W-Period Technique for Parallel String Matching

S.Viswanadha Raju [†] , A.Vinaya Babu [††] , G.V.S.Raju [†††],, and K.R. Madhavi [†],

[†]Gokaraju Rangaraju Institute of Engineering and Technology, [†††] SCET, [††] JNTUniversity, Hyderabad, 500072, India.

Summary

In this paper, we present new approach for parallel string matching. Some known parallel string matching algorithms are considered based on duels by witness which focuses on the strengths and weaknesses of the currently known methods. This has applications such as string databases, Information Retrieval and computational biology. The new 'divide and conquer' approach has been introduced for parallel string matching, called the W-period, which is used for parallel preprocessing of the pattern and has optimal implementations in a various models of computation. The idea, common for every parallel string matching algorithm is slightly different from sequential ones as Knuth-Morris-Pratt or Boyer-Moore algorithm.

Key words:

Divide and conquer, duel, PRAM, string matching, W-period

1. Introduction

String matching deals with the problem of finding all occurrences of a single string in a given text. This string matching is one of the most extensive problems in computer technologies during past two decades. Pattern matching is widely implemented in information retrieval, web search engine, DNA sequencing, artificial intelligence and several other fields.

String matching problem can be defined as: Given an alphabet (finite sequence of characters) ĉ, a text string T = T[1] T[2]T[3]……T[m] of length m and a string pattern P= P[1]P[2]P[3]…..P[n] of length n where both text and pattern string are sequences of character from ĉ with n<=m. Sequential algorithms for this string matching problem can be found in [2,6,9,10]. The string matching problem solved in two stages namely preprocessing and searching by different authors. As the goal of preprocessing the pattern, one has to compute all periods of the pattern and all witnesses against nonperiods. In addition, super fast string matching algorithms need effective computation of the sequence of deterministic samples. There are two optimal algorithms known for computing witnesses and periods in the CRCW PRAM[8]. The first logarithmic one was introduced by Vishkin [7]. The next double logarithmic one was presented by Breslauer and Galil in [1]. The cost of the preprocessing in

super fast string matching algorithms was forced by the computation of the deterministic samples. Both algorithms (those of Vishkin and Galil) have expensive ($\log^2 m/\log\log m$) - time preprocessing. But there were no known optimal algorithms for computing periods and witnesses in other PRAM models, body centered hypercube [12] and 2D-mesh[13]. The problem of computing all periods and witnesses became the bottleneck in the pattern preprocessing. These problems can be solved by W-period technique.

The rest of the paper organized as follows: In section 2, we provide brief description of duel technique with an example. In section 3, the searching stage, all text positions are potential occurrences of the pattern .In section 4 we introduce new approach for our parallel string matching algorithm and Conclusions are reported in section 5.

2. Duel Technique

Duel technique is a natural scheme for parallel computation. It was originally used to compute the minimum (election of a leader for instance) among n elements [7].

2.1 Dueling in Group of Pairs

Consider the minimum finding problem for n-size input. For simplicity of presentation all the numbers may be assumed to be different. Dueling algorithms for the CREW and EREW PRAM models have a simple scheme in common. All computations are divided into logarithmic number of stages. The output of the previous stage is an input for the next one. At every stage all candidates (minimum) are grouped in pairs. Every pair of candidates performs a duel. A candidate who has smaller value survives, while the other loses and is not considered as a candidate for minimum later. After every stage the

Manuscript received September 5, 2007
Manuscript revised September 20, 2007

number of candidates is decreasing by a half. The working time of the algorithm corresponds to equation $T(n) = T(n/2) + O(1)$. There are no read/write conflicts, so simply we get logarithmic approach for both CREW and EREW PRAM models, and has been obtained [12].

2.2 Dueling by Squaring

A dueling idea has more efficient application for the CRCW PRAM model.

Fact 1

It is possible to find the minimum between n numbers, in constant time, in case when we use $O(n^2)$ processors.

Proof: Every input number has n processors to perform in one step and duels with all other candidates. After this step every candidate has associated binary vector of length n (1- won, 0- lost duel). Only one candidate, who won all duels, has associated binary vector filled by ones. This could be recognized by computing boolean n-bit AND in constant time. The candidate with positive AND is the desired minimum.

