A Parallel Algorithm for the Best k-mismatches Alignment Problem

Harleen Kaur Ahuja
San Jose State University
Email: harleen0489@gmail.com

Veena Reddy
San Jose State University
Email: veena.reguri@gmail.com

Abstract- With the development of parallel technology and wide range of applications in this field, the performance analysis has become one of the most challenging aspect of Parallel Paradigm. This paper is about a research work on a well-known parallel algorithm – Best k-mismatches Alignment Problem. The objective of this paper is to attain and implement the practicality of this parallel algorithm to achieve faster computational speed. The motive for this algorithm is to improve the processing by decreasing the workload or in other terms reducing the search space.

The real time implementation for this algorithm exists in the field of bioinformatics, where this approach is used to find the best match between a genome cells. The actual algorithm proposed is in compatible to PLX, FERMI supercomputers and ERNE(Extended Randomized Numerical aligner) package. In this paper, to implement the solution the concept of multithreading in Java has been used.

I. Introduction

With the improvement in Bioinformatics field, the expansion of new technologies leads to the invention of new Generation Sequencers. These sequencers can generate huge amount of data in very less time. These sequencers are in fact used to generate a number of string patterns like {A C G T N} which are nucleotide bases. In biological terms, these alphabets imply Adenine, Cytosine, Guanine and Thymine. Once this step of generating sequences of genome is done, the next phase is the alignment phase. There are two strings - a reference and various set of strings (genome) generated by Next Generation sequencers (NGS). We need to align these two set of strings searching the correct position for each pattern against the reference with a limited number of mismatches allowed. Now, at this part the k-mismatch algorithm comes into picture.
The throughput of the sequencers is increasing with a factor of 5 every year. This forces us to implement such features that can results in the reduction of computation time taken by the alignment phase. Since it is essential that higher is the amount of data generated, higher is the need to process the data as fast as possible.

II. Problem and its Solutions

The problem is we have a long string containing millions of characters that needs to be broken into short length strings. The number of mismatches allowed is given as k. These substrings are then distributed among various nodes and the alignment check is done with the reference string in parallel on all nodes. Each node will start finding a string matching with minimum number of mismatches. When one node finds a better solution than the other it will broadcast the value of k (which is the number of mismatches found) to all other nodes thus reducing the search space as well as the computation time.

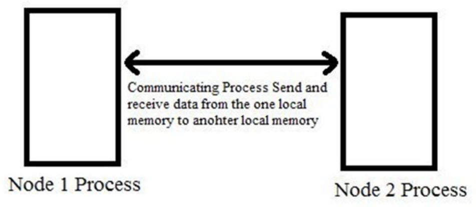
The problem can be defined mathematically as below:
Consider two strings P (Pattern) and R (Reference) with the characters in the space {A,C,T,G,N}. Since we have some number of mismatches allowed, we can say we are actually implementing an approximate string matching algorithm instead of an exact matching. The best algorithm for calculating the approximation in the field of bioinformatics is the Hamming Distance. The Hamming Distance counts the number of differences between the two strings. Now, in our problem the alignment means to find the best k value. The best will be the minimum value.

Though the naïve solution will be to compare each character of pattern with the reference string but this will results in the computation time of m*n (m and n as the length of pattern and reference strings). But, unfortunately this cannot act as a practical solution due to the large dimensions of the reference strings which vary from few thousands to billions of base pairs. To improve this computation time this algorithm uses a technique that reduces the search space that is by dividing the string parts into number of nodes and sharing the value of k among those nodes.

Let's take an example to describe the situation more clearly. There is a node A find a string match with number of mismatches less than given value of k. It will broadcast a message to all other nodes " I have found a match with the x number of mismatches where $x < k$ so now the new value of k is x". This technique results in reduction in processing time required to perform the alignment between the strings. To implement this technique, communication between nodes is required.

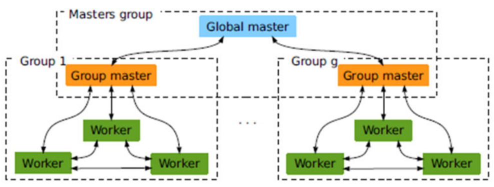
There are two parallel alignment approaches which can be used to implement this model explained above. The first one is MPI Message Passing Interface. The message is a special method of communication to transfer signal from one domain to another. The domain here can vary depending upon the scenario one is working. It can be threads or processes. Two threads that are communicating and exist in the same processor are using intra-processor messaging. Two threads that are communicating but exists in two different processors are using inter-process messaging. In more general, rather than using the term thread, we use process-process message passing.

To perform this message passing between processors, there must be an interface or a medium. This medium is called as Message Passing Interface. It is a protocol or a library that allows developers to perform message passing between different nodes.


