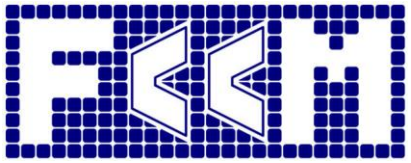
Hardware Accelerated Next Generation Sequencing Analysis

Arun Subramaniyan

Staff Bioinformatics Scientist, Illumina



Workshop on Secure and Adaptive Computing from Genomics to Healthcare
The 33rd Annual Symposium on Field-Programmable Custom Computing Machines
May 4th 2025

*All views are own and do not reflect those of employer

Sequencing data is increasing in volume and diversity

Illumina Genome Analyzer, 2005

1 Gbases per day

Illumina NovaSeq X, 2025

4 Tbases per day

4000x increase in sequencing machine throughput

Credits: Illumina, DataBase Center for Life Science (DBCLS), https://www.ecseq.com/support/ngs/do_you_have_two_colors_or_four_colors_in_Illumina

Sequencing data is increasing in volume and diversity

Illumina Genome Analyzer, 2005

1 Gbases per day

Illumina NovaSeq X, 2025

4 Tbases per day

illumina®

Read length: 100-350bp

Per base inaccuracy: 0.1%

Oxford NANOPORE Technologies

Read length: 1kb-1Mbp

Per base inaccuracy: 1-15%

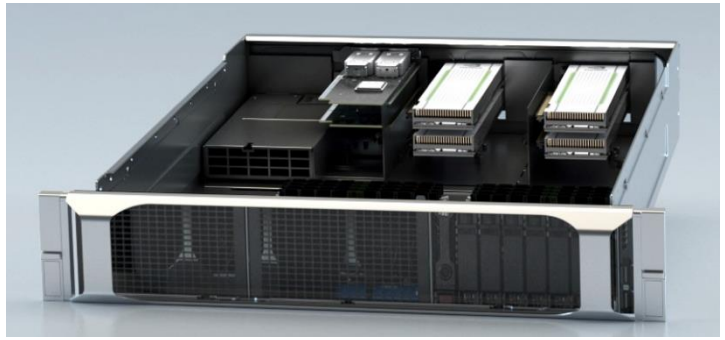
4000x increase in sequencing machine throughput

1000x increase in sequencing fragment length

10 - 100x increase in sequencing error rate

Credits: Illumina, DataBase Center for Life Science (DBCLS), https://www.ecseq.com/support/ngs/do_you_have_two_colors_or_four_colors_in_Illumina

GPUs and custom hardware are being used for sequence analysis



NVIDIA Parabricks

Microsoft

illumina

DRAGEN Bio-IT Platform

Microsoft Genomics and Stanford University

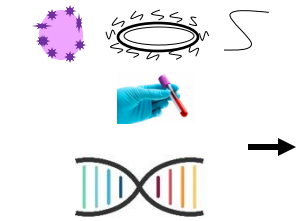
intel

GPU

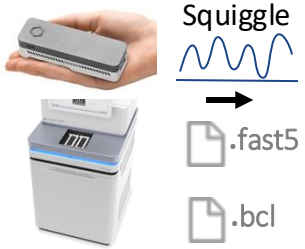
FPGA

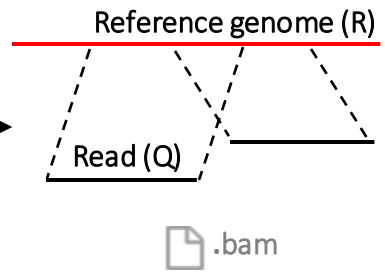
Modern Sequence Analysis Pipelines

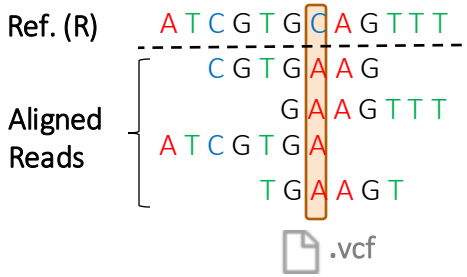
(a) Reference-Guided Assembly



Genome sizes:
~Mega -- Gigabases



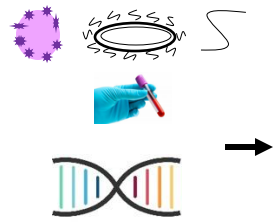




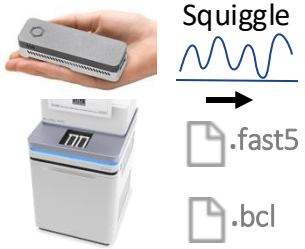
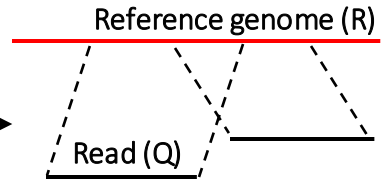
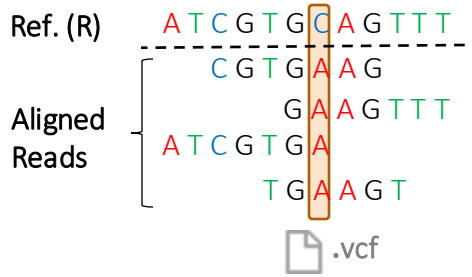
	Sequencing	Basecalling	Read Alignment	Variant Calling
Software tools:				
Short Reads		Bustard	BWA-MEM2 (~30-40 %)	GATK Haplotype Caller, Platypus (~40%)
Long Reads		Bonito	Minimap2	Clair

Modern Sequence Analysis Pipelines

(a) Reference-Guided Assembly

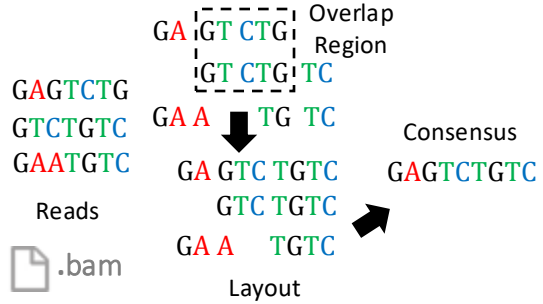
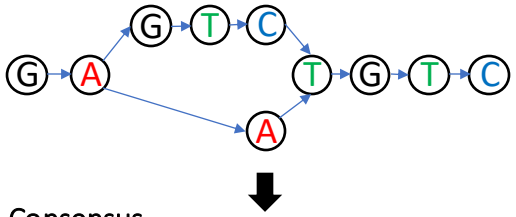


Genome sizes:
~Mega -- Gigabases

	Sequencing	Basecalling	Read Alignment	Variant Calling
		CGTGAAG GAAGTTT Reads (~millions-billions) .fastq	 .bam	
Software tools:	**Short Reads**	Bustard	BWA-MEM2 (~30-40 %)	GATK Haplotype Caller, Platypus (~40%)
	Long Reads	Bonito	Minimap2	Clair

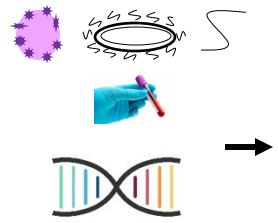
(b) De-Novo Assembly

Long Reads

Overlap, Layout, Consensus	Error Correction and Polishing
	
Flye, Minimap2	Racon, Nanopolish

Modern Sequence Analysis Pipelines

(a) Reference-Guided Assembly



Genome sizes:
~Mega -- Gigabases

Sequencing

Squiggle

.fast5

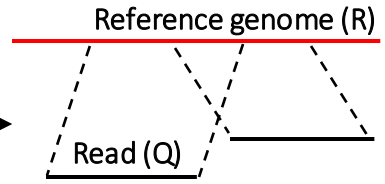
.bcl

Basecalling

CGTGAAG
GAAGTTT
Reads
(~millions-billions)

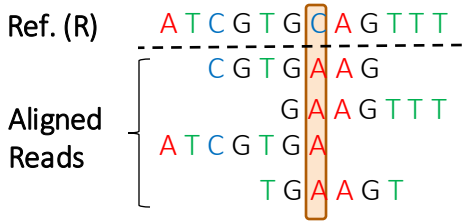
.fastq

Read Alignment



.bam

Variant Calling



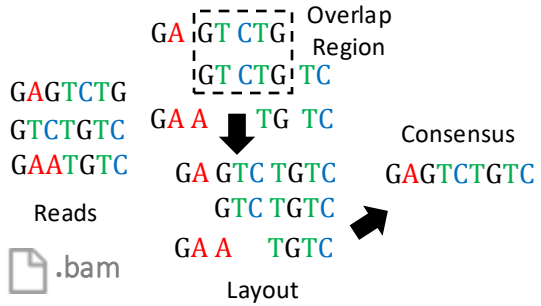
.vcf

Software tools:			
Short Reads	Bustard	BWA-MEM2 (~30-40 %)	GATK Haplotype Caller, Platypus (~40%)
Long Reads	Bonito	Minimap2	Clair

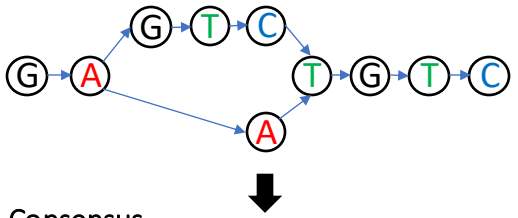
(b) De-Novo Assembly

Long Reads

Overlap, Layout, Consensus



Error Correction and Polishing

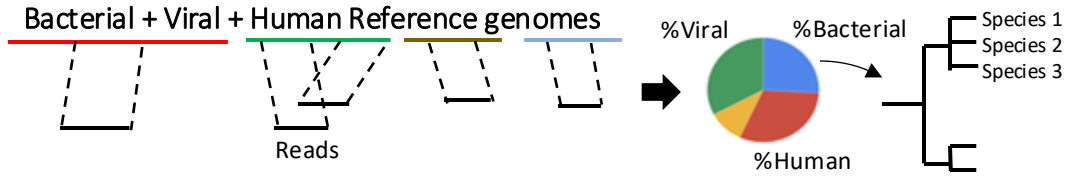


Flye, Minimap2 | Racon, Nanopolish

(c) Metagenomics Classification

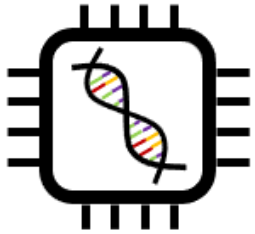
Long Reads

Abundance Estimation



Read Alignment .bam Phylogenetic Tree

Centrifuge, Minimap2



GenomicsBench

https://genomicsbench.eecs.umich.edu

- Computationally intensive kernels drawn from well maintained software tools
- Covers the major steps of modern sequence analysis pipelines
- Includes both short and long read analysis algorithms
- Small/large input datasets

Dynamic programming algorithms are common

Benchmark	Input Datatype	Applications	Chosen Tool	% Time Spent in Tool (single-thread)	Parallelism Motif
fmi	Short reads	Read Alignment Metagenomics Classification	BWA-MEM2	38%	Tree Traversal
bsw	Short reads	Read Alignment De-Novo Assembly	BWA-MEM2	31%	Dynamic Programming
dbg	Short reads	Variant Calling De-Novo Assembly	Platypus	65%	Graph Construction Hash Table
phmm	Short reads	Variant Calling Error Correction	GATK Haplotype Caller	70%	Dynamic Programming
chain	Long reads	De-Novo Assembly Read Alignment	Minimap2	47.4 %	Dynamic Programming (1D)
spoa	Long reads	Error Correction	Racon	75 %	Dynamic Programming Graph Construction
abea	Long reads	Basecalling Variant Calling	Nanopolish	71.4%	Dynamic Programming
grm	NA	Population Genomics	PLINK2	92.8 %	Dense Matrix Multiplication
nn-base	Long reads	Basecalling	Bonito	95 %	FP Matrix Multiplication
nn-variant	Long reads	Variant Calling	Clair	57.2 %	FP Matrix Multiplication

Subramaniyan, Arun, et al. "GenomicsBench: A Benchmark Suite for Genomics." *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2021.

Dynamic programming algorithms are common

Benchmark	Input Datatype	Applications	Chosen Tool	% Time Spent in Tool (single-thread)	Parallelism Motif
fmi	Short reads	Read Alignment Metagenomics Classification	BWA-MEM2	38%	Tree Traversal
bsw	Short reads	Read Alignment De-Novo Assembly	BWA-MEM2	31%	Dynamic Programming
dbg	Short reads	Variant Calling De-Novo Assembly	Platypus	65%	Graph Construction Hash Table
phmm	Short reads	Variant Calling Error Correction	GATK Haplotype Caller	70%	Dynamic Programming
chain	Long reads	De-Novo Assembly Read Alignment	Minimap2	47.4 %	Dynamic Programming (1D)
spoa	Long reads	Error Correction	Racon	75 %	Dynamic Programming Graph Construction
abea	Long reads	Basecalling Variant Calling	Nanopolish	71.4%	Dynamic Programming
grm	NA	Population Genomics	PLINK2	92.8 %	Dense Matrix Multiplication
nn-base	Long reads	Basecalling	Bonito	95 %	FP Matrix Multiplication
nn-variant	Long reads	Variant Calling	Clair	57.2 %	FP Matrix Multiplication

Subramaniyan, Arun, et al. "GenomicsBench: A Benchmark Suite for Genomics." *2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2021.

Dynamic programming algorithms are common

- 5 dynamic programming kernels in GenomicsBench
- Contribute 30-75% execution time in their respective software tools

Dynamic programming algorithms are common

- 5 dynamic programming kernels in GenomicsBench
- Contribute 30-75% execution time in their respective software tools
  - Chaining
  - Banded Smith-Waterman
  - Pairwise Hidden Markov Model
  - Partial Order Alignment
  - Adaptive Banded Event Alignment

Dynamic programming algorithms are common

- 5 dynamic programming kernels in GenomicsBench
- Contribute 30-75% execution time in their respective software tools
  - Chaining
  - Banded Smith-Waterman
  - Pairwise Hidden Markov Model
  - Partial Order Alignment
  - Adaptive Banded Event Alignment

Diverse compute requirements

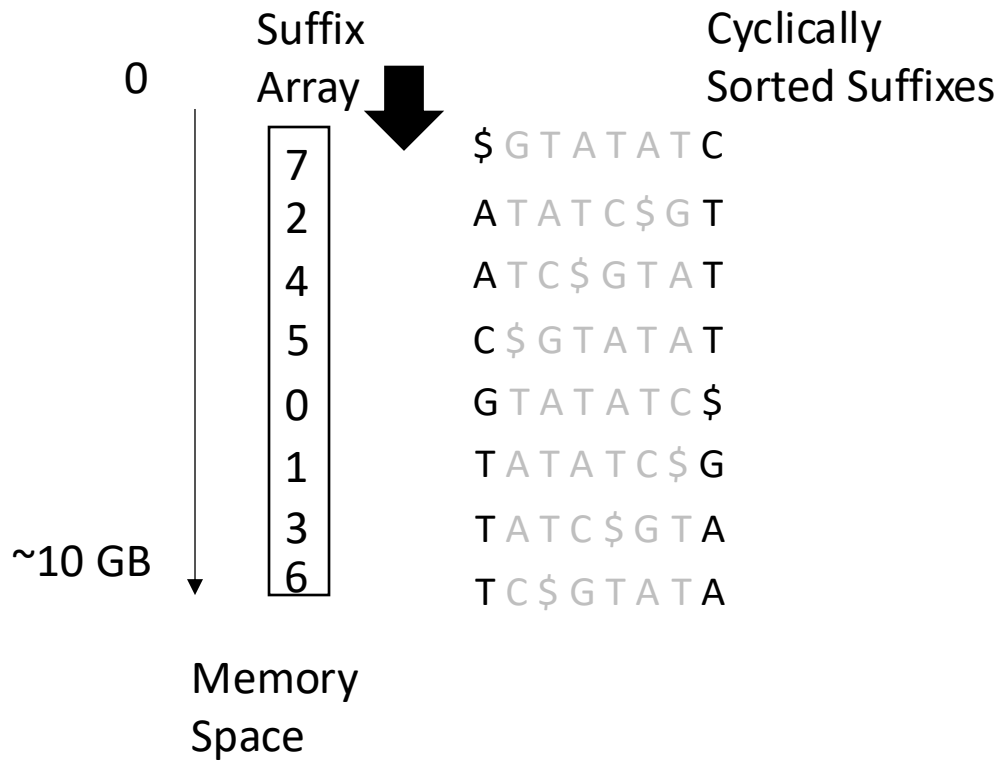
Data dependencies: 1-D, 2-D

Inputs: Sequence, Graph

Computation type: Integer, Floating point

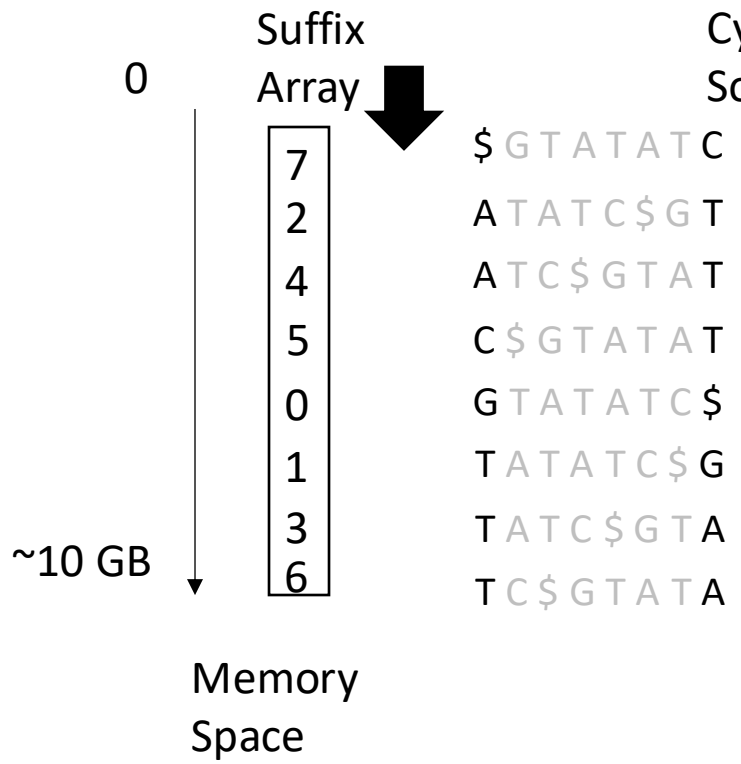
Memory-Intensive Index Lookup Benchmarks: e.g., FM-index search