The main algorithm for the CRCW PRAM model could be presented by the simple recursive approach. When the whole n-size input is divided into $\sqrt{n}$ consecutive $\sqrt{n}$ - size sub blocks, in every sub block (independently in parallel) minimum can be found recursively. After coming back from recursive calls in every sub block, there is still only one candidate. The total number of candidates is $\sqrt{n}$, to compute the minimum among $\sqrt{n}$ candidates in constant time with n processors, according to fact 1. The total time $T\{n)$ of the algorithm satisfies the equation $T(n) = T(\sqrt{n}) + O(1)$, which finally gives double logarithmic working time O(log log n).

2.3 Optimality

It may be noted that both the algorithms are not optimal. Both use linear number of processors. Thus the work of the first one is O (n log n) and the second one is O(n log log n). To get an optimal (linear in the sense of work) logarithmic time algorithm use only O(n/log n) processors must be used. In the first step every processor gets O (log n) numbers and searches sequentially looking for minimum. After this step there are only O(n/log n) candidates. Now nonoptimal logarithmic time algorithm can be used with linear number of processors. Finally an optimal O(log n)-time algorithm can be obtained and implemented on O(n/log n) processors.

Similarly, to get an optimal double logarithmic time algorithm every processor first finds minimum in its O(log log n)-block, and then non optimal O(log log n) time algorithm on reduced input can be used.

3. Searching a text

In parallel string matching, a duel plays an important role too. In the beginning of the searching stage all text positions are potential occurrences (candidates) of the pattern. We can perform duels by witnesses, between candidates, according to rules given below.

3.1 Duels for a text searching

Assume that all witnesses (existing against all positions $r \leq m/2$) of the pattern have been computed. Two candidates $r < s$ may be considered such that t is a witness against s - r (i.e., $P_t \neq P_{t-s+r}$). See fig. 3.1.

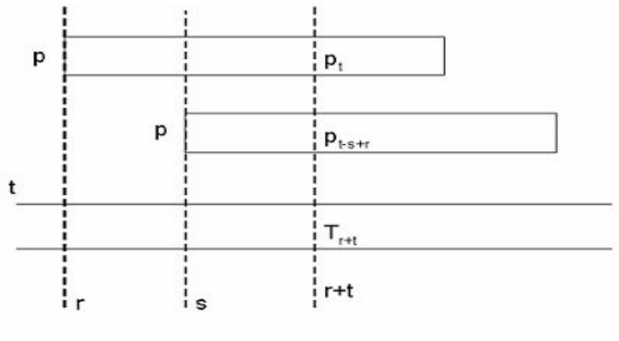

Figure 3.1: A duel between candidates r and s

- If $T_{r+t} \neq P_t$, r + t is a witness to non-occurrence r,
- IF $T_{r+t} = P_{t-s+r}$, r + t is a witness to non-occurrence s.

The two tests above are called a duel between r and s [7]. A duel can remove one or both the candidates. Having witnesses, the string matching searching stage in logarithmic time for the CREW PRAM model can be implemented optimally (similar to minimum finding algorithm). The string matching searching can be implemented in double logarithmic time for the CREW PRAM model.

Assume that pattern P, $|P| = m$, is nonperiodic and during the preprocessing we have computed all witnesses against all positions $r \leq m/2$, the whole text T, $|T| = n$, be divided into consecutive blocks of size m/2.

Fact 2

There is at most one occurrence of nonperiodic pattern P.

$|P| = m$, in every m/2 -block of the text *T*.

***Proof*:** If there are two occurrences of the pattern P, r and s, such that r < s, in some m/2-block of *T* implies that pattern P has a period s - r < m/2, which contradicts that pattern P is nonperiodic.

In every m/2-block, duels may be performed by witnesses and finally at least one candidate among m/2 ones. Afterwards there are at the most O (n/m) candidates in the whole text. Every candidate could be checked naively if it is a real occurrence with work O (m). So the total work of checking survivors is O(n/m, m) = O(n). Finally linear work algorithm is obtained according to Brent's theorem, could be implemented optimally in O (log m) time in the CREW PRAM model and in O(log log m) time in the CRCW PRAM model [5].

