International Journal of Cognitive Informatics and Natural Intelligence
Volume 15 • Issue 4 • October-December 2021

Hybrid Approach for Enhancing Performance of Genomic Data for Stream Matching

Gururaj T., Department of Information Science and Engineering, Ramaiah Institute of Technology, Visvesvaraya Technological University, Belagavi, India

Siddesh G. M., Department of Information Science and Engineering, Ramaiah Institute of Technology, Visvesvaraya Technological University, Belagavi, India

ABSTRACT

In gene expression analysis, the expression levels of thousands of genes are analyzed, such as separate stages of treatments or diseases. Identifying a particular gene sequence pattern is a challenging task with respect to performance issues. The proposed solution addresses the performance issues in genomic stream matching by involving assembly and sequencing. Counting the k-mer based on k-input value and while performing DNA sequencing tasks, the researches need to concentrate on sequence matching. The proposed solution addresses performance issue metrics such as processing time for k-mer counting, number of operations for matching similarity, memory utilization while performing similarity search, and processing time for stream matching. By suggesting an improved algorithm, revised Rabin Karp (RRK), for basic operation and also to achieve more efficiency, the proposed solution suggests a novel framework based on Hadoop MapReduce blended with Pig and Apache Tez. The measure of memory utilization and processing time proposed model proves its efficiency when compared to existing approaches.

KEYWORDS

Apache Tez, DNA Sequencing, K-MER, Levenshtein, Pig, Rabin Karp

INTRODUCTION

For many bioinformatics applications using data from Next-Generation Sequencing (NGS), the frequency distribution of k-mers is quite useful (Behera et al., 2018). A few of these examples involve assembly based on de Bruijn, read error correction, an approximation of genome size, and digital normalization. Counting (or estimating) k-mers with low frequency is a pre-processing phase while designing tools for such applications. The amount of k-mers in genome data, and even more specifically the frequency distribution of k-mers, is a central component of many bioinformatics applications in genome data (Carvalho et al., 2016; Rangavittal et al., 2017).

The analysis consists of Input, Output, Counting k-mer, Sequencing, Similarity search, and Assembly. Out of these tasks, k-mer and similarity search in a sequence data are major modules to address. An alignment (Mapping/Searching) task involves arranging sequences to get the highest similarity level. Alignment helps in generating a phylogenetic tree. Two types of alignments: Local (the only portion of the sequence is aligned) & Global alignment. Basically, living things are

DOI: 10.4018/IJCINI.20211001.oa38

related by evolution. Thus DNA, RNA and protein sequence (Amino acid) of different organisms are related to one another in evolution and show similarities. Reconstruction of DNA sequence by combining and aligning small fragments refers to the Assembly process which is a part of novel DNA Sequencing which includes (a) Cutting the DNA into small pieces (b) Reading the small fragments (c) Reconstituting the original DNA by merging the information on the various fragment.

While computing genomic data, k-mers (Miller et al., 2008) are used in sequence assembly and sequence alignment. K-mer generates all possible substrings of length 'len' from an input DNA Sequence. The total number of k-mers generated from a length 'len' is len-K+1, When performing sequence assembly task, choice of size(K) is very important. Therefore, if experiments have lower sized k-mer, they will decrease the edges in the graph, in turn, less space is required to store sequence and if experiments have larger size k-mer, they will result in greater edges and memory to store sequence. Next Generation Sequencing (NGS) technologies (Manekar et al., 2018) generate billions of operations for every run. There is always a scope for designing a framework that is efficient in terms of memory and time. Many of the researchers work on disk based, and they use local resources that are limited by nature. The compressed input will have better results in terms of processing time. So, the authors main concentration is related to k-mer generation and counting.

As the technology improves, there is a substantial increase in the quantity of data, which has intensified the need to reform a new effective technique to speed up the search for compatible DNA sequences in a large data collection(Li et al., 2019; Hiraishi, 2019). One of the main problems of matching approaches is the variability in the length of sequences in a given sample, which will affect the results. There will always be the most common subsequence since the longest series between the others(Alazzam et al., 2018). Two key variables are used during the matching process to determine the string matching algorithm efficiency, which is the total number of character similarities and the total number of tries(Sameer et al., 2017).

As computing becomes easy and the source of genomes expand, the dominion of bioinformatics is sure to increase and change radically, allowing us to build new models of complexity and usefulness. When sequence analysis reveals the cause for a disease, the trace of the number of occurrences of the sequence defines the possibility of the disease. As the genome is a huge database, the authors propose a Stream/String and Pattern matching technique to find out a particular sequence in the given large input sequence. Bloom filters use a proper data structure for classification while performing sequencing, (NajamL Jin et al., 2018) proposed Multiple Bloom Filters which locate the specific pattern in a DNA also tell the number of repetitions. While predicting any sort of disease these two factors are very important. This proposal focuses on a new approach for detecting the patterns present in the gene database. Stream matching is to find out the exact location of a specified pattern.

Table 1. illustrates the notations the authors use in representing DNA, RNA, and k-mers, while working on genomic data (El-Metwally et al., 2014). Each row represents the possible characters chosen from a set of alphabets for DNA, RNA, and k-mers respectively with example in the last row.

Researches are finding novel approaches to work on the human genome that results in the advancement of big data analytics. Certain tasks the researches can perform on the genomic data includes the following: Sequencing, Alignment, Sequence alignment, Next generation sequencing, Sequence analysis, sequence assembly, Gene expression, and et al. Let us consider the process of Sequence Analysis, where the researchers use different analytical methods on DNA/RNA/Peptide sequence to extract feature, function, structure or evolution.

Proposed approach lists various existing string matching techniques like Brute force, fuzzy string searching, Robin karp, dynamic programming, Needleman wunsch, Smith waterman, etc., Table 2. compares most frequently used 'Brute Force and Robin Karp' (researchers are interested in improving) which is represented by two columns. Comparison is based on certain parameters highlighted with different rows like additional efforts for pre-processing O(m) is needed in case of Robin karp while no pre-processing is required in Brute Force technique. In both cases, the type of search left to the right remains the same. Running time of Robin Karp is O(n+m) which is less compared to Brute

Table 1. Mathematical notations for DNA, RNA, and k-mers

A string is a predetermined sequence of nucleotides, naturally characters.	
Characters used to represent a DNA sequences	Σ = { A, G, C, T }
Characters used to represent a RNA sequences	Σ = { A, G, C, U }
k-mers - Recognize regions of attention within sequences	Σ2 = { CC, CT, CA, CG, TC, TT, TA, TG, AC, AT, AA, AG, GC, GT, GA, GG} Σ* = Σ 0 È Σ 1 È Σ 2 È …
Σ are ordered lexicographically based on the relationship	ε < C < CC < CCC < … < CCCA < CCCT < CCCG < CCA …
Example illustrates, the string CATG has	4-Prefixes (C, CA, CAT, and CATG) 4-Suffixes (G, TG,ATG, and CATG).

Force value O(nm). Rabin karp approach uses hash based technique which is far better than that of linear search, used by Brute Force. And finally, search idea is given in the last row of table. Table 2. Clearly indicates that Robin karp is more efficient. Further details of Robin karp will be discussed in the next section.

