Fast and Efficient Genome Analysis
via New Algorithms and Architectures

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

6 May 2025
Inria – GenScale & Symbiose

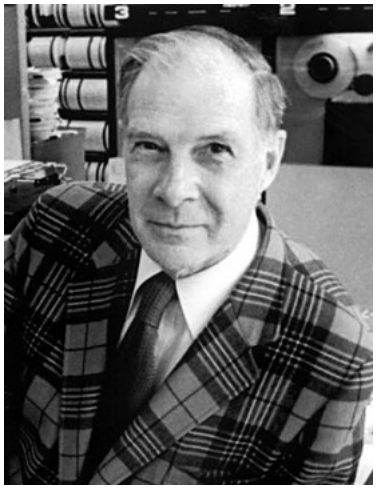
ETH zürich

SAFARI

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.

Computing is Bottlenecked by Data

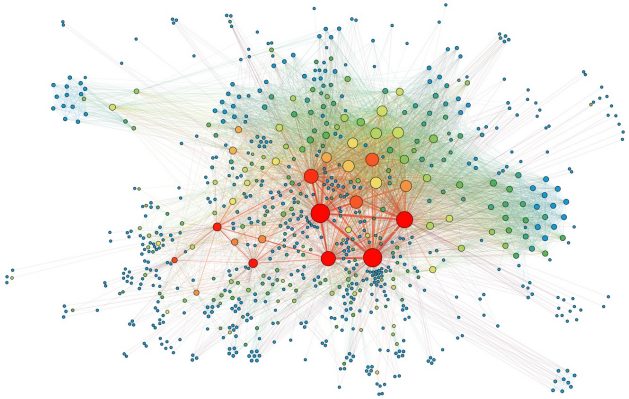
Large recommender systems

ChatGPT

Gemini

Claude 3.5 Sonnet

Large Language Models



Graph/Tree Processing

Genomics

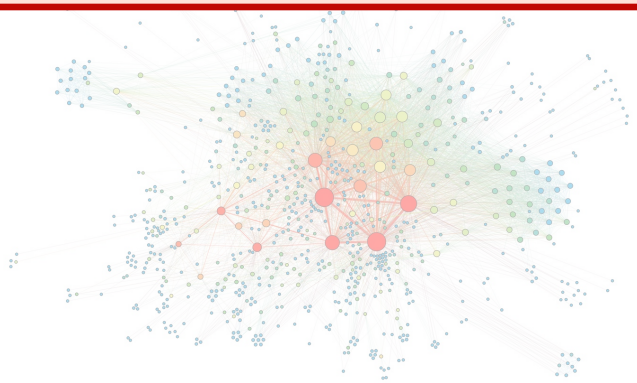
Computing is Bottlenecked by Data

ChatGPT

Gemini

Claude 3.5 Sonnet

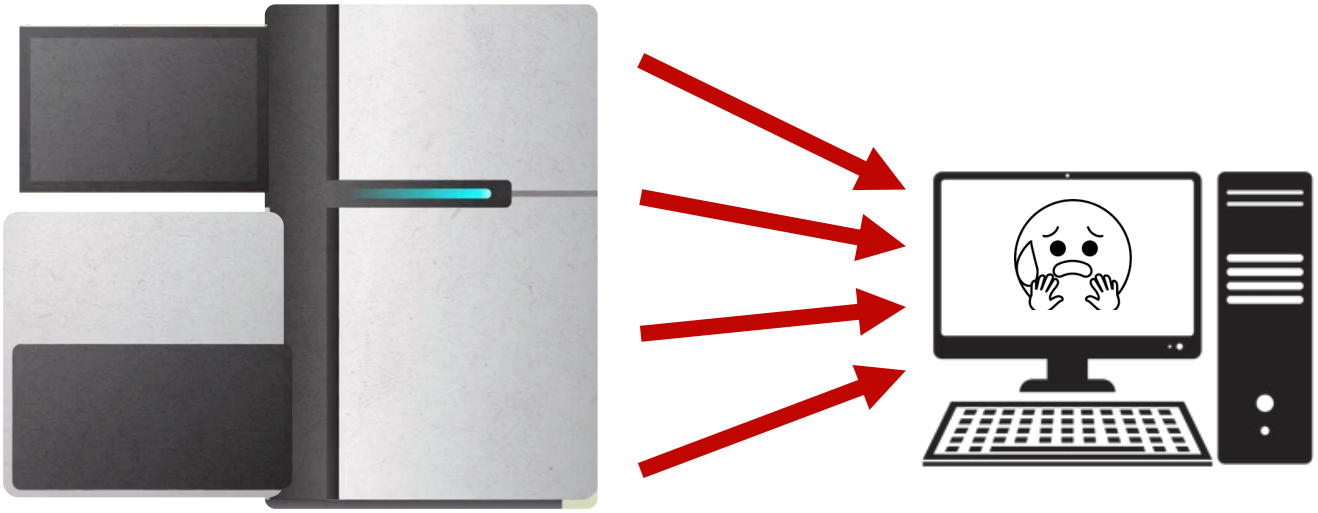
Data movement →
Performance & Energy Bottleneck



Graph/Tree Processing

Genomics

Problems with Data Analysis Today



Special-Purpose Machine
for **Data Generation**

FAST

General-Purpose Machine
for **Data Analysis**

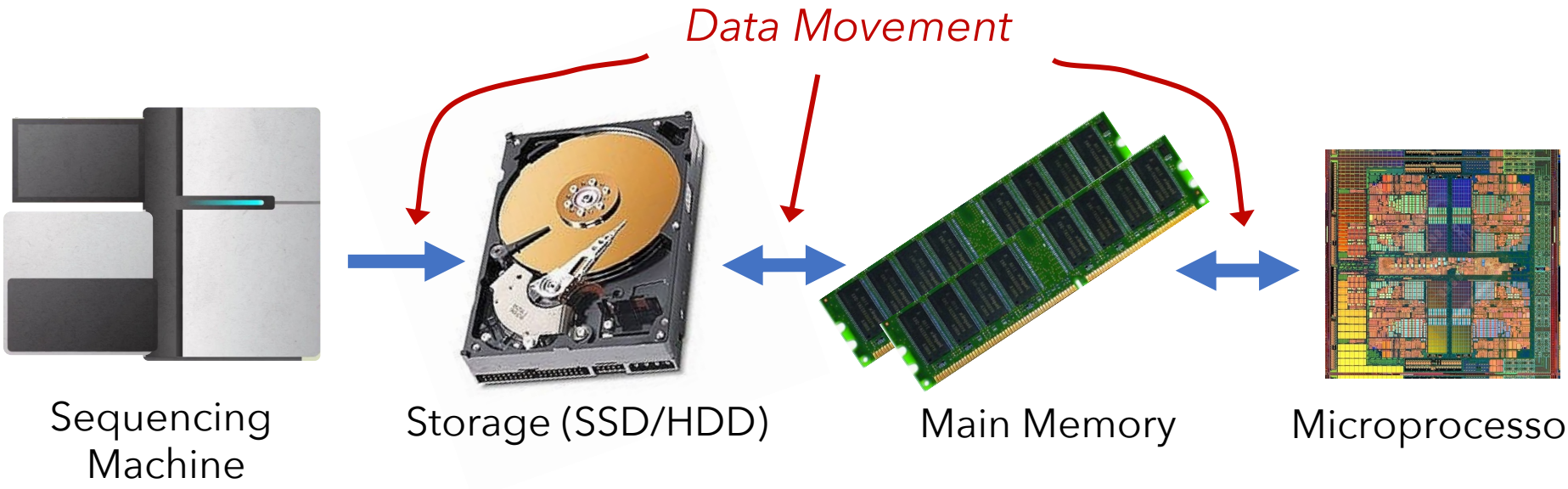
SLOW

Processing capabilities oblivious to data analysis requirements

Large amounts of data movement

Data Movement Dominates Performance

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



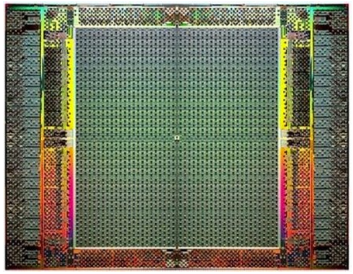
Single memory request consumes **>160× - 800× more energy** compared to performing an addition operation

- 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

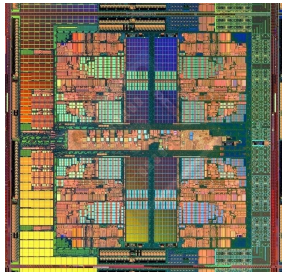
We need to **co-design**
algorithms and **architecture**
to handle data well

Pushing Towards New Architectures

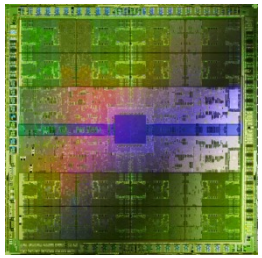
Modern systems



FPGAs

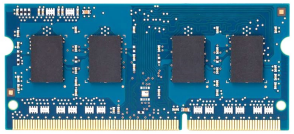


Heterogeneous Processors & Accelerators

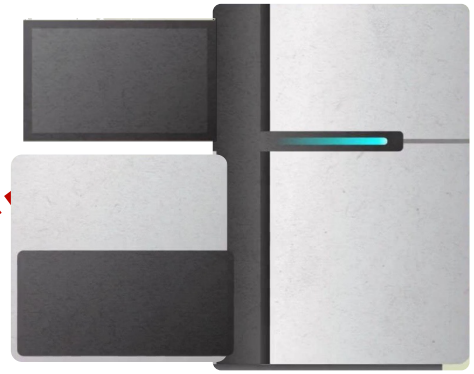


GPUs

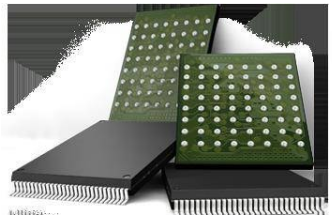
?



Hybrid Main Memory

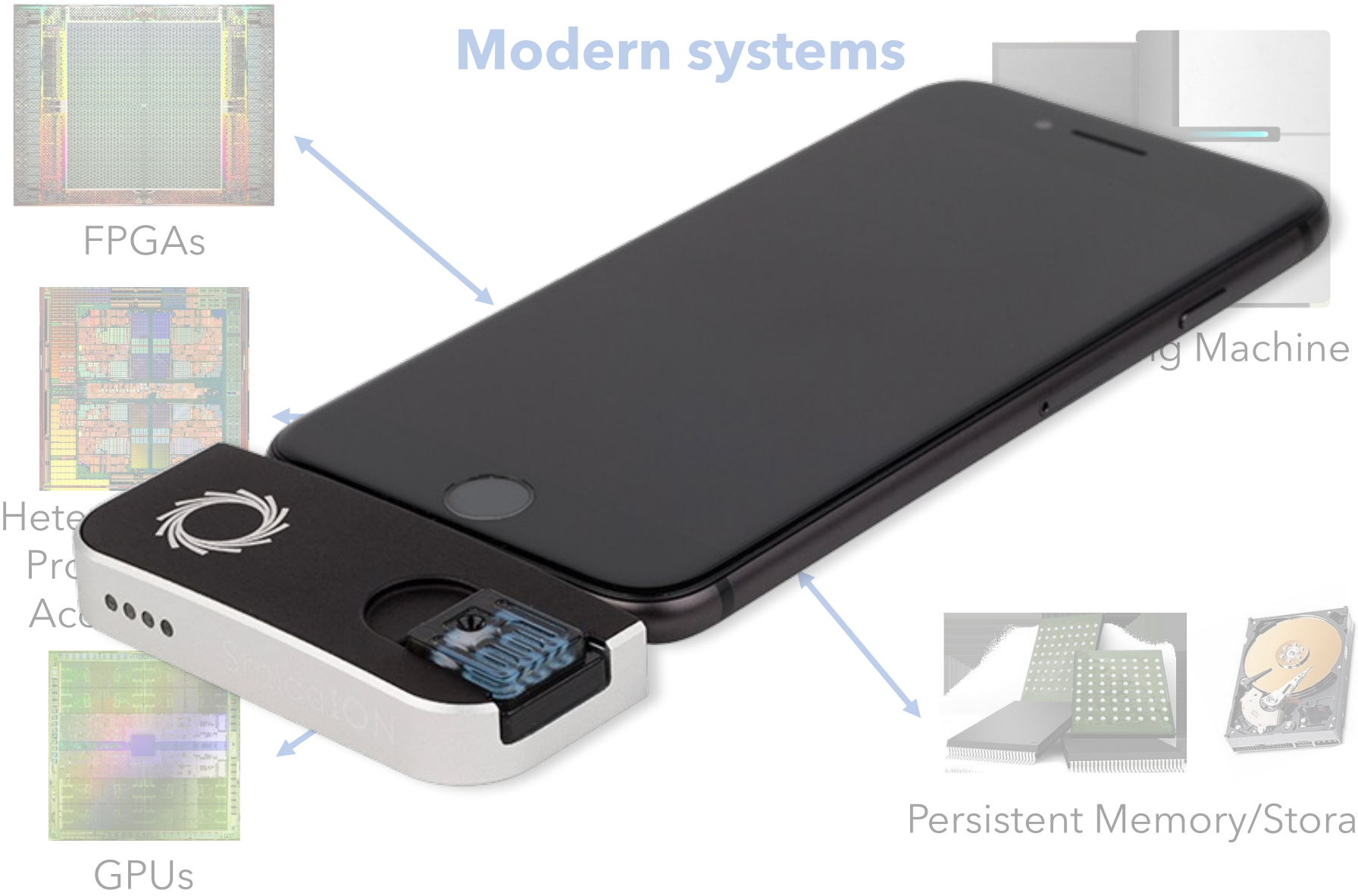


Sequencing Machine



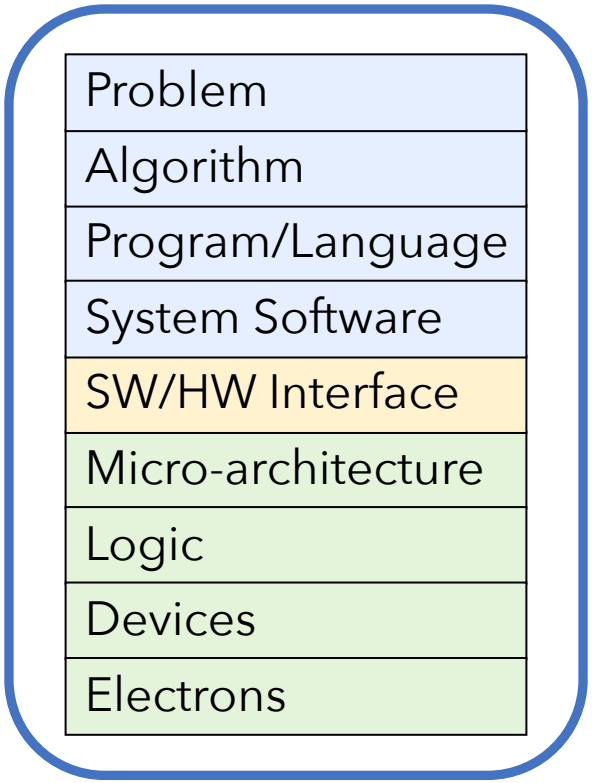
Persistent Memory/Storage

Pushing Towards New Architectures



Algorithm-Architecture Co-design is Critical

Computer Architecture (expanded view)



Accelerating Genome Analysis [DAC '23]

- 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.
  [Related Invited Paper]
  [arXiv version]
  [Slides (pptx) (pdf)]
  [Talk Video at DAC 2023] (38 minutes, including Q&A)

Analyzing Genomes Reveals Key Insights

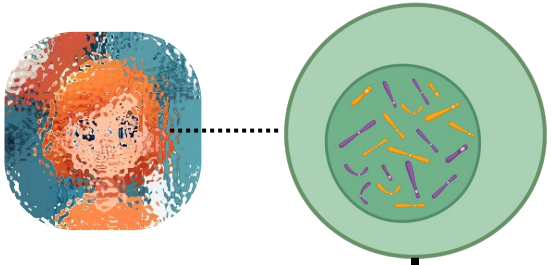
Personalized Medicine

Altering genomes to solve fundamental challenges of life

Detecting **pathogens** in the environment

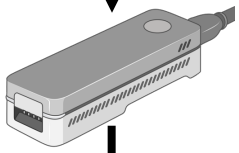
Rapid surveillance of **disease outbreaks**

Generating the Entire Genomic Sequence



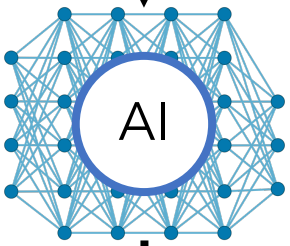
Whole Genome: 6.4B bps
(Non-human-readable)

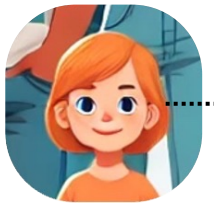
Sequencing Device



Raw Genomic Data
(Human-readable?)

Translator
(Basecalling)

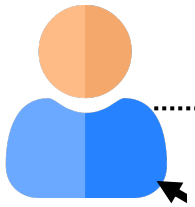




CGAAATGCC...AGATTAAACGCT...TGCCCTAA...GAATGGCGT

Whole Human Genome: 6.4B bps (Human-readable)

Genomic Data: Giant Jigsaw Puzzle to Solve



Human (Reference) Genome: 3.2 Billion bps (haploid)

CGAAATGCC...AGATTAAACGCT...TGCCCTAA...GAATGGCGT

?

CGAAATGCC...AGATTAAATGGT...TGCCCTAA...GAATGGCGT

Entire Genomic Sequence of an Organism

- **Small pieces (reads)**
- **Unknown origins**
- **Non-identical reference genome**
- **Large volume of data to analyze**

Data Science for Massive Datasets (TBs to PBs)

Solving the Puzzle: The Naïve Way

Pairwise Approximate String Matching (Alignment) → **Entire Genome** → **For All Reads**

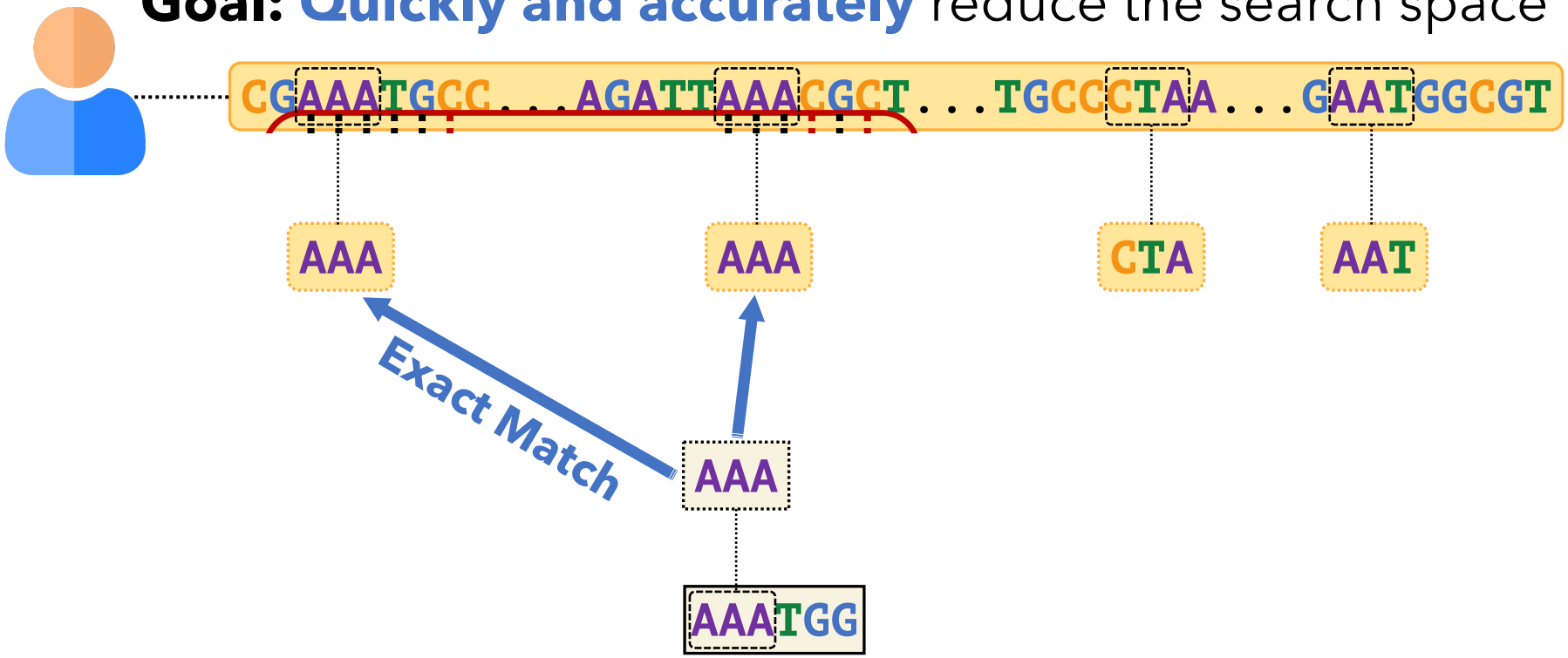
$O(nmk)$

$\cong O(10^{20})$

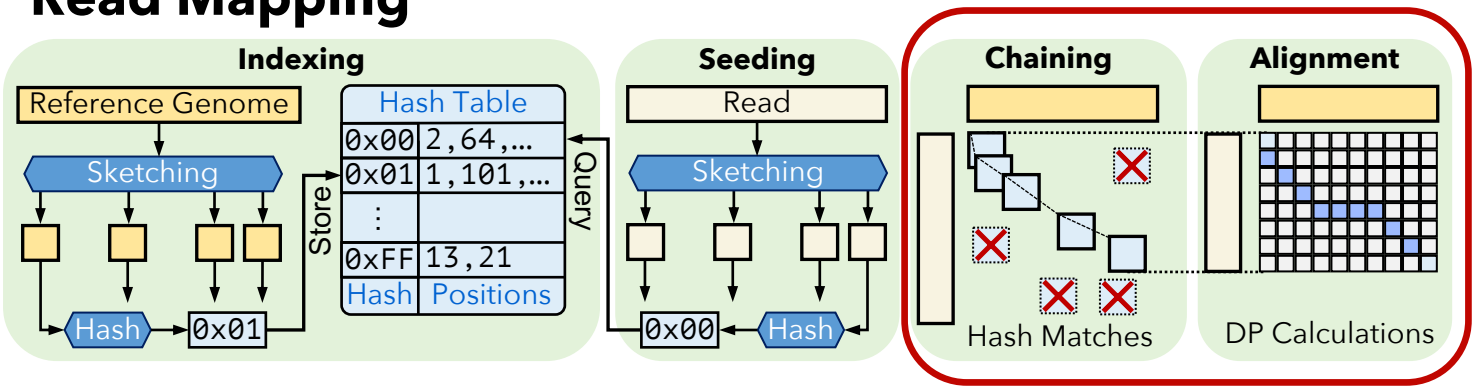
≅Years

Solving the Puzzle: The Practical Way

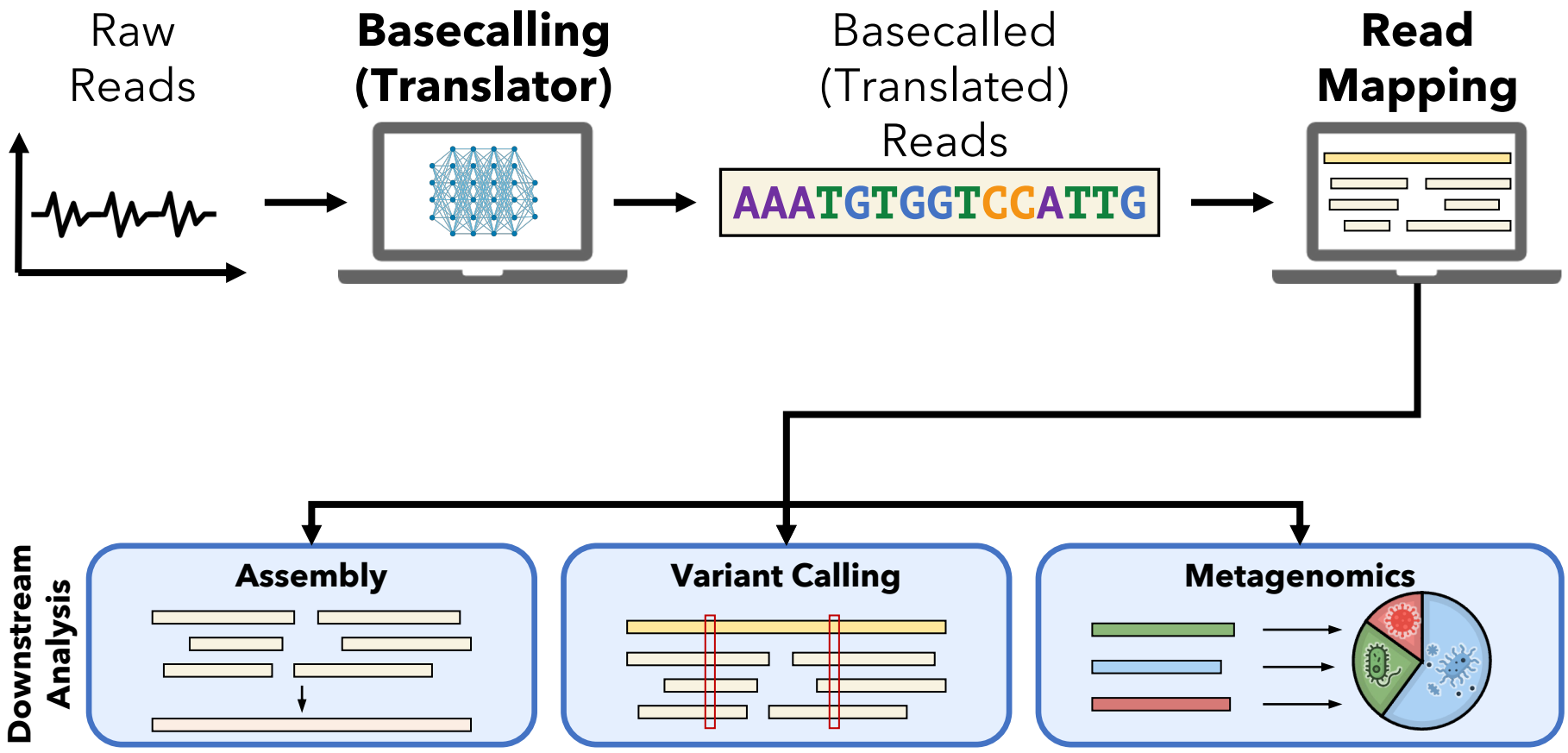
Goal: Quickly and accurately reduce the search space



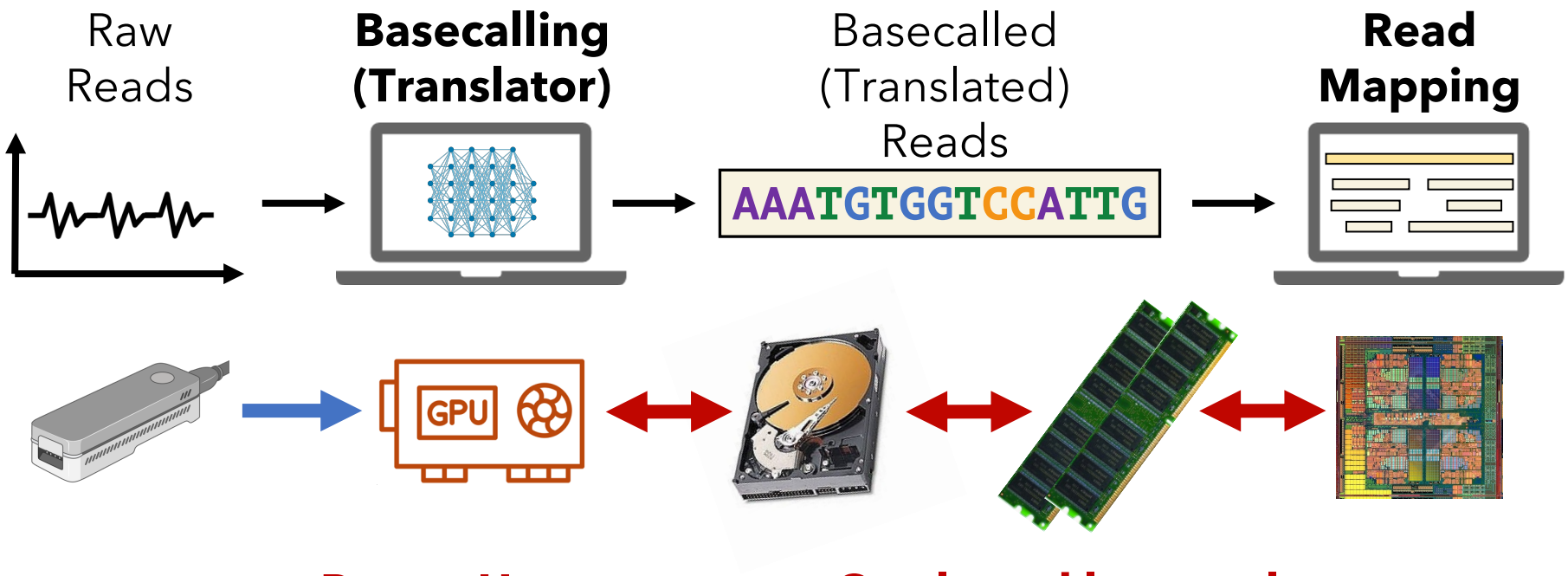
Read Mapping



A Common Genome Analysis Pipeline



Genome Analysis is Energy Inefficient



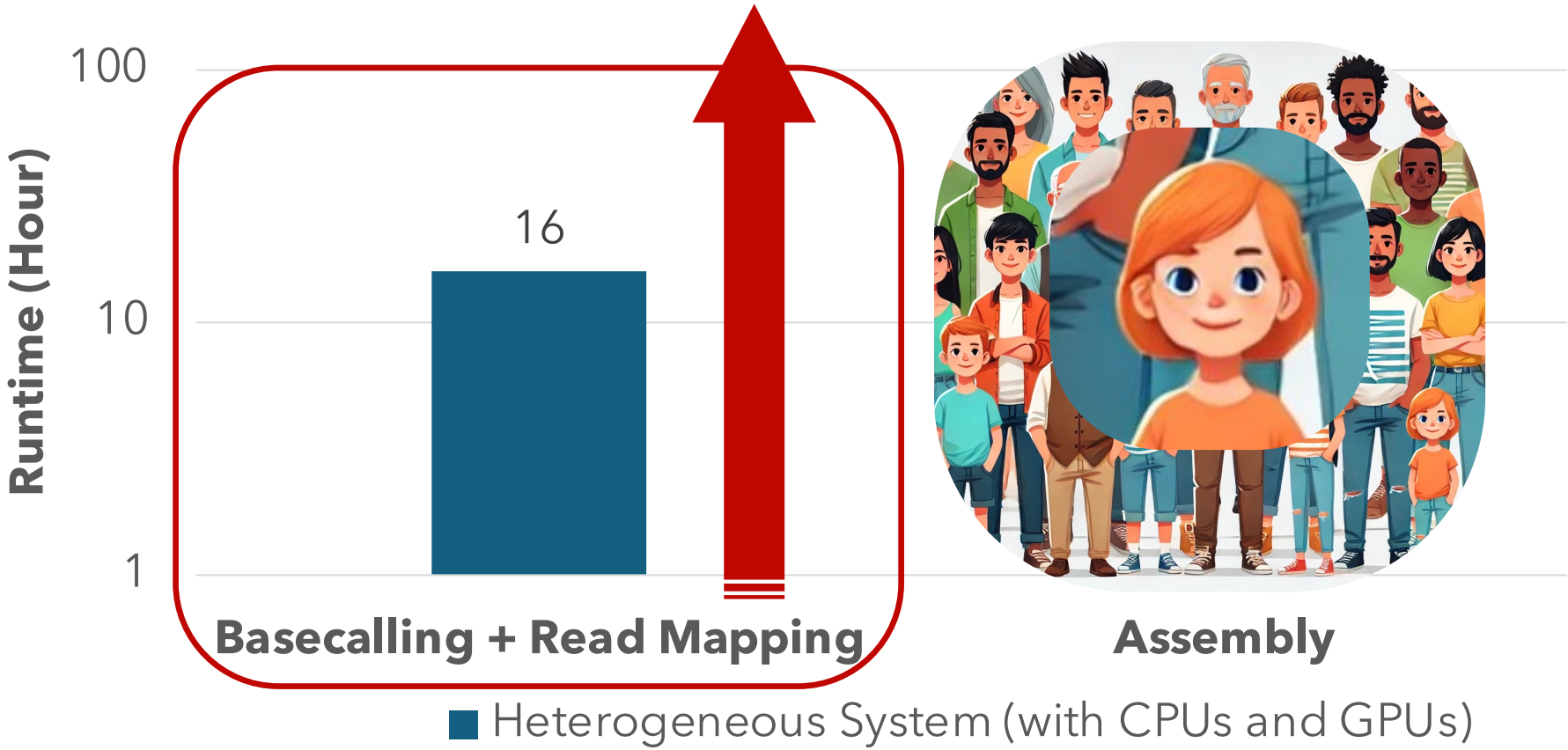
Power-Hungry Devices

Costly and large volumes of data movement

Single memory request consumes **>160× - 800× 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

Genome Analysis is Computationally Costly



Significant computation and energy overhead

Limited Application Scope and Accuracy

Energy-efficient analysis for Resource-constrained devices

Urgent analysis within minutes

Large-scale analysis without sacrificing accuracy and performance

Enabling New Directions in Genome Analysis **via New Algorithms**

New Frontiers: Raw Signal Analysis [ISMB '23]

- 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**)*, July 2023
  [Online link at Bioinformatics Journal]
  [arXiv version]
  [Slides (pptx) (pdf)]
  [Talk Video at ISMB/ECCB 2023] (19 minutes)
  [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.)

Real-Time Raw Signal Analysis [Bioinform. '24]

- Can Firtina, Melina Soysal, Joël Lindegger, and Onur Mutlu,
  "RawHash2: Mapping Raw Nanopore Signals Using Hash-Based Seeding and Adaptive Quantization"
  Bioinformatics, July 2024.
  [Online link at Bioinformatics Journal]
  [arXiv version]
  [RawHash Talk Video] (19 minutes)
  [RawHash2 Source Code]

Bioinformatics, 2024, **40(8)**, btae478
https://doi.org/10.1093/bioinformatics/btae478
Advance Access Publication Date: 30 July 2024
Applications Note

OXFORD

Sequence analysis

RawHash2: mapping raw nanopore signals using hash-based seeding and adaptive quantization

Can Firtina [1,*], Melina Soysal [1], Joël Lindegger [1], Onur Mutlu [1,*]

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

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

New Directions: Assembling Raw Signals

- Can Firtina, Maximilian Mordig, Harun Mustafa, Sayan Goswami, Nika Mansouri Ghiasi, Stefano Mercogliano, Furkan Eris, Joël Lindegger, Andre Kahles, and Onur Mutlu
  "Rawsamble: Overlapping and Assembling Raw Nanopore Signals using a Hash-based Seeding Mechanism"
  arXiv, October 2024
  [arXiv version]
  [Talk Video at CSHL Data Science 2024] (16 minutes)
  [Source Code]

arXiv

Source code

Rawsamble: Overlapping and Assembling Raw Nanopore Signals using a Hash-based Seeding Mechanism

Can Firtina[1] Maximilian Mordig[1,2] Harun Mustafa[1,3,4] Sayan Goswami[1] Nika Mansouri Ghiasi[1]
Stefano Mercogliano[1] Furkan Eris[1] Joël Lindegger[1] Andre Kahles[1,3,4] Onur Mutlu[1]
[1]*ETH Zurich* [2]*Max Planck Institute for Intelligent Systems*
[3]*University Hospital Zurich* [4]*Swiss Institute of Bioinformatics*

Quickly Aligning Raw Signals

- 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"
  IEEE Access, December 2024.
  [Online link at IEEE Access Journal]
  [arXiv version]
  [RawAlign Source Code]

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

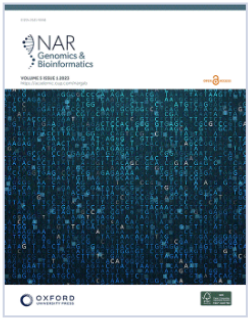
JOËL LINDEGGER, CAN FIRTINA, NIKA MANSOURI GHIASI, MOHAMMAD SADROSADATI, MOHAMMED ALSER, AND ONUR MUTLU, (Fellow, IEEE)
Department of Information Technology and Electrical Engineering, ETH Zürich, 8092 Zürich, Switzerland

Corresponding authors: Joël Lindegger (jmlindegger@gmail.com), Can Firtina (firtinac@ethz.ch), and Onur Mutlu (omutlu@ethz.ch)

This work was supported in part by the Semiconductor Research Corporation (SRC), in part by the ETH Future Computing Laboratory, in part by European Union's Horizon Programme for Research and Innovation under Grant 101047160-BioPIM, and in part by the Swiss National Science Foundation (SNSF) under Grant 200021_213084.

