Enabling Fast, Accurate & Efficient Real-Time Genome Analysis via New Algorithms and Architectures

Can Firtina
canfirtina@gmail.com
https://cfirtina.com

16 January 2024

Technical University of Munich (TUM)
School of Computation, Information and Technology

SAFARI

ETH zürich

Brief Self Introduction

- **Can Firtina**
  - Ph.D. Student in SAFARI Research Group led by Prof. Onur Mutlu

- **Research interests:** Bioinformatics & Computer Architecture
  - Real-time genome analysis
  - Similarity search in a large space of genomic data
  - Hardware-Algorithm co-design to accelerate genome analysis
  - Genome editing
  - Error correction

- Get to know **our group and our research**
  - **Group website:** https://safari.ethz.ch/
  - **Contact me:** canfirtina@gmail.com
  - **Website:** https://cfirtina.com
  - **Twitter (aka X):** https://twitter.com/FirtinaC

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
  - …

SAFARI Research Group

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

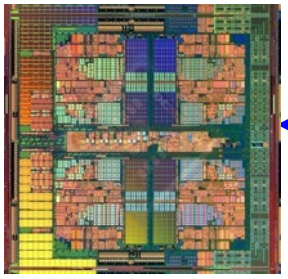
SAFARI
SAFARI Research Group
safari.ethz.ch

40+ Researchers

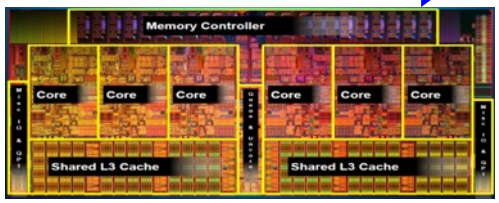
Think BIG, Aim HIGH!

Current Research Mission

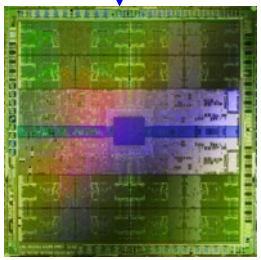
Computer architecture, HW/SW, systems, bioinformatics, security



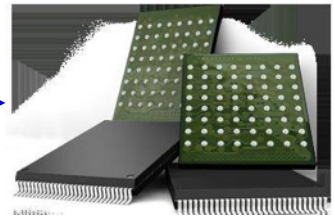
Heterogeneous Processors and Accelerators



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

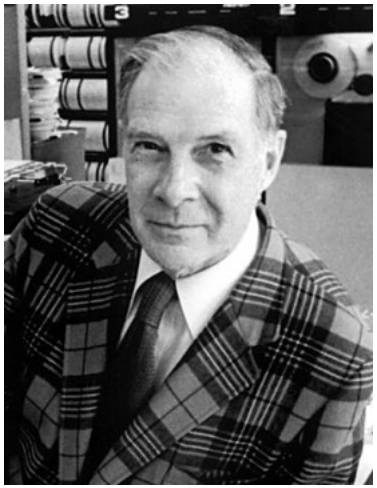
Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
  - Intelligent genome analysis
- Enabling Fast and Accurate Real-time Analysis
  - RawHash and RawHash2
- Graph & ML Acceleration in Genomics
  - ApHMM
- Conclusion

The Goal of Computing: Beyond Numbers

"The purpose of **computing** is [to gain] **insight**, not numbers"

Richard Hamming

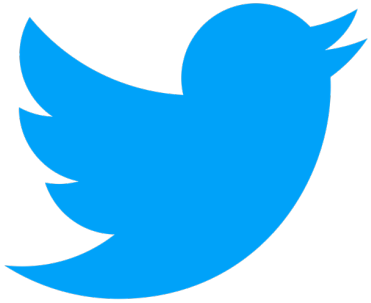


"Numerical Methods for Scientists and Engineers," Richard Hamming, 1962.

We need to gain insights
and observations
much more efficiently
than ever before

Big Data is Everywhere

Astronomy
25 zetta-bytes/year



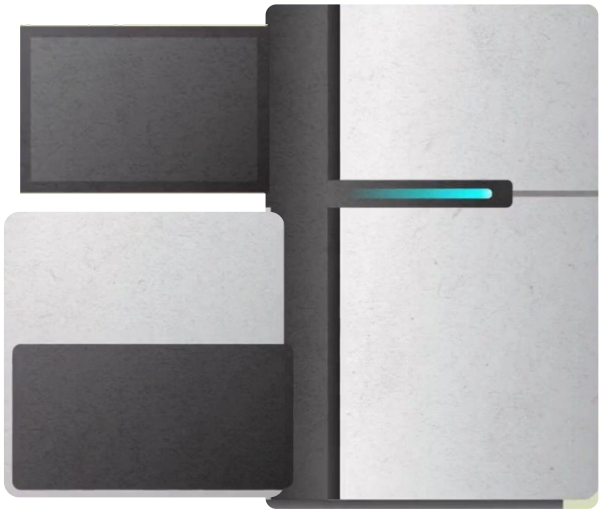
Twitter (now X)
0.5-15 billion tweets/year

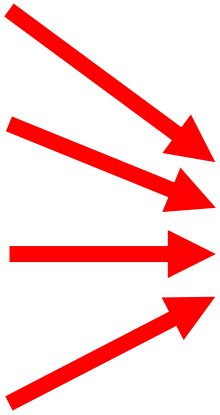
YouTube
500-900 million hours/year

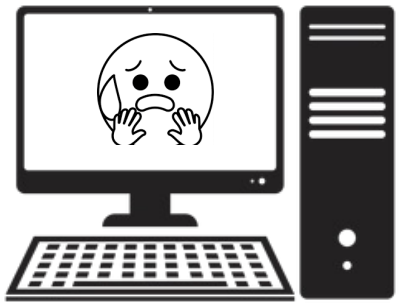
Genomics
1 zetta-bases/year

"Big data: astronomical or genomical?", PLoS biology, 2015.

Problems with Data Analysis Today







Special-Purpose Machine
for **Data Generation**

General-Purpose Machine
for **Data Analysis**

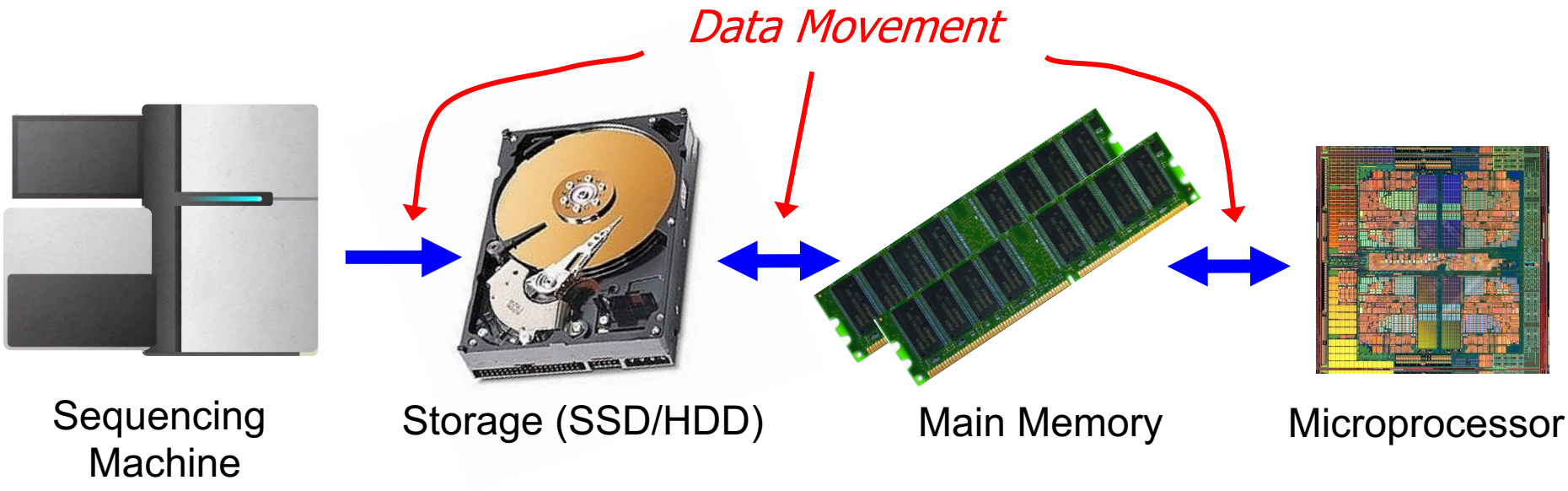
FAST

SLOW

Slow and inefficient processing capability
Large amounts of data movement

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

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

Does intelligent genome analysis really matter?

Intelligent Genome Analysis

Mohammed Alser, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu
"From Molecules to Genomic Variations: Intelligent Algorithms and Architectures for Intelligent Genome Analysis"
Computational and Structural Biotechnology Journal, 2022
[Source code]

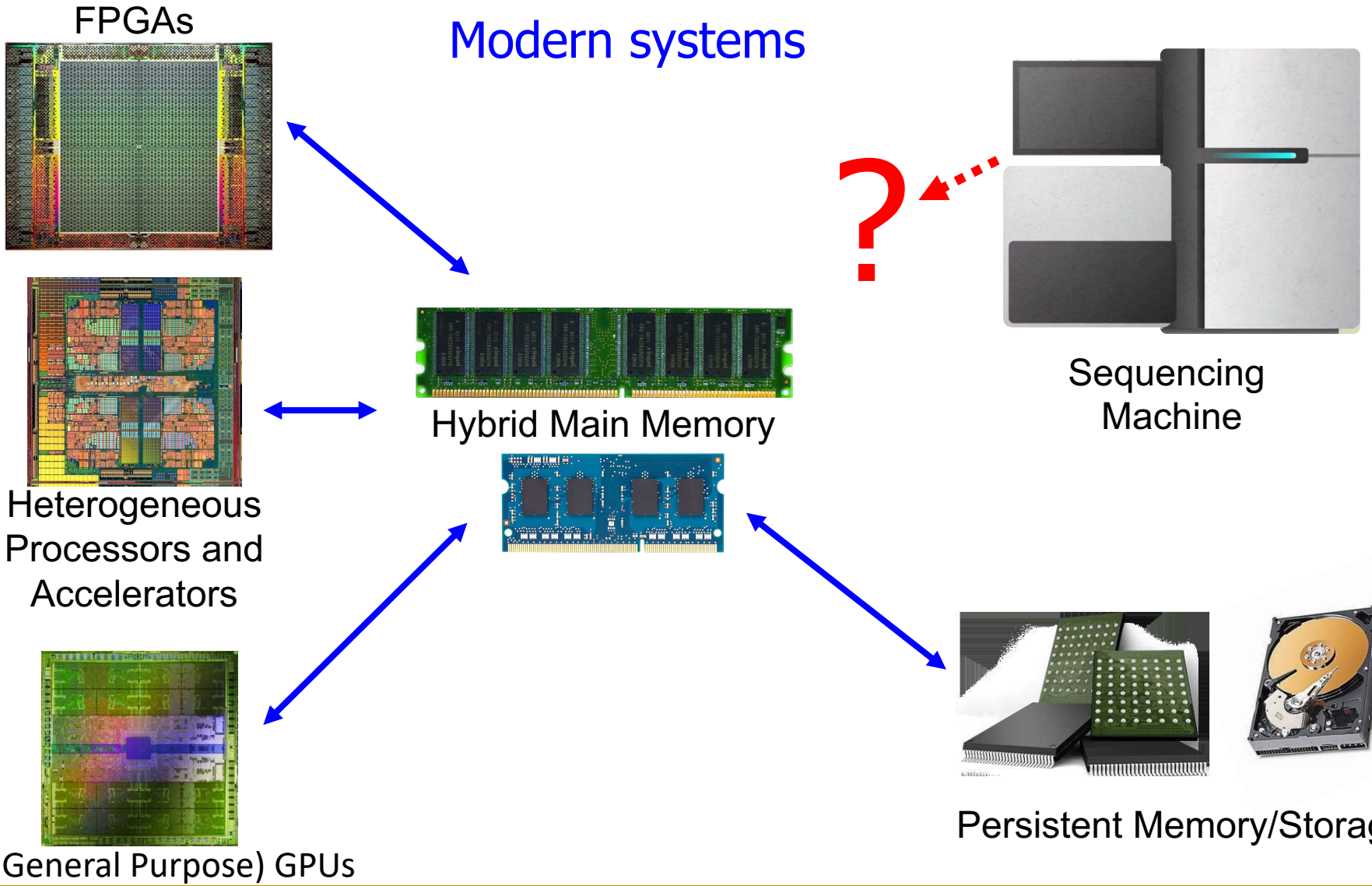
Review

From molecules to genomic variations: Accelerating genome analysis via intelligent algorithms and architectures

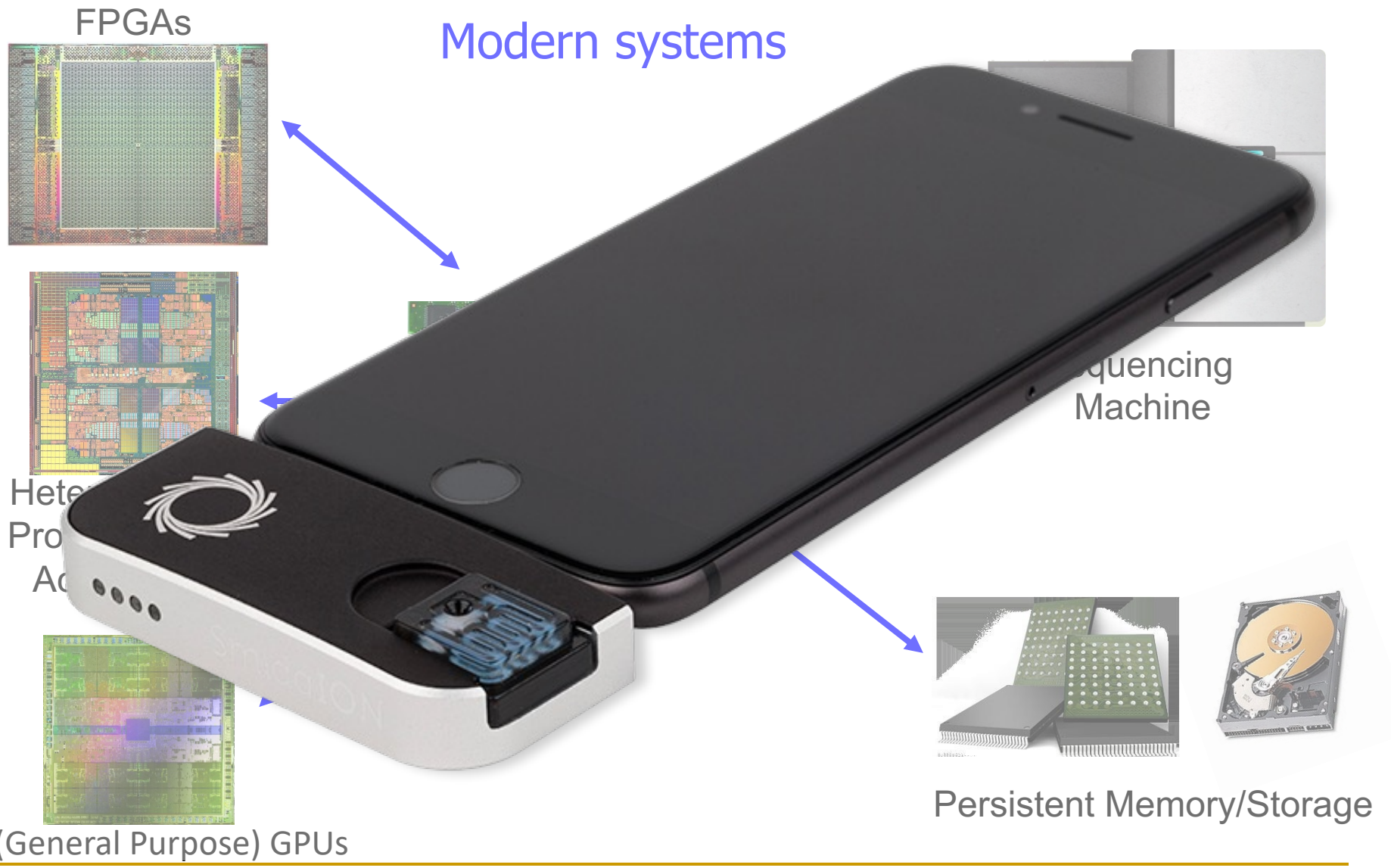
Mohammed Alser *, Joel Lindegger, Can Firtina, Nour Almadhoun, Haiyu Mao, Gagandeep Singh, Juan Gomez-Luna, Onur Mutlu *

ETH Zurich, Gloriastrasse 35, 8092 Zürich, Switzerland

Pushing Towards New Architectures



Pushing Towards New Architectures



https://nanoporetech.com/products/smidgion

Fast Genome Analysis

Fast genome analysis

in mere **seconds**

using **limited** computational resources

(e.g., a mobile device).

Accurate Genome Analysis

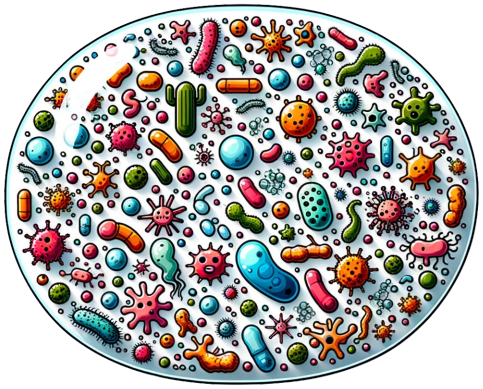
Accurate genome analysis

to **make life-critical decisions**

and improving the quality of life

Faster, Scalable & Accurate Genome Analysis

Understanding **genetic variations, species, and evolution**



Predicting the **presence of pathogens** in an environment

Surveillance of **disease outbreaks**

Personalized medicine

And, many, many other applications …

Personalized Medicine in UK

"From 2019, all seriously ill children in UK will be offered whole genome sequencing as part of their care"

NHS
National Institute for Health Research

Rapid Surveillance of Disease Outbreaks

Real-time, portable genome sequencing for Ebola surveillance

Figure 1: Deployment of the portable genome surveillance system in Guinea.

Subscribe Sign In

University spinout's portable DNA sequencer has proved invaluable in tracking the global spread of coronavirus

Scalable SARS-CoV-2 Testing

nature biomedical engineering

Article | Published: 01 July 2021

Massively scaled-up testing for SARS-CoV-2 RNA via next-generation sequencing of pooled and barcoded nasal and saliva samples

Joshua S. Bloom ✉, Laila Sathe, [...] Valerie A. Arboleda ✉

Nature Biomedical Engineering **5**, 657–665 (2021) | Cite this article

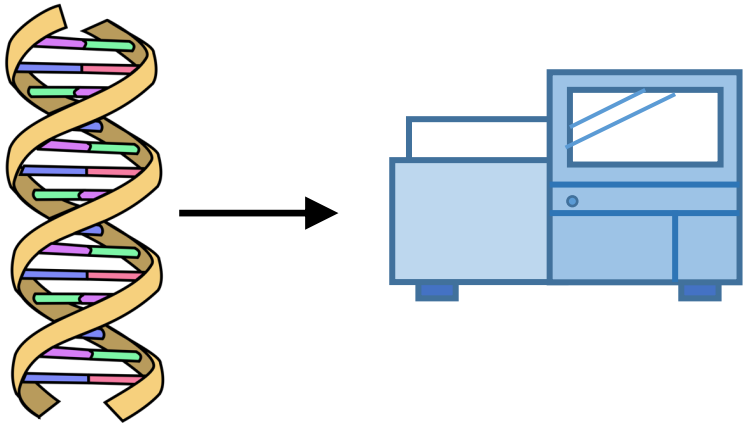
Bloom+, "Swab-Seq: A high-throughput platform for massively scaled up SARS-CoV-2 testing", *Nature Biomedical Engineering*, 2021

Large Scale Analysis

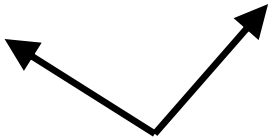
https://blog.wego.com/7-crowded-places-and-events-that-you-will-love/

Genome Analysis – How?

- **Genome sequencing machines** can quickly convert biological molecules
  - Into sequences of characters for analysis



Biological Molecule (e.g., DNA)

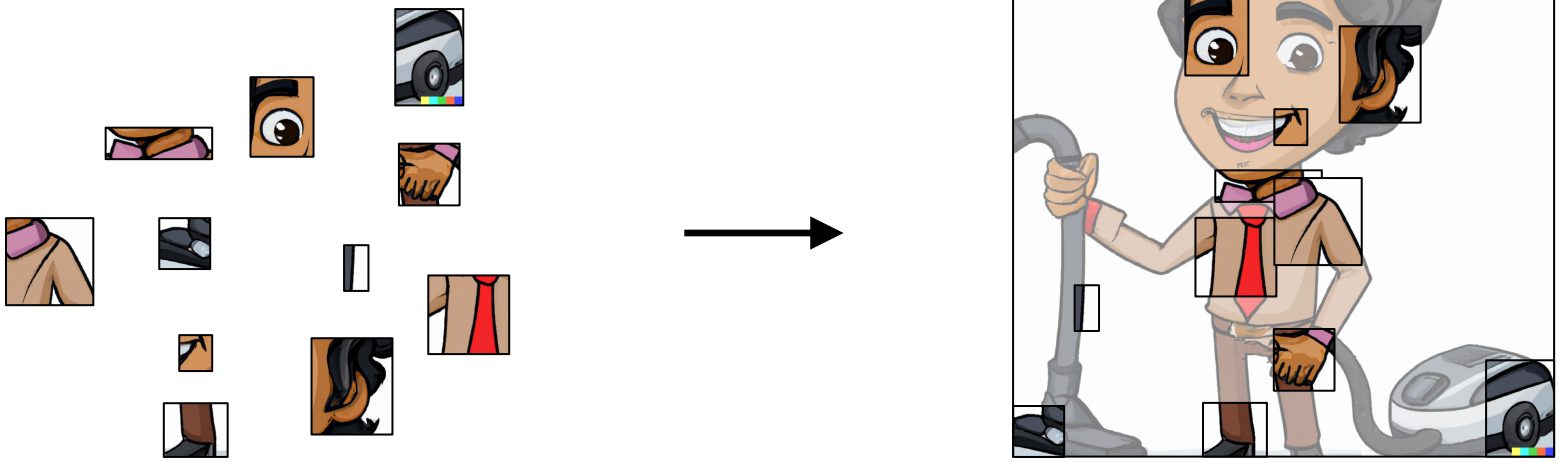


Sequences from DNA

Sequence Comparison is Essential

- Analyze sequences by accurately and quickly **comparing** them
  - To **each other**
  - To a **template sequence** representative of a species, a certain group…

Biological Sequences (e.g., DNA, proteins)



- Essential to understand functionality of a sequence, mutations, diseases…

Applications
are only limited
by our imagination

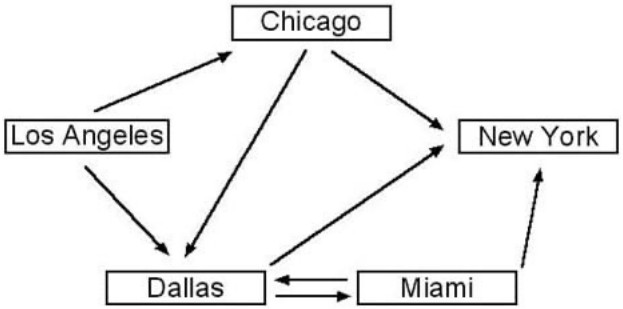
Genome Editing

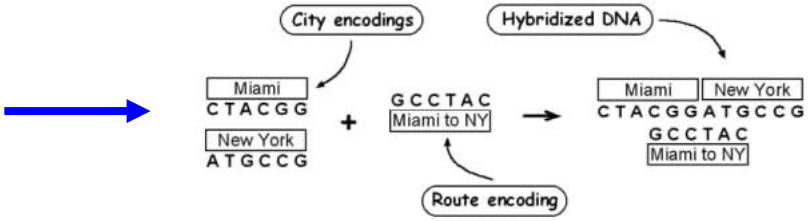
The Nobel Prize in Chemistry 2020

awarded "for the development of a method of genome editing"

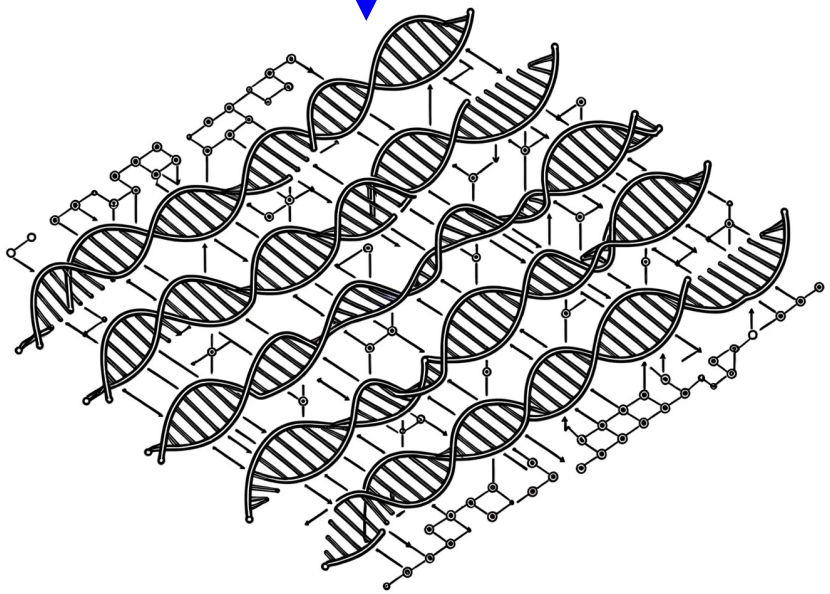
https://www.nobelprize.org/prizes/chemistry/2020/press-release/

DNA Computing





Massive parallelism to solve (hard) problems!



Accelerating Genome Analysis [DAC 2023]

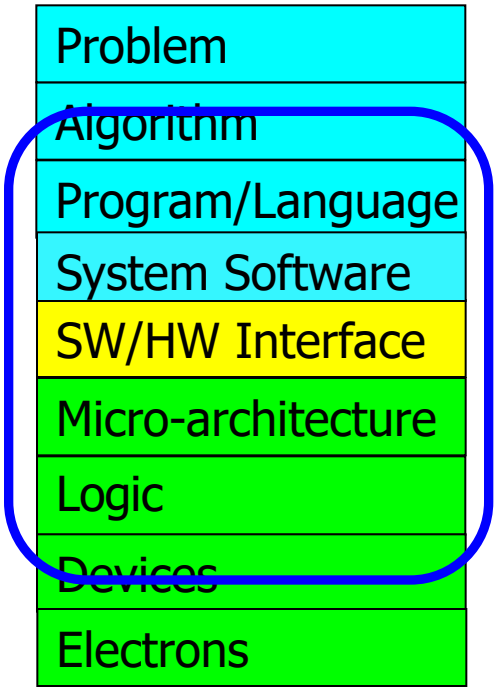
- Onur Mutlu and Can Firtina,
  "Accelerating Genome Analysis via Algorithm-Architecture Co-Design"
  *Invited Special Session Paper in Proceedings of the 60th Design Automation Conference (**DAC**)*, San Francisco, CA, USA, July 2023.
  [Slides (pptx) (pdf)]
  [Talk Video (38 minutes, including Q&A)]
  [Related Invited Paper]
  [arXiv version]

Accelerating Genome Analysis via Algorithm-Architecture Co-Design

Onur Mutlu Can Firtina
ETH Zürich

Algorithm-Arch-Device Co-Design is Critical

Computer Architecture (expanded view)



We need intelligent algorithms and intelligent architectures that handle data well

New Frontiers: Raw Signal Analysis [ISMB 2023]

- Can Firtina, Nika Mansouri Ghiasi, Joel Lindegger, Gagandeep Singh, Meryem Banu Cavlak, Haiyu Mao, and Onur Mutlu,
  "RawHash: Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes"
  *Proceedings of the 31st Annual Conference on Intelligent Systems for Molecular Biology (**ISMB**) and the 22nd European Conference on Computational Biology (**ECCB**)*, Jul 2023
  [Bioinformatics Journal version]
  [Slides (pptx) (pdf)]
  [RawHash Source Code]

Bioinformatics, 2023, **39**, i297–i307
https://doi.org/10.1093/bioinformatics/btad272
ISMB/ECCB 2023

OXFORD

RawHash: enabling fast and accurate real-time analysis of raw nanopore signals for large genomes

Can Firtina [1,*], Nika Mansouri Ghiasi [1], Joel Lindegger [1], Gagandeep Singh [1], Meryem Banu Cavlak [1], Haiyu Mao [1], Onur Mutlu [1,*]

[1]Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland

*Corresponding author. Department of Information Technology and Electrical Engineering, ETH Zurich, Gloriastrasse 35, 8092 Zurich, Switzerland.
E-mail: firtinac@ethz.ch (C.F.), omutlu@ethz.ch (O.M.)

Fast and Accurate Real-Time Genome Analysis

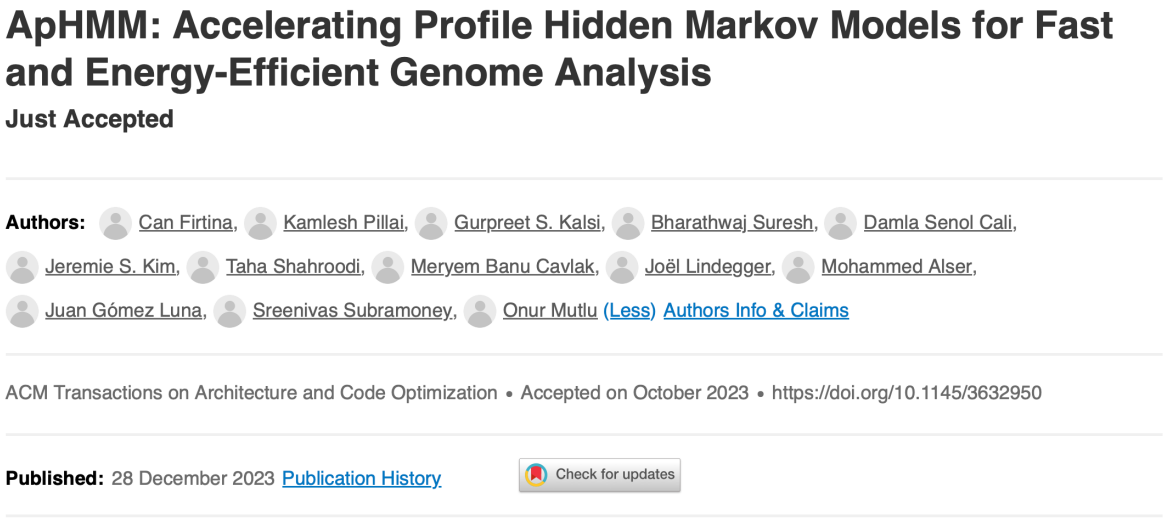
- Can Firtina, Melina Soysal, Joel Lindegger, and Onur Mutlu,
  "RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism"
  Preprint on ***arxiv***, September 2023.
  [arXiv version]
  [RawHash2 Source Code]

RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism

Can Firtina Melina Soysal Joel Lindegger Onur Mutlu
ETH Zürich

Accelerating ML & Genome Graphs [ACM TACO '23]

- Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, and Onur Mutlu,
  "ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis"
  ACM TACO, Dec 2023.
  [Online link at ACM TACO]
  [arXiv preprint]
  [ApHMM Source Code]



ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis

Just Accepted

Authors: Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, Onur Mutlu (Less) Authors Info & Claims

ACM Transactions on Architecture and Code Optimization • Accepted on October 2023 • https://doi.org/10.1145/3632950

Published: 28 December 2023 Publication History

Check for updates

Genome Similarity Identification [NARGAB 2023]

- 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 in Genome Analysis"
  NAR Genomics and Bioinformatics, March 2023.
  [Online link at NAR Genomics and Bioinformatics Journal]
  [arXiv preprint]
  [bioRxiv preprint]
  [BLEND Source Code]

Volume 5, Issue 1
March 2023

JOURNAL ARTICLE

BLEND: a fast, memory-efficient and accurate mechanism to find fuzzy seed matches in genome analysis

Can Firtina ✉, Jisung Park, Mohammed Alser, Jeremie S Kim, Damla Senol Cali, Taha Shahroodi, Nika Mansouri Ghiasi, Gagandeep Singh, Konstantinos Kanellopoulos, Can Alkan, Onur Mutlu ✉

NAR Genomics and Bioinformatics, Volume 5, Issue 1, March 2023, lqad004,

New Applications: Frequent Database Updates

- Jeremie S. Kim*, Can Firtina*, M. 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"
  Proceedings of the 21st Asia Pacific Bioinformatics Conference (***APBC***), Changsha, China, April 2023.
  [AirLift Source Code]
  [arxiv.org Version (pdf)]
  [Talk Video at BIO-Arch 2023 Workshop]

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*]

 *Equal contribution

Error Correction using ML [Bioinform. 2020]

- 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 ▾**

Accelerating String Matching [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.
  [Lightning 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*

Accelerating Genome Graphs [ISCA 2022]

- Damla Senol Cali, Konstantinos Kanellopoulos, Joel Lindegger, Zulal Bingol, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie Kim, Nika MansouriGhiasi, Gagandeep Singh, Juan Gomez-Luna, Nour Almadhoun Alserr, Mohammed Alser, Sreenivas Subramoney, Can Alkan, Saugata Ghose, and Onur Mutlu,
  "SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping"
  *Proceedings of the 49th International Symposium on Computer Architecture (**ISCA**)*, New York, June 2022.
  [arXiv version]

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

Genome Analysis via PIM [MICRO 2022]

- Haiyu Mao, Mohammed Alser, Mohammad Sadrosadati, Can Firtina, Akanksha Baranwal, Damla Senol Cali, Aditya Manglik, Nour Almadhoun Alserr, and Onur Mutlu,
  "GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping"
  *Proceedings of the 55th International Symposium on Microarchitecture (**MICRO**)*, Chicago, IL, USA, October 2022.
  [Slides (pptx) (pdf)]
  [Longer Lecture Slides (pptx) (pdf)]
  [Lecture Video (25 minutes)]
  [arXiv version]

GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping

Haiyu Mao[1] Mohammed Alser[1] Mohammad Sadrosadati[1] Can Firtina[1] Akanksha Baranwal[1]
Damla Senol Cali[2] Aditya Manglik[1] Nour Almadhoun Alserr[1] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bionano Genomics*

Food Microbiome Profiling using PIM

Taha Shahroodi, Mahdi Zahedi, Can Firtina, Mohammed Alser, Stephan Wong, Onur Mutlu, Said Hamdioui
"Demeter: A Fast and Energy-Efficient Food Profiler using Hyperdimensional Computing in Memory"
IEEE Access, 2022

Demeter: A Fast and Energy-Efficient Food Profiler Using Hyperdimensional Computing in Memory

TAHA SHAHROODI[1], MAHDI ZAHEDI[1], CAN FIRTINA[2], MOHAMMED ALSER[2], STEPHAN WONG[1], (Senior Member, IEEE), ONUR MUTLU[2], (Fellow, IEEE), AND SAID HAMDIOUI[1], (Senior Member, IEEE)
[1]Q&CE Department, EEMCS Faculty, Delft University of Technology (TU Delft), 2628 CD Delft, The Netherlands
[2]SAFARI Research Group, D-ITET, ETH Zürich, 8092 Zürich, Switzerland

Fast and Accurate Real-Time Genome Analysis

- Joel Lindegger, Can Firtina, Nika Mansouri Ghiasi, Mohammad Sadrosadati, Mohammed Alser, and Onur Mutlu,
  "RawAlign: Accurate, Fast, and Scalable Raw Nanopore Signal Mapping via Combining Seeding and Alignment"
  Preprint on ***arxiv***, October 2023.
  [arXiv version]
  [RawAlign Source Code]

RawAlign: Accurate, Fast, and Scalable Raw Nanopore Signal Mapping via Combining Seeding and Alignment

Joël Lindegger[§] Can Firtina[§] Nika Mansouri Ghiasi[§]
Mohammad Sadrosadati[§] Mohammed Alser[§] Onur Mutlu[§]

[§]*ETH Zürich*

Machine Learning in Genomics

- M. Banu Cavlak, Gagandeep Singh, Mohammed Alser, Can Firtina, Joel Lindegger, Mohammad Sadrosadati, Nika Mansouri Ghiasi, Can Alkan, and Onur Mutlu, **"TargetCall: Eliminating the Wasted Computation in Basecalling via Pre-Basecalling Filtering"**
  *Proceedings of the 21st Asia Pacific Bioinformatics Conference (**APBC**)*, Changsha, China, April 2023.
  [TargetCall Source Code]
  [arxiv.org Version]
  [Talk Video at BIO-Arch 2023 Workshop]

TargetCall: Eliminating the Wasted Computation in Basecalling via Pre-Basecalling Filtering

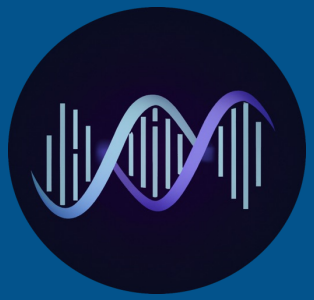
Meryem Banu Cavlak[1] Gagandeep Singh[1] Mohammed Alser[1] Can Firtina[1] Joël Lindegger[1]
Mohammad Sadrosadati[1] Nika Mansouri Ghiasi[1] Can Alkan[2] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bilkent University*

https://arxiv.org/pdf/2301.09200.pdf

Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
  - Intelligent genome analysis
- Enabling Fast and Accurate Real-time Analysis
  - RawHash and RawHash2
- Graph & ML Acceleration in Genomics
  - ApHMM
- Conclusion