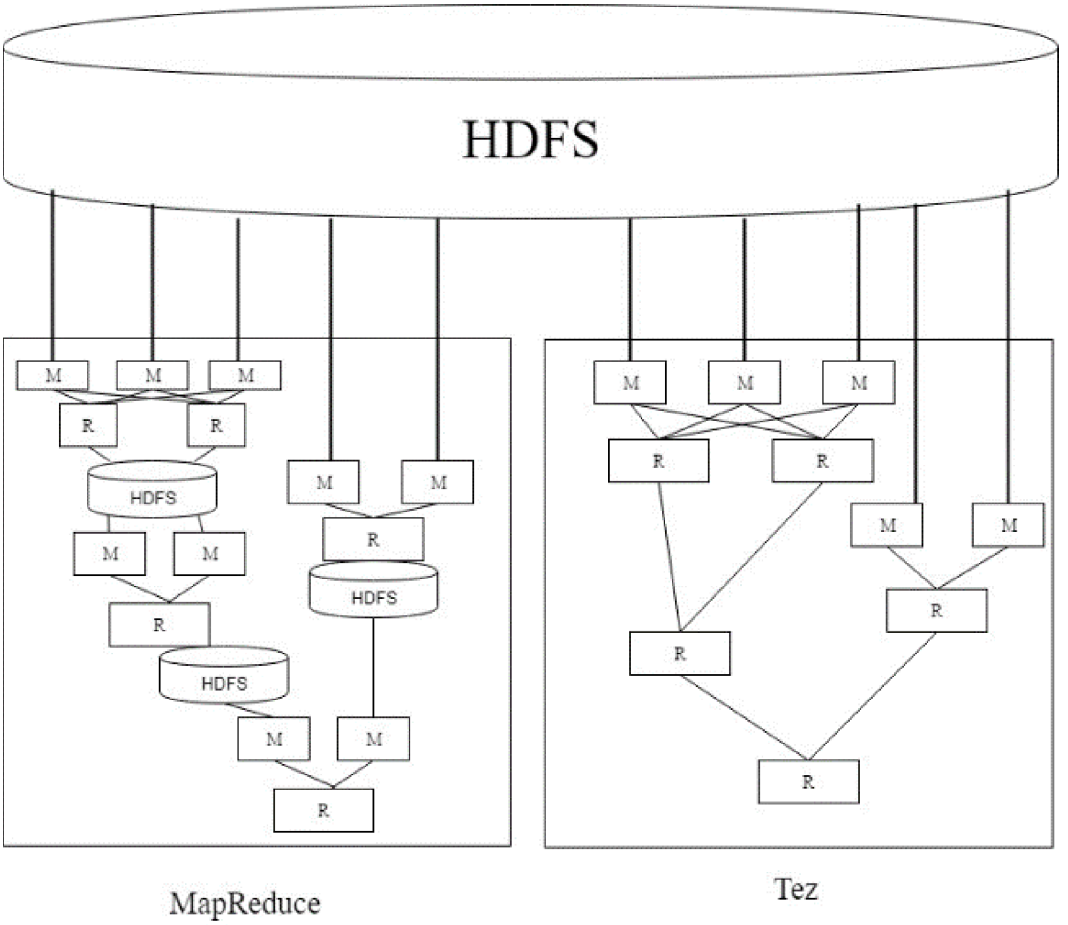
Hadoop MapReduce (Eadline, 2016; Jin et al., 2018) is a two-step process that includes a mapping followed by a reducing. Figure 1. refers to a comparison with Map reduce with Tez; here Tez came up with better results and performance comparing with the old ways. Many researchers in the field of bioinformatics have contributed to changing the lives of human beings. The medical field largely depends on the processing and analyzing of large genome data sets (Tian & Liu, 2019), hence Hadoop is one such platform that gives a solution to many bioinformatics problems because of its ability to execute fast and also accurately and reliably. The proposed approach incorporates enhanced Hadoop (Navarro et al., 2019) with a different environment using PIG and Apache Tez and managing work with the help of pig scripts and execute MapReduce jobs.

Hadoop blended with pig enable easy parallelism and support large datasets. In the proposed approach authors have considered pig which is based on Hadoop tool kit for tuning parameters. Tuning, in respect of the size of the block, size of the heap and compressing of data it results in improving performance with respect to time and memory. Pig scripting tool analyzes data both locally and on a Hadoop node and to execute a job on the cluster author used optimized Tez. The proposed approach is more efficient because of Tez (which uses containers efficiently), multiple reduce phases (without map phases) and effective use of HDFS. The software environment used in this framework can be listed as OS: Linux, Hortonworks HDP 2.2 with Hadoop version: 2.6 & Pig version: 0.14.0.

Table 2. Comparison of two string matching technique

Parameters	Algorithms	
	Robin karp	**Brute Force**
Pre-processing	O(m)	No
Search type	L -> R	L -> R
Running time (Worst case)	O(n+m)	O(nm)
Approach	Hashing based	Linear search
Search Idea	Compares hash values of text and pattern	Search by matching all character

Figure 1. Comparing regular Map Reduce Paradigm with Tez



RELATED WORK

(Tripathi et al.,2016) highlighted basic steps for Genome data analysis and various tasks included in the process. Primary tasks can be listed as Sequence file management, Quality check, Alignment, Mapping, and Analysis. For each of the tasks various tools are developed, the majority of the tasks can be accomplished by SEAL tool Various tasks, related software and tools have been listed in Table 3. This will give an overview of primary tasks and available software. Every algorithm is suited for its own purpose as explained in Table 3., The authors have considered the major issues related to sequencing, alignment, and assembly. The authors have come up with the best possible design as explained in the next section.

The solution to k-mer counting problems can be given in different ways. The basic way is by generating hash value & storing (key, value) pair. The process, where k-mers as keys, and counts as value, are considered it becomes slow (Purcell et al., 2005). Some of the advanced k-mer counters (Kurtz et al., 2008) uses a suffix array, but to process big sequence, data parallelism needs to be used. In order to increase the efficiency of stream matching technique author uses Graphics Processing Units (GPU) which is robust and parallel processing in nature with Rabin karp algorithm (Shah et al., 2017) this results in higher efficiency which speeds up the process by tuning number of threads. In the current scenario, the Hadoop MapReduce paradigm will be used to improve the speed of

Table 3. Tools related to Sequencing, Alignment and Assembly

Function	Algorithm/ Tool	Description Reference
Genomic sequence Mapping	**CloudAligner**	NGS (Next Generation Sequencing) using MapReduce based application for mapping short reads.
	CloudBurst	NGS using a parallel read mapping technique for comparing One genome and another genome.
	SEAL	It's a distributed application for manipulating, aligning and analyzing short DNA sequences.
	BlastReduce	Optimized parallel short sequence read mapping for aligning sequence in SNP, personal genomics and genotyping.

computation. Authors have proposed an algorithm that uses Hadoop with Apache Tez which helps in reducing time thereby increasing the performance (Pahadia et al.,2015).

K-mer counting will be used for different steps involved in bioinformatics like Sequence analysis, assembly, and alignment. Incorrect reading in huge input can be eliminated by using Brujin graphs while performing sequence assembling. This can be used to align multiple sequences, as the size of k-mer increases most of the computation time will be for a read operation (Shi et al., 2015). Hence k-mer counting is a complex problem in genomic data analysis. This problem raises issues of memory. To address this scalability issue our proposed solution uses the Pig environment with HDFS. This suits a big dataset and k-mer size to boost up the speed and to decrease the number of reading operations.

