Optimizing Traceback in the Smith-Waterman Algorithm for GPU architectures

LeAnn M. Lindsey[1], Hari Sundar[1](Advisor), Muaaz Awan[2] (Advisor)

[1]University of Utah, [2]Lawrence Berkeley National Laboratory

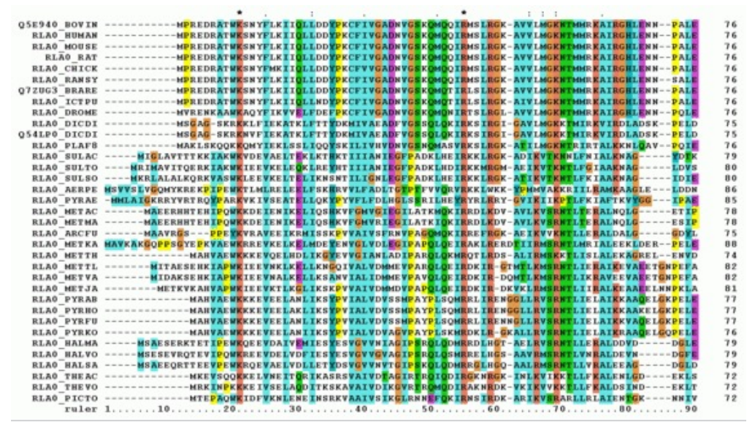
Abstract

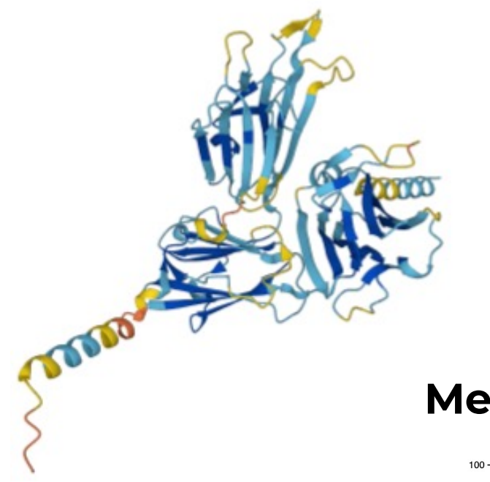
The traceback phase of the Smith-Waterman algorithm requires a significant amount of memory and introduces an irregular memory access pattern which makes it challenging to be performant on GPU architectures.   We developed an efficient traceback algorithm, using diagonal major indexing and binary representation of the traceback matrix. This implementation was integrated into the ADEPT[5] sequence alignment library. To demonstrate its efficacy in high performance software, we integrated our traceback implementation into Metahipmer, which is a widely used metagenome assembler. Our proposed implementation is 3.6x faster than traceback in GASAL2[6], and 51x faster than traceback in Striped Smith Waterman[10], currently the fastest GPU and CPU algorithms, respectively. It sped up the final alignment step in Metahipmer[8] by an average of 44% and improved the overall execution time of MHM2 by an average of 13%.

Background

Sequence Alignment is fundamental to many bioinformatics algorithms:

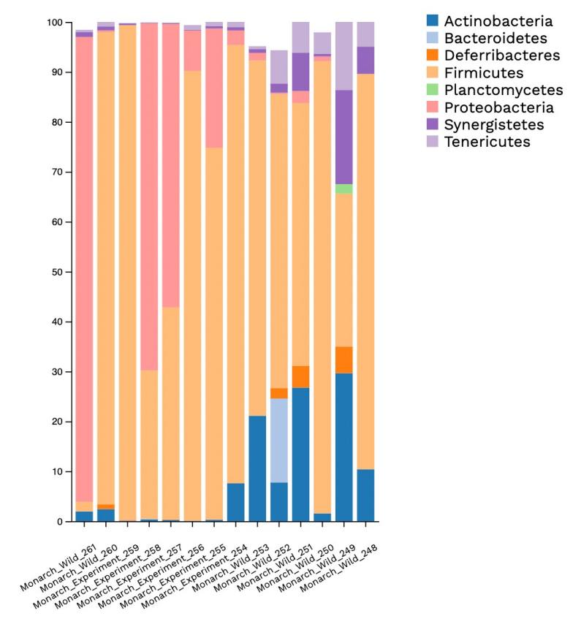
Multiple Sequence Alignment[1]



