GC Content of Assemblies



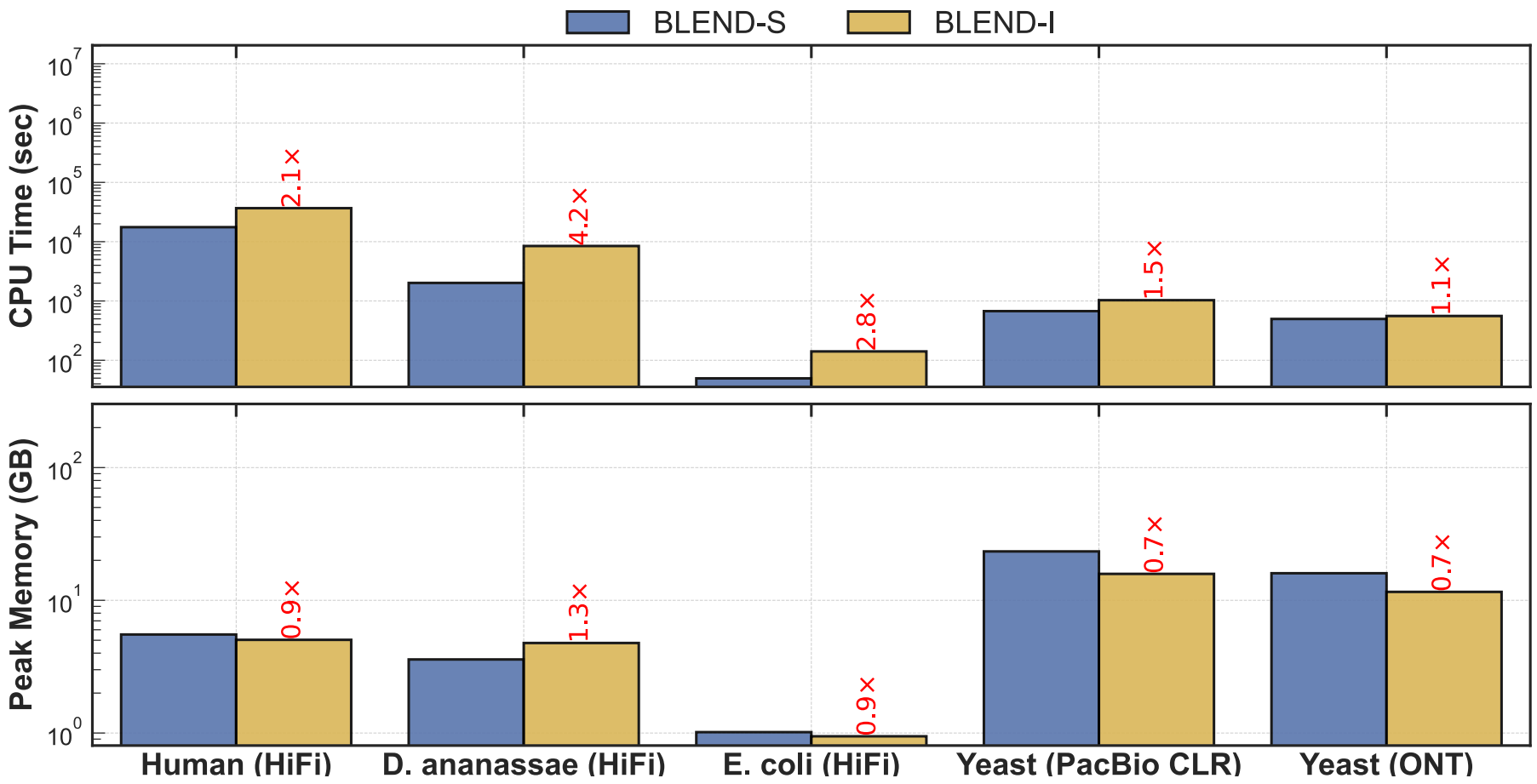
BLEND – Read Mapping Quality

Dataset	Tool	Mean Depth of Cov. (×)	Breadth of Coverage (%)	Aligned Reads (#)	Properly Paired (%)
Human CHM13	BLEND	**16.58**	**99.99**	3,171,916	NA
	minimap2	**16.58**	**99.99**	**3,172,261**	NA
	LRA	16.37	99.06	3,137,631	NA
	Winnowmap2	**16.58**	**99.99**	3,171,313	NA
D. ananassae	BLEND	57.37	99.66	1,223,388	NA
	minimap2	**57.57**	**99.67**	1,245,931	NA
	LRA	57.06	99.60	1,235,098	NA
	Winnowmap2	57.40	99.66	**1,249,575**	NA
E. coli	BLEND	**99.14**	99.90	39,048	NA
	minimap2	**99.14**	99.90	**39,065**	NA
	LRA	99.10	99.90	39,063	NA
	Winnowmap2	**99.14**	99.90	39,036	NA
Yeast (PacBio)	BLEND	195.84	**99.98**	269,804	NA
	minimap2	**195.86**	**99.98**	**269,935**	NA
	LRA	194.65	99.97	267,399	NA
	Winnowmap2	192.35	**99.98**	259,073	NA
Yeast (ONT)	BLEND	97.86	**99.97**	134,721	NA
	minimap2	**97.88**	99.96	**134,885**	NA
	LRA	97.25	99.95	132,862	NA
	Winnowmap2	97.04	99.96	130,978	NA
Yeast (Illumina)	BLEND	**79.93**	99.97	**6,494,489**	95.89
	minimap2	79.91	99.97	6,492,994	95.89

