Parallelization of KMP String Matching Algorithm on Different SIMD architectures: Multi-Core and GPGPU's

Akhtar Rasool, Nilay Khare
Maulana Azad National Institute of Technology,
Bhopal-462051, India

ABSTRACT
String matching is a classical problem in computer science. After the study of the Naïve string search, Brute Force and the KMP algorithm, several advantages and disadvantages of the algorithms have been analyzed. Considering KMP in particular concept of parallelization has been introduced to improve the performance of the KMP algorithm. The algorithm is designed to work on SIMD parallel architecture where text is divided for parallel processing and special searching at division point is required for consistent and complete searching. This algorithm reduces the number of comparisons and parallelization improves the time efficiency. This algorithm achieves a better result as compared to the multithreaded version of the algorithm where again by text dividing, the parallelization is achieved.

General Terms
Algorithm; String Matching

Keywords
KMP, SIMD, Parallel KMP

1. INTRODUCTION OF STRING MATCHING
The interpretation of string pattern matching is that substring position in the parent string is found and it is an important algorithm for various applications. Few of the well Known algorithms are BM (Boyer Moore), and the KMP algorithm [1, 2, 3]. Researchers had been doing research to improve the algorithm, especially the KMP algorithm and BM algorithm [4,5,6,7], and the improved algorithm achieved a certain level of matching efficiency. The worst case searching time of the two algorithms is linear. Here a more efficient KMP algorithm is introduced using the concept of parallel processing.

2. OVERVIEW OF PARALLEL PROCESSING
Parallel processing is the use of multiple processing units to execute different parts of the same program simultaneously. The main goal of parallel processing is to Reduce Wall Clock Time. Other goals of parallel processing include:

- Cheapest Possible Solution Strategy.
- Local versus Non-Local Resources.
- Memory Constraints.

Why Use Parallel Processing to Reach These Goals?

- Limits to transmission speed.
- Limits to miniaturization.
- Economics limitations.

The most common strategies for increasing speed involve:

- Faster Processors.
- Higher-density Packaging.
- Thinner Substrates.
- Fairly fast processors are inexpensive.

Bottom line: Processors in parallel are relatively less expensive than a single high speed processor. Also number of instruction processed per second cannot increase up to certain limit because it can produce more heat and circuit can burn.
Some of the parallel architectures are SIMD (Single Instruction Single Data), MISD (Multiple Instruction Single Data) and MIMD (Multiple Instruction Multiple Data).

3. KMP (KNUTH MORRIS PRATT) ALGORITHM
Knuth, Morris and Pratt proposed a linear time algorithm for the string matching problem. A matching time of O(n) is achieved by avoiding comparisons with elements of text 'S' that have previously been involved in comparison with some element of the pattern 'p' to be matched. i.e., backtracking on the string 'S' never occurs.

3.1Components of KMP algorithm
The prefix function, Π
The prefix function, Π for a pattern encapsulates knowledge about how the pattern matches against shifts of itself. This information can be used to avoid useless shifts of the pattern 'p'. In other words, this enables avoiding backtracking on the string 'S'.

The KMP Matcher
With string 'S', pattern 'p' and prefix function 'Π' as inputs, the occurrence of 'p' in 'S' is found and the algorithm returns the number of shifts of 'p' after which the occurrence is found.

Running - time analysis
The running time for computing the prefix function is Θ (m) and running time of matching function is Θ (n). [12]

3.2KMP ALGORITHM WITH PROPOSED PARALLELIZATION
The KMP algorithm has been modified, to work on parallel architecture supporting strings of larger size. The concept of parallelization has introduced to improve the performance of the algorithm.
Using the concept of parallelization, a very large size string is divided into parts independent of the pattern size. The same pattern is executed on different parts of string in parallel, thereby reducing the time complexity of the algorithm.
Speaking in terms of memory and processors , a much reliable multiple execution can be achieved in parallel The same concept of KMP matcher can be applied for matching the pattern in the strings which are divided in multiple parts and executed in parallel. Here we are just illustrating a parallelization method with the help of an example. Suppose there are four processors available. So we divide the text into four parts and shared memory keeps the pattern's KMP Prefix Function and four different parts are processed by four different processors. In this parallelization process SIMD (Single Instruction Multiple Data) architecture is applied. Here the KMP algorithm is applied on separate data for

parallel processing. Main Problem in this algorithm is that if pattern comes at the data division part or connection point it is not detected because the data is processed in different processors.

