Darwin: A Genomics Co-processor Provides up to 15,000 × acceleration on long read assembly

Yatish Turakhia Gill Bejerano William J. Dally

Stanford University

ASPLOS'18

João Sanches Ferreira

ETH Zürich

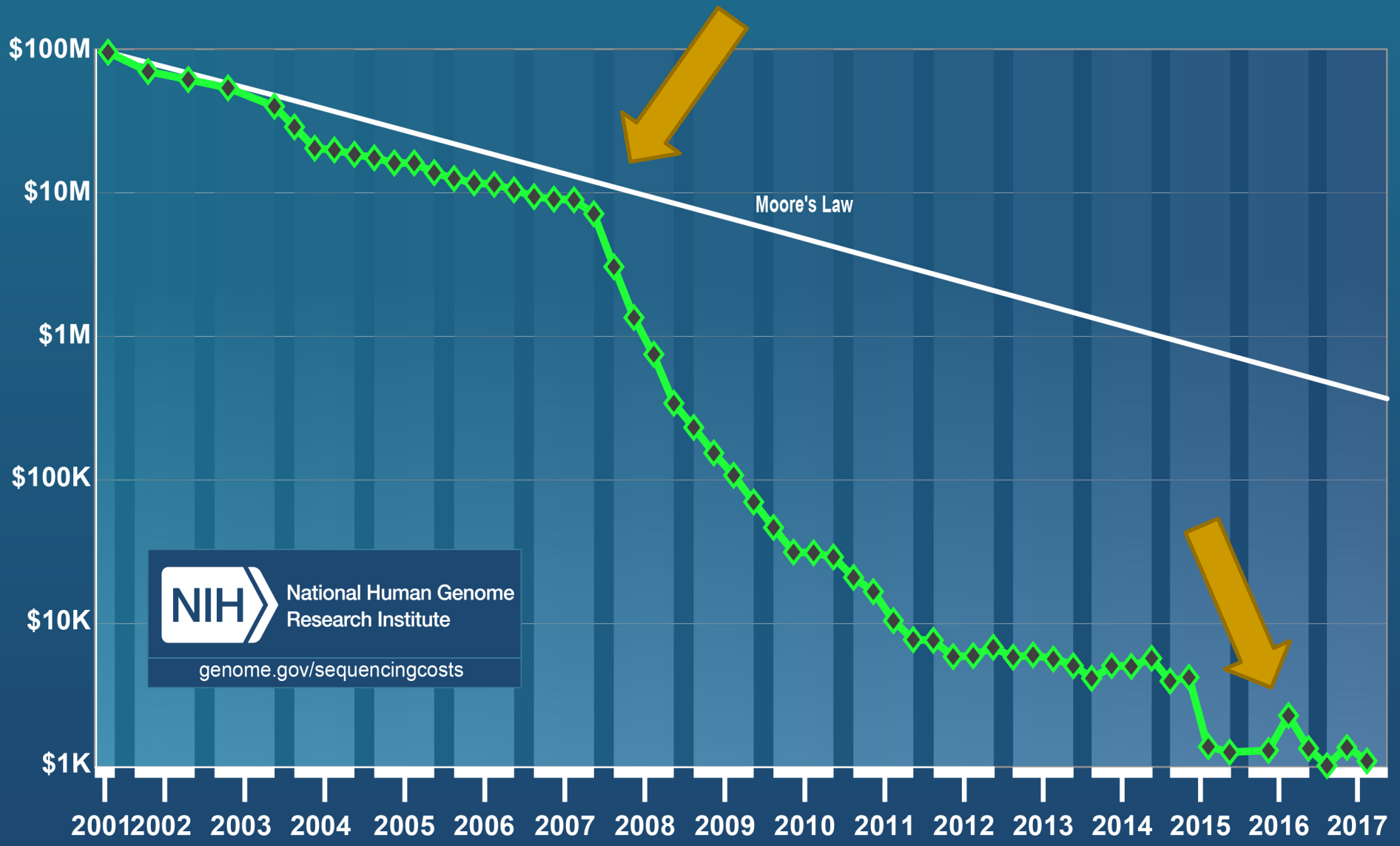
9 May 2019

Executive Summary

- **Motivation:** DNA sequencing technological improvements have resulted in longer reads, which results in higher quality genome assembly.

Problem

Cost per Genome

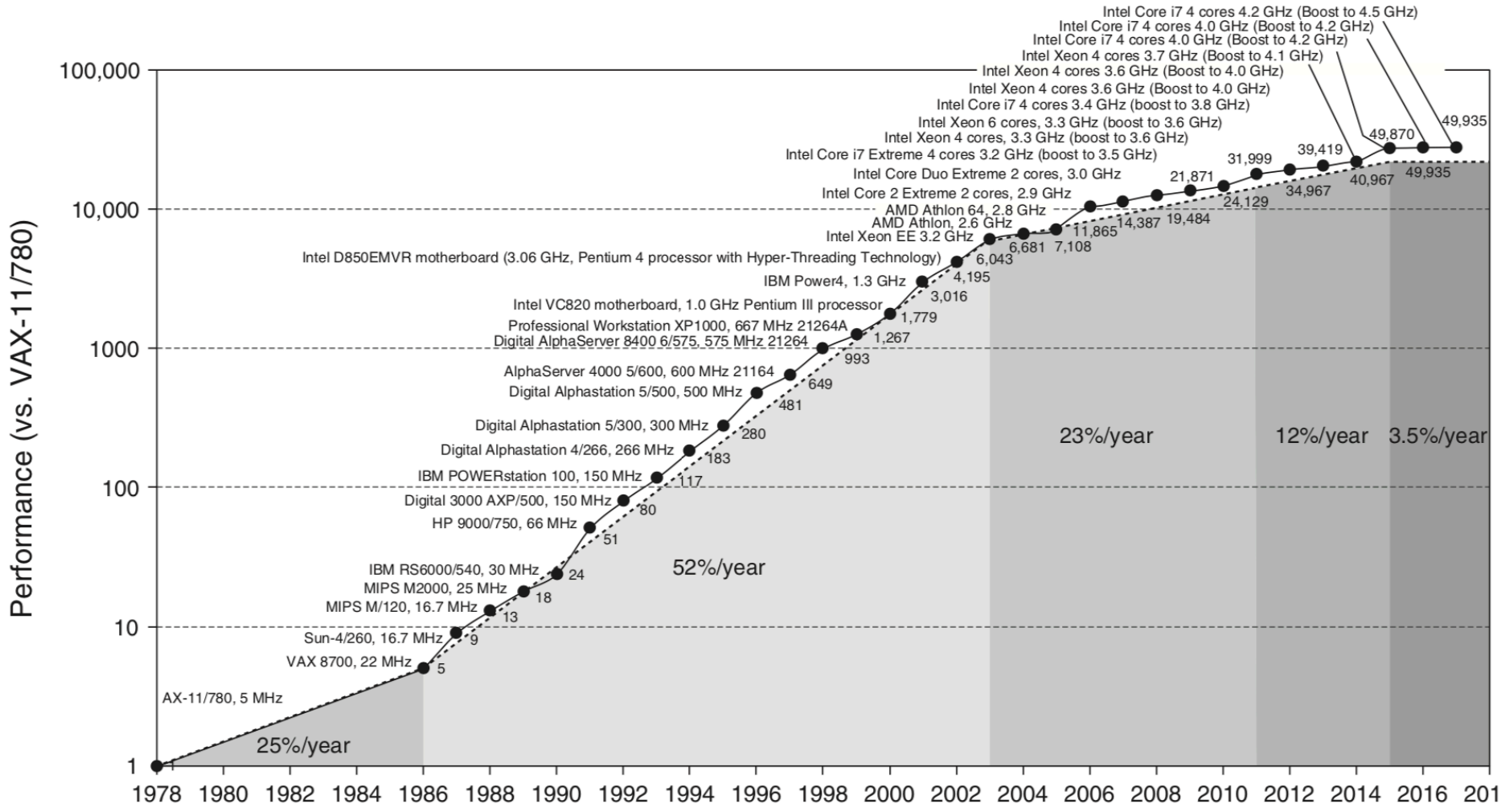


[1] Wetterstrand KA. DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP) Available at: www.genome.gov/sequencingcostsdata. Accessed 04/05/2019.

Comparison of genomic sequencing generations

	First generation	Second generation	Third generation
Fundamental technology	Size-separation of specifically end-labeled DNA fragments	Wash-and-scan SBS	Single molecule real time sequencing
Resolution	Averaged across many copies of the DNA molecule	Averaged across many copies of the DNA molecule	Single DNA molecule
Current raw read accuracy	High	High	Lower
Current read length	Moderate (800-1000 bp)	Short (generally much shorter than Sanger sequencing)	> 1000 bp
Current throughput	Low	High	High
Current cost	High cost per base, Low cost per run	Low cost per base, High cost per run	Low cost per base, High cost per run
RNA-sequencing method	cDNA sequencing	cDNA sequencing	Direct RNA sequencing
Time to result	Hours	Days	< 1 day
Sample preparation	Moderately complex, PCR amplification is not required	Complex, PCR amplification is required	Various
Data analysis	Routine	Complex (due to large data volumes & short reads)	Complex
Primary results	Base calls with quality values	Base calls with quality values	Base calls with quality values