ISMB/ECCB 2023



RawHash

Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes

Can Firtina

Nika Mansouri Ghiasi Joel Lindegger Gagandeep Singh

Meryem Banu Cavlak Haiyu Mao Onur Mutlu

Paper

Code

SAFARI

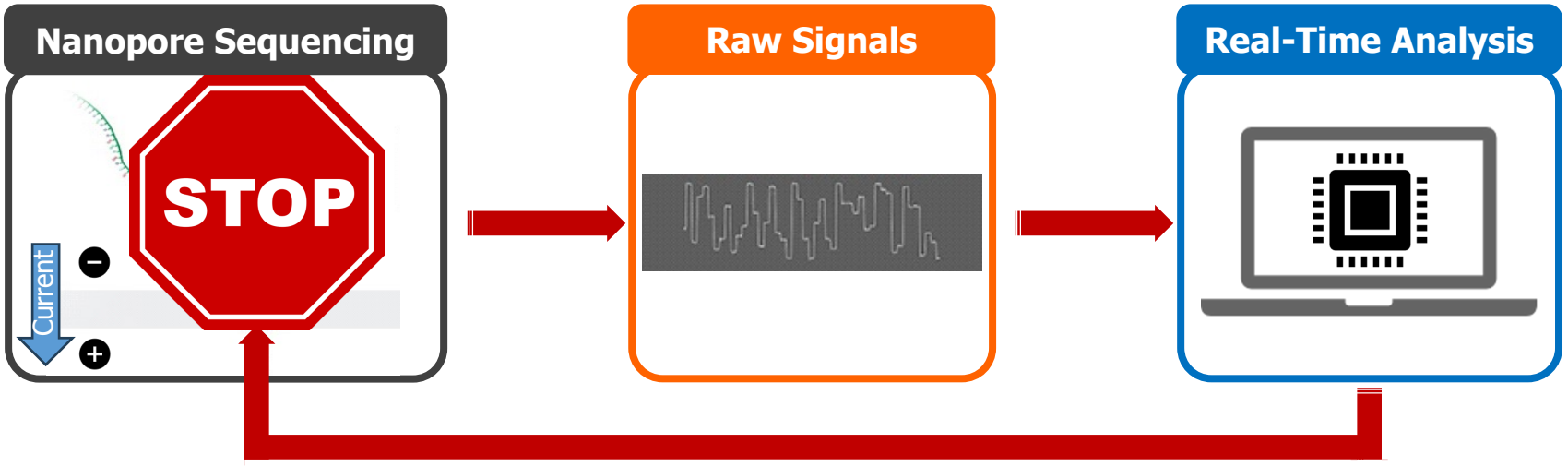
ETH zürich

Nanopore Sequencing

Nanopore Sequencing: a widely used sequencing technology

- Can sequence large fragments of nucleic acid molecules (up to >2Mbp)
- Offers high throughput
- Cost-effective
- Enables **real-time genome analysis**

Real-Time Analysis with Nanopore Sequencing



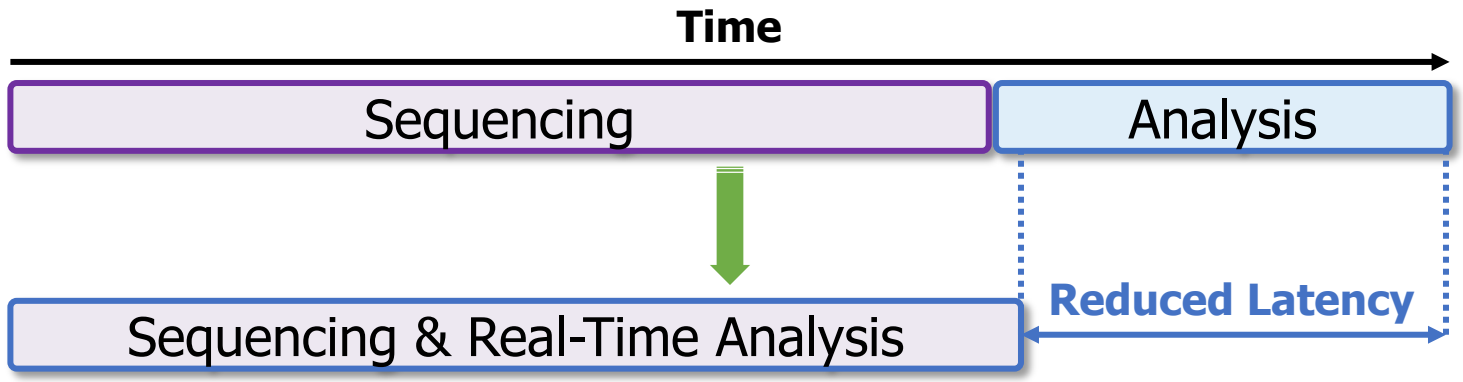
Raw Signals: Ionic current measurements generated at a certain **throughput**

Real-Time Analysis: Analyzing all raw signals by **matching the throughput**

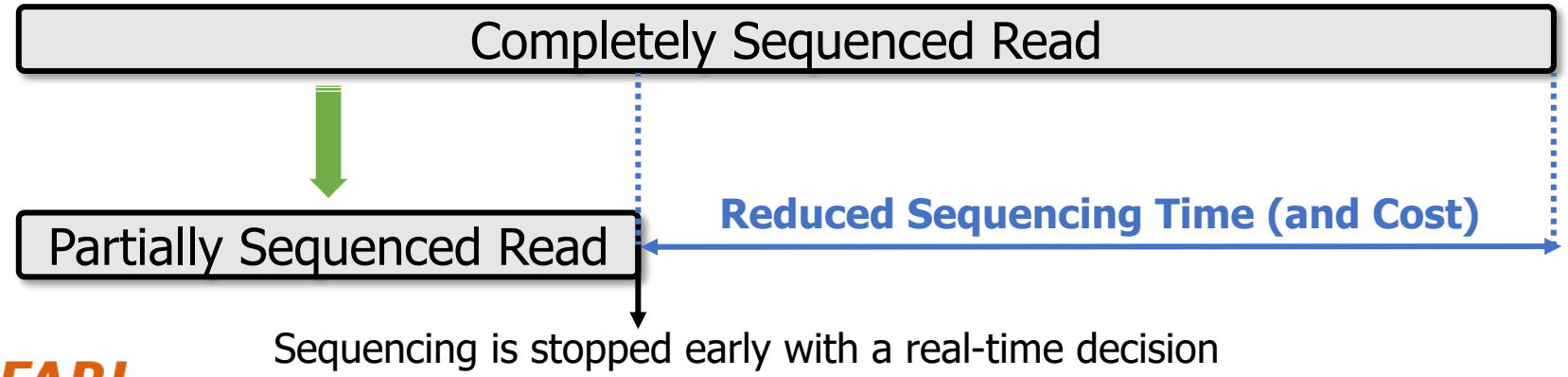
Real-Time Decisions: Stopping sequencing **early** based on real-time analysis

Benefits of Real-Time Genome Analysis

✓ **Reducing latency** by overlapping the sequencing and analysis steps



✓ **Reducing sequencing time and cost** by stopping sequencing early



Challenges in Real-Time Genome Analysis

Rapid analysis to match the nanopore sequencer throughput

Timely decisions to stop sequencing as early as possible

Accurate analysis from noisy raw signal data

Power-efficient computation for scalability and portability

Executive Summary

Problem: Real-time analysis of nanopore raw signals is **inaccurate** and **inefficient for large genomes**

Goal: Enable **fast** and **accurate** real-time analysis of raw signals for **large genomes**

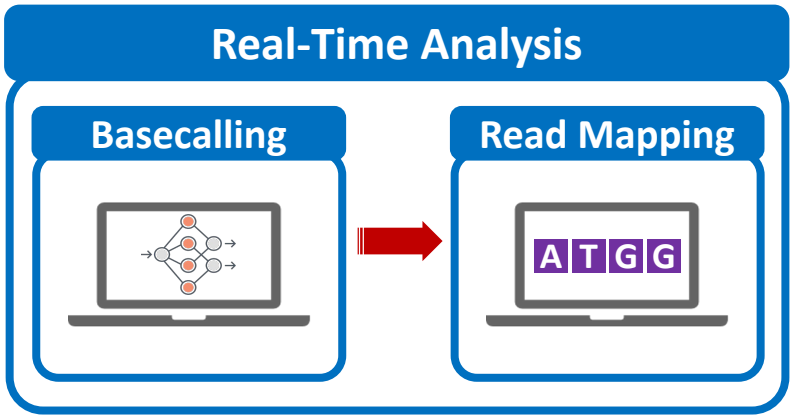
Key Contributions:
1) The **first hash-based mechanism** that can quickly and accurately analyze raw nanopore signals for **large genomes**

2) The novel **Sequence Until** technique can accurately and **dynamically stop the entire sequencing of all reads at once** if further sequencing is not necessary

Key Results: Across 3 use cases and 5 genomes of varying sizes, RawHash provides
- **25.8× and 3.4× better average throughput** compared to two state-of-the-art works
- **1.14× – 2.13× more accurate mapping results** for **large genomes**
- Sequence Until **reduces the sequencing time and cost by 15×**

Existing Solutions

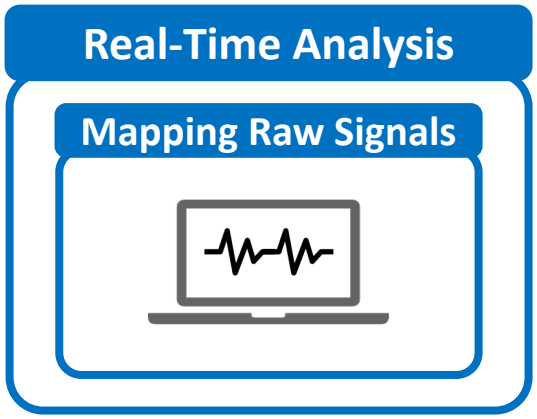
1. Deep neural networks (**DNNs**) for translating **signals** to **bases**



Less noisy analysis from basecalled sequences

Costly and power-hungry computational requirements

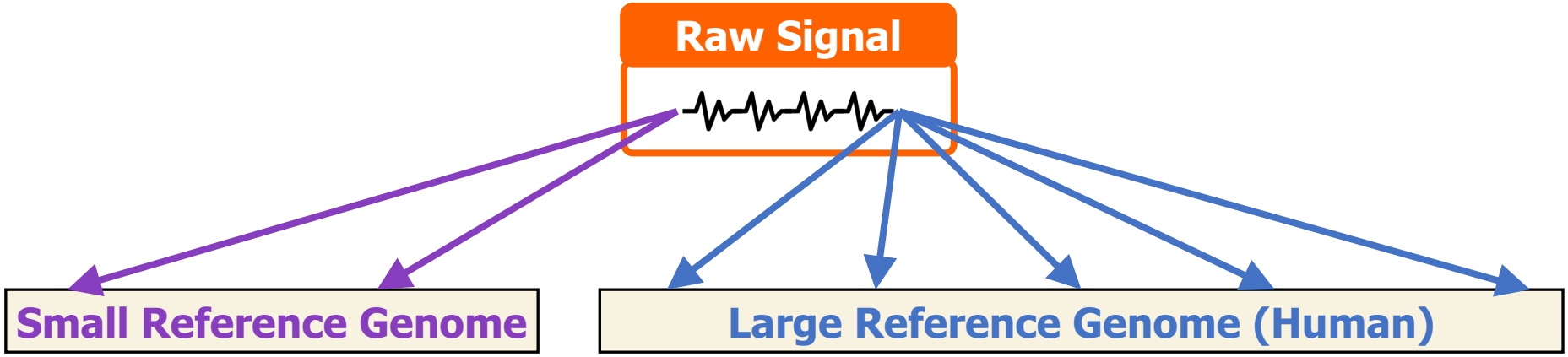
2. Mapping **signals** to reference genomes **without** basecalling



Raw signals contain richer information than bases

Efficient analysis with better scalability and portability

The Problem – Mapping Raw Signals



Fewer candidate regions in **small genomes**

Substantially **larger number of regions** to check **per read** as the genome size increases

Accurate mapping

Problem: Probabilistic mechanisms on **many regions → inaccurate mapping**

High throughput

Problem: Distance calculation on **many regions → reduced throughput**

The Problem – Mapping Raw Signals

Existing solutions are
inaccurate or inefficient
for large genomes

Accurate mapping

on **many regions → inaccurate mapping**

High throughput

Problem: Distance calculation
on **many regions → reduced throughput**

Outline

Background

RawHash

Evaluation

Conclusion

Goal

Enable **fast and accurate real-time analysis** of raw nanopore signals **for large genomes**

RawHash

The **first hash-based search mechanism**
to quickly and accurately map raw nanopore signals
to reference genomes

Sequence Until can accurately and **dynamically stop**
the entire sequencing run at once
if further sequencing is unnecessary

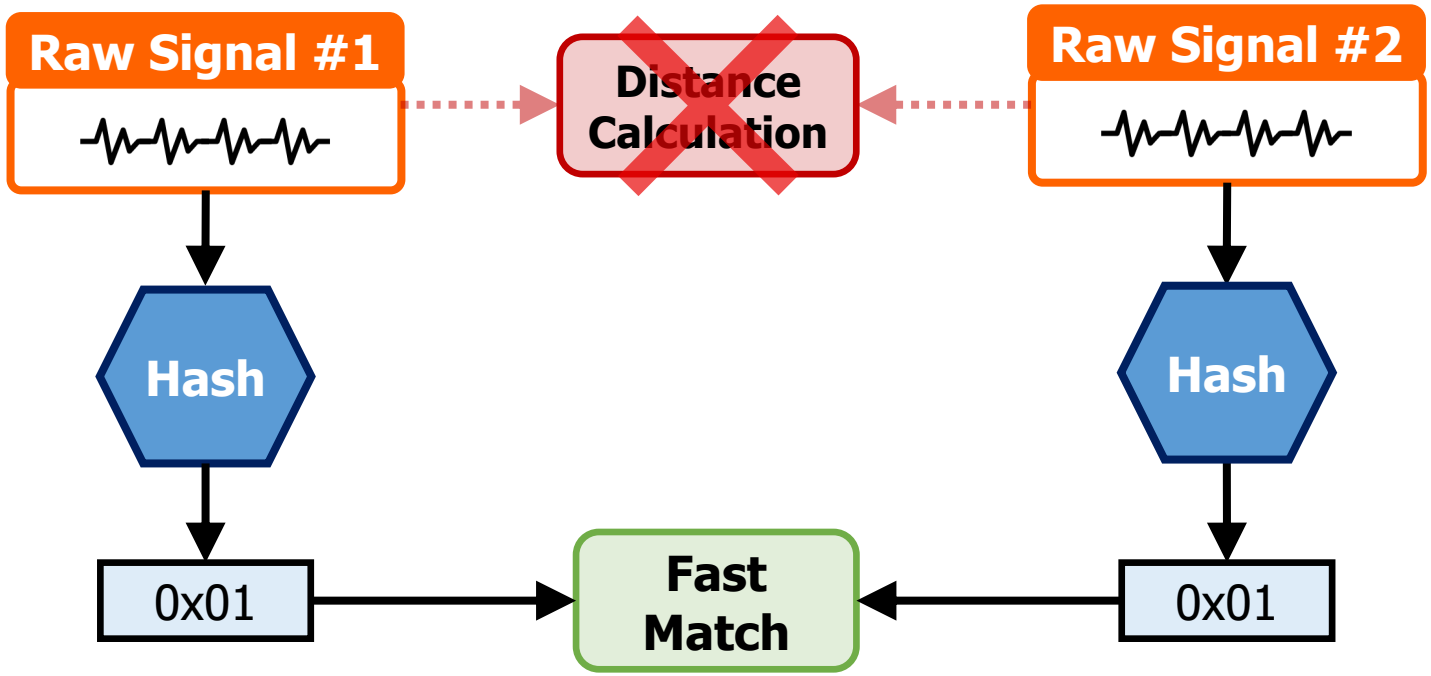
RawHash

The **first hash-based search mechanism**
to quickly and accurately map raw nanopore signals
to reference genomes

Sequence Until can accurately and **dynamically stop**
the entire sequencing run at once
if further sequencing is unnecessary

RawHash – Key Idea

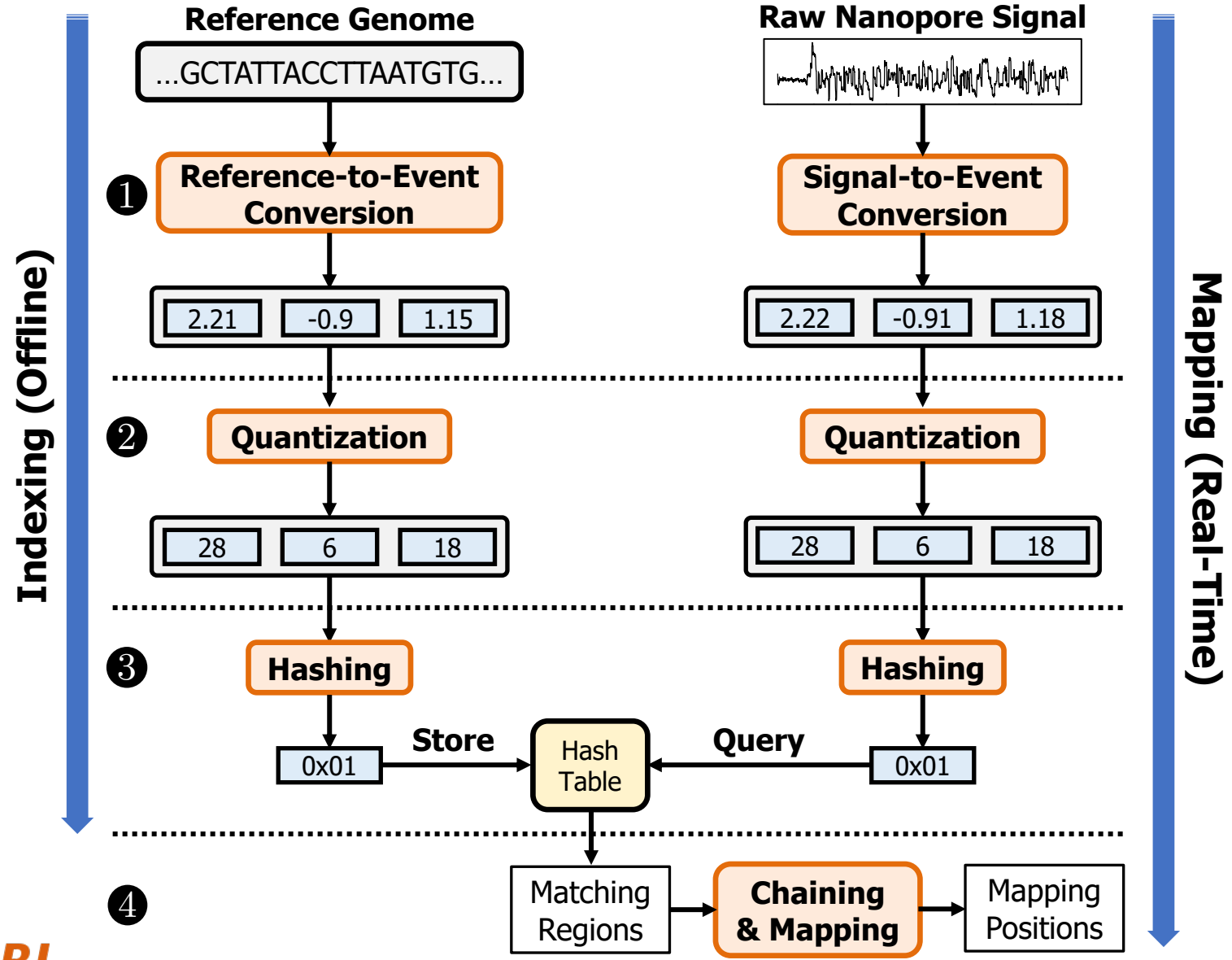
Key Observation: **Identical** nucleotides generate **similar** raw signals



Challenge #1: Generating the **same** hash value for **similar enough** signals

Challenge #2: **Accurately** finding similar regions **as few as possible**

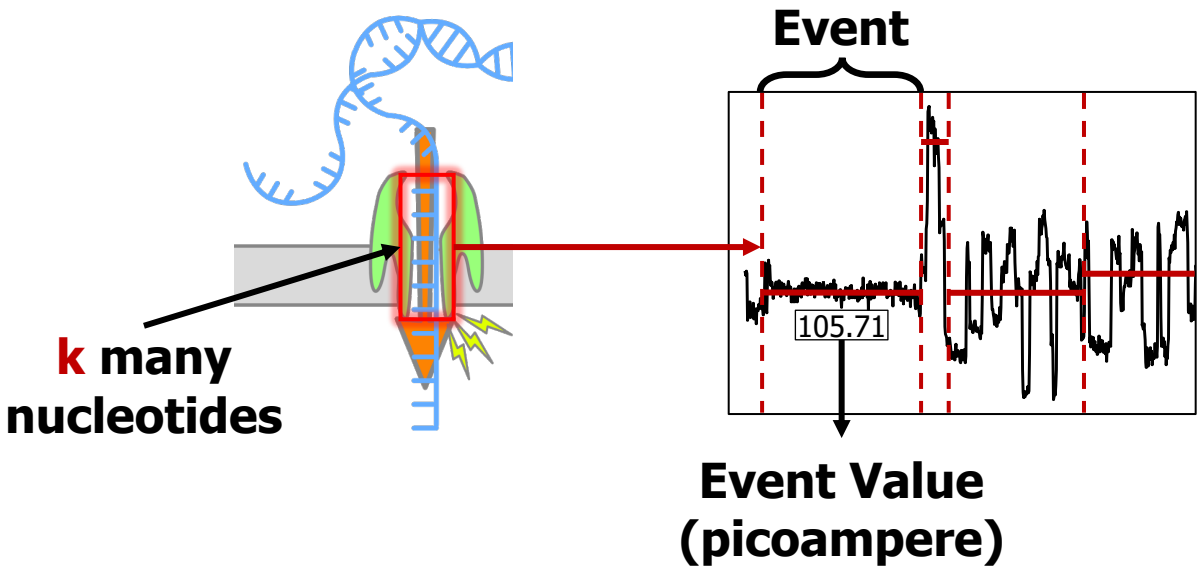
RawHash Overview



Events in Raw Nanopore Signals

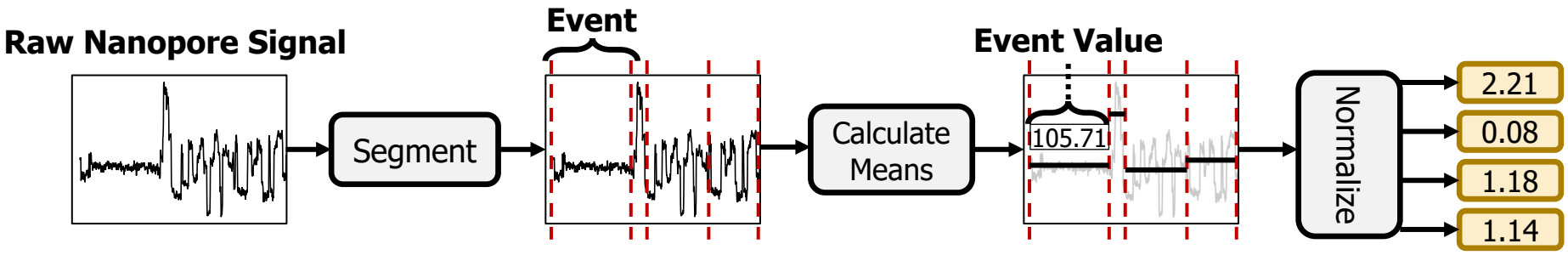
- **Event:** A **segment** of the raw signal
  - Corresponds to a **particular k**-mer

- **Event detection** finds these segments to identify **k**-mers
  - Start and end positions are marked by abrupt signal changes
  - Statistical methods identify these abrupt changes
  - **Event value:** average of signals **within an event**



Signal-to-Event Conversion

- **Event detection:** Identifies signal regions corresponding to specific k-mers
  - Uses statistical test (**segmentation**) to spot abrupt signal changes



- Consecutive events ➔ consecutive k-mers

Signal-to-Event Conversion

- **Event detection:** Identifies signal regions corresponding to specific k-mers
  - Uses statistical test (**segmentation**) to spot abrupt signal changes

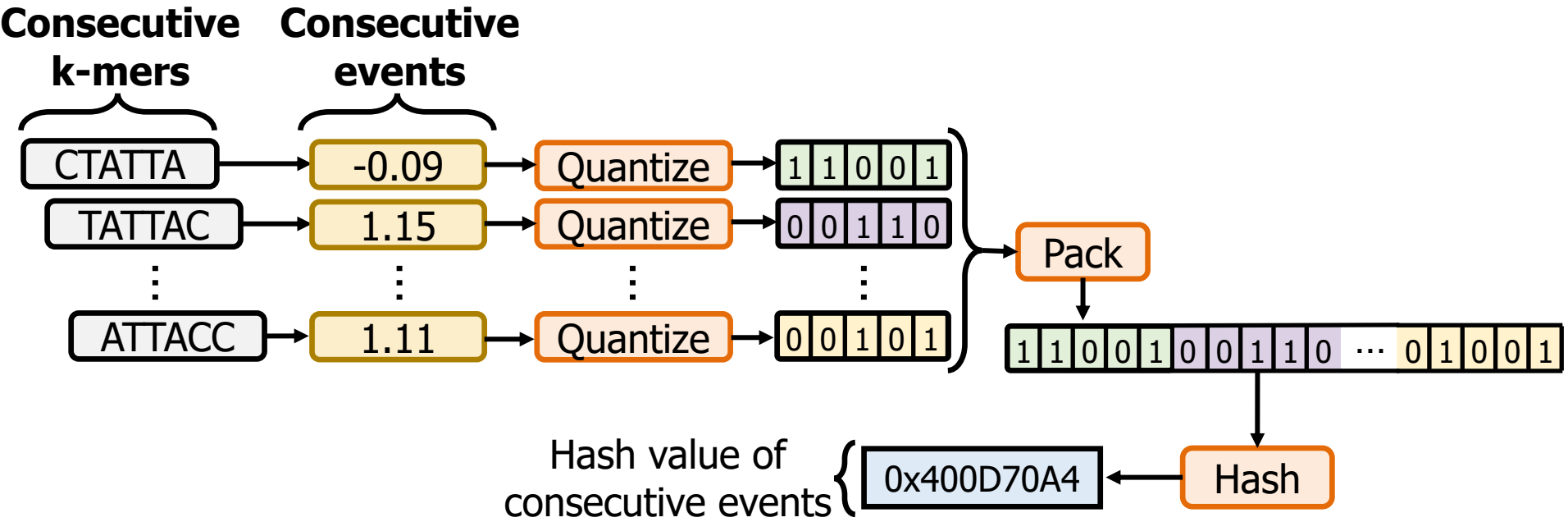
Can we directly match signals to each other?

- Consecutive events ➔ consecutive k-mers

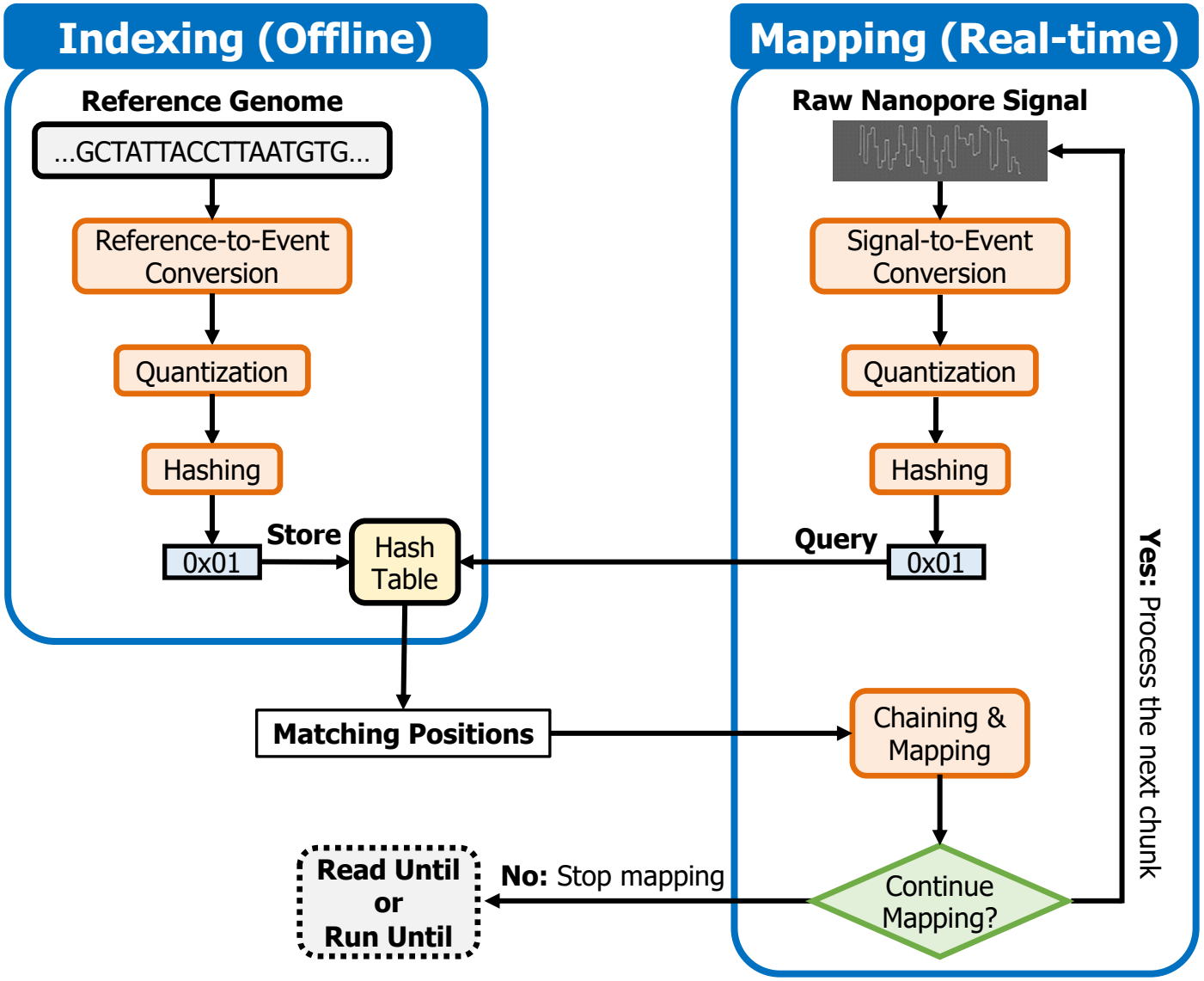
Hashing for Fast Similarity Search

- Each event usually represents a very small k-mer (6 to 9 characters)
  - **Challenge:** Short k-mers are likely to appear in many locations

- **Key Idea:** Create longer k-mers from many **consecutive events**
- **Key Benefit:** Directly match hash values to quickly identify similarities



Real-Time Mapping using Hash-based Indexing



RawHash

The **first hash-based search mechanism**
to quickly and accurately map raw nanopore signals
to reference genomes

Sequence Until can accurately and **dynamically stop**
the entire sequencing run at once
if further sequencing is unnecessary

RawHash

The **first hash-based search mechanism**
to quickly and accurately map raw nanopore signals
to reference genomes

Sequence Until can accurately and **dynamically stop**
the entire sequencing run at once
if further sequencing is unnecessary

The Sequence Until Mechanism

- **Problem:**
  - Unnecessary sequencing waste time, power and money
- **Key Idea:**
  - **Dynamically** decide if further sequencing of the entire sample is necessary to achieve high accuracy
  - Stop sequencing early without sacrificing accuracy
- **Potential Benefits:**
  - Significant **reduction in sequencing time and cost**
- Example real-time genome analysis use case:
  - **Relative abundance estimation**

Outline

Background

RawHash

Evaluation

Conclusion

Evaluation Methodology

- Compared to **UNCALLED** [Kovaka+, Nat. Biotech. 2021] and **Sigmap** [Zhang+, ISMB/ECCB 2021]
  - **CPU baseline:** AMD EPYC 7742 @2.26GHz
  - **32 threads** for each tool

- **Use cases** for real-time genome analysis:
  1. Read mapping
  2. Relative abundance estimation
     - **Benefits of Sequence Until**
  3. Contamination analysis

Evaluation Methodology

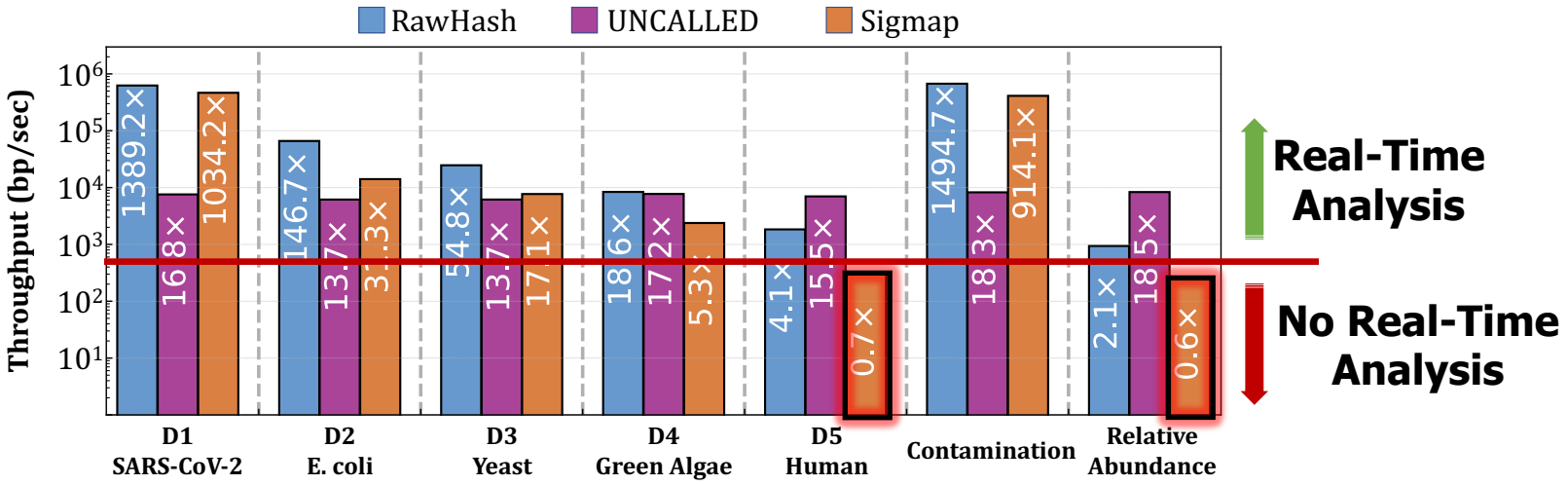
- Evaluation metrics:
  - **Throughput** (bases processed per second)
  - Potential reduction in **sequencing time and cost**
  - **Accuracy**
    - **Baseline:** Mapping basecalled reads using minimap2
    - Precision, recall, and F1 scores
    - Relative abundance estimation distance to ground truth

- **Datasets:**

	Organism	Reads (#)	Bases (#)	Genome Size
Read Mapping				
D1	*SARS-CoV-2*	1,382,016	594M	29,903
D2	*E. coli*	353,317	2,365M	5M
D3	*Yeast*	49,989	380M	12M
D4	*Green Algae*	29,933	609M	111M
D5	*Human HG001*	269,507	1,584M	3,117M
Relative Abundance Estimation				
	D1-D5	2,084,762	5,531M	3,246M
Contamination Analysis				
	D1 and D5	1,651,523	2,178M	29,903

Throughput

- **Real-time analysis requires** faster throughput than sequencer
  - Throughput of a nanopore sequencer: **~450 bp/sec (data generation speed)**

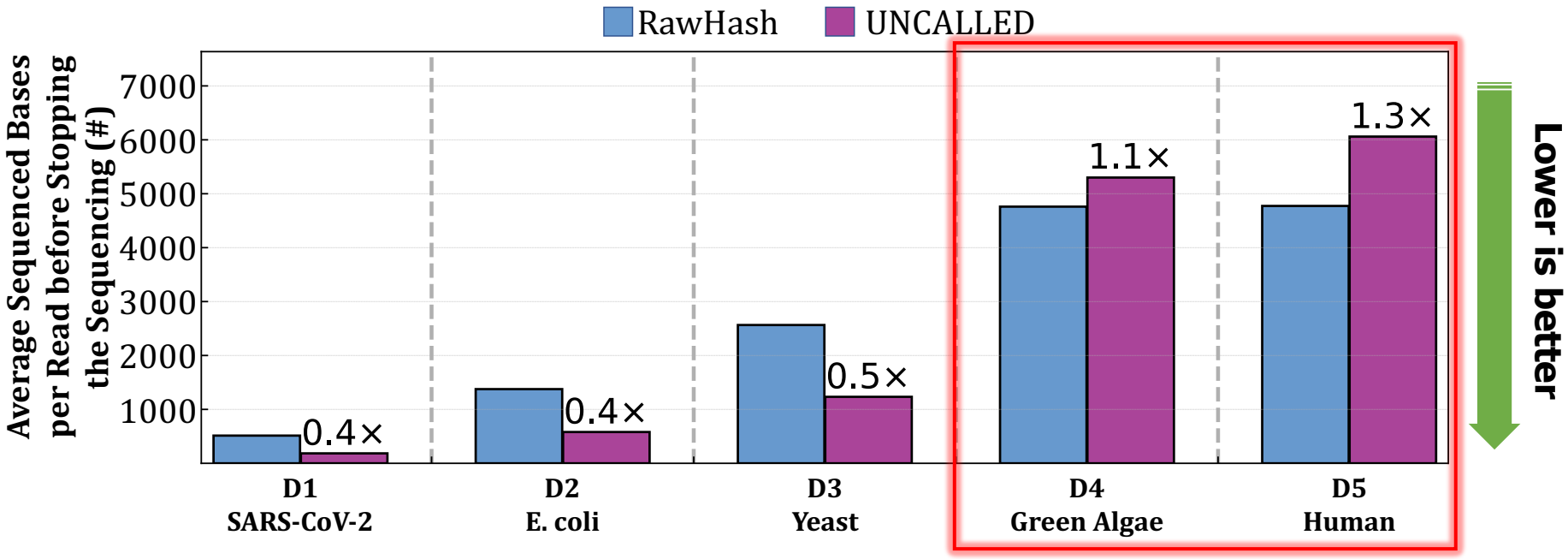


25.8× and 3.4× better average throughput compared to **UNCALLED and Sigmap**, respectively

