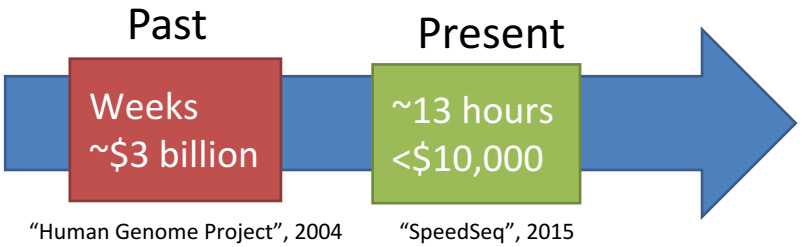
Accelerating Genomic Sequence Alignment Workloads with SVE—the Scalable Vector Extension

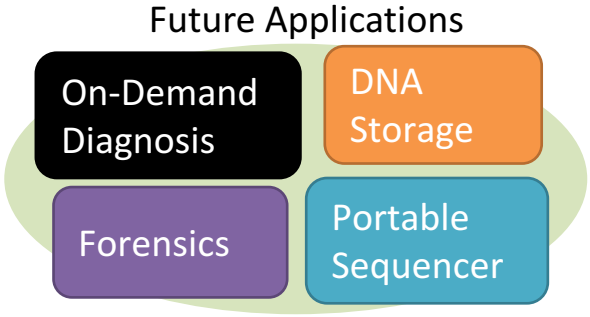
Trevor Mudge
Bredt Family Professor of Computer Science and Engineering
The University of Michigan, Ann Arbor

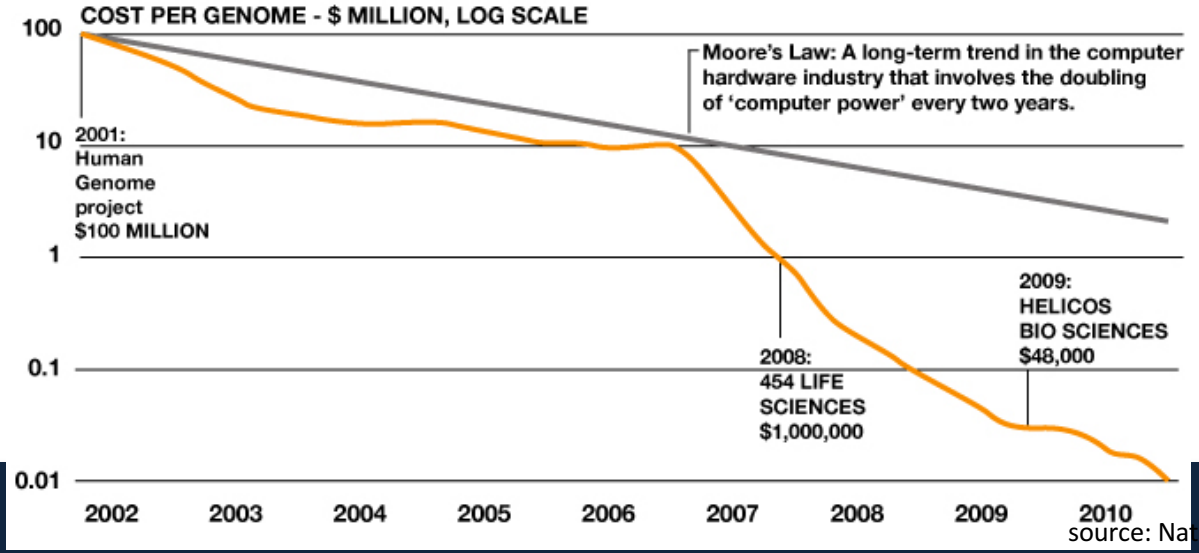
Outline

- Motivation:
  - Emerging Application: Genomics
  - ARM's Scalable Vector Extension
- Improving Sequence Alignment
  - Smith-Waterman: Batch, Sliced, and Wavefront
- Experimental Setup
- Results and Conclusion

Computational Biology







source: National Institute of Health

Human Genome:
3.2 billion base pairs

Need to sample at 30-50x coverage

Whole Genome Sequencing Pipeline

10-20k
in length

reference gene

reconstructed sequence

Read/Extract Sequences

- Reading fragment samples of whole genome
- Signal/Image processing

Sequence Alignment

- Matching overlaps across multiple sequences
- Dynamic vs heuristic algorithm

Assembly

- Reconstructing the original sequence
- de-novo vs mapping assembly

Analysis

- Identifying gene variants and abnormalities
- Pattern matching, HMM, DNN

Target Architecture:

Scalable Vector Extension

ARM's Scalable Vector Extension (SVE)

- Designed to complement existing SIMD architecture (NEON)

- Key Features:
  - Scalable Vector Length (128, 256, 512, and 1024-bits)
  - Per-lane Predication (32 SIMD Reg. + 16 Predicate Reg.)
  - Gather-load and scatter-store
  - Horizontal vector operations

Vector Length Agnostic Code

ARM's Scalable Vector Extension (SVE)

- Genomic sequences are sampled at different lengths depending on the device used for sampling:
  - Illumina HiSeq System: 30-300 bps
  - Sanger 3730xl: 400-900 bps

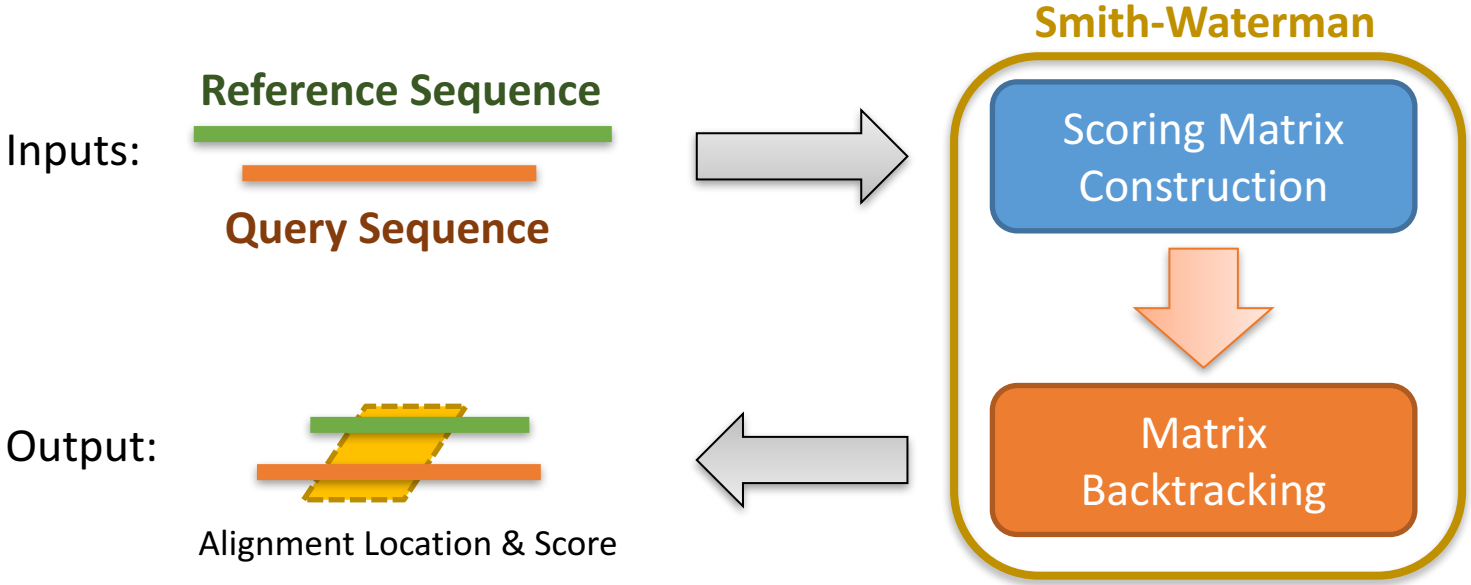
Vector-Length Agnostic Code can be used to **Dynamically Choose the Optimal SIMD Width**

Target Algorithm:

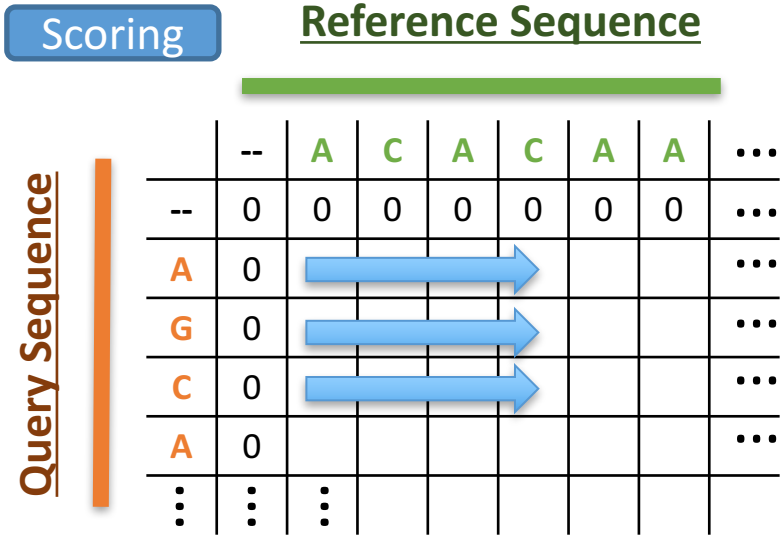
Smith-Waterman
Sequence Alignment

Smith-Waterman Algorithm

Local sequence alignment algorithm developed in 1981



Scoring Matrix Construction



$$H(m,n) = max \begin{cases} E(m,n) \\ F(m,n) \\ H(m-1,n-1) + S(a_m, b_n) \end{cases}$$

$$E(m,n) = max \begin{cases} H(m,n-1) - g_o \\ E(m,n-1) - g_e \end{cases}$$

$$F(m,n) = max \begin{cases} H(m-1,n) - g_o \\ F(m-1,n) - g_e \end{cases}$$

Scoring Matrix Construction

Scoring

Reference Sequence

Query Sequence

	--	A	C	A	C	A	A	…
--	0	0	0	0	0	0	0	…
A	0	2	1	2	1	2	2	…
G	0	1	1	1	1	1	1	…
C	0	0	3	2	3	2	1	…
A	0	2	2	5				…
⋮	⋮	⋮						

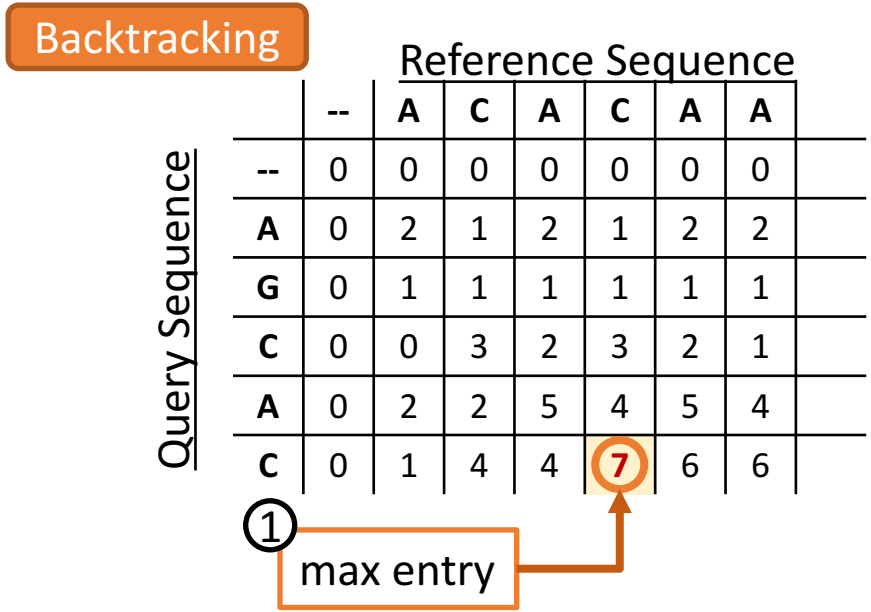
$$H(m,n) = max \begin{cases} E(m,n) \\ F(m,n) \\ H(m-1,n-1) + S(a_m, b_n) \end{cases}$$

$$E(m,n) = max \begin{cases} H(m,n-1) - g_o \\ E(m,n-1) - g_e \end{cases}$$

$$F(m,n) = max \begin{cases} H(m-1,n) - g_o \\ F(m-1,n) - g_e \end{cases}$$

Backtracking

Finds the best local alignment from the scoring matrix

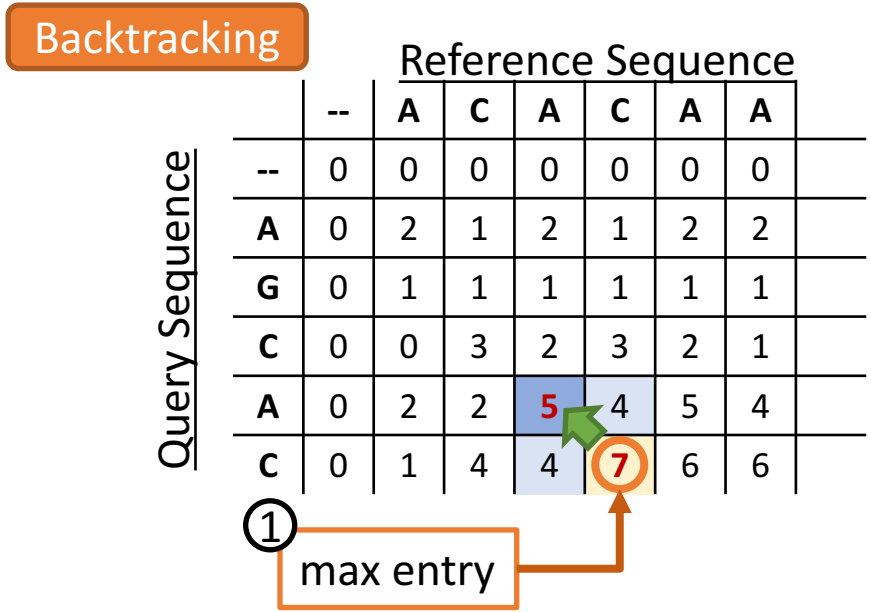


Step 1.

Search through the matrix and find the entry with the largest score

Backtracking

Finds the best local alignment from the scoring matrix

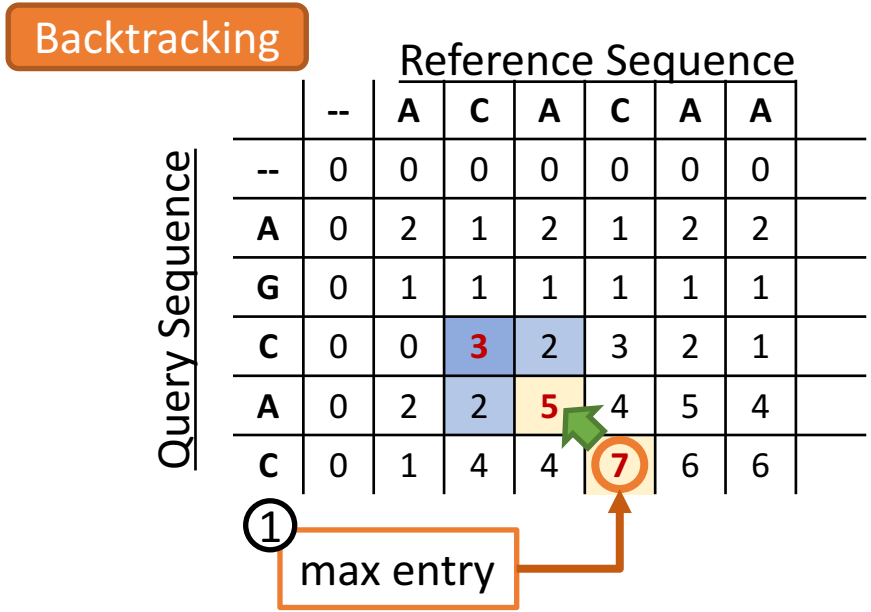


Step 2.