Tolerating Noise in String Matching [NARGAB '23]

- 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 version]
  [bioRxiv version]
  [Talk Video at RECOMB 2023] (23 minutes)
  [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,

Mitigating Useless Computations [TCBB '24]

- Jeremie S. Kim*, Can Firtina*, Meryem Banu Cavlak, Damla Senol Cali, Nastaran Hajinazar, Mohammed Alser, Can Alkan, and Onur Mutlu,
  "AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes"
  IEEE/ACM TCBB, August 2024.
  [Online link at IEEE/ACM TCBB Journal]
  [arXiv version]
  Presented at the 21st Asia Pacific Bioinformatics Conference (**APBC**), Changsha, China, April 2023.
  [Slides (pptx) (pdf)]
  [Talk Video at BIO-Arch 2023 Workshop] (22 minutes)
  [AirLift Source Code]

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

Jeremie S. Kim[1,†] Can Firtina[1,†] Meryem Banu Cavlak[1] Damla Senol Cali[2]
Nastaran Hajinazar[1,3] Mohammed Alser[1] Can Alkan[4] Onur Mutlu[1,2,4]
[1]*ETH Zurich* [2]*Carnegie Mellon University* [3]*Simon Fraser University* [4]*Bilkent University*

*Equal contribution

Mitigating Useless Computations [Bioinform. '22]

- Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Can Alkan, and Onur Mutlu,
  "FastRemap: A Tool for Quickly Remapping Reads between Genome Assemblies"
  Bioinformatics, October 2022.
  [Online link at Bioinformatics Journal]
  [arXiv preprint]
  [FastRemap Source Code]

Bioinformatics, 38(19), 2022, 4633–4635
https://doi.org/10.1093/bioinformatics/btac554
Advance Access Publication Date: 17 August 2022
Applications Note

OXFORD

Genome analysis

FastRemap: a tool for quickly remapping reads between genome assemblies

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

[1]Department of Computer Engineering, ETH Zurich, D-ITET, Zurich 8006, Switzerland, [2]Department of Computer Engineering, Bionano Genomics, San Diego, CA 92121, USA and [3]Department of Computer Engineering, Bilkent University, Ankara 06800, Turkey

Utilizing AI/ML in

Genome Analysis

via New Algorithms and Architectures

Error Correction using ML [Bioinform. '20]

- 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.
  [Online link at Bioinformatics Journal]
  [arXiv version]
  [Slides (pptx)][(pdf)]
  [Source Code]

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 ML & Genome Graphs [TACO '24]

- Can Firtina, Kamlesh Pillai, Gurpreet S. Kalsi, Bharathwaj Suresh, Damla Senol Cali, Jeremie S. Kim, Taha Shahroodi, Meryem Banu Cavlak, Joel Lindegger, Mohammed Alser, Juan Gomez Luna, Sreenivas Subramoney and Onur Mutlu, **"ApHMM: Accelerating Profile Hidden Markov Models for Fast and Energy-efficient Genome Analysis"**
  ACM Transactions on Architecture and Code Optimization (***TACO***), Feb 2024.
  [ACM Digital Library version]
  [arXiv version]
  Presented at the 19th HiPEAC Conference, Munich, Germany, January 2024.
  [Slides (pptx) (pdf)]
  [Talk Video HiPEAC 2024] (35 minutes)
  [ApHMM Source Code]

RESEARCH-ARTICLE | OPEN ACCESS |

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

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, Volume 21, Issue 1 • Article No.: 19, Pages 1 - 29
https://doi.org/10.1145/3632950

Translating Raw Signals using ML [Genome. Biol. '24]

- Gagandeep Singh, Mohammed Alser, Kristof Denolf, Can Firtina, Alireza Khodamoradi, Meryem Banu Cavlak, Henk Corporaal and Onur Mutlu, **"RUBICON: A Framework for Designing Efficient Deep Learning-Based Genomic Basecallers"**
  Genome Biology, February 2024.
  [arXiv version]
  [Online link at Genome Biology Journal]
  [RUBICON Source Code]

Method | Open access | Published: 16 February 2024

RUBICON: a framework for designing efficient deep learning-based genomic basecallers

Gagandeep Singh, Mohammed Alser, Kristof Denolf, Can Firtina ✉, Alireza Khodamoradi, Meryem Banu Cavlak, Henk Corporaal & Onur Mutlu ✉

Genome Biology **25**, Article number: 49 (2024) | Cite this article

Eliminating Useless Basecalling [Front. In Genetics '24]

- 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"
  Frontiers in Genetics, October 2024.
  [Online link at Frontiers in Genetics Journal]
  [arXiv Version]
  [Talk Video at BIO-Arch 2023 Workshop]
  [TargetCall Source Code]

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] and Onur Mutlu[1]*

[1]SAFARI Research Group, Department of Information Technology and Electrical Engineering, ETH Zurich, Zurich, Switzerland, [2]Department of Computer Engineering, Bilkent University, Ankara, Türkiye

ML-based Genome Analysis via PIM [MICRO '22]

- 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*

Basecalling using PIM [MICRO '23]

- Taha Shahroodi, Gagandeep Singh, Mahdi Zahedi, Haiyu Mao, Joel Lindegger, Can Firtina, Stephan Wong, Onur Mutlu, and Said Hamdioui, **"Swordfish: A Framework for Evaluating Deep Neural Network-based Basecalling using Computation-In-Memory with Non-Ideal Memristors"** *Proceedings of the 56th International Symposium on Microarchitecture (**MICRO**), Toronto, ON, Canada, November 2023.*
  [Slides (pptx) (pdf)]
  [arXiv version]

Swordfish: A Framework for Evaluating Deep Neural Network-based Basecalling using Computation-In-Memory with Non-Ideal Memristors

Taha Shahroodi[1] Gagandeep Singh[2,3] Mahdi Zahedi[1] Haiyu Mao[3] Joel Lindegger[3] Can Firtina[3]
Stephan Wong[1] Onur Mutlu[3] Said Hamdioui[1]

[1]TU Delft [2]AMD Research [3]ETH Zürich

Fast, Accurate, and Efficient

Genome Analysis

via Algorithm-Architecture co-design

Accelerating String Matching [MICRO '20]

- 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 '22]

- Damla Senol Cali, Konstantinos Kanellopoulos, Joel Lindegger, Zulal Bingol, Gurpreet S. Kalsi, Ziyi Zuo, Can Firtina, Meryem Banu Cavlak, Jeremie Kim, Nika Mansouri Ghiasi, 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.
[Slides (pptx) (pdf)]
[arXiv version]
[SeGraM Source Code and Datasets]
[Talk Video (22 minutes)]

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 Filtering for Genomics [ASPLOS '22]

- 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, Feb 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

In-Storage Metagenomics [ISCA '24]

- Nika Mansouri Ghiasi, Mohammad Sadrosadati, Harun Mustafa, Arvid Gollwitzer, Can Firtina, Julien Eudine, Haiyu Mao, Joel Lindegger, Meryem Banu Cavlak, Mohammed Alser, Jisung Park, and Onur Mutlu,
  "MegIS: High-Performance and Low-Cost Metagenomic Analysis with In-Storage Processing"
  *Proceedings of the 51st Annual International Symposium on Computer Architecture (**ISCA**)*, Buenos Aires, Argentina, July 2024.
  [Slides (pptx) (pdf)]
  [arXiv version]

MegIS: High-Performance, Energy-Efficient, and Low-Cost Metagenomic Analysis with In-Storage Processing

Nika Mansouri Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

Accelerating Genome Analysis [DAC '23]

- 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.
  [Related Invited Paper]
  [arXiv version]
  [Slides (pptx) (pdf)]
  [Talk Video at DAC 2023] (38 minutes, including Q&A)

My Involvements During my Ph.D.

Goal: Enable **Fast, Energy-Efficient, Accurate, and Scalable Genome Analysis** by Designing Algorithms and Architectures

Today's Talk

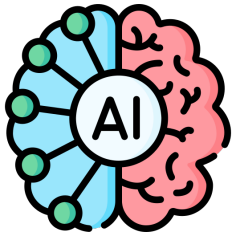
Real-Time and Raw Sequencing Data Analysis

[ISMB/ECCB'23]

[Bioinformatics'24]

[IEEE Access'24]

[arXiv'24 & HiTSeq'24]



Algorithm & Hardware Design for AI in Genome Analysis

[Bioinformatics'20]

[ACM TACO'24]

[Genome Biology'24]

[Front. in Genetics'24]

Accurate and Fast Algorithm Design for Read Mapping

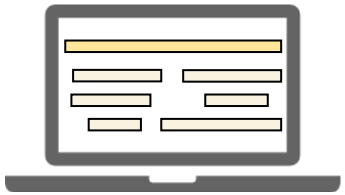
[Bioinformatics'22]

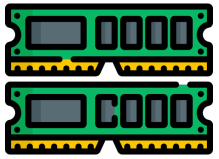
[NARGAB'23 & RECOMB'23]

[arXiv'23]

[arXiv'23]

[IEEE/ACM TCBB'24]





Data-Centric Architectures for Genome Analysis

[MICRO'20]

[MICRO'22]

[ISCA'22]

[ASPLOS'22]

[IEEE Access'22]

[MICRO'23]

[ISCA'24]

First/Co-First Author Publications

Reviews: [CSBJ'22] [DAC'23] [Nature Protocols'24]

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1

RawHash (ISMB/ECCB'23)

Enabling real-time and accurate read mapping without basecalling

2

Rawsamble (arXiv'24 & HiTSeq'24)

Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

3

ApHMM (TACO'24 & HiPEAC'24)

Accelerating ML operations in Genome Graphs

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

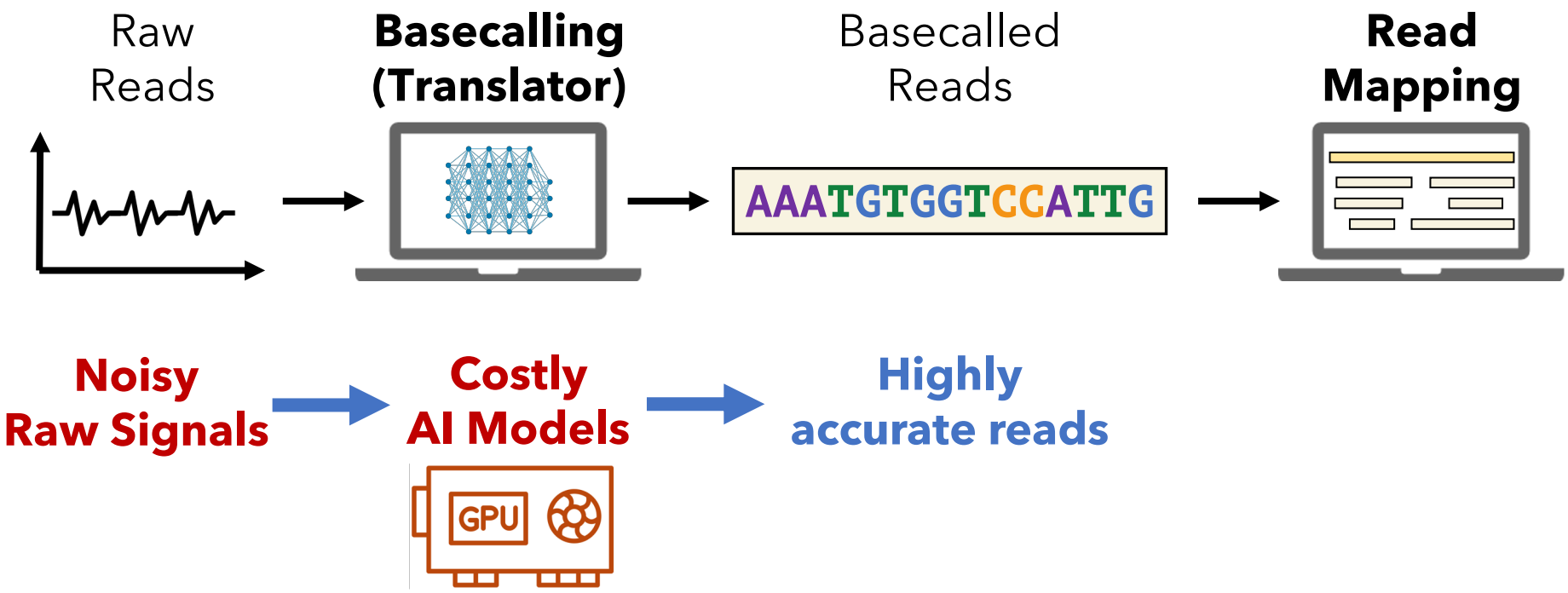
Real-Time and Raw Sequencing Data Analysis

1. **RawHash (ISMB/ECCB'23)**
   Enabling real-time and accurate read mapping without basecalling
2. **Rawsamble (arXiv'24 & HiTSeq'24)**
   Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

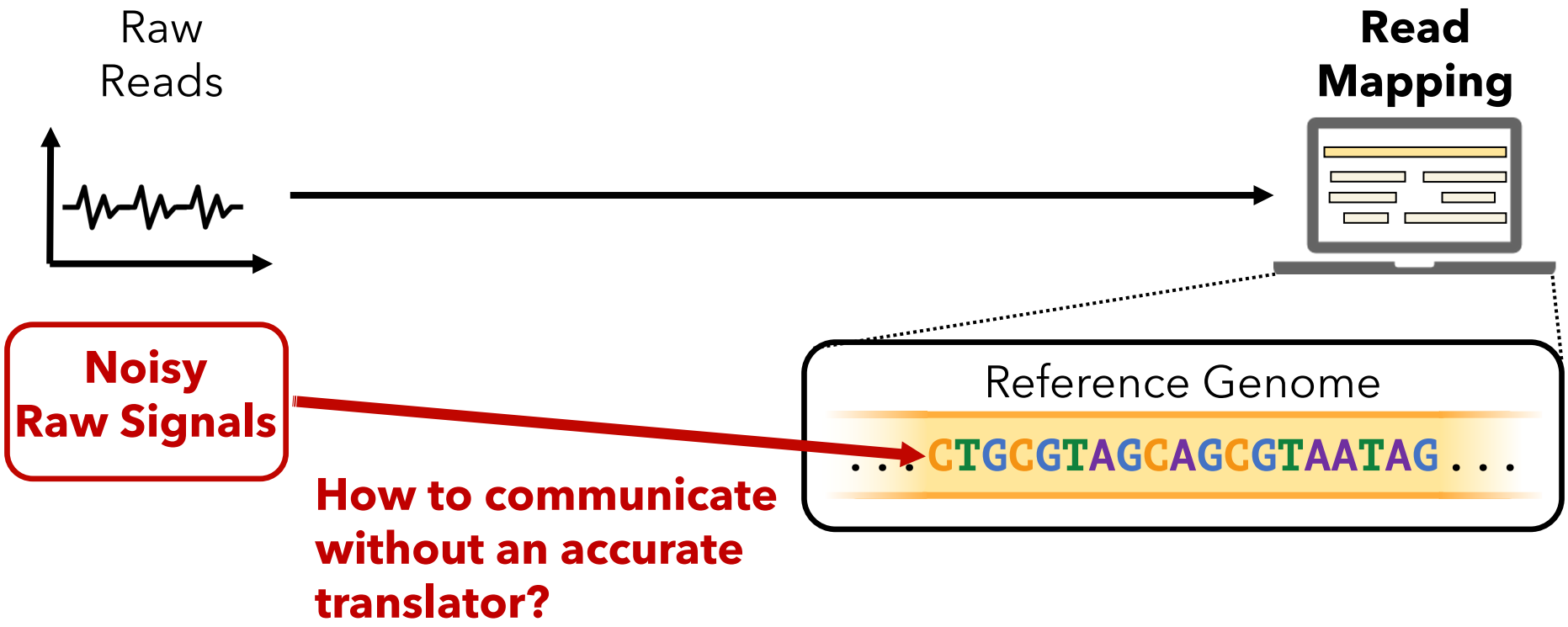
3. **ApHMM (TACO'24 & HiPEAC'24)**
   Accelerating ML operations in Genome Graphs

Basecalling is Accurate yet Costly

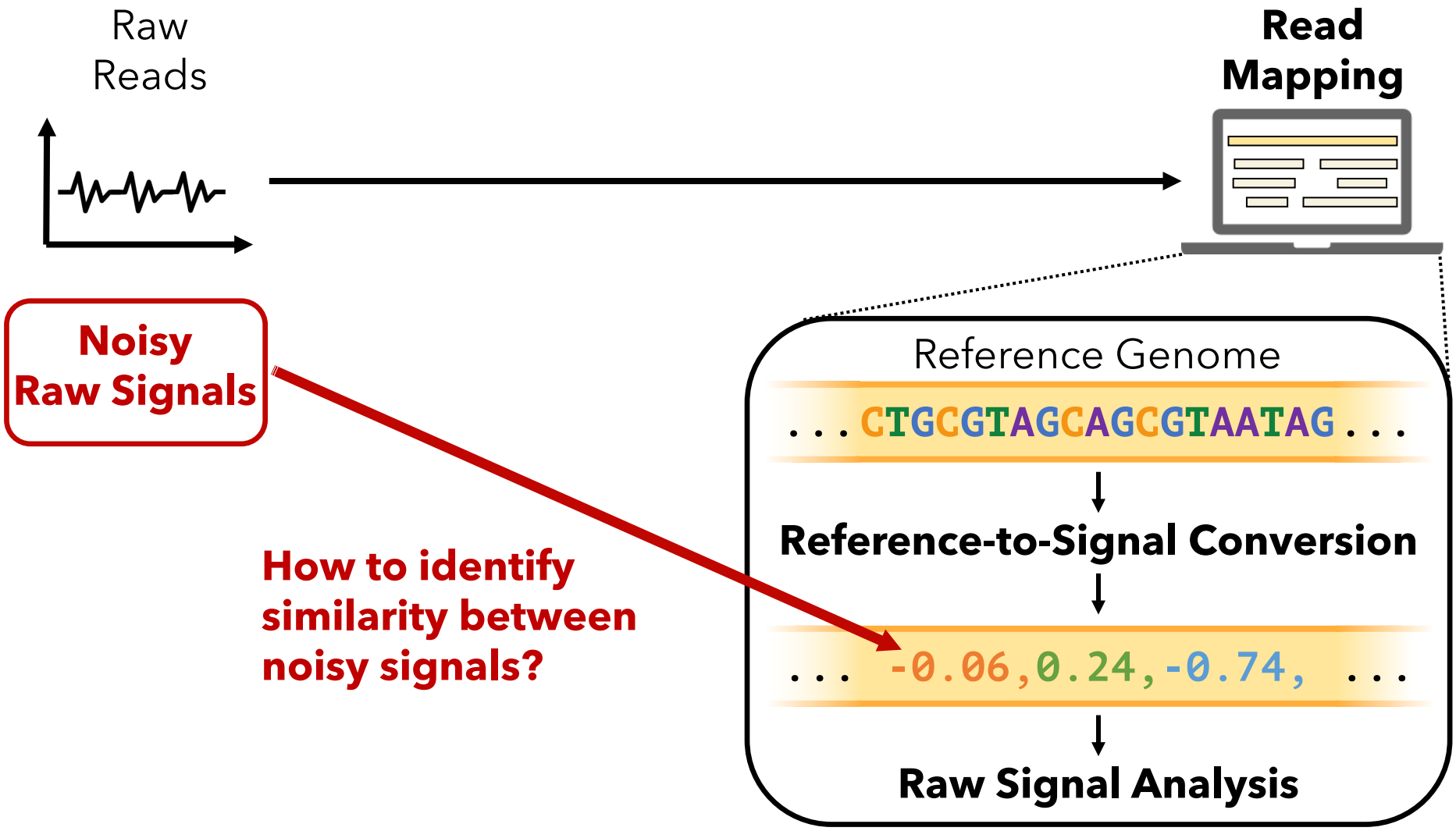


How to make this pipeline cheaper?

Can We Survive Without Translation?



Can We Survive With Noisy Signal Analysis?

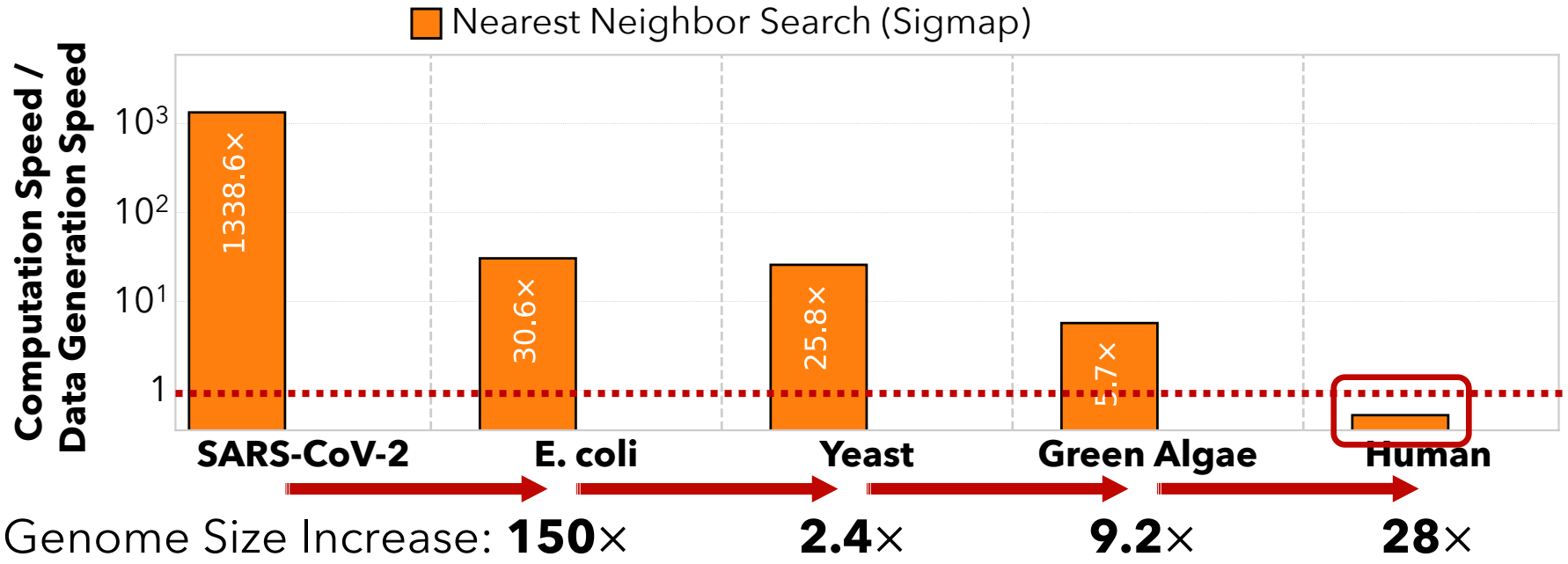


Noise in Raw Electrical Signals

Observation: Identical molecules generate similar raw signals

Problem: Too Costly for Large Genomes

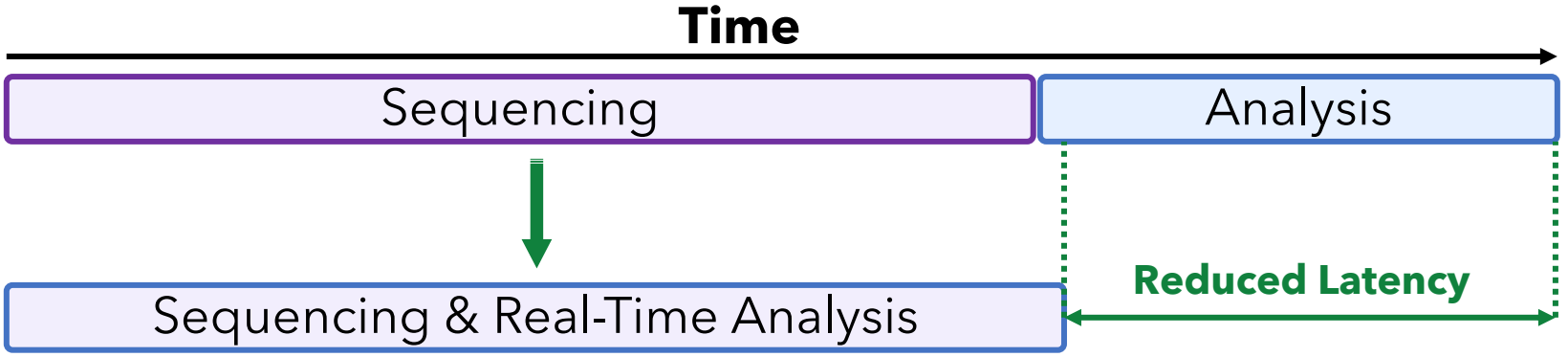
Scalability Issues Limit Real-Time Analysis



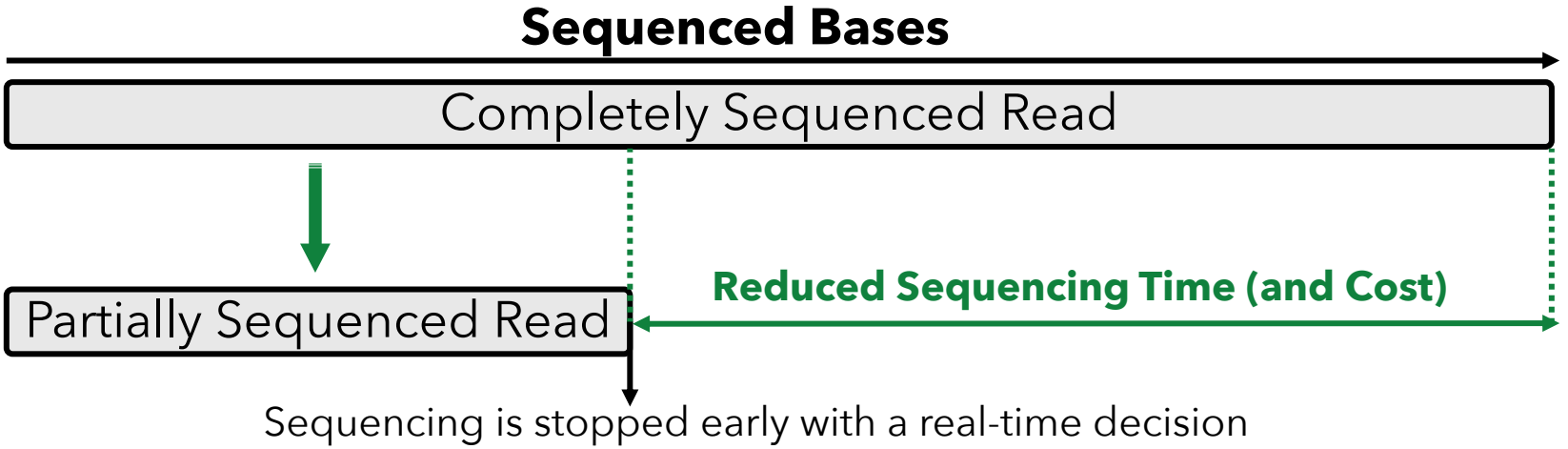
No Real-Time Analysis for Larger Genomes:
Searching mechanism does not scale well

Benefits of Real-Time Analysis

✓ **Reducing latency** by overlapping analysis with sequencing



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

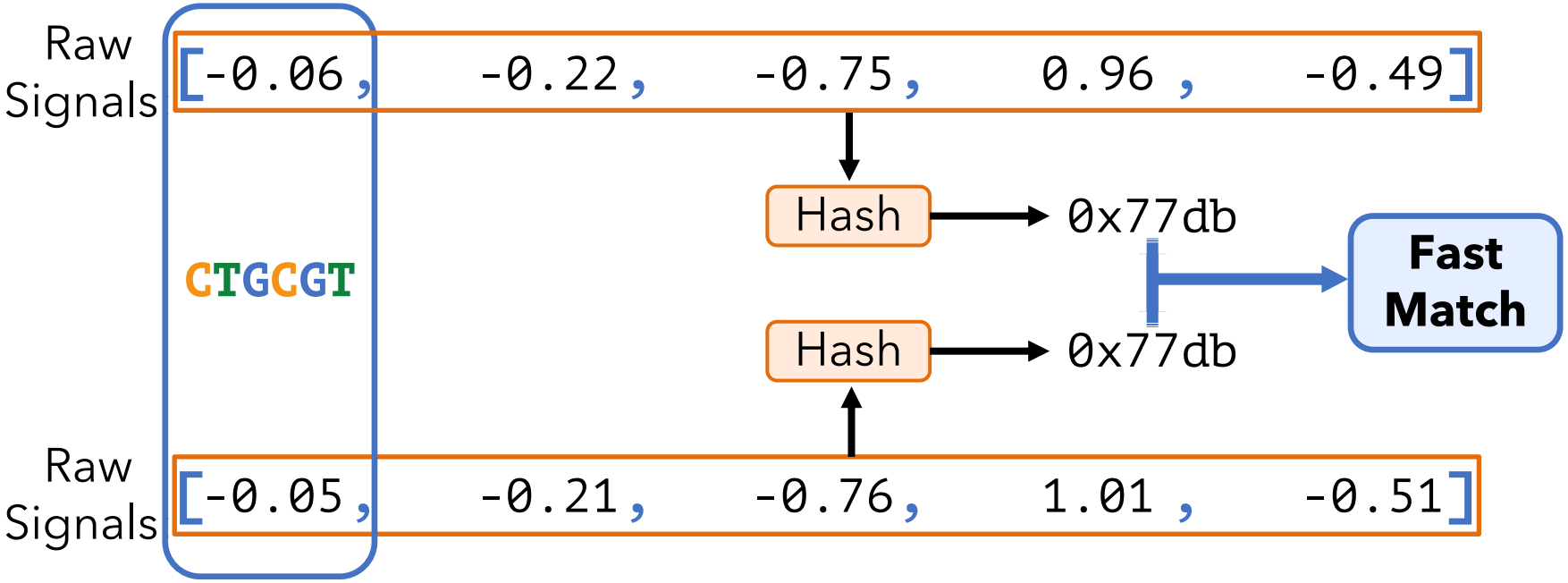


Goal

Enable **Fast, Accurate, and Scalable Real-Time Analysis** for Larger Genomes

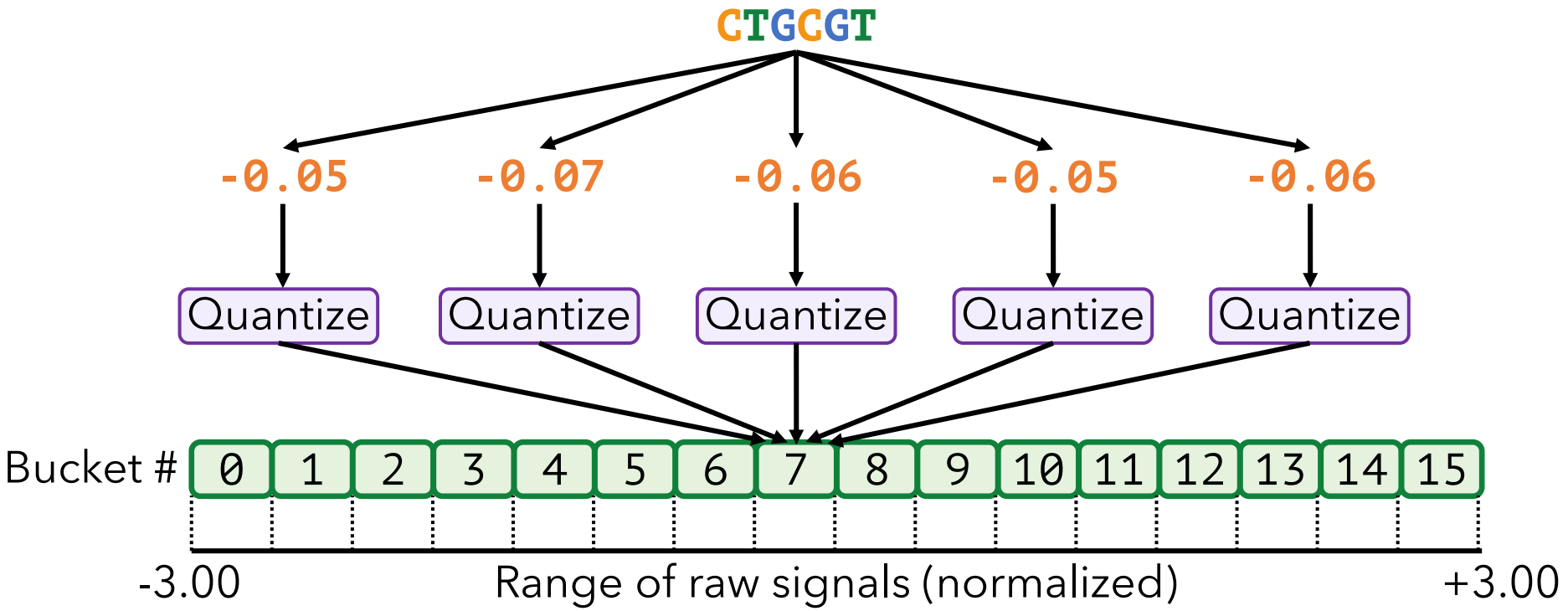
Key Idea

Key Idea: Perform similarity estimation based on **quick exact matches**



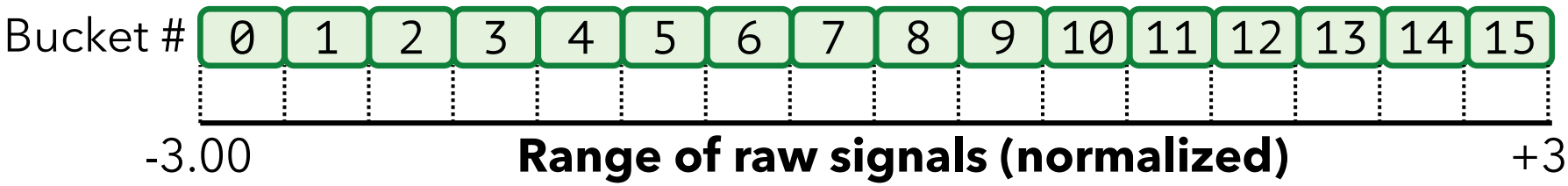
Challenge: How to generate **the same hash value** for **similar enough signals with various noise types?**

RawHash Key Idea – Quantization

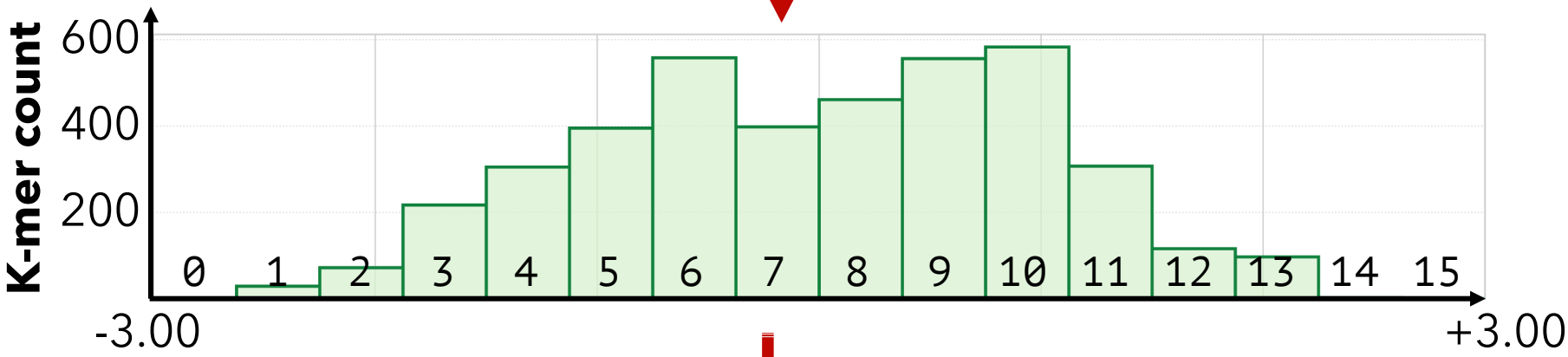


✓ **Enables matching raw signals** by eliminating slight differences

How to Better Quantize Signals?



