SMITH-WATERMAN ALGORITHM IN PARALLEL

CSE633 MPI Project

Jian Chen

University at Buffalo The State University of New York

Outline

Sequential Sequence alignment algorithm

- Smith-Waterman & Needleman–Wunsch

Parallele Smith-Waterman

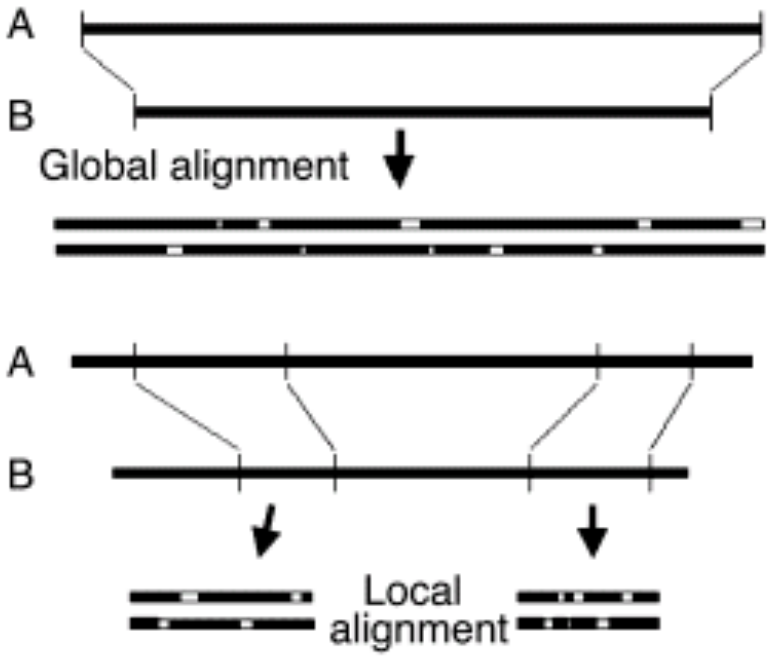
- Time Complexity
- Memory cost

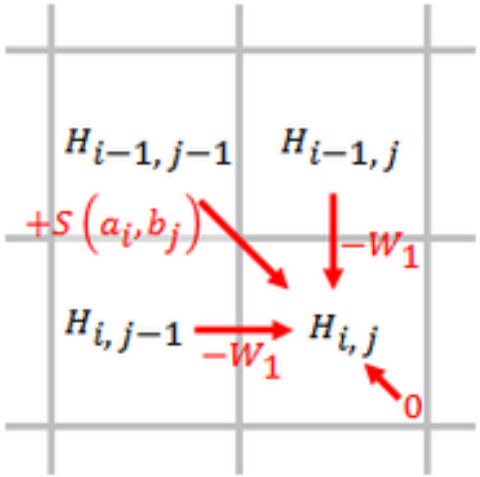
Experiment Result

- Fix data, increase number of processor
  - On single node
  - On multiple nodes
- Fix data per processor, increase data and processor number

Reference

Needleman–Wunsch





Smith-Waterman

Smith-Waterman

Traceback

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

3

```
G T T - A C
| | |   | |
G T T G A C
```

Needleman–Wunsch

Needleman-Wunsch

match = 1 mismatch = -1 gap = -1

		G	C	A	T	G	C	U
	0	-1	-2	-3	-4	-5	-6	-7
G	-1	1	0	-1	-2	-3	-4	-5
A	-2	0	0	1	0	-1	-2	-3
T	-3	-1	-1	0	2	1	0	-1
T	-4	-2	-2	-1	1	1	0	-1
A	-5	-3	-3	-1	0	0	0	-1
C	-6	-4	-2	-2	-1	-1	1	0
A	-7	-5	-3	-1	-2	-2	0	0

```
GCA-TGCU
|  | | . | .
G -ATTACA
```

Smith-Waterman Algorithm

Fill the scoring matrix

		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

Substitution matrix: $S(a_i, b_j) = \begin{cases} +3, & a_i = b_j \\ -3, & a_i \neq b_j \end{cases}$