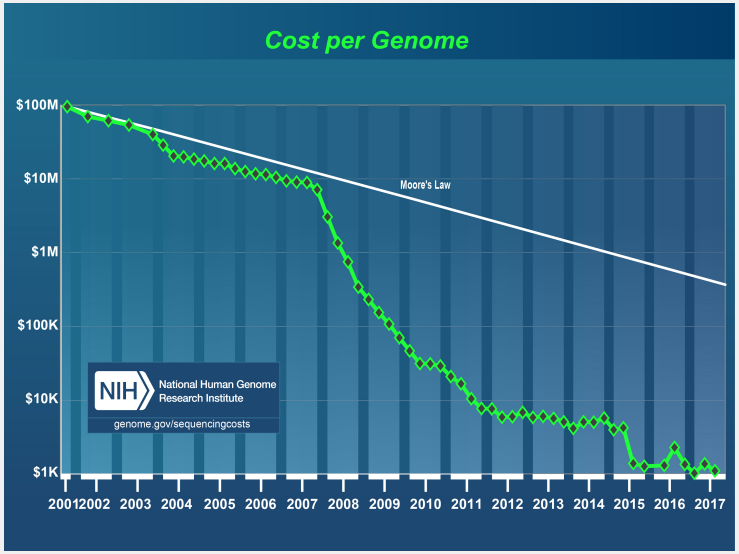
Adapted from Schadt, et al. Hum Mol Genet 2010[13]

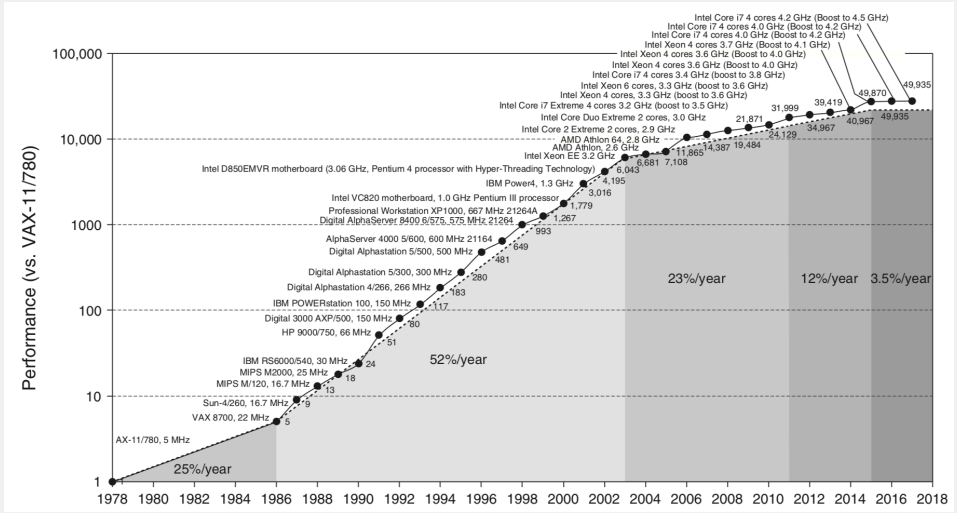


[2] Hennessy, John L. *Computer Architecture: A Quantitative Approach*. Sixth edition, Morgan Kaufmann Publishers, 2019.

Executive Summary

- **Motivation:** DNA sequencing technological improvements have resulted in longer reads, which results in higher quality genome assembly.
- **Problem:** Genomic sequencing technology is scaling, compute performance isn't.





Executive Summary

- **Motivation:** DNA sequencing technological improvements have resulted in longer reads, which results in higher quality genome assembly.
- **Problem:** Genomic sequencing technology is scaling, compute performance isn't.
- **Goal:** Introduce a co-processor to accelerate genomic sequence alignment – **Darwin**.

Executive Summary

- **Motivation:** DNA sequencing technological improvements have resulted in longer reads, which results in higher quality genome assembly.
- **Problem:** Genomic sequencing technology is scaling, compute performance isn't.
- **Goal:** Introduce a co-processor to accelerate genomic sequence alignment – **Darwin**.
- **Solution:** Co-design algorithms and hardware targeted at long (3rd-gen) read assembly.

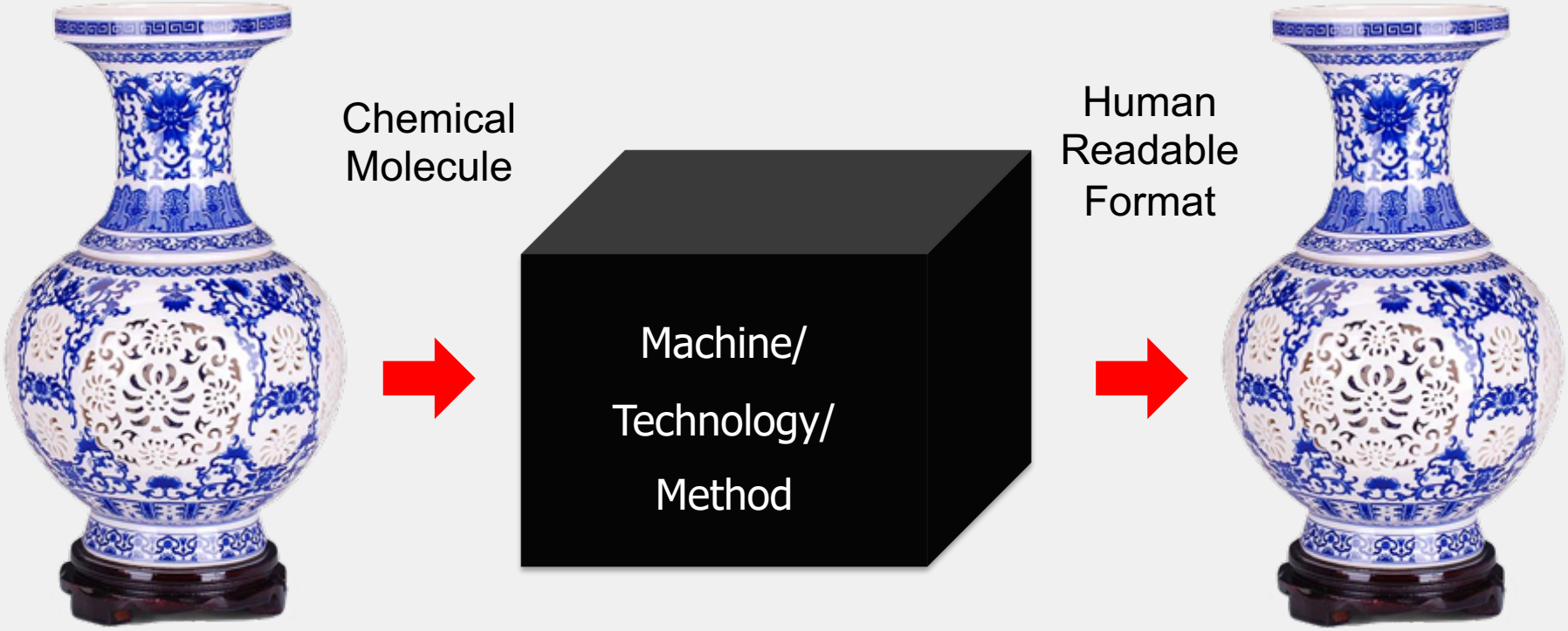
Executive Summary

- **Motivation:** DNA sequencing technological improvements have resulted in longer reads, which results in higher quality genome assembly.
- **Problem:** Genomic sequencing technology is scaling, compute performance isn't.
- **Goal:** Introduce a co-processor to accelerate genomic sequence alignment – **Darwin**.
- **Solution:** Co-design algorithms and hardware targeted at long (3rd-gen) read assembly.
- **Evaluation:**
  - 3-4 orders of magnitude faster reference-guided assembly
  - 2 orders of magnitude faster *de novo* assembly

Background

DNA Sequencing

- **Goal:** Find the complete sequence of A, C, G, T's in DNA.



- **Challenge:** There is no machine/technology/method that takes long DNA as an input, and gives the complete sequence as output.

DNA Sequencing

All sequencing machines chop DNA into pieces and identify relatively small pieces (but not how they fit together).

DNA Sequencing

short reads

- 50-300 bp
- low error rate (~0.1%)

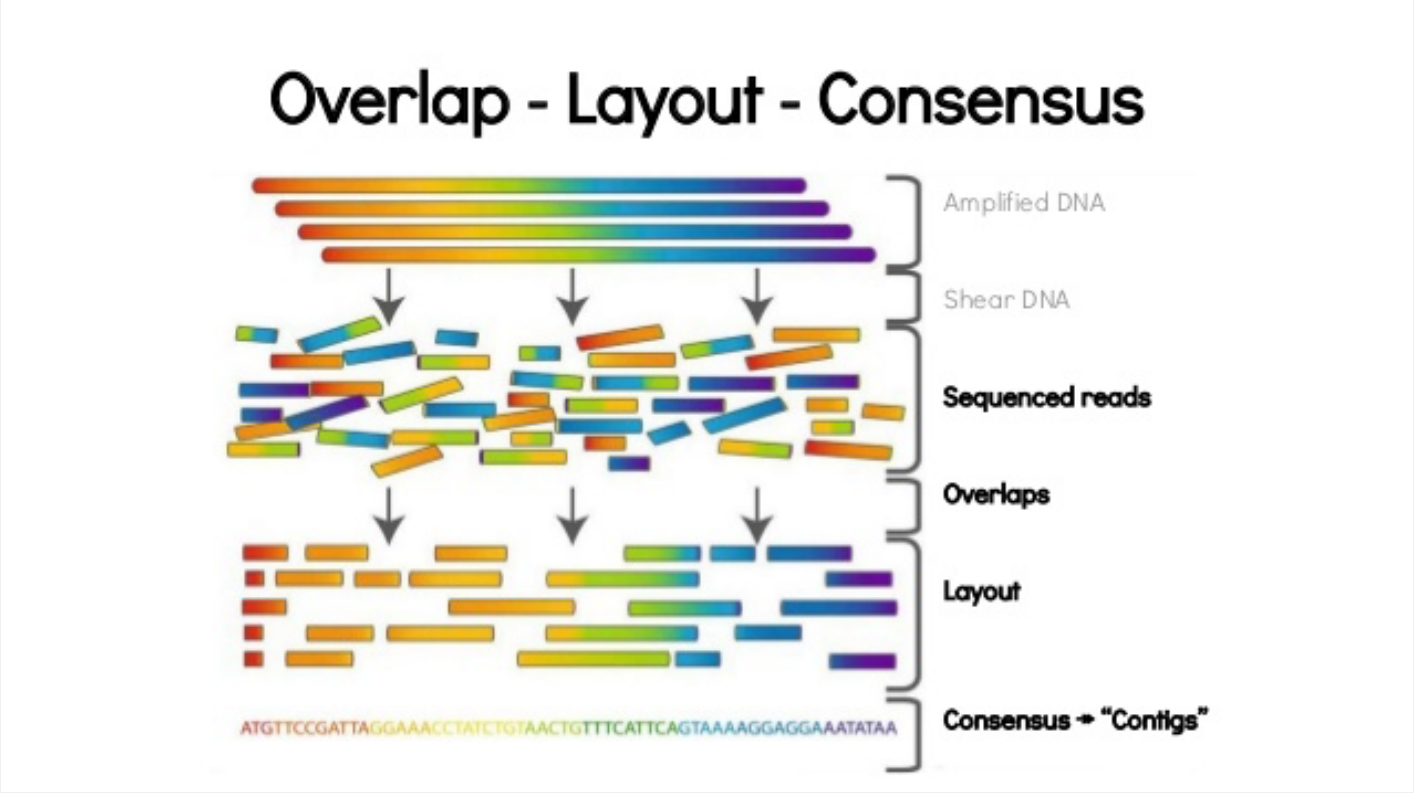
long reads

- 10K-100K bp
- high error rate (~15%)

