Enabling Fast, Accurate, and Efficient Real-Time Genomic Sequence Analysis via New Algorithms and Architectures

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

13 September 2024
AMD

SAFARI

ETH zürich

Brief Self Introduction

- **Can Firtina**
  - Senior Ph.D. student in the SAFARI Research Group and a lecturer at ETH Zurich

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

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

Professor Mutlu

- Onur Mutlu
  - Full Professor @ ETH Zurich ITET (INFK), since September 2015
  - Strecker Professor @ Carnegie Mellon University ECE/CS, 2009-2016, 2016-…
  - PhD from UT-Austin, worked at Google, VMware, Microsoft Research, Intel, AMD
  - https://people.inf.ethz.ch/omutlu/
  - omutlu@gmail.com (Best way to reach)
  - https://people.inf.ethz.ch/omutlu/projects.htm
- Research and Teaching in:
  - Computer architecture, computer systems, hardware security, bioinformatics
  - Memory and storage systems
  - Hardware security, safety, predictability
  - Fault tolerance
  - Hardware/software cooperation
  - Architectures for bioinformatics, health, medicine
  - …

Four Key Current Directions at SAFARI

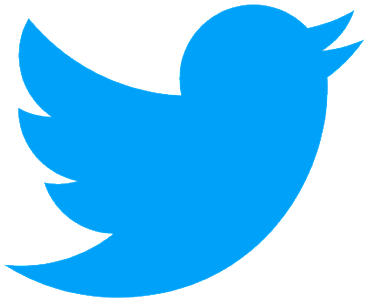
- Fundamentally Secure/Reliable/Safe Architectures
- Fundamentally Energy-Efficient Architectures
  - Memory-centric (Data-centric) Architectures
- Fundamentally Low-Latency and Predictable Architectures
- Algorithms & Architectures for AI/ML, Genomics, Medicine

Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
- Enabling Scalable Real-time Genome Analysis
- Graph & ML Acceleration for Genomics
- Conclusion

Big Data is Everywhere

Astronomy
25 zetta-bytes/year



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

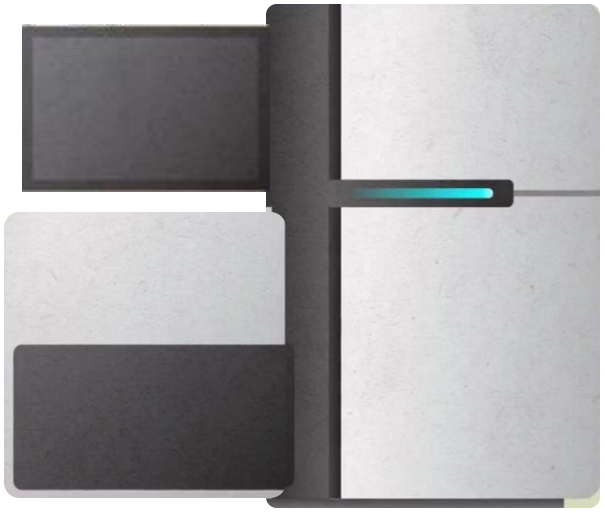
YouTube
500-900 million hours/year

Genomics
1 zetta-bases/year

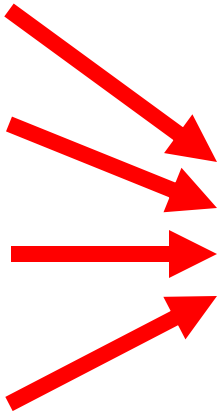
"Big data: astronomical or genomical?", PLoS biology, 2015.

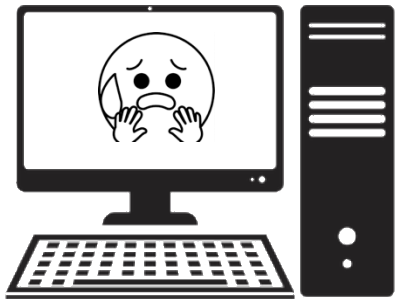
We need to gain insights
and observations
much more efficiently
than ever before

Problems with Data Analysis Today



Special-Purpose Machine
for **Data Generation**





General-Purpose Machine
for **Data Analysis**

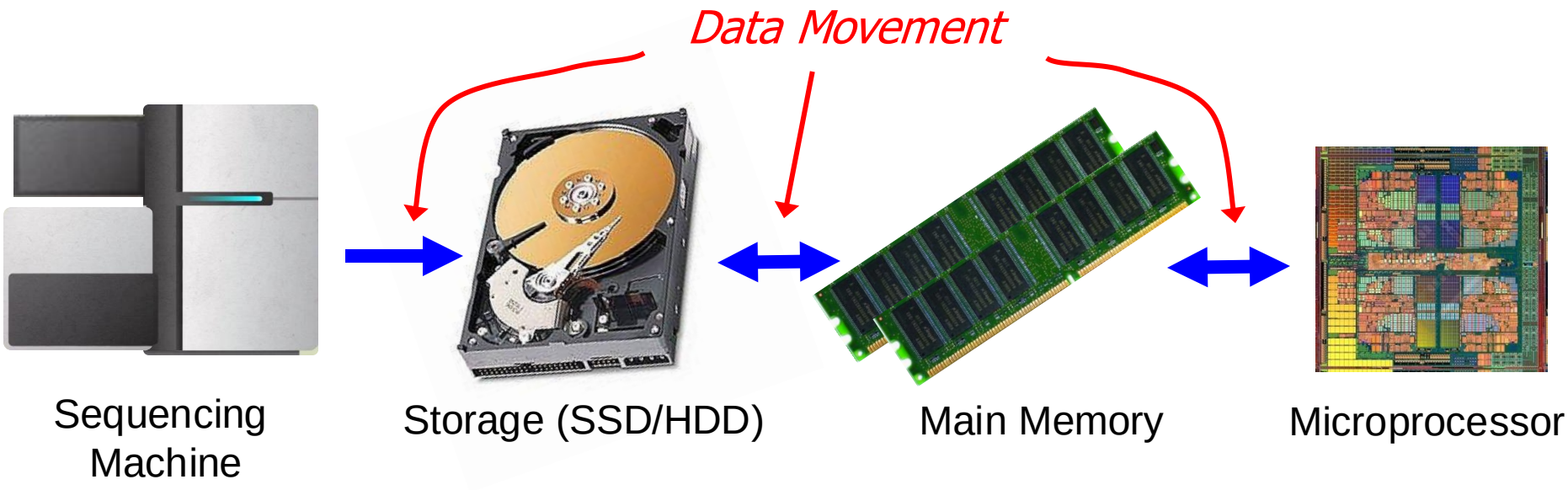
FAST

SLOW

Slow and inefficient processing capability
Large amounts of data movement

Data Movement Dominates Performance

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

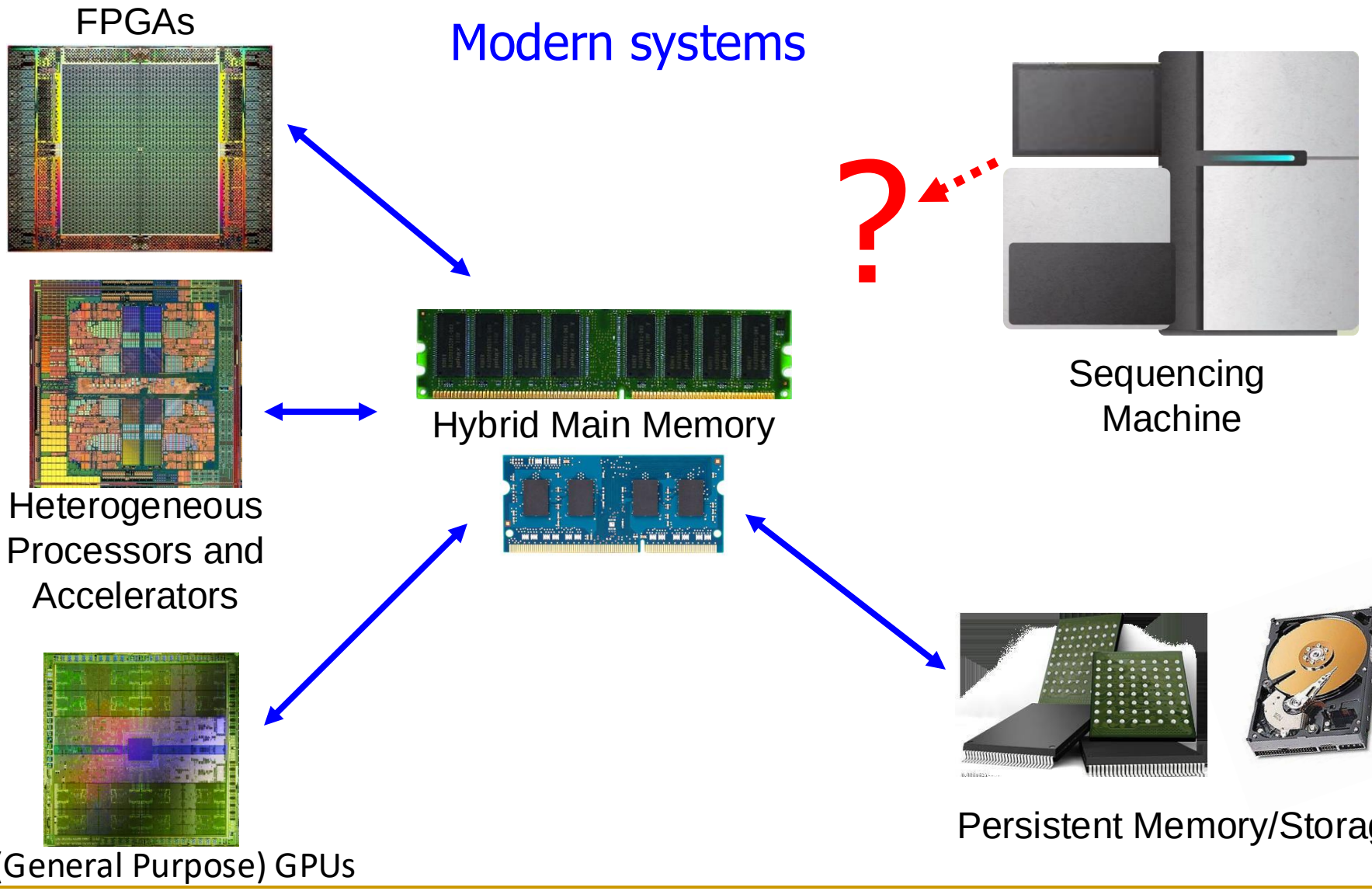


Single memory request consumes >160x-800x more energy compared to performing an addition operation

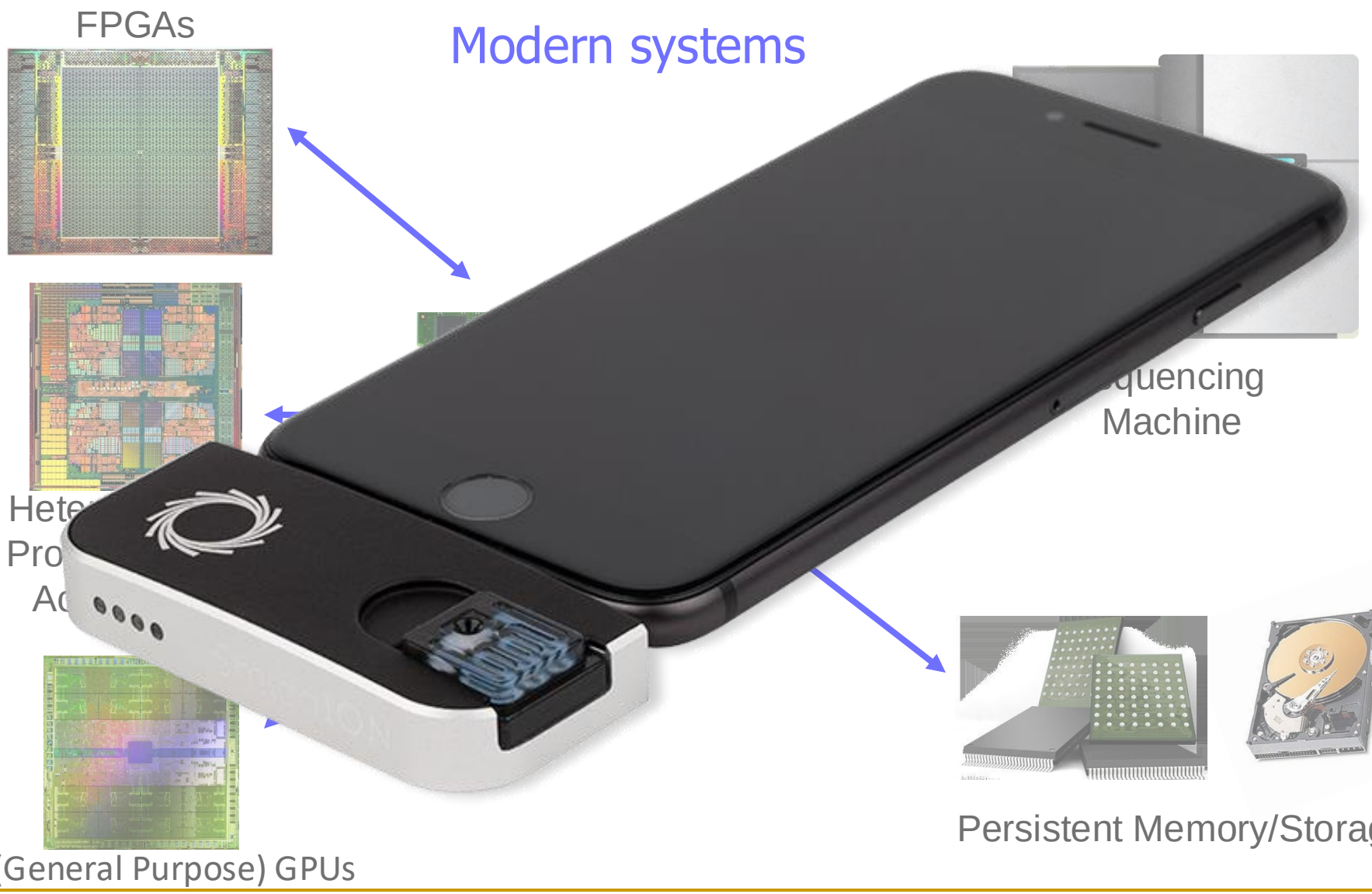
* Boroumand et al., "Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks," ASPLOS 2018
⋆ Kestor et al., "Quantifying the Energy Cost of Data Movement in Scientific Applications," IISWC 2013
☆ Pandiyan and Wu, "Quantifying the energy cost of data movement for emerging smart phone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

We need intelligent algorithms and intelligent architectures that handle data well

Pushing Towards New Architectures



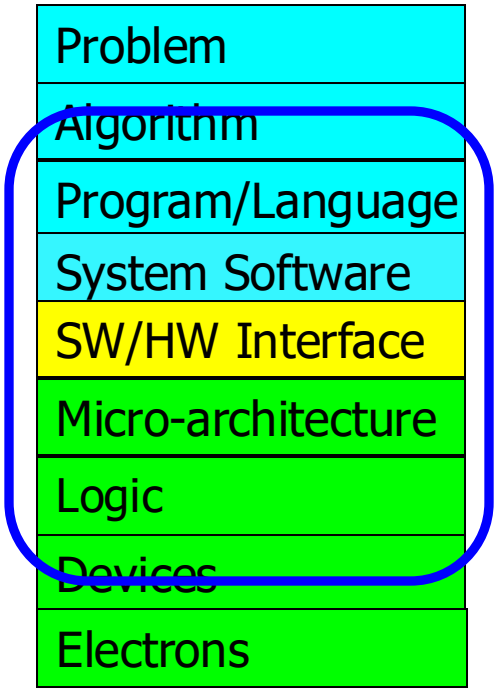
Pushing Towards New Architectures



https://nanoporetech.com/products/smidgion

Algorithm-Arch-Device Co-Design is Critical

Computer Architecture (expanded view)



Accelerating Genome Analysis [DAC 2023]

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

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**)*, Jul 2023
  [Bioinformatics Journal 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, 30 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, Joël Lindegger, Yan Zhu, Andre Kahles, and Onur Mutlu, **"Rawsamble: Overlapping and Assembling Raw Nanopore Signals using a Hash-based Seeding Mechanism"**
  Presented at ISMB (HiTSeq), Montreal, QC, Canada, Jul. 2024.
  [Slides (pptx) (pdf)]
  [Rawsamble 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] Joël Lindegger[1] Yan Zhu[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*

New Directions: 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"
  *Preprint on **arXiv***, October 2023.
  [arXiv version]
  [RawAlign Source Code]

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

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

[§]*ETH Zürich*

Genome Similarity Identification [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 preprint]
  [bioRxiv preprint]
  [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 preprint]
  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. '24]

- Jeremie S. Kim, Can Firtina, Meryem Banu Cavlak, Damla Senol Cali, Can Alkan, Onur Mutlu,
  "FastRemap: A Tool for Quickly Remapping Reads between Genome Assemblies"
  Bioinformatics, 1 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 Machine Learning in Genome Analysis **via New Algorithms and Architectures**

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]
  [Journal Article]
  [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 Raw Signal Translation

- M. Banu Cavlak, Gagandeep Singh, Mohammed Alser, Can Firtina, Joel Lindegger, Mohammad Sadrosadati, Nika Mansouri Ghiasi, Can Alkan, and Onur Mutlu, **"TargetCall: Eliminating the Wasted Computation in Basecalling via Pre-Basecalling Filtering"**
  *Proceedings of the 21st Asia Pacific Bioinformatics Conference (**APBC**)*, Changsha, China, April 2023.
  [arxiv.org 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] Onur Mutlu[1]
[1]*ETH Zürich* [2]*Bilkent University*

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.
  [Source Code]
  [Online link at Bioinformatics Journal]

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

FREE

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

Bioinformatics, Volume 36, Issue 12, 15 June 2020, Pages 3669–3679,
https://doi.org/10.1093/bioinformatics/btaa179
Published: 13 March 2020 **Article history ▾**

Accelerating 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***), February 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

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*

Raw Signal Translation 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, & 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 2022]

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

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

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

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

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

In-Storage Metagenomics [ISCA 2024]

- 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

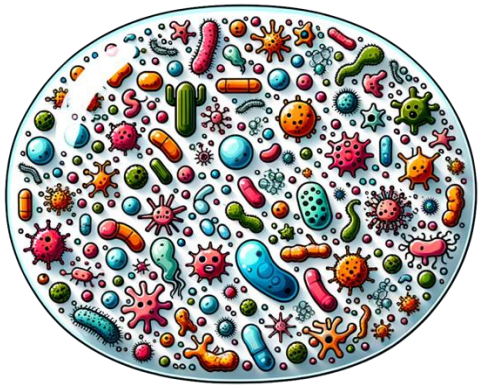
Nika Mansouri Ghiasi[1] Mohammad Sadrosadati[1] Harun Mustafa[1] Arvid Gollwitzer[1]
Can Firtina[1] Julien Eudine[1] Haiyu Mao[1] Joël Lindegger[1] Meryem Banu Cavlak[1]
Mohammed Alser[1] Jisung Park[2] Onur Mutlu[1]
[1]ETH Zürich [2]POSTECH

Accelerating Genome Analysis [DAC 2023]

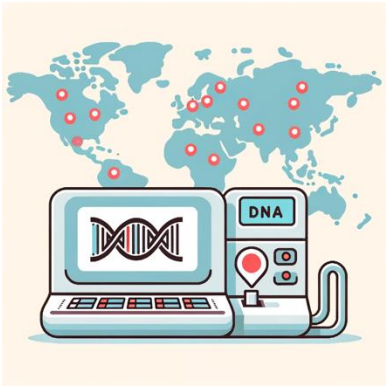
- Onur Mutlu and Can Firtina,
  "Accelerating Genome Analysis via Algorithm-Architecture Co-Design"
  *Invited Special Session Paper in Proceedings of the 60th Design Automation Conference (**DAC**)*, San Francisco, CA, USA, July 2023.
  [Related Invited Paper]
  [arXiv version]
  [Slides (pptx) (pdf)]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

Genomic Sequence Analysis – Why?

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



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



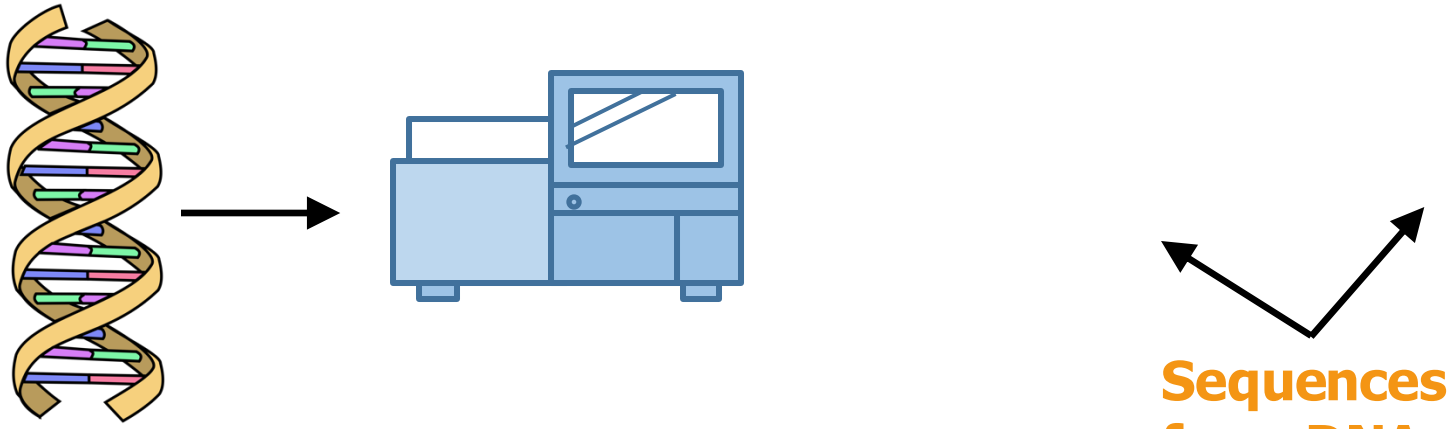
Surveillance of **disease outbreaks**

Personalized medicine

And, many, many other applications …

Genomic Sequence Analysis – How?

- **High throughput sequencing machines**
  - Convert biological molecules into DNA characters for analysis



Biological Molecule (e.g., DNA)

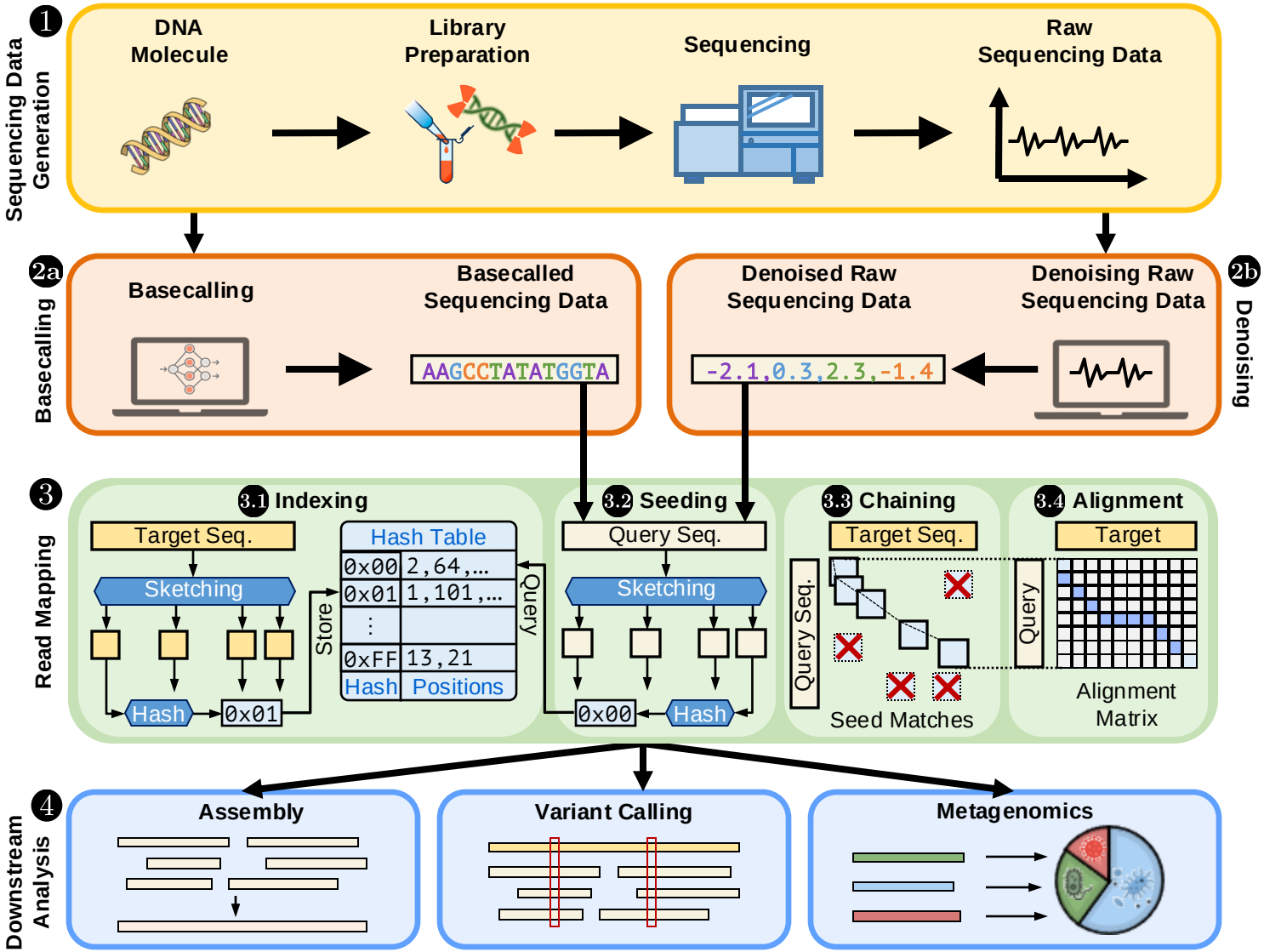
Sequences from DNA (Reads)

Challenge 1: Origin locations of sequences are unknown

Challenge 2: Sequences may contain differences and errors **(noise)**

Challenge 3: Many sequences to analyze

A Genome Analysis Pipeline



Recall: Challenges in Genome Analysis

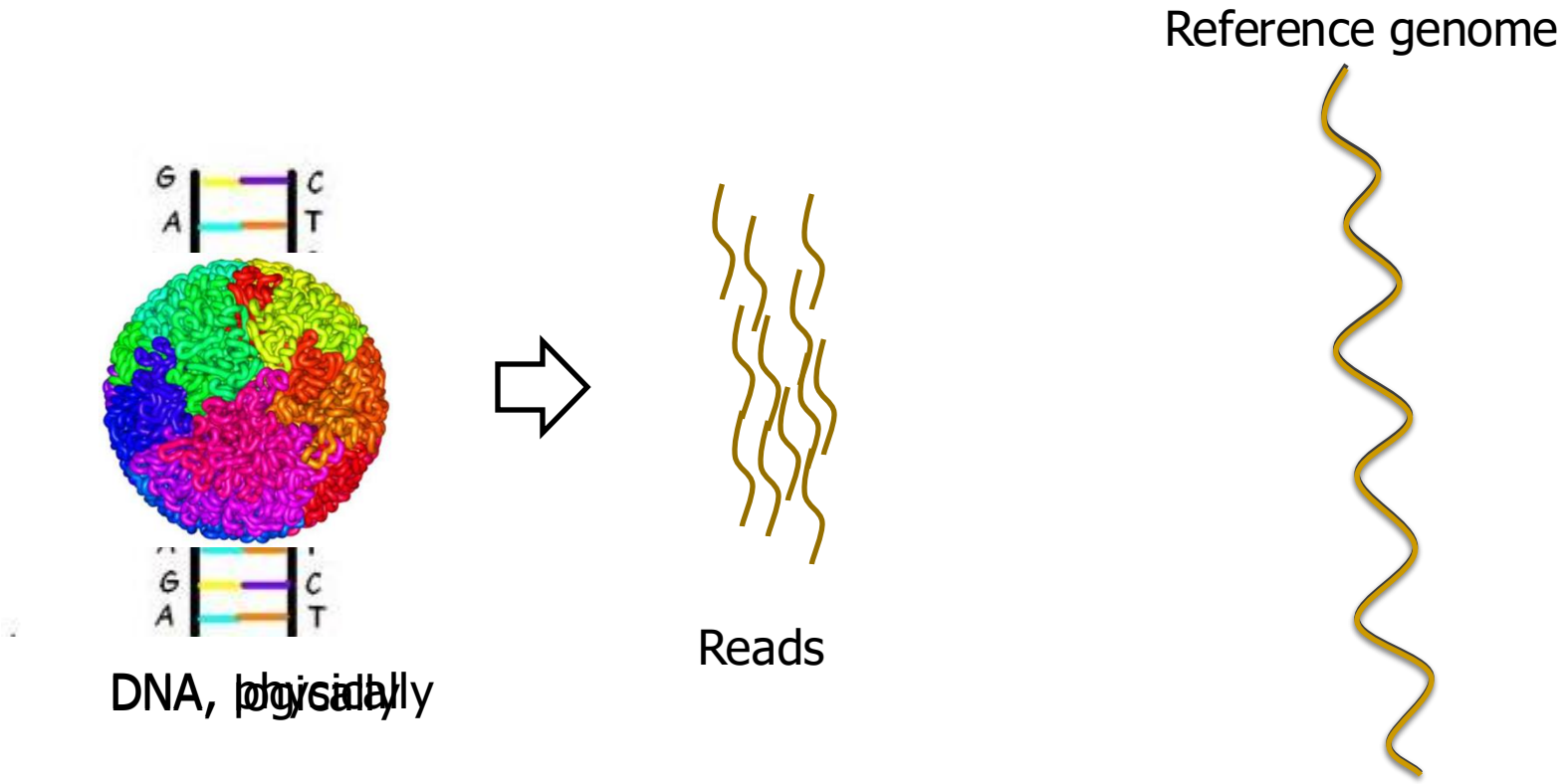
Challenge 1: Origin locations of sequences are unknown

Challenge 2: Sequences may contain differences and errors **(noise)**

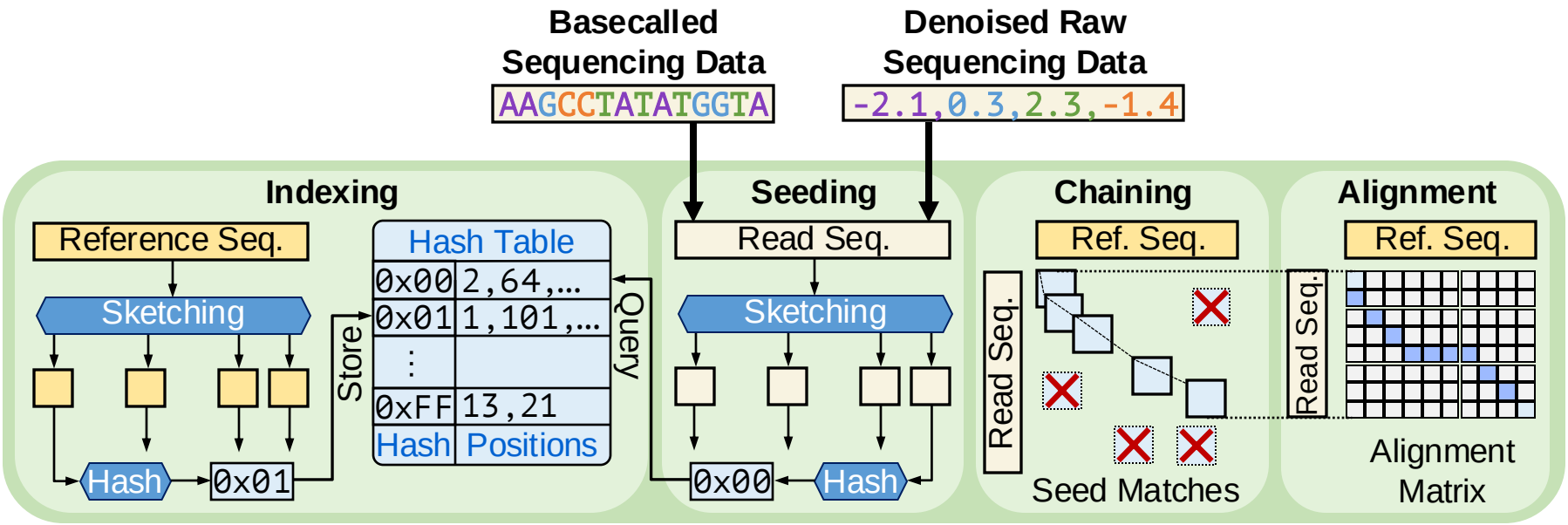
Challenge 3: Many sequences to analyze

Pairwise Sequence Comparison is Essential

- **Read mapping:** Map many reads to a known reference genome with some differences allowed



Practical Read Mapping



Accurate, fast, and scalable

genome analysis is critical

to **make life-critical decisions**

and improving the quality of life

Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
- Enabling Scalable Real-time Genome Analysis
- Graph & ML Acceleration for Genomics
- Conclusion

RawHash

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

Can Firtina

Nika Mansouri Ghiasi Joel Lindegger Gagandeep Singh

Meryem Banu Cavlak Haiyu Mao Onur Mutlu

RawHash

RawHash2

Code

SAFARI

ETH zürich

Outline

Background

RawHash

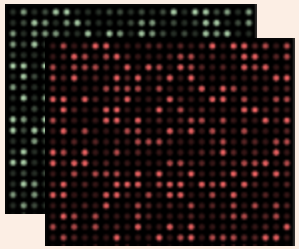
RawHash2

Evaluation

Conclusion

Different Raw Sequencing Data

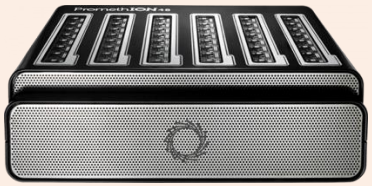
Illumina

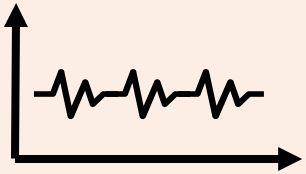


Multiple images

.BCL/.CBCL

Nanopore

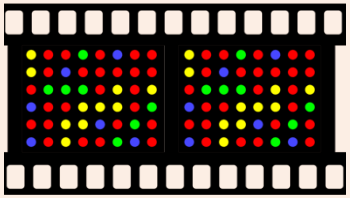




Electrical Signal

.POD5

PacBio



30-hour movie

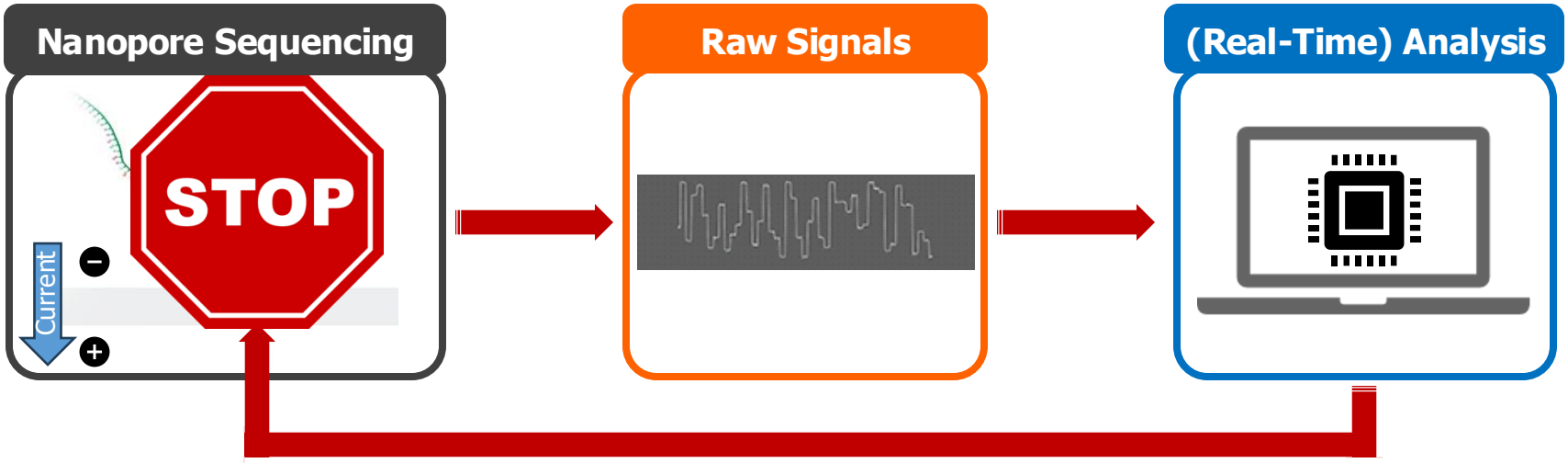
.BAM

Nanopore Sequencing

Nanopore Sequencing: a widely used sequencing technology

- Can sequence **large fragments** of nucleic acid molecules (>2Mbp)
- **Portable** sequencing and **real-time analysis**
- **Cost-effective**