A similarity search in a sequence can be addressed in different ways. In some techniques sequence alignment (Huang et al.,2011) is not required, while other techniques require alignment. This process of alignment includes adding or removing sequences which result in better alignment (Rahman et al.,2006). One of the popular algorithms is Maximum Common SubStream (MCS) technique used to find similarity between sequences. In (M et al.,2013) different variations of MCS are proposed. Rabin karp (RK) technique uses Hashing. Which will be adopted for large data. In RK technique input substring is compared for M letters iteratively, and if it's equivalent then there is a match else it will calculate next M character sequence for comparison. For sequence similarity, the (Tripathi et al.,2016) paper had proposed the comparison between MCS and Rabin karp method. These two techniques were implemented and the author came up with a conclusion that Rabin karp is more efficient compared to Maximum Common SubStream, because Maximum Common SubStream takes more time than RK algorithm. Further, it results in the best performance and accuracy in the case of RK. By considering the results of (Tripathi et al.,2016), the authors proposed an improved version of Rabin karp (Revised RK) and proved to be better than the previous versions of Rabin karp.

Rabin karp works on the principle of comparing hash value of two strings, provided the hash values are the same. It is successful in terms of hash value, but sometimes it may be spurious only after searching the actual stream of data. It can be finalized as a successful search is itself a drawback where two strings may have the same hash value but not similar in the stream. To overcome authors have proposed a new hybrid technique which is a combination of Levenshtein algorithm and modifications while calculating the remainder along with quotient for improving comparison with actual data and searching data.

PROPOSED SOLUTION

Overview of Genomic analysis with respect to proposed work is shown in Figure 2. Based on the requirements of quality of input, reads are pre-processed with different techniques like reading conversion, Quality check and Trimming (The data in experiments used are pre-processed). The next process will be of sequence assembly that requires different tasks. In this paper, authors have

Figure 2. Overview of Genomic data analysis

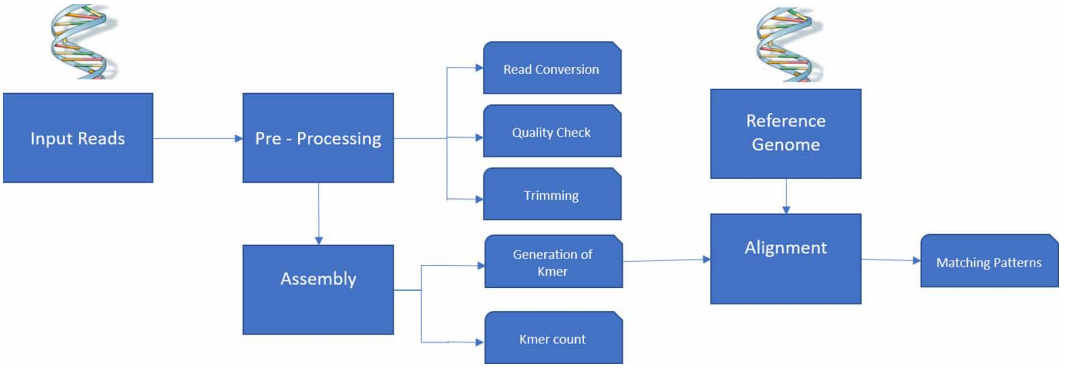


Figure 3. Proposed algorithm for Analysis of Genomic Data

Input: DNA Sequence from NCBI.
Output: Results in terms of increased efficiency.

Overall flow of genomic data analysis in the proposed framework:

Step 1: Input reads from NCBI will be stored using PIG.
Step 2: Identified two tasks in assembly phase i.e., Generation of k-mers and k-mer counting.
Step 3: Output of step 2 will be given as input for Matching patterns (one of the Alignment task).
Step 4: Displaying Results.

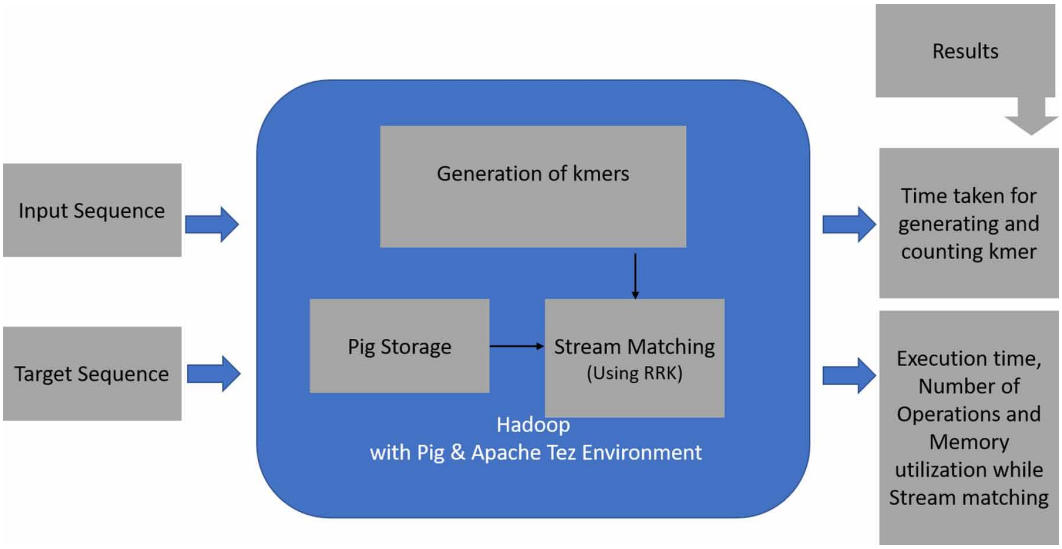
Stage One: Generation and counting k-mers

Step 1: Read data set
Step 2: Fix k size
Step 3: Generation of k-mer using Algorithm and given as input to stage two.
Step 4: Output recorded based on Time taken for generating and Counting k-mers

Stage Two: Stream Matching process

Step 1: Input data from Target Sequence and stage one output.
Step 2: Apply Stream matching technique using Revised RK Algorithm.
Step 3: Output recorded based on Time taken for Execution, Number of operations & memory utilization for Stream matching.

Figure 4. Block diagram of proposed system



concentrated on the generation of k-mers & counting k-mer. As a second stage Matching patterns/ Similarity search will be done with the reference genome by alignment process. Finally, the visual representation of the data analysis techniques is investigated with genomic datasets.

Figure 3. discusses the proposed algorithm for the analysis of genomic data, which consists of two stages. The first stage is related to the collection of data set from NCBI sources and is stored in Pig storage. By fixing the value of k and using k-mer generation algorithm, output values are recorded. The output of the first stage (generated k-mer) will be given as input to stage2 which will be matched with the target sequence by using the Revised RK algorithm and the final results will be collected. Figure 4. Illustrates the two stages of the proposed system. Here input sequence is given for stage one to count and generate k-mers. Then target sequence is matched with generated k-mers using Revised Rabin karp in stage two. The final results are obtained.

Counting K-Mers

To generate and count the k-mer which is stage one process, here authors have proposed an algorithm that includes Hadoop with pig and Apache Tez. Researchers have compared the performance issues with the enhanced version with the basic Hadoop version. Datasets were collected with three different kinds, the detailed description will be given in the result section. Figure 5. gives the methodology used for k-mer generation and counting steps include read data set and store in HDFS. Fixing k value specifying size of k-mer and then extracting using an algorithm is mentioned in the next section. While running this, authors have used two different modes. As a start to get the results of basic Hadoop execution, researchers have used I/O operations of HDFS and basic MapReduce operations. The results were stored in a temporary file. Hence from all kinds of datasets, extracted k-mer will be stored in a temporary file. The extracted k-mers with higher k size will be given as input for the next stage, where authors perform stream matching technique with the target sequences. Based on the requirement authors have created temporary files against all input datasets and stored.