Check the adjacent entries for the next largest score

Backtracking

Finds the best local alignment from the scoring matrix



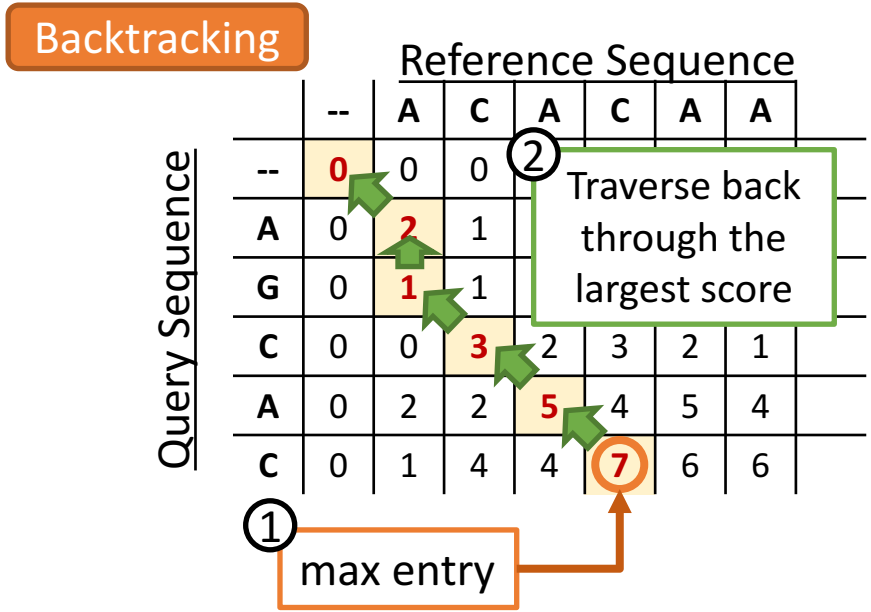
Step 2.

Check the adjacent entries for the next largest score

Move to the entry with the largest score and continue the path

Backtracking

Finds the best local alignment from the scoring matrix



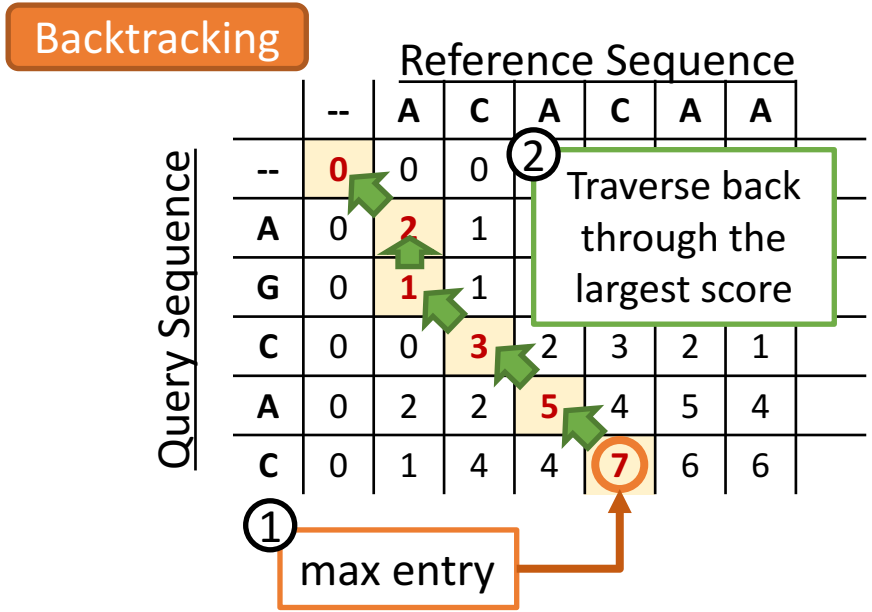
Step 2.

Check the adjacent entries for the next largest score

Move to the entry with the largest score and continue the path

Backtracking

Finds the best local alignment from the scoring matrix



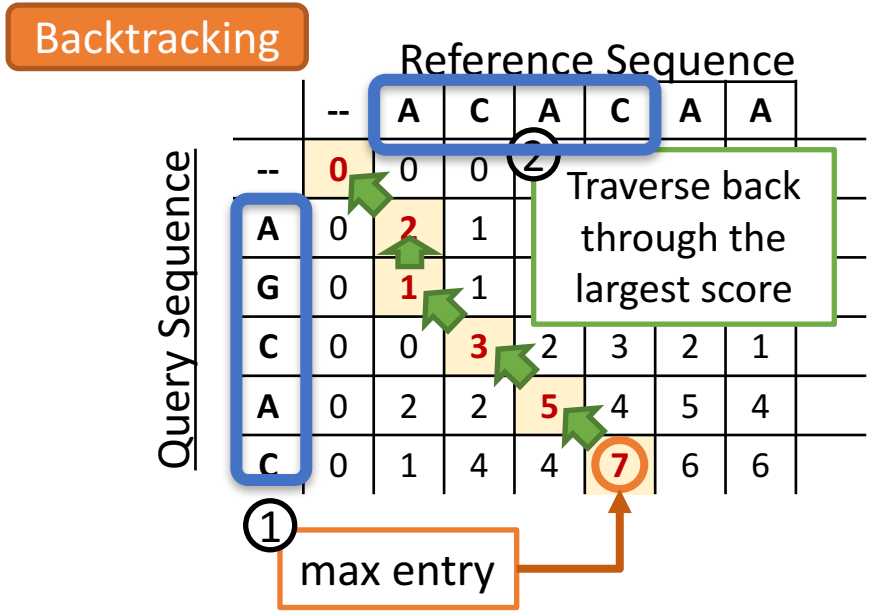
Step 3.

Get the resulting alignment

Path Direction	Alignment
Horizontal	Deletion
Vertical	Insertion
Diagonal	Match

Backtracking

Finds the best local alignment from the scoring matrix



Step 3.

Get the resulting alignment

③

Reference: A–CAC

Query: AGCAC

Insertion

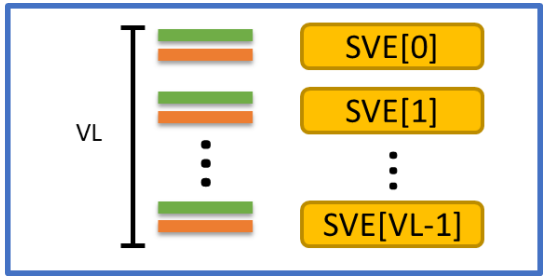
Alignment Score: 7

Smith-Waterman Vectorization:

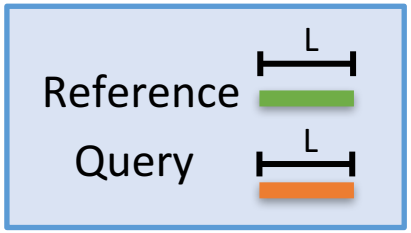
Batch, Sliced, and Wavefront

Vectorization

BATCH:

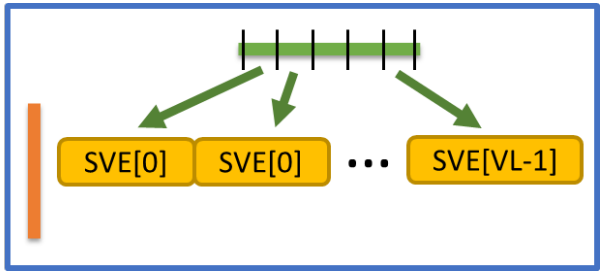


$$VL = \frac{\text{SIMD\_WIDTH}}{32 \text{ bits}}$$



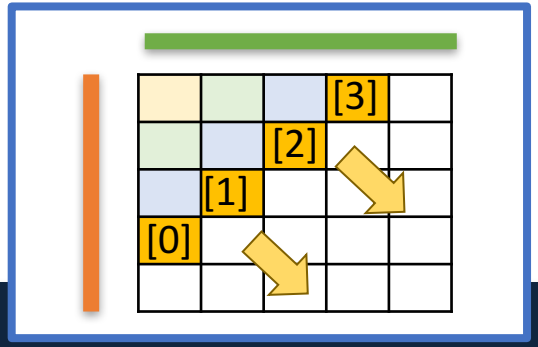
X 1000 samples

SLICED:

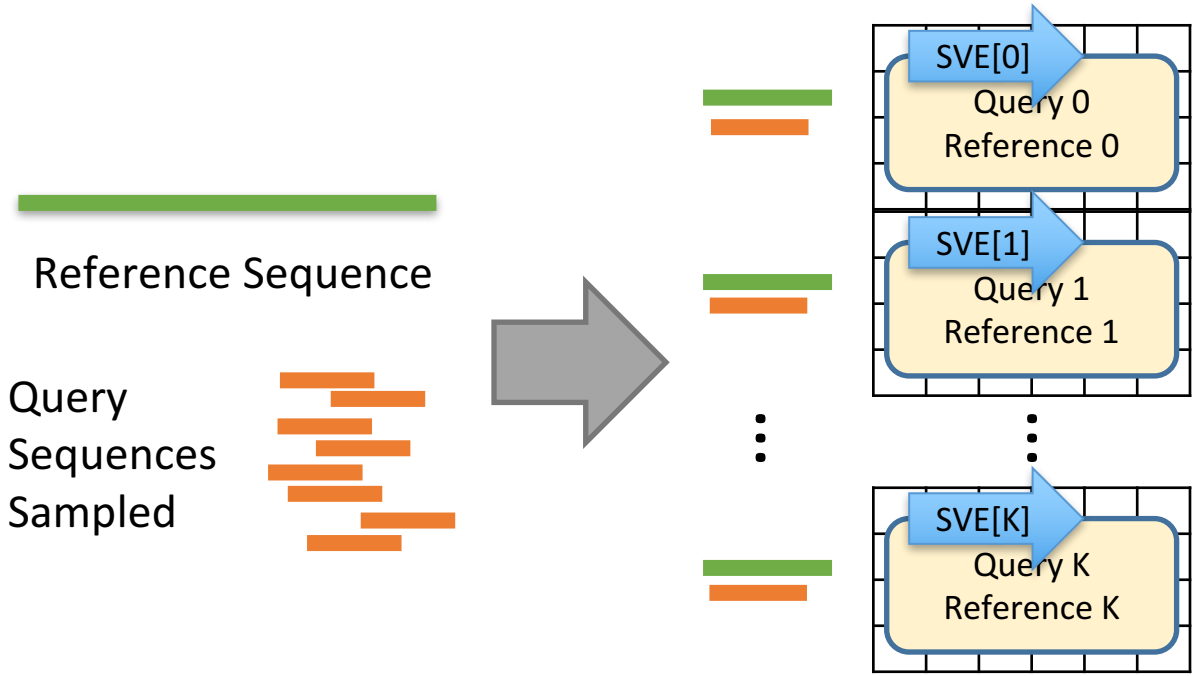


Alignment Location & Score

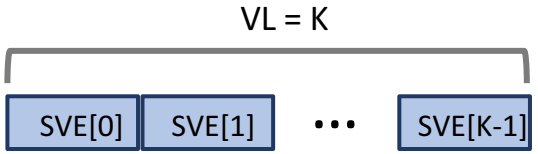
Wavefront:

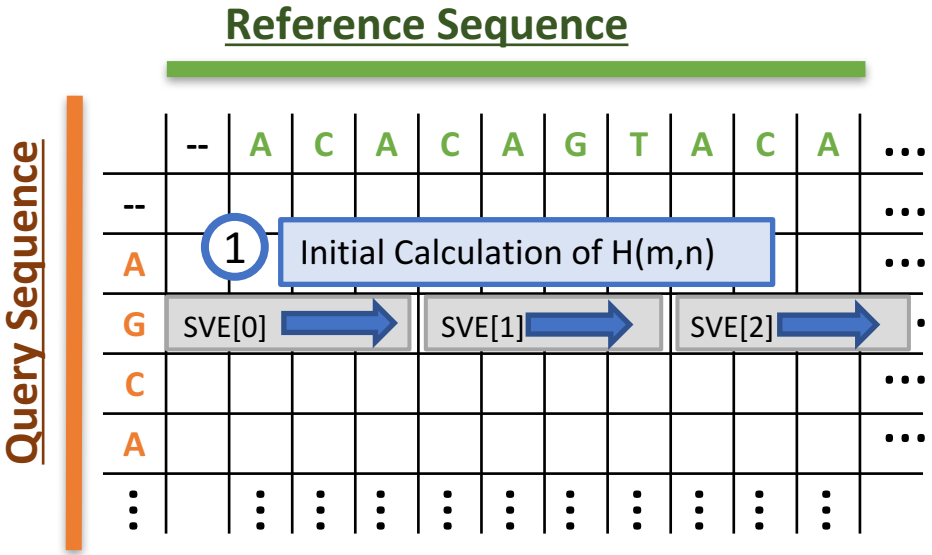


Batch Smith-Waterman

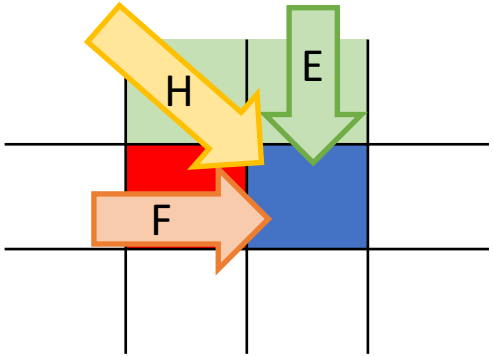


Sliced Smith-Waterman

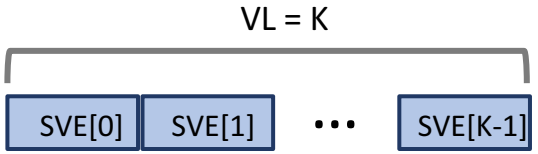


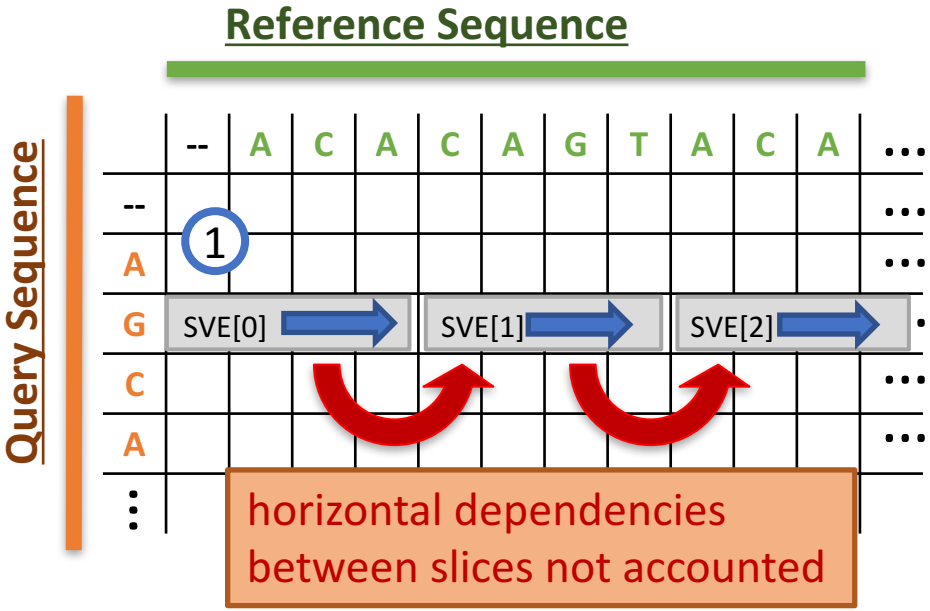


$$H(m,n) = max \begin{cases} E(m,n) \\ F(m,n) \\ H(m-1,n-1) + S(a_m, b_n) \end{cases}$$