Nanopore Sequencing



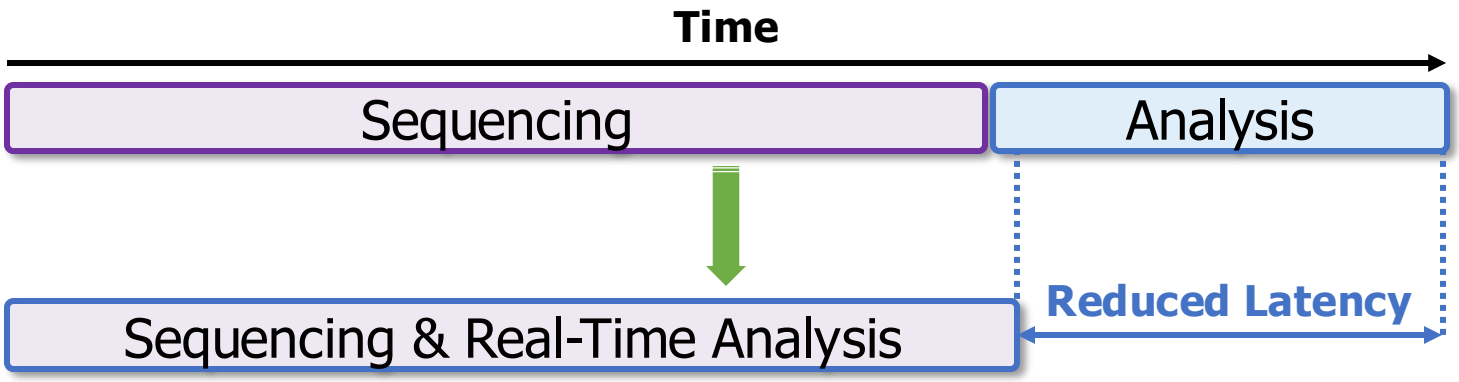
Raw Signals: Ionic current measurements generated at a certain **throughput**

(Real-Time) Analysis: Signals can be analyzed while they are generated

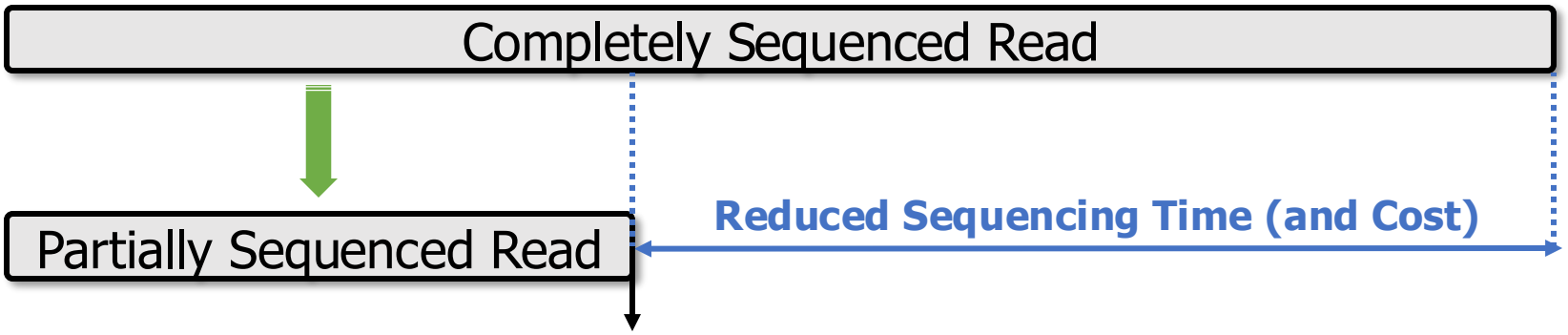
Real-Time Decisions: Stopping sequencing **early** based on real-time analysis

Benefits of Real-Time Analysis

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



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



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

Outline

Background

RawHash

RawHash2

Evaluation

Conclusion

Executive Summary

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

Goal: Enable **fast** and **accurate** real-time analysis of raw nanopore signals

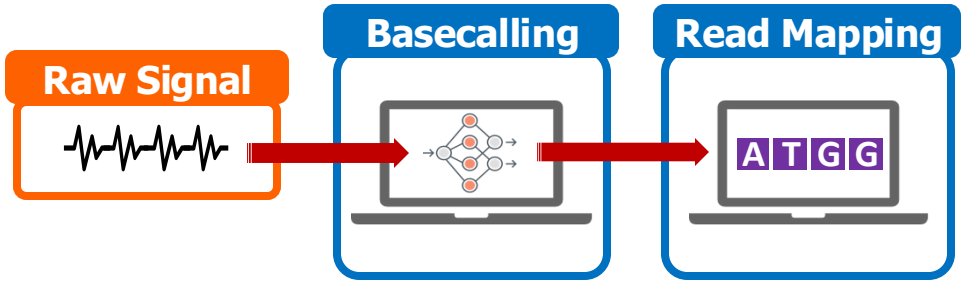
Key Contributions:
1) The **first hash-based mechanism** for mapping raw nanopore signals

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

Key Results: Across 3 use cases and 5 genomes of varying sizes
- **27× 19×, and 4× better average throughput** compared to the state-of-the-art works
- **Most accurate raw signal mapper for all datasets**
- Sequence Until **reduces the sequencing time and cost by 15×**

Analyzing Raw Nanopore Signals

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

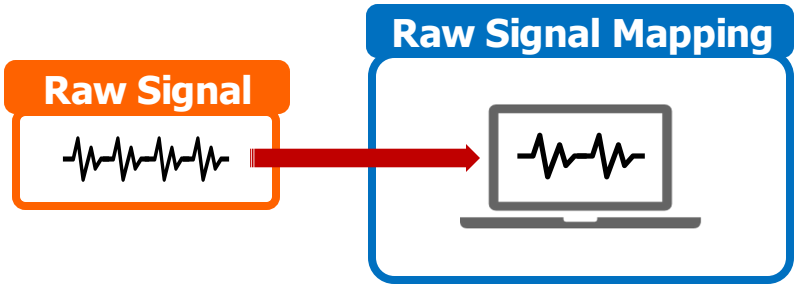


✓ Basecalled sequences are less noisy than raw signals

✓ Many analysis tools use basecalled sequences

✗ Costly and power-hungry computational requirements

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

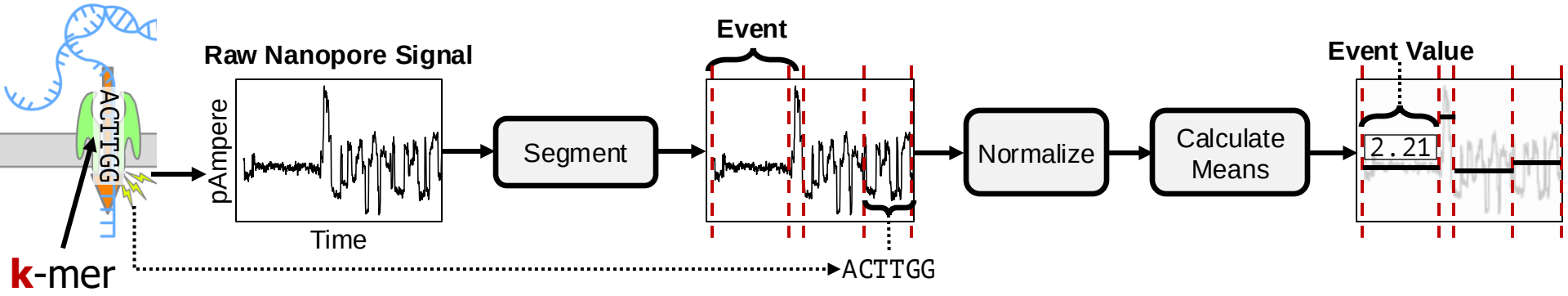


✓ Efficient analysis with better scalability and portability

✓ Raw signals retain more information than just bases

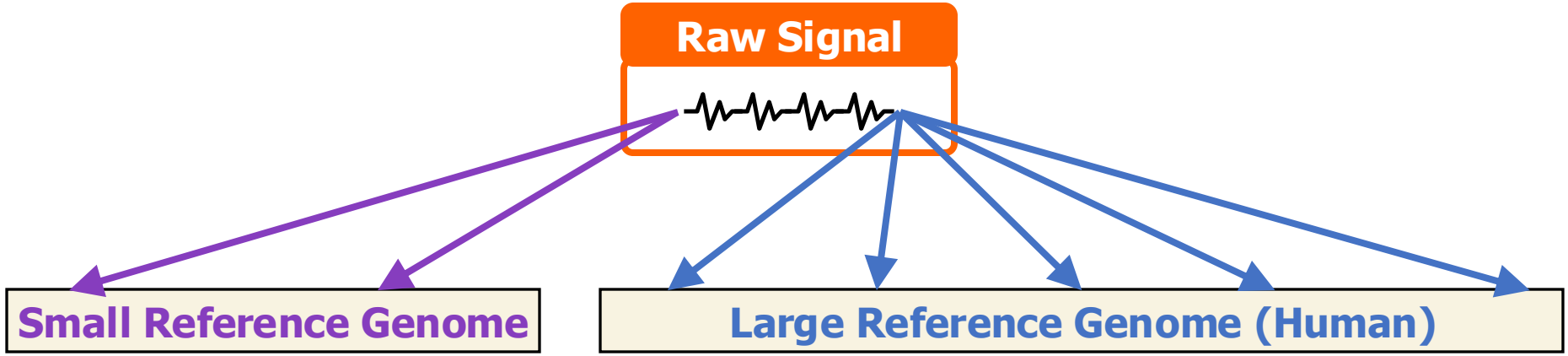
Enabling Analysis From Electrical Signals

- 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)

The Problem – Mapping Raw Signals



Fewer candidate regions in **small genomes**

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

Accurate mapping

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

High throughput

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

The Problem – Mapping Raw Signals

Prior works are
inaccurate or inefficient
for large genomes

Accurate mapping

on **many regions → inaccurate mapping**

High throughput

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

Goal

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

RawHash

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

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

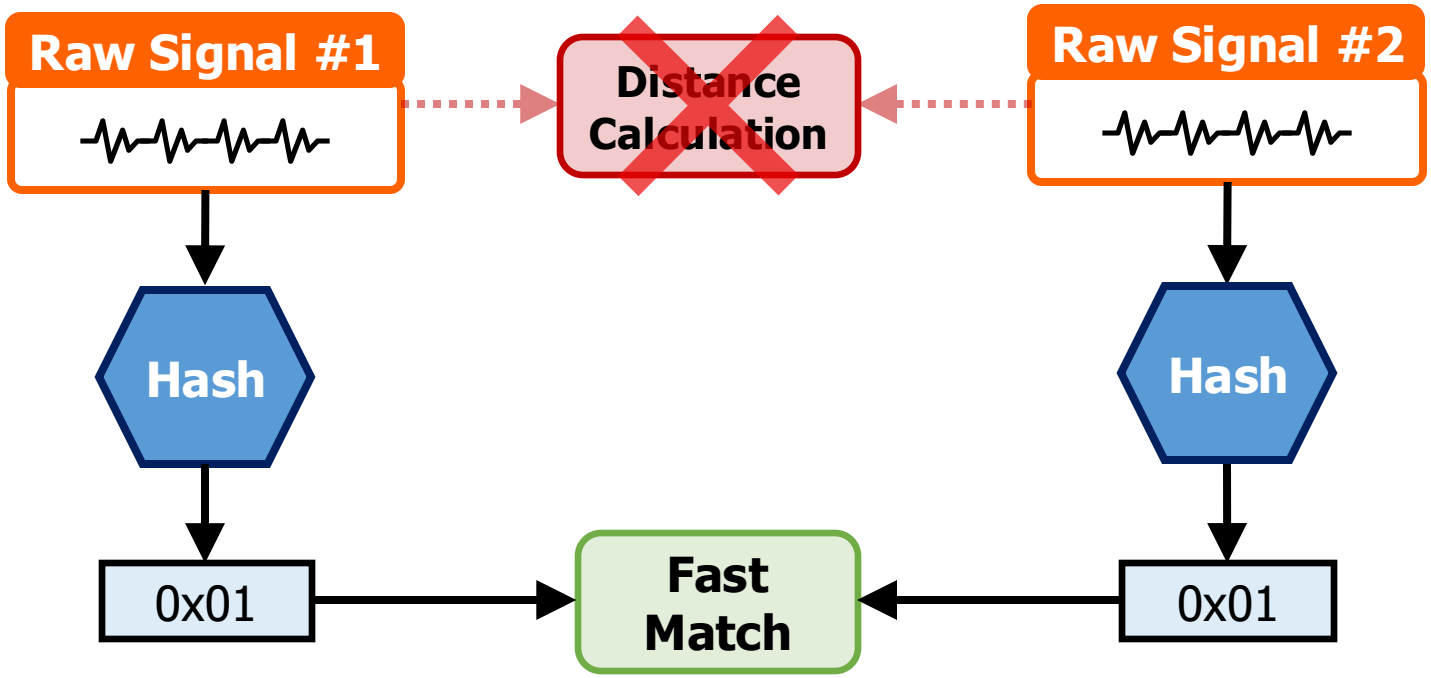
RawHash

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

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

RawHash – Key Idea

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



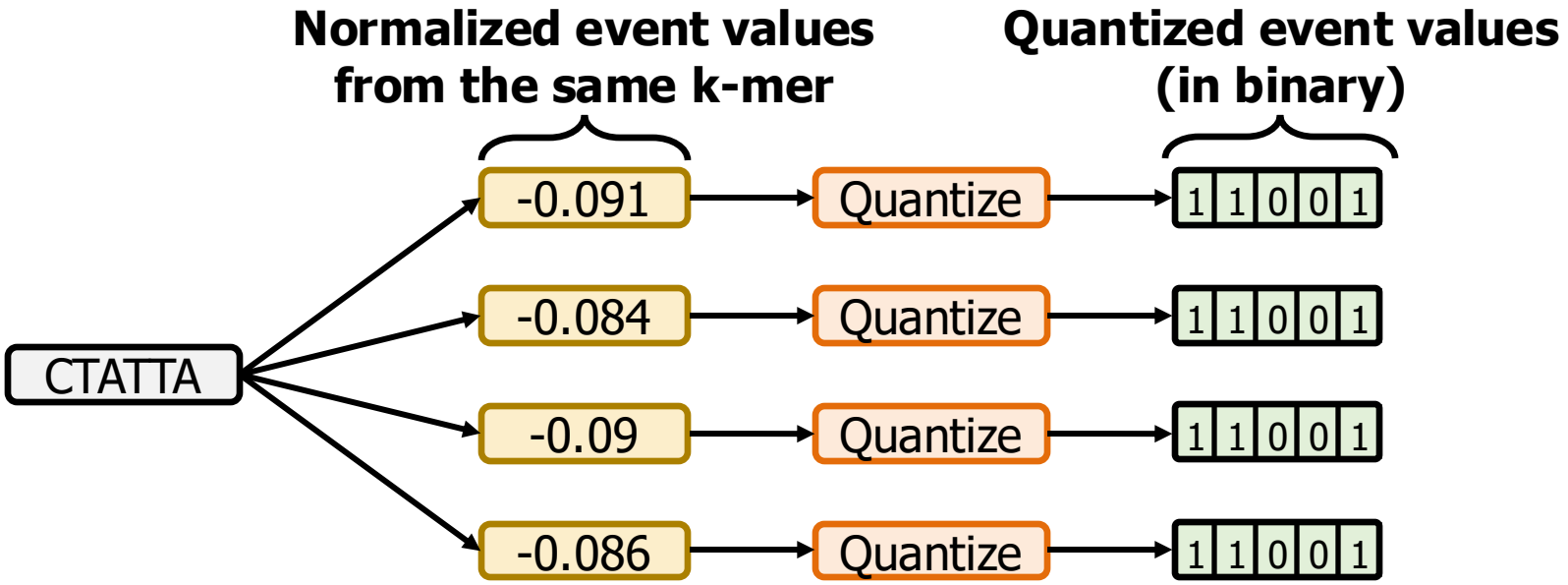
Challenge #1: Generating the **same** hash value for **similar enough** signals

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

Solution for Challenge #1: Quantization

- **Noise:** Slight differences in raw signals from identical k-mers
  - **Challenge:** Generating the **same** hash value for **similar enough** signals

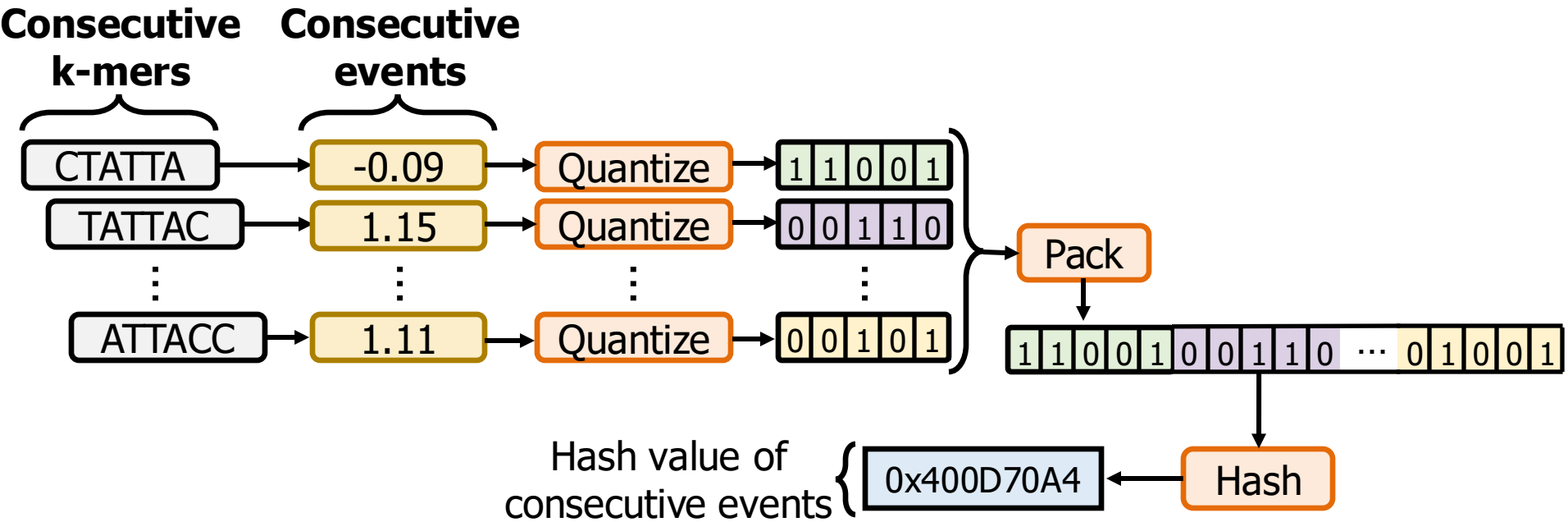
- **Key Idea:** Quantize the event values **to reduce noise**
  - Enables assigning **identical quantized values** to **similar event values**



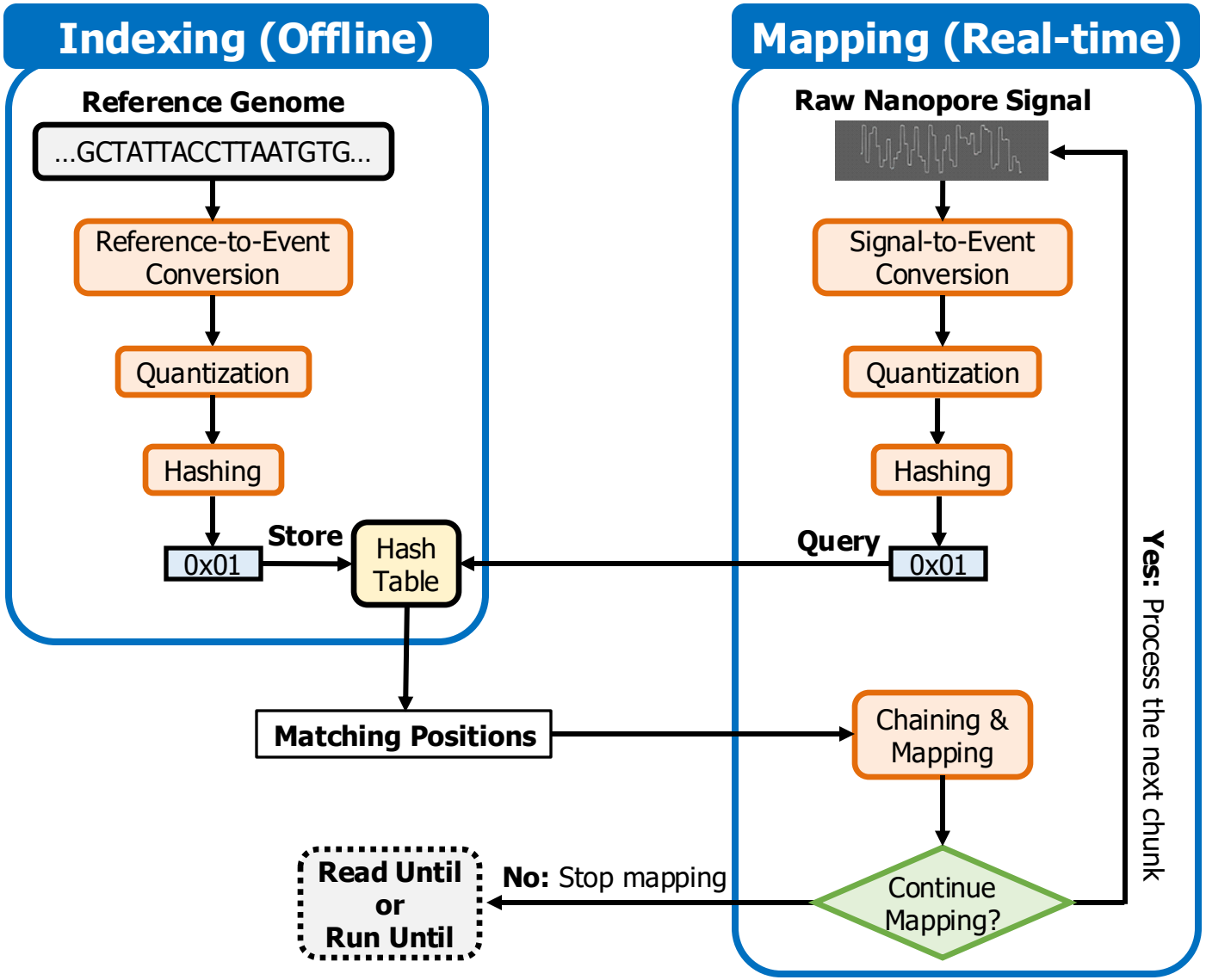
Solution for Challenge #2: Packing

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

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



Real-Time Mapping using RawHash



RawHash

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

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

RawHash

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

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

Outline

Background

RawHash

RawHash2

Evaluation

Conclusion

Key Contributions in RawHash2

A new adaptive quantization to better fit the expected nanopore signal pattern to achieve **high accuracy**

Improved chaining algorithm with sensitive penalty scores

Weighted decision making for more robust mapping

Frequency filter and **minimizer sketching** to reduce seed matches for faster and space-efficient mapping

Outline

Background

RawHash

RawHash2

Evaluation

Conclusion

Evaluation Methodology

- Two settings for RawHash2:
  - **RawHash2:** All hash values without sampling
  - **RawHash2-Minimizer:** Minimizer sketching

- Compared to **UNCALLED** [Kovaka+, Nat. Biotech. '21], **Sigmap** [Zhang+, ISMB '21] and **RawHash** [Firtina+, ISMB '23]

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

Evaluation Methodology

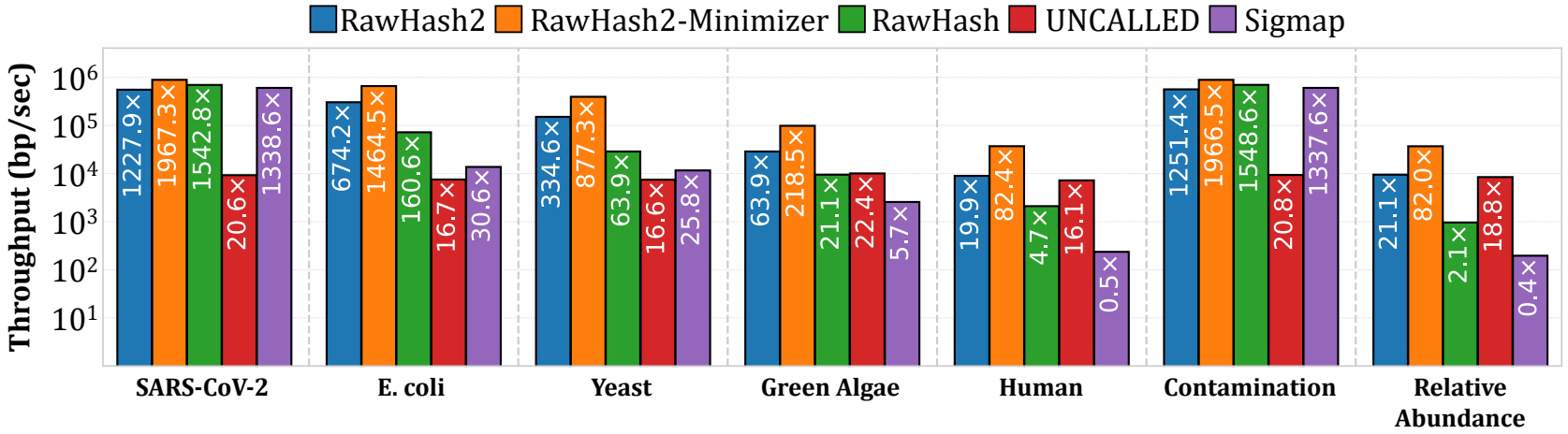
- Evaluation metrics:
  - **Throughput** (bases processed per second per CPU thread)
  - Potential reduction in **sequencing time and cost**
  - **Accuracy**
    - **Baseline:** Mapping basecalled reads using minimap2
    - Precision, recall, and F1 scores
    - Relative abundance estimation distance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

- **Datasets:**

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

Throughput

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

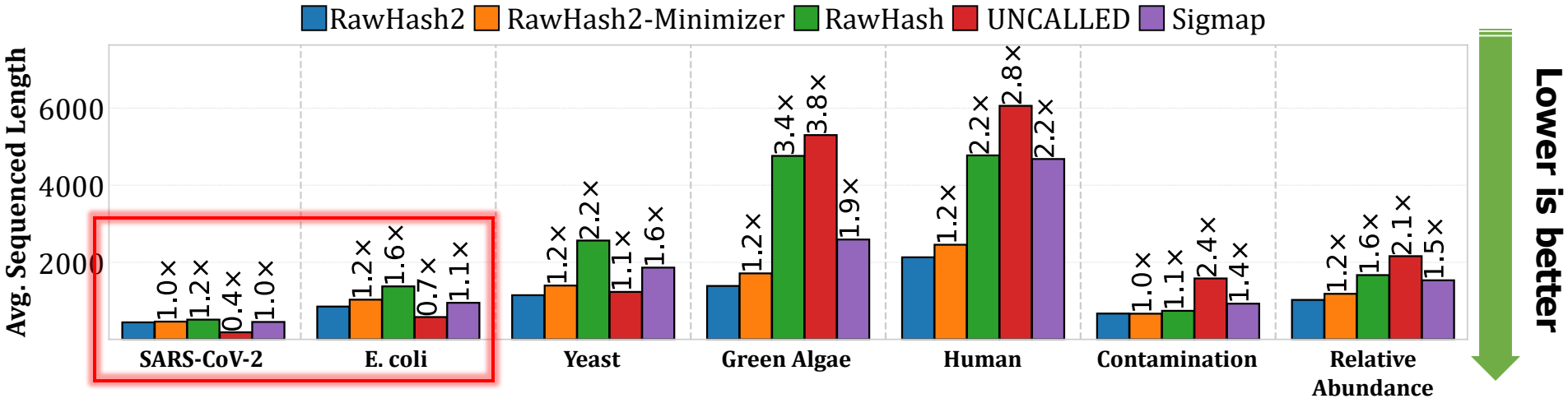


RawHash2: 27×, 19×, and 4× better average throughput compared to **UNCALLED, Sigmap** and **RawHash**, respectively

RawHash2-Minimizer further improves the throughput by **2.5×** compared to **RawHash2**

Average Sequenced Length

- Fewer bases to sequence ➔ Less unnecessary sequencing



RawHash2 **reduces sequencing time and cost**

on average by **1.9**× compared to UNCALLED and RawHash

RawHash2 leads to **sequencing the least amount of bases**

for larger genomes

Accuracy

- Read mapping, contamination, and relative abundance estimation accuracy (baseline: basecalled mapping)

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

Best results are **highlighted**.

RawHash2 provides the **most accurate read mapping**

RawHash2-Minimizer provides an **on-par accuracy with RawHash2** while improving **the throughput substantially**

Outline

Background

RawHash

RawHash2

Evaluation

Conclusion

Conclusion

Key Contributions:
1) The **first hash-based mechanism** for mapping raw nanopore signals
2) The novel **Sequence Until** technique can accurately and **dynamically stop the entire sequencing of all reads at once** if further sequencing is not necessary

Key Results: Across 3 use cases and 5 genomes of varying sizes
- **27× 19×, and 4× better average throughput** compared to the state-of-the-art works
- **Most accurate raw signal mapper for all datasets**
- Sequence Until **reduces the sequencing time and cost by 15×**

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

RawHash

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

Can Firtina

Nika Mansouri Ghiasi Joel Lindegger Gagandeep Singh

Meryem Banu Cavlak Haiyu Mao Onur Mutlu

RawHash

RawHash2

Code

SAFARI

ETH zürich

Enabling Many New Directions with Raw Signal Analysis

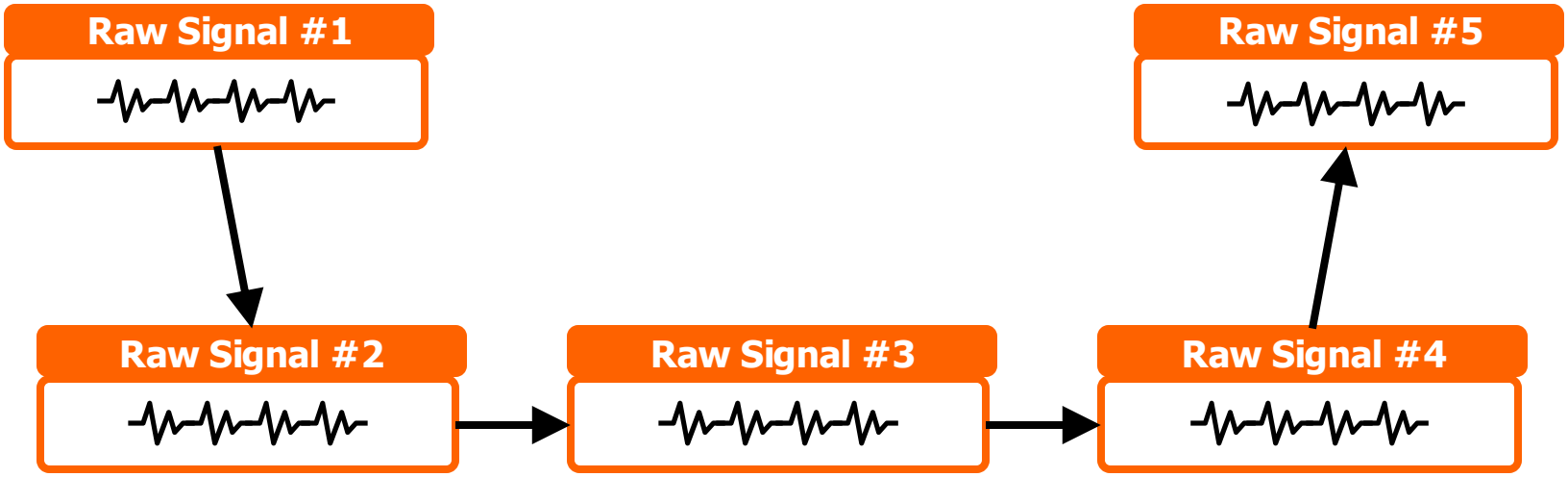
Genome Construction from Raw Signals

- Can Firtina, Maximilian Mordig, Harun Mustafa, Sayan Goswami, Nika Mansouri Ghiasi Stefano Mercogliano, Joël Lindegger, Yan Zhu, Andre Kahles, and Onur Mutlu, **"Rawsamble: Overlapping and Assembling Raw Nanopore Signals using a Hash-based Seeding Mechanism"**
  Presented at ISMB (HiTSeq), Montreal, QC, Canada, Jul. 2024.
  [Slides (pptx) (pdf)]
  [Rawsamble 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] Joël Lindegger[1] Yan Zhu[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*

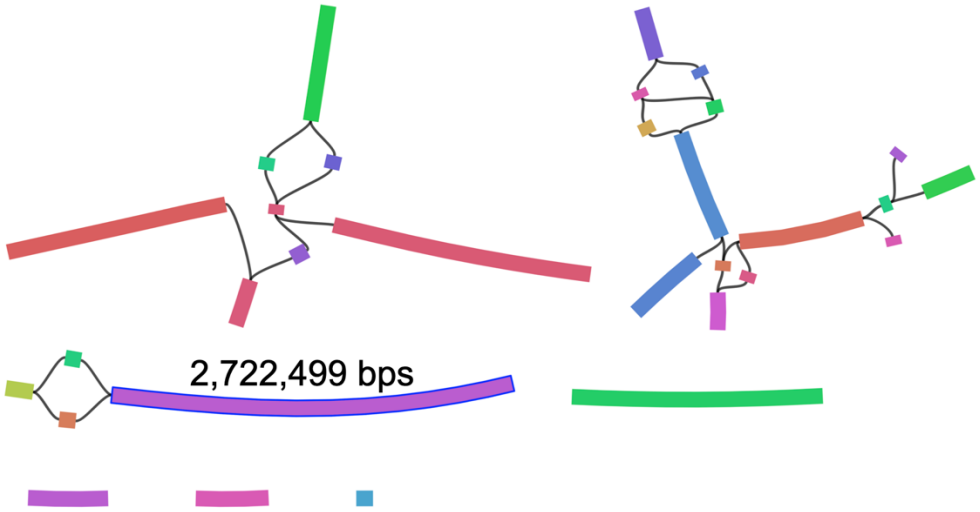
Beyond Reference Mapping: Overlapping



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] Joël Lindegger[1] Yan Zhu[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*

Genome Assembly from Raw Signal Overlaps



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] Joël Lindegger[1] Yan Zhu[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*