Gap penalty: $W_k = kW_1$, $W_1 = 2$

Traceback

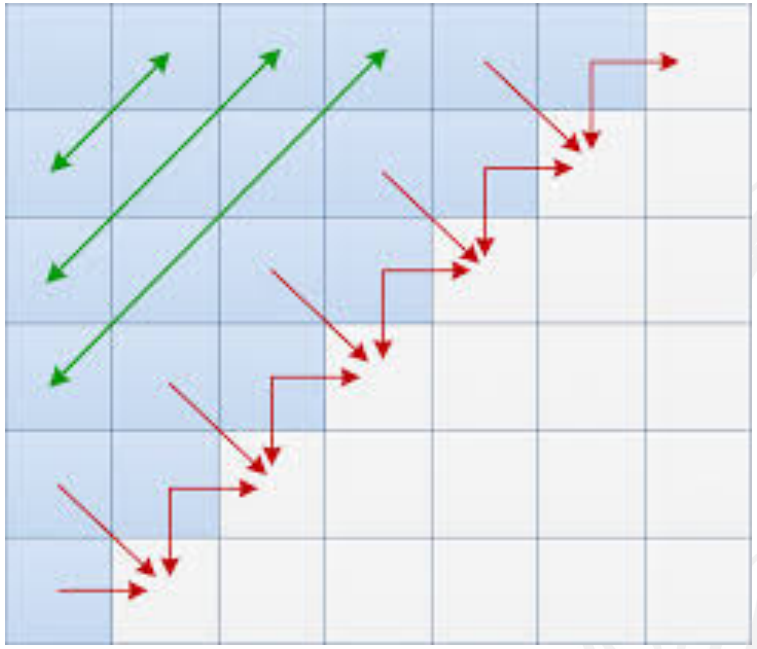
		T	G	T	T	A	C	G	G
	0	0	0	0	0	0	0	0	0
G	0	0	3	1	0	0	0	3	3
G	0	0	3	1	0	0	0	3	6
T	0	3	1	6	4	2	0	1	4
T	0	3	1	4	9	7	5	3	2
G	0	1	6	4	7	6	4	8	6
A	0	0	4	3	5	10	8	6	5
C	0	0	2	1	3	8	13	11	9
T	0	3	1	5	4	6	11	10	8
A	0	1	0	3	2	7	9	8	7

3

G T T - A C

G	T	T	G	A	C

How to parallel



	-	C	G	G	G	T	A	T	C
-	0	0	0	0	0	0	0	0	0
C	0	T1	T2	T3	T4	T5	T6	T7	T8
C	0	T2	T3	T4	T5	T6	T7	T8	T9
C	0	T3	T4	T5	T6	T7	T8	T9	T10
T	0	T4	T5	T6	T7	T8	T9	T10	T11
A	0	T5	T6	T7	T8	T9	T10	T11	T12
G	0	T6	T7	T8	T9	T10	T11	T12	T13
G	0	T7	T8	T9	T10	T11	T12	T13	T14
T	0	T8	T9	T10	T11	T12	T13	T14	T15

Figure 2. Cases calculable at the same time T_i.

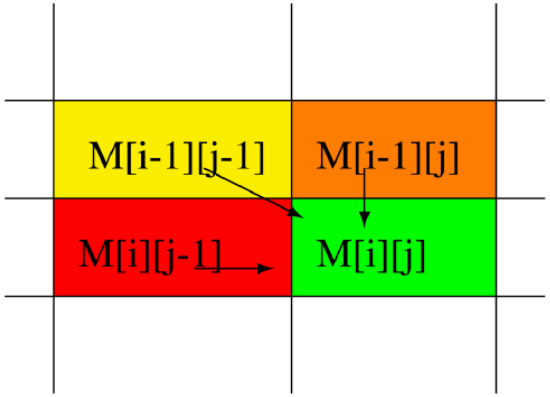


Figure 4. Calculating $M[i][j]$ dependency.

follow the update rule, each diagonal can be computed at the same time

-	C	G	G	G	T	A	T	C								
0	0	0	0	0	0	0	0	0								
-	0	T1	T2	T3	T4	T5	T6	T7	T8							
	C	0	T2	T3	T4	T5	T6	T7	T8	T9						
		C	0	T3	T4	T5	T6	T7	T8	T9	T10					
			C	0	T4	T5	T6	T7	T8	T9	T10	T11				
				T	0	T5	T6	T7	T8	T9	T10	T11	T12			
					A	0	T6	T7	T8	T9	T10	T11	T12	T13		
						G	0	T7	T8	T9	T10	T11	T12	T13	T14	
							G	0	T8	T9	T10	T11	T12	T13	T14	T15
								T	0							

Figure 3. Linear representation of the parallelizable boxes.