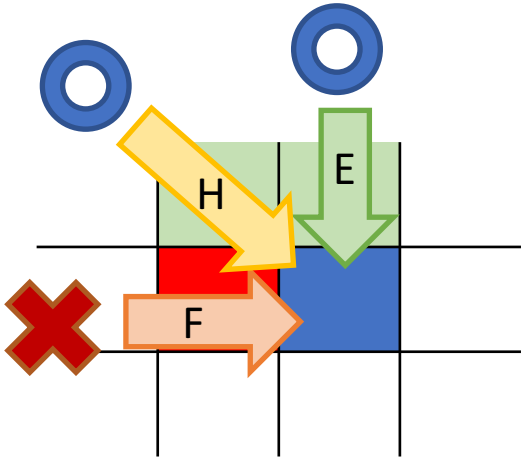


Sliced Smith-Waterman

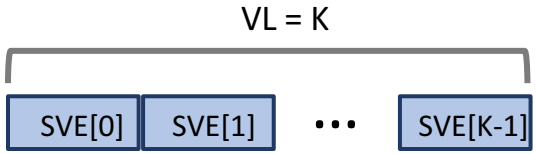


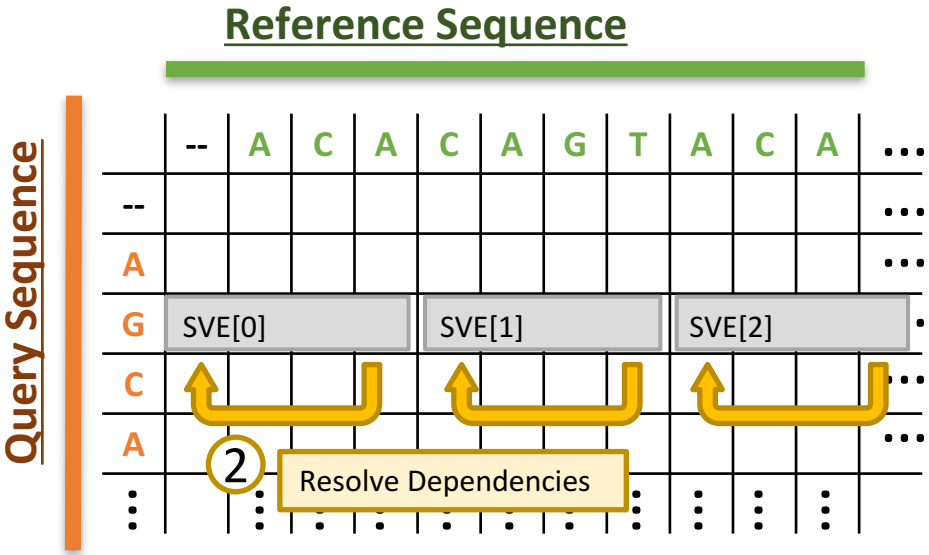


Value of F need to be re-calculated



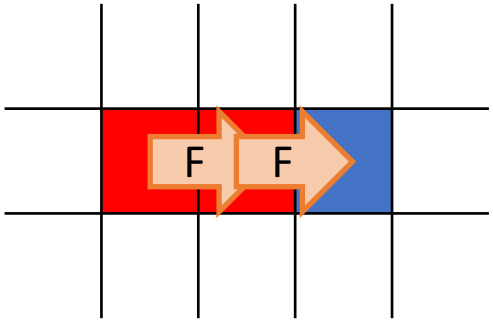
Sliced Smith-Waterman





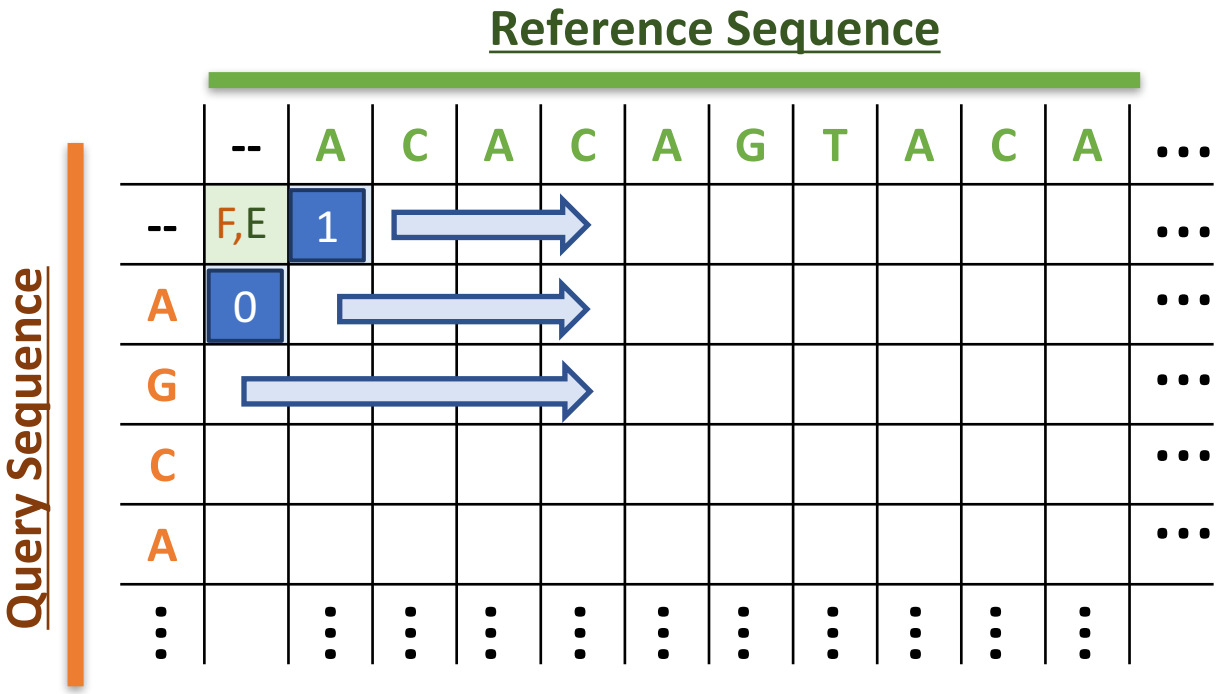
$$F(m,n) = max\begin{cases} H(m-1,n) - g_o \\ F(m-1,n) - g_e \end{cases}$$

$$H(m,n) = \max\left(F(m,n),\ H(m,n)\right)$$

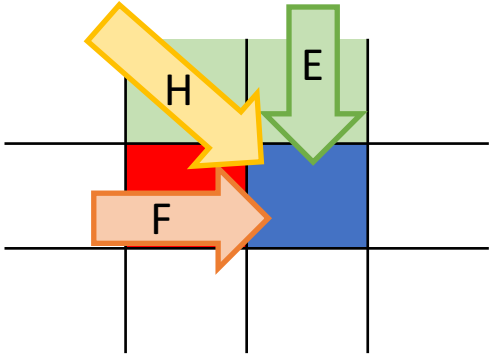


Costly resolution as one need to traverse through the entire row

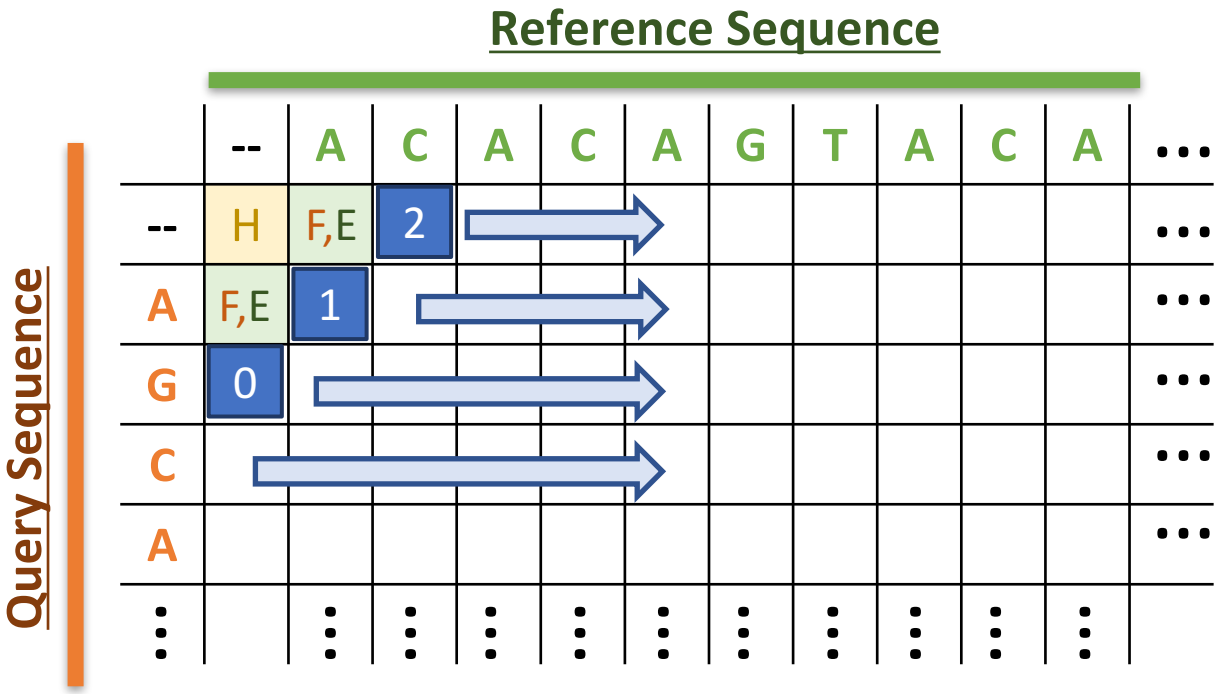
Wavefront Smith-Waterman



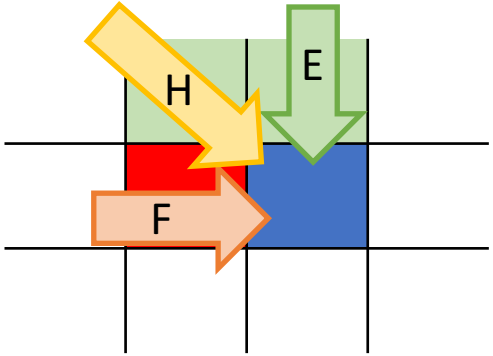
All dependency comes from previous execution



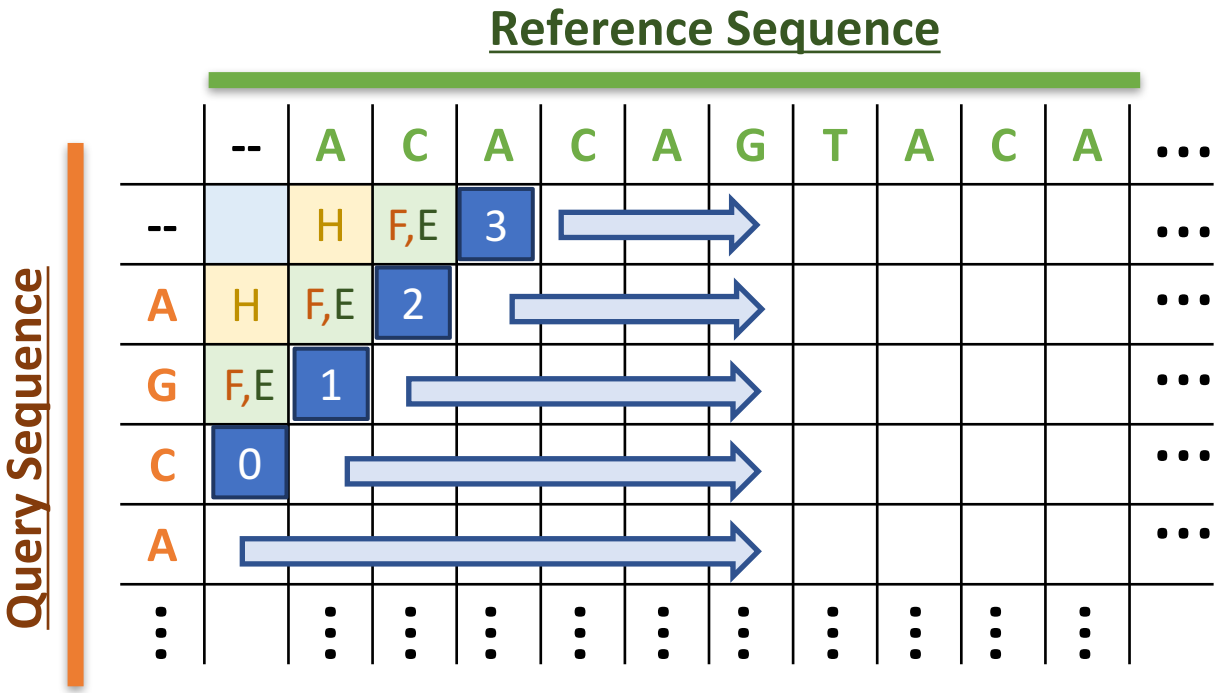
Wavefront Smith-Waterman



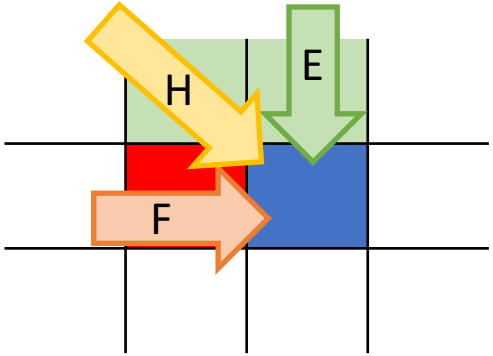
All dependency comes from previous execution