Size of human genome: **3.2 Billion bp**

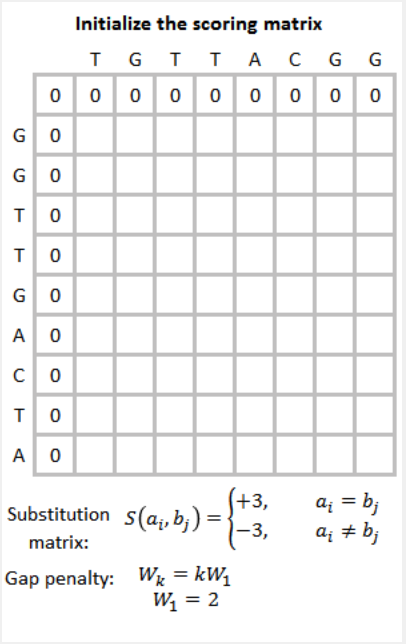
DNA Sequence Alignment

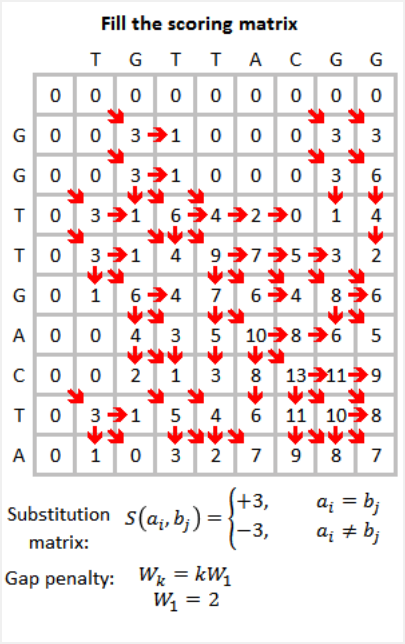
- Compare a query sequence Q and a reference sequence R, to maximize an alignment score.
  - Identify insertions, deletions or mismatches.

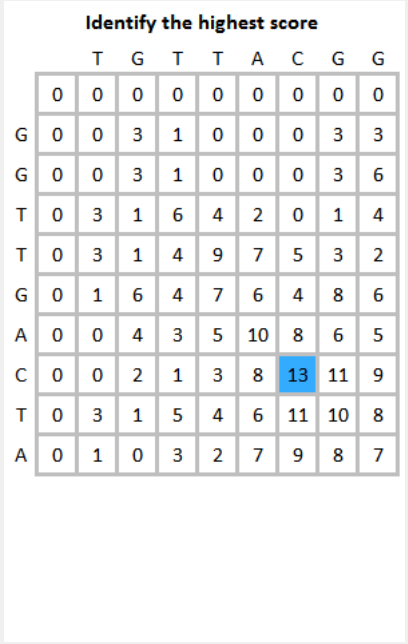


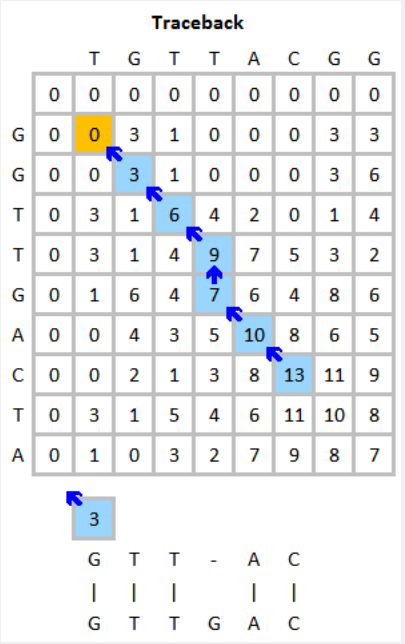
Alignment Algorithms

- **Smith-Waterman algorithm**
  - Identifies similar regions between two input sequences
  - Compares segments of all possible lengths
  - Ensures optimal local alignment





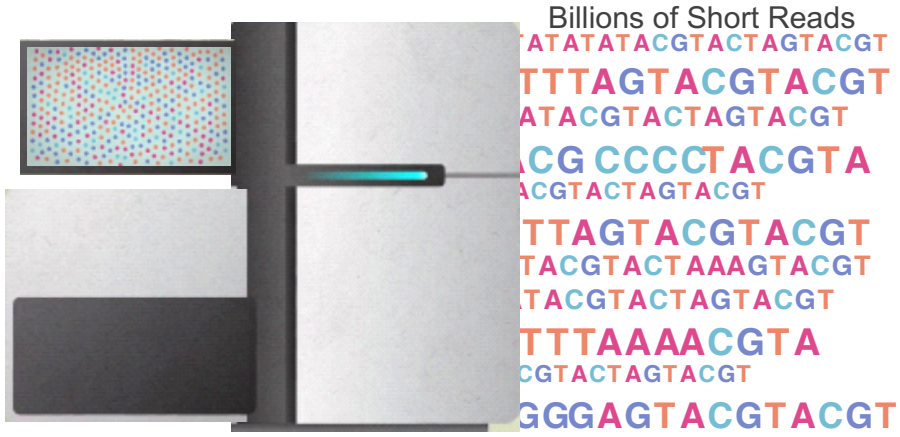




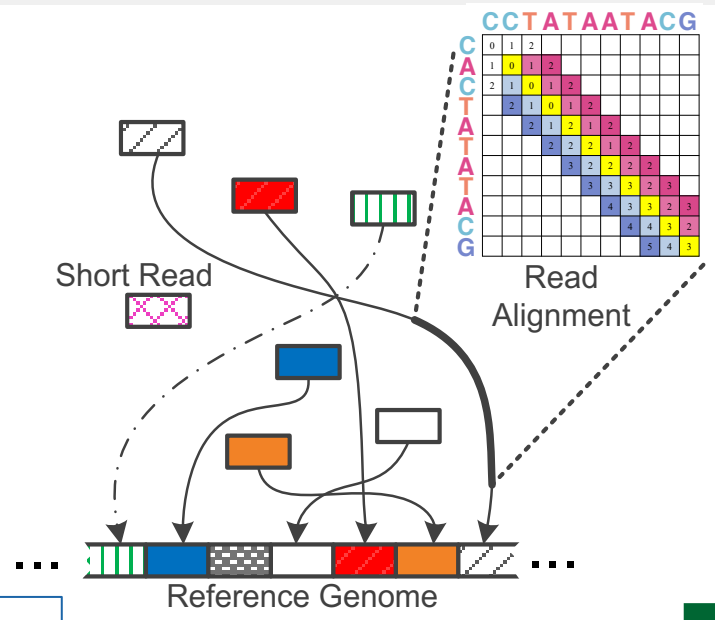
Filtering Algorithms

- **Problem**
  - Smith-Waterman (and similar algorithms) are computationally expensive.

- **Solution**
  - Use filtering step based on seed-and-extend paradigm.
  - This approach uses seeds, substrings of fixed size k from Q, and finds their exact matches in R, called seed hits.



1 Sequencing



Read Mapping 2

Genome Analysis

```
reference: TTTATCGCTTCCATGACGCAG
read1:       ATCGCATCC
read2:      TATCGCATC
read3:           CATCCATGA
read4:         CGCTTCCAT
read5:               CCATGACGC
read6:             TTCCATGAC
```