Raw Signal Alignment Coupled with Mapping

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

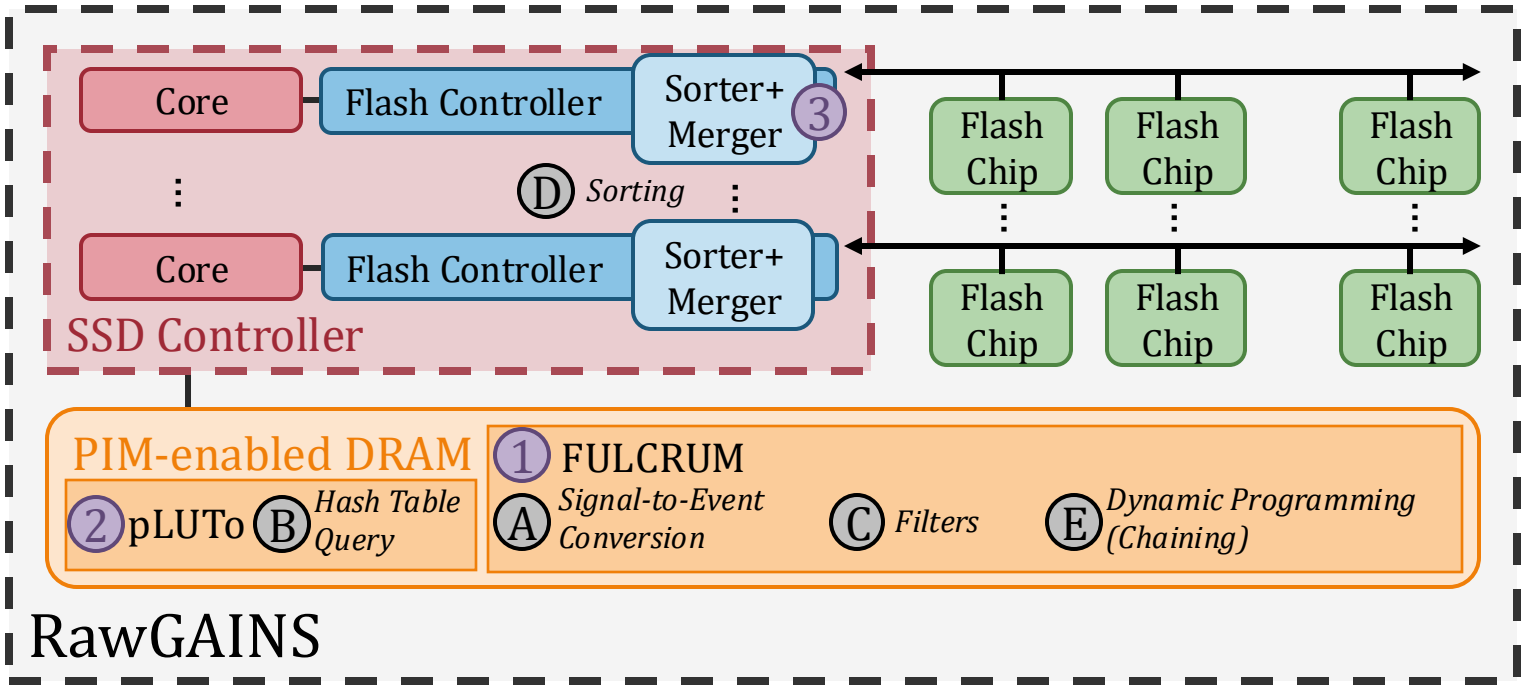
RawAlign: Accurate, Fast, and Scalable Raw Nanopore Signal Mapping via Combining Seeding and Alignment

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

[§]*ETH Zürich*

Storage-Centric Design for Raw Signal Analysis

RawGAINS: A Heterogeneous Storage-Centric Processing System for Raw Signal Genome Analysis



Opportunities for New Applications

- Improving the basecalling accuracy
  - Using the overlapping information
- Full downstream analysis fully using raw nanopore signals
- Cooperating the raw signal analysis and basecalled sequence analysis together
- Many, many more keeping the underlying hardware in mind

Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
- Enabling Scalable Real-time Genome Analysis
- Graph & ML Acceleration for Genomics
- Conclusion

HiPEAC 2024

ApHMM

Paper

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

Can Firtina

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

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

SAFARI ETH zürich

intel TU Delft Carnegie Mellon

Executive Summary

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

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

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

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

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

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

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

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

Outline

Background & Problem

ApHMM

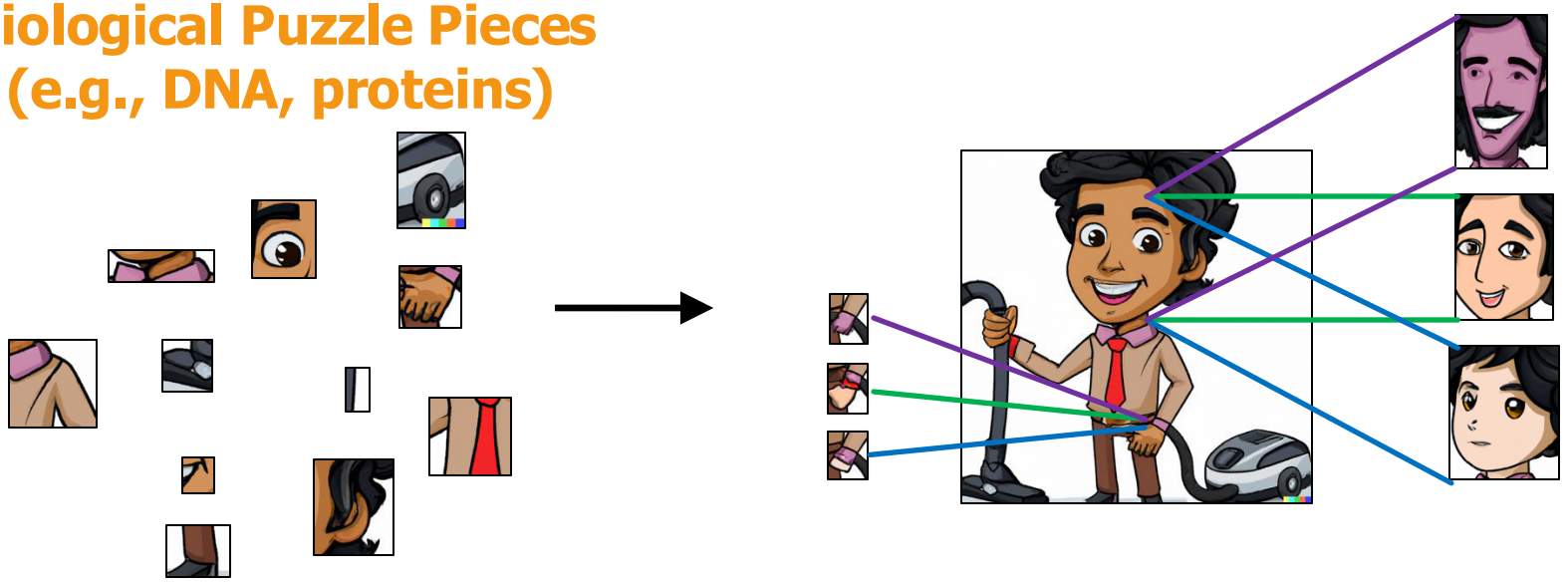
Evaluation

Conclusion

Graphs for Sequence Comparisons

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

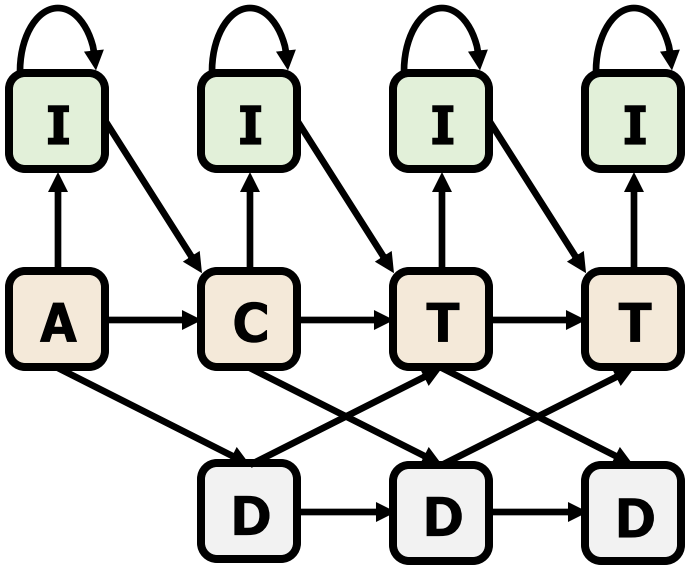
Biological Puzzle Pieces (e.g., DNA, proteins)



Profile Hidden Markov Models

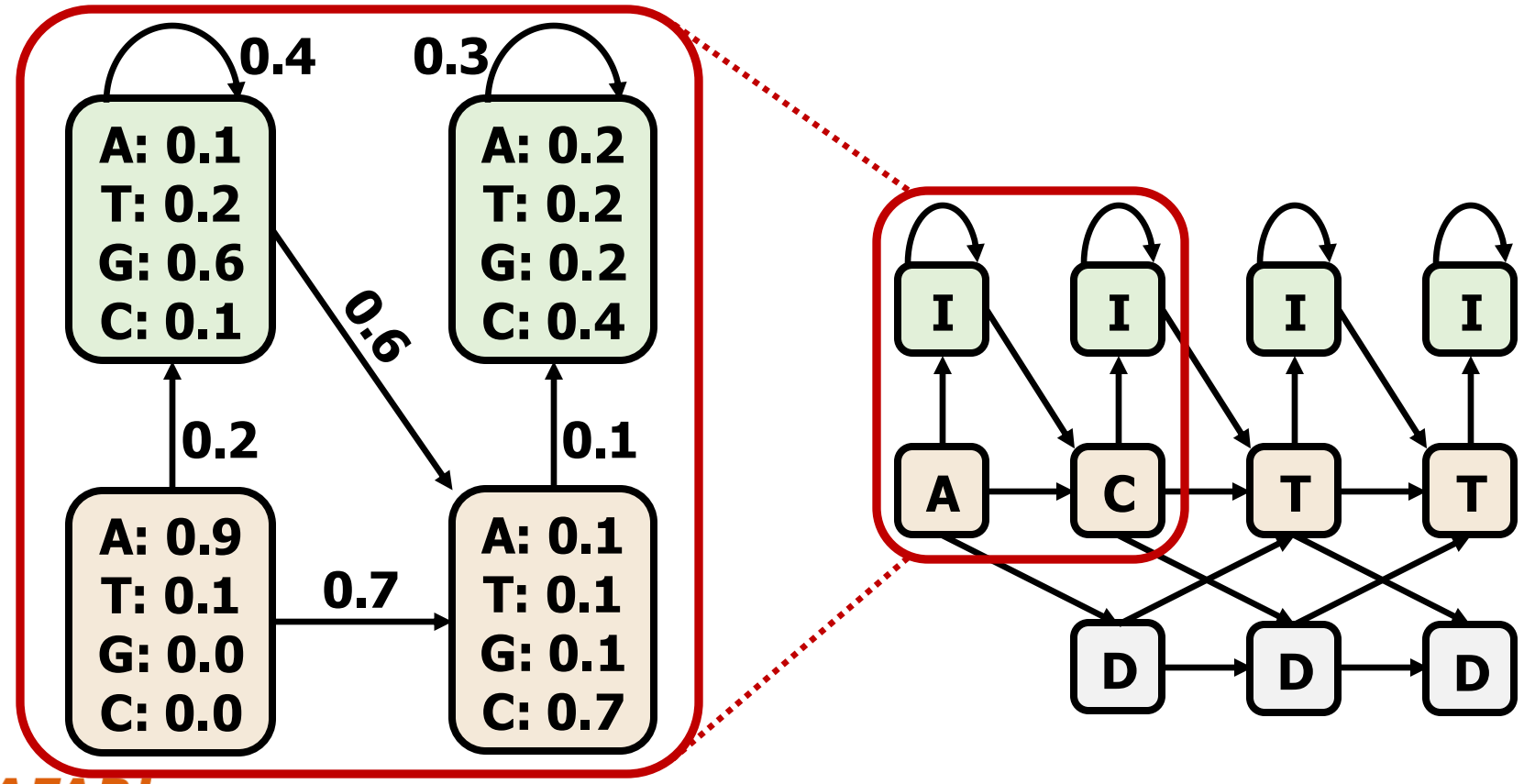
- Profile Hidden Markov Models (pHMMs) are powerful and common **graph structures** for sequence comparison
  - **Goal:** Identify variations between sequences **probabilistically**
  - **Variations:** No variation, Substitutions, Insertions, Deletions
  - Each **state** outputs a biological character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

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



Probabilities in pHMMs

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



Utilizing Probabilities in pHMMs

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

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

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

Forward Calculations

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

Backward Calculations

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

Updating Transition Probabilities

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

Updating Emission Probabilities

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

Utilizing Probabilities in pHMMs

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

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

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

Forward Calculations

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

Backward Calculations

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

Updating Transition Probabilities

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

Updating Emission Probabilities

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

Utilizing Probabilities in pHMMs

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

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

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

Forward Calculations

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

Backward Calculations

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

Training Step

Updating Transition Probabilities

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

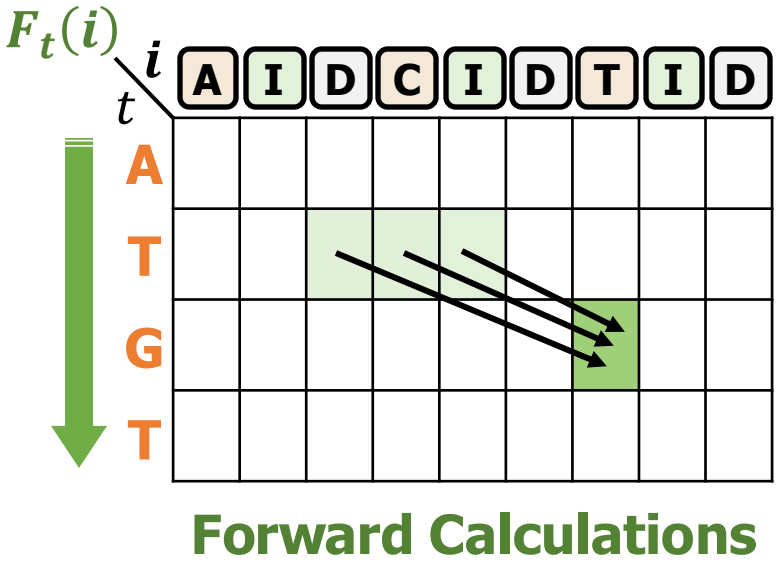
Updating Emission Probabilities

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

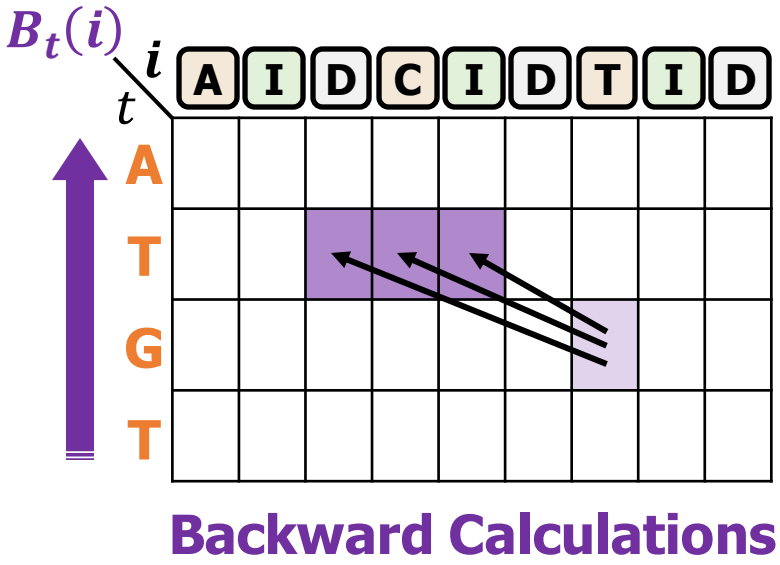
Forward & Backward Calculations

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

Observed Sequence: ATGT



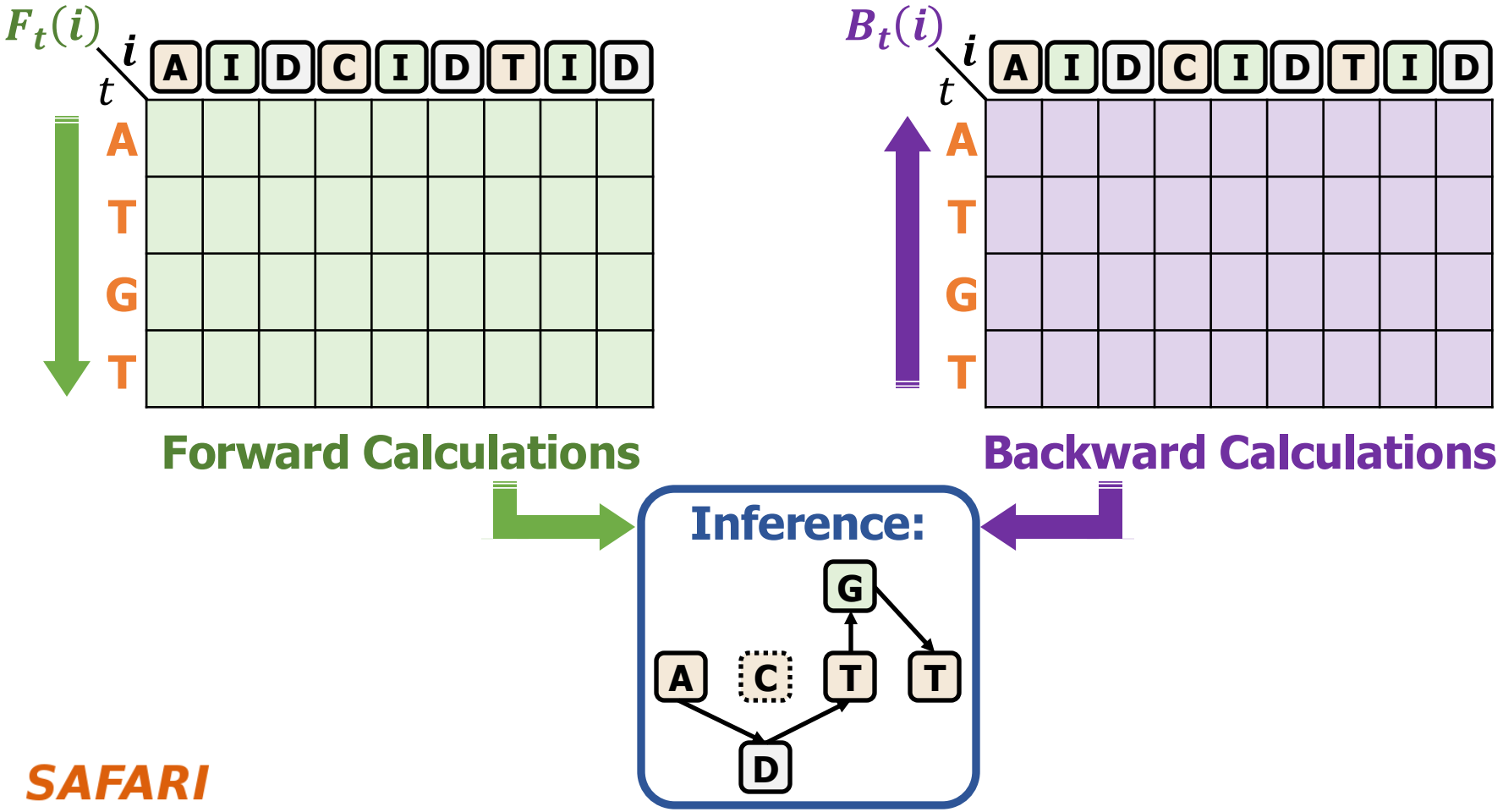
Forward Calculations



Backward Calculations

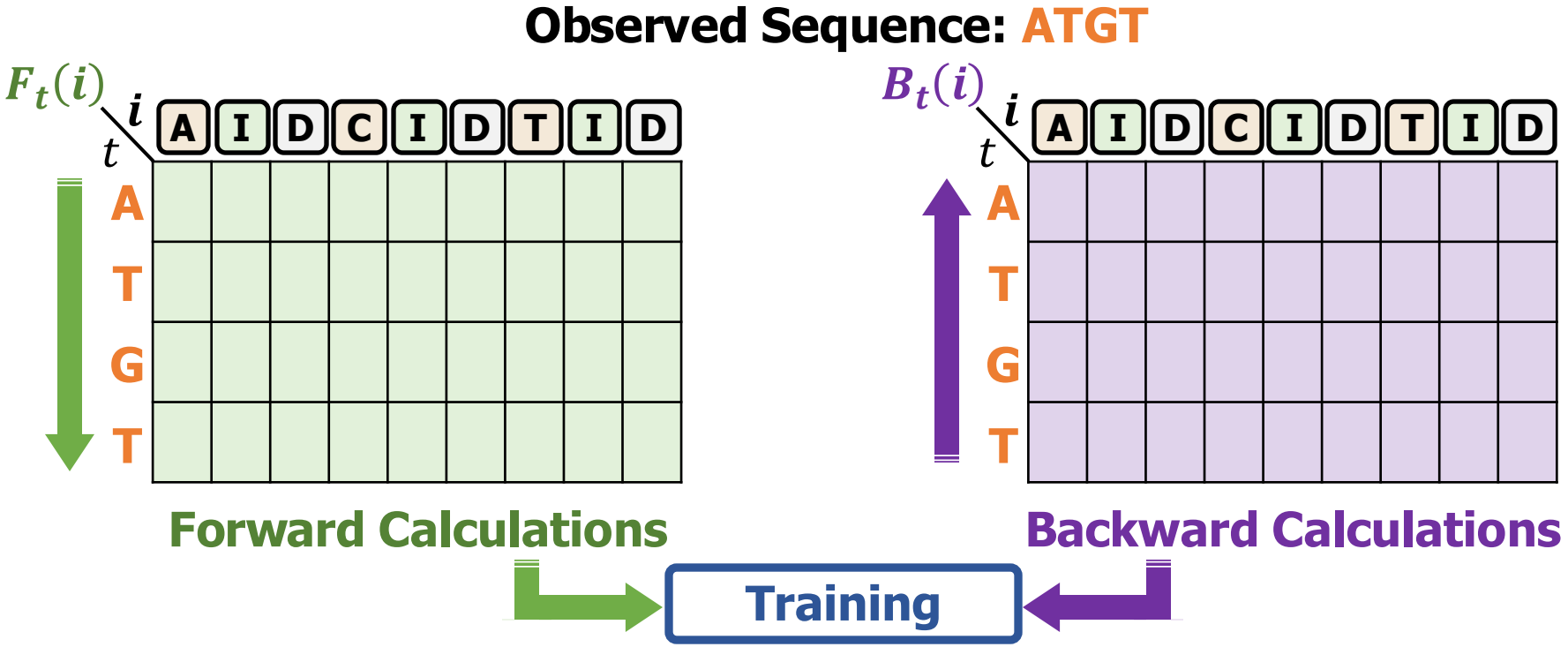
Inference using pHMMs

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



Training using pHMMs

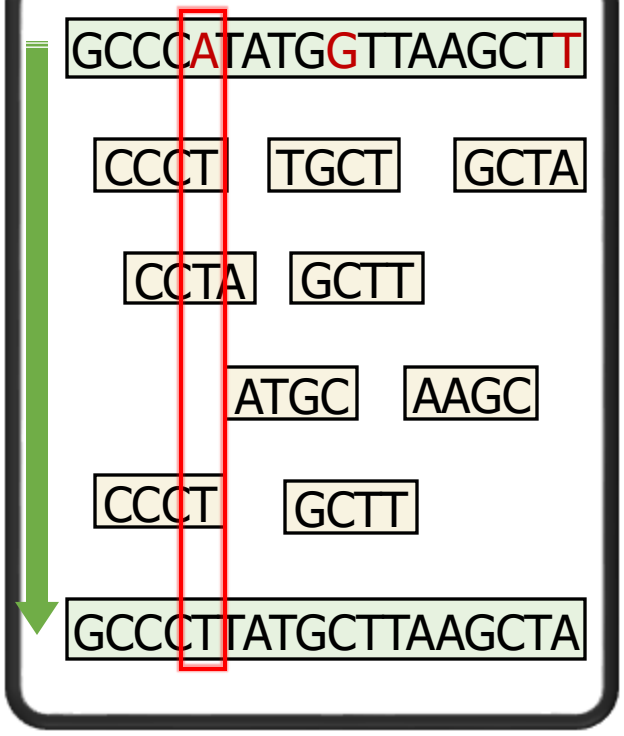
- **Goal:** Maximizing parameters to observe certain variations
  - **Training** using the parameter updating steps in the Baum-Welch algorithm



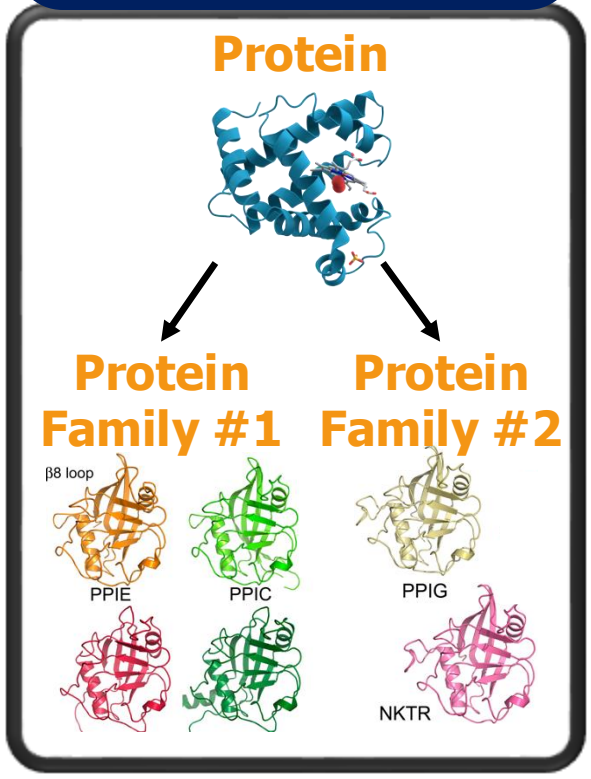
pHMMs in Genomics Workloads

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

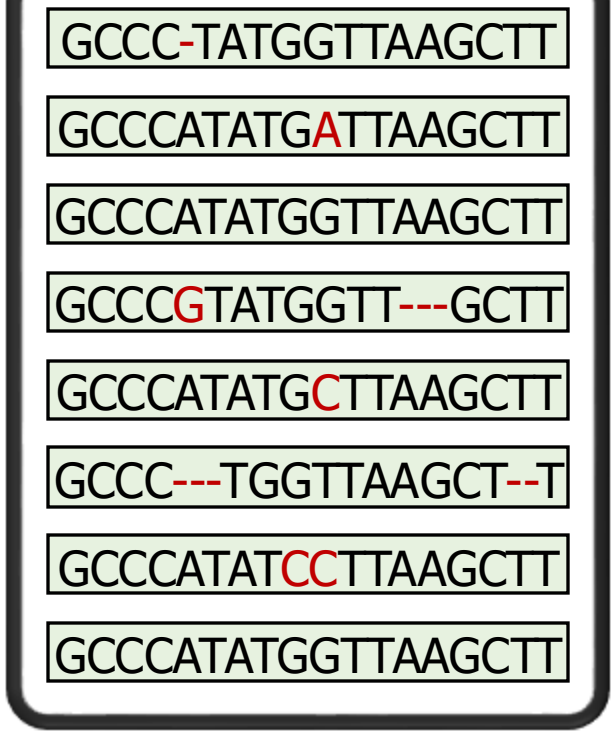
1. Error Correction



2. Protein Family Search

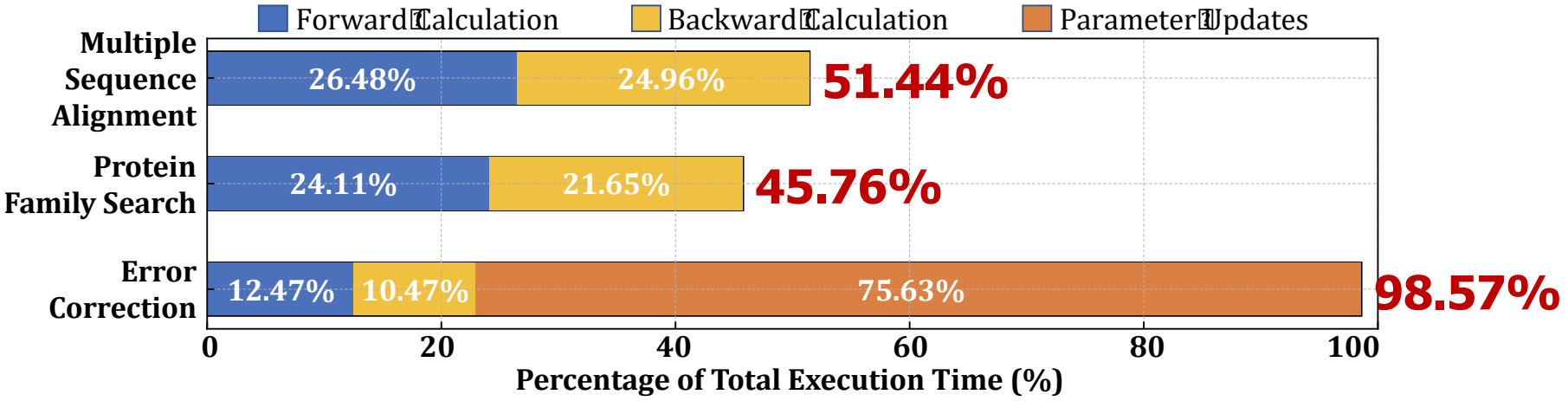


3. Multiple Sequence Alignment



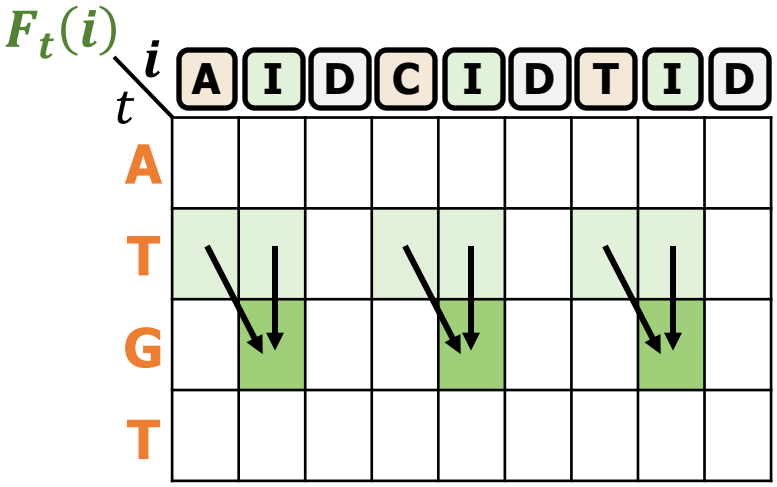
The Baum-Welch Algorithm is Costly

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

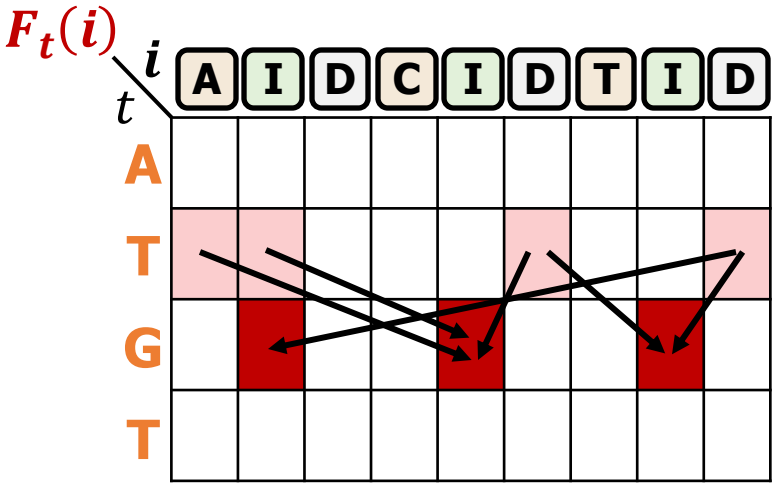


Existing Solutions are Ineffective

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



Forward Calculations in pHMMs

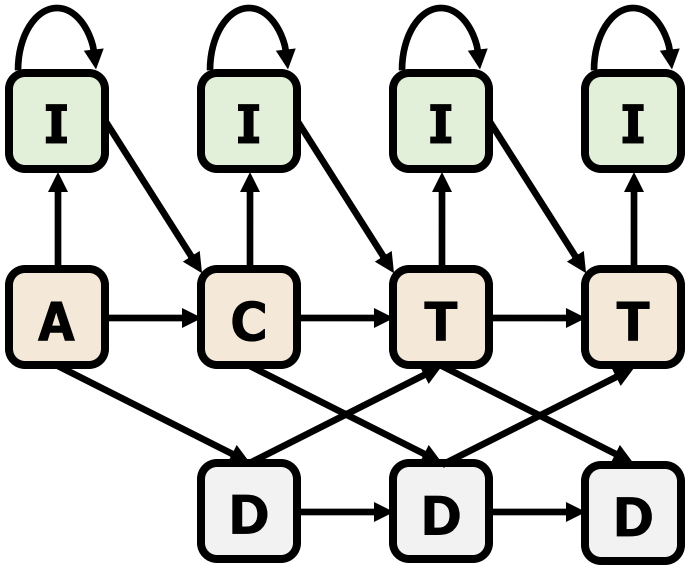


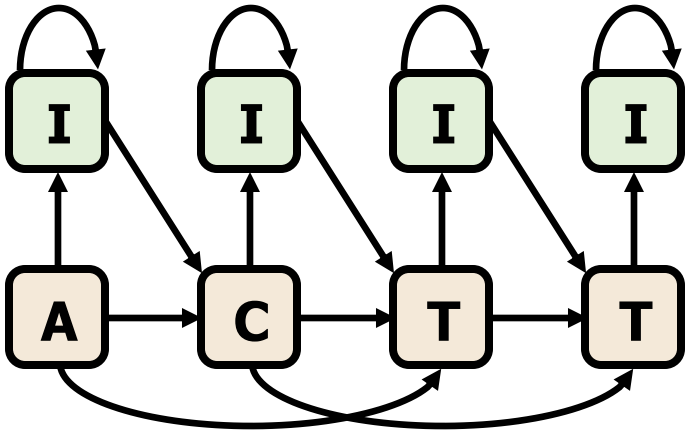
Forward Calculations in HMMs

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

Existing Solutions are Inflexible

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





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

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

Existing Solutions are Inefficient

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

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

The Problem

Using pHMMs causes major performance overhead in important genomics applications

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

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Goal

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

ApHMM

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

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

Key Software & Hardware Optimizations

- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering

SW

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

- **Reduce data movement** by exploiting fixed data pattern

HW

- **Flexible and efficient** control logic and hardware design

Key Software & Hardware Optimizations

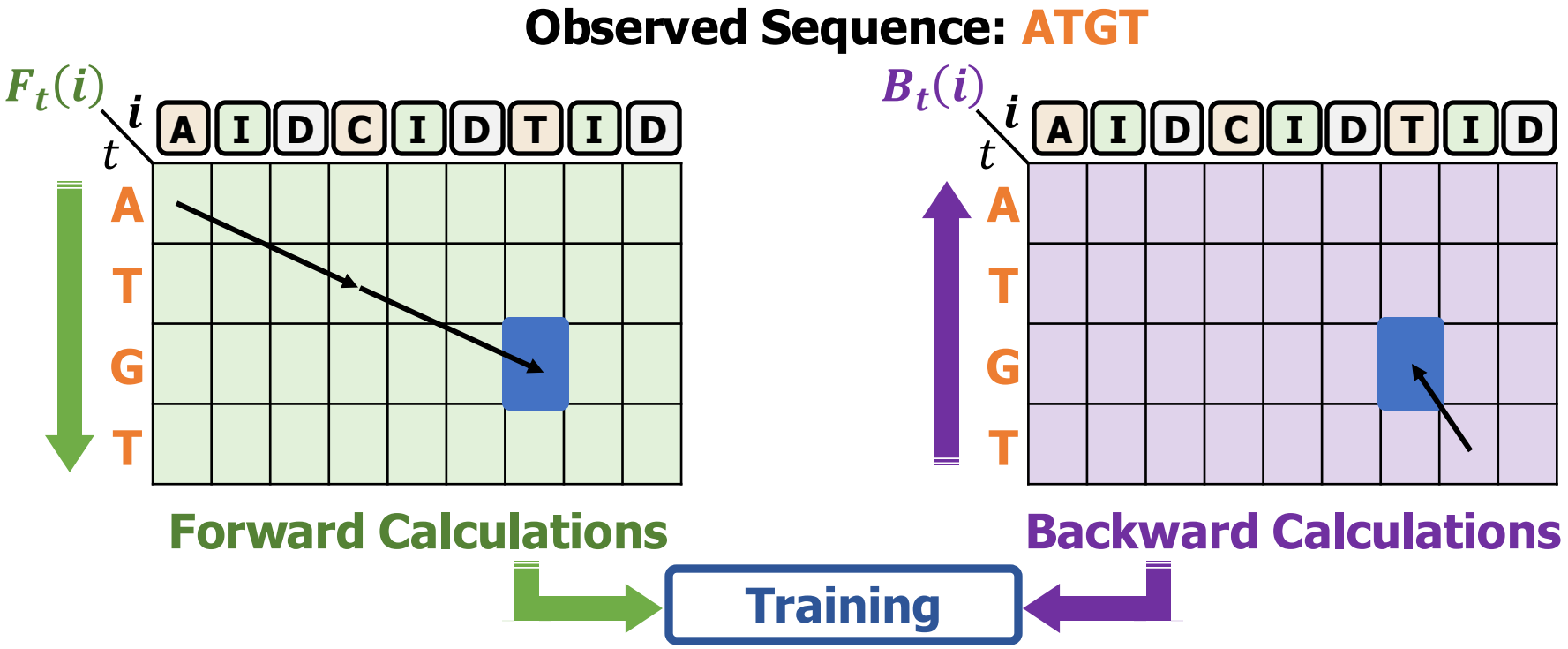
- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering

SW

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

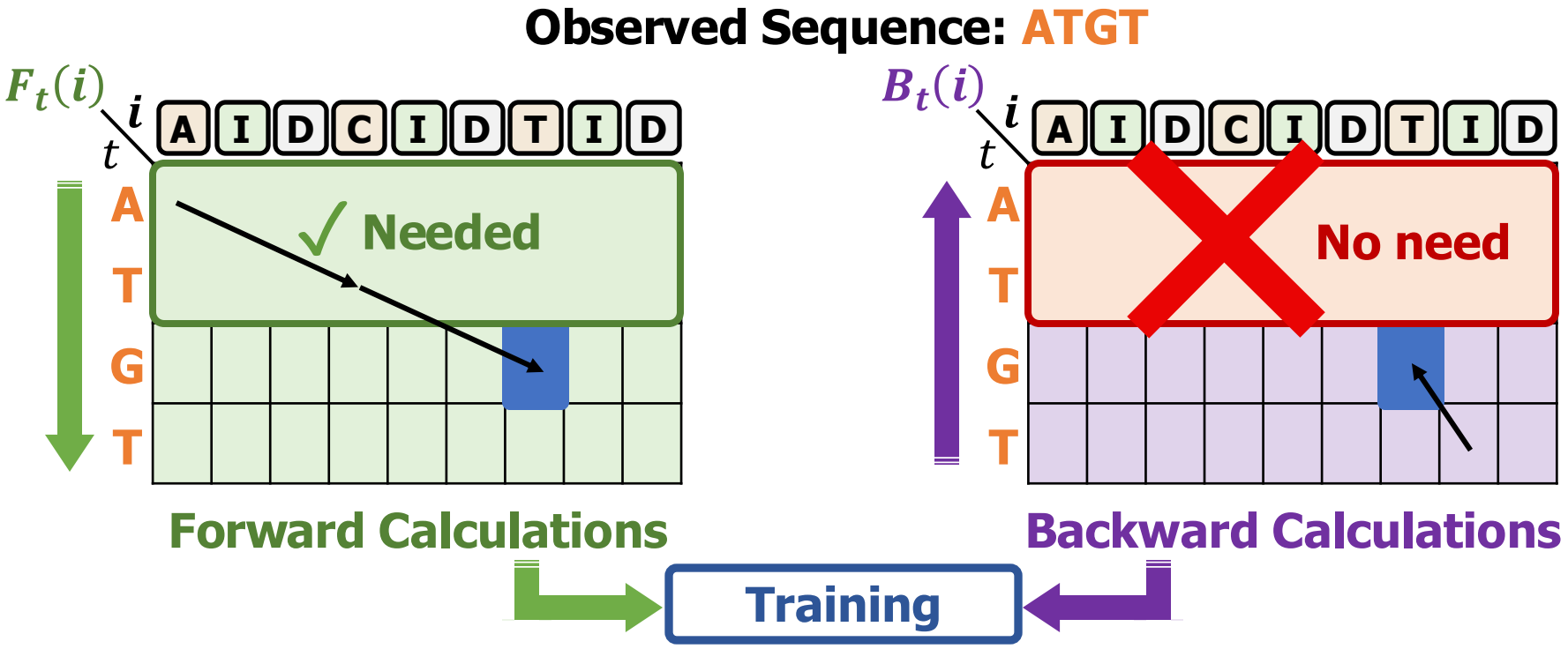
SW: Minimizing Redundant Storage

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



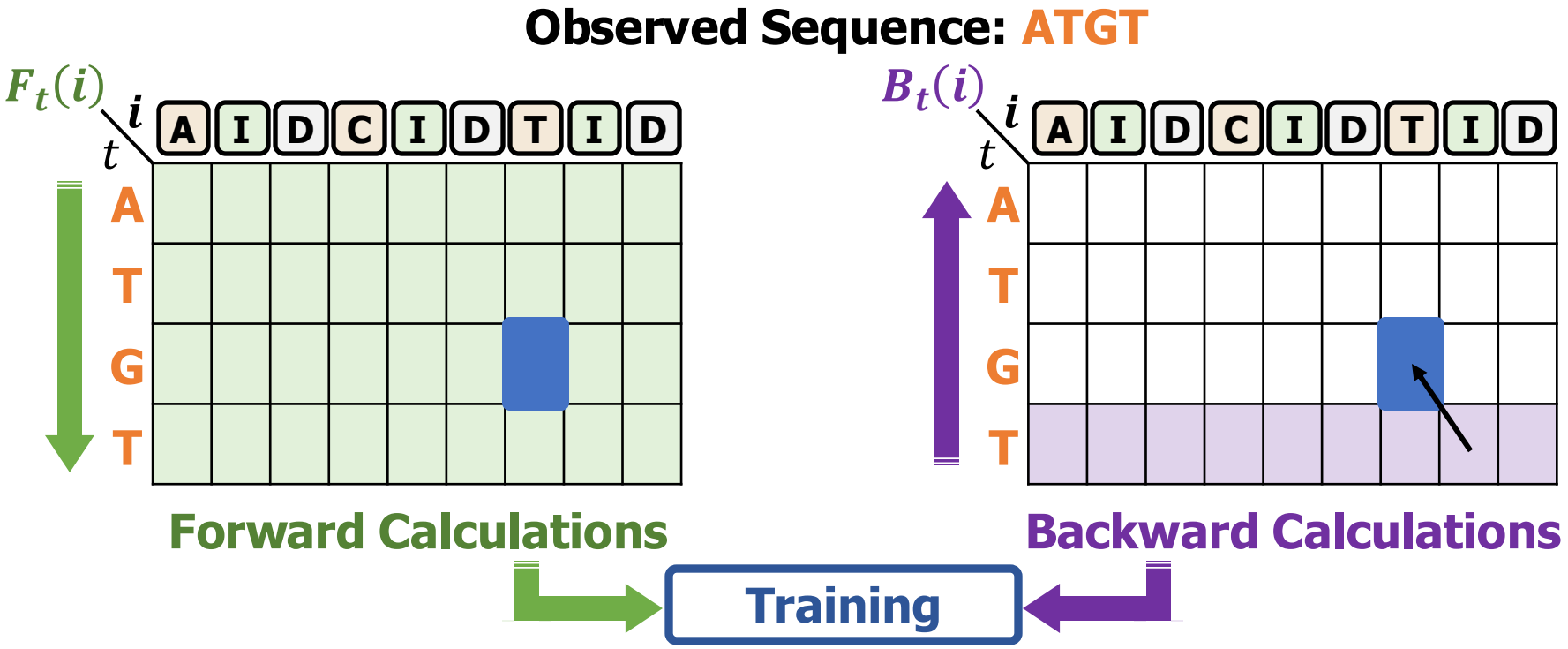
SW: Minimizing Redundant Storage

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



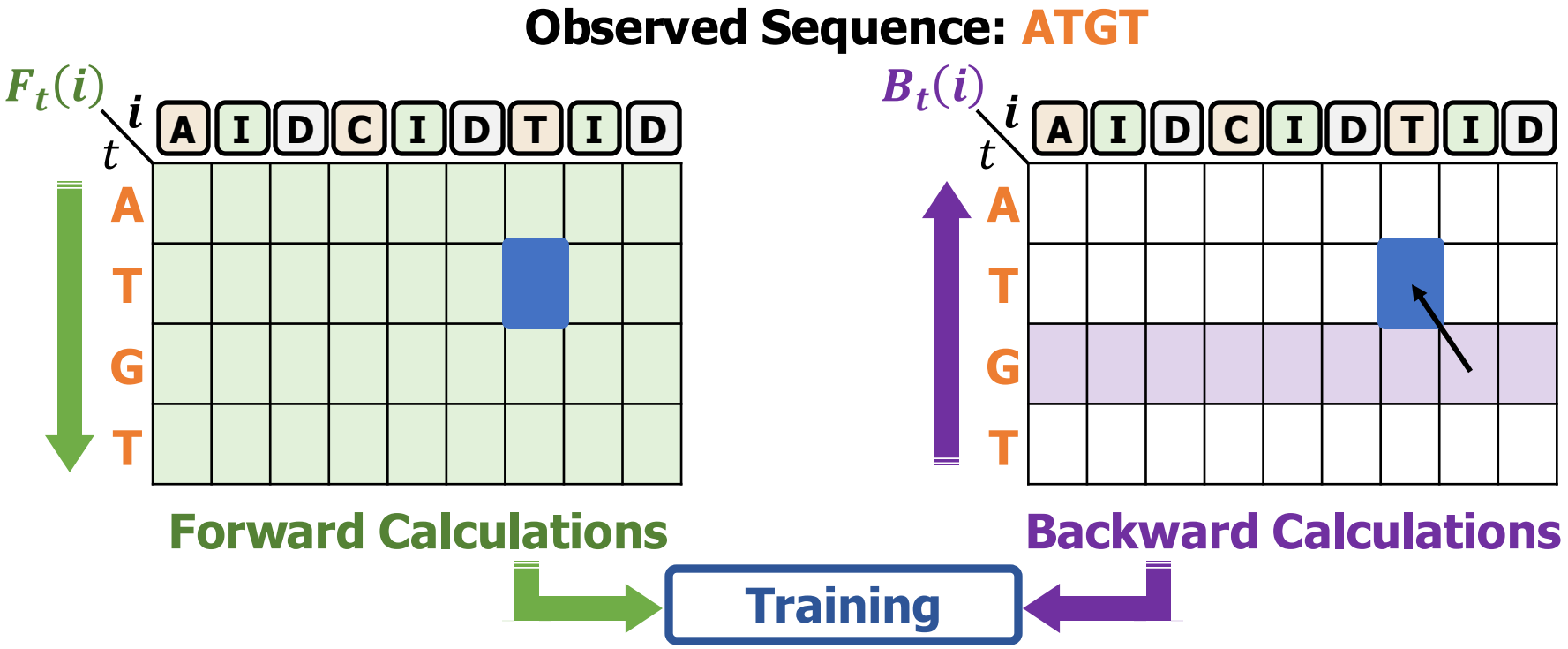
SW: Minimizing Redundant Storage

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



SW: Minimizing Redundant Storage

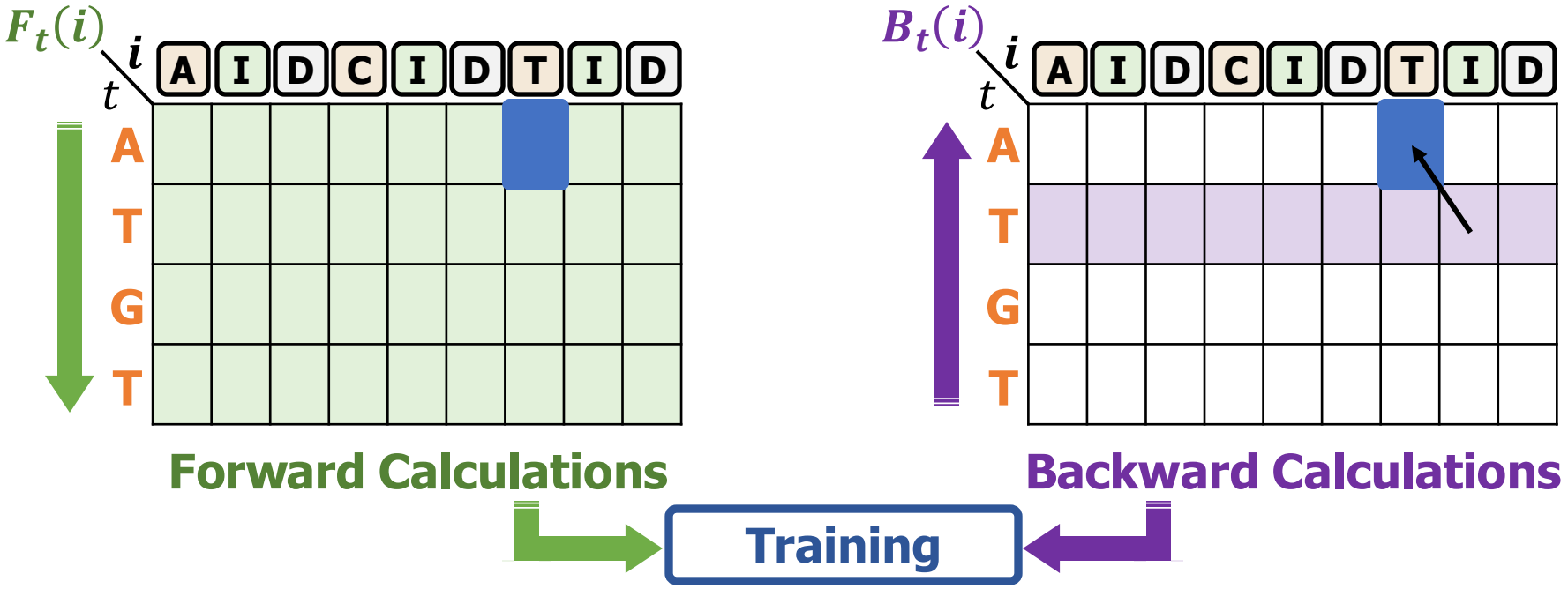
- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value
  - **Partial compute approach:** Only a single row should be **fully stored**



SW: Minimizing Redundant Storage

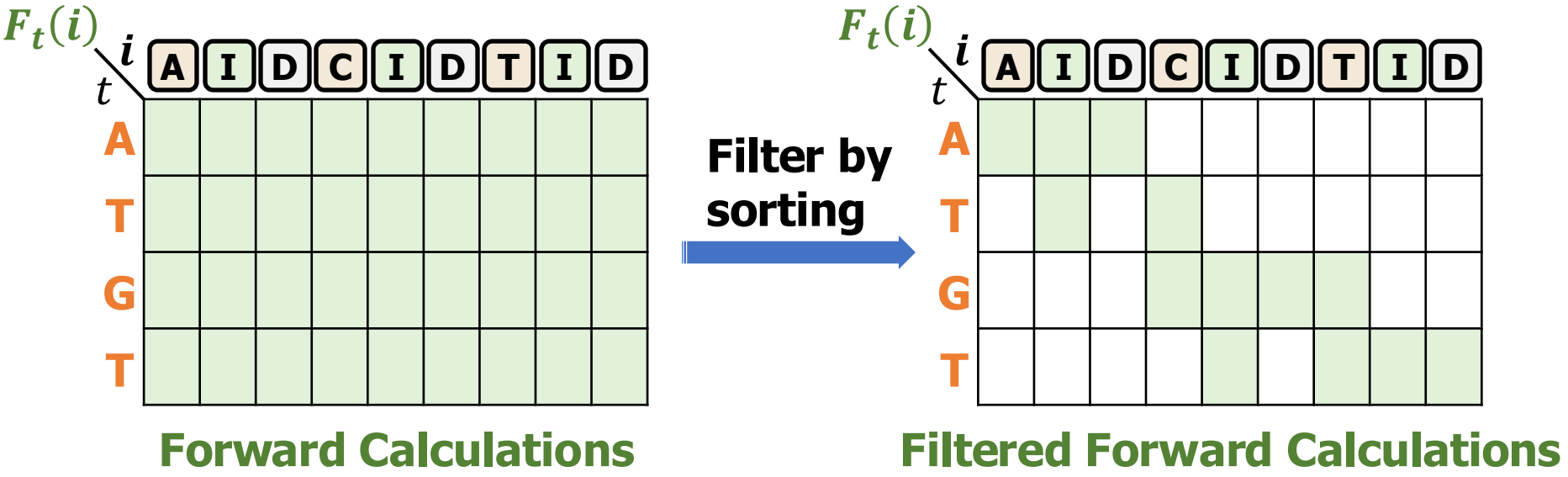
- **Observation:** Filling the entire Backward table is unnecessary
  - **Pipelining opportunities** to directly consume a Backward value
  - **Partial compute approach:** Only a single row should be **fully stored**
  - **Reduces the storage requirements** during training

Observed Sequence: ATGT



SW: Reducing Unnecessary Computations

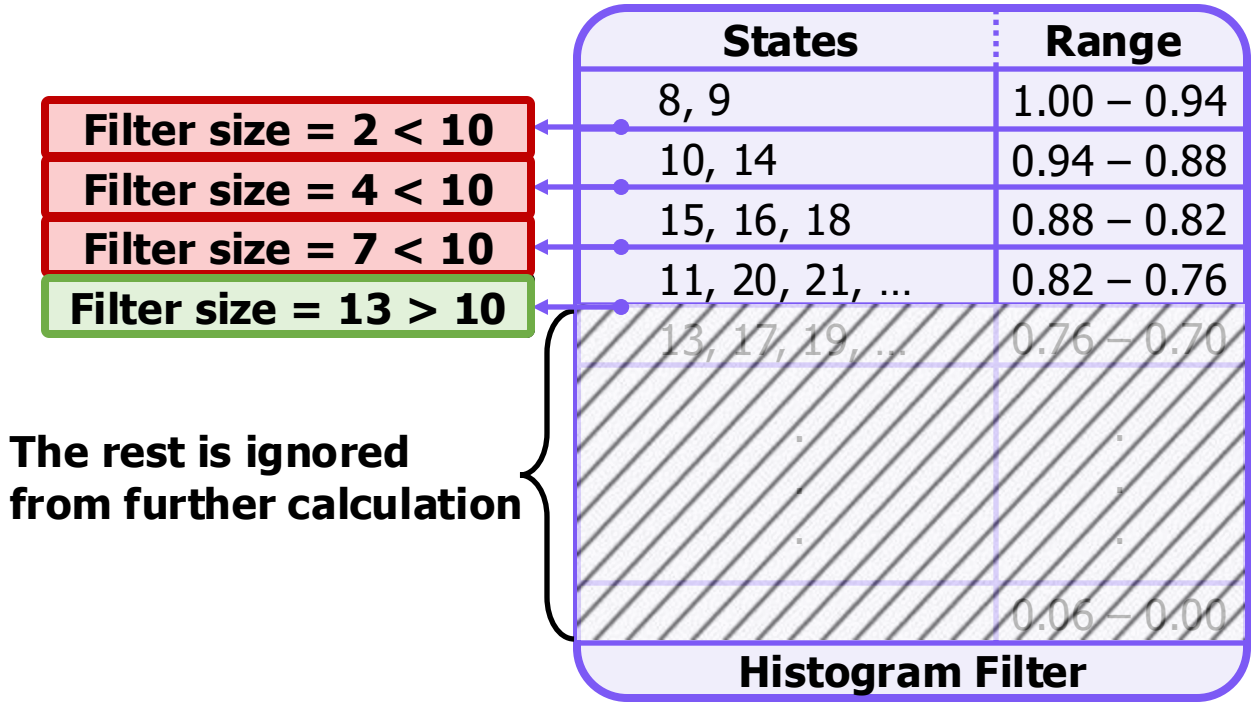
- **Observation:** 'Negligible' cells can be ignored without significantly reducing overall accuracy
  - **Filtering:** Non-negligible states are identified by sorting
  - **Sorting** to find **exactly** n states with **largest** Forward or Backward values



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

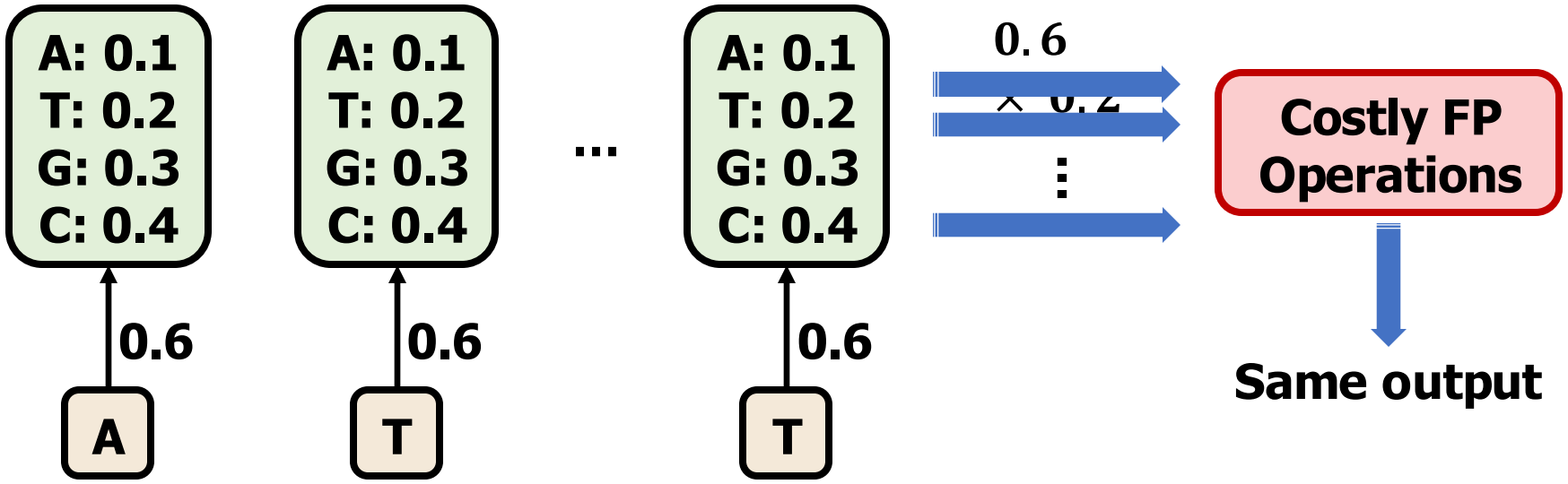
SW: Reducing Unnecessary Computations

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



SW: Avoiding Repeated Operations

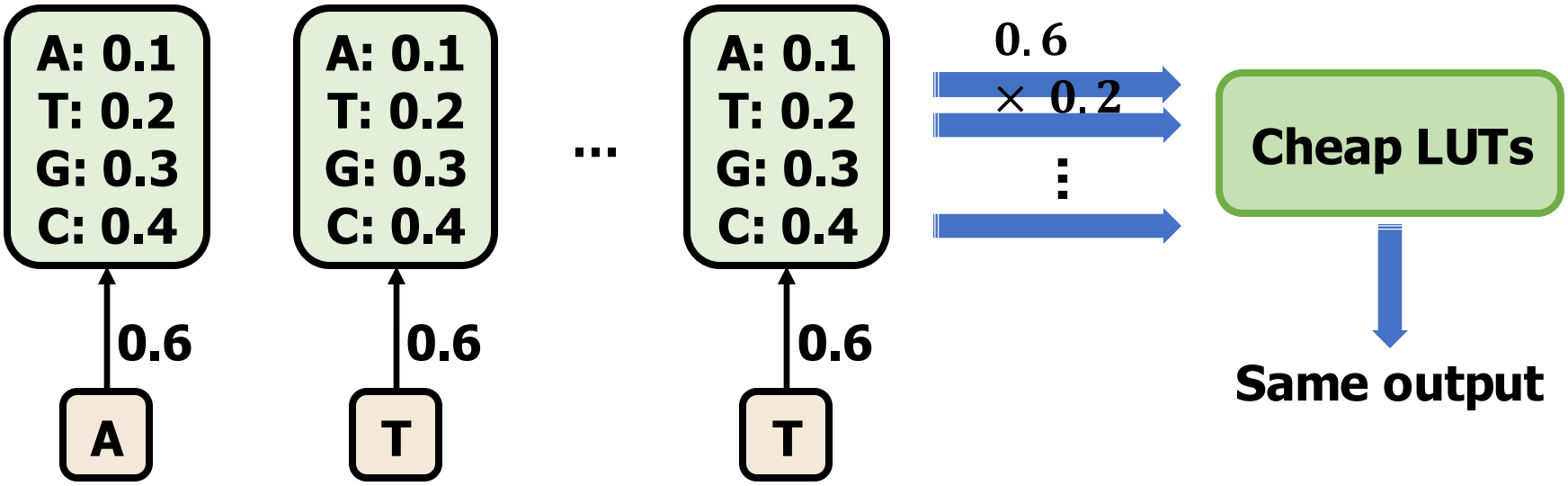
- **Observation:** Same multiplications are redundantly performed
  - **Same default values are used** for each possible connection in pHMMs
  - **Fixed connection patterns** generate a fixed set of multiplication results



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

SW: Avoiding Repeated Operations

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



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

Key Software & Hardware Optimizations

- **Minimize redundant data storage** by efficient pipelining
- **Reduce unnecessary computations** with quick filtering

SW

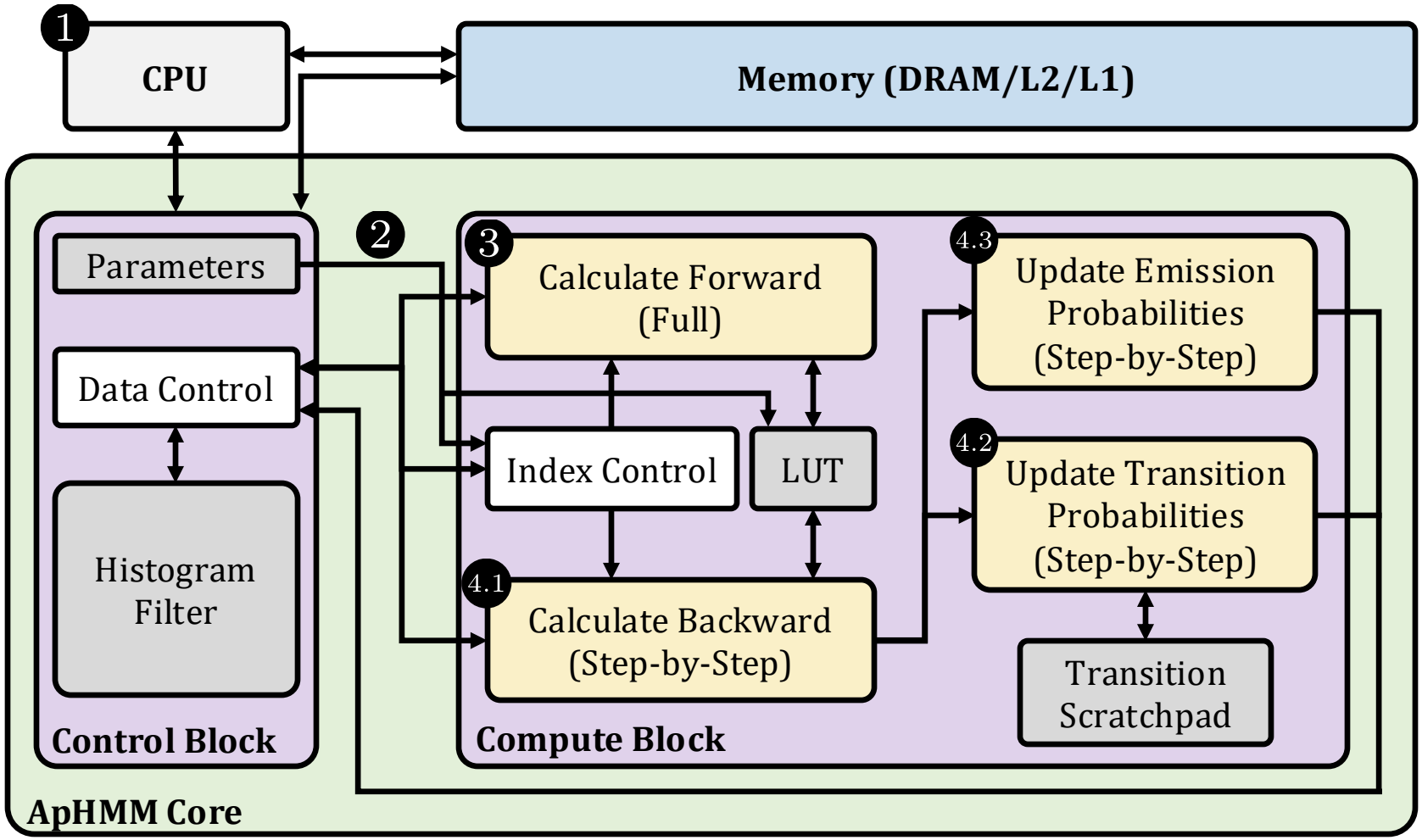
- **Avoid repeated operations** by utilizing lookup tables

- **Reduce data movement** by exploiting fixed data pattern

HW

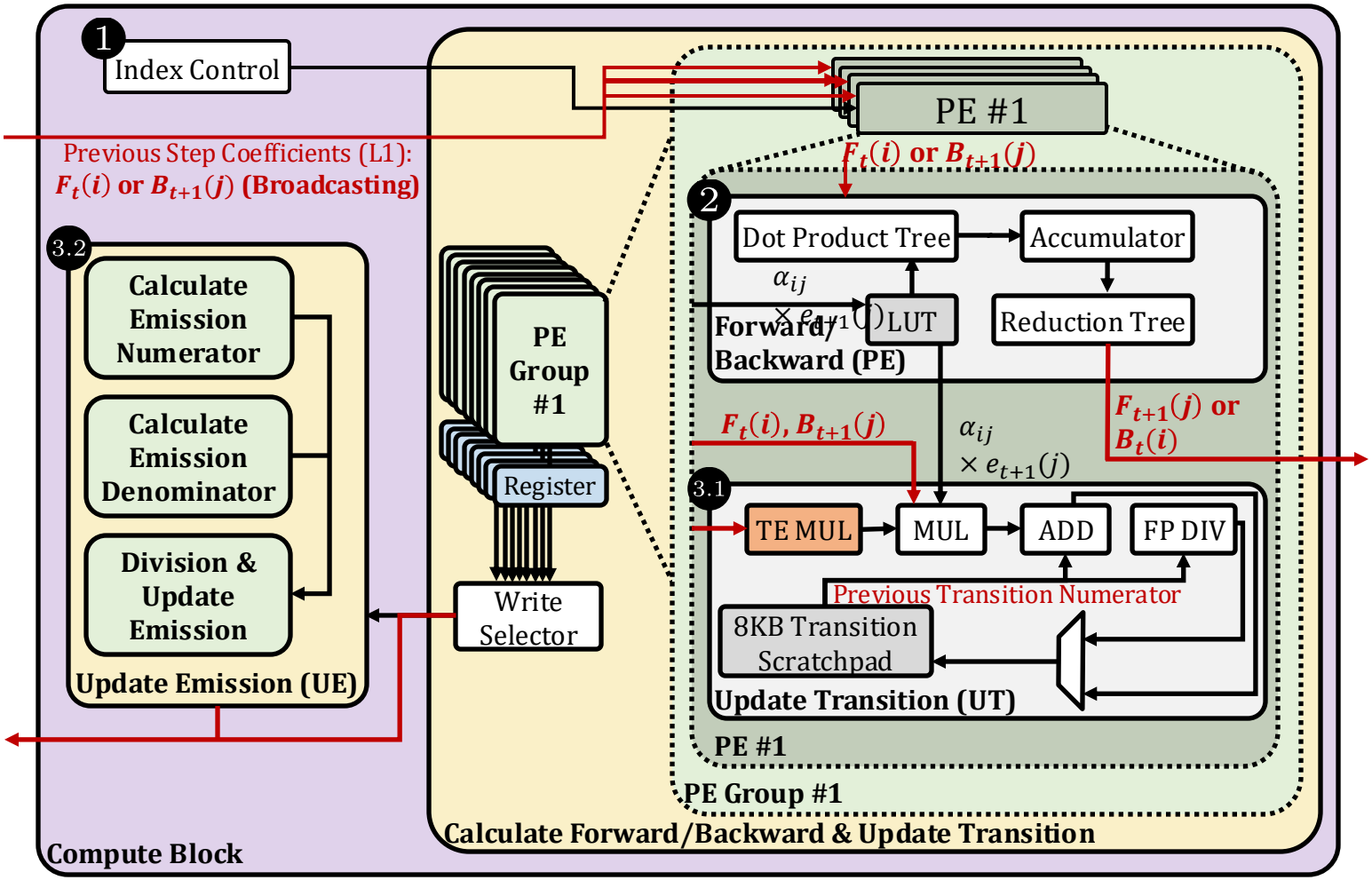
- **Flexible and efficient** control logic and hardware design

Overview of ApHMM Design



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

Computing the Baum-Welch in ApHMM



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

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Evaluation Methodology

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

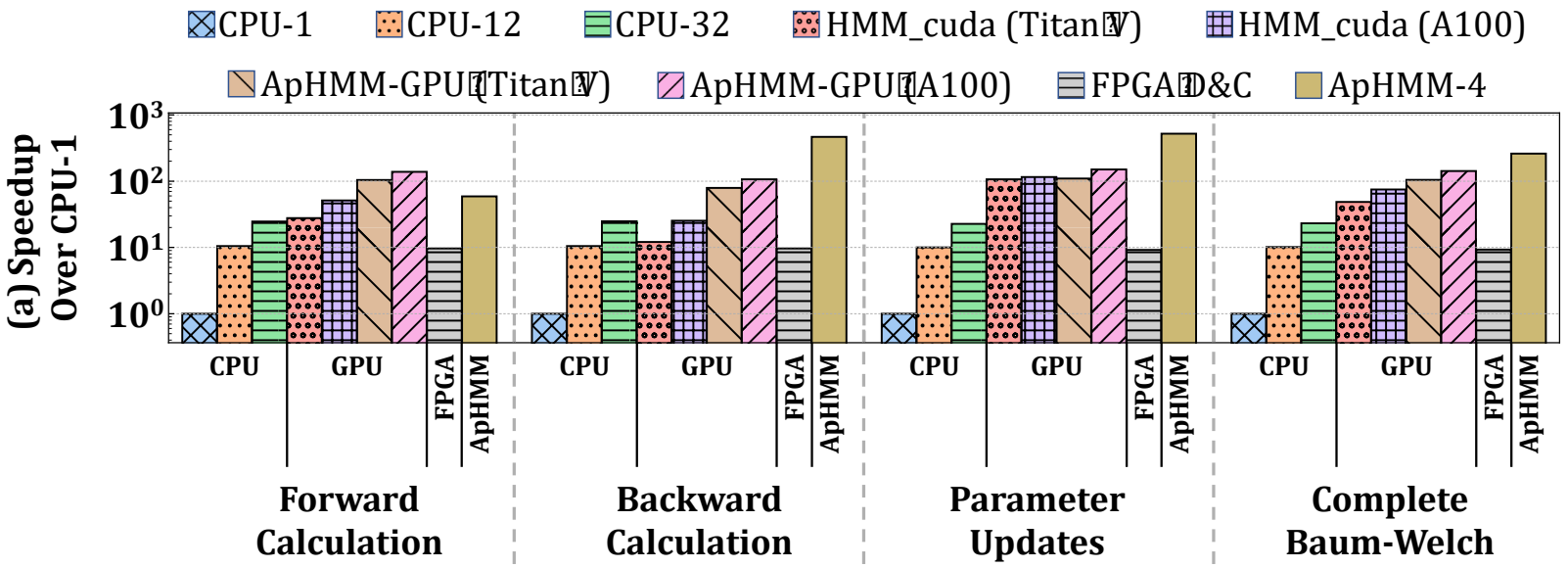
- **Use cases** and their software baseline:
  1. Error Correction – Apollo
  2. Protein Family Search – HMMER
  3. Multiple Sequence Alignment – HMMER