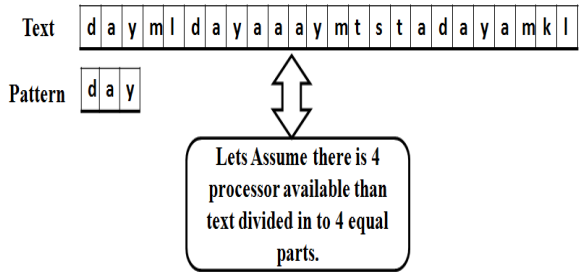

Figure 1: Before division

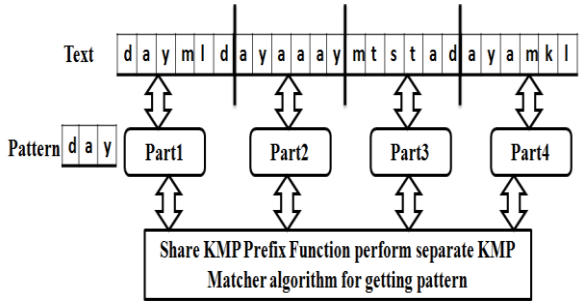

Figure 2: After division

For solving this problem we have process one more data string at each connection points. Suppose pattern size is n than n-1 elements from end of part1 and n-1 elements from start of part2 is taken at each connection or division part shows in figure above and create a new data set which uses KMP algorithm for pattern searching. These connection points' strings can be parallelized for getting better performance.

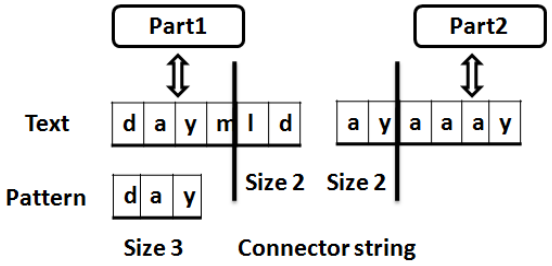

Figure 3: Connection string before joining

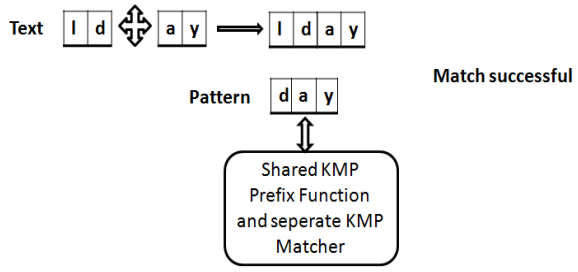

Figure 4: Connection string after joining

Proposed Parallelize SIMD based KMP algorithm:

//KMP algorithm function

KMPSearch (start, end, pattern, position)

Start- Starting position of data in the text array.
End- End position of data in the text array.
Pattern- String to be search.
Position- Successful match positions array

ParallelKMP:

// calculate KMP prefix function for processing.
Given: A text of n elements stored in A[1....n]. A pattern of m elements stored in pattern [1..m].where m<n.
Goal: To find pattern P in the text A.
Global: A [1..n], pattern [1..m],position array
Begin
//Controlling processor activates the processors.
Spawn (p_1, p_2,..p_k);
//Where k is the number of processors and its value //depends upon the available architecture.
For all processors p_i where $1 \le i \le k$ do
{// divide text data and apply KMPSearch on parts. $\mathbf{KMPSearch}((\mathbf{i}-\mathbf{1}) \times \left\lfloor \frac{\mathbf{n}}{\mathbf{k}} \right\rfloor, \mathbf{i} \times \left\lfloor \frac{\mathbf{n}}{\mathbf{k}} \right\rfloor, \mathbf{pattern}, \mathbf{position})$;
} end for
//each processor perform searching
// After getting all processing elements process //complete response again do KMP parallel search for connecting parts.
Spawn (p_1, p_2... p_{k-1});
For all processors p_i where $1 \le i \le k-1$ do
{// KMPSearch on connection or division parts.
$\mathbf{KMPSearch}(\mathbf{i} \times \left\lfloor \frac{\mathbf{n}}{\mathbf{k}} \right\rfloor - (\mathbf{m}-\mathbf{1}), \mathbf{i} \times \left\lfloor \frac{\mathbf{n}}{\mathbf{k}} \right\rfloor + (\mathbf{m}-\mathbf{1}), \mathbf{pattern}, \mathbf{position})$;
}//end for
End// process complete

3.3ALGORITHM ANALYSIS

This method greatly improves the performance of string matching algorithms.The best case time complexity of the string matching algorithms are O(n),where n is the text size in which string to be searched. Suppose the number of processors available for parallelization is equal to p.The text size in which pattern to be search is n and the pattern string size is m. Here assume p is equal to k number of division , means number of processors are equal to number of divisions.This is basically a optimum load. All available processors are fully utilized. Here the text is divided in to various parts so time complexity of the algorithms are $O\left(\left\lfloor \frac{n}{k=p} \right\rfloor + \frac{(p-1)(2n-2)}{(p-1)} + c\right)$ where c is the constant which represents overheard depends upon the architecture for parallelization initialization and combining the results.2n-2 string is process again for detecting connection point string.