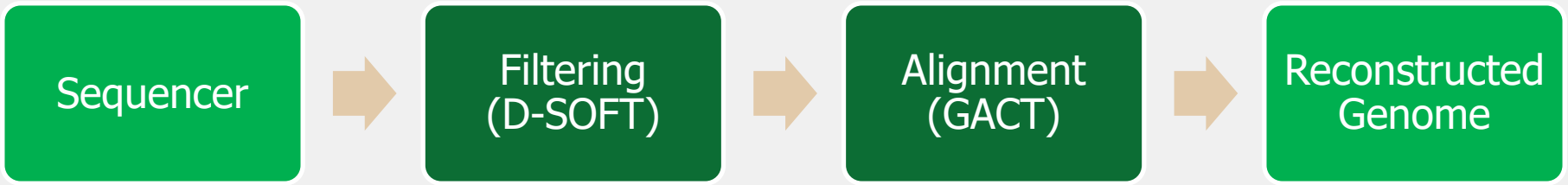
3 Variant Calling

Scientific Discovery 4

Novelty

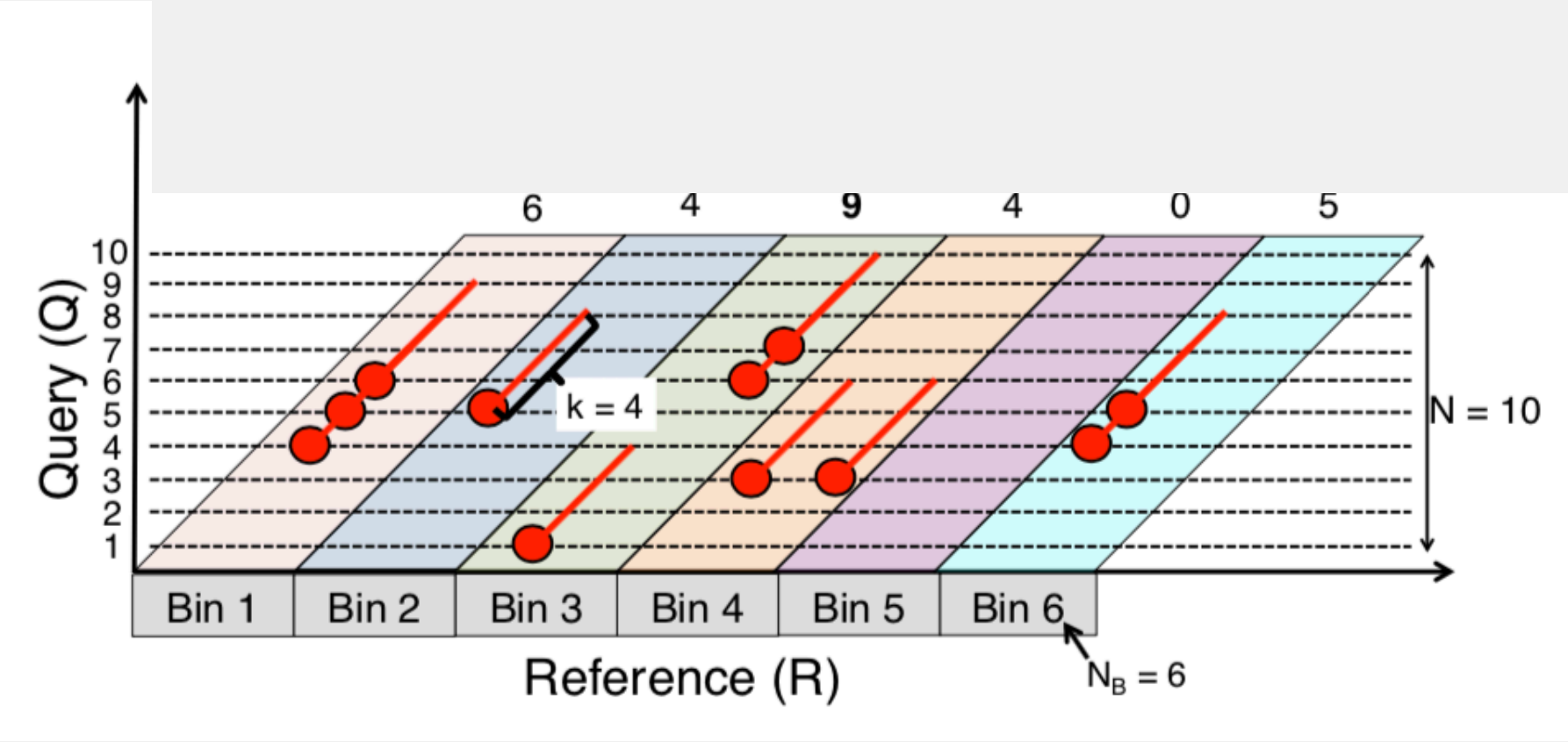
Novelty

- **D-SOFT –** Filtering algorithm
  - Tunable sensitivity (tolerance to inexact matches).
  - High precision.
- **GACT –** Alignment algorithm
  - Arbitrarily long sequences, with optimal alignment for error rates of up to 40%.
  - Constant memory for the compute-intensive step.
- **Darwin implementation**
  - FPGA.
  - ASIC (simulated, by scaling up frequency).



Mechanisms

Mechanisms – D-SOFT



Mechanisms – SeedLookup

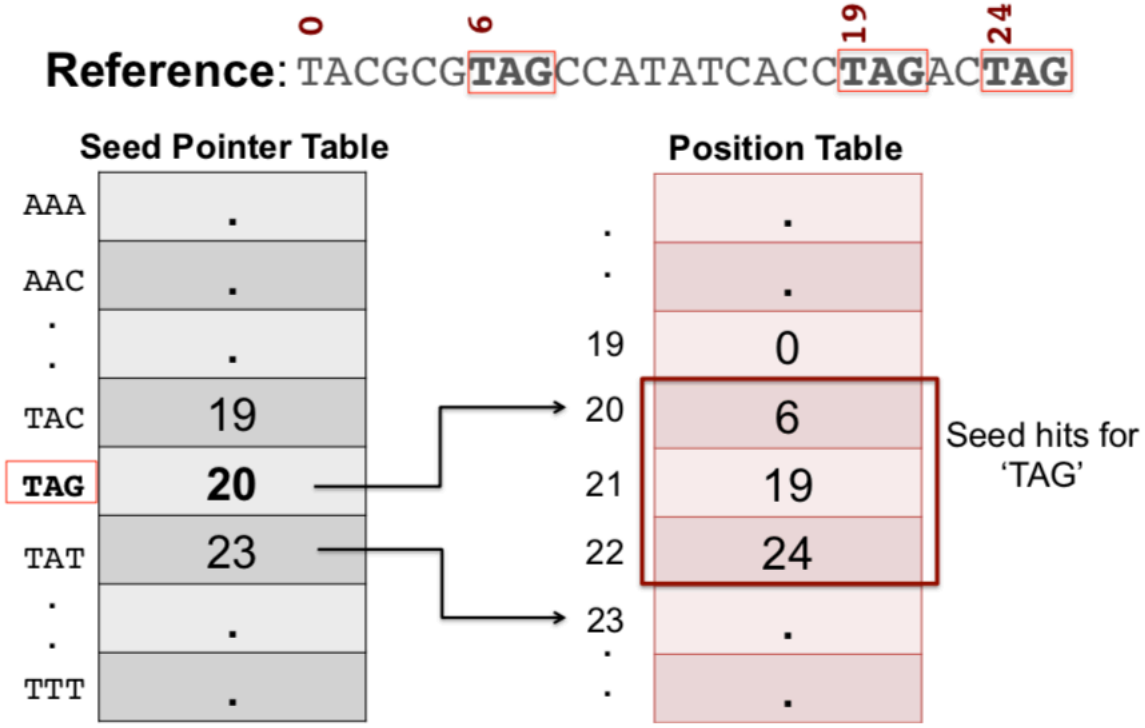
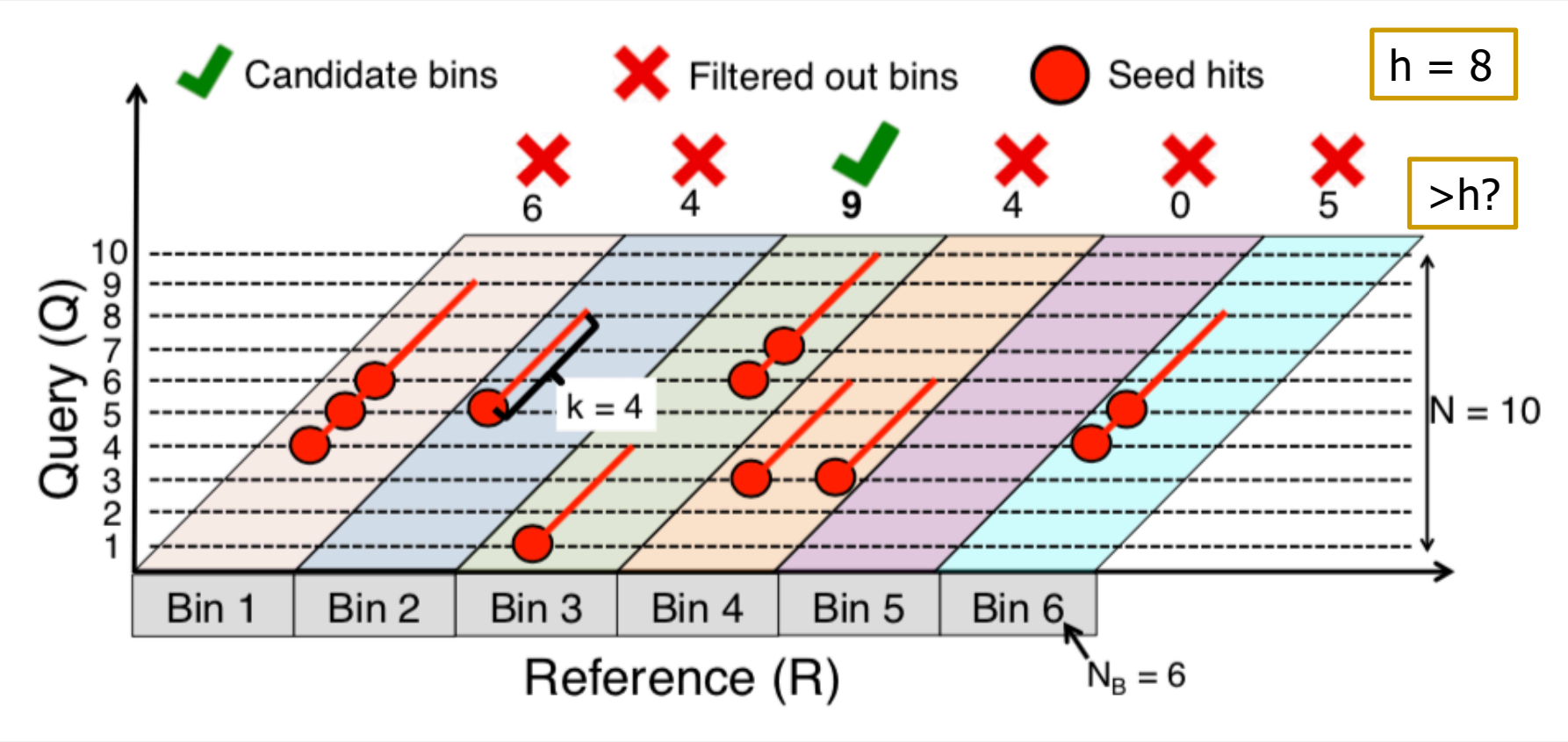
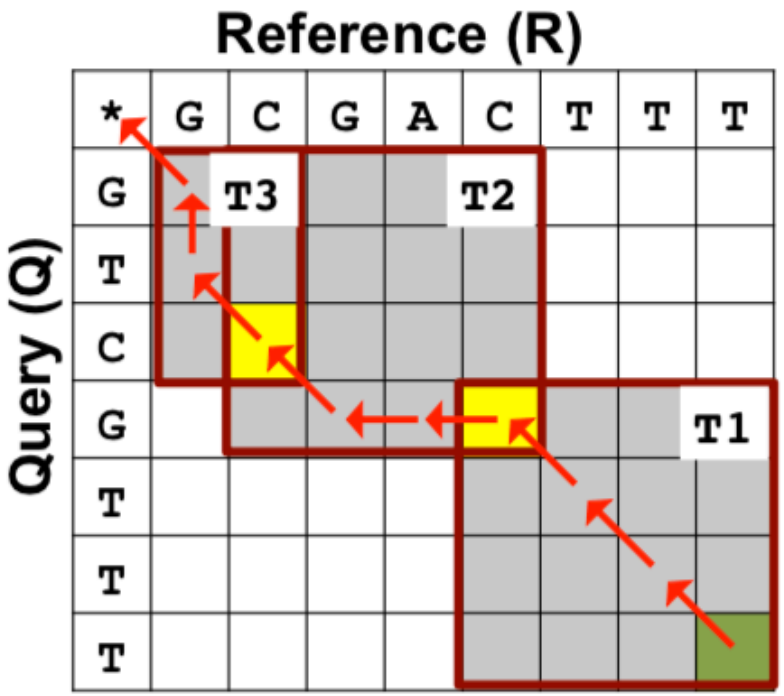


