Darwin-WGA

A Co-processor Provides Increased Sensitivity in Whole Genome Alignments with High Speedup

Yatish Turakhia*, **Sneha D. Goenka*,**

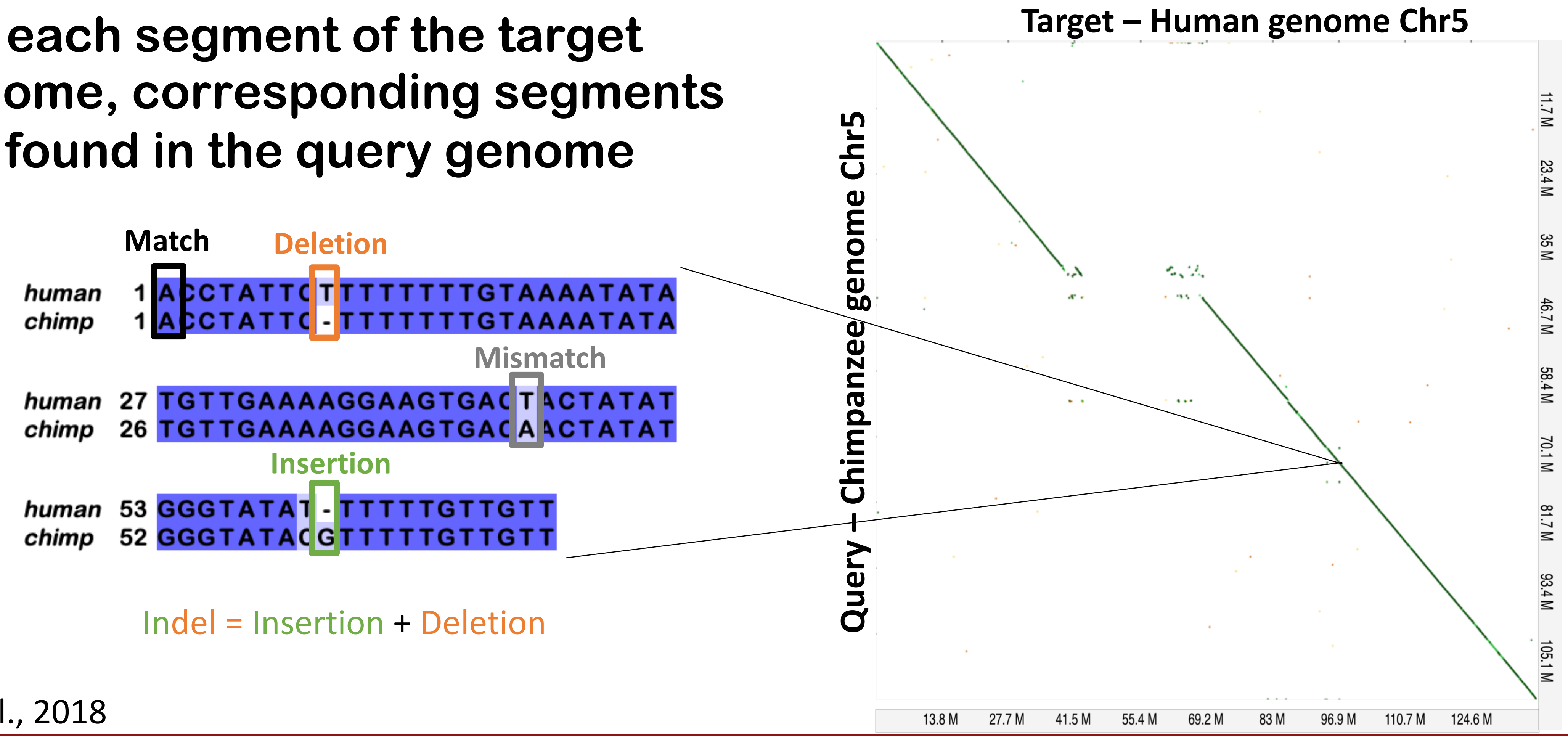
Prof. Gill Bejerano, Prof. William J. Dally

* Equal contribution

What are whole genome alignments (WGA)?

WGA is correspondence between genomes

- For each segment of the target genome, corresponding segments are found in the query genome


Match
Deletion
human 1 ACCTATTCTTTTTTTTGTAAAATATA
chimp 1 ACCTATTC-TTTTTTTTGTAAAATATA
Mismatch
human 27 TGTTGAAAAGGAAGTGACTACTATAT
chimp 26 TGTTGAAAAGGAAGTGACAACTATAT
Insertion
human 53 GGGTATAT-TTTTTGTTGTT
chimp 52 GGGTATACGTTTTTGTTGTT
Target – Human genome Chr5
Query – Chimpanzee genome Chr5
13.8 M
27.7 M
41.5 M
55.4 M
69.2 M
83 M
96.9 M
110.7 M
124.6 M
11.7 M
23.4 M
35 M
46.7 M
58.4 M
70.1 M
81.7 M
93.4 M
105.1 M

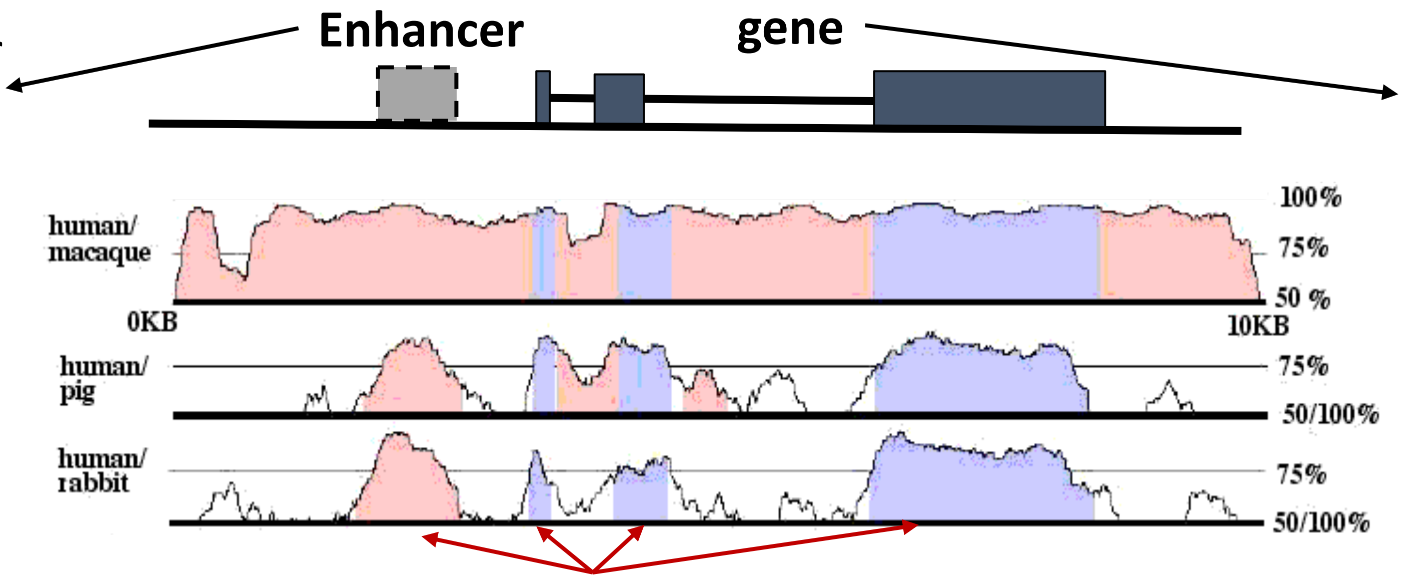
Indel = Insertion + Deletion

Cabanettes et al., 2018

Why are whole genome alignments important?

WGA help predict functional elements

Control logic for "gene expression"



Protein recipes that encode function

Regions with sequence conservation

(Mayor et al. , 2000)

Classical alignment algorithm

Smith-Waterman Algorithm

Inputs

Target sequence (r)
`GTGTCACTA` ($L_r = 9$)

Query sequence (q)
`GGCCAACTA` ($L_q = 9$)

Scoring parameters

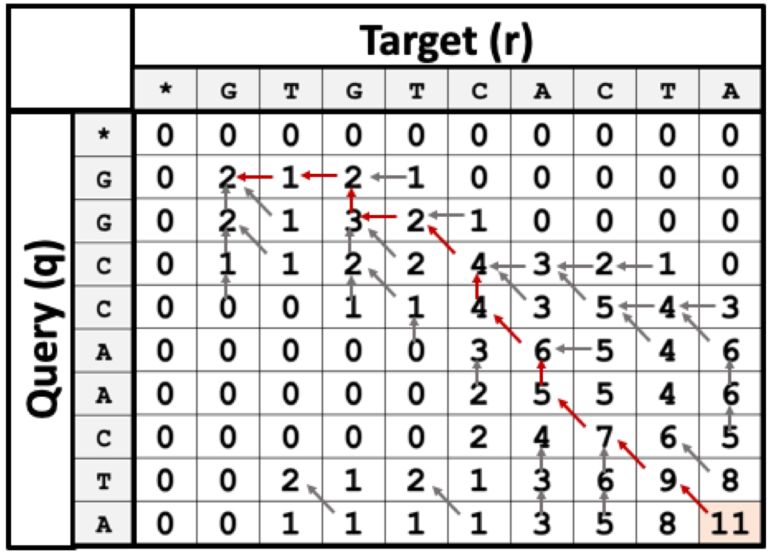
W =

	A	C	G	A
A	2	−1	−1	−1
C	−1	2	−1	−1
G	−1	2	2	−1
T	−1	−1	−1	2

Gap penalty = 1

Smith-waterman equations

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



		Target (r)									
		*	G	T	G	T	C	A	C	T	A
Query (q)	*	0	0	0	0	0	0	0	0	0	0
	G	0	2	1	2	1	0	0	0	0	0
	G	0	2	1	3	2	1	0	0	0	0
	C	0	1	1	2	2	4	3	2	1	0
	C	0	0	0	1	1	4	3	5	4	3
	A	0	0	0	0	0	3	6	5	4	6
	A	0	0	0	0	0	2	5	5	4	6
	C	0	0	0	0	0	2	4	7	6	5
	T	0	0	2	1	2	1	3	6	9	8
	A	0	0	1	1	1	1	3	5	8	11

Alignment

```
GTGTC-A-CTA
G-G-CCAACTA
```

Smith Waterman algorithm intractable on whole genomes

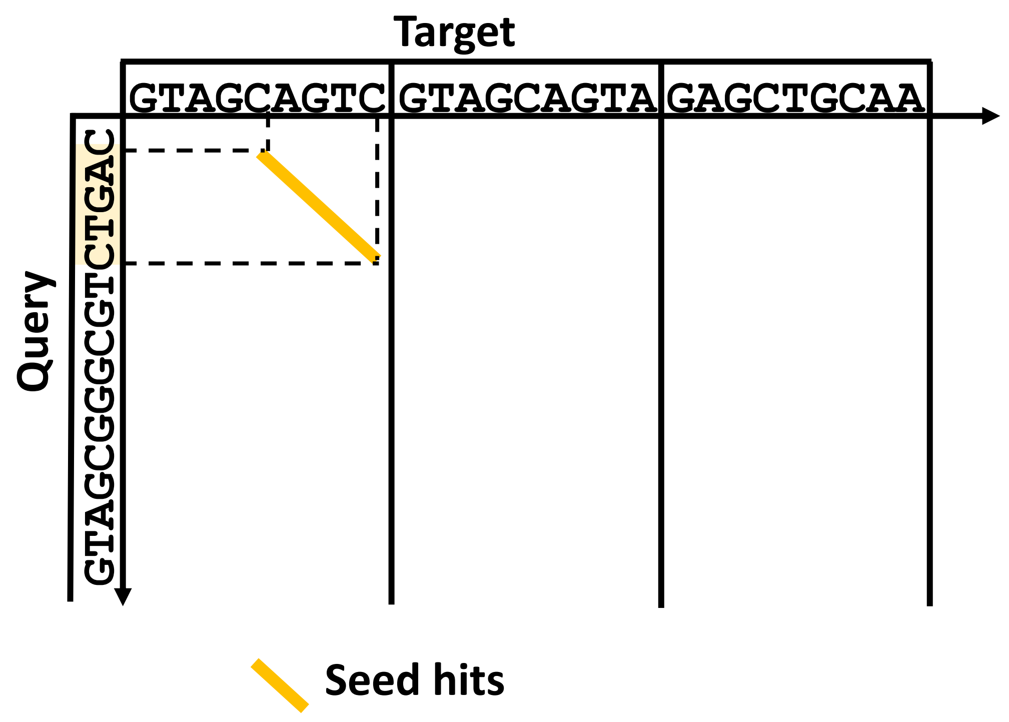
- Smith Waterman algorithm time and space complexity ~ $O(L_r . L_q)$

