Global Journals LaTeX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

Analysis of Parallel Boyer-Moore String Search Algorithm

Abdulellah A.Alsaheel, Abdullah H.Alqahtani[1]

[1] King Saud University

Received: 7 December 2012 Accepted: 31 December 2012 Published: 15 January 2013

Abstract

Boyer Moore string matching algorithm is one of the famous algorithms used in string search algorithms. Widely, it is used in sequential form which presents good performance. In this paper a parallel implementation of Boyer Moore algorithm is proposed and evaluated. Experimental results show that it is valuable with zero overhead and cost optimal. The comparison between sequential and parallel showed that the parallel implementation was faster and more useful.

Index terms— parallel processing, parallel algorithm, boyer-moore, string matching algorithm, performance analysis.

1 Introduction

tring matching algorithm is a huge area used when there is a need to search for the occurrence of a pattern of a string in a text. The applications of string matching algorithms is widely used like in security field when a protection system searching for a malicious pattern in a big text string. Boyer Moore is one of the most algorithms used in this field. The Boyer Moore algorithm search for the occurrence of pattern P **??**1-N] in a text T[1-M] by comparing the pattern in the text from right to left and shifting the pattern from left to right when mismatch occur using one of two functions that are bad character shift or good suffix shift [1].

The sequential Boyer Moore searching algorithm requires O(NM) in worst case when the pattern occurs in the text. And in the best performance is O(M/N) [4], where length of a pattern is N, and length of a text string is M. Actually, This algorithm is not efficiently working with small size strings but it is efficient with big size strings.

Figure **??** : BM algorithm mechanism make comparison from right to left and shift from left to right when mismatch occur In this paper implementing a parallel Boyer Moore is proposed and evaluating its performance and compare it with sequential version. Most of performance metrics used in evaluating any algorithm is used to make a clear comparison between sequential and parallel algorithm and presents valuable metrics for algorithm.

2 II.

3 Related Works

A few works on parallelizing Boyer Moore algorithm proposed. In [2] implements a parallel algorithm in clustered computing environment for many kinds of string matching algorithms and Boyer Moore algorithm is one of them. They made a comparison between the performance of this algorithm in sequential and parallel on different sizes of data and in different number of nodes. The result for their experiment and comparison demonstrated that Boyer Moore in parallel is taken less time than in sequential.

Authors ? ? ? : King Saud University, College of computer and information sciences, Saudi Arabia. E-mail : cs_saheel@hotmail.com Popa et al [3] has made an implementation for a parallel algorithm for Boyer Moore and calculating the speed up and efficiency in various numbers of processes. He used single program multiple data stream (SPMD) as a model and C as programming language with using parallel virtual machine (PVM) functions.

4 III.

5 The Parallel Algorithm

Parallel processing is the concept of executing the same problem on different processing units concurrently. The parallel implementation is made to work on one control unit with different data that called single instruction stream multiple data stream (SIMD). The communication model is shared address space where the algorithm implemented on the same platform.

In this algorithm different tasks assigned to P processors to present results. While these tasks are homogeneous and there are no dependencies between them, the bag of tasks has been used as a design pattern for the parallel algorithm. The idea is to divide the text T to multiple partitions and make the comparison over the same copy of pattern and every worker concurrently implement M/P [1] where M is the length of text T. The partitioning is occurred between lines of file text and cannot be occurred in the middle of line and assuming that the pattern that being searched for, will not be occurred between any two lines. a) Theoretical Analysis Sequential algorithm has been evaluated by its serial runtime ?? ?? , that taken from the start of the execution of the algorithm to the end of it. The sequential time of Boyer Moore when pattern occur is ?? ?? =??(????) in worst case [4] [5]. The parallel runtime that denoted by ?? ?? is the time taken from the first processor starts to the end of last processor. When our algorithm depends on dividing a text ?? on the available workers P, the parallel time will be then ?? ?? =??(* ??) and the total cost is ???? ?? = . In the following some of the most famous performance metrics used for parallel systems measurement applied on the proposed parallel algorithm [6]:

? Total overhead: the total time that spent by processers over sequential time ?? ?? =???? ?? ? ????=0.