Sigmap **cannot** perform real-time analysis **for large genomes**

Sequencing Time

- Fewer bases to sequence ➔
  - Reduction in sequencing time and cost



RawHash **reduces sequencing time and cost**

for large genomes up to **1.3**× compared to UNCALLED

Mapping Accuracy

- Read mapping accuracy of each tool and each use case

Dataset		UNCALLED	Sigmap	RawHash
Read Mapping				
D1	Precision	0.9547	**0.9929**	0.9868
SARS-CoV-2	Recall	**0.9910**	0.5540	0.8735
	F_1	**0.9725**	0.7112	0.9267
D2	Precision	0.9816	**0.9842**	0.9573
E. coli	Recall	**0.9647**	0.9504	0.9009
	F_1	**0.9731**	0.9670	0.9282
D3	Precision	0.9459	0.9856	**0.9862**
Yeast	Recall	**0.9366**	0.9123	0.8412
	F_1	0.9412	**0.9475**	0.9079
D4	Precision	0.8836	**0.9741**	0.9691
Green Algae	Recall	0.7778	**0.8987**	0.7015
	F_1	0.8273	**0.9349**	0.8139
D5	Precision	0.4867	0.4287	**0.8959**
Human HG001	Recall	0.2379	0.2641	**0.4054**
	F_1	0.3196	0.3268	**0.5582**

Dataset		UNCALLED	Sigmap	RawHash
Relative Abundance Estimation				
D1-D5	Precision	0.7683	0.7928	**0.9484**
	Recall	0.1273	0.2739	**0.3076**
	F_1	0.2184	0.4072	**0.4645**
Contamination Analysis				
D1, D5	Precision	**0.9378**	0.7856	0.8733
	Recall	**0.9910**	0.5540	0.8735
	F_1	**0.9637**	0.6498	0.8734

For Large Genomes: RawHash provides the **best accuracy** in all metrics, resulting in **1.14×** - **2.13×** improvement in F_1 score

Relative Abundance Estimation Accuracy

- Estimating the ratio of genomes in a sample in real-time
  - **Distance:** Euclidean distance compared to the ground truth distance
  - The dataset includes a large reference genome

	Estimated Relative Abundance Ratios					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
Ground Truth	0.0929	0.4365	0.0698	0.1179	0.2828	N/A
UNCALLED	0.0026	0.5884	0.0615	0.1313	0.2161	0.1895
Sigmap	0.0419	0.4191	0.1038	0.0962	0.3390	0.0877
RawHash	0.1249	0.4701	0.0957	0.0629	0.2464	**0.0847**

RawHash provides the **best relative abundance estimation** closest to the ground truth estimation

Simulating Sequence Until

- Real relative abundance results using the entire set of reads

	Estimated Relative Abundance Ratios					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
Ground Truth	0.0929	0.4365	0.0698	0.1179	0.2828	N/A
UNCALLED	0.0026	0.5884	0.0615	0.1313	0.2161	0.1895
Sigmap	0.0419	0.4191	0.1038	0.0962	0.3390	0.0877

UNCALLED and RawHash benefit from **Sequence Until** significantly **by up to 100×** reductions in sequencing time and costs

Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
Ground Truth	0.0929	0.4365	0.0698	0.1179	0.2828	N/A
UNCALLED (25%)	0.0026	0.5890	0.0613	0.1332	0.2139	0.1910
RawHash (25%)	0.0271	0.4853	0.0920	0.0786	0.3170	**0.0995**
UNCALLED (10%)	0.0026	0.5906	0.0611	0.1316	0.2141	0.1920
RawHash (10%)	0.0273	0.4869	0.0963	0.0772	0.3124	**0.1004**
UNCALLED (1%)	0.0026	0.5750	0.0616	0.1506	0.2103	0.1836
RawHash (1%)	0.0259	0.4783	0.0987	0.0882	0.3088	**0.0928**
UNCALLED (0.1%)	0.0040	0.4565	0.0380	0.1910	0.3105	0.1242
RawHash (0.1%)	0.0212	0.5045	0.1120	0.0810	0.2814	**0.1136**
UNCALLED (0.01%)	0.0000	0.5551	0.0000	0.0000	0.4449	0.2602
RawHash (0.01%)	0.0906	0.6122	0.0000	0.0000	0.2972	**0.2232**

More in the Paper

- **More Results**
  - **Mapping time** per read
  - Overall **computational resources** required by each tool
    - Peak memory usage, CPU time and real time in the indexing and mapping steps
  - **Performance breakdown** of the steps in RawHash

- **Details of all mechanisms and configurations**
  - Details of the **quantization** and **hashing** mechanism
  - Details of the **parameter configurations**
  - Trade-offs between the **DNN-based approaches** and raw signal mapping approaches

RawHash

- Can Firtina, Nika Mansouri Ghiasi, Joel Lindegger, Gagandeep Singh, Meryem Banu Cavlak, Haiyu Mao, and Onur Mutlu,
  "RawHash: Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes"
  *Proceedings of the 31st Annual Conference on Intelligent Systems for Molecular Biology (**ISMB**) and the 22nd European Conference on Computational Biology (**ECCB**)*, Jul 2023
  [arXiv preprint]
  [Source Code]

Bioinformatics, 2023, **39**, i297–i307
https://doi.org/10.1093/bioinformatics/btad272
ISMB/ECCB 2023

OXFORD

RawHash: enabling fast and accurate real-time analysis of raw nanopore signals for large genomes

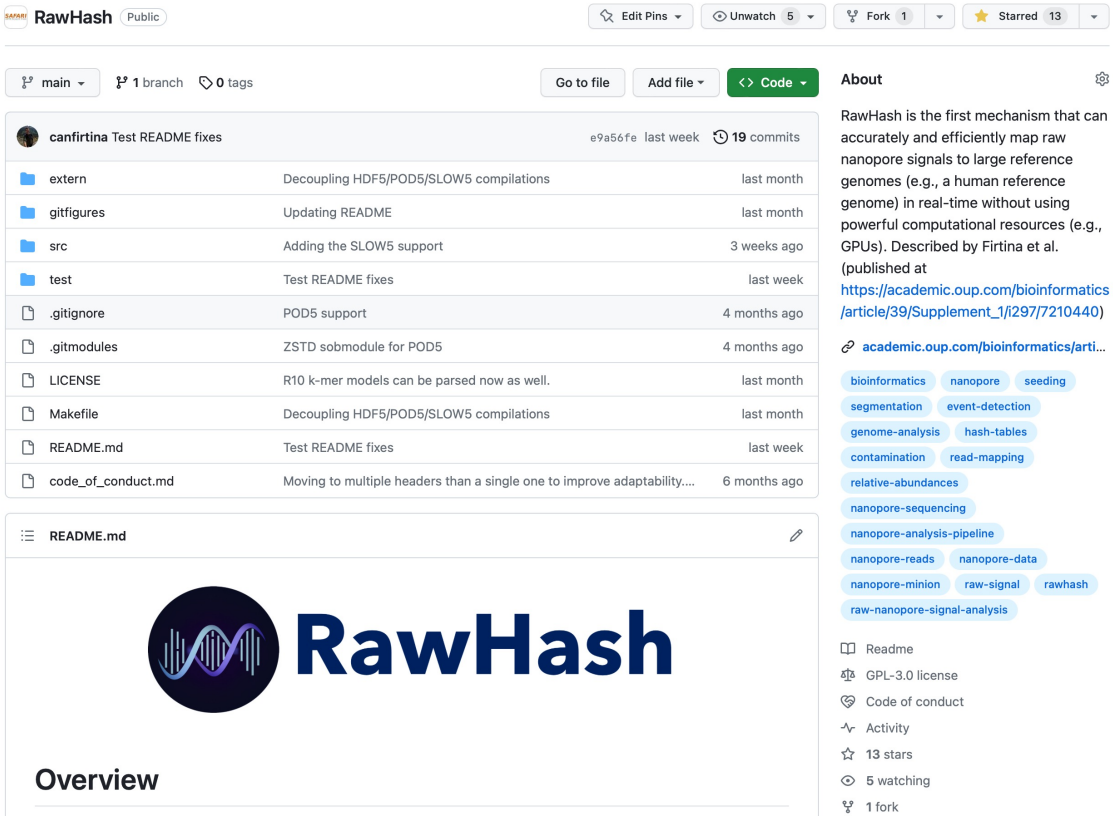
Can Firtina [1,*], Nika Mansouri Ghiasi [1], Joel Lindegger [1], Gagandeep Singh [1], Meryem Banu Cavlak [1], Haiyu Mao [1], Onur Mutlu [1,*]

[1]Department of Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland

*Corresponding author. Department of Information Technology and Electrical Engineering, ETH Zurich, Gloriastrasse 35, 8092 Zurich, Switzerland.
E-mail: firtinac@ethz.ch (C.F.), omutlu@ethz.ch (O.M.)

RawHash Source Code

- Supports **all major raw signal file formats and flow cell versions**
  - FAST5, POD5, S/BLOW5 file formats
- Easy-to-use scripts
  - To download all the datasets
  - To reproduce all of our results
- You can write your outlier function for Sequence Until
  - Easily integrate Sequence Until
- Upcoming Feature:
  - Integrating the MinKNOW API



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

SAFARI

Outline

Background

RawHash

Evaluation

Conclusion

Conclusion

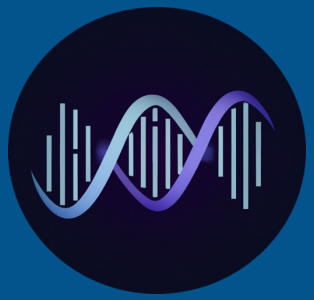
Key Contributions:
1) The **first hash-based mechanism** that can quickly and accurately analyze raw nanopore signals for **large genomes**

2) The novel **Sequence Until** technique can accurately and **dynamically stop the entire sequencing of all reads at once** if further sequencing is not necessary

Key Results: Across 3 use cases and 5 genomes of varying sizes, RawHash provides
- **25.8× and 3.4× better average throughput** compared to two state-of-the-art works
- **1.14× – 2.13× more accurate mapping results** for **large genomes**
- Sequence Until **reduces the sequencing time and cost by 15×**

Many opportunities for analyzing raw nanopore signals in real-time:
- Many hash-based **sketching techniques** can now be used for raw signals
- **Indexing is very cheap:** Many future use cases with the on-the-fly index construction
- We should rethink the algorithms to perform downstream analysis fully using raw signals

ISMB/ECCB 2023



RawHash

Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes

Can Firtina

Nika Mansouri Ghiasi Joel Lindegger Gagandeep Singh

Meryem Banu Cavlak Haiyu Mao Onur Mutlu

Paper

Code

SAFARI

ETH zürich

Fast and Accurate Real-Time Genome Analysis

- Can Firtina, Melina Soysal, Joel Lindegger, and Onur Mutlu,
  "RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism"
  Preprint on ***arxiv***, September 2023.
  [arXiv version]
  [RawHash2 Source Code]

RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism

Can Firtina Melina Soysal Joel Lindegger Onur Mutlu
ETH Zürich

Optimizations in RawHash2 (1)

- **More sensitive** chaining implementation with penalty scores
  - Benefits: Enables filtering dissimilar regions quickly
  - Downside: Additional computations with costly log operations

- **Weighted mapping decisions**
  - Benefit #1: 'Learned' mapping decisions based on the weights chosen from empirical analysis
  - Benefit #2: Faster and more accurate decisions

- Frequency **filters**
  - Filters the seeds that frequently appear before chaining
  - Benefits: Reduced workload on chaining without significantly affecting accuracy
  - Downside: Less sensitive mapping due to removed seeds

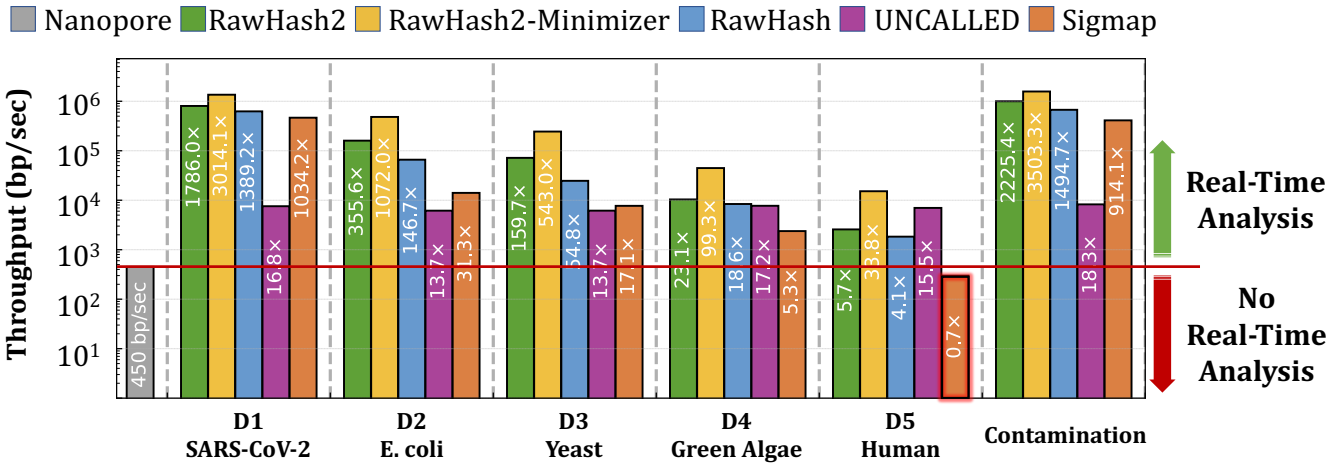
Optimizations in RawHash2 (2)

- New sketching techniques such as **minimizers** and **BLEND**
  - Enables integration of widely studied sketching techniques
  - Benefits: Can take advantage of these techniques (e.g., reduced storage requirements)

- Support for the recent improvements in the technology
  - Support for **new data formats**: POD5 and S/BLOW5
  - Support for **newer nanopore chemistry** versions: R10.4

Results – Throughput

- **Real-time analysis requires** faster throughput than sequencer
  - Throughput of a nanopore sequencer: **~450 bp/sec (data generation speed)**



2.3× better average throughput RawHash

Results – Accuracy

Dataset		UNCALLED	Sigmap	RawHash	RawHash2	RawHash2-Minimizer
			Read Mapping			
D1	Precision	0.9547	**0.9929**	0.9868	0.9857	0.9602
SARS-CoV-2	Recall	**0.9910**	0.5540	0.8735	0.8842	0.7080
	F_1	**0.9725**	0.7112	0.9267	0.9322	0.8150
D2	Precision	0.9816	0.9842	0.9573	**0.9864**	0.9761
E. coli	Recall	**0.9647**	0.9504	0.9009	0.8934	0.7805
	F_1	**0.9731**	0.9670	0.9282	0.9376	0.8674
D3	Precision	0.9459	0.9856	**0.9862**	0.9567	0.9547
Yeast	Recall	**0.9366**	0.9123	0.8412	0.8942	0.7792
	F_1	0.9412	**0.9475**	0.9079	0.9244	0.8581
D4	Precision	0.8836	**0.9741**	0.9691	0.9264	0.9198
Green Algae	Recall	0.7778	**0.8987**	0.7015	0.8659	0.6711
	F_1	0.8273	**0.9349**	0.8139	0.8951	0.7760
D5	Precision	0.4867	0.4287	**0.8959**	0.8830	0.8111
Human HG001	Recall	0.2379	0.2641	0.4054	**0.4317**	0.1862
	F_1	0.3196	0.3268	0.5582	**0.5799**	0.3028
			Contamination			
D1 and D5	Precision	0.9378	0.7856	0.8733	**0.9393**	0.9330

RawHash2 is more accurate than RawHash **in all cases**

Results – Average Sequencing Length

Tool	*SARS-CoV-2*	*E. coli*	*Yeast*	*Green Algae*	*Human*	*Contamination*
Average sequenced base length per read						
UNCALLED	**184.51**	**580.52**	**1,233.20**	5,300.15	6,060.23	1,582.63
RawHash	513.95	1,376.14	2,565.09	4,760.59	4,773.58	742.56
RawHash2	488.46	1,234.39	1,715.31	**2,077.39**	**3,441.43**	**681.94**
RawHash2-Minimizer	566.42	1,763.76	2,339.41	2,891.55	4,090.68	787.82
Average sequenced number of chunks per read						
Sigmap	**1.01**	**2.11**	**4.14**	5.76	10.40	2.06
RawHash	1.24	3.20	5.83	10.72	10.70	2.41
RawHash2	1.18	2.93	4.02	**4.84**	**7.78**	**1.68**
RawHash2-Minimizer	1.39	4.16	5.45	6.66	9.17	1.89

RawHash2 uses fewer bases to sequence than RawHash in all cases

RawHash2 uses the smallest number of bases to sequence for larger genomes

Fast and Accurate Real-Time Genome Analysis

- Can Firtina, Melina Soysal, Joel Lindegger, and Onur Mutlu,
  "RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism"
  Preprint on ***arxiv***, September 2023.
  [arXiv version]
  [RawHash2 Source Code]

RawHash2: Accurate and Fast Mapping of Raw Nanopore Signals using a Hash-based Seeding Mechanism

Can Firtina Melina Soysal Joel Lindegger Onur Mutlu
ETH Zürich

The Future is Bright for Genome Analysis

- Enabling cost-effective, portable, fast, and accurate genome analysis has many implications
  - What are the new applications we can enable with these new unique benefits?
- Can we do even better?
  - Understanding and modifying the sequencing process for analyzing other types of biological data
- How about other sequencing technologies?

Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
  - Intelligent genome analysis
- Enabling Fast and Accurate Real-time Analysis
  - RawHash and RawHash2
- Graph & ML Acceleration in Genomics
  - ApHMM
- Conclusion

HiPEAC 2024

ApHMM

Paper

Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis

Can Firtina

canfirtina@gmail.com
https://cfirtina.com

Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, Onur Mutlu

SAFARI ETH zürich

intel TU Delft Carnegie Mellon

Executive Summary

Motivation: Graph structures such as **profile Hidden Markov Models (pHMMs)** are commonly used to accurately analyze biological sequences

Problem: The parameters used in pHMMs are mainly trained and used with a **computationally intensive Baum-Welch algorithm**, causing major performance and energy overhead for many genomics workloads

Goal: Enable rapid, power-efficient, and flexible use of pHMMs for genomics workloads

ApHMM: the first flexible and hardware-software accelerator for pHMMs that can

1) Substantially reduce unnecessary data storage, data movement, and computations by effectively co-designing hardware and software together
2) Provide a flexible design to support several genomics workloads that use pHMMs

Key Results: Our ASIC implementation compared to CPU, GPU, and FPGA baselines across 3 workloads

- **15.55×–260.03×, 1.83×–5.34×, and 27.97× better performance**
- **Up to 2622.94×** reduction in **energy consumption**

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

Outline

Background & Problem

ApHMM

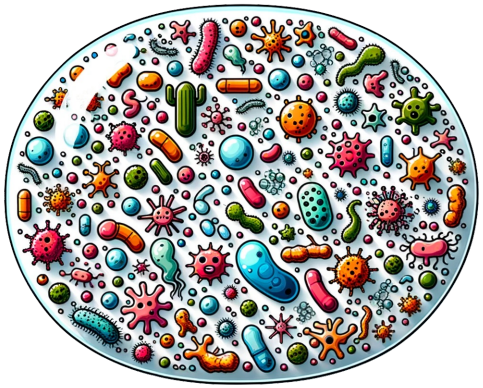
Evaluation

Conclusion

Genome Analysis – Why?

- **Fast and accurate** genome analysis is important for:

Understanding **genetic variations, species, and evolution**



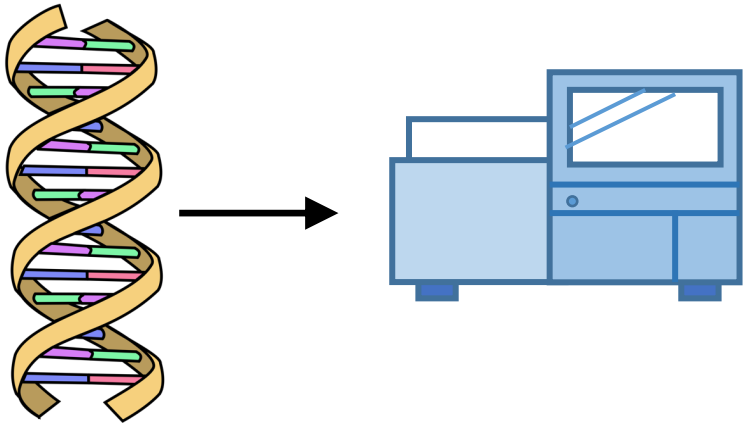
Predicting the **presence of pathogens** in an environment

Surveillance of **disease outbreaks**

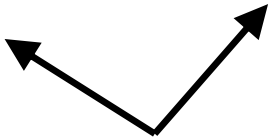
Personalized medicine

Genome Analysis – How?

- **Genome sequencing machines** can quickly convert biological molecules
  - Into sequences of characters for analysis



Biological Molecule (e.g., DNA)

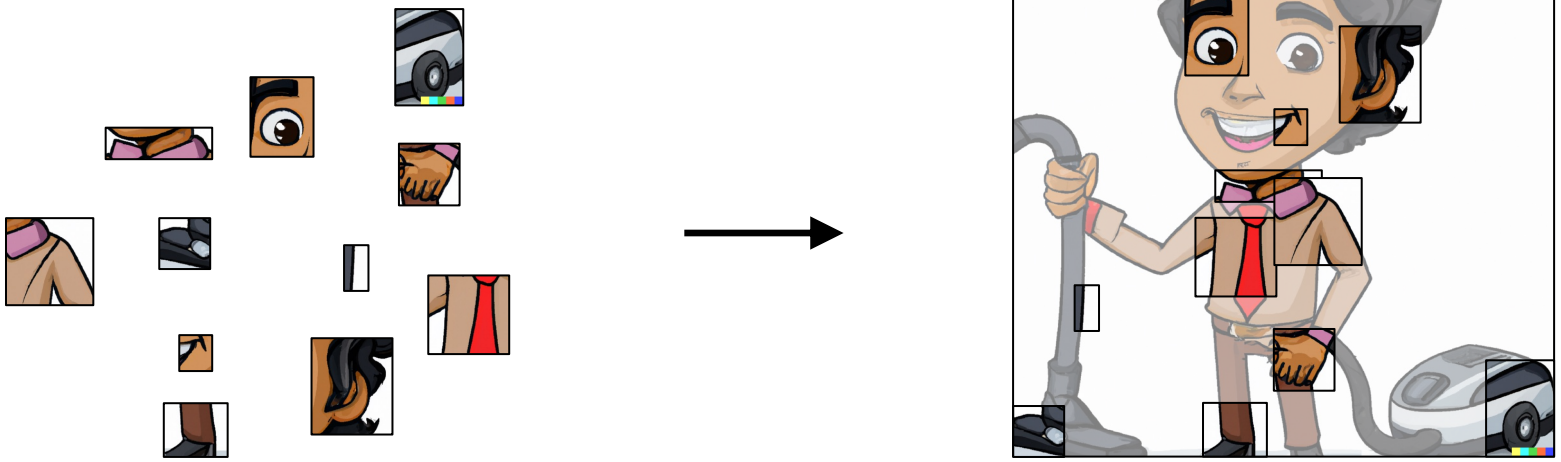


Sequences from DNA

Sequence Comparison is Essential

- Analyze sequences by accurately and quickly **comparing** them
  - To **each other**
  - To a **template sequence** representative of a species, a certain group…

Biological Sequences (e.g., DNA, proteins)

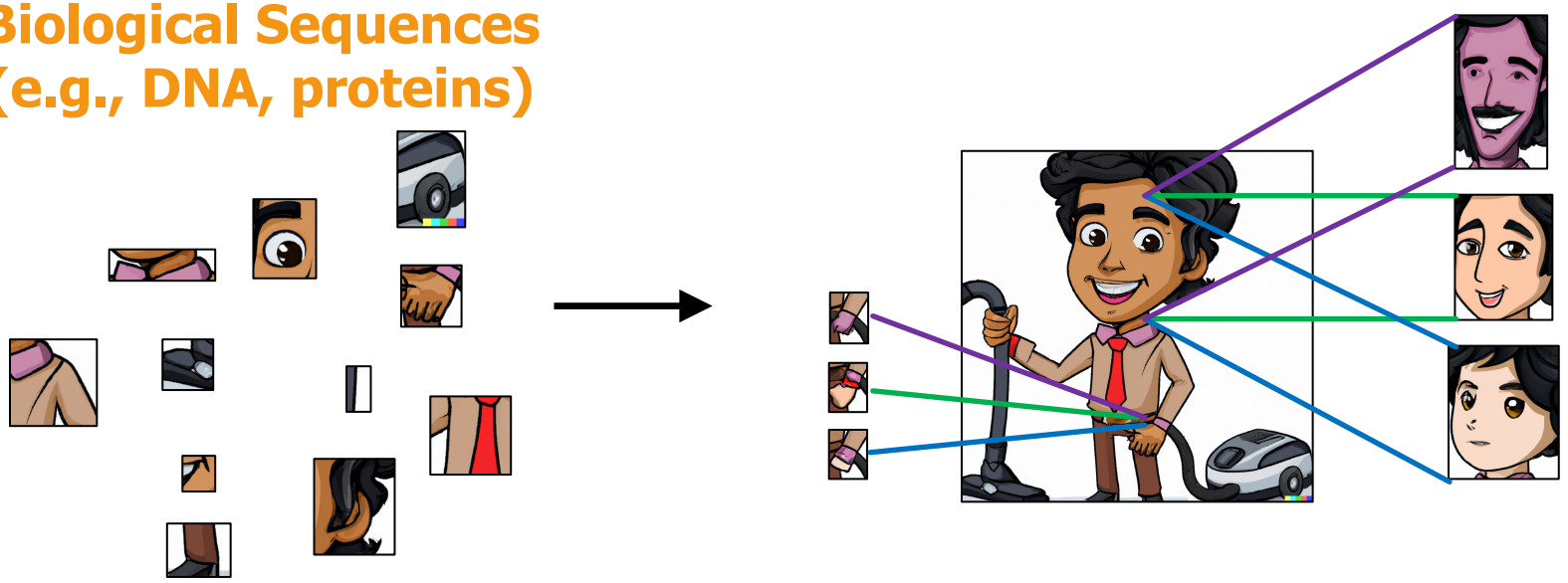


- Essential to understand functionality of a sequence, mutations, diseases…

Graphs for Sequence Comparisons

- **Graphs are commonly used** in sequence comparisons
  - **Can avoid redundant** comparisons and storage
  - Provides **rich information** on **expected variations** between sequences

Biological Sequences (e.g., DNA, proteins)

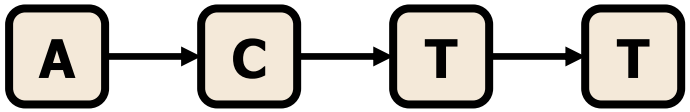


Profile Hidden Markov Models

- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - Each **state** outputs a biological character (**emission**) when visited
  - States are visited via **transitions** (edges) based on **observed variations**
  - **Variations:** No variation

Expected sequence: ACTT

Observed Sequence #1: ACTT
(No variation)

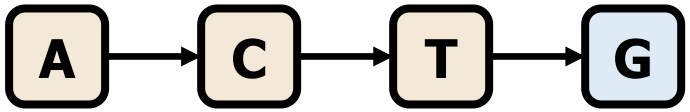


Profile Hidden Markov Models

- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - Each **state** outputs a biological character (**emission**) when visited
  - States are visited via **transitions** (edges) based on **observed variations**
  - **Variations:** No variation, Substitutions

Expected sequence: ACTT

Observed Sequence #2: ACTG (Substitutions)

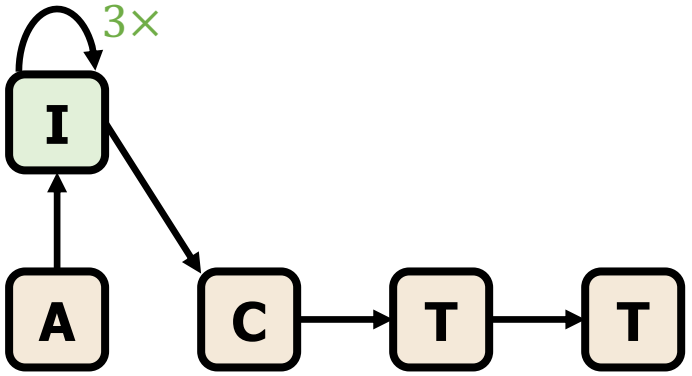


Profile Hidden Markov Models

- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - Each **state** outputs a biological character (**emission**) when visited
  - States are visited via **transitions** (edges) based on **observed variations**
  - **Variations:** No variation, Substitutions, Insertions

Expected sequence: ACTT

Observed Sequence #3: AGGGCTT
(I: Insertions)

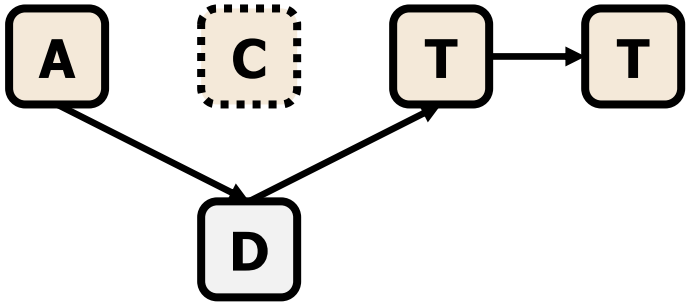


Profile Hidden Markov Models

- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - Each **state** outputs a biological character (**emission**) when visited
  - States are visited via **transitions** (edges) based on **observed variations**
  - **Variations:** No variation, Substitutions, Insertions, Deletions

Expected sequence: ACTT

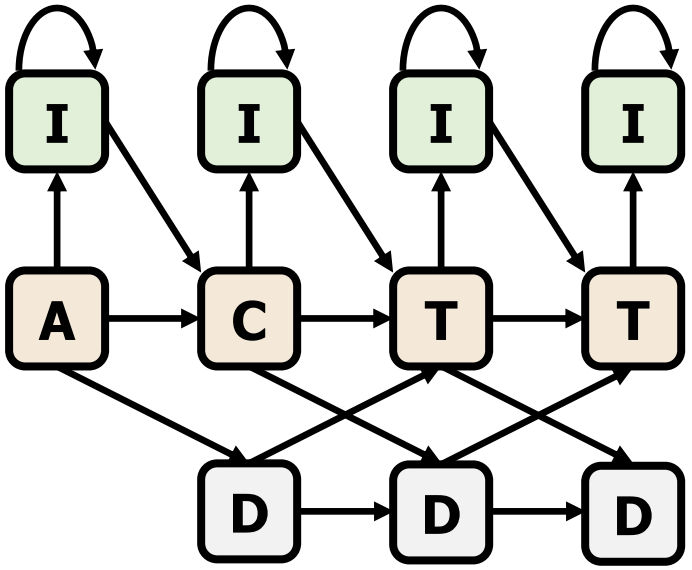
Observed Sequence #4: ATT
(D: Deletions)



Profile Hidden Markov Models

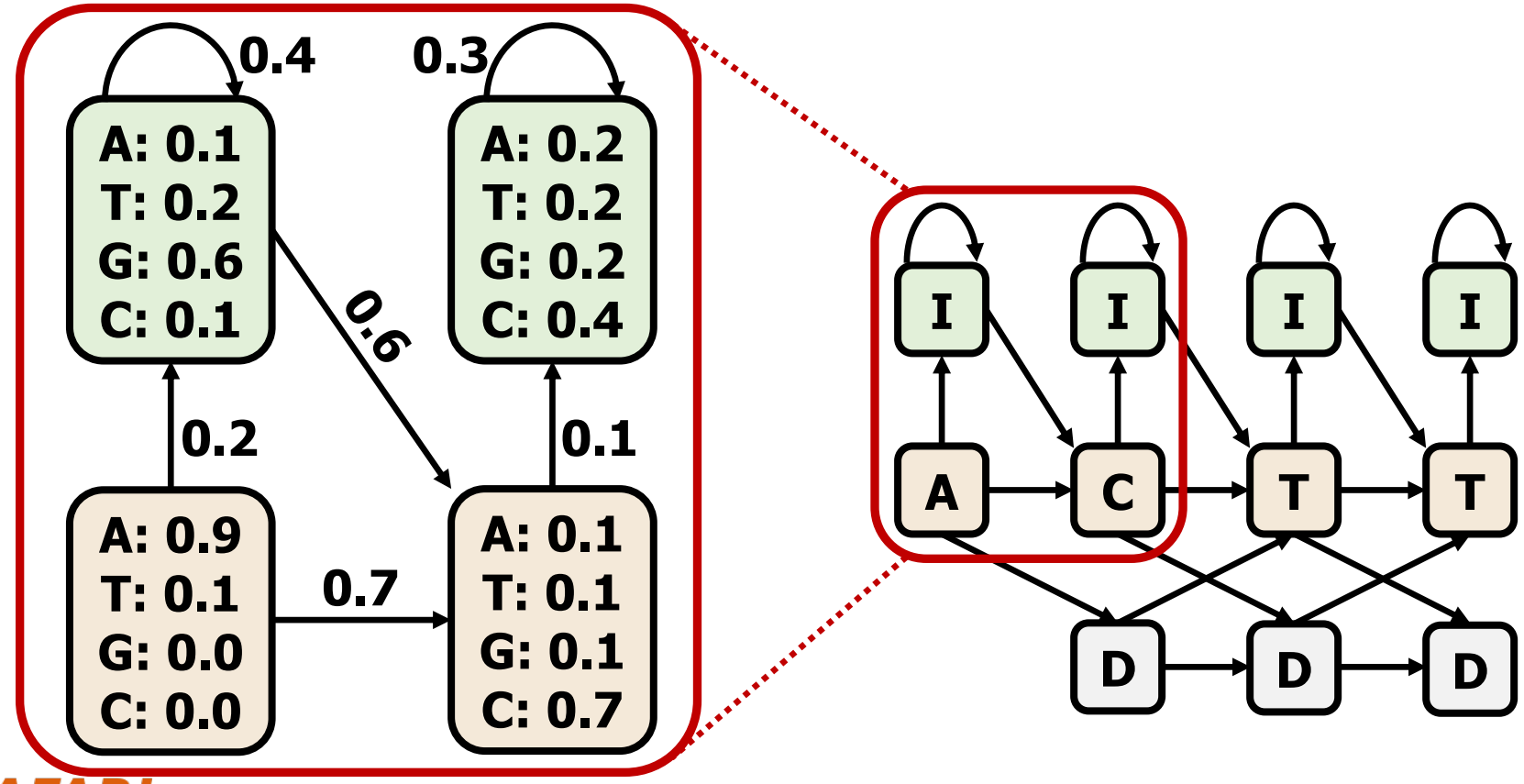
- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - Each **state** outputs a biological character (**emission**) when visited
  - States are visited via **transitions** (edges) based on **observed variations**
  - **Variations:** No variation, Substitutions, Insertions, Deletions

Observed Sequence #1: ACTT
Observed Sequence #2: ACTG
Observed Sequence #3: AGGGCTT
Observed Sequence #4: ATT
…



Probabilities in pHMMs

- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**



Utilizing Probabilities in pHMMs

- **The Baum-Welch algorithm** is commonly used with pHMMs
  - For both **inference and training** by effectively utilizing the probabilities

- **Inference:** Identifying the variations between sequences

- **Training:** Maximizing parameters to observe certain variations

Forward Calculations

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

Backward Calculations

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

Updating Transition Probabilities

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

Updating Emission Probabilities

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

Utilizing Probabilities in pHMMs

- **The Baum-Welch algorithm** is commonly used with pHMMs
  - For both **inference and training** by effectively utilizing the probabilities

- **Inference:** Identifying the variations between sequences

- **Training:** Maximizing parameters to observe certain variations

Forward Calculations

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

Backward Calculations

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

Updating Transition Probabilities

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

Updating Emission Probabilities

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

Utilizing Probabilities in pHMMs

- **The Baum-Welch algorithm** is commonly used with pHMMs
  - For both **inference and training** by effectively utilizing the probabilities