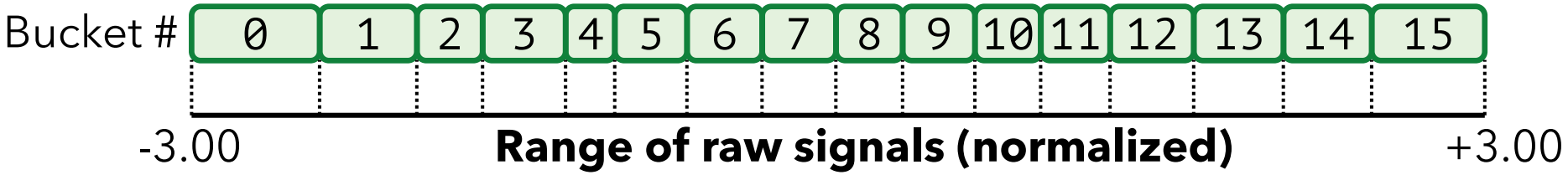
Equal-width buckets leads to
unbalanced bucket loading



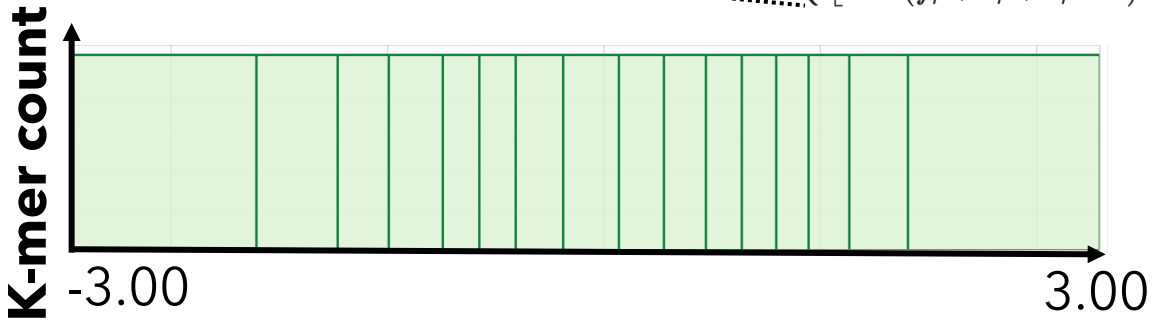
Reduced uniqueness and poor resource utilization

Adaptive Quantization

- **Key Idea:** Quantizing raw signals with **non-equal bucket widths** **by leveraging raw signal distribution**

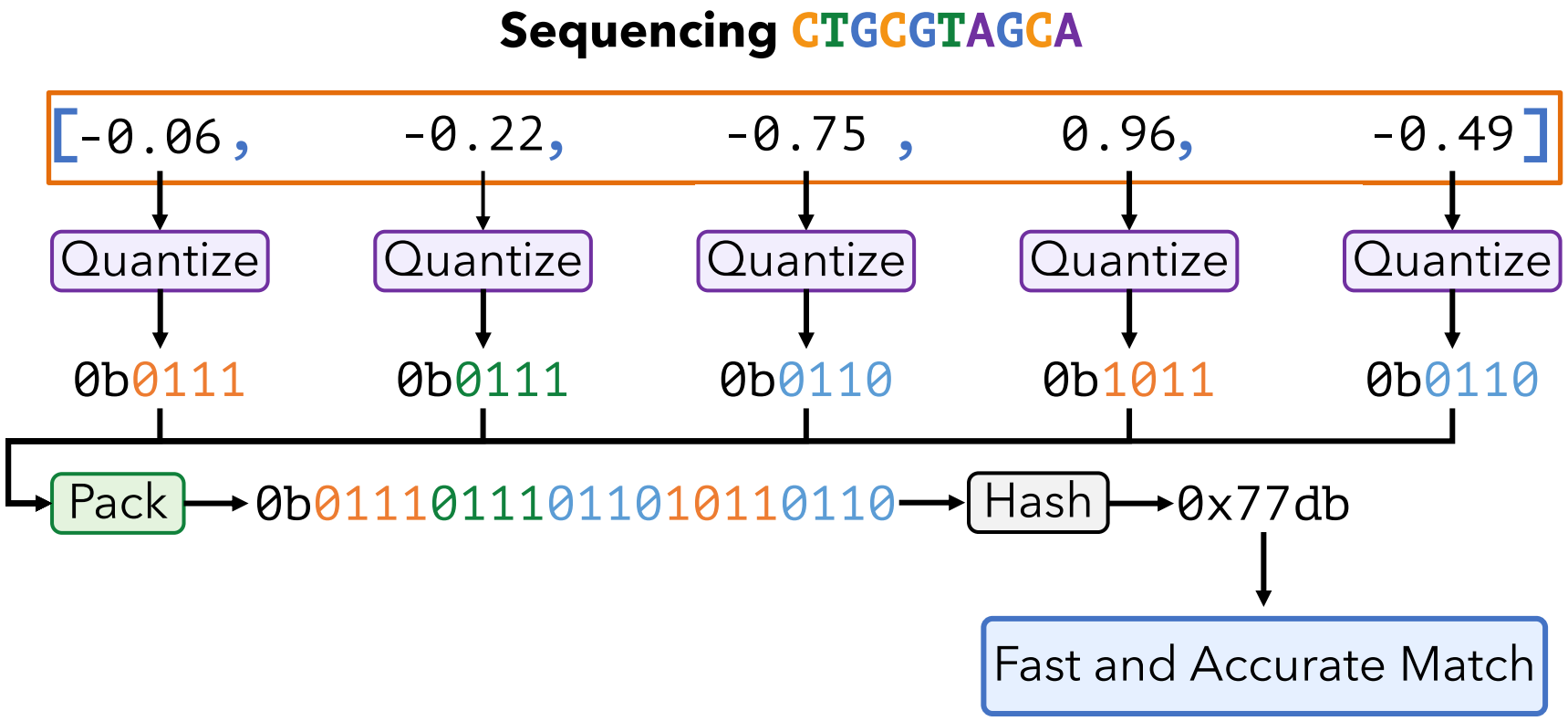


$$q(s) = \begin{cases} \lfloor n \times \left(f_r \times \frac{(s - f_{min})}{f_{max} - f_{min}} \right) & \text{if } f_{min} \leq s \leq f_{max} \\ \lfloor n \times (f_r + c_r \times s) & \text{if } s < f_{min} \\ \lfloor n \times (f_r + c_r + c_r \times s) & \text{if } s > f_{max} \end{cases}$$



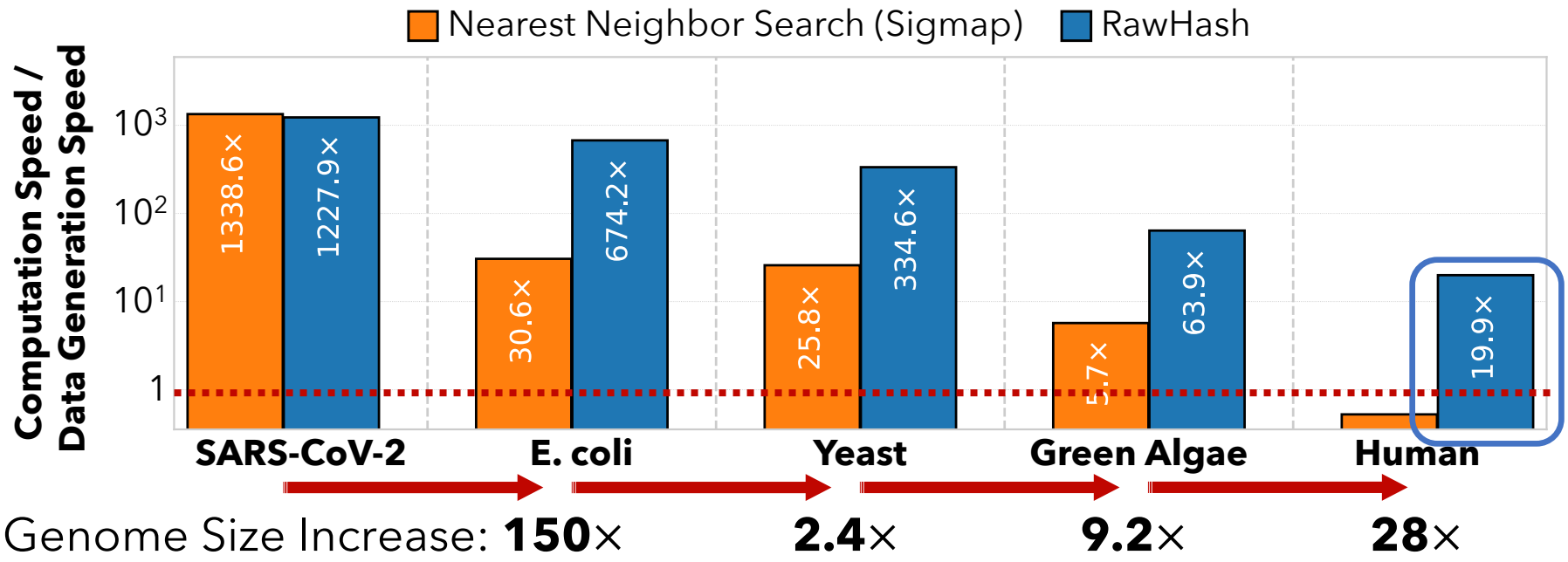
Better bucket utilization

RawHash Key Idea – Hash-based Matching



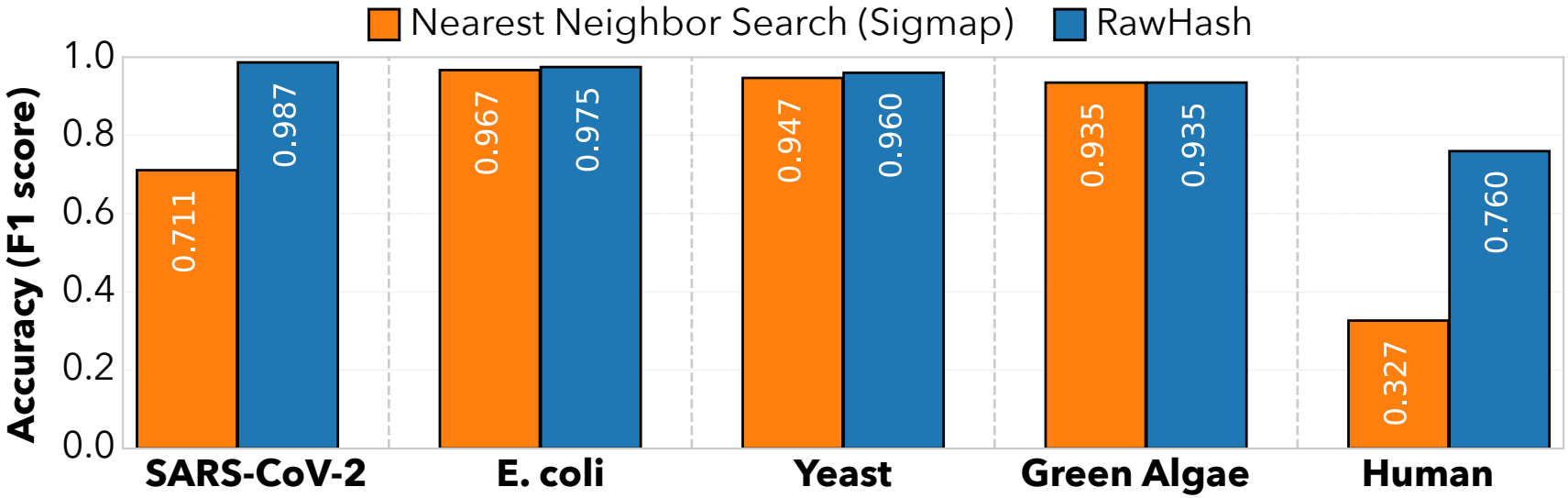
Key Contribution: The first hash-based search (and indexing) mechanism for raw signals

Real-Time Analysis Benefits from Fast Search



Real-time Analysis for Human Genomes:
Fast search instead of costly neighbor search

Fast Search Leads to Accurate Search



Best accuracy for all genomes:
Hash values span longer sequences than vectors ->
More unique and informative matches

(Almost) Identical accuracy to basecalled analysis:
Is the small difference because RawHash is even better?

RawHash is Open Source

RawHash Public

Edit Pins | Unwatch 8 | Fork 5 | Starred 51

main | 9 Branches | 3 Tags | Go to file t | Add file | Code

canfirtina updating README — d3956ea · 3 months ago — 46 Commits

extern	Rawsamble: overlapping signals and some fixes in RawHa...	8 months ago
gitfigures	rawsamble info in README	3 months ago
src	Experimental options	3 months ago
test	Rawsamble new preset parameters	4 months ago
.gitignore	Rawsamble new preset parameters	4 months ago
.gitmodules	ZSTD sobmodule for POD5	2 years ago
LICENSE	R10 k-mer models can be parsed now as well.	2 years ago
Makefile	A minor fix in Makefile	2 years ago
README.md	updating README	3 months ago
code_of_conduct.md	Moving to multiple headers than a single one to improve a...	2 years ago

README | Code of conduct | GPL-3.0 license

About

RawHash can accurately and efficiently map raw nanopore signals to reference genomes of varying sizes (e.g., from viral to a human genomes) in real-time without basecalling. Described by Firtina et al. (published at https://academic.oup.com/bioinformatics/article/39/Supplement_1/i297/7210440).

academic.oup.com/bioinformatics/arti...

bioinformatics | nanopore | seeding | segmentation | event-detection | genome-analysis | hash-tables | contamination | read-mapping | relative-abundances | nanopore-sequencing | nanopore-analysis-pipeline | nanopore-reads | nanopore-data | nanopore-minion | raw-signal | rawhash | raw-nanopore-signal-analysis

Readme
GPL-3.0 license
Code of conduct
Activity
Custom properties
51 stars
8 watching

RawHash's Impact

RawHash

Several works building on RawHash:

Alignment without basecalling [IEEE Access'24]

Assembly without basecalling [arXiv'24]

In-storage processing [Accepted to ICS'25]

In-memory processing [Unpublished work]

Collaboration with Industry to Accelerate Genome Analysis

AMD

[Ongoing]

Collaboration with Medical Centers to Accelerate Sequence-to-Answer

Universitätsspital Basel

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1. **RawHash (ISMB/ECCB'23)**
   Enabling real-time and accurate read mapping without basecalling
2. **Rawsamble (arXiv'24 & HiTSeq'24)**
   Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

3. **ApHMM (TACO'24 & HiPEAC'24)**
   Accelerating ML operations in Genome Graphs

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1. **RawHash (ISMB/ECCB'23)**
   Enabling real-time and accurate read mapping without basecalling
2. **Rawsamble (arXiv'24 & HiTSeq'24)**
   Enabling new applications by assembling raw sequencing data

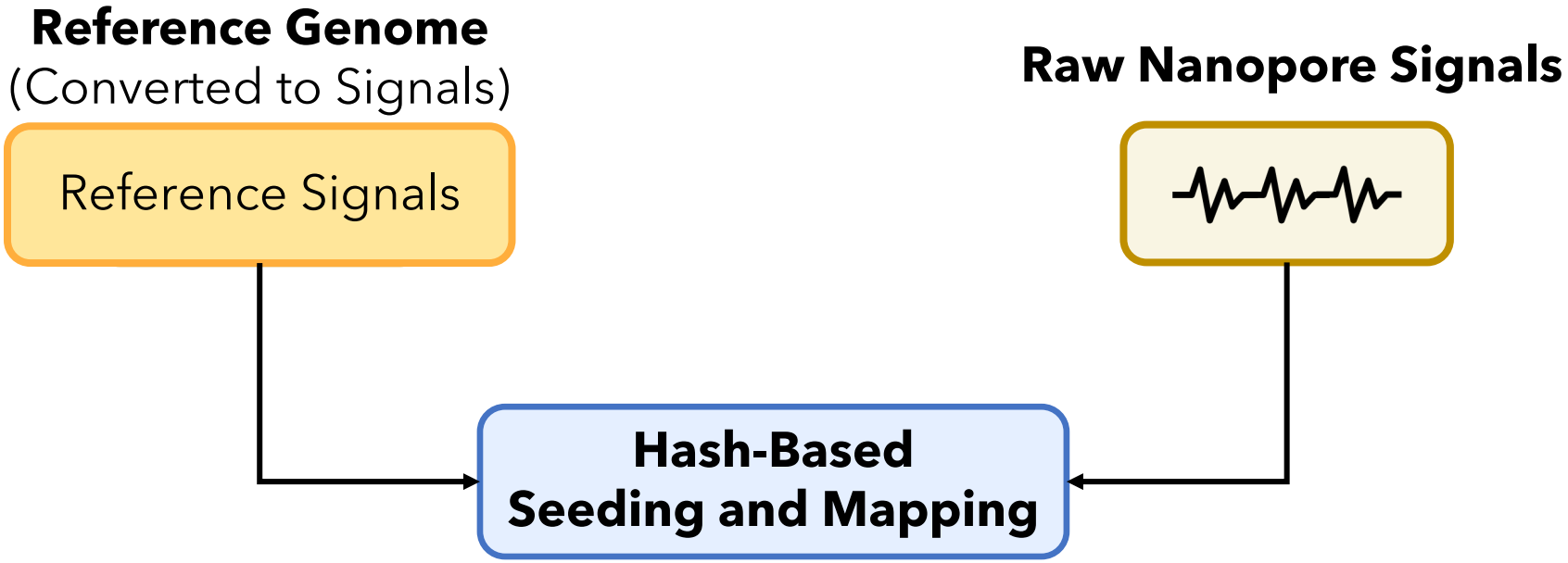
Algorithm & Hardware Design for AI in Genome Analysis

3. **ApHMM (TACO'24 & HiPEAC'24)**
   Accelerating ML operations in Genome Graphs

Pushing the Boundaries of Application Scope

via New Algorithms

Beyond Reference Mapping: Overlapping

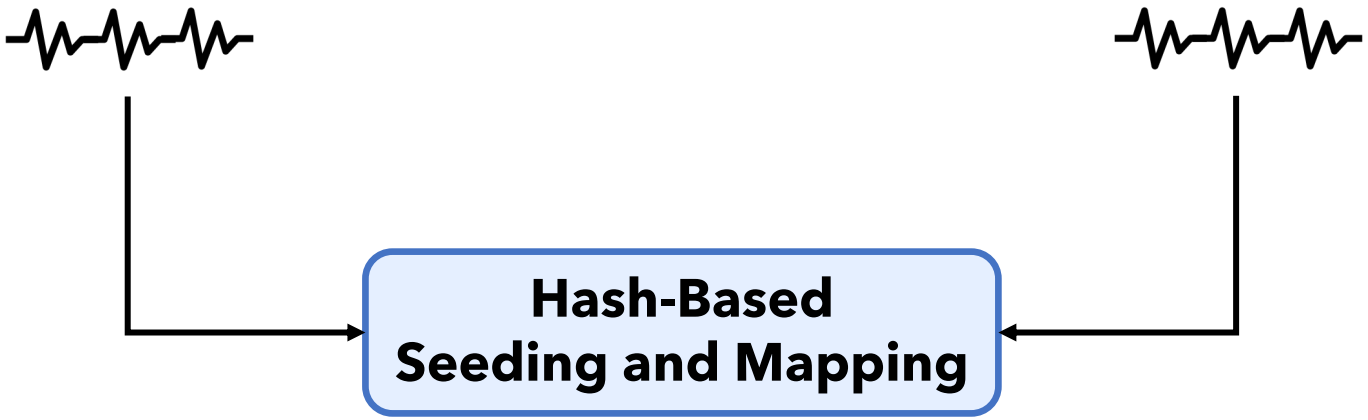


Challenge: Reference genome is not always available

Assembly: Constructing genome from **overlapping reads**

Existing solutions cannot find overlapping reads **without basecalling**

Challenges with Overlapping Raw Signals



Challenge: Identifying hash matches **when both signals are noisy**

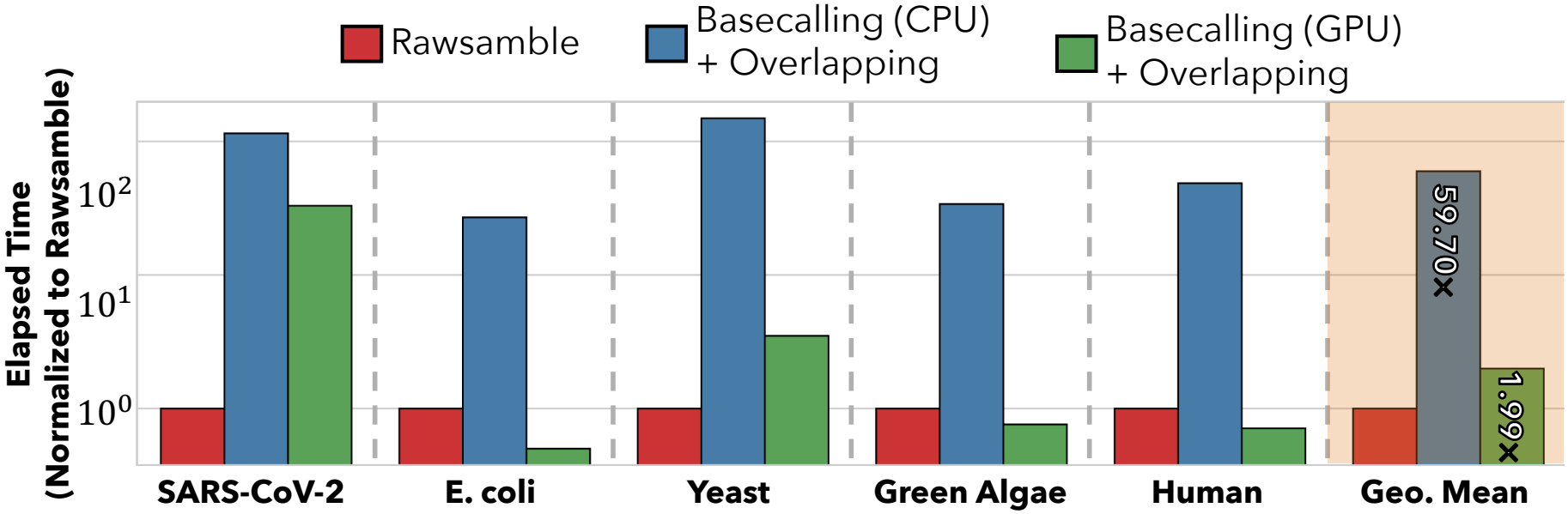
Challenge: Finding **many** useful overlapping pairs (all-vs-all overlapping)

Challenge: Generating **long paths** from useful overlaps

Rawsamble

The first mechanism that can **perform all-vs-all overlapping from raw signals**

Eliminating Basecalling from the Pipeline

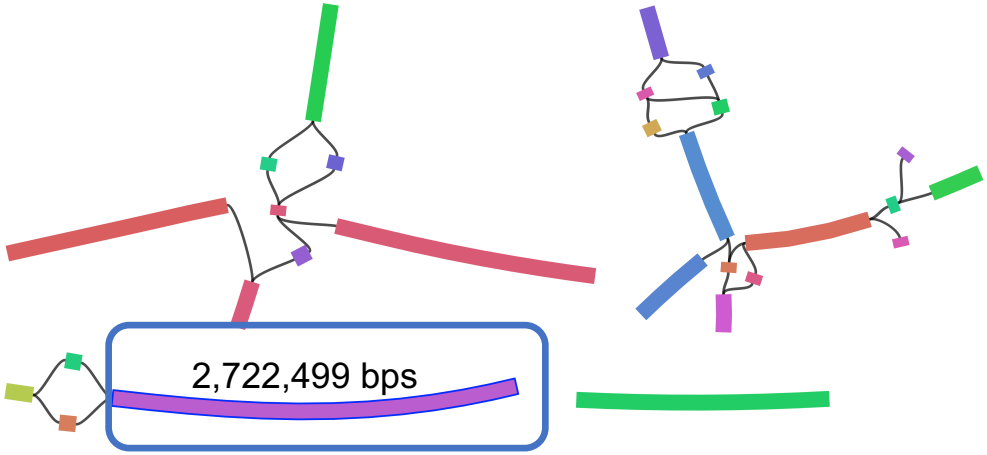


Average speedup of **60**× than CPU-based basecalling:
Eliminating basecalling leaves a significant room
for performance improvements

Average speedup of **2**× than the **GPU-based basecalling**:
GPU vs. GPU comparison is likely to provide even better results

Key Results – First Ever Assemblies

E. coli Assembly
(From the Rawsamble Overlaps)



First assemblies ever constructed without basecalling

~400× longer assembled pieces than the average read length: Informative overlaps lead to longer paths

It is time to rethink if we want to translate individual reads separately

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1. **RawHash (ISMB/ECCB'23)**
   Enabling real-time and accurate read mapping without basecalling
2. **Rawsamble (arXiv'24 & HiTSeq'24)**
   Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

3. **ApHMM (TACO'24 & HiPEAC'24)**
   Accelerating ML operations in Genome Graphs

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1. RawHash (ISMB/ECCB'23)
   Enabling real-time and accurate read mapping without basecalling
2. Rawsamble (arXiv'24 & HiTSeq'24)
   Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

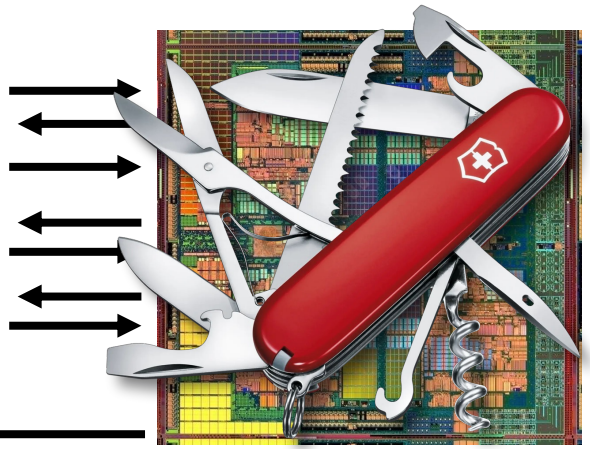
3. **ApHMM (TACO'24 & HiPEAC'24)**
   Accelerating ML operations in Genome Graphs

Algorithm Design Demands General-Purpose Computing Capabilities

Algorithm Design Demand General-Purpose Computing Capabilities

Your Favorite Algorithm

Function 1
Function 2
Function 3



Your Favorite Output

General-Purpose Systems (e.g., CPUs)

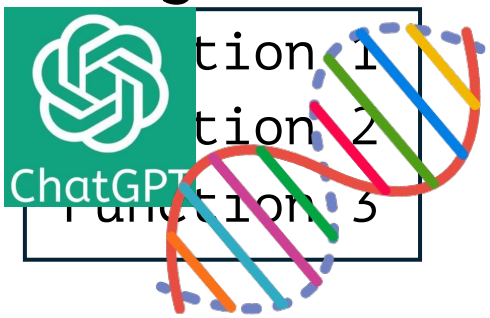
Paid Costs for Generality:

Supporting unnecessary operations
Inefficient data and control flow
Limited parallelism & bandwidth

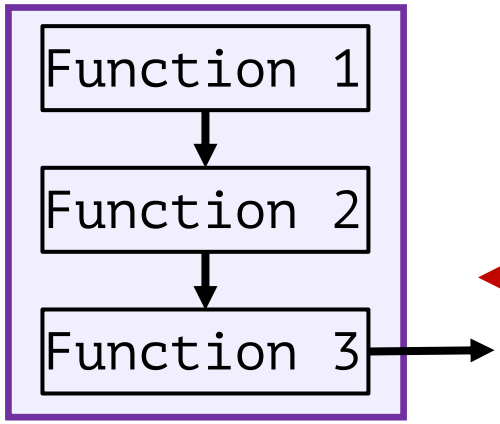
→ **High latency & Substantial energy**

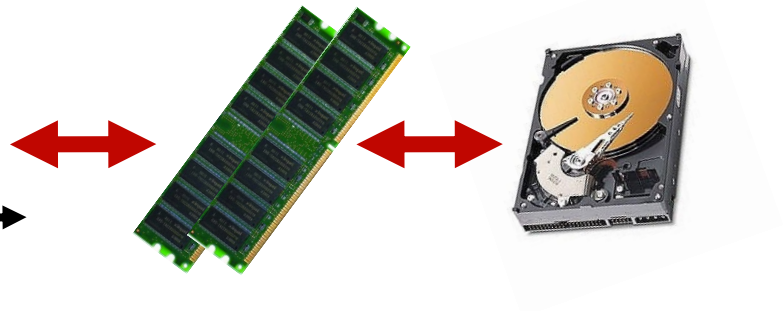
Great Algorithms Demand Specialization

Everyone's Favorite Algorithm



Application-Specific Architecture





Goal: Avoid wasting computation & energy

Opportunities:

Better data flow with minimal latency & power

Efficient processors customized for the task

Better parallelism

Paid Costs for Specialization:

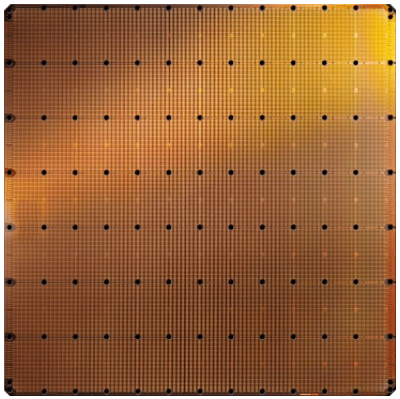
Poor programmability

High manufacturing and engineering costs

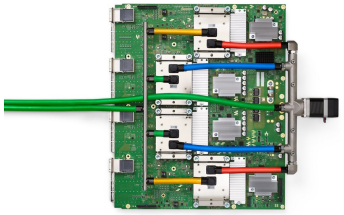
Acceleration Efforts for AI and Genomics

Cerebras WSE-3

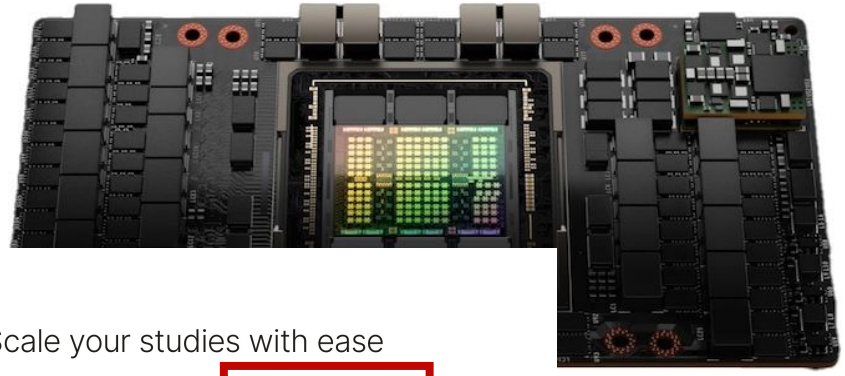
The largest ML accelerator chip



Google's TPUs:



NVIDIA H100



Scale your studies with ease

Process high-throughput data quickly with hardware acceleration

Process high-throughput data quickly with hardware acceleration. With four field-programmable gate arrays (FPGAs) onboard you have the most powerful DRAGEN analysis ever, enabling you to process NovaSeq X System data easily. Perform up to four simultaneous applications per flow cell in a single run.

Reduce data footprint, manage and store data easily with lower costs and lower energy consumption, with built-in compression that reduces FASTQ file sizes by up to 80%.2

Stream data directly to Illumina Connected Analytics or BaseSpace Sequence Hub on the cloud for scalable data management, analysis, and aggregation.

Absci and AMD Announce Collaboration and Strategic Investment to Accelerate the Future of AI Drug Discovery

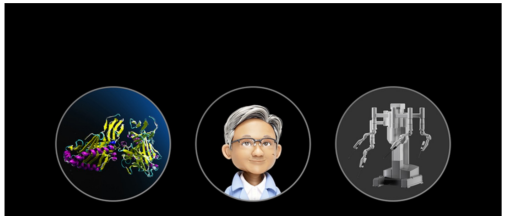
01/08/2025

PDF Version

NVIDIA Partners With Industry Leaders to Advance Genomics, Drug Discovery and Healthcare

IQVIA, Illumina, Mayo Clinic and Arc Institute Harness NVIDIA AI and Accelerated Computing to Transform $10 Trillion Healthcare and Life Sciences Industry

January 13, 2025



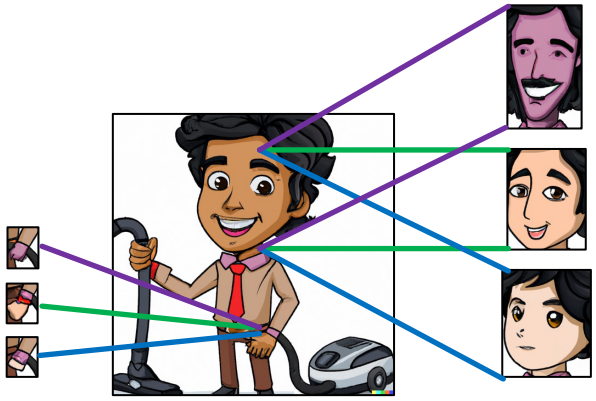
Important and Commonly Used Applications Demand Specialization

Graphs in Genomics

Graphs are commonly used
to identify variations between sequences

To avoid **redundant** comparisons and storage

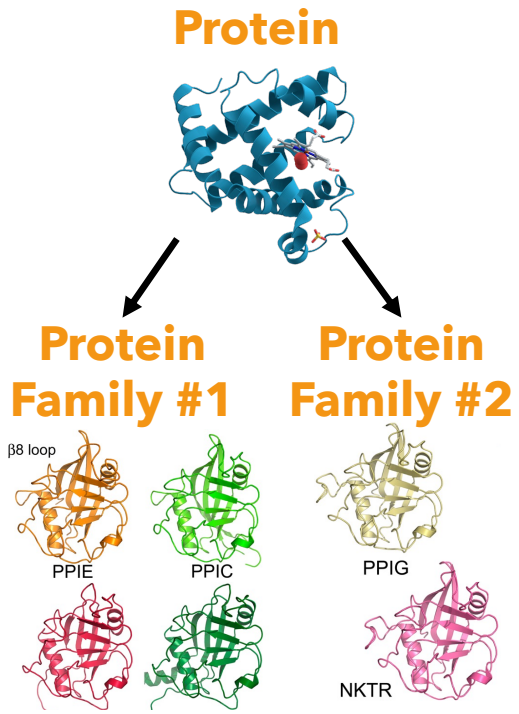
To provide **rich information** on expected variations



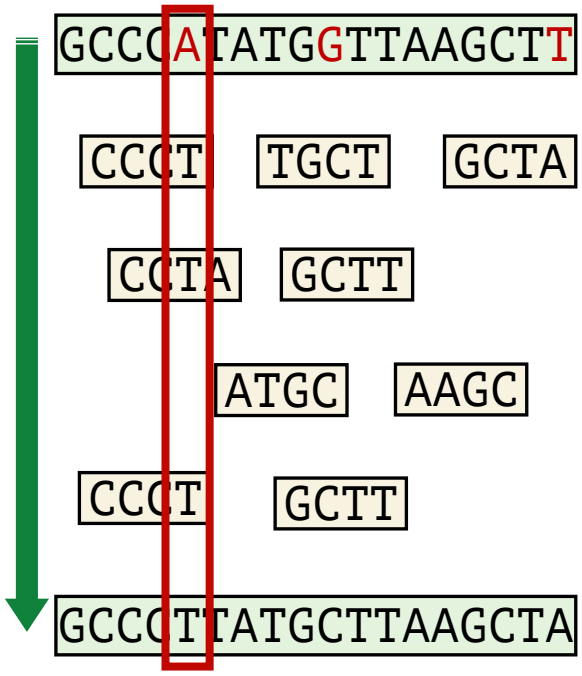
Reference Genome
Graph

Graph-Based Applications in Genomics

Protein Family Search with Graph Alignment



Error Correction with Probabilistic Graphs

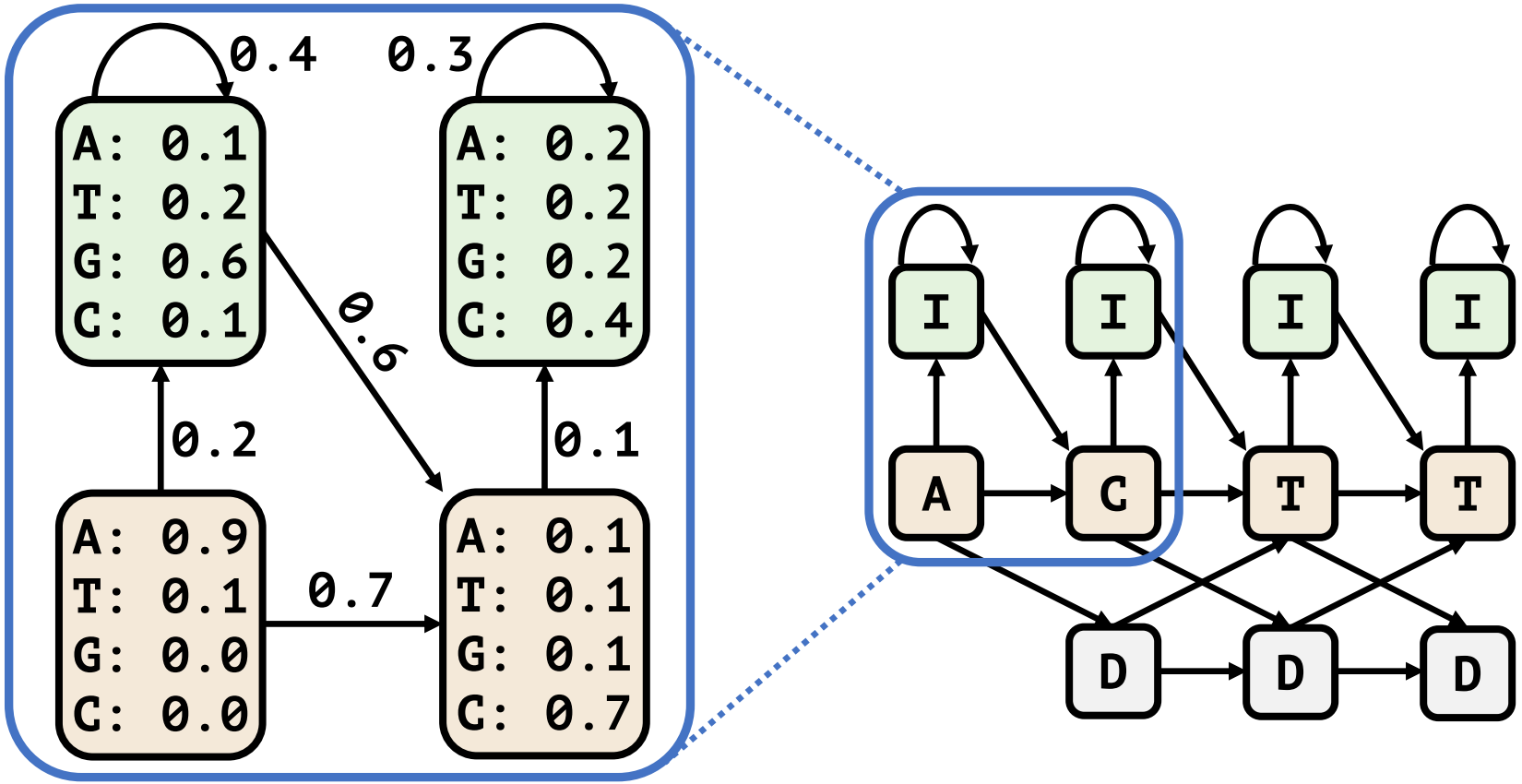


Multiple Sequence Alignment

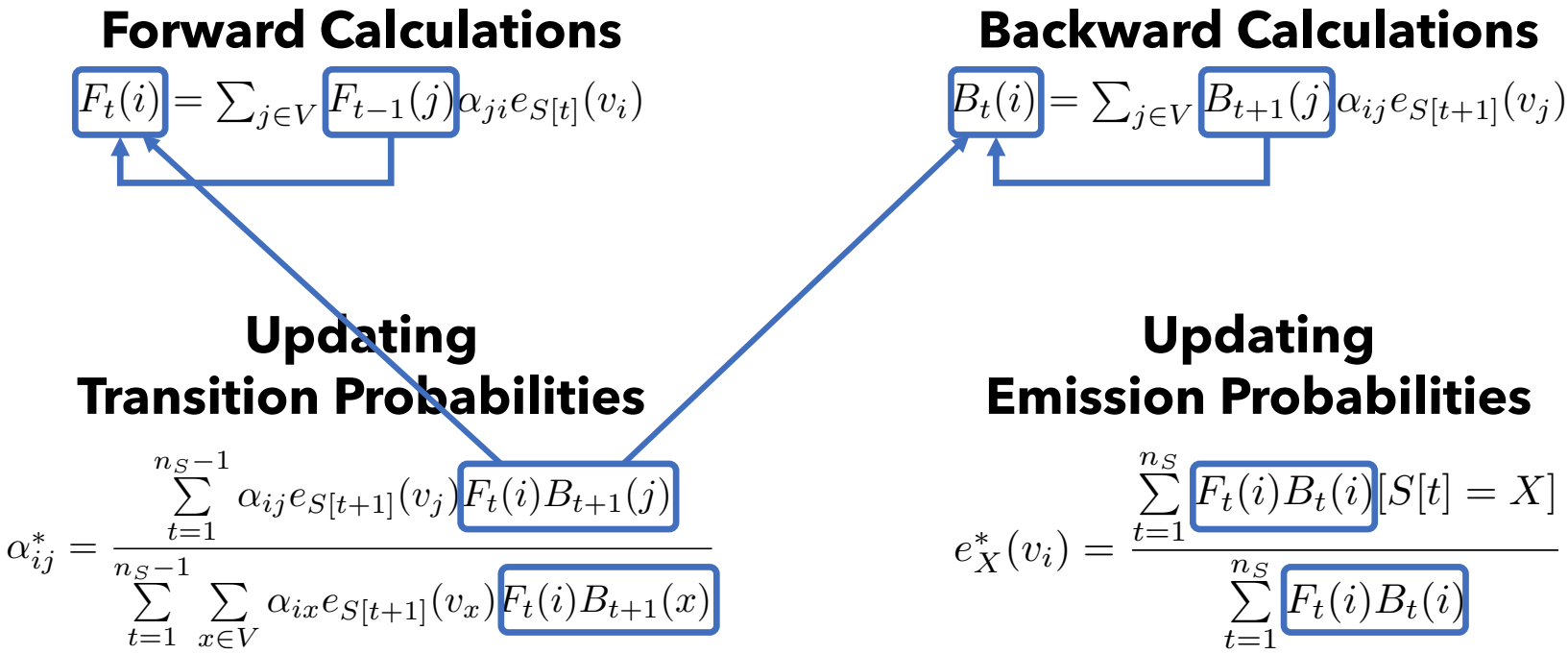
```
GCCC-TATGGTTAAGCTT
GCCCATATGATTAAGCTT
GCCCATATGGTTAAGCTT
GCCCGTATGGTT--GCTT
GCCCATATGCTTAAGCTT
GCCC---TGGTTAAGCT-
GCCCATATCCTTAAGCTT
GCCCATATGGTTAAGCTT
```