- Mammalian genomes ~ 10^9-10^{10} base-pairs

- Use heuristics based approaches – *seed-filter-extend*

Seed-Filter-Extend algorithm (LASTZ)

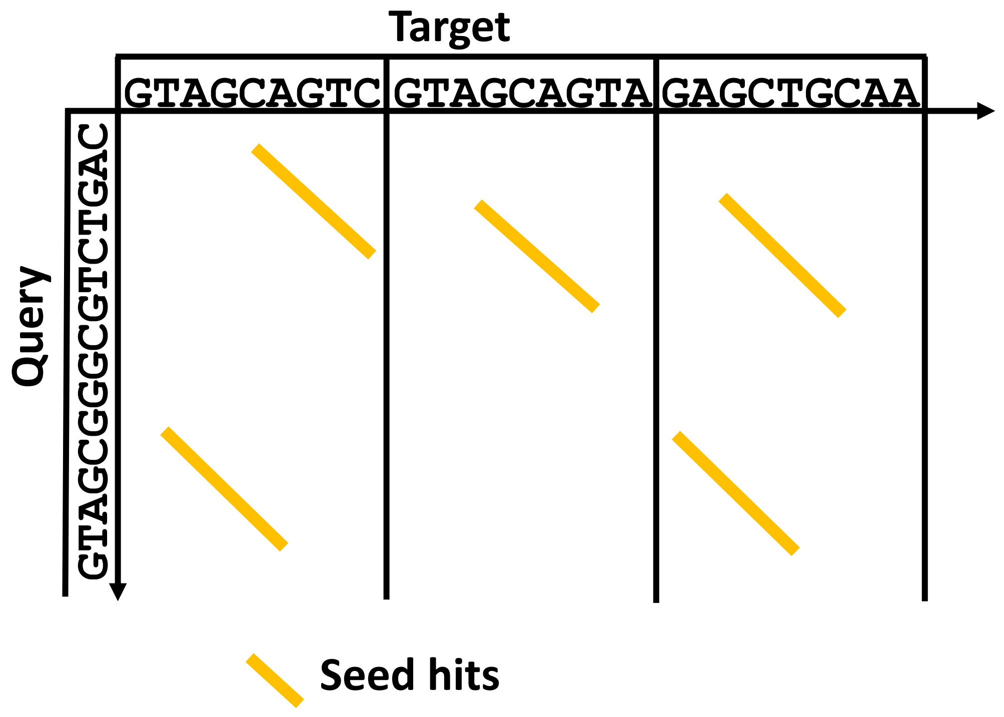
Seeding finds small matching local patterns



Seed

- Seeding finds local matching patterns of fixed length s
- Substrings of query of length s are compared to the target
- Substrings start from position 0

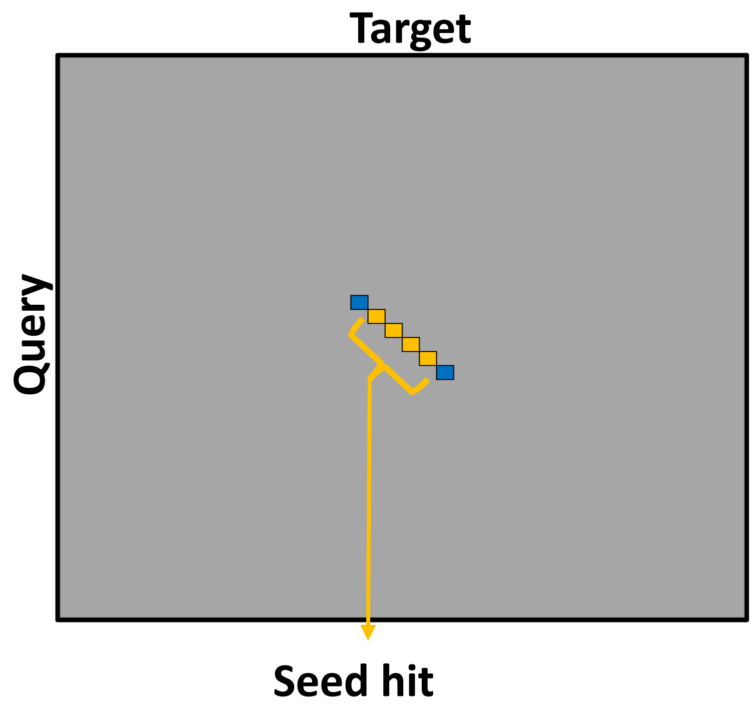
Seeding finds small matching local patterns



- Seeding finds local matching patterns of fixed length s
- Substrings of query of length s are compared to the target
- Substrings start from position 0

Seed

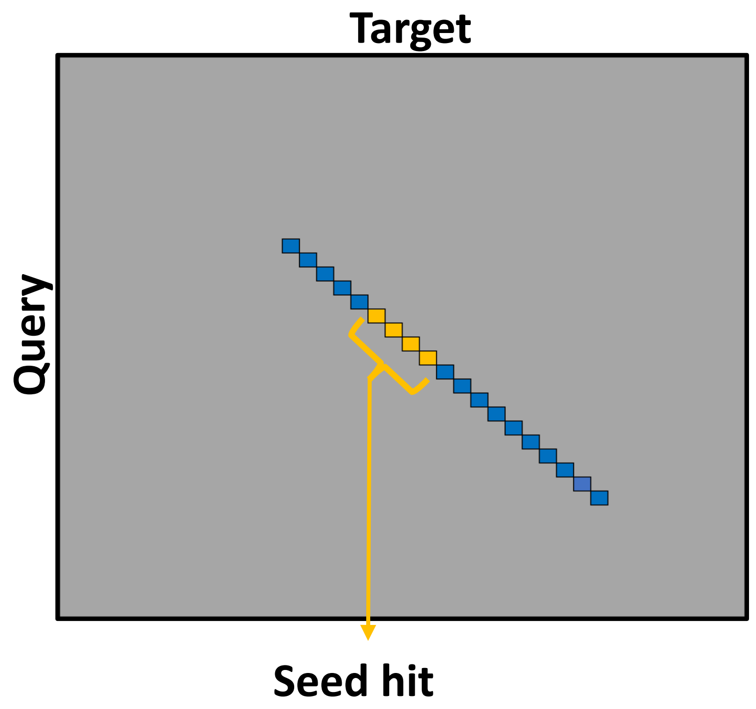
Filtering aligns ~100bp around seed hits



Filter (Ungapped)

- Calculate scores along the seed hit diagonal (match or mismatch)
- Track maximum score
- Stop as soon as score falls below (max_score - x)
- Does not consider indels

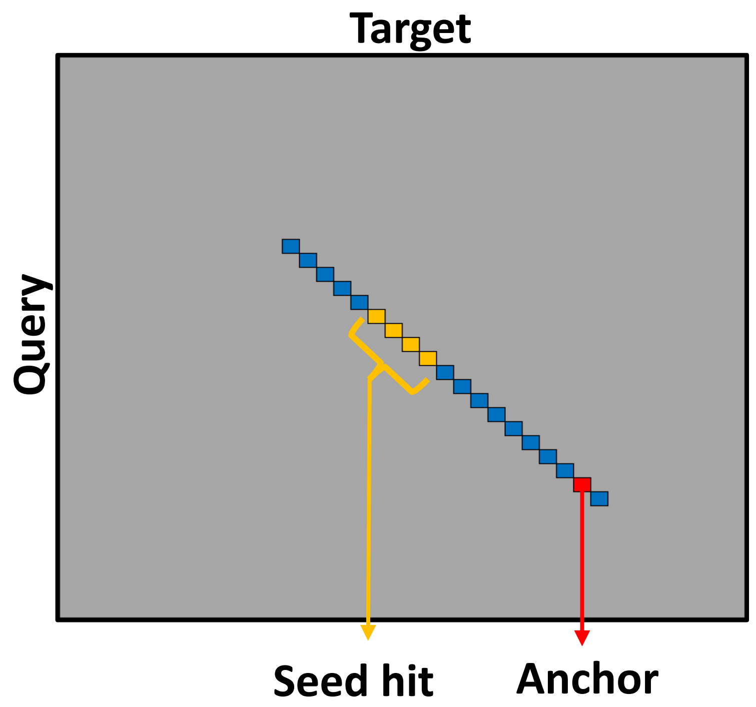
Filtering aligns ~100bp around seed hits



Filter (Ungapped)

- Calculate scores along the seed hit diagonal (match or mismatch)
- Track maximum score
- Stop as soon as score falls below (max_score - x)
- Does not consider indels

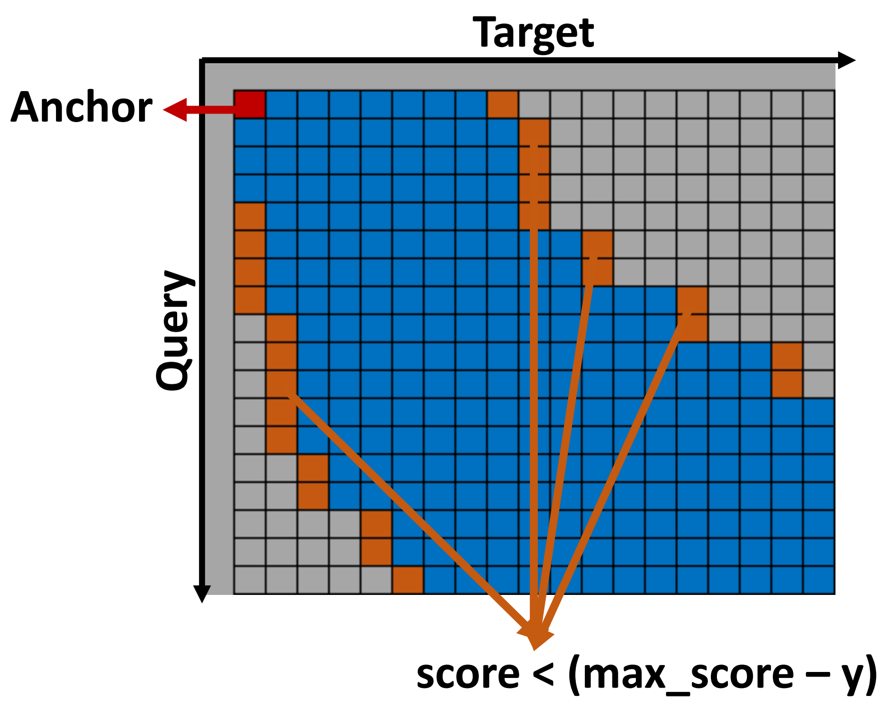
Filtering aligns ~100bp around seed hits



Filter (Ungapped)

- Calculate scores along the seed hit diagonal (match or mismatch)
- Track maximum score
- Stop as soon as score falls below (max_score - x)
- Does not consider indels

Extension uses Y-drop algorithm



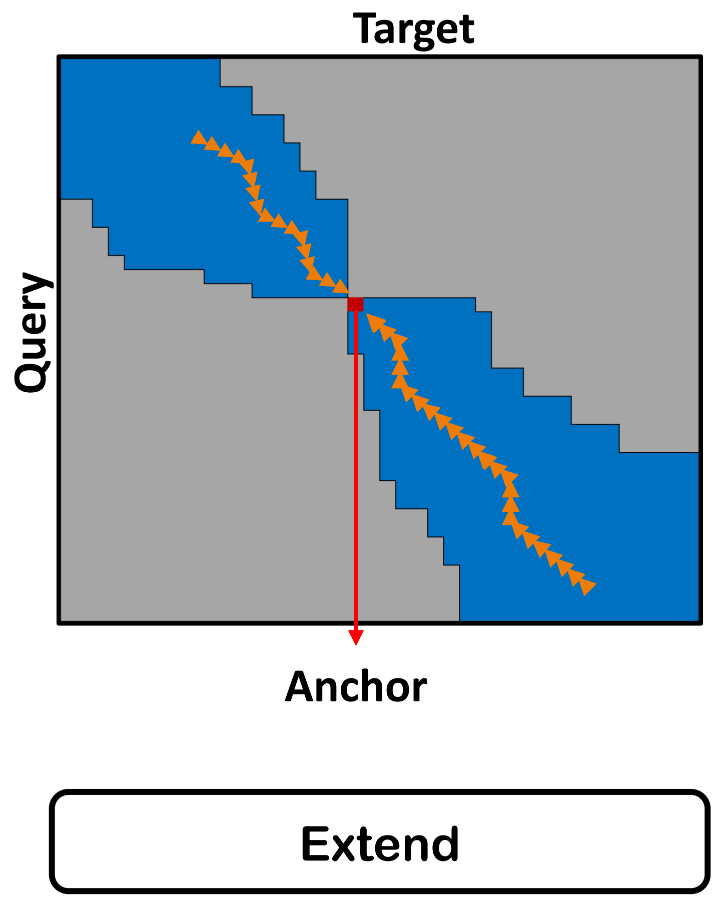
- Start computing score along a row when

  score > (max_score - y)

