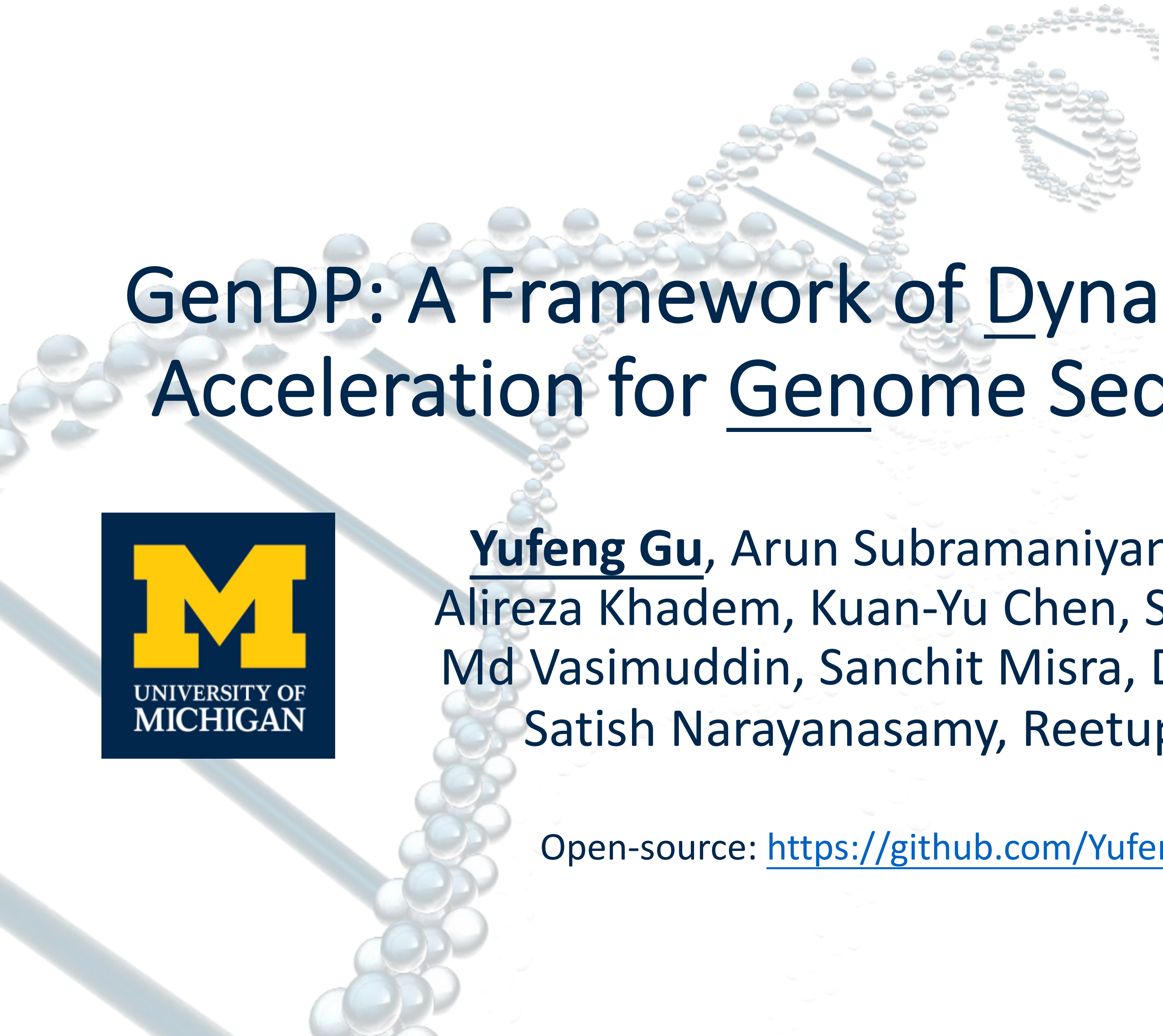


GenDP: A Framework of Dynamic Programming Acceleration for Genome Sequencing Analysis

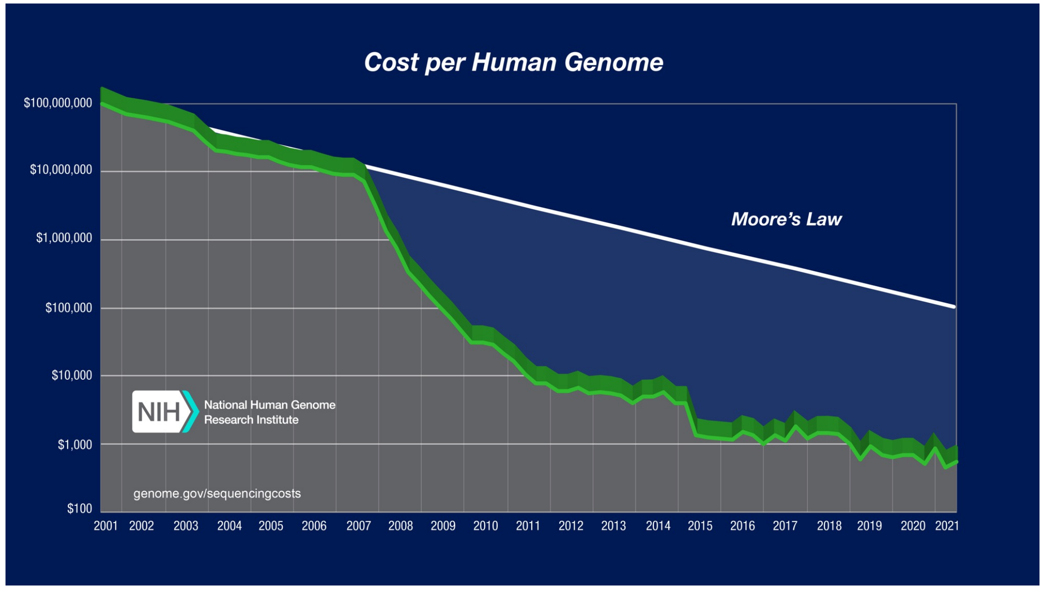
Yufeng Gu, Arun Subramaniyan, Tim Dunn, Alireza Khadem, Kuan-Yu Chen, Somnath Paul, Md Vasimuddin, Sanchit Misra, David Blaauw, Satish Narayanasamy, Reetuparna Das