The below Figure 6. explains the process of extracting k-mer by splitting input read sequences (4mer in the example). As a first step, the initial input of DNA is sequenced. Then the alignment and sequencing of the input reads are made (output reads). In the third step, reads are being split into 4-mers. Finally, repeated 4-mers would be discarded in the fourth step and alignment done.

Figure 5. Steps in KMER task

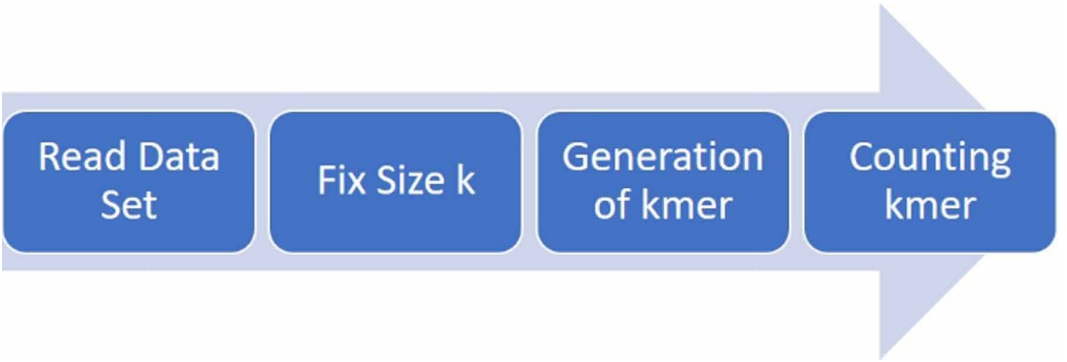
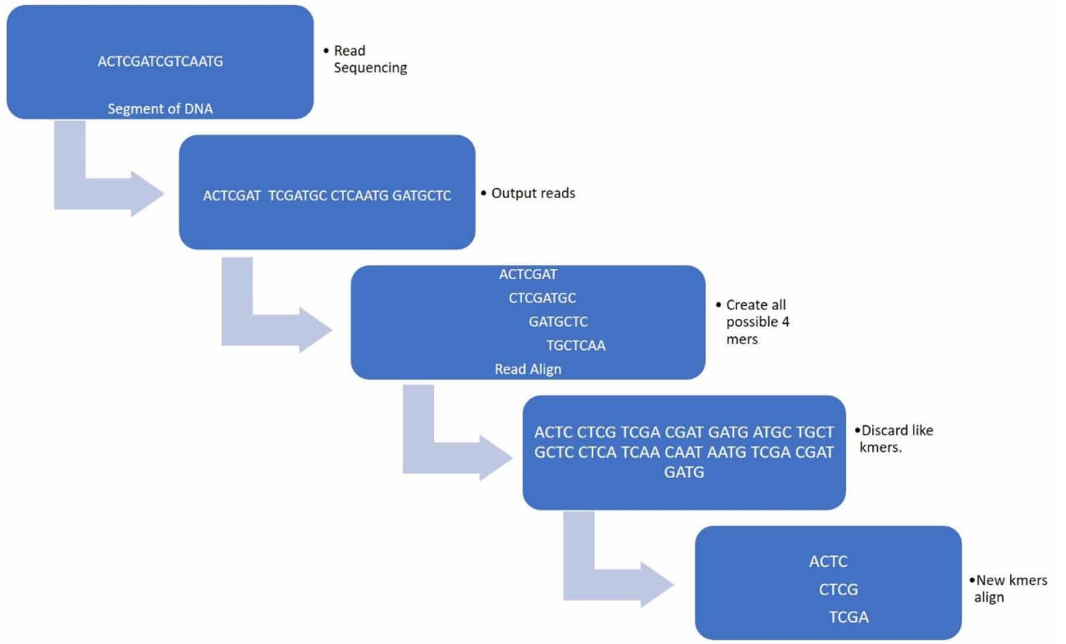


Figure 6. Process of KMER generation

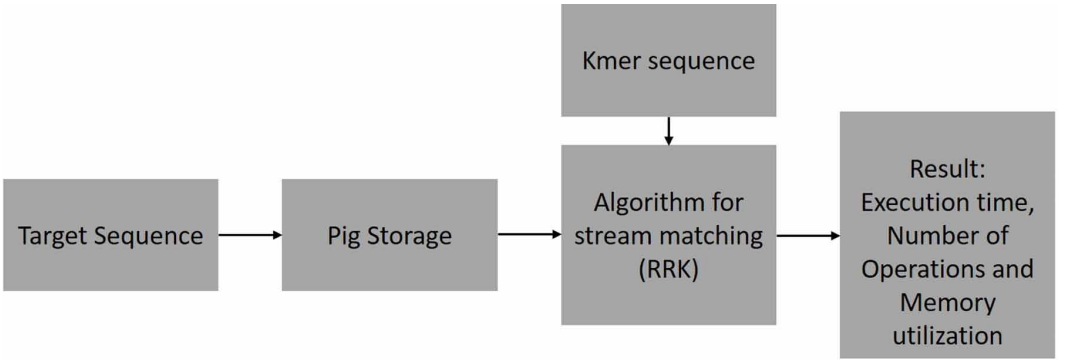


Many functional modules like I/O, k-mer counting, Assembly, Blast & Similarity need more memory capacity to process in standard computers. Among these tasks k-mer counting is important and is used in constructing Bruijn graphs in the assembly process. As the size of k-mer increases, a lot of reading operations need to be done. To address this complication. Pig provides a suitable environment using Hadoop framework which is scalable in nature.

Similarity Search

Nowadays most popular, scalable and capable of processing with large datasets, is possible only with Hadoop based technologies. Authors have used this approach for the task of genomic analysis with the enhanced environment by adding pig programmability and Speeding up the task by Tez. Figure

Figure 7. Steps in stream matching task



7. specifies the stream matching technique which is a stage two process, where the target sequence is compared with k-mer sequence by using novel RK technique (RRK). Pig programmability helps in reducing time for comparing with database sequences, as Pig is having a special feature like compressed storage capability that helps to reduce the number of operations and memory utilization and is directly proportional to the increased efficiency.

Proposed Revised Rabin Karp (RRK)

To search a pattern 'pat' from a text 'tex', Rabinkarp divides the pattern with predefined prime number 'p' to calculate the mod of pattern 'pat'. Then the process starts with finding m=|pat| from text 'tex' to compute the remainder of m characters from the text 'tex'. If the remainder of the 'pat' and remainder of 'tex' are equal then only compare the characters of text with the 'pat' pattern, otherwise there is no need to compare. This process is repeated for the next set of characters from a text with all possible shifts from 0 to nm, where n refers to the length of text and m denotes the length of the pattern. Thus, REM (pat/p) = REM (tex/p) after division, it will have three options: i. If two REM matches and also the characters of pat and tex are the same, then it will be successful. ii. Two REM are equal but characters of pat and tex are not equal then it becomes the spurious result. iii. If two REM are not equal, then there is no need to compare pat and tex. For example, tex = 3145032147888878821454, Pat = 231450 and p = 11 then REM (tex) = 314503 / 11 =2 and REM (pat) = 231450 / 11 = 3 here in this case remainders are not equal hence no need to compare. Move on to other sets of the same length and repeat the process. For the sake of improvisation, instead of calculating and comparing the only remainder, this proposal further considers the quotient i.e., QUO (pat/p) and QUO (tex/p). Then comparing both remainder and quotient it will be a successful search if it matches otherwise unsuccessful. So that the third choice of spurious result can be avoided. No extra computation time required and efficiency directly increases.