- **Inference:** Identifying the variations between sequences

- **Training:** Maximizing parameters to observe certain variations

Forward Calculations

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

Backward Calculations

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

Training Step

Updating Transition Probabilities

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

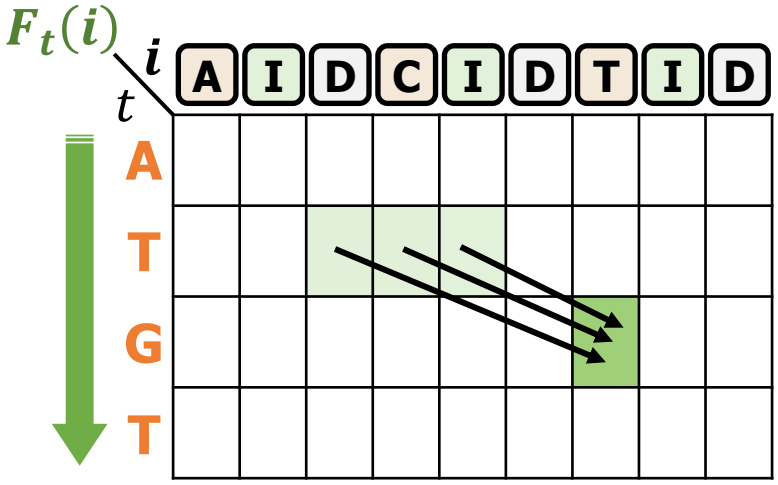
Updating Emission Probabilities

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

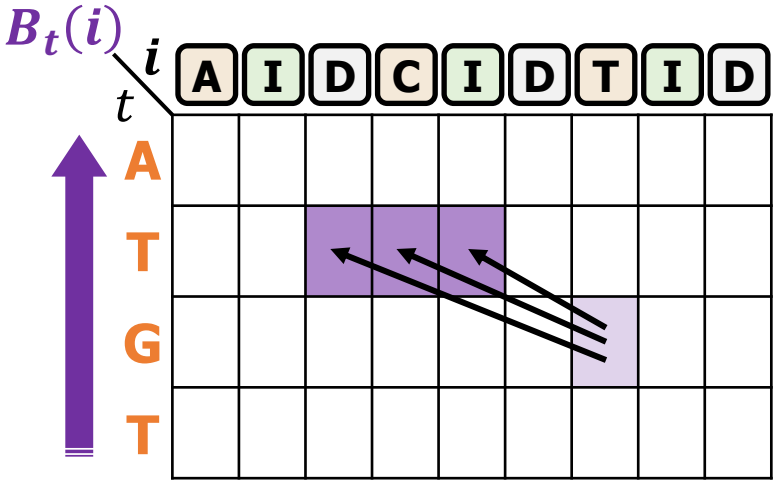
Forward & Backward Calculations

- **A dynamic programming** approach
  - Calculate the 'possibility' of visiting each state in a pHMM
  - Given an observed sequence (from both directions of the sequence)

Observed Sequence: ATGT



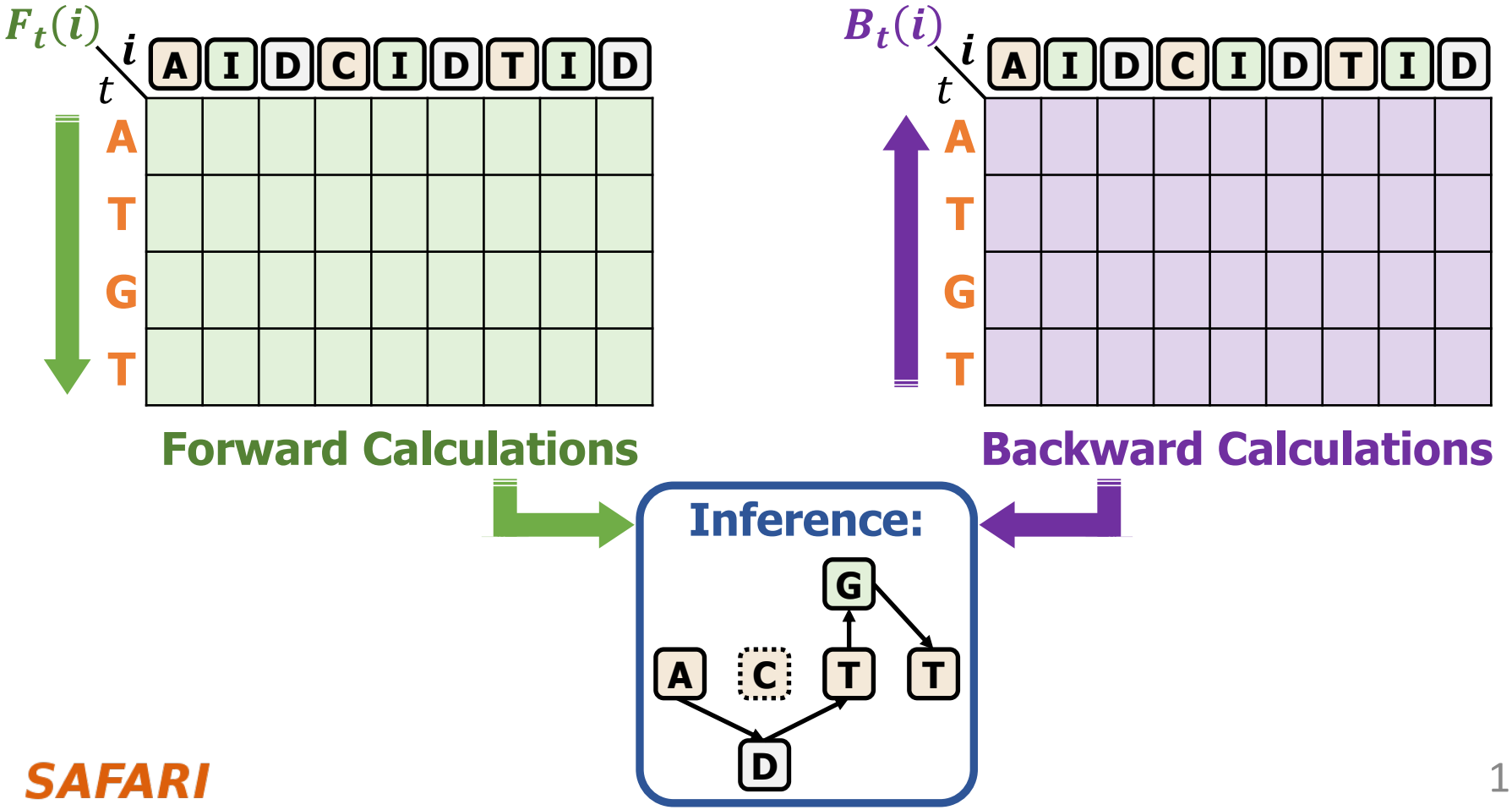
Forward Calculations



Backward Calculations

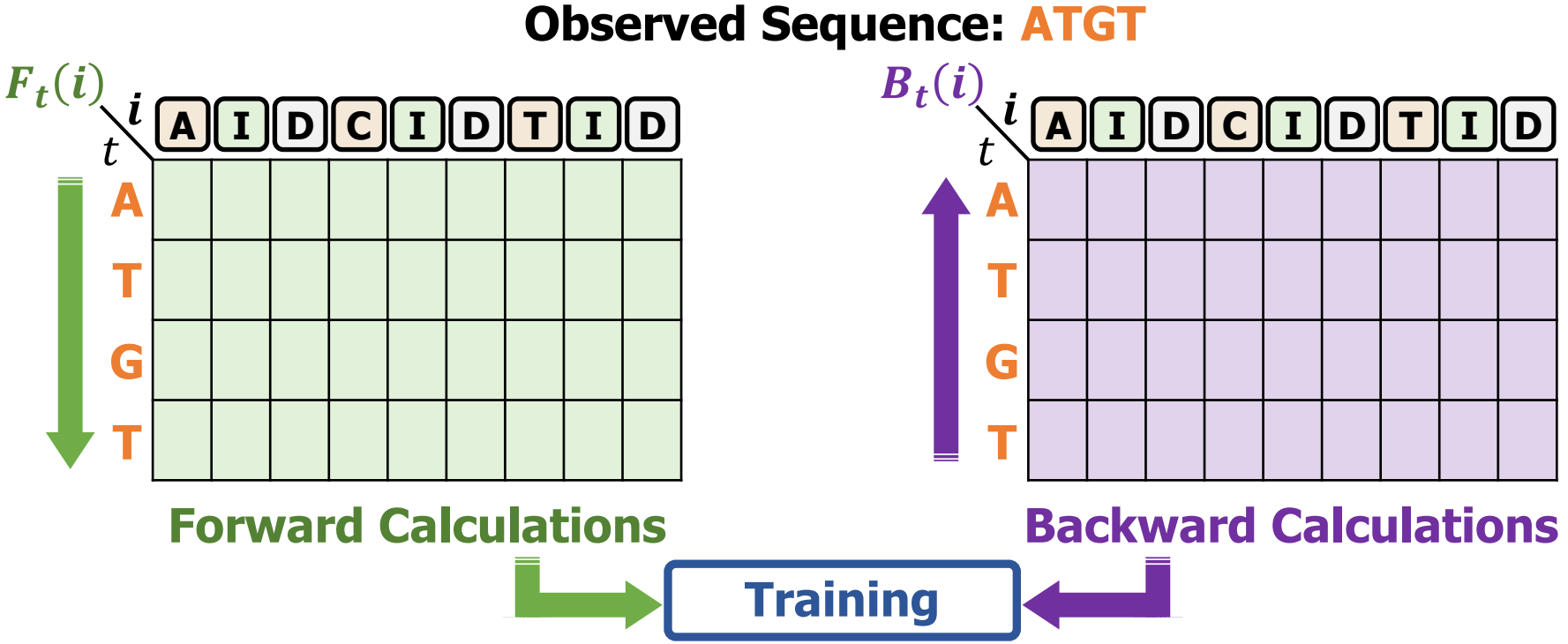
Inference using pHMMs

- **Goal:** Identifying the variations between sequences
  - **Inference** by using decoding algorithms (e.g., the Viterbi Algorithm)



Training using pHMMs

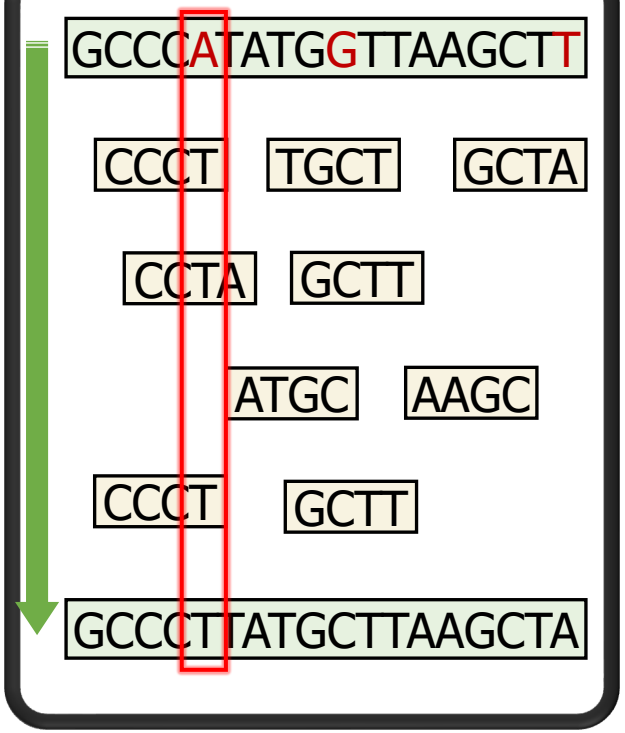
- **Goal:** Maximizing parameters to observe certain variations
  - **Training** using the parameter updating steps in the Baum-Welch algorithm



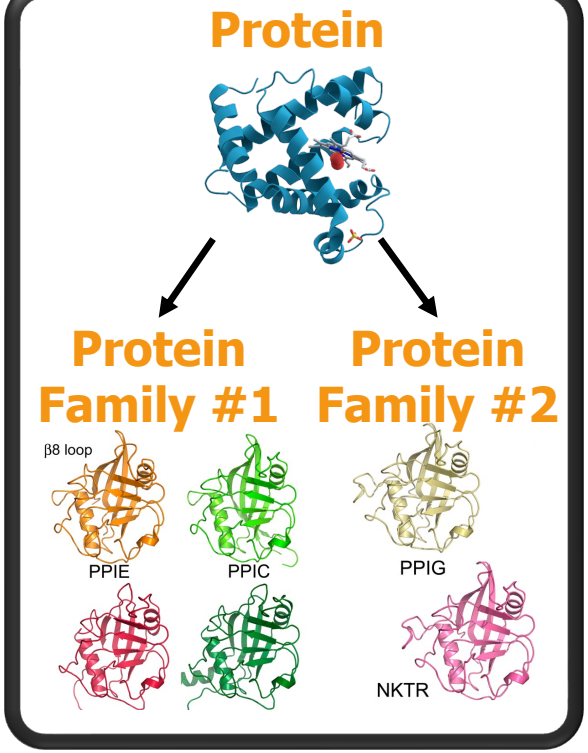
pHMMs in Genomics Workloads

- **pHMMs** are commonly used in many genomics applications

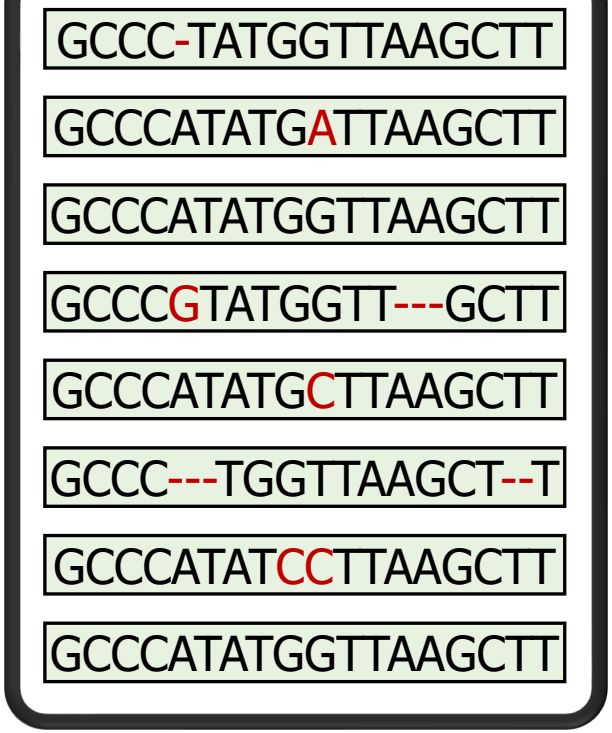
1. Error Correction



2. Protein Family Search

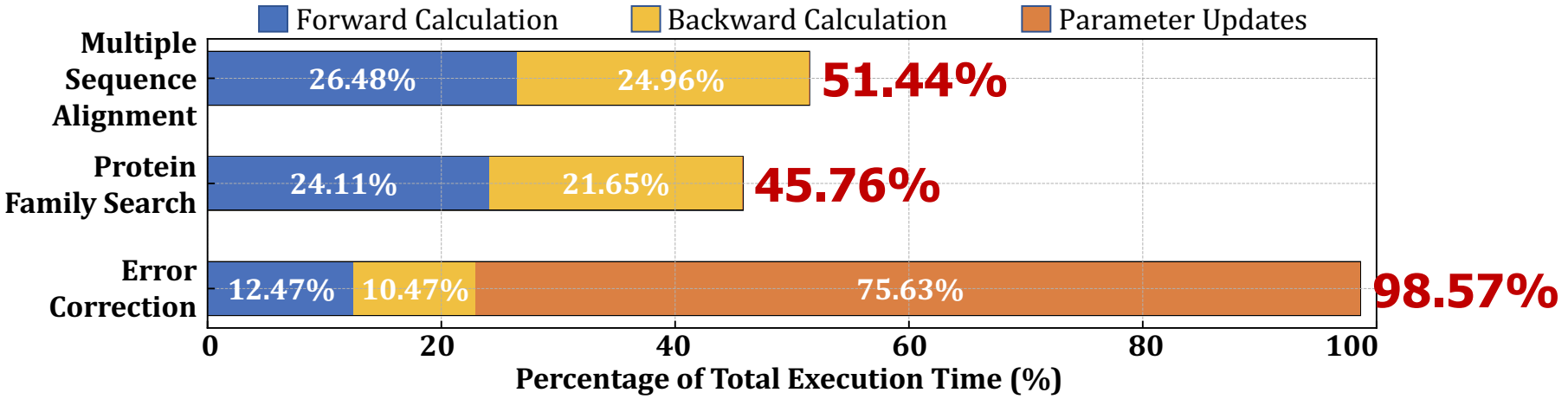


3. Multiple Sequence Alignment



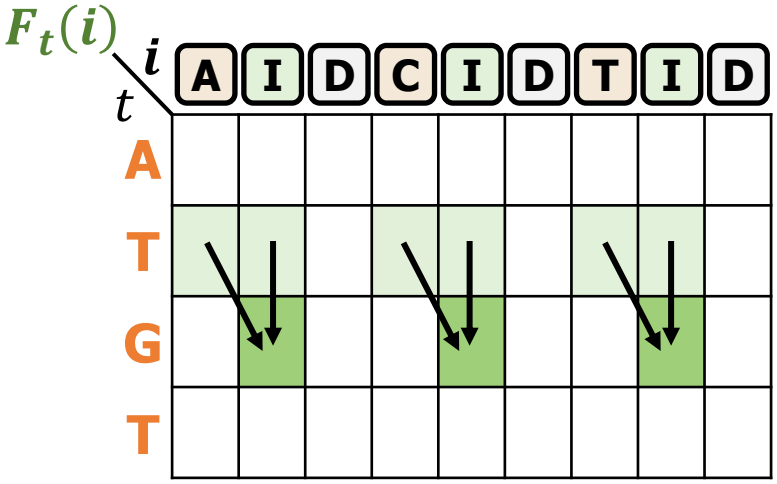
The Baum-Welch Algorithm is Costly

- The Baum-Welch algorithm causes a **major computational overhead** in genomics workloads
  - Taking up from **46% to 99% of the overall execution time**
  - Computationally complex dynamic programming calculations
  - Compute intensive many floating-point operations

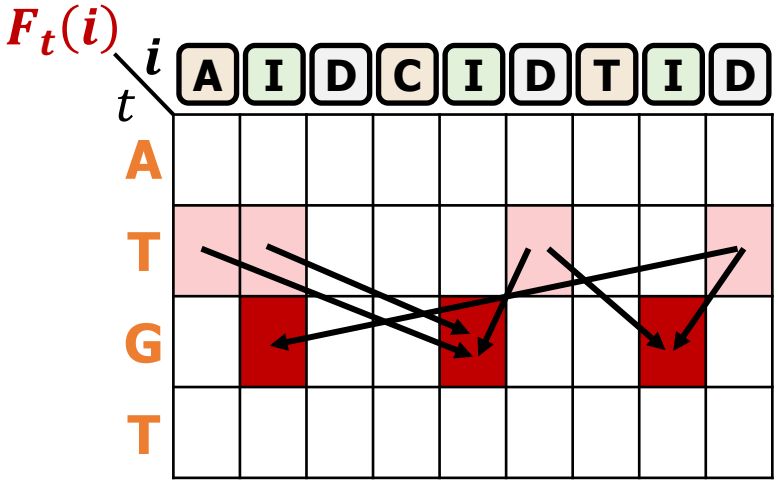


Existing Solutions are Ineffective

- pHMMs are specialized version of **Hidden Markov Models (HMMs)** with **fixed patterns** on states and transitions



Forward Calculations in pHMMs

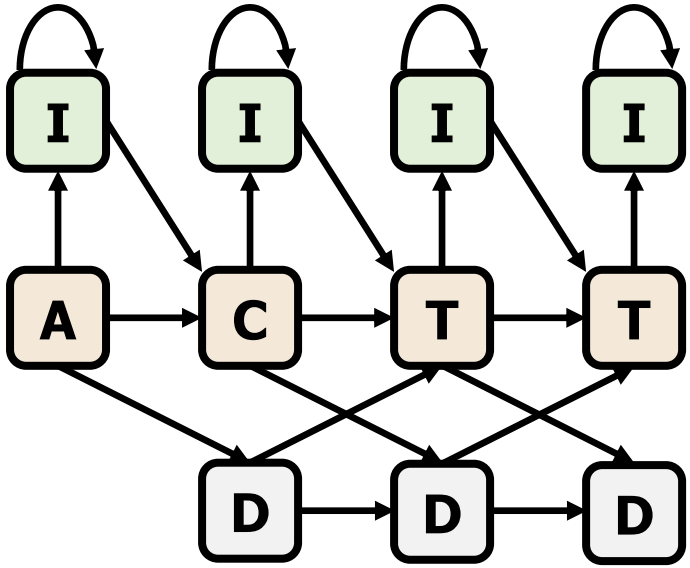


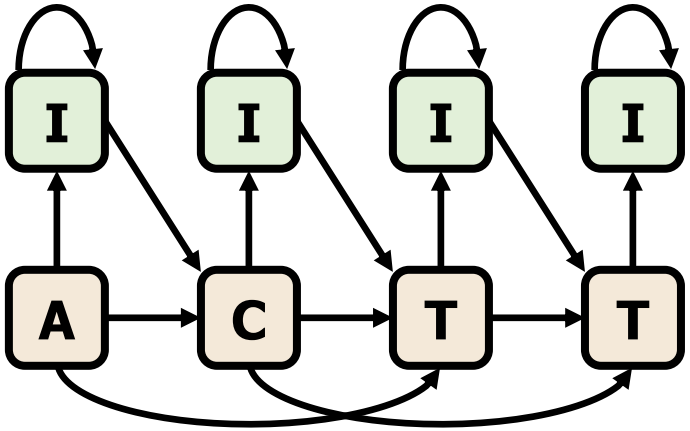
Forward Calculations in HMMs

Generic HMM accelerators **cannot exploit the fixed data dependency pattern** of pHMMs

Existing Solutions are Inflexible

- pHMM requirements can change based on the application
  - **Different pHMM designs:**





  - **Different alphabet sizes**: DNA (4 letters), protein (20 letters)

Lack of **flexible mechanisms**
to handle different design choices

Existing Solutions are Inefficient

- Suboptimal vectorization of SIMD-based solutions on CPUs and GPUs
  - High warp divergence, branching, low port utilization…
- A significant portion of the floating-point operations in dynamic programming is redundant
  - Same multiplications results can redundantly be computed during training
  - Unnecessary data movements

Existing solutions provide suboptimal solutions due to **inefficient hardware of software design**

The Problem

The Baum-Welch algorithm causes major performance overhead in important genomics applications

Hardware- or software-only solutions are not sufficient for effectively accelerating pHMMs

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Goal

Enable **rapid, power-efficient, and flexible** use of pHMMs when using the Baum-Welch algorithm

ApHMM

The first flexible hardware-software co-designed acceleration framework that can significantly reduce the computational overhead of the Baum-Welch algorithm for pHMMs

ApHMM-GPU: The first GPU implementation of the Baum-Welch algorithm for pHMMs

Key Software & Hardware Optimizations

- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering
- **Avoid repeated operations** by utilizing lookup tables

SW

- **Reduce data movement** by exploiting fixed data pattern
- **Flexible and efficient** control logic and hardware design

HW

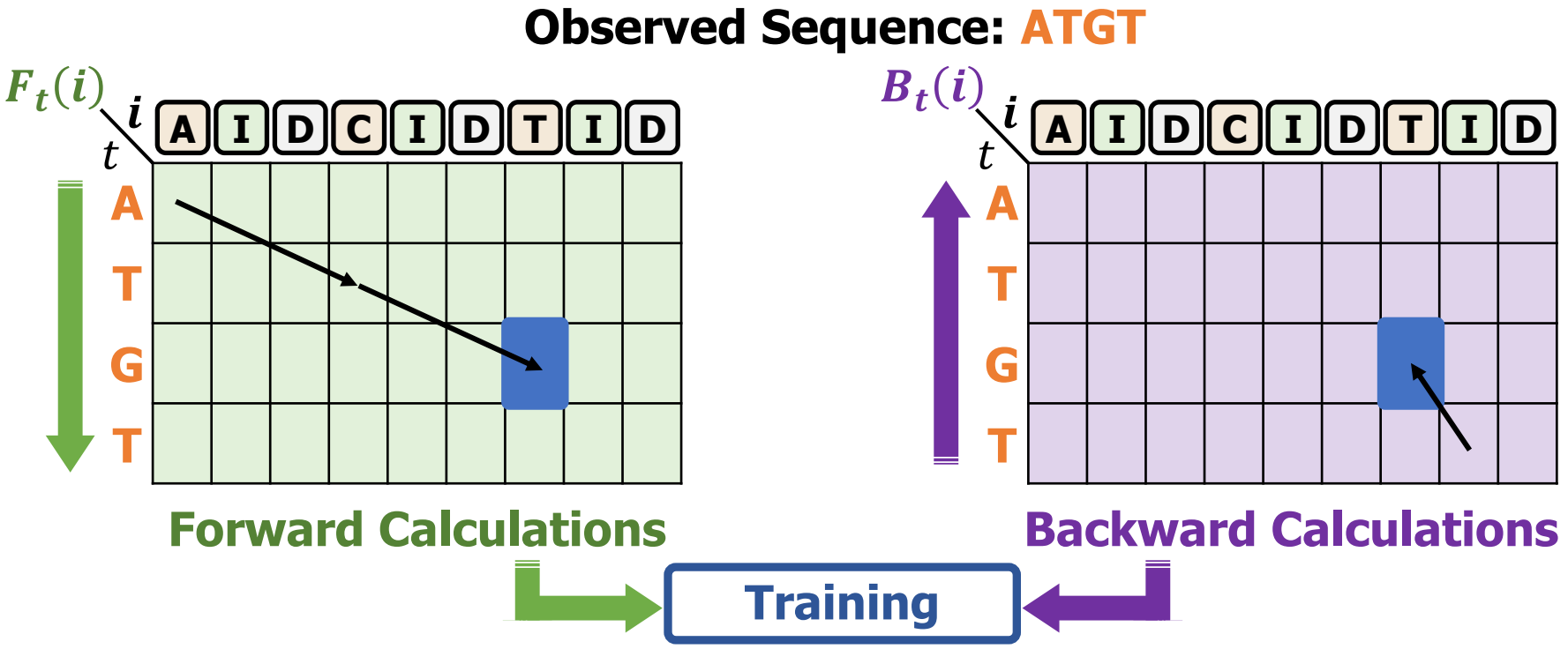
Key Software & Hardware Optimizations

- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering
- **Avoid repeated operations** by utilizing lookup tables

SW

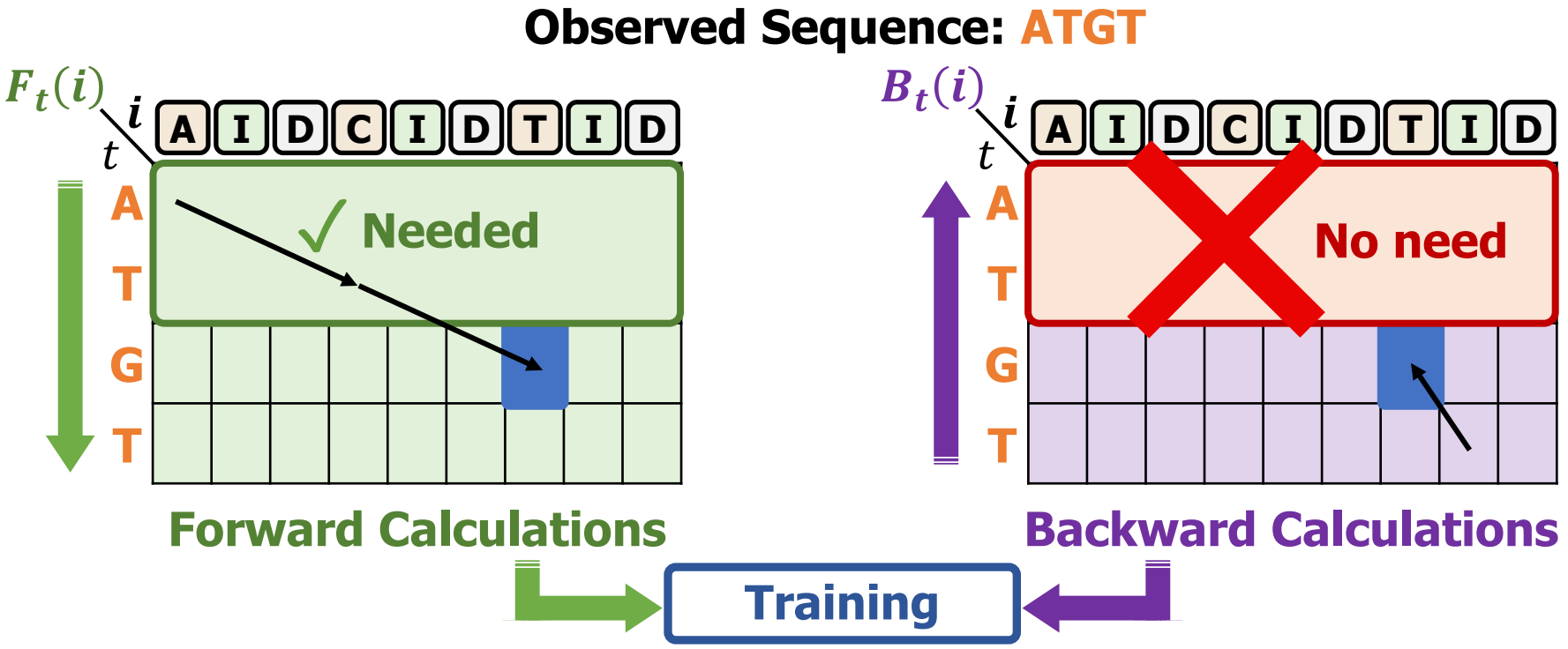
SW: Minimizing Redundant Storage

- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value



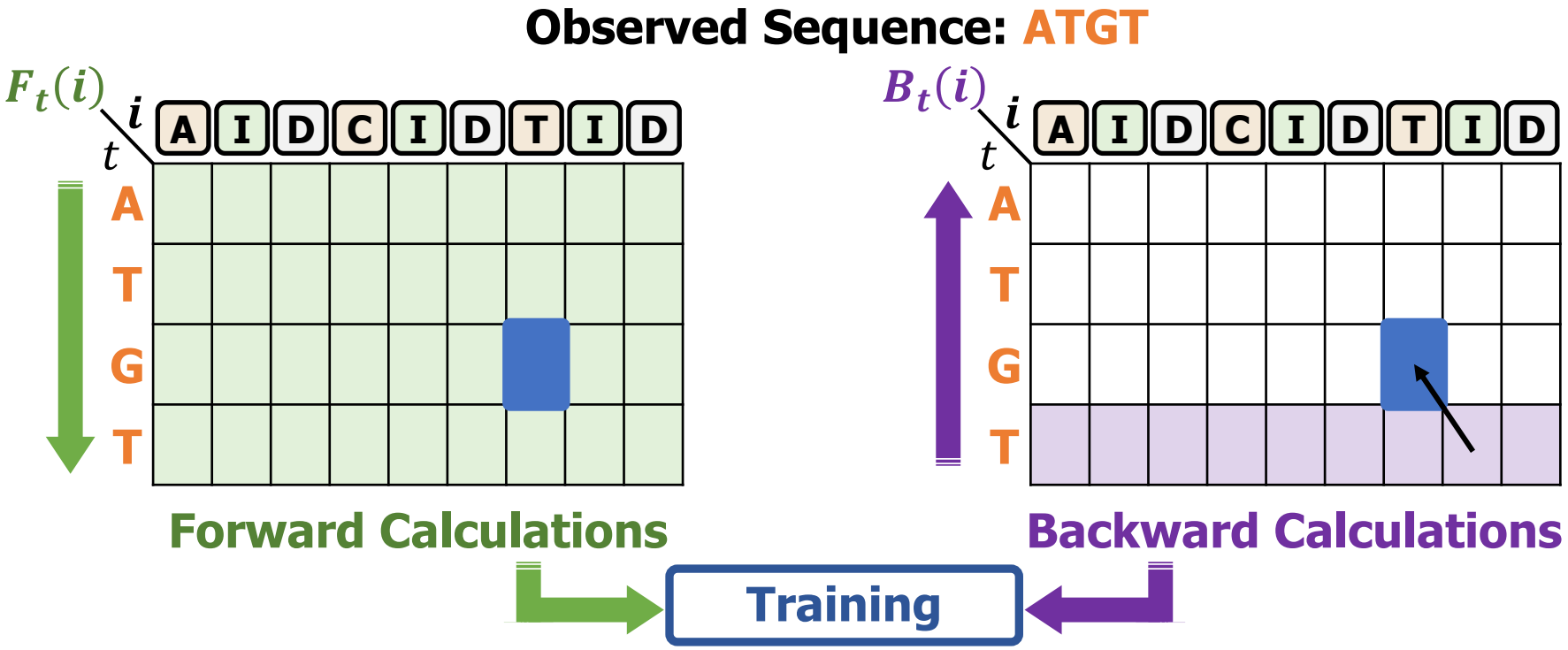
SW: Minimizing Redundant Storage

- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value



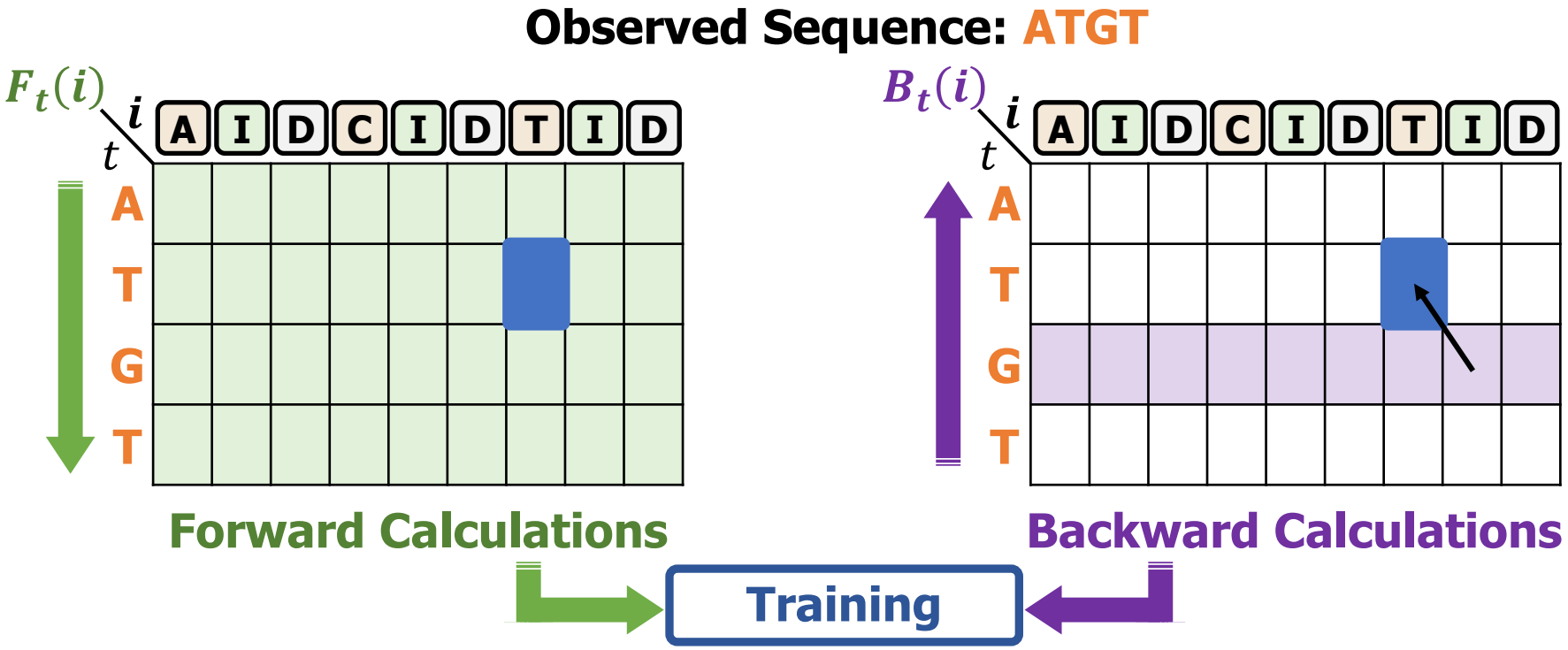
SW: Minimizing Redundant Storage

- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value
  - **Partial compute approach:** Only a single row should be **fully stored**



SW: Minimizing Redundant Storage

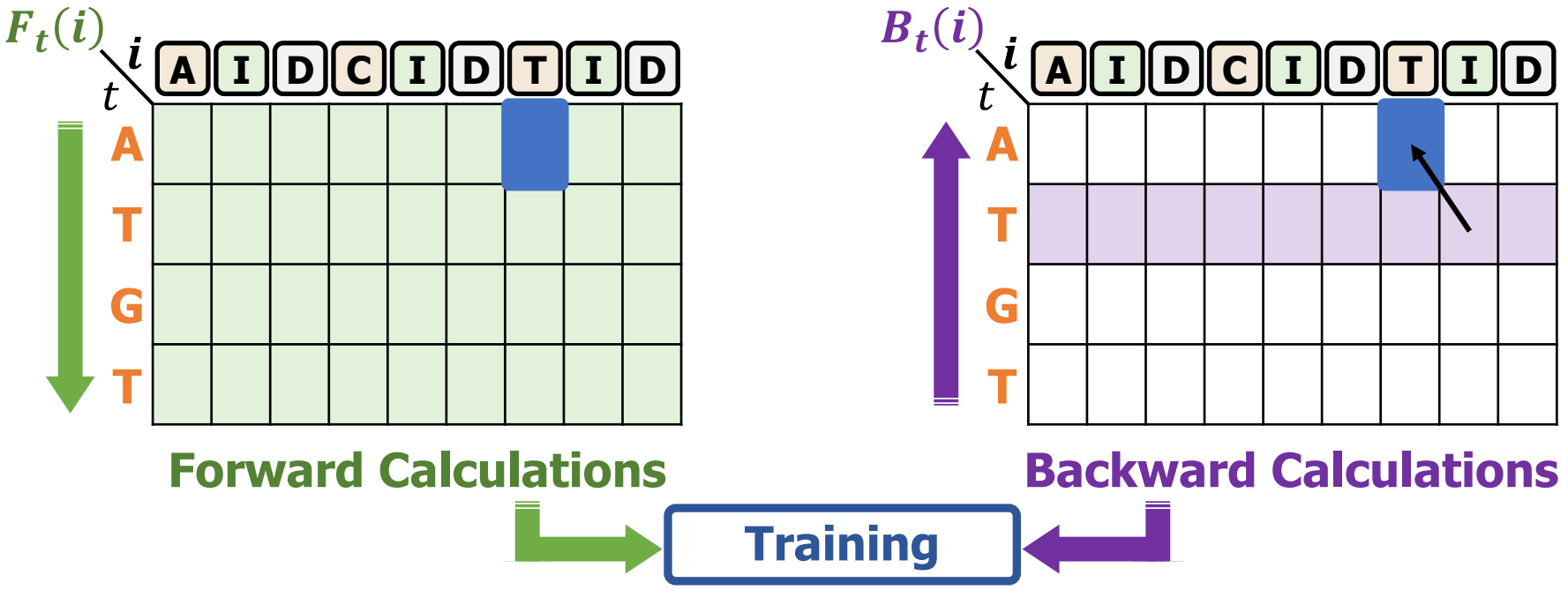
- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value
  - **Partial compute approach:** Only a single row should be **fully stored**