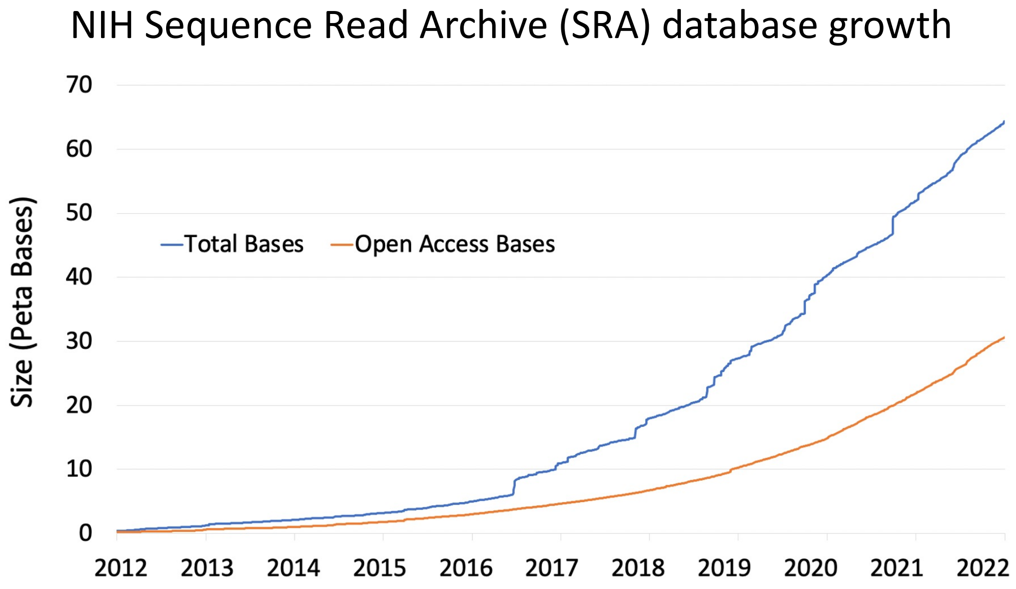
UNIVERSITY OF MICHIGAN

intel

Open-source: https://github.com/Yufeng98/GenDP

Genetic data is produced cheaper and faster

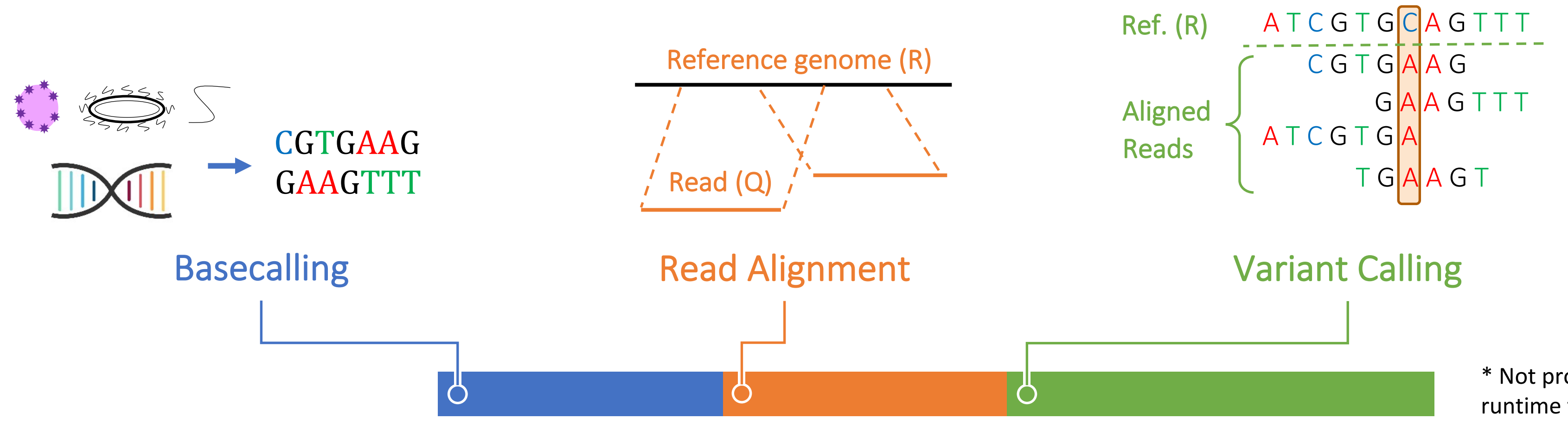




The explosive growth of genetic data motivates hardware acceleration for genome sequencing analysis

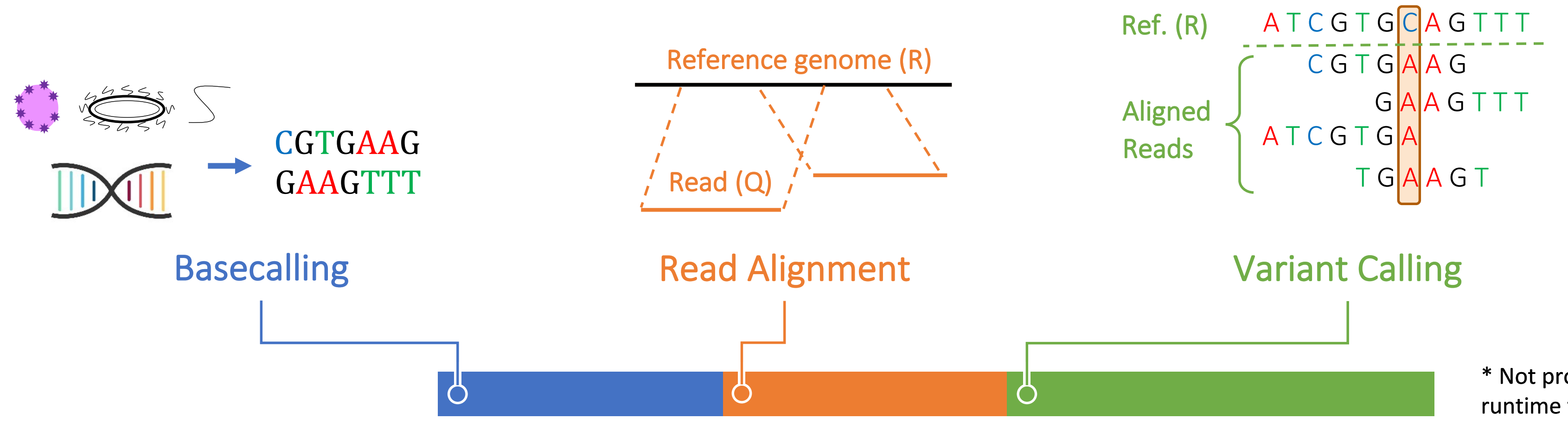
https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data
Data Source: https://www.ncbi.nlm.nih.gov/Traces/sra/sra_stat.cgi

How to accelerate genome sequencing pipelines



* Not proportional to the runtime for each stage

How to accelerate genome sequencing pipelines



* Not proportional to the runtime for each stage

Software tools

Short Reads	Bustard	BWA-MEM2	GATK Haplotype Caller, Platypus
Long Reads	Bonito, Guppy	Minimap2	Clair

General-Purpose Hardware

Intel Genomics Kernel Library (GKL)

NIVIDIA Clara Parabricks Toolkit

Performance

Design Cost

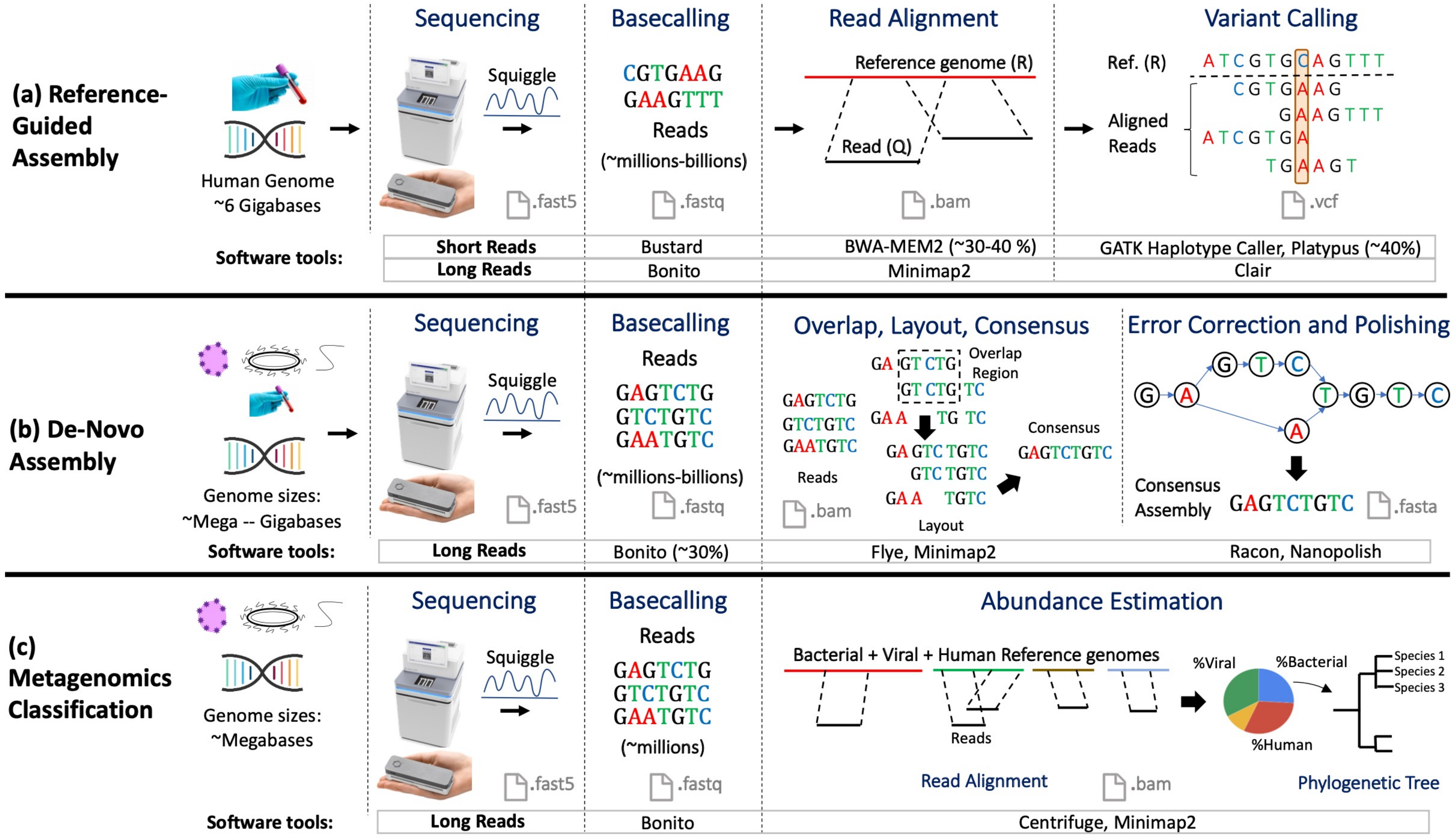
Domain specific accelerator bridges the gap between general-purpose processing and custom accelerator

SquiggleFilter | GenAx SeedEx ERT GenASM Darwin | Prune-PairHMM

Custom Accelerators

......	......	......