4. EXPERIMENTAL RESULTS AND ANALYSIS

Generalized Text Division Method implemented for KMP string matching algorithm on different SIMD architectures and provides massive improvement in pattern matching time efficiency.

4.1Experimental Environment

Processor: i3
RAM: 4 GB
OS: windows 7
Language: visual C++ runs on visual studios 2008
GPGPUs: AMD Radeon HD 6800 series.
Language (parallel Implementation): OpenCL

4.2.Experimental Data for Single Pattern String Matching

Text File: Text of size 251 MB, having large number of occurrences of pattern.
Pattern File: Three different Pattern of length 8, 16 and 25.
Here we are taking 20 threads execution results for multithreaded single CPU and multi-core CPU. On OpenCL we are taking 6000 work-items without setting any local workgroup size. If local workgroup size cannot be set to any value in this case the OpenCL implementation will determine how to be break the global work-items into appropriate work-group instances. So in that case GPU cores utilization are maximum in respect of global memory. These are the best case of un-optimized GPU implemented algorithms.

4.3.Experiment

KMP algorithm is implemented in three different ways:

- Serial
- Multithreaded CPU
- Multicore CPU using OpenCL
- Parallel on GPGPUs using language OpenCL

Experimental results of these are taken and analysis of KMP algorithm on above four implementation is shown below.
The experimental result is shown in table 1 and comparison is shown in graph below. In multithreaded implementation a speedup of 2.05, in multicore implementation a speedup of 4.33 and in GPU implementation speedup of 9.52 is achieved in comparison to serial implementation.

Table1 : KMP Algorithm Experimental Results

KMP Algorithm		Implementations			
		Serial	Multithreaded single CPU	Multicore OpenCL	GPU
Pattern Length	8	860	400	203	87
	16	830	425	200	93
	25	900	435	195	92

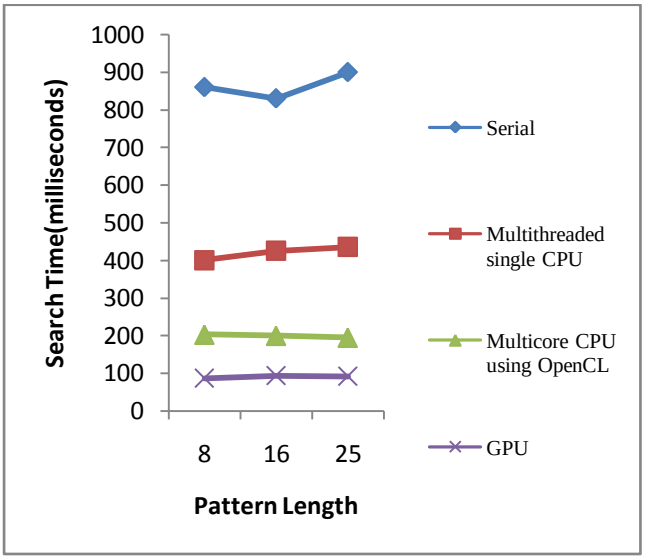

Figure 5: Comparison of KMP algorithm of different Implementation for different pattern length

5. CONCLUSION

The KMP algorithm with Parallelization greatly improves the matching efficiency if the text size is very large and a sufficient numbers of processors are available. The most important characteristic of the KMP algorithm is that by making better use of next bit characters, a maximum moving distance is achieved .Further parallelization of KMP algorithm provides parallel computing of pattern searching on the text in SIMD architecture. The use of pattern matching is very broad and efficient parallel pattern matching algorithms can improve system performance in various problems of computer science.

6.REFERENCES

[1]. Published IEEE papers related to confined topics of KMP and other string matching algorithms, references as , Knuth DE,Morris JH,Pratt V R.Fast pattern in strings[J]. SIAM Journal on Computing,1977,6(2):323-350.
[2]. Tang Va-ling. KMP algorithm in the calculation of next array. Computer Technology and Development [J] .2009,19 (6) :98-101.
[3]. Published IEEE papers related to confined topics of KMP and other string matching algorithms, references as, Knuth DE,Morris JH,Pratt V R.Fast pattern in strings[J]. SIAM Journal on Computing, 1977,6(2):323-350.
[4]. WANG Jian-guo, ZHENG Jia-Heng. "BM string matching algorithm for an improved algorithm". Computer Engineering and Science, 2007, 29 (5): 94-95.
[5]. ZHANG Hong-mei, Fan Ming-yu."Pattern matching BM Algorithm" [J]. Computer Applications .2009,26 (9): 3249- 3252.
[6]. YUAN Jing-bo, ZHENG Ji-sen, DING Shun-Ii. "A kind of BM pattern matching algorithm" [J] Computer Engineering and Applications, 2009,45 (17) :105-107.
[7]. Wu Xi-hong, Ling Jie. "BM pattern matching algorithm" in Computer Engineering and Design, 2007,28 (I): 29-30.