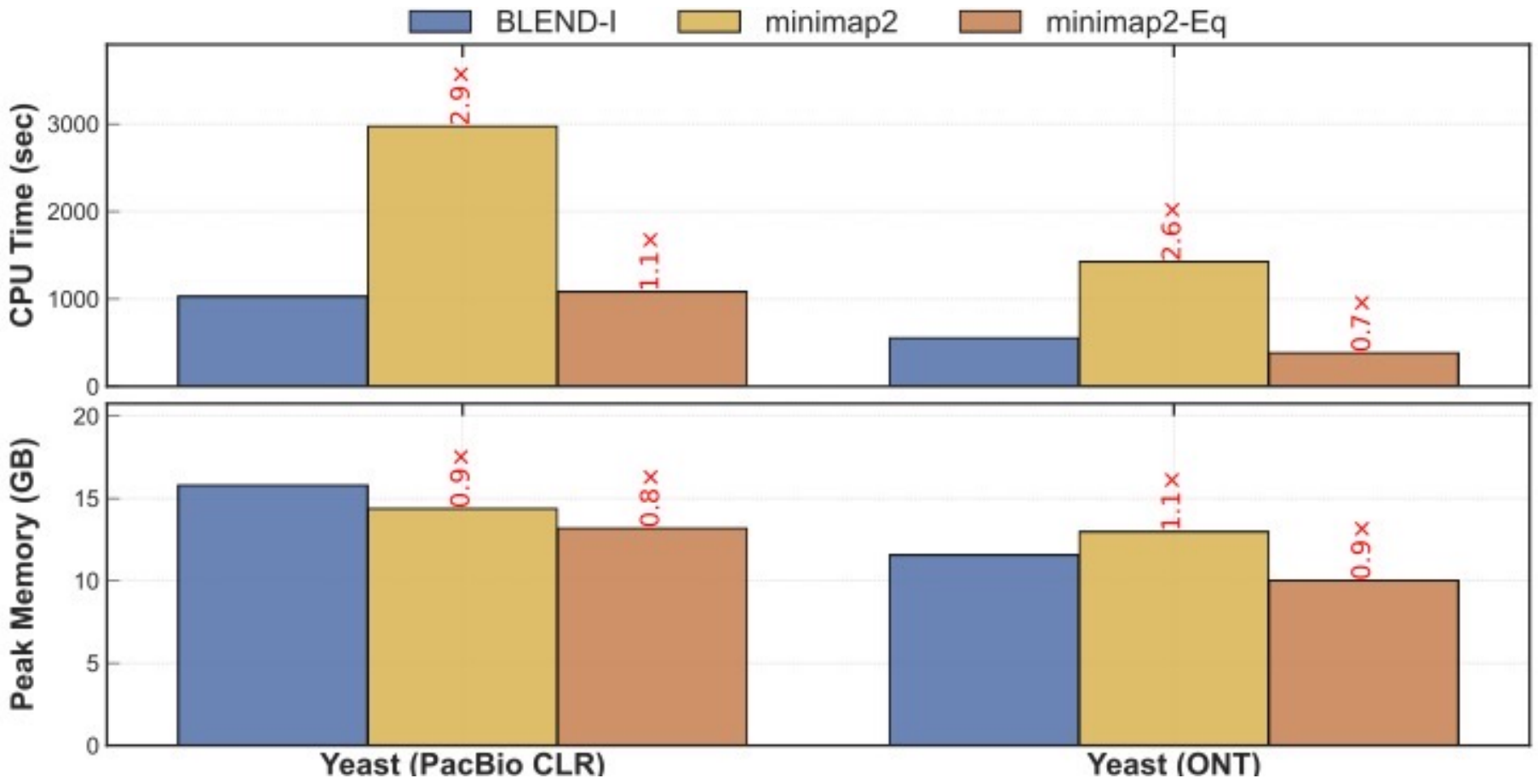
BLEND – Read Mapping Accuracy

Dataset	Tool	Error Rate (%)	High Quality True Mappings (#)
Yeast (PacBio)	BLEND	0.2524054	**266,382**
	minimap2	0.2504307	265,709
	LRA	99.9958863	11
	Winnowmap2	**0.2474206**	247,024
	S-conLSH	97.2181306	3,822
Yeast (ONT)	BLEND	**0.2404970**	133,228
	minimap2	0.2468770	**133,784**
	LRA	99.9954840	5
	Winnowmap2	0.2534777	126,899
	S-conLSH	96.6753628	2,361