- Stop computing score along a row when

  score < (max_score - y)

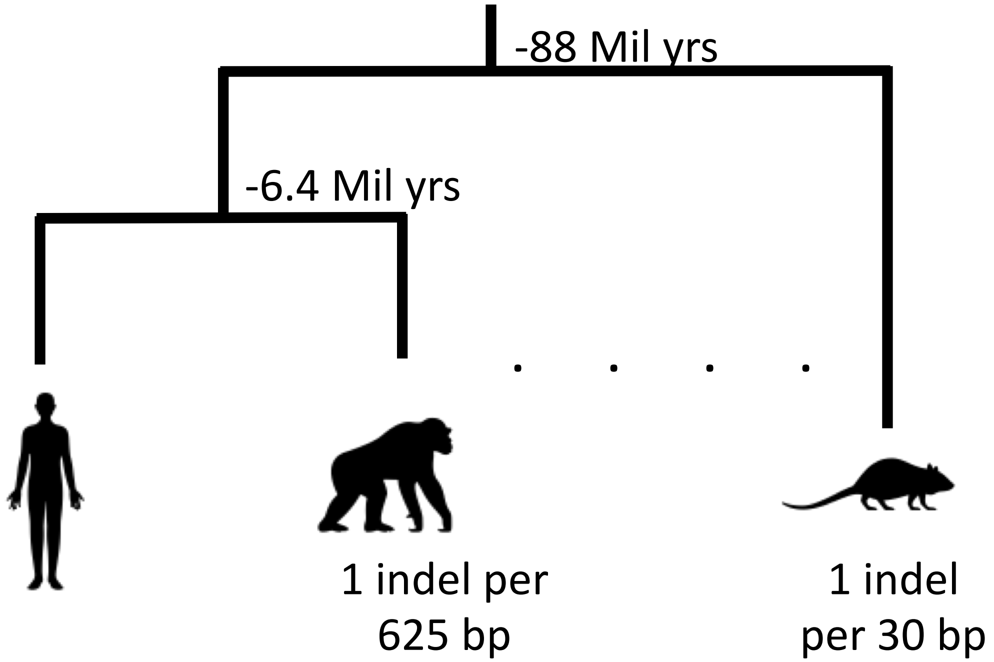
Extension provides the final alignments

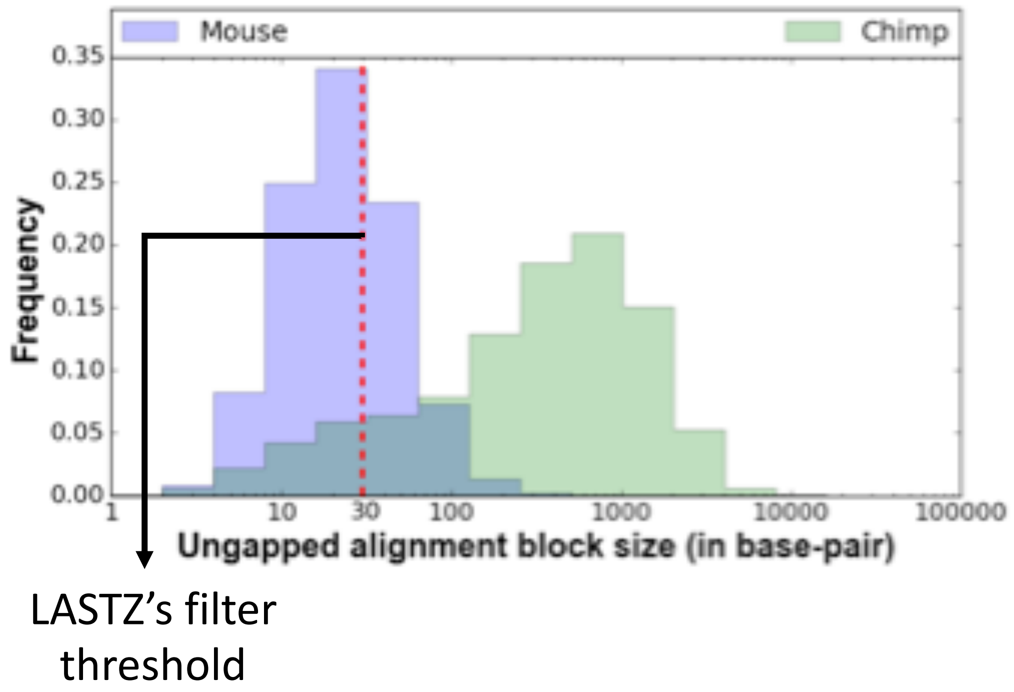


Final alignment = right extension + left extension

Why is LASTZ less sensitive?

Increasing indel frequency => increasing need for gapped filtering



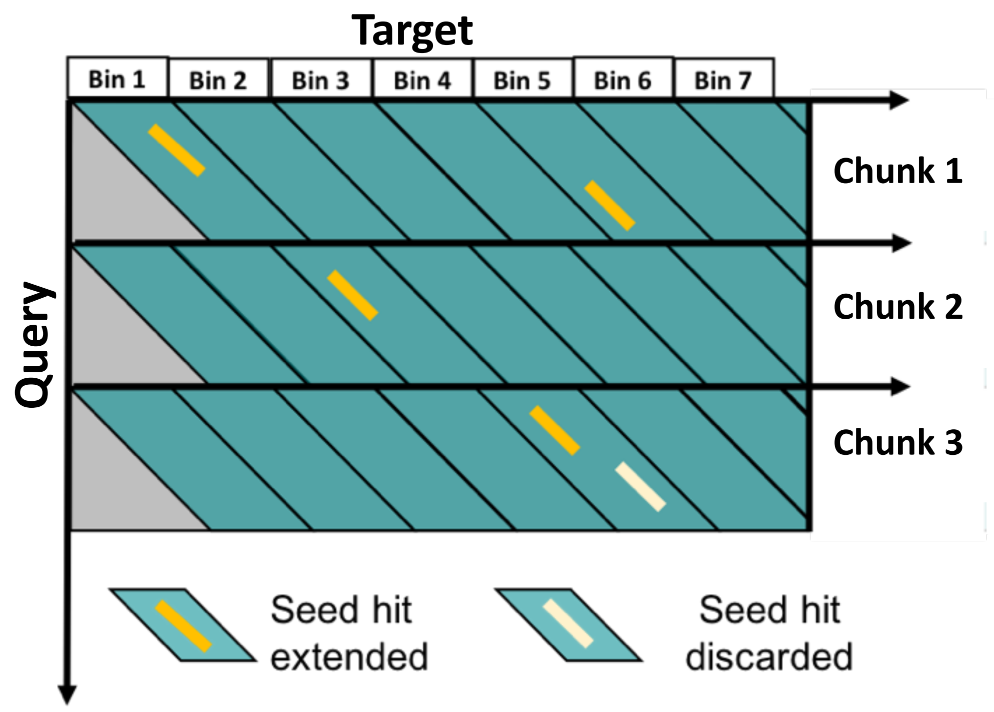


Replacing ungapped filtering by gapped filtering slows down the software by 200x

Darwin-WGA algorithm overview

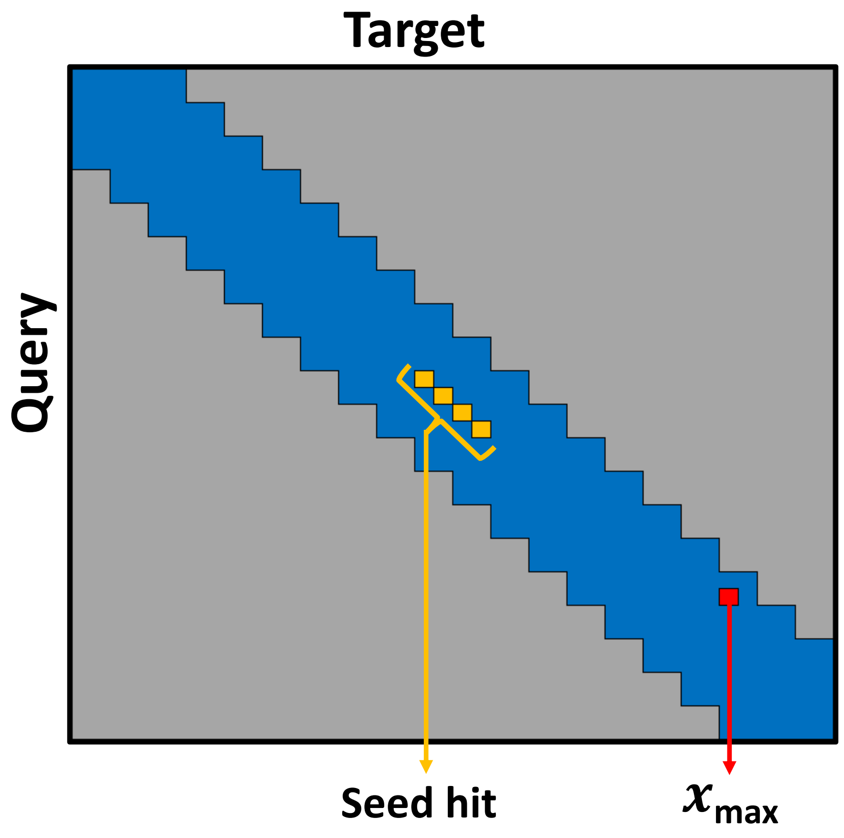
Seeding – D-SOFT

- Target bin and Query chunk determine diagonal band
- Each seed hit falls in a single diagonal band
- At most 1 seed hit per diagonal band is extended



Gapped Filtering – Banded Smith Waterman

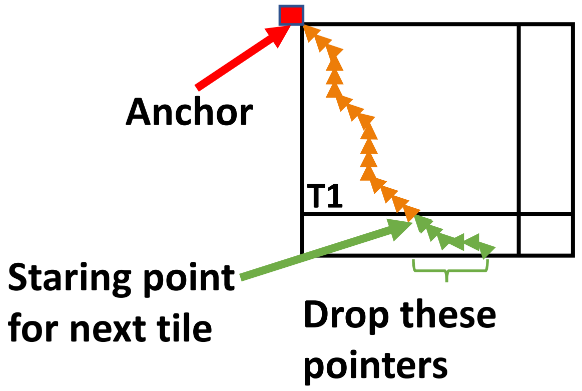
- Seed hit extended using banded Smith-Waterman
- Pre-determined band with no traceback
- If (max_score > threshold), the maximum score position (x_{max}) is the *anchor* or the starting point for alignment extension
- Gaps considered => better alignments for species further apart

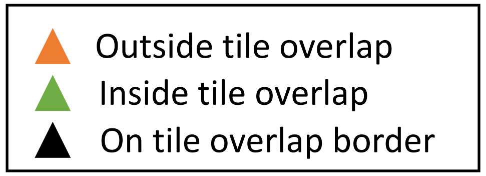


Extension - GACT-X algorithm

- Tiled (tile size T, overlap O) implementation inspired by GACT in Darwin*

- Origin of the next tile lies at the intersection of the current traceback path with the overlap