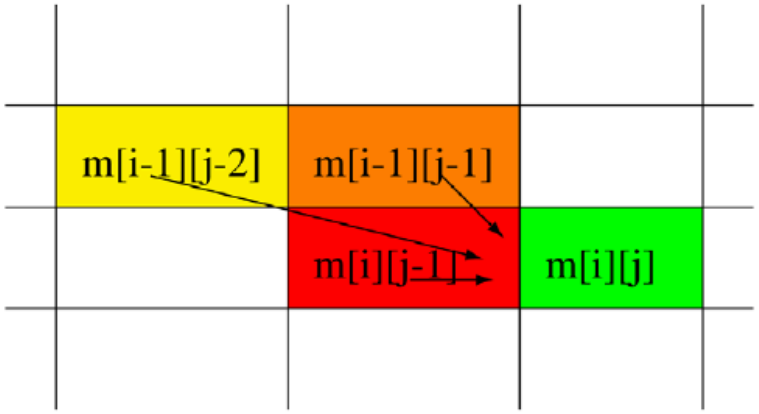
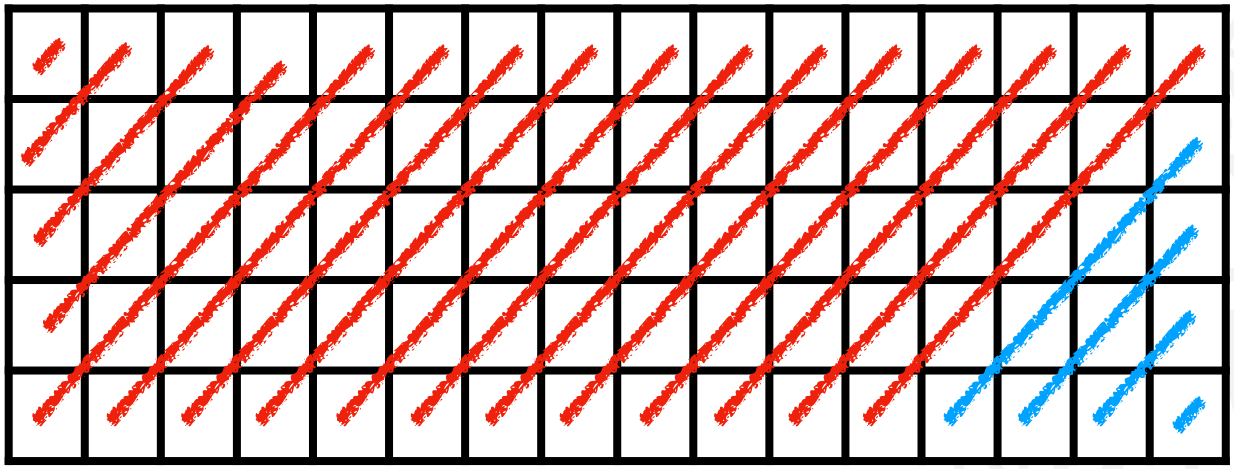


Figure 5. New dependencies computing $m[i][j]$.

follow the update rule, each column can be computed at the same time

How Many steps?