? Speed Up S: the ration of time taken by the algorithm taken in sequential to the same algorithm in parallel, S = P .

? Efficiency E: is the relation between speed up and number of processors, E = 1 which is in this case is optimal.

By applying these metrics on parallel algorithm, it showed that using Boyer Moore parallel algorithm is more efficient and better than using Boyer Moore sequential algorithm.

6 b) Practical Analysis

The algorithm has been implemented by Java programming language with having sequential and parallel versions of Boyer-Moore algorithm. The experiment has been applied on a platform with 8 processing units and 8 GB RAM (Random Access Memory). This implementation used bad rule character that scan the pattern starting from the rightmost character against the text and the pattern will be shifted to the right whenever a mismatch occurs, pattern shifting will be according to the number of positions that stated at skip table that has stored the right most occurrence of that mismatched character at the pattern. The performance of the sequential algorithm is stable and easy to read as Table1 has demonstrated, since adding 500 elements each time will requires 50 Milliseconds approximately; despite the number of data elements being processed it will usually consume the same amount of time for processing the same amount of data. On the other hand there is the performance of the parallel processing which is presented at Table1 and may requires more analysis to be understood, however the table demonstrated that adding 500 data elements at each time will affect the processing time significantly whenever N is small; whereas adding the same amount of data to large number of elements will not affect the processing time that much. Conclusion is that the parallel version of Boyer-Moore algorithm works better with larger data. Adding more workers will significantly improve the processing time especially whenever the number of workers is small. Notice that adding more workers not necessarily will improve processing performance, this due to the fact that Boyer-Moore algorithm will have worse case whenever there are more mismatches which require more shifting to do; and since the parallel algorithm has partitioned the data across multiple workers, that partitioning sometimes may not guarantee a better performance then what would you have if you have less workers especially whenever the number of workers is large where adding one more worker may not improve the performance significantly. Some data partitioning introduces an additional delay since some data portions that held by some workers may have more mismatches and more shifting then what other workers may need. So the data nature and the number of workers are critical for the parallel algorithm performance especially when there is large number of elements to be processed and where adding more workers may not significantly improve the performance. To have more workers than the existing number of processors may lead to significant performance improvement, since having more workers means having smaller sub-problems which are faster to be processed. But one should consider that having more workers than the existing number of processors will introduce an additional overhead due to the context switching. So, identifying a threshold-like which specifies the maximum allowed number of workers that can work safely in performance wise. The experiment showed that on a platform with P=8 processors, there is a safe and better performance can be achieved by having 2 * P = 16 threads. Increasing the number of threads more than that may result in better or worse performance so one is advised to not exceed that threshold due to instability.

7 Conclusion

Sequential and parallel implementations of Boyer-Moore string matching algorithm have been evaluated. Theoretically, it proved that the performance of the parallel algorithm is cost optimal with zero overhead. In practical experiment, different sizes of data have been used to show that the parallel algorithm is the data is large. Also, using different number of processors we demonstrated that the parallel Boyer-Moore works better when the number of workers get increased unless the new data partitioning result in having some workers with data has many shifting which end up in delaying all of the workers. Number of workers/threads threshold has been proposed that can specify the number of threads which can be used efficiently with having stable performance, 2 * ?? ?????????????, where P is the number of existing processors.

8 Global

2

Figure 1: Figure 2 :