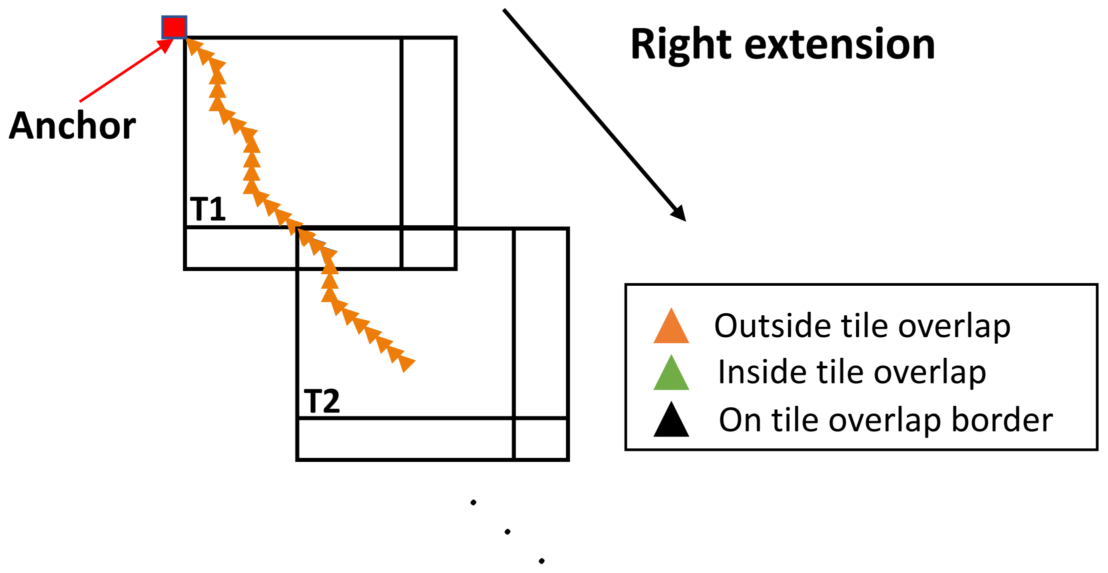




* Turakia et al. , ASPLOS'18

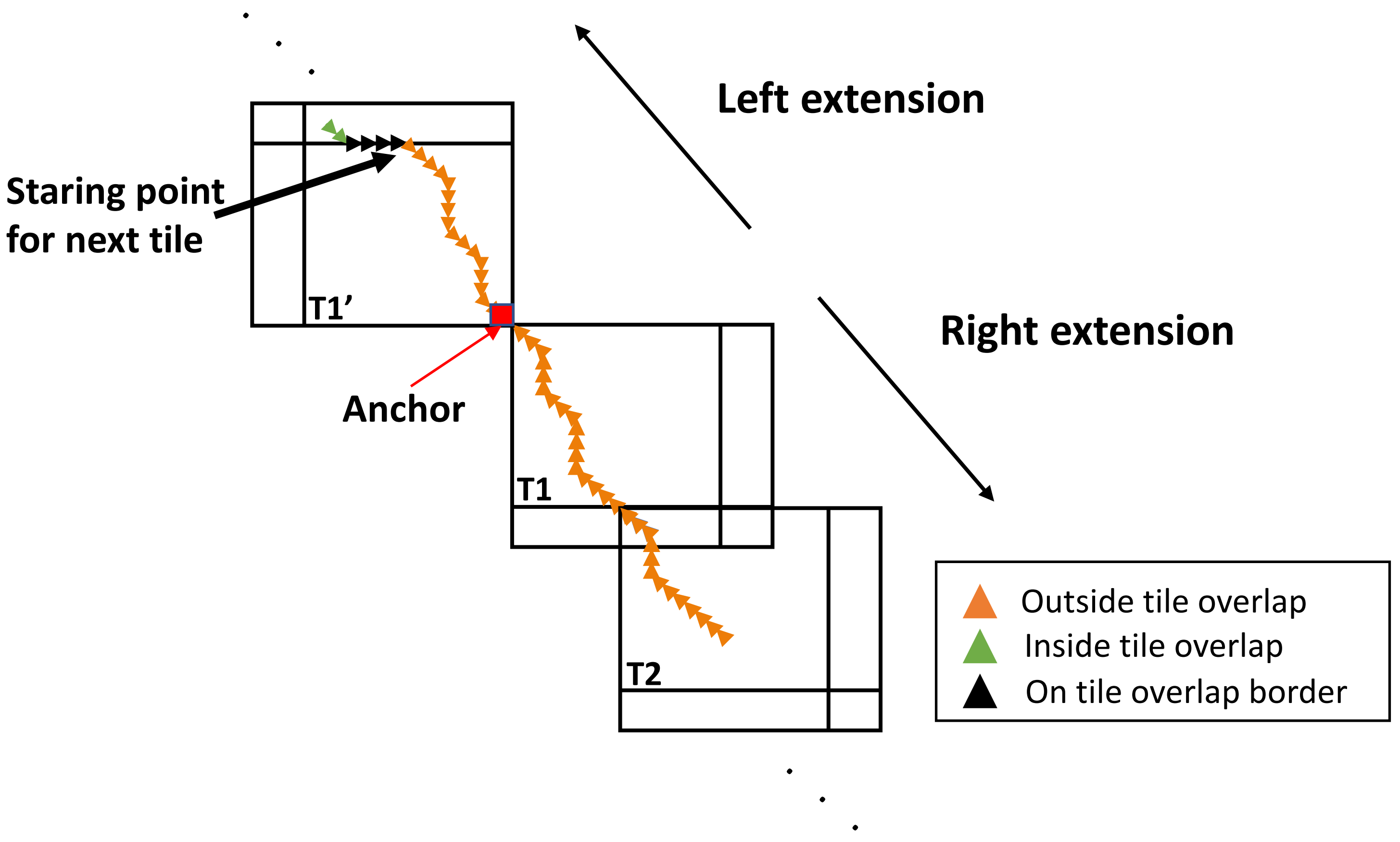
Extension - GACT-X algorithm

- Extension along a direction continues until a tile is encountered with a non-positive maximum score



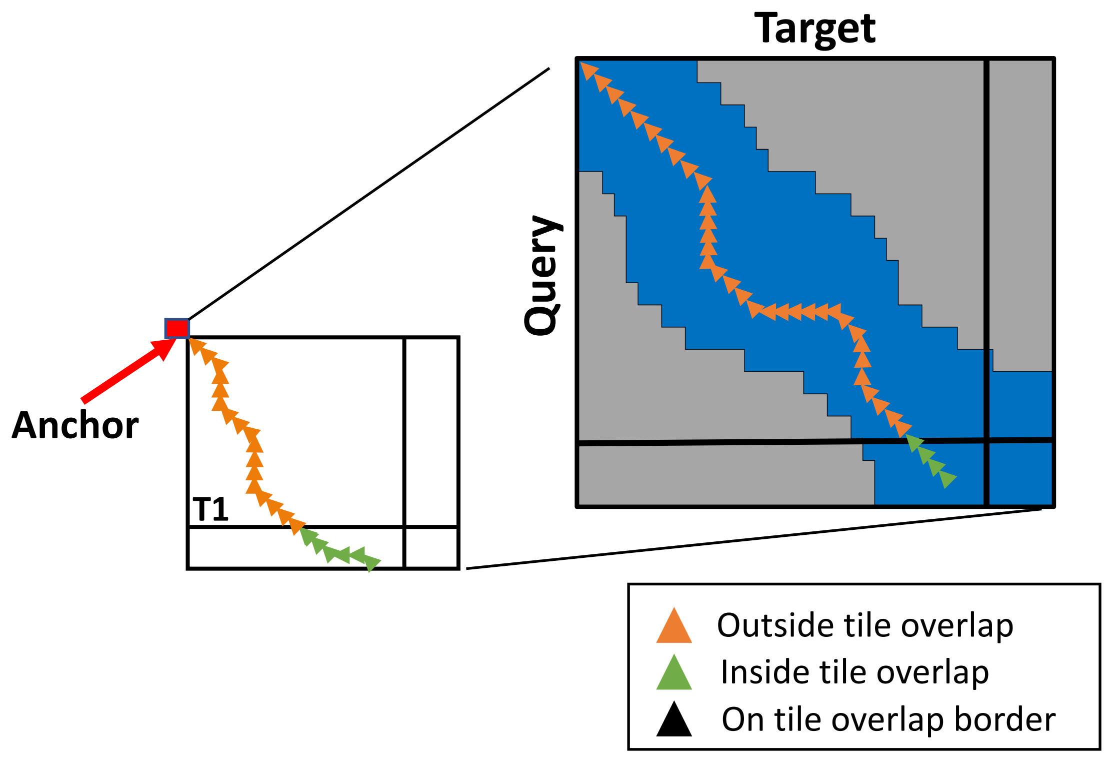
Extension - GACT-X algorithm

- Final alignment combines left and right extension



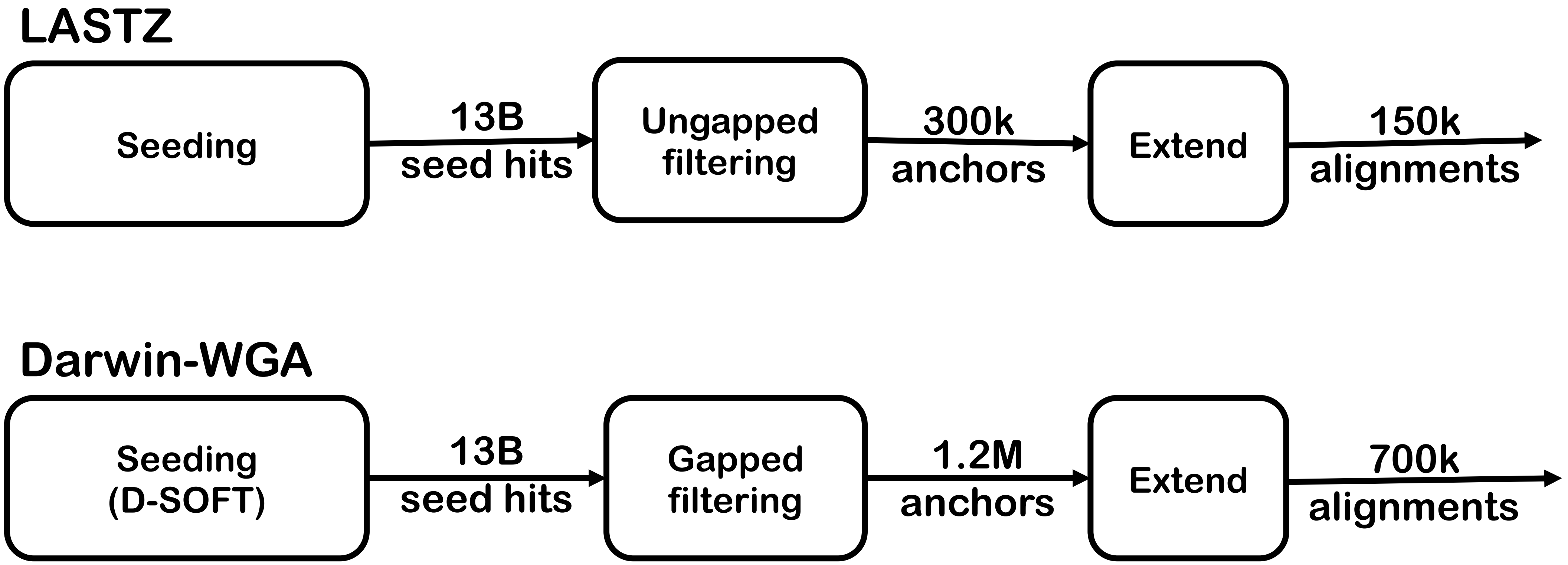
Extension - GACT-X algorithm

- Y-drop implementation within each tile
- Adaptive band with traceback
- Reduces on-chip memory requirement compared to computing whole tile
- Reduces compute time