Genomic Sequencing Pipelines



(a) Software tools:

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

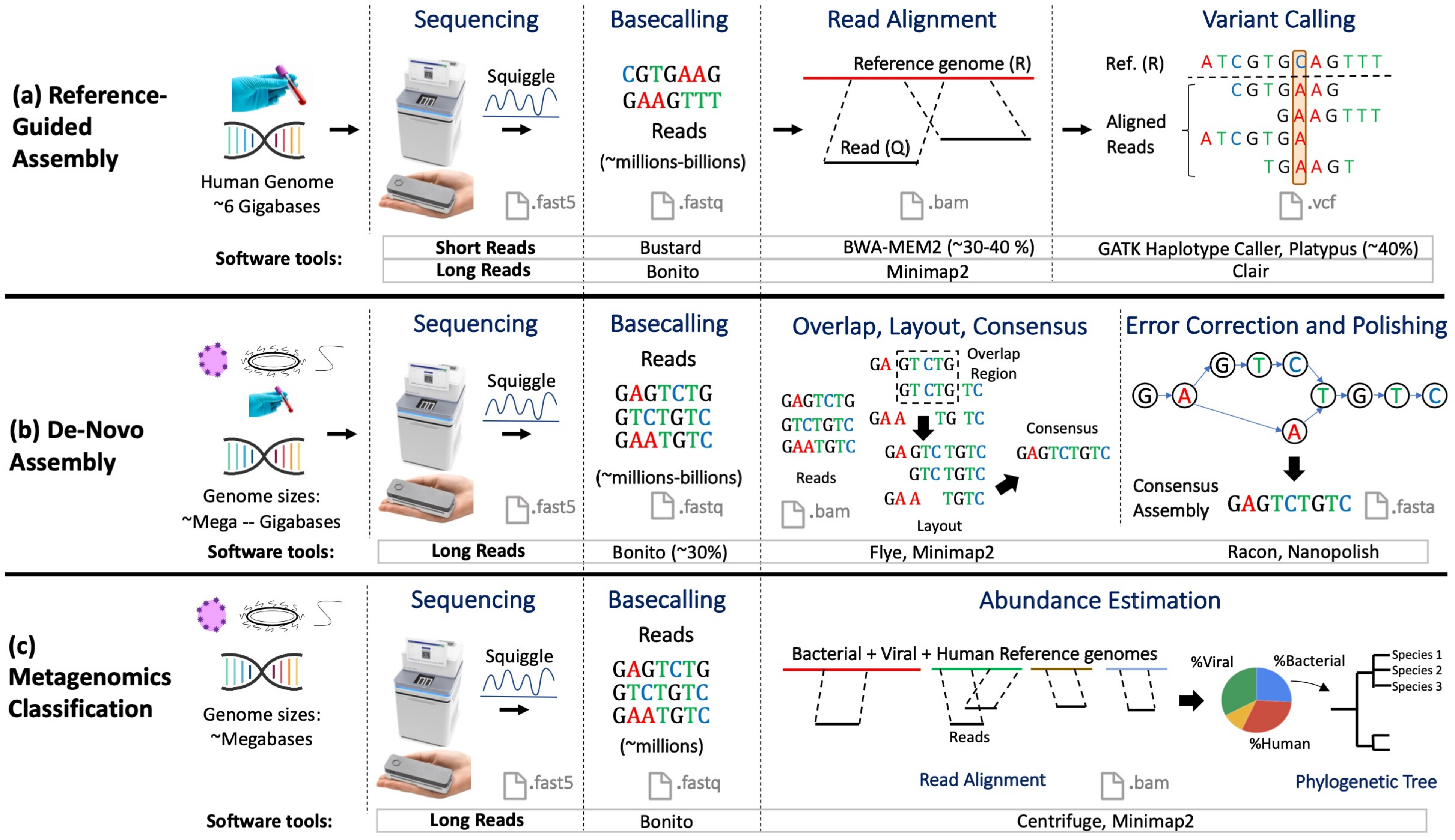
(b) Software tools:

	Sequencing	Basecalling	Overlap, Layout, Consensus	Error Correction and Polishing
	Long Reads	Bonito (~30%)	Flye, Minimap2	Racon, Nanopolish

(c) Software tools:

	Sequencing	Basecalling	Abundance Estimation
	Long Reads	Bonito	Centrifuge, Minimap2

Genomic Sequencing Pipelines



Software tools:

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

Software tools:

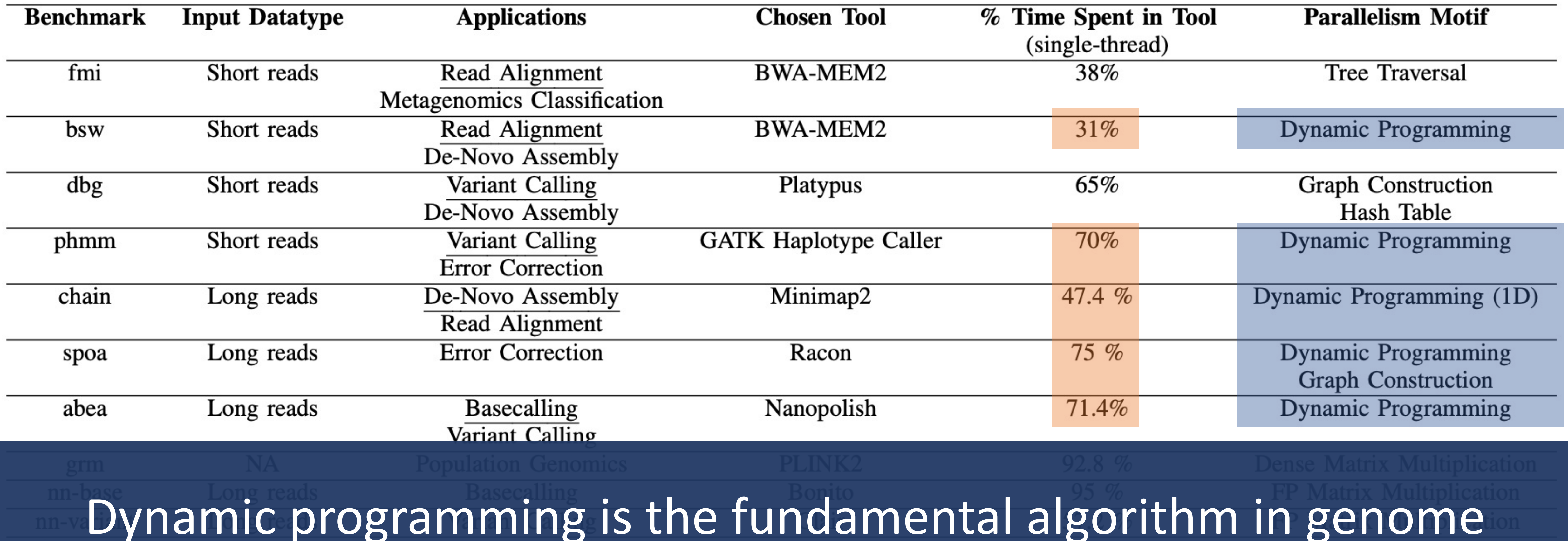
Sequencing	Basecalling	Overlap, Layout, Consensus	Error Correction and Polishing
Long Reads	Bonito (~30%)	Flye, Minimap2	Racon, Nanopolish

Software tools:

Sequencing	Basecalling	Abundance Estimation
Long Reads	Bonito	Centrifuge, Minimap2

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

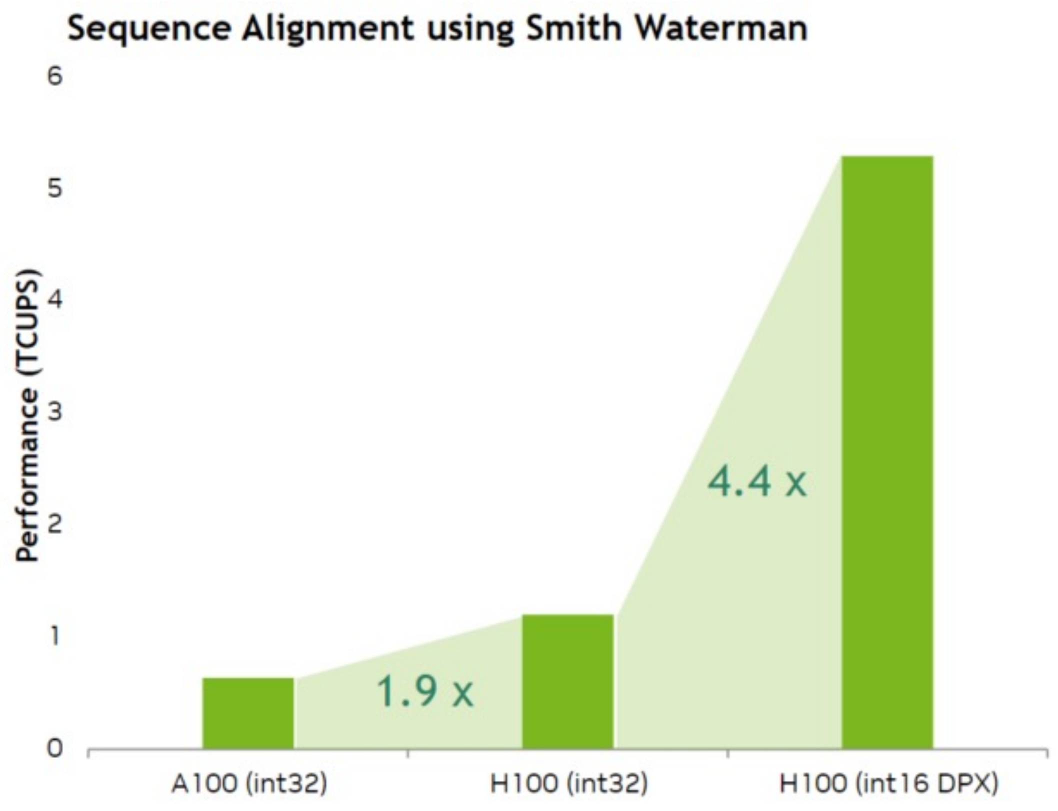
Dynamic Programming Kernels in GenomicsBench