Our proposed Revised Rabin Karp (RRK), adopts Levenshtein algorithm which calculates hash distance in both pattern and text resulting in improving the accuracy. The algorithm takes pattern and text as input parameters and uses the hash value, which is generated by Rabin karp. Then sequence similarity will be checked based on the value obtained by Levenshtein technique. The value generated by Levenshtein gives the similarity level of different input reads. And also, it is used to calculate the minimum effort needed to transform one input data to another string by doing minimum modifications. Hence, the combination of improved Rabin karp with additional checking of quotient and adopting a Levenshtein algorithm, results in higher efficiency and accuracy.

Figure 8. Example of downloaded data for Influenza A virus

```
LOCUS       HC668016                6130 bp    DNA     linear
PAT 03-MAY-2010
DEFINITION  Sequence 1 from Patent WO2010036948.
ACCESSION   HC668016
VERSION     HC668016.1
KEYWORDS    .
SOURCE      Influenza A virus
  ORGANISM  Influenza A virus
            Viruses; ssRNA viruses; ssRNA negative-strand
viruses;
            Orthomyxoviridae; Influenzavirus A.
REFERENCE   1
  AUTHORS   Nabel,G.J., Wei,C.J. and Yang,Z.Y.
  TITLE     Dna prime/inactivated vaccine boost immunization to
influenza virus
  JOURNAL   Patent: WO 2010036948-A2 1 01-APR-2010;
            Department Of Health And Human Services (US)
FEATURES             Location/Qualifiers
     source          1..6130
                     /organism="Influenza A virus"
                     /mol_type="unassigned DNA"
                     /db_xref="taxon:11320"
ORIGIN
        1 tcgcgcgttt cggtgatgac ggtgaaaacc tctgacacat gcagctcccg
gagacggtca
       61 cagcttgtct gtaagcggat gccgggagca gacaagcccg tcagggcgcg
tcagcgggtg
      121 ttggcgggtg tcggggctgg cttaactatg cggcatcaga gcagattgta
ctgagagtgc
      181 accatatgcg gtgtgaaata ccgcacagat gcgtaaggag aaaataccgc
atcagattgg
      241 ctattggcca ttgcatacgt tgtatccata tcataatatg tacatttata
ttggctcatg
      301 tccaacatta ccgccatgtt gacattgatt attgactagt tattaatagt
aatcaattac
      361 gggaacttcc atagcccata tatggagttc cgcgttacat aacttacggg
aatttccaaa
      421 cctggctgac cgcccaacga cccccgccca ttgacgtcaa taatgacgta
tgttcccata
```

IMPLEMENTATION

Data Collection

Data sets from NCBI are used to measure the performance of the proposed technique (NCBI, 2019). Figure 8. Illustrates the downloaded dataset. Data sets include: Influenza A virus (ACCESSION NO. HC668016, VERSION HC668016.1 consists of nearly 6130 base pairs), Mycoplasma Bacteria (ACCESSION NO. LS991952.1 bacteria consists of 981408 base pairs), Salmonella enterica bacteria(ACCESSION NO.CP030026 VERSION CP030026.1 consists of nearly 6125373 base pairs) and Macaca fascicularis animal(ACCESSION NO.CM001276 AEHL01000000 VERSION CM001276.1, consists of nearly 232296185 base pairs). Data sets are classified based on the size of

Figure 9. Algorithm for generation and counting K-Mers

```
Function kmer(InputString, k: size of each k-mer)
        n = len(InputString)  //Repeat over length of string that can made k-mers of n length
        for(i=1;i<=n-k+1;i++) //output individual k-mer of len k, from i to i+k in given string
        Output InputString[i:i+k]
        End for
End function
Example:
Read  : AAAGTCGAGC
3mers : AAA AAG AGT GTC TCG CGA GAG AGC
5mers : AAAGT AAGTCG AGTCG GTCGA TCGAG CGAGC
```

organisms a few KBs to 100's of MBs. Dataset1 refers to organisms ranging from 1-100 KB, dataset2 refers to organisms ranging from 10-100 MB and dataset3 is a collection of 100- 1000 MB (1 GB).

Generation and Counting K-MERS - Stage One

To generate k-mers of input read, the process is done by the iterating length of a string. Every time taking out each substring of k length, the function for the above task is discussed in algorithm as given in Figure 9. The relationship between k-mer number, base number, genome size, and sequencing/ coverage depth can be calculated by equations (4) and (5) shown as below. Let nbase – Total number of bases, nkmers – Total no. of k-mers from data input, Cbase – Expected coverage depth for bases as given in equation (1), Ckmer – depth coverage of k-mers as given in equation (2).

$$c_{base} = \frac{n_{base}}{G} \tag{1}$$

$$c_{kmer} = \frac{n_{kmer}}{G} \tag{2}$$

(L-K+1) k-mers were generated for every read having length L, $\frac{n_{kmer}}{n_{base}} = \frac{\left(L-K+1\right)}{L}$ (3)

Hence,

$$c_{base} = c_{kmer} \times \frac{L}{\left(L-K+1\right)} \tag{4}$$

$$G = \frac{n_{kmer}}{C_{kmer}} = \frac{n_{base}}{c_{base}} \tag{5}$$

Above equations (1)(2) deduced by Michael S.Waterman's group. Given: L-Length and K-kmer size are constant values, to calculate the depth and DNA size with the help of equations (4)(5). As a start determine, total no. of kmers as n and probable kmer coverage depth as c and the kmer results helps in generating n value.

Stream Matching Process – Stage Two

The formula for calculating the probability ratio is as mentioned below, the Hypothesis is that target pattern matching leads to successful, with H_0=1. The generalized equation as shown in equation (6).

$$PR = \frac{P\left(\frac{E}{H_0.I}\right)}{P\left(\frac{E}{H_1.I}\right)} = \frac{1}{P\left(\frac{E}{H_1.I}\right)} \tag{6}$$

H0 The source of DNA is from the target sequence

H1 The source of DNA is from stage 1(K-mer)

Solution 1: Assuming the target sequence is not matched, then genotype frequency is given by equation (7).

$$P\left(\frac{E}{H_0, I}\right) = \begin{cases} P_i^2, G_C = G_S = A_i.A_i \\ 2P_iP_j, G_C = G_S = A_i.A_j \end{cases} \tag{7}$$

Solution 2: To check the relatedness with the target sequence and output of stage1, genotype frequency is given by a mathematical model using Bayesian networks using PR approach is given by equation (8).

$$PR = \frac{P_r\left(\frac{E}{H_p}\right)}{P_r\left(\frac{E}{H_d}\right)} = \frac{P_r\left(\frac{E}{H_p}\right)}{\sum_{i=2}^{N} P_r\left(\frac{E}{H_i, H_d}\right) P_r\left(\frac{H_i}{H_d}\right)} \tag{8}$$

where; *Hp The DNA came from the Target sequence & Hd The DNA came from the output of stage1 not related to the target.*

Example:

Input: TarSeq[] = "ACCGCCAT AACGATTTA AAAAAGGG" & KmSeq[] = "TTTA"

Output: Sequence found at index 14

Input: TarSeq[] = "AAGAACAATAAGAAGA" & KmSeq[] = "AAGA"