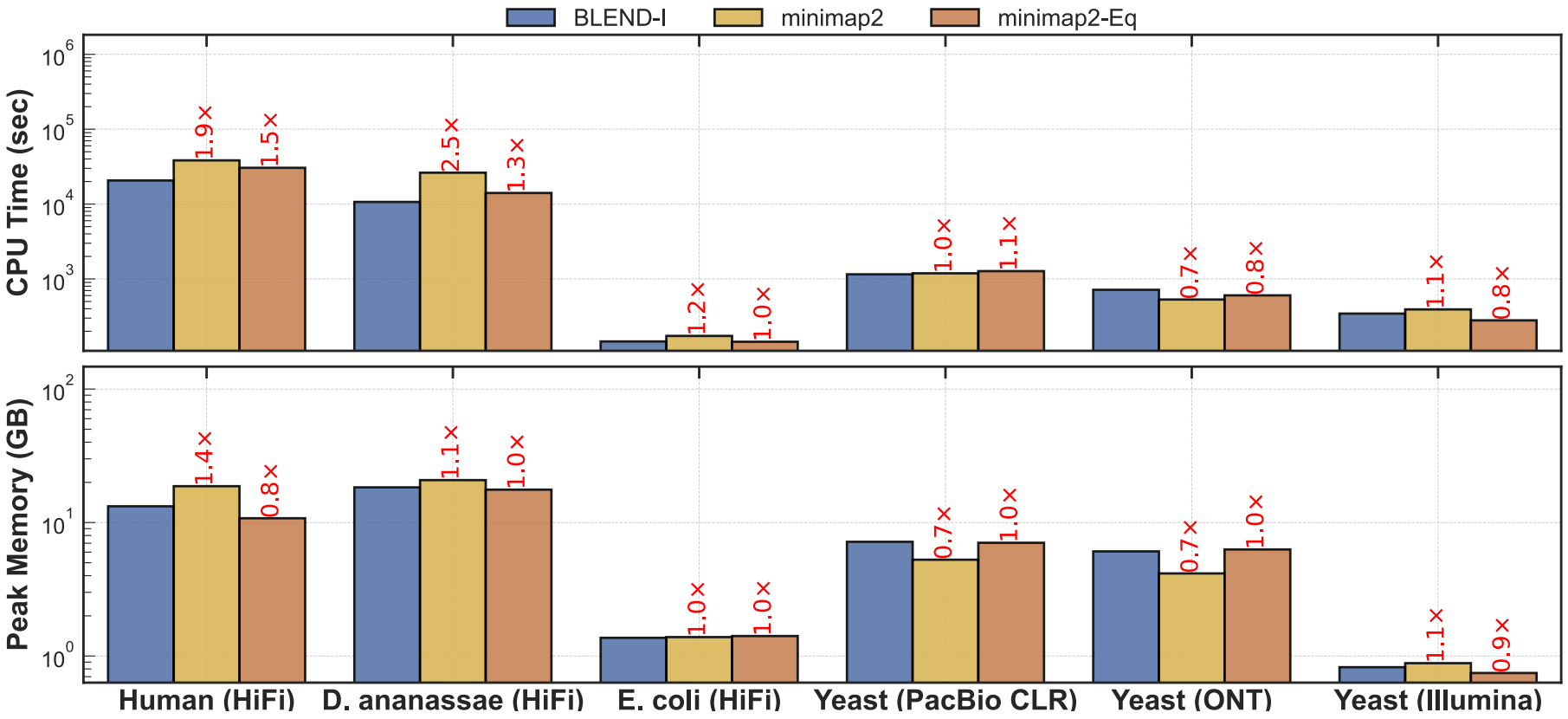
BLEND – *BLEND-I* vs. *BLEND-S*



BLEND – Equal Parameters



BLEND – Equal Parameters

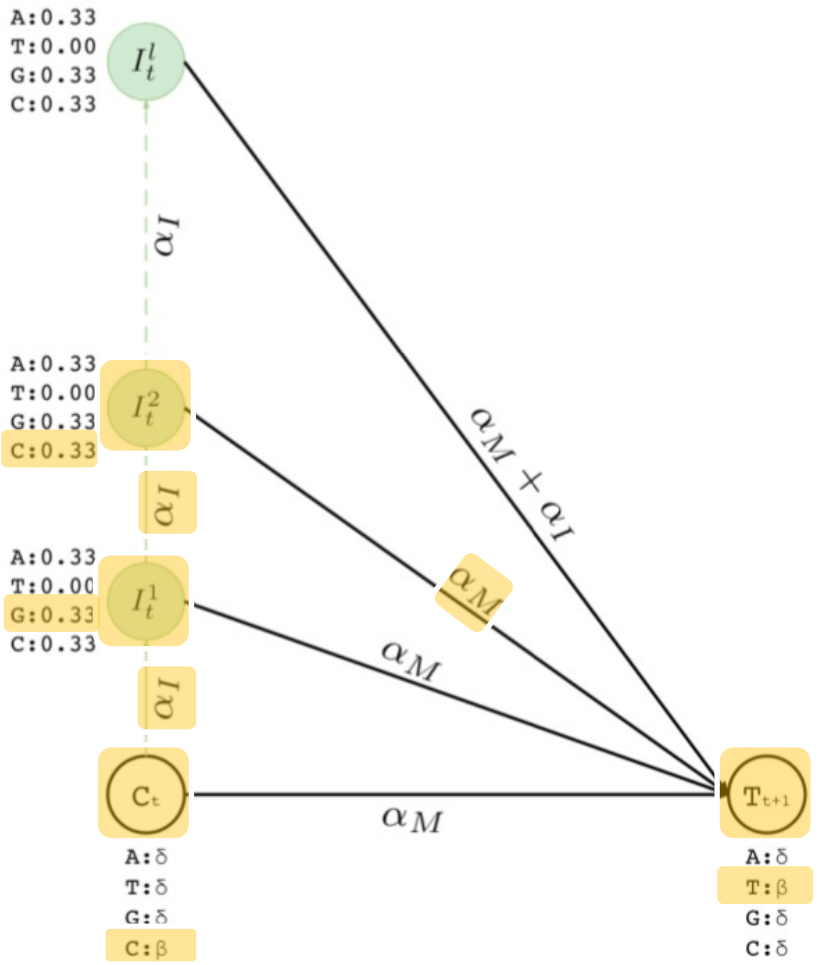


BLEND – Equal Parameters

Dataset	Tool	Mean Depth of Cov. (×)	Breadth of Coverage (%)	Aligned Reads (#)	Properly Paired (%)
Human CHM13	BLEND-I	16.58	99.99	**3,172,313**	NA
	minimap2	16.58	99.99	3,172,261	NA
	minimap2-Eq	16.58	99.99	3,172,312	NA
D. ananassae	BLEND-I	57.50	99.65	1,249,731	NA
	minimap2	**57.57**	**99.67**	1,245,931	NA
	minimap2-Eq	57.49	99.65	**1,250,816**	NA
E. coli	BLEND-I	99.14	99.90	39,065	NA
	minimap2	99.14	99.90	39,065	NA
	minimap2-Eq	99.14	99.90	39,065	NA
Yeast (PacBio)	BLEND-I	195.84	99.98	269,804	NA
	minimap2	**195.86**	99.98	**269,935**	NA
	minimap2-Eq	195.81	99.98	269,493	NA
Yeast (ONT)	BLEND-I	97.86	**99.97**	134,721	NA
	minimap2	**97.88**	99.96	**134,885**	NA
	minimap2-Eq	97.82	99.96	134,578	NA
Yeast (Illumina)	BLEND-I	**79.93**	99.97	**6,494,489**	**95.89**
	minimap2	79.91	99.97	6,492,994	**95.89**
	minimap2-Eq	79.83	99.97	6,485,540	95.72

