Parallelization of the Smith-Waterman Algorithm with HPX

Stephanie Crillo,[1] Bryce Adelstein-Lelbach,[2,3] Hartmut Kaiser[2,3]
[1]Belmont Abbey College, [2]Lousisana State University, [3]LSU Center for Computation and Technology

STE||AR
stellar.cct.lsu.edu
LSU CENTER FOR COMPUTATION & TECHNOLOGY

Abstract

Concepts introduced by the Smith-Waterman algorithm are utilized by various bioinformatics systems today, such as BLAST. The algorithm, although accurate, slows greatly when working with larger data sets, due to its $O(n^2)$ complexity(1). We wrote a simple Smith-Waterman algorithm, in both serial and parallel, in an attempt to improve the scalability of Smith-Waterman, as part of our larger effort to improve the scalability of BLAST through parallelization.

Introduction

The Smith-Waterman algorithm, (Figure 1), is designed to find the optimal local alignment of two sequences.

$$H_{(i,j)} = max\begin{Bmatrix} H(i-1,j-1) + w(a_i, b_j) \\ H(i-1,j) + w(a_i, -) \\ H(i,j-1) + w(-, b_j) \\ 0 \end{Bmatrix}, 1 \le i \le n, 1 \le j \le m$$

Figure 1: The Smith-Waterman Algorithm
The first term describes a match/mismatch, the second term describes a deletion, and the third term describes an insertion.

The algorithm builds a matrix based on a specific scoring scheme (Figure 2). In our runs, matches were granted a score of +2, while deletions and insertions (gaps) were granted a score of -1.

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

Figure 2: Smith-Waterman Scoring Matrix
This matrix was built in the comparison of the two sequences TGCATACT (our "database" sequence) and TCGTATGT (our "queried" sequence).

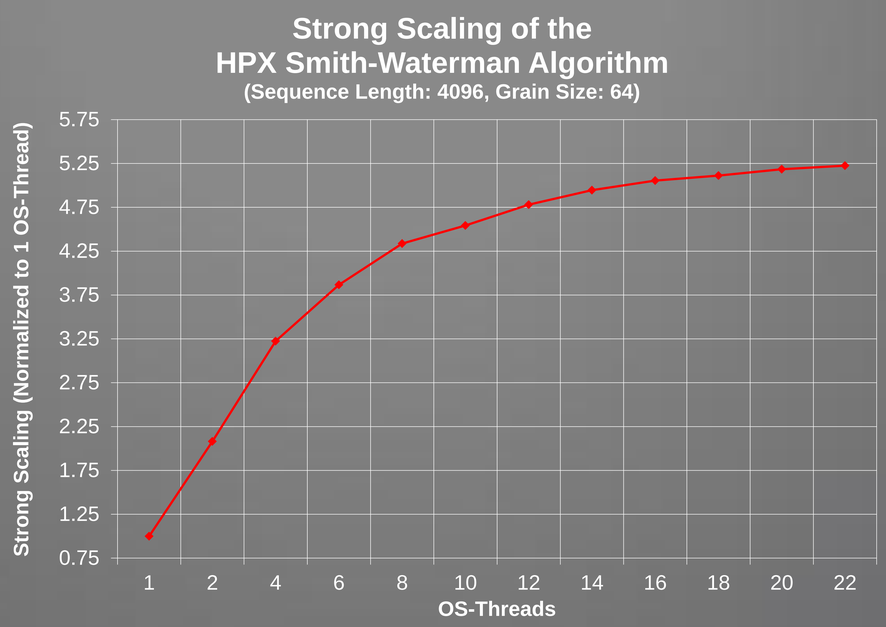
Upon completion of the matrix, the algorithm backtracks through the matrix to obtain an optimal alignment (Figure 3).