SW: Minimizing Redundant Storage

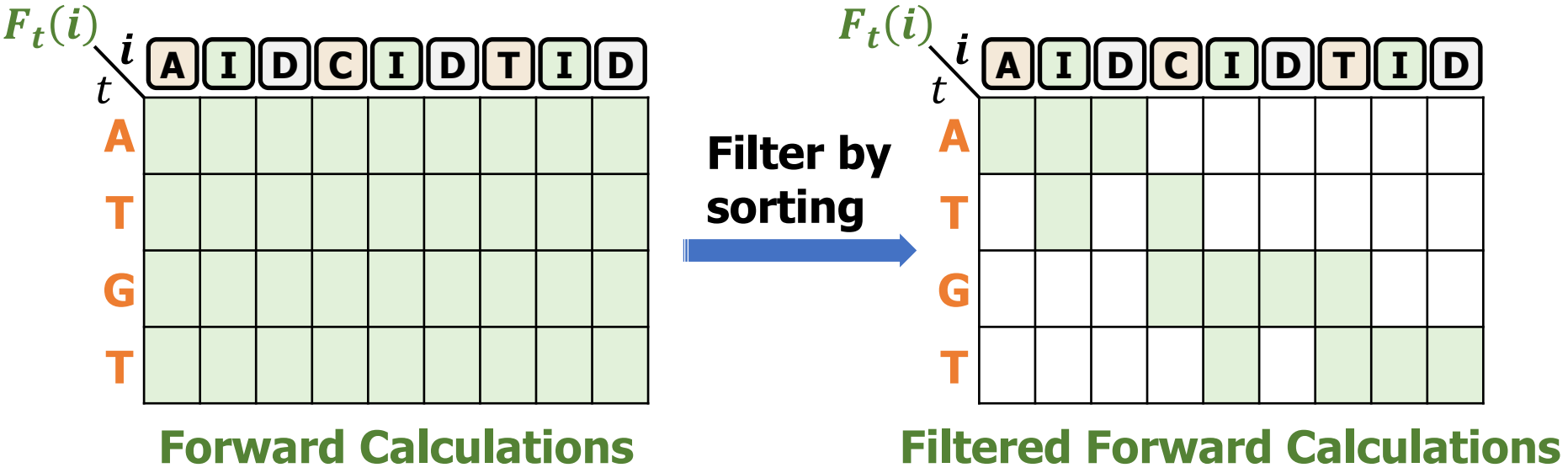
- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value
  - **Partial compute approach:** Only a single row should be **fully stored**
  - **Reduces the storage requirements** during training

Observed Sequence: ATGT



SW: Reducing Unnecessary Computations

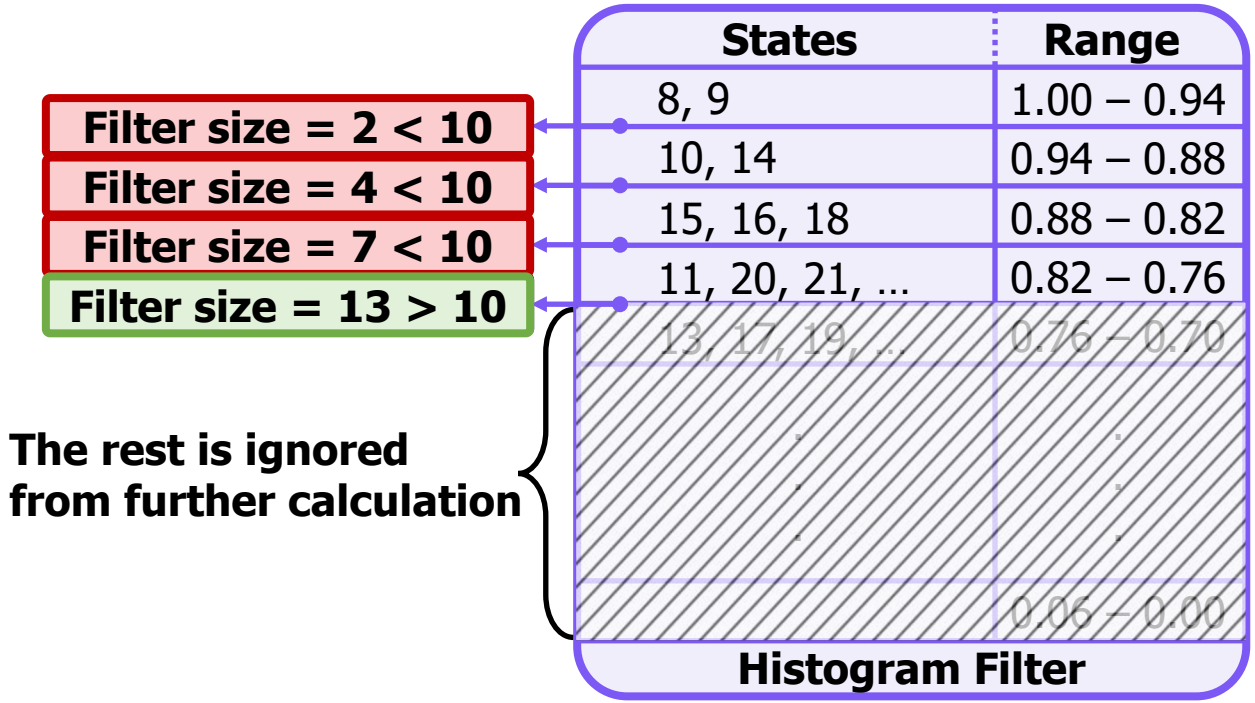
- **Observation:** 'Negligible' cells can be ignored without significantly reducing overall accuracy
  - **Filtering:** Non-negligible states are identified by sorting
  - **Sorting** to find **exactly** n states with **largest** Forward or Backward values



- **Sorting is complex** to implement in hardware (and costly)
  - Can we filter without sorting?

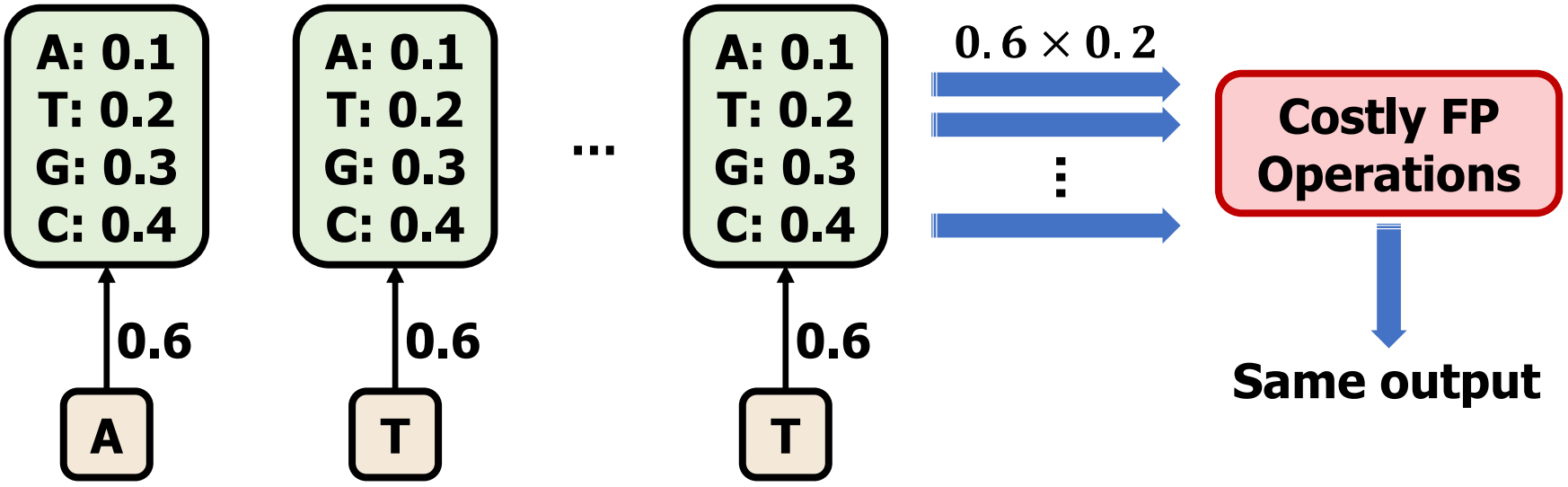
SW: Reducing Unnecessary Computations

- **Observation:** 'Negligible' cells can be ignored without significantly reducing overall accuracy
  - **Goal:** Find **at least** n states with largest Forward and Backward values
  - **Histogram-based filtering:** Placing the states into buckets corresponding to a range of values
  - Filter is full as soon we find **at least n states (e.g., $n = 10$)**



SW: Avoiding Repeated Operations

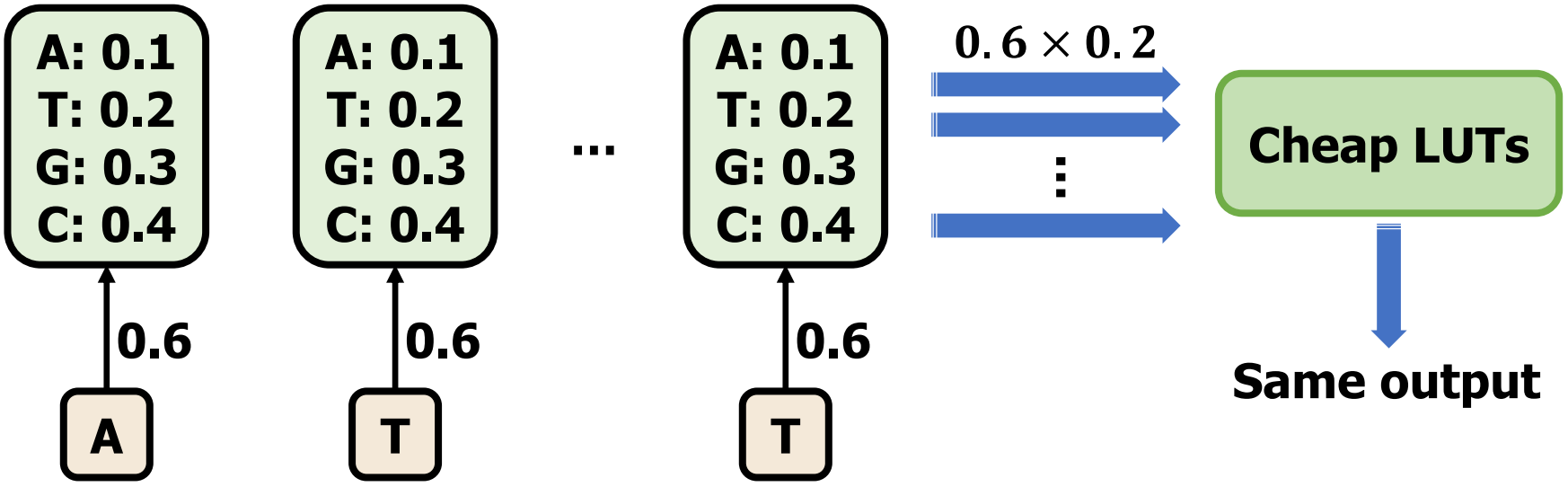
- **Observation:** Same multiplications are redundantly performed
  - **Same default values are used** for each possible connection in pHMMs
  - **Fixed connection patterns** generate a fixed set of multiplication results



- **Goal:** Avoid redundant computations
  - By enabling efficient reuse of the common multiplications results

SW: Avoiding Repeated Operations

- **Observation:** Same multiplications are redundantly performed
  - **Same default values are used** for each possible connection in pHMMs
  - **Fixed connection patterns** generate a fixed set of multiplication results



- **Goal:** Avoid redundant computations
  - By enabling efficient reuse of the common multiplications results
  - **Lookup tables (LUTs)** to efficiently store and use these common results

Key Software & Hardware Optimizations

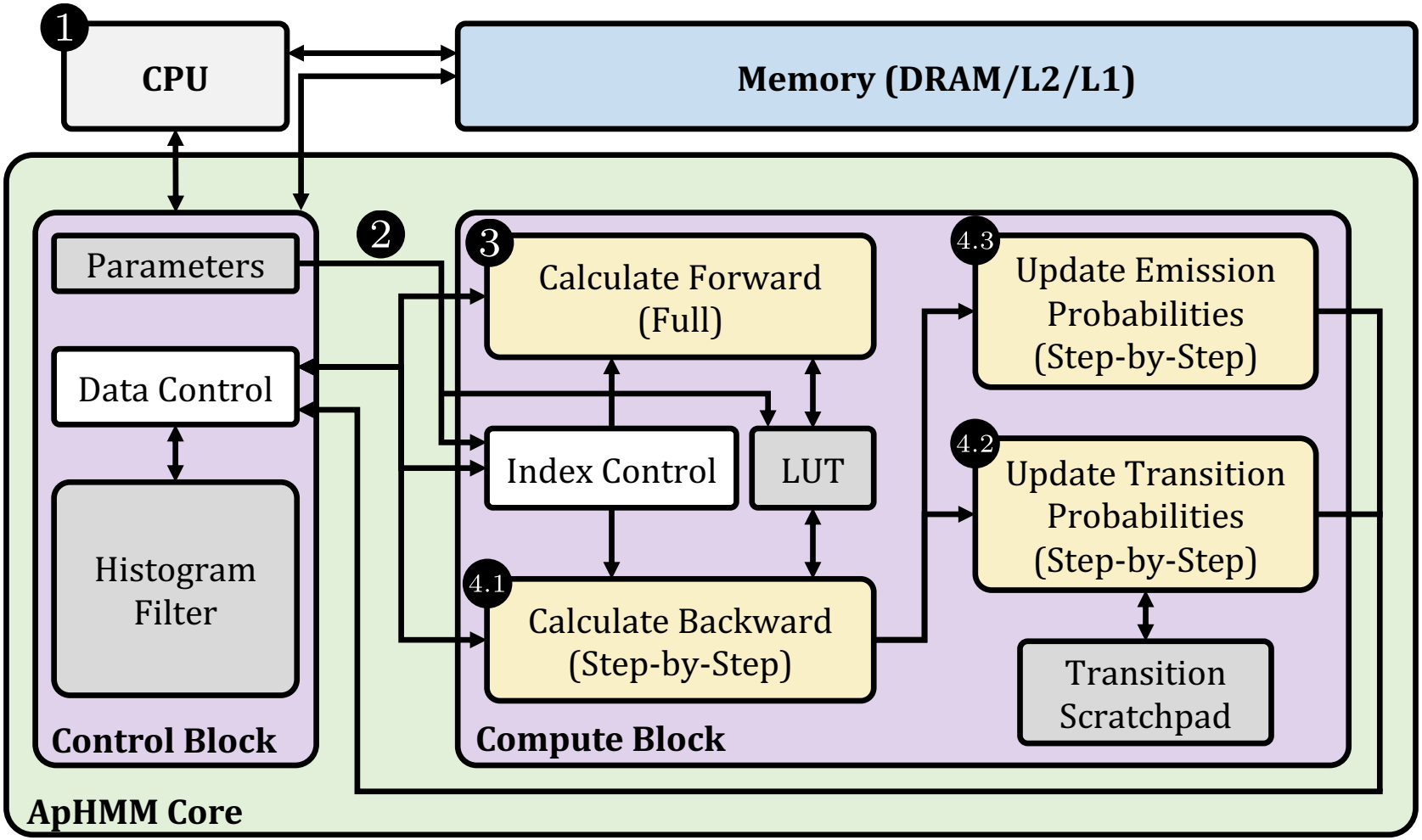
- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering
- **Avoid repeated operations** by utilizing lookup tables

SW

- **Reduce data movement** by exploiting fixed data pattern
- **Flexible and efficient** control logic and hardware design

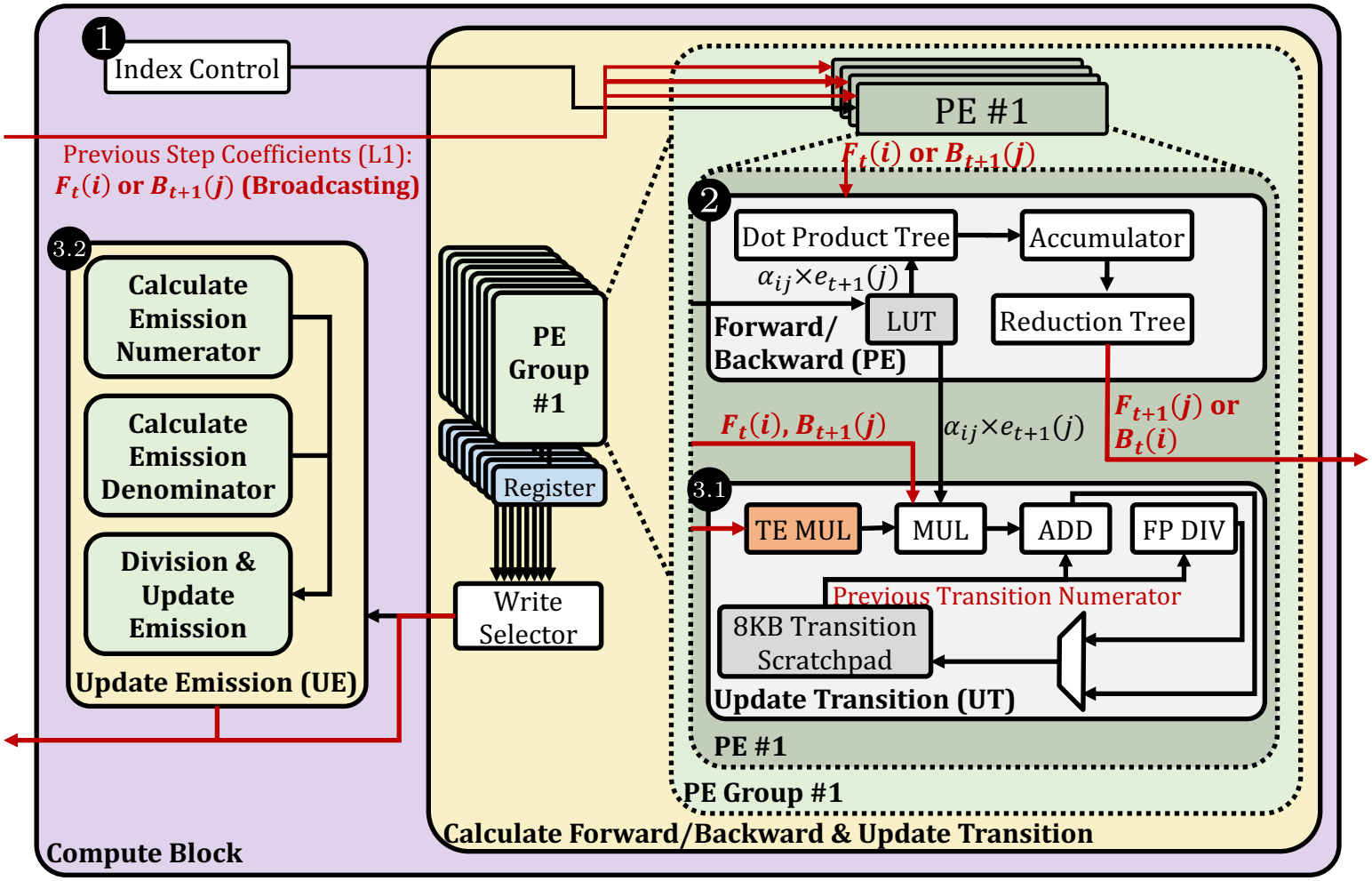
HW

Overview of ApHMM Design



✓ **Flexible and efficient control logic and hardware** design enables opting out from heuristics and supporting different pHMM designs

Computing the Baum-Welch in ApHMM



Efficiently exploiting data locality, broadcasting, memoization, streaming, and pipelining with our SW optimizations for an effective HW-SW co-design

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Evaluation Methodology

- **Performance, Area, and Power Analysis:**
  - Synthesized SystemVerilog Model in a 28nm process @1GHz
  - **CPU baseline:** AMD EPYC 7742 @2.26GHz (1, 12, 32 threads)
  - **GPU baselines:** Titan V & A100
  - **FPGA baseline:** FPGA D&C

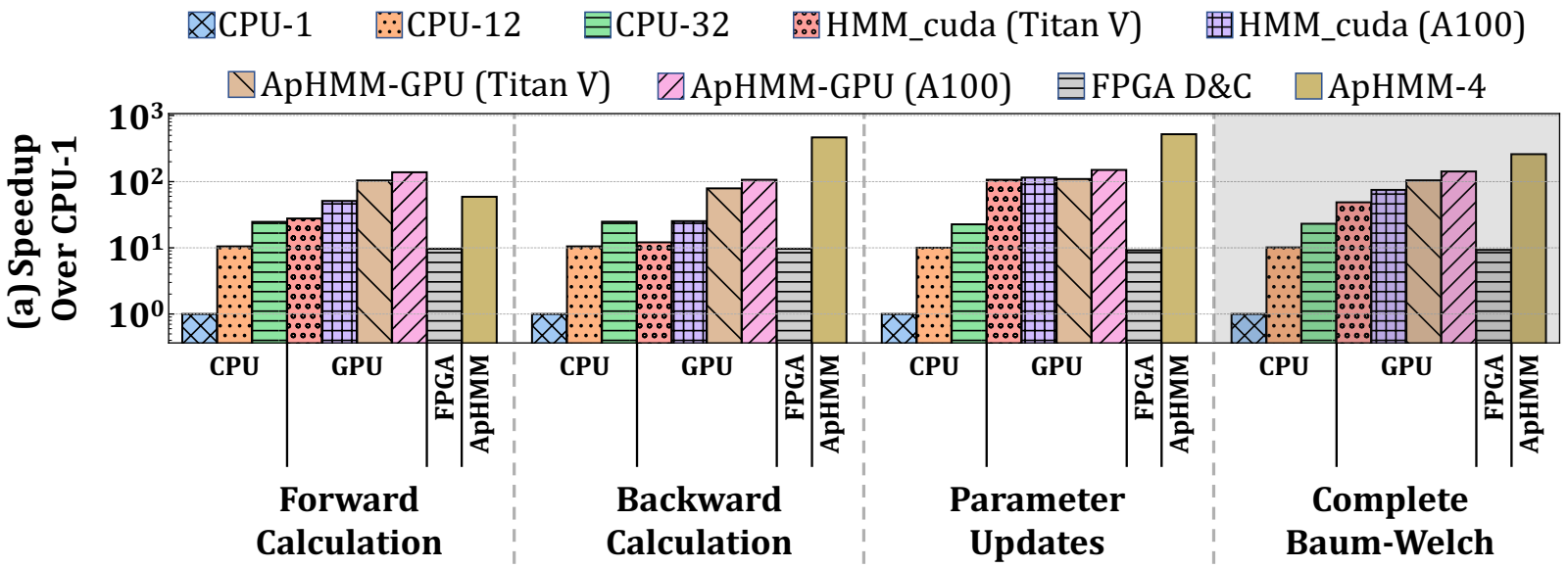
- **Use cases** and their software baseline:
  1. Error Correction – Apollo
  2. Protein Family Search – HMMER
  3. Multiple Sequence Alignment – HMMER

Evaluation Methodology

- **Comparison Points**
  - CPU: Apollo, HMMER
  - GPU: ApHMM-GPU, HMM_cuda
  - FPGA: FPGA D&C

- **Datasets**
  - Error correction: **Real 10,000 DNA sequences** from Escherichia coli (*E. coli*) with average 5,128 read length
  - Protein family search: Entire Pfam database (**19,632 pHMMs**) and **real 214,393 protein sequences** from Mitochondrial carrier
  - Multiple sequence alignment: Aligning over **~1 million protein sequences** from Pfam database

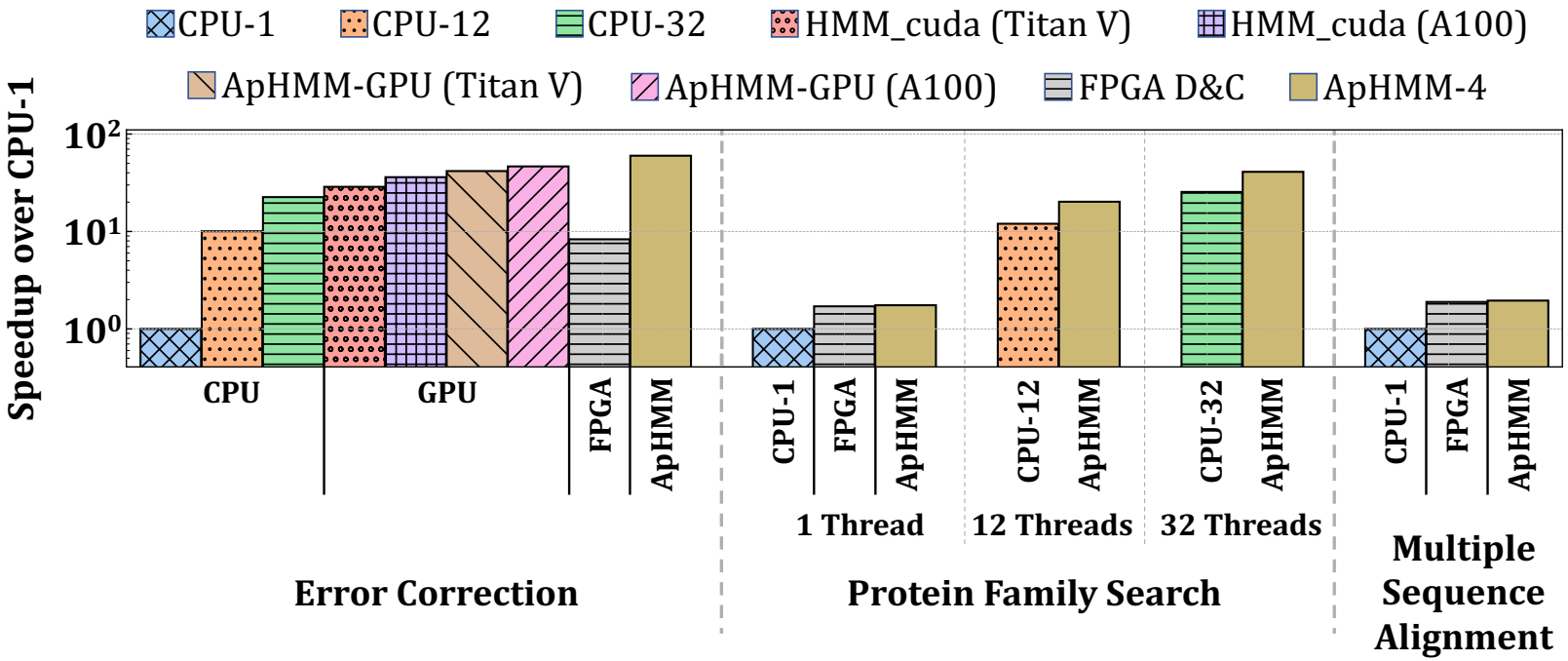
Performance: The Baum-Welch Algorithm



15.55×–260.03×, 1.83×–5.34×, and 27.97× faster than the CPU, GPU, and FPGA implementations of the Baum-Welch algorithm

GPUs provide **better performance for Forward calculations** due to frequent off-chip memory accesses in ApHMM during Forward calculation

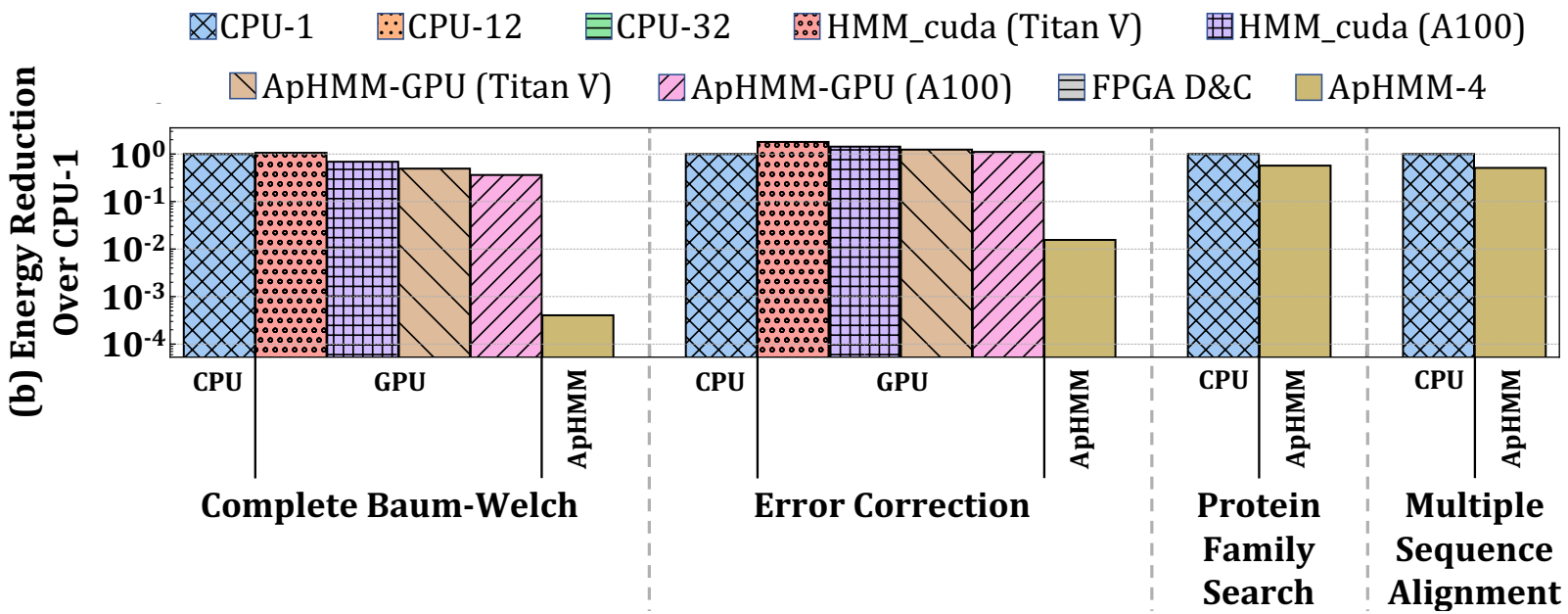
Performance: Workload Acceleration



1.29×–59.94×, 1.03×–1.75×, and 1.03×–1.95× better performance compared to the CPU, GPU, and FPGA baselines

Error correction benefits most from the acceleration due to **frequent and costly training**

Energy: Overall Comparisons



For the Baum-Welch algorithm: **2474.09× and 896.70×–2622.94×** reduction in energy consumption compared to CPU-1 and GPU implementations

For the workloads: **64.24×, 1.75×, and 1.96×** reduction compared to CPU-1

Speedup of Each Optimization

- We analyze the speedup that each optimization provides over the CPU baseline

Optimization	Speedup (×)
Histogram Filter	1.07
LUTs	2.48
Broadcasting and Partial Compute	3.39
Memoization	1.69
Overall	15.20

Broadcasting and partial compute together is only possible

with an efficient HW-SW co-design

Area and Power

- We analyze the **area and power for ApHMM-4** using the Synopsys Design Compiler with a 28nm process @1GHz:

Module Name	Area (mm^2)	Power (mW)
Control Block	0.011	134.4
64 Processing Engines (PEs)	1.333	304.2
64 Update Transitions (UTs)	5.097	0.8
4 Update Emissions (UEs)	0.094	70.4
Overall	6.536	509.8
128 KB L1-Memory	0.632	100

UTs require the largest area due to several complex units such as multiplexer, division pipeline, and local memory

ApHMM can significantly accelerate pHMMs with relatively small area and power requirements

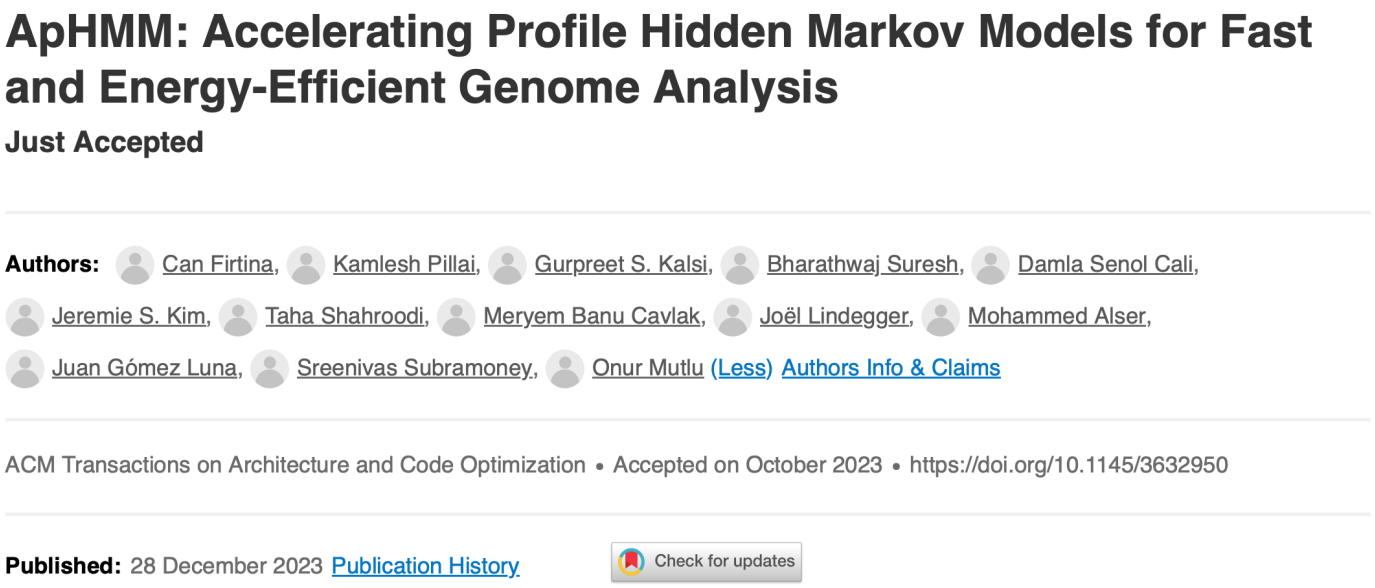
More in the Paper

- **More Results**
  - Detailed discussion on the results generated per use case
  - Justification of the dataset and baseline choices

- **Details of all mechanisms and configurations**
  - Details of our design space exploration
  - Data distribution and memory layout
  - Control and execution flow of ApHMM cores
  - Related work discussion (e.g., Pair HMMs vs pHMMs)
  - Detailed background on the equations and algorithms

ApHMM

- Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, and Onur Mutlu,
  "ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis"
  ACM TACO, Dec 2023.
  [Online link at ACM TACO]
  [arXiv preprint]
  [ApHMM Source Code]



ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis

Just Accepted

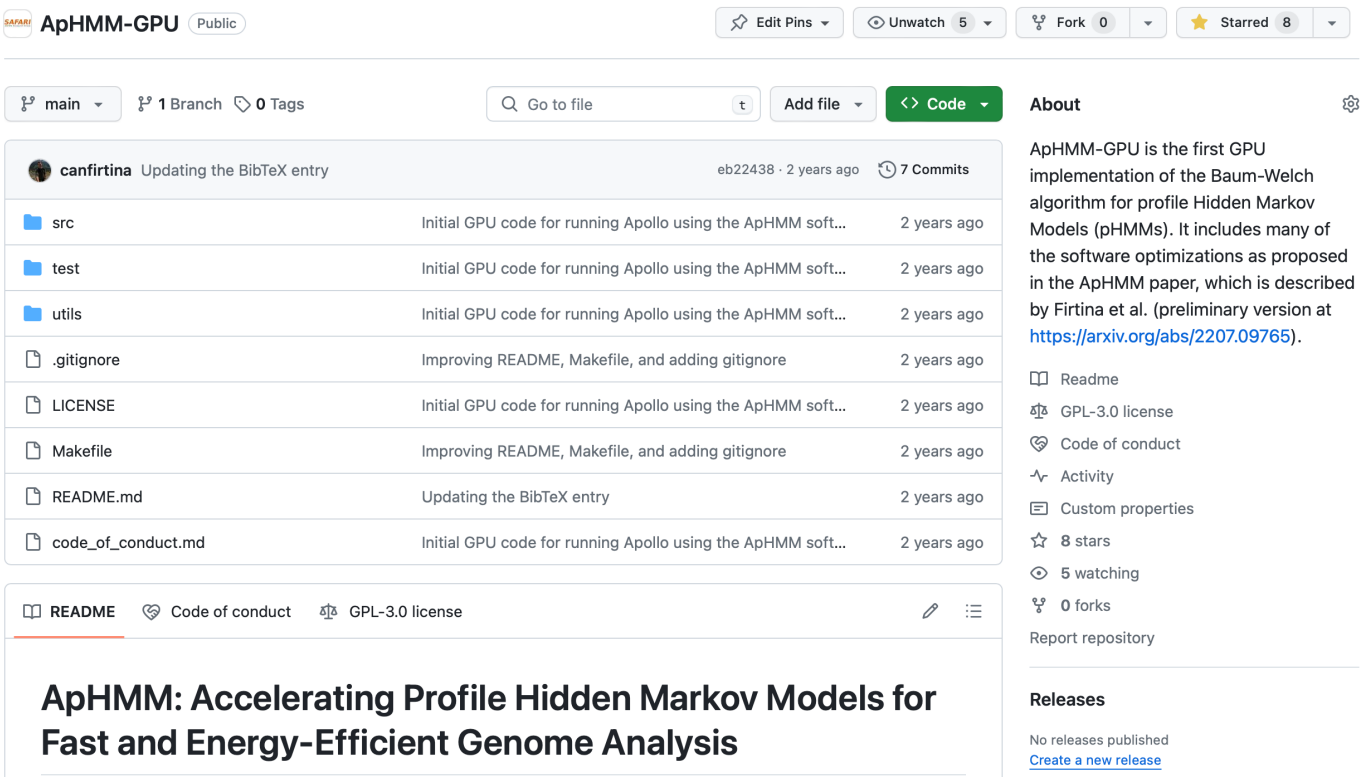
Authors: Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, Onur Mutlu (Less) Authors Info & Claims