Figure 3: An example reference sequence and seed position table used in *SeedLookup*.

Mechanisms – D-SOFT



Mechanisms – GACT



T1

	C	T	T	T
G	0	0	0	0
T	0	2	2	2
T	0	2	4	4
T	0	2	4	6

(i_{off}=3, j_{off}=3)

Key Results: Methodology and Evaluation

Key Results

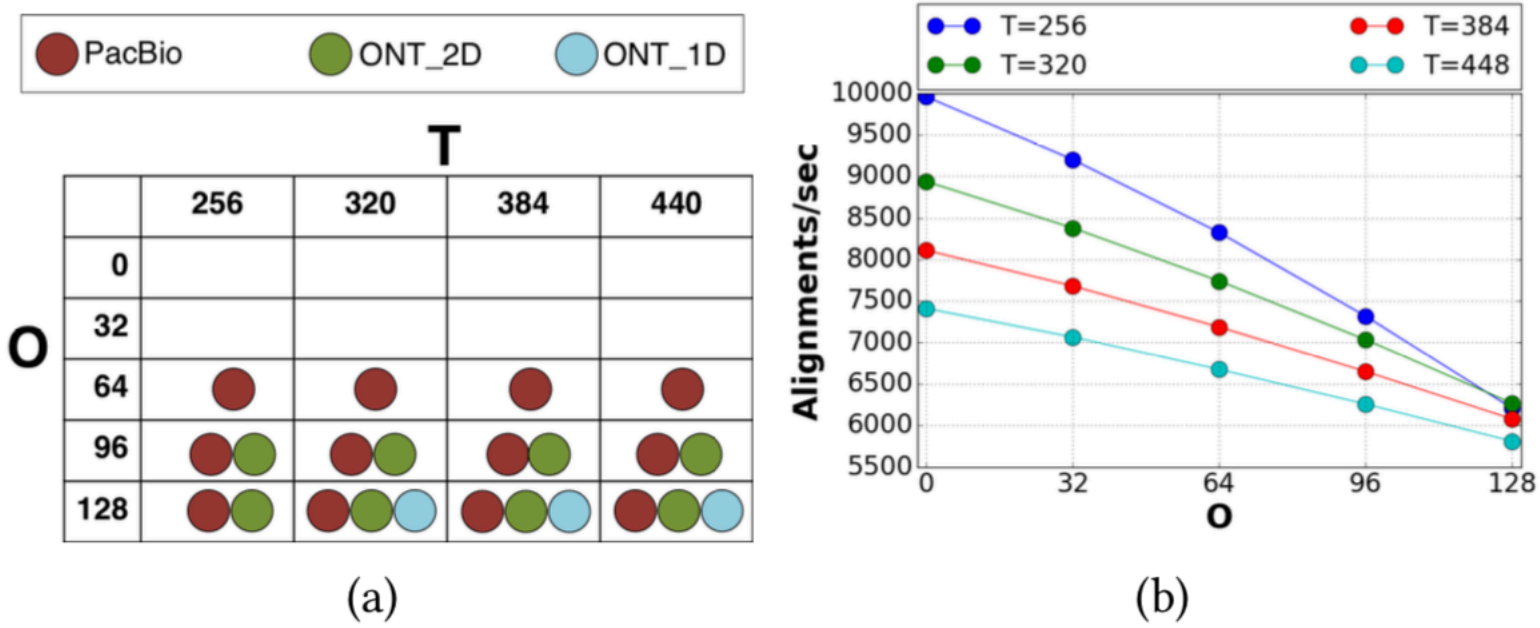


Figure 9: (a) (T, O) settings of GACT for different read types for which all 200,000 observed alignments were optimal. (b) Throughput of a single GACT array for pairwise alignment of 10Kbp sequences for different (T, O) settings.

Key Results

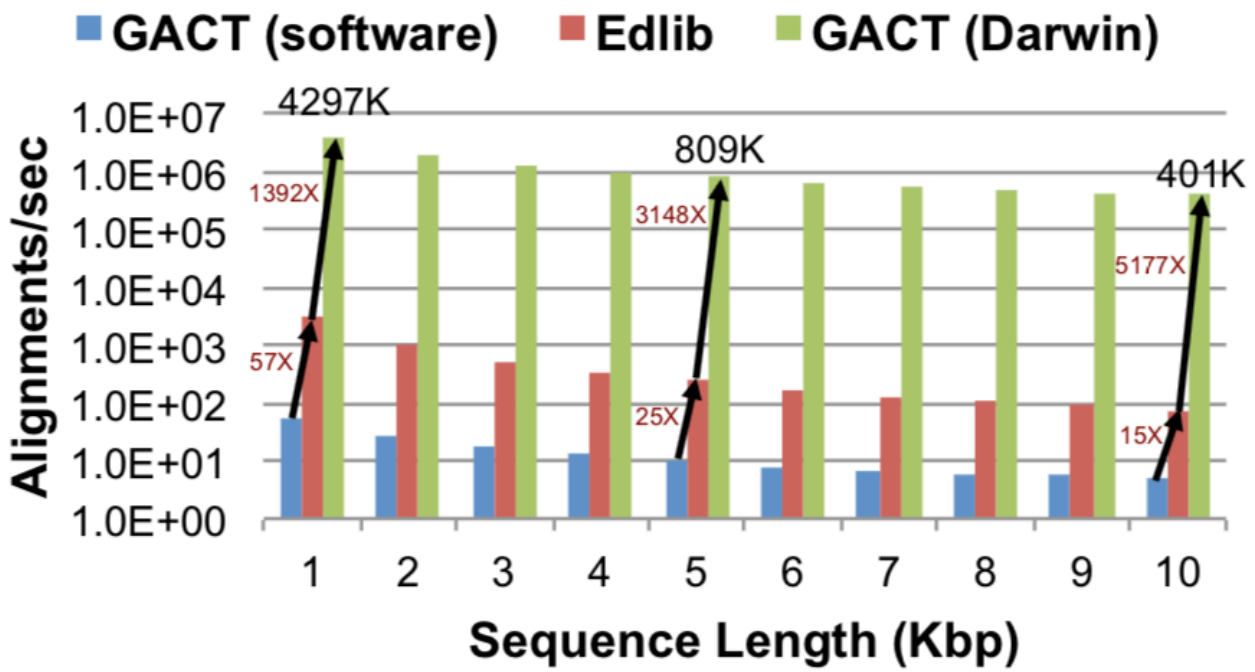


Figure 10: Throughput (alignments/second) comparison for different sequence lengths between a software implementation of GACT, Edlib library and the hardware-acceleration of GACT in Darwin.

Key Results

Size (k)	hits/seed (GRCh38)	Throughput (Kseeds/sec)		
		Software	Darwin	Speedup
11	1866.1	16.6	1,426.9	85×
12	491.6	66.2	5,422.6	82×
13	127.3	259.3	19,081.7	73×
14	33.4	869.5	55,189.2	63×
15	8.7	2,257.1	91,138.7	40×

Table 3: Average number of seed hits for different seed sizes (k) and throughput comparison of D-SOFT on Darwin and its software implementation using human genome (GRCh38) for seed position table. 45% of available memory cycles in Darwin are reserved for GACT.