* Turakhia et al. , ASPLOS’18

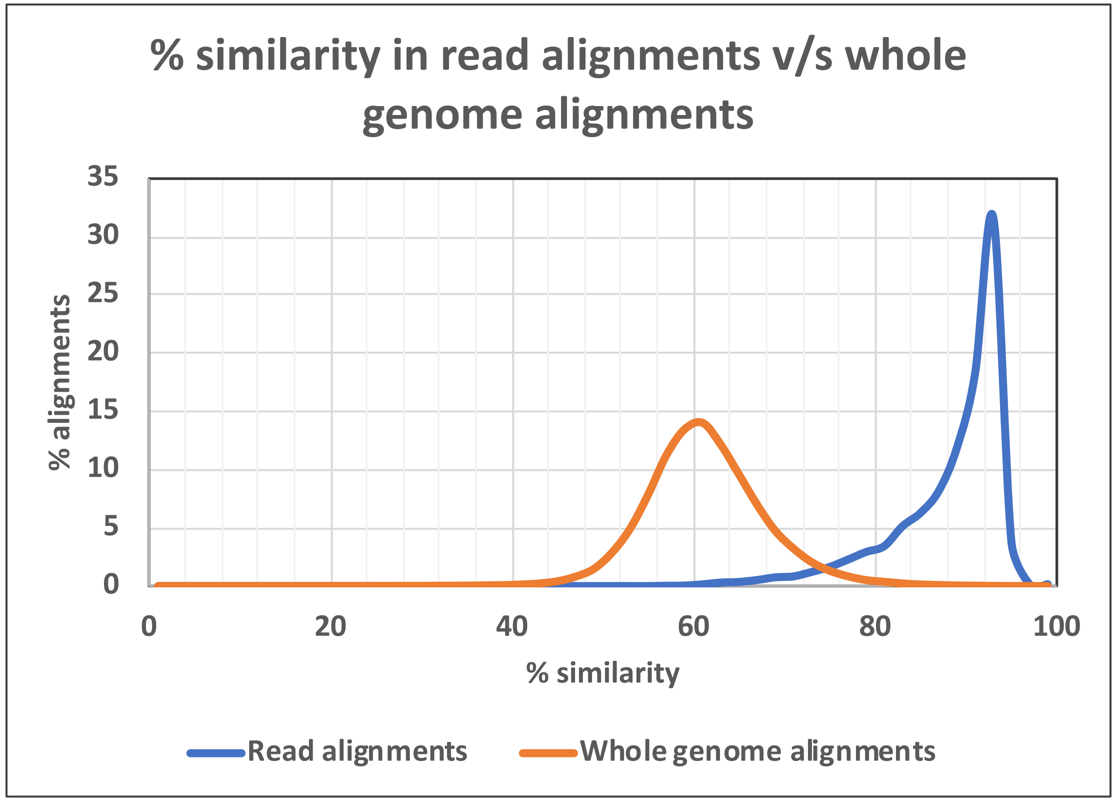
Workloads in LASTZ v/s Darwin-WGA



Whole genome alignment
v/s
Read alignment

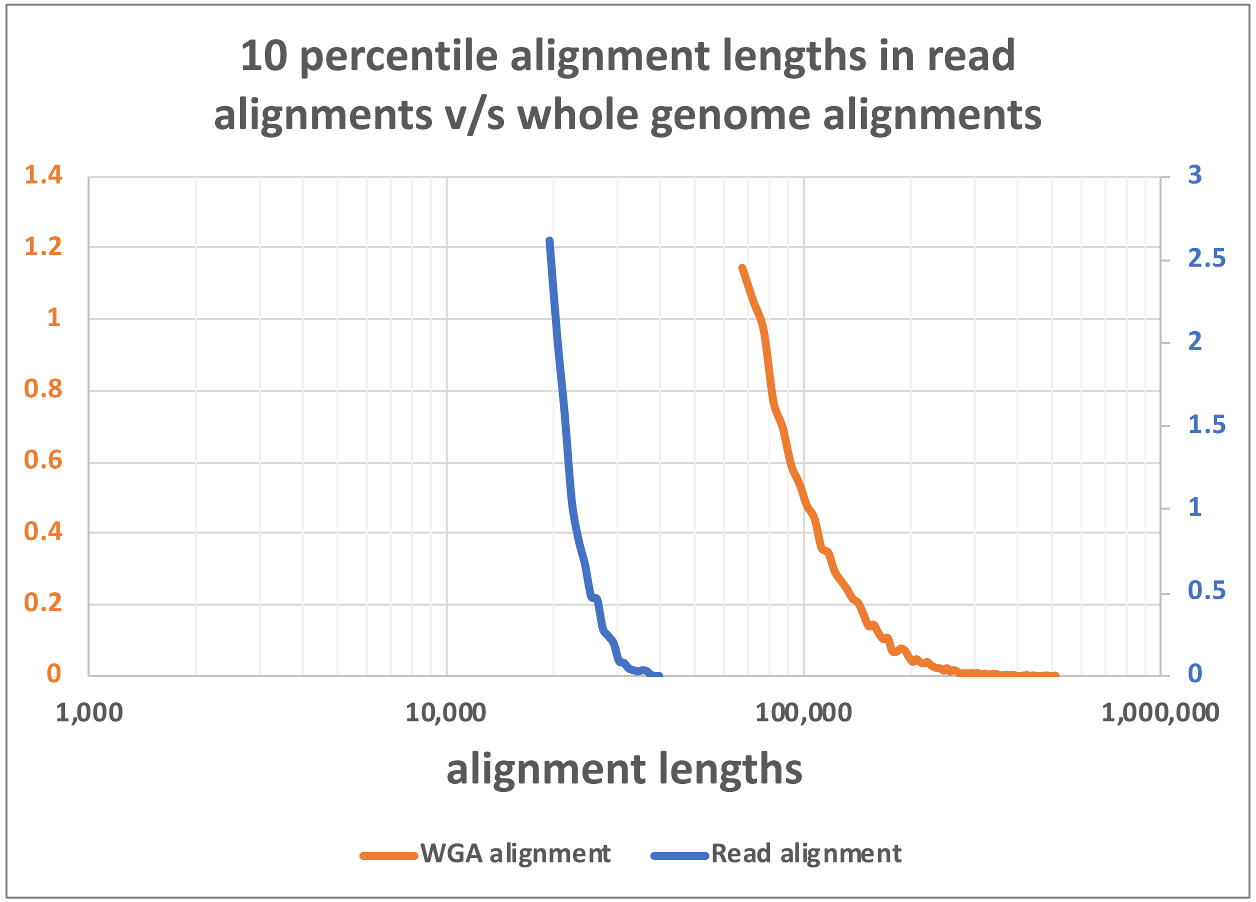
1. WGA requires aligns less similar sequences

- Genomes may diverge considerably over evolutionary timescales and have low sequence similarity
- Read alignment deals with highly similar sequences (well-characterized sequencing error model)

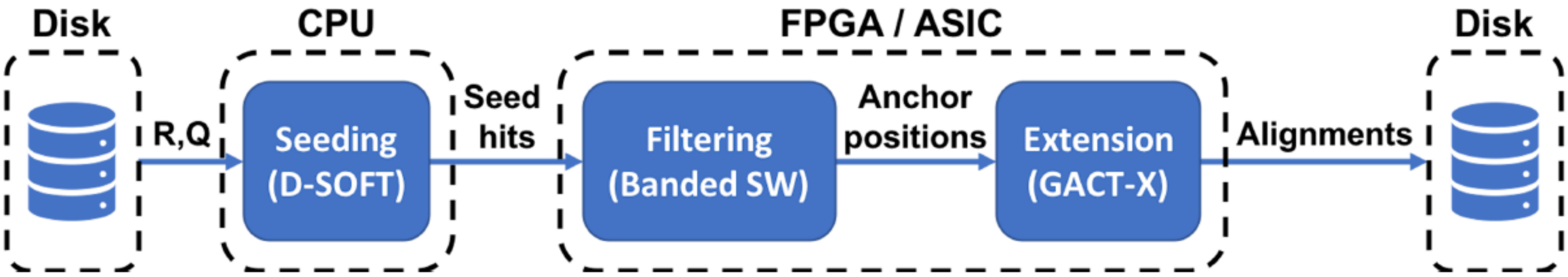


2. WGA has longer alignments with large indels

- Whole genome alignments can span millions of base-pairs with large indels
- Read alignments span not more than tens of thousands of base-pairs with much shorter indels
- Previous hardware accelerators would require high on-chip memory



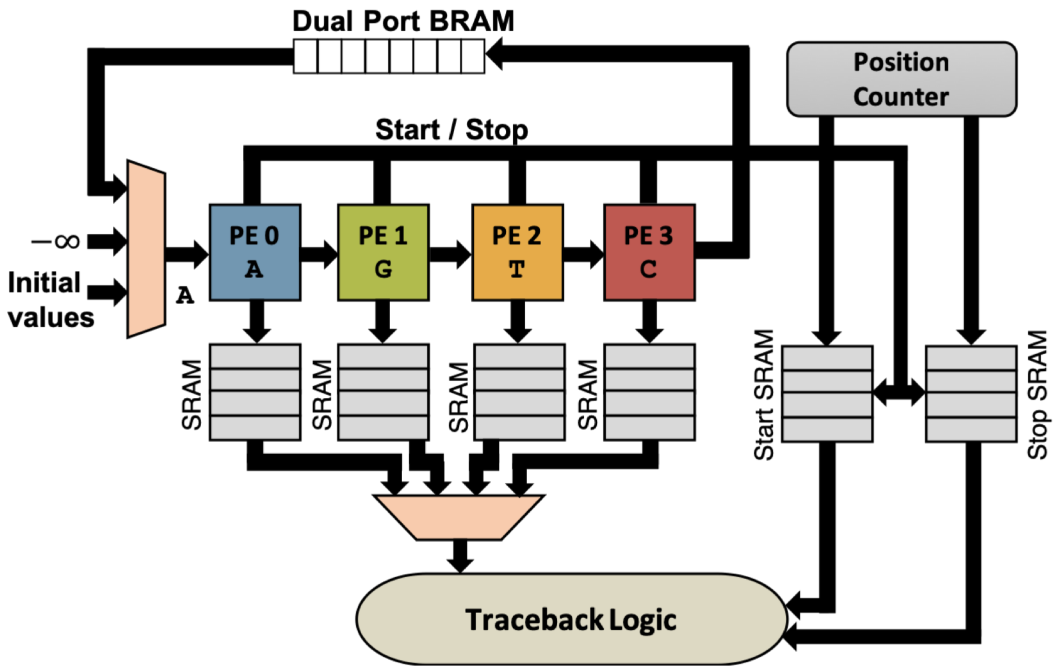
Darwin-WGA Framework

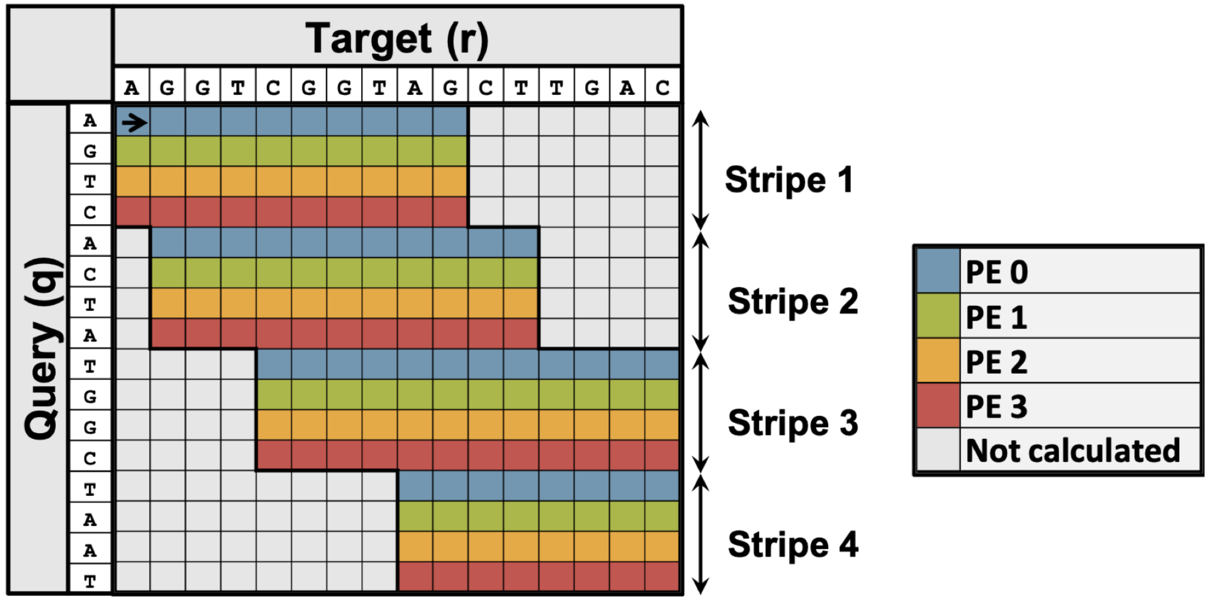


- Seeding done in software
- Banded Smith-Waterman and GACT-X accelerated in hardware as bounded Dynamic Programming with Systolic Arrays