Reference : G T A T A T C $



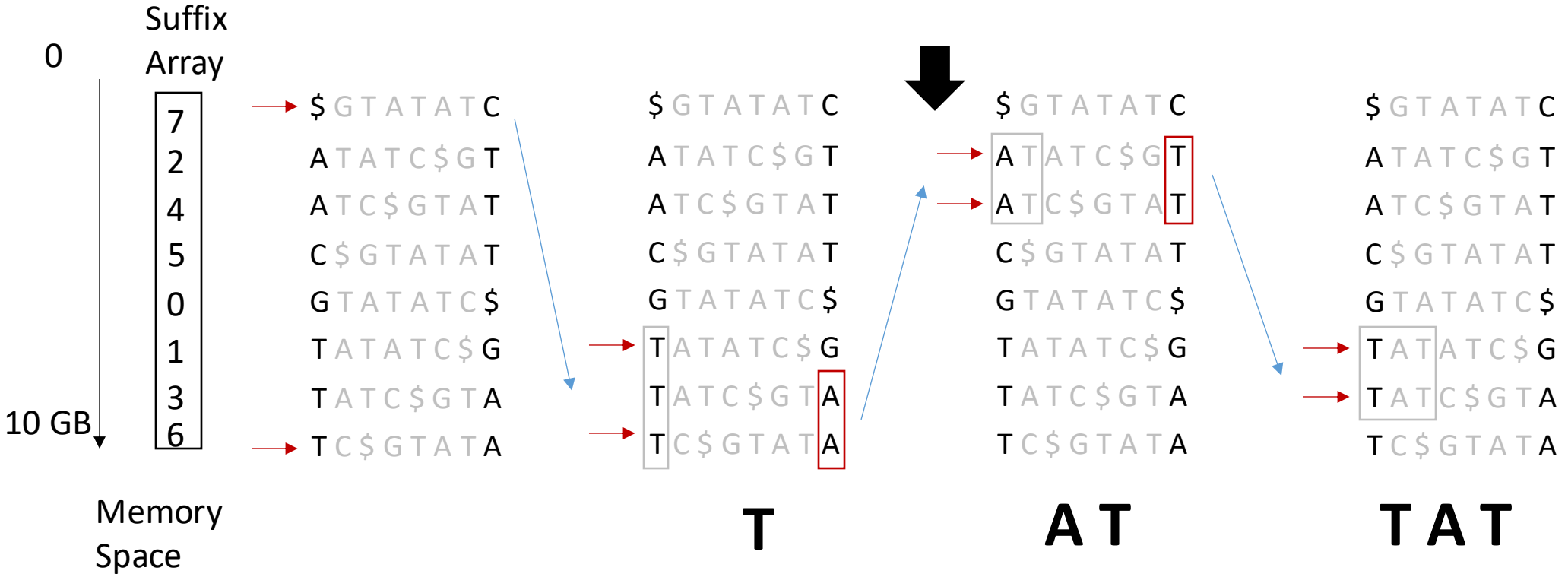
Memory-Intensive Index Lookup Benchmarks: e.g., FM-index search

Reference : G T A T A T C $

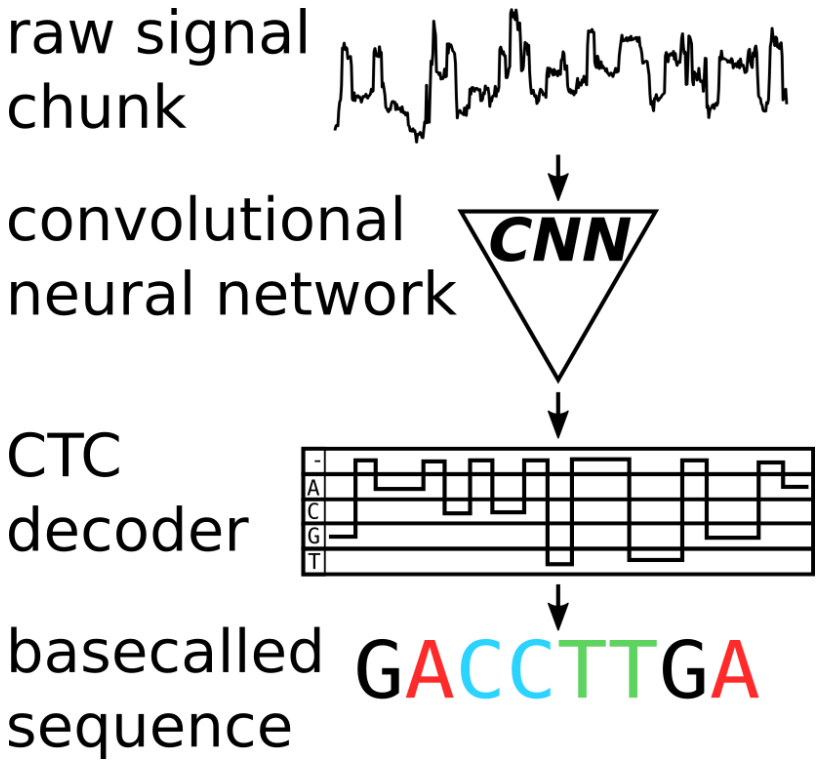


FM-index(T) – compressed representation of cyclically sorted suffixes of T

Irregular memory accesses in FM-index search

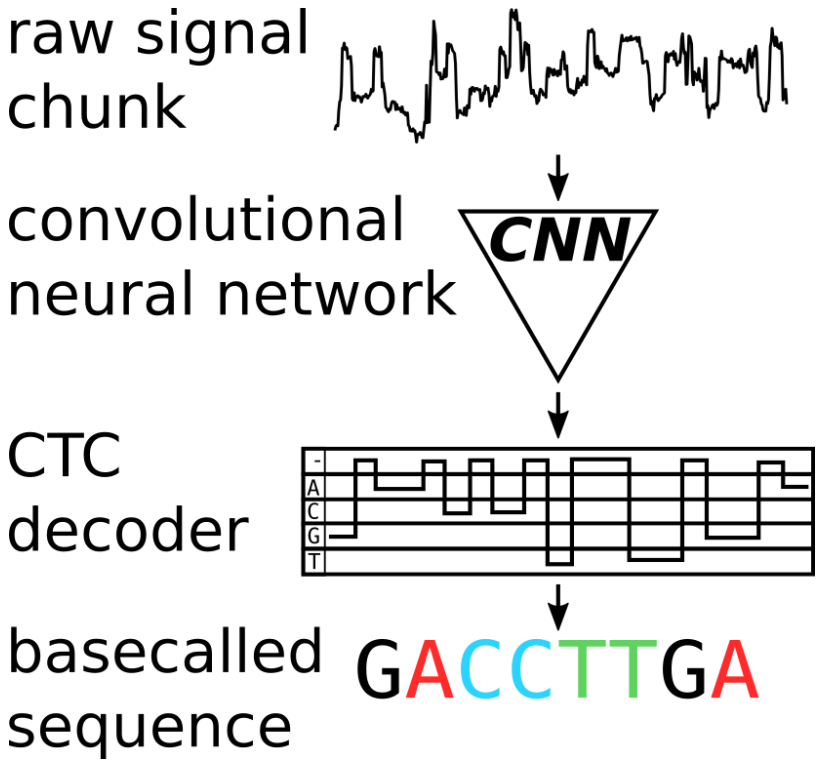


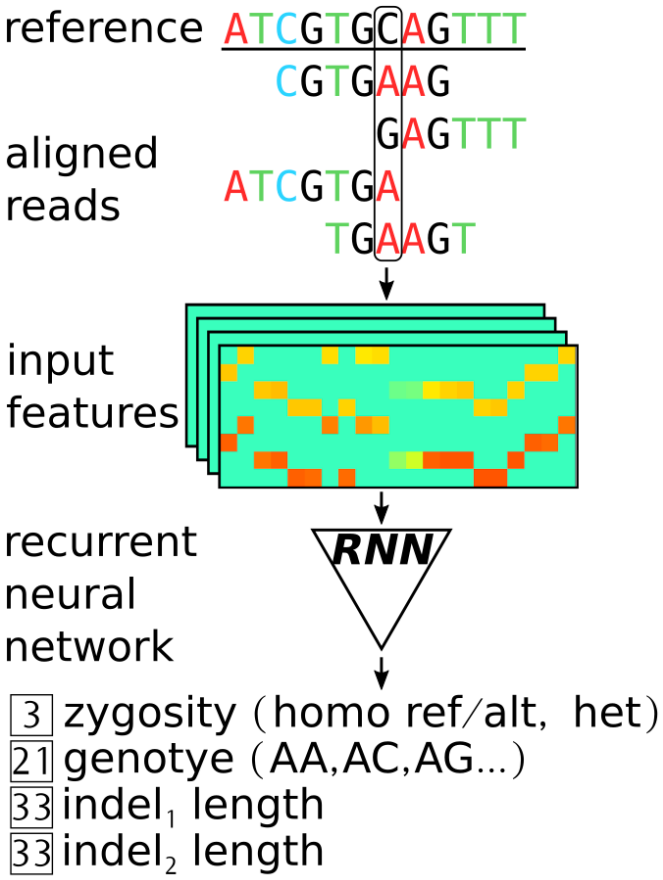
Machine learning algorithms are also gaining traction



nn-base (Basecalling)

Machine learning algorithms are also gaining traction





nn-base (Basecalling)

nn-variant (Variant calling)

Abundant data parallelism in sequence analysis applications

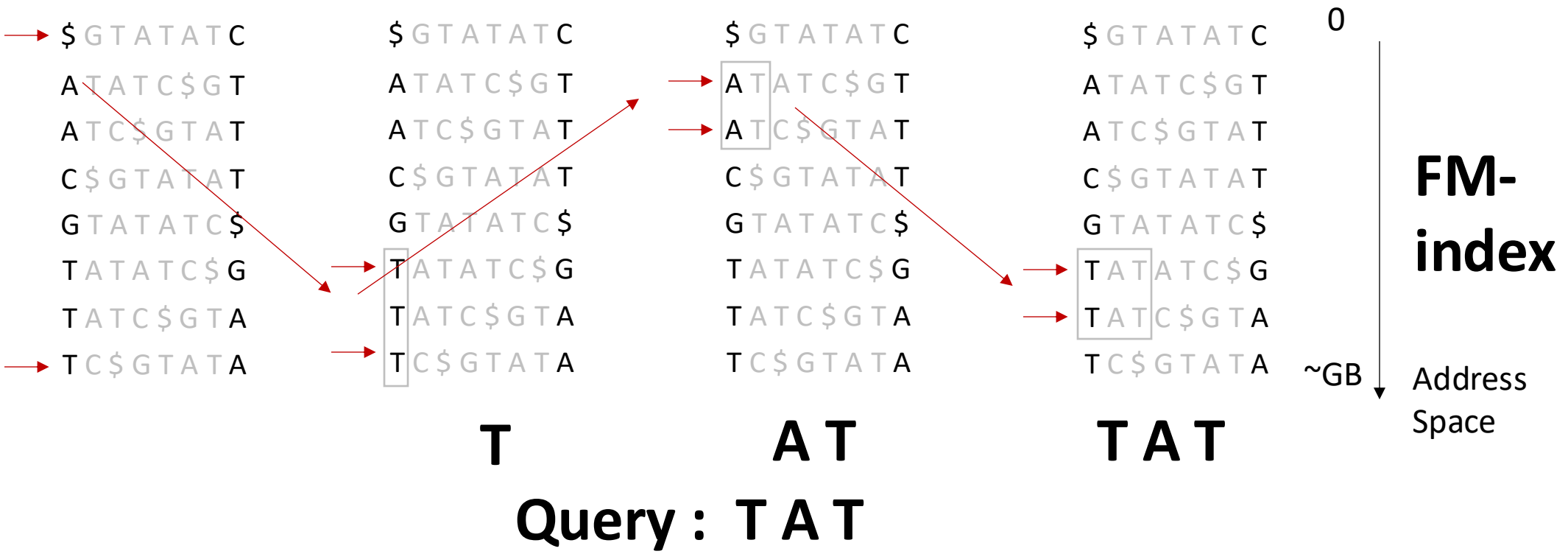
Kernel	Parallelism granularity	Source of parallelism
FM-index search (fmi)	Read	# OCC Table Lookups
Banded Smith-Waterman (bsw)	Seed	# Cell Updates
Pairwise Hidden Markov Model (phmm)	Genome Region	# Cell Updates
Chaining (chain)	Read	# Input Anchors
Partial Order Alignment (poa)	Read Chunk	# Cell Updates

Abundant data parallelism available

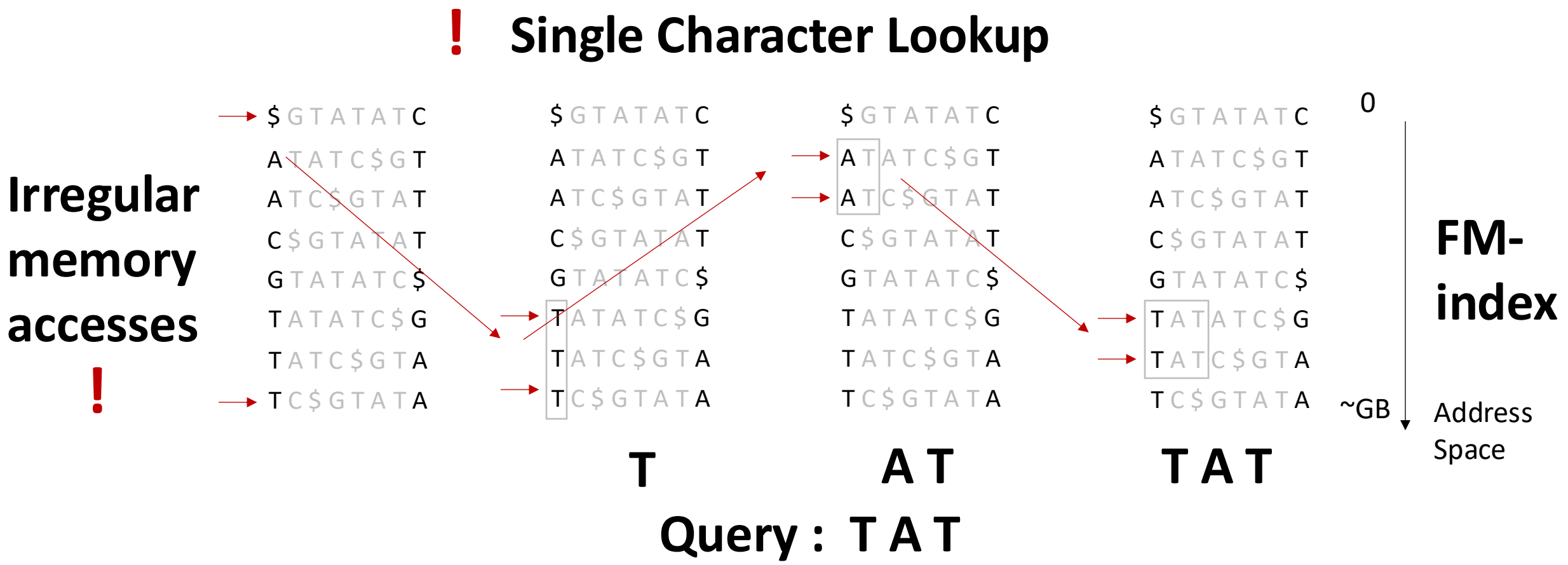
~billions of ***independent*** reads
~millions of ***independent*** genome regions