Evaluation Methodology

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

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

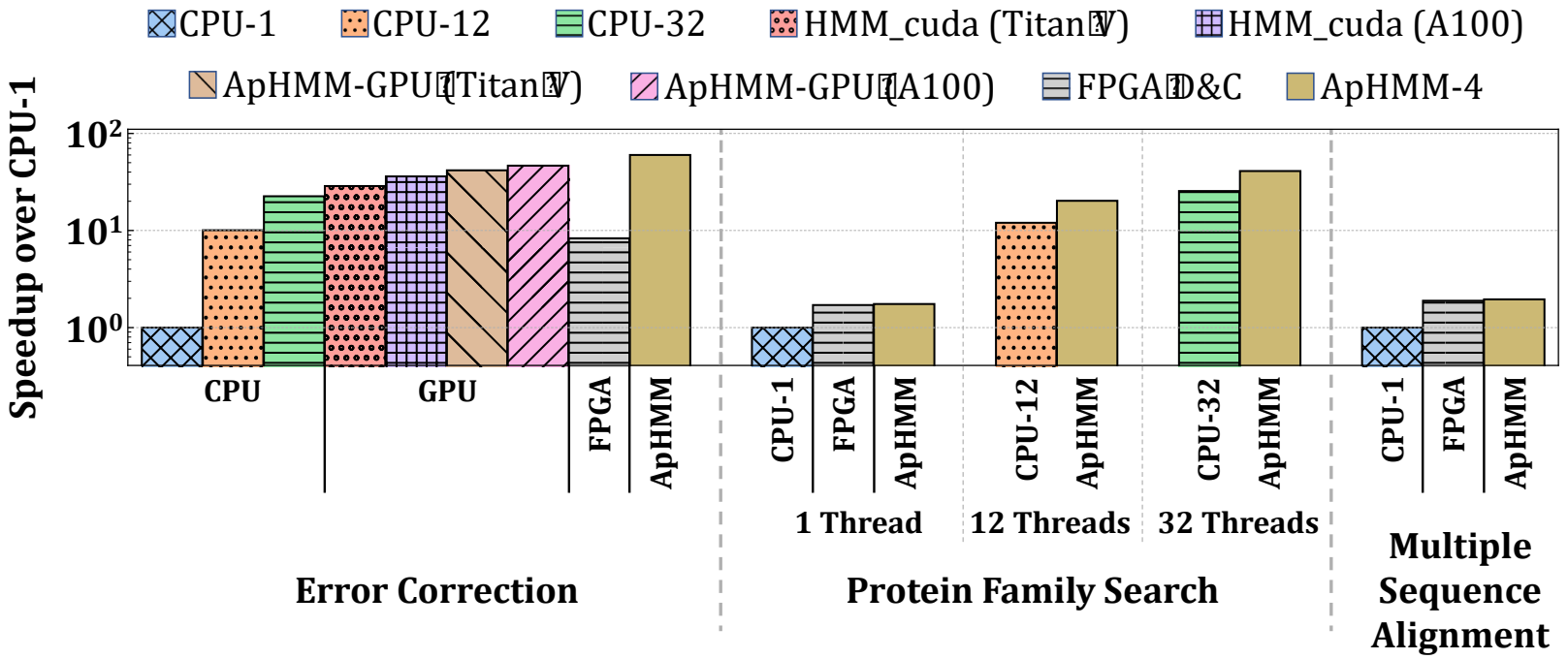
Performance: The Baum-Welch Algorithm



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

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

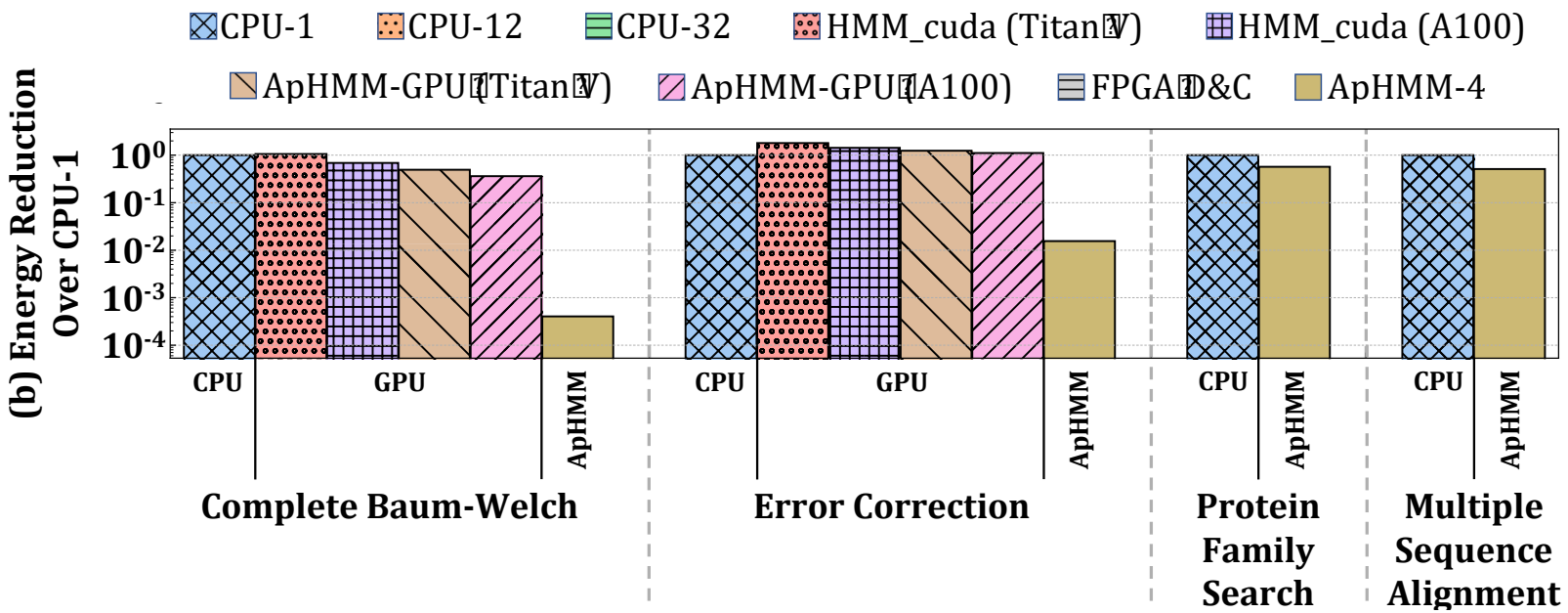
Performance: Workload Acceleration



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

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

Energy: Overall Comparisons



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

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

More in the Paper

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

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

ApHMM [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***), February 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]

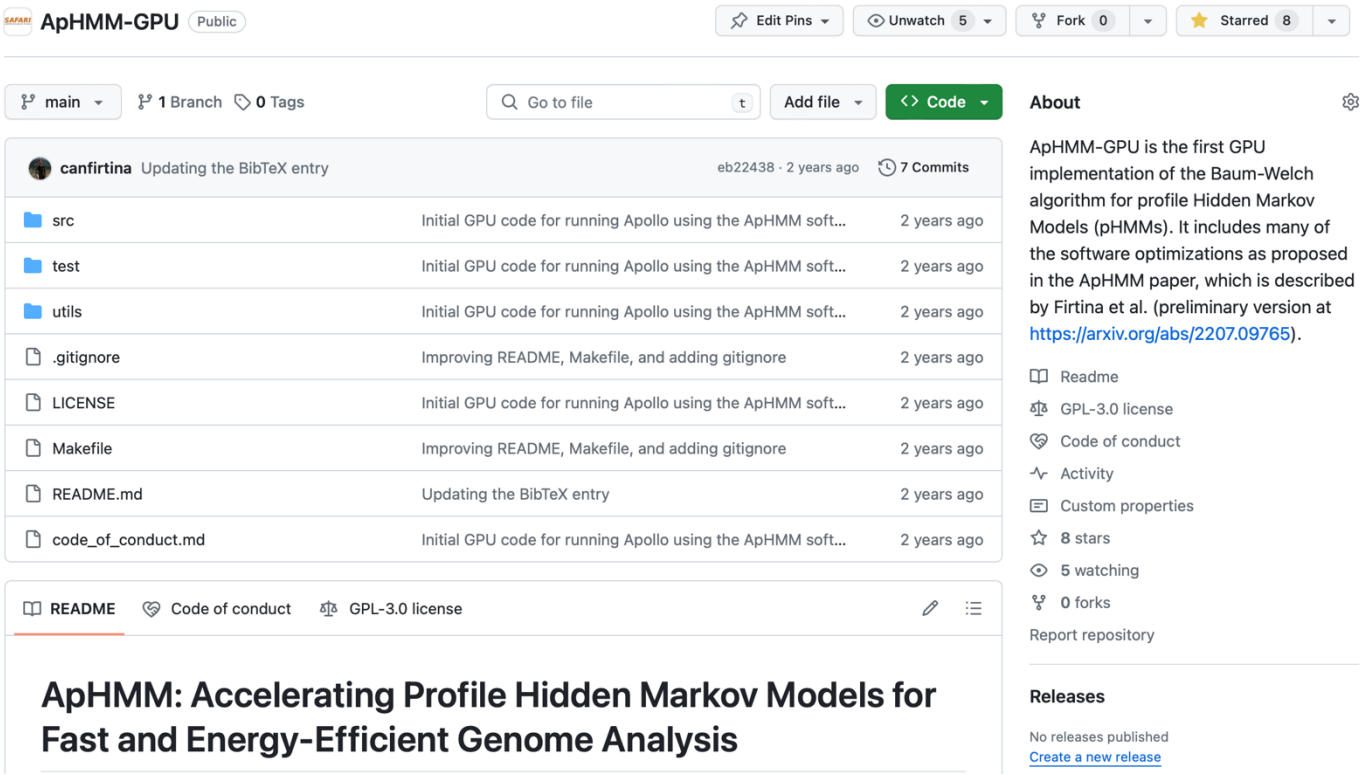
RESEARCH-ARTICLE | OPEN ACCESS |

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

GPU Source Code of ApHMM



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

Outline

Background & Problem

ApHMM

Evaluation

Conclusion

Conclusion

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

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

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

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

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

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

HiPEAC 2024

ApHMM

Paper

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

Can Firtina

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

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

SAFARI ETH zürich

intel TU Delft Carnegie Mellon

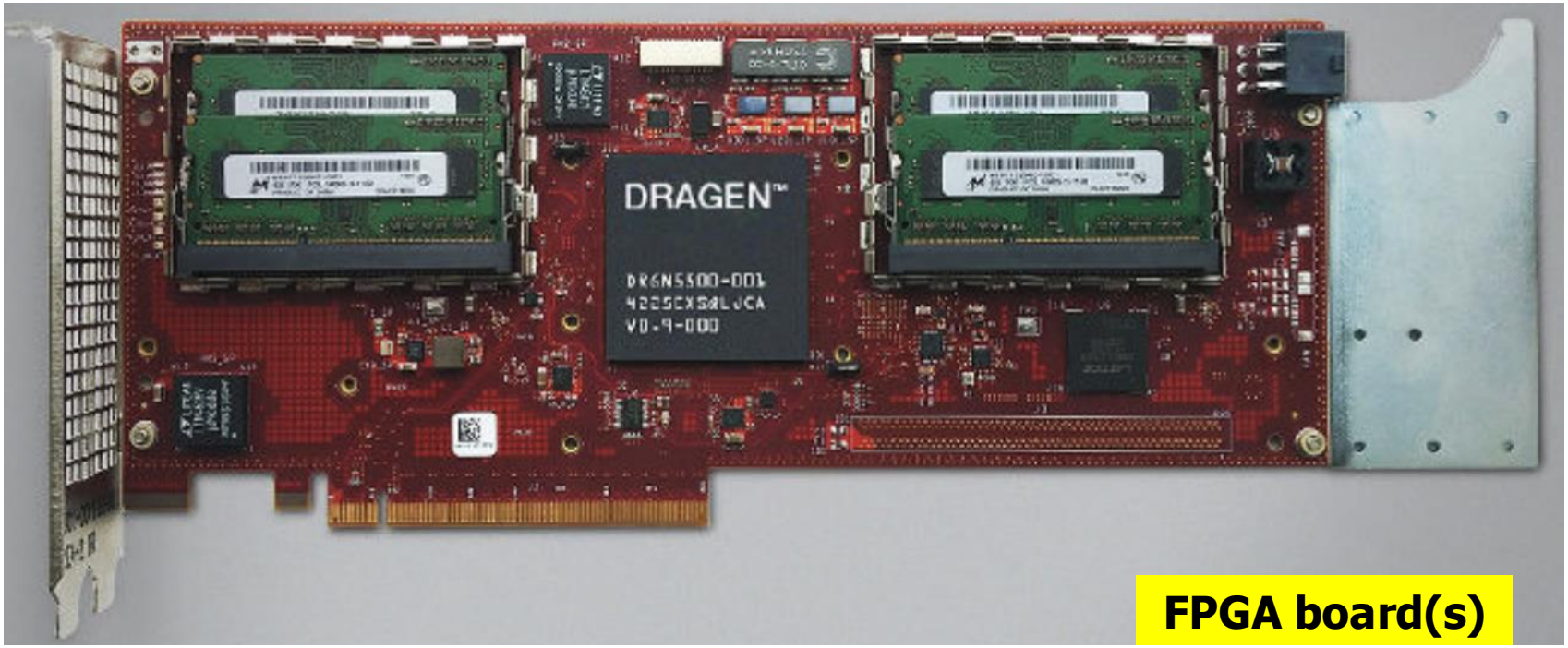
Agenda for Today

- Cutting-edge in Accelerating Genome Analysis
- Enabling Scalable Real-time Genome Analysis
- Graph & ML Acceleration for Genomics
- Conclusion

Things Are Happening In Industry

Illumina DRAGEN Bio-IT Platform (2018)

- Processes whole genome at 34x coverage in ~30 minutes with hardware support for data compression



emea.illumina.com/products/by-type/informatics-products/dragen-bio-it-platform.html
emea.illumina.com/company/news-center/press-releases/2018/2349147.html
illumina.com/content/dam/illumina/gcs/assembled-assets/marketing-literature/dragen-bio-it-data-sheet-m-gl-00680/dragen-bio-it-data-sheet-m-gl-00680.pdf

Nova/NextSeq with Analysis Capability

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.

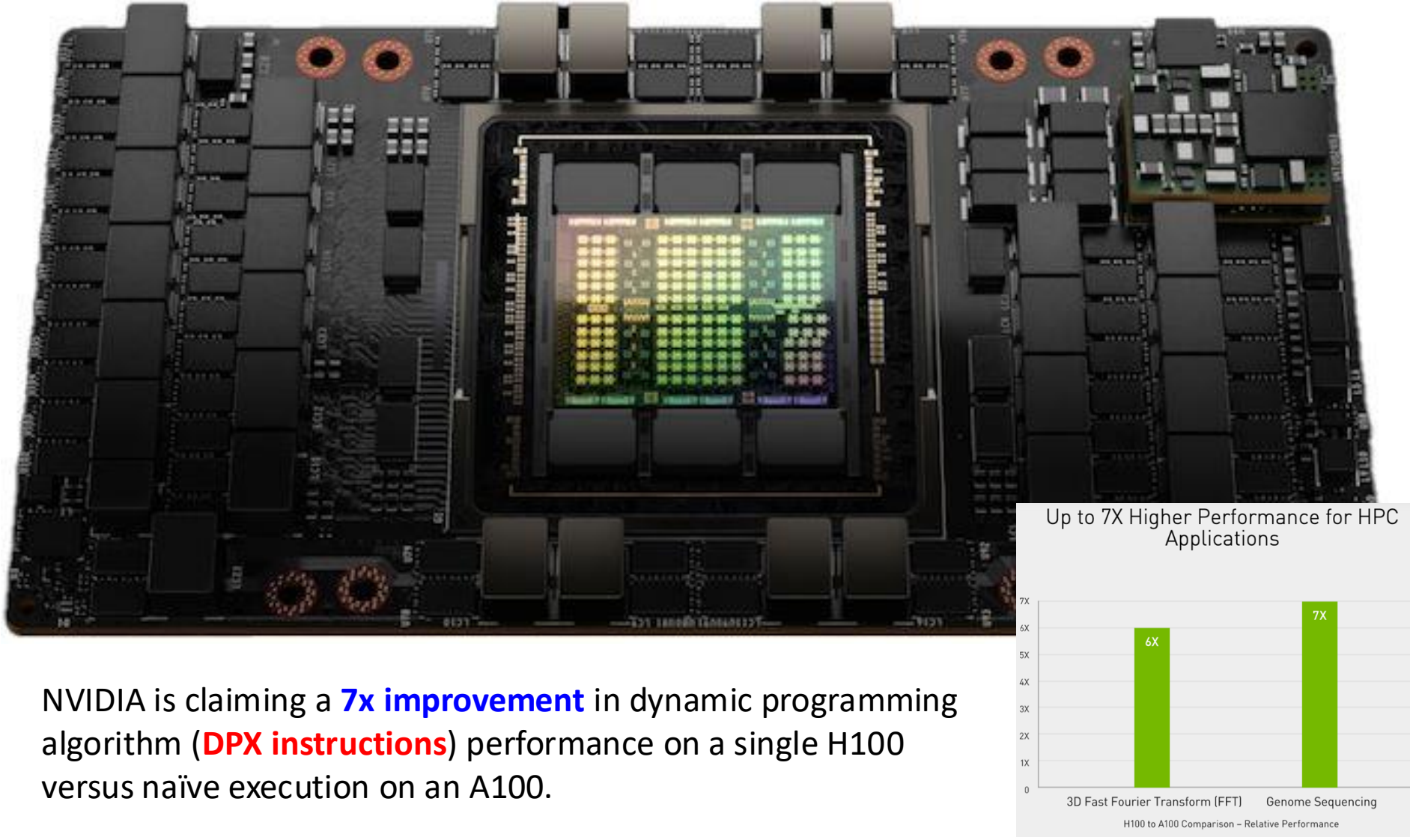
Reduce data footprint, manage and store data easily with lower costs and lower energy consumption, with built-in compression that reduces FASTQ file sizes by up to 80%.2

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

https://www.illumina.com/content/dam/illumina/gcs/assembled-assets/marketing-literature/dragen-bio-it-data-sheet-m-gl-00680/dragen-bio-it-data-sheet-m-gl-00680.pdf

https://emea.illumina.com/systems/sequencing-platforms/novaseq-x-plus/products-services/software.html

NVIDIA H100 (2022)



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

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

Bionano & NVIDIA:

Accelerating Analysis for Fast Time to Results

Technological solution to **support higher throughput**

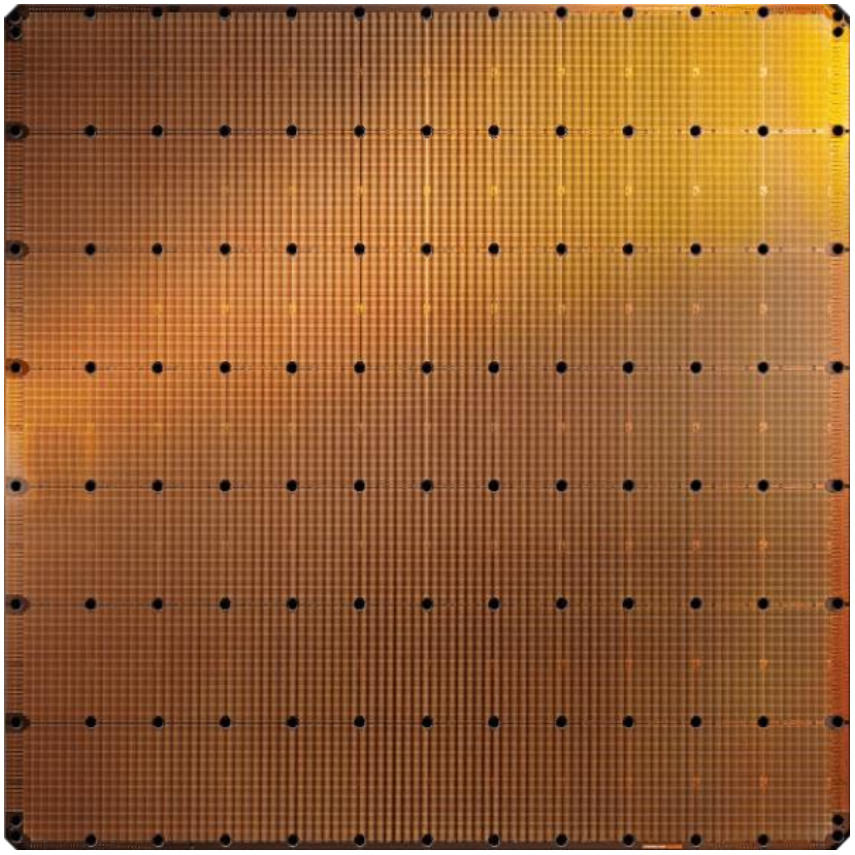
New high-performance algorithms from Bionano

Powered by NVIDIA RTX™ 6000 Ada Generation GPUs

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

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

Cerebras's Wafer Scale Engine (2024)



- The largest ML accelerator chip (2024)
- 900,000 cores

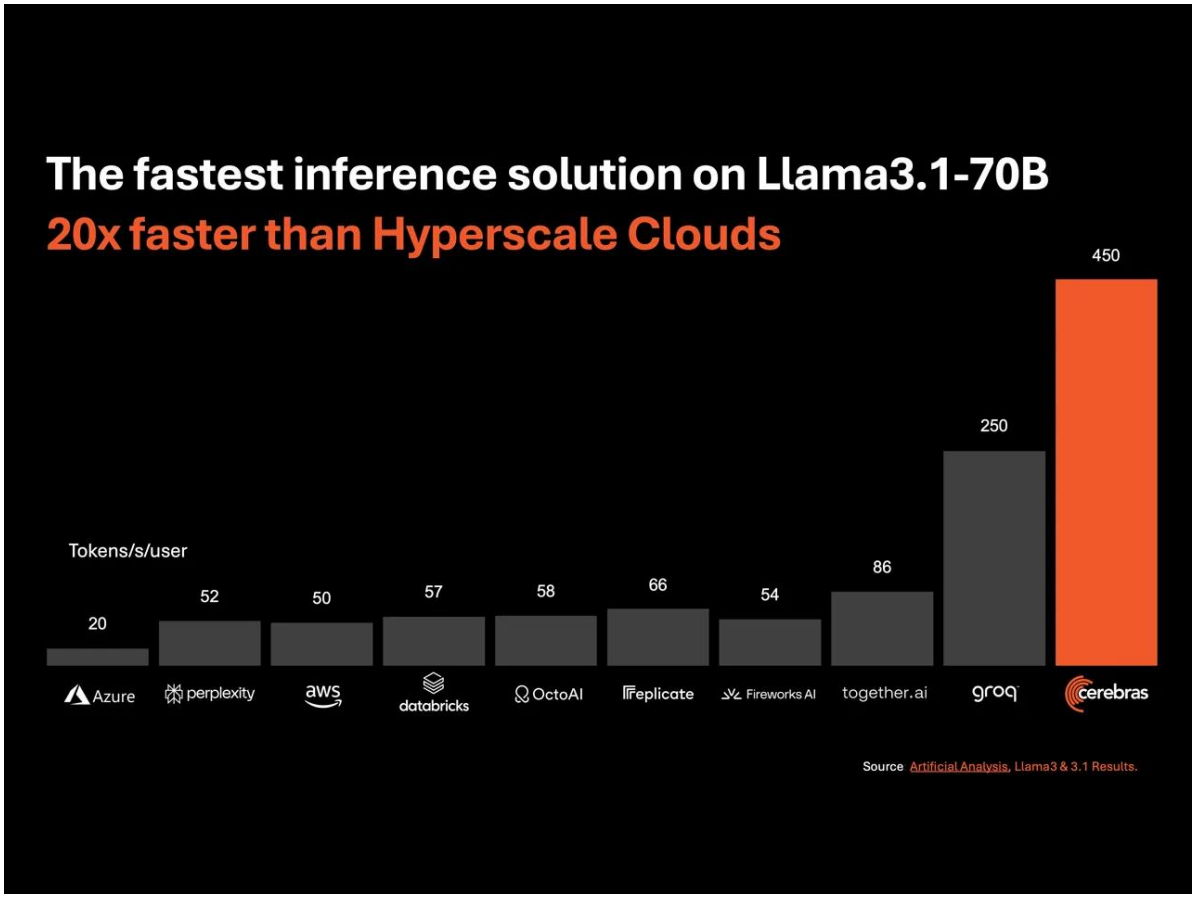
Cerebras WSE-3
~4 Trillion transistors
46,225 mm^2

NVIDIA H100 GPU
80 Billion transistors
814 mm^2

https://cerebras.ai/blog/cerebras-cs3

https://www.cerebras.net/cerebras-wafer-scale-engine-why-we-need-big-chips-for-deep-learning/

Cerebras's Wafer Scale Engine (2024)

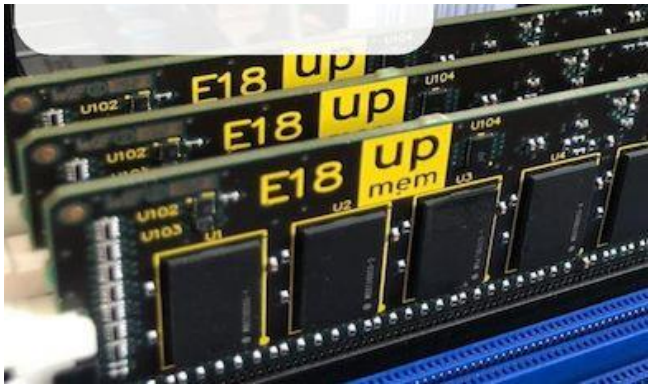


https://cerebras.ai/blog/cerebras-cs3

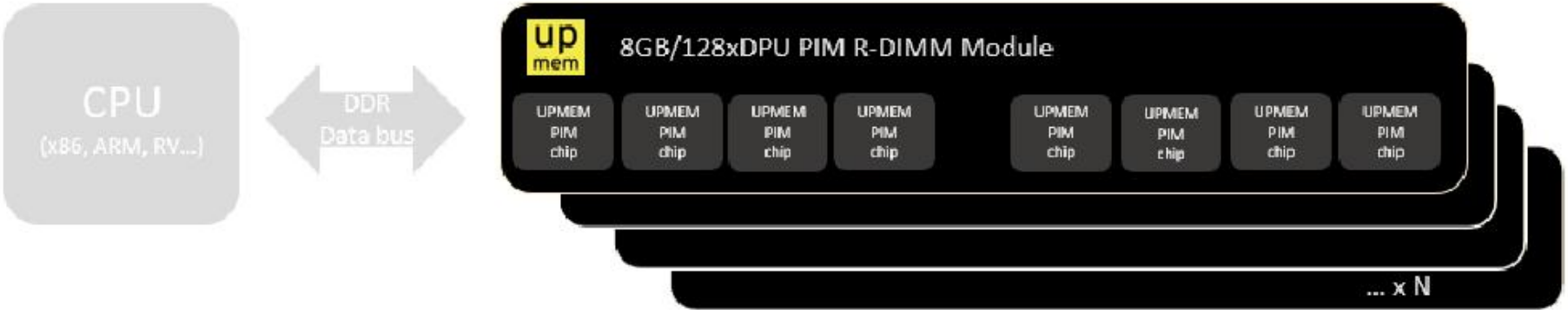
https://cerebras.ai/press-release/cerebras-launches-the-worlds-fastest-ai-inference

UPMEM Processing-in-DRAM Engine (2019)

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



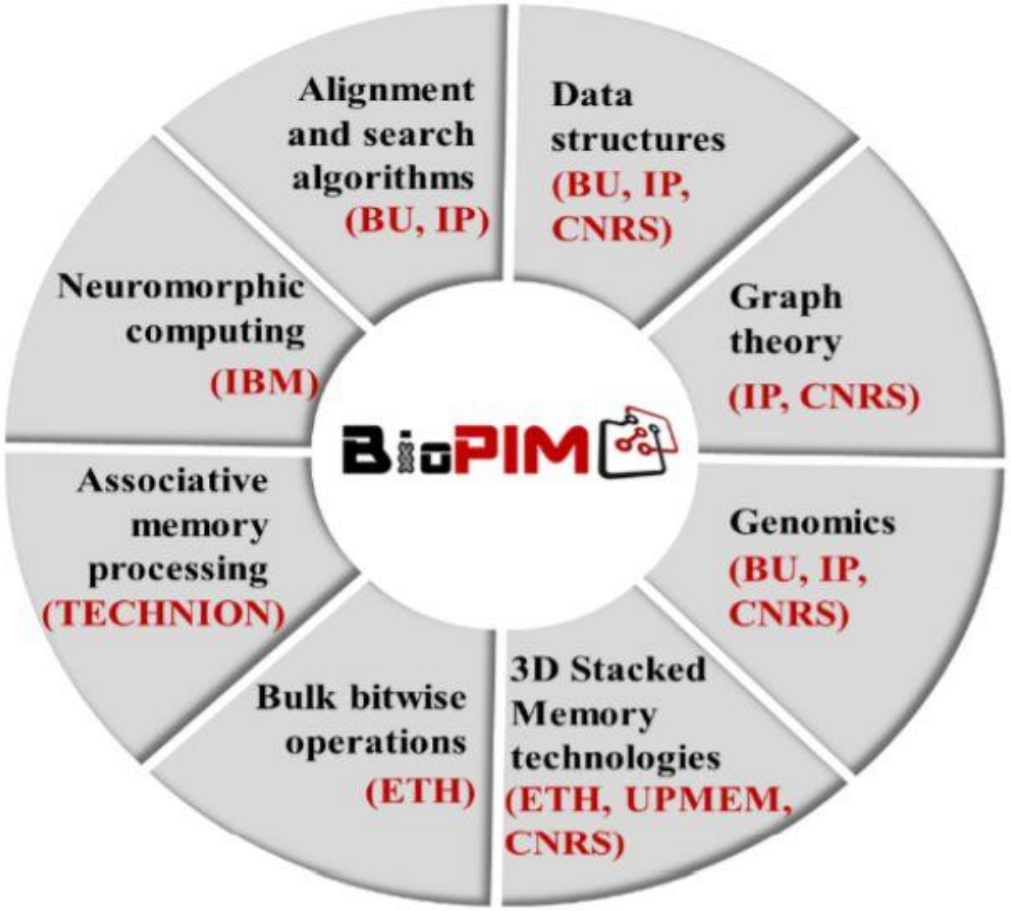
- Replaces **standard** DIMMs
  - DDR4 R-DIMM modules
    - 8GB+128 DPUs (16 PIM chips)
    - Standard 2x-nm DRAM process
  - **Large amounts of** compute & memory bandwidth



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

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

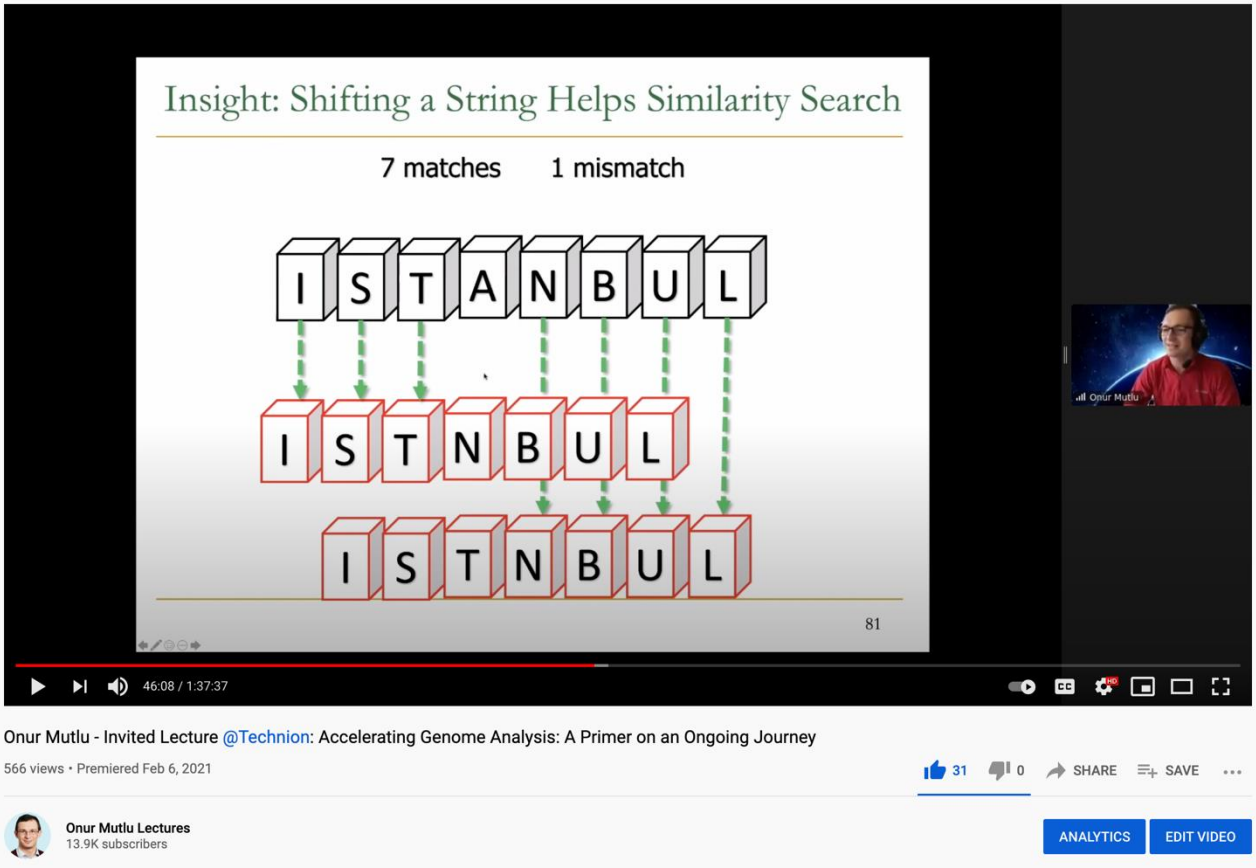
BioPIM (2022)



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

Fast Genome Analysis…

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



More on Fast Genome Analysis…

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

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

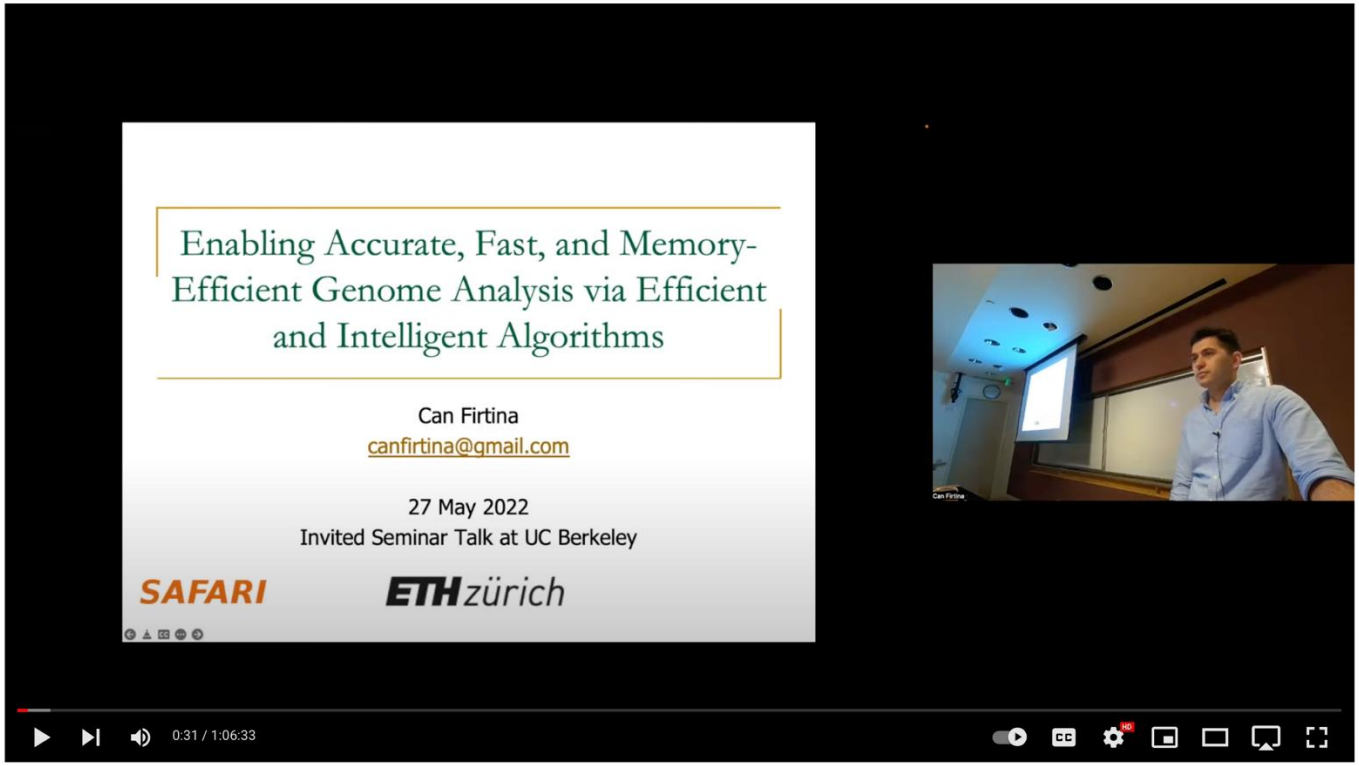
Onur Mutlu Lectures
36.6K subscribers
Analytics Edit video
Share Save