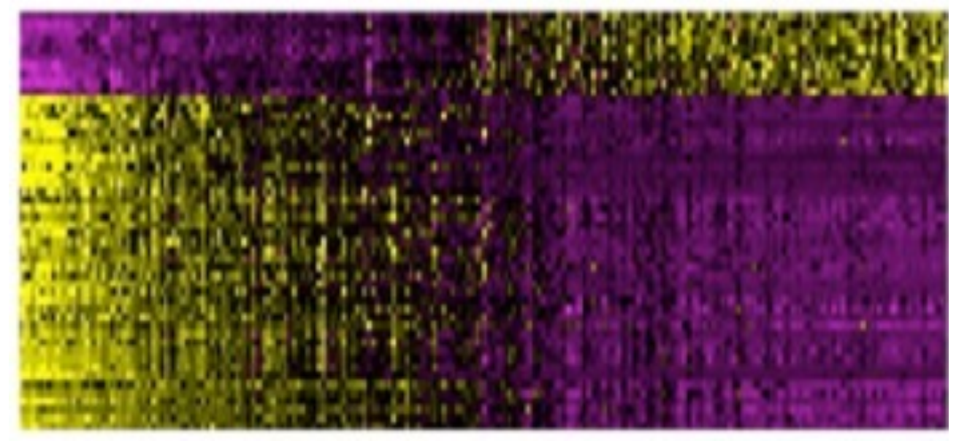


Protein Structure Prediction[2,3]

Metagenomic Analysis[10]



Genome Wide Association Studies[4]



Smith-Waterman Algorithm

Sequence Alignment Algorithms are required to *stitch together the short read sequences* produced by Next Generation Sequencers such as Illumina sequencing.

The **Smith-Waterman Algorithm** for local sequence alignment (Smith & Waterman, 1981) is the *base algorithm* used in many modern sequence alignment software packages.

The algorithm locates regions of similarity between two DNA or Amino Acid Sequences

- Guaranteed to find the optimal local alignment
- Time Complexity of *O(mn)*
- Space Complexity of *O(mn)*
- Computationally Intensive for large data sets

Affine Gap Scoring:

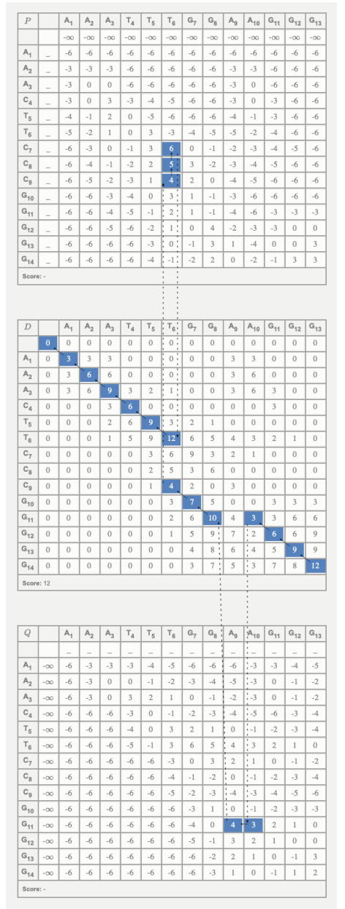
- Different penalties for a gap opening G_{init} and gap extension G_{ext}
- More biologically realistic
- Increased required memory to implement by adding E & F matrices.