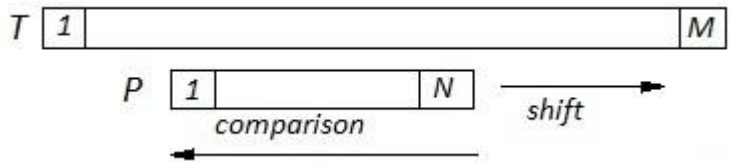
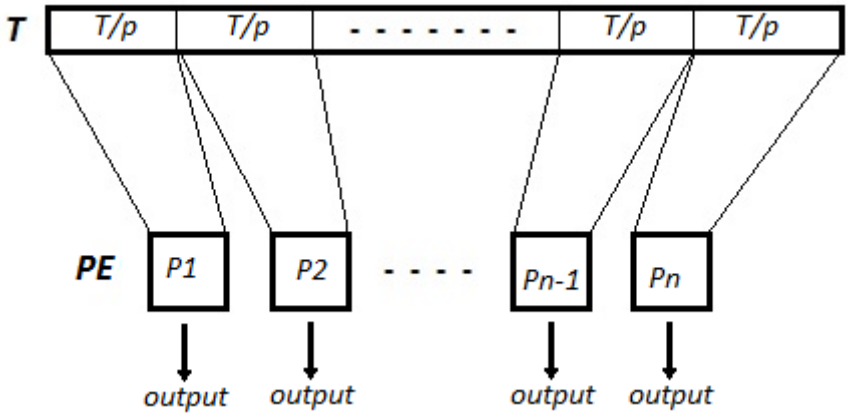


Figure 2:

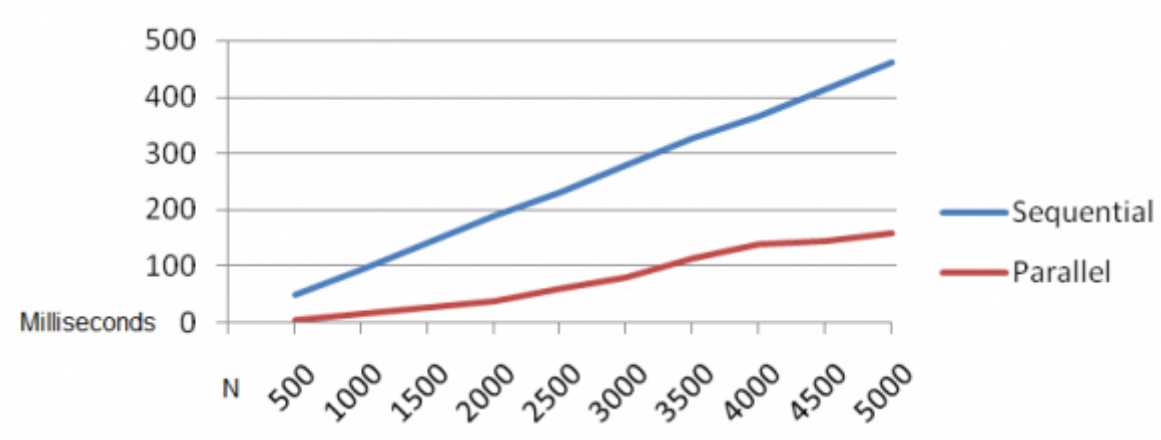
[1]

[1] © 2013 Global Journals Inc. (US)



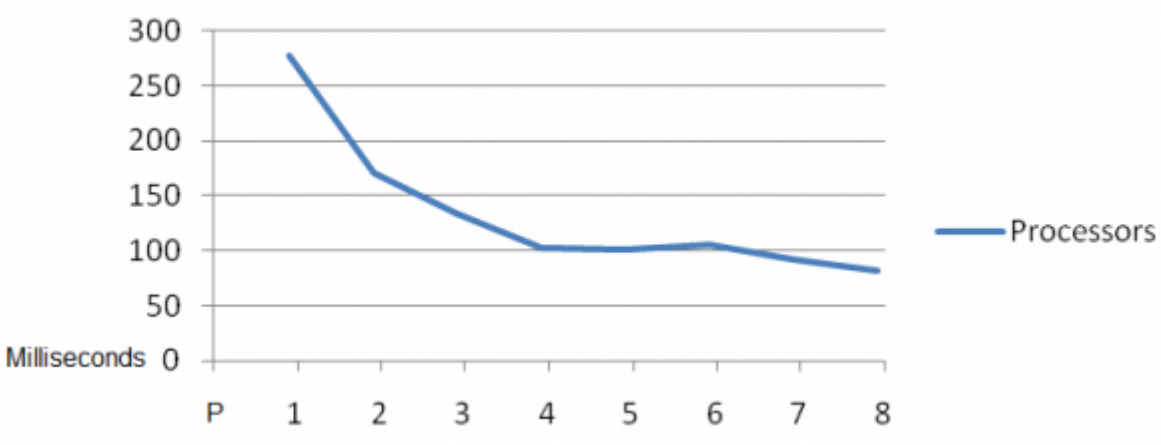
3

Figure 3: Figure 3 :