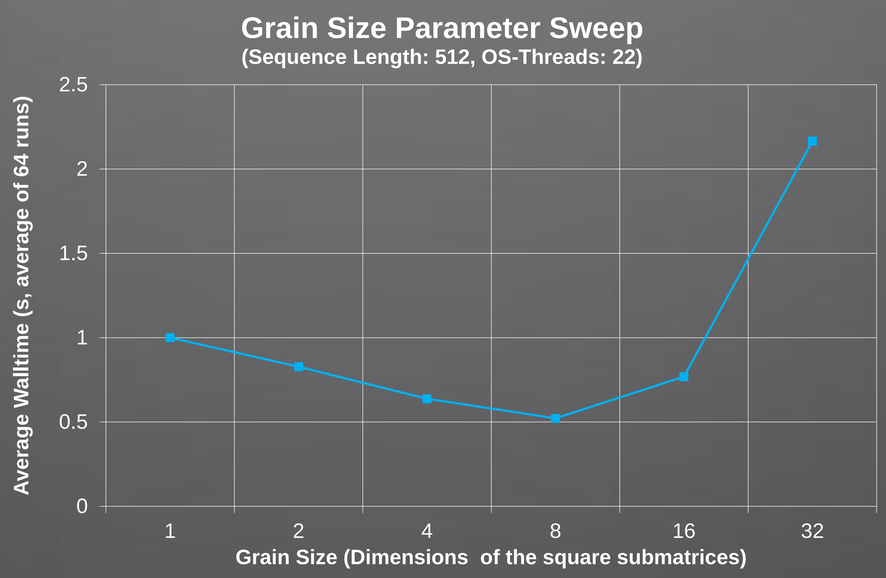
	–	T	C	G	T	A	T	G	T
–	0	0	0	0	0	0	0	0	0
T	0	**2**	1	0	2	1	2	1	2
G	0	1	**1**	3	2	1	1	4	3
C	0	0	**3**	2	2	1	0	3	3
A	0	0	2	**2**	1	4	3	2	2
T	0	2	1	1	**4**	3	6	5	4
A	0	1	1	0	3	**6**	5	5	4
C	0	0	3	2	2	5	**5**	4	4
T	0	2	2	2	4	4	**7**	6	6

Figure 3: Backtracking
Backtracking begins at the highest number(2) (here, 7), and continues through the matrix until a point surrounded by zeros is reached(2). Diagonal movement indicates a match or mismatch, top-down movement indicates a deletion, and left-right movement (not shown) indicates an insertion.

Results

Parallelization of the Smith-Waterman code was done by dividing the matrix into smaller submatrices and computing the scores of these submatrices in parallel where possible (see Figure 4 and 5). Additionally, backtracking of each of these components was computed as the matrix was being constructed.





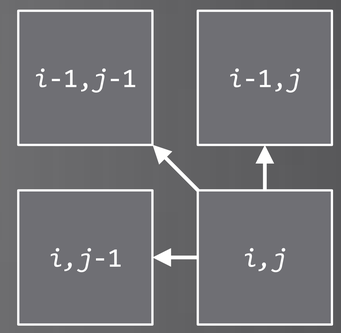


Figure 4: Dependencies of the Algorithm
To compute the value (*i, j*), the values immediately above, diagonal and left of (*i, j*) are needed.

Figure 5: Hot Zones
Darker, or "hotter" cells represent regions in the matrix which have a higher potential for parallelization, because these regions are independent of each other.

References

(1) Torbjørn Rognes, 2011. Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation. BMC Bioinformatics. [Internet]. [cited 2012 July 12]; 12:221. Available from: http://www.biomedcentral.com/1471-2105/12/221/.
(2) Smith T.F., Waterman M. S., 1981. Identification of Common Molecular Subsequences. Journal of Molecular Biology. [Internet]. [cited 2012 July 12]; 147:195-197. Available from: http://dx.doi.org/10.1016/0022-2836(81)90087-5.

Acknowledgements

- NSF Grants 1240655, 1117470, 1048019 and 1029161
- DOE ASCR X-Stack Program
- Center for Computation and Technology