Parallel Position-Specific Iterated Smith-Waterman Algorithm Implementation

Witold Rudnicki, Rafal Maszkowski,
Lukasz Bolikowski, Maciej Cytowski, Maciej Dobrzynski, Maria Fronczak
Interdisciplinary Center of Mathematical
and Computational Modeling,
Warsaw University

May 2006

ABSTRACT: *We have implemented a parallel version of Position-Specific Iterated Smith-Waterman algorithm using UPC. The core Smith-Waterman implementation is vectorized for Cray X1E, but it is possible to use an FPGA core instead. The quality of results and performance are discussed.*

KEYWORDS: Cray X1E, UPC, bioinformatics, Smith-Waterman, sequence alignment

1 Introduction

The aim of this paper is to briefly summarize some technical aspects of implementing the Position-Specific Iterated Smith-Waterman algorithm for local sequence alignment on Cray architecture.

The project was initiated at the Interdisciplinary Center of Modeling in 2002. At first, our goal was to test the capabilities of the BMM unit[1] by implementing the original Smith-Waterman algoritm on Cray SV1. At that time there was no *BioLib*[2] package, nevertheless an equivalent tool was necessary. It turned out that the Smith-Waterman algoritm itself cannot substantially benefit from the BMM, but a better understanding of the unit allowed us to design certain filters that would limit the number of calls to the S-W routine. This work was presented by Lukasz Bolikowski on CUG 2005 conference in Albuquerqe.

Finally, as Cray X1 appeared at ICM, it was feasible to implement the iterated version of S-W (PSI S-W), in a fashion similar to the PSI BLAST[3]. It was tempting to check whether such approach could be competitive to PSI BLAST in terms of speed, and whether the alignments obtained by PSI S-W could be substantially better than ones obtained by PSI BLAST. We have parallelized our code with UPC[4]

2 Theory

2.1 Sequence alignments

The purpose of Smith-Waterman algorithm is to find a *local alignment* of two *sequences* that is maximal according to the chosen *scoring system*.

The sequences are simply strings of letters: there are 20 letters to choose from in case of amino acids and 4 letters in case of nucleotides. Let us explain what a local alignment is by an example. Take two amino acid sequences:
`MDRKVTPGSTCAVFGLGGVGLSAIMGFIL`
and
`MKLNPGSSGHGGMGATMTSAVMGDRNN`.

[1] Bit Matrix Multiply (BMM) is a hardware functional unit available on Cray SV1 and Cray X1 architectures that performs fast multiplication of two 64×64 bit matrices (the adds and multiplies are *modulo* 2)

[2] Cray Bioinformatics Library (BioLib) is a set of routines for nucleotide and amino acid sequence manipulation

[3] Position-Specific Iterated Basic Local Alignment Search Tool (PSI BLAST) is the most popular tool for local sequence alignment. It is a very fast heuristic, while S-W is an exact and time consuming algorithm

[4] Unified Parallel C - is an extension of the C programming language designed for high-performance computing on large-scale parallel machines, including those with a common global address space (SMP and NUMA) and those with distributed memory (eg. clusters).

The optimal local alignment for the two is:

```
Query: 4  KVTPGSTCAVFGLGGVG---LSAIMG 26
          K+ PGS+    G GG+G    SA+MG
Sbjct: 2  KLNPGSS----GHGGMGATMTSAVMG 23
```

The top row is a fragment of the first sequence, the bottom row is a fragment of the second one. Each letter of a sequence is assigned to a letter in the other sequence, or to a gap between two consecutive letters in the other sequence.

A scoring system for all alignments shall be defined to select the optimal alignment from the set of all possibilities. The system used by the Smith-Waterman algorithm is based on the probability of various mutations, such as replacements, insertions and deletions, in the protein sequences.

The score of an alignment is computed as follows: each aligned pair of letters is given an integer value, and the values are summed up. Then each gap of length l is given a penalty $G + lE$, where G and E are constants named *gap opening penalty* and *gap extension penalty*. Finally, all the penalties are substracted from the sum and the result is the alignment score.

The scoring system is, thus, respresented by a symmetric table T of scores for each pair of letters, and a pair (G, E) of penalties. The table is 20×20 for amino acids and 4×4 for nucleotides. A fragment of BLOSUM62, a popular amino acid scoring table, is shown below:

	A	R	N	D	C	Q	E
A	4	-1	-2	-2	0	-1	-1
R	-1	5	0	-2	-3	1	0
N	-2	0	6	1	-3	0	0
D	-2	-2	1	6	-3	0	2
C	0	-3	-3	-3	9	-3	-4
Q	-1	1	0	0	-3	5	2
E	-1	0	0	2	-4	2	5

Note that the diagonal elements are always positive and greater than any other one in the row. This is natural, since alignment of a sequence A with itself (a perfect alignment) should have score greater than any alignment of A with a different sequence.