Communicating Process Send and receive data from the one local memory to anohter local memory
Node 1 Process
Node 2 Process
SEND & RECEIVE are the two important Fundamentls of the MPI

The second approach that can be used is a framework produced by Google known as MapReduce. This technique can make use of few to thousands of machines as per the requirement. In this approach there are two procedures (a) Map – The input is split into number of parts and is distributed among different nodes. The string alignment algorithm is executed simultaneously on all the nodes. (b) Reduce – This procedure is to combine the results from all the nodes and provide the final result for the query ran on that cluster.

This is a Master/workers model where master divide the input and provide instructions to its workers regarding which task needs to be executed and when. The workers communicate among them. As per the research, the biological references do not allow the split the references into more than 10-30 parts. So in order to utilize all the nodes available the substrings are replicated on the nodes left. This logical structure can be depicted in the figure below:

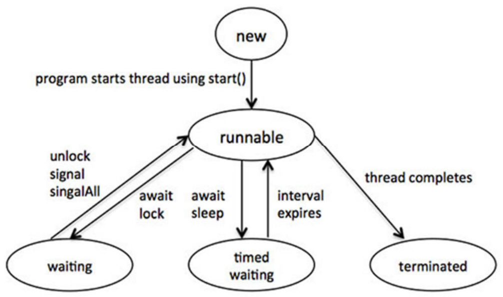


The above structure is divided into two parts- Master and the workers. The responsibility of Master is data distribution and result collection. Whereas, on the other hand the workers which is a set of nodes perform alignment tasks. The most challenging part of this implementation is the update of best k values. This is because the time at which the k value is updated is unpredictable and we are not sure which node source updated the value of node. In order to avoid network traffic, we can use an efficient buffer mechanism. During the communication, a common thread deals with updating of k value. So whenever the threads are executing they can always check the value of common buffered value of k without wasting time in communicating the k value.

III. Implementation

The real time implementation though requires the use of supercomputers as well as the sequencers to generate the long pattern reads and the references. Therefore, in this paper to implement this k-mismatch algorithm applying the parallel technique, we are utilizing the concept of multithreading in Java. Java being a multithreaded language can make two or more threads run concurrently. Actually the way CPU handles it is by switching rapidly between the threads thus giving an illusion of threads running at the same time. Using this each thread can handle different task at the same time. This results in optimal use of available resources.

Each thread goes through a life cycle. It can be waiting, running or in completed state. This can be shown as:



In order to create threads, two methods can be used. The first is by extending the thread class. The second is by implementing the runnable interface. In our algorithm, we have created four different threads. The input string is taken from the user. The string is then divided in to four substrings and distributed among the four threads. Now each thread while running is actually working on different part of string. In case any one thread finds an appropriate match with number of mismatches less than k. it will update the value of the globally declared value k. Now from this time, all the threads will be looking for a match with the number of mismatches allowed equals the recently updated value.

This is how we are reducing the search space for the nodes (threads in our case) and improving the time taken to find an approximate match. The main concern or challenge in this algorithm was how to share the updated value of k and how to communicate among other threads executing. Though this approximate matching makes more sense if we talk about DNA/RNA matching with a reference genome, But this concept of multithreading has perfectly illustrated the algorithm.

IV. Conclusion

In this paper, we have explored the k mismatch algorithm and the alignment problem. We stated the problem faced from biological perspective and understand the two important approaches that can be used as a solution to this problem. The two aspects were MPI – Message Passing Interface and the Map Reduce paradigm. Building the base to the problem we tried to work out on the implementation of this algorithm where a task is divided into number of nodes.

To implement this concept we have make use of Multi-threading concept in Java. In order to improve the computation time, the results of different nodes were made to share among them thus resulting in reduction of search space or in other terms reducing the workload on each node. Finally, we tested our algorithm using real time data and also obtained our objective that was to share the k value among nodes and improving the processing in the alignment phase.

In future enhancements, we are planning to take this implementation to a higher level by using Map Reduce approach and passing information among different clusters instead of simple nodes.

V. References

[1] Cristian Del Fabro, Fabio Tardivo, Alberto Policriti, “A Parallel Algorithm for the Best k-mismatches Alignment Problem”, IEEE, 2014.

[2] Yaser Jararweh and Quassi Yaseen, “Multi-Threading Based Map Reduce Tasks Scheduling”, IEEE, 2014.

[3]John, “Starting Threads and Using Anonymous Classes”, http://bit.ly/1sPBYn9, 2011

[4] tutorialspoint, “Java Multi-Threading” , http://bit.ly/1GZP7nq

[5] JB Park, FK Yu, “A Message Passing interface for parallel and distributed computing”, 1983.

[6] Longjiang Guo, “Parallel Algorithm for Approximate Sting Matching with K differences”, IEEE, 2013.

[7] Koloud Al-Khamaiseh, “A survey of Sting Matching Algorithms”, 2014