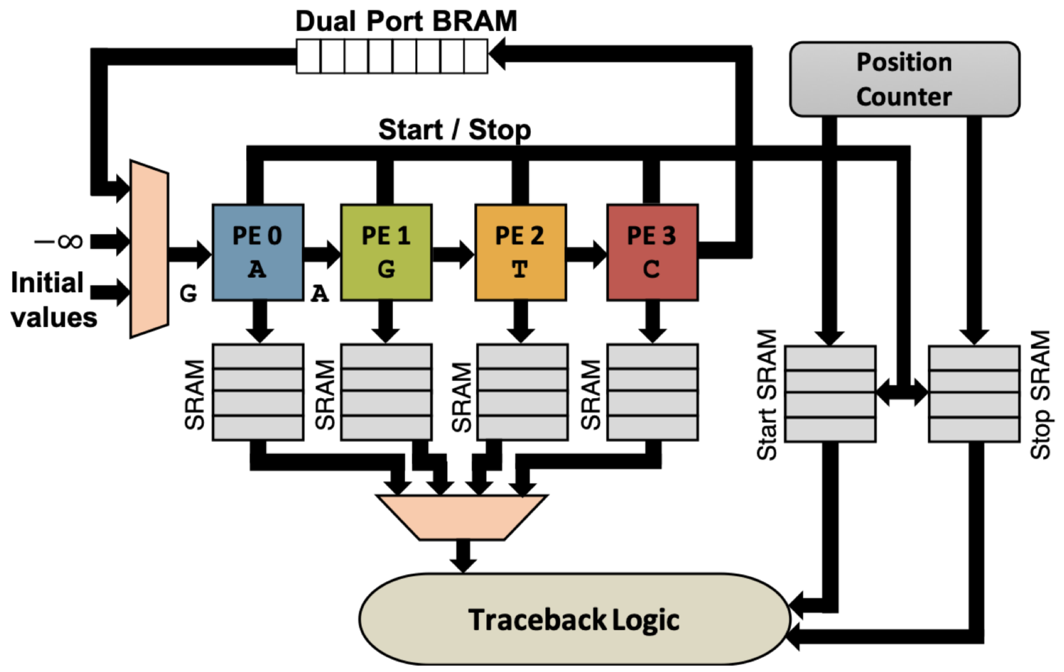
Hardware Acceleration

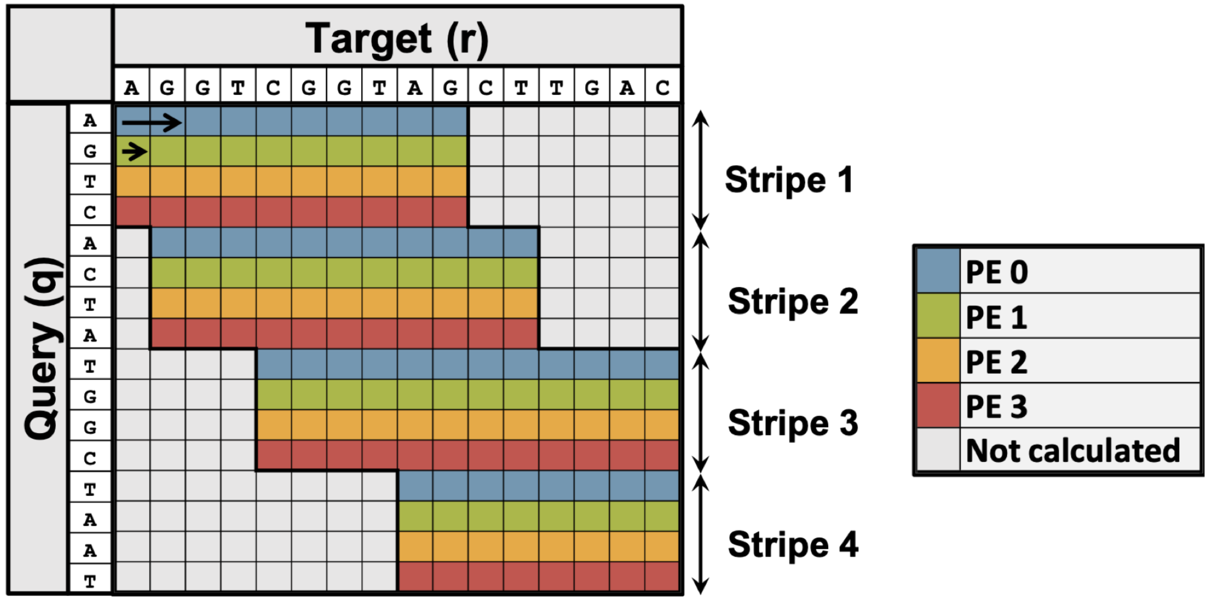
Accelerating bounded Dynamic Programming with Systolic Arrays



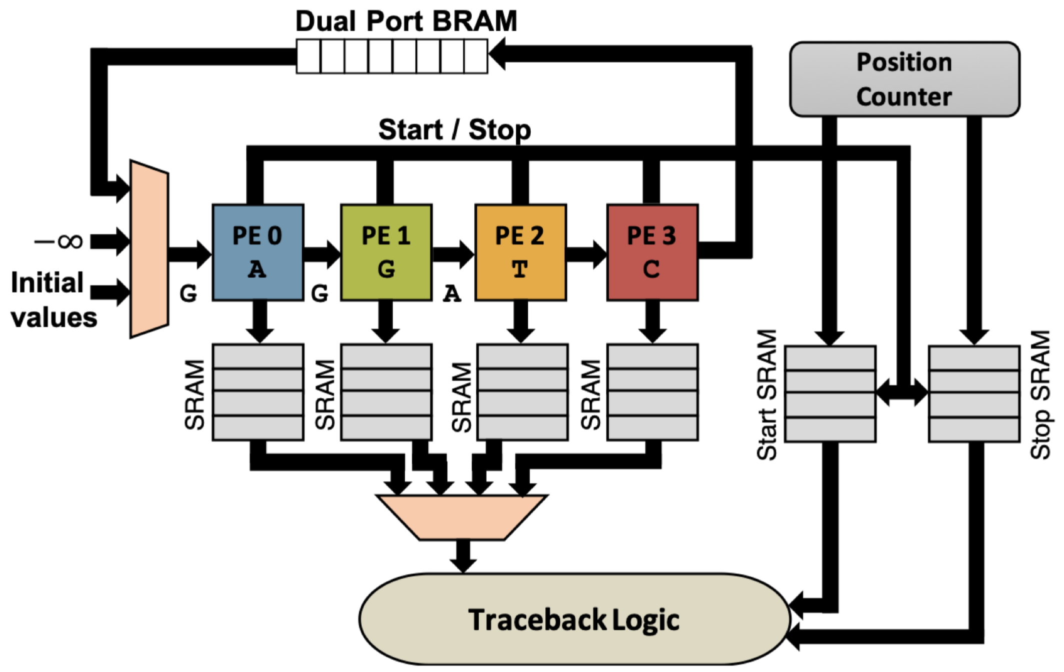


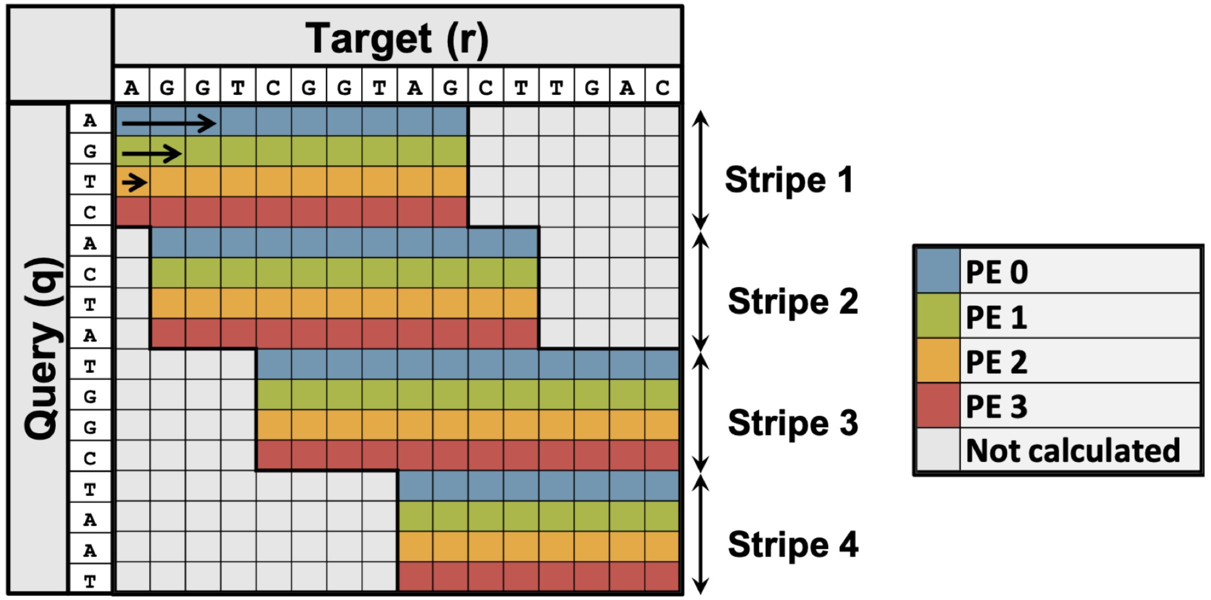
Accelerating bounded Dynamic Programming with Systolic Arrays