Case Study – Accelerating FM-index Search

FMD-index seeding trades off memory bandwidth for space



FMD-index seeding trades off memory bandwidth for space

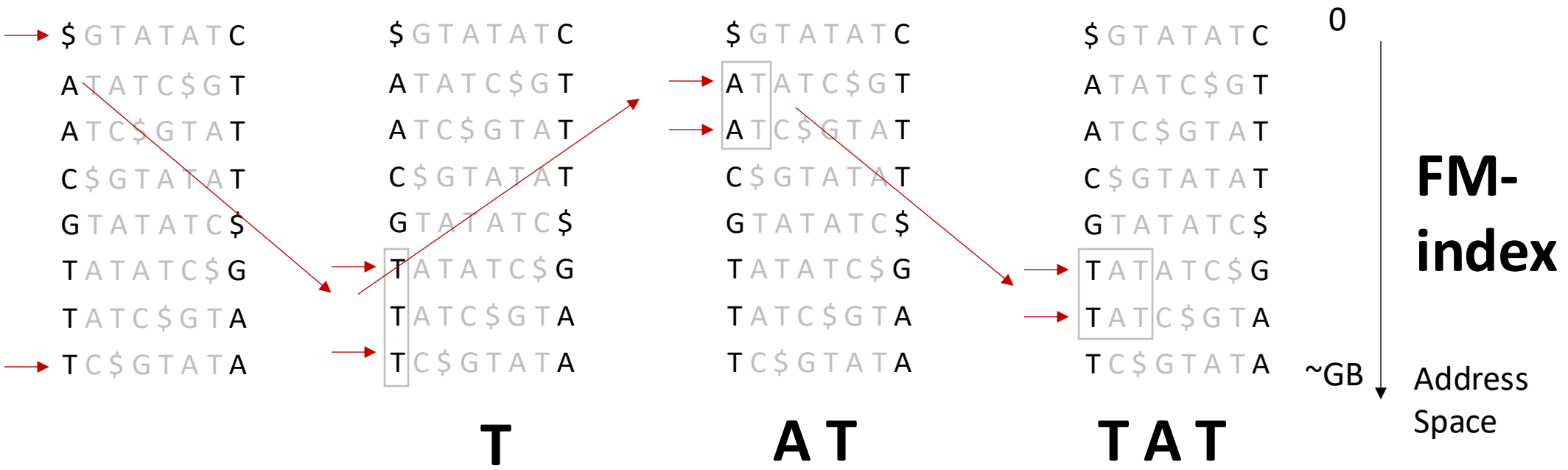


FMD-index seeding trades off memory bandwidth for space

BWA-MEM : 102 kB / read
BWA-MEM2 : 61 kB / read

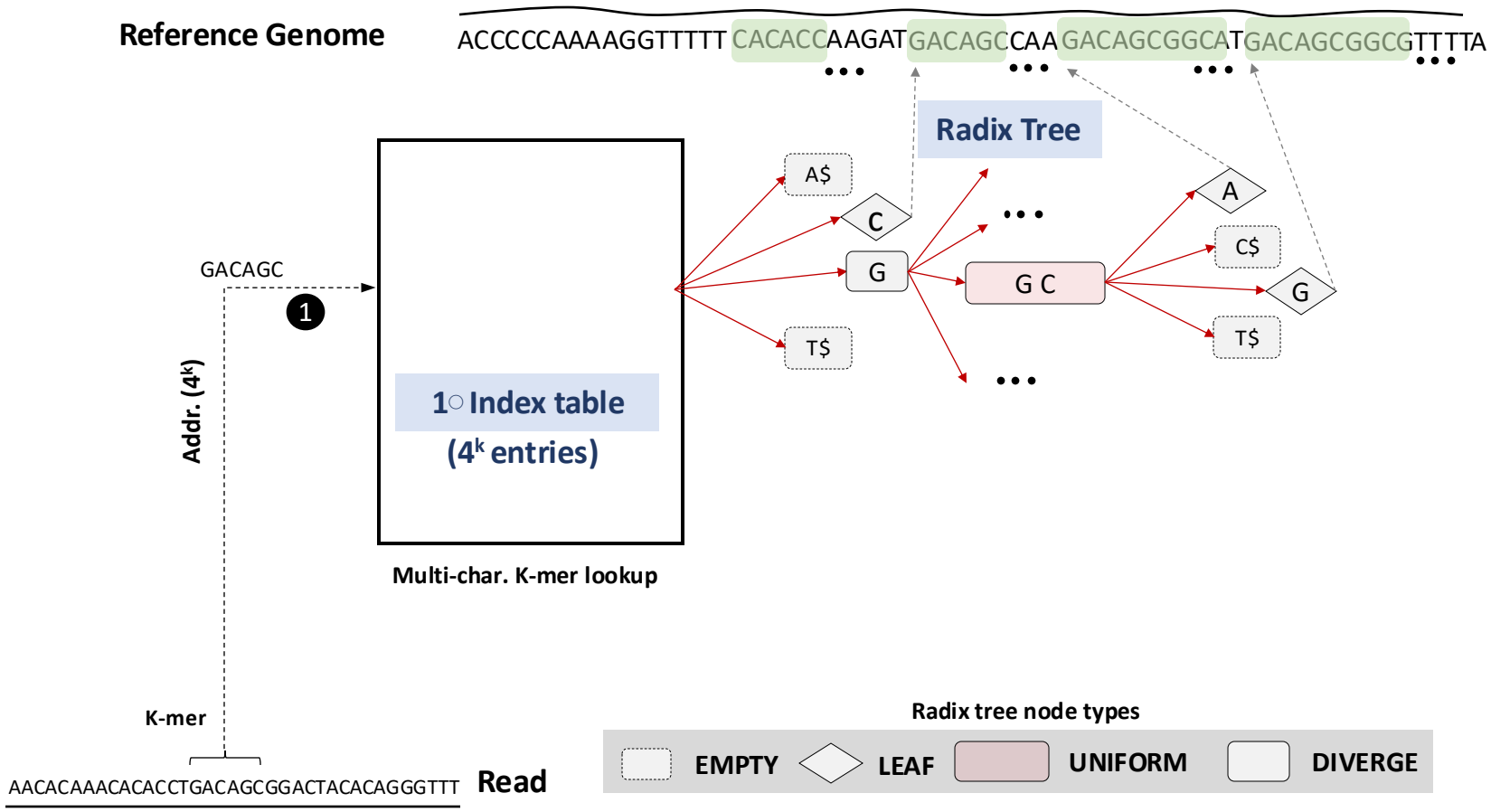
! Single Character Lookup

Irregular memory accesses !



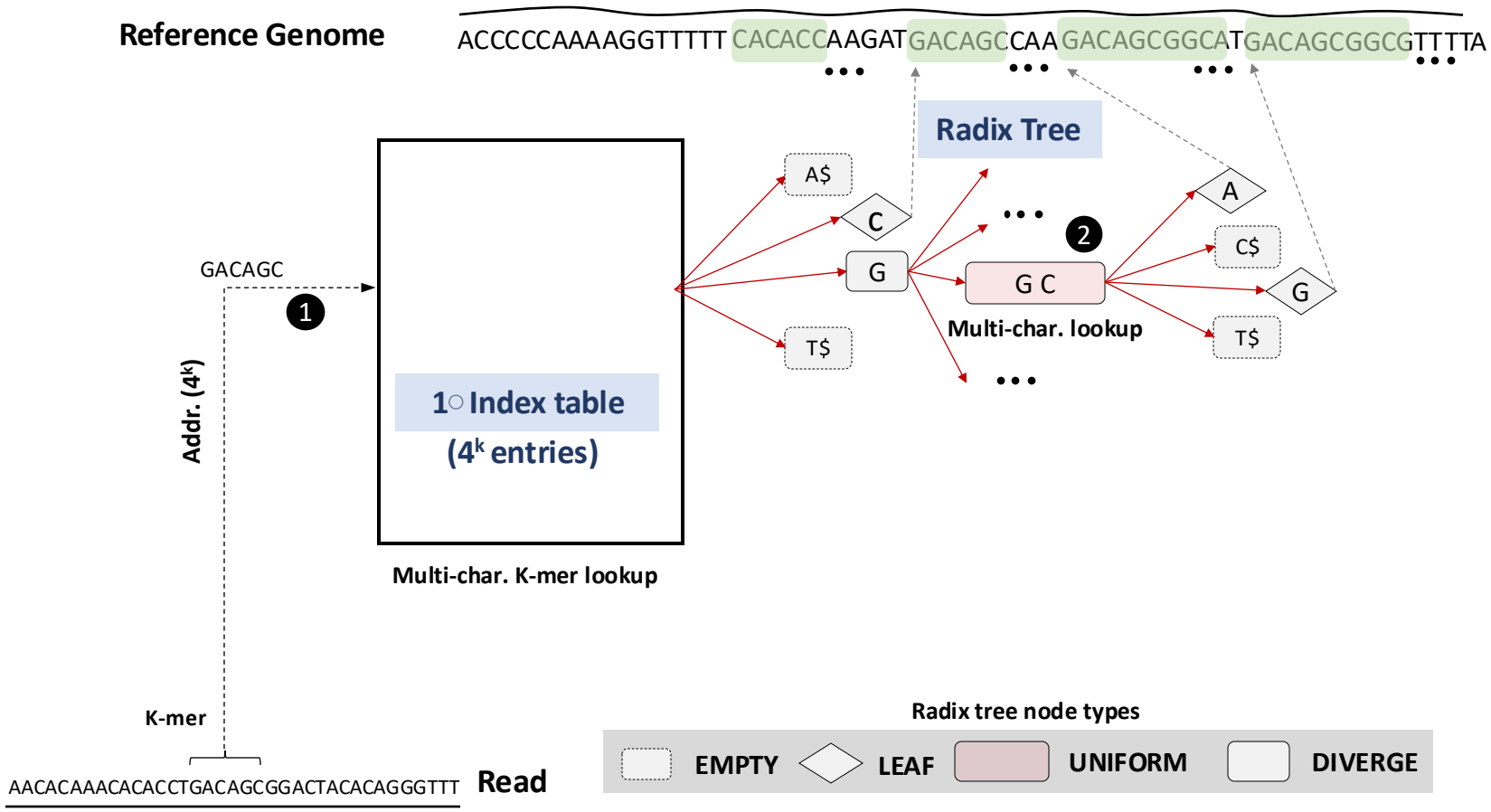
Multi-character lookup trading space for reducing memory bandwidth

ERT: Multi-level index table with a space-optimized radix tree



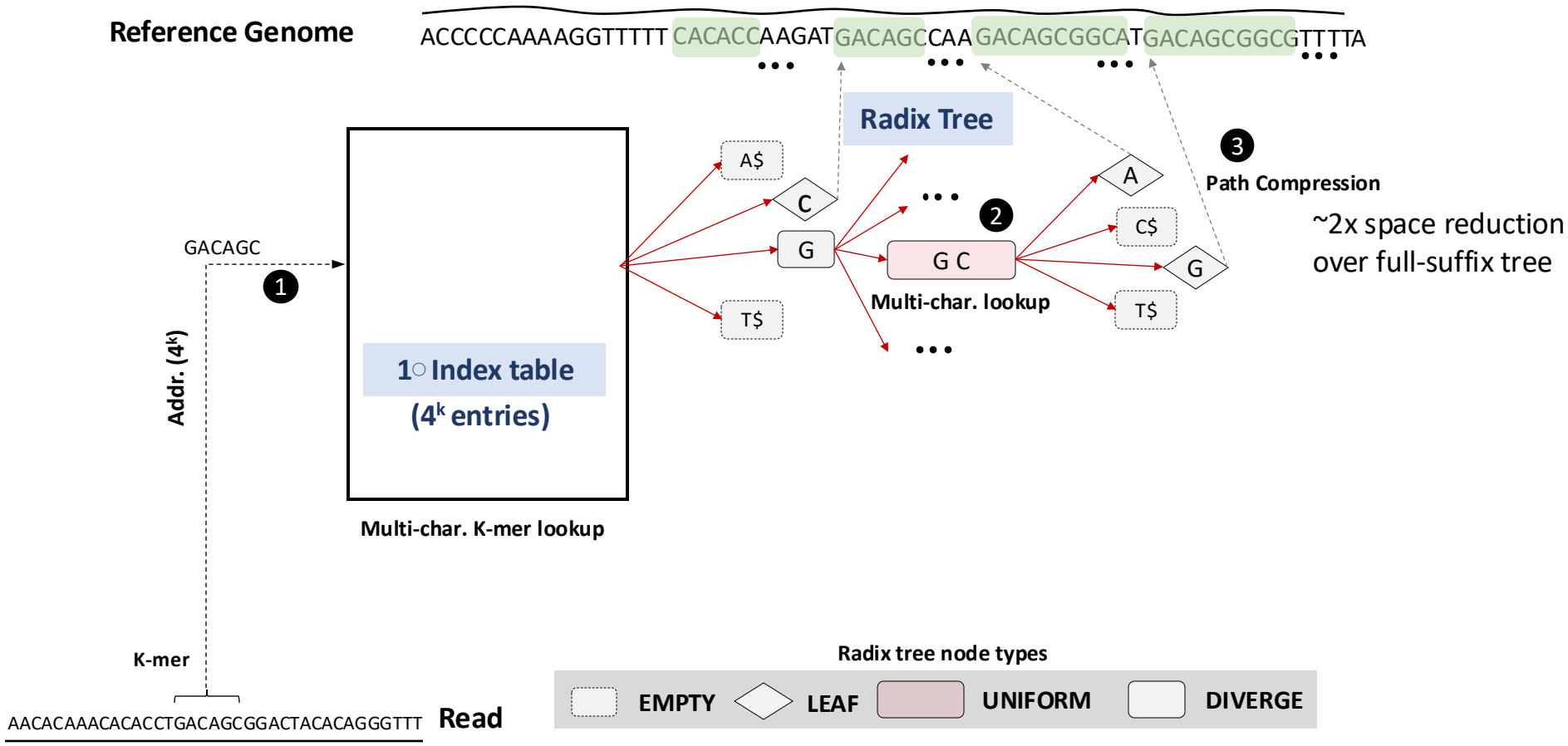
Multi-character lookup trading space for reducing memory bandwidth

ERT: Multi-level index table with a space-optimized radix tree



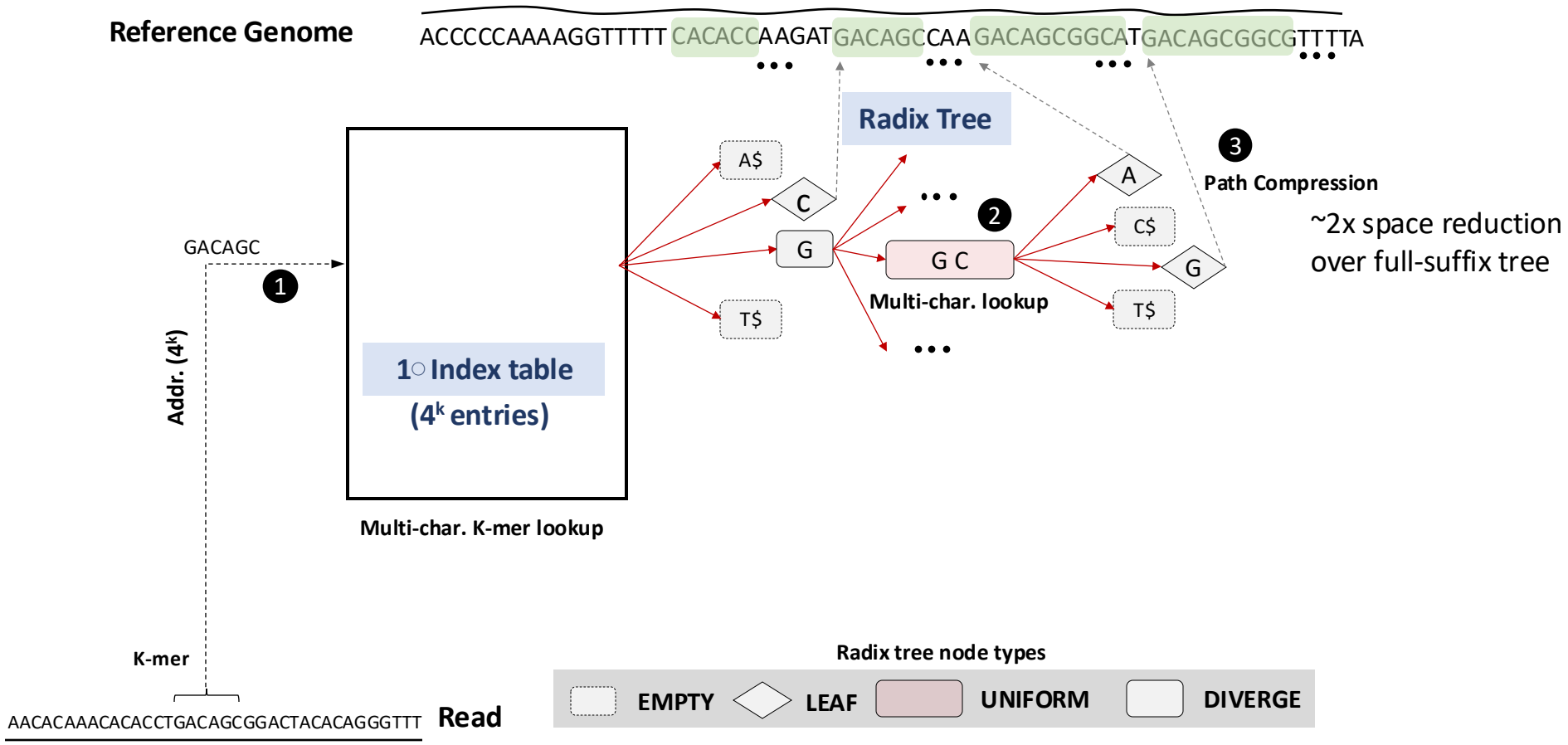
Multi-character lookup trading space for reducing memory bandwidth

ERT: Multi-level index table with a space-optimized radix tree



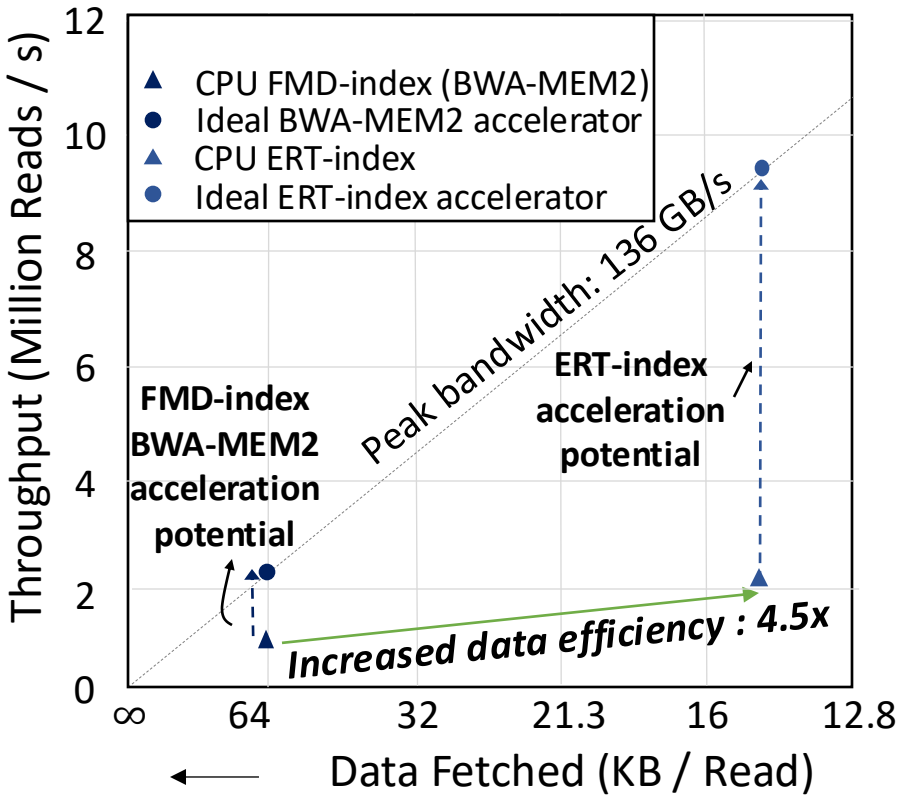
Multi-character lookup trading space for reducing memory bandwidth