Key Results

Reference-guided assembly (human)							
Read type	D-SOFT	Sensitivity		Precision		Reads/sec	
	(k, N, h)	Baseline	Darwin	Baseline	Darwin	Baseline	Darwin
PacBio	(14, 750, 24)	95.95%	+3.76%	95.95%	+3.96%	3.71	9,916×
ONT_2D	(12, 1000, 25)	98.11%	+0.09%	99.10%	+0.22%	0.18	15,062×
ONT_1D	(11, 1300, 22)	97.10%	+0.30%	98.20%	+0.72%	0.18	1,244×
De novo assembly (*C. elegans*)							
Read type	D-SOFT	Sensitivity		Precision		Runtime (sec)	
	(k, N, h)	Baseline	Darwin	Baseline	Darwin	Baseline	Darwin (Speedup)
PacBio	(14, 1300, 24)	99.80%	+0.09%	88.30%	+1.80%	47,524	123×
De novo assembly (human)							
Read type	D-SOFT	Estimated Runtime (hours)					
	(k, N, h)	Baseline		Darwin (Speedup)			
PacBio	(14, 1300, 24)	15,600		710×			

Table 4: Comparison of Darwin with baseline techniques on reference-guided and *de novo* assembly. Darwin values are relative to the baseline technique.

Key Results

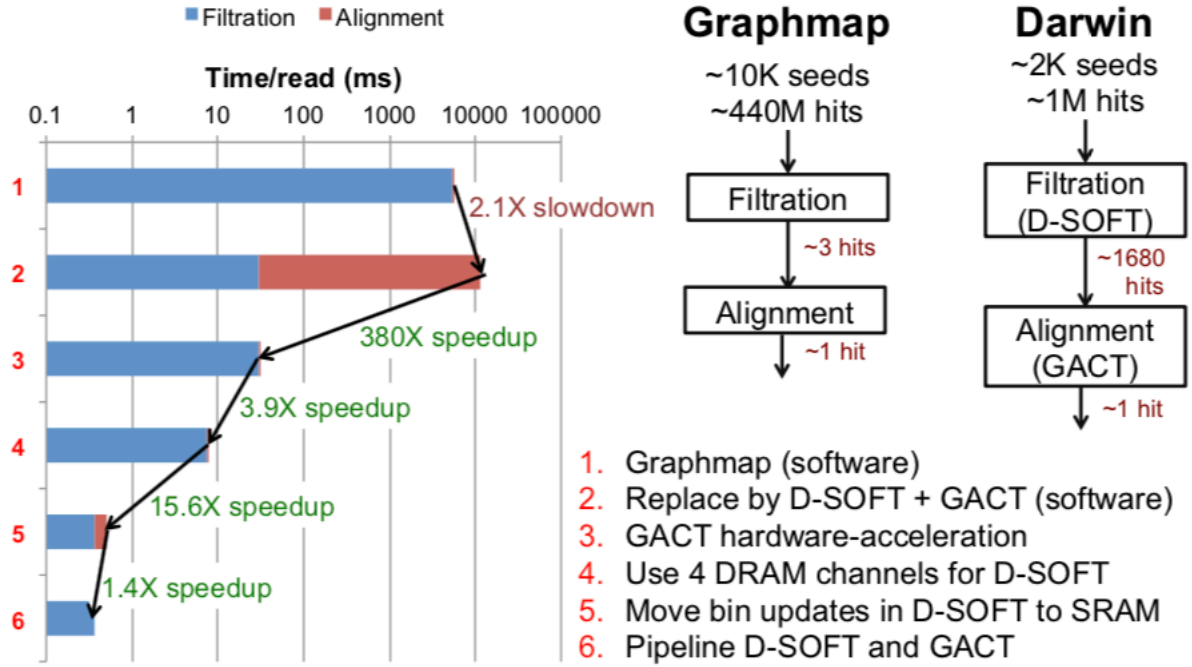


Figure 13: Timing breakdown between filtration and alignment stages for reference alignment of a single ONT_2D read in a series of steps going from Graphmap to Darwin.

Summary

Summary

- Volume of genomic data is rapidly increasing.
  - Need for efficient sequence alignment, unmet by present-day hardware.
- **Darwin** – a co-processor for genomic sequence alignment that combines hardware-accelerated alignment (GACT) and filtering (D-SOFT) algorithms.
  - **D-SOFT**
    - Tunable sensitivity.
  - **GACT**
    - Can process arbitrarily long sequences
    - Requires constant memory for the compute-intensive step.
- **2-4** orders of magnitude improvement in sequencing performance, compared to baseline.

Strengths

Strengths

- HW/SW co-design
  - Memory system is optimized for filtering (which is more expensive than alignment).
    - 4 DRAM channels store identical copies of the seed position table.
    - Seed hits are stored sequentially.
- Filtering algorithm offers tunable sensitivity.
- Alignment algorithm is linear-time and constant-memory.
- Filtering and alignment can be used in other genomics applications:
  - Whole sequence alignments, metagenomics, multiple sequence alignments…

Weaknesses

Weaknesses

- Poor baseline
  - Baseline is single-threaded CPU
  - Hardware/Accelerator baseline is missing
    - Multi-threaded / PIM / GPU / FPGA…
    - Speedups are only given with reference to CPU, running a single-thread.
- Tiling is not novel – used typically in greed mapping.
  - Seems heuristic?
  - Guarantee of optimality?
- ASIC performance is only simulated by scaling up the frequency, with the FPGA version as a baseline.
- No direct comparison of D-SOFT / GACT with other filtering/alignment algorithms?

Thoughts and Ideas

Thoughts and Ideas

- What other algorithms can be modified to maximally exploit HW/SW codesign?
- What if we used different filters?
  - How would that affect sensitivity?
  - What's the sensitivity of this filtering? How does it respond to alignments that are not true alignment?
- Can we use PIM? Could we do (some?) of the computation in-memory, to avoid having to move data from the DRAM memory to the accelerators?

Takeaways

Takeaways

- Specialized hardware is increasingly important
  - Specialization gives efficiency, parallelization gives speedup.
  - Specialization may require changes to the algorithms
    - Case in point: GACT, D-SOFT.

- Memory access time dominates
  - Optimizing access patterns is critical for performance.
  - Computation in memory pays off.

- Previous points show the importance of **HW/SW co-design**.

Open Discussion

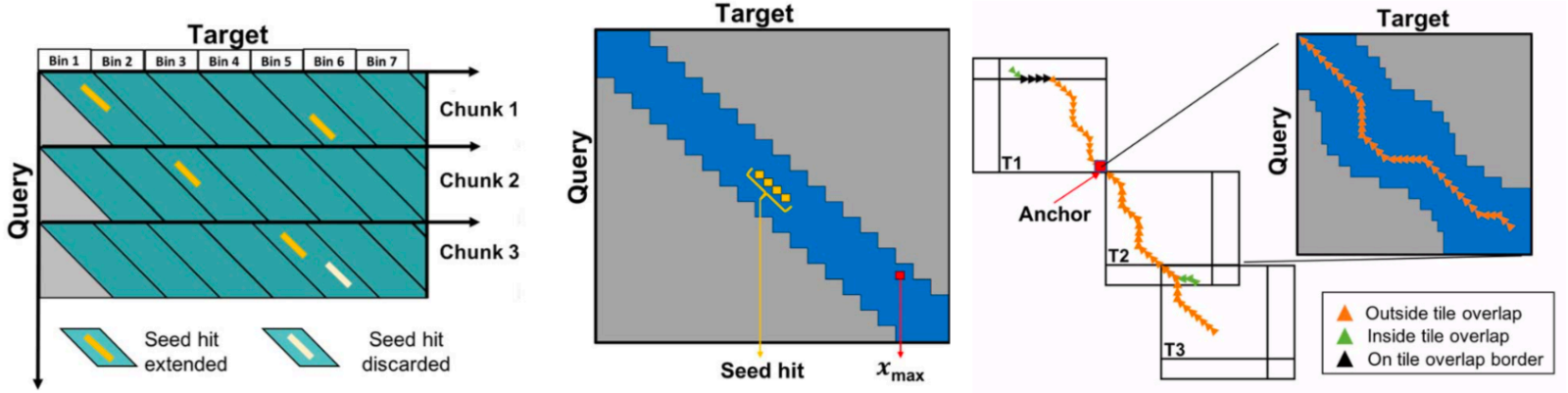
Discussion Starters

Could Darwin be used for Whole Genome Alignment?

whole genome alignment (WGA) refers to the computational process of aligning entire genome sequences of two or more species in order to study their evolutionary relationship. In particular, WGA helps in identifying set of sequences that are *orthologous* (diverged after a speciation event) or *paralogous* (diverged after a duplication event) [28]. As

Discussion

- What must be changed in Darwin to achieve WGA?



(a) D-SOFT filtering (b) Gapped filtering (c) GACT-X extension

Figure 4: Overview of Darwin-WGA (a) Target bins and query chunks constitute a diagonal band, at most 1 seed hit to be extended per diagonal band. (b) Tile for banded Smith-Waterman algorithm, blue represents the band calculated, yellow positions with the seed at its center. (c) Right and left extension from the anchor, tile overlapping and alignment reconstruction from the traceback pointers. The blue band on the right represents the calculated portion of the dynamic programming matrix.

The threshold parameter h concerns the number of seed hits per diagonal band.

At most 1 seed hit is extended per diagonal band. This reduces redundant extensions for seed hits within the same diagonal band.

Discussion Starters

Can merge the filter + alignment operations be merged to gain efficiency?

For example, by incorporating them directly into the sequencer?