Scoring Matrices:

- E matrix tracks insertions
- F matrix tracks deletions
- H matrix tracks aligned residues

To watch a Smith Waterman animation **SCAN** here

SCAN ME



$$F_{i,j} = max \begin{cases} F_{i-1,j} + G_{ext} \\ H_{i-1,j} + G_{init} \end{cases}$$

$$H_{i,j} = max \begin{cases} E_{i,j} \\ F_{i,j} \\ H_{i-1,j-1} + S(q_i, r_i) \\ 0 \end{cases}$$

$$E_{i,j} = max \begin{cases} E_{i,j-1} + G_{ext} \\ H_{i,j-1} + G_{init} \end{cases}$$

Challenges of Implementing on GPUs

CHALLENGE #1 - Minimize Memory Footprint on GPU

- Limited Shared Memory (~96 kb)
- Limited Global Memory (~32 GB)
- Global Memory access time ~100x slower than L1 or registers so reducing Global Memory footprint increases performance

The information that is needed for traceback is the *pointer* back to the cell that gave the maximum value in the scoring calculation.

$F_{i-1,j-1}$	$F_{i-1,j} + G_{ext}$
$F_{i,j-1}$	$F_{i,j}$

$H_{i-1,j-1} + S(q_j, r_i)$	$H_{i-1,j} + G_{init}$
$H_{i,j-1} + G_{init}$	$H_{i,j}$

$E_{i-1,j-1}$	$E_{i-1,j}$
$E_{i,j-1} + G_{ext}$	$E_{i,j}$

A top-of-the-line GPU has approximately *40 GB* of global memory, and the memory required to store the three matrices for 1 million alignments is *135-540 GB*

SOLUTION #1: Minimal Binary representation of all 3 matrices

- Reduces size of Global Memory Footprint to 1 byte per cell
- Total Global Memory Footprint is N*m*n (N=number of sequences)

where E = {0 - stay in H matrix,
1 - move to E matrix}

where F = {0 - stay in H matrix,
1 - move to F matrix}

where H = {11 - diagonal cell is max,
10 - top cell is max,
01 - left cell is max ,
00 - score is 0}

3x Global Memory Footprint Reduction

binary representation
0000HHEF
--------one byte--------

Reducing Global Memory Footprint minimizes the total number of Global Memory Accesses during the Scoring Phase of the SW algorithm

CHALLENGE #2 – Irregular Memory Access Pattern

- Dependencies in the recurrence result in parallelism along the diagonals
- Traditional Row Major or Column Major indexing of the Scoring Matrix in memory results in **uncoalesced memory accesses**

Traditional Matrix Indexing:

- In real data, m~1200 and n~150
- Diagonal values will span multiple cache lines leading to *uncoalesced memory accesses*

Uncoalesced Memory Accesses dramatically increase execution time as each 128 byte cache line must be loaded from Global Memory.

Row Major Indexing

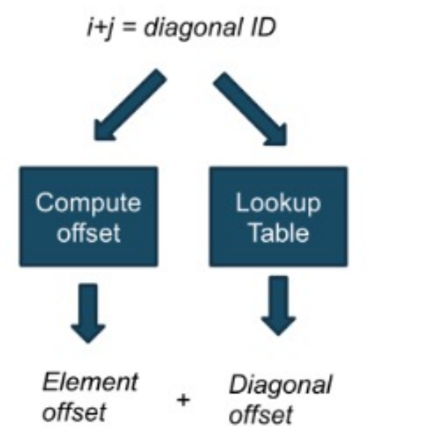
	0	1	2	3	4	5
0	0	1	2	3	4	5
1	6	7	8	9	10	11
2	12	13	14	15	16	17
3	18	19	20	21	22	23
4	24	25	26	27	28	29
5	30	31	32	33	34	35
6	36	37	38	39	40	41
7	42	43	44	45	46	47
8	48	49	50	51	52	53

SOLUTION #2 – Maximize Coalesced Memory accesses - Diagonal Major Indexing

- The SW algorithm is parallel on the diagonal, so we use diagonal major indexing to create coalesced memory accesses on write.