4. W-period Technique

As the goal of preprocessing the pattern, one has to compute all periods of the pattern and all witnesses against nonperiods. As a contrast to CRCW PRAM model, there were no known optimal algorithms for computing periods and witnesses in other PRAM models, body centered hypercube or 2D-mesh. The new 'divide and conquer' parallel approach for computing the period and witnesses of the pattern, called a w-period technique has been introduced.

The W-period has an operational definition. Given a string z of length r, we compute a W-period q by computing witnesses against all non-multiples $i \le r - \Psi(r)$ of q, where function $\Psi(r)$ is relatively small. This study holds that $\Psi(r) = O(\sqrt{r})$ will be used.

Fact 3
If a word z is periodic with the period p ($p \le |z|/2$) the W-period q must divide the period p of z.

Proof: Since it has computed all witnesses against all nonmultiples $i < |z|/2$ of q, the short period must be divisible by q.

In the case of the period of a word z, it may the noted that it has already computed W-period q. Witnesses for other positions also have already been computed. Desired witnesses can be caused only by symbols with distance iq apart, for all $1 \le i \le |z|/q$. This gives a natural division of z into q distinct classes of symbols forming a set of new smaller words:

$z^{(0)} = z_0 z_q z_{2q}..$, $z^{(1)} = z_1 z_{q+1} z_{2q+1}..,\ldots$, $z^{(q-1)} = z_{q-1} z_{2q-1} z_{3q-1}\ldots\ldots$

Compute the real periods $p^{(0)}p^{(1)},\ldots, p^{(q-1)}$ and witnesses recursively, for all words $z^{(0)}z^{(1)} .., z^{(q-1)}$, independently in parallel using recursive method.

Fact 4

The period p of the whole word z is determined by the $\mu.q$, where $\mu = \mathrm{LCM}(p^{(0)}, p^{(1)}, \ldots.. p^{(q-1)})$ and LCM stands for the lowest common multiple.

Proof: Witnesses against all nonmultiples of $\mu.q$ are computed.

4.1 Example

Let us consider a word z =vuuvuuuuvwvuuvuuuuvwvuuvu, |z| = 25, and assume that somehow W-period q = 5 for z which have been computed. This means that all witnesses against positions i < 20 (let $\Psi(25) = 5$) have been computed already.

$WITN_z[0,..,19]$=[0,1,2,7,4,0,6,7,9,9,0,11,12,16,14,0,16,17, 19,19]
It may be noted that for all multiples i of 5 $WITN_z[i] = 0$. The period of z is a multiple of 5 in case z is periodic. The word z is divided into 5 blocks (of length 5) and ith class $z^{(i)}$ is formed by ith symbols from every block:
z = vuuvu|uuuvw|vuuvu|uuuvw|vuuvu
$z^{(0)}$ = vuvuv, $z^{(1)}$ = uuuuu, $z^{(2)}$; = uuuuu. $z^{(3)}$ = vvvvv, $z^{(4)}$ = uwuwu with corresponding periods $p^{(0)} = 2$, $p^{(1)} = 1$, $p^{(2)} = 1$, $p^{(3)} = 1$, $p^{(4)} = 2$. The period p of z is $\mu.q$ = LCM (2, 1, 1, 1, 2). 5 = 10.

Finally, to compute witnesses (unknown before) for nonmultiples of the period 10 in z: $WITN_z[5]$ and $WITN_z[15]$. Both witnesses are reconstructed from witnesses of z (°)…z $^{(4)}$ where:
$WITN_{z(0)}[]$ =$WITN_z()$ = [0,1,0,3, 0] and
$WITN_z(1)$ [] =WITNz(2) [] =$WITN_{z(3)}[]$ = [0,0,0,0,0]
In the following way
$WITN_z[5]=5.WTTN_{z.(o)}[1] + 0 = 5 . 1 = 5$ or equivalently
$WITN_z[5] = 5.WITN_{z(2)}[1]+4 = 5.1+4 = 9$ and

$WITN_z[15] = 5.WITN_{z(o)}[3] + 0 = 5.3 = 5$ or equivalently
$WITN_z [15] = 5.WITN_{z(4)}[3] + 4 = 5.3 + 4 = 19$.
Witness is chosen in an arbitrary manner. This means in practice, we take first one from the left.