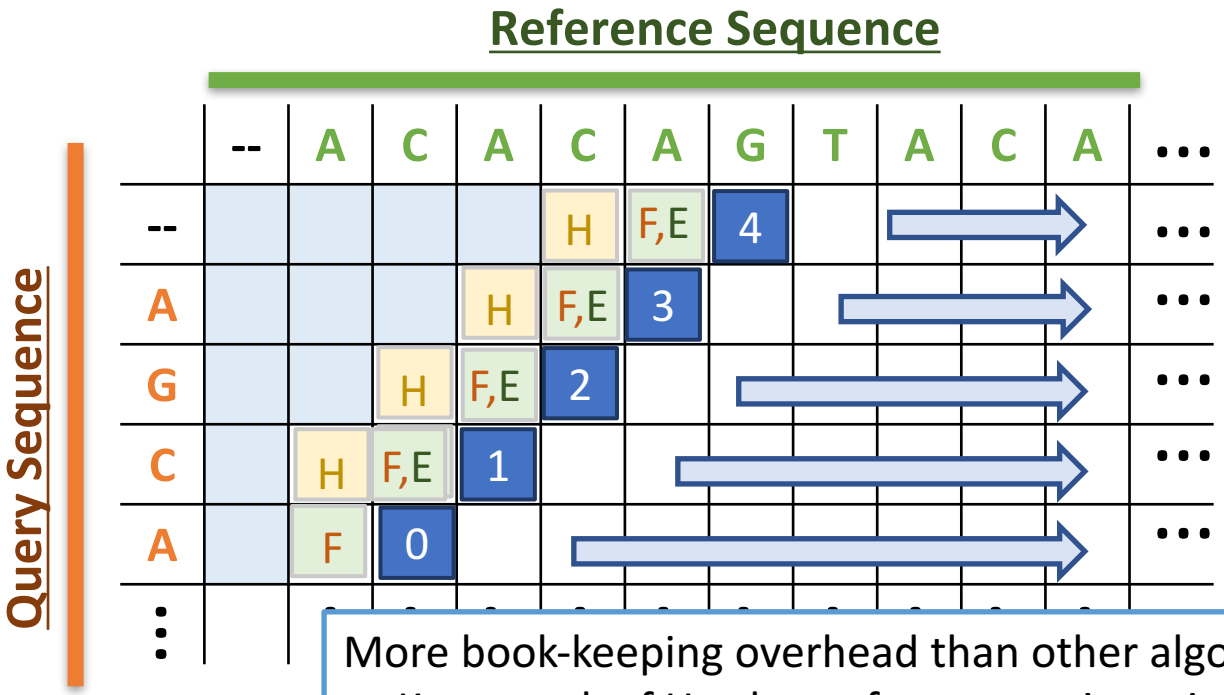
Wavefront Smith-Waterman



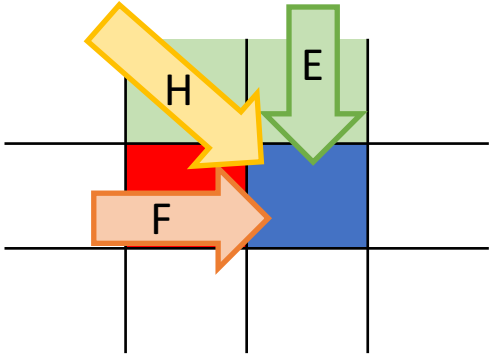
All dependency comes from previous execution



Wavefront Smith-Waterman



All dependency comes from previous execution



More book-keeping overhead than other algorithms:

- Keep track of H values of two prev. iterations
- F and E values from prev. iteration

Experimental Evaluation:

Smith-Waterman on gem5 w/ SVE

Experimental Setup

Gem5 Simulator w/ ARM SVE Simulation

Component	Configuration
Core	Single-Core out-of-order 64-bit ARM, 1GHz, 8-issue SIMD Width: 128-bit (NEON), 128/256/512/1024-bit (SVE)
Cache	32KB private L1 instruction cache, 2-way associative 64KB private L1 data cache, 2-way associative 4MB private L2 inclusive cache, 8-way associative
DRAM	Capacity: 8GB Latency: 30 ns Memory Controller Bandwidth: 12.8 GB/s

Experimental Setup

Application:

Smith-Waterman – Batch, Sliced, and Wavefront

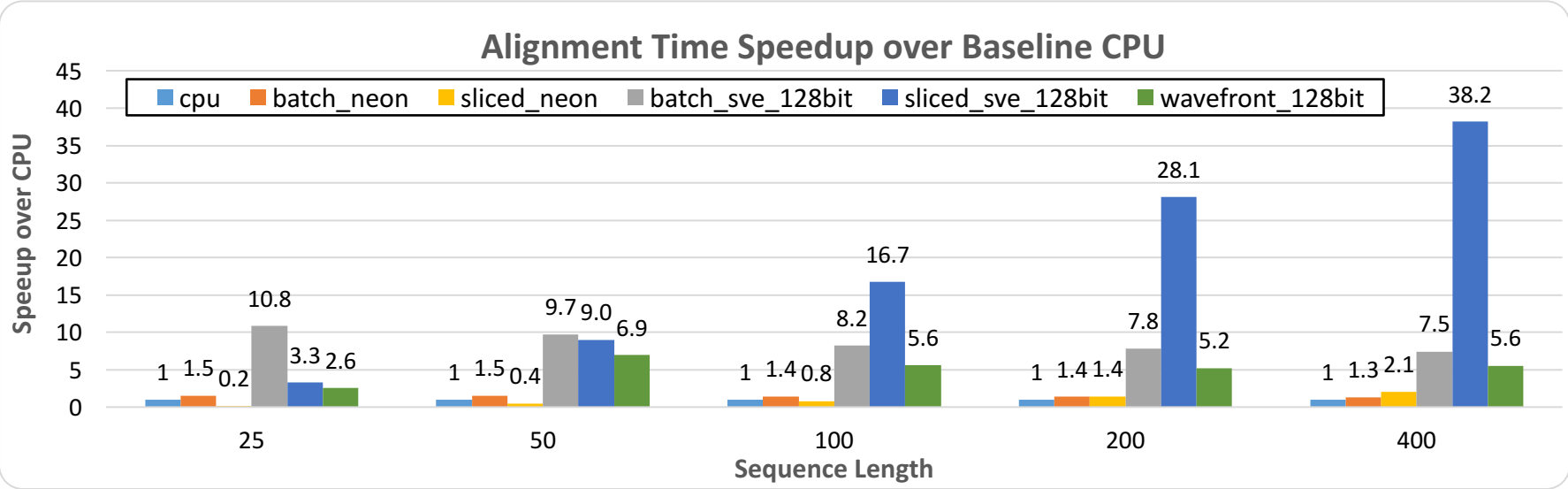
- Reference :

  25-400 bps samples from *E. Coli 536* Gene (4.9 Mbps)
- Query :

  1000 x 25-400 bps samples through WGSim

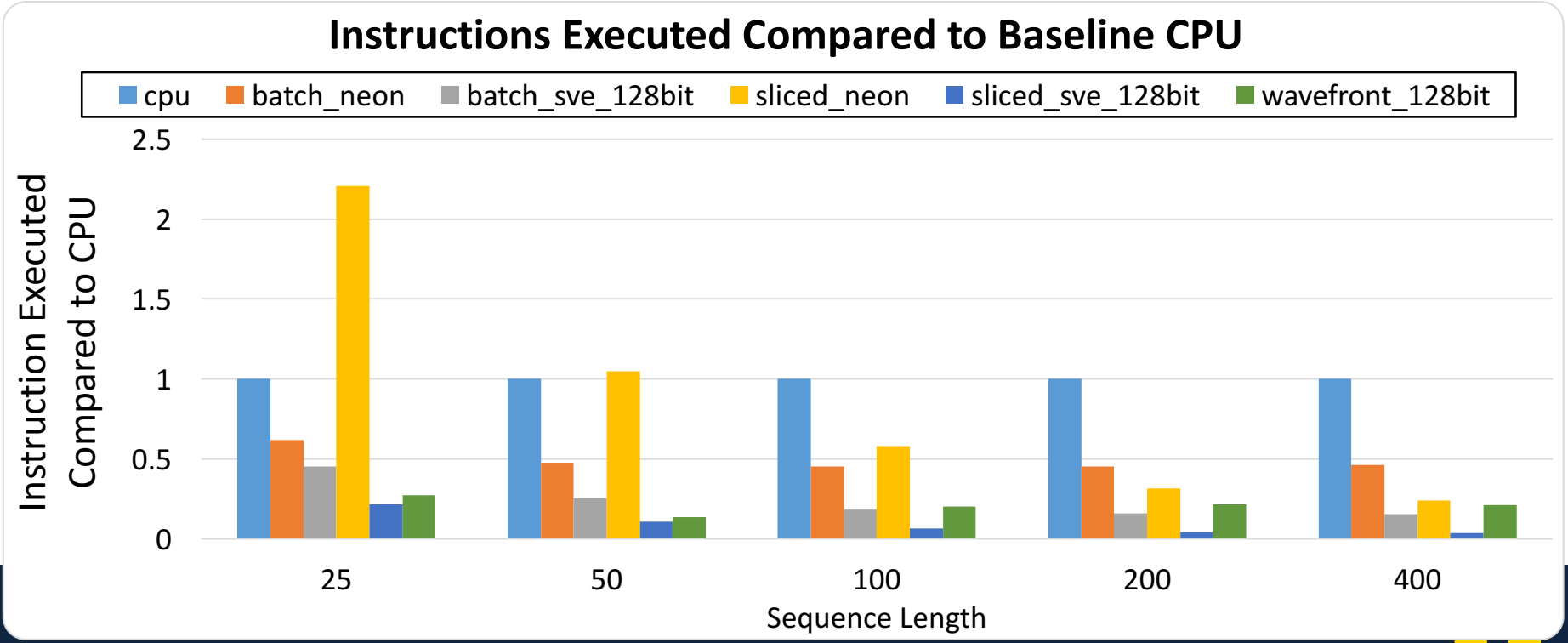
Advantage of SVE over Traditional System

- CPU, NEON implementation written in C. SVE hand-written in assembly.
- SVE outperforms both CPU and NEON implementations by at least 3x
- Batch, Sliced use 16-bit vectors. Wavefront use 64-bit vectors.



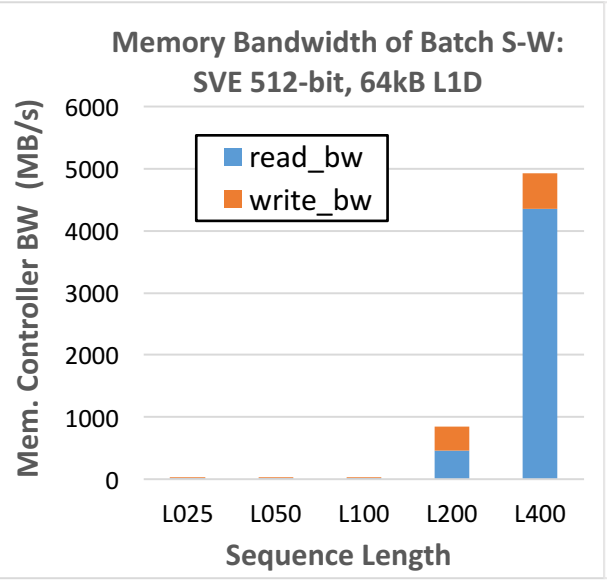
Advantage of SVE over Traditional System

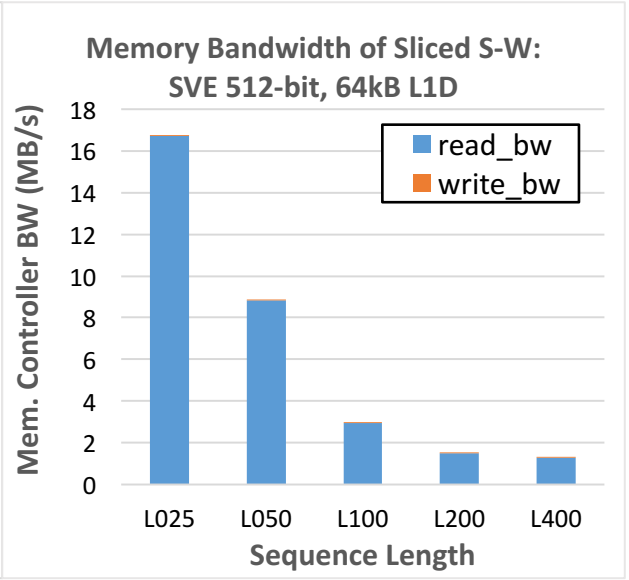
- SVE reduces the instruction execution significantly compared to CPU or NEON

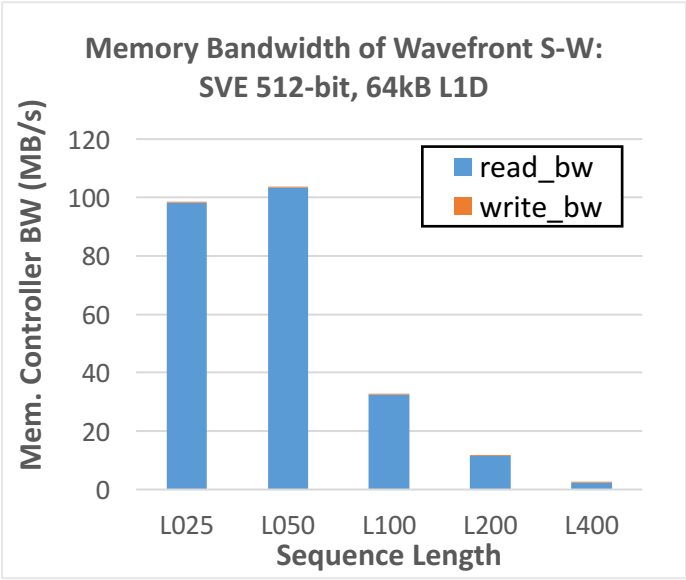


Memory Bandwidth Comparison

- Sliced and Wavefront significantly reduce the memory bandwidth compared to the Batch algorithm