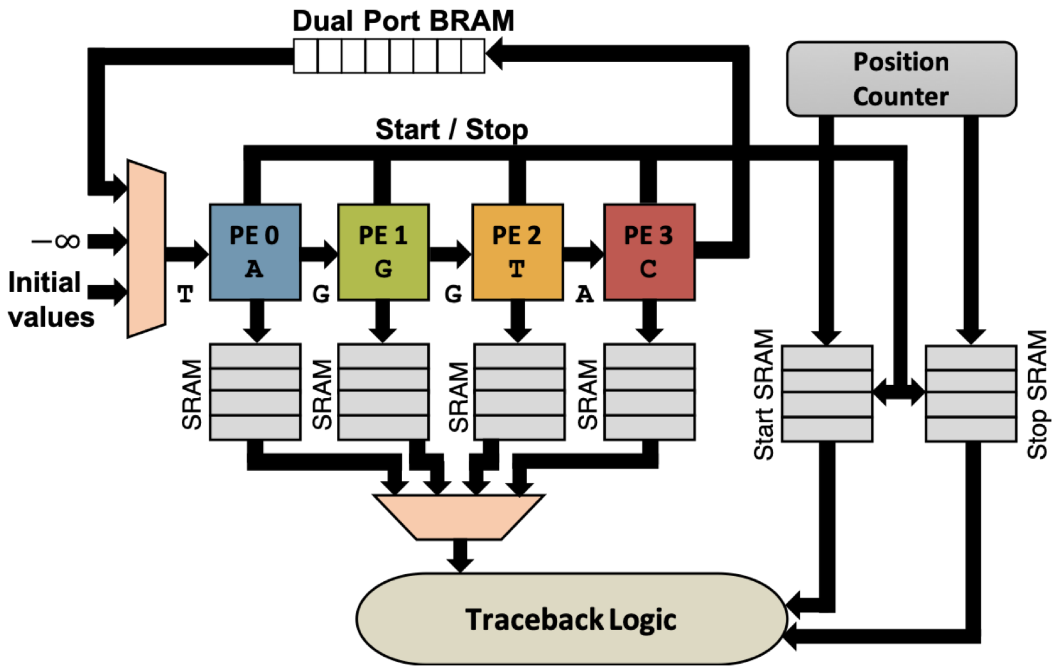


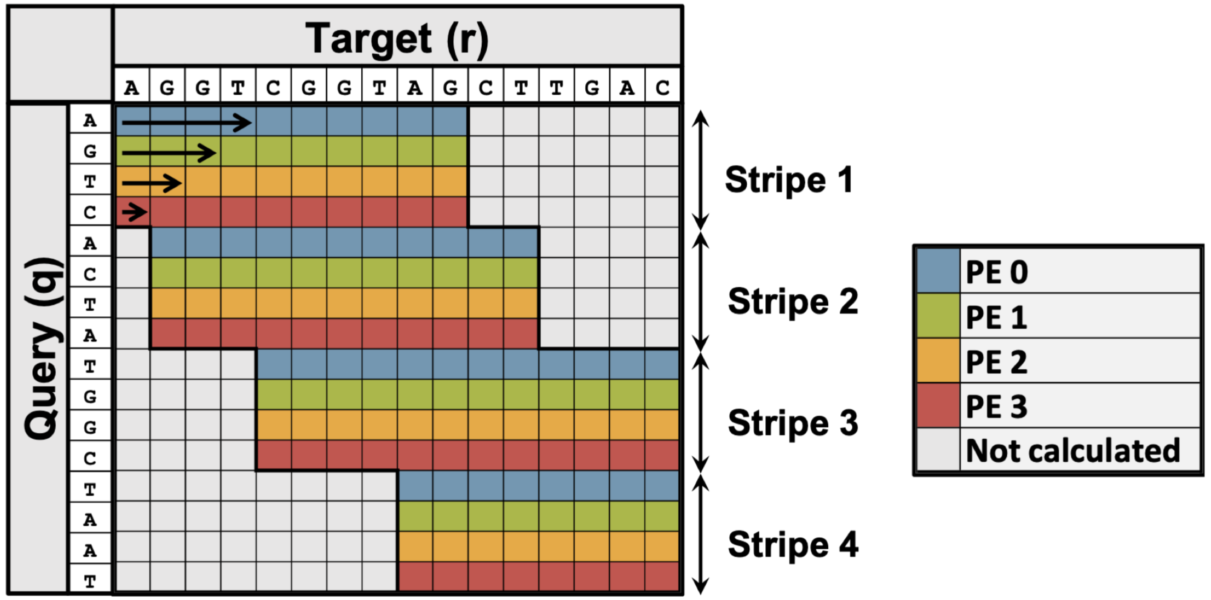
Accelerating bounded Dynamic Programming with Systolic Arrays



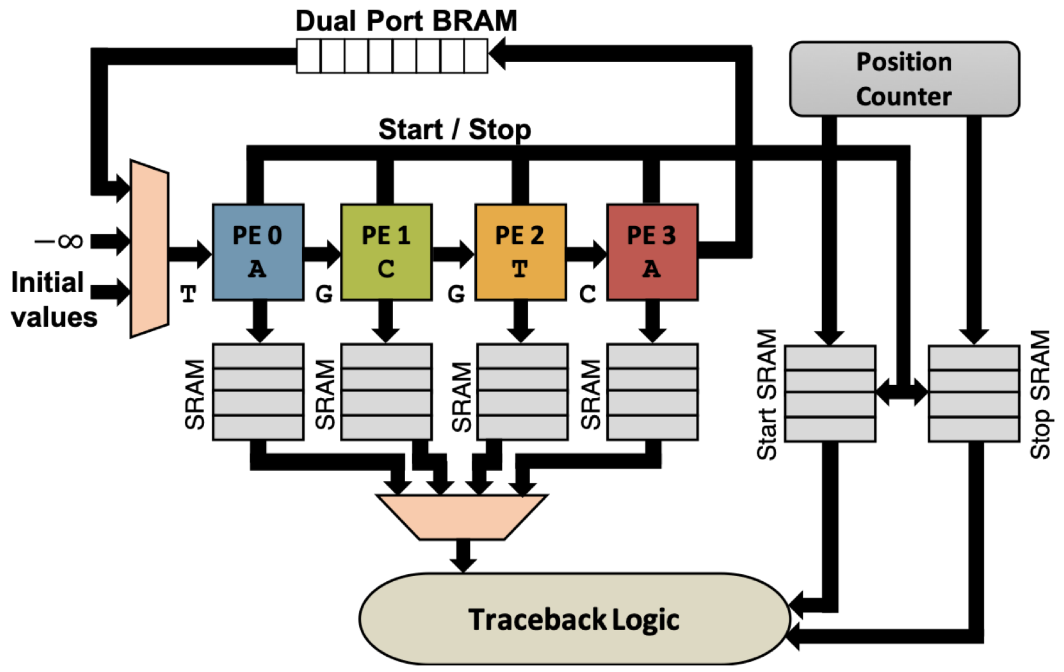


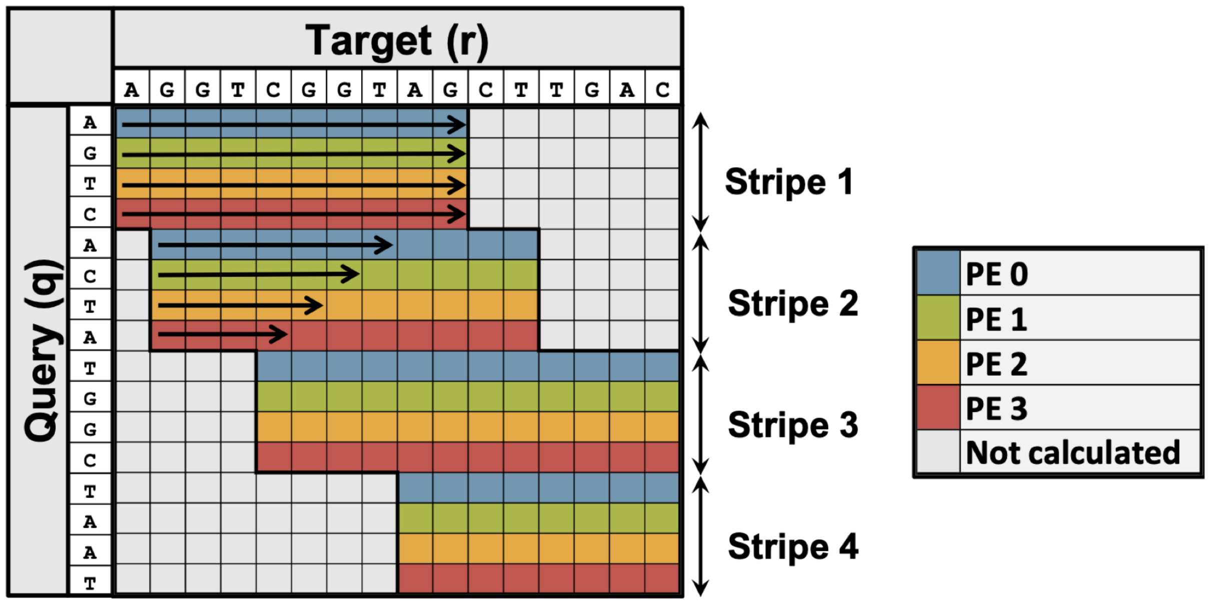
Accelerating bounded Dynamic Programming with Systolic Arrays





Accelerating bounded Dynamic Programming with Systolic Arrays





- Banded Smith-Waterman - preset band and no traceback
- GACT-X - adaptive band with traceback

Evaluation Framework

Experimental Setup

CPU (baseline)

- AWS c4.8xlarge instance
- 36 vCPUs (18 physical cores)
- LASTZ as software baseline
- Parasail to estimate iso-sensitive runtime
- $1.59/hour

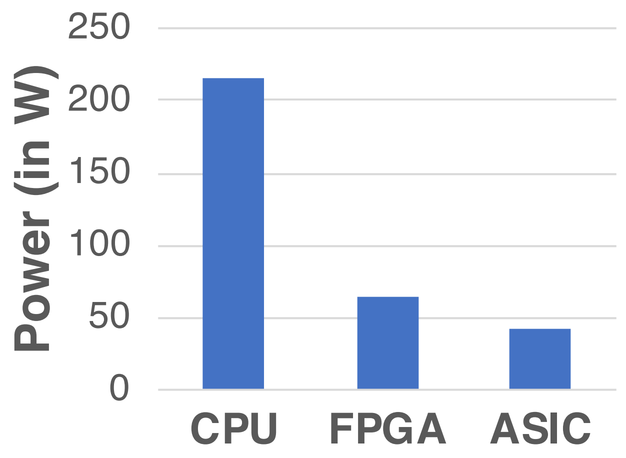
FPGA (Darwin-WGA)

- AWS f1.2xlarge instance
- 1 Xilinx Virtex Ultrascale+ FPGA (50 BSW and 2 GACT-X arrays with 32PEs)
- 8 vCPUs
- $1.65/hour

ASIC

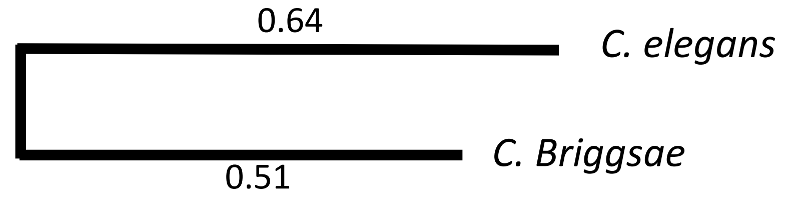
TSMC 40nm DC synthesis (not a chip prototype)

		Configuration	Area (mm^2)	Power (W)
BSW	Logic	64 x (64PE array)	16.6	25.6
GACT-X	Logic	12 x (64PE array)	4.2	6.72
	Traceback SRAM	12 x (64PE x 16KB/PE)	15.1	7.92
DRAM	DDR4-2400R	4 x 32GB	-	3.10
TOTAL			35.9	43.34

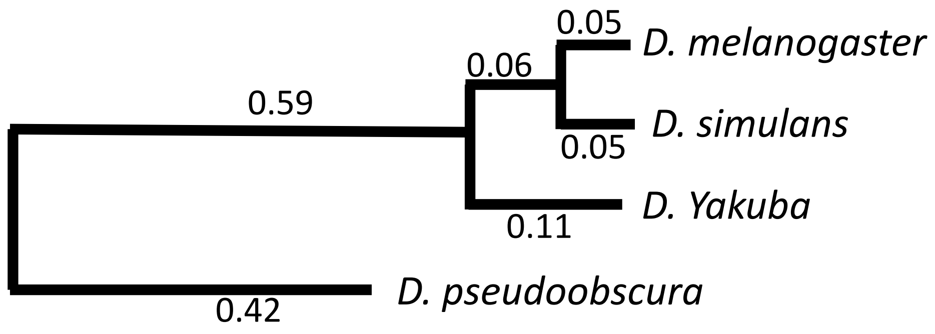


Species and Genome Assembly

Target Species	Size (Mbp)	Query Species	Size (Mbp)
C. elegans (ce11)	100	*C. briggsae* (cb4)	105
D. melanogaster (dm6)	137	*D. simulans* (droSim1)	110
		D. Yakuba (droYak2)	120
		D. pseudoobscura (dp4)	127