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-va	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

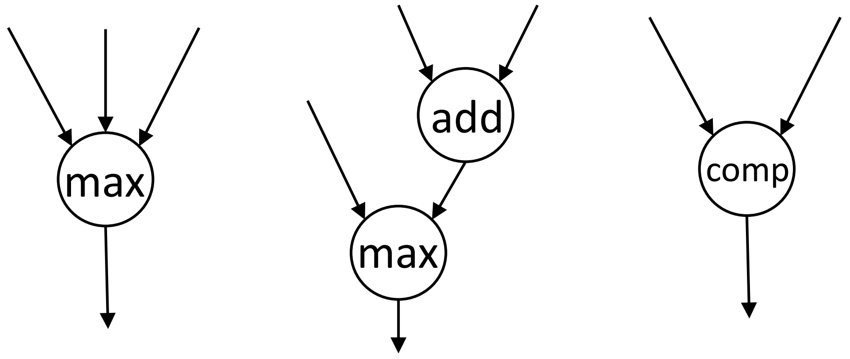
Dynamic programming is the fundamental algorithm in genome sequencing analysis and motivates a domain specific accelerator

DPX Instruction Extension in NVIDIA Hopper GPU



Affine gap alignment (score calculation)
Weights used from BWA.
Data: HG002 (NA24385) paired-end protocol using Illumina Sequencers.

ISA Extension

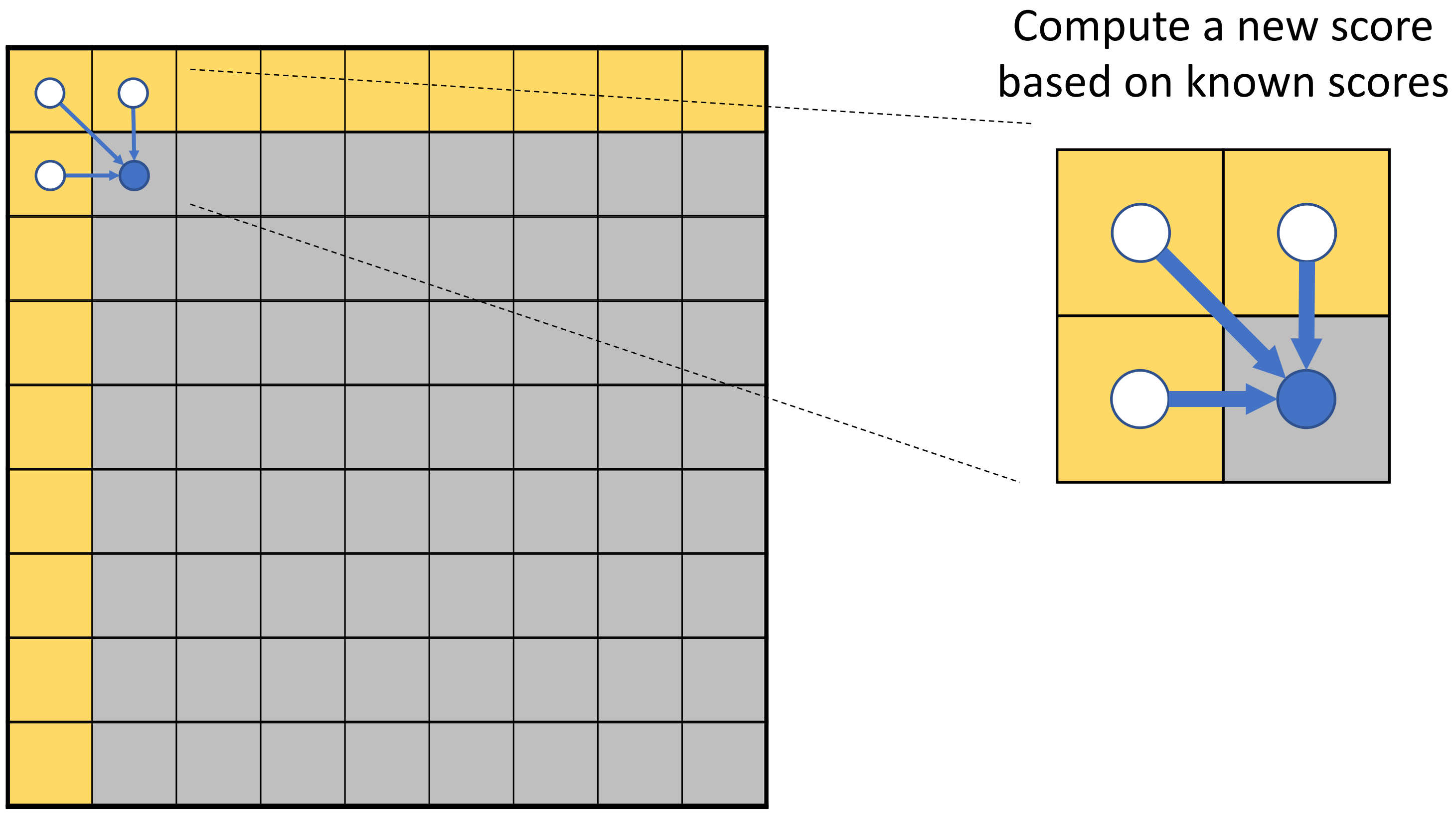


(a) (b) (c)

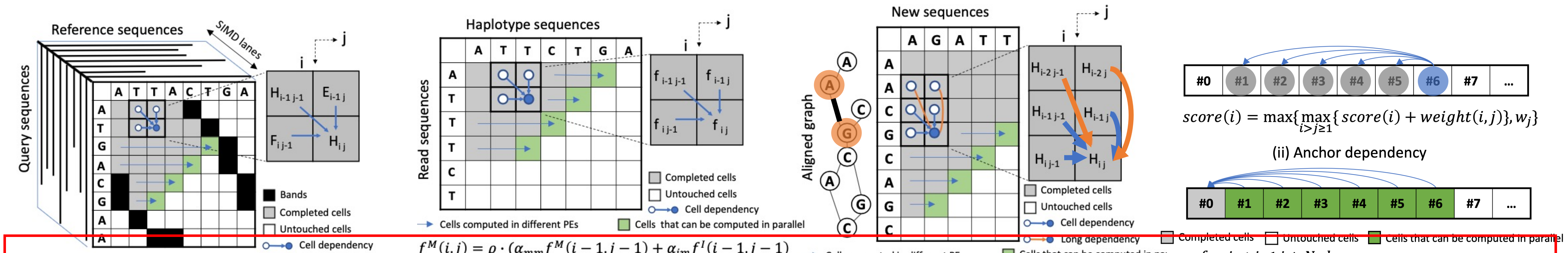
https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/

Dynamic Programming (DP)

- 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.



Genomics DP Kernels



$$H(i,j) = \max\{H(i-1,j-1) + S(X(i),Y(j)), E(i,j), F(i,j)\}$$
$$E(i+1,j) = \max\{H(i,j) - g_o, E(i,j) - g_e, 0\}$$
$$F(i,j+1) = \max\{H(i,j) - g_o, F(i,j) - g_e, 0\}$$

(a) Banded Smith-Waterman

$$f^M(i,j) = \rho \cdot (\alpha_{mm} f^M(i-1,j-1) + \alpha_{im} f^I(i-1,j-1) + \alpha_{dm} f^D(i-1,j-1))$$
$$f^I(i,j) = \alpha_{mi} f^M(i-1,j) + \alpha_{ii} f^I(i-1,j)$$
$$f^D(i,j) = \alpha_{md} f^M(i,j-1) + \alpha_{dd} f^D(i,j-1)$$
$$Result = f^M(N_r,N_n) + f^D(N_r,N_n)$$

(b) Pairwise Hidden Markov Model

$$H(i,j) = \max\{H(p(i),q(j)) + S(X(i),Y(j)), H(p(i),j) + g(j), H(i,q(j)) + g(i)\}$$

(c) Partial Order Alignment

$$score(i) = \max\{\max_{i>j\geq 1}\{score(i) + weight(i,j)\}, w_j\}$$

(ii) Anchor dependency

$for\ i < j \leq i + N, do$
$score(j) = \max\{score(i) + weight(i,j), w_j\}$

(iii) Reordered anchor dependency

(d) Chain

Kernel	Application	Dimension and Size	Dependency	Data Type
bsw	Read Alignment	2D ~120×60	Last 2 Wave-fronts	Int 8/16