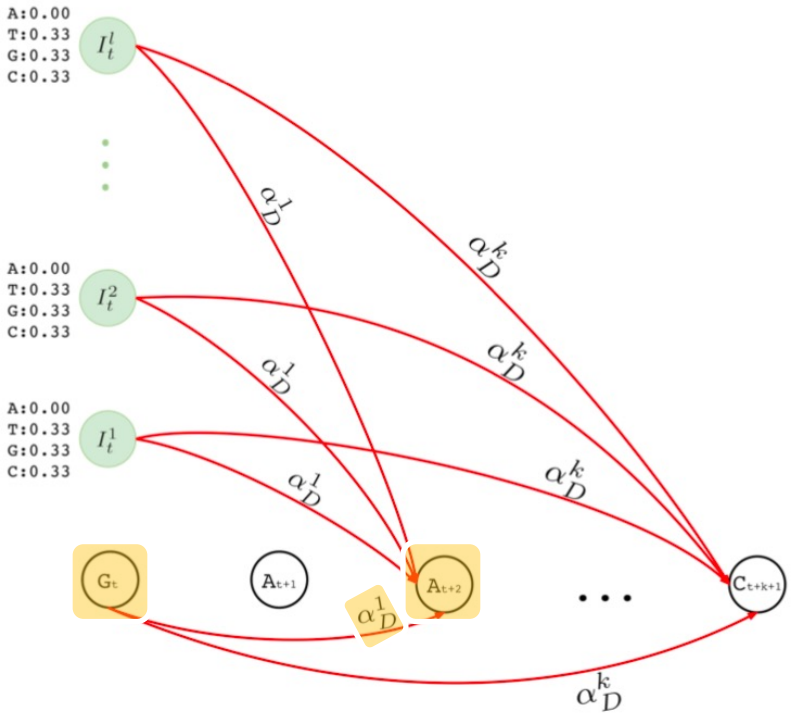
Resolving deletion errors

- 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



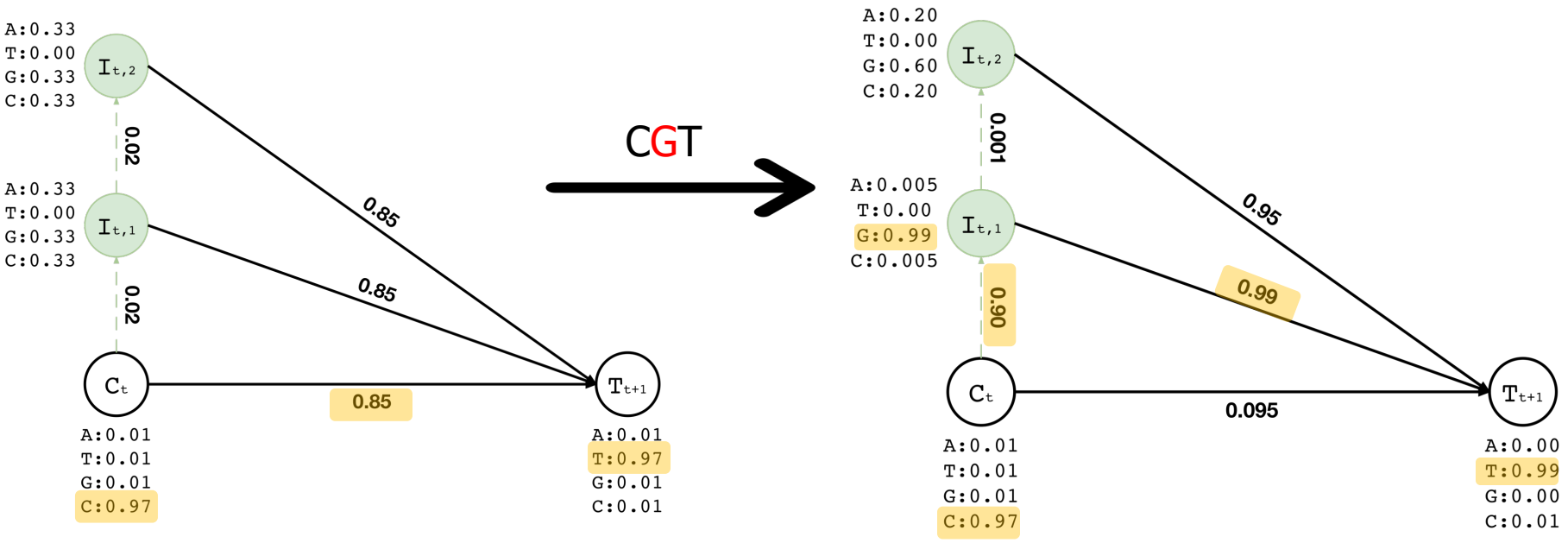
Resolving insertion errors

- 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



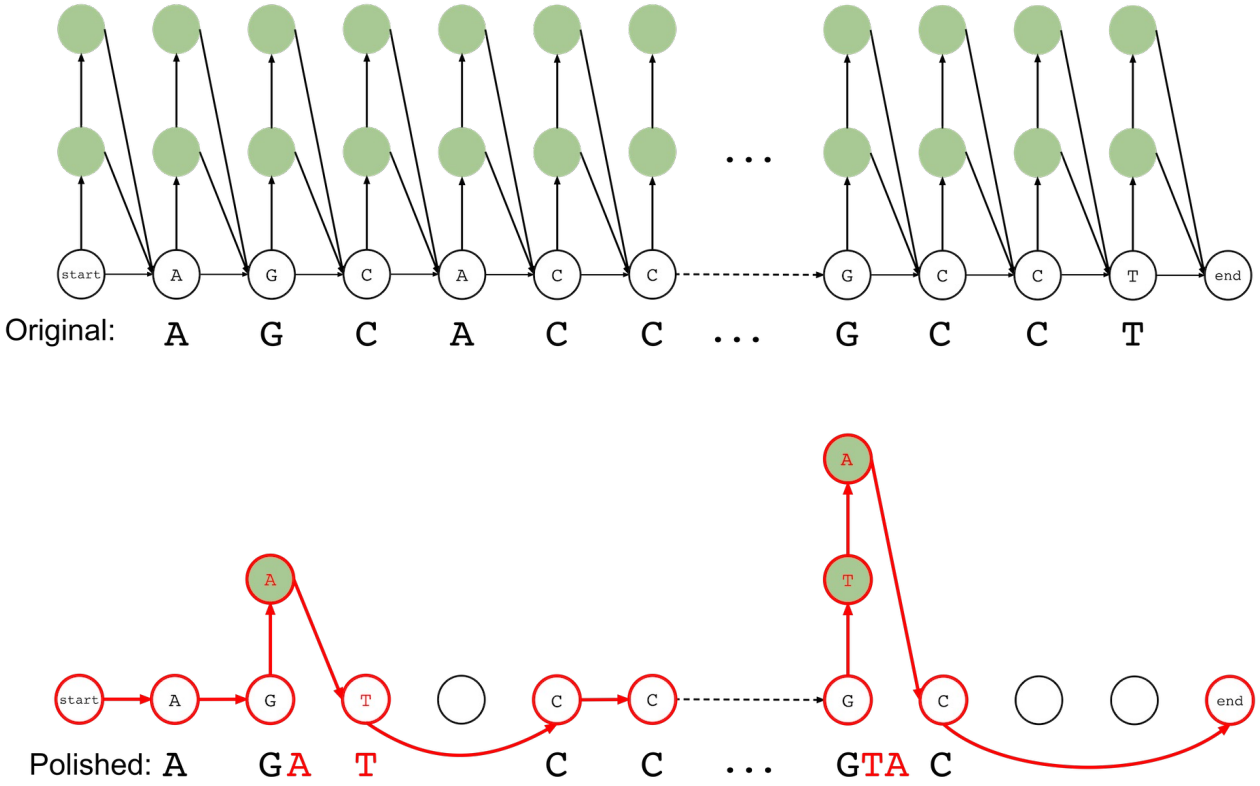
Training

- 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**



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"