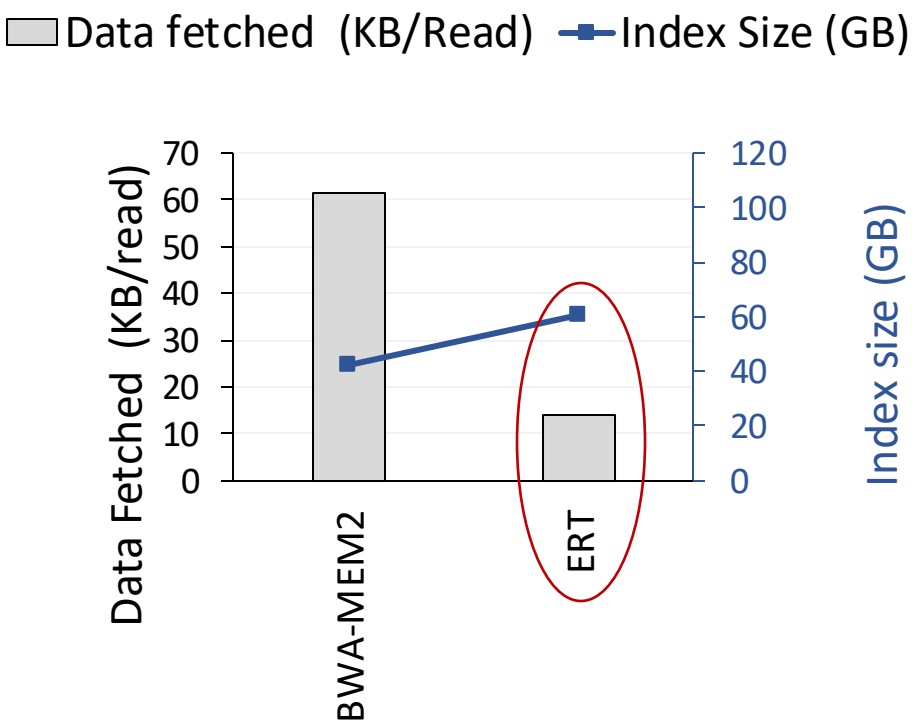
ERT: Multi-level index table with a space-optimized radix tree



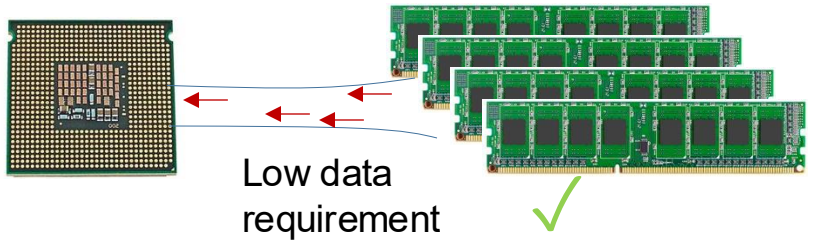
Also, present is a 2nd level index table for 15-mers having >= 256 occurrences

Instead trade-off memory capacity for reducing memory bandwidth





Enumerated Radix Tree (ERT)



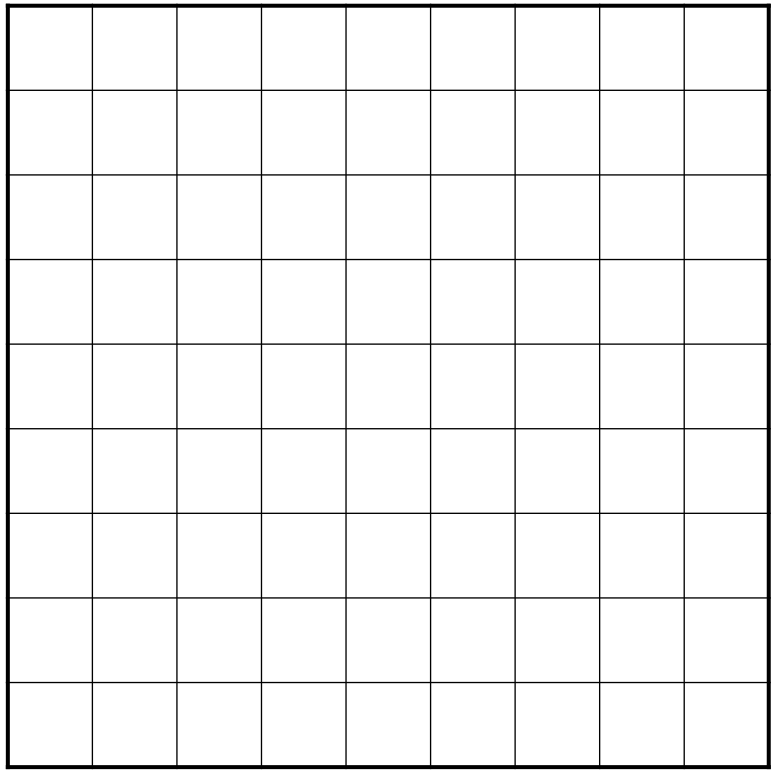
ERT ~60 GB human

https://github.com/bwa-mem2/bwa-mem2/tree/ert

"Accelerated seeding for genome sequence alignment with enumerated radix trees." In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, pp. 388-401. IEEE, 2021.

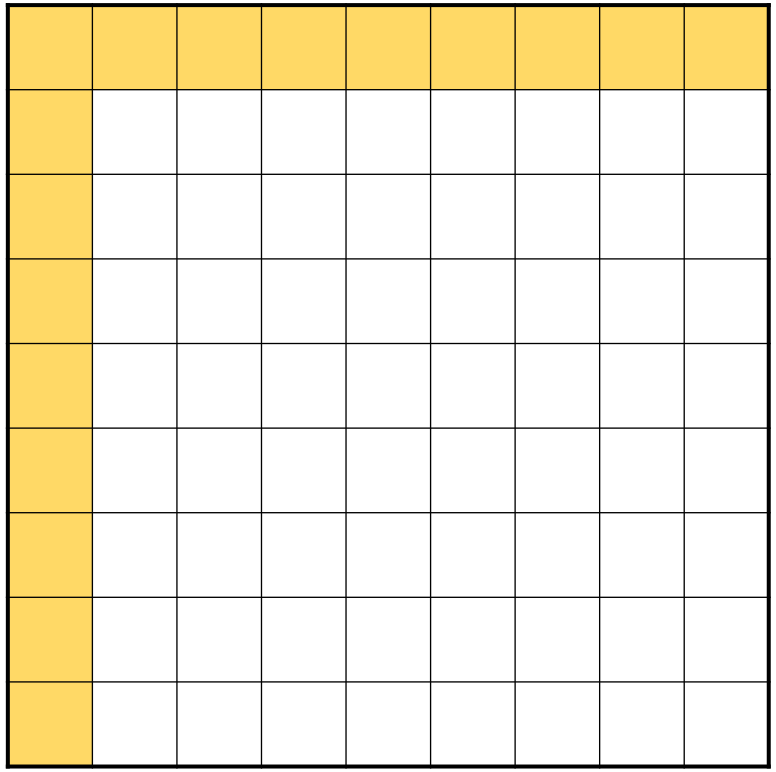
Dynamic programming for sequence similarity estimation

- Task: calculate the scores in the entire table.



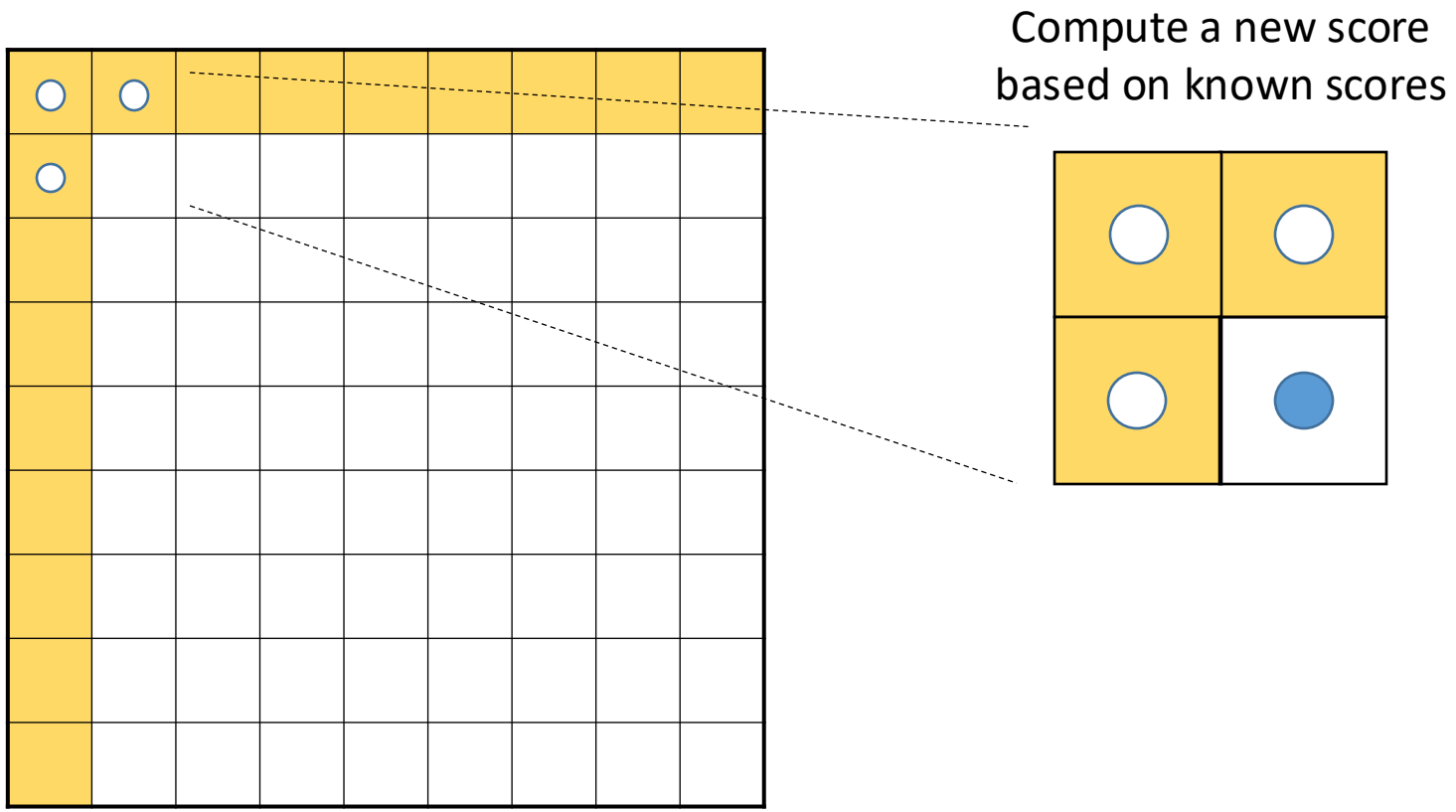
Dynamic programming for sequence similarity estimation

- Task: calculate the scores in the entire table.
- Initialization: the scores in first row and first column.



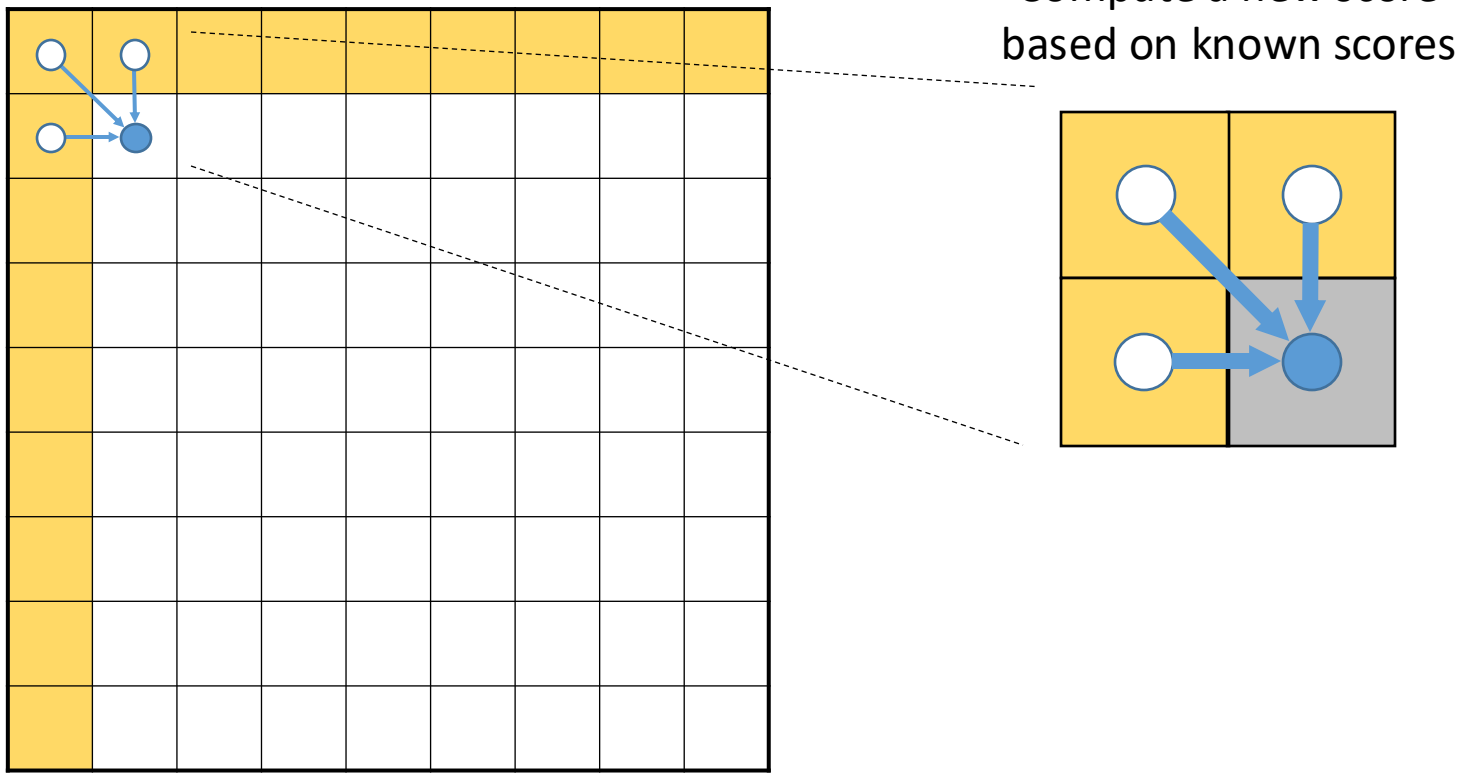
Dynamic programming for sequence similarity estimation

- Task: calculate the scores in the entire table.
- Initialization: the scores in first row and first column.
- Objective Function: calculate the score of a cell based on its upper, left and diagonal neighbors.



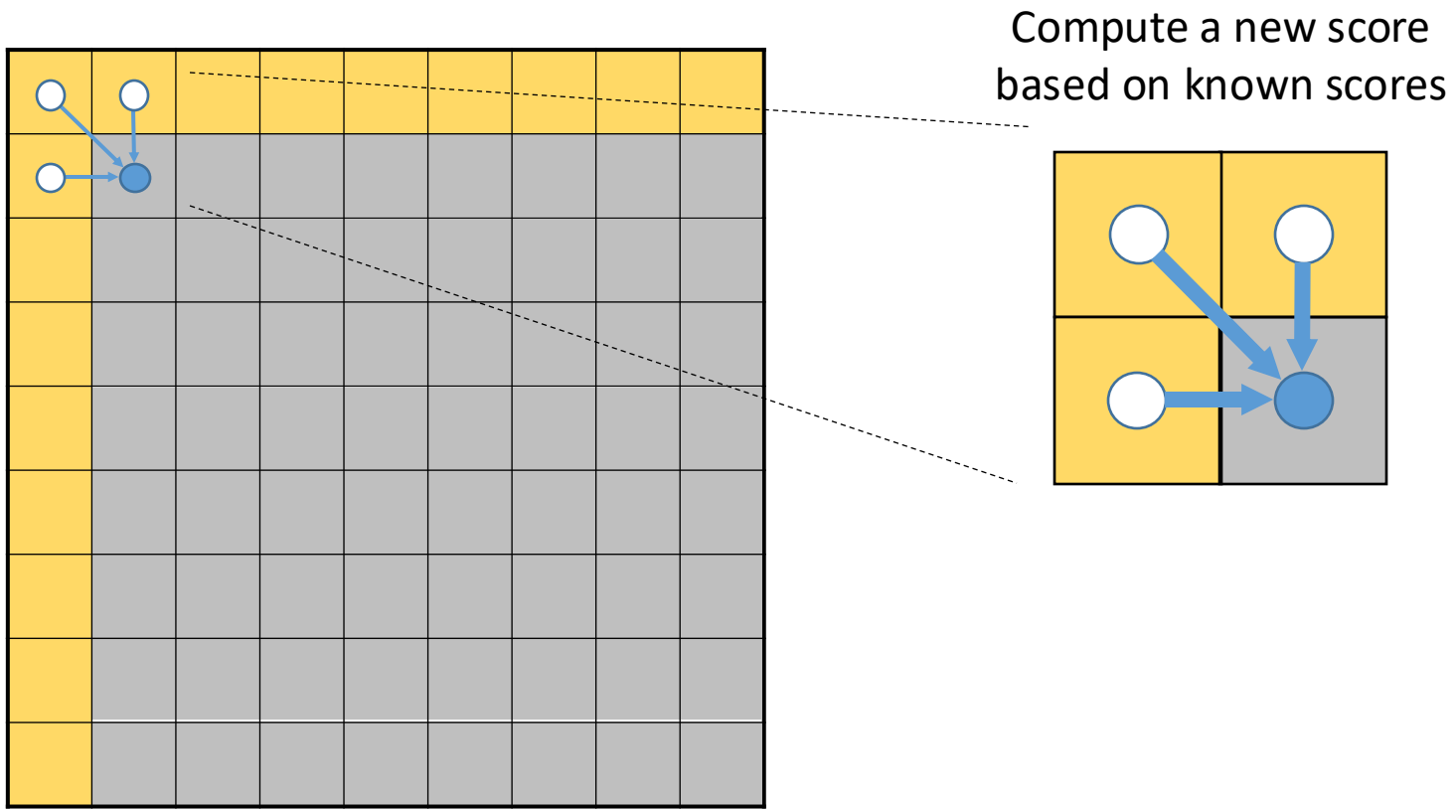
Dynamic programming for sequence similarity estimation

- Task: calculate the scores in the entire table.
- Initialization: the scores in first row and first column.
- Objective Function: calculate the score of a cell based on its upper, left and diagonal neighbors.