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

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

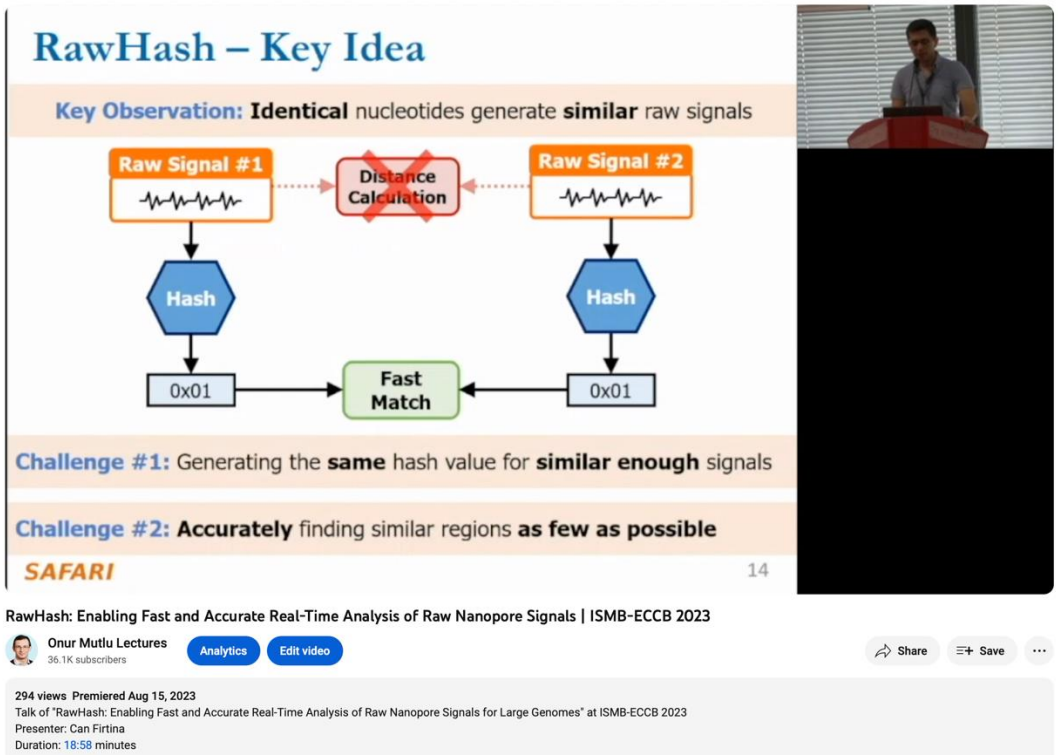
More on Accelerating Genome Analysis

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



More on Real-Time Genome Analysis

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



Accelerating Genome Analysis [DAC 2023]

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

BIO-Arch Workshop at RECOMB 2023

- April 14, 2023

BIO-Arch: Workshop on Hardware Acceleration of Bioinformatics Workloads

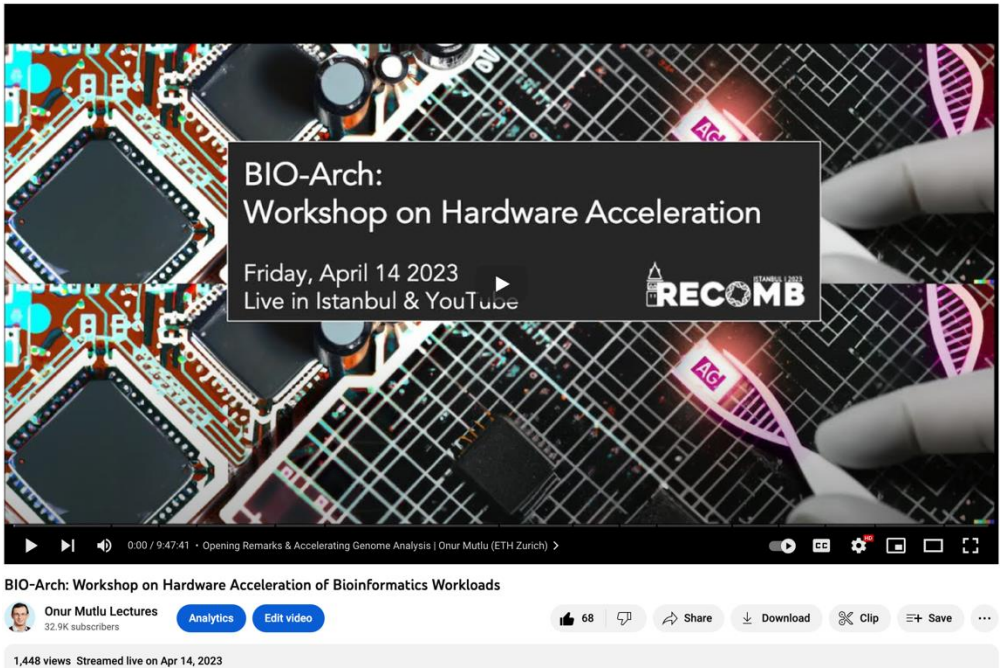
About

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

Venue

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

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



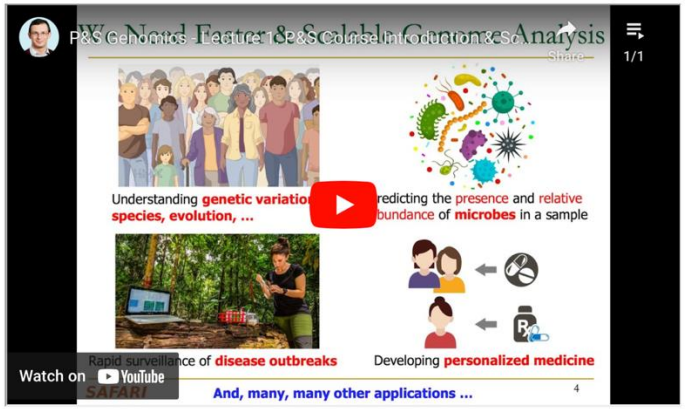
https://www.youtube.com/watch?v=2rCsb4-nLmg

Genomics Course (Spring 24)

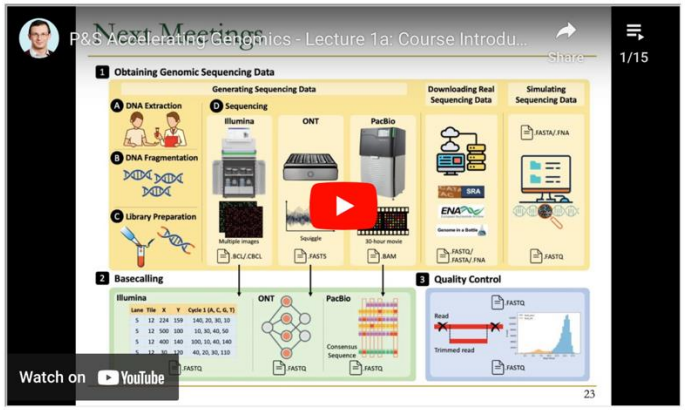
- **Spring 2024 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/spring2024/doku.php?id=bioinformatics
- **Fall 2023 Edition:**
  - https://safari.ethz.ch/projects_and_seminars/fall2023/doku.php?id=bioinformatics
- **Youtube Livestream (Fall 2023):**
  - https://youtube.com/playlist?list=PL5Q2soXY2Zi_O0wyOjiMShG4t2QPZoeE3
- Project course
  - Taken by Bachelor's students
  - Genomics lectures
  - Hands-on research exploration
  - Many research readings

https://www.youtube.com/onurmutlulectures

Lecture Playlist (Fall 2023):



Complete Lecture Playlist (Spring 2023):



Edit

Fall 2023 Schedule

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

Conclusion

- We covered various recent ideas to
  - Accelerate genome analysis
  - Analyze genomes in ways that were not possible before

- Enabling cost-effective, portable, fast, and accurate genome analysis has many implications
  - What are the new applications to enable with these unique benefits?

- Can we do even better?
  - Understanding and modifying the sequencing process for analyzing other types of biological data

- **Many future opportunities exist**
  - **Especially with new sequencing technologies**
  - **Especially with new applications and use cases**

Enabling Fast, Accurate, and Efficient Real-Time Genomic Sequence Analysis via New Algorithms and Architectures

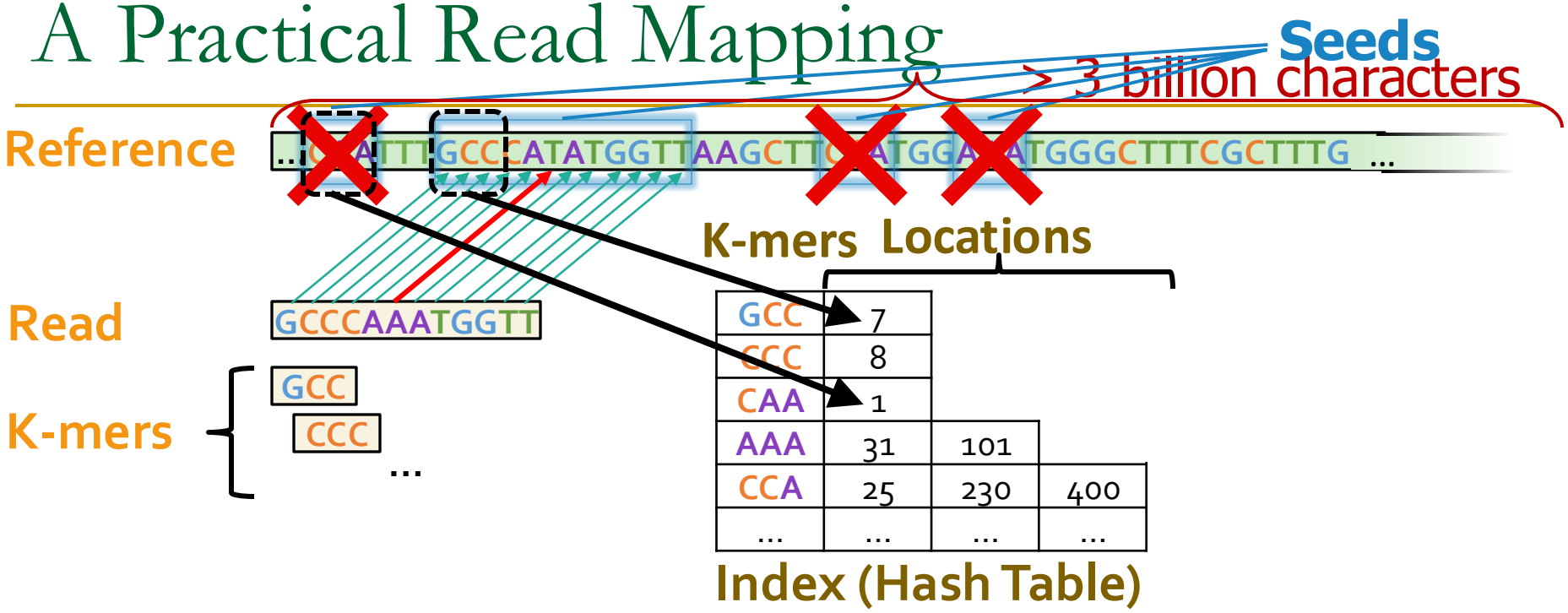
Can Firtina
canfirtina@gmail.com
https://cfirtina.com

13 September 2024
AMD

SAFARI

ETH zürich

A Practical Read Mapping



Indexing	Store certain k-mers with their positions for fast query
Seeding	Determine potential matching regions (seeds)
Seed Filtering	Prune uninformative/unreliable seeds
Alignment	Determine the exact differences

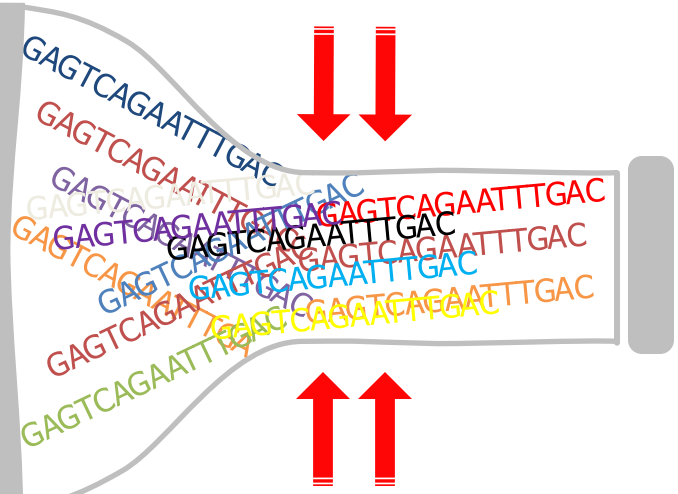
Analysis is Bottlenecked in Read Mapping!!

48 Human whole genomes

at 30× coverage

in about 2 days

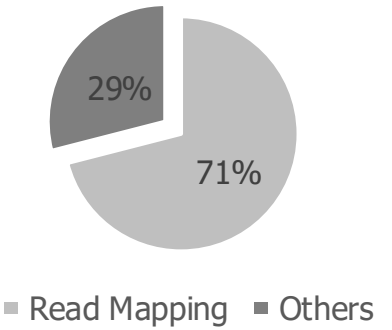
Illumina NovaSeq 6000



1 Human genome

32 CPU hours

on a 48-core processor



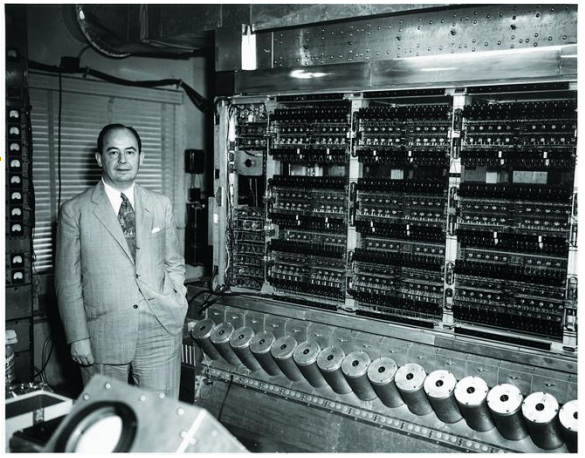
Goyal+, "Ultra-fast next generation human genome sequencing data processing using DRAGENTM bio-IT processor for precision medicine", *Open Journal of Genetics*, 2017.

A Tsunami of Sequencing Data

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

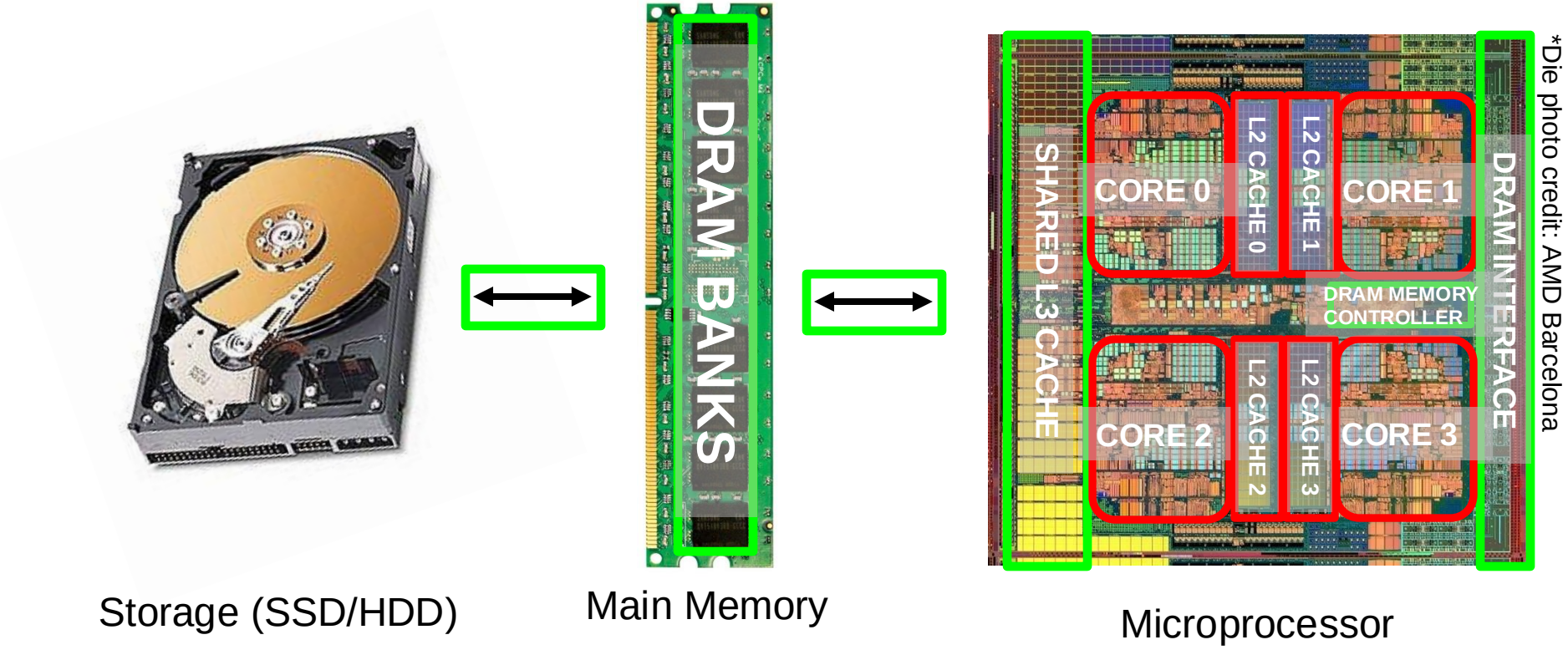
Source: @kyrpides

Today's Computing Systems



von Neumann model, 1945

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



Storage (SSD/HDD)

Main Memory

Microprocessor

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

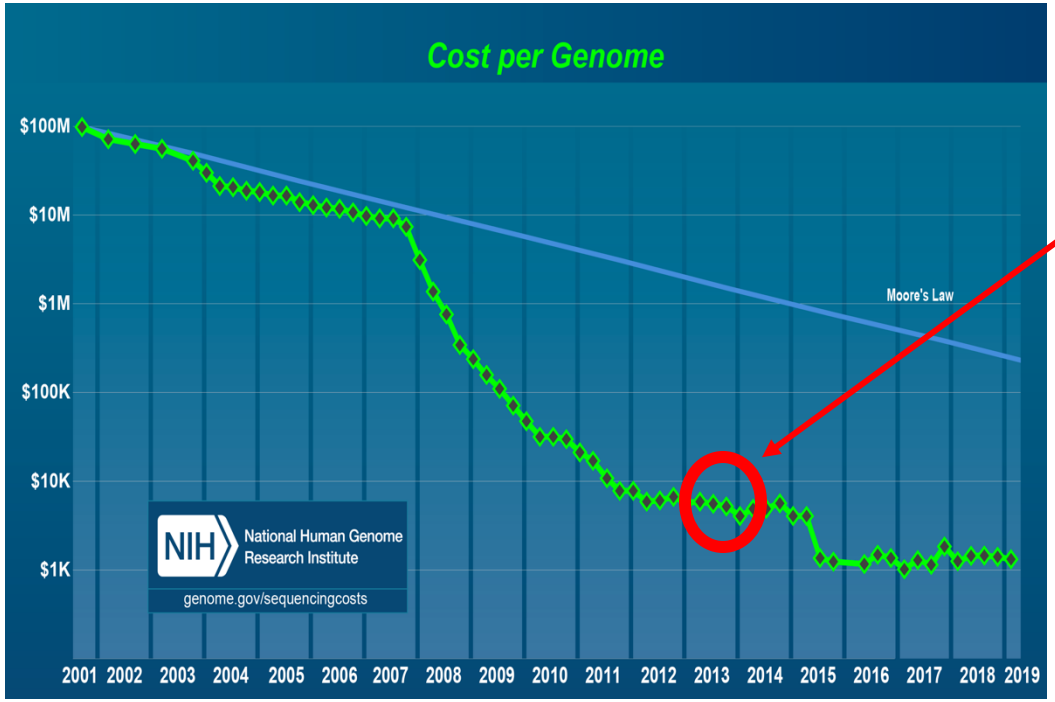
The Problem

Data analysis

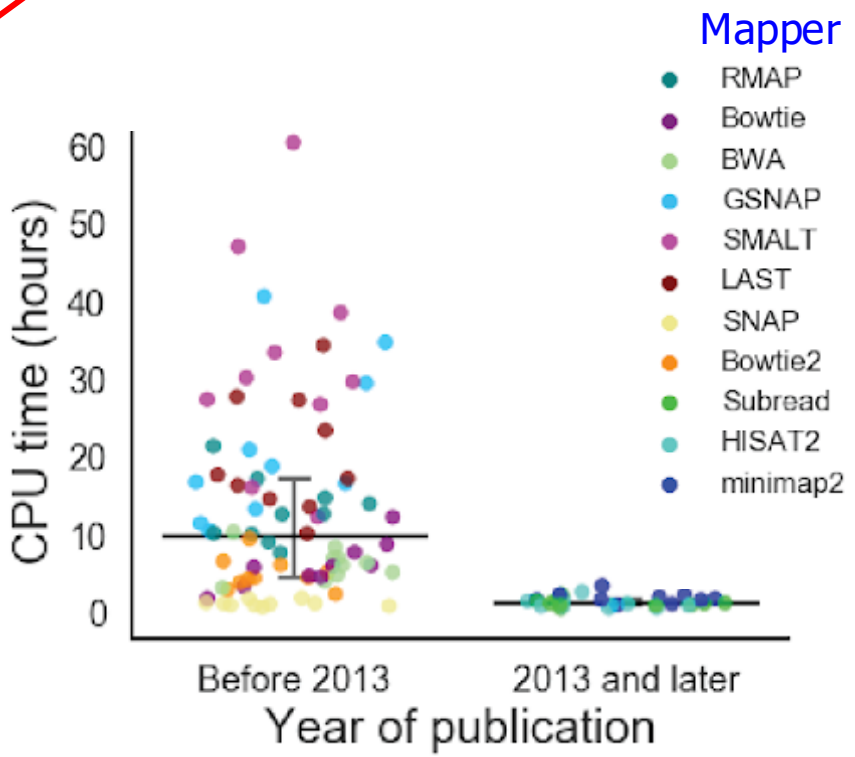
is performed

far away from the data

The Need for Speed



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



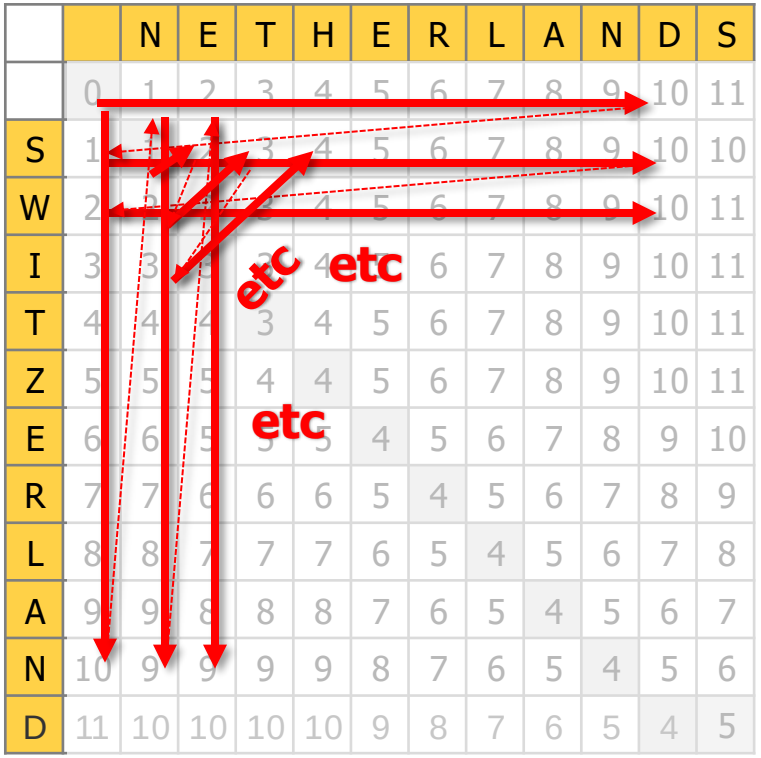
Alser+, "Technology dictates algorithms: Recent developments in read alignment", Genome Biology, 2021

Sequence Alignment in Unavoidable

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

  Enumerating all possible prefixes

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



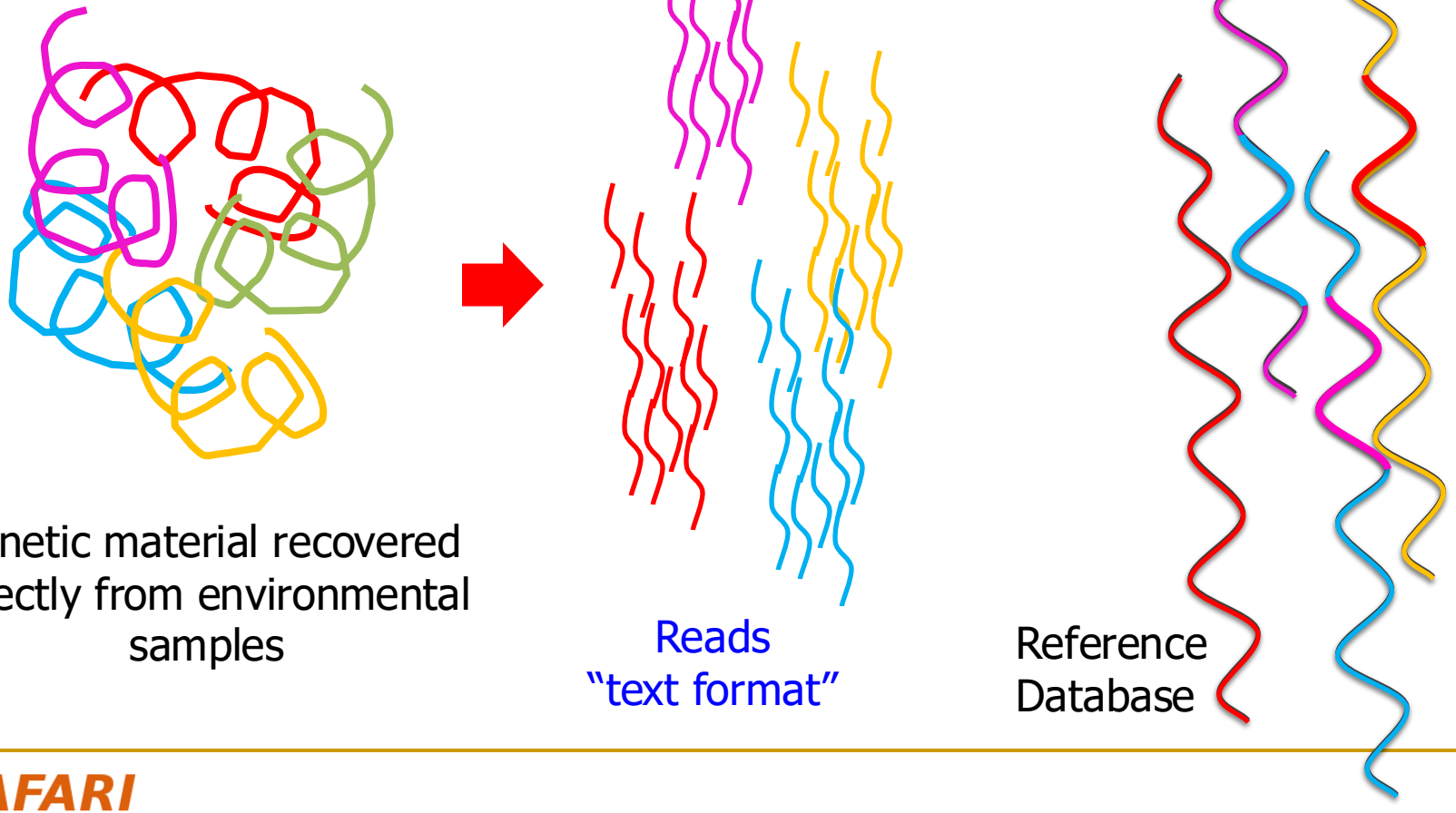
Sequence Alignment in Unavoidable

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

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

Metagenomics Analysis

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

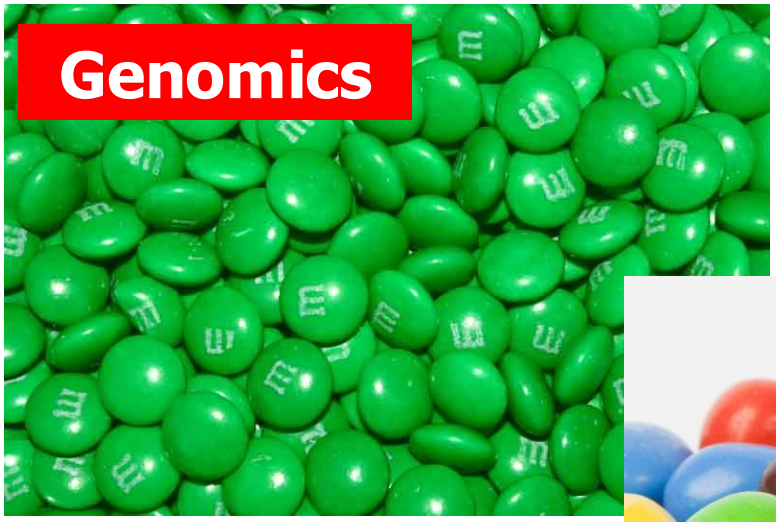


genetic material recovered directly from environmental samples

Reads "text format"

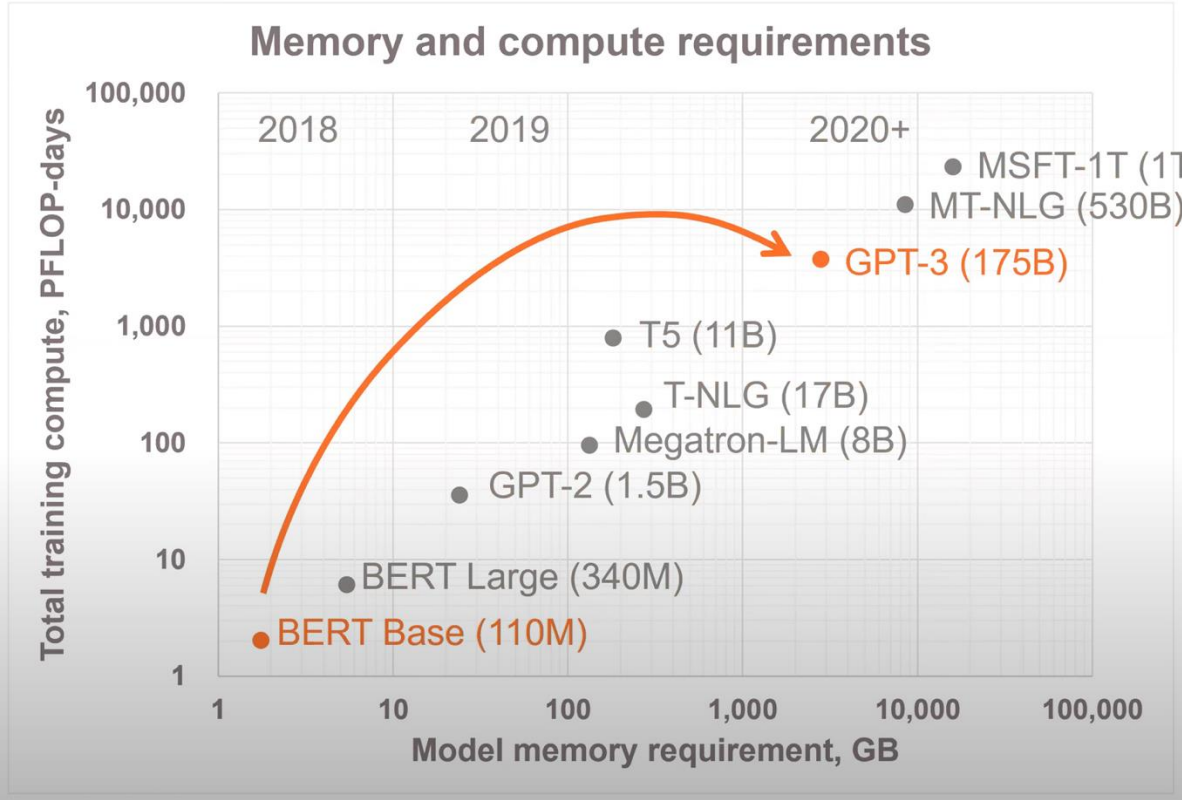
Reference Database

Genomics vs. Metagenomics



Huge Demand for Performance & Efficiency

Exponential Growth of Neural Networks



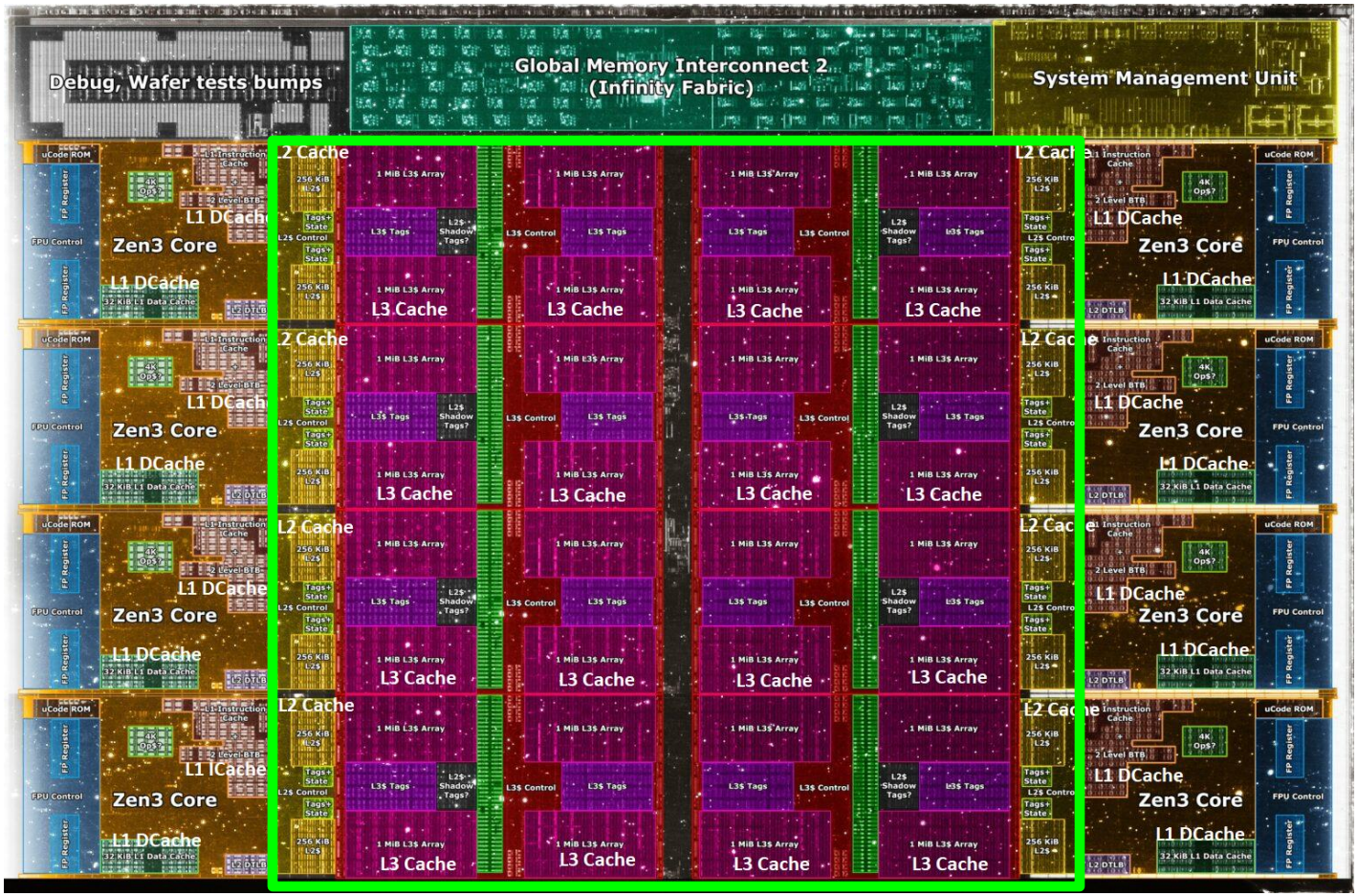
1800x more compute

In just **2 years**

Tomorrow, **multi-trillion** parameter models

Source: https://youtu.be/Bh13Idwcb0Q?t=283

Deeper and Larger Memory Hierarchies



AMD Ryzen 5000, 2020

Core Count:
8 cores/16 threads

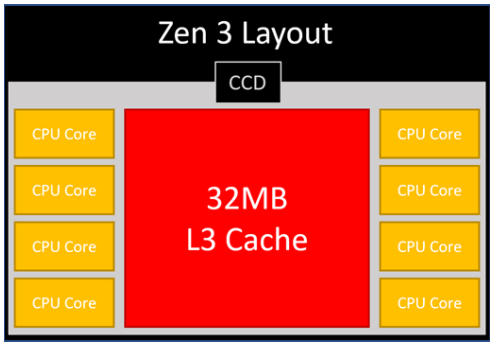
L1 Caches:
32 KB per core

L2 Caches:
512 KB per core

L3 Cache:
32 MB shared

https://wccftech.com/amd-ryzen-5000-zen-3-vermeer-undressed-high-res-die-shots-close-ups-pictured-detailed/