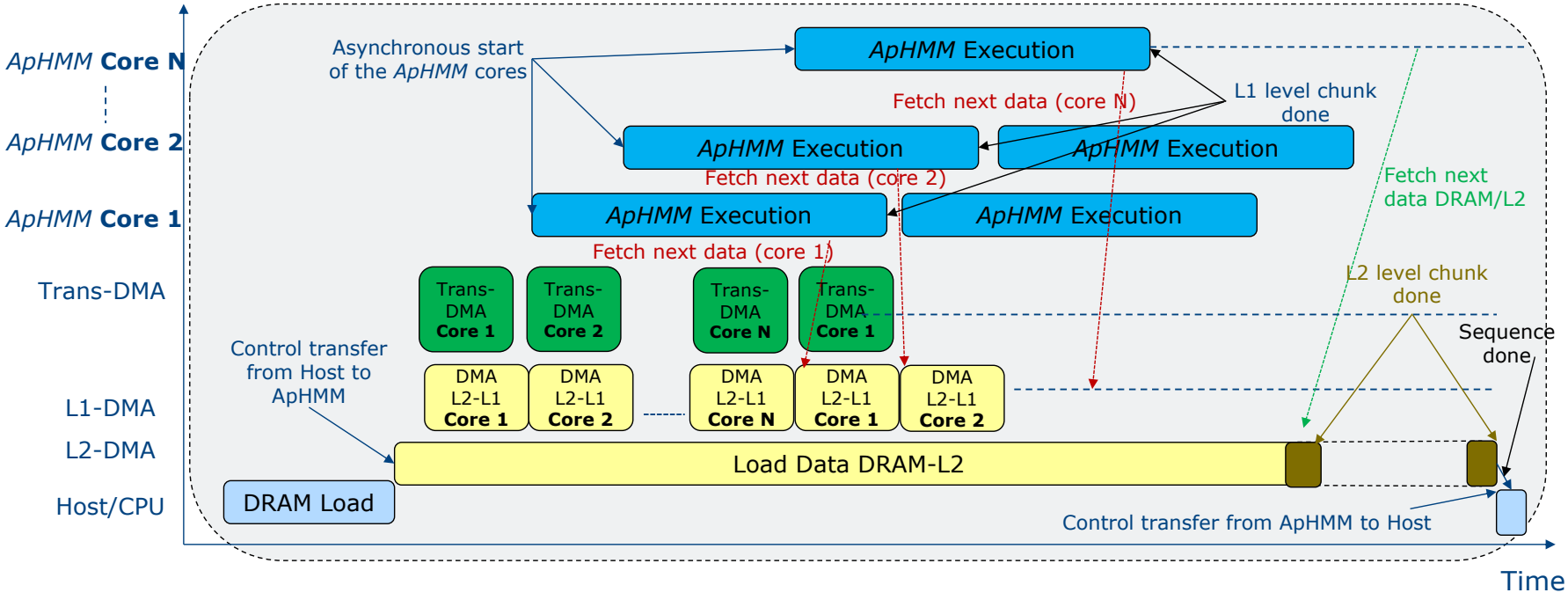
Apollo -- Data Sets

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)