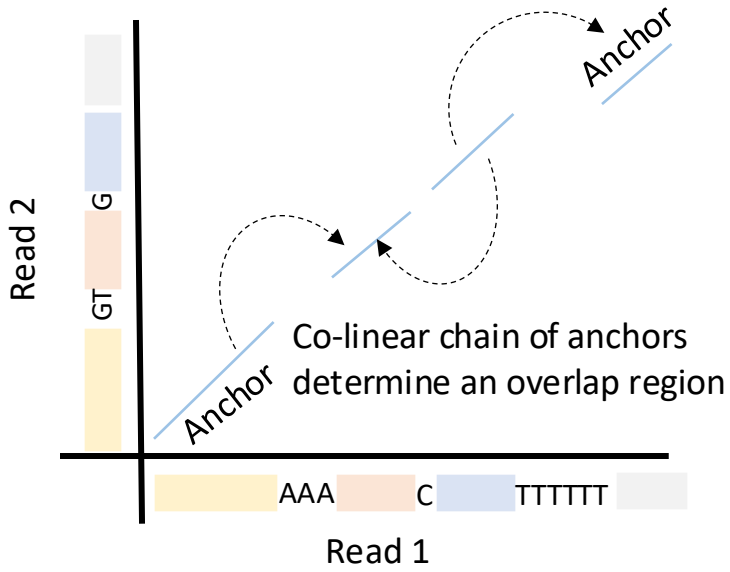


Dynamic programming for sequence similarity estimation

- Task: calculate the scores in the entire table.
- Initialization: the scores in first row and first column.
- Objective Function: calculate the score of a cell based on its upper, left and diagonal neighbors.



1D dynamic programming



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

Score for anchor i

Score for anchor j

matching bases

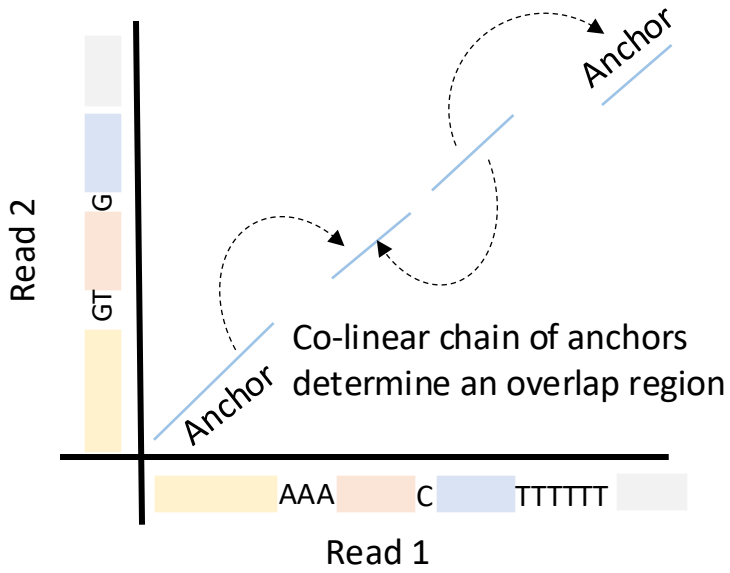
gap length

anchor length

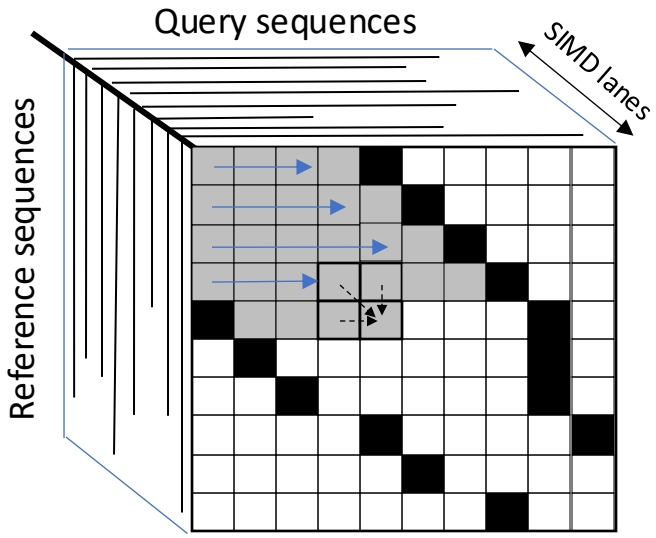
Chaining

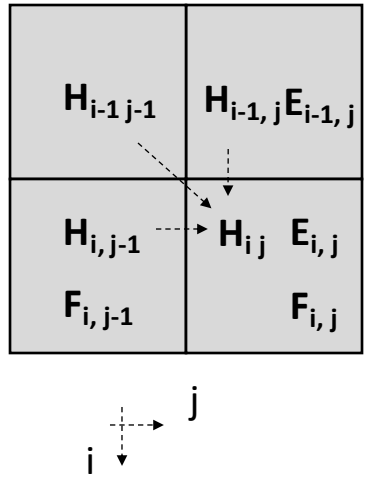
L. Guo, J. Lau, Z. Ruan, P. Wei, and J. Cong, "Hardware Acceleration of Long Read Pairwise Overlapping in Genome Sequencing: A Race Between FPGA and GPU," in 2019 IEEE 27th International Symposium On Field-Programmable Custom Computing Machines (FCCM), April 2019.

1D dynamic programming



2D dynamic programming





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

Score for anchor i

Score for anchor j

matching bases

gap length

anchor length

$$H_{ij} = \max\{H_{i-1,j-1} + s(i,j), E_{ij}, F_{ij}\}$$
$$E_{i+1,j} = \max\{H_{ij} - q, E_{ij}\} - e$$
$$F_{i,j+1} = \max\{H_{ij} - q, F_{ij}\} - e$$

Affine gap scoring

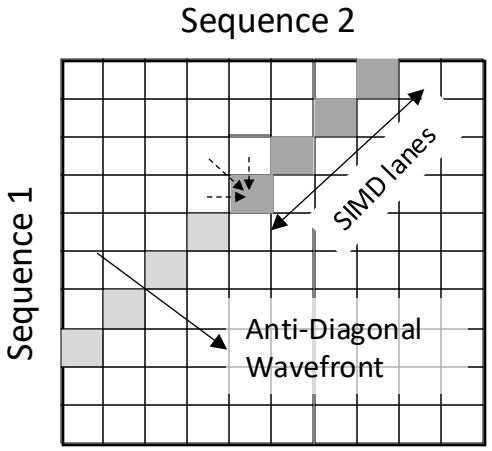
Chaining

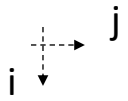
Banded Smith-Waterman

L. Guo, J. Lau, Z. Ruan, P. Wei, and J. Cong, "Hardware Acceleration of Long Read Pairwise Overlapping in Genome Sequencing: A Race Between FPGA and GPU," in 2019 IEEE 27th International Symposium On Field-Programmable Custom Computing Machines (FCCM), April 2019.

Dynamic programming benchmarks: Floating point 2D

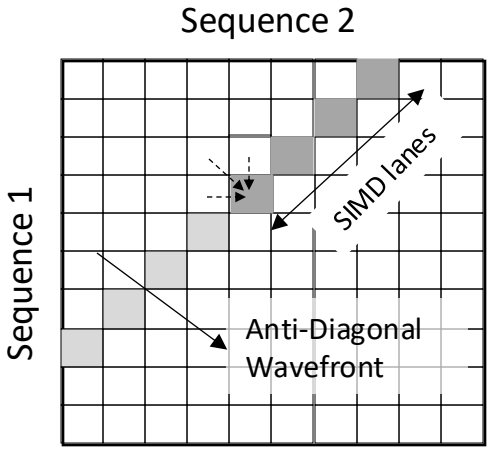
Pairwise Hidden Markov Model

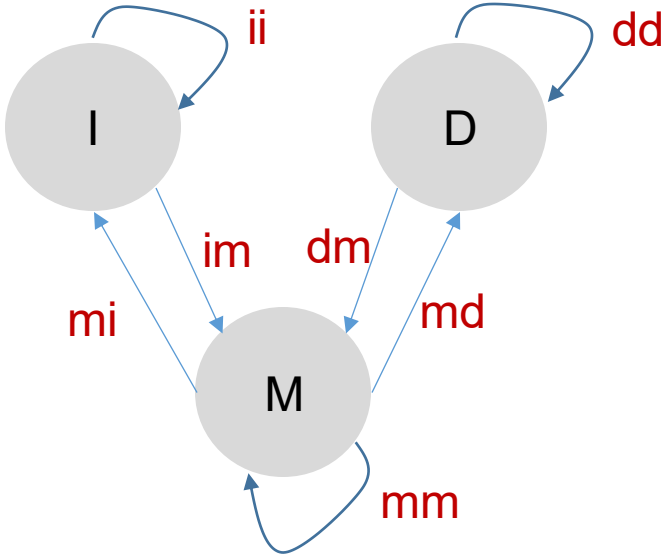


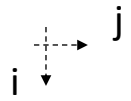


Dynamic programming benchmarks: Floating point 2D

Pairwise Hidden Markov Model

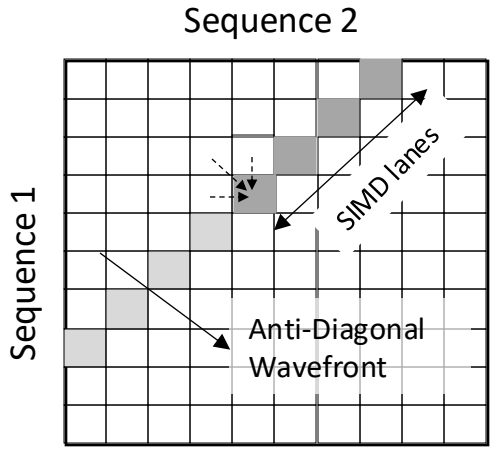


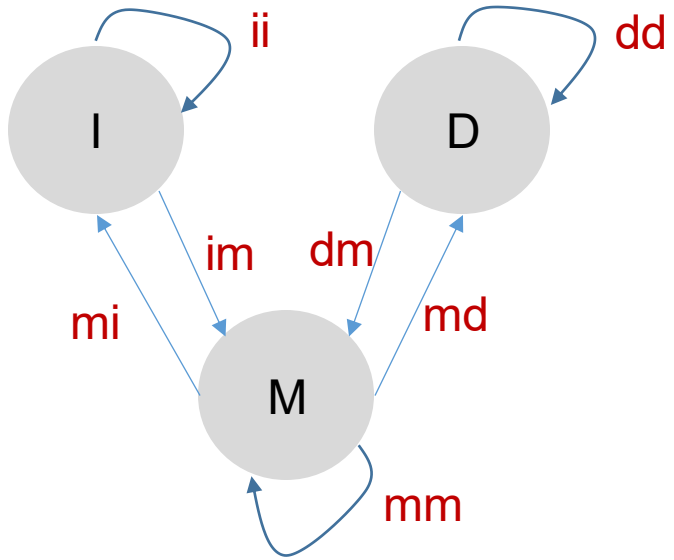


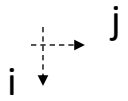


Dynamic programming benchmarks: Floating point 2D

Pairwise Hidden Markov Model







$$M(i,j) = (\ mm.\ M(i-1,j-1) + im.\ I(i-1,j-1) + dm.\ D(i-1,j-1)\ )$$

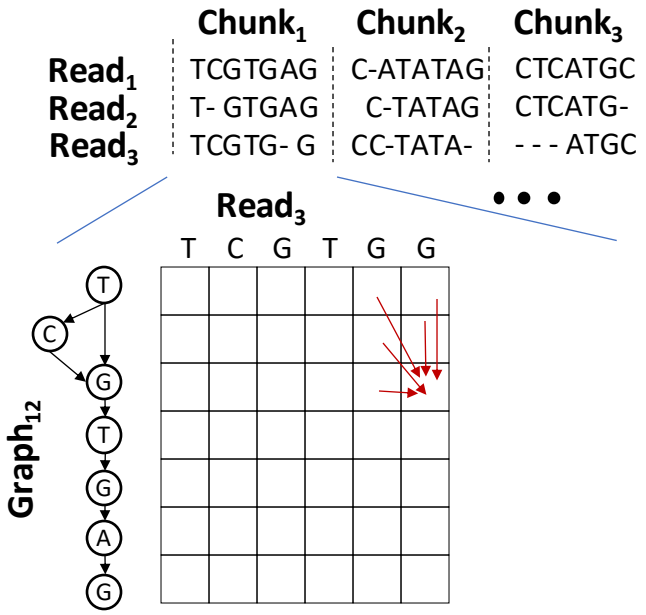
$$I(i,j) = mi.\ (i-1,j) + ii.\ I(i-1,j)$$

$$D(i,j) = md.\ (i,j-1) + dd(I,j-1)$$

Floating Point

Dynamic programming benchmarks: Sequence to Graph

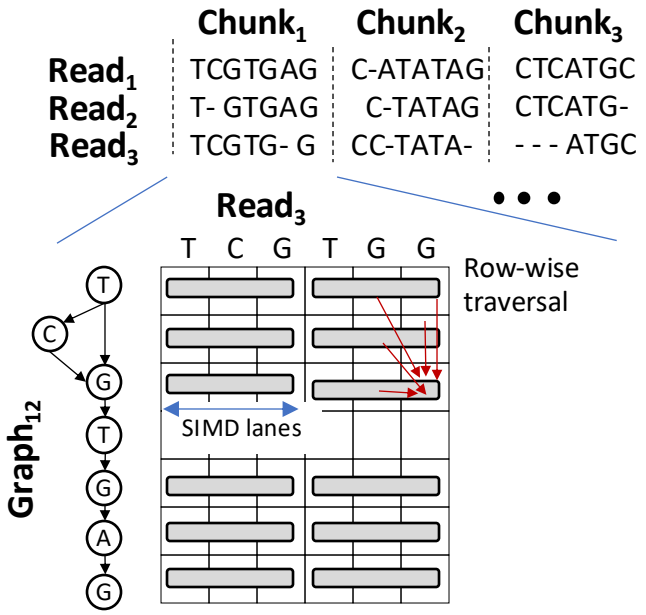
Partial Order Alignment



Sequence to Graph Alignment

Dynamic programming benchmarks: Sequence to Graph

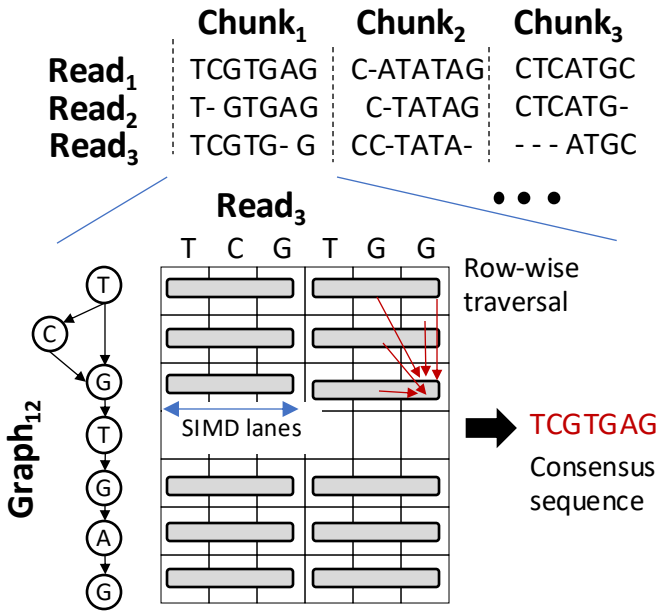
Partial Order Alignment



Sequence to Graph Alignment

Dynamic programming benchmarks: Sequence to Graph

Partial Order Alignment

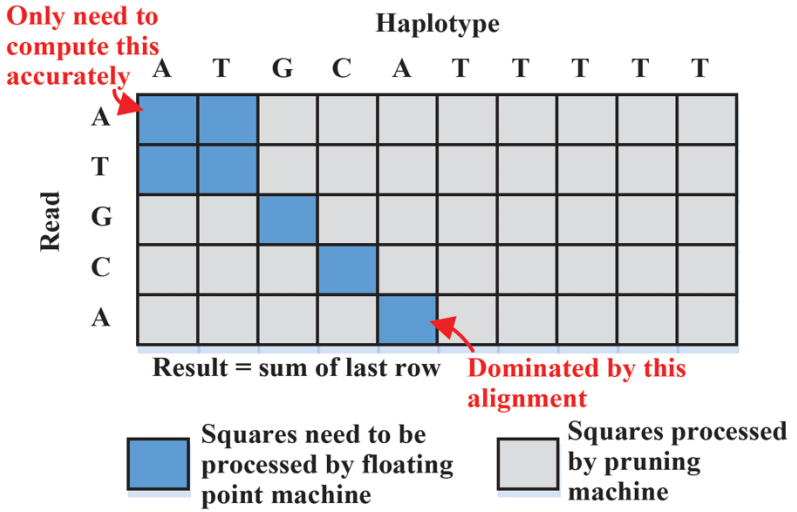


Sequence to Graph Alignment

Case Study – Accelerating Pairwise Hidden Markov Model (pairHMM)

Pruning algorithm for pairHMM with same output guarantees

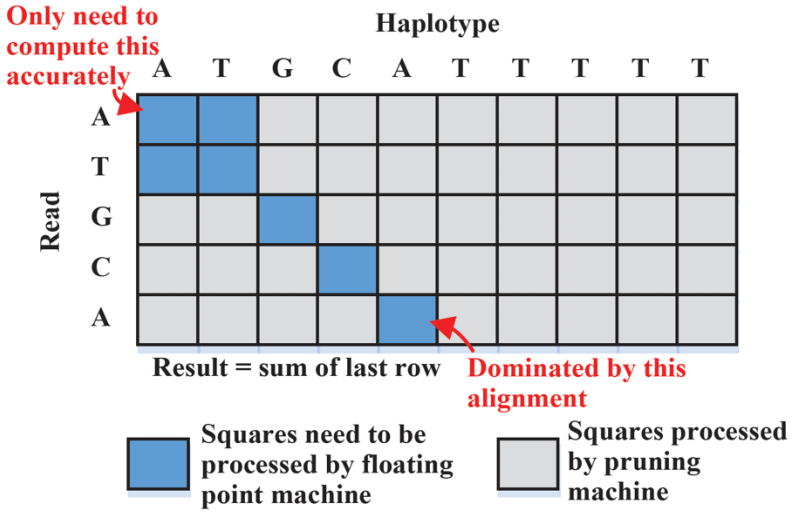
- Compute upper bound probability by replacing floating-point multiplication with log-domain addition and rounding up ($high_{res}$)
- Prune matrix and identify promising paths
- Compute lower bound probability with single-precision floating-point only for small unpruned area (low_{res})
- Recompute matrix in floating point if bound checks fail



"17.3 GCUPS pruning-based pair-hidden-Markov-model accelerator for next-generation DNA sequencing." In *2020 IEEE Symposium on VLSI Circuits*, pp. 1-2. IEEE, 2020.