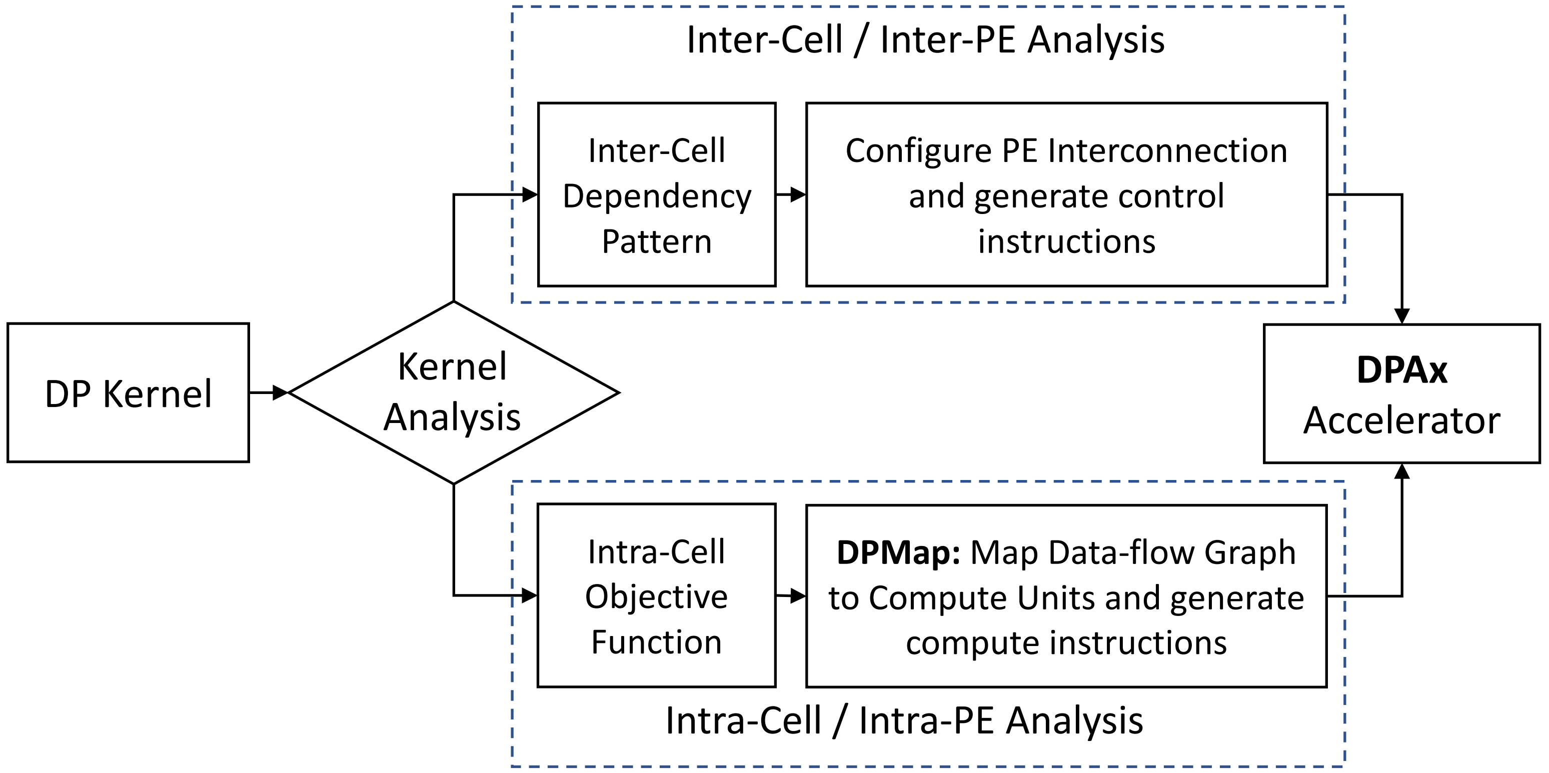
The divergence of dynamic programming kernels make it challenging to design a domain-specific accelerator