ACM Transactions on Architecture and Code Optimization • Accepted on October 2023 • https://doi.org/10.1145/3632950

Published: 28 December 2023 Publication History

Check for updates

ApHMM-GPU Source Code



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

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Conclusion

Goal: Enable rapid, power-efficient, and flexible use of pHMMs for genomics workloads

ApHMM: the first flexible and hardware-software accelerator for pHMMs that can

1) Substantially reduce unnecessary data storage, data movement, and computations by effectively co-designing hardware and software together
2) Provide a flexible design to support several genomics workloads that use pHMMs

Key Results: Our ASIC implementation compared to CPU, GPU, and FPGA baselines across 3 workloads

- **15.55×–260.03×, 1.83×–5.34×, and 27.97× better performance**
- **Up to 2622.94×** reduction in **energy consumption**

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

HiPEAC 2024

ApHMM

Paper

Accelerating Profile Hidden Markov Models for Fast and Energy-Efficient Genome Analysis

Can Firtina

canfirtina@gmail.com
https://cfirtina.com

Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joël Lindegger, Mohammed Alser, Juan Gómez Luna, Sreenivas Subramoney, Onur Mutlu

SAFARI ETH zürich

intel TU Delft Carnegie Mellon

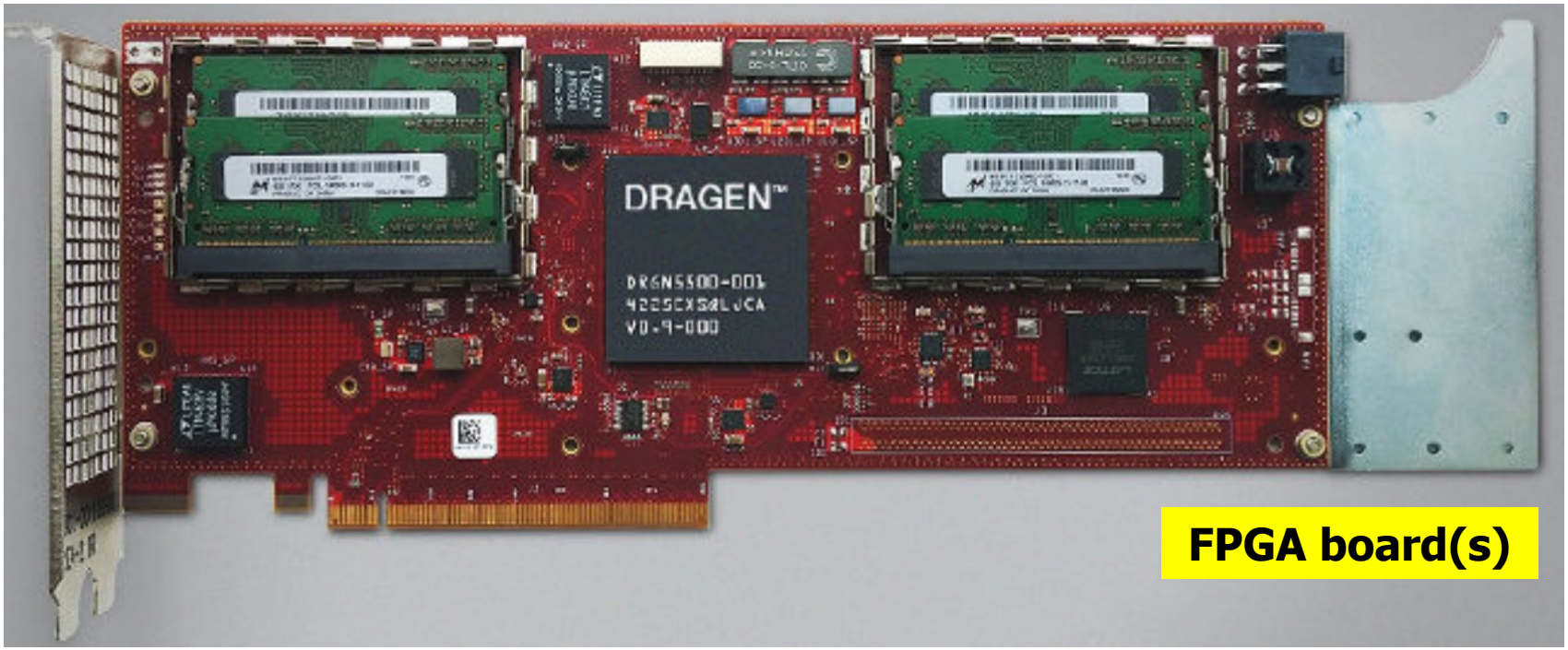
Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
  - Intelligent genome analysis
- Enabling Fast and Accurate Real-time Analysis
  - RawHash and RawHash2
- Graph & ML Acceleration in Genomics
  - ApHMM
- Conclusion

Things Are Happening In Industry

Illumina DRAGEN Bio-IT Platform (2018)

- Processes whole genome at 30x coverage in ~25 minutes with hardware support for data compression



emea.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html
emea.illumina.com/company/news-center/press-releases/2018/2349147.html

NextSeq 2000 with Analysis Capability

NextSeq 1000/2000 Integrates DRAGEN Bio-IT Platform On-Board

DRAGEN Bio-IT platform:

- Fast
- Accurate
- Industry standard pipelines
- For both novice and expert users

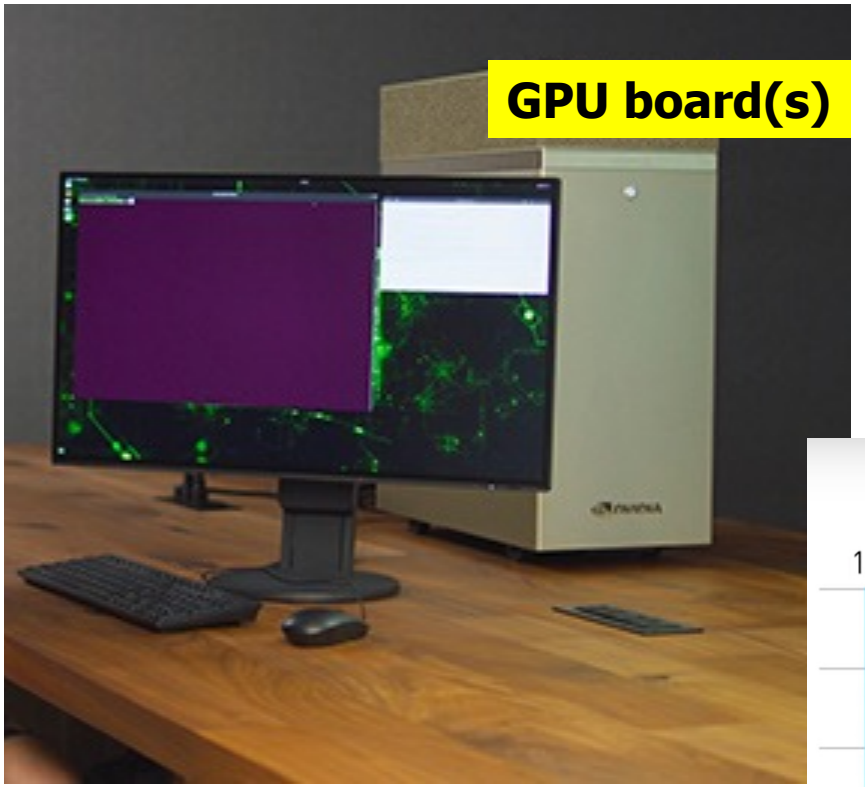
Pipelines available on-board:

- DRAGEN Enrichment pipeline
- DRAGEN RNA pipeline
- DRAGEN Germline
- DRAGEN Single Cell RNA
- Generate FASTQ via BCL Convert
- *Additional pipelines available in BaseSpace Sequence Hub*

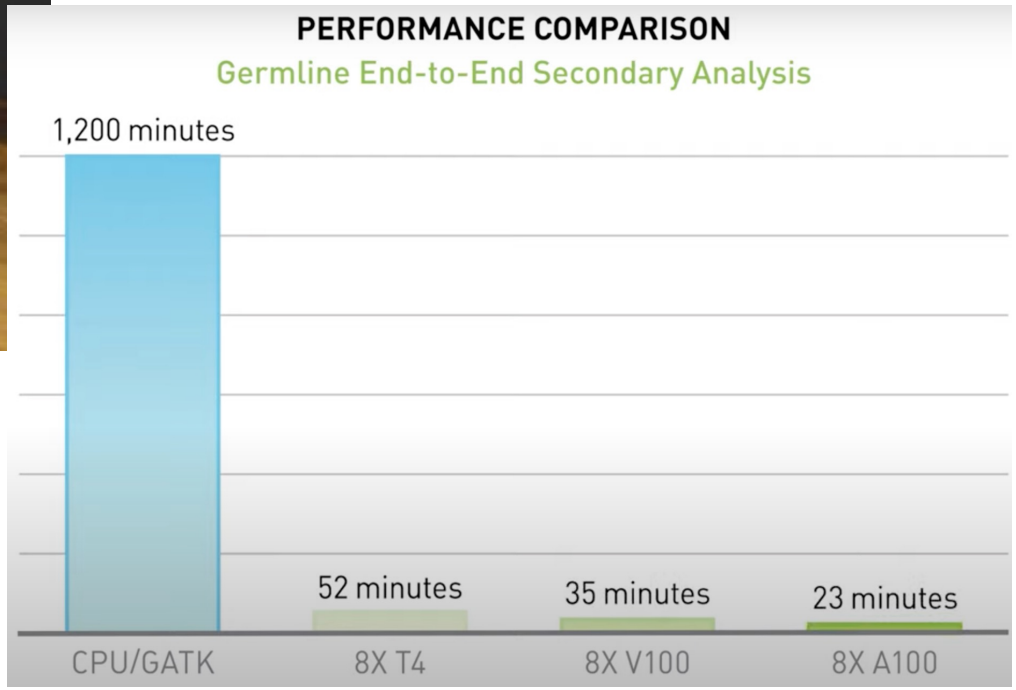
illumina

For Research Use Only.
Not for use in diagnostic procedures.

NVIDIA Clara Parabricks (2020)



A University of Michigan startup in 2018 joined NVIDIA in 2020

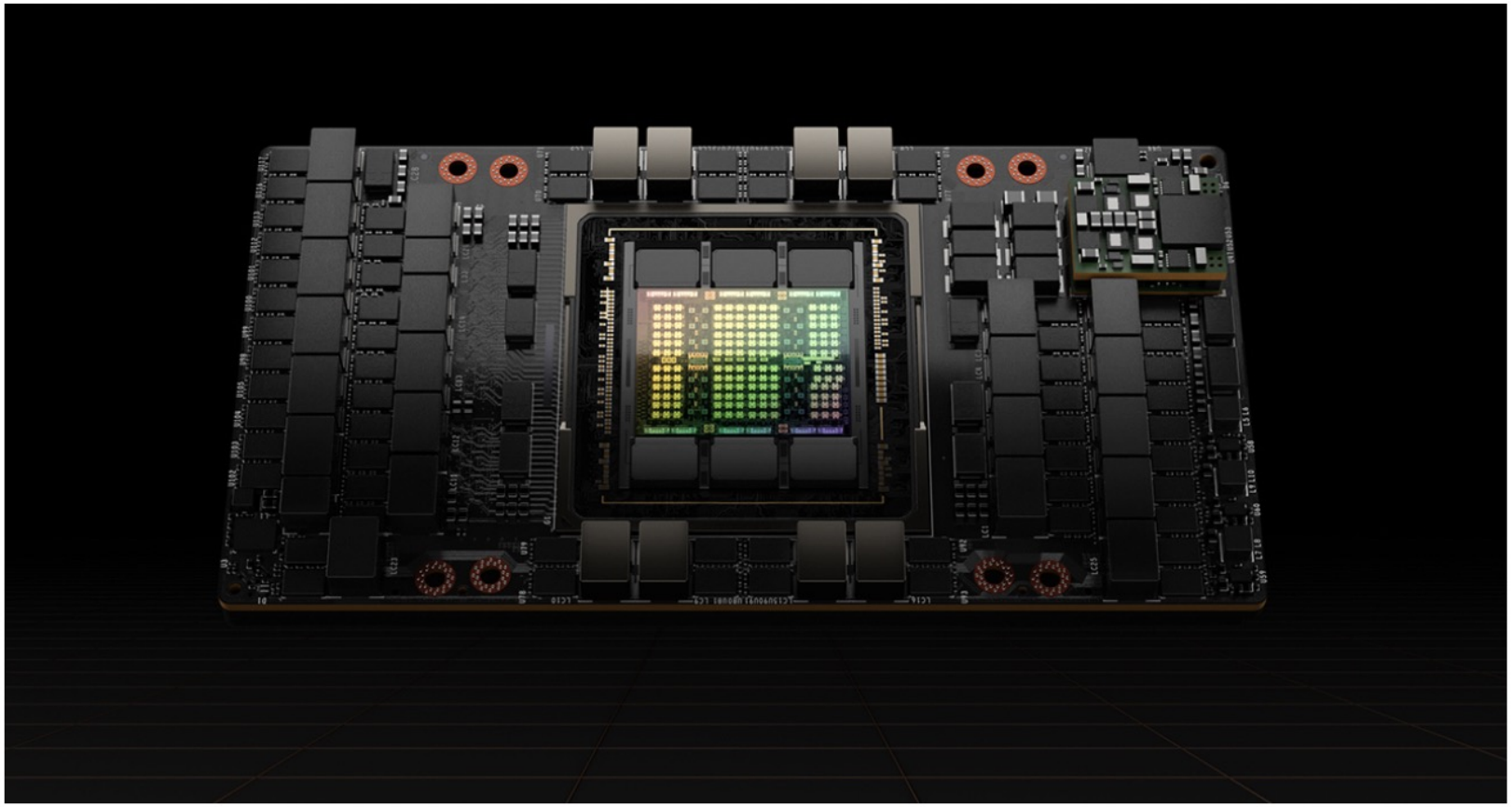


NVIDIA Hopper DPX Instructions (2022)

NVIDIA Hopper GPU Architecture Accelerates Dynamic Programming Up to 40x Using New DPX Instructions

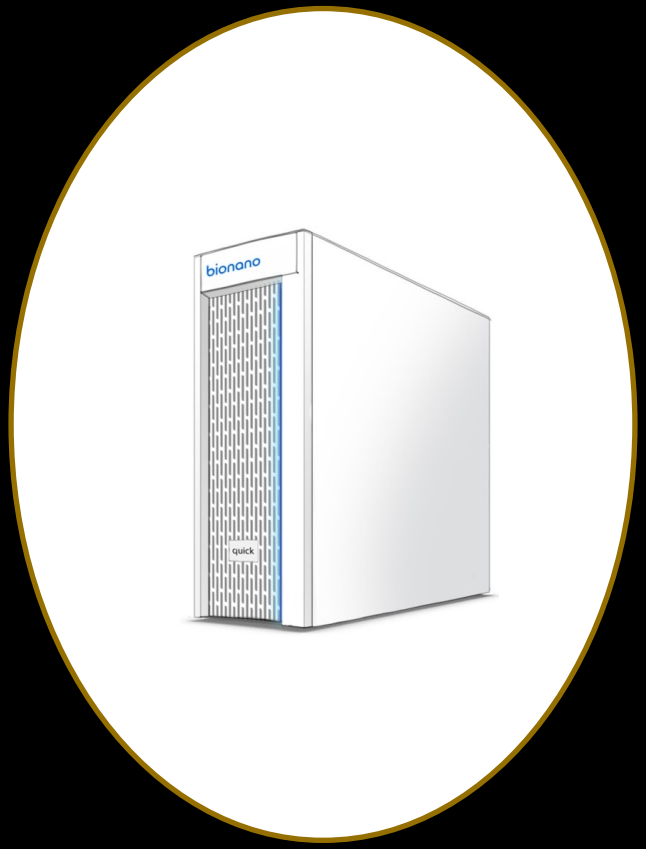
Dynamic programming algorithms are used in healthcare, robotics, quantum computing, data science and more.

March 22, 2022 by DION HARRIS



https://blogs.nvidia.com/blog/2022/03/22/nvidia-hopper-accelerates-dynamic-programming-using-dpx-instructions/

- **We are accelerating the transformation in how we analyze the human genome!**



Bionano & NVIDIA:

Accelerating Analysis for Fast Time to Results

Technological solution to **support higher throughput**

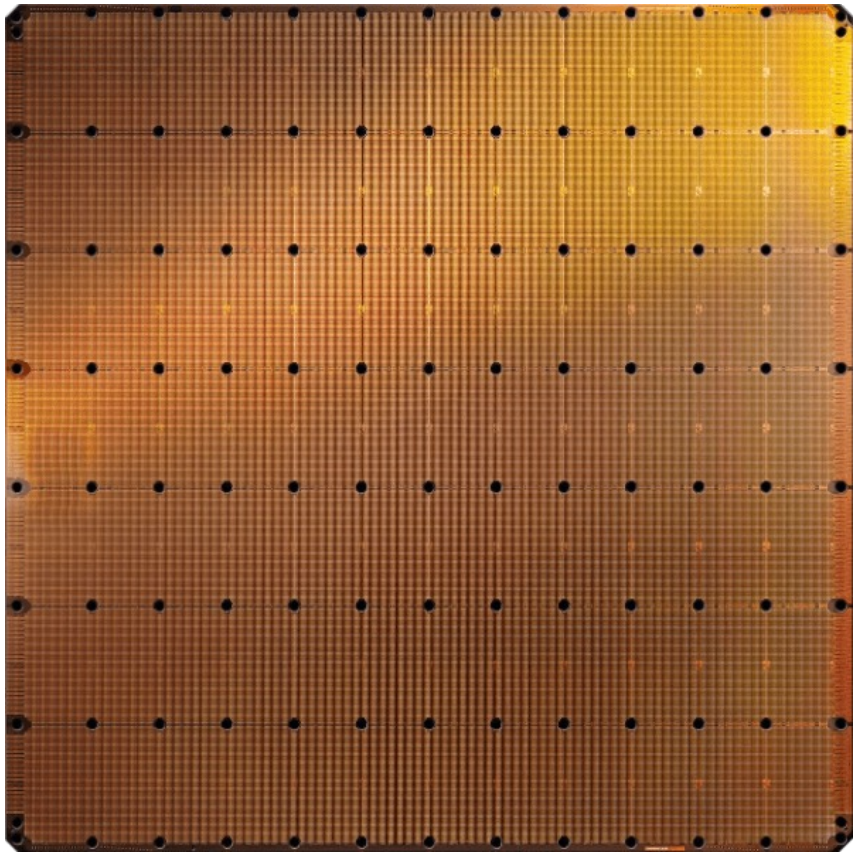
New high-performance algorithms from Bionano

Powered by NVIDIA RTX™ 6000 Ada Generation GPUs

Analysis of highly complex cancer whole genomes in **less than 2 hours**

Workflow tailored for a **small lab and IT footprint**

Cerebras's Wafer Scale Engine (2021)



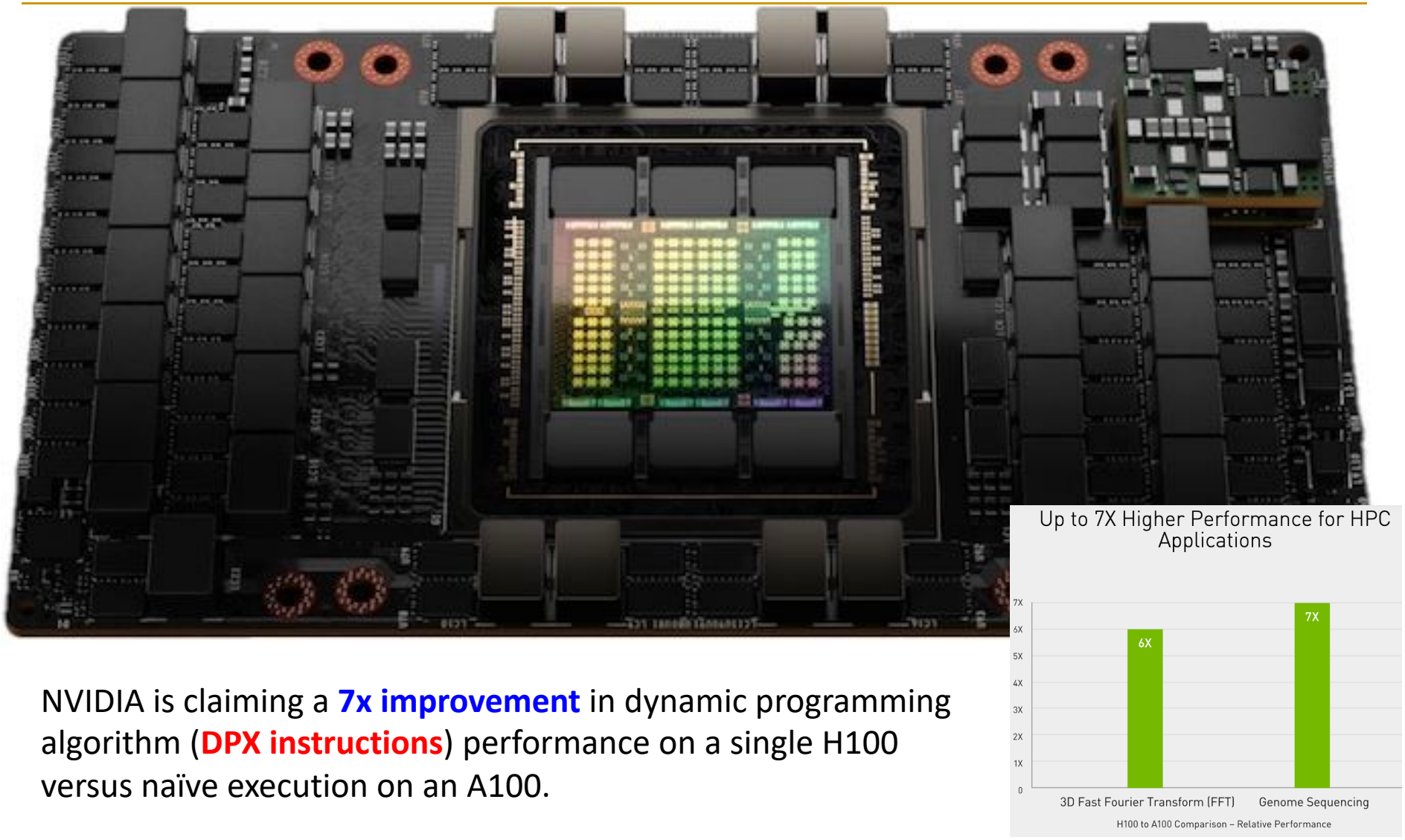
- The largest ML accelerator chip (2021)
- 850,000 cores

Cerebras WSE-2
2.6 Trillion transistors
46,225 mm^2

Largest GPU
54.2 Billion transistors
826 mm^2
NVIDIA Ampere GA100

https://www.anandtech.com/show/14758/hot-chips-31-live-blogs-cerebras-wafer-scale-deep-learning
https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

NVIDIA H100 (2022)

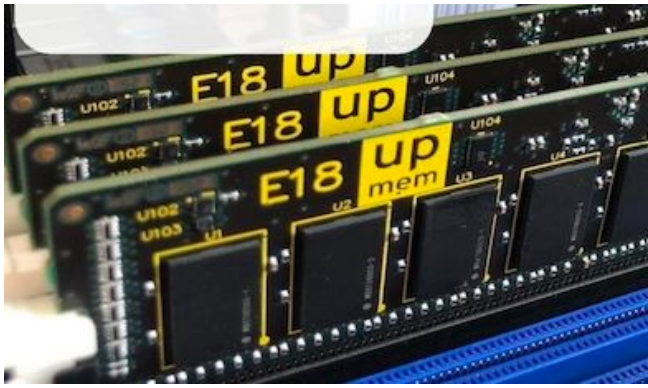


NVIDIA is claiming a **7x improvement** in dynamic programming algorithm (**DPX instructions**) performance on a single H100 versus naïve execution on an A100.

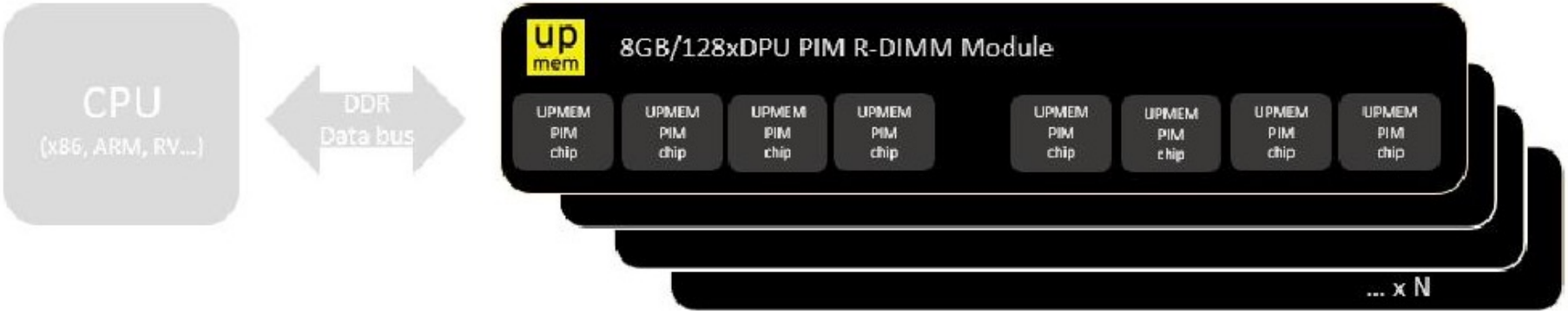
https://www.nvidia.com/en-us/data-center/h100/

UPMEM Processing-in-DRAM Engine (2019)

- Processing in DRAM Engine
- Includes **standard DIMM modules**, with a **large number of DPU processors** combined with DRAM chips.



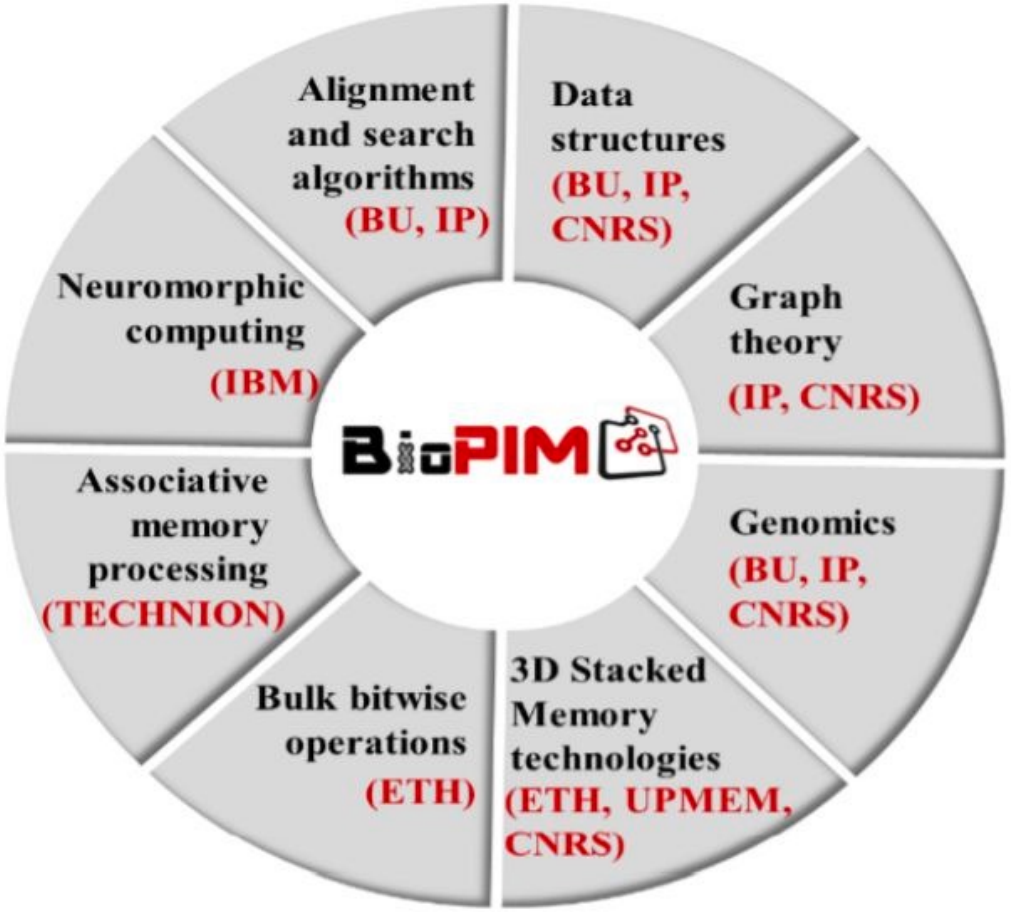
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



https://www.anandtech.com/show/14750/hot-chips-31-analysis-inmemory-processing-by-upmem

https://www.upmem.com/video-upmem-presenting-its-true-processing-in-memory-solution-hot-chips-2019/

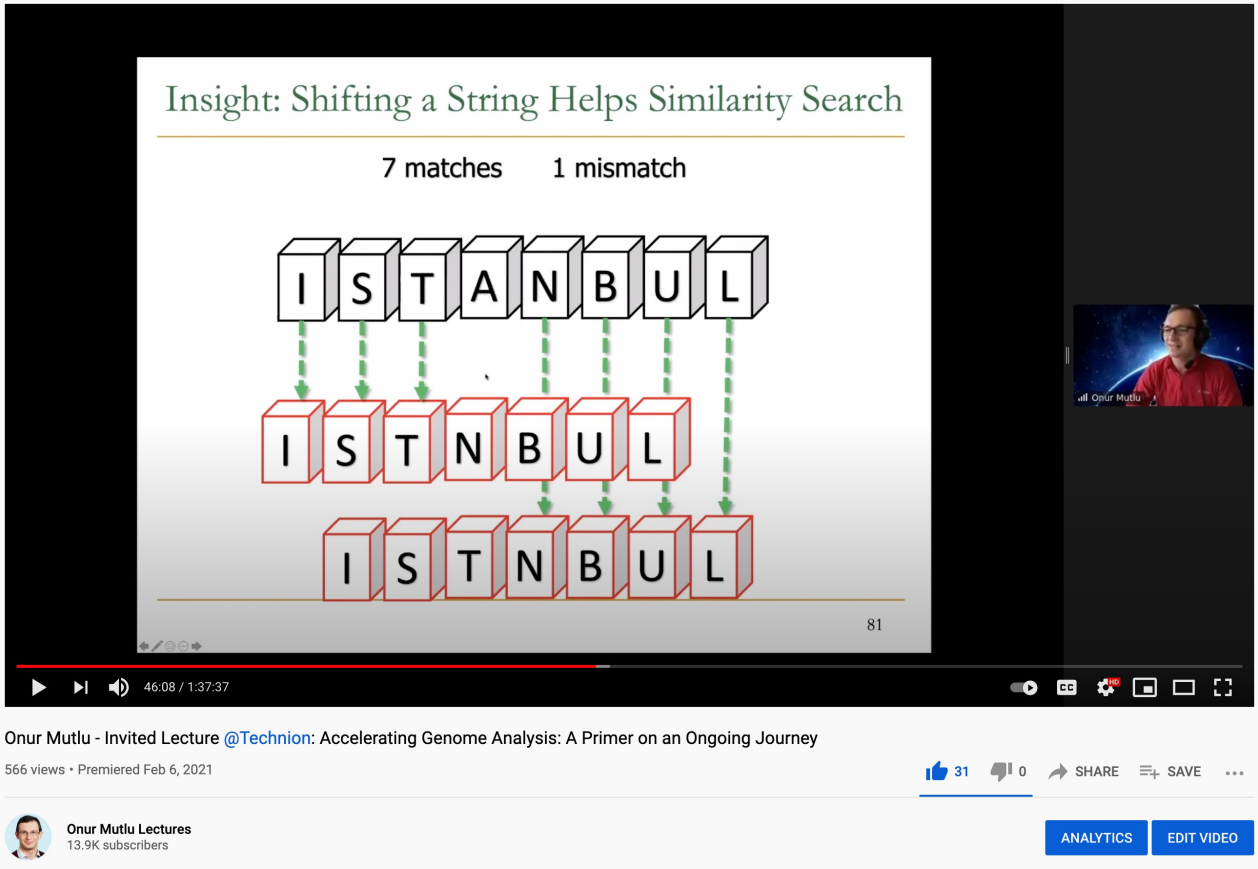
BioPIM (2022)



The vision of BioPIM is the realization of **cheap, ultra-fast and ultra-low energy mobile genomics** that eliminates the current dependence of sequence analysis on large and power-hungry computing clusters/data-centers.