Pruning algorithm for pairHMM with same output guarantees

- Compute upper bound probability by replacing floating-point multiplication with log-domain addition and rounding up ($high_{res}$)
- Prune matrix and identify promising paths
- Compute lower bound probability with single-precision floating-point only for small unpruned area (low_{res})
- Recompute matrix in floating point if bound checks fail



Original algorithm

Downstream: Call the most likely mutation

	Genotype	Probability	
G1	GG	0.1	
G2	AA	0.2	Called
G3	GA	0.1	

Pruned version

Downstream: Bound checks & call mutation

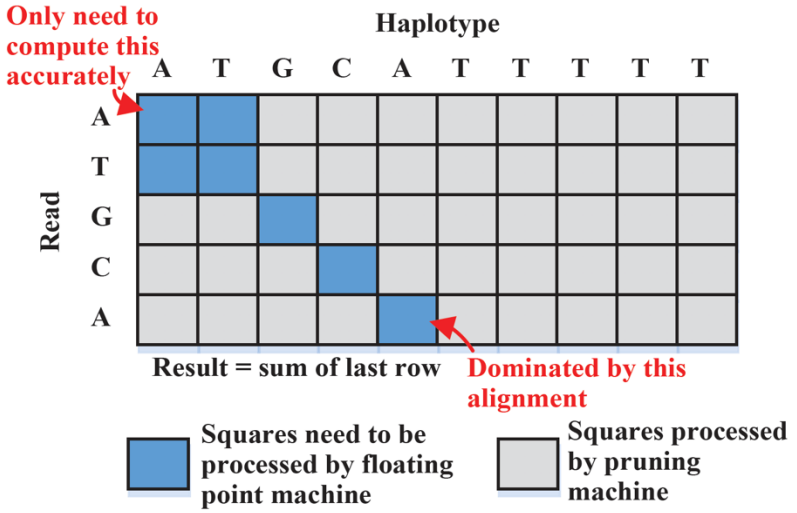
		Genotype	Probability
	G1	GG	[0.05,0.15]
Called	G2	AA	[0.19,0.25]
	G3	GA	[0.01,0.15]

Nonoverlapping bounds: G2 is the correct call

"17.3 GCUPS pruning-based pair-hidden-Markov-model accelerator for next-generation DNA sequencing." In *2020 IEEE Symposium on VLSI Circuits*, pp. 1-2. IEEE, 2020.

Pruning algorithm for pairHMM with same output guarantees

- Compute upper bound probability by replacing floating-point multiplication with log-domain addition and rounding up ($high_{res}$)
- Prune matrix and identify promising paths
- Compute lower bound probability with single-precision floating-point only for small unpruned area (low_{res})
- Recompute matrix in floating point if bound checks fail



Original algorithm

Downstream: Call the most likely mutation

	Genotype	Probability	
G1	GG	0.1	
G2	AA	0.2	Called
G3	GA	0.1	

Pruned version

Downstream: Bound checks & call mutation

	Genotype	Probability	
G1	GG	[0.05,0.15]	
Called G2	AA	[0.19,0.25]	Nonoverlapping bounds:
G3	GA	[0.01,0.15]	G2 is the correct call

Results:

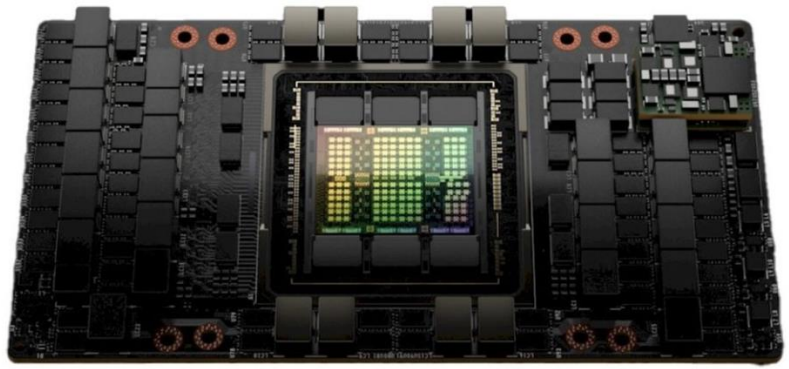
43x fewer cells computed in precise floating point

8.3x higher throughput (GCUPS) than floating-point ASIC of the same area

"17.3 GCUPS pruning-based pair-hidden-Markov-model accelerator for next-generation DNA sequencing." In *2020 IEEE Symposium on VLSI Circuits*, pp. 1-2. IEEE, 2020.

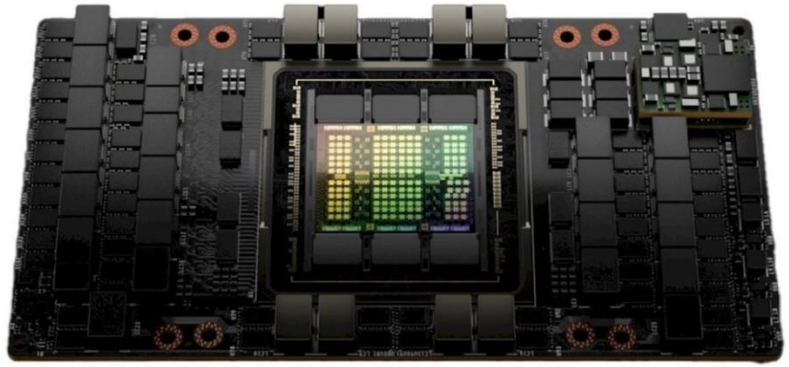
Case Study – Generalized Dynamic Programming Acceleration

DPX Instruction Extension in NVIDIA Hopper GPU



https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

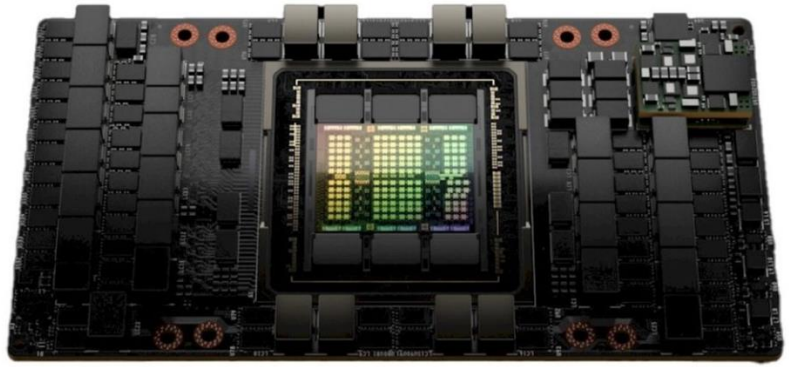
DPX Instruction Extension in NVIDIA Hopper GPU



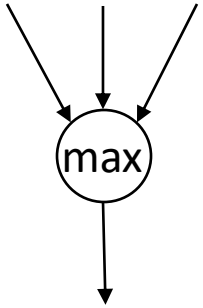
ISA Extension

https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

DPX Instruction Extension in NVIDIA Hopper GPU



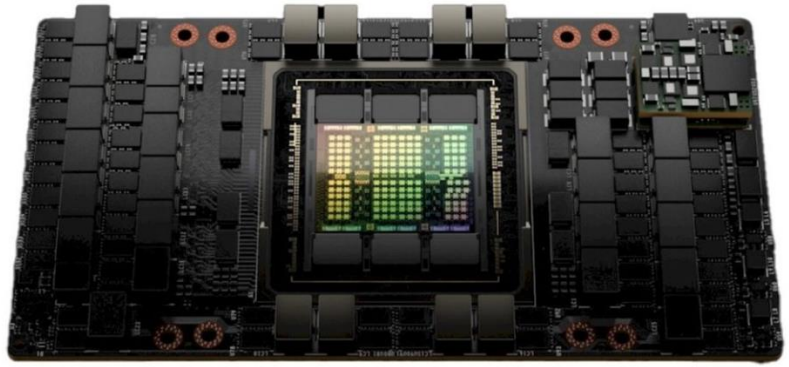
ISA Extension



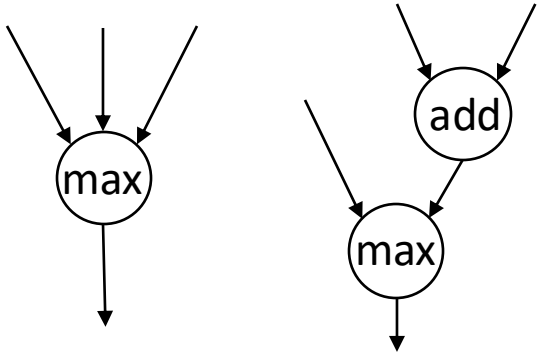
(a)

https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

DPX Instruction Extension in NVIDIA Hopper GPU

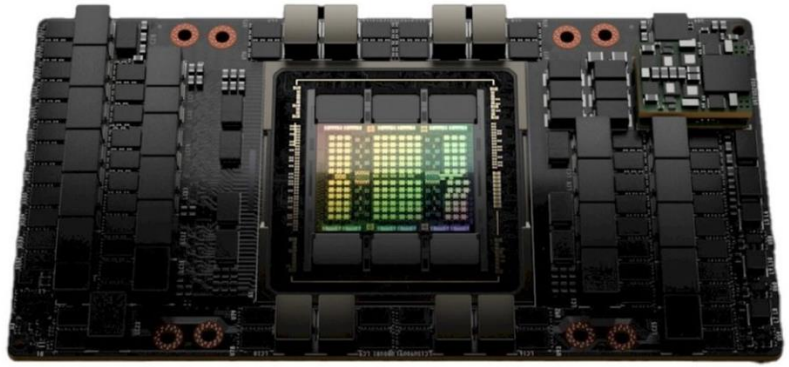


ISA Extension

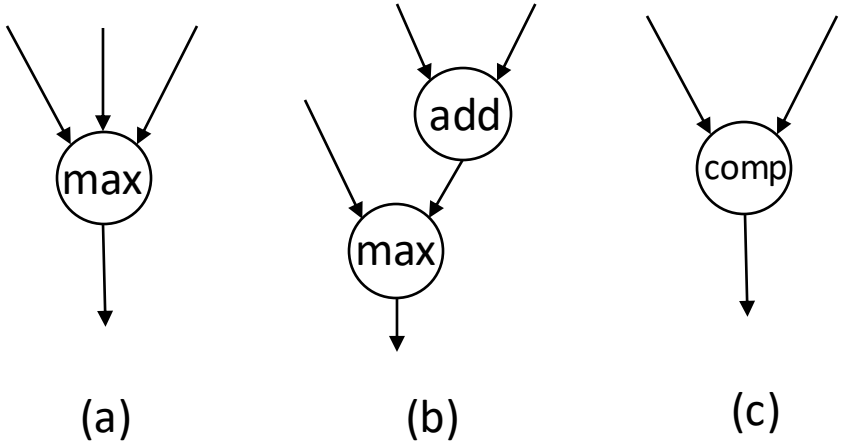


https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

DPX Instruction Extension in NVIDIA Hopper GPU

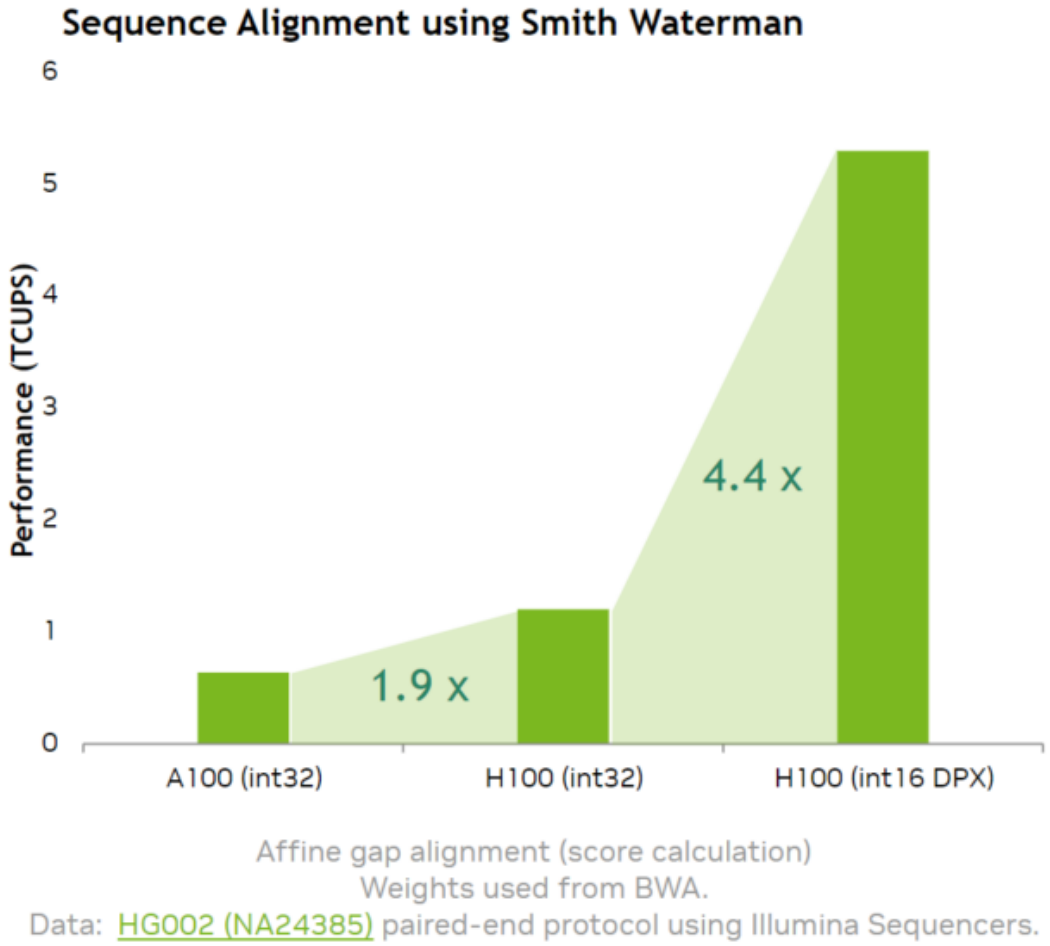


ISA Extension

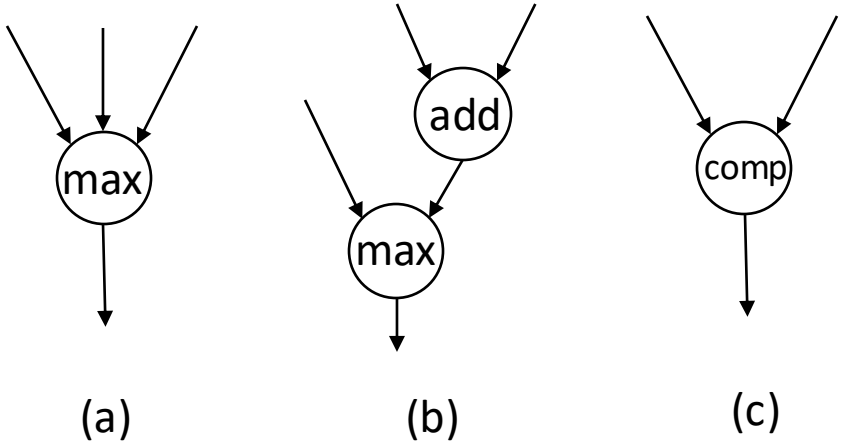


https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

DPX Instruction Extension in NVIDIA Hopper GPU



ISA Extension



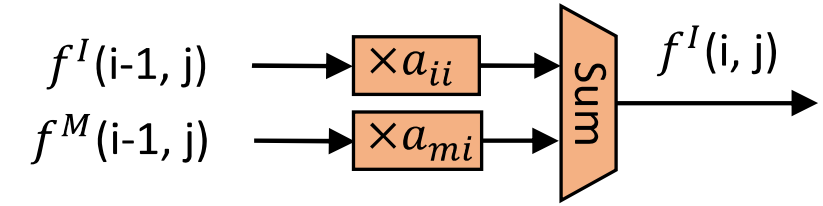
https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