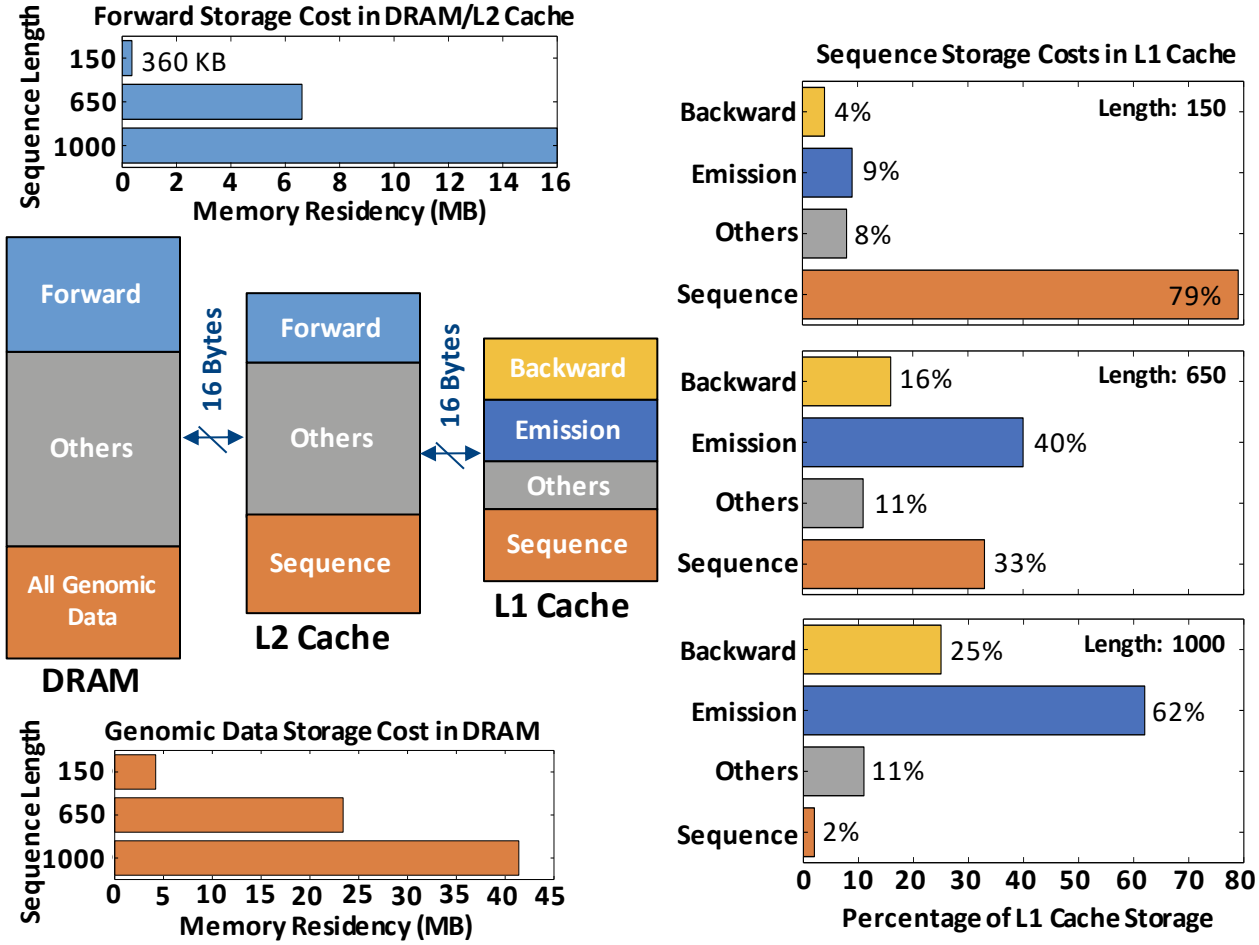
Apollo -- Experimental Setup

- 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

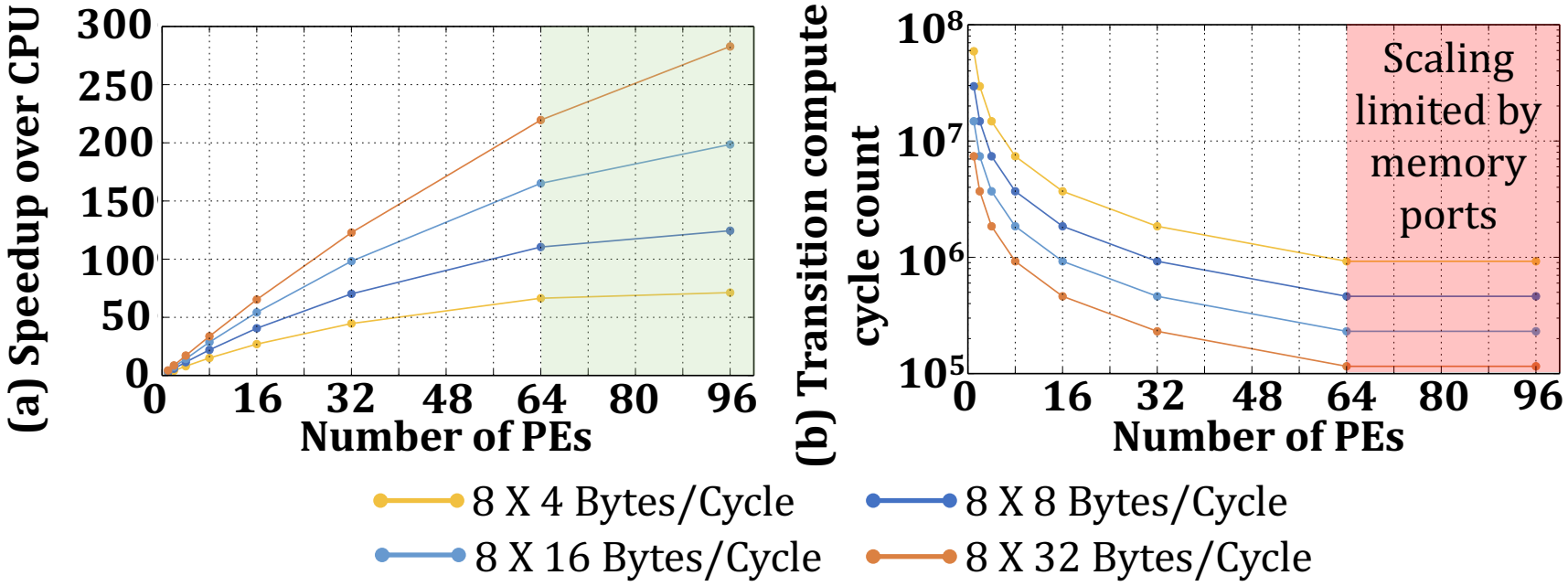
ApHMM – Execution Flow



ApHMM – Storage Distribution



ApHMM – Hardware Configuration



Memory	Memory BW (Bytes/cycle): 16, Memory Ports (#): 8 L1 Cache Size: 128KB
Processing Engine	PEs (#): 64, Multipliers per PE (#): 4, Adders per PE (#): 4 Memory per PE: 8, Transition Pipes (#): 64, Emission Pipes (#): 4