Fast 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)]



More on Fast Genome Analysis…

- Onur Mutlu,
  "Accelerating Genome Analysis"
  *Invited Talk at the Barcelona Supercomputing Center (**BSC**)*, Barcelona, Spain, 6 September 2022.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 35 minutes, including Q&A)]
  [Related Invited Paper (at IEEE Micro, 2020)]
  [Related Invited Paper (at Computational and Structural Biology Journal, 2022)]

Accelerating Genome Analysis - Onur Mutlu's Invited Talk at the Barcelona Supercomputing Center

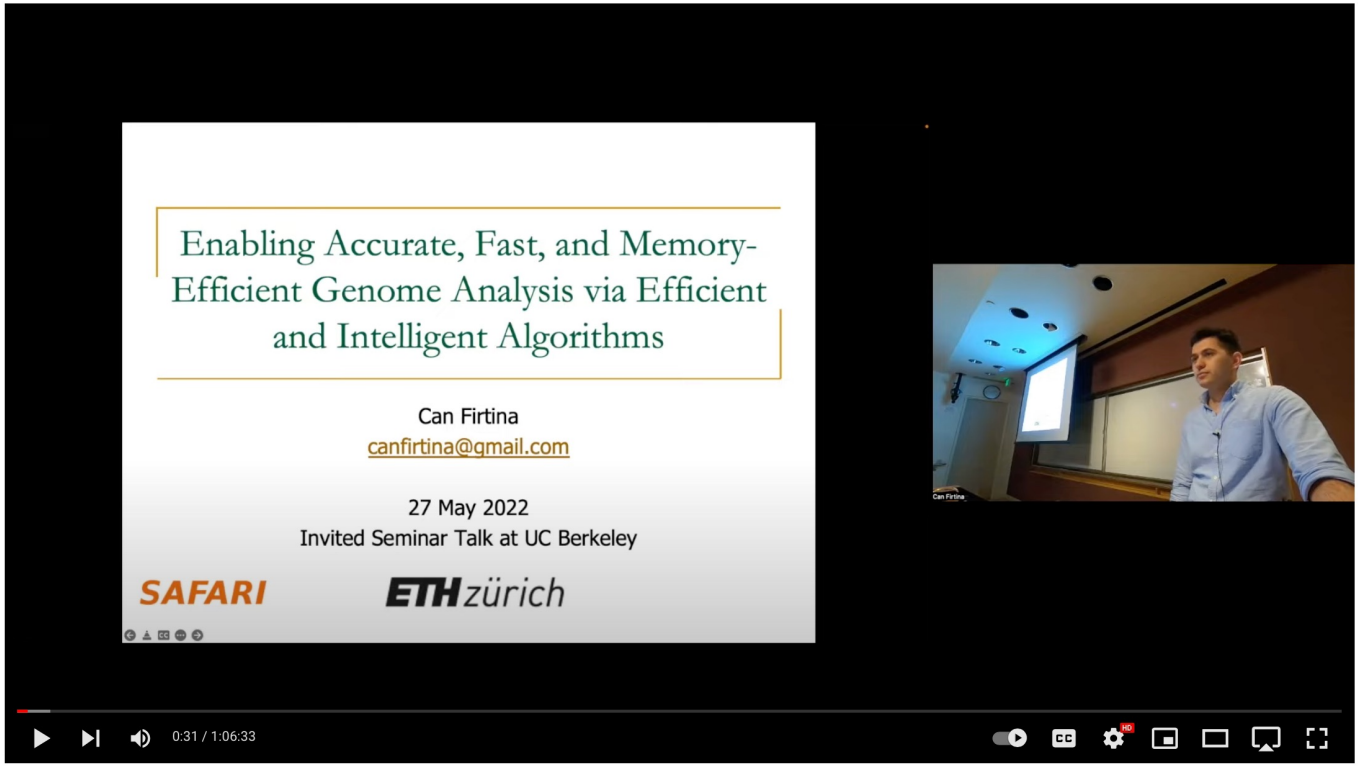
Onur Mutlu Lectures
36.6K subscribers
Analytics | Edit video | Share | Save

413 views Premiered Feb 2, 2023
Invited Talk at the Barcelona Supercomputing Center (BSC)
Accelerating Genome Analysis - A Primer on an Ongoing Journey

Presenter: Professor Onur Mutlu (https://people.inf.ethz.ch/omutlu/)
Date: September 6, 2022
1 hour 35 minutes (including Q&A)

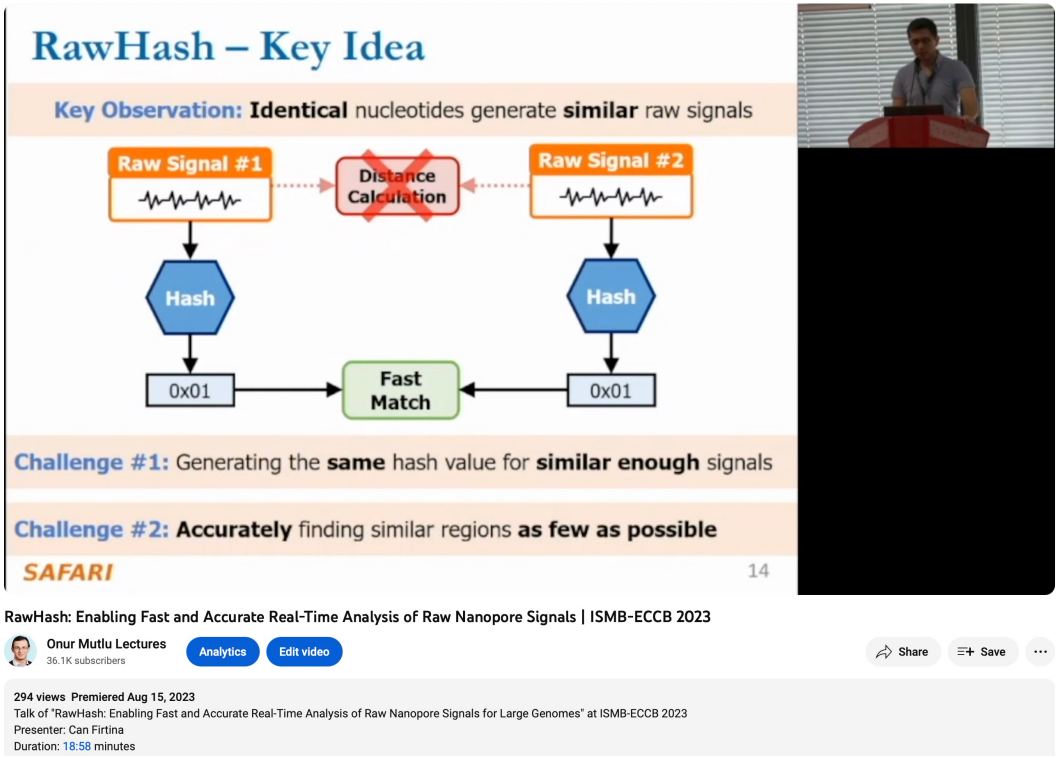
More on Accelerating Genome Analysis

- Can Firtina,
  "Enabling Accurate, Fast, and Memory-Efficient Genome Analysis via Efficient and Intelligent Algorithms"
  Talk at UC Berkeley, Berkeley, CA, United States, May 27, 2022.
  [Slides (pptx) (pdf)]
  [Talk Video (1 hour 6 minutes)]



More on Real-Time Genome Analysis

- Can Firtina,
  "RawHash: Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes"
  Proceedings Talk at ISMB-ECCB, Lyon, France, 25 July 2023.
  [Slides (pptx) (pdf)]
  [Talk Video (18 minutes]



Accelerating Genome Analysis [DAC 2023]

- Onur Mutlu and Can Firtina,
  "Accelerating Genome Analysis via Algorithm-Architecture Co-Design"
  *Invited Special Session Paper in Proceedings of the 60th Design Automation Conference (**DAC**)*, San Francisco, CA, USA, July 2023.
  [Slides (pptx) (pdf)]
  [Talk Video (38 minutes, including Q&A)]
  [Related Invited Paper]
  [arXiv version]

Accelerating Genome Analysis via Algorithm-Architecture Co-Design

Onur Mutlu Can Firtina
ETH Zürich

BIO-Arch Workshop at RECOMB 2023

- April 14, 2023

BIO-Arch: Workshop on Hardware Acceleration of Bioinformatics Workloads

About

BIO-Arch is a new forum for presenting and discussing new ideas in accelerating bioinformatics workloads with the co-design of hardware & software and the use of new computer architectures. Our goal is to discuss new system designs tailored for bioinformatics. BIO-Arch aims to bring together researchers in the bioinformatics, computational biology, and computer architecture communities to strengthen the progress in accelerating bioinformatics analysis (e.g., genome analysis) with efficient system designs that include hardware acceleration and software systems tailored fo new hardware technologies.

Venue

BIO-Arch will be held in The Social Facilities of İstanbul Technical University on **April 14**. Detailed information about how to arrive at the venue location with various transportation options can be found on the RECOMB website.

Our panel discussion will be held in conjunction with the main RECOMB conference. The panel discussion will be held in Marriott Şişli on **April 17 at 17:00**. You can find

https://www.youtube.com/watch?v=2rCsb4-nLmg

Genomics Course (Fall 2023)

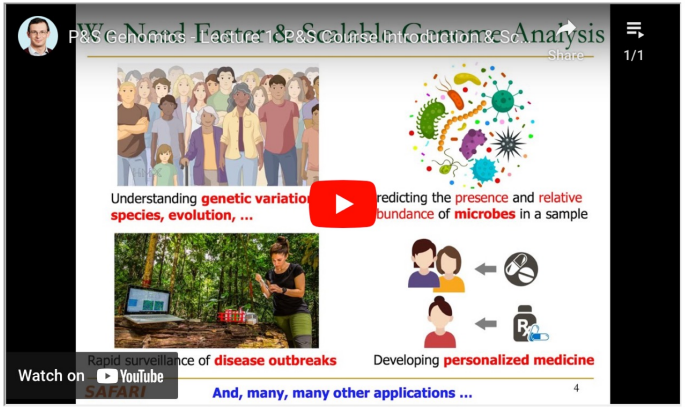
- **Fall 2023 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2023/doku.php?id=bioinformatics
- **Spring 2023 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2023/doku.php?id=bioinformatics

- **Youtube Livestream (Fall 2023):**
  - https://youtube.com/playlist?list=PL5Q2soXY2Zi_O0wyOjiMShG4t2QPZoeE3

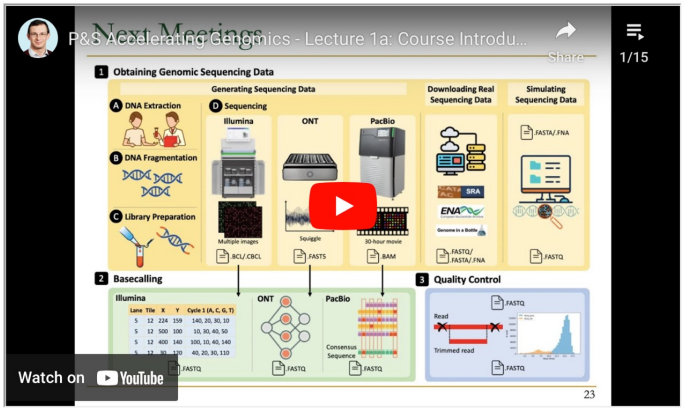
- Project course
  - Taken by Bachelor's/Master's students
  - Genomics lectures
  - Hands-on research exploration
  - Many research readings

https://www.youtube.com/onurmutlulectures

Lecture Playlist (Fall 2023):



Complete Lecture Playlist (Spring 2023):



Edit

Fall 2023 Schedule

Week	Date	Livestream	Meeting
W0	05.10 Thu.		L0: Project Introductions and Q&A
W1	11.10 Wed.	YouTube Live	L1: P&S Course Introduction & Scope (PDF) (PPT)
W2	25.10 Wed.		L2: Introduction to Genome Analysis (PDF) (PPT)
W3	01.11 Wed.		L3: From Molecules to Data: An Overview of DNA Sequencing Technologies (PDF) (PPT)
W4	08.11 Wed.		L4a: Fundamentals of Sequence Alignment: Algorithms and Applications (PDF) (PPT)
			L4b: Optimizing Sequence Search: Hashing, Indexing, and Filtering Techniques (PDF) (PPT)

Conclusion

- System design for bioinformatics is a critical problem
  - It has large scientific, medical, societal, personal implications
- We covered various recent ideas to
  - Accelerate genome analysis
  - Analyze genomes in ways that were not possible before
- **Many future opportunities exist**
  - **Especially with new sequencing technologies**
  - **Especially with new applications and use cases**

Enabling Fast, Accurate & Efficient Real-Time Genome Analysis via New Algorithms and Architectures

Can Firtina
canfirtina@gmail.com
https://cfirtina.com

16 January 2024

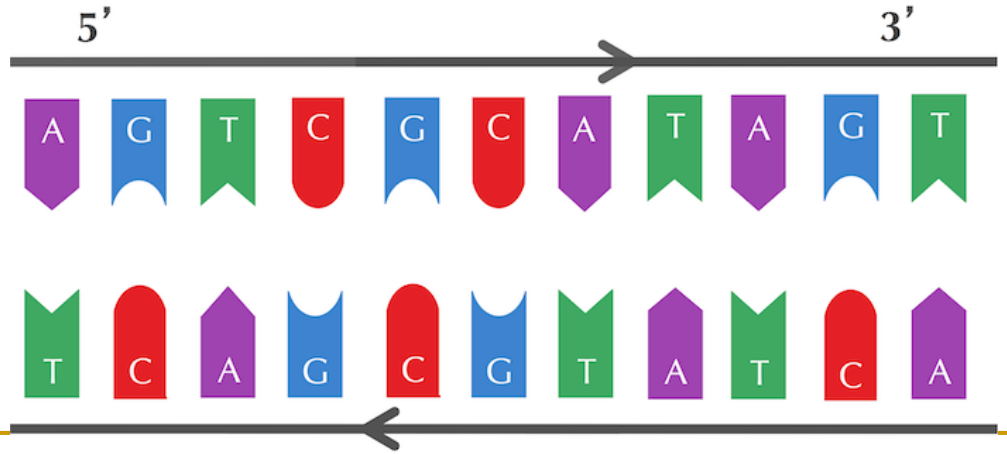
Technical University of Munich (TUM)

SAFARI

ETH zürich

Challenges in Read Mapping

- Need to find many mappings of each read
- Need to tolerate variances/sequencing errors in each read
- Need to map each read very fast (i.e., performance is important, life critical in some cases)
- Need to map reads to both forward and reverse strands



https://www.bioinformaticsalgorithms.org/bioinformatics-chapter-1

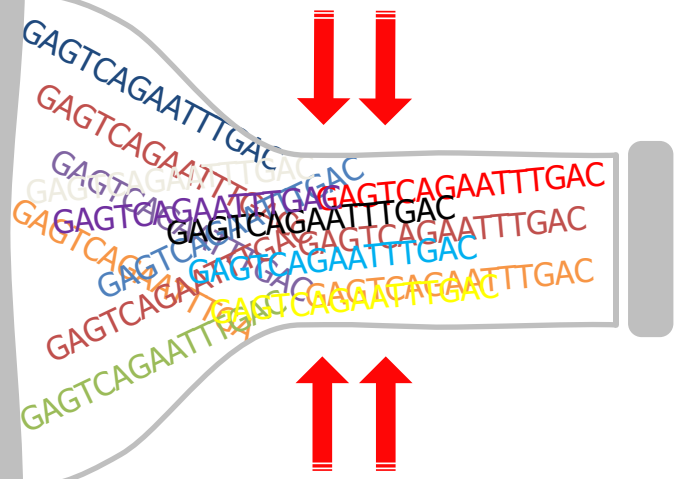
Analysis is Bottlenecked in Read Mapping!!

48 Human whole genomes

at 30× coverage

in about 2 days

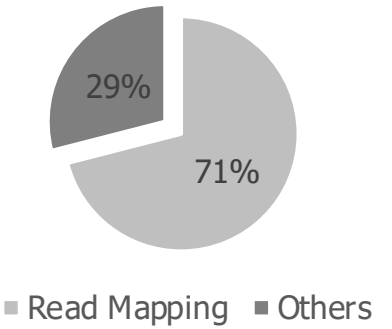
Illumina NovaSeq 6000



1 Human genome

32 CPU hours

on a 48-core processor



Goyal+, "Ultra-fast next generation human genome sequencing data processing using DRAGENTM bio-IT processor for precision medicine", *Open Journal of Genetics,* 2017.

A Tsunami of Sequencing Data

A Tera-scale increase in sequencing production in the past 25 years		
Genes & Operons	1990	**Kilo** = 1,000
Bacterial genomes	1995	**Mega** = 1,000,000
Human genome	2000	**Giga** = 1,000,000,000
Human microbiome	2005	**Tera** = 1,000,000,000,000
50K Microbiomes	2015	**Peta** = 1,000,000,000,000,000
what is expected for the next 15 years ? (a Giga?)		
200K Microbiomes	2020	**Exa** = 1,000,000,000,000,000,000
1M Microbiomes	2025	**Zetta** = 1,000,000,000,000,000,000,000
Earth Microbiome	2030	**Yotta** = 1,000,000,000,000,000,000,000,000

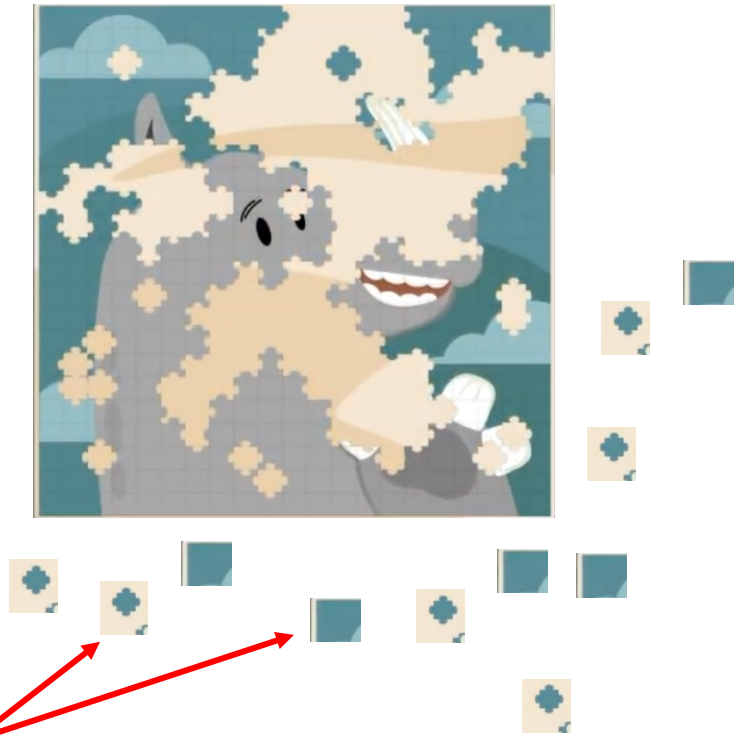
Source: @kyrpides

Solving the Puzzle

.FASTA file

Reference genome

.FASTQ file



Reads

https://www.pacb.com/smrt-science/smrt-sequencing/hifi-reads-for-highly-accurate-long-read-sequencing/

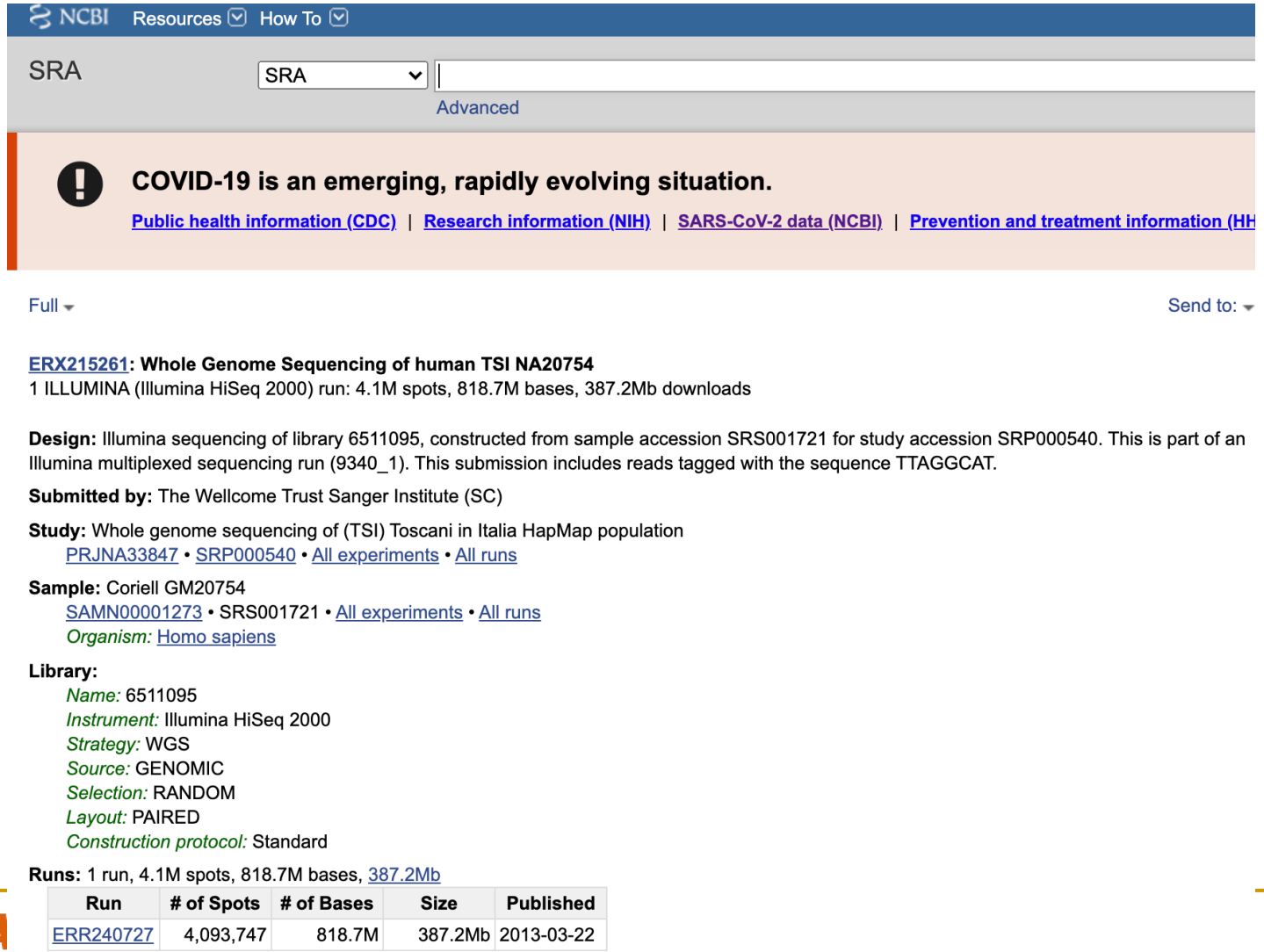
Obtaining the Human Reference Genome

- **GRCh38.p13**
- Description: Genome Reference Consortium Human Build 38 patch release 13 (GRCh38.p13)
- Organism name: Homo sapiens (human)
- Date: 2019/02/28
- 3,099,706,404 bases
- Compressed .fna file (964.9 MB)
- https://www.ncbi.nlm.nih.gov/assembly/GCF_000001405.39

```
>NC_000001.11 Homo sapiens chromosome 1, GRCh38.p13 Primary Assembly
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
NNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNNN
....
```

Obtaining .FASTQ Files

- https://www.ncbi.nlm.nih.gov/sra/ERR240727



NCBI Resources How To

SRA

SRA

Advanced

COVID-19 is an emerging, rapidly evolving situation.

Public health information (CDC) | Research information (NIH) | SARS-CoV-2 data (NCBI) | Prevention and treatment information (HH

Full

Send to:

ERX215261: **Whole Genome Sequencing of human TSI NA20754**
1 ILLUMINA (Illumina HiSeq 2000) run: 4.1M spots, 818.7M bases, 387.2Mb downloads

Design: Illumina sequencing of library 6511095, constructed from sample accession SRS001721 for study accession SRP000540. This is part of an Illumina multiplexed sequencing run (9340_1). This submission includes reads tagged with the sequence TTAGGCAT.

Submitted by: The Wellcome Trust Sanger Institute (SC)

Study: Whole genome sequencing of (TSI) Toscani in Italia HapMap population
PRJNA33847 • SRP000540 • All experiments • All runs

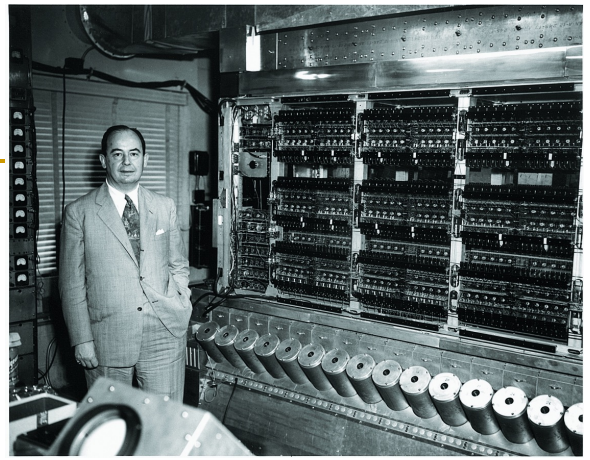
Sample: Coriell GM20754
SAMN00001273 • SRS001721 • All experiments • All runs
Organism: Homo sapiens

Library:
Name: 6511095
Instrument: Illumina HiSeq 2000
Strategy: WGS
Source: GENOMIC
Selection: RANDOM
Layout: PAIRED
Construction protocol: Standard

Runs: 1 run, 4.1M spots, 818.7M bases, 387.2Mb

Run	# of Spots	# of Bases	Size	Published
ERR240727	4,093,747	818.7M	387.2Mb	2013-03-22

Today's Computing Systems

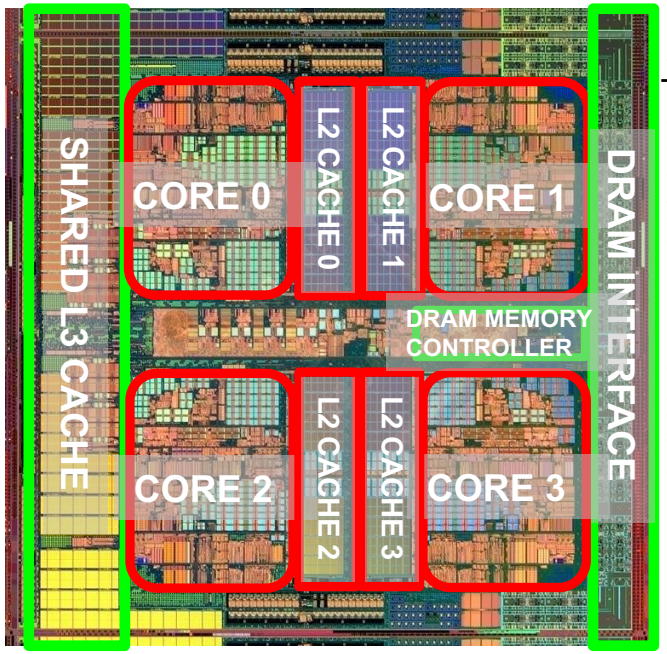


von Neumann model, 1945

where the **CPU** can **access data** stored in an off-chip main memory only through **power-hungry bus**

Storage (SSD/HDD)

Main Memory



*Die photo credit: AMD Barcelona

Microprocessor

Burks, Goldstein, von Neumann, "Preliminary discussion of the logical design of an electronic computing instrument," 1946.

The Problem

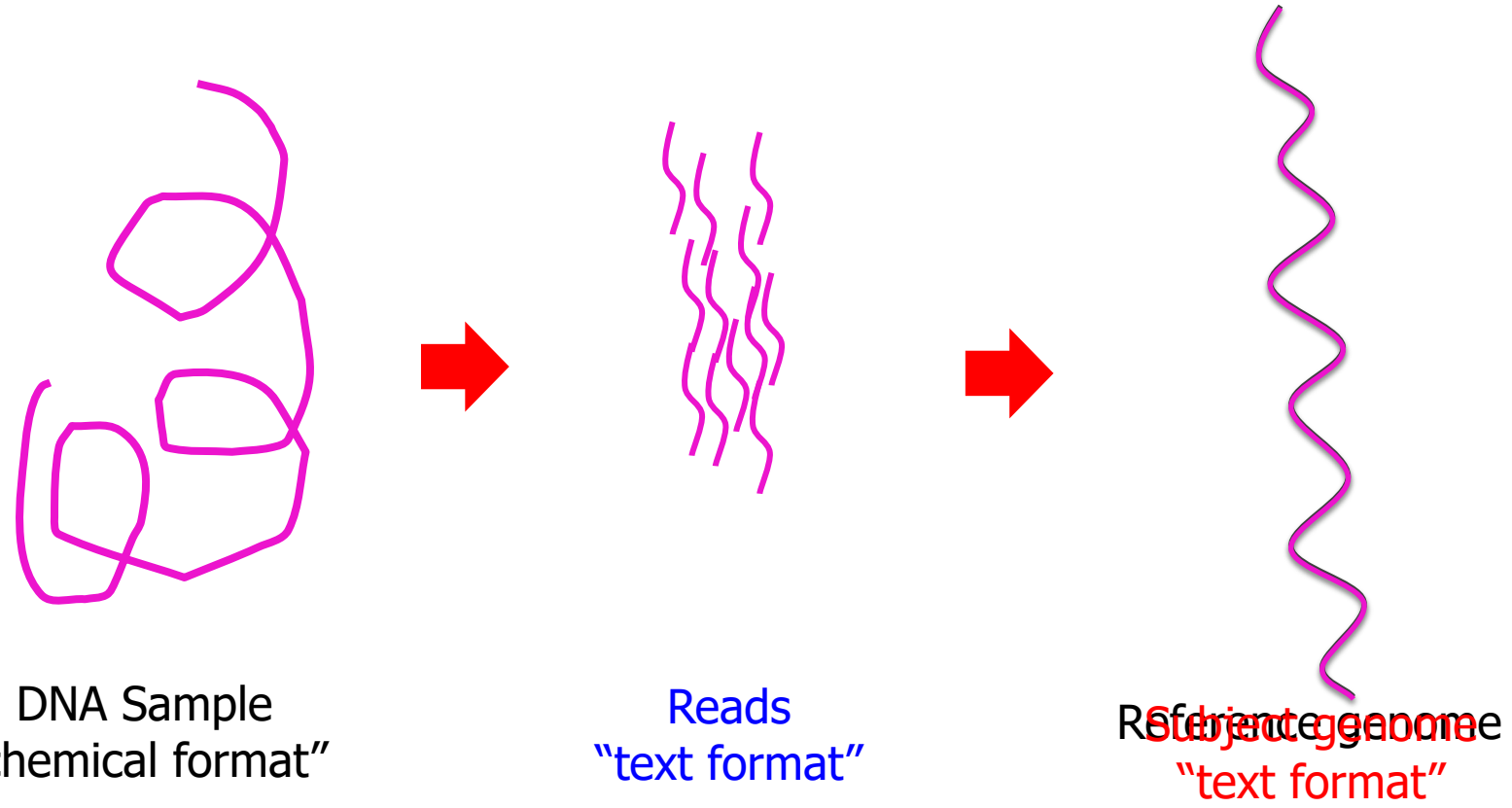
Data analysis

is performed

far away from the data

Read Mapping

Map reads to a known reference genome with some minor differences allowed



Read Mapping Algorithms: Two Styles

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

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

An Example of Hash Table Based Mappers

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

nature genetics

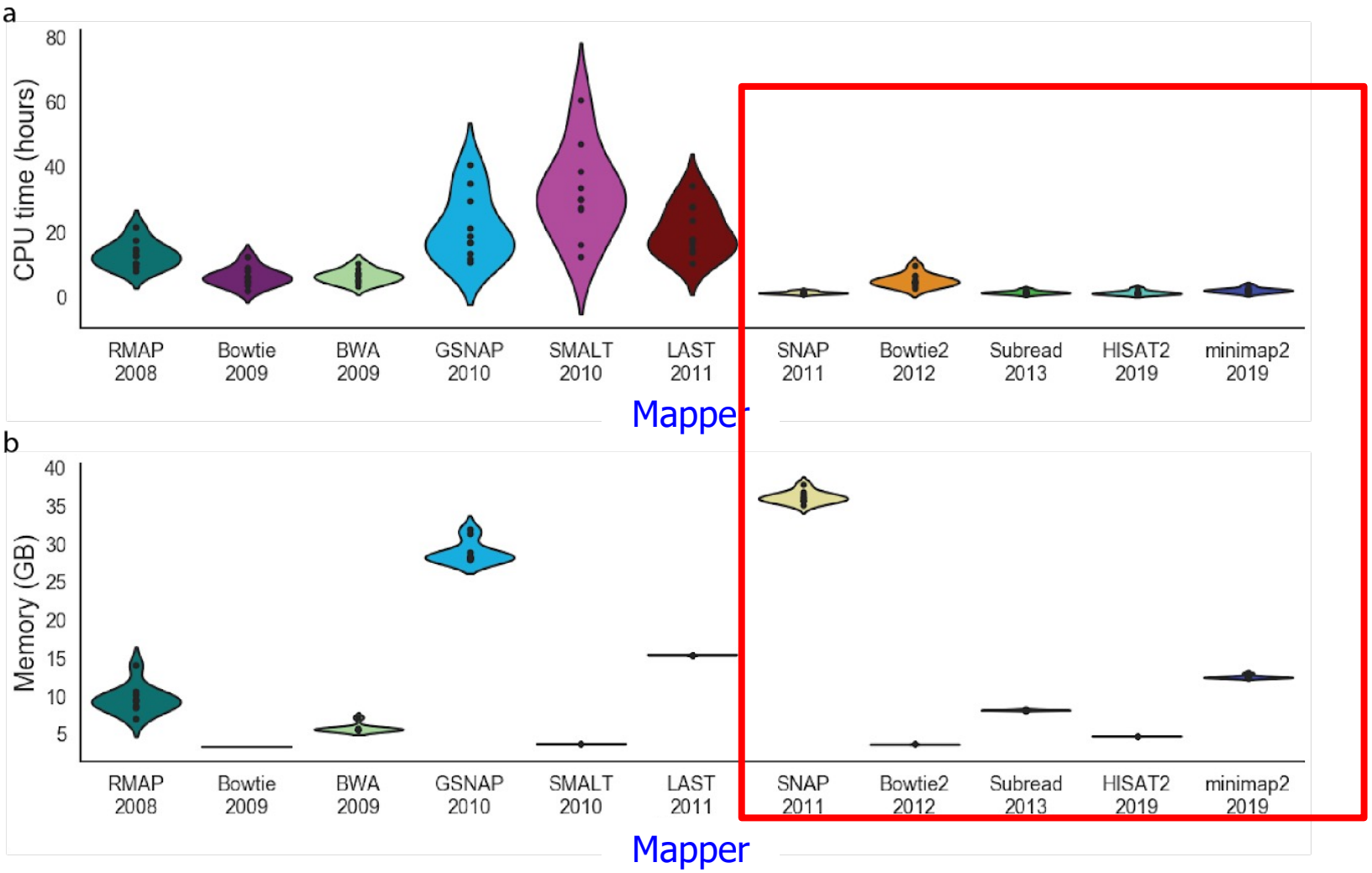
https://github.com/BilkentCompGen/mrfast

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

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

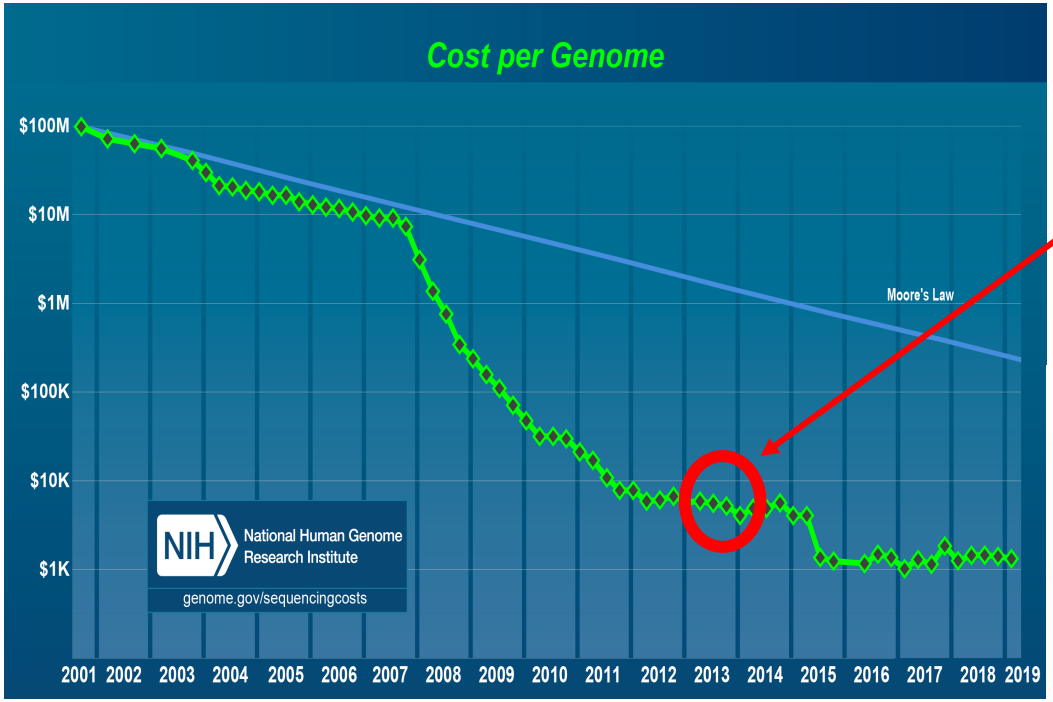
Alkan+, "Personalized copy number and segmental duplication maps using next-generation sequencing", Nature Genetics 2009.

Performance of Read Mapping

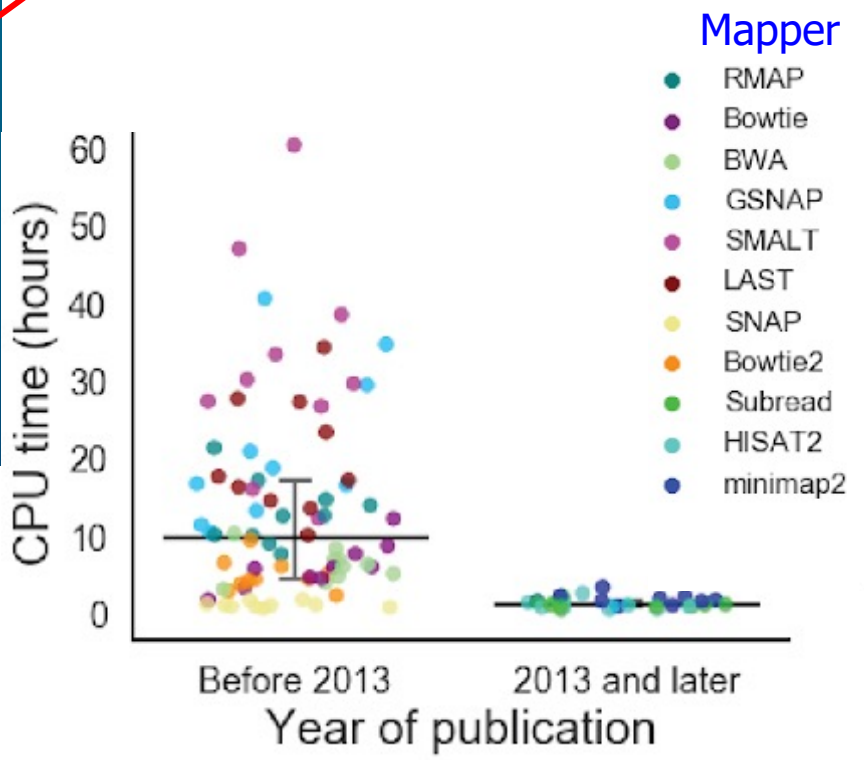


Alser+, "Technology dictates algorithms: Recent developments in read alignment", Genome Biology, 2021

The Need for Speed



Did we realize the **need** for **faster** genome analysis?