Similarity ⬇ Customization

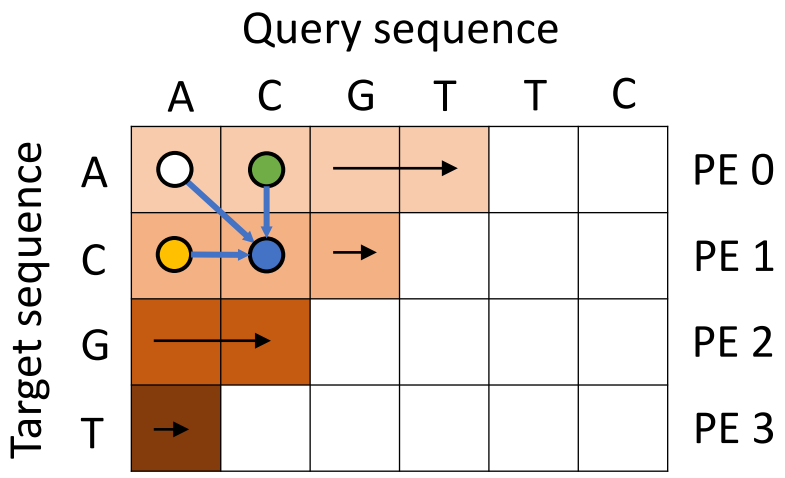
Difference ⬆ Programmability

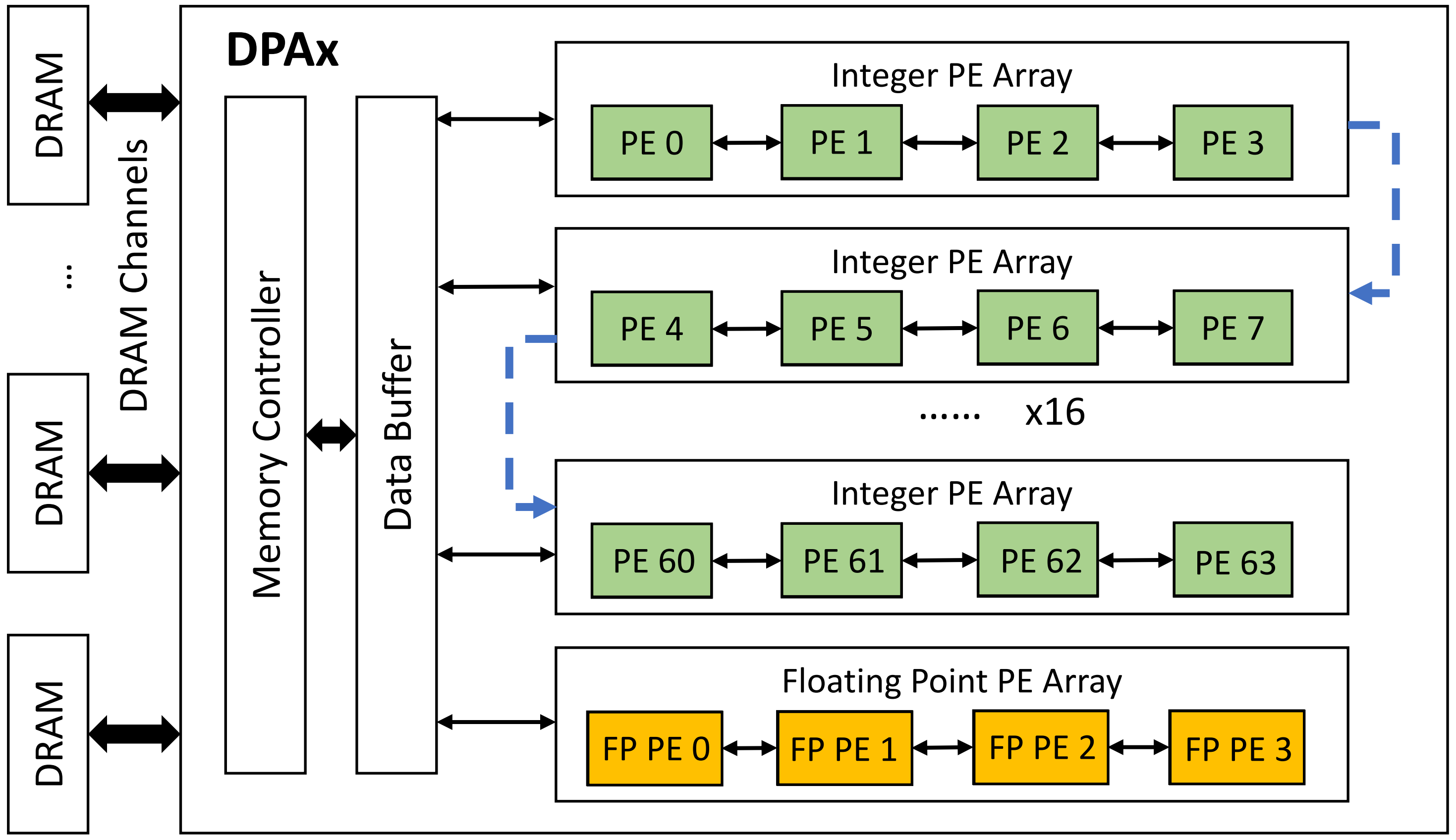
GenDP Framework

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

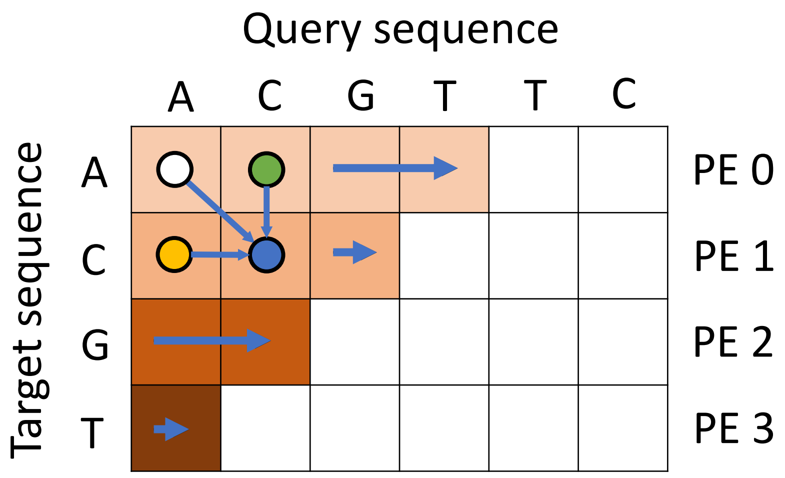

Inter-Cell / Inter-PE Analysis
Inter-Cell Dependency Pattern
Configure PE Interconnection and generate control instructions
DP Kernel
Kernel Analysis
DPAx Accelerator
Intra-Cell Objective Function
DPMap: Map Data-flow Graph to Compute Units and generate compute instructions
Intra-Cell / Intra-PE Analysis