A Common Graph Structure in Genomics

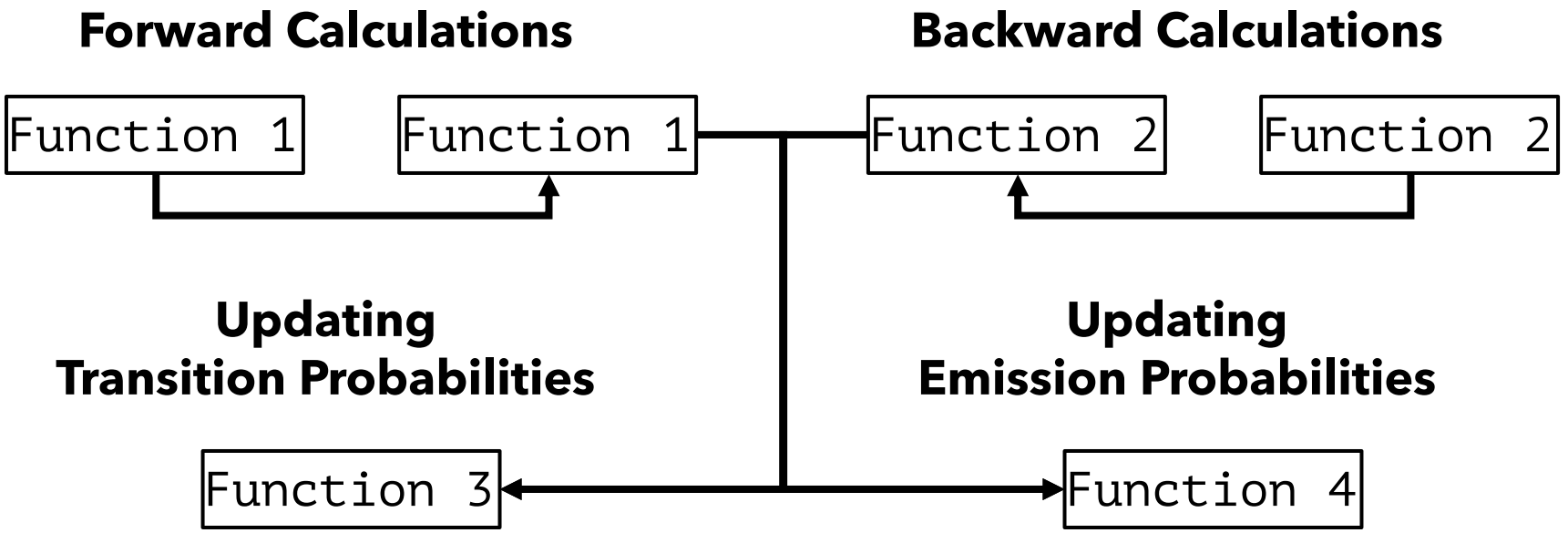
Profile Hidden Markov Models (PHMMs): A probabilistic graph structure



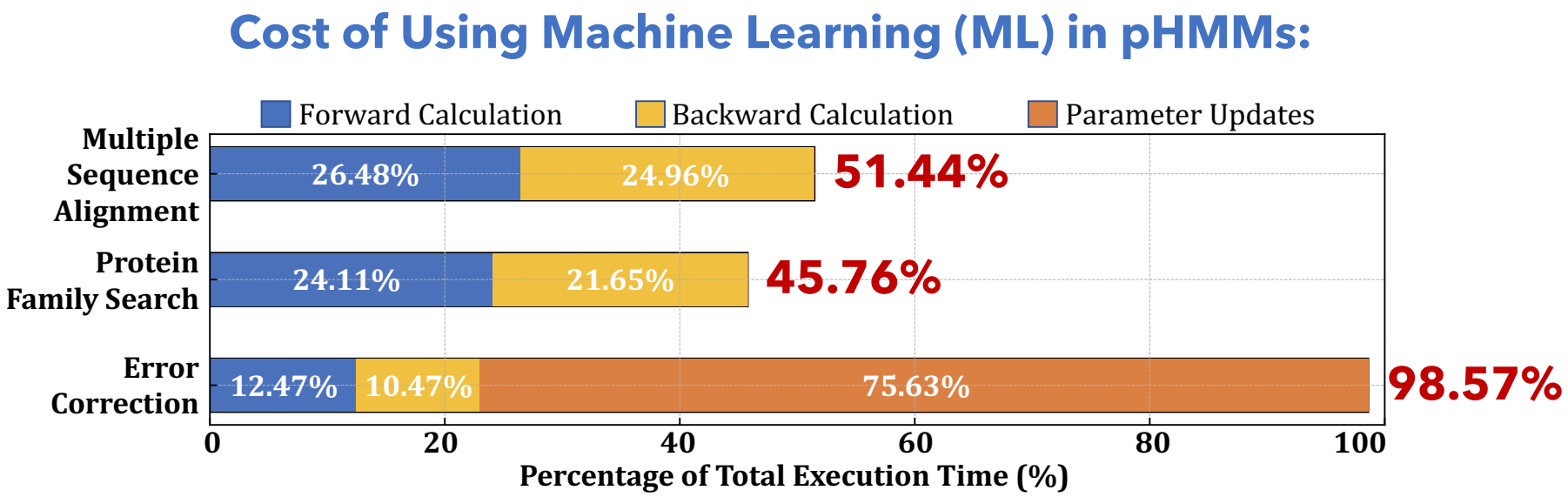
Underlying ML Technique in Genomic Graphs



Underlying ML Technique in Genomic Graphs



Costly Use of ML in Genomic Graphs



Goal

Enable **rapid and power-efficient**
use of the underlying ML technique in genome graphs

Key Approach: HW-SW Co-Design

- **Reduce redundant data storage** by utilizing the fixed data pattern
- **Reduce unnecessary computations** with quick filtering

SW

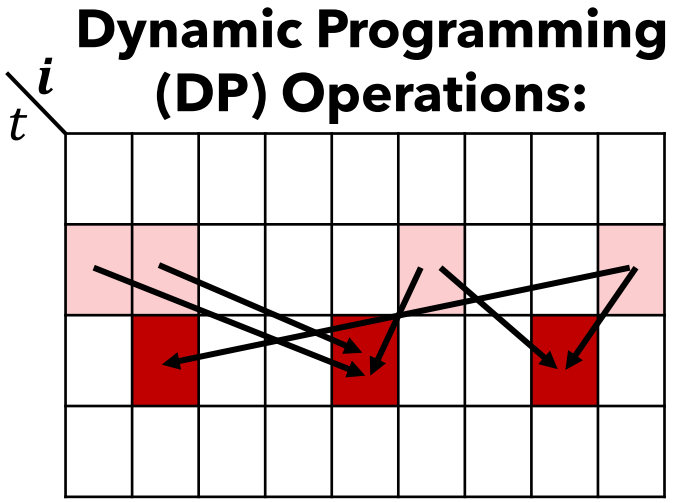
- **Avoid repeated operations** by utilizing lookup tables

- **Reduce data movement** by processing data where it makes sense

HW

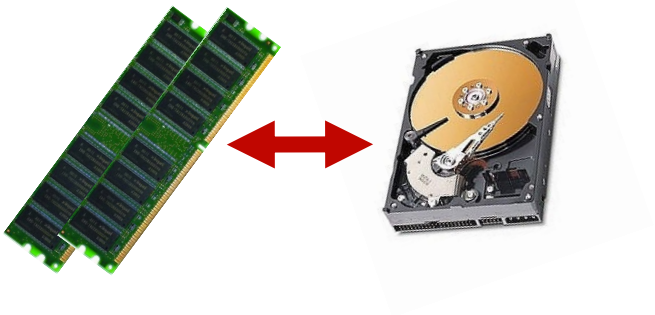
- **Flexible and efficient** hardware design

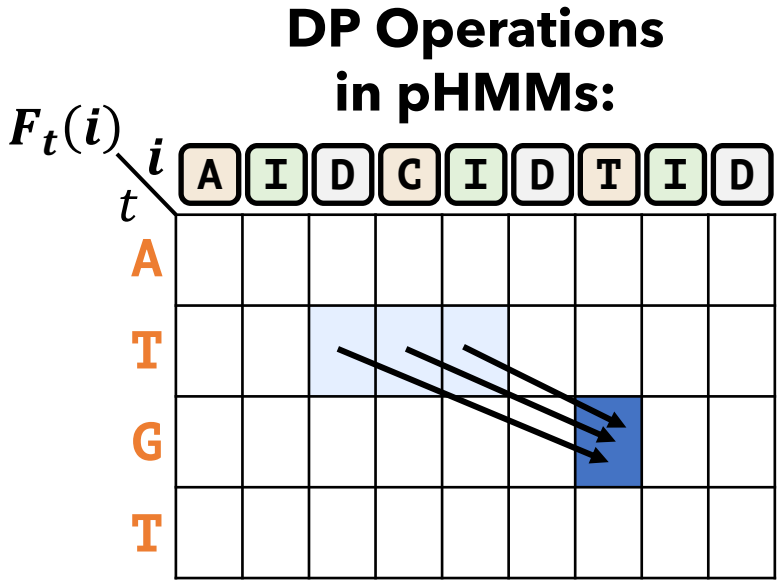
HW: Simple Data Pattern in DP Calculations



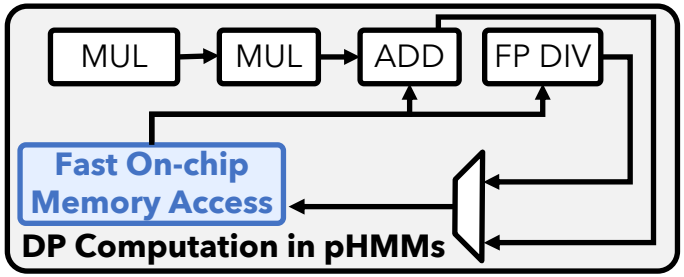
Many Different Patterns

Frequent off-chip access





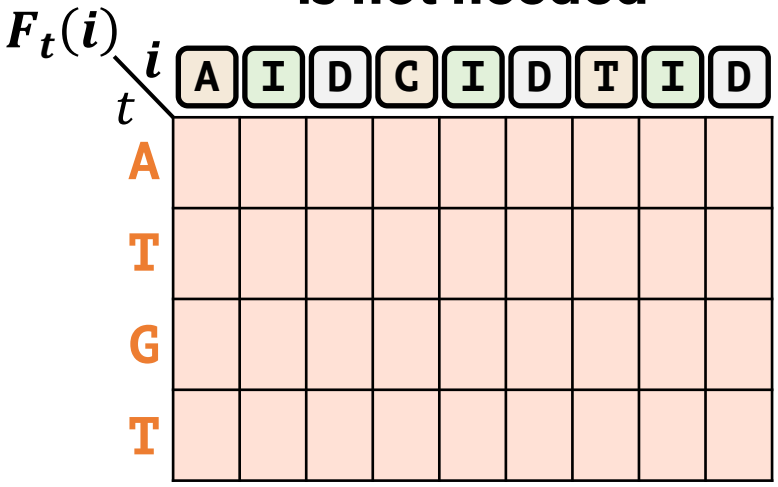
Simple Pattern



Store and access data where it makes sense

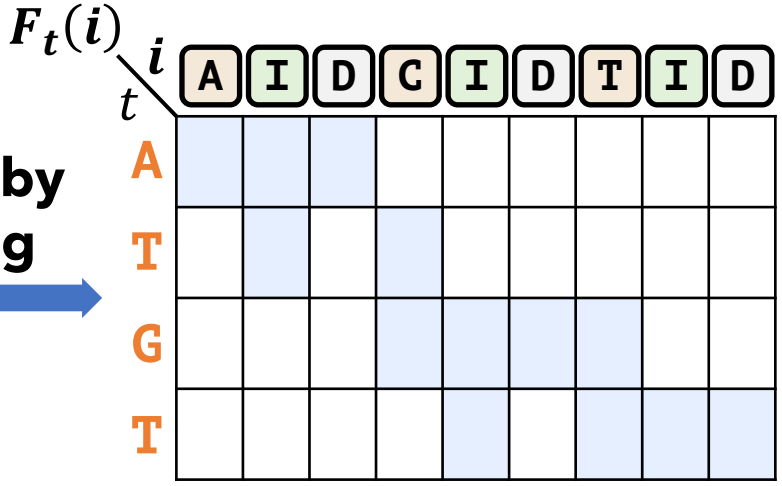
SW: Reducing Unnecessary Computations

Observation:
Calculating the entire DP is not needed



Dense DP Calculations

Filter by sorting



Sparse (Filtered) DP Calculations

Challenge:
Sorting is costly in hardware

SW: Reducing Unnecessary Computations

Software co-design:
Replace sorting with a cheaper filtering

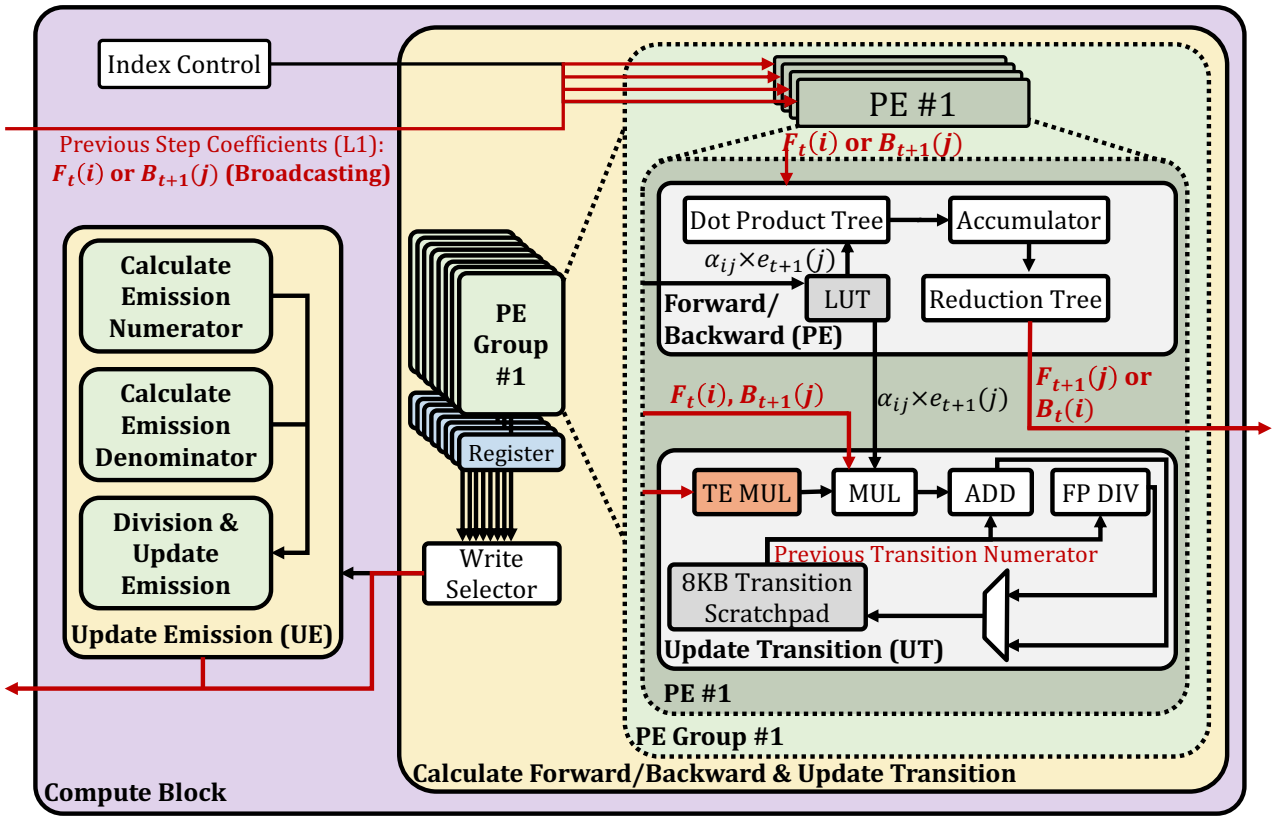
State IDs	Range
8, 9	1.00 – 0.94
10, 14	0.94 – 0.88
15, 16, 18	0.88 – 0.82
11, 20, 21, …	0.82 – 0.76
13, 17, 19, …	0.76 – 0.70
⋮	⋮
	0.06 – 0.00

Histogram Filter

Putting the Optimizations Together

Effectively implementing the hardware and software optimizations

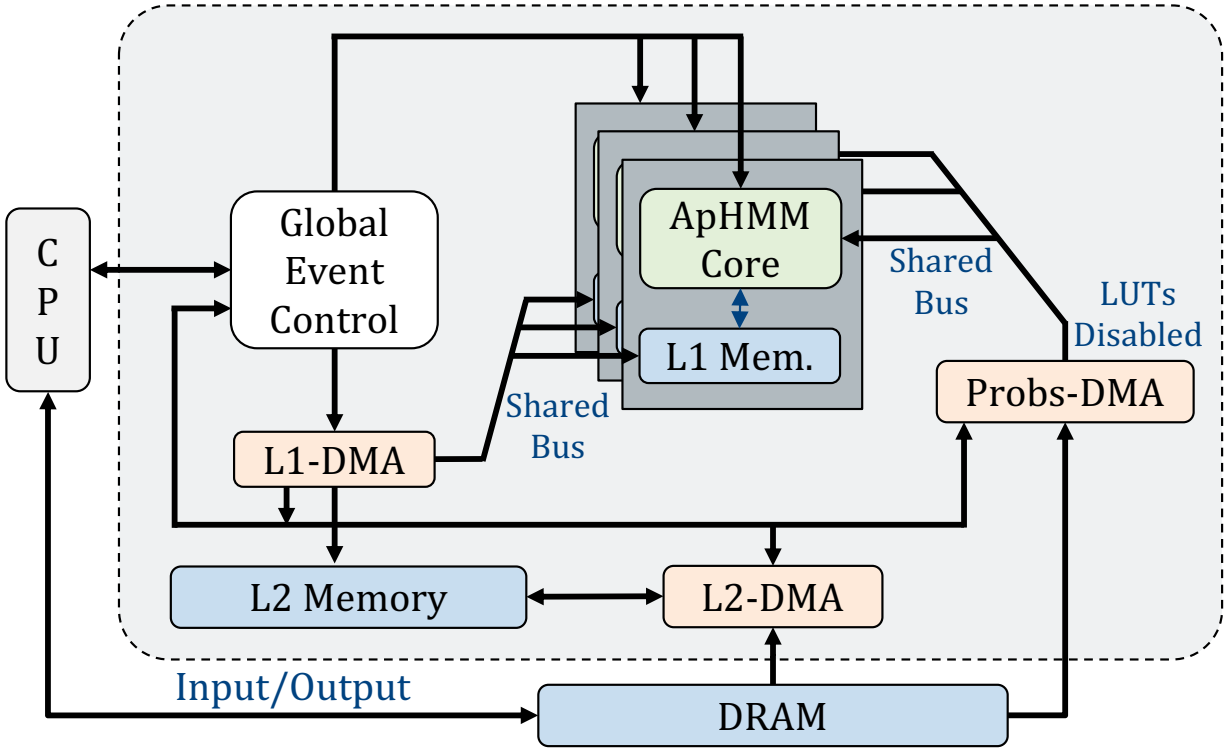
to improve performance and reduce power requirements



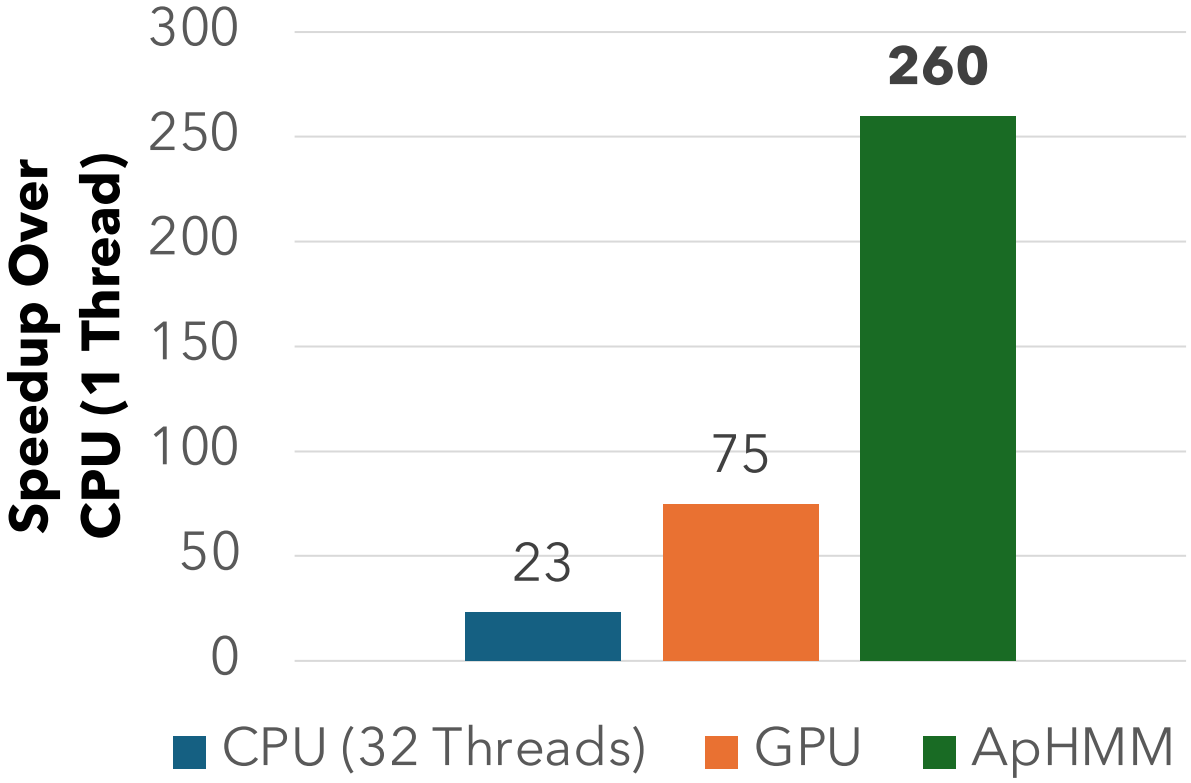
Integrating ApHMM in a Complete System

The specialized task can be **offloaded**

to an accelerator **whenever needed**

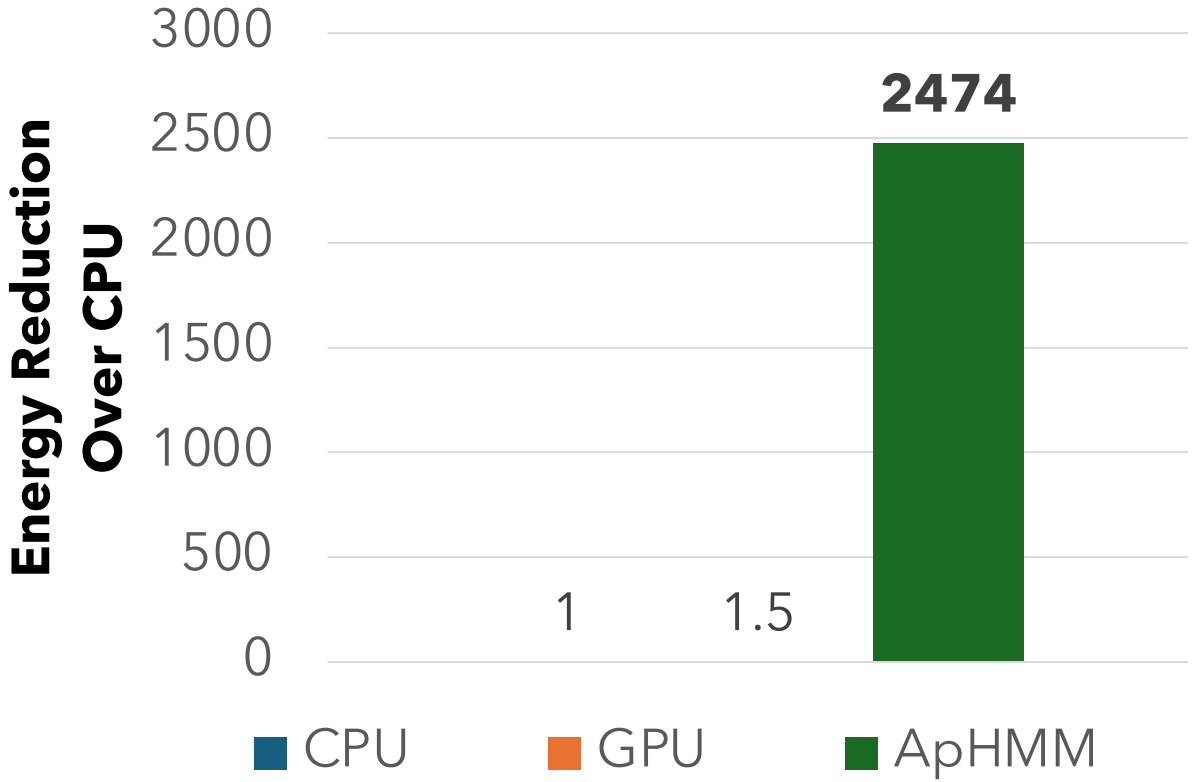


Key Results: Performance



Specialized design enables computation
up to two orders of magnitude faster:
Better data flow with reduced latency and improved parallelism

Key Results: Energy Consumption



…and a significant reduction in energy consumption
by up to three orders of magnitude:
simpler hardware with a minimized off-chip memory accesses and data movement

Specialization Provides Significant Performance and Energy Benefits

Today's Talk

Significant computation and energy overhead

Limited accuracy and application scope

Real-Time and Raw Sequencing Data Analysis

1

RawHash (ISMB/ECCB'23)

Enabling real-time and accurate read mapping without basecalling

2

Rawsamble (arXiv'24 & HiTSeq'24)

Enabling new applications by assembling raw sequencing data

Algorithm & Hardware Design for AI in Genome Analysis

3

ApHMM (TACO'24 & HiPEAC'24)

Accelerating ML operations in Genome Graphs

Acknowledgements

- **Collaborators:**

SAFARI ETH zürich AMD UNIVERSITY OF TORONTO

Carnegie Mellon University

İhsan Doğramacı Bilkent University 1984

intel

TU Delft

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY 1986 포항공과대학교

USC

ILLINOIS

- **Funding agencies and industry sponsors:**
  - BioPIM, SNSF, Intel, Google, Huawei, Microsoft, VMware, and SRC

Fast and Efficient Genome Analysis
via New Algorithms and Architectures

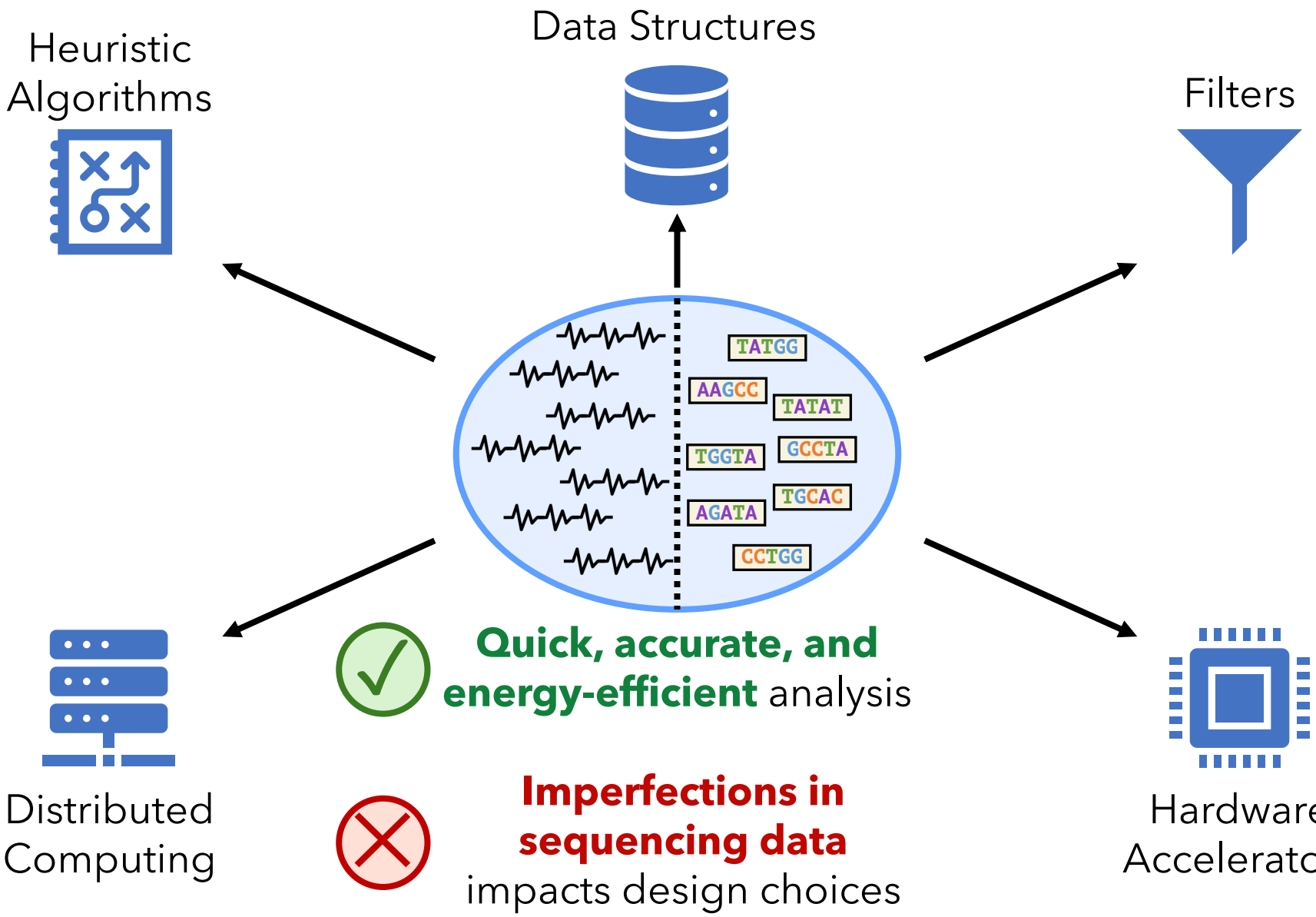
Can Firtina
canfirtina@gmail.com
https://canfirtina.com