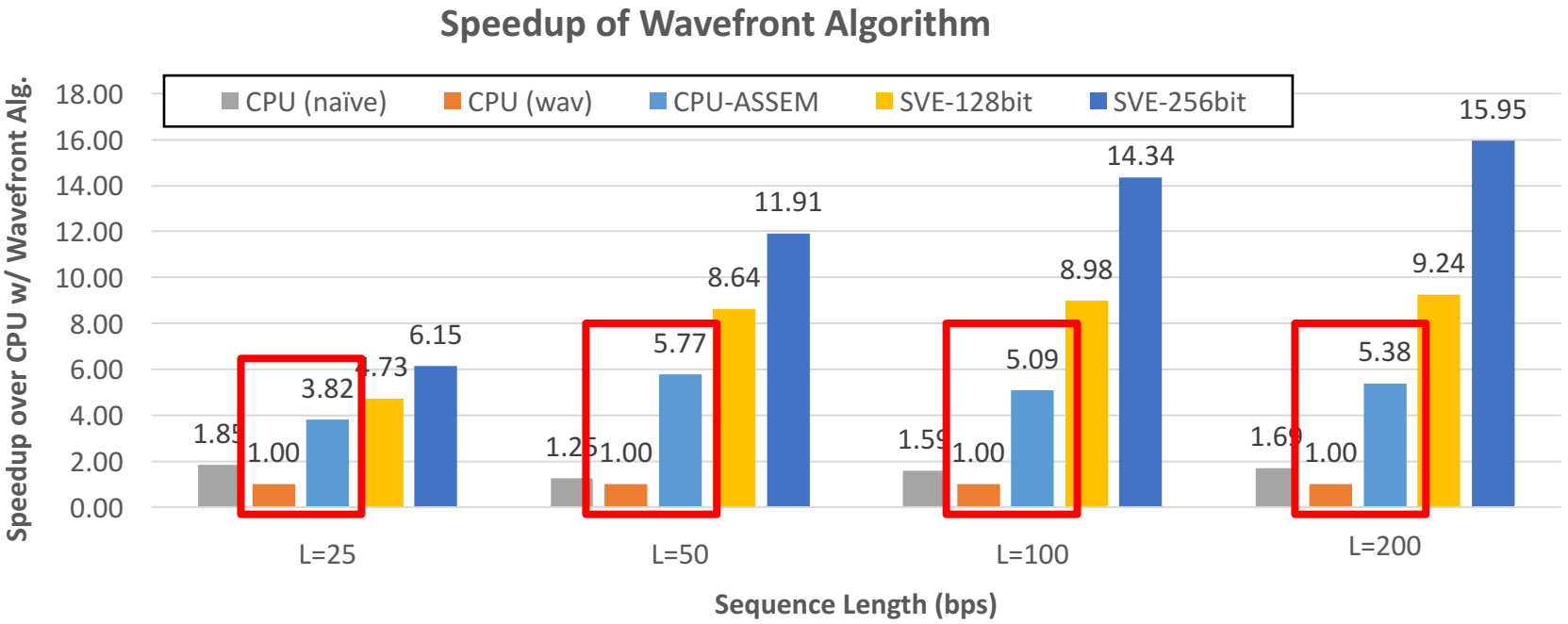




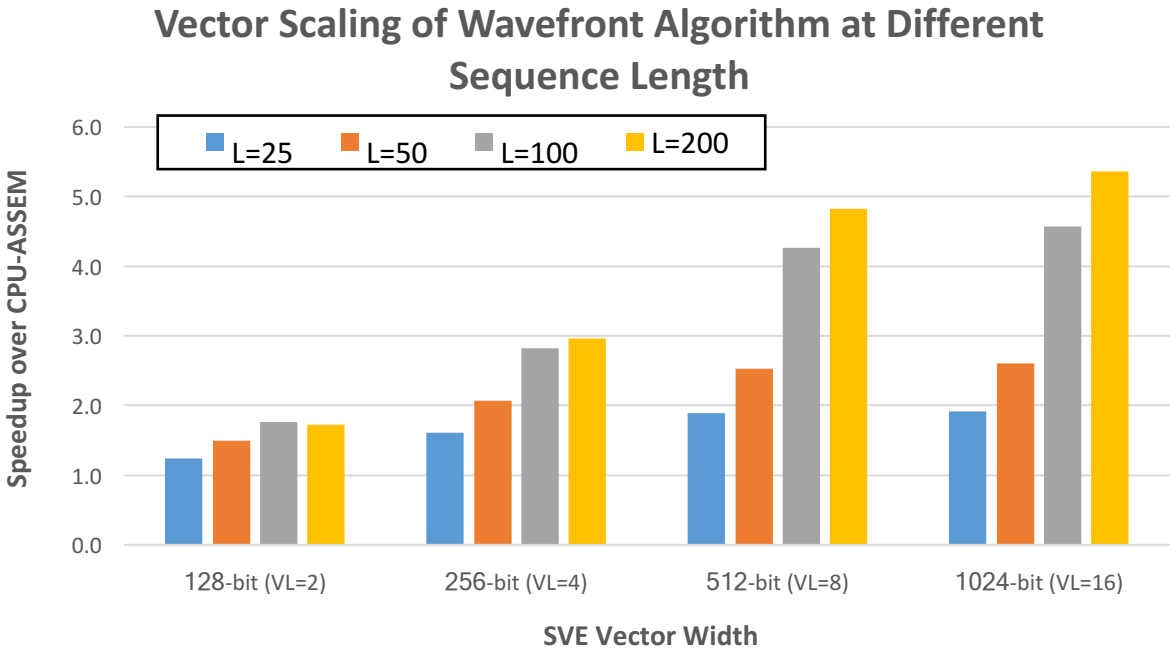


Performance Breakdown of Wavefront Algorithm

- Hand-written assembly code of Wavefront Algorithm has 4-6x speedup over C code.



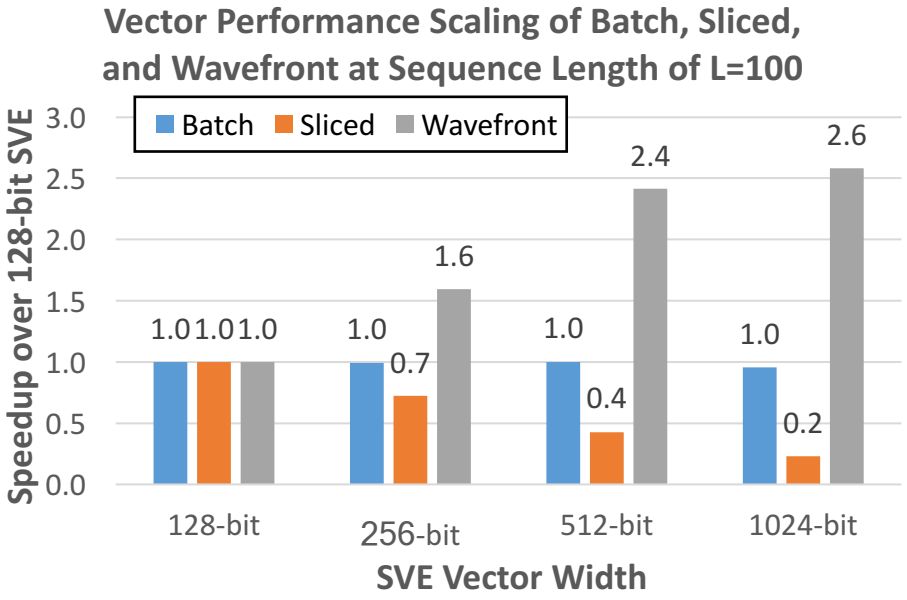
Waveform Algorithm – Vector Scaling

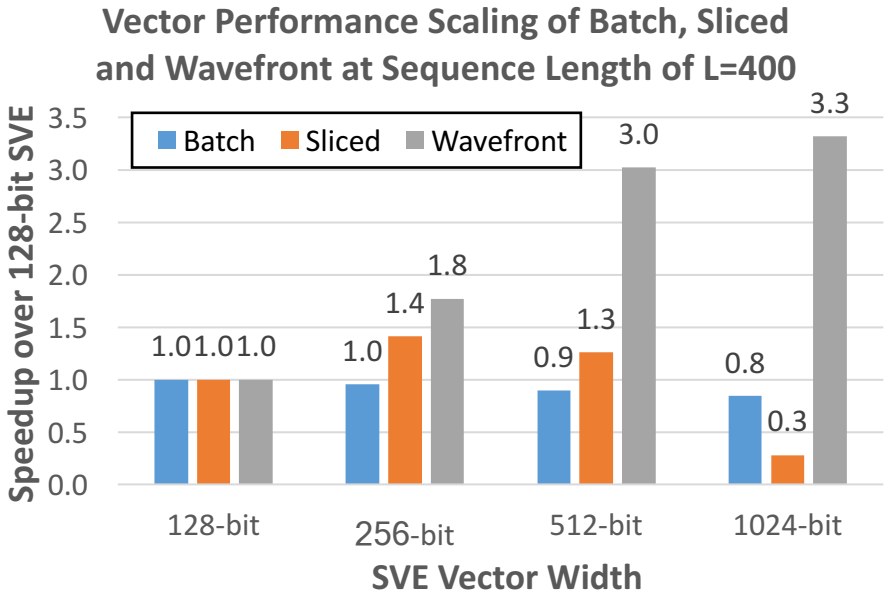


- Wavefront algorithm shows good performance scaling with increased vector length.
- Longer sequences have better utilization of wide vector lanes, but with diminishing return.

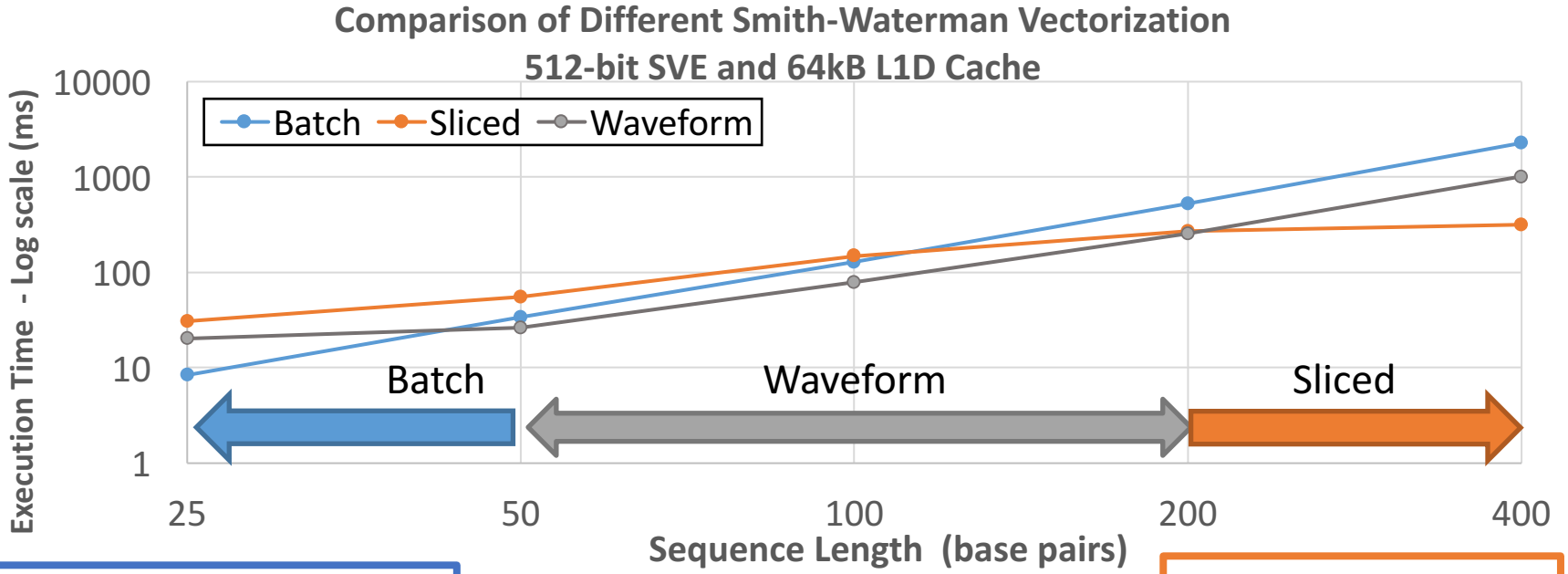
Vector Scaling of Different Algorithms

- Batch and Sliced show marginal improvement with increasing vector length
- Difficult to keep up with increased memory demand
- Need to resolve dependencies.





Fixed HW Options: Batch vs Sliced vs Waveform @512-bit



Batch
- Low overhead.
- Poor scaling.
- Fastest for short seq.

Waveform
- Efficient use of vector Lanes
- Fastest for medium seq.

Sliced
- High overhead.
- Execution bypassing
- Fastest for long seq.

HW with Variable Vector Length

- Given freedom to choose the hardware for each sequence length, we can establish a set of optimal algorithm-hardware pair.

Read Length	Algorithm	Vector Length	Speedup Over 512-bit Wavefront
< 50 bps	Batch	128-bit	2.77
50-100 bps	Wavefront	1024-bit	1.03
100-400 bps	Sliced	256-bit	1.23-3.06

Conclusion

Smith-Waterman on SVE:

+ **Select Optimal Vector Length & Algorithm depending on Input**

+ **Lower Instruction Footprint**

- Improvements to memory controller can lead to improved performance
- Wavefront algorithm use 64-bit vectors due to limitations on gather-scatter instruction addressing.

Key References

Smith TF, Waterman MS, "*Identification of common molecular subsequences*" J Mol Biol 147

Farrar M, "*Striped Smith-Waterman speeds database searches six times over other SIMD implementations*" Bioinformatics 23

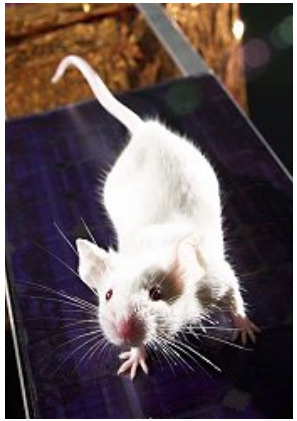
Zhao M, Lee W, Garrison E., Marth G. "*SSW Library: An SIMD Smith-Waterman C/C++ Library for Use in Genomic Applications*"

Li H, Durbin R. "*Fast and accurate short read alignment with Burrows-Wheeler transform*" Bioinformatics 25

Steinfadt S. "*SWAMPT+: Enhanced Smith-Waterman Search for Parallel Models*"

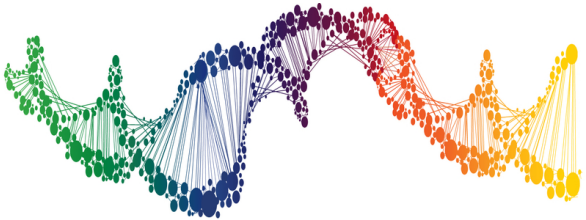
Questions?

DNA Sequencing

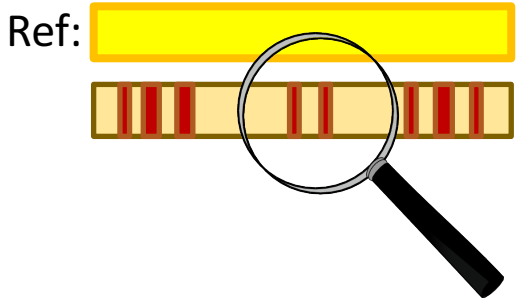


Subject Organism

Genomic Code



Detect Abnormalities



Human Genome:
3.2 billion base pairs

Need to sample at 30-50x coverage

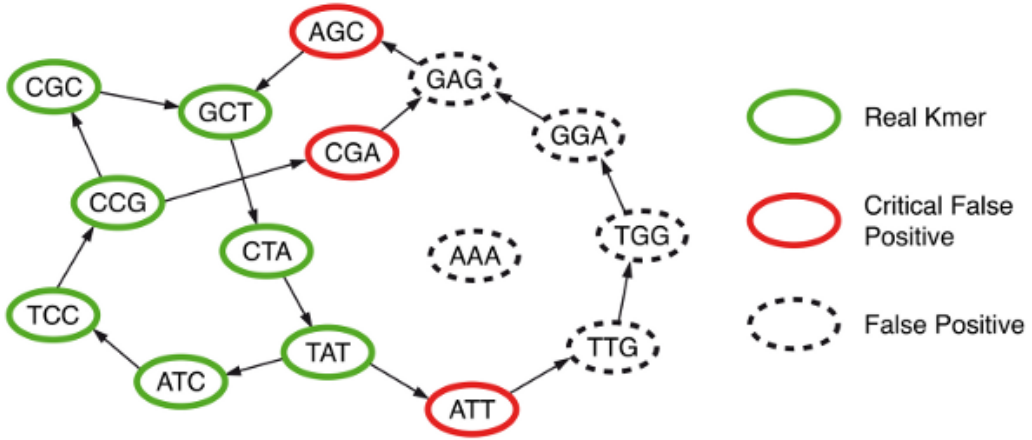
source: National Institute of Health

Future Work

Sequence Assembly: de novo assembly

Assembling the query samples without aligning them to a reference

Construct and traverse the De Bruijn graph



Target Genome (Unknown)