2.2 Smith-Waterman algorithm

The Smith-Waterman is a simple dynamic programming algorithm. Let A and B be the two sequences being aligned. It starts by allocating a table S of dimensions equal to the lengths of A and B. Then it fills the table from upper-left to bottom-right (by rows, columns or antidiagonals – any of these orders is good) with value:

$$S_{i,j} = \max \begin{pmatrix} S_{i-1,j-1} + T(A_i, B_j) \\ \max_{l=1,j}(S_{i,j-l} - (G + E * l)) \\ \max_{k=1,i}(S_{i-k,j} - (G + E * k)) \\ 0 \end{pmatrix} \quad (1)$$

G and E here stands for gap penalties: G when a gap is opened and E, when it is extended. The intuitive meaning of a value $S_{i,j}$ is the score of the best local alignment ending at A_i, B_j.

At each step, when a maximum value is found, it should be recorded from which direction the result was obtained: diagonal, top, left, or none. The table S together with information about the directions provide enough data to find the best local alignment.

To find the best alignment, one has to do as follows. First, find the maximum value within table S. This is the end of the highest scoring local alignment. Then, backtrace from the cell in the recorded direction. A diagonal move from $S_{i-1,j-1}$ to $S_{i,j}$ represents an aligned letter pair A_i, B_j; a horizontal move from $S_{i-1,j}$ to $S_{i,j}$ represents A_i aligned with a gap between B_{j-1} and B_j; a vertical move is analogous to the horizontal one; while max $= 0$ at $S_{i,j}$ means the optimal local alignment starts at $S_{i+1,j+1}$.

2.3 Database searches and PSI S-W

A typical use of a local alignment algorithm is such: one sequence, *a query*, is aligned against *a database* of sequences, one by one. The most significant alignments, and their corresponding sequences are presented to the user.

Position Specific Iterated Smith Waterman (PSI-SW) is an extension of the original SW algorithm analogous to the PSI-BLAST extension of the BLAST algorithm. In this algorithm the single scoring table is replaced by the position specific scoring matrix (*PSSM* or *profile*). A set of unique 20 scores for each amino acid in the query sequence is used. These scores are obtained in the iterative self-consistent procedure. In the first step the scores from the similarity matrix, such as BLOSUM62 are used to find the set of homologous sequences. Then all sequences are aligned with the query and frequencies of the aminoacid appearance at each position is

computed and translated into the scores in the position specific scoring matrix. Then a new search for the homologous sequences is performed using new PSSM, presumably leading to finding more homologous sequences than in the search with the original socring matrix. This procedure is repeated until no more new sequences are found.

3 Implementation

3.1 Core S-W

The core Smith-Waterman algorithm is quite straightforward to implement. One crucial observation is that updating the S table should be done by antidiagonals, since all operations on a antidiagonal are then independent of each other, which enables vectorization.

3.2 PSI S-W

The implementation of PSI S-W algorithm was also straightforward. It required certain changes to the original S-W code, since the iterated version introduces Position-Specific Scoring Matrices (PSSM) instead of the traditional scoring tables.

3.3 Parallel UPC implementation

Application of the Smith-Waterman algorithm for the database search is a naturally parallel task, since one can split the task into N sub-tasks, each on the seperate fragment of the database. We use this approach in this study.

There are two possible bottlenecks in the parallel implementation of the algorithm. First is a possible load inbalance due to the unbalanced database split. We applied the following procedure to minimize this inbalance: All proteins in the database were sorted with respect to their lenght. Then, starting with the shortest sequences we put each sequence one by one into actually smallest set. Note that usually one compares sequences to some well known databases which can be split into N parts in advance.

Another possible bottleneck arises due to the application of the parallel library in the implementation. The library enforces co-ordination between processors, which may slow down the execution, when proteins of different lengths are executed on different CPU's. Nevertheless we decided to use the parallel library, because we need to collect the results from each processor after each single iteration in order to compute a new position specific matrix. Such a matrix is than used as a substitution matrix in next iteration. The UPC was used, because it is user friendly parallel library, which is also well suited for the CrayX1E architecture.

3.4 Fast S-W step

Classical S-W algorithm consists of two main steps:

- finding the maximal score by dynamic programming scheme
- reconstructing the path to the maximal score

To perform the second step one needs to keep aditional data to remember where we came from in each single entry of our dynamic programming matrix. This slows down the code considerably. Also, usually the number of statistically significant hits is significantly smaller then number of proteins in database. Therefore we don't need to perform path reconstruction in most cases.