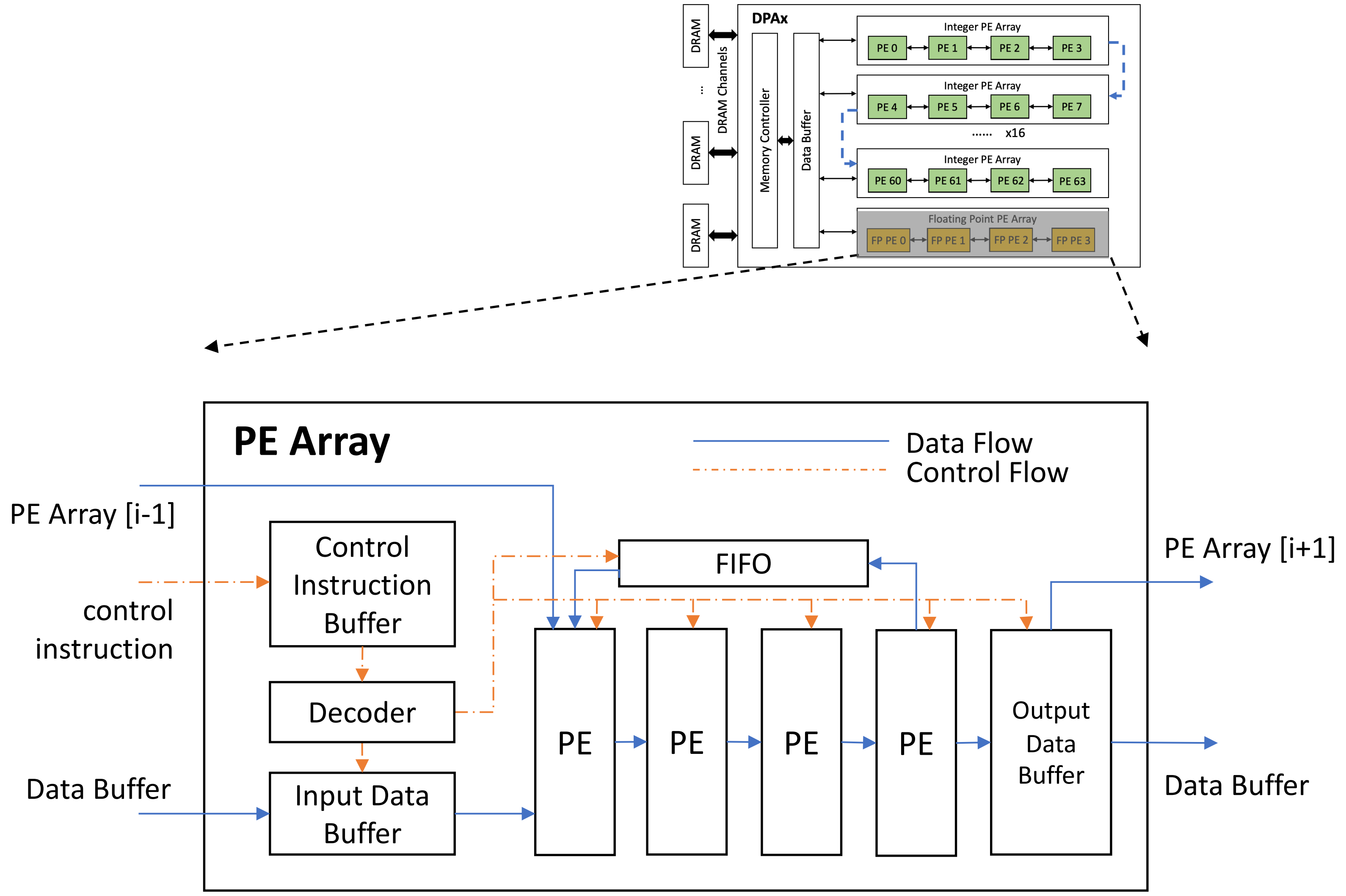
DPAx Accelerator



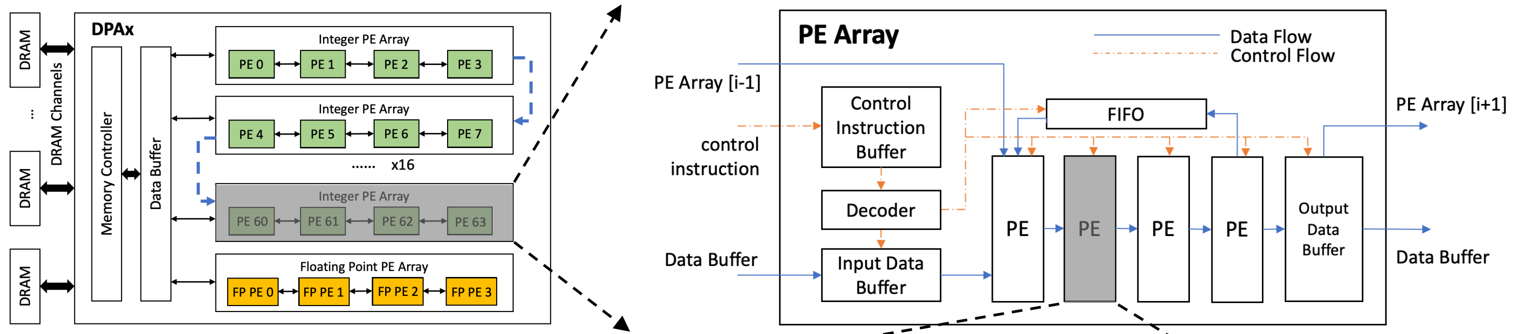


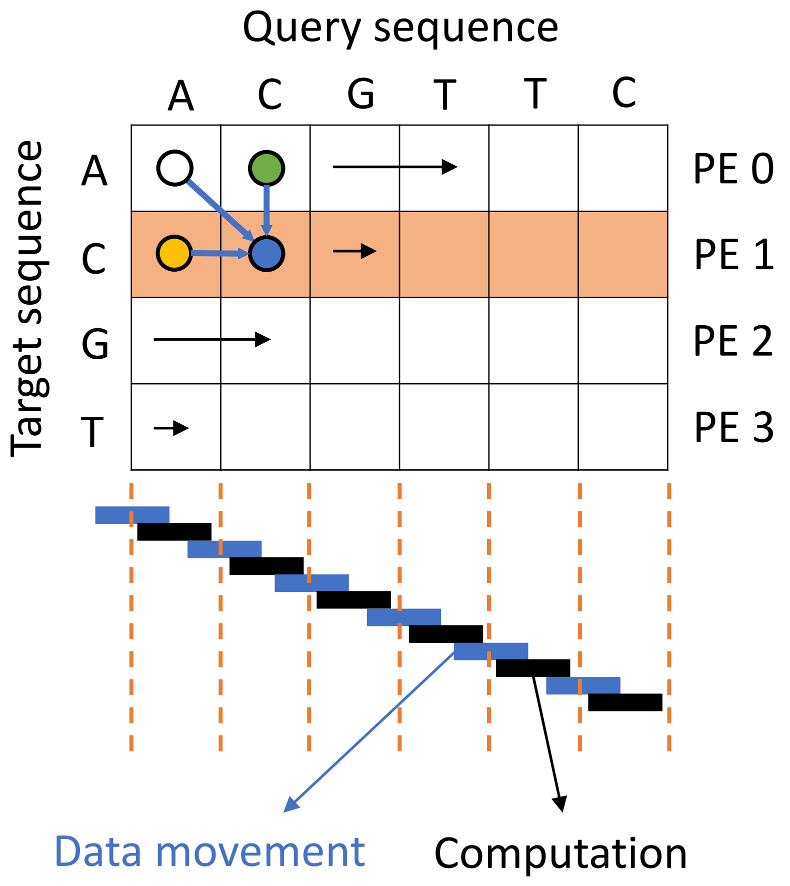
PE Array Architecture

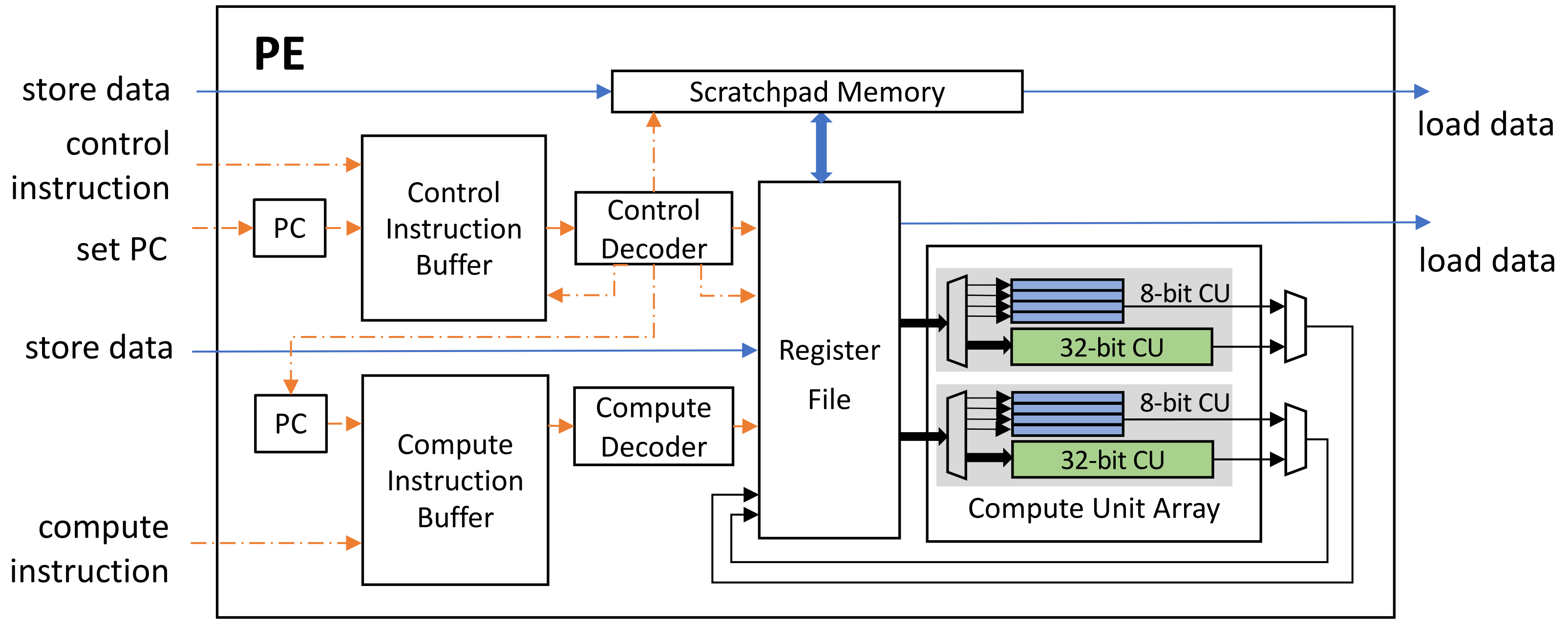




PE Architecture

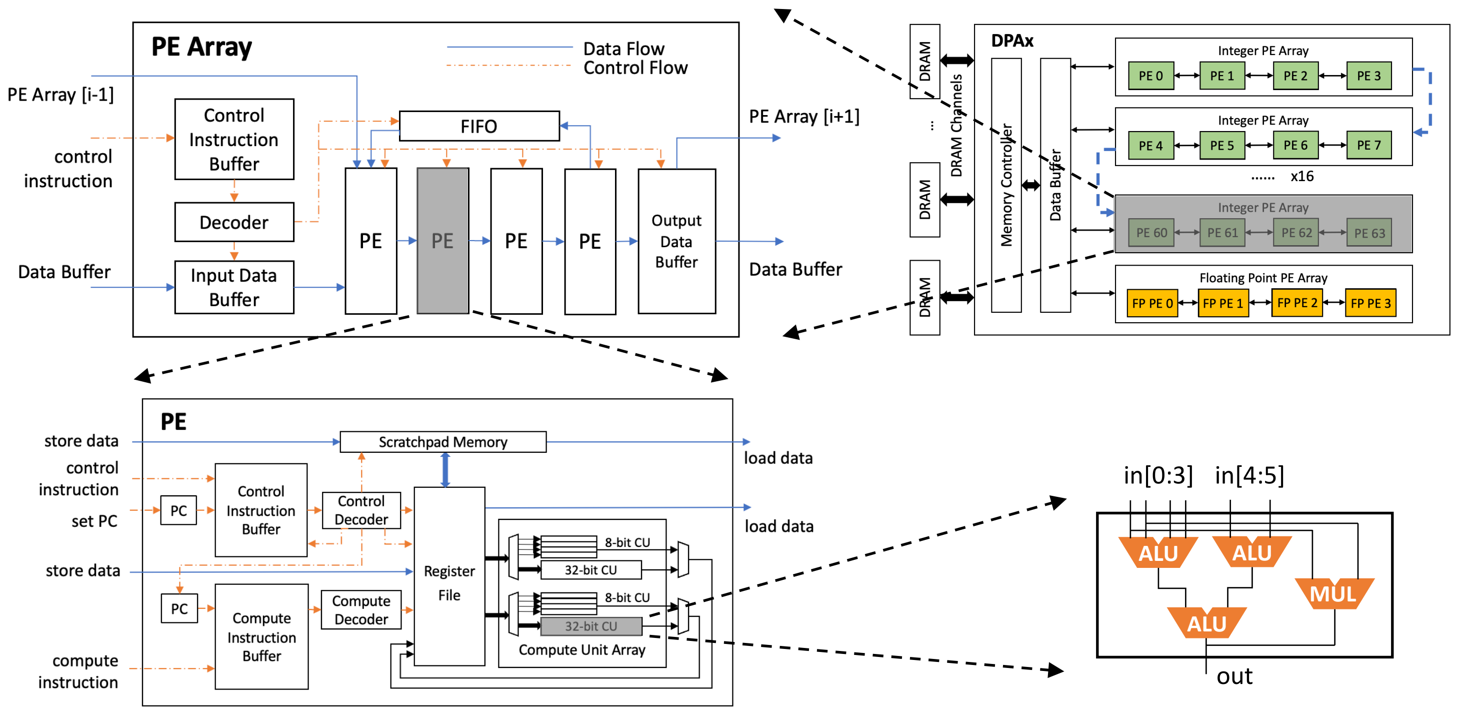


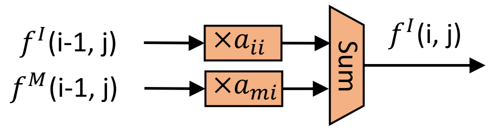




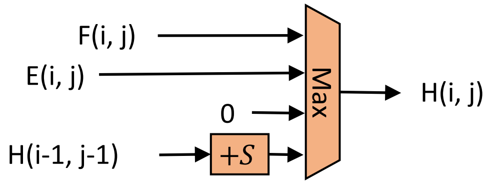
Design Choice Take Away

Similarity	➢ Local dependency	✓ 1-Dimension systolic PE array with FIFO
	➢ Reduction tree data path	✓ Compute unit – 2-level reduction tree
Difference	➢ Precision requirement	✓ 16 Integer PE array (SIMD compute unit) and 1 FP PE array
	➢ Dependency patterns	✓ PE arrays could execute separately or combined
	➢ Long dependency	✓ Software managed scratchpad memory
	➢ Objective func. and datapath	✓ Custom ISA for control and computation