ATGCTATGCGT

Sampling

ATGCTA
CTATGC
ATGCGT

Sampled Sequences

String processing

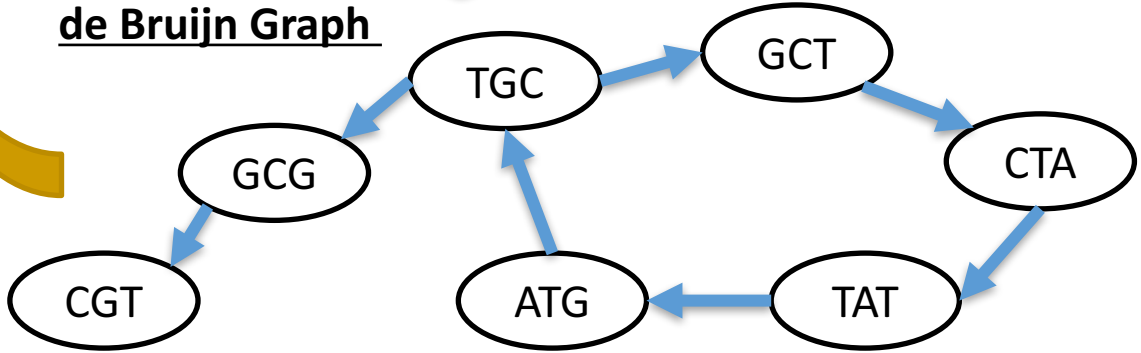
K-mers (k=4):

ATGC TGCT GCTA CTAT
TATG ATGC TGCG GCGT

Graph Traversal

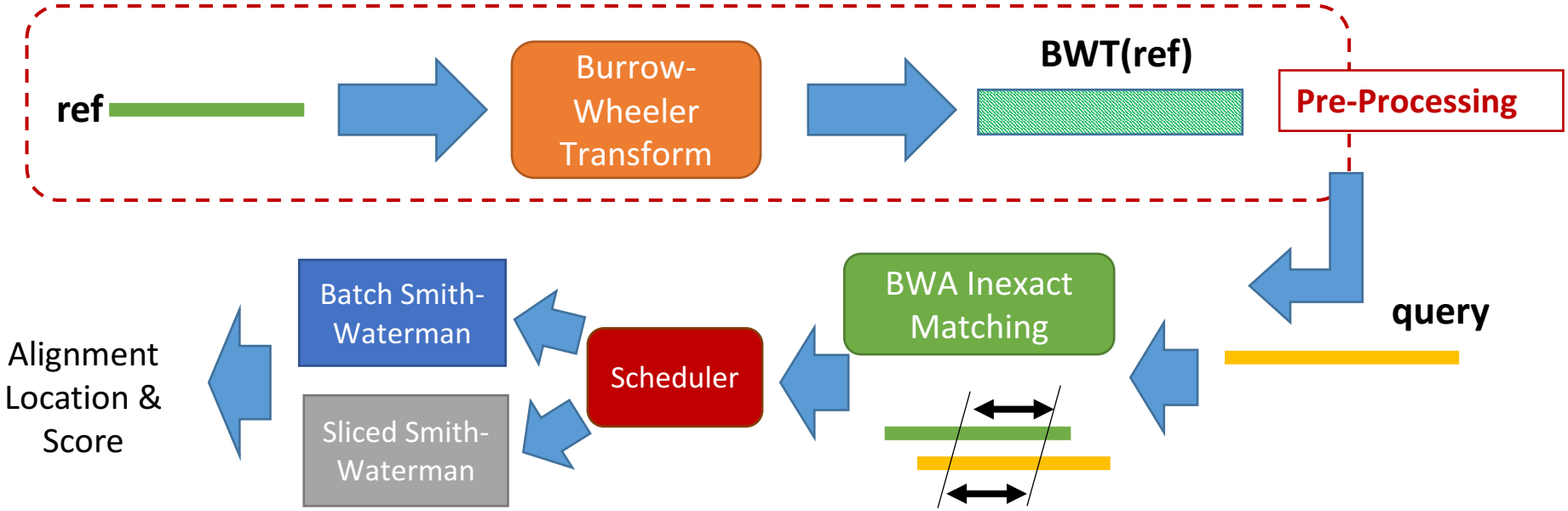
Graph Construction

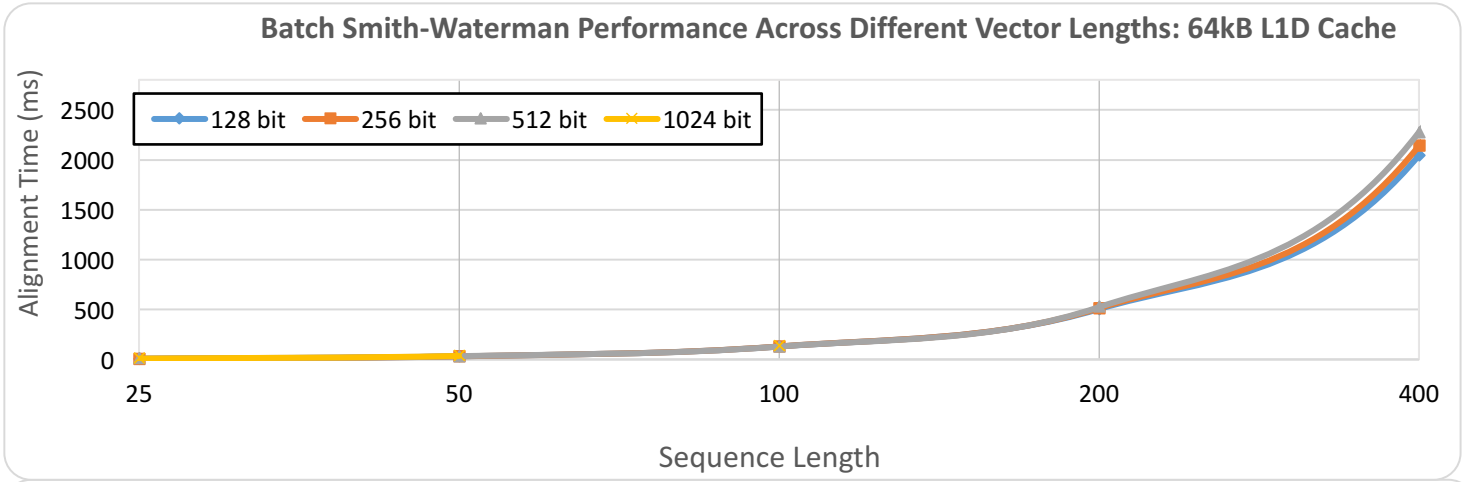
de Bruijn Graph

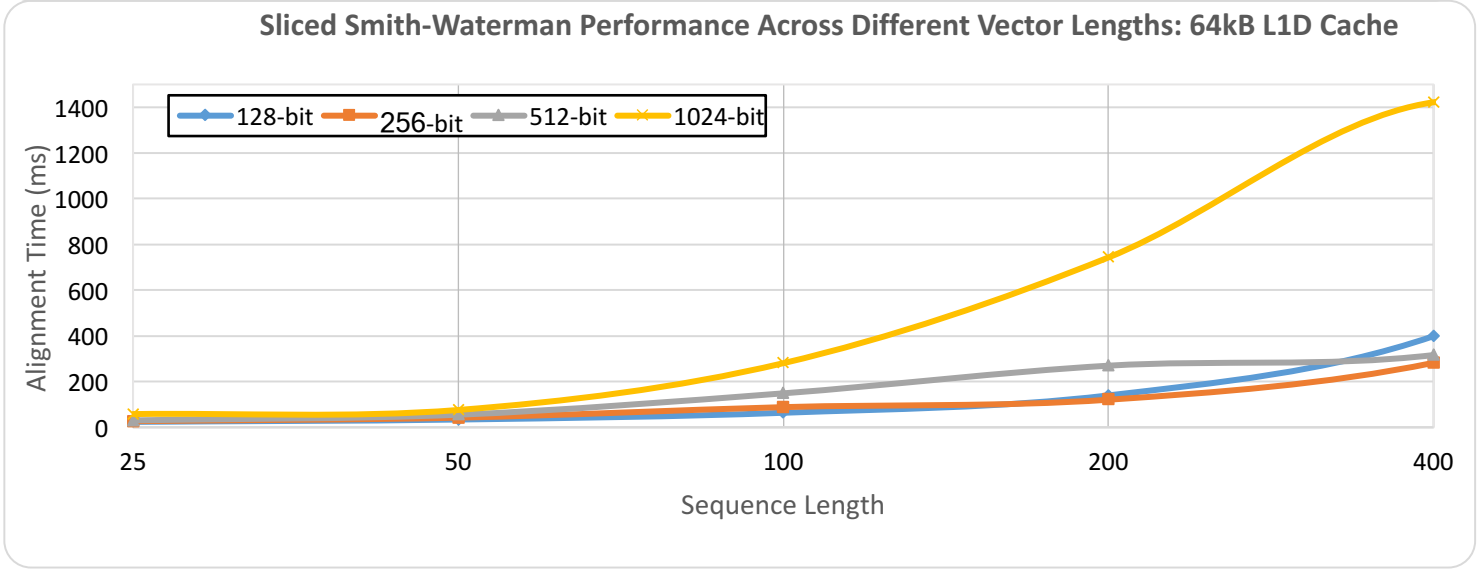


Eulerian path of de Bruijn graph gives us the original sequence

Hybrid Alignment Selection – BWA + SW

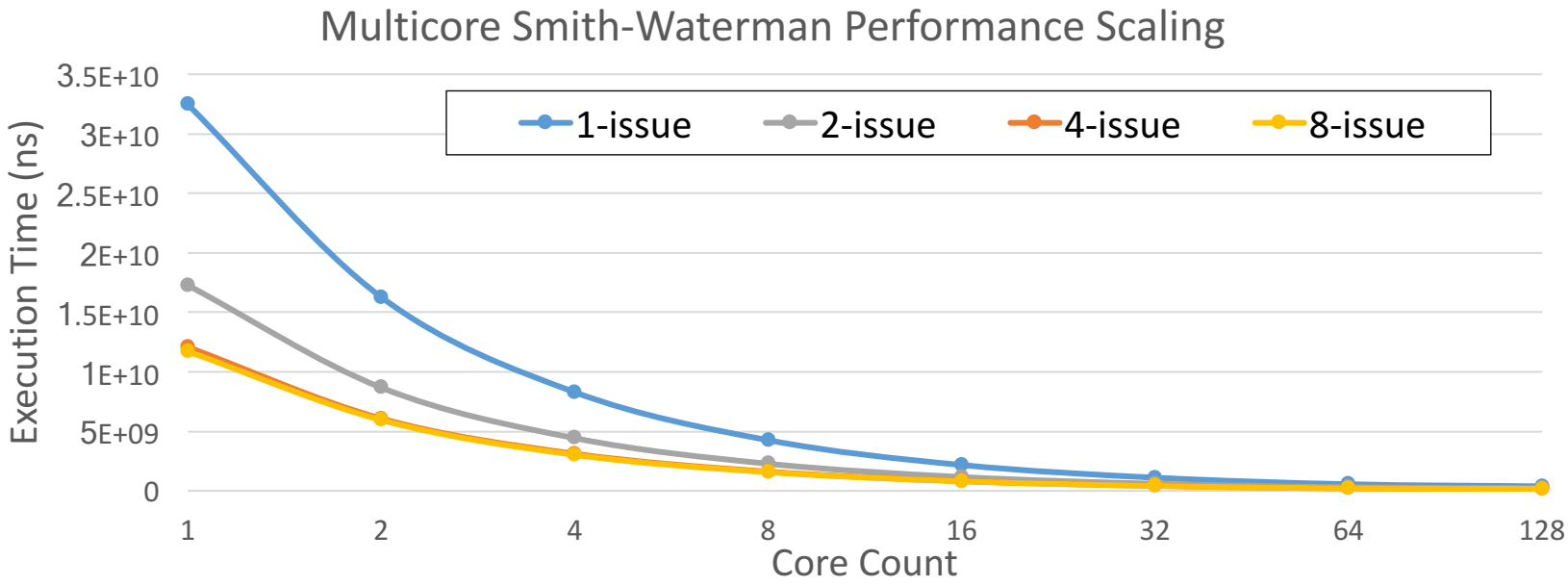



Batch Smith-Waterman Performance Across Different Vector Lengths: 64kB L1D Cache
Alignment Time (ms)
2500
2000
1500
1000
500
0
128 bit
256 bit
512 bit
1024 bit
25
50
100
200
400
Sequence Length

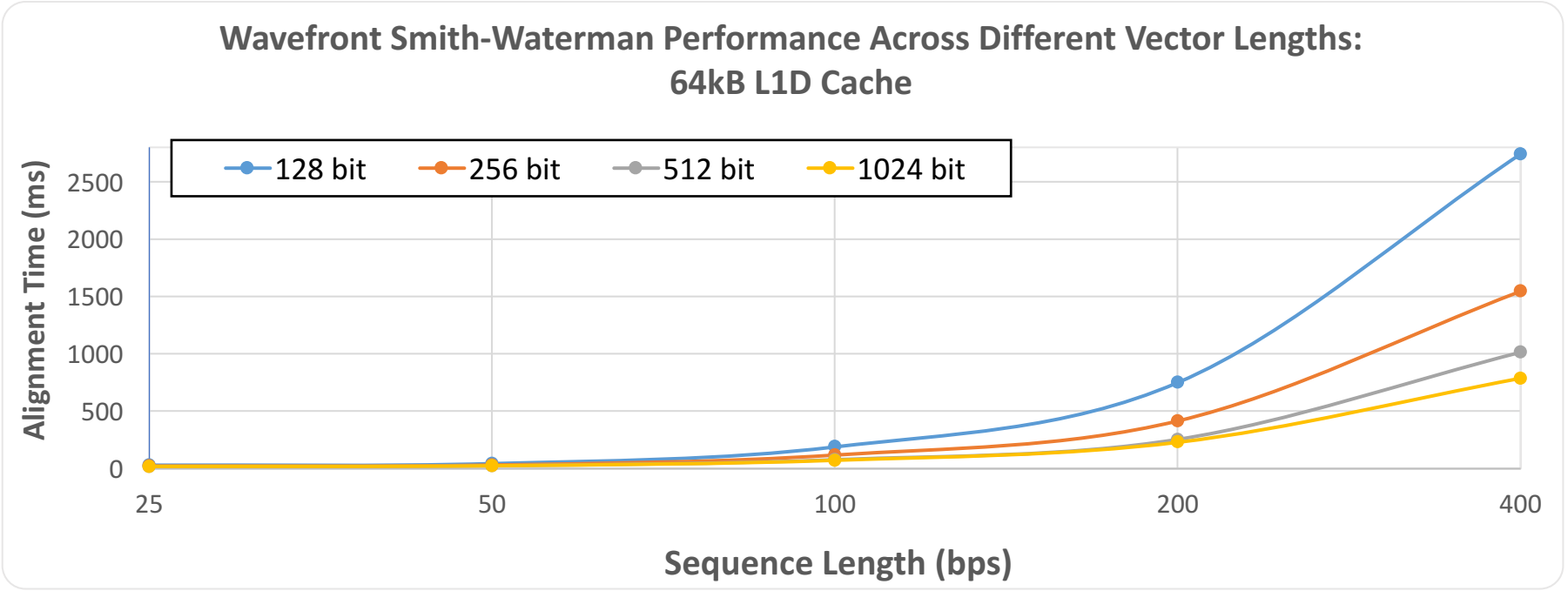

Sliced Smith-Waterman Performance Across Different Vector Lengths: 64kB L1D Cache
Alignment Time (ms)
1400
1200
1000
800
600
400
200
0
128-bit
256-bit
512-bit
1024-bit
25
50
100
200
400
Sequence Length

Sequence Alignment in Multicore System

- Smith-Waterman scales efficiently with core count
- Better single-thread performance can dramatically improve performance