Lookup Table:

- diagonals are different sizes for each alignment
- Each CUDA block calculates a lookup table for the diagonal offset and stores it in shared memory.



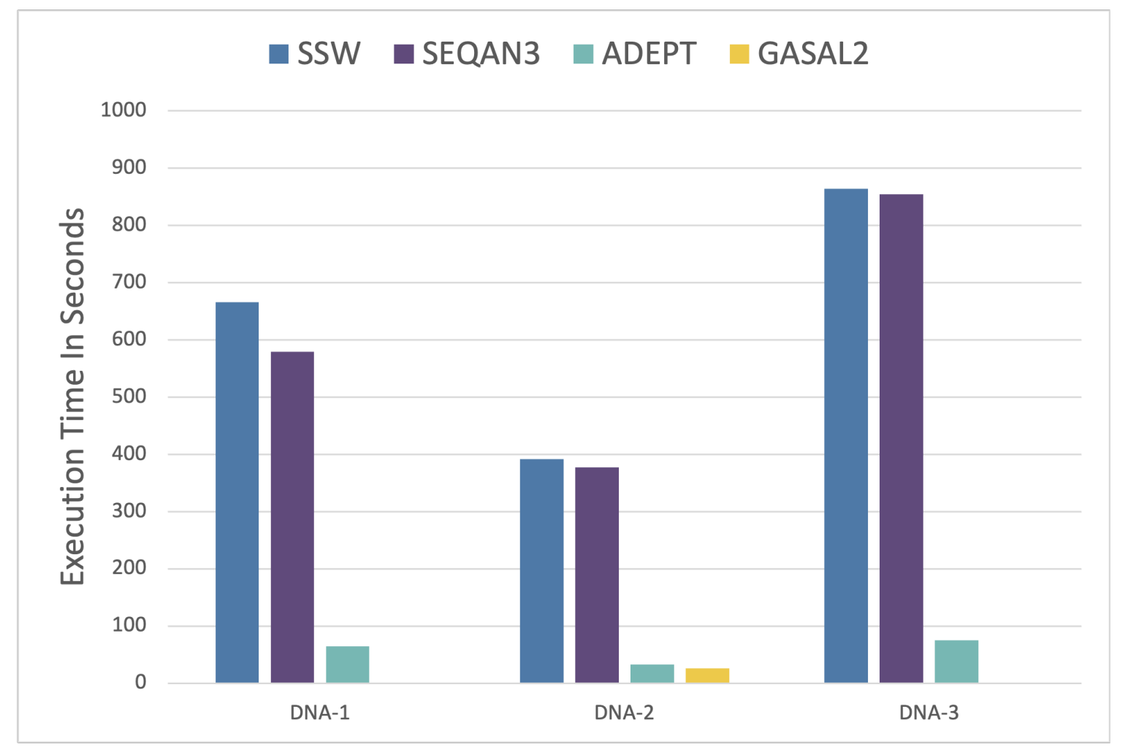
Diagonal Major Indexing

	0	1	2	3	4	5
0	0	2	5	9	14	20
1	1	4	8	13	19	26
2	3	7	12	18	25	32
3	6	11	17	24	31	38
4	10	16	23	30	37	43
5	15	22	29	36	42	47
6	21	28	35	41	46	50
7	27	34	40	45	49	52
8	33	39	44	48	51	53

Using Diagonal Major Indexing maximizes the number of coalesced global memory accesses during the Scoring Phase of the SW algorithm