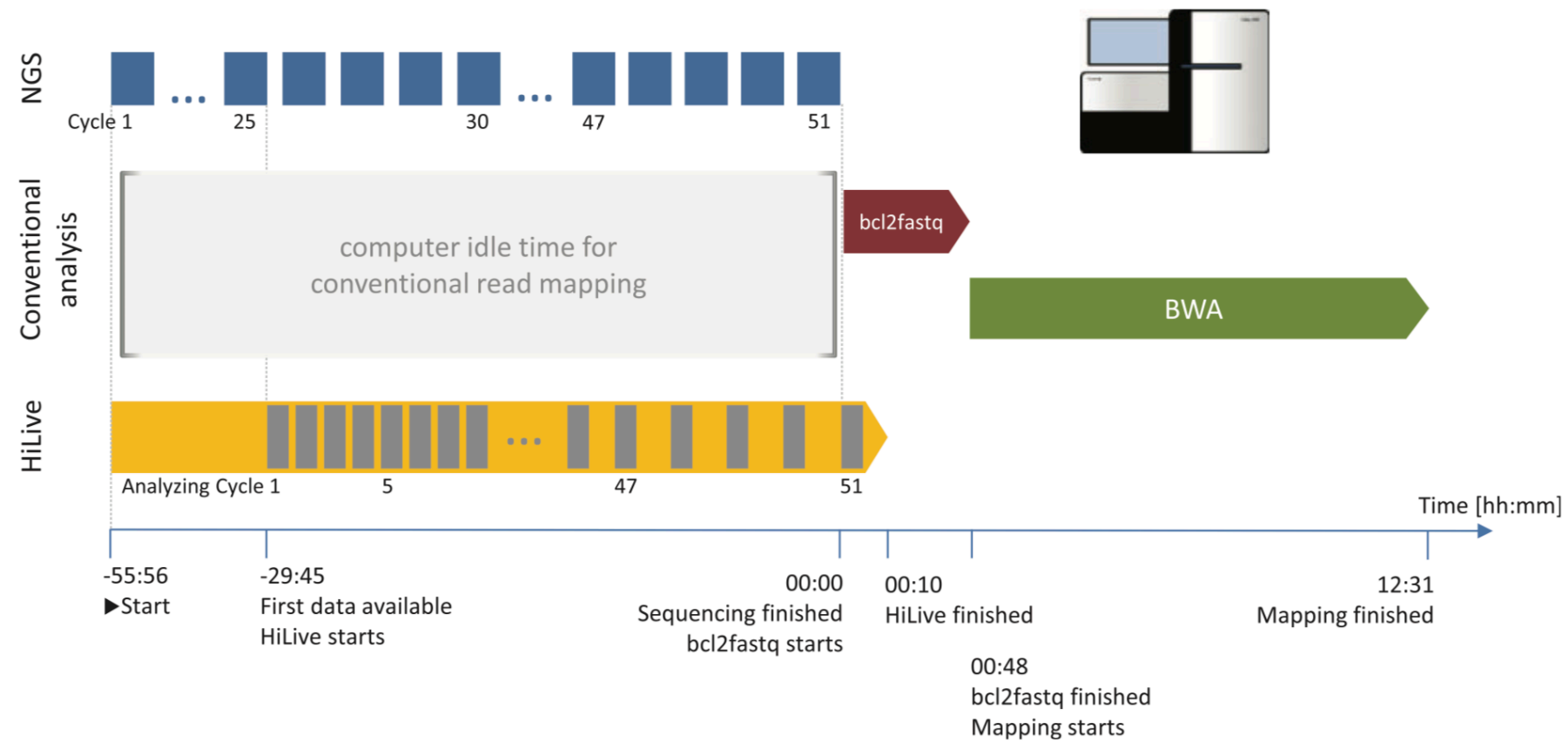
Discussion

HiLive: real-time mapping of illumina reads while sequencing

Martin S. Lindner[1,†], Benjamin Strauch[1], Jakob M. Schulze[1], Simon H. Tausch[1,2], Piotr W. Dabrowski[1,2], Andreas Nitsche[2] and Bernhard Y. Renard[1,*]

Motivation: Next Generation Sequencing is increasingly used in time critical, clinical applications. While read mapping algorithms have always been optimized for speed, they follow a sequential paradigm and only start after finishing of the sequencing run and conversion of files. Since Illumina machines write intermediate output results, HiLive performs read mapping while still sequencing and thereby drastically reduces crucial overall sample analysis time, e.g. in precision medicine.

Discussion


NGS
Cycle 1
25
30
47
51
Conventional analysis
computer idle time for conventional read mapping
bcl2fastq
BWA
HiLive
Analyzing Cycle 1
5
47
51
Time [hh:mm]
-55:56
▶Start
-29:45
First data available
HiLive starts
00:00
Sequencing finished
bcl2fastq starts
00:10
HiLive finished
00:48
bcl2fastq finished
Mapping starts
12:31
Mapping finished

Darwin: A Genomics Co-processor Provides up to 15,000 × acceleration on long read assembly

Yatish Turakhia Gill Bejerano William J. Dally

Stanford University

ASPLOS'18

João Sanches Ferreira

ETH Zürich

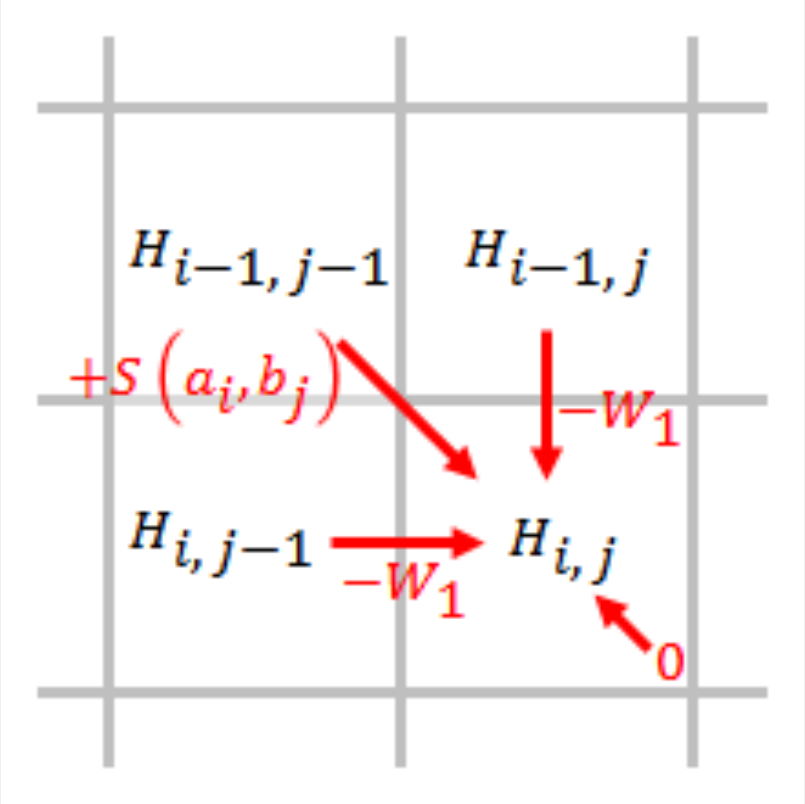
9 May 2019

Backup Slides

Alignment Algorithms

- **Smith-Waterman algorithm**

$$H_{ij} = \max \begin{cases} H_{i-1,j-1} + s(a_i, b_j), \\ H_{i-1,j} - W_1, \\ H_{i,j-1} - W_1, \\ 0 \end{cases}$$



Approximate String Matching

- **Edit distance** is defined as the minimum number of edits (i.e. insertions, deletions, or substitutions) needed to make the read exactly match the reference segment.

NETHERLANDS x SWITZERLAND

N	E	-	T	H	E	R	L	A	N	D	S
S	W	I	T	Z	E	R	L	A	N	D	-

match
deletion
insertion
mismatch

Algorithm – D-SOFT

Algorithm 1: D-SOFT

```
candidate_pos ← [] ;
last_hit_pos ← [−k  for i in range(N_B)] ;
bp_count ← [0  for i in range(N_B)] ;
for j in start : stride : end do
    seed ← Q[j : j + k];
    hits ← SeedLookup(R, seed) ;
    for i in hits do
        bin ← ⌈(i − j)/B⌉ ;
        overlap ← max(0, last_hit_pos[bin] + k − j);
        last_hit_pos[bin] ← j;
        bp_count[bin] ← bp_count[bin] + k − overlap ;
        if (h + k − overlap > bp_count[bin] ≥ h) then
            candidate_pos.append(< i, j >) ;
        end
    end
end
return candidate_pos;
```

Algorithm – GACT

Algorithm 2: GACT for Left Extension

$tb\_left \leftarrow []$;
$(i_{curr}, j_{curr}) \leftarrow (i^*, j^*)$;
$t \leftarrow 1$;
while $((i_{curr} > 0)$ ***and*** $(j_{curr} > 0))$ **do**
  $(i_{start}, j_{start}) \leftarrow (max(0, i_{curr} - T), max(0, j_{curr} - T))$;
  $(R^{tile}, Q^{tile}) \leftarrow (R[i_{start} : i_{curr}], Q[i_{start} : i_{curr}])$;
  $(TS, i_{off}, j_{off}, i_{max}, j_{max}, tb) \leftarrow$
   $Align(R^{tile}, Q^{tile}, t, T - O)$;
  $tb\_left.prepend(tb)$;
  if $(t == 1)$ **then**
    $(i_{curr}, j_{curr}) \leftarrow (i_{max}, j_{max})$;
    $t \leftarrow 0$;
  end
  if $((i_{off} == 0)$ ***and*** $(j_{off} == 0))$ **then**
    break;
  end
  else
    $(i_{curr}, j_{curr}) \leftarrow (i_{curr} - i_{off}, j_{curr} - j_{off})$
  end
end
return $(i_{max}, j_{max}, tb\_left)$;