6 May 2025
Inria – GenScale & Symbiose

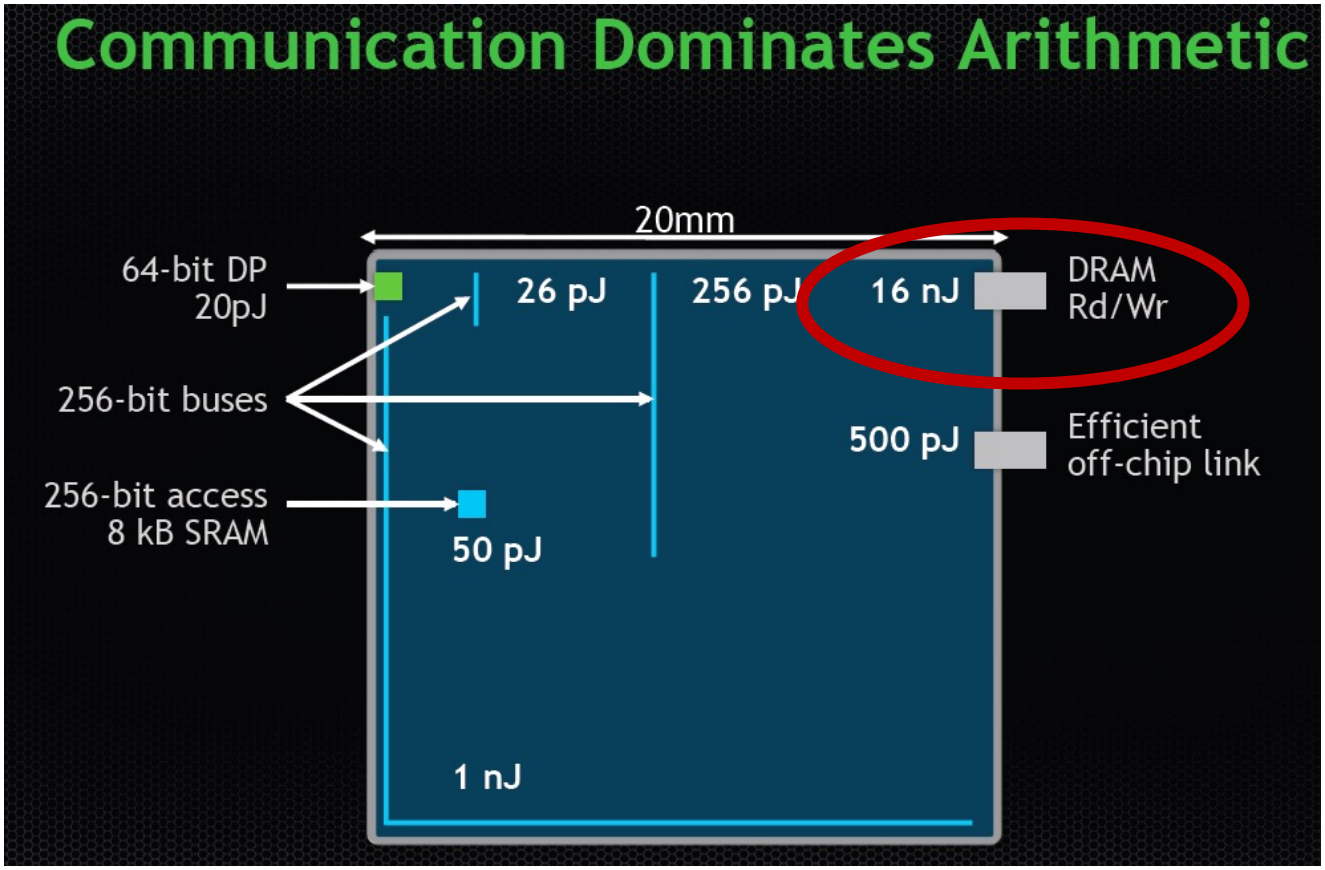
ETH zürich

SAFARI

Sequencing Data Analysis



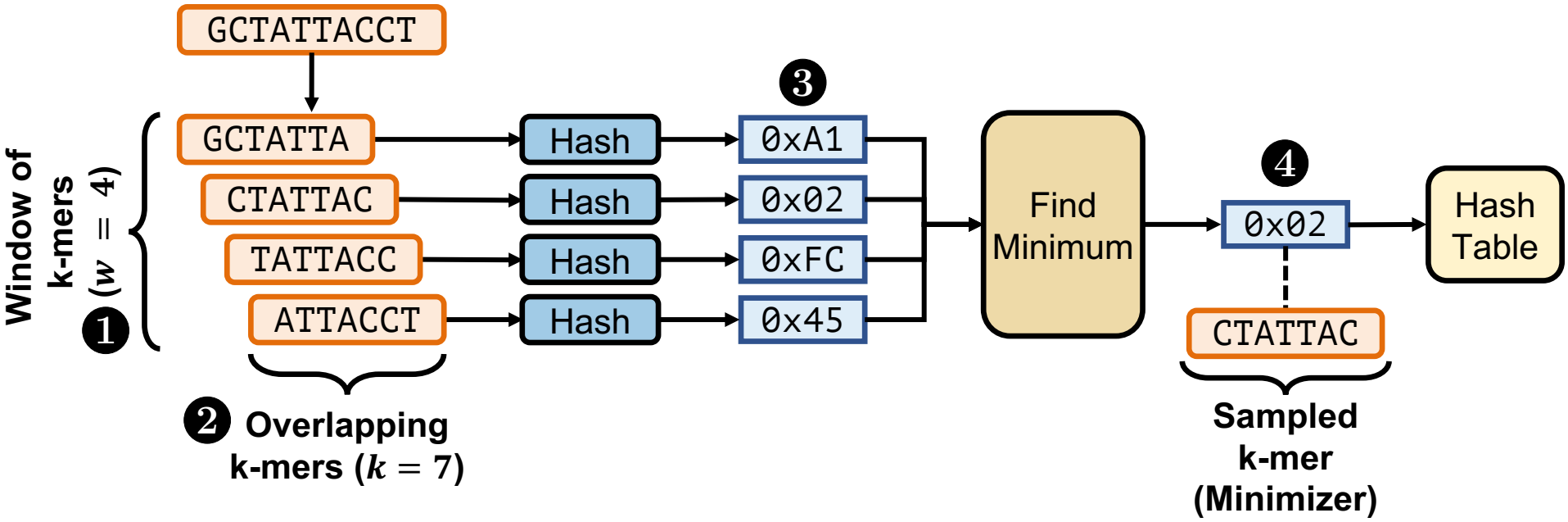
Data Movement Overhead



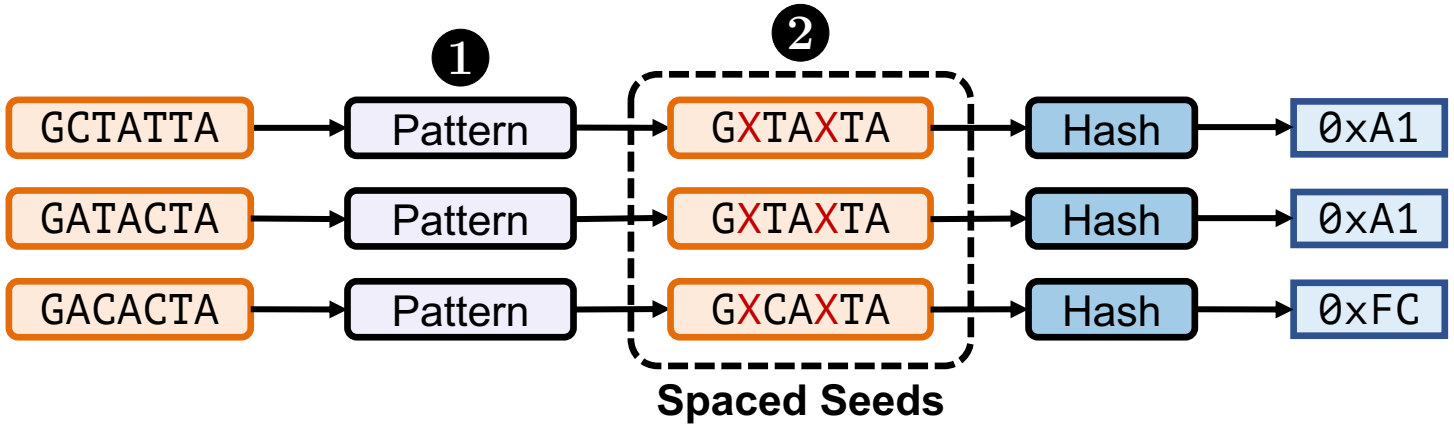
William Dally,
HiPEAC 2015

A memory access consumes ~100-1000X the energy of a complex addition

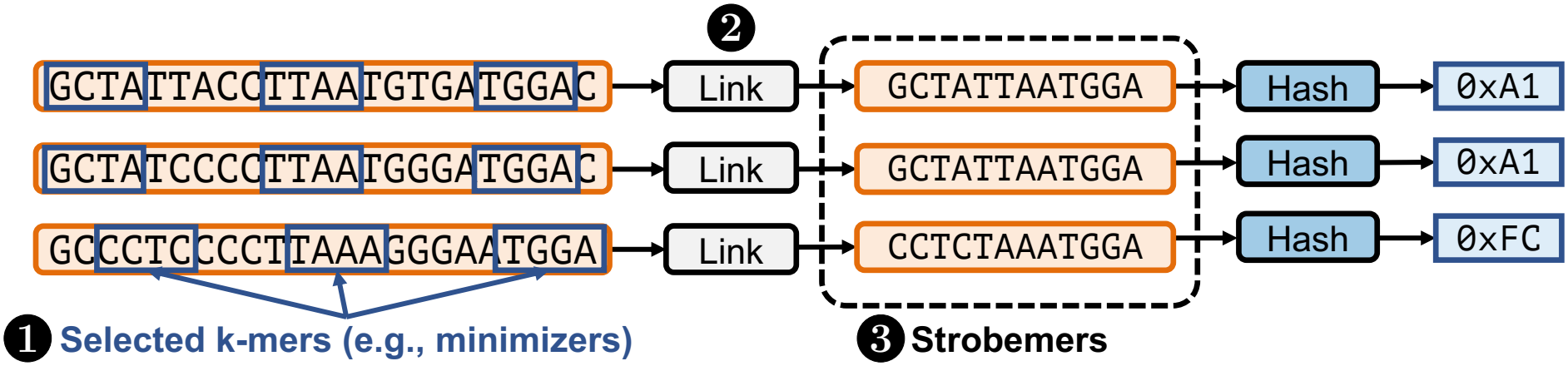
Minimizer Sketching



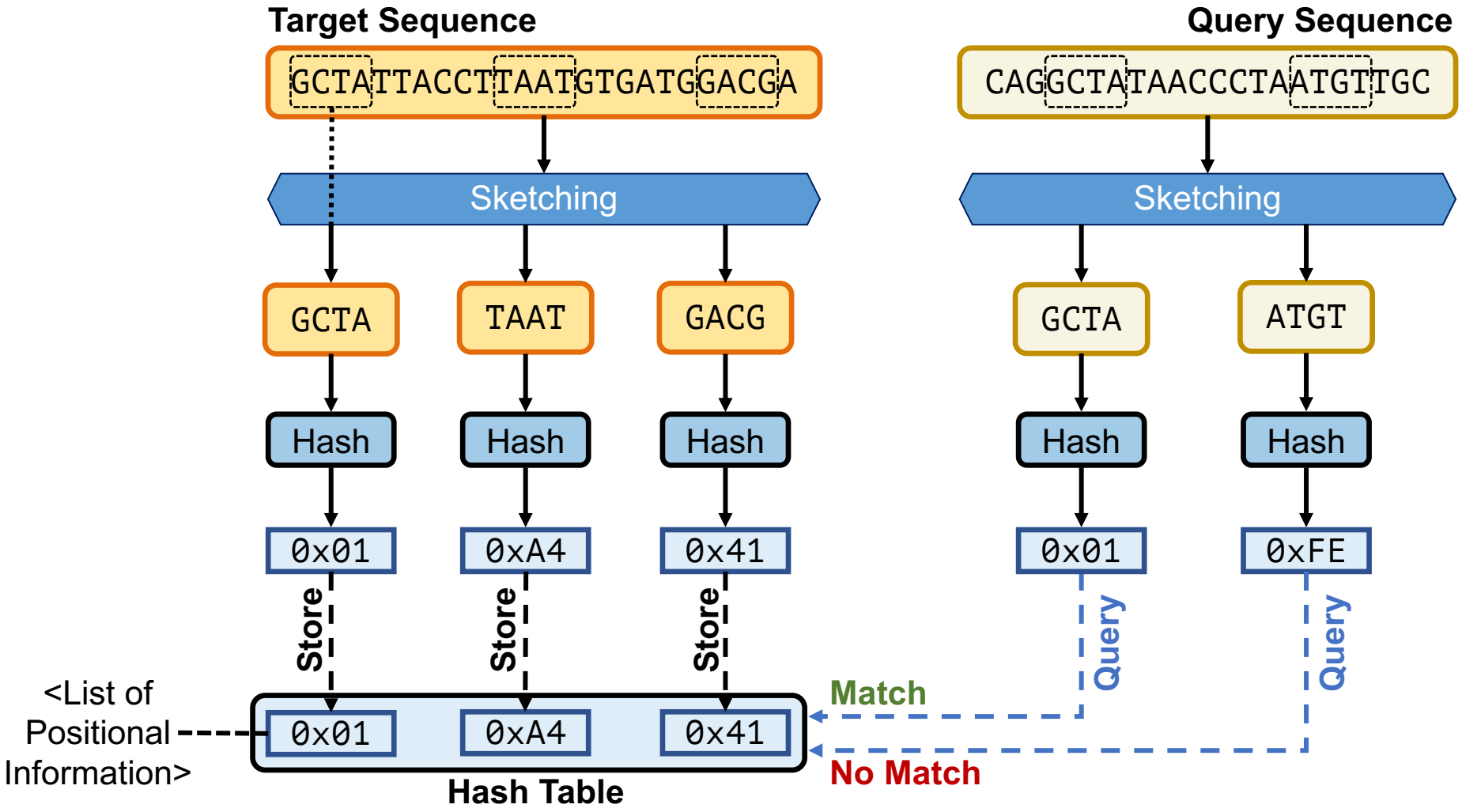
Spaced Seeding



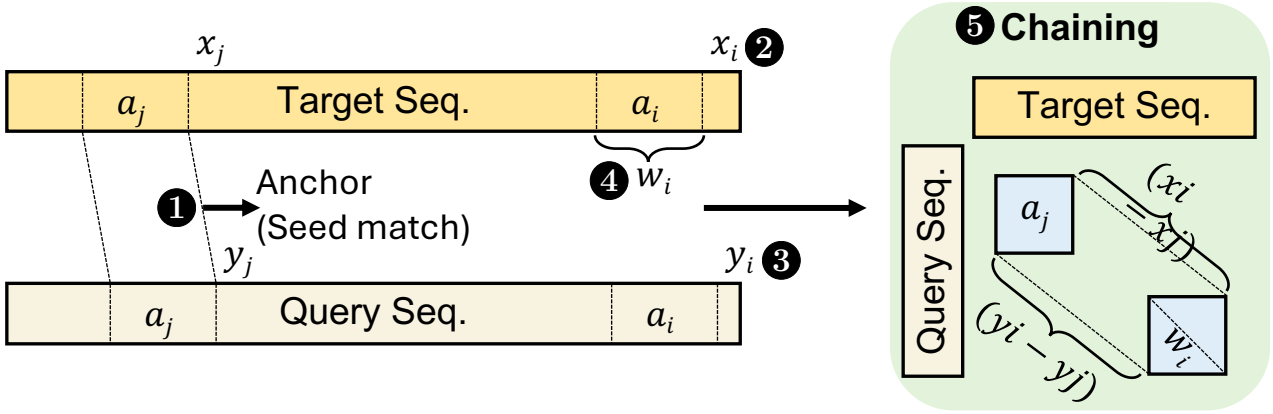
Strobemer Sketches



Hash-Based Sketching and Seed Matching



Chaining (Two Points)

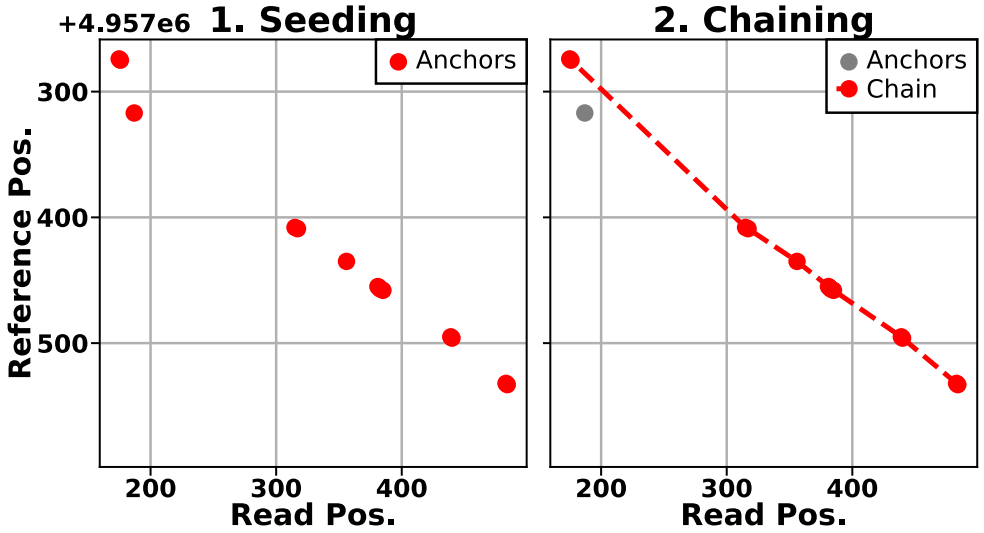


Chaining (Multiple Points)

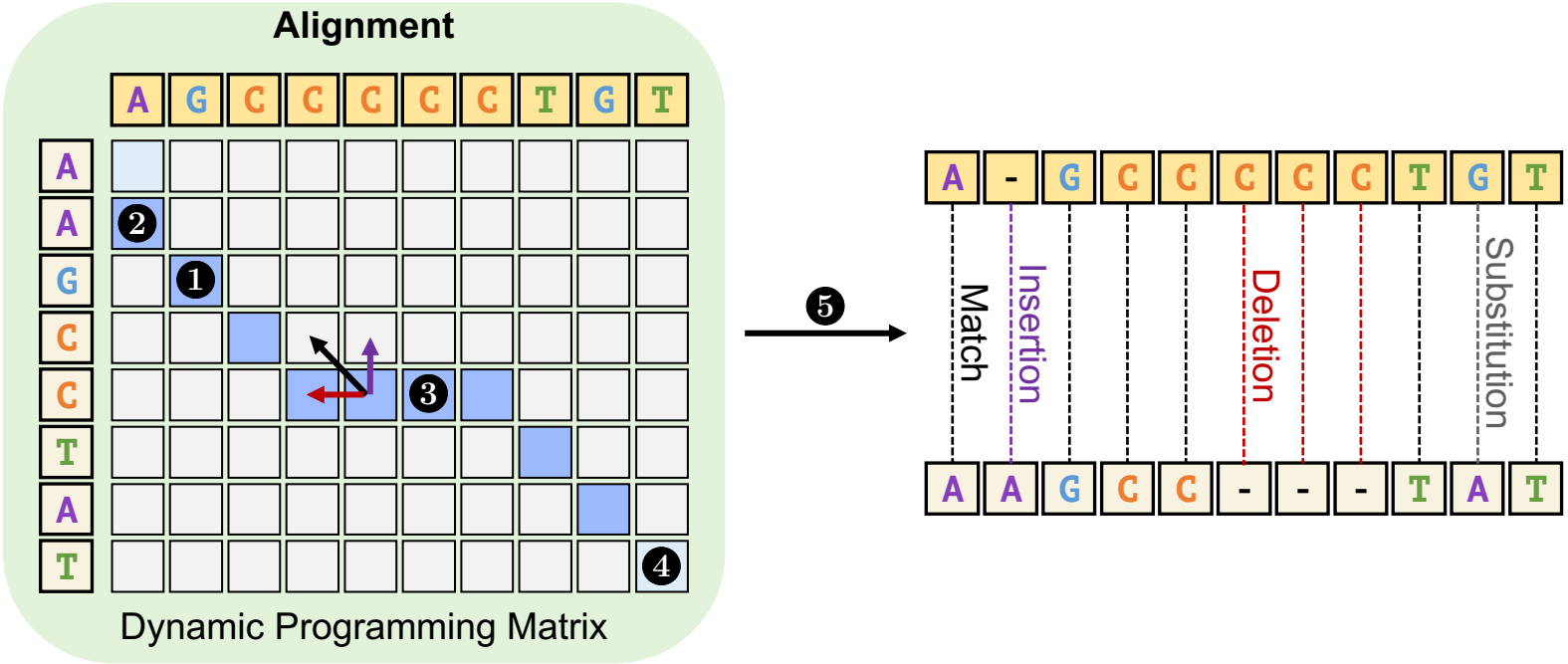
- **Exact hash value matches:** Needed for finding matching regions between a reference genome and a read

- What if there are mutations or errors?
  - **No hash (seed) match** will occur in such positions

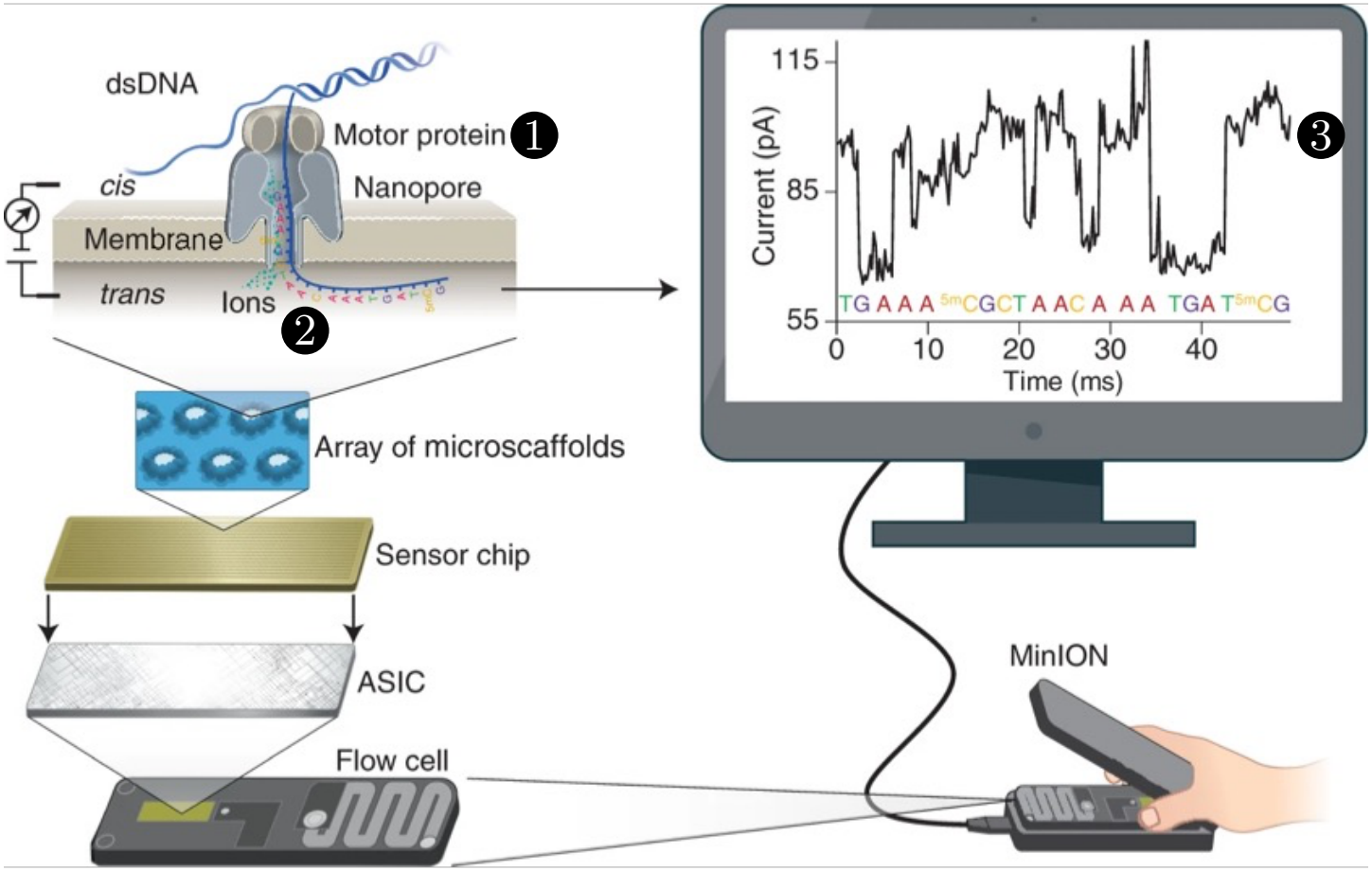
- The chaining algorithm links **exact matches in a proximity** even though there are gaps (no seed matches) between them



Sequence Alignment



Nanopore Sequencing

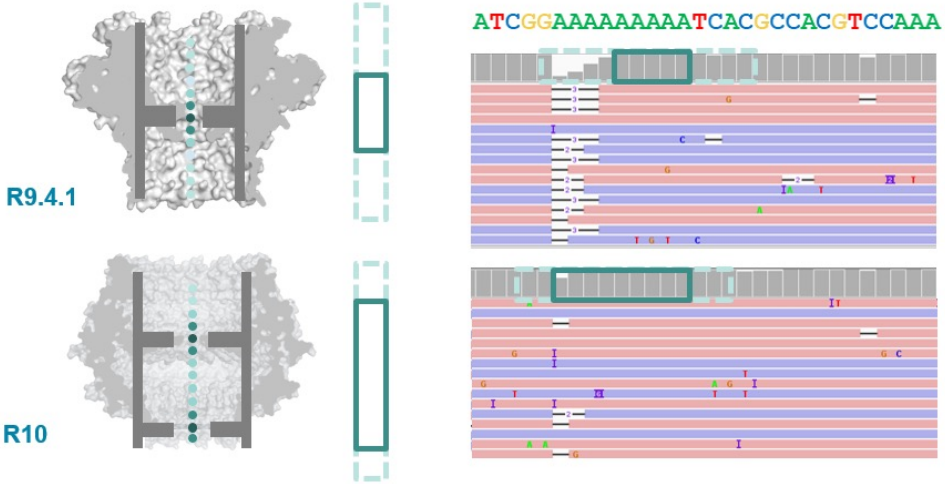


Source of Noise in Nanopore Sequencing

- **Stochastic thermal fluctuations in the ionic current**
  - Random ionic movement due to inherent thermal energy (Brownian motion)
- **Variations in the translocation speed**
  - Mainly due to the motor protein
- **Environmental factors**
  - **Temperature:** Affecting enzymes including the motor protein
  - **pH levels:** Affecting charge and the shape of molecules
- **Maybe: Aging & material-related noise between nanopores**
  - Their effects potentially can be minimized with normalization techniques

R9 vs. R10 Chemistries

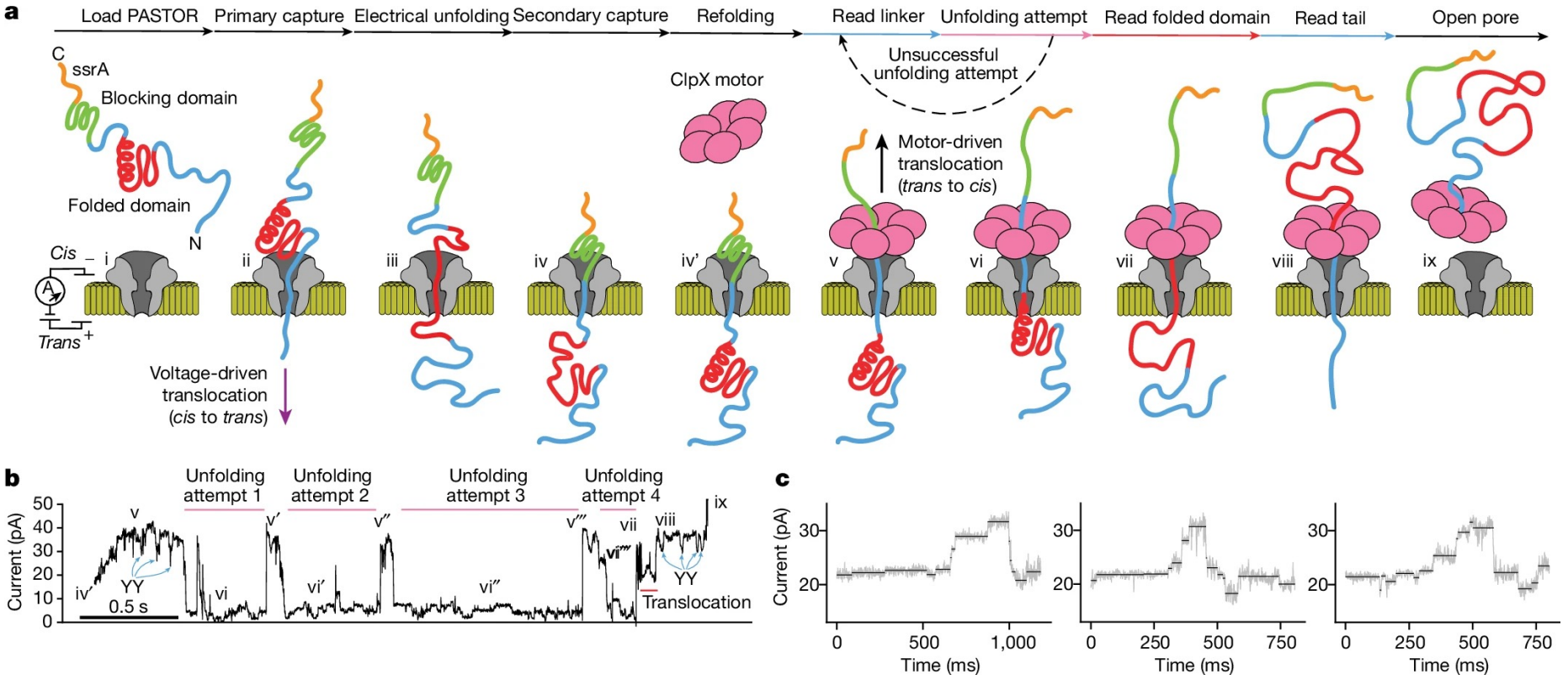
- **Dual reader head**



- **Motor protein** with more consistent translocation speed in R10

- **Duplex sequencing** in R10

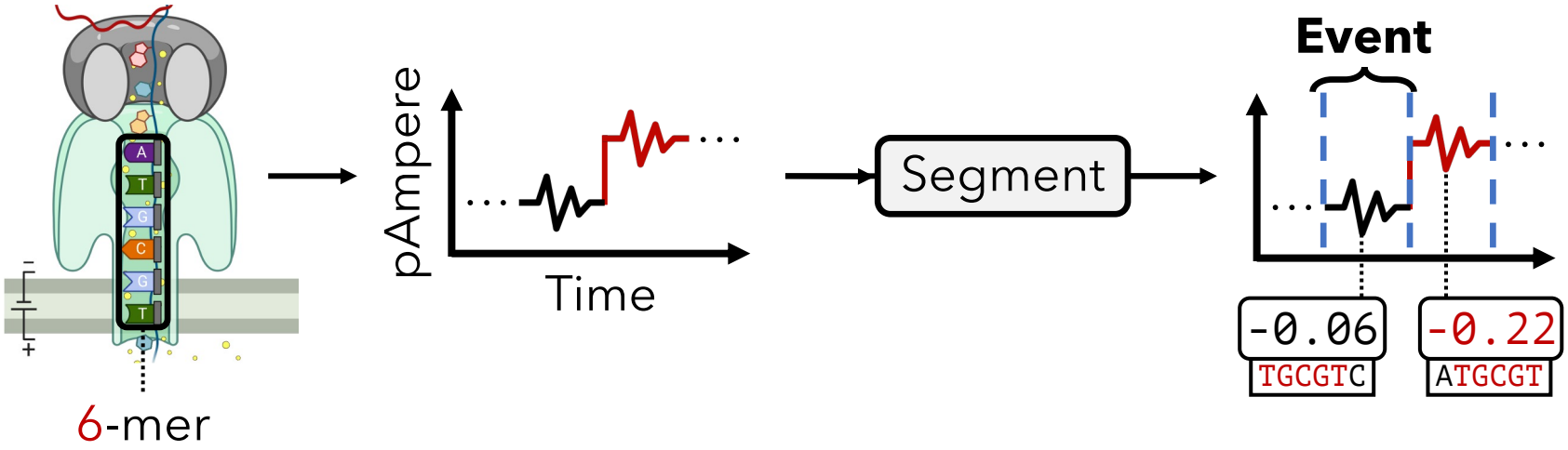
Proteomics with Nanopores



Motone+, "Multi-pass, single-molecule nanopore reading of long protein strands", Nature 2024.

Processing Raw Nanopore Signals

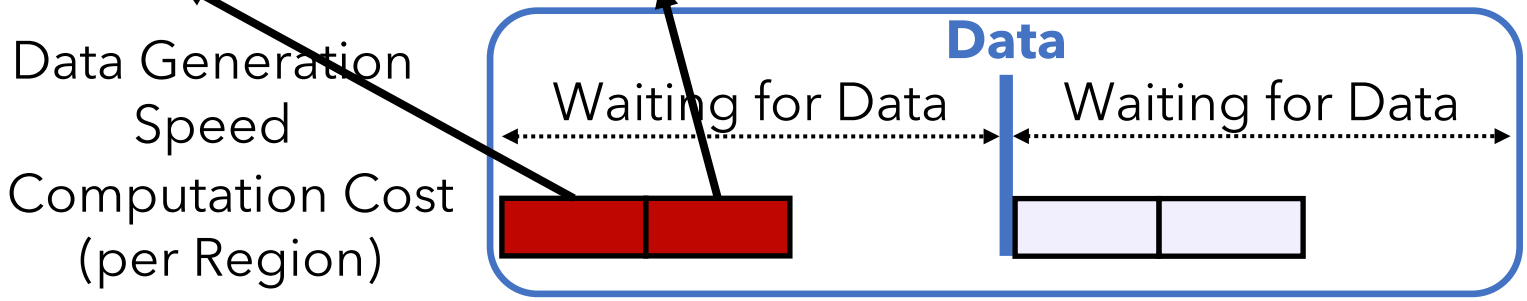
- K many nucleotides (k-mers) sequenced at a time
- **Observation:** Abrupt change in the signal as DNA moves inside a nanopore (e.g., when sequencing a new k-mer)
- **Goal:** Identify **raw signal segments** corresponding each sequenced k-mer
  - **Statistical tests (segmentation)** to identify abrupt changes
  - **Event:** A raw signal **segment** corresponding to a **particular k-mer**



Scalability Issues with Costly Calculations

A Small Genome (Bacteria)

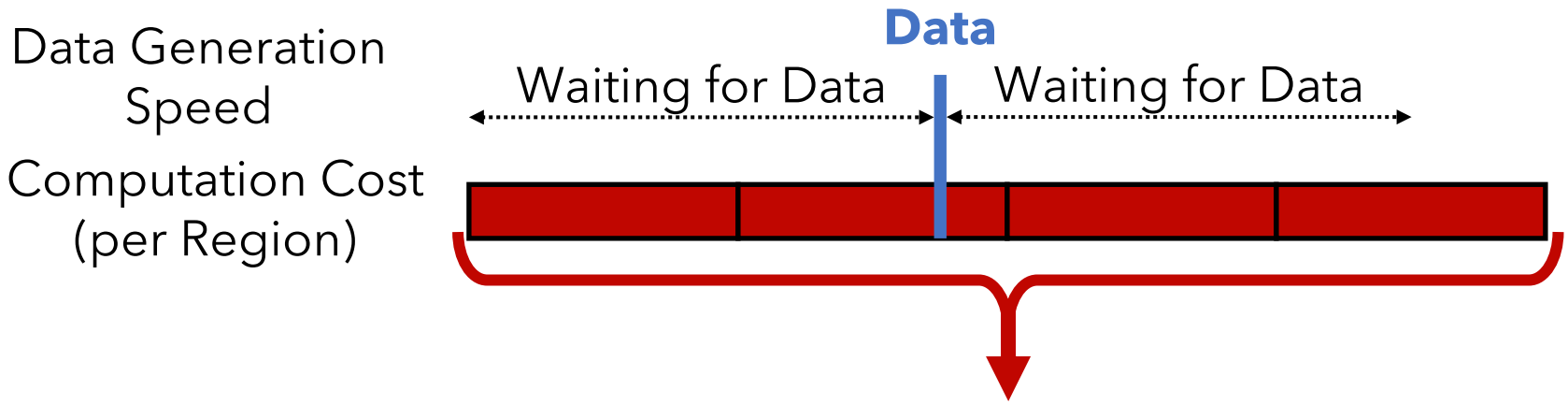
CGAAATGCC . . . AGATTAAATGGGCTTTCT



Real-Time Analysis: Data is consumed before next one arrives

A Large Genome (Human)

CGAAATGCC . . . AGATTAAATGGGCTTTCT . . . TGCCCTAA . . . GAATGGCGT



Increased Latency

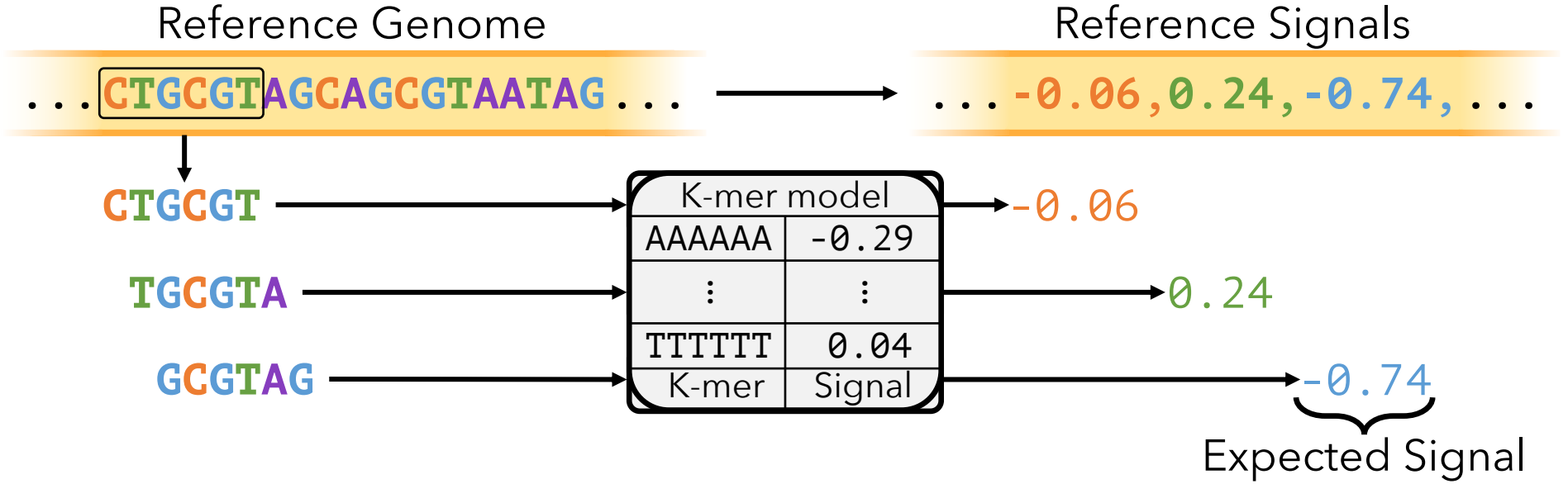
No Real-Time Analysis

Reference-to-Signal Conversion

- **Goal:** Enable direct comparison to raw signals by converting reference genome into its synthetic signal (one-time task)

- **K-mer model:** Provides **expected** signal values **for every possible k-mer**
  - A lookup table preconstructed based on nanopore's characteristics

- Use the **k-mer model** to convert **all k-mers** of a reference genome to their **expected** signal values



Challenges in Real-Time 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

Applications of Read Until

Depletion: Reads mapping to a particular reference genome is ejected

- Microbiome studies by removing host DNA
- Eliminating known residual DNA or RNA (e.g., mitochondrial DNA)
- High abundance genome removal

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

- Removing contaminated organisms
- Targeted sequencing (e.g., to a particular region of interest in the genome)
- Low abundance genome enrichment

Applications of Run Until & Sequence Until

Run Until: Stopping the entire sequencing run

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

Sequence Until: Run Until with accuracy-aware decision making

- 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
- …

In Vitro (e.g., PCR) vs. In Silico

- **Polymerase Chain Reaction (PCR)** as a way of in vitro "analysis"
  - Can increase the quantity of DNA in a sample
  - **Non-dynamic** targeted sequencing (e.g., low abundance *known* targets)
  - **Requires additional resources:** Time and money for preparation and execution of PCR

- **Adaptive sampling** as a way of in silico (i.e., computational) analysis
  - **Cannot** increase the existing quantity of DNA in a sample
  - **Dynamic targeted sequencing:** Decisions can be made based on real-time analysis (e.g., Sequence Until)
  - Minimal additional resources
    - ***Almost* no additional resources** for preparation and execution
    - **Simultaneous** enrichment and depletion is possible
    - Better suited for rapid whole genome sequencing
  - *Beauty* of computational analysis (e.g., high flexibility – no need for primers)

- PCR and adaptive sampling can be combined depending on the analysis type

Finding Mapping Positions

- Useful for **any application** that requires exact genomic position
  - Variant calling in downstream analysis
  - Specifically: Identifying rare variants in cancer genomics
  - Methylation profiling

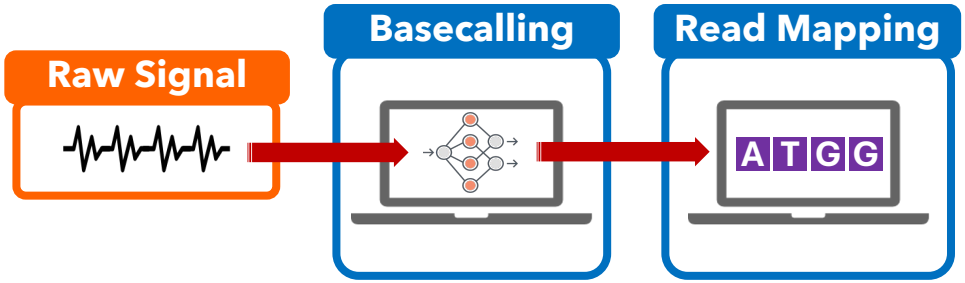
- Accurate and flexible **depth of coverage estimation**
  - **Alternative: DNA quantification** (without computational analysis)
    - DNA quantification is challenging for metagenomics analysis
  - **Computational method:** We can map to almost entire set of known reference genomes to accurately estimate the coverage of a metagenomics sample

- **Transcriptome analysis**
  - Accurately quantifying expression levels & alternative splicing

- **Better resolution** (i.e., more sensitive analysis) for any other application that does not specifically require mapping positions

Analyzing Raw Nanopore Signals

Traditional: Translating (**basecalling**) signals to bases **before** analysis

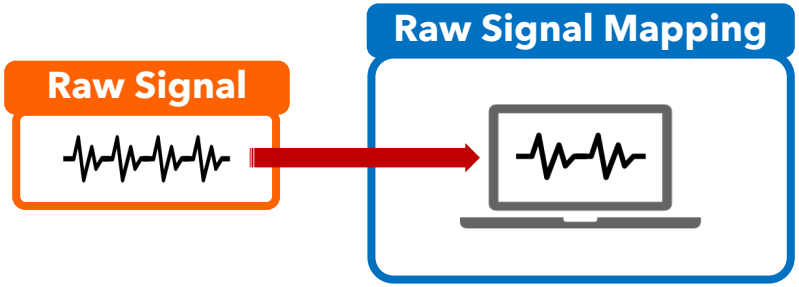


✓ Basecalled sequences are less noisy than raw signals

✓ Many analysis tools use basecalled sequences

✗ Costly and power-hungry computational requirements

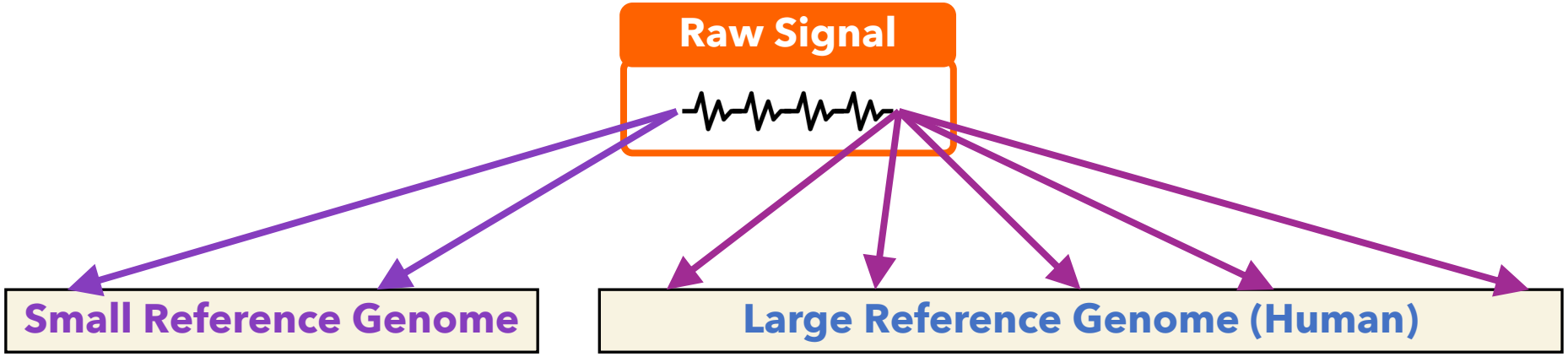
Recent Work: Directly analyzing signals **without basecalling**