AMD's 3D Last Level Cache (2021)

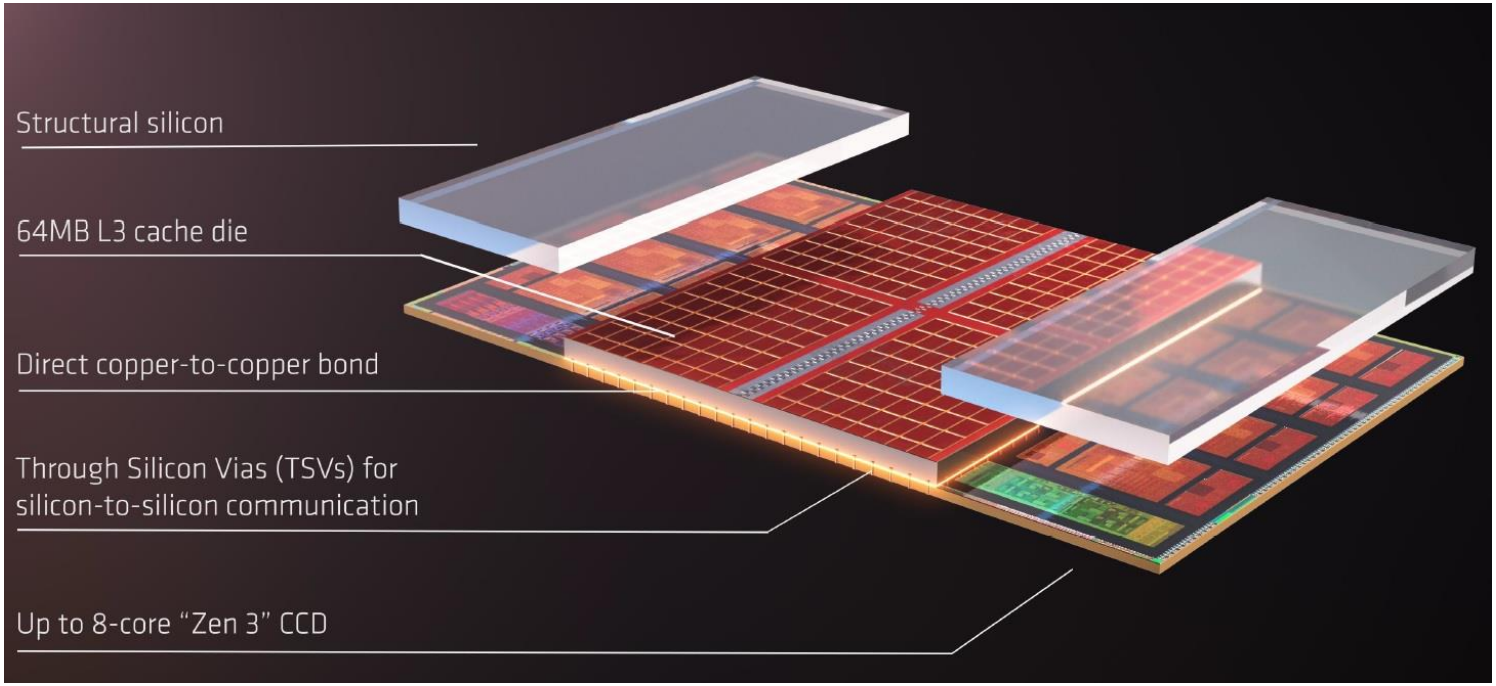


https://community.microcenter.com/discussion/5134/comparing-zen-3-to-zen-2

AMD increases the L3 size of their 8-core Zen 3 processors from 32 MB to 96 MB

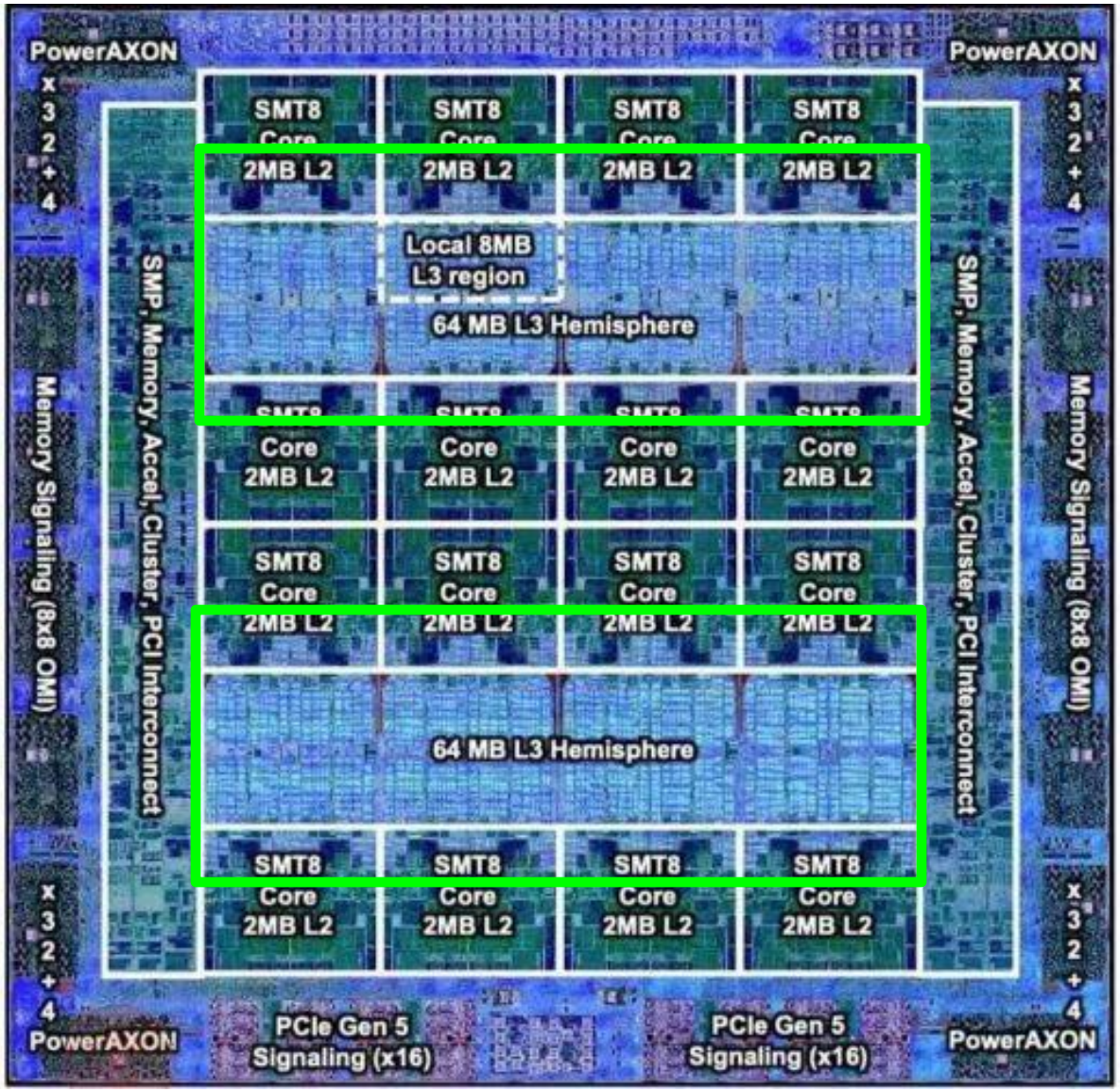
Additional 64 MB L3 cache die stacked on top of the processor die

- Connected using Through Silicon Vias (TSVs)
- Total of 96 MB L3 cache



https://youtu.be/gqAYMx34euU
https://www.tech-critter.com/amd-keynote-computex-2021/

Deeper and Larger Memory Hierarchies



IBM POWER10, 2020

Cores:

15-16 cores,
8 threads/core

L2 Caches:

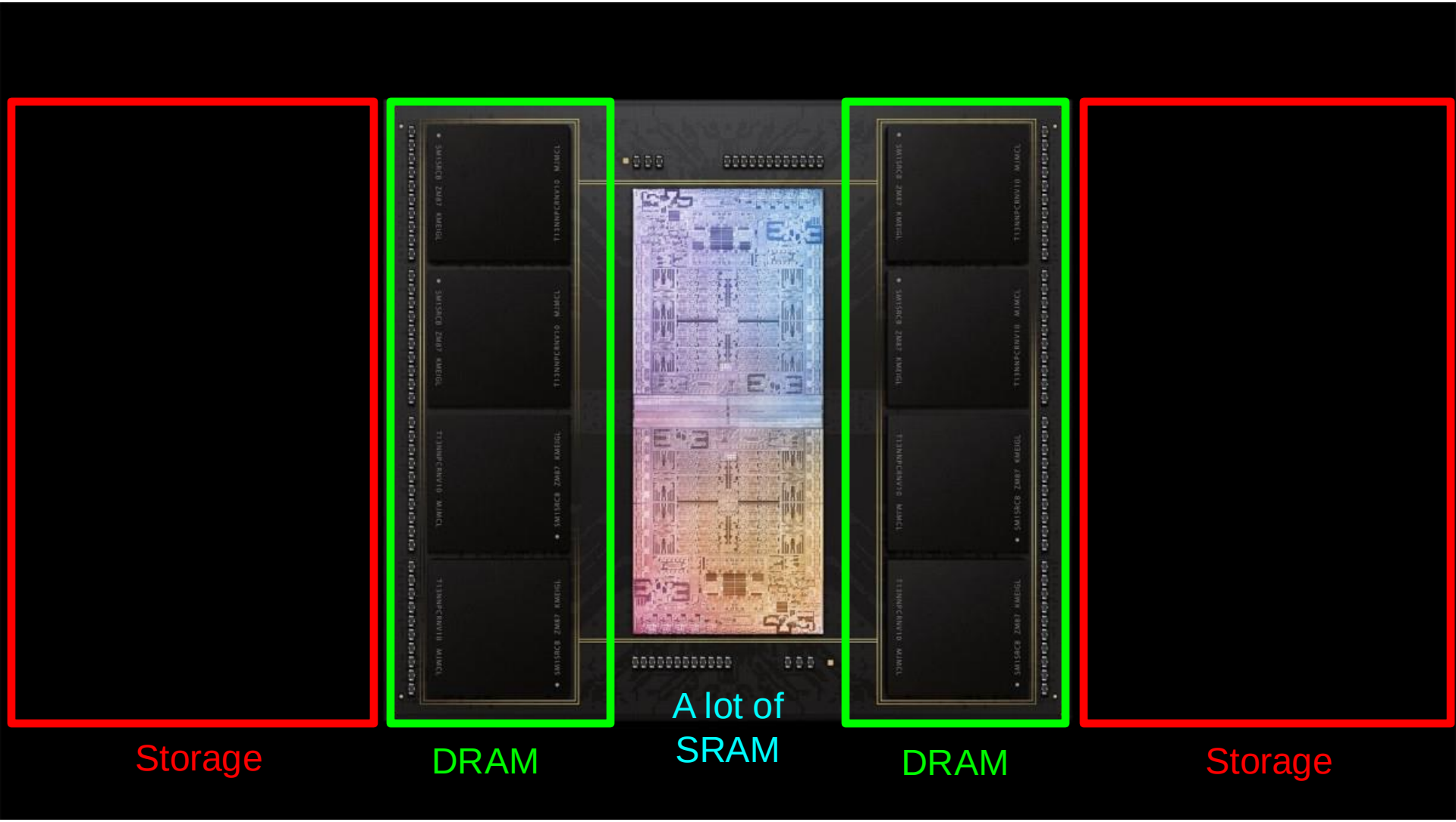
2 MB per core

L3 Cache:

120 MB shared

https://www.it-techblog.de/ibm-power10-prozessor-mehr-speicher-mehr-tempo-mehr-sicherheit/09/2020/

Deeper and Larger Memory Hierarchies



Apple M1 Ultra System (2022)

https://www.gsmarena.com/apple_announces_m1_ultra_with_20core_cpu_and_64core_gpu-news-53481.php

Data Movement Overwhelms Modern Machines

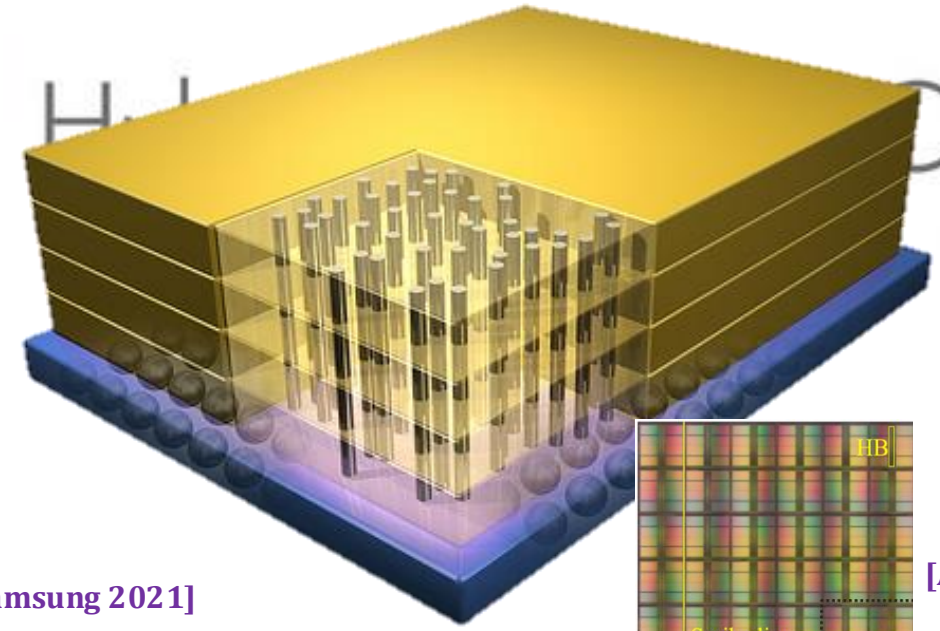
- Amirali Boroumand, Saugata Ghose, Youngsok Kim, Rachata Ausavarungnirun, Eric Shiu, Rahul Thakur, Daehyun Kim, Aki Kuusela, Allan Knies, Parthasarathy Ranganathan, and Onur Mutlu, **"Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks"** *Proceedings of the 23rd International Conference on Architectural Support for Programming Languages and Operating Systems (**ASPLOS**)*, Williamsburg, VA, USA, March 2018.

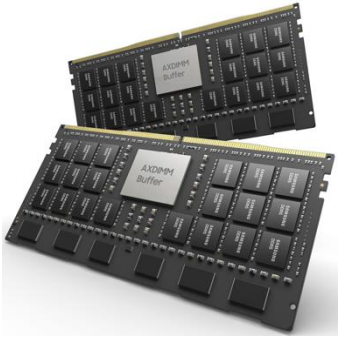
62.7% of the total system energy is spent on data movement

Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks

Amirali Boroumand[1] Saugata Ghose[1] Youngsok Kim[2]
Rachata Ausavarungnirun[1] Eric Shiu[3] Rahul Thakur[3] Daehyun Kim[4,3]
Aki Kuusela[3] Allan Knies[3] Parthasarathy Ranganathan[3] Onur Mutlu[5,1]

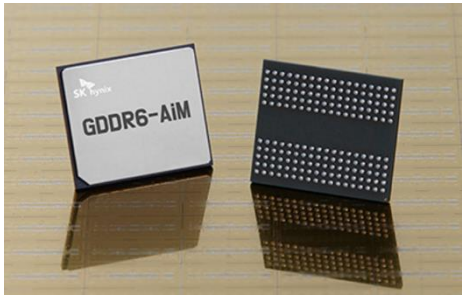
Processing-in-Memory Landscape Today



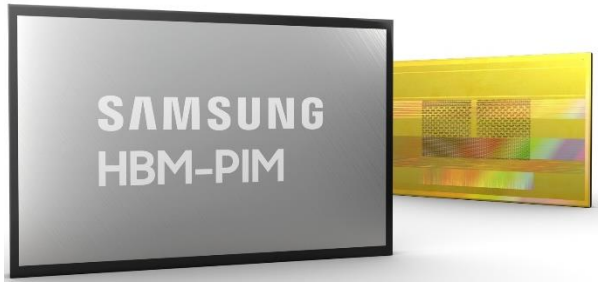


[Samsung 2021]

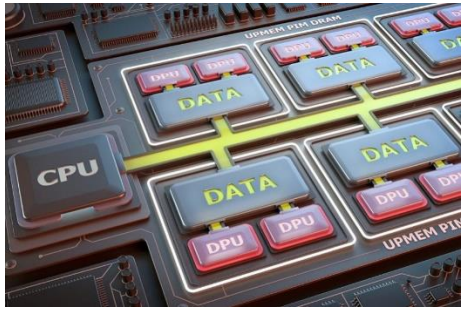
[Alibaba 2022]



[SK Hynix 2022]



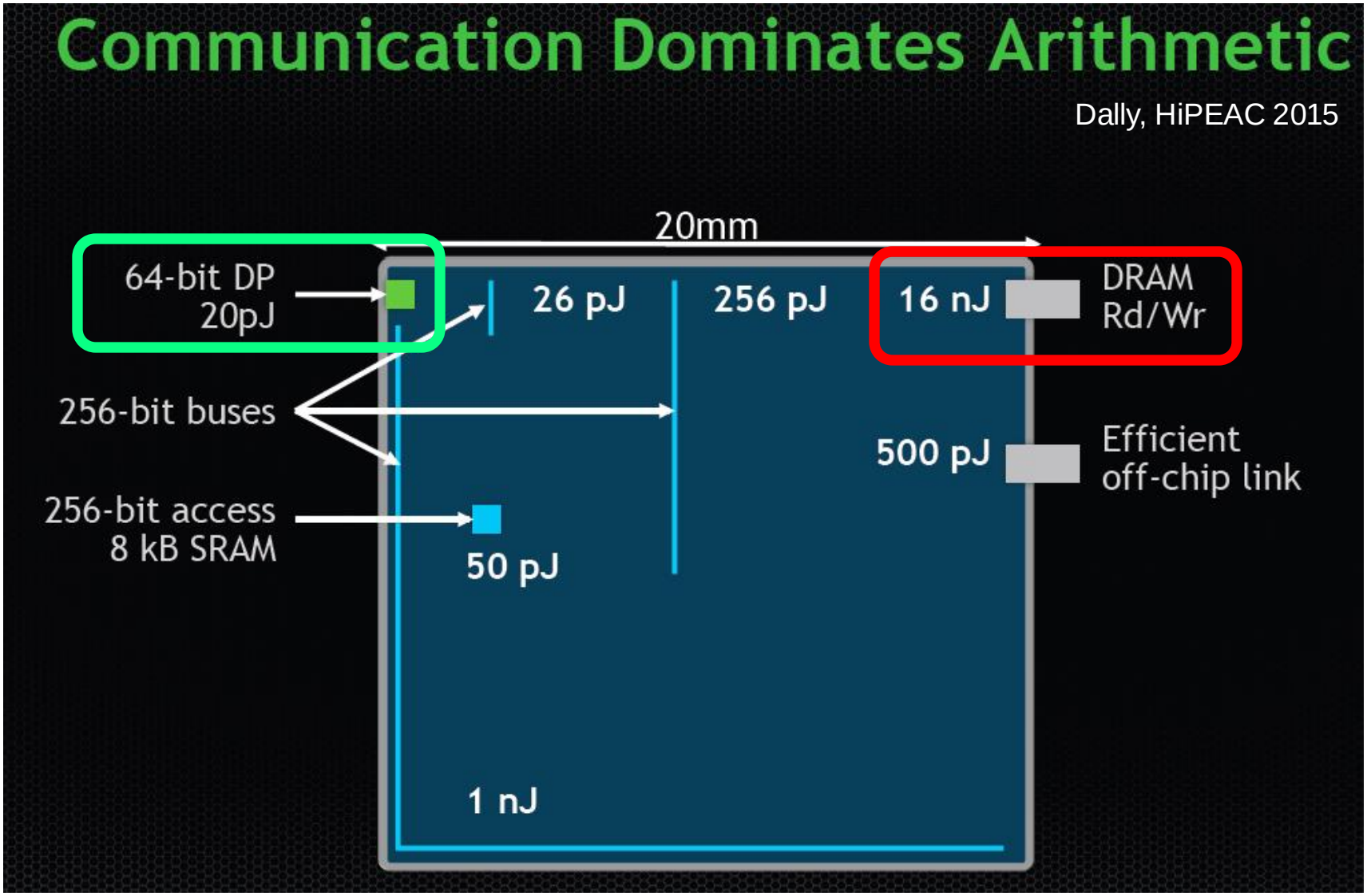
[Samsung 2021]



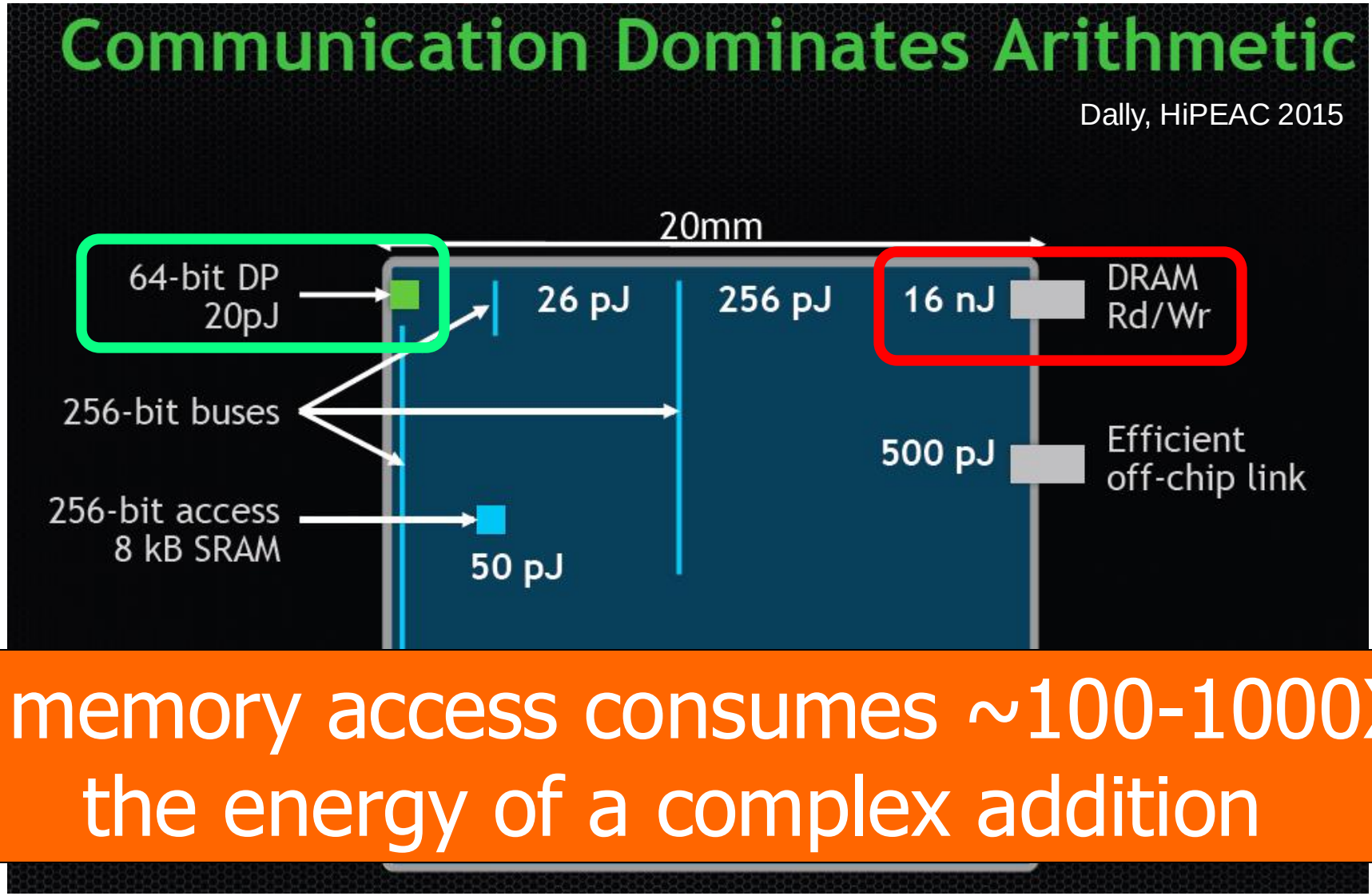
[UPMEM 2019]

And, many other experimental chips and startups

The Energy Perspective

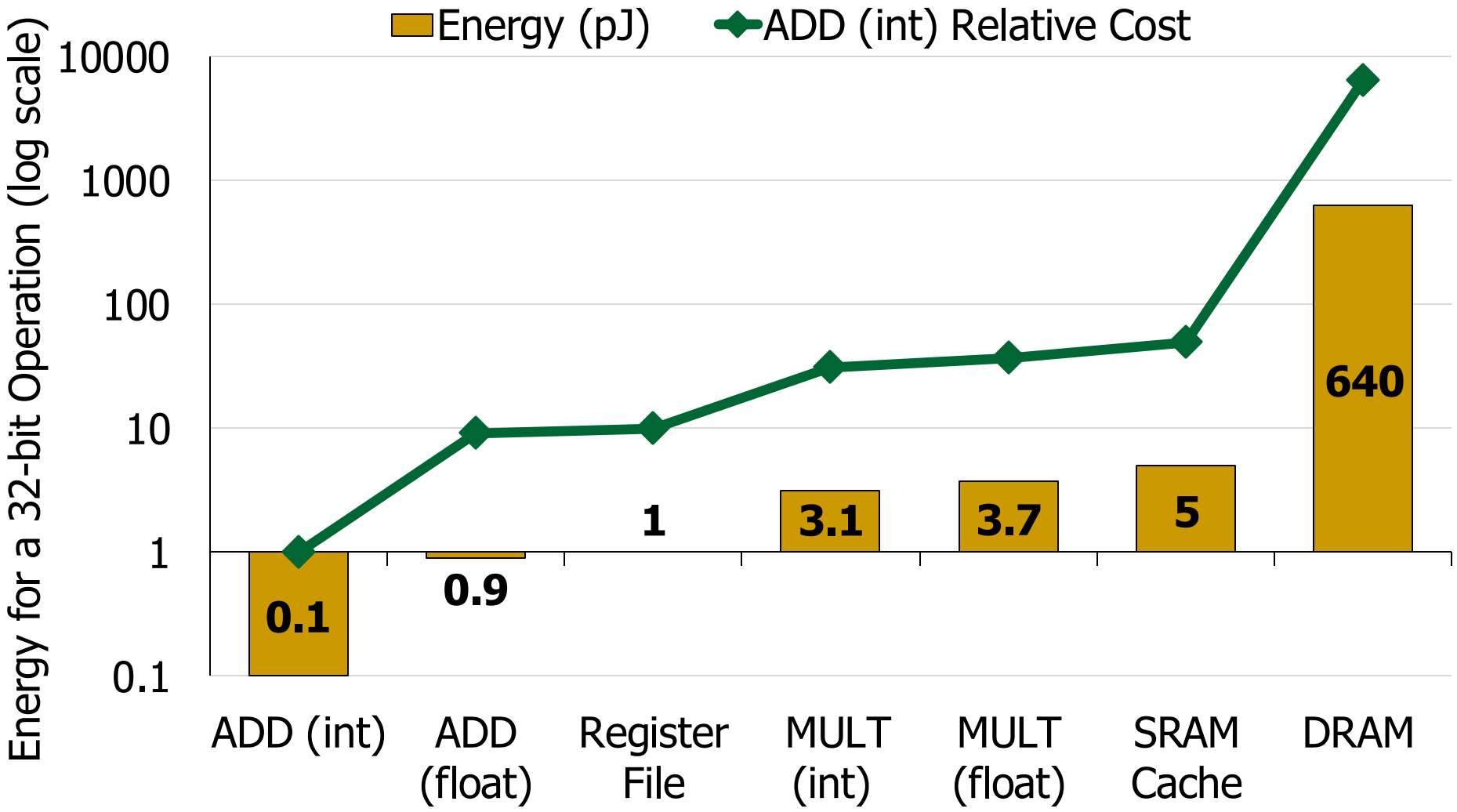


Data Movement vs. Computation Energy



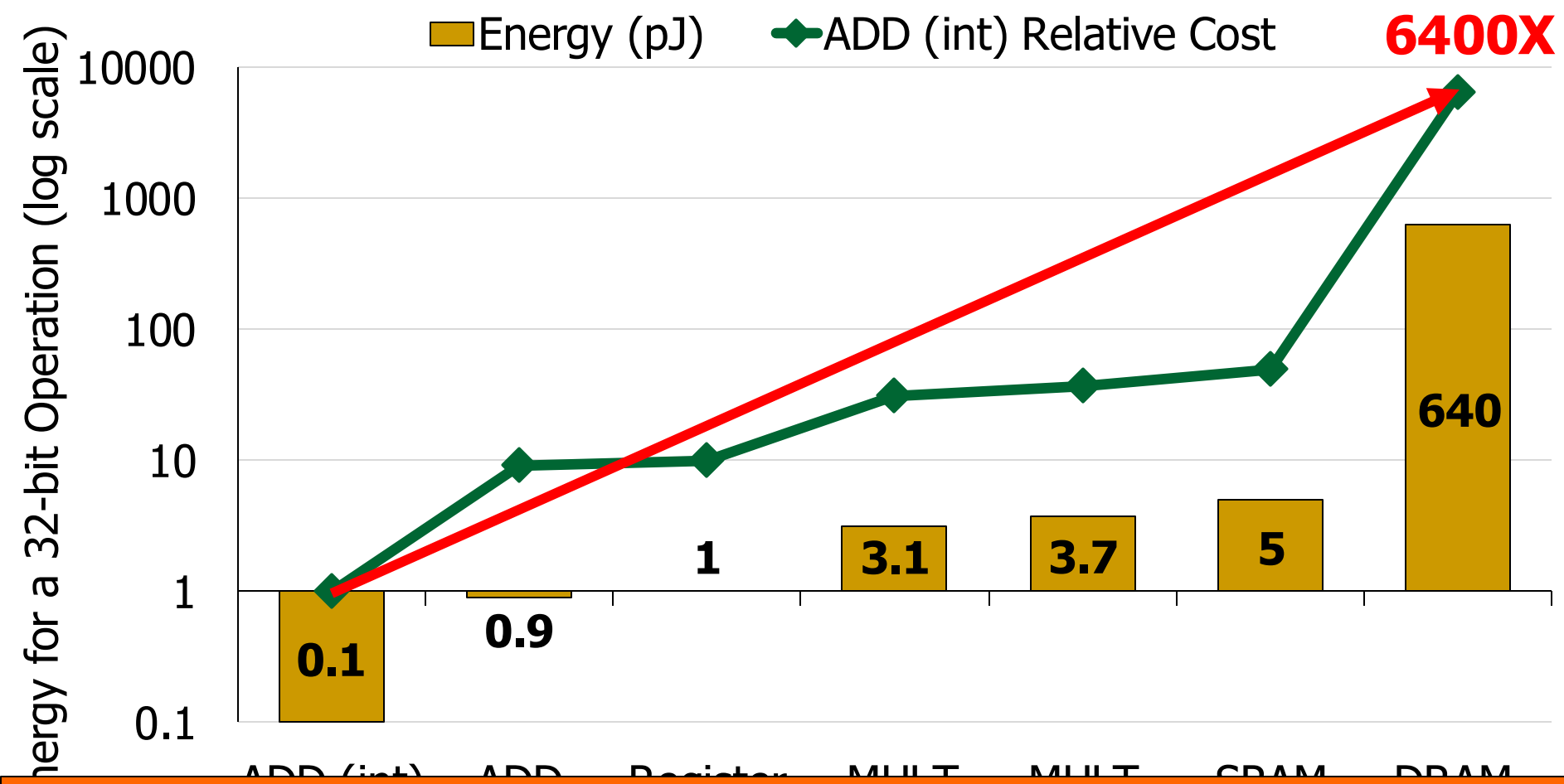
A memory access consumes ~100-1000X the energy of a complex addition

Data Movement vs. Computation Energy



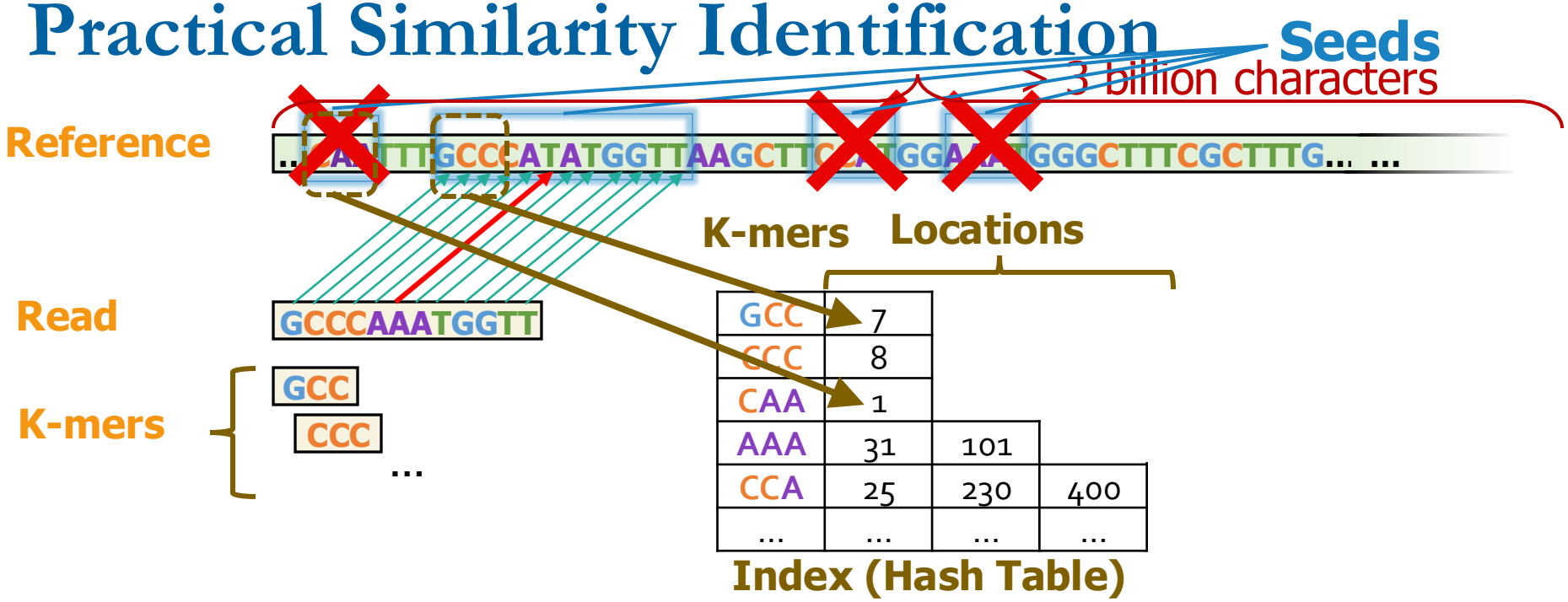
Han+, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," ISCA 2016.