4

Figure 4: Figure 4 :



5

Figure 5: Figure 5 :

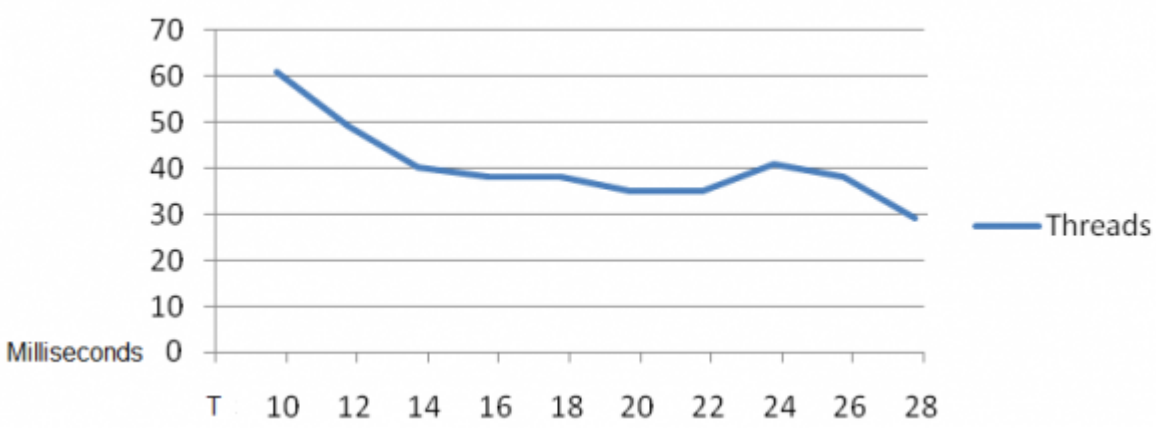


Figure 6:

1

```
    int tableSize;
    int[] table;
    String pattern;
    public BoyerMoore(String pattern) {
?   ? ?                 this.tableSize = 256; this.pattern = pattern; table = new
?                       int[tableSize];
*
?
                        for (int c = 0; c < tableSize; c++)
                        table[c] = -1;
                        for (int j = 0; j < pattern.length(); j++)
                        table[pattern.charAt(j)] = j;
    }
    public int search(String text) {
                        int N = pattern.length();
                        int M = text.length();
                        int skip;
                        for (int i = 0; i <= M -N; i += skip) {
                        skip = 0;
                        for (int j = N-1; j >= 0; j–) {
                        if (pattern.charAt(j) != text.charAt(i+j)) {
                        skip = Math.max(1, j -table[text.charAt(i+j)]);
                        break;
                        }
                        }
                        if (skip == 0)
                        return i;
                        }
                        return M;
    }
```

Partitioning the text into sub-amounts of work will divide the problem into smaller sub-problems, so

Figure 7: Table 1 :

[Grama et al. ()] , A Grama , G Karypis , V Kumar , A Gupta . 2003. Addison-Wesley. (Introduction to parallel computing, second edition)

[Boyer and Moore (1977)] 'A fast string searching algorithm'. R S Boyer , J S Moore . *Communications of the ACM* Session10, Oct.1977. 20 p. .

[Breslauer and Galil ()] 'A Parallel implementation of Boyer-Moore String Searching Algorithm'. D Breslauer , Z Galil . *Sequencess II* 1993. p. .

[Galil (1979)] *On improving the worst case running time of the Boyer-Moore string matching algorithm" Communication of the*, Z Galil . Sept. 1979. ACM Vo. 22 p. .

[Panicker ()] 'Single Pattern Search Implementations in a Cluster Computing Environment'. PJ , K S Panicker . *4th IEEE International Conference on Digital Ecosystems and Technologies*, 2010.

[Cole ()] 'Tight bounds on the complexity of the Boyer-Moors string matching algorithm'. R Cole . *SODA '91 Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms USA. Philadelphia*, 1991. p. .