Waveform Algorithm Performance Overview



Majority of Runtime

Alignment Algorithms

Smith-Waterman

- Dynamic algorithm
- Construct and trace alignment matrix

Accurate, but slow

Hash Table Based

- BLAST/BLAT

Prefix/Suffix Based

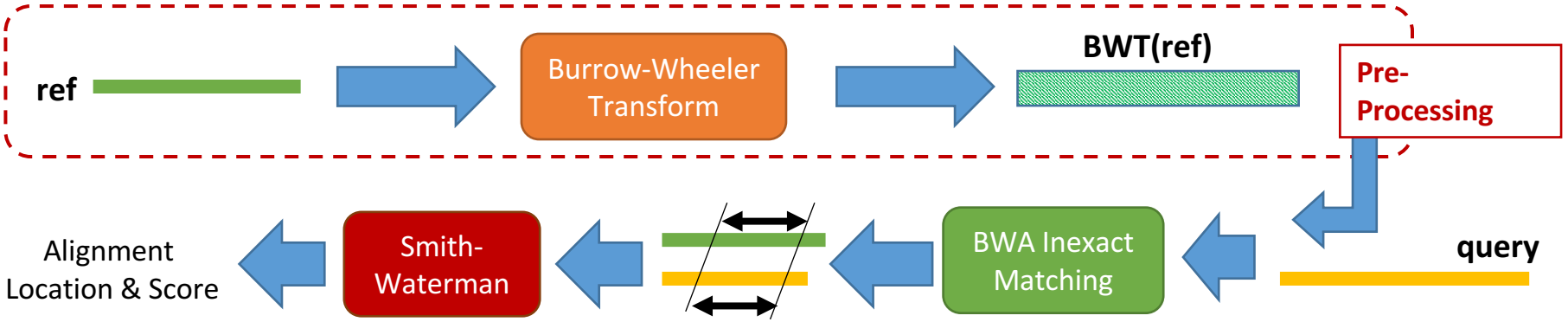
- **BWA,** STAR

Target Algorithm:

Burrow-Wheeler

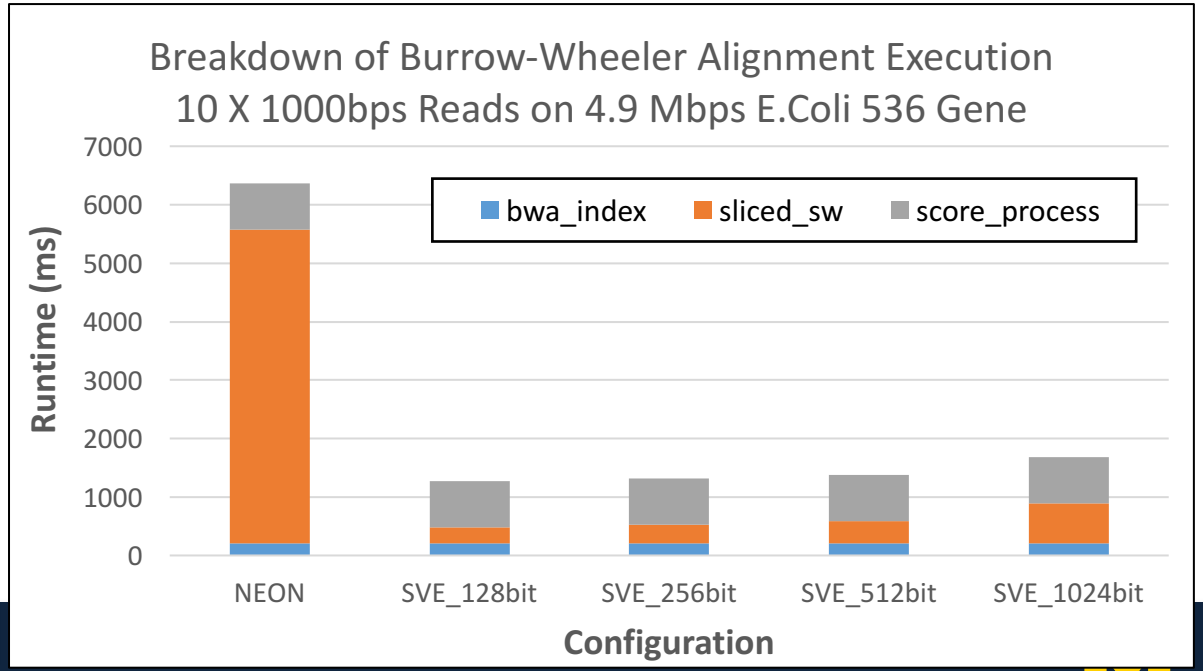
Why Burrow-Wheeler Alignment

- Performance of Smith-Waterman does not scale well with large sequence lengths.
- BWA help narrow down the search space (while losing some precision)
- Use Smith-Waterman to perform the precise alignment

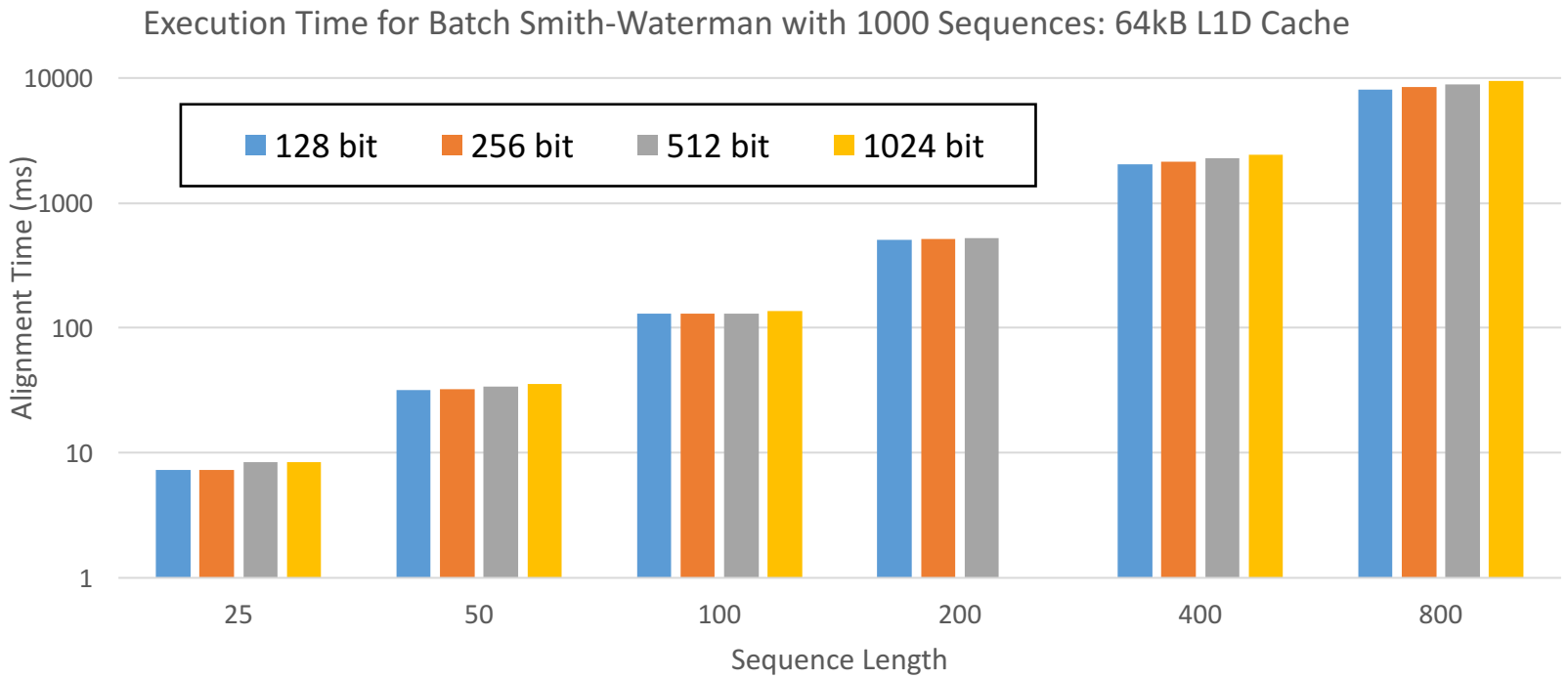


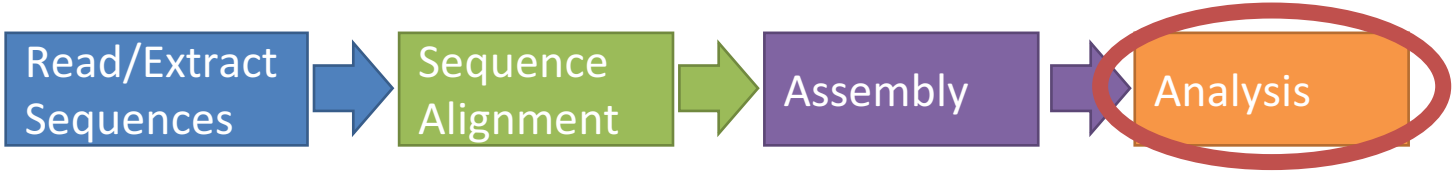
Burrow-Wheeler Alignment

- SVE significantly reduce the overhead of invoking Smith-Waterman alignment



Batch Smith-Waterman





Planted Motif Search: Identifying common patterns (motifs)

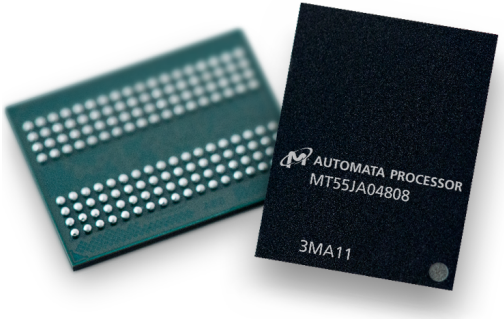
Micron's Automata Processor

- Reconfigurable automata processor built on DDR3 SDRAM memory arrays

(l,d) Pair	48 CPU	Micron Automata
(25,10)	21 min.	12 min.
(26,11)	2,814 min.	14 min.

1.8x

201x



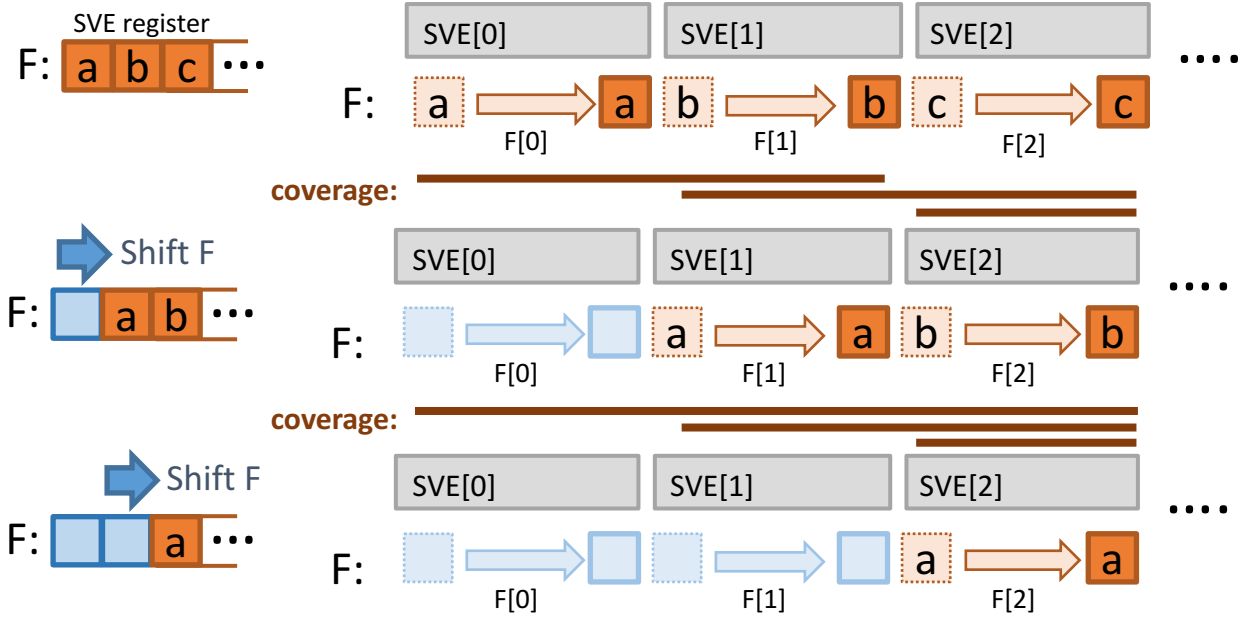
Sliced Smith-Waterman

$$F(m,n) = max \begin{cases} H(m-1,n) - g_o \\ F(m-1,n) - g_e \end{cases}$$

Dataflow of F(m,n)

1. Initial Calculation of $H(m,n)$

2. Resolving Dependencies

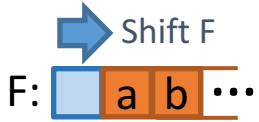


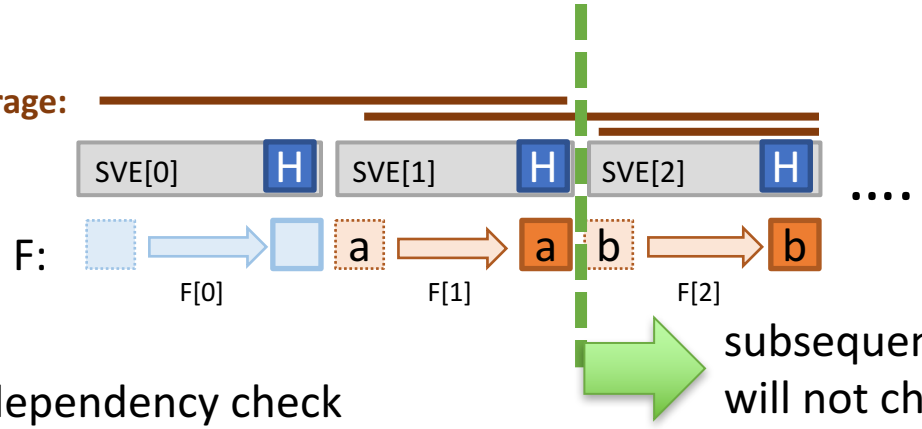
Sliced Smith-Waterman

$$F(m,n) = max\begin{cases} H(m-1,n) - g_o \\ F(m-1,n) - g_e \end{cases}$$

Dataflow of F(m,n)

2 Resolving Dependencies





subsequent columns will not change

For all vector elements:

If (H $-g_o$ **>** F $-g_e$ **) :** dependency check

Early Exit → All Dependency Resolved

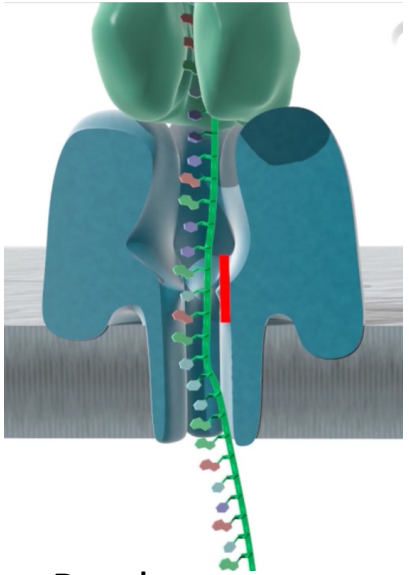
Worst Case: K Iterations $O\left(\frac{N^2}{K} + N^2\right)$