Diversity of dynamic programming in genomic kernels

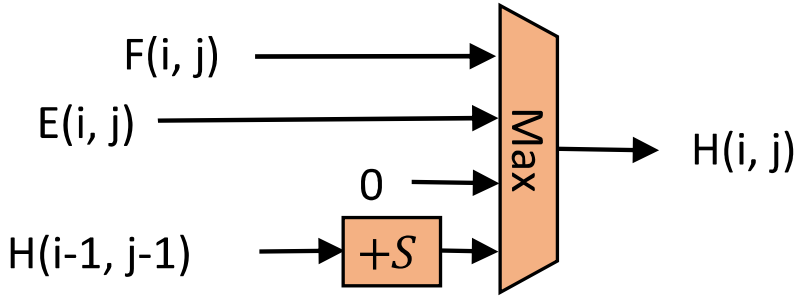
Kernel	Application	Dimension and Size	Dependency	Data Type
bsw	Read Alignment	2D $\sim 120 \times 60$	Last 2 Wave-fronts	Int 8/16
pairHMM	Variant Calling	2D $\sim 100 \times 60$	Last 2 Wave-fronts	Floating Point
poa	Error Correction	2D $\sim 1000 \times 500$	Graph Structure	Int 32
chain	Read Alignment	1D ~ 20000	Last N (~ 25) Anchors	Int 32

Different objective functions computed in each cell

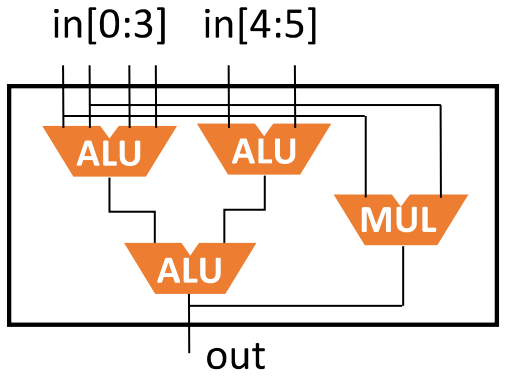
GenDP: Support for intra-cell objective functions



(a) Reduction Data-Flow in PairHMM



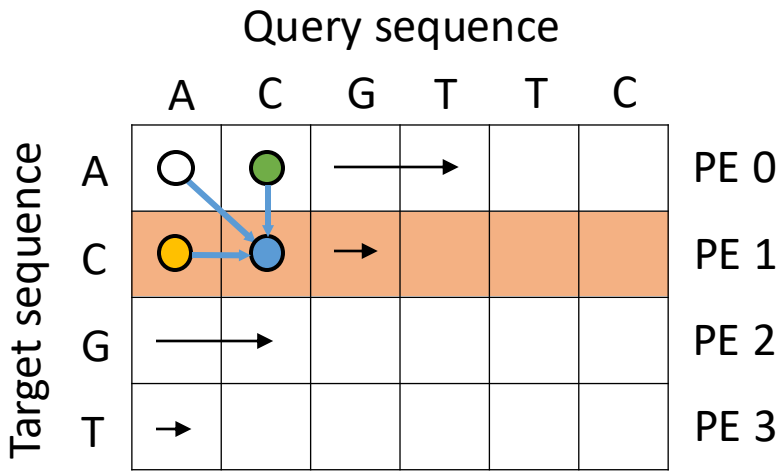
(b) Reduction Data-Flow in BSW



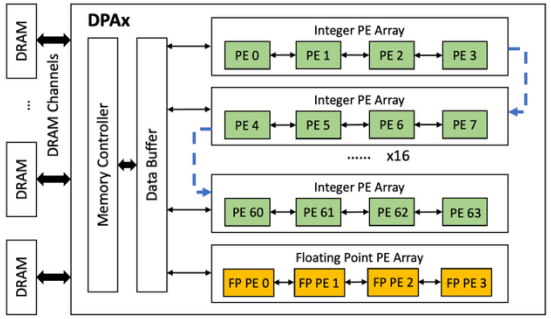
Compute unit – 2-level reduction tree

"GenDP: A framework of dynamic programming acceleration for genome sequencing analysis." In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1-15. 2023.
Communications of the ACM, Volume 68, Issue 05 Pages 81 - 90 https://doi.org/10.1145/3712168

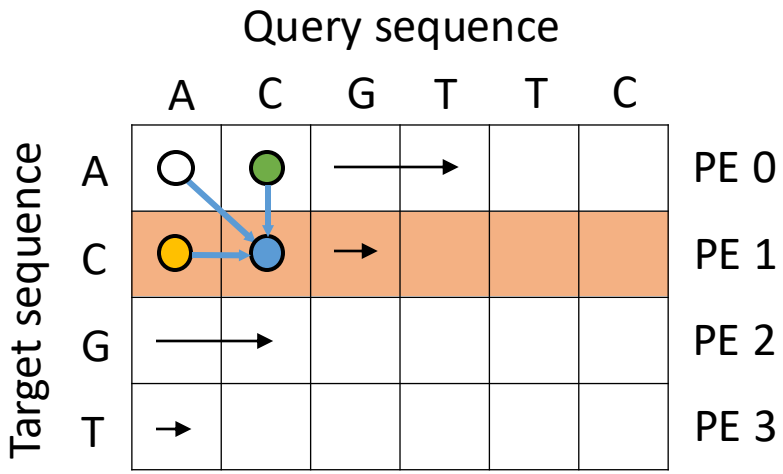
GenDP: Inter-cell dependencies and multi-precision support



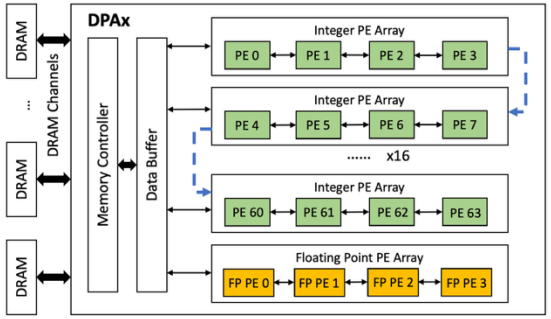
GenDP: Inter-cell dependencies and multi-precision support



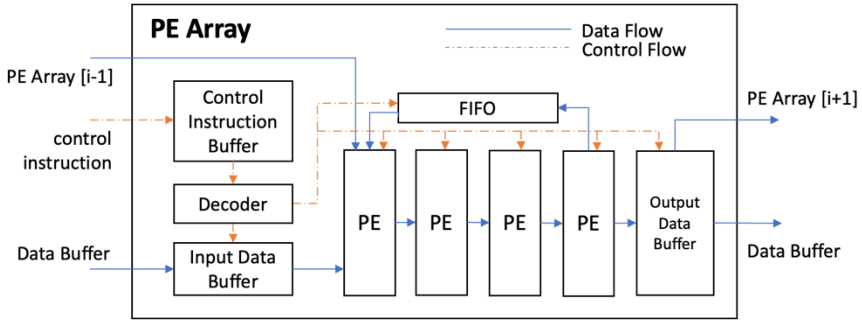
Integer and floating point PE array



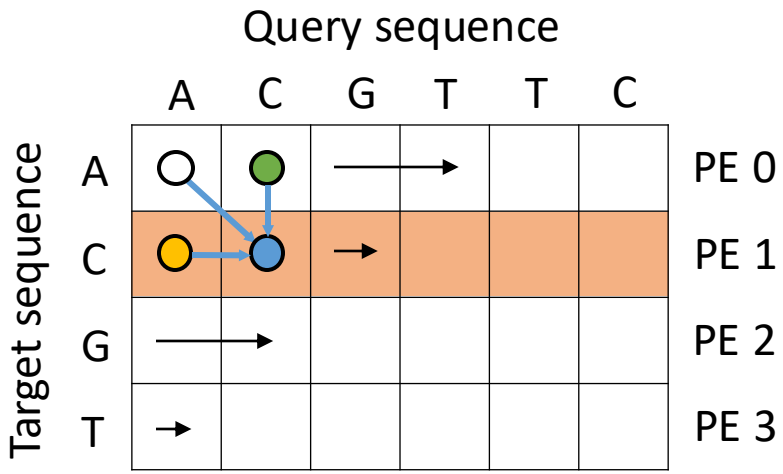
GenDP: Inter-cell dependencies and multi-precision support



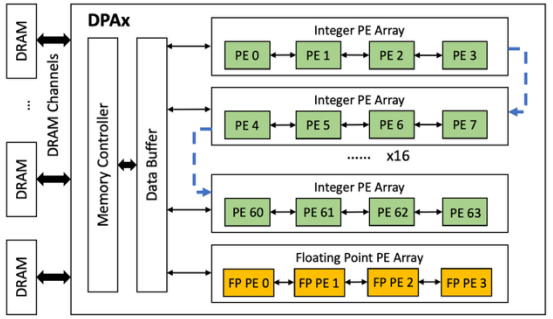
Integer and floating point PE array



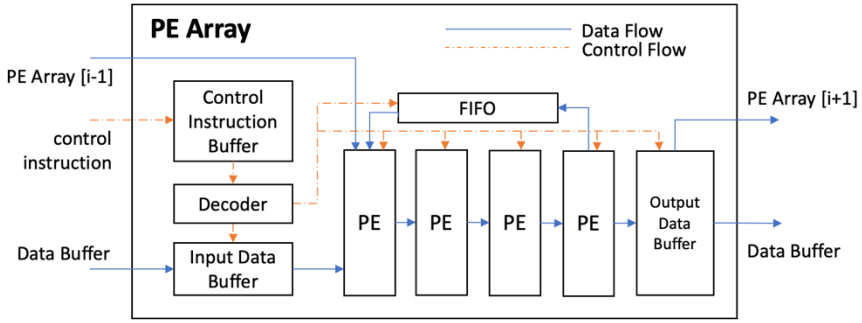
Local dependency - 1-Dimension systolic PE array with FIFO



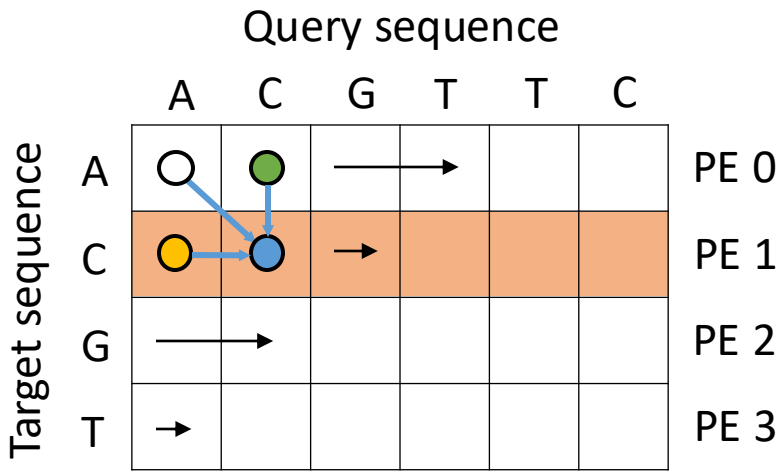
GenDP: Inter-cell dependencies and multi-precision support

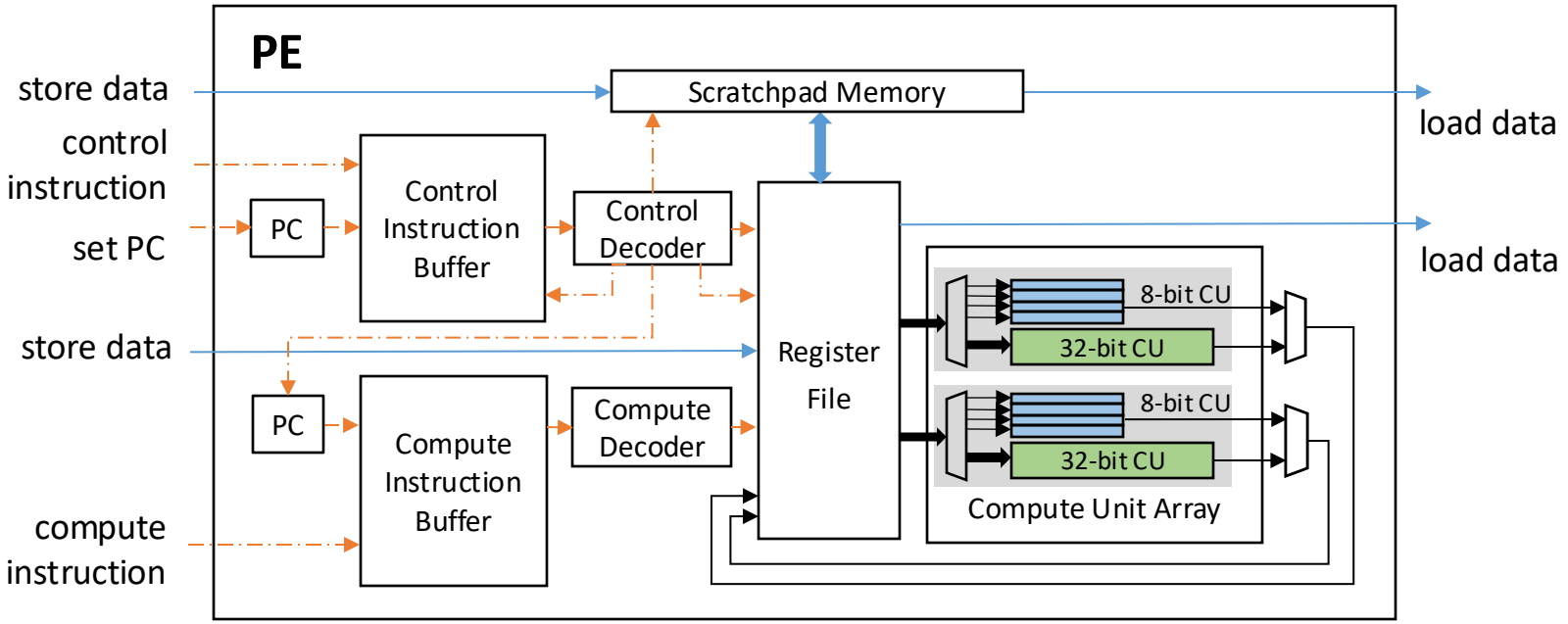


Integer and floating point PE array



Local dependency - 1-Dimension systolic PE array with FIFO





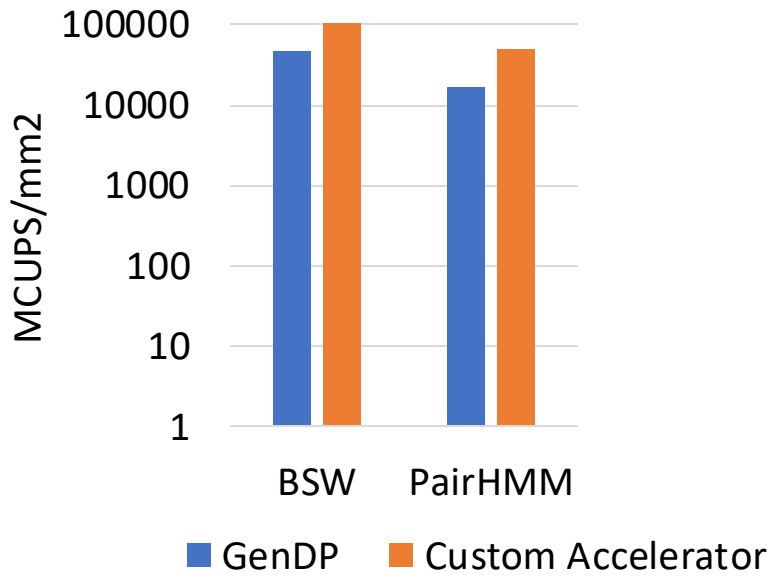
Software managed scratchpad memory for long range dependencies

GenDP performance

- Metrics: Throughput/Area – Million Cell Updates per Second/*mm*2 (MCUPS/*mm*2)

GenDP performance

- Metrics: Throughput/Area – Million Cell Updates per Second/*mm*2 (MCUPS/*mm*2)
- GenDP has 2.8x slowdown when compared to custom accelerators



Conclusion

- Genomic sequence data is rapidly increasing in volume and diversity
- Modern sequence analysis pipelines present unique computation challenges that multi-core CPUs and GPUs have struggled to meet
- The GenomicsBench benchmark suite is an initial attempt to capture this diversity with the goal of enabling the development of custom hardware architectures
- GenomicsBench has already helped to develop several hardware accelerators
  - Enumerated Radix Tree accelerator for memory-intensive exact match search
  - GenDP for dynamic programming acceleration
  - Pruning-based accelerator for PairHMM

Acknowledgement

University of Michigan

Reetuparna Das (Advisor)
Satish Narayanasamy
David Blaauw
Jack Wadden
Kush Goliya
Xiao Wu
Nathan Ozog

Intel Labs

Sanchit Misra
Md. Vasimuddin
Somnath Paul

Microsoft Research/UC Berkeley AMPLab

Bill Bolosky
Ravi Pandya
Matei Zaharia
David Patterson

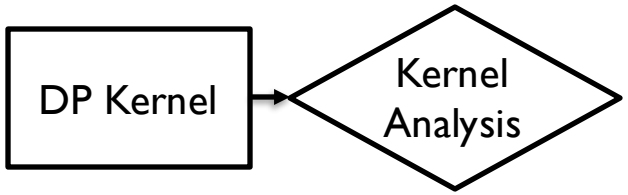
GenDP: A General Dynamic Programming Accelerator for Genomics

- **DPAx**: programmable dynamic programming (DP) accelerator.
- **DPMap**: map the objective function of DP algorithm to DPAx accelerator.