ApHMM – Area and Power Breakdown

Module Name	**Area (mm^2)**	**Power (mW)**
Control Logic	0.011	134.4
64 Processing Engines	1.333	304.2
64 Transition Pipelines	5.097	0.8
4 Emission Pipelines	0.094	70.4
Overall	6.536	509.8
128KB L1-Memory	0.632	100

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]

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

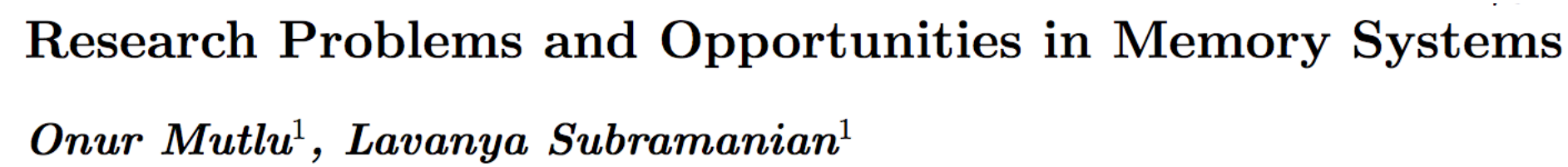
ONUR MUTLU
ETH Zürich and Carnegie Mellon University

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.