Data Movement vs. Computation Energy



A memory access consumes 6400X the energy of a simple integer addition

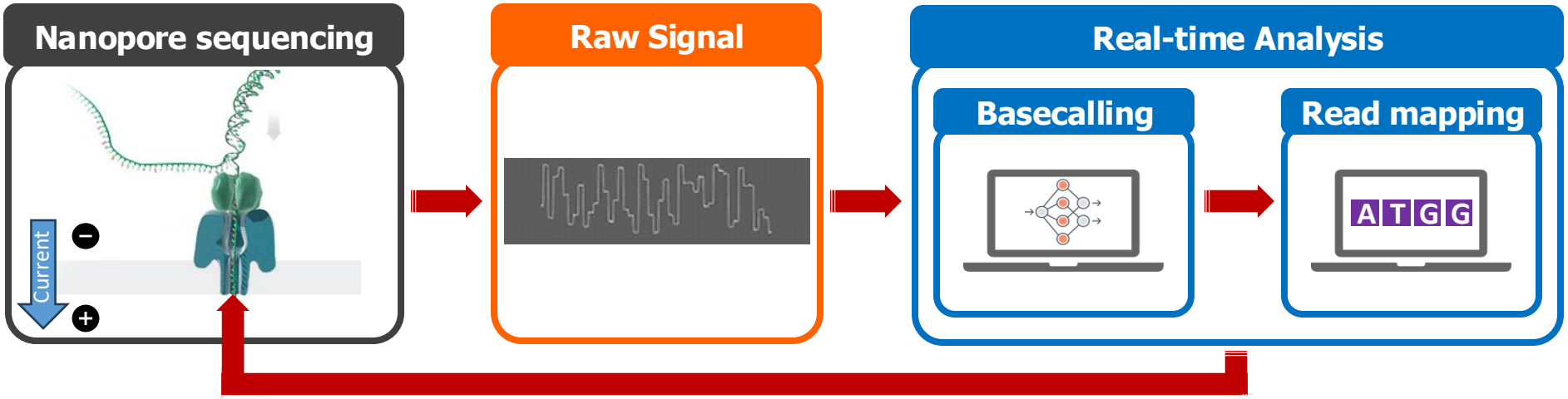
Practical Similarity Identification



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

Existing Solutions – Real-time Basecalling

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

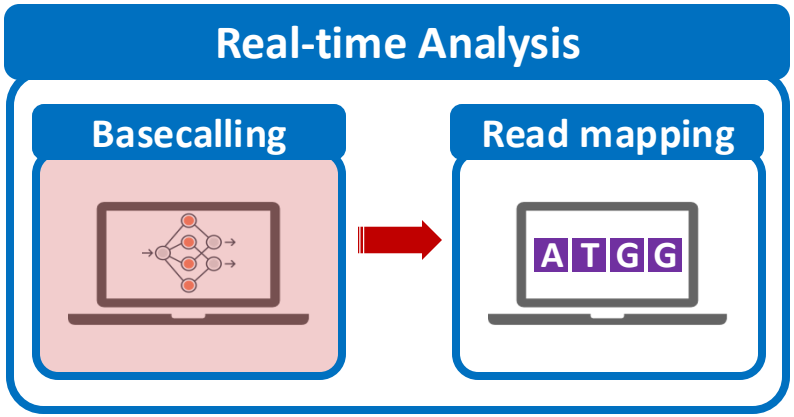


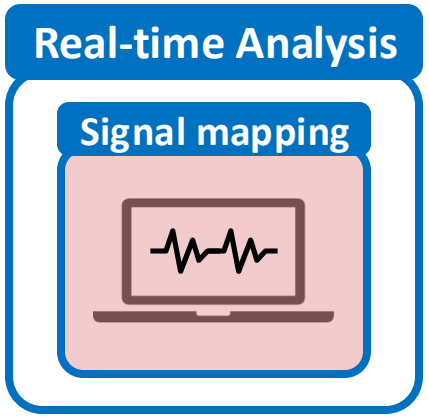
DNNs provide **less noisy analysis** from basecalled sequences

Costly and power-hungry computational requirements

The Problem

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





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

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

Applications of Read Until

Depletion: Reads mapping to a particular reference genome is ejected

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

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

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

Applications of Run Until and Sequence Until

Run Until: Stopping the sequencing without informative decision from analysis

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

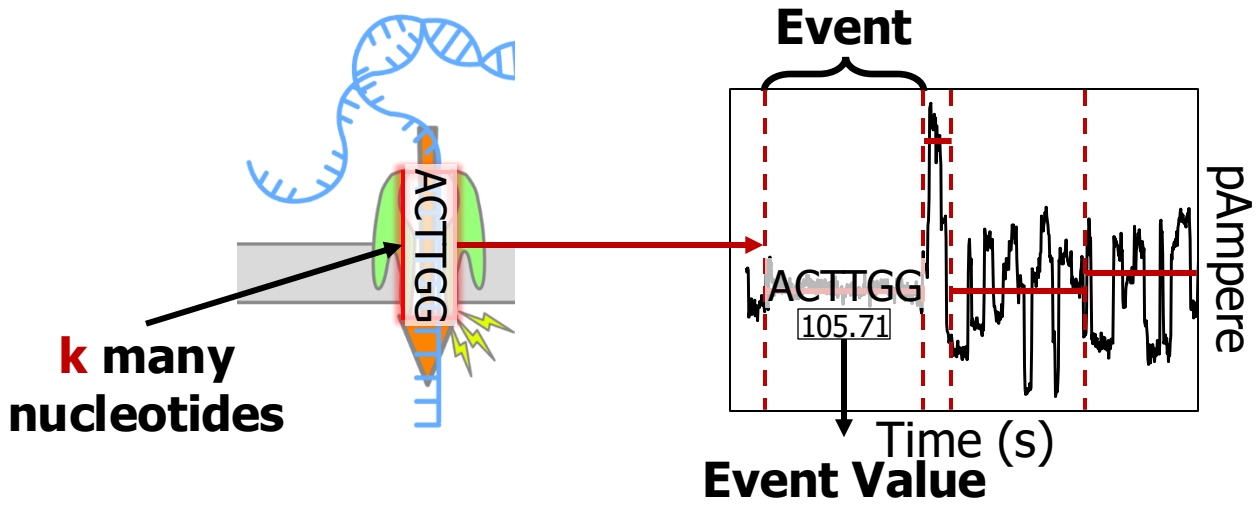
Sequence Until: Stopping the sequencing based on information decision

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

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
  - Abrupt signal changes show sequencing of a new k-mer
  - **Statistical methods** can find these abrupt changes
  - **Event value:** average of signals **within an event**

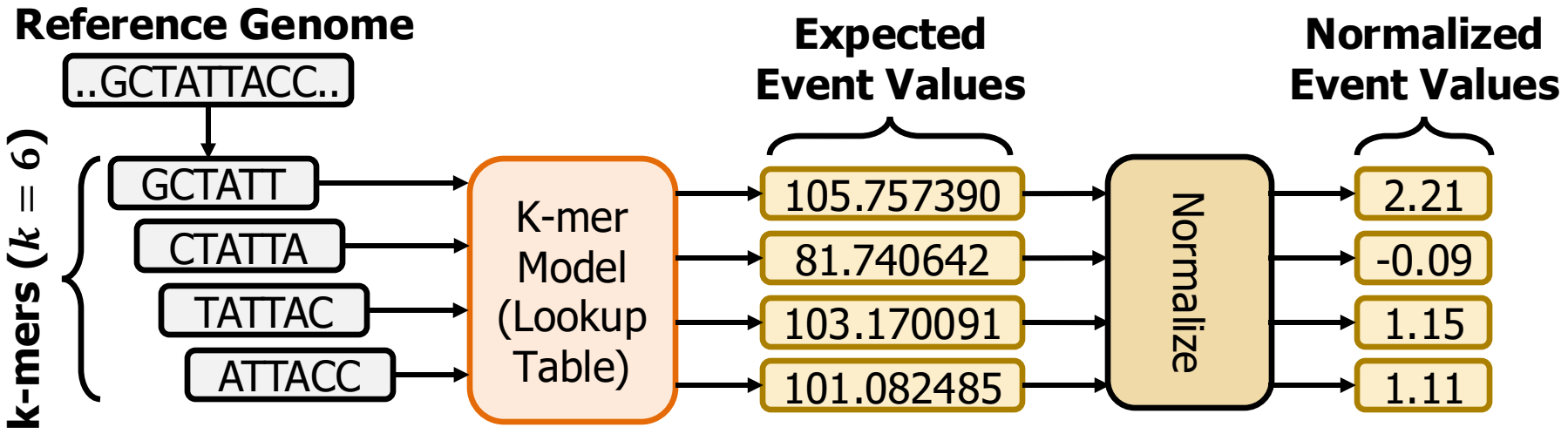
- **Observation: Identical** k-mers generate **similar** event values during sequencing



Details: Reference-to-Event Conversion

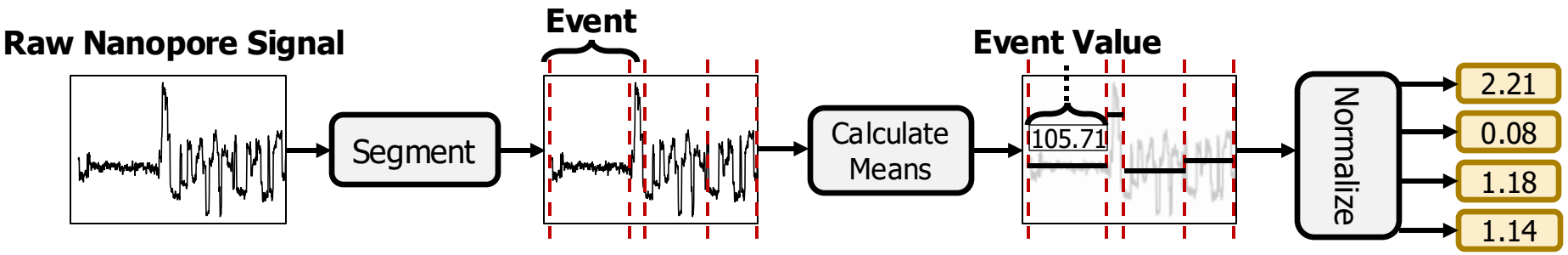
- **K-mer model:** Provides **expected** event values **for each k-mer**
  - Preconstructed based on nanopore sequencer characteristics

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



Details: Signal-to-Event Conversion

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



- Consecutive events ➔ consecutive k-mers

Details: Signal-to-Event Conversion

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

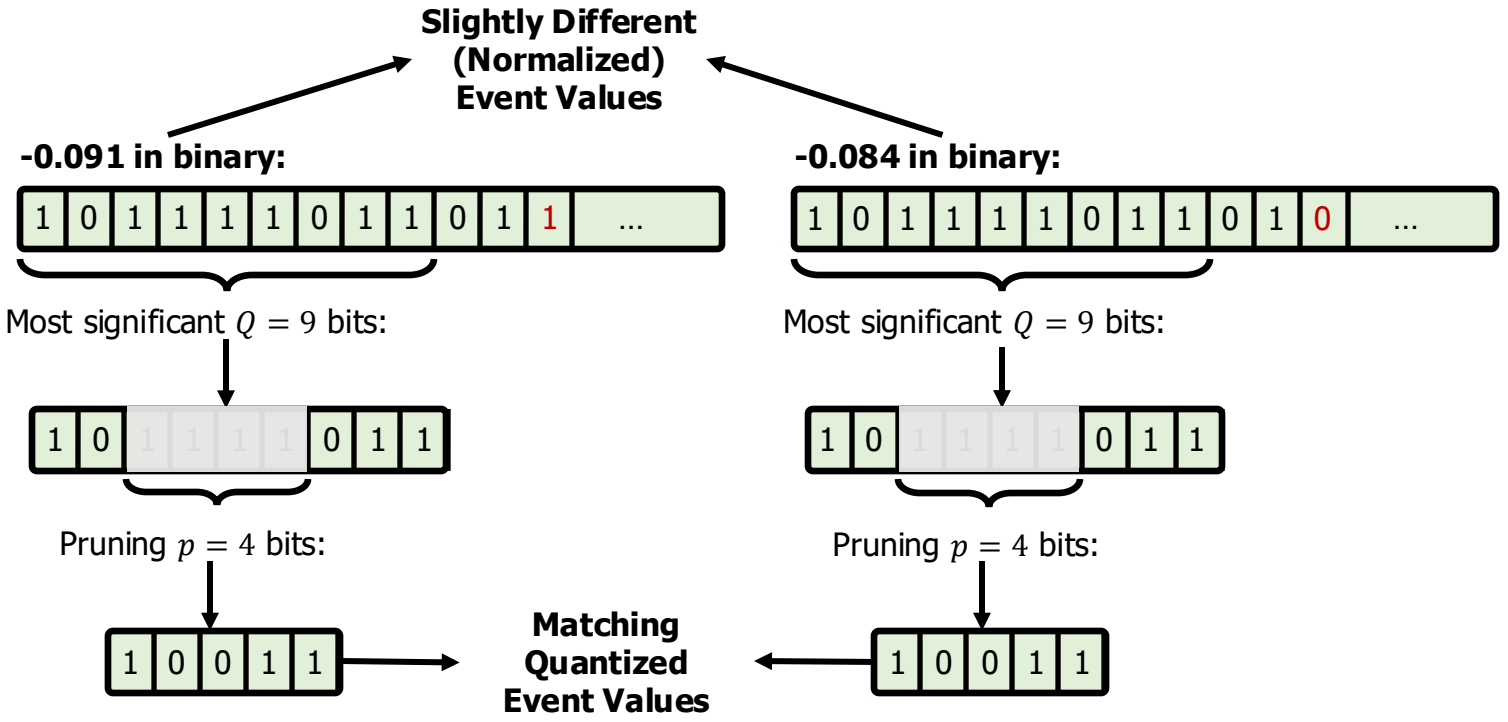
Can we directly match signals to each other?

- Consecutive events ➔ consecutive k-mers

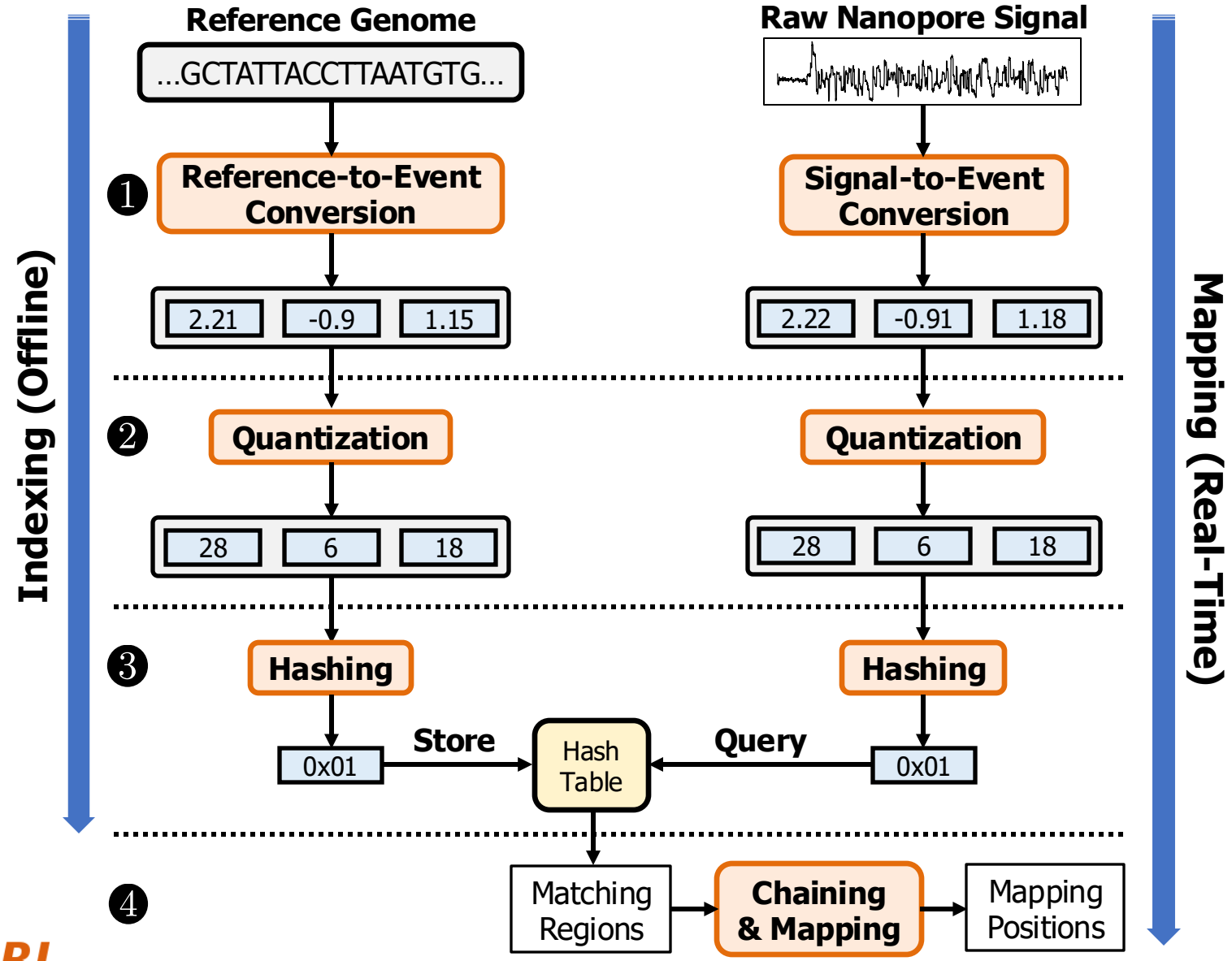
Details: Quantizing the Event Values

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

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



RawHash Overview



Breakdown Analysis of the RawHash Steps

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

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

Full Read Mapping Accuracy

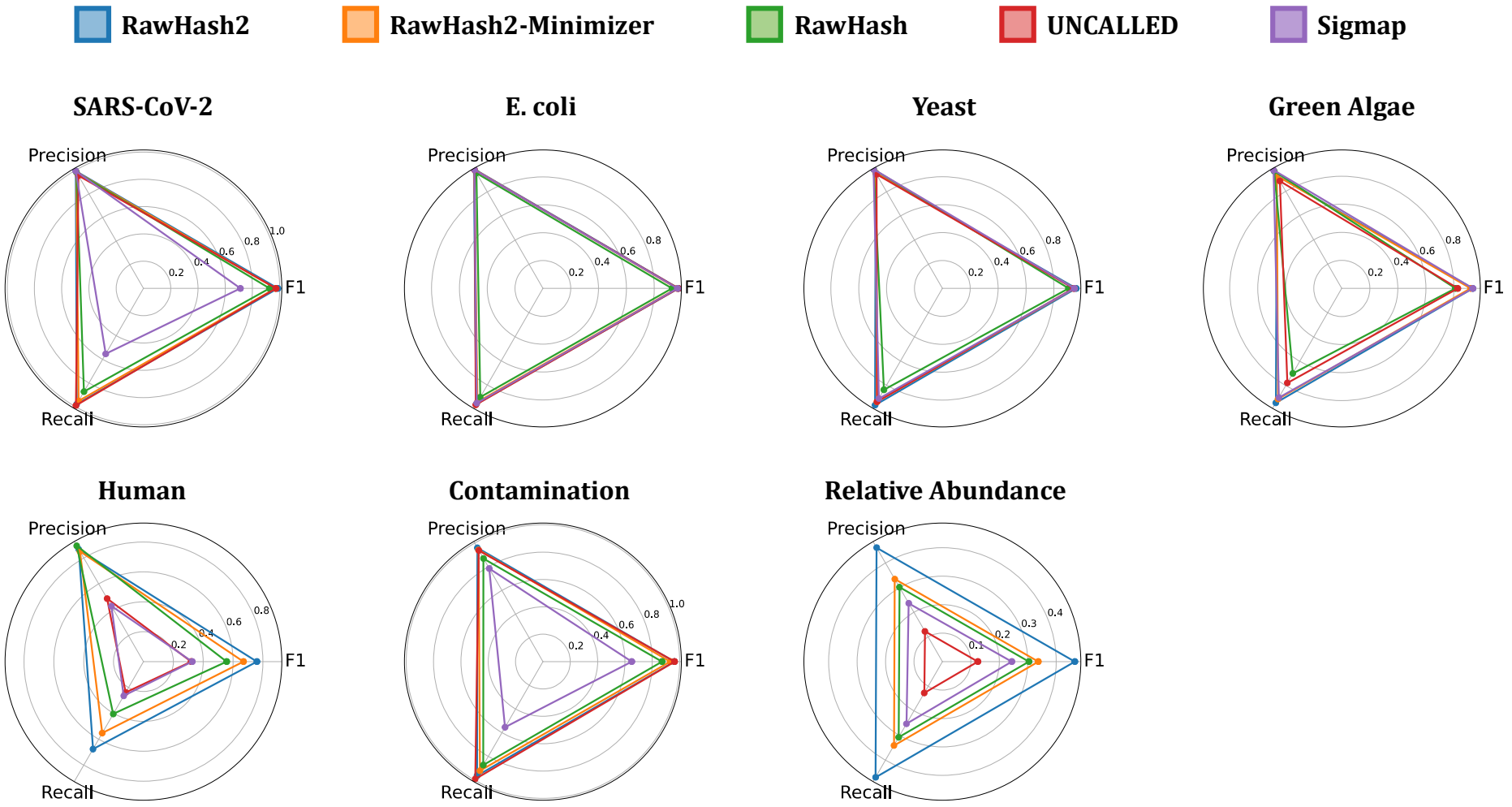
Dataset	Metric	RH2	RH2-Min.	RH	UNCALLED	Sigmap
SARS-CoV-2	F1	**0.9867**	0.9691	0.9252	0.9725	0.7112
	Precision	**0.9939**	0.9868	0.9832	0.9547	0.9929
	Recall	0.9796	0.9521	0.8736	**0.9910**	0.5540
E. coli	F1	**0.9748**	0.9631	0.9280	0.9731	0.9670
	Precision	**0.9904**	0.9865	0.9563	0.9817	0.9842
	Recall	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

Best results are **highlighted**.

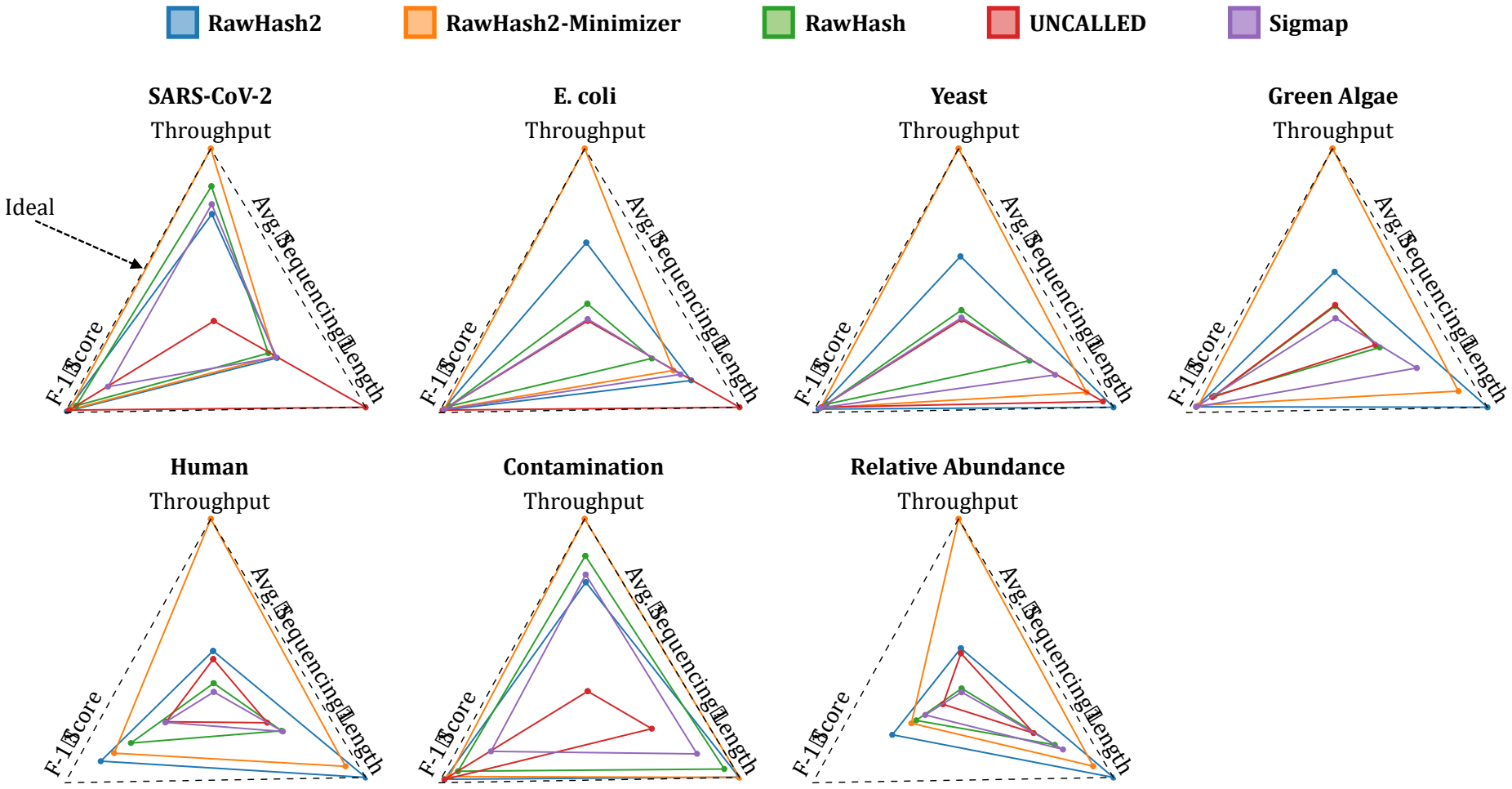
R10.4 Accuracy and Performance Results

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

Full Read Mapping Accuracy – Spider Plot



Different Metrics Combined – Spider Plot



Required Computation Resources in Indexing

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

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

Required Computation Resources in Mapping

Dataset	RH2	RH2-Min.	RH	UNCALLED	Sigmap
			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
			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

The mapping step of RawHash2 (and RawHash2-Minimizer)

is **significantly faster than all tools**

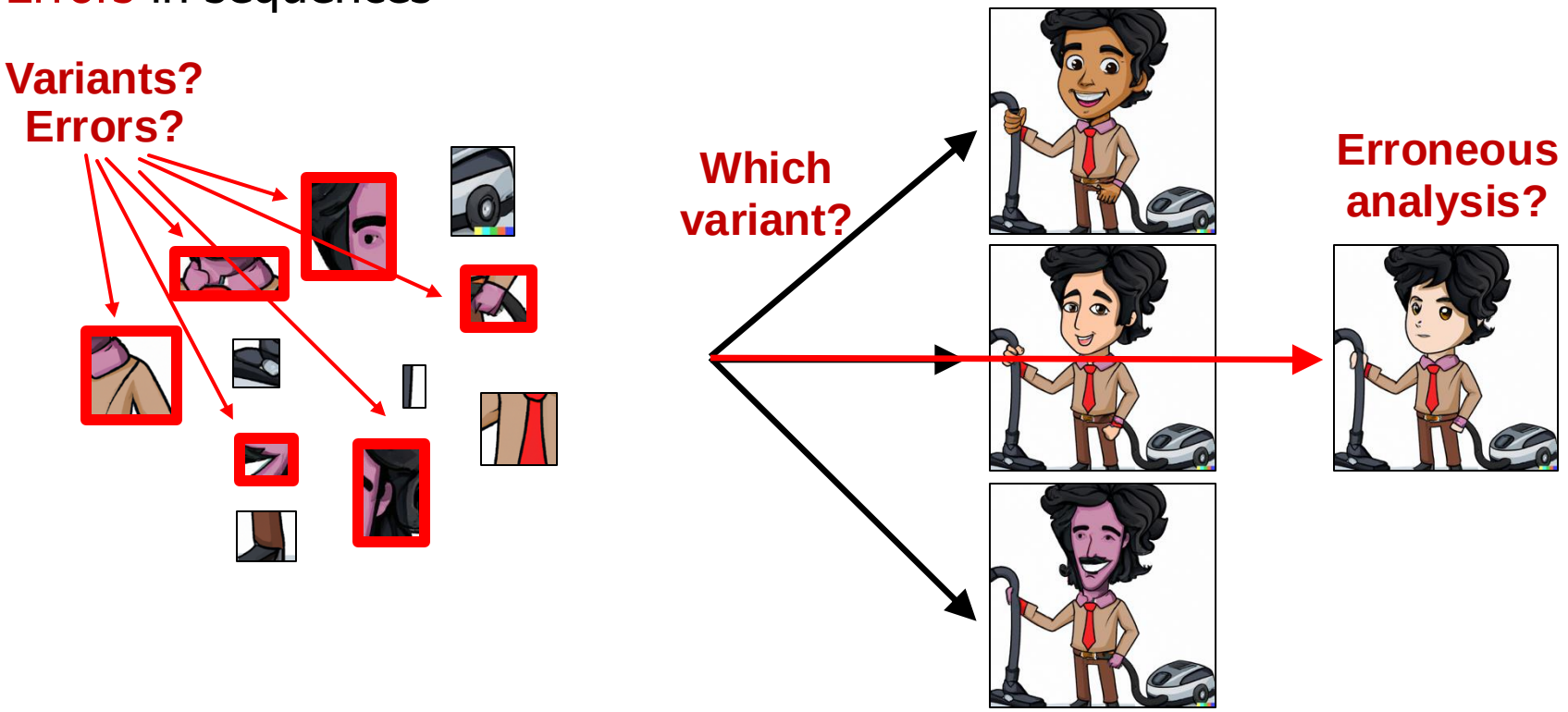
Required CPU Threads For the Entire Flow Cell

Dataset	RH2	RH2-Min.	RH	UNCALLED	Sigmap
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

RawHash2 (and RawHash2-Minimizer) requires the least amount of CPU threads to process the entire MinION flowcell

Why Graphs are Useful

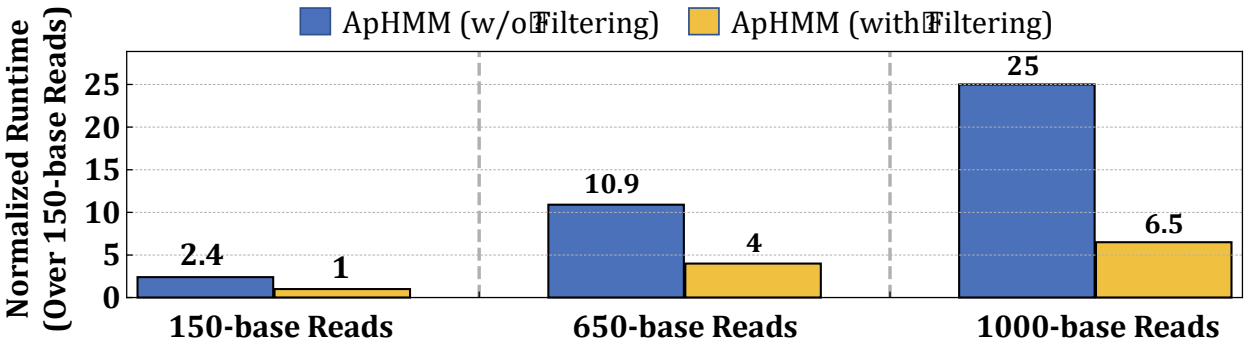
- Accurate comparison requires identifying changes (**insertions, deletions, substitutions**) between sequences due to
  - Variations between individuals and template sequences
  - Errors in sequences



- How to avoid unnecessary (and costly) comparisons?

Filtering – Performance Benefits

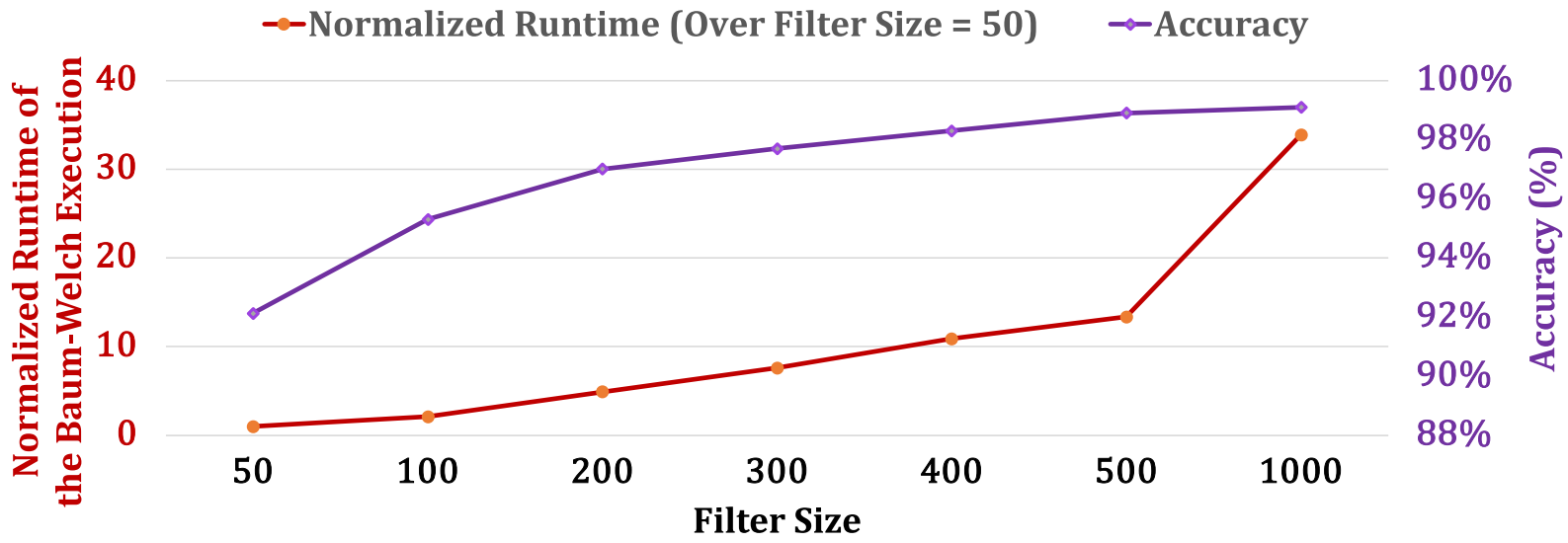
- Filtering heuristics aim to reduce unnecessary computations



Motivational Study: ~2.5x performance improvements with filtering

Filtering – Accurate but Costly Sorting

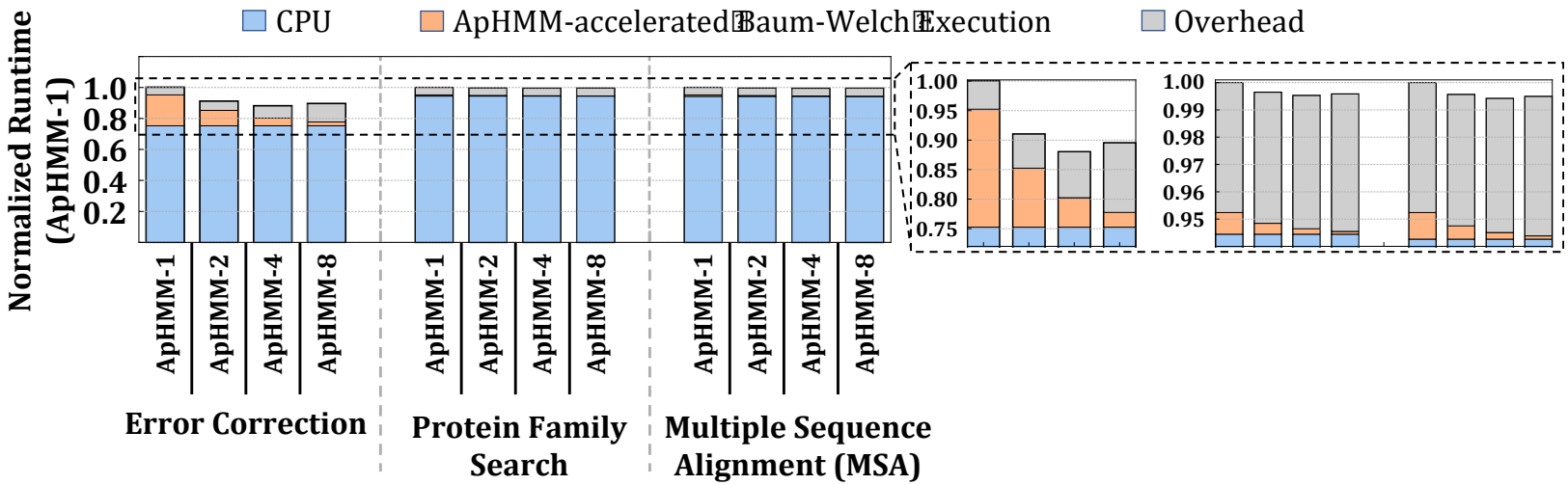
- Software-based filtering heuristics aim to reduce unnecessary computations
  - High-accuracy can be achieved with filtering with correct setting



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

Choosing the Right Amount of Cores

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



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

Speedup of Each Optimization

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

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

Broadcasting and partial compute together is only possible

with an efficient HW-SW co-design

Area and Power

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

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

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

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