✓ Efficient analysis with better scalability and portability

✓ Raw signals retain more information than just bases

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

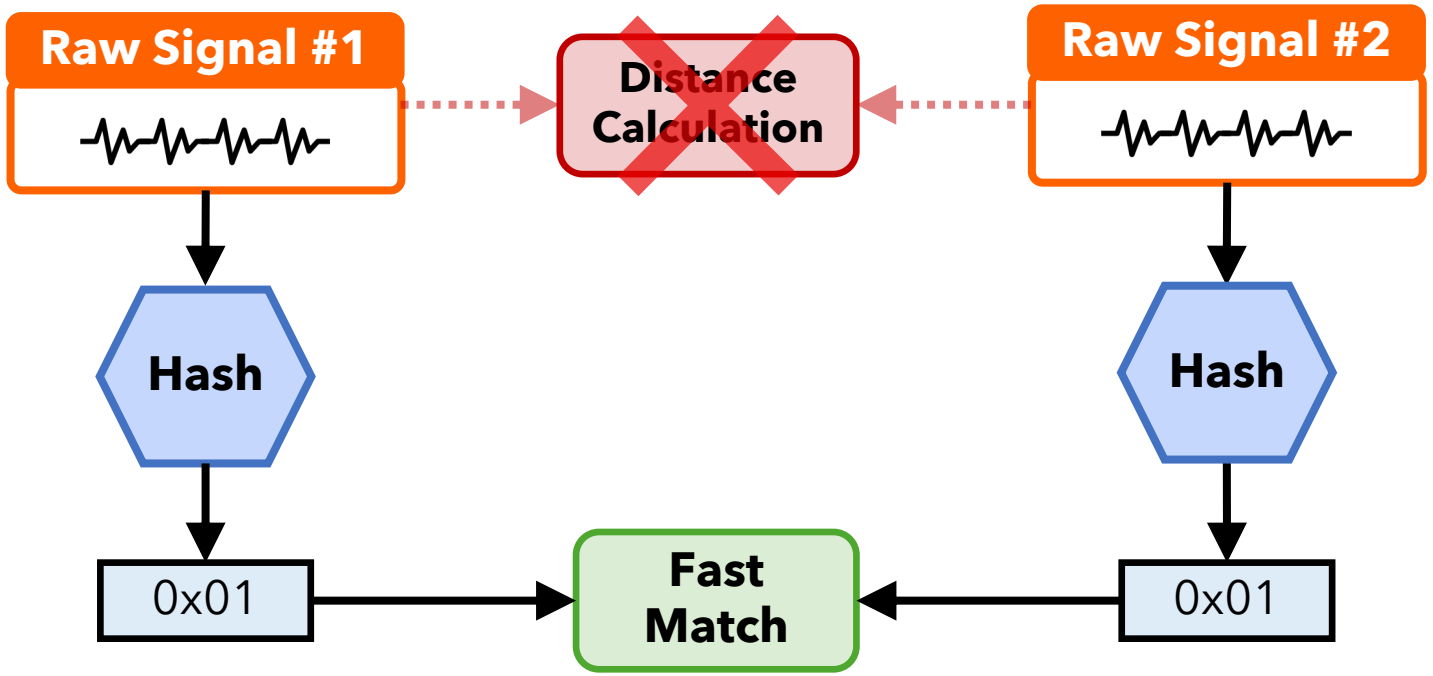
Problem: Probabilistic mechanisms on **many regions → inaccurate mapping**

High throughput

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

RawHash – Key Idea

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



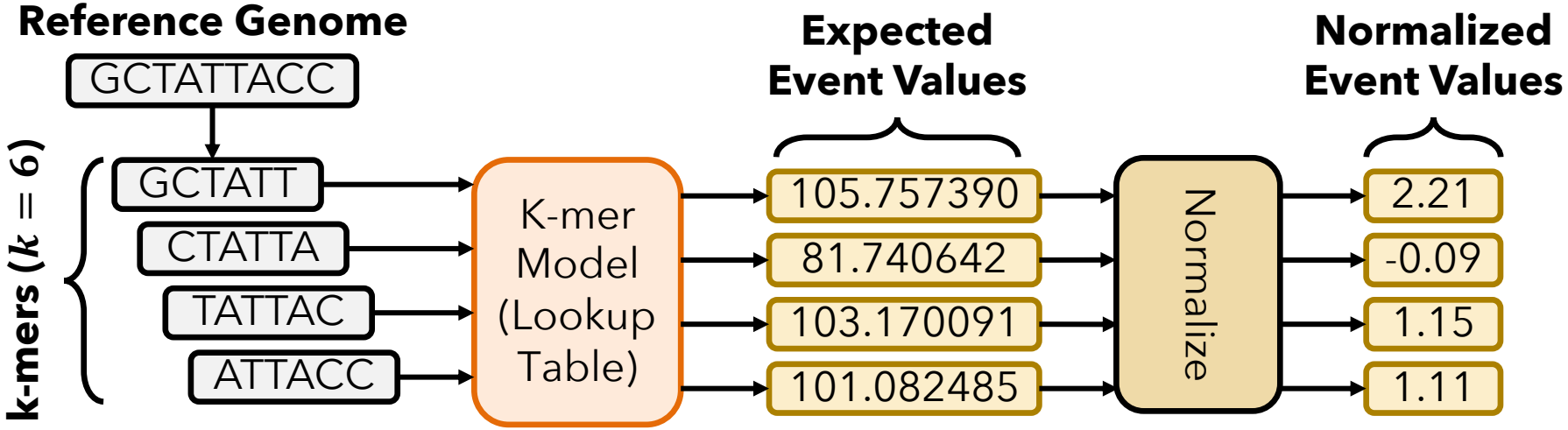
Challenge #1: Generating the **same** hash value for **similar enough** signals

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

Reference-to-Event Conversion

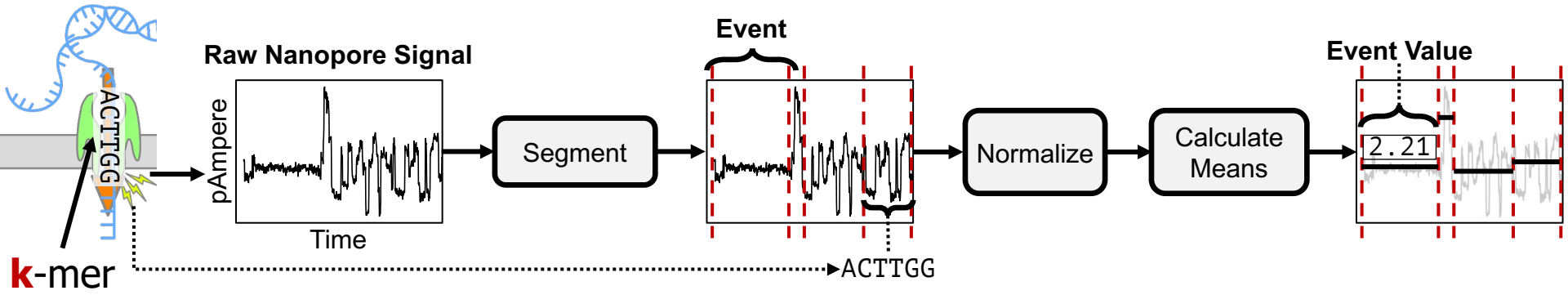
- **K-mer model:** Provides **expected** event values **for each k-mer**
  - Preconstructed based on nanopore sequencer characteristics

- Use the **k-mer model** to convert **all k-mers** of a reference genome to their **expected** event values



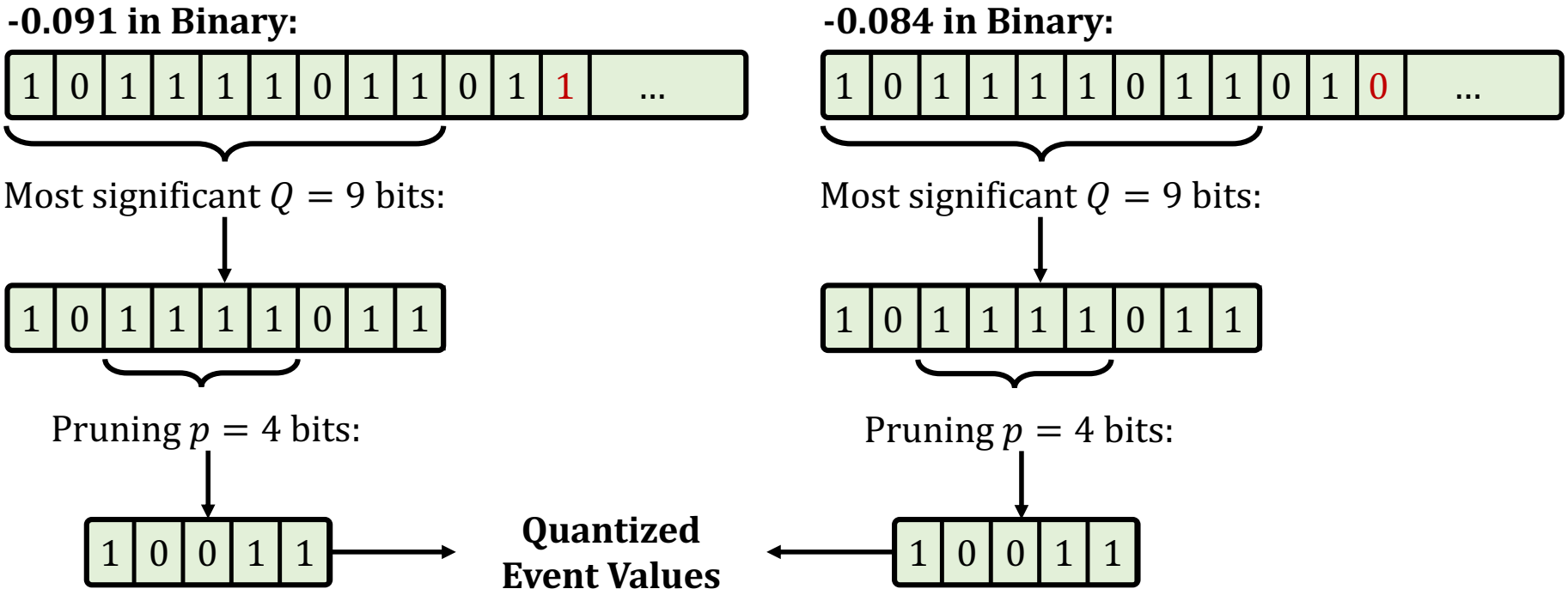
Enabling Analysis From Electrical Signals

- K many nucleotides (k-mers) sequenced at a time
- **Event:** A **segment** of the raw signal
  - Corresponds to a **particular** k-mer

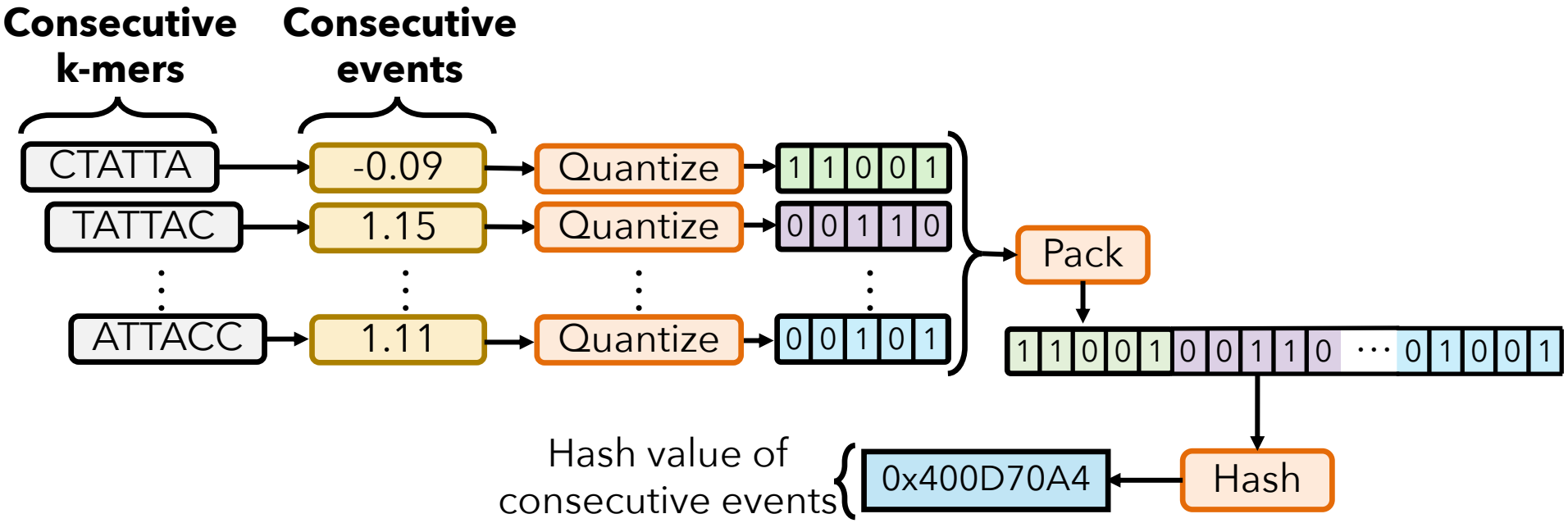


- **Observation:** Event values generated after sequencing **the same k-mer** are **similar** in value (not necessarily the same)

Quantization -- RawHash



Packing and Hashing

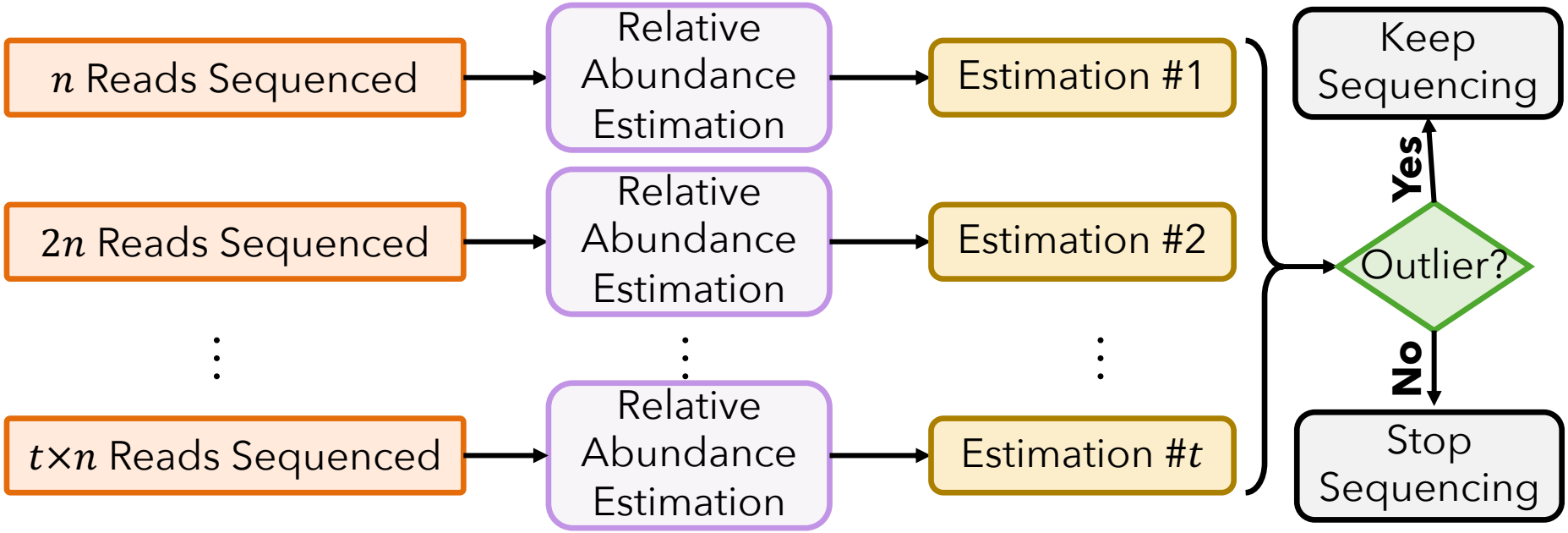


The Sequence Until Mechanism

- **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**

The Sequence Until Mechanism

- **Key Steps:**
  1. Continuously generate relative abundance estimation after every n reads
  2. Keep the last t estimation results
  3. **Detect outliers** in the results via **cross-correlation** of the recent t results
  4. Absence of outliers indicates **consistent results**
     - Further sequencing **is likely** to generate consistent results ➔ Stop the sequencing



Sequence Until – RawHash & UNCALLED

	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 (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**

Sequence Until – RawHash

	Estimated Relative Abundance Ratios in 50,000 Random Reads					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
RawHash (100%)	0.0270	0.3636	0.3062	0.1951	0.1081	N/A
RawHash + *Sequence Until* (7%)	0.0283	0.3539	0.3100	0.1946	0.1133	0.0118

Presets

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 – RawHash

Tool	Version
RawHash	0.9
UNCALLED	2.2
Sigmap	0.1
Minimap2	2.24

Related Works

- **Basecalled real-time analysis**
  - ReadFish, ReadBouncer, RUBRIC: Basecalled read mapping
  - SPUMONI, SPUMONI 2: Basecalled binary classification using r-index
  - Coriolis: Basecalled metagenomics classification
  - baseLess: k-mer calling for classification

- **Raw signal analysis without basecalling**
  - SquiggleNet, DeepSelectNet, RawMap: Target/non-target classification
  - Sigmoni: Target/non-target classification using r-index
  - UNCALLED, Sigmap, RawHash: Read mapping

Adaptive Quantization

$$q(s) = \begin{cases} \lfloor n \times (f_r \times \frac{(s - f_{min})}{f_{max} - f_{min}}) & \text{if } f_{min} \leq s \leq f_{max} \\ \lfloor n \times (f_r + c_r \times s) & \text{if } s < f_{min} \\ \lfloor n \times (f_r + c_r + c_r \times s) & \text{if } s > f_{max} \end{cases}$$

Chaining Scores – RawHash vs RawHash2

- **RawHash Chaining**

$$f(i) = \max\left\{ \max_{i>j\geq 1}\{f(j) + \alpha(j, i)\}, w_i \right\}$$

$$\alpha(j, i) = \min\left\{ \min\{y_i - y_j, x_i - x_j\}, w_i \right\}$$

- **RawHash2 Chaining**

$$f(i) = \max\left\{ \max_{i>j\geq 1}\{f(j) + \alpha(j, i) - \beta(j, i)\}, w_i \right\}$$

$$\beta(j, i) = \gamma_c\big((y_i - y_j) - (x_i - x_j)\big)$$

$$\gamma_c(l) = \begin{cases} 0.01 \cdot \bar{w} \cdot |l| + 0.5 \log_2 |l| & (l \neq 0) \\ 0 & (l = 0) \end{cases}$$

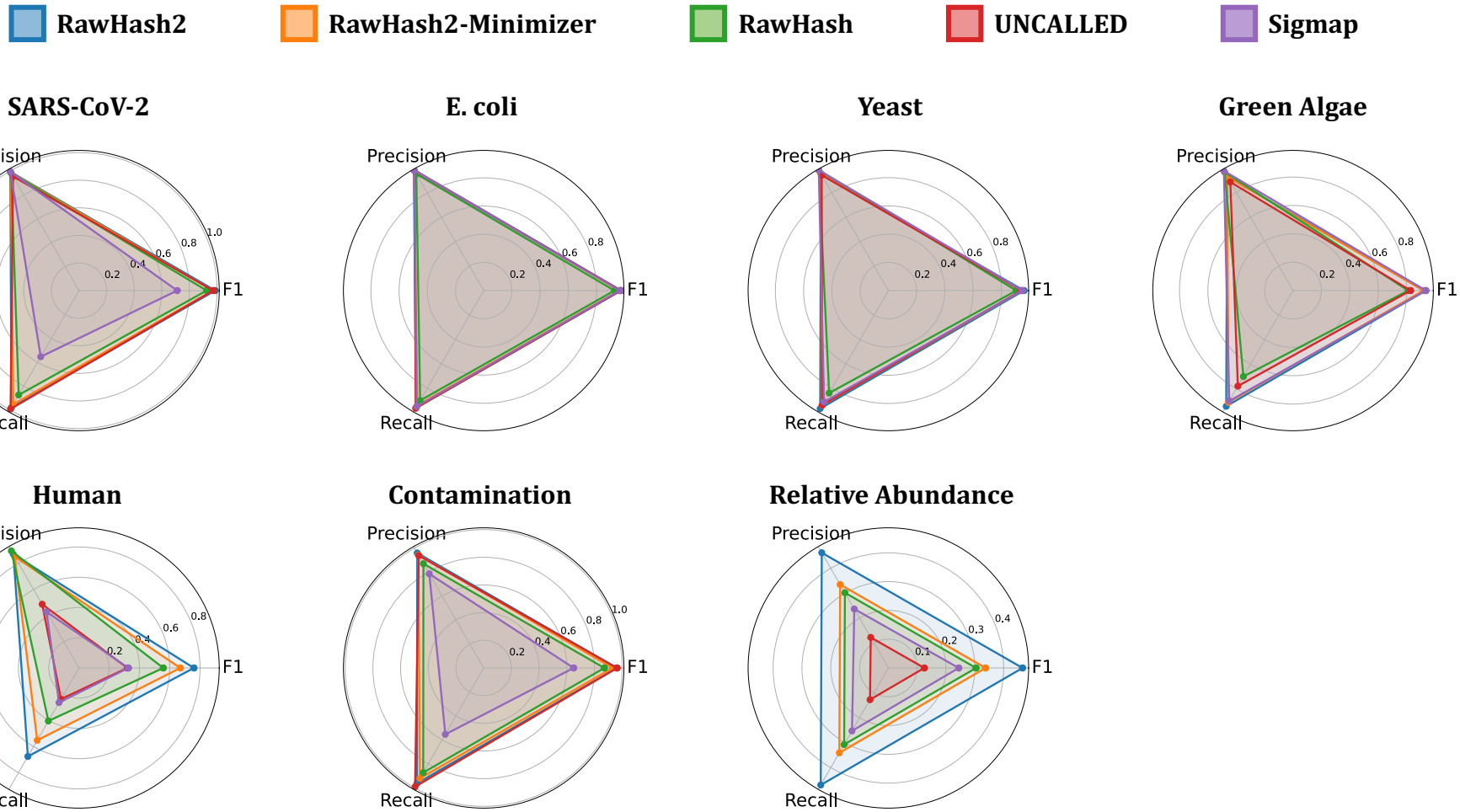
Datasets

	Organism	Device Type	Flow Cell Type	Transloc. Speed	Sampling Frequency	Basecaller Model	Reads (#)	Bases (#)	SRA Accession	Reference Genome	Genome Size
					Read Mapping						
D1	*SARS-CoV-2*	MinION	R9.4.1 e8 (FLO-MIN106)	450	4000	Guppy HAC v3.2.6	1,382,016	594M	CADDE Centre	GCF_009858895.2	29,903
D2	*E. coli*	GridION	R9.4.1 e8 (FLO-MIN106)	450	4000	Guppy HAC v5.0.12	353,317	2,365M	ERR9127551	GCA_000007445.1	5M
D3	*Yeast*	MinION	R9.4.1 e8 (FLO-MIN106)	450	4000	Albacore v2.1.7	49,989	380M	SRR8648503	GCA_000146045.2	12M
D4	*Green Algae*	PromethION	R9.4.1 e8 (FLO-PRO002)	450	4000	Albacore v2.3.1	29,933	609M	ERR3237140	GCF_000002595.2	111M
D5	*Human*	MinION	R9.4.1 e8 (FLO-MIN106)	450	4000	Guppy Flip-Flop v2.3.8	269,507	1,584M	FAB42260	T2T-CHM13 (v2)	3,117M
D6	*E. coli*	GridION	R10.4 e8.1 (FLO-MIN112)	450	4000	Guppy HAC v5.0.16	1,172,775	6,123M	ERR9127552	GCA_000007445.1	5M
D7	*S. aureus*	GridION	R10.4 e8.1 (FLO-MIN112)	450	4000	Dorado SUP v0.5.3	407,727	1,281M	SRR21386013	GCF_000144955.2	2.8M
					Contamination Analysis						
			D1 and D5				1,651,523	2,178M	D1 and D5	D1	29,903
					Relative Abundance Estimation						
			D1-D5				2,084,762	5,531M	D1-D5	D1-D5	3,246M

Accuracy

Dataset	Metric	RH2	RH2-Min.	RH	UNCALLED	Sigmap
SARS-CoV-2	F1	**0.9867**	0.9691	0.9252	0.9725	0.7112
E. coli	F1	**0.9748**	0.9631	0.9280	0.9731	0.9670
Yeast	F1	**0.9602**	0.9472	0.9060	0.9407	0.9469
Green Algae	F1	**0.9351**	0.9191	0.8114	0.8277	0.9350
Human	F1	**0.7599**	0.6699	0.5574	0.3197	0.3269
Contamination	Precision	**0.9595**	0.9424	0.8702	0.9378	0.7856
Rel. Abundance	Distance	**0.2678**	0.4243	0.4385	0.6812	0.5430

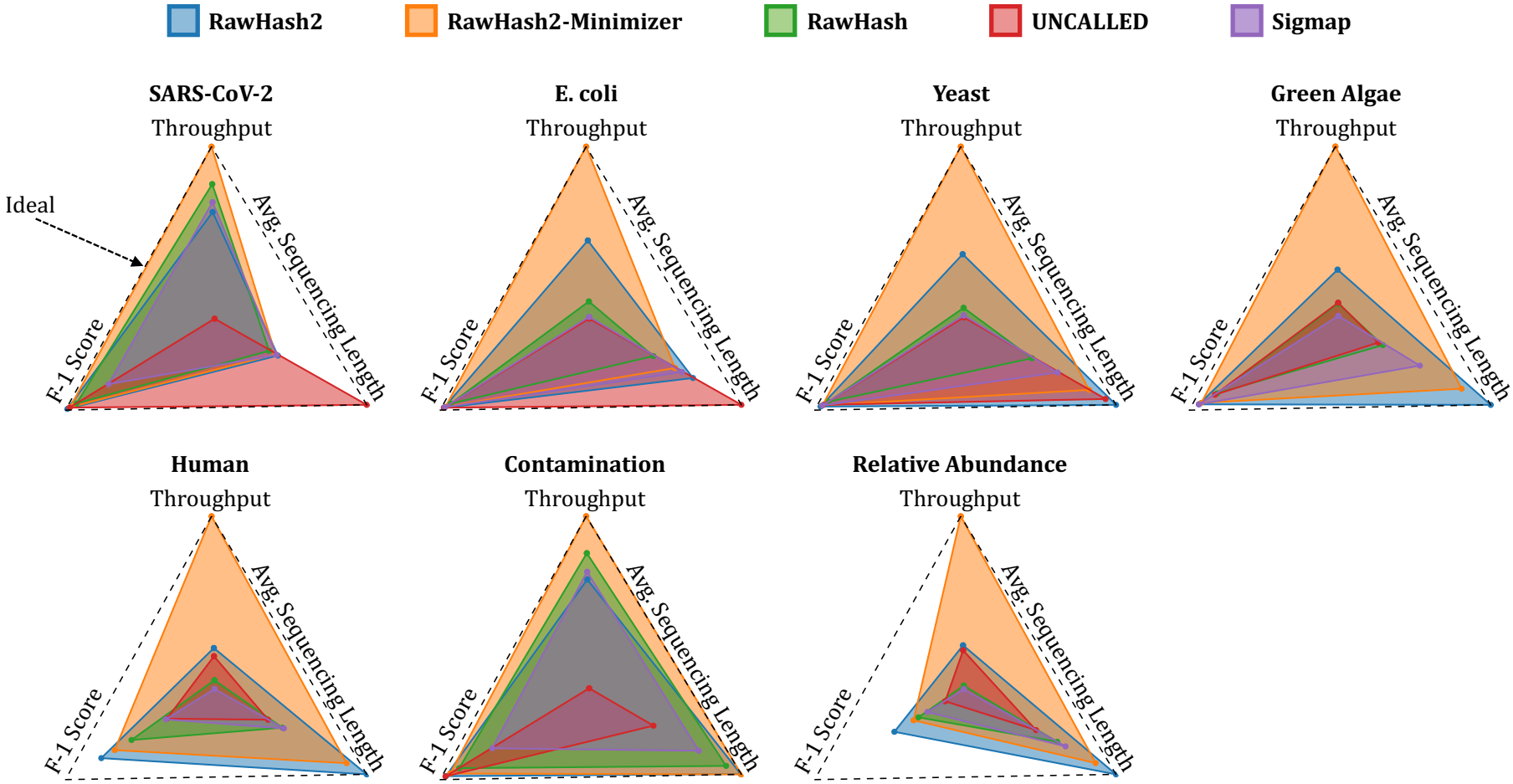
Mapping Accuracy – Radar



Mapping Accuracy – All Metrics

Dataset	Metric	RH2	RH2-Min.	RH	UNCALLED	Sigmap
SARS-CoV-2	F1	**0.9867**	0.9691	0.9252	0.9725	0.7112
	Precision	**0.9939**	0.9868	0.9832	0.9547	0.9929
	Recall	0.9796	0.9521	0.8736	**0.9910**	0.5540
E. coli	F1	**0.9748**	0.9631	0.9280	0.9731	0.9670
	Precision	**0.9904**	0.9865	0.9563	0.9817	0.9842
	Recall	0.9597	0.9408	0.9014	**0.9647**	0.9504
Yeast	F1	**0.9602**	0.9472	0.9060	0.9407	0.9469
	Precision	0.9553	0.9561	0.9852	0.9442	**0.9857**
	Recall	**0.9652**	0.9385	0.8387	0.9372	0.9111
Green Algae	F1	**0.9351**	0.9191	0.8114	0.8277	0.9350
	Precision	0.9284	0.9280	0.9652	0.8843	**0.9743**
	Recall	**0.9418**	0.9104	0.6999	0.7779	0.8987
Human	F1	**0.7599**	0.6699	0.5574	0.3197	0.3269
	Precision	0.8675	0.8511	**0.8943**	0.4868	0.4288
	Recall	**0.6760**	0.5523	0.4049	0.2380	0.2642
Contamination	F1	0.9614	0.9317	0.8718	**0.9637**	0.6498
	Precision	**0.9595**	0.9424	0.8702	0.9378	0.7856
	Recall	0.9632	0.9212	0.8736	**0.9910**	0.5540
Rel. Abundance	F1	**0.4659**	0.3375	0.3045	0.1249	0.2443
	Precision	**0.4623**	0.3347	0.3018	0.1226	0.2366
	Recall	**0.4695**	0.3404	0.3071	0.1273	0.2525

Combined Benefits – Radar



Sequenced Length

Dataset	RH2	RH2-Min.	RH	UNCALLED	Sigmap
SARS-CoV-2	443.92	460.85	513.95	**184.51**	452.38
E. coli	851.31	1,030.74	1,376.14	**580.52**	950.03
Yeast	**1,147.66**	1,395.87	2,565.09	1,233.20	1,862.69
Green Algae	**1,385.59**	1,713.46	4,760.59	5,300.15	2,591.16
Human	**2,130.59**	2,455.99	4,773.58	6,060.23	4,680.50
Contamination	670.69	**667.89**	742.56	1,582.63	927.82
Rel. Abundance	**1,024.28**	1,182.04	1,669.46	2,158.50	1,533.04

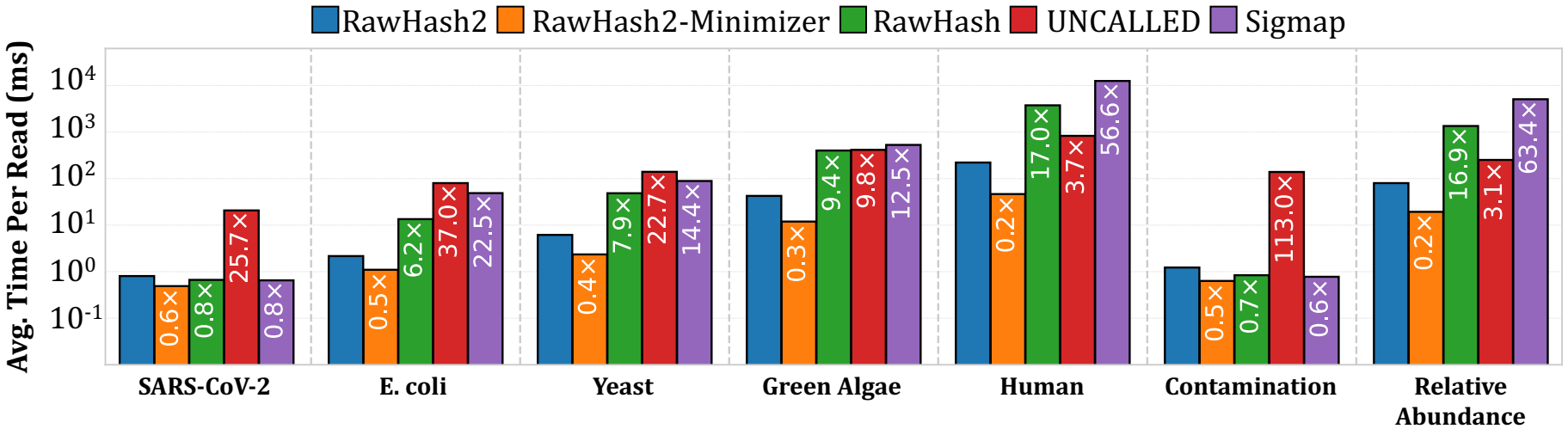
Computational Resources #1

Dataset	RH2	RH2-Min.	RH	UNCALLED	Sigmap
			Indexing CPU Time (sec)		
SARS-CoV-2	0.12	0.06	0.16	8.40	**0.02**
E. coli	2.48	**1.61**	2.56	10.57	8.86
Yeast	4.56	**3.02**	4.44	16.40	25.29
Green Algae	27.60	**17.73**	24.51	213.13	420.25
Human	1,093.56	**588.30**	809.08	3,496.76	41,993.26
Contamination	0.13	0.06	0.15	8.38	**0.03**
Rel. Abundance	747.74	**468.14**	751.67	3,666.14	36,216.87
			Indexing Peak Memory (GB)		
SARS-CoV-2	**0.01**	**0.01**	**0.01**	0.06	**0.01**
E. coli	0.35	0.19	0.35	**0.11**	0.40
Yeast	0.75	0.39	0.76	**0.30**	1.04
Green Algae	5.11	**2.60**	5.33	11.94	8.63
Human	80.75	**40.59**	83.09	48.43	227.77
Contamination	**0.01**	**0.01**	**0.01**	0.06	**0.01**
Rel. Abundance	152.59	75.62	152.84	**47.80**	238.32
			Mapping CPU Time (sec)		
SARS-CoV-2	1,705.43	**1,227.05**	1,539.64	29,282.90	1,413.32
E. coli	1,296.34	**787.49**	7,453.21	28,767.58	22,923.09
Yeast	545.77	**246.37**	4,145.38	7,181.44	7,146.32
Green Algae	2,135.83	**657.63**	22,103.03	12,593.01	26,778.44
Human	100,947.58	**21,860.05**	1,825,061.23	245,128.15	6,101,179.89
Contamination	3,783.69	**2,332.28**	3,480.43	234,199.60	3,011.78
Rel. Abundance	250,076.90	**62,477.76**	4,551,349.79	569,824.13	15,178,633.11

Computational Resources #2

Mapping Peak Memory (GB)					
SARS-CoV-2	4.15	4.16	4.20	**0.17**	28.26
E. coli	4.13	4.03	4.18	**0.50**	111.12
Yeast	4.38	4.12	4.37	**0.36**	14.66
Green Algae	6.11	4.98	11.77	**0.78**	29.18
Human	48.75	25.04	52.43	**10.62**	311.94
Contamination	4.16	4.14	4.17	**0.62**	111.70
Rel. Abundance	49.14	25.82	54.89	**8.99**	486.63
Mapping Throughput (bp/sec)					
SARS-CoV-2	552,561.25	**885,263.48**	694,274.92	9,260.31	602,380.96
E. coli	303,382.45	**659,013.57**	72,281.32	7,515.76	13,750.97
Yeast	150,547.61	**394,766.80**	28,757.15	7,471.48	11,624.82
Green Algae	28,742.46	**98,323.70**	9,488.79	10,069.41	2,569.89
Human	8,968.78	**37,086.38**	2,099.35	7,225.67	236.45
Contamination	563,129.81	**884,929.30**	696,873.20	9,343.95	601,936.49
Rel. Abundance	9,501.37	**36,919.79**	962.79	8,437.70	196.48
CPU Threads Needed for the entire MinION Flowcell (512 pores)					
SARS-CoV-2	**1**	**1**	**1**	25	**1**
E. coli	**1**	**1**	4	31	17
Yeast	2	**1**	9	31	20
Green Algae	9	**3**	25	23	90
Human	26	**7**	110	32	975
Contamination	**1**	**1**	**1**	25	**1**
Rel. Abundance	25	**7**	240	28	1173

Average Time Spent per Read



FAST5 vs. POD5. vs S/BLOW5

Tool	*E. coli*	*Yeast*
Elapsed Time (mm:ss)		
RH2-FAST5	19:27	08:35
RH2-POD5	16:55	07:33
RH2-BLOW5	17:32	07:38
RH2-Min.-FAST5	12:13	03:56
RH2-Min.-POD5	09:42	02:56
RH2-Min.-BLOW5	10:16	03:02

Flow Cell Types R9 vs R10.4

Flow Cell		RH2	RH2-Min.
Read Mapping Accuracy (E. coli)			
R9.4	F1	0.9748	0.9631
	Precision	0.9904	0.9865
	Recall	0.9597	0.9408
R10.4	F1	0.8960	0.8389
	Precision	0.9506	0.9325
	Recall	0.8473	0.7623
Read Mapping Accuracy (S. aureus)			
R10.4	F1	0.7749	0.6778
	Precision	0.8649	0.8167
	Recall	0.7018	0.5793
Performance (E. coli)			
R9.4	Throughput [bp/sec]	303,382.45	659,013.57
	Mean time per read [ms]	2.161	1.099
R10.4	Throughput [bp/sec]	175,351.94	480,471.75
	Mean time per read [ms]	6.598	2.505
Performance (S. aureus)			
R10.4	Throughput [bp/sec]	256,680.4	617,308.7
	Mean time per read [ms]	5.478	2.243

Ratio of Filtered Seed Hits

Dataset	Average Filtered Ratio
SARS-CoV-2	0.0627
E. coli	0.5505
Yeast	0.5356
Green Algae	0.8106
Human	0.5104
E. coli (R10.4)	0.6895
S. aureus (R10.4)	0.6003

Presets

Preset	Corresponding parameters	Usage
viral	-e 6 -q 4 –max-chunks 5 –bw 100 –max-target-gap 500 –max-target-gap 500 –min-score 10 –chain-gap-scale 1.2 –chain-skip-scale 0.3	Viral genomes
sensitive	-e 8 -q 4 –fine-range 0.4	Small genomes (i.e., < 500M bases)
fast	-e 8 -q 4 –max-chunks 20	Large genomes (i.e., > 500M bases)
	Other helper parameters	
depletion	–best-chains 5 –min-mapq 10 –w-threshold 0.5 –min-anchors 2 –min-score 15 –chain-skip-scale 0	Contamination analysis
r10	-k9 –seg-window-length1 3 –seg-window-length2 6 –seg-threshold1 6.5 –seg-threshold2 4 –seg-peak-height 0.2 –chain-gap-scale 1.2	For R10.4 Flow Cells