Best Case: Stop at first Iteration $O\left(\frac{N^2}{K}\right)$

Energy Efficient Sequencing

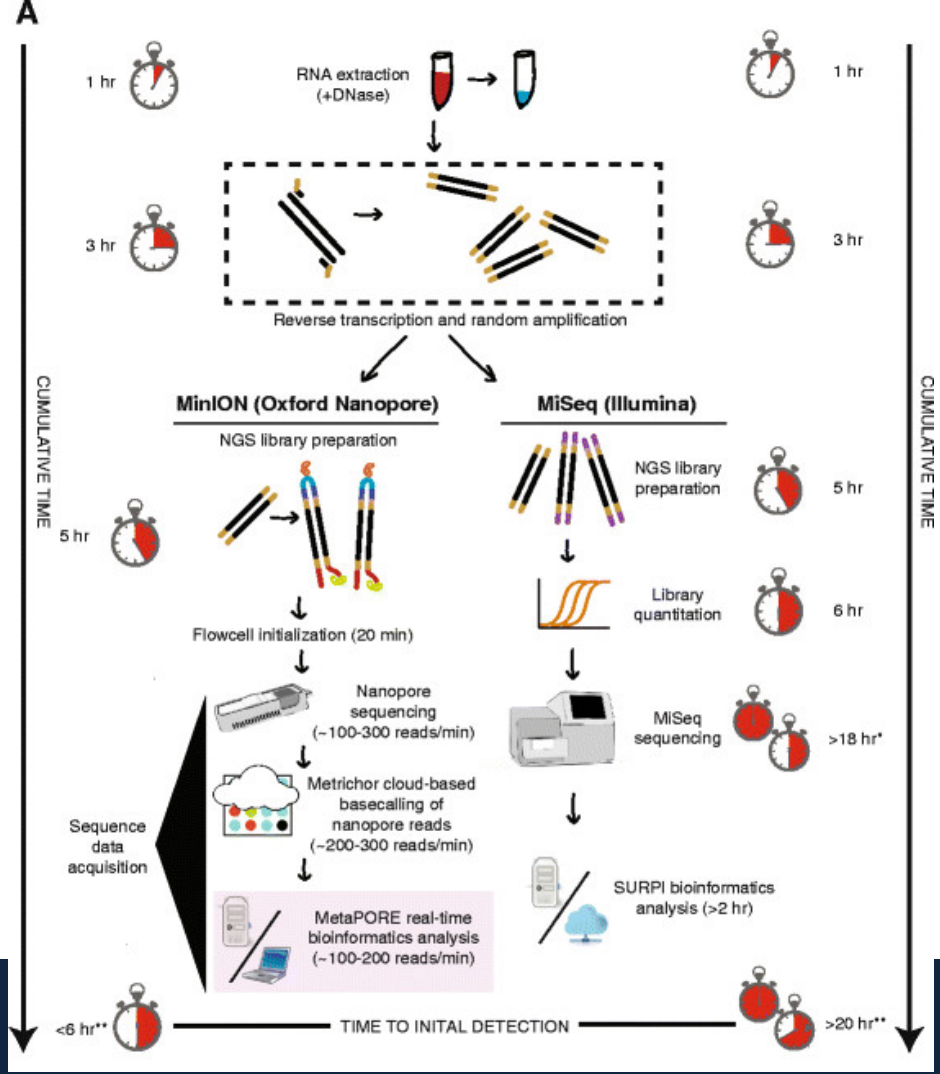
MinION: Portable, Real-time Sequencer

Up to 2.5M reads at standard speed (280bps)
$1,000 per Equipment



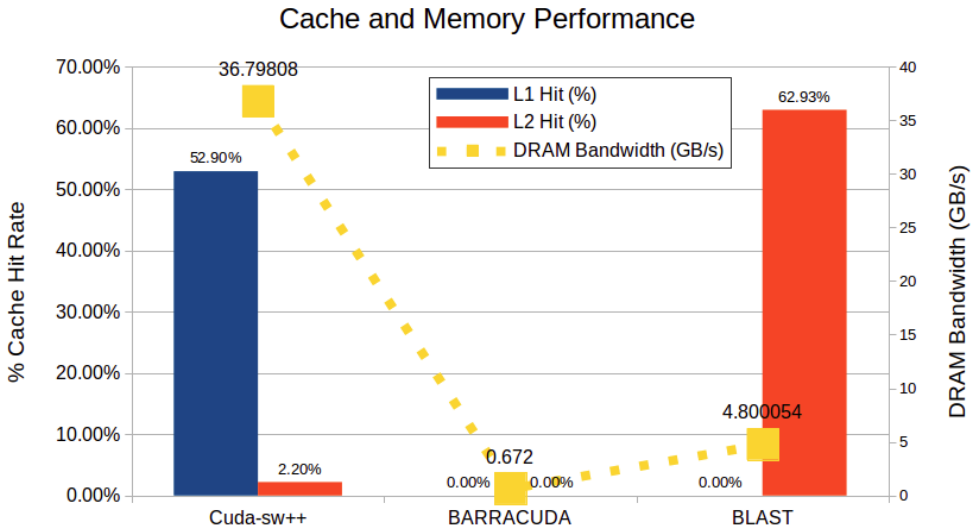
Sequence Read:
GGTTGTTCTGTTCTGCC

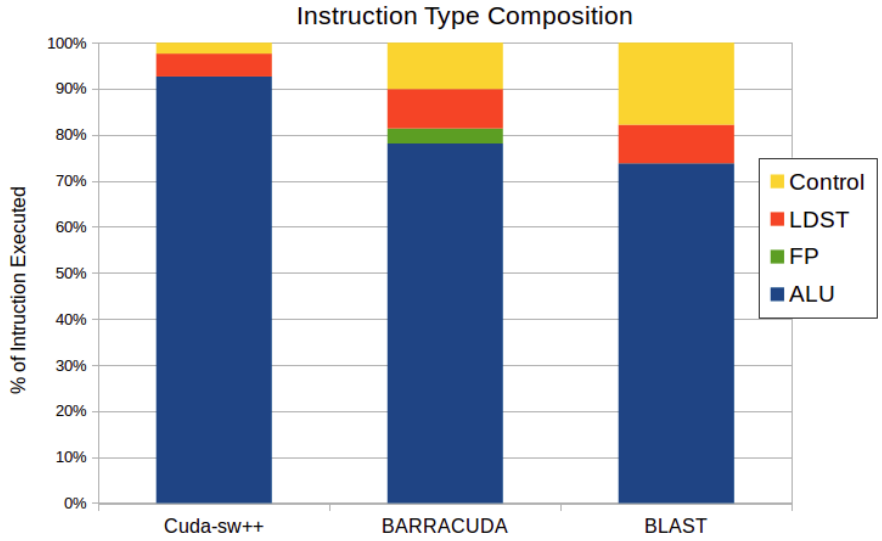
Oxford NANOPORE Technologies


A
1 hr
RNA extraction
(+DNase)
1 hr
3 hr
3 hr
Reverse transcription and random amplification
CUMULATIVE TIME
MinION (Oxford Nanopore)
MiSeq (Illumina)
NGS library preparation
5 hr
NGS library
preparation
5 hr
Library
quantitation
6 hr
Flowcell initialization (20 min)
Nanopore
sequencing
(~100-300 reads/min)
MiSeq
sequencing
>18 hr*
Metrichor cloud-based
basecalling of
nanopore reads
(~200-300 reads/min)
Sequence
data
acquisition
SURPI bioinformatics
analysis (>2 hr)
MetaPORE real-time
bioinformatics analysis
(~100-200 reads/min)
<6 hr**
TIME TO INITAL DETECTION
>20 hr**
CUMULATIVE TIME

GPU-based Alignment Workloads

- Cuda-SW++ - Smith-Waterman algorithm in GPU
- BARRACUDA – Burrow-Wheeler Alignment (prefix-based)
- BLAST – hash table based alignment



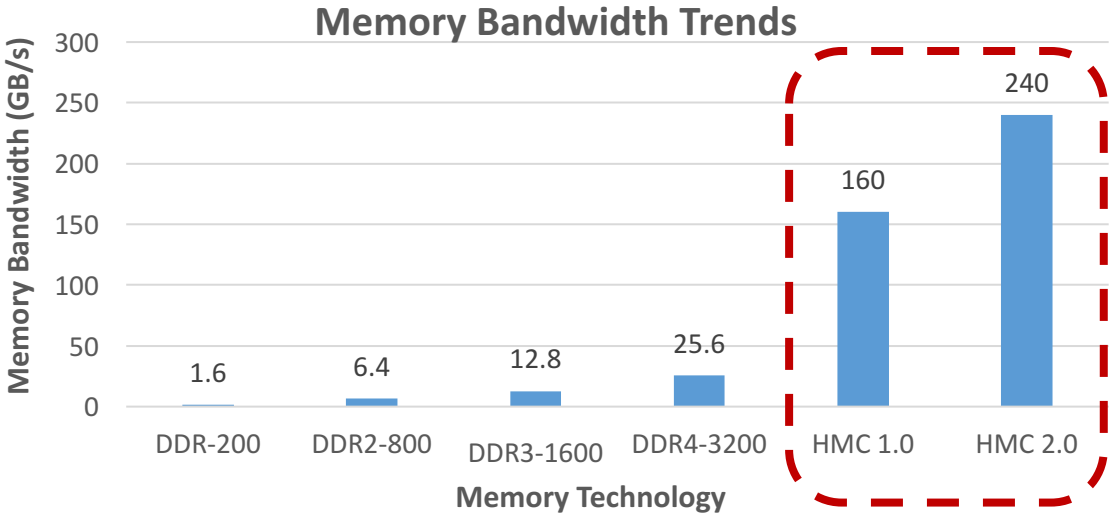


Why Compute Near Memory? - Bandwidth

I/O channels cannot keep up with the bandwidth of 3-D stacked memory

PCIe	Bandwidth
2.0	8GB/s
3.0	15.8 GB/s
4.0*	31.5 GB/s
5.0*	63.0 GB/s

*Latest version of PCIe is 4.0



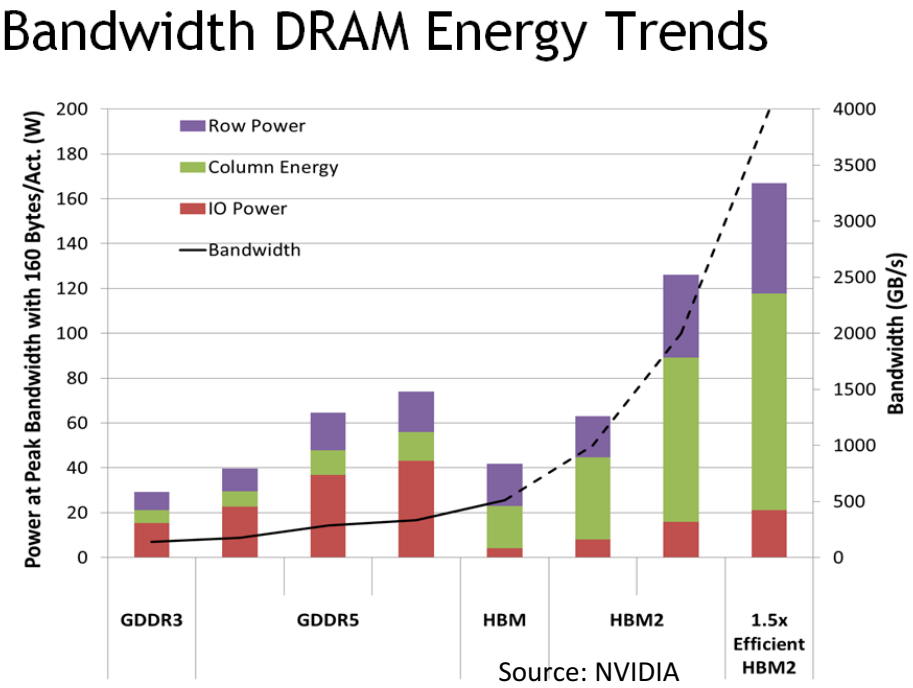
Source: Wikipedia, HMC

Why Compute Near Memory? - Power

High-bandwidth data transfer requires wide I/O channels

Cost of transferring large chunks of data incur significant I/O power

PIM helps reduce the I/O overhead



Scoring Matrix Construction

Scoring

Reference Sequence

Query Sequence

	--	A	C	A	C	A	A	…
--	0	0	0	0	0	0	0	…
A	0	2	1	2	1	2	2	…
G	0	1	1	1	1	1	1	…
C	0	0	3	2	3	2	1	…
A	0	2	2	**5**				…
⋮	⋮	⋮						

S(A,A) = +2

Given: $S(a,b) = +2$

Gap penalty = 1

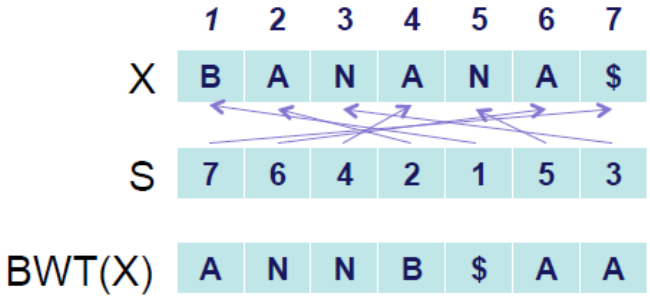
$H(3,4) =$

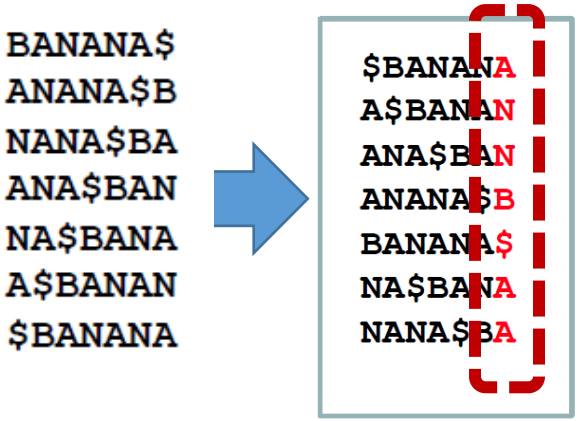
$$\max \begin{bmatrix} F(3,4) = 2-1 \\ E(3,4) = 2-1 \\ \mathbf{3+2=5} \end{bmatrix}$$

$$H(3,4) = 5$$

Burrow-Wheeler Transform

1. Get all possible rotations of sequence X
2. Sort the iterations by the suffix
3. Get the last character string, which is BWT(X)
4. The index of the original position is the suffix array S





BWT matrix of string ‘BANANA’

BWA Inexact Matching

- **Exact Matching:**

  Given a query W, get the lower bound L(W) and upper bound U(W) of index in BWT matrix where W is the prefix:

  $L(aW) = C(a) + i + 1$,
  where i = # ‘a’s up to L(W) – 1 in BWT(X)

  $U(aW) = C(a) + j$,
  where j = # ‘a’s up to U(W) in BWT(X)

  C(a): # of characters smaller than ‘a’

- **Inexact Matching:** allow ‘n’ number of mismatches or gaps during the matching process. Recursively iterate possible combinations.

Smith-Waterman Step

- Using the lower and upper bound indices from the BWA inexact matching, perform Smith-Waterman on the selected region to get a more accurate alignment.

- Accelerated through Sliced Smith-Waterman