N Column

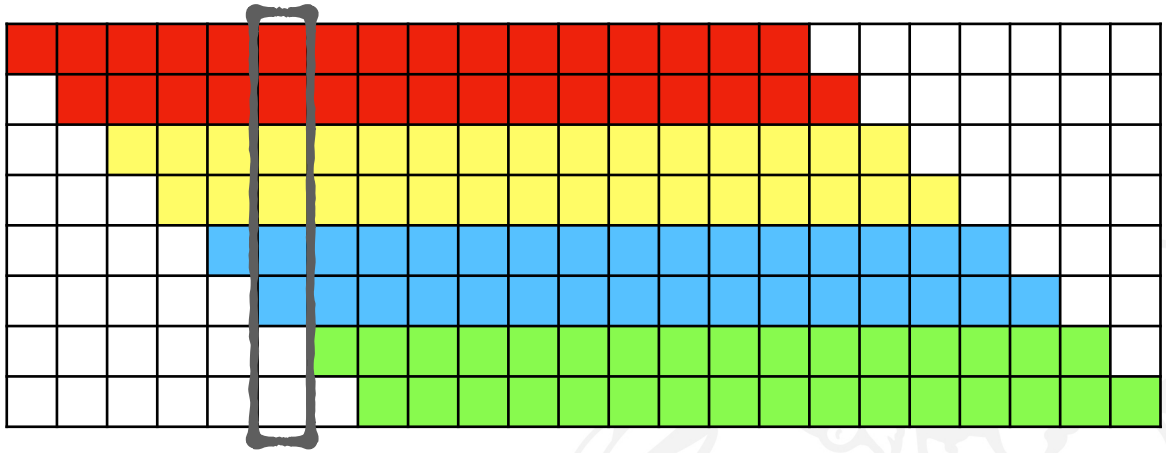


M Row

M + N -1 steps

Speed-up factor

8 row, 16 column, 4 Processor



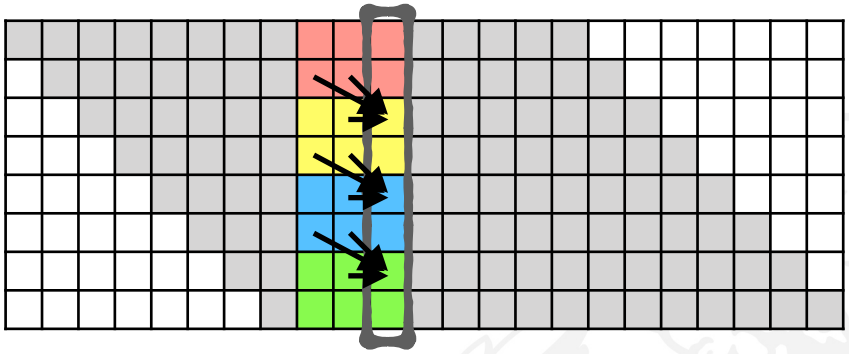
For M row, N column, P processors

Suppose 1 process fill 1 entry cost t seconds. each processor got N/P rows
In each step each process need to fill N/P entry cost (N/P) t.
All M + N -1 step. Cost [(M + N -1) x N / P] t

While sequentially cost M*N, The ratio is (M+N-1) / MP
(when M = N, the factor is about 2/P)