We have implemented two S-W procedures: one which finds the maximum score and one which performs full S-W algorithm including the path reconstruction step. Each time we run the first fast procedure to check if E-value of the best alignment is better than preset threshold. If it does, the full S-W procedure is called. On average this implementation is 50% faster than full S-W procedure.

3.5 FPGA implementation

The fast S-W step can be even faster if we use a special architecture. The work towards implementation of the Smith-Waterman algorithm are under way in our laboratory. Preliminary tests showed that FPGA implementation can be up to 100 times faster then a standard PC implementation, comparable in speed with BLAST. This opens very interesting option for implementation of the PSI-SW for Cray XD1 architecture.

3.6 Statistics, user interface

The tool output is similar to the reports generated by BLAST for at least two important reasons: the reports are easy to read, and is familiar for most potential users.

An important step in selecting the alignments that are to be reported to the user is to assess their

statistical significance. The tool assesses the significance using Karlin-Altschul-Dembo theory, the same that is used by BLAST ([4], [5]).

4 Performance

4.1 Quality of results

We have performed the test using the SCOP classification of the protein domains. We checked sensitivity of the searches, for all proteins from PDB with similarity lower than 98%. We compared results of PSI BLAST and PSI S-W algorithms.

The preliminary results suggest, that for identical parameters for the PSI procedure, the PSI SW has slightly higher sensitivity, but lower specificity. The testing procedure is under way and results will be reported elsewhere.

4.2 Speed of execution

We have tested the scaling of our parallel implementation. The results are shown in Figure 1:

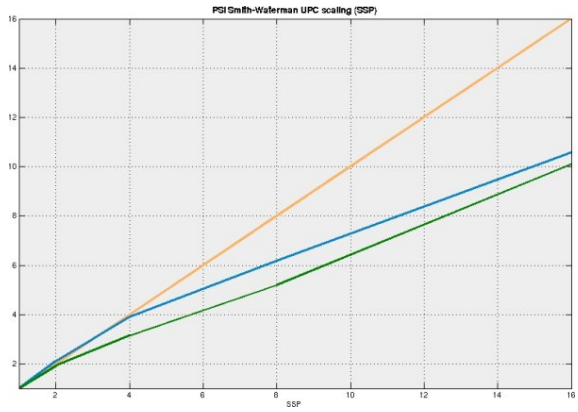

Fig.1 Scaling of PSI S-W parallel UPC implementation on Cray X1E.

5 Conclusions

The Smith-Waterman algorithm has been implemented on X1 architecture, independently from the implementation included in Cray BioLib, with similar performance. The parallel UPC version of the program has been implemented, and reasonable parallel scaling has been achieved. The performance of the code on the vector architecture is significantly higher than on the PC, neverthless it is still at least order of magnitude slower than that of PSI-BLAST - the standard tool used for homology searches. Implementation of SW core algorithm to the FPGA architecture is underway in our laboratory, which might have performance comparable to current implementations of BLAST.

Sensitivity and specificity of the PSI S-W has been compared with PSI BLAST, with PSI S-W displaying slightly higher sensitivity and PSI BLAST displaying higher specificity, for the same set of PSI parameters.

About the authors

The project was led by Dr. Witold Rudnicki, Deputy Director, HPC Division, ICM. His scientific research focuses on bioinformatics and HPC. He can be reached at: ICM Warsaw University, Pawinskiego 5A blok D, 02-106 Warsaw, Poland, e-mail: rudnicki@icm.edu.pl. Lukasz Bolikowski, Maciej Cytowski, Maria Fronczak and Rafal Maszkowski are Software Developers at ICM. Maciej Dobrzynski was a student of Dr. Rudnicki in an early stage of the project.

A large portion of work was done by Witold Rudnicki and Rafal Maszkowski, who initiated the project. Witold was responsible for the theoretical design, while Rafal implemented the core S-W.

References

[1] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman D.J. (1990) Basic local alignment search tool, *J. Mol. Biol.,* **215**, 403-410.

[2] Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389-402.

[3] Henikoff, S. and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks, *Proc. Natl. Acad. Sci. USA,* **89**, 10915-10919.

[4] Karlin, S., Altschul, S.F. (1990) Method for assessing the statistical significance of molecular sequence features by using general scoring schemes, *Proceedings of the National Academy of Science, USA* **87**, 2264-2268.

[5] Karlin, S., and Dembo, A. (1992) Limit distributions of maximal segmental score among markov-dependent partial sums, *Advances in Applied Probability* **24**, 113-140.

[6] Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison, *P. Natl Acad. Sci. USA,* **85**, 2444-2448.

[7] Wootton, J. C. and S. Federhen (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Computers in Chemistry* **17**, 149-163.

[8] Wootton, J. C. and S. Federhen (1996). Analysis of compositionally biased regions in sequence databases. *Methods in Enzymology* **266**, 554-571.