Versions

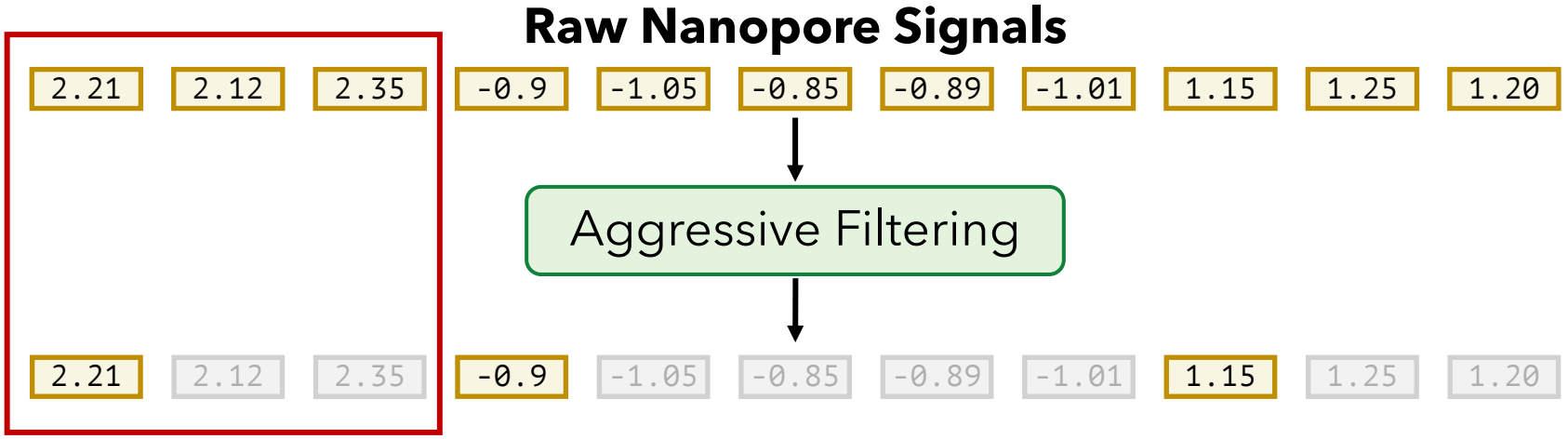
Tool	Version
RawHash2	2.1
RawHash	1.0
UNCALLED	2.3
Sigmap	0.1
Minimap2	2.24

FAST5 (HDF5)	1.10
POD5	0.2.2
S/BLOW5	1.2.0-beta

Indexing using Raw Signals

1. Constructing the hash table index **from raw nanopore signals**
   - Reference-to-signal conversion **is mainly free from noise** (e.g., stay errors)
   - Indexed raw nanopore signals **are not free from noise**

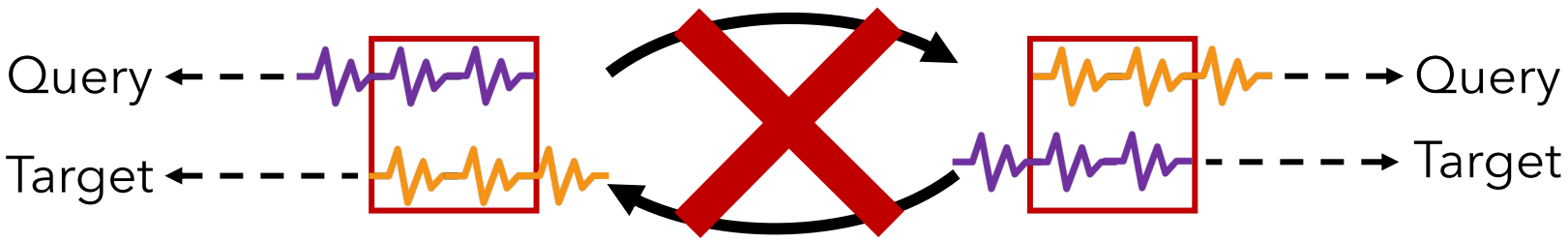
2. **Aggressively filtering** consecutive and similar signals to **substantially reduce noise** at the cost of data loss



Chaining and Outputting Overlaps

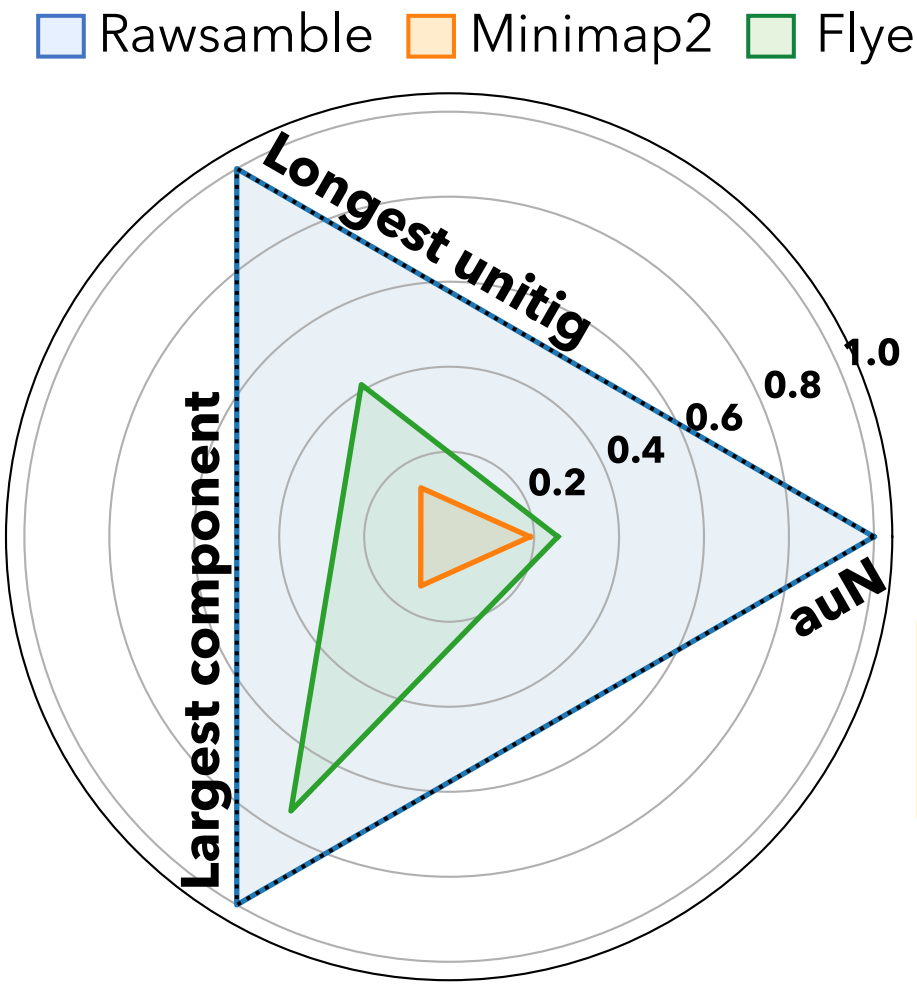
3. **Adjusting the minimum chaining score** to avoid false chains
   - All-vs-all overlapping tends to find a larger number of seed hits than mapping to a reference genome
   - Minimum score for a chain during overlapping is set to be **~5× larger than mapping**
   - **All such chains are reported** (instead of a single best mapping)

4. **Avoid cyclic overlaps** with deterministic comparisons



Low-Coverage Human Genome Assembly

- Results are shown **relative to the best result from each metric**
  - **Metrics:** auN, longest unitig, largest connected unitigs (component)
  - **Coverage: 0.6x**



Rawsamble leads to
better contiguity
than using basecalled reads
at a low coverage dataset

Can **the richer information in raw signals improve assembly quality** where basecalled analysis falls short?

Datasets

	Organism	Device Type	Reads (#)	Bases (#)	Avg. Read Length	Estimated Coverage (×)	SRA Accession
D1	*SARS-CoV-2*	MinION	10,001	4.02M	402	135×	CADDE Centre
D2	*E. coli*	GridION	353,948	2,332M	6,588	445×	ERR9127551
D3	*Yeast*	MinION	50,023	385M	7,698	32×	SRR8648503
D4	*Green Algae*	PromethION	30,012	622M	20,731	5.6×	ERR3237140
D5	*Human*	MinION	270,006	1,773M	6,567	0.6×	FAB42260

Throughput

	D1 *SARS-CoV-2*	D2 *E. coli*	D3 *Yeast*	D4 *Green Algae*	D5 *Human*
Throughput	2,065,764	2,720,702	2,128,800	1,668,065	3,579,472

Performance

Organism	Tool	Elapsed time (hh:mm:ss)	CPU time (sec)	Peak Mem. (GB)
D1	**Rawsamble**	0:00:03	33	1.07
SARS-CoV-2	**Minimap2**	0:00:01 (0.33×)	19 (0.58×)	0.16 (0.15×)
	Minimap2 + Dorado CPU (Fast)	0:01:45 (35.00×)	3,227 (97.79×)	44.93 (41.99×)
	Minimap2 + Dorado CPU (HAC)	0:05:45 (115.00×)	5,457 (165.36×)	57.98 (54.19×)
	Minimap2 + Dorado GPU (HAC)	0:01:41 (33.67×)	NA	0.8 (0.75×)
	Minimap2 + Dorado GPU (SUP)	0:25:47 (515.67×)	NA	1.23 (1.15×)
D2	**Rawsamble**	1:12:44	132,758	6.72
E. coli	**Minimap2**	0:14:25 (0.20×)	25,721 (0.19×)	26.73 (3.98×)
	Minimap2 + Dorado CPU (Fast)	7:17:05 (6.01×)	583,358 (4.39×)	50.43 (7.50×)
	Minimap2 + Dorado CPU (HAC)	32:26:12 (26.76×)	1,335,697 (10.06×)	38.0 (5.65×)
	Minimap2 + Dorado GPU (HAC)	0:36:14 (0.50×)	NA	26.73 (3.98×)
	Minimap2 + Dorado GPU (SUP)	1:30:30 (1.24×)	NA	26.73 (3.98×)
D3	**Rawsamble**	0:01:18	2,241	6.39
Yeast	**Minimap2**	0:00:21 (0.27×)	290 (0.13×)	5.25 (0.82×)
	Minimap2 + Dorado CPU (Fast)	0:54:04 (41.59×)	71,796 (32.04×)	56.13 (8.78×)
	Minimap2 + Dorado CPU (HAC)	3:13:56 (149.18×)	193,640 (86.41×)	65.43 (10.24×)
	Minimap2 + Dorado GPU (HAC)	0:04:33 (3.50×)	NA	5.25 (0.82×)
	Minimap2 + Dorado GPU (SUP)	0:10:33 (8.12×)	NA	5.92 (0.93×)
D4	**Rawsamble**	0:07:57	14,064	8.67
Green Algae	**Minimap2**	0:00:47 (0.10×)	882 (0.06×)	8.7 (1.00×)
	Minimap2 + Dorado CPU (Fast)	1:16:35 (9.63×)	79,606 (5.66×)	50.88 (5.87×)
	Minimap2 + Dorado CPU (HAC)	4:30:07 (33.98×)	286,362 (20.36×)	64.07 (7.39×)
	Minimap2 + Dorado GPU (HAC)	0:06:01 (0.76×)	NA	8.7 (1.00×)
	Minimap2 + Dorado GPU (SUP)	0:14:54 (1.87×)	NA	8.7 (1.00×)
D5	**Rawsamble**	0:28:56	51,975	6.0
Human	**Minimap2**	0:01:52 (0.06×)	1,372 (0.03×)	20.21 (3.37×)
	Minimap2 + Dorado CPU (Fast)	6:42:24 (13.91×)	802,983 (15.45×)	81.98 (13.66×)
	Minimap2 + Dorado CPU (HAC)	23:27:18 (48.64×)	1,219,043 (23.45×)	46.12 (7.69×)
	Minimap2 + Dorado GPU (HAC)	0:20:24 (0.71×)	NA	20.31 (3.38×)
	Minimap2 + Dorado GPU (SUP)	1:05:48 (2.27×)	NA	20.21 (3.37×)

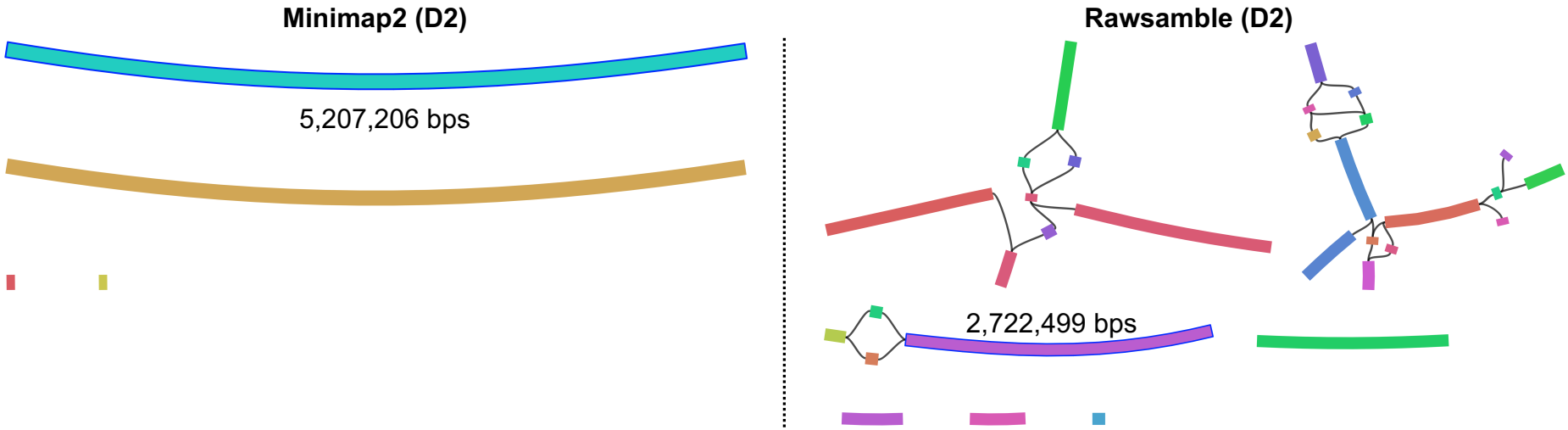
Overlapping Statistics

	Organism	Unique to Rawsamble (%)	Unique to Minimap2 (%)	Shared Overlaps (%)
D1	*SARS-CoV-2*	11.55	15.27	73.18
D2	*E. coli*	8.33	50.62	41.05
D3	*Yeast*	24.94	35.17	39.89
D4	*Green Algae*	3.76	78.64	17.61
D5	*Human*	32.69	56.18	11.13

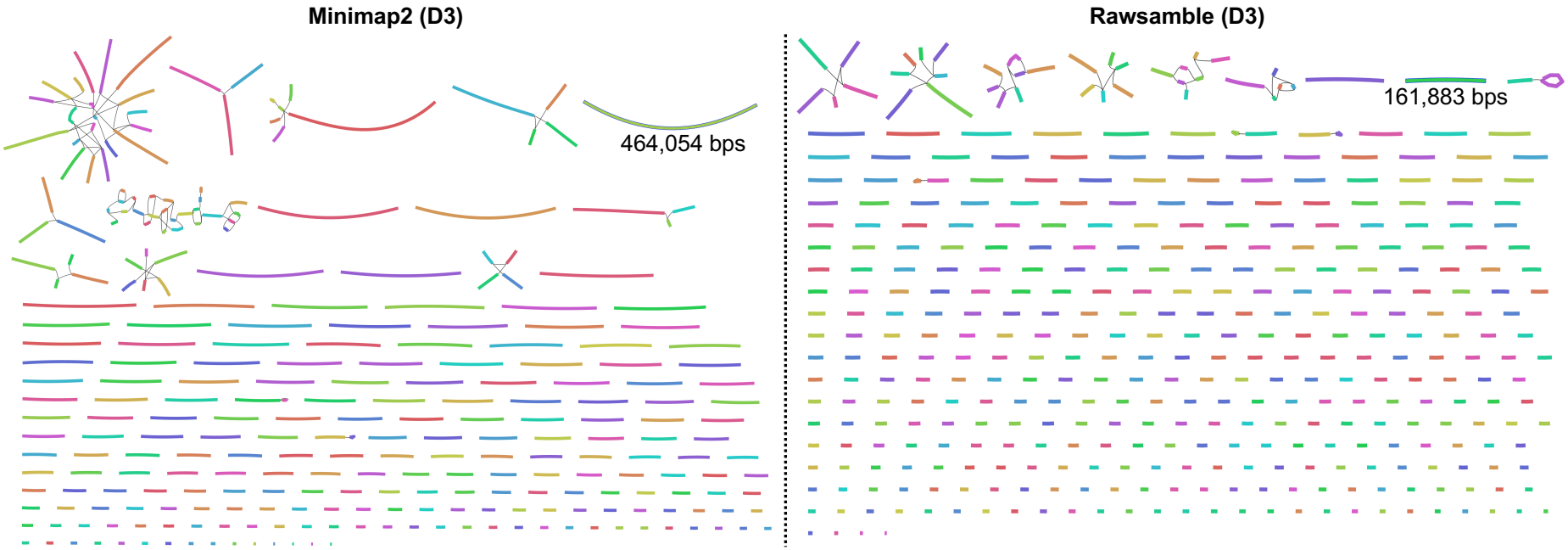
Assembly Statistics

Dataset	Tool	Total Length (bp)	Largest Comp. (bp)	N50 (bp)	auN (bp)	Longest Unitig (bp)	Unitig Count
D2	Rawsamble	14,525,505	4,841,669	1,535,079	1,309,738	2,722,499	31
E. coli	minimap2	10,434,542	5,207,206	5,204,754	5,194,738	5,207,206	4
	Gold standard	5,235,343	5,235,343	5,235,343	5,235,343	5,235,343	1
D3	Rawsamble	13,898,208	362,050	41,118	48,106	161,883	396
Yeast	minimap2	23,755,455	1,611,876	134,050	150,908	464,054	282
	Gold standard	11,963,521	11,835,059	640,934	623,210	1,073,346	68
D4	Rawsamble	3,448,899	448,422	93,111	108,818	252,038	50
Green Algae	minimap2	2,117,190	198,709	63,310	88,906	198,709	55
	Gold standard	106,479,288	2,255,807	452,774	538,136	1,667,975	420
D5	Rawsamble	1,850,419	493,004	51,300	116,049	364,113	48
Human	minimap2	747,607	65,951	19,476	22,103	48,424	61
	Gold standard	8,365,210	367,305	19,329	29,697	150,470	592

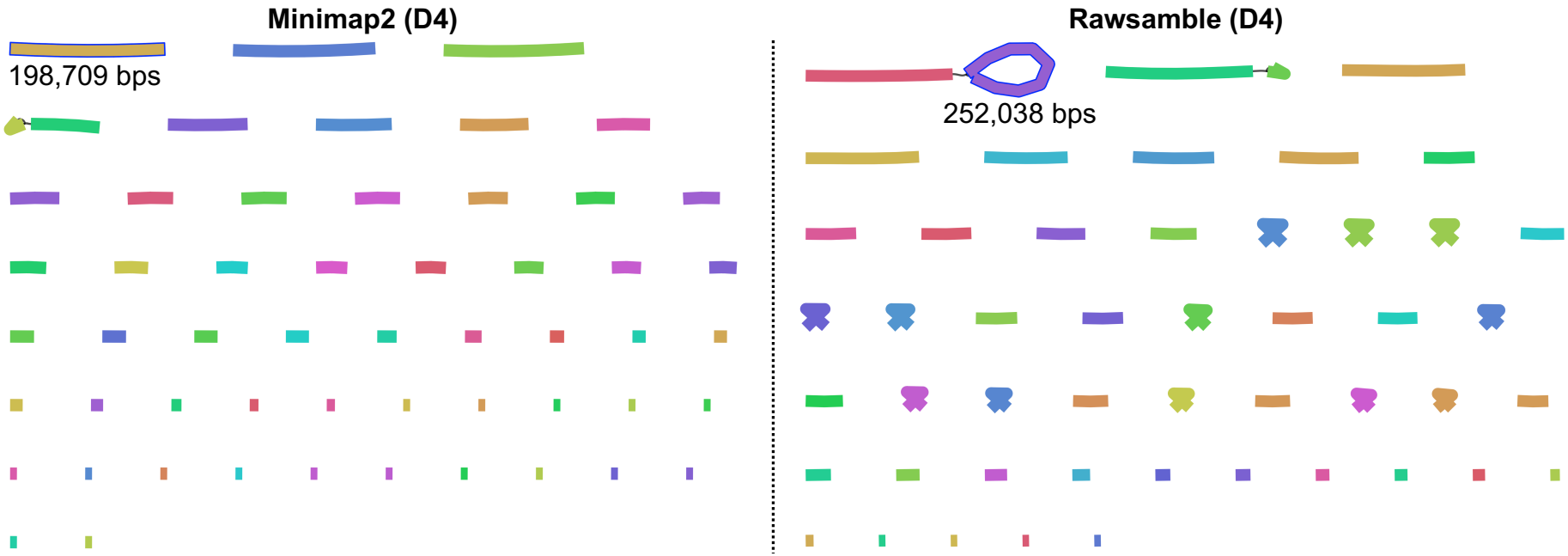
Visualizing the E. coli Assembly Graph



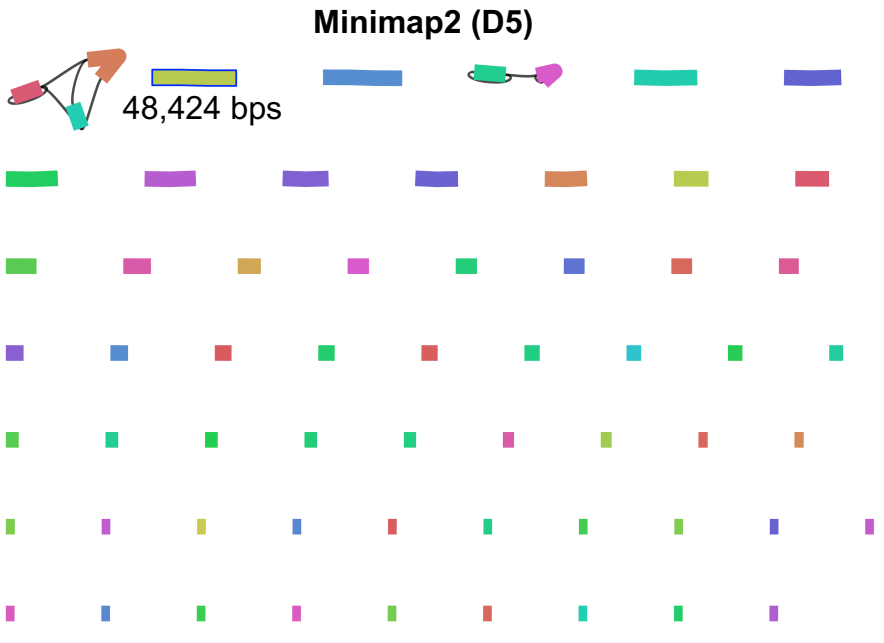
Visualizing the Yeast Assembly Graph

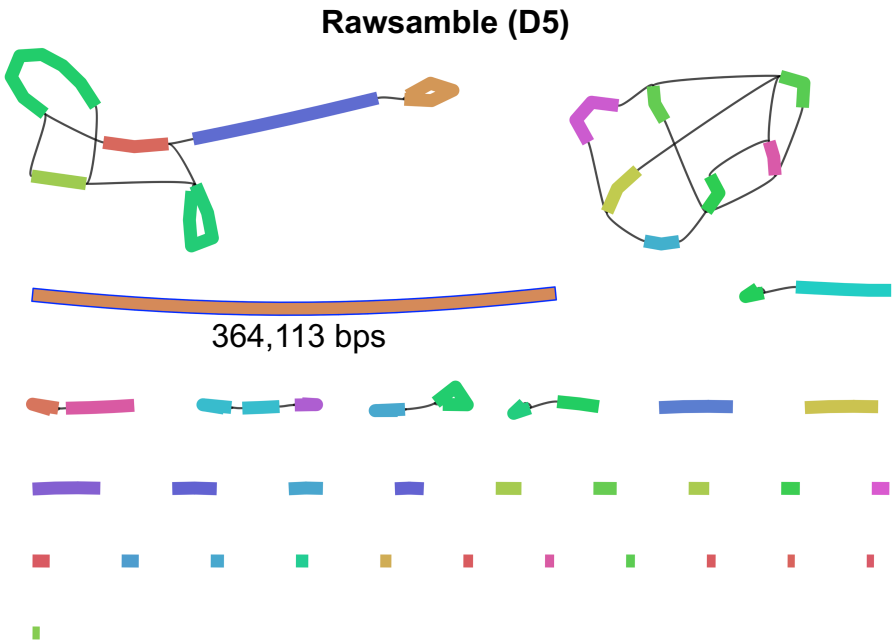


Visualizing the Green Algae Assembly Graph



Visualizing the Human Assembly Graph





HERRO Correction Before and After

Dataset	Coverage Before Correction	Coverage After Correction
D2 *E. coli*	445×	240×
D3 *Yeast*	32×	12×
D4 *Green algae*	5.6×	3.7×
D5 *Human*	0.6×	0.002×

Parameters

Tool	D1 *SARS-CoV-2*	D2 *E. coli*	D3 *Yeast*	D4 *Green Algae*	D5 *Human*
Rawsamble	-x ava-viral -t 32	-x ava -t 32	-x ava -t 32	-x ava -t 32	-x ava –chain-gap-scale 0.6 -t 32
Minimap2	-x ava-ont –for-only -t 32				
Dorado CPU (Fast)	basecaller -x cpu dna_r9.4.1_e8_fast@v3.4				
Dorado CPU (HAC)	basecaller -x cpu dna_r9.4.1_e8_hac@v3.3				
Dorado GPU (HAC)	basecaller dna_r9.4.1_e8_hac@v3.3				
Dorado GPU (SUP)	basecaller dna_r9.4.1_e8_sup@v3.3				
Miniasm					

Presets

Preset	Corresponding parameters	Usage
ava-viral	-e 6 -q 4 -w 0 –sig-diff 0.45 –fine-range 0.4 –min-score 20 –min-score2 30 –min-anchors 5 –min-mapq 5 –bw 1000 –max-target-gap 2500 –max-query-gap 2500 –chain-gap-scale 1.2 –chain-skip-scale 0.3	Viral genomes
ava	-e 8 -q 4 -w 3 –sig-diff 0.45 –fine-range 0.4 –min-score 40 –min-score2 75 –min-anchors 5 –min-mapq 5 –bw 5000 –max-target-gap 2500 –max-query-gap 2500	Default case

Versions

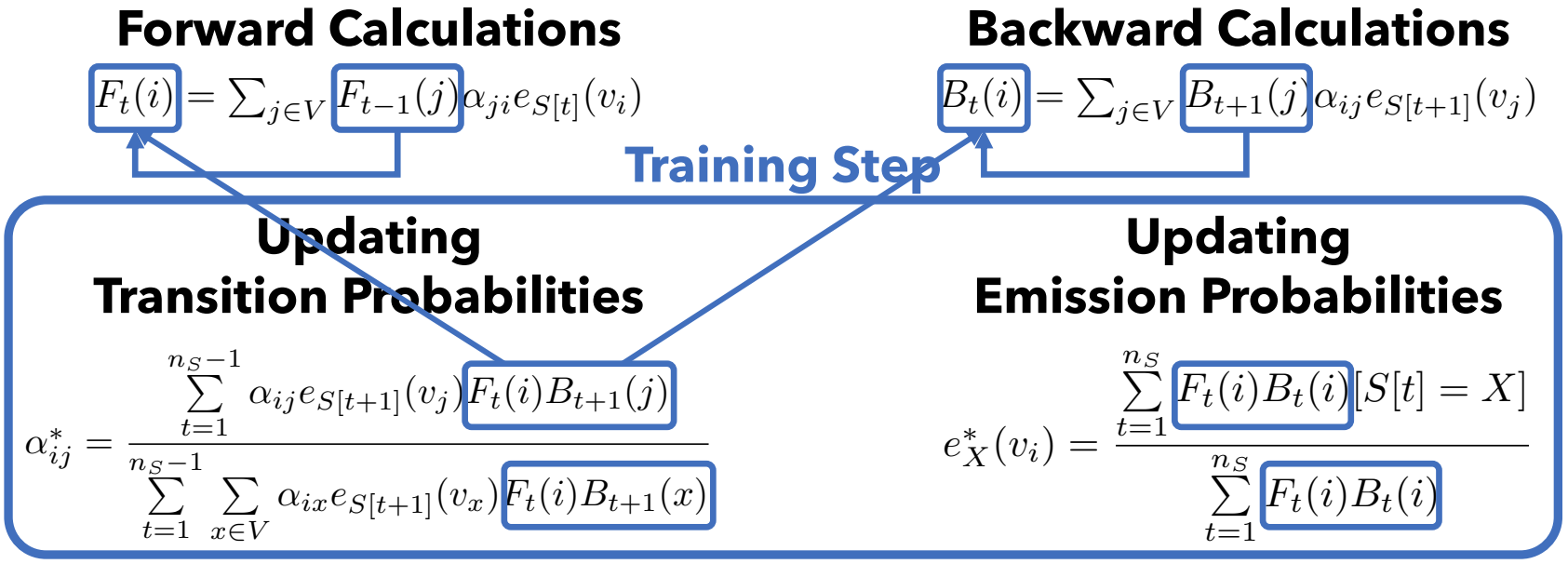
Tool	Version
Rawsamble	2.1
Minimap2	2.24
Dorado	0.7.3
Miniasm	0.3-r179
Rawasm	main
Flye	2.9.5
HERRO	0.1

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

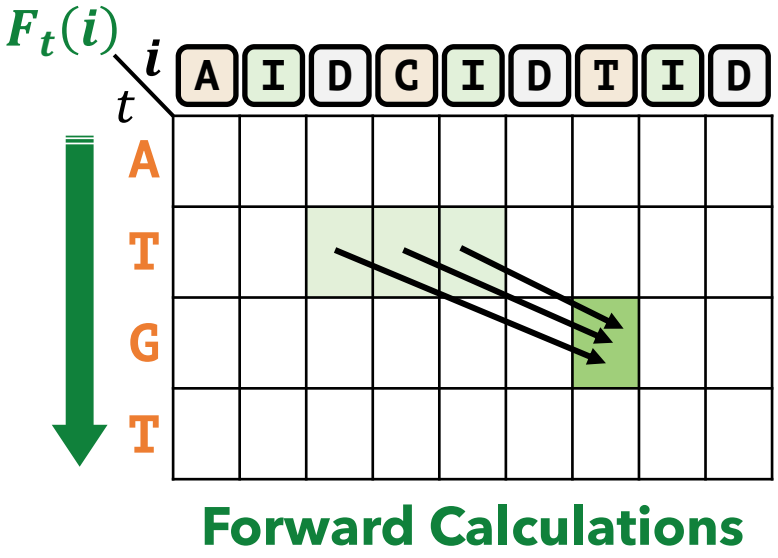
- **Training:** Maximizing parameters to observe certain variations