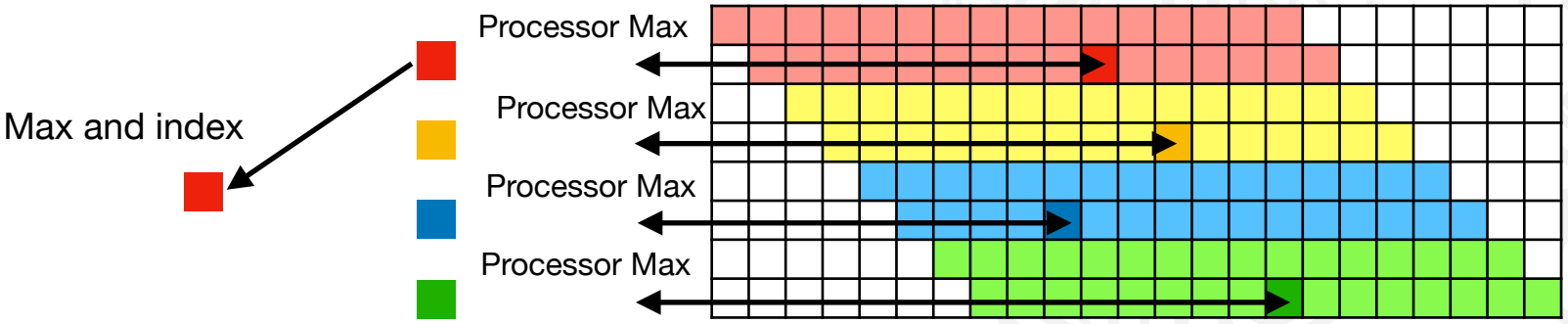
Save memory to scale the data

8 row, 16 column, 4 Processor

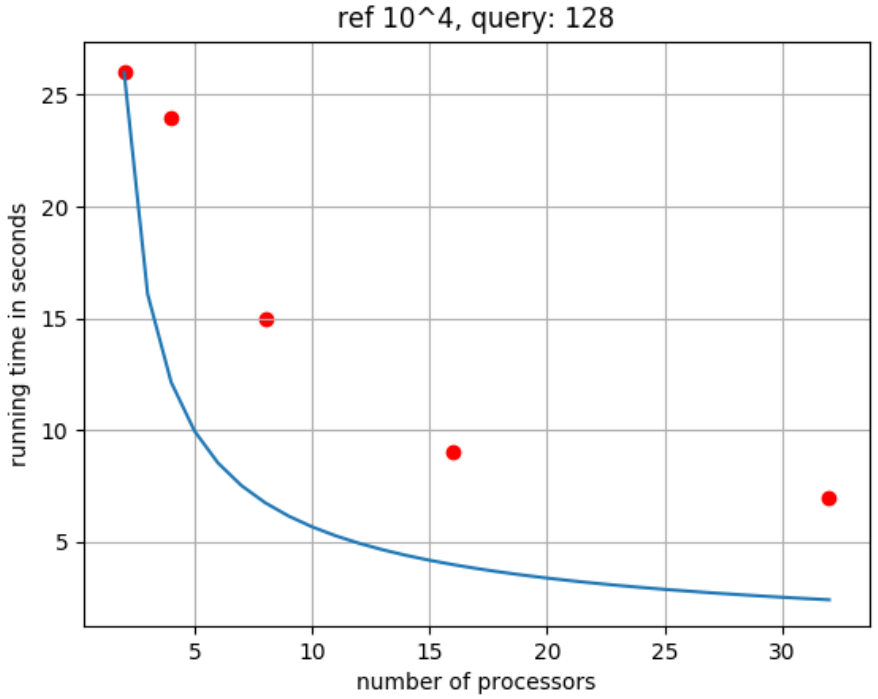


Only save 3 column max and index

Only save the maximum and index



Fix data, increase number of processor

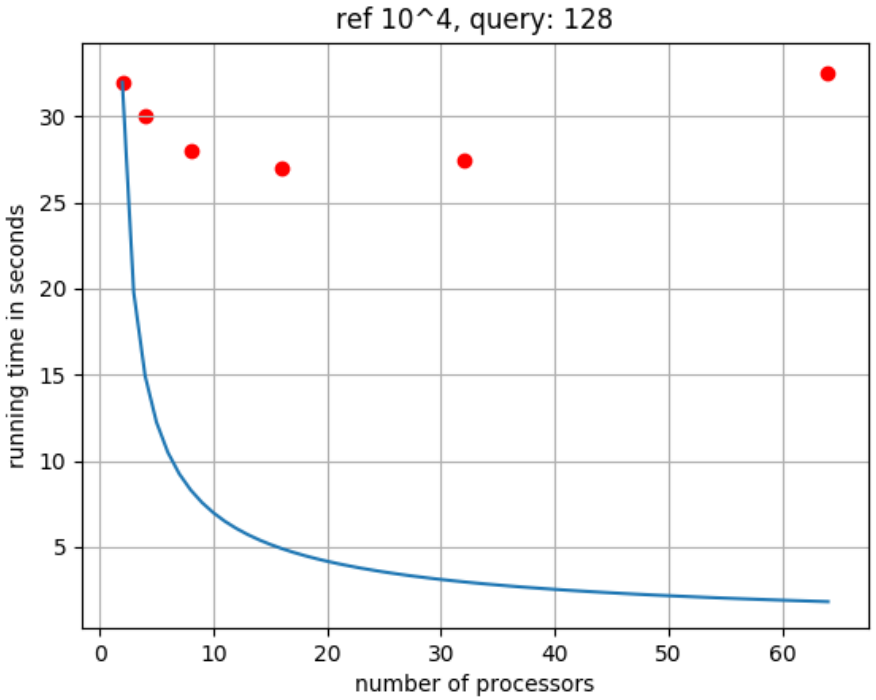


P	2	4	8	16	32
T	26s	24s	15s	9s	7s

Running time on 1 node with 32 cores