"GenDP: A framework of dynamic programming acceleration for genome sequencing analysis." In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1-15. 2023.
Communications of the ACM, Volume 68, Issue 05 Pages 81 - 90 https://doi.org/10.1145/3712168

GenDP: A General Dynamic Programming Accelerator for Genomics

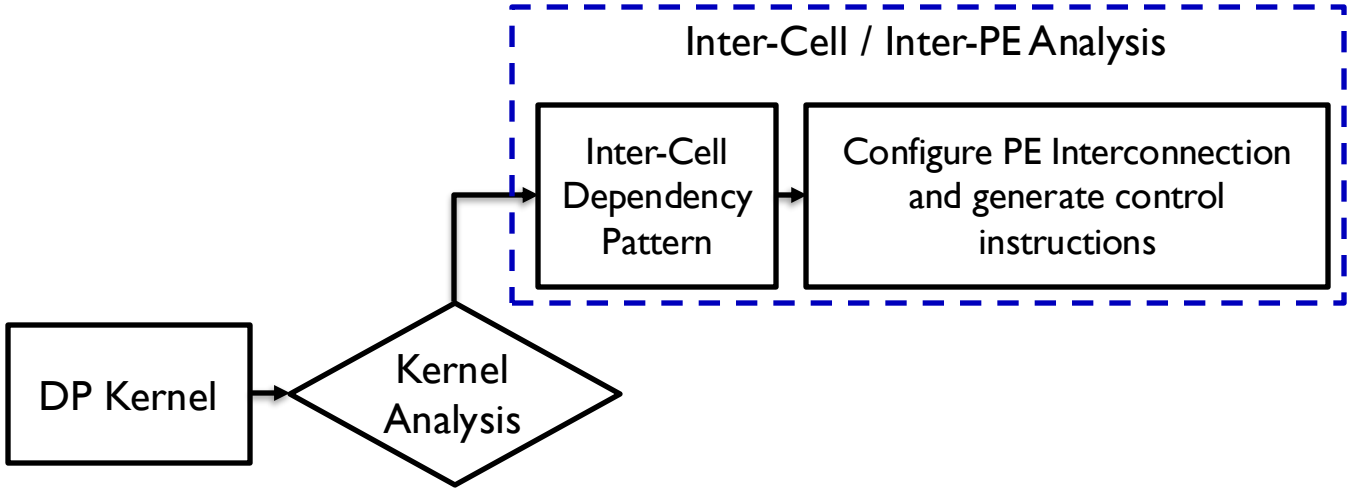
- **DPAx**: programmable dynamic programming (DP) accelerator.
- **DPMap**: map the objective function of DP algorithm to DPAx accelerator.



"GenDP: A framework of dynamic programming acceleration for genome sequencing analysis." In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1-15. 2023.
Communications of the ACM, Volume 68, Issue 05 Pages 81 - 90 https://doi.org/10.1145/3712168

GenDP: A General Dynamic Programming Accelerator for Genomics

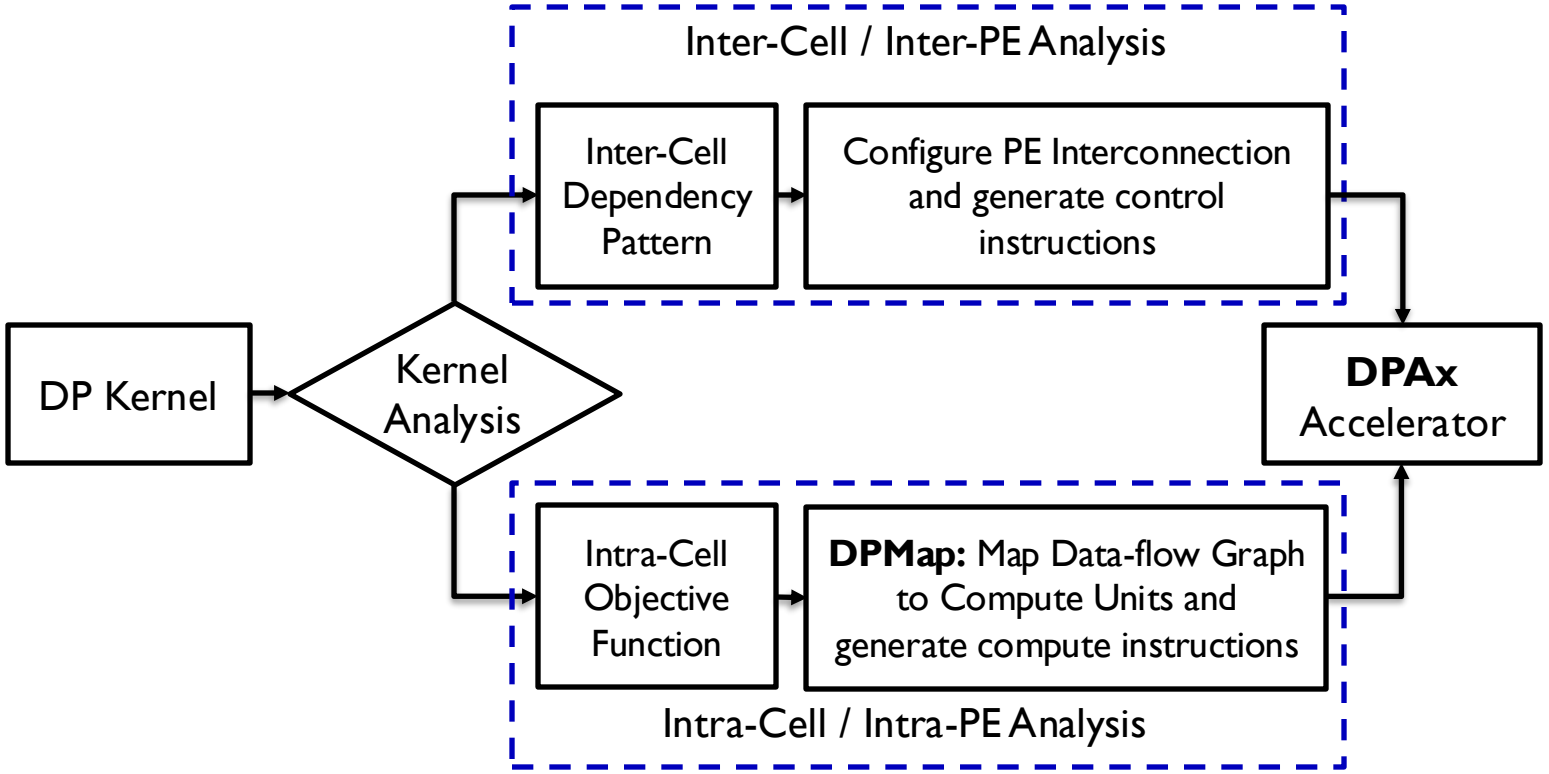
- **DPAx**: programmable dynamic programming (DP) accelerator.
- **DPMap**: map the objective function of DP algorithm to DPAx accelerator.



"GenDP: A framework of dynamic programming acceleration for genome sequencing analysis." In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1-15. 2023.
Communications of the ACM, Volume 68, Issue 05 Pages 81 - 90 https://doi.org/10.1145/3712168

GenDP: A General Dynamic Programming Accelerator for Genomics

- **DPAx**: programmable dynamic programming (DP) accelerator.
- **DPMap**: map the objective function of DP algorithm to DPAx accelerator.



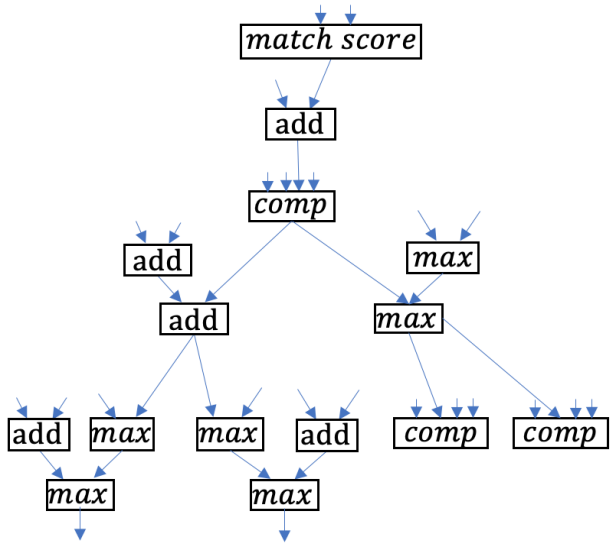
"GenDP: A framework of dynamic programming acceleration for genome sequencing analysis." In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pp. 1-15. 2023.
Communications of the ACM, Volume 68, Issue 05 Pages 81 - 90 https://doi.org/10.1145/3712168

DPMap Algorithm

- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.

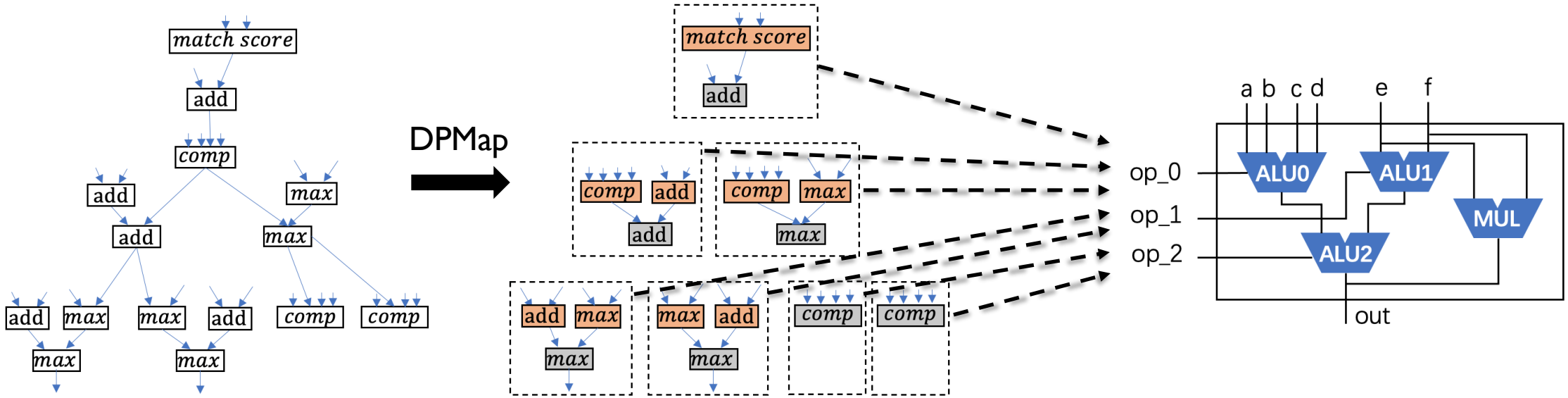
DPMap Algorithm

- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.



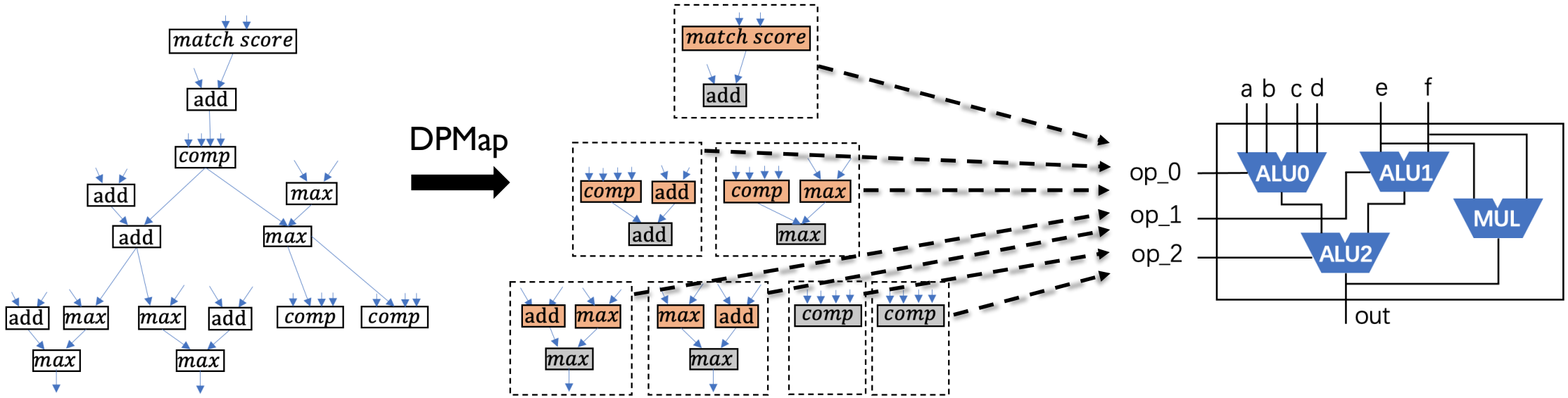
DPMap Algorithm

- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.



DPMap Algorithm

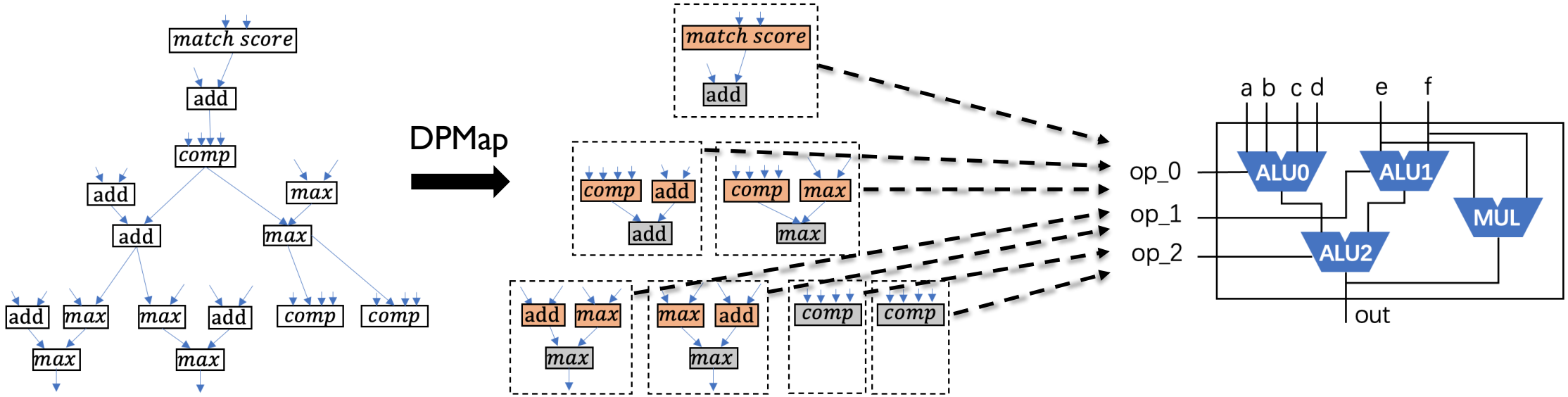
- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.
  - **Partitioning**: Break the data-flow graph with 4-input ALU and Multiplier
  - **Seeding**: Look for vertices that could be mapped to the 2nd level
  - **Refinement**: Break the single-strand structure



DPMap Algorithm

- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.
  - **Partitioning**: Break the data-flow graph with 4-input ALU and Multiplier
  - **Seeding**: Look for vertices that could be mapped to the 2nd level
  - **Refinement**: Break the single-strand structure

More details in the paper

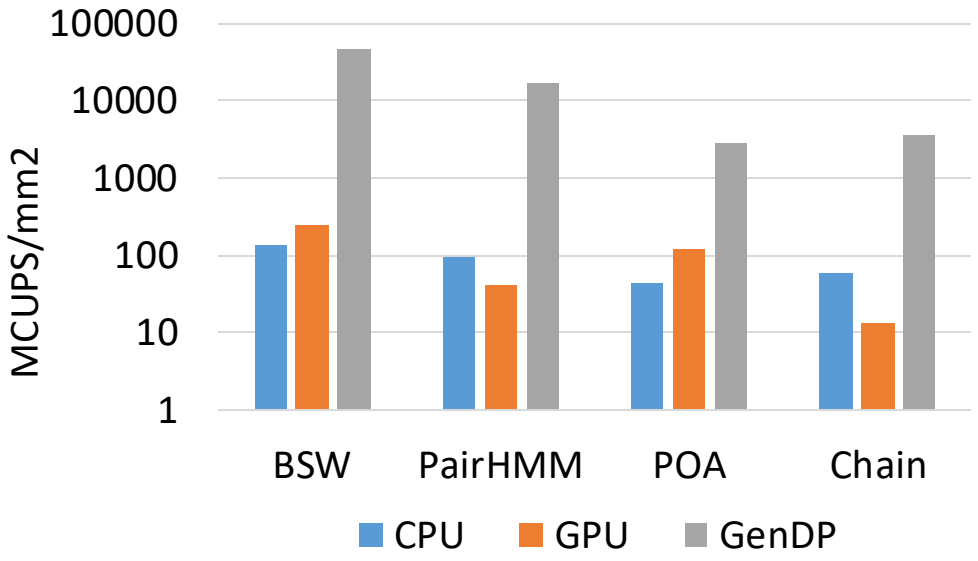


GenDP performance

- Metrics: Throughput/Area – Million Cell Updates per Second/*mm*2 (MCUPS/*mm*2)

GenDP performance

- Metrics: Throughput/Area – Million Cell Updates per Second/*mm*2 (MCUPS/*mm*2)
- GenDP achieves 157.8× throughput/mm^2 over GPU



GenDP performance

- Metrics: Throughput/Area – Million Cell Updates per Second/*mm*2 (MCUPS/*mm*2)
- GenDP achieves 157.8× throughput/mm^2 over GPU
- GenDP has 2.8x slowdown when compared to custom accelerators