4.2 General structure

Finally a new 'divide and conquer' parallel approach for computing the period of a given string has been obtained.
Algorithm FIND-PERIOD(z, |z|);
1. Find a W-period q of z;

2. Divide the word z into q substrings $z^{(i)} = z_i z_{i+q} z_{i+2q} \ldots$ for all i = 1,.....,q;
3. For all substrings $z^{(i)}$ in parallel do
   $P^{(i)}$ = FIND-PERIOD($z^{(i)}$,|z|/q);
4. Collect periods $p^{(i)}$ and witnesses from preprocessed $z^{(i)}$s (multiples of q) and compute $\mu = LCM(p^{(0)}, p^{(1)}, \ldots, p^{(q-1)})$;.
5. Return (μ.q).

Note that algorithm FIND-PERIOD computes also all witnesses against nonperiods of P. Extending witnesses from $P^{(i)}$ to P is done according to the following rule:

$WITN_p[j * q] = q.WITNP_{p(i)}[j] + i$

5. Conclusion

We studied some known parallel string matching algorithms based on duels by witness which focuses on the strengths and weaknesses. We introduced W-period technique for computing period and witnesses of pattern. Finally, a general structure to find a period for pattern string was designed. Our future research, is to implement the FIND-PERIOD algorithm for various parallel models as PRAM's, Body centered hypercube [12] and two dimensional mesh structure [13].

Acknowledgements

We would like to thank our colleagues in the Department of Computer Science and Computer applications for their advice and helpful discussions. We thank Prof P.S.Raju, Director, and Dr.J.N.Murthy, Principal, Dr.Eliah Professor in English, GRIET for their moral support and providing Laboratory. We are grateful to the British Library, JNTUniversity Library for their services.

References

[1] D. Breslauer and Z. Galil, A lower bound for parallelstring matching, SIAM Journal of. Computing 21, 1992, pp. 856-862.
[2] M.Crochemore and W.Rytter, Text Algorithms, Oxford
University Press, 1994.
[3] Z.Galil and K.Park, Truly alphabet-independent two-
Dimensional pattern matching,Proc 33rd IEEE symposium, Foundation of Computer science, 1992, 247-256.
[4] Z. Galil, A constant-time optimal parallel string-matching algorithm, Proc. 24th Annual ACM Symposium on Theory of Computing, 1992.
[5] Z.Galil, Optimal parallel algorithms for string matching, Information and Control 67 1985, pp. 144-157.
[6] D.E. Knuth. J.H. Morris, and V.R. Pratt, Fast pattern matching in strings, SIAM Journal of Computers, 6, 1977, pp. 323-350.
[7] U.Vishkin, Optimal parallel matching in strings, Information and Control 67 ,1985, . 91-113.
[8] S.Viswanadha Raju and A.Vinayababu, (2004) "Performance in the design of Parallel Programming", *Proc ObComAPC-2004, Allied Publications*, pp.380 to 392.
[9] S.Viswanadha Raju,A.Vinayababu and M.Mrudula ,(2006) "Backend Engine for Parallel string Matching using Boolean Matrix" , *Proc PARELEC-2006, IEEE Computer Society , 281-283.*
[10] S.Viswanadha Raju,S.R.Mantena,A.Vinayababu and GVSRaju, (2006) "Efficient Parallel String Matching Using Partition Method", *Proc PDCAT-2006,IEEE Computer Society, 281-284.*
[11] S.Viswanadha Raju, A.Vinayababu,S.P.Yanaiah and GVSRaju, (2006) "Parallel Approach for K String Matching", *Proc NCIMDiL-2006, Indian Institute Of Technology, Kharagpur , 5-10.*
[12] S.Viswanadha Raju and A.Vinaya Babu, (2006) " Optimal Parallel String Matching Algorithm on Body Centered Hypercube", *International Journal of Mathematics and Computer Science*,**1** No.4, 473-484.
[13] S.Viswanadha Raju and A.Vinaya Babu, (2006)" Optimal Parallel algorithm for String Matching on Mesh Network Structure", *International Journal applied mathematica Sciences*, **3** No.2, 167-175