Because communication cost much more than computing in each time circle

Fix data, increase number of processor

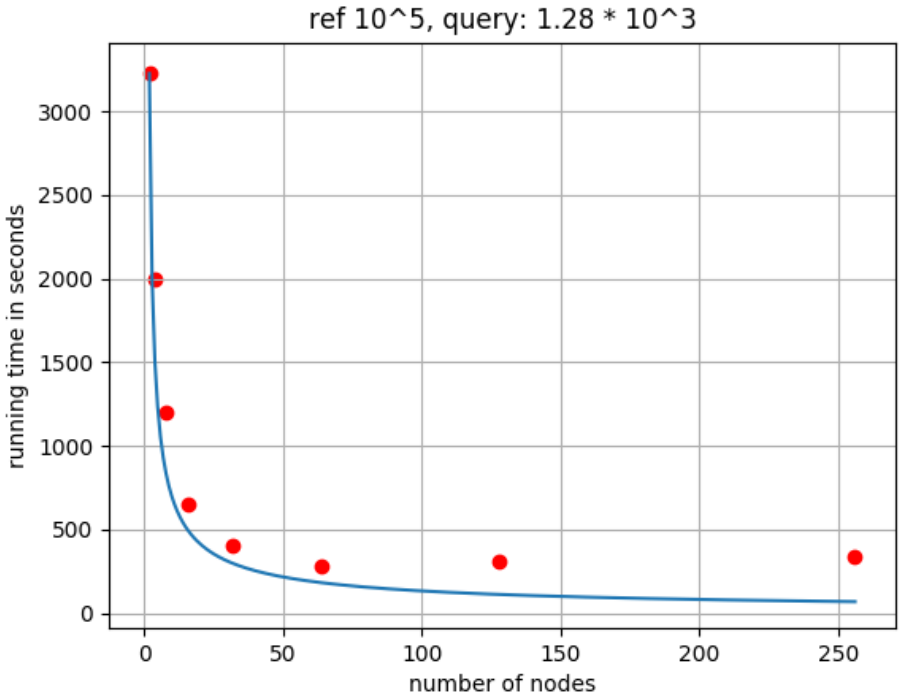


P	2	4	8	16	32	64
T	32s	30s	28s	27s	27.5s	32.5s

Running time on multiple nodes with 1 process per node

Running time increase for large number of processors. Hard to observe speed up on multi-node, Because communication between nodes are even more costly