Output: Sequence found at index 1

Sequence found at index 10

Sequence found at index 13

Figure 10. Algorithm for Stream Matching

```
//Algorithm   : Stream Matching.
// Abbreviations:
        ISeq: Input sequence, Pat: Pattern to search, e: value for that letter(256 for character), p: Pre
defined Prime number, tlen: text length, plen: Pattern length, q_pat: quotient post hash calculation for
pattern, q_tex: quotient post hash calculation for portions of text of size plen.
//Input       : Input data from Target Sequence and stage one output.
//Output      :Output recorded Stream matching based on Time taken for Execution,
              Number of operations & memory utilization for Stream matching.
Begin
RevisedRabinKarp( ISeq, Pat, e, p )
        1.Initialize tlen= length(ISeq), plen=length(Pat), hash=e^(plen-1) mod p, pp=0, t0=0,
          q_pat=0, and q_tex=0.
        2. for i= 1 to plen
        3. do t_pat = ( e * pp + Pat[i])
        4. q_pat = t_pat mod p
        5. pp= t_pat mod p
        6. t_tex = ( e*t0 + ISeq[i])
        7. q_tex = temp mod p
        8. t0=temp mod p
        9. leve = lev ISeq,Pat(|ISeq|,|Pat|)
        10. for j = 0 to tlen – plen //comparison
        11. do if q= (tj && q_pat =q_tex) && leve
        12. then "Pattern occurs with j shifts"
        13. if j < tlen – plen
        14. t_tex =(e * (tj – Iseq[j+1] * hash) + ISeq[j+plen+1])/ p
        15. q_tex = t_tex / p
        16. t j+1 = ( e *(tj – Iseq[j+1] * hash) + ISeq[j + plen +1] mod p
        17. tj = tj +1
End
```

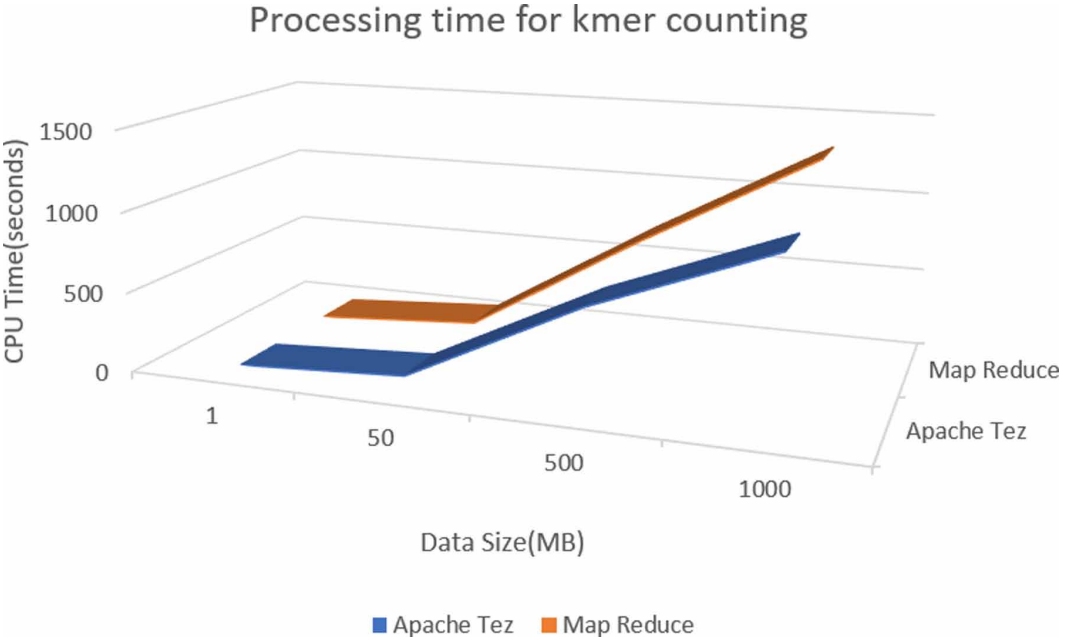
Mathematical definition for Levenshtein distance between two sequences as given by equation (9) (Equation source: Efficient Recursive Levenshtein (edit) Distance Algorithm), a and b (|a|=length of a and |b|=length of b), is given as lev a,b(|a|,|b|):

$$lev_{a,b}(i,j) = \begin{cases} \max(i,j)_{if\ \min(i,j)=0} \\ \min = \begin{cases} lev_{a,b}(i-1,j)+1 \\ lev_{a,b}(i,j-1)+1 \\ lev_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} \quad otherwise. \end{cases} \tag{9}$$

Here, a_i [1] b_i is the function equal to 0 when a_i=b_i & it will be 1 or else. Lev a,b(I,j) is length among the first I letters of a and the first j letters of b.

Figure 10. discusses the algorithm for process of stream matching, let us consider sequence as "acgtcadatga" and in order to search 3mer, then the hash value of the first sub sequence, "acg" using

Figure 11. Performance comparison for K-MER counting



base 101 is given by: ASCII values of a = 97, c =99, g =103, t=116, Hash ("acg") = (97 * 101^2) + (99 * 101^1) + (103 * 101^0) = 999599.

Similarly author try to calculate the hash value of the next sub sequence, i.e. "cga", by considering previous calculated Hash("acg") by deducting the number for the first 'a' of "acg", that is 97 * 102^2, multiply it by the base 101 and add for the new a of "cga", which gives 97 * 102^0. For ex. Hash ("cga") = [101 * (999599- (97 * 101^2))] + (97 * 101^0) = 1020399. Similarly hash can be calculated as, Hash("cgt") = [101 * (999599-(97 * 101^2))] + (116 *101^0) = 1020408.

RESULTS AND DISCUSSION

Figure 11. shows processing time for k-mer counting, where X – axis indicates the size of the file varying from 1 MB to 1 GB and Y – axis represents time taken in seconds for generating k-mers. Here in this result, authors have used 8 as the size for k. Results clearly indicate that by using Apache Tez with Pig for giving a better performance with a reduced time of 16.5% compared to basic MapReduce concept.

Figure 12. shows the improved performance of the proposed system compared to basic Hadoop. Pig programmability with RRK helps in reducing time while comparing with database sequences. Pig is having a special feature like compressed storage capability, this helps in reducing the number of operations. Around 36% of efficiency is achieved. Here Sequence1 is of 7-mer "AAGGCTA", Sequence 2 is of 6-mer "GGACTA", Sequence 3 is of 5-mer "AACCT" and Sequence 4 is of 4-mer "AAAA". These are the sequences with different k size. By observing the case of Sequence 4, the number of read operations remains almost the same in both cases. This because of the sequence that exists in all data blocks. Hence, the number of operations remains almost the same in both cases.