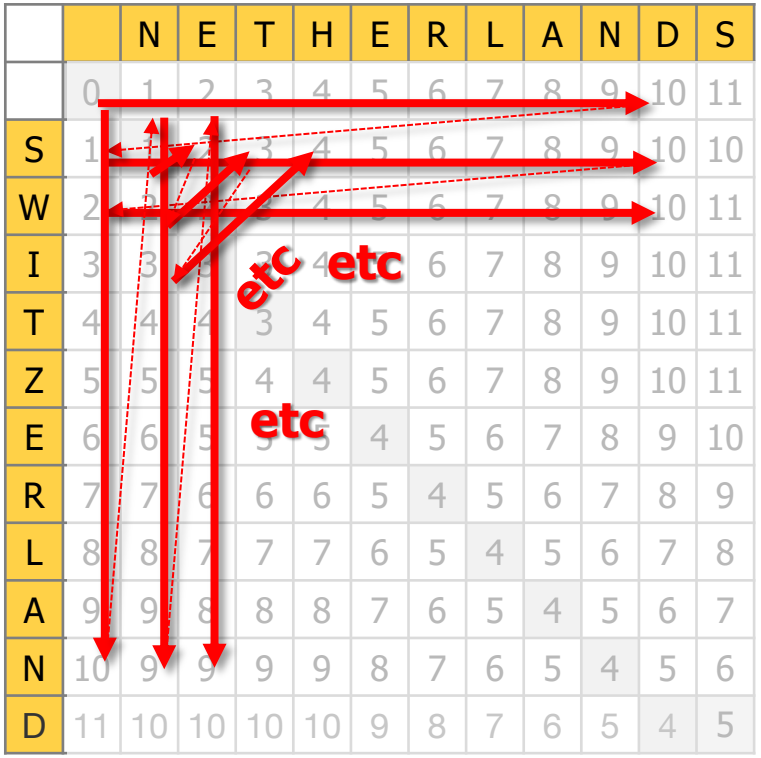
Alser+, "Technology dictates algorithms: Recent developments in read alignment", Genome Biology, 2021

Sequence Alignment in Unavoidable

- Quadratic-time dynamic-programming algorithm **WHY?!**

  Enumerating all possible prefixes

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



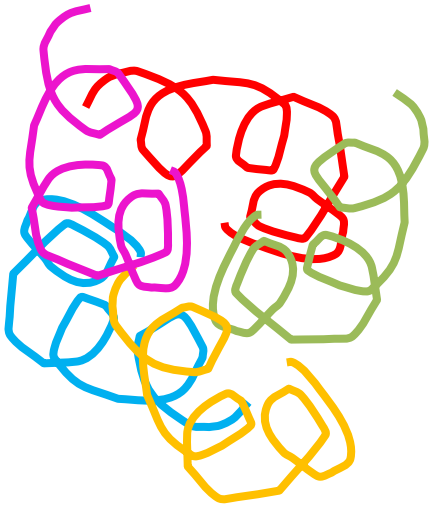
Sequence Alignment in Unavoidable

- **Quadratic-time** dynamic-programming algorithm
  - Enumerating all possible prefixes
- **Data dependencies** limit the computation parallelism
  - Processing row (or column) after another
- **Entire matrix** is computed even though strings can be dissimilar.
  - Number of differences is computed only at the backtraking step.

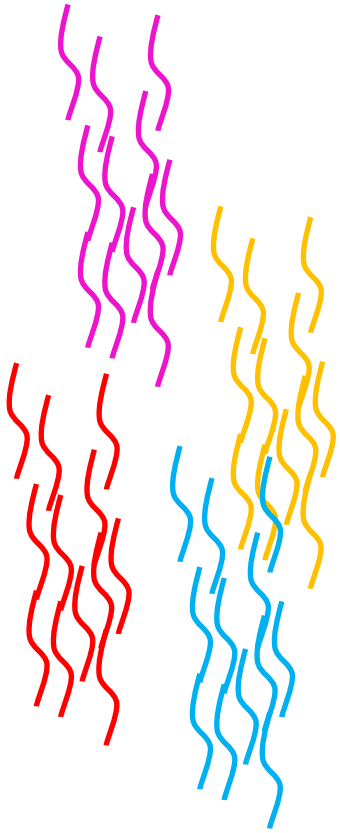
		N	E	T	H	E	R	L	A	N	D	S
	0	1	2	3	4	5	6	7	8	9	10	11
S	1	1	2	3	4	5	6	7	8	9	10	10
W	2	2	2	3	4	5	6	7	8	9	10	11
I	3	3	3	3	4	5	6	7	8	9	10	11
T	4	4	4	3	4	5	6	7	8	9	10	11
Z	5	5	5	4	4	5	6	7	8	9	10	11
E	6	6	5	5	5	4	5	6	7	8	9	10
R	7	7	6	6	6	5	4	5	6	7	8	9
L	8	8	7	7	7	6	5	4	5	6	7	8
A	9	9	8	8	8	7	6	5	4	5	6	7
N	10	9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

Metagenomics Analysis

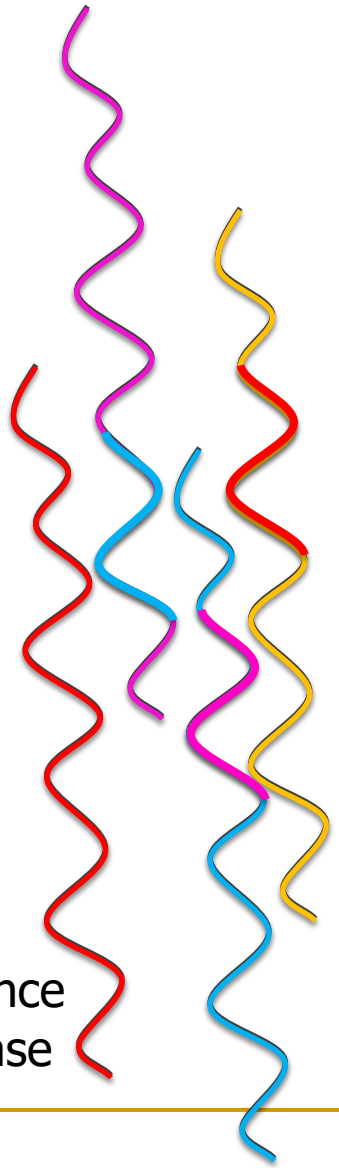
Reads from different unknown donors at sequencing time are mapped to many known reference genomes



genetic material recovered directly from environmental samples

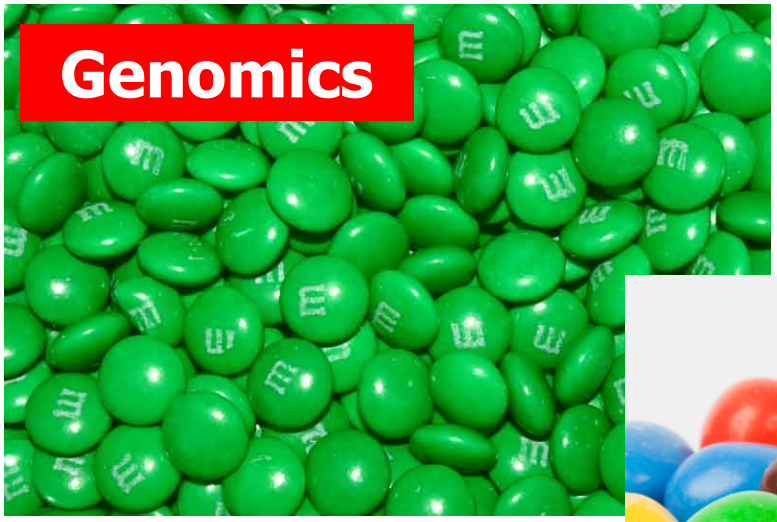


Reads "text format"

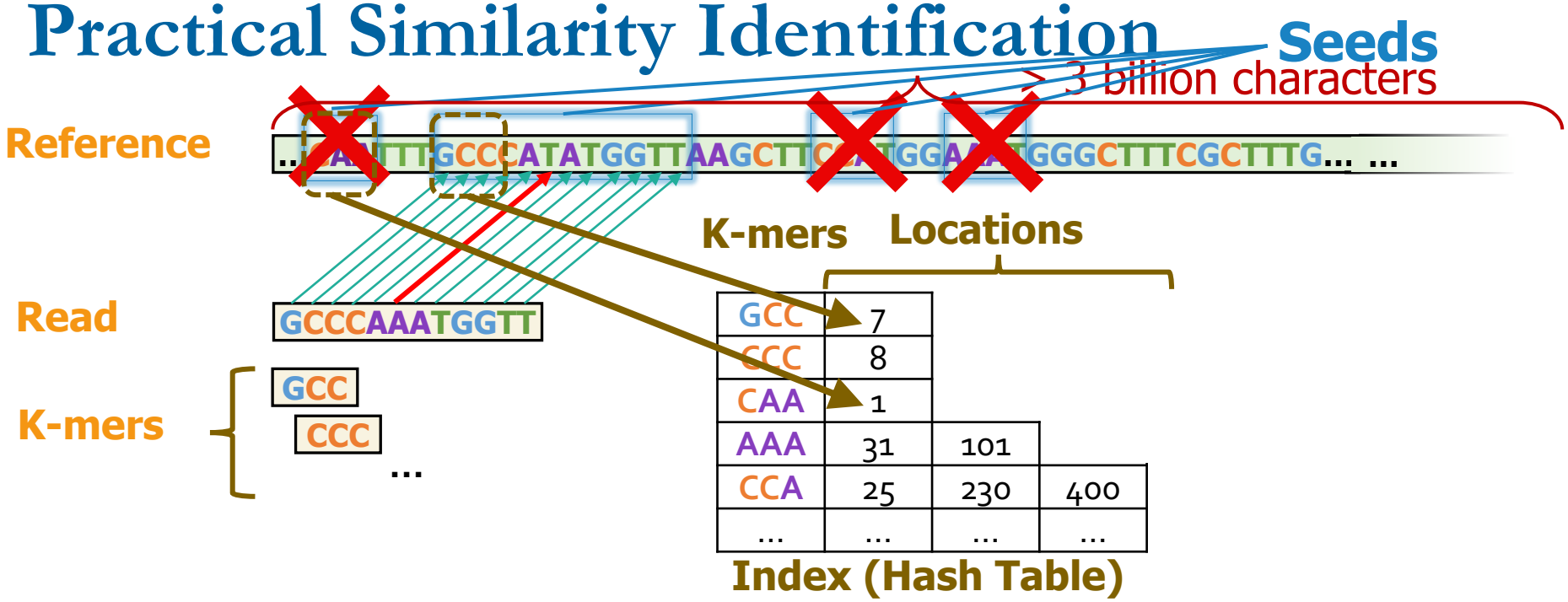


Reference Database

Genomics vs. Metagenomics



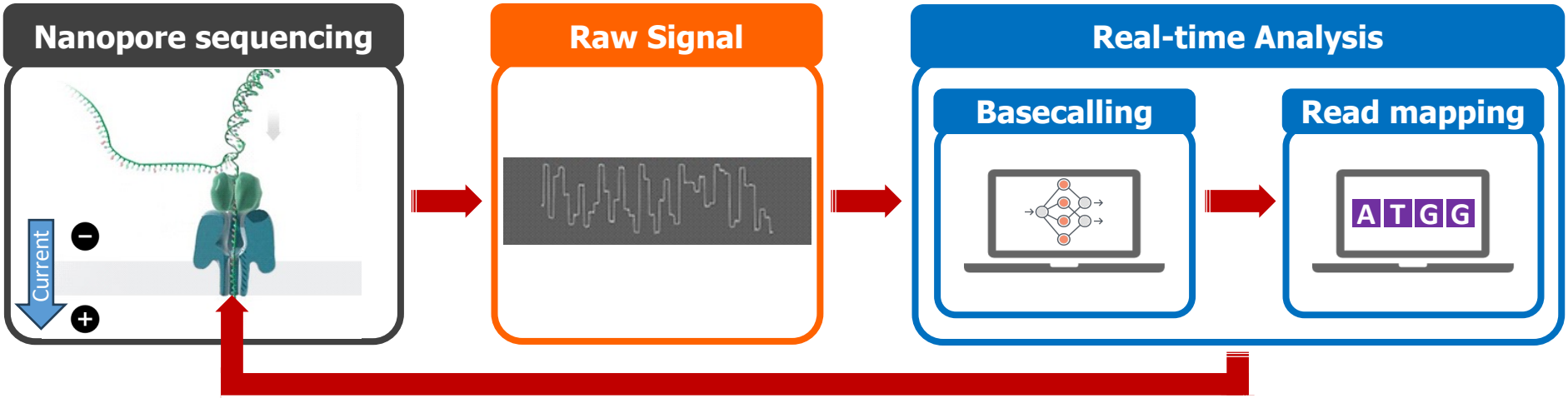
Practical Similarity Identification



Seeding	Determine potential matching regions (seeds) in the reference genome
Seed Filtering (e.g., Chaining)	Prune some seeds in the reference genome
Alignment	Determine the exact differences between the read and the reference genome

Existing Solutions – Real-time Basecalling

Deep neural networks (**DNNs**) for translating **signals** to **bases**

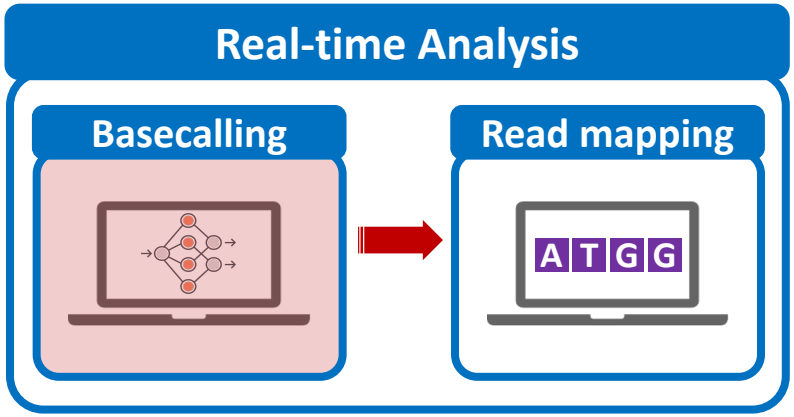


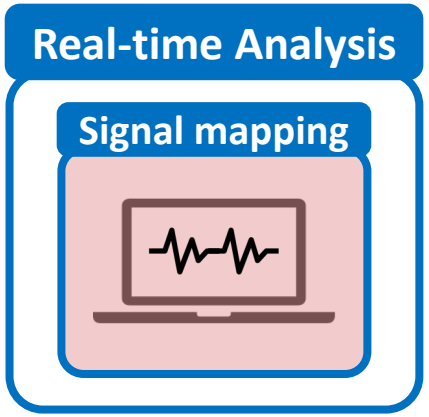
DNNs provide **less noisy analysis** from basecalled sequences

Costly and power-hungry computational requirements

The Problem

The existing solutions are **ineffective for large genomes**





Costly and energy-hungry computations to basecall each read:
Portable sequencing becomes challenging with resource-constrained devices

Larger number of reference regions **cannot be handled accurately or quickly**, rendering existing solutions **ineffective for large genomes**

Applications of Read Until

Depletion: Reads mapping to a particular reference genome is ejected

- Removing contaminated reads from a sample
- Relative abundance estimation
- Controlling low/high-abundance genomes in a sample
- Controlling the sequencing of depth of a genome

Enrichment: Reads **not** mapping to a particular reference genome is ejected

- Purifying the sample to ensure it contains only the selected genomes
- Removing the host genome (e.g., human) in contamination analysis

Applications of Run Until and Sequence Until

Run Until: Stopping the sequencing without informative decision from analysis

- Stopping when reads reach to a particular depth of coverage
- Stopping when the abundance of all genomes reach a particular threshold

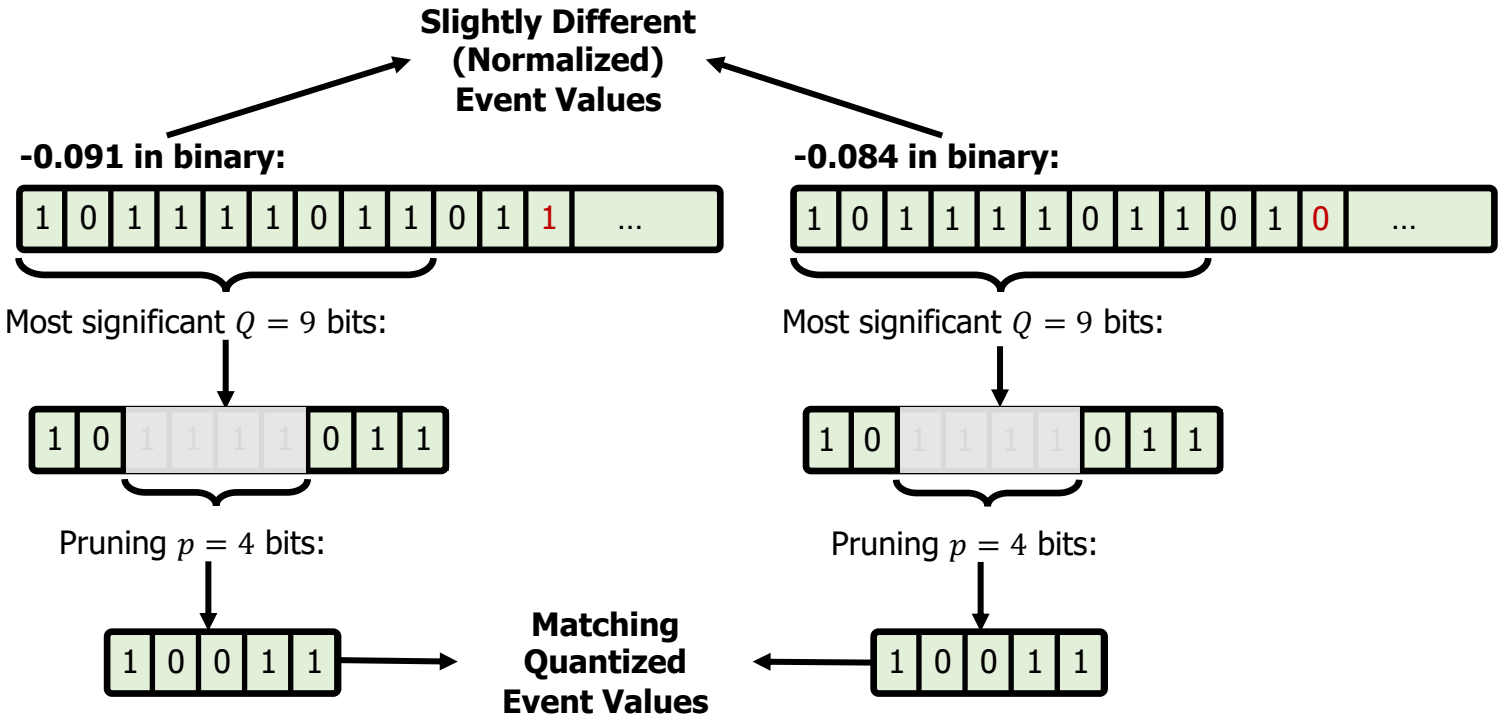
Sequence Until: Stopping the sequencing based on information decision

- Stopping when relative abundance estimations do not change substantially (for high-abundance genomes)
- Stopping when finding that the sample is contaminated with a particular set of genomes
- …

Details: Quantizing the Event Values

- **Observation:** Identical k-mers generate similar raw signals
  - **Challenge:** Their corresponding event values can be slightly different

- **Key Idea:** Quantize the event values
  - To enable assigning the **same quantized value** to the **similar event values**



Average Sequenced Bases and Chunks

Tool	*SARS-CoV-2*	*E. coli*	*Yeast*	*Green Algae*	*Human*
Average sequenced base length per read					
UNCALLED	**184.51**	**580.52**	**1,233.20**	5,300.15	6,060.23
RawHash	513.95	1,376.14	2,565.09	**4,760.59**	**4,773.58**
Average sequenced number of chunks per read					
Sigmap	**1.01**	**2.11**	**4.14**	**5.76**	**10.40**
RawHash	1.24	3.20	5.83	10.72	10.70

RawHash **reduces sequencing time and cost for large genomes**

up to **1.3**× compared to UNCALLED

Although Sigmap processes less number of chunks than RawHash, it fails to provide real-time analysis capabilities for large genomes

Breakdown Analysis of the RawHash Steps

	Fraction of entire runtime (%)				
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***
File I/O	0.00	0.00	0.00	0.00	0.00
Signal-to-Event	21.75	1.86	1.01	0.53	0.02
Sketching	0.74	0.06	0.04	0.03	0.00
Seeding	3.86	4.14	3.52	6.70	5.39
Chaining	73.50	93.92	95.42	92.43	94.46
Seeding + Chaining	77.36	98.06	98.94	99.14	99.86

The entire runtime is **bottlenecked by the chaining step**

Required Computation Resources in Indexing

Tool	*Contamination*	*SARS-CoV-2*	*E. coli*	*Yeast*	*Green Algae*	*Human*	*Relative Abundance*
			CPU Time (sec)				
UNCALLED	8.72	9.00	11.08	18.62	285.88	4,148.10	4,382.38
Sigmap	0.02	0.04	8.66	24.57	449.29	36,765.24	40,926.76
RawHash	0.18	0.13	2.62	4.48	34.18	1,184.42	788.88
			Real time (sec)				
UNCALLED	1.01	1.04	2.67	7.79	280.27	4,190.00	4,471.82
Sigmap	0.13	0.25	9.31	25.86	458.46	37,136.61	41,340.16
RawHash	0.14	0.10	1.70	2.06	15.82	278.69	154.68
			Peak memory (GB)				
UNCALLED	0.07	0.07	0.13	0.31	11.96	48.44	47.81
Sigmap	0.01	0.01	0.40	1.04	8.63	227.77	238.32
RawHash	0.01	0.01	0.35	0.76	5.33	83.09	152.80

The indexing step of RawHash is **orders of magnitude faster** than the indexing steps of UNCALLED and Sigmap, especially **for large genomes**

RawHash requires **larger memory space** than UNCALLED

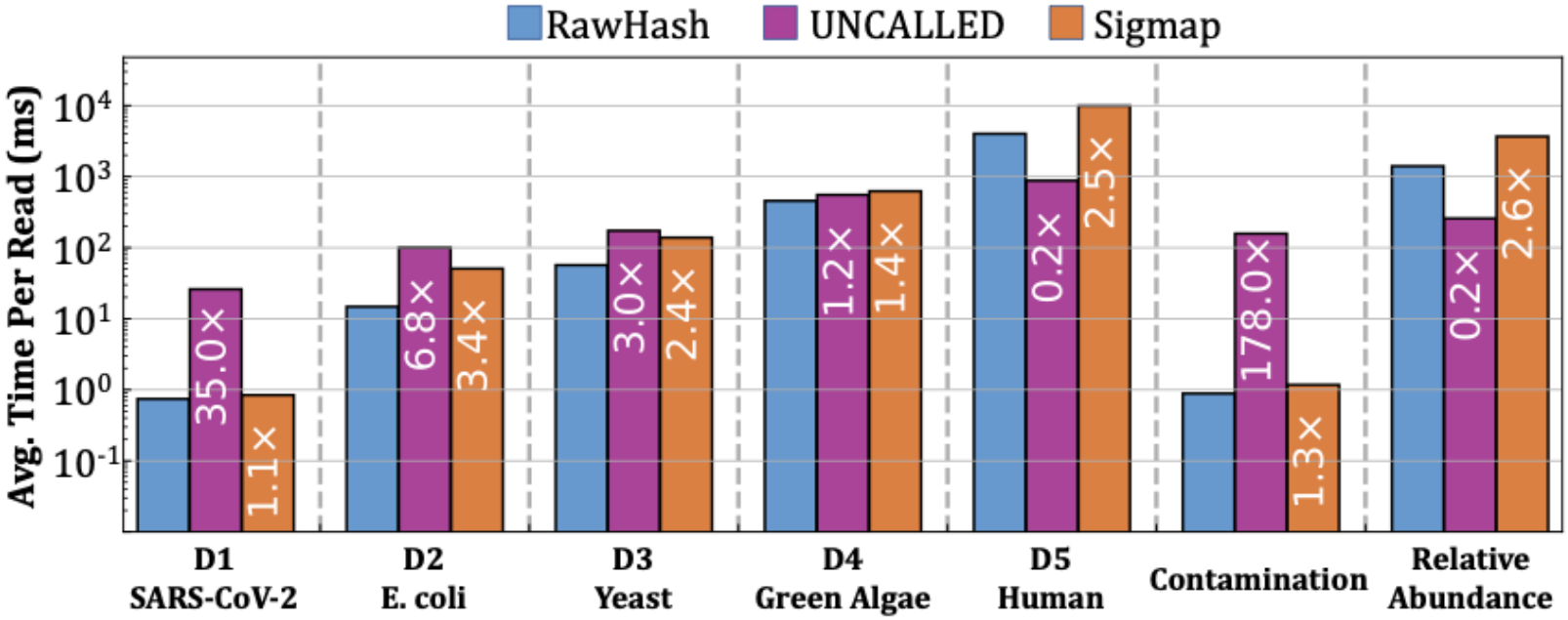
Required Computation Resources in Mapping

Tool	*Contamination*	*SARS-CoV-2*	*E. coli*	*Yeast*	*Green Algae*	*Human*	*Relative Abundance*
				CPU Time (sec)			
UNCALLED	265,902.26	36,667.26	35,821.14	8,933.52	16,769.09	262,597.83	586,561.54
Sigmap	4,573.18	1,997.84	23,894.70	11,168.96	31,544.55	4,837,058.90	11,027,652.91
RawHash	3,721.62	1,832.56	8,212.17	4,906.70	25,215.23	2,022,521.48	4,738,961.77
				Real time (sec)			
UNCALLED	20,628.57	2,794.76	1,544.68	285.42	2,138.91	8,794.30	19,409.71
Sigmap	6,725.26	3,222.32	2,067.02	1,167.08	2,398.83	158,904.69	361,443.88
RawHash	3,917.49	1,949.53	957.13	215.68	1,804.96	65,411.43	152,280.26
				Peak memory (GB)			
UNCALLED	0.65	0.19	0.52	0.37	0.81	9.46	9.10
Sigmap	111.69	28.26	111.11	14.65	29.18	311.89	489.89
RawHash	4.13	4.20	4.16	4.37	11.75	52.21	55.31

The mapping step of RawHash is **significantly faster than Sigmap** for all genomes, and **faster than UNCALLED for small genomes**

RawHash requires **larger memory space** than UNCALLED

Average Mapping Time per Read



The mapping step of RawHash is **significantly faster than Sigmap** for all genomes, and **faster than UNCALLED for small genomes**

Parameter Configurations

Tool	*Contamination*	*SARS-CoV-2*	*E. coli*	*Yeast*	*Green Algae*	*Human*	*Relative Abundance*
RawHash	-x viral -t 32	-x viral -t 32	-x sensitive -t 32	-x sensitive -t 32	-x fast -t 32	-x fast -t 32	-x fast -t 32
UNCALLED	map -t 32						
Sigmap	-m -t 32						
Minimap2	-x map-ont -t 32						

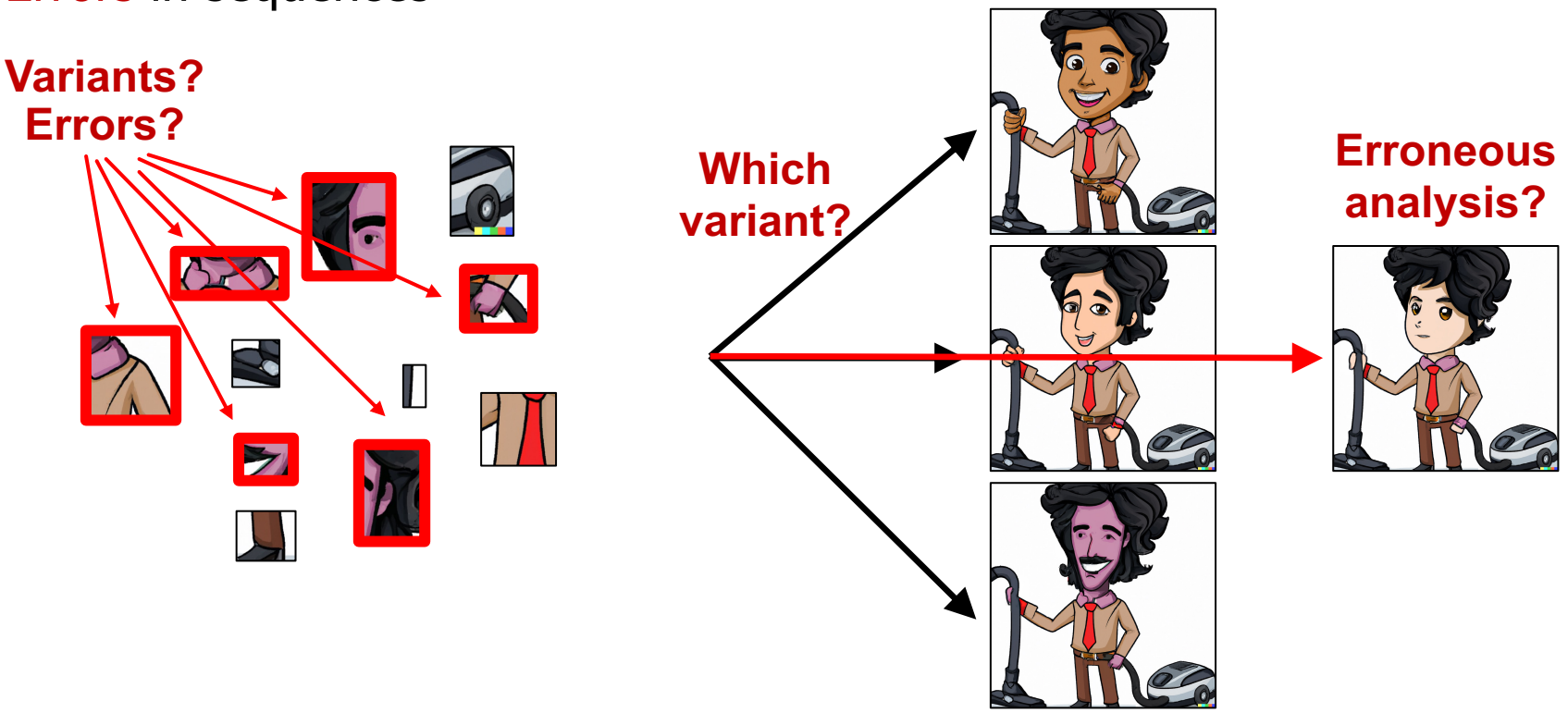
Preset (-x)	**Corresponding parameters**	Usage
viral	-e 5 -q 9 -l 3	Viral genomes
sensitive	-e 6 -q 9 -l 3	Small genomes (i.e., $< 50M$ bases)
fast	-e 7 -q 9 -l 3	Large genomes (i.e., $> 50M$ bases)

Versions

Tool	Version	Link to the Source Code
RawHash	0.9	https://github.com/CMU-SAFARI/RawHash/tree/8042b1728e352a28fcc79c2efd80c8b631fe7bac
UNCALLED	2.2	https://github.com/skovaka/UNCALLED/tree/74a5d4e5b5d02fb31d6e88926e8a0896dc3475cb
Sigmap	0.1	https://github.com/haowenz/sigmap/tree/c9a40483264c9514587a36555b5af48d3f054f6f
Minimap2	2.24	https://github.com/lh3/minimap2/releases/tag/v2.24

Why Graphs are Useful

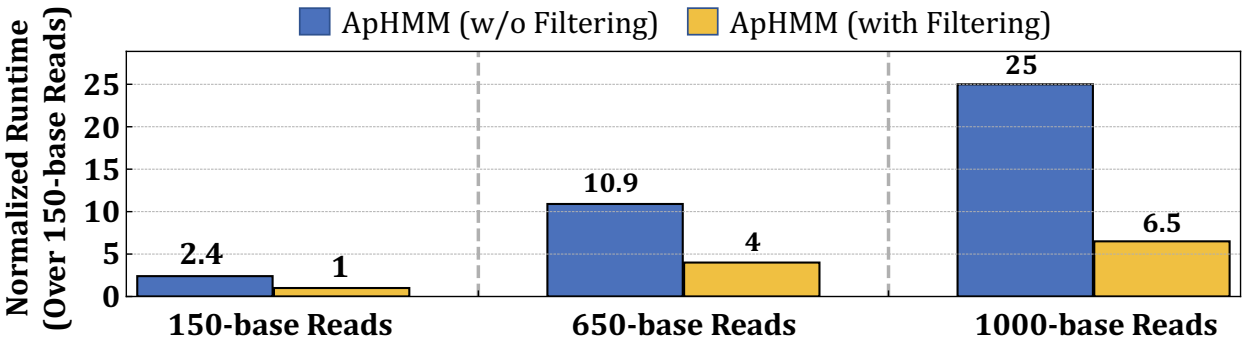
- Accurate comparison requires identifying changes (**insertions, deletions, substitutions**) between sequences due to
  - Variations between individuals and template sequences
  - Errors in sequences



- How to avoid unnecessary (and costly) comparisons?

Filtering – Performance Benefits

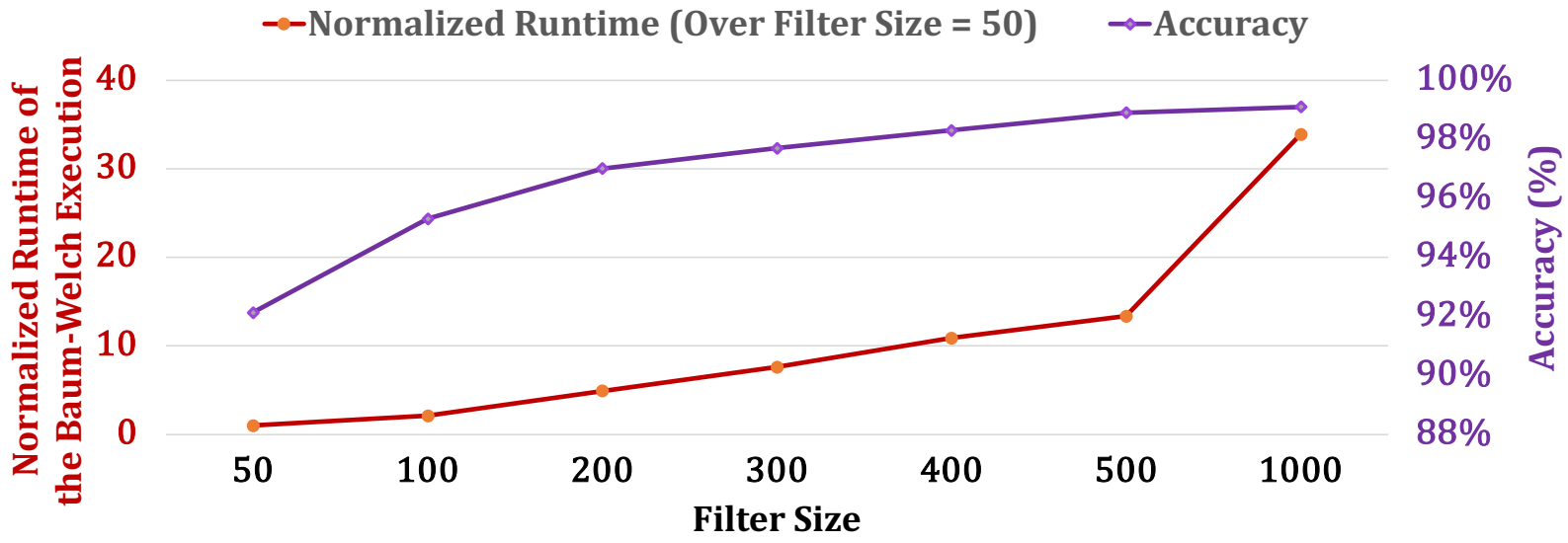
- Filtering heuristics aim to reduce unnecessary computations



Motivational Study: ~2.5x performance improvements with filtering

Filtering – Accurate but Costly Sorting

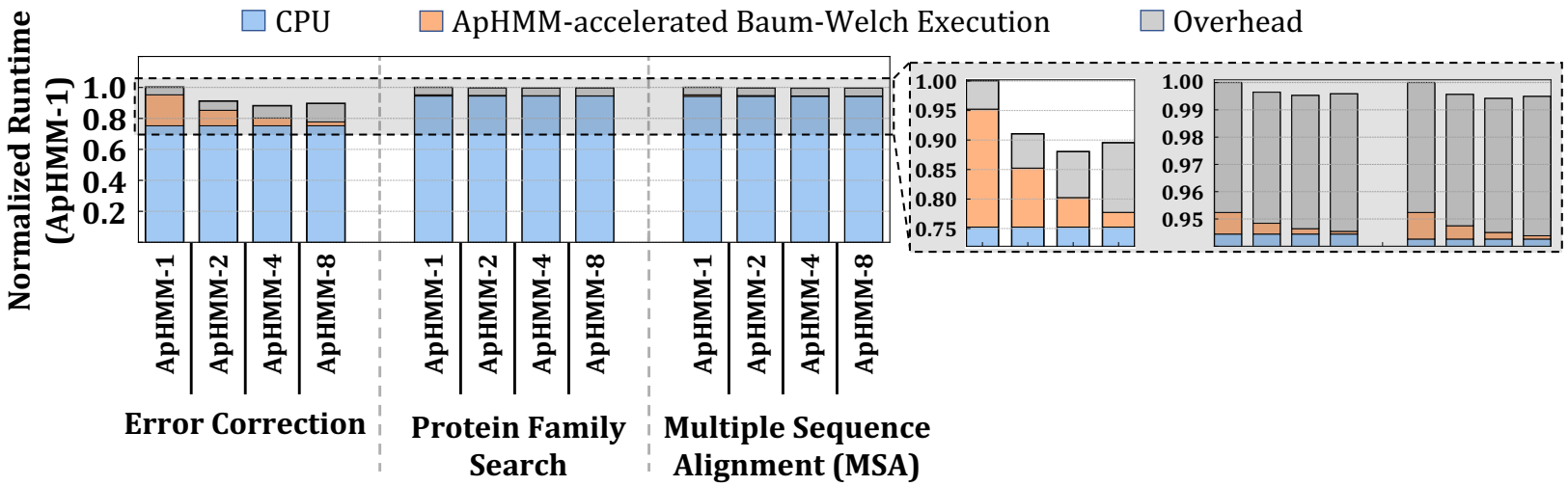
- Software-based filtering heuristics aim to reduce unnecessary computations
  - High-accuracy can be achieved with filtering with correct setting



Filtering takes up ~8.5% of the overall execution time **due to sorting**

Choosing the Right Amount of Cores

- We analyze maximum number of cores that ApHMM can utilize
  - Before it is bottlenecked by memory bandwidth for genomics applications



ApHMM with 4 cores (ApHMM-4) provides the best overall speedup