Fix data, increase number of processor

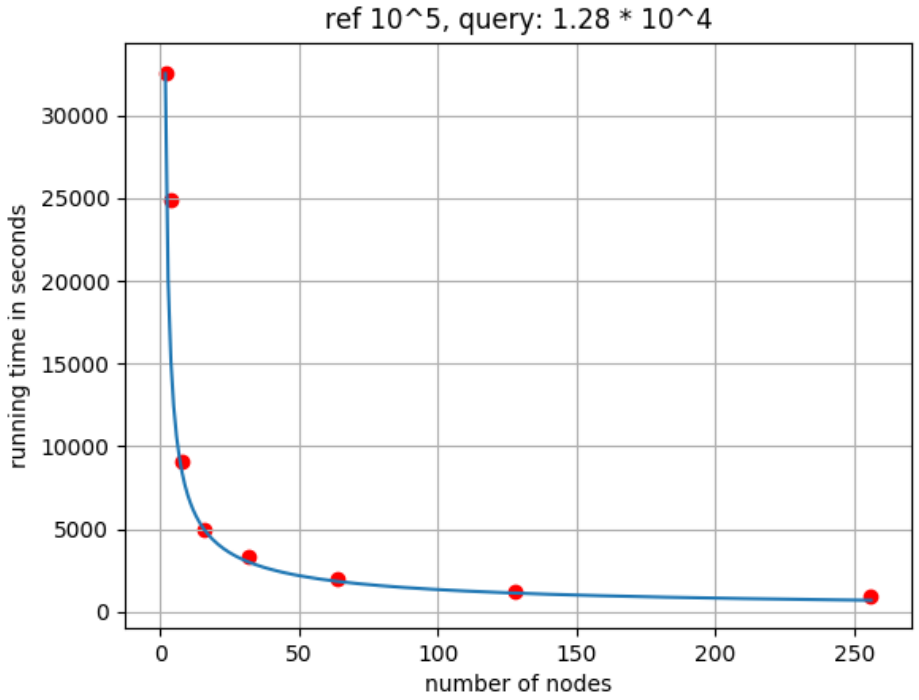


Running time on multiple nodes with 1 process per node, Increase query size 1.28*10^3 and reference size 10^5

Still observing running time increase after 64 processors

P	2	4	8	16	32	64	128	256
T	3225	2016	1257	651	403	281	313	343

Fix data, increase number of processor

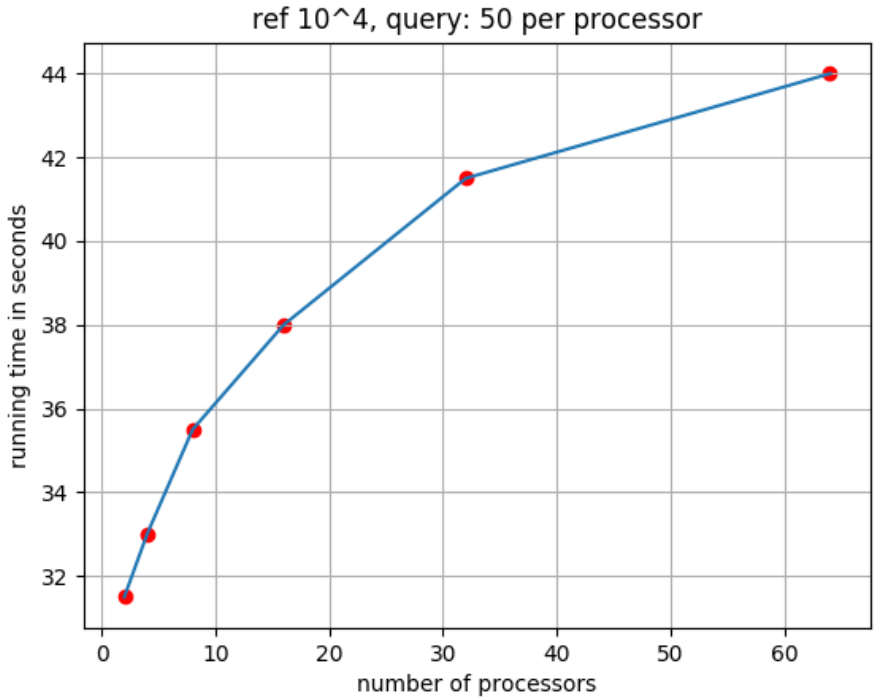


Running time on multiple nodes with 1 process per node, Increase query size 1.28*10^4 and reference size 10^5 and reference size

Running time get closer to logarithm curve

P	2	4	8	16	32	64	128	256
T	32535	24915	9075	4980	3315	1980	1201	902

Fix data per processor



P	2	4	8	16	32	64
T	31.5s	33s	35.5s	38	41.5s	44s

Running time on multiple nodes with 1 process per node, keep query per processor to be 50.

Running time slightly increase. Because the increasing of communication.

Reference:

- Liao, Hsien-Yu, Meng-Lai Yin, and Yi Cheng. "A parallel implementation of the Smith-Waterman algorithm for massive sequences searching." *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. Vol. 2. IEEE, 2004.

- Chaibou, Amadou, and Oumarou Sie. "Comparative study of the parallelization of the Smith-Waterman algorithm on OpenMP and Cuda C." *Journal of Computer and Communications* 3.06 (2015): 107.

- Baker, Matthew, Aaron Welch, and Manjunath Gorentla Venkata. "Parallelizing the Smith-Waterman algorithm using OpenSHMEM and MPI-3 one-sided interfaces." *Workshop on OpenSHMEM and Related Technologies*. Springer, Cham, 2014.

Thanks