Pig based framework for Revised RK algorithm is superior to the ordinary MapReduce paradigm. From above Figure 13. it's clear that execution memory taken by pig storage is less by nearly 10%

Figure 12. Number of operations while performing sequence similarity

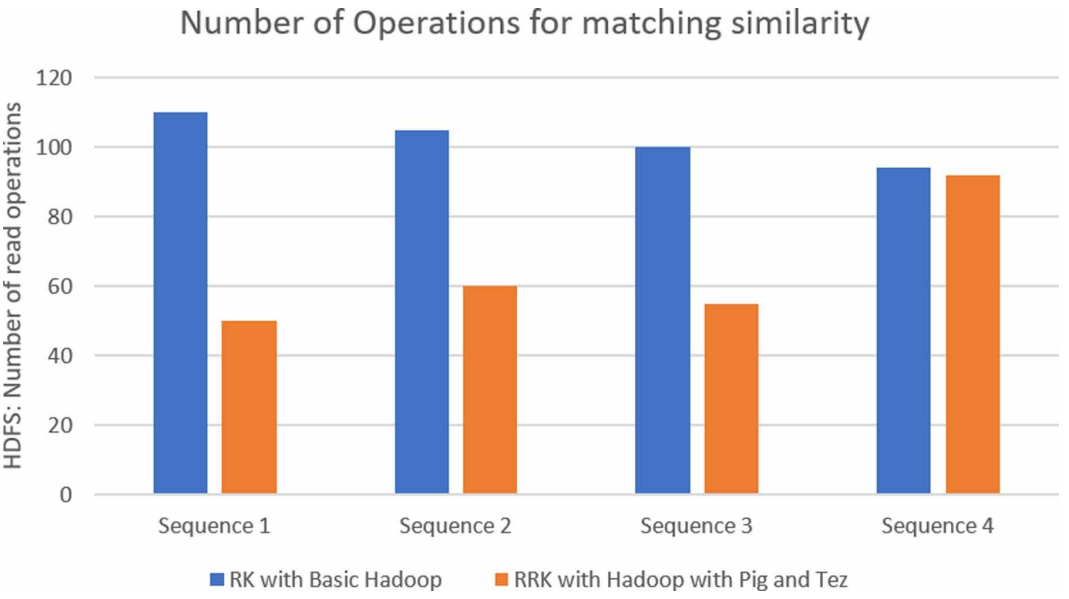
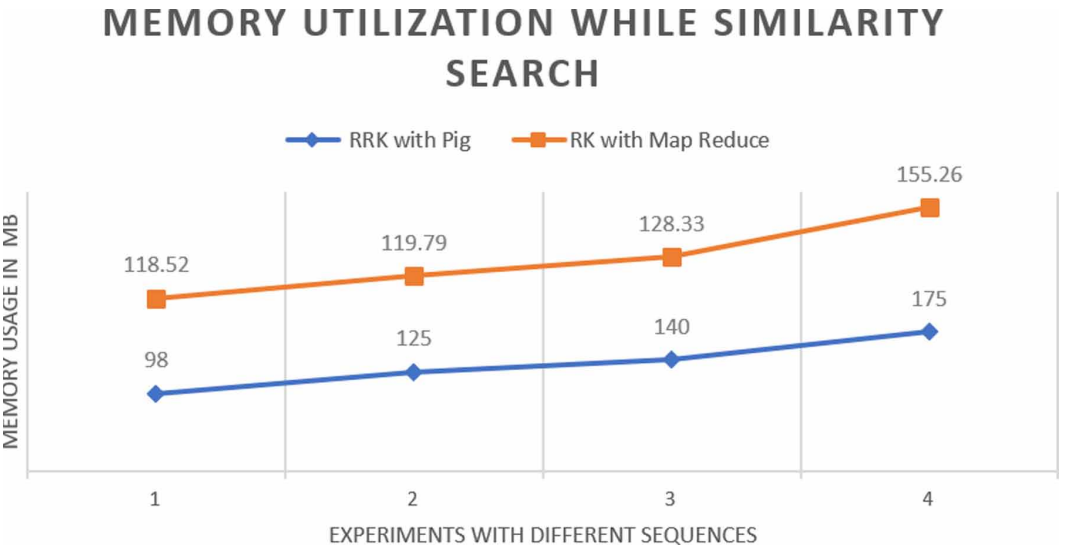


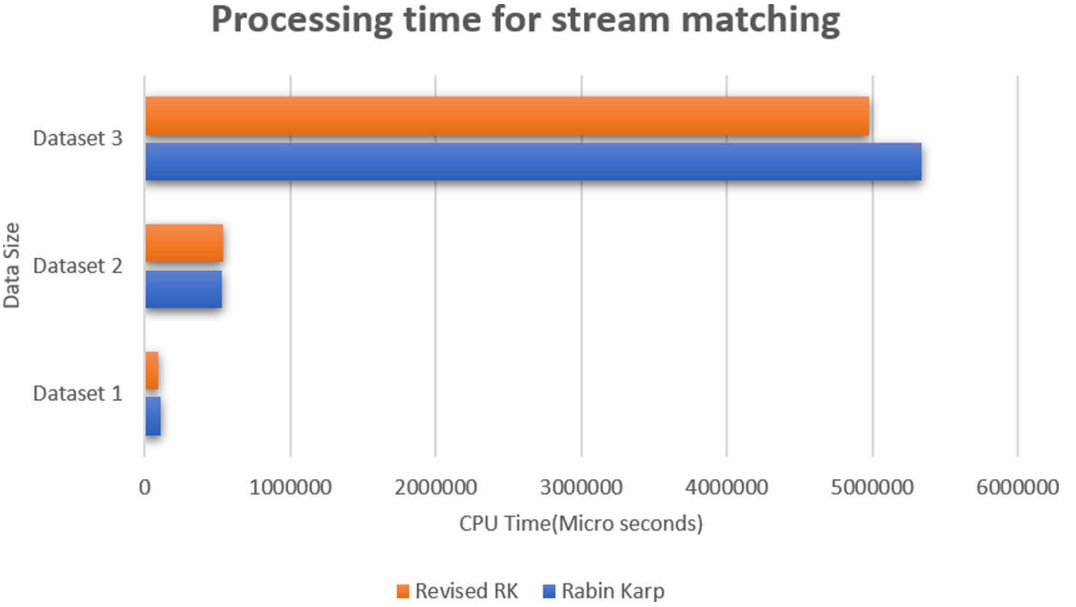
Figure 13. Comparison between execution memory



comparing to ordinary technique. Different sequences considered in X- axis is the same as set of sequences in Figure 9.

Results in Figure 14. depicts the performance of Rabin karp with Revised RK techniques for three different datasets. X- axis represents CPU time in Microseconds and Y-axis represents different datasets. For comparing two techniques authors use Dataset1 with 1MB of file size, Dataset2 with 50 MB of file size and finally Dataset3 of size 500MB. Results illustrate that Revised RK performs well compare to original RK with a time reduction by 8%.

Figure 14. Relating Rabin KARP with Revised RK



CONCLUSION

Sequence Assembly is a process of reconstruction of DNA Sequence by combining and aligning small fragments. Sequencing process is challenging to identify particular gene sequence patterns which can be accomplished using stream matching approaches. These tasks involve huge among of data and the amount of computation required is also very high. Hadoop based technologies are more popular, scalable and have the capacity of processing with large data sets. Proposed solution involved these approaches for genomic analysis with the enhanced environment using pig programmability and speeding up the task by Tez. Pig helps in reducing time while comparing with database sequences. It also helps in compressed storage capability that helps in reducing the number of operations and memory. The revised algorithm proposed in this work is novel. Results conclude that Revised Rabin Karp is superior to existing Rabin karp as it uses additional checking of quotients. It adopts Levenshtein algorithm that calculates hash distance in both pattern and text, resulting in improving accuracy. The authors conclude this work by considering four metrics. These include processing time for k-mer counting reduced by 16.5%, number of operations for matching similarity reduced by 36%, memory utilization while performing similarity search reduced by 10% and finally processing time for stream matching reduced by 8%. Future enhancement can be made possible by reducing I/O operation, that uses Bio pig with Apache Spark. Accuracy related comparisons can be made using an existing algorithm for stream matching and can be enhanced. Researchers can even focus on algorithms that will enhance performance by implementing Genetic algorithms. These findings are sure to assist future researchers whoever undertakes study in this field of science.

ACKNOWLEDGMENT

This research was supported by Ramaiah Institute of Technology (MSRIT), Bangalore-560054 and Visvesvaraya Technological University, Jnana Sangama, Belagavi -590018.