(a) Reduction Data-Flow in PairHMM

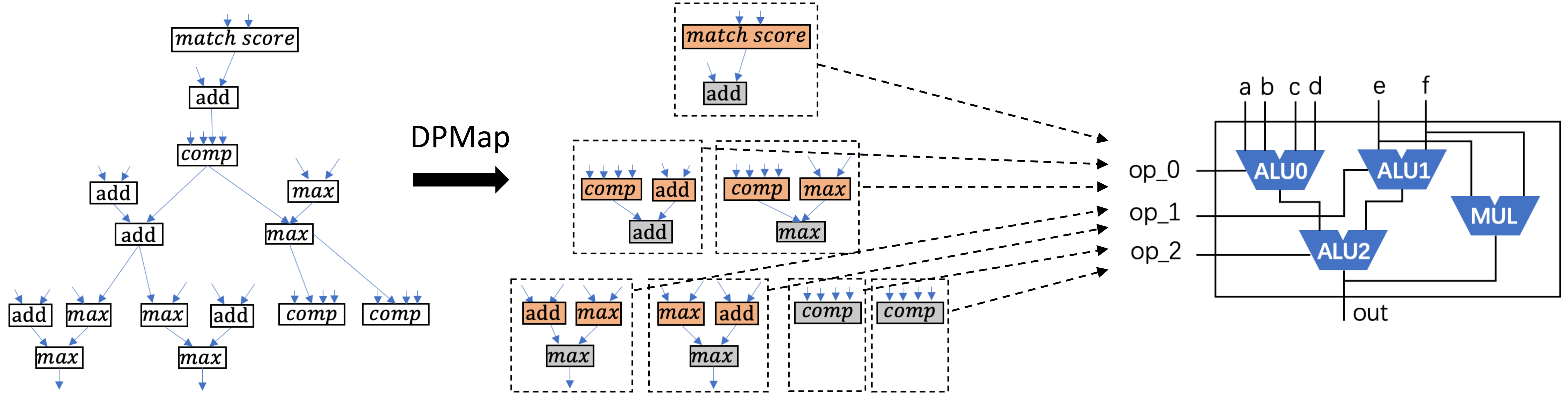


(b) Reduction Data-Flow in BSW

DPMap Algorithm

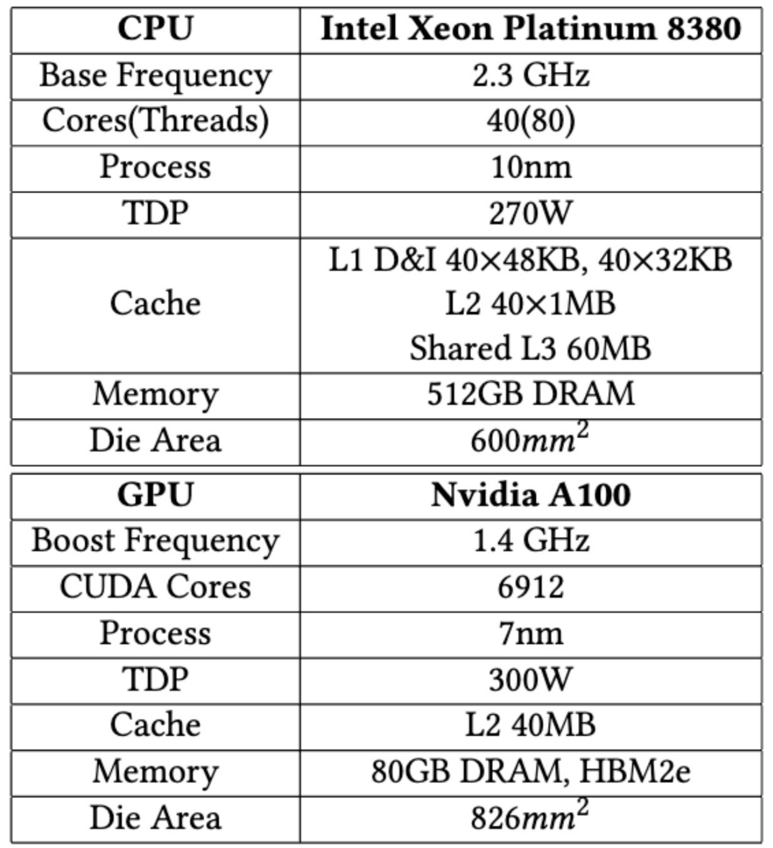
- **DPMap**: map the objective function of DP algorithm to programmable compute units in DPAx.
    - **Partitioning**: Break the data-flow graph with 4-input ALU and Multiplier
    - **Seeding**: Look for vertices that could be mapped to the 2nd level
    - **Refinement**: Break the single-strand structure

More details in the paper



Methodology

- Evaluate 4 representative genomics DP kernels
  - Banded Smith-Waterman (BSW)
  - Pairwise Hidden Markov Model (PairHMM)
  - Partial Order Alignment (POA)
  - Chain
- CPU Baseline
  - Intel Xeon Platinum 8380
  - SIMD optimization with AVX512 and 80 threads
- GPU Baseline
  - NVIDIA A100
- GenDP
  - Obtain throughput with an in-house cycle-accurate simulator.
  - Synthesis with a TSMC 28nm process to estimate the area and power of DPAx.

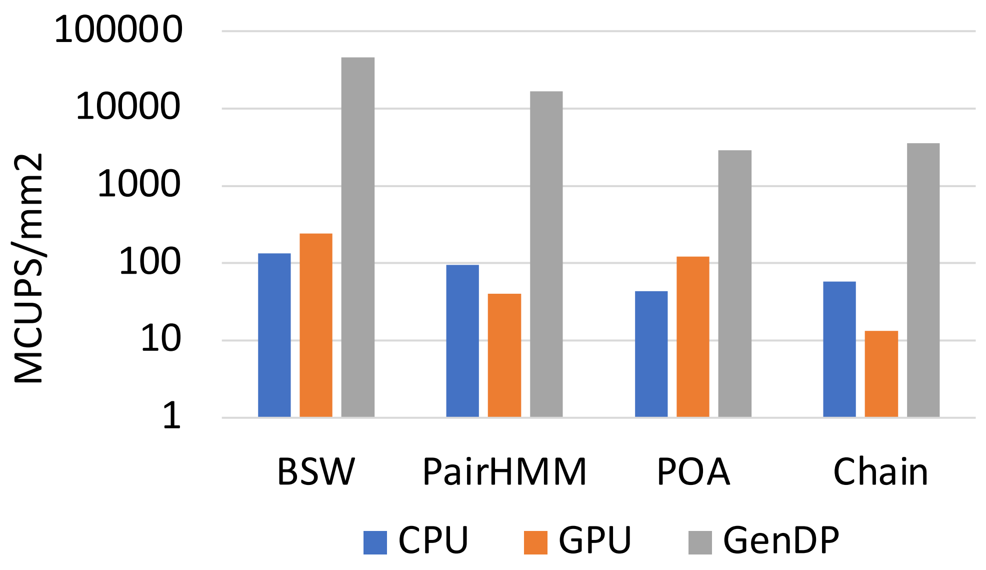


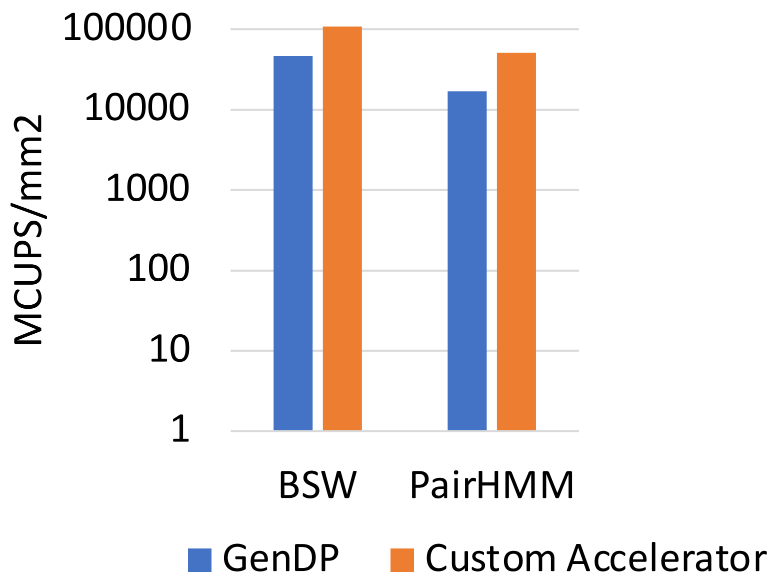
CPU	**Intel Xeon Platinum 8380**
Base Frequency	2.3 GHz
Cores(Threads)	40(80)
Process	10nm
TDP	270W
Cache	L1 D&I 40×48KB, 40×32KB L2 40×1MB Shared L3 60MB
Memory	512GB DRAM
Die Area	600mm^2

GPU	**Nvidia A100**
Boost Frequency	1.4 GHz
CUDA Cores	6912
Process	7nm
TDP	300W
Cache	L2 40MB
Memory	80GB DRAM, HBM2e
Die Area	826mm^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 has 2.8x slowdown when compared to custom accelerators
- Generality on DP algorithms in other domains
  - Dynamic time warping – speech detection
  - Bellman-Ford – Robot motion planning





GenDP: A Framework of Dynamic Programming Acceleration for Genome Sequencing Analysis

Yufeng Gu, Arun Subramaniyan, Tim Dunn, Alireza Khadem, Kuan-Yu Chen, Somnath Paul, Md Vasimuddin, Sanchit Misra, David Blaauw, Satish Narayanasamy, Reetuparna Das

Open-source: https://github.com/Yufeng98/GenDP

Acknowledgement