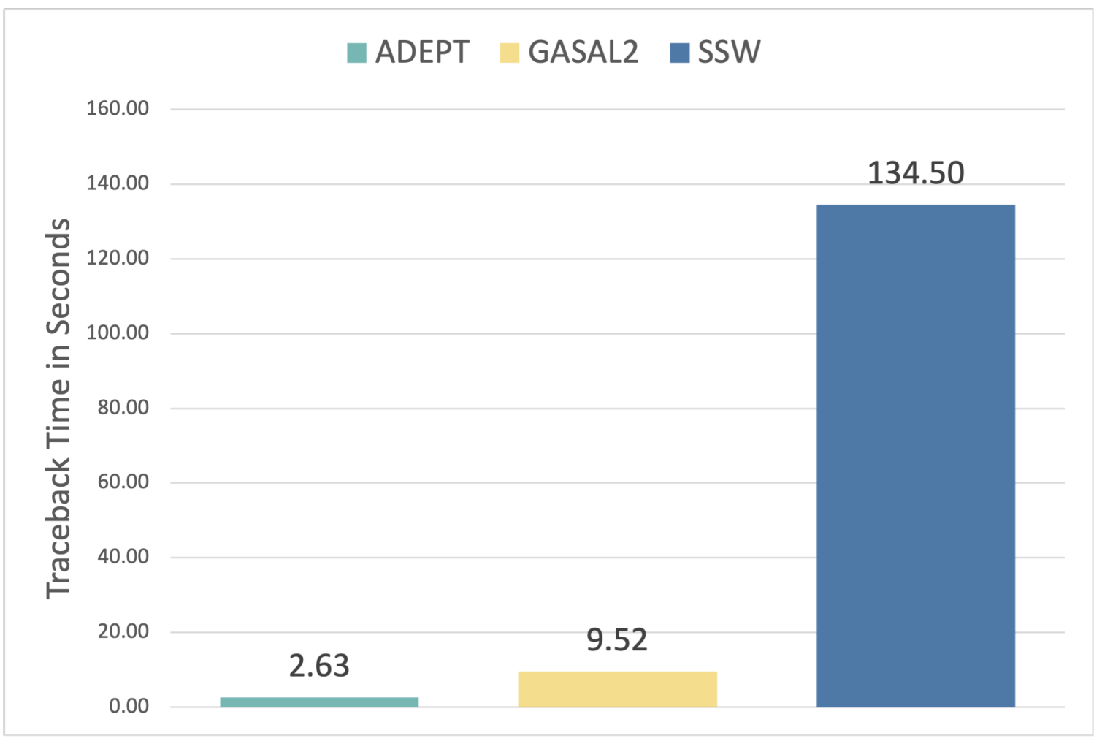
Performance Evaluation against Comparison Libraries