Roundworm family

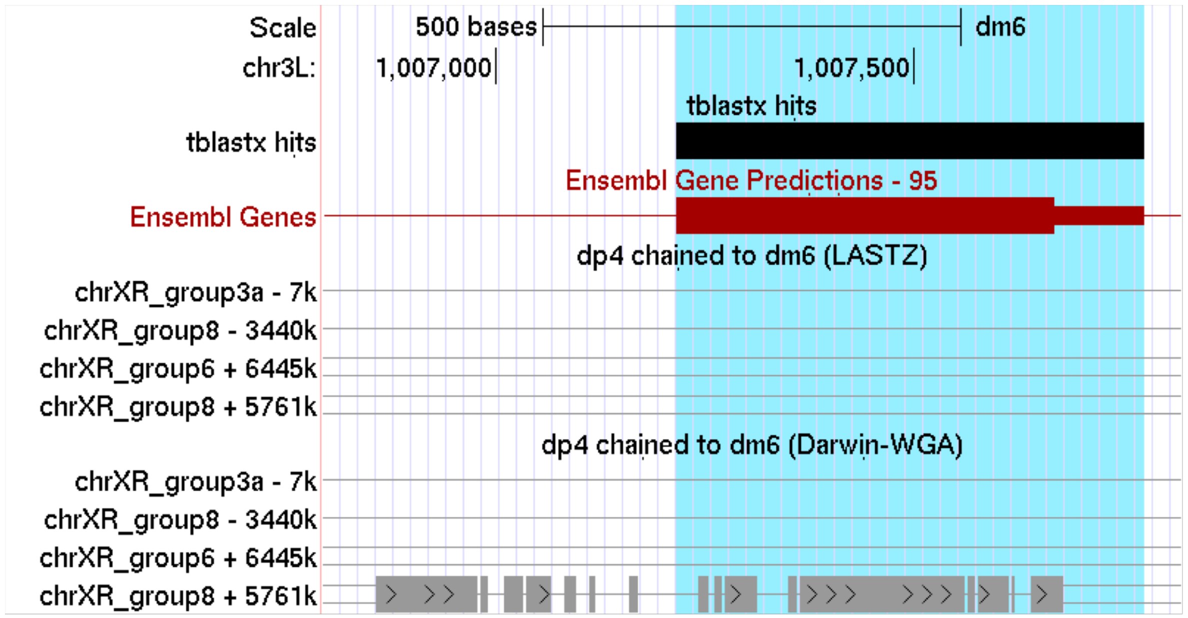


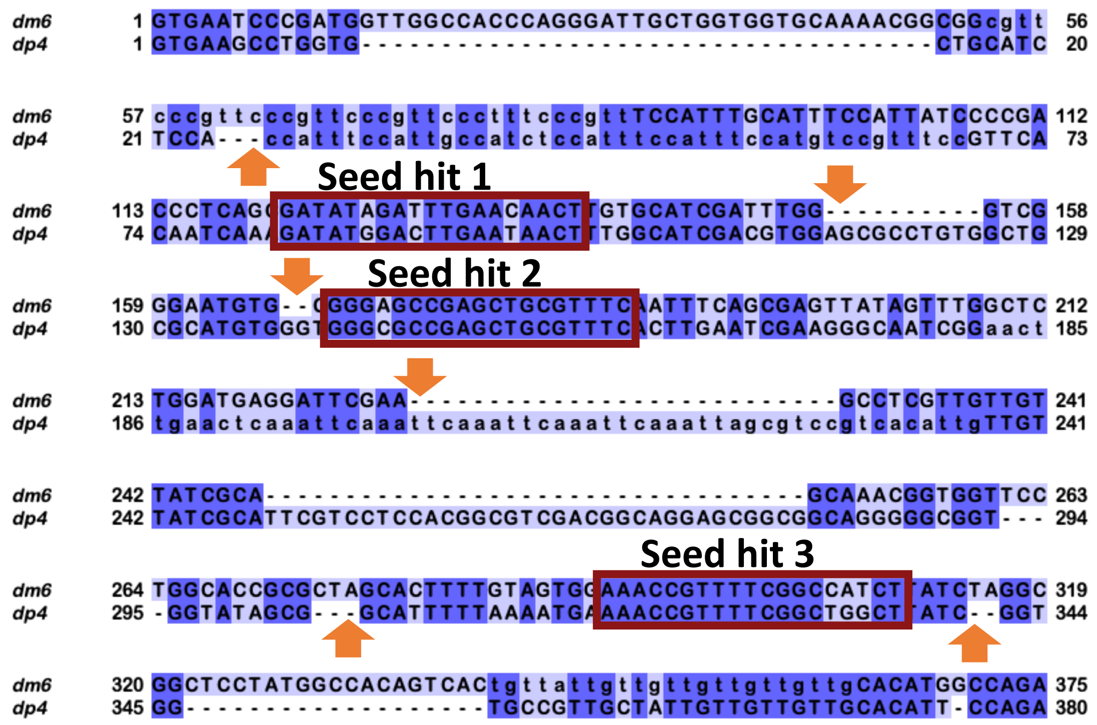
Fruit fly family

- dm6-droSim1 molecular distance comparable to human-monkey
- dm6-dp4 molecular distance comparable to human-chicken

Results

Darwin-WGA finds genes that LASTZ does not





```
dm6     1 GTGAATCCCGATGGTTGGCCACCCAGGGATTGCTGGTGGTGCAAAACGGCGGcgtt  56
dp4     1 GTGAAGCCTGGTG-------------------------------------CTGCATC  20

dm6    57 cccgttcccgttcccgttccctttcccgttTCCATTTGCATTTCCATTATCCCCGA 112
dp4    21 TCCA---ccatttccattgccatctccatttccatttccatgtccgtttccGTTCA  73

dm6   113 CCCTCAGCGATATAGATTTGAACAACTTGTGCATCGATTTGG----------GTCG 158
dp4    74 CAATCAAAGATATGGACTTGAATAACTTTGGCATCGACGTGGAGCGCCTGTGGCTG 129

dm6   159 GGAATGTG--CGGGAGCCGAGCTGCGTTTCAATTTCAGCGAGTTATAGTTTGGCTC 212
dp4   130 CGCATGTGGGTGGGCGCCGAGCTGCGTTTCACTTGAATCGAAGGGCAATCGGaact 185

dm6   213 TGGATGAGGATTCGAA-------------------------GCCTCGTTGTTGT 241
dp4   186 tgaactcaaattcaaattcaaattcaaattcaaattagcgtccgtcacattgTTGT 241

dm6   242 TATCGCA-----------------------------------GCAAACGGTGGTTCC 263
dp4   242 TATCGCATTCGTCCTCCACGGCGTCGACGGCAGGAGCGGCGGCAGGGGGCGGT--- 294

dm6   264 TGGCACCGCGCTAGCACTTTTGTAGTGGAAACCGTTTTCGGCCATCTTATCTAGGC 319
dp4   295 -GGTATAGCG---GCATTTTTAAAATGAAAACCGTTTTCGGCTGGCTTATC--GGT 344

dm6   320 GGCTCCTATGGCCACAGTCACtgttattgttgttgttgttgCACATGGCCAGA 375
dp4   345 GG-------------------TGCCGTTGCTATTGTTGTTGTTGCACATT-CCAGA 380
```

Indels (shown by arrows) around each seed hit – dropped by ungapped filtering (LASTZ) but retained by gapped filtering (Darwin-WGA)

Darwin-WGA Sensitivity Improvement Versus LASTZ

Species pair	Top-10 Alignment Chain Scores	Matching Base-pairs within Alignments	Number of Aligning Exons (protein-coding genes)
dm6-droSim1	+0.03%	1.25x	+0.20%
dm6-droYak2	+0.05%	1.41x	+0.09%
dm6-dp4	+1.86%	1.42x	+0.41%
ce11-cb4	+5.73%	3.12x	+2.70%

Top-10 Alignment Chain Scores: Represent orthologous sequences (derived from "speciation")

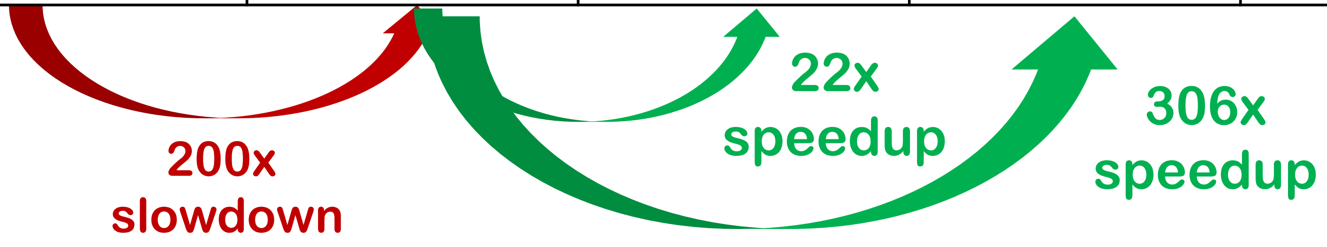
Matching Base-pairs within Alignments: Represent paralagous sequences (derived from "duplication")

Number of Aligning Exons: Represent functionally relevant orthologous sequences, under some selective pressure (at least in the target species)

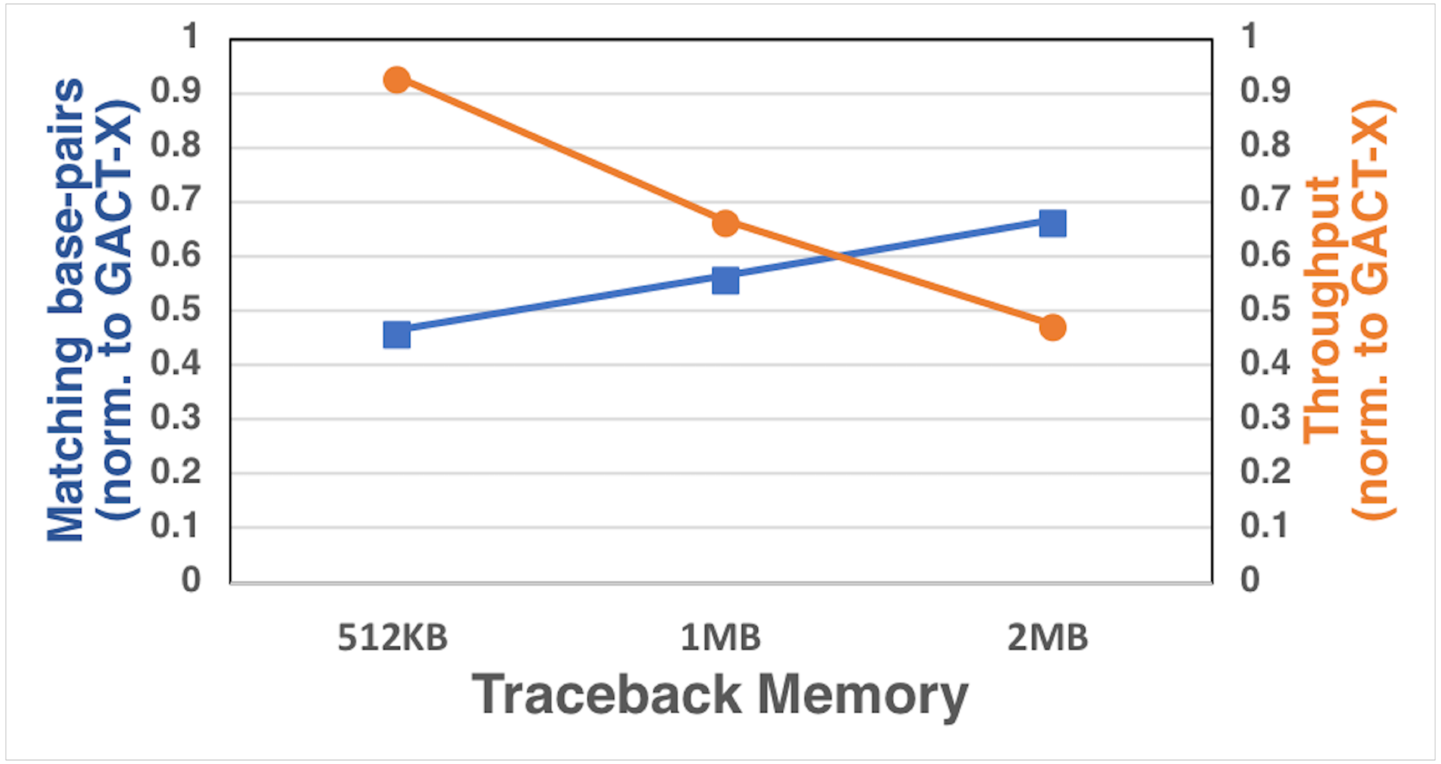
False positive rate (2-mer shuffled genome): 0.0007%

Runtime and Cost Comparison

Species pair	LASTZ runtime (sec)	Iso-sensitive s/w runtime (sec)	Darwin-WGA runtime (sec)		Darwin-WGA Improvement	
			FPGA	ASIC	FPGA (Perf/$)	ASIC (Perf/W)
ce11-cb4	481	64,960	3,823	219	19.1x	1,478x
dm6-droSim1	643	142,627	5,936	461	23.2x	1,547x
dm6-droYak2	654	144,454	6,001	469	23.2x	1,539x
dm6-dp4	557	125,700	4,987	404	24.3x	1,553x



GACT-X uses 3x less space and time as compared to GACT



Summary

- Darwin-WGA replaces ungapped filtering in LASTZ by Banded Smith-Waterman algorithm for higher sensitivity
  - up to 3x matching base-pairs
  - up to 5.7% more orthologs
  - up to 2.1% more exons
- Darwin-WGA outperforms iso-sensitive software
  - FPGA: 24x performance/$ improvement
  - ASIC: 1,500x performance/Watt improvement
- GACT-X provides 3x improvement in speed and storage efficiency compared to GACT

Thank You!