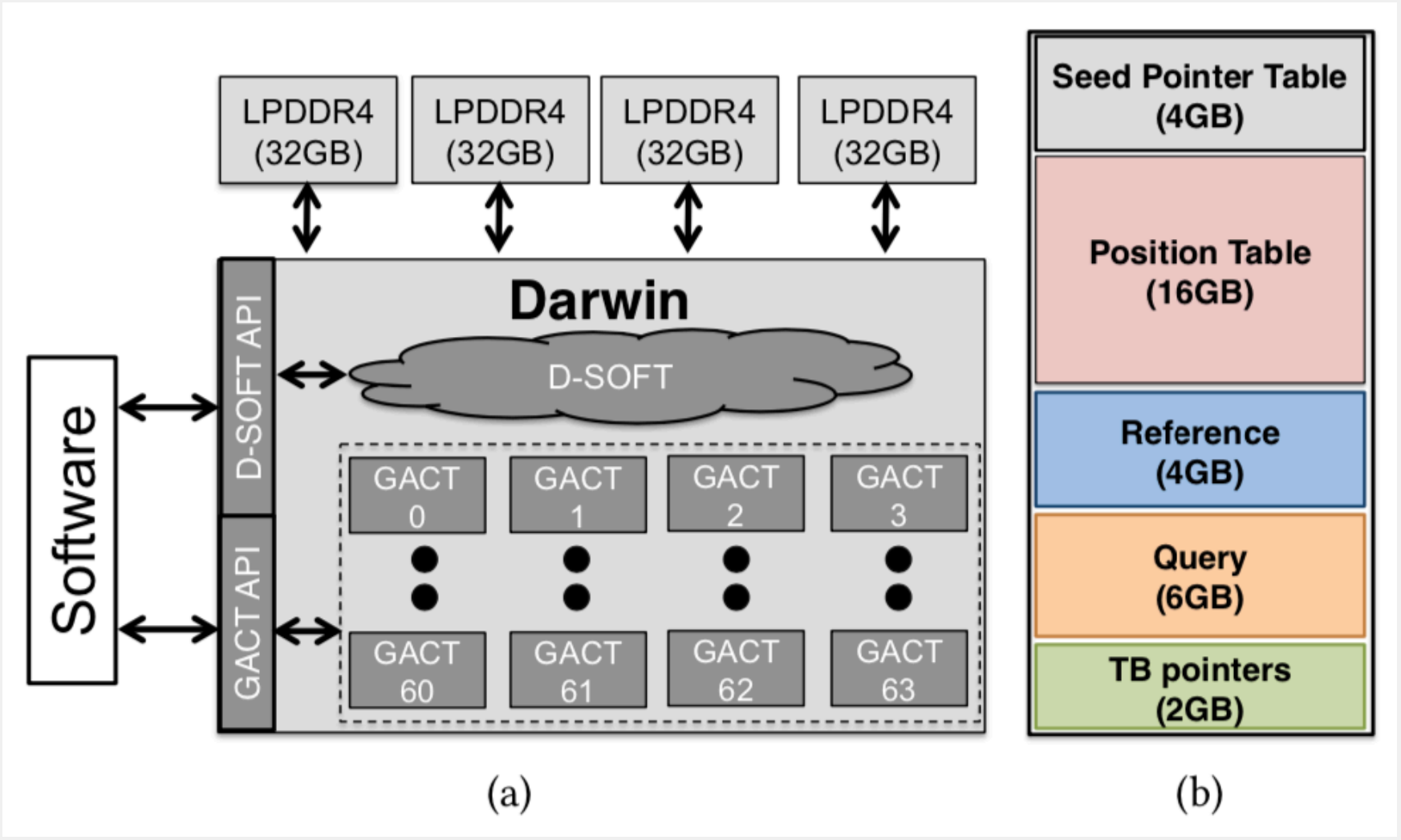
Mechanisms – GACT

$$I(i,j) = \max \begin{cases} H(i,j-1) - o \\ I(i,j-1) - e \end{cases}$$

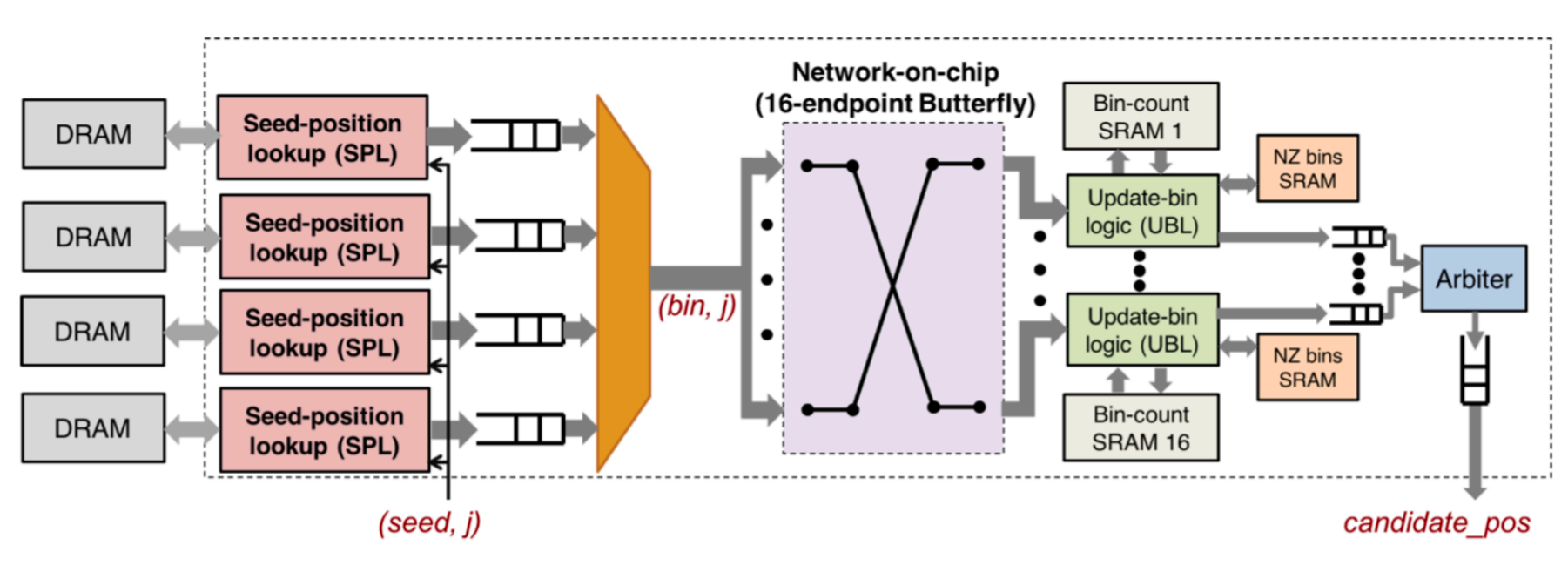
$$D(i,j) = \max \begin{cases} H(i-1,j) - o \\ D(i-1,j) - e \end{cases}$$

$$H(i,j) = \max \begin{cases} 0 \\ I(i,j) \\ D(i,j) \\ H(i-1,j-1) + W(r_i, q_j) \end{cases}$$

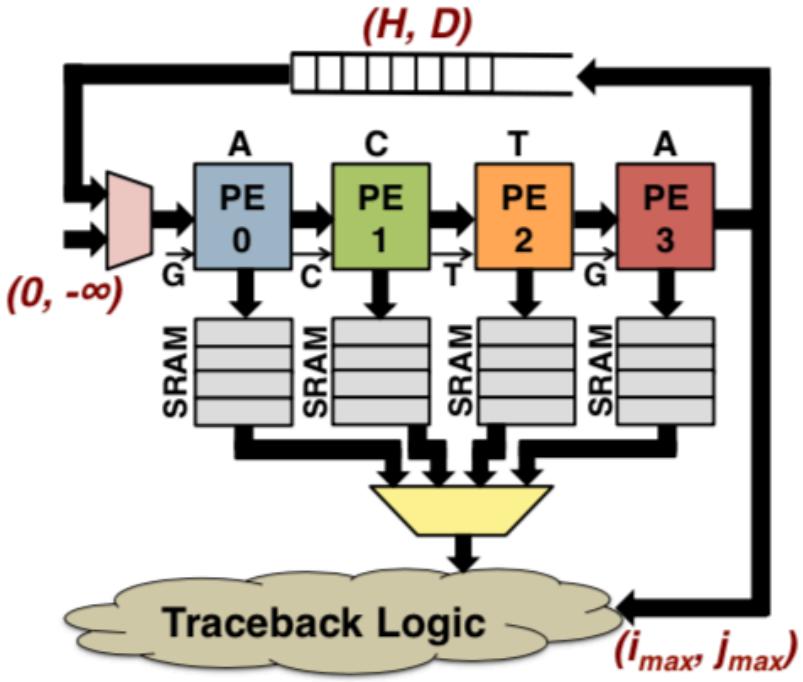
Mechanisms – Darwin Overview



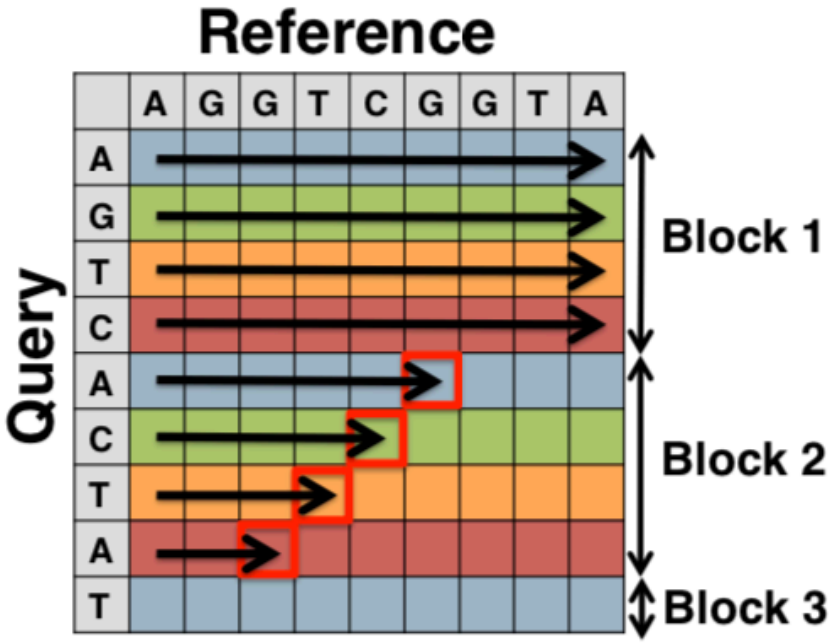
Mechanisms – D-SOFT HW Implementation



Mechanisms – GACT HW Implementation



(a)



(b)