Overall Execution Time - DNA



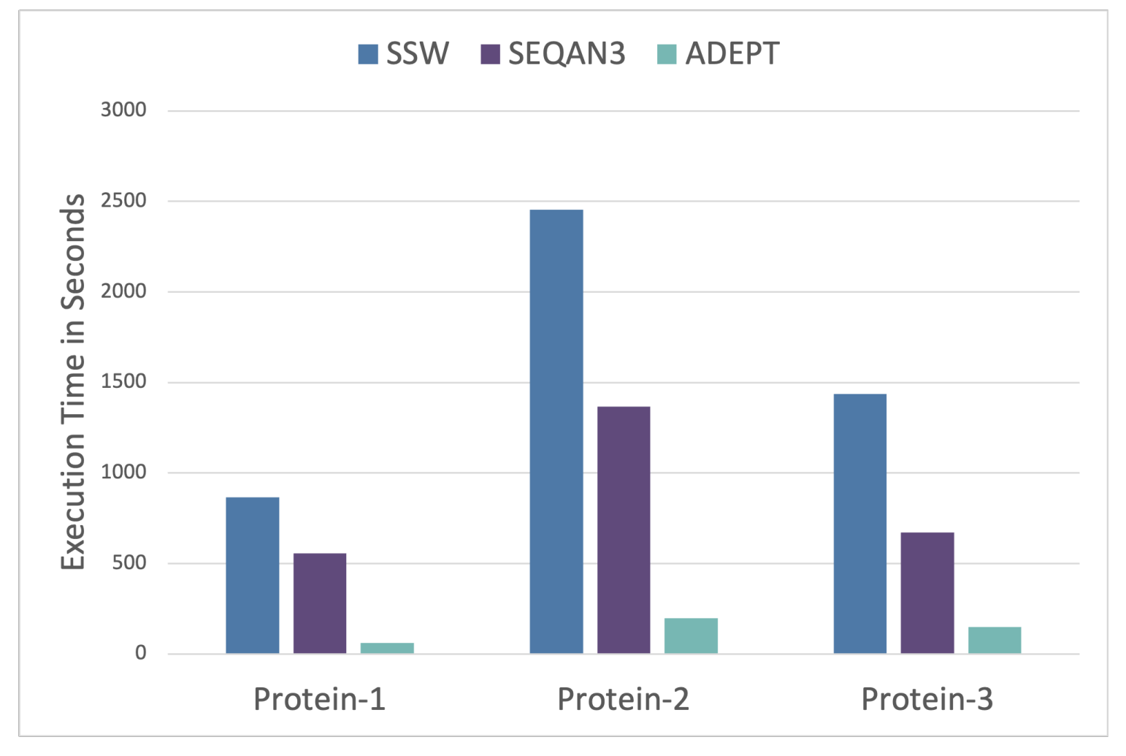
DNA - Overall Speed Ups:
10.5x speed up over SEQAN3
11.1x speed up over SSW

Traceback Time



Traceback Speed Ups:
3.6x speedup over GASAL2
51x speedup over SSW

Overall Execution Time – Proteins



Protein - Overall Speed Ups:
6.8x speedup over SEQAN3
11.8x speedup over SSW

Other Libraries:

- Striped-Smith-Waterman[9] (SSW)
- SeqAn3[7]
- Gasal2[8] (aligns only DNA sequences)

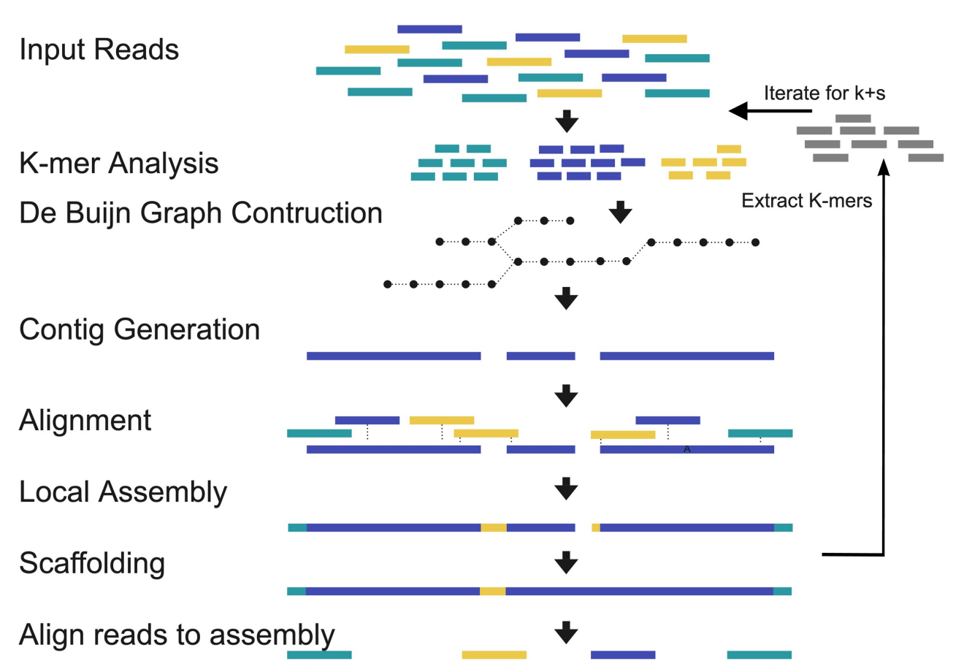
Notes:

1. Traceback Time is defined as additional execution time taken when traceback is turned on.
2. Overall Execution Time excludes I/O
3. Traceback Time was reported only on DNA Data Set 2 because GASAL2 had CUDA memory errors on all other data sets. This error was reported to the authors on June 2, 2022. (GASAL2 does not align Protein sequences)
4. SeqAn3 does not have an option to align without traceback, so it was not included in the Traceback Time results

Metahipmer2 Integration

Metahipmer2[8] (MHM2) is a high performance de novo metagenomic short read assembler. The iterative alignment phases do not require traceback and are performed on GPU using the ADEPT library. The user option **"--post-asm-align**" is used to align all reads to the generated contigs and produce a *CIGAR string* and a *SAM output file*

MHM2 Workflow



The "—post-asm-align" option requires traceback and previously MHM2 used the Striped Smith Waterman Library (SSW) on CPU to complete this last step.

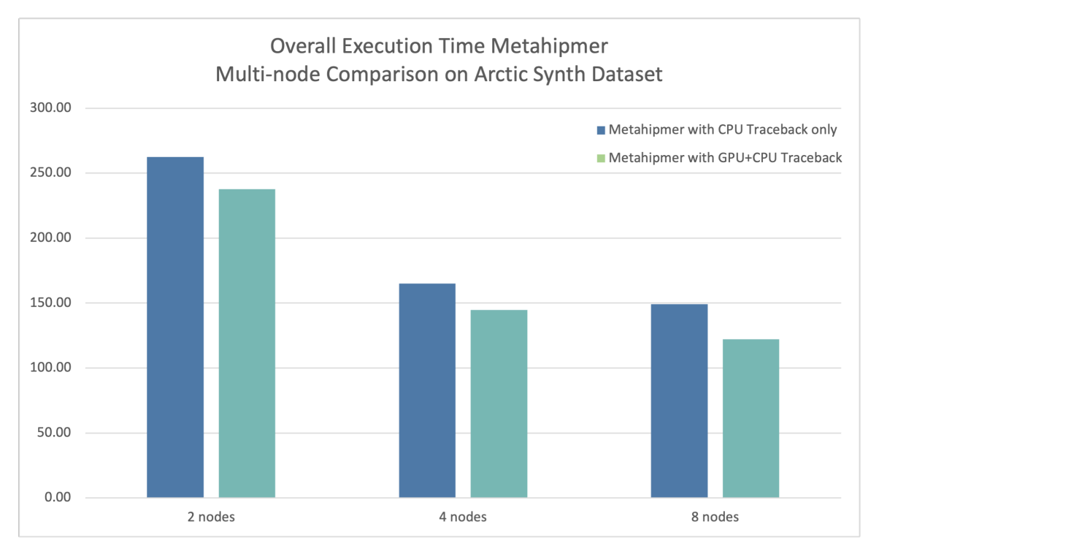
This final step produces a *Sequence Alignment Map* or SAM file that uses *CIGAR strings* to indicate the alignment of the reads to the assembly created by MHM2.