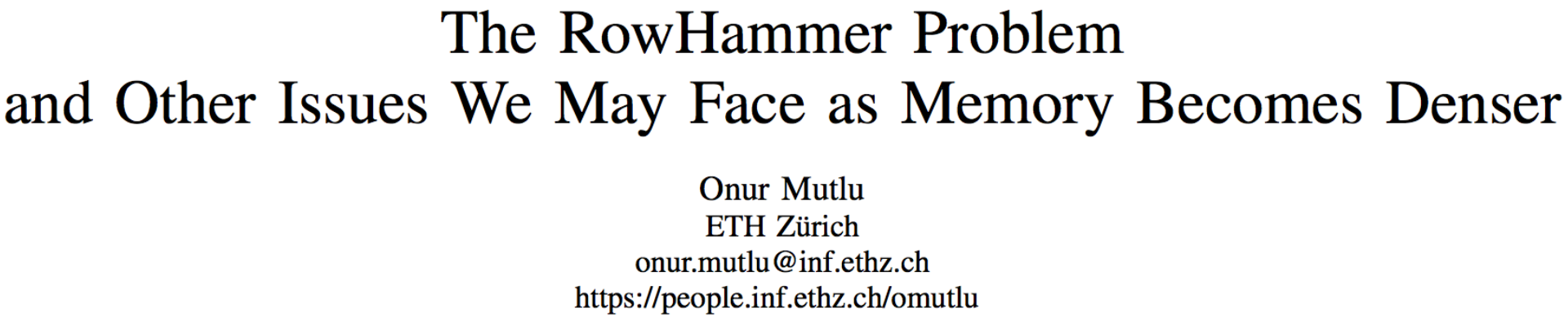

Research Problems and Opportunities in Memory Systems

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

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

Reference Overview Paper V

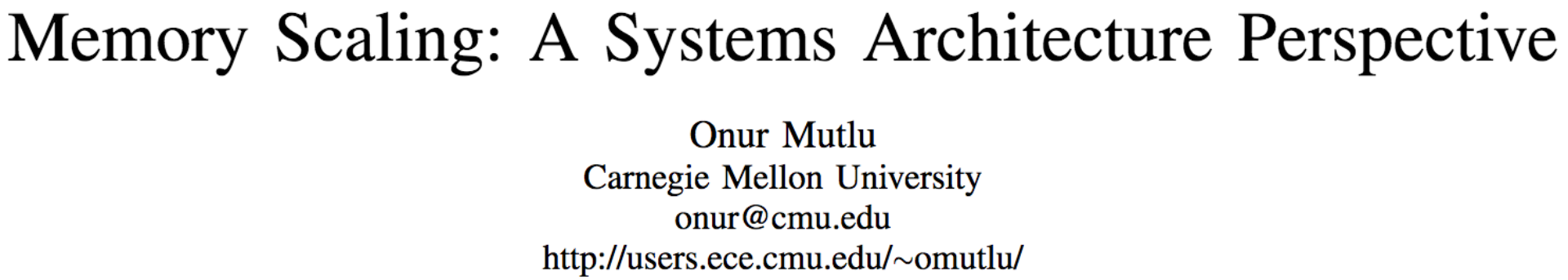
- Onur Mutlu,
  "The RowHammer Problem and Other Issues We May Face as Memory Becomes Denser"
  *Invited Paper in Proceedings of the Design, Automation, and Test in Europe Conference (**DATE**)*, Lausanne, Switzerland, March 2017.
  [Slides (pptx) (pdf)]


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

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/

Reference Overview Paper VII

Proceedings of the IEEE, Sept. 2017

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

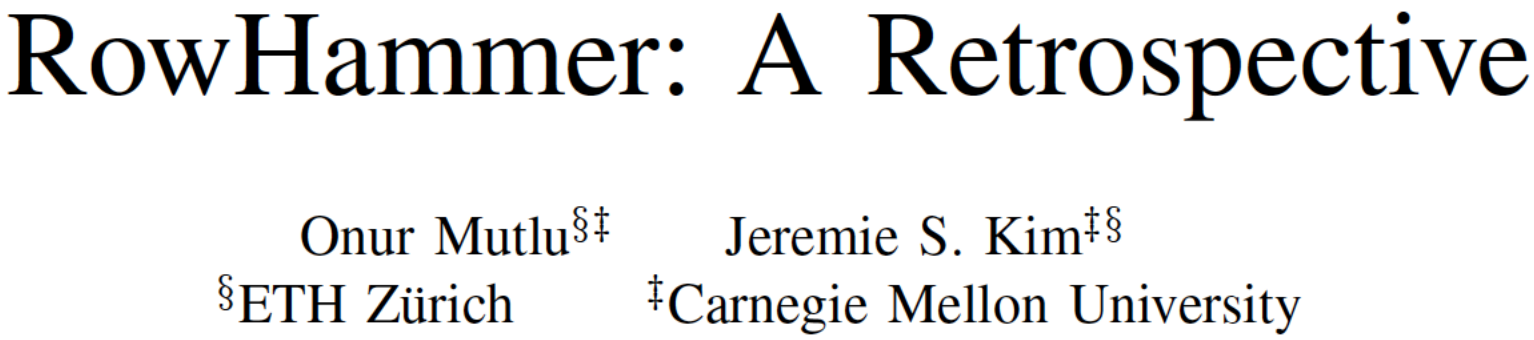
This paper reviews the most recent advances in solid-state drive (SSD) error characterization, mitigation, and data recovery techniques to improve both SSD's reliability and lifetime.

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**

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