REFERENCES

Alazzam, H., & Sharieh, A. (2018). Parallel DNA Sequence Approximate Matching with Multi-Length Sequence Aware Approach. *International Journal of Computers and Applications*, *180*(26), 1–6. doi:10.5120/ijca2018916594

Behera, S., Gayen, S., Deogun, J. S., & Vinodchandran, N. V. (2018). KmerEstimate: A Streaming Algorithm for Estimating k-mer Counts with Optimal Space Usage. *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics - BCB 18*. doi:10.1145/3233547.3233587

Carvalho, A. B., Dupim, E. G., & Goldstein, G. (2016). Improved assembly of noisy long reads byk-mer validation. *Genome Research*, *26*(12), 1710–1720. doi:10.1101/gr.209247.116 PMID:27831497

Eadline, D. (2016). *Hadoop® 2 quick-start guide: Learn the essentials of big data computing in the Apache Hadoop2 ecosystem*. Addison-Wesley, Pearson.

El-Metwally, S., Ouda, O. M., & Helmy, M. (2014). *Next Generation Sequencing Technologies and Challenges in Sequence Assembly*. SpringerBriefs in Systems Biology. doi:10.1007/978-1-4939-0715-1

Hiraishi, H. (2019). Qualitative and Cognitive Analysis and Modeling Tool for Biological Data. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(2), 30–47. doi:10.4018/IJCINI.2019040103

Huang, G., Zhou, H., Li, Y., & Xu, L. (2011). Alignment-free comparison of genome sequences by a new numerical characterization. *Journal of Theoretical Biology*, *281*(1), 107–112. doi:10.1016/j.jtbi.2011.04.003 PMID:21536050

Jin, S., Peng, J., & Xie, D. (2018). A New MapReduce Approach with Dynamic Fuzzy Inference for Big Data Classification Problems. *International Journal of Cognitive Informatics and Natural Intelligence*, *12*(3), 40–54. doi:10.4018/IJCINI.2018070103

Kurtz, S., Narechania, A., Stein, J. C., & Ware, D. (2008). A new method to compute K-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics*, *9*(1), 517. doi:10.1186/1471-2164-9-517 PMID:18976482

Li, R., Qiu, L., & Zhang, D. (2019). Research on an Improved Coordinating Method Based on Genetic Algorithms and Particle Swarm Optimization. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(2), 18–29. doi:10.4018/IJCINI.2019040102

M, I. T., & R, S. (2013). Study of DNA Sequence Analysis Using DSP Techniques. *Journal of Automation and Control Engineering, 1*(4), 336-342.

Manekar, S. C., & Sathe, S. R. (2018). A benchmark study of k-mer counting methods for high-throughput sequencing. *GigaScience*. Advance online publication. doi:10.1093/gigascience/giy125 PMID:30346548

Miller, J. R., Delcher, A. L., Koren, S., Venter, E., Walenz, B. P., Brownley, A., Johnson, J., Li, K., Mobarry, C., & Sutton, G. (2008). Aggressive assembly of pyrosequencing reads with mates. *Bioinformatics (Oxford, England)*, *24*(24), 2818–2824. doi:10.1093/bioinformatics/btn548 PMID:18952627

Najam, M., Rasool, R. U., Ahmad, H. F., Ashraf, U., & Malik, A. W. (2019). Pattern Matching for DNA Sequencing Data Using Multiple Bloom Filters. *BioMed Research International*, *2019*, 1–9. doi:10.1155/2019/7074387 PMID:31111064

Navarro, F. C. P., Mohsen, H., Yan, C., Li, S., Gu, M., Meyerson, W., & Gerstein, M. (2019). Genomics and data science: An application within an umbrella. *Genome Biology*, *20*(1).

NCBI. (2019). *NCBI Repository – Influenza A virus, Mycoplasma Bacteria, Salmonella Enterica bacteria & Macaca fascicularis animal*. Retrieved from https://www.ncbi.nlm.nih.gov/nuccore

Pahadia, M., Srivastava, A., Srivastava, D., & Patil, N. (2015). Genome Data Analysis Using MapReduce Paradigm. *2015 Second International Conference on Advances in Computing and Communication Engineering*. doi:10.1109/ICACCE.2015.68

Purcell, C., & Harris, T. (2005). Non-blocking Hashtables with Open Addressing. *Lecture Notes in Computer Science Distributed Computing,* 108-121.

Rahman, M. S., & Iliopoulos, C. S. (2006). Algorithms for Computing Variants of the Longest Common Subsequence Problem. Algorithms and Computation Lecture Notes in Computer Science, 399-408.

Rangavittal, S., Harris, R. S., Cechova, M., Tomaszkiewicz, M., Chikhi, R., Makova, K. D., & Medvedev, P. (2017). RecoverY: K-mer-based read classification for Y-chromosome-specific sequencing and assembly. *Bioinformatics (Oxford, England)*, *34*(7), 1125–1131. doi:10.1093/bioinformatics/btx771 PMID:29194476

Sameer, S., Bint, N., Abdul, M., & Hazim, N. (2017). Fast Hybrid String Matching Algorithm based on the Quick-Skip and Tuned Boyer-Moore Algorithms. *International Journal of Advanced Computer Science and Applications*, *8*(6).

Shah, P., & Oza, R. (2017). Improved Parallel Rabin-Karp Algorithm Using Compute Unified Device Architecture. *Information and Communication Technology for Intelligent Systems*, *2*, 236–244.

Shi, L., Wang, Z., Yu, W., & Meng, X. (2015). Performance evaluation and tuning of BioPig for genomic analysis. *Proceedings of the 2015 International Workshop on Data-Intensive Scalable Computing Systems - DISCS 15*.

Tian, G., & Liu, Y. (2019). Simple Convolutional Neural Network for Left-Right Hands Motor Imagery EEG Signals Classification. *International Journal of Cognitive Informatics and Natural Intelligence*, *13*(3), 36–49.

Tripathi, R., Sharma, P., Chakraborty, P., & Varadwaj, P. K. (2016). Next-generation sequencing revolution through big data analytics. *Frontiers in Life Science*, *9*(2), 119–149.

Tripathi, S., & Pandey, A. K. (2016). Identifying DNA sequence by using stream matching techniques. *2016 International Conference System Modeling & Advancement in Research Trends (SMART)*.

Gururaj T. is a Research Scholar at Ramaiah Institute of Technology(MSRIT), Bangalore affiliated to VTU. He received his Master of Technology in Computer Science & Engineering from J. N. National College of Engineering, Shivamogga. (VTU). He is a part-time research scholar at MS Ramaiah Institute of Technology, Bangalore and currently working as an Associate Professor in the Department of Computer Science and Engineering at S. J. M. Institute of Technology, Chitradurga. His main area of interest includes studies related to big data and its applications and Bioinformatics.

Siddesh G. M. is currently working as an Associate professor in the Department of Information Science & Engineering, MS Ramaiah Institute of Technology, Bangalore. He is the recipient of Seed Money to Young Scientist for Research (SMYSR) for FY 2014-15, from Government of Karnataka, Vision Group on Science and Technology (VGST). He has published a good number of research papers in reputed International Conferences and Journals. He is a member of ISTE, IETE, etc., He has authored books on Network Data Analytics, Statistical Programming in R, Internet of Things with Springer, Oxford University Press, and Cengage publishers, respectively. He has edited research monographs in the area of Cyber Physical Systems, Fog Computing and Energy Aware Computing with CRC Press and IGI Global respectively. His research interests include Internet of Things, Distributed Computing and Data Analytics.