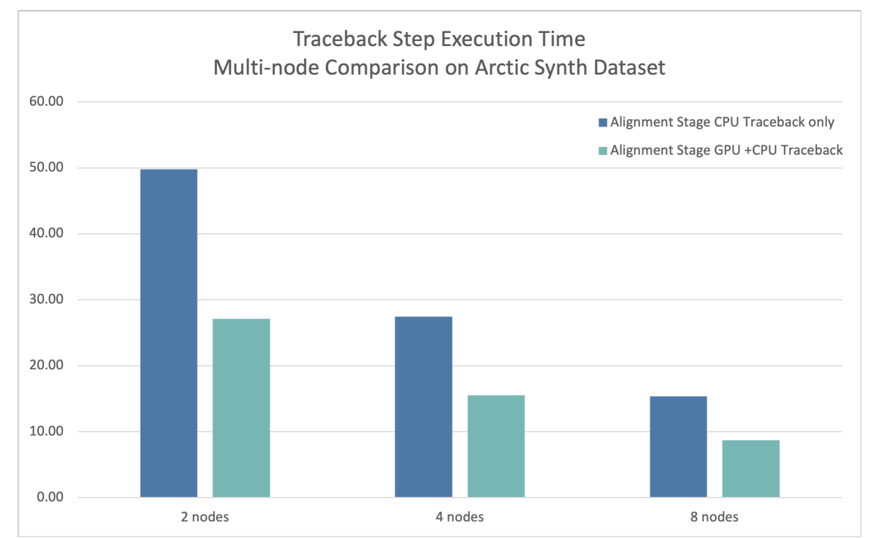
To learn more about SAM Files and CIGAR strings, **SCAN** here

SCAN ME

Performance Improvements:

- Traceback Step Execution Time was reduced by an average of **44%**
- Overall Execution Time was reduced by an average of **13%**





Acknowledgements

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of two U.S. Department of Energy organizations (Office of Science and the National Nuclear Security Administration) responsible for the planning and preparation of a capable exascale ecosystem, including software, applications, hardware, advanced system engineering, and early testbed platforms, in support of the nation's exascale computing imperative. This research was also funded by National Science Foundation grants 1808652 and 2008772.

References

[1] M. A. Larkin *et al.*, "Clustal W and Clustal X version 2.0," *Bioinformatics*, vol. 23, no. 21, pp. 2947–2948, Nov. 2007, doi: 10.1093/bioinformatics/btm404.
[2] J. Jumper *et al.*, "Highly accurate protein structure prediction with AlphaFold," *Nature*, vol. 596, no. 7873, Art. no. 7873, Aug. 2021, doi: 10.1038/s41586-021-03819-2.
[3] M. Varadi *et al.*, "AlphaFold Protein Structure Database: massively expanding the structural coverage of protein-sequence space with high-accuracy models," *Nucleic Acids Res.*, vol. 50, no. D1, pp. D439–D444, Jan. 2022, doi: 10.1093/nar/gkab1061.
[4] R. Satija, J. A. Farrell, D. Gennert, A. F. Schier, and A. Regev, "Spatial reconstruction of single-cell gene expression data," *Nat. Biotechnol.*, vol. 33, no. 5, Art. no. 5, May 2015, doi: 10.1038/nbt.3192.
[5] M. G. Awan *et al.*, "ADEPT: a domain independent sequence alignment strategy for gpu architectures," *BMC Bioinformatics*, vol. 21, no. 1, p. 406, Sep. 2020, doi: 10.1186/s12859-020-03720-1.
[6] M. G. Awan *et al.*, "Accelerating large scale de novo metagenome assembly using GPUs," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, Nov. 2021, pp. 1–11. doi: 10.1145/3458817.3476212.
[7] A. Döring, D. Weese, T. Rausch, and K. Reinert, "SeqAn an efficient, generic C++ library for sequence analysis," *BMC Bioinformatics*, vol. 9, p. 11, Jan. 2008, doi: 10.1186/1471-2105-9-11.
[8] N. Ahmed, J. Lévy, S. Ren, H. Mushtaq, K. Bertels, and Z. Al-Ars, "GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data," *BMC Bioinformatics*, vol. 20, no. 1, p. 520, Oct. 2019, doi: 10.1186/s12859-019-3086-9.
[9] M. Farrar, "Striped Smith–Waterman speeds database searches six times over other SIMD implementations," *Bioinformatics*, vol. 23, no. 2, pp. 156–161, Jan. 2007, doi: 10.1093/bioinformatics/btl582.
[10] https://leannmlindsey.github.io/dataviscourse-pr-Visualization-of-Metagenomic-Data/FinalProject.html