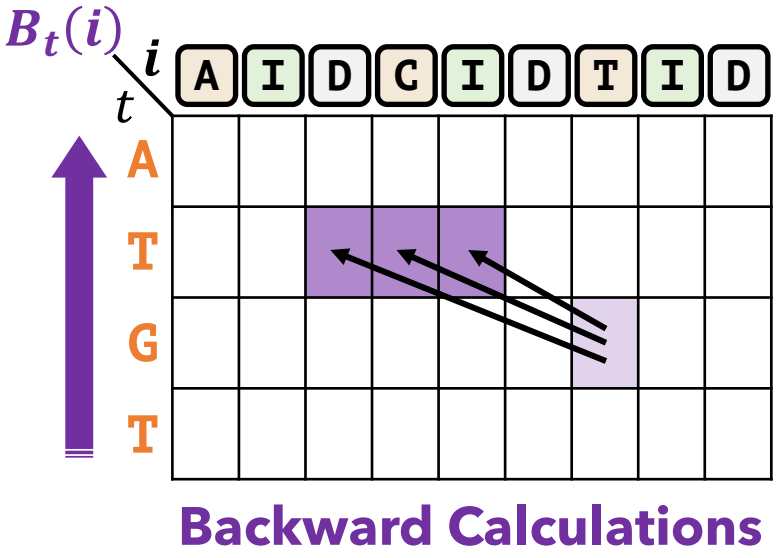
Forward & Backward Calculations

- **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



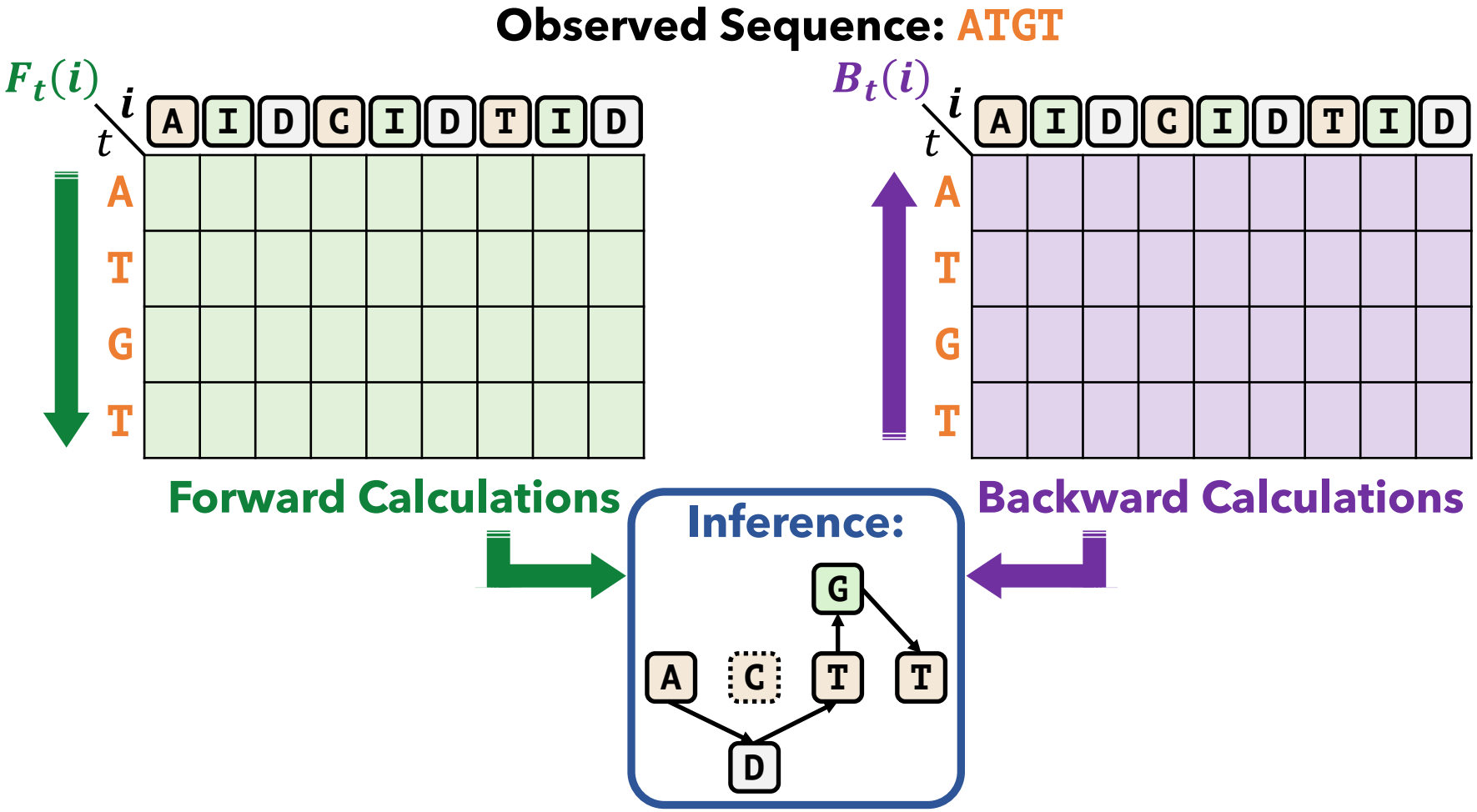
Forward Calculations



Backward Calculations

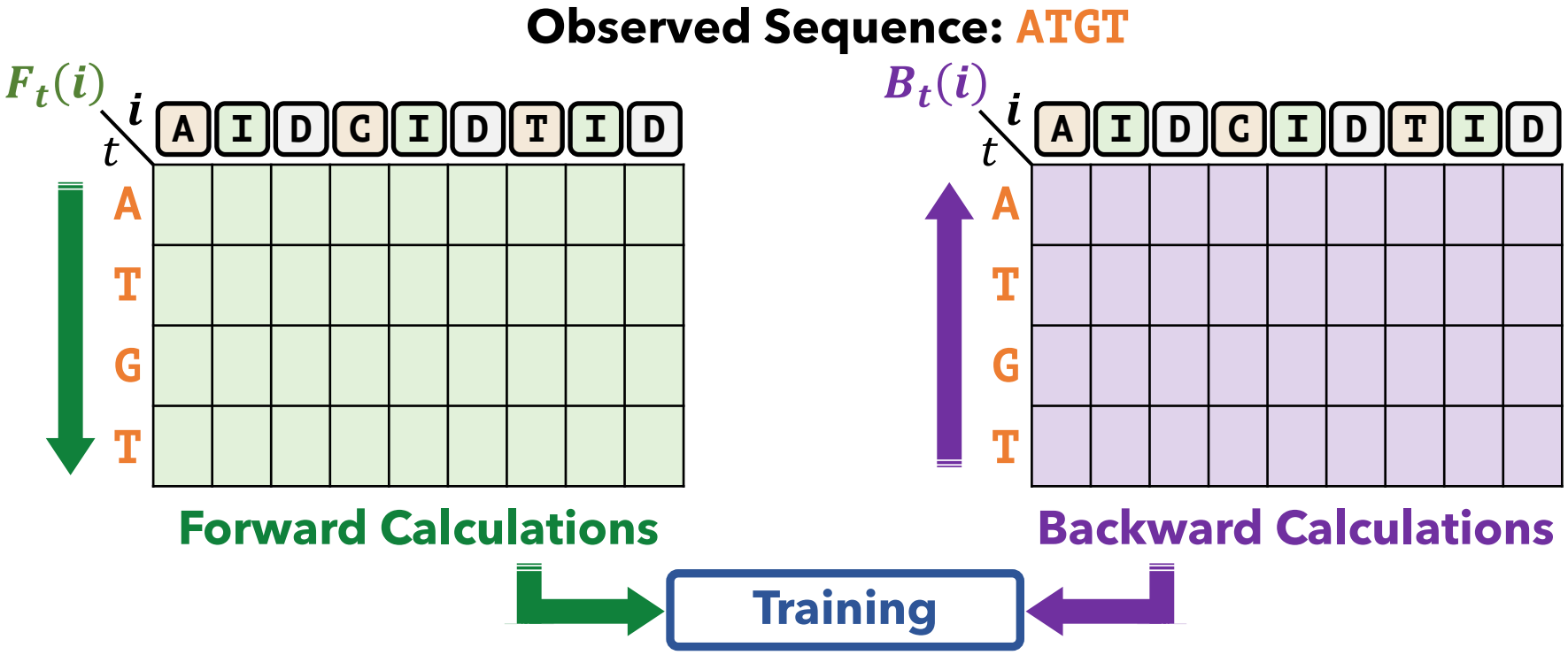
Inference using pHMMs

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



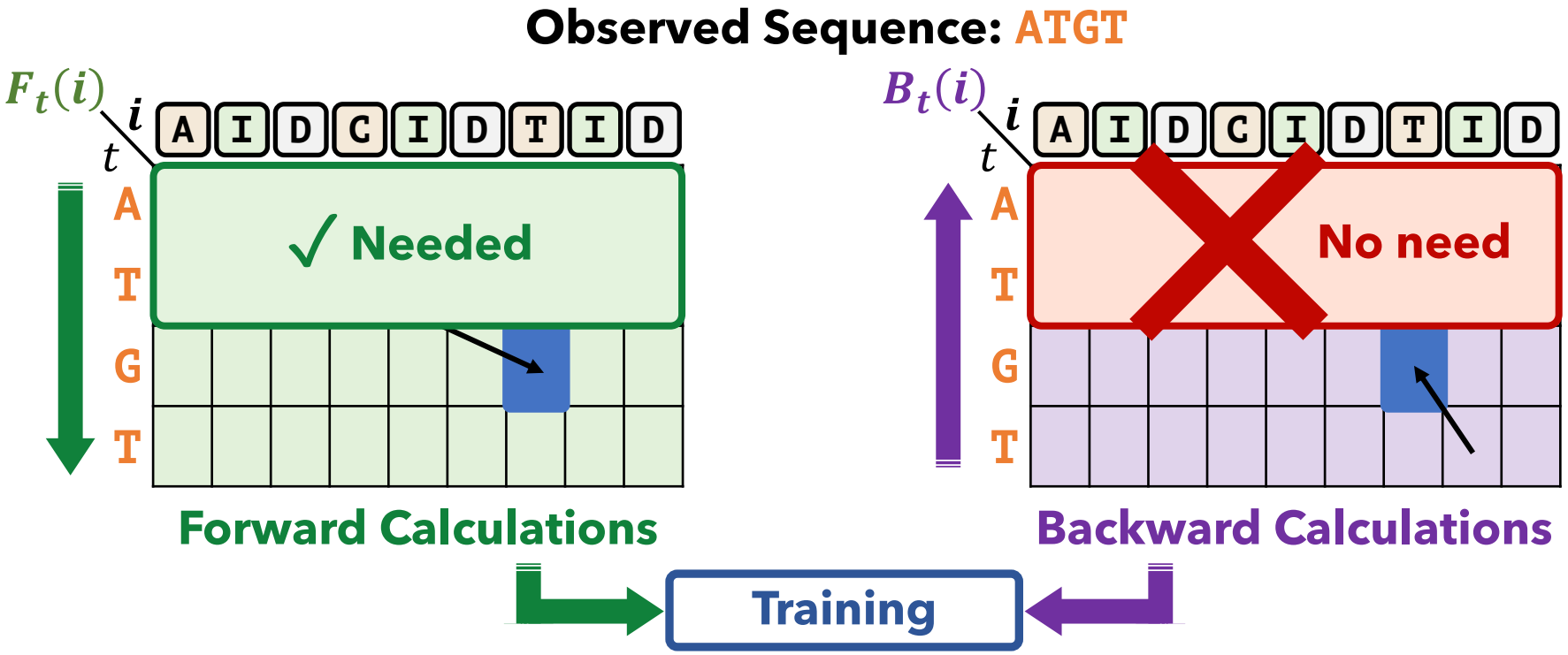
Training using pHMMs

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



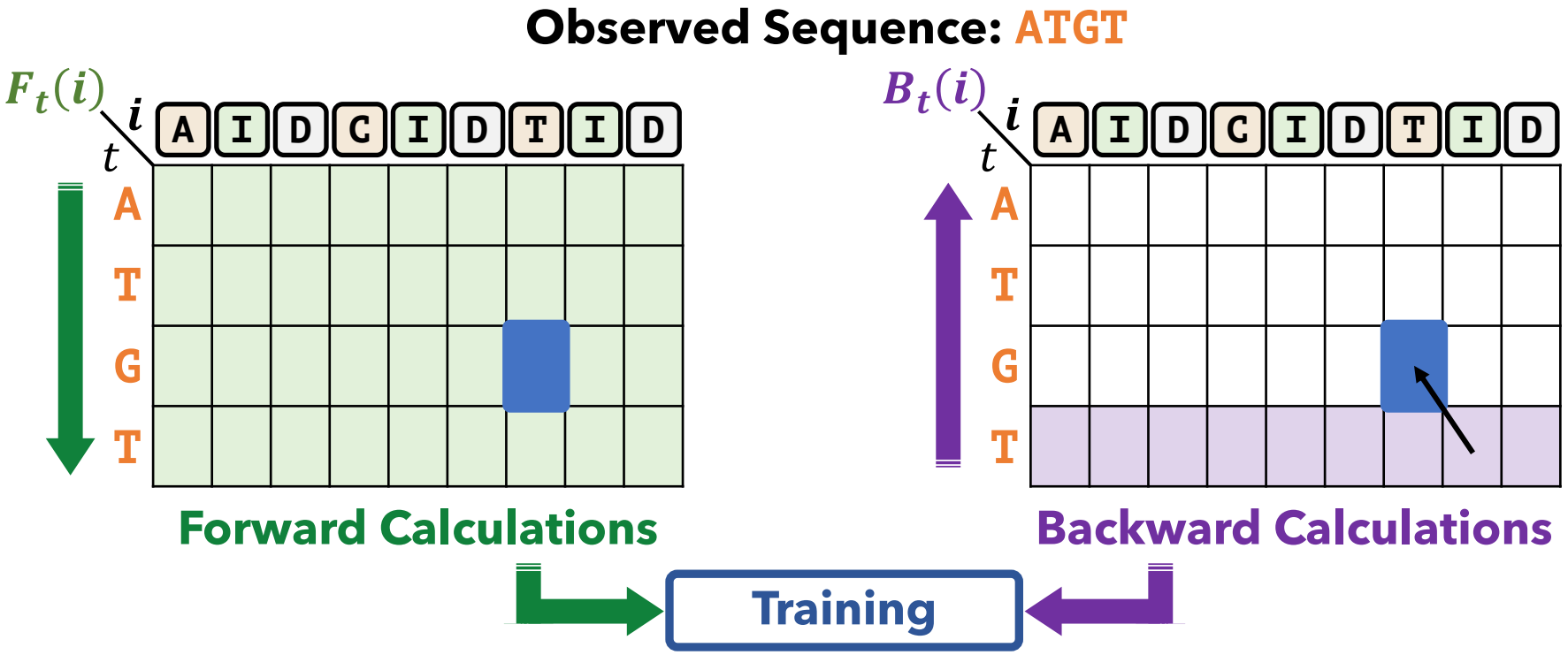
SW: Minimizing Redundant Storage

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



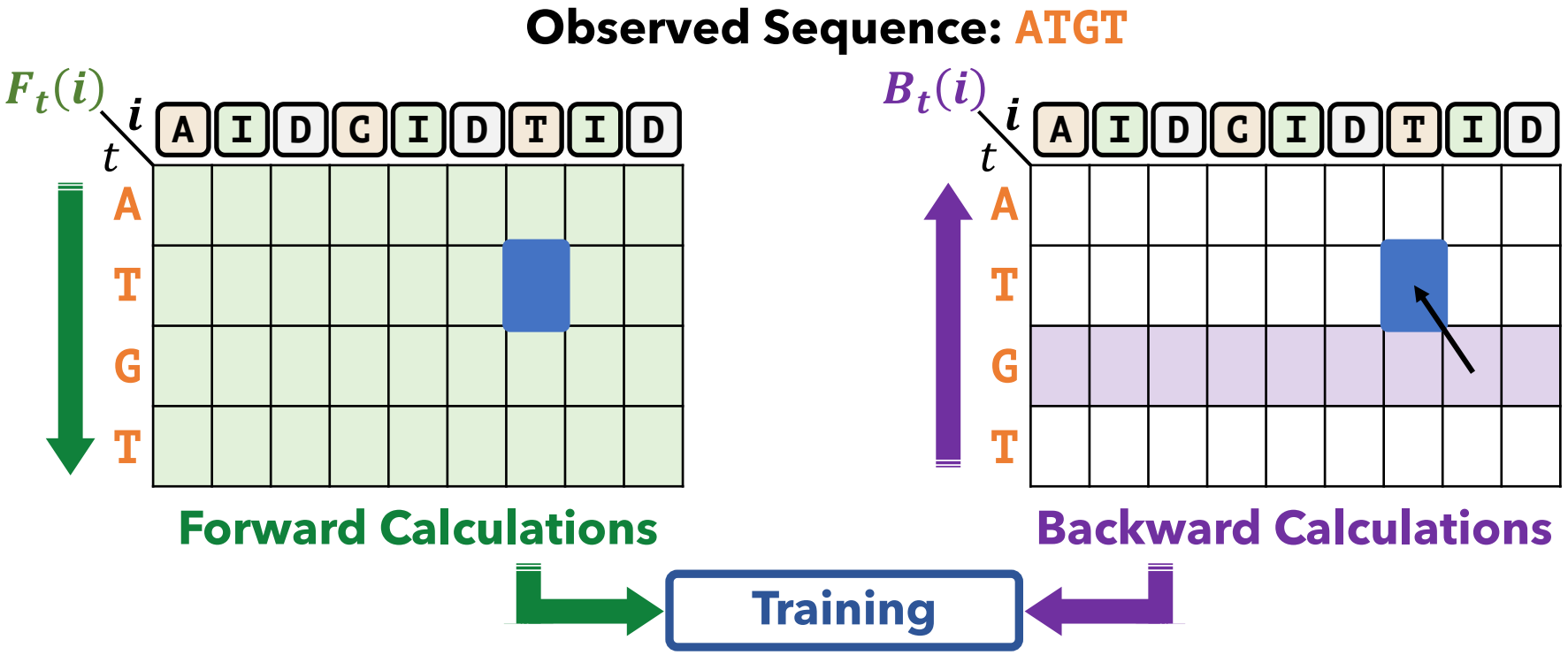
SW: Minimizing Redundant Storage

- **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**



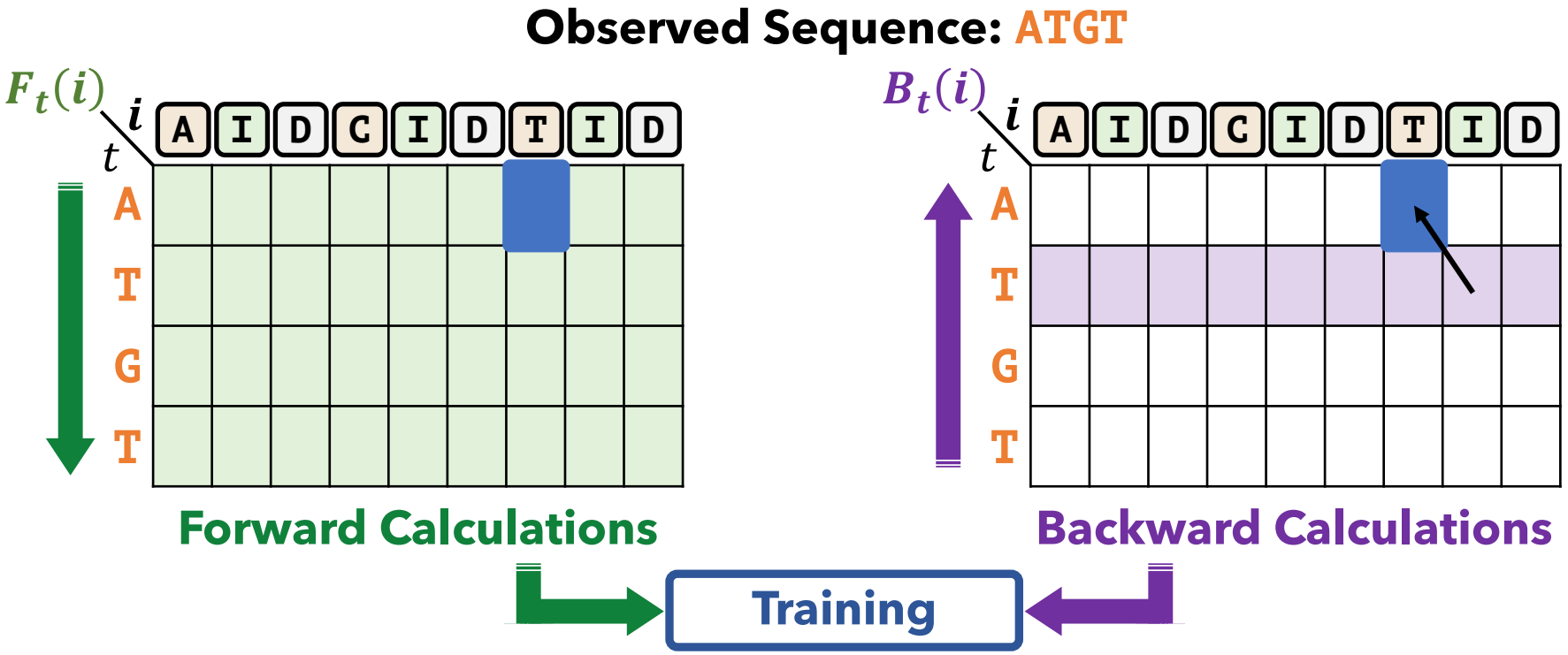
SW: Minimizing Redundant Storage

- **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**



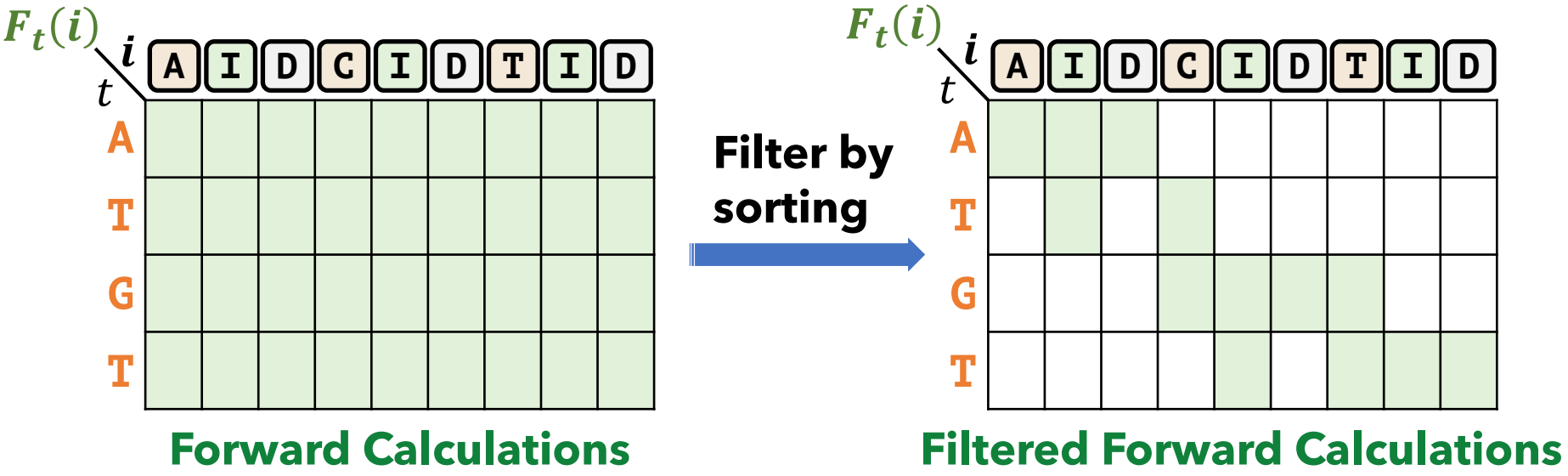
SW: Minimizing Redundant Storage

- **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



SW: Reducing Unnecessary Computations

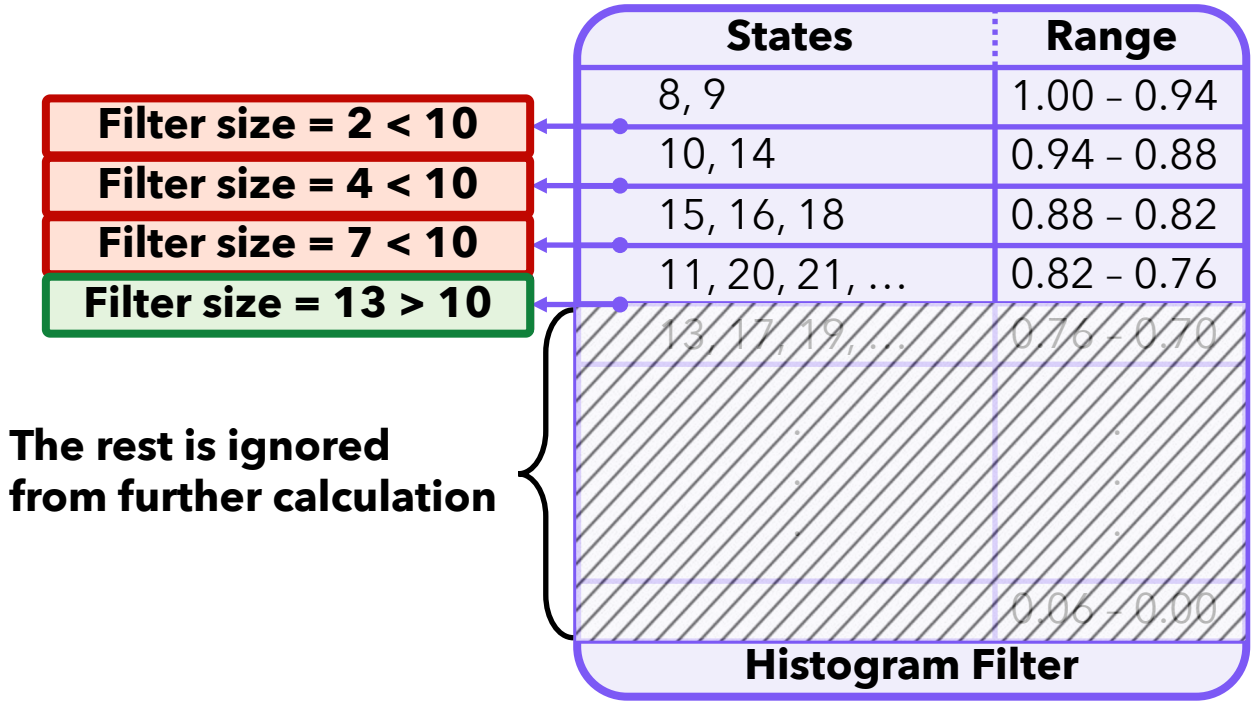
- **Observation:** 'Negligible' cells can be ignored without significantly reducing overall accuracy
  - **Filtering:** Non-negligible states are identified by sorting
  - **Sorting** to find **exactly** n states with **largest** Forward or Backward values



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

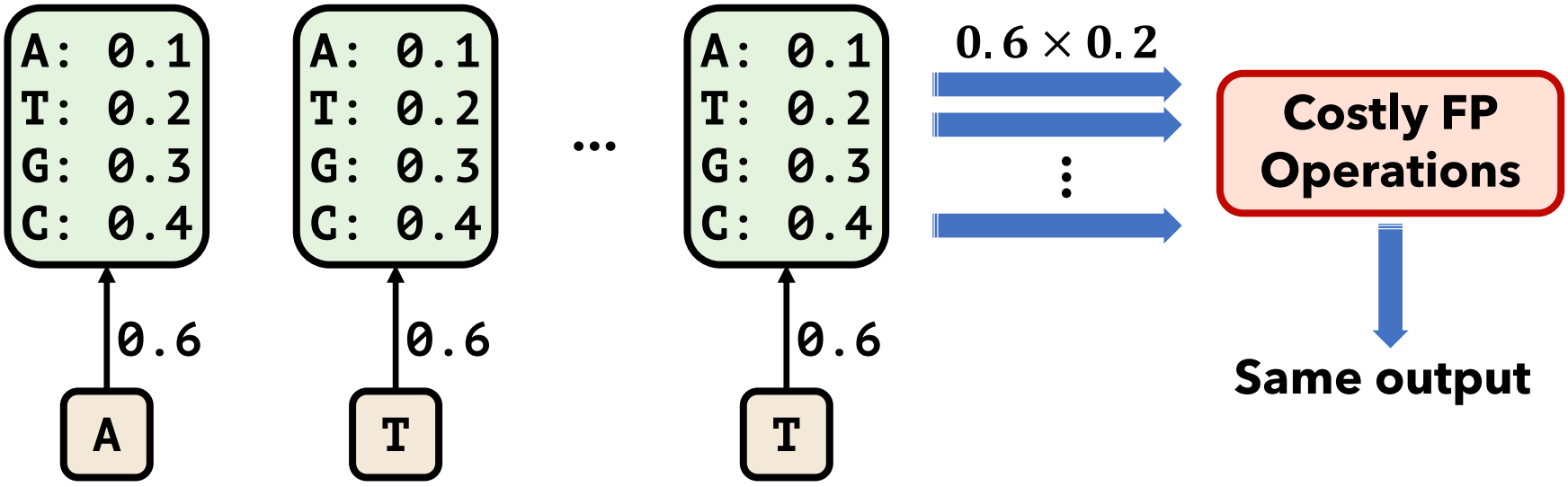
SW: Reducing Unnecessary Computations

- **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

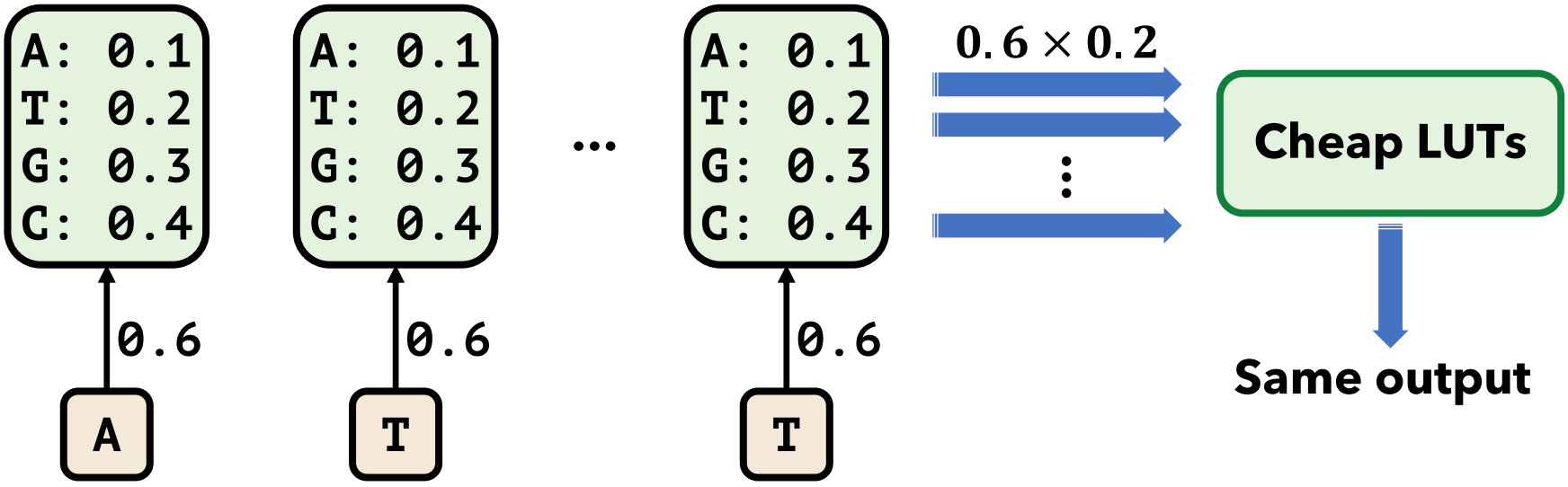
- **Observation:** Same multiplications are redundantly performed
  - **Same default values are used** for each possible connection in pHMMs
  - **Fixed connection patterns** generate a fixed set of multiplication results



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

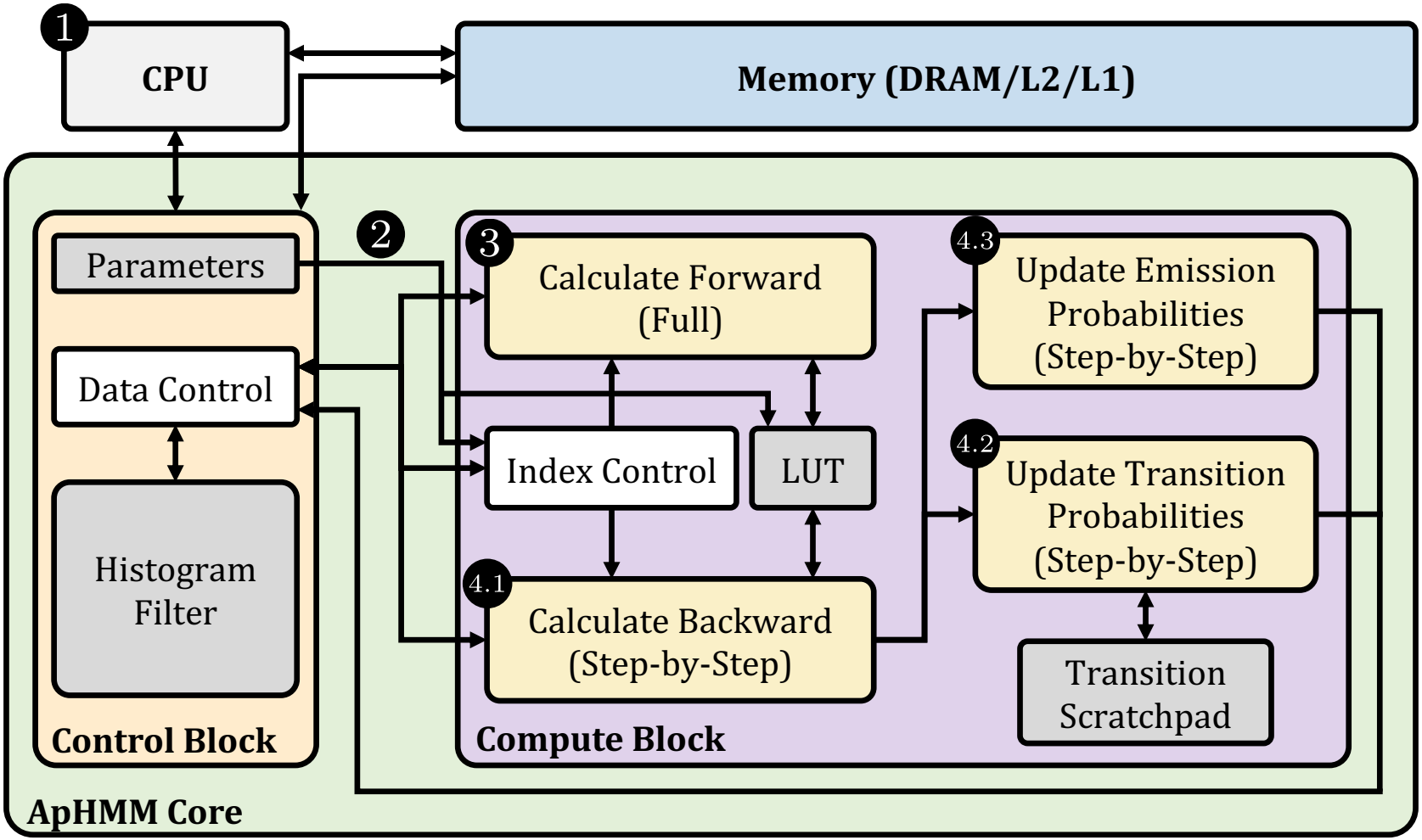
SW: Avoiding Repeated Operations

- **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

Overview of ApHMM Design



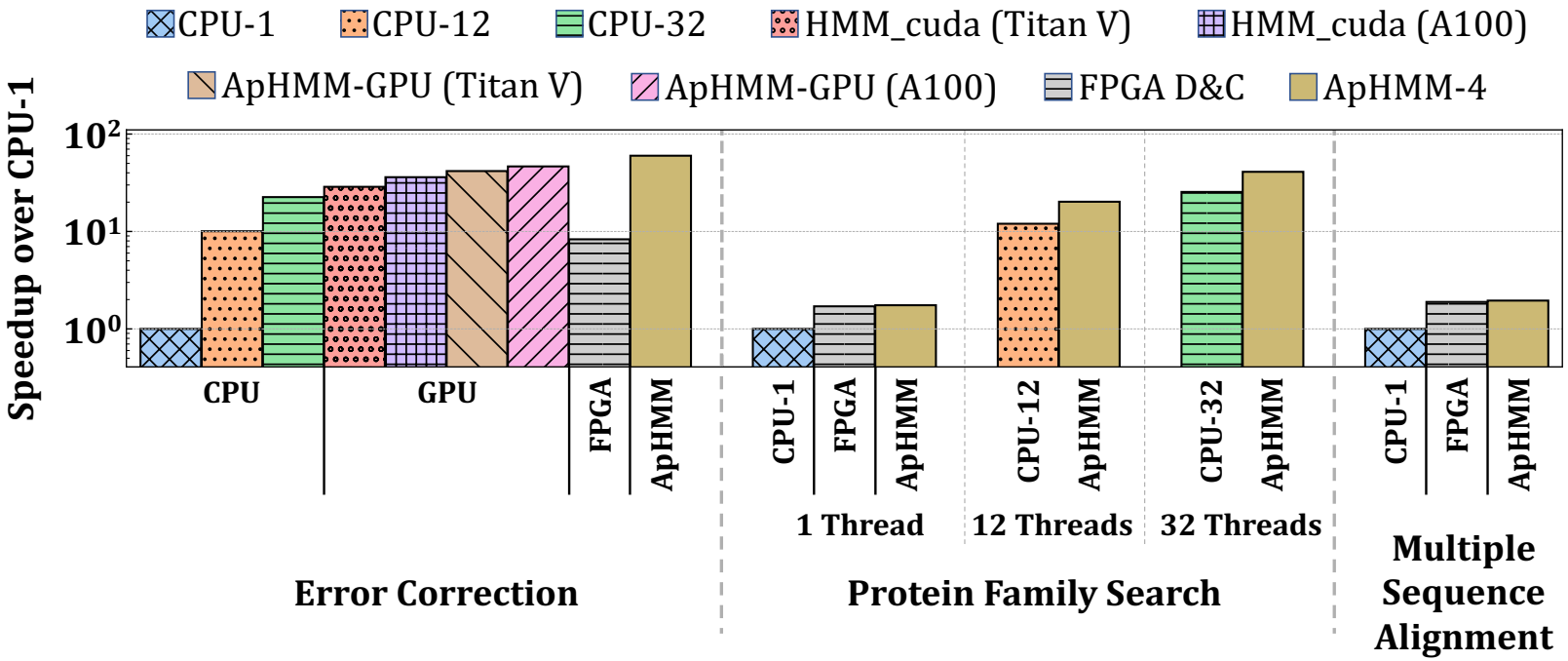
✓ **Flexible and efficient control logic & hardware design**

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: Workload Acceleration



Up to 60× (CPU-1), 1.75× (GPU), and 1.95× (FPGA) faster execution of the end-to-end applications

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

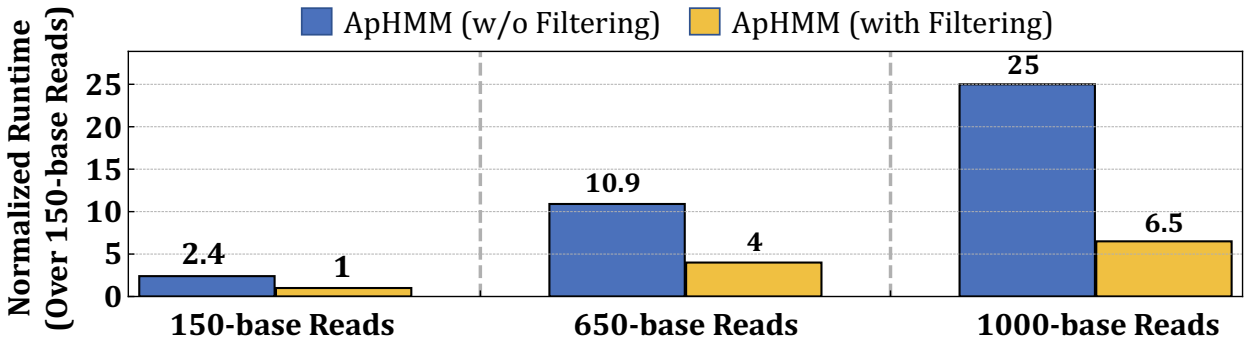
More in the Paper

- **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

Filtering – Performance Benefits

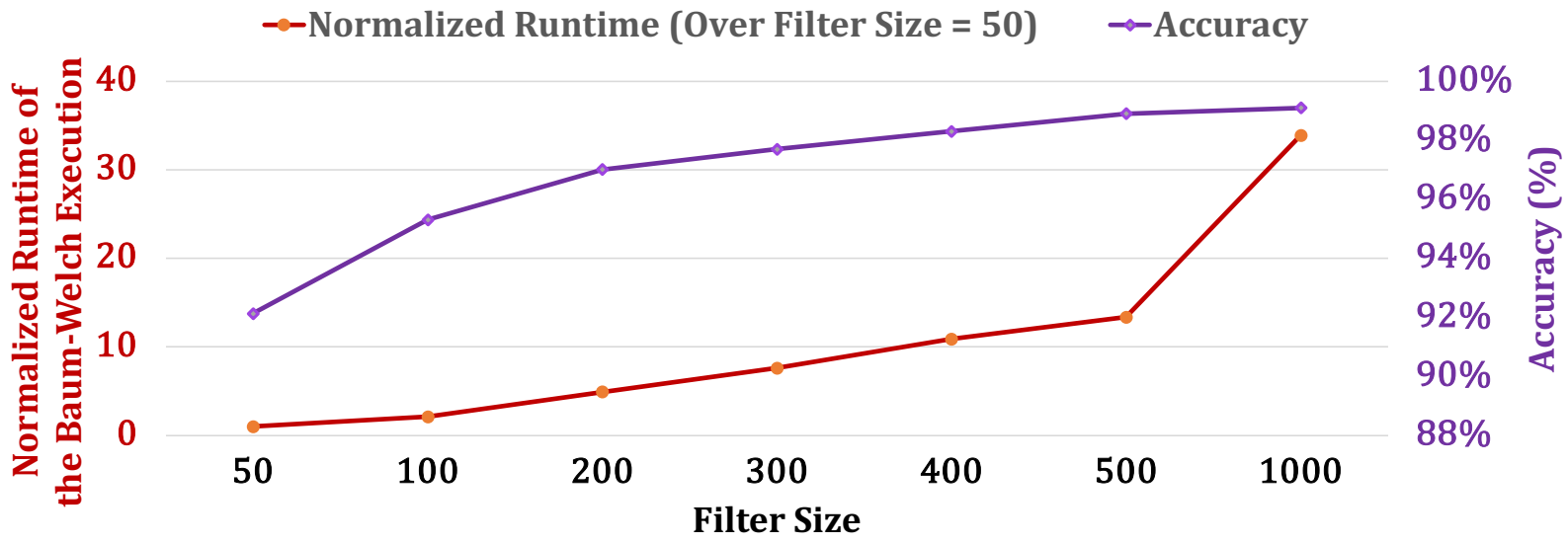
- Filtering heuristics aim to reduce unnecessary computations



Motivational Study: ~2.5x performance improvements with filtering

Filtering – Accurate but Costly Sorting

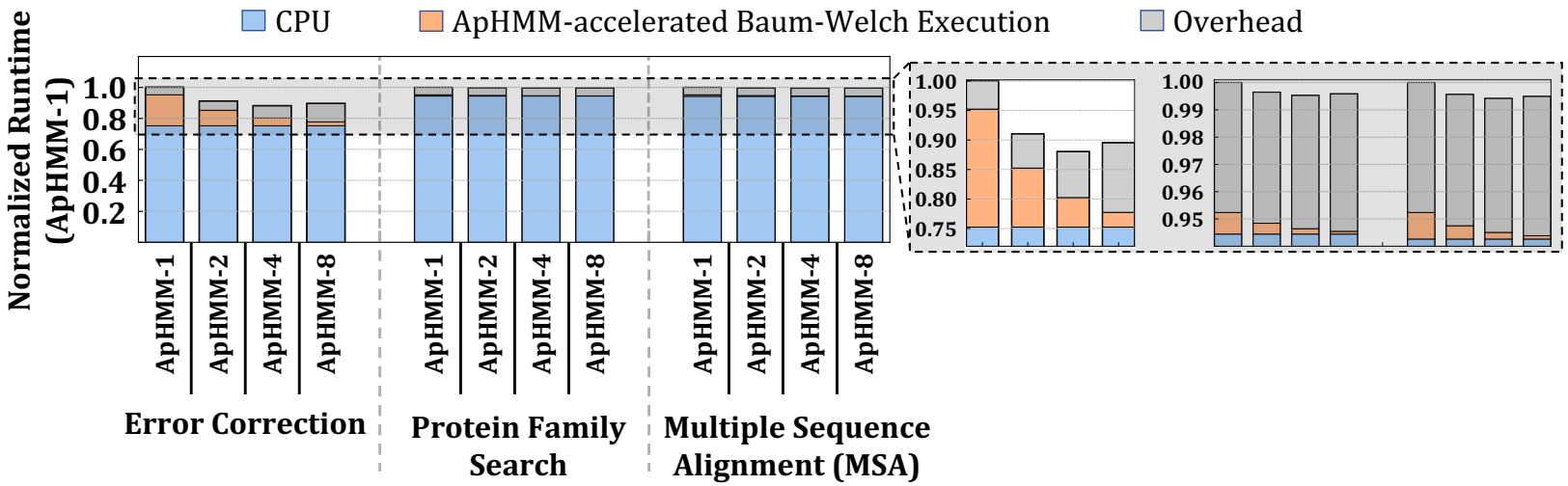
- Software-based filtering heuristics aim to reduce unnecessary computations
  - High-accuracy can be achieved with filtering with correct setting



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