# Finding simple sequence repeats (SSRs) within human genome using MapReduce based K-mer algorithm

Sudip Mondal
*Department of Computer Science and Engineering*
*University of Calcutta*
Kolkata, India
sudip.wbsu@gmail.com

Sunirmal Khatua
*Department of Computer Science and Engineering*
*University of Calcutta*
Kolkata, India
skhatuacomp@caluniv.ac.in

***Abstract*—Simple sequence repeats (SSRs) is a collection of the different short and repetitive oligonucleotides sequence that is primarily observed in between any large DNA sequence. In comparison to the standard DNA sequence, these short oligonucleotides may have distinct functions and structural properties. In this article, we have presented a massive parallel processing model for basic operations with k-mers from DNA sequences, based on SSRs function. We introduce a novel approach that finds SSRs using k-mer in the first step. We implement the MapReduce based K-mer algorithm for the advantage of parallel execution, which also enables the analysis of large datasets using of the distributed platform. We have compared the result with KAnalyze, a pipelined K-mer toolkit for finding different sizes of k-mer, and found significant improvement in terms of computational time. Our study shows that MapReduce based k-mer has the potential which reduces memory footprint and accurately identifies every SSR of any specified length. Our proposed MapReduce based k-mer method is useful for finding any Simple sequence repeats (SSRs) in large DNA sequences on any distributed platform.**

***Keywords*—simple sequence repeats, K-mer, MapReduce, Hadoop, Distributed Computing**

## I. Introduction

High throughput sequencing technologies create a huge volume of short reads in a single run. The modern sequencers machine randomly extracts different positions from the DNA sequence and provide the set of short sequence reads [1]. The large dataset produced by these sequencing technologies are very error prone as compared to conventional Sanger sequencing methods [2]. These errors [3] are important and removing them is one of the major computational challenges [4]. The different known method includes tandem repeats, simple sequence repeats, long terminal repeats [5] are the technique among those types commonly found in eukaryotic species.

Simple sequence repeats (SSRs) have different name includes simple tandem repeats (STRs) or microsatellites which are nothing but a short repeat of one to six nucleotides [6]. Oligo-nucleotides indicates the combination of short DNA or RNA moleules and it has an application for different fields including genetic testing, genetic research,etc. We study the problem of detecting Simple sequence repeats (SSRs) within a large sequence. It plays a major role in NGS data analysis [7]. The Simple sequence repeats (SSRs) are short duplicated regions found in DNA or RNA sequence where one or more bases are tandemly replicated several times because of slipped-strand mispairing and errors occurring in DNA replication [8].

Finding SSR is an important task because it is not only essential for different species to calculate the match region, but also required for the same species within different chromosomes. [9]. SSRs reveal characteristic functions of DNA replication and improvement, they are essential in investigating biological systems, as well as studying different diseases using next-generation sequencing data. There are various areas where SSR can be used for investigating different functionalities of the biological sequences. One of the studies [6] involves SSR which shows that length preference has regulatory functions in human genomes. They state that a few SSRs with specific longer repeat lengths are enriched in the genomes. In case of regulatory RNAs like microRNAs (miRNAs), SSRs are used for analysis to understand

978-1-5386-6026-3/18/$31©2018 IEEE

the biological processes. The interaction among SSRs and miRNAs is not yet clearly explained. In the study [10] where SSRs in primary miRNA (pre-miRNAs) transcripts gives evidence to determine the role of microRNAs. The outcome of the research is to show the portion of SSRs in all the available primary miRNAs including plants, animals, and viruses by performing computational analysis.

Progress has been made in the development of molecular markers and hence the application of molecular markers is more superior nowadays. SSRs are the tandem repeats of DNA sequence but now we can use it for developing genetic markers. These molecular markers are important tools for fundamental and applied biological studies. The work [11] described a novel protocol for SSR marker development and SSR utilization at the whole genome system. They have used the genome-wide SSR characterization and development of marker across seven Nicotiana genomes.

Other more recent works like, in silico analysis of SSRs could help to disclose different aspects of the relationships and dynamics of SSRs in eukaryotic genomes [12]. They have identified and described SSRs and their motifs, and with detailed surveyed of their distributions and variations in intragenic and intergenic regions. Apart from this they have also addressed the abundance of different SSR types and checking motifs are similar or not in different genomic regions. Another novel work [13] shows the analysis of evolutionary trends of microsatellite or SSRs distribution across 15 taxonomic subgroups. They have also used a comprehensive SSR mining tool to identify microsatellites in 719 eukaryotic species and identified a large number of novel taxon-specific SSR enrichment patterns. As a result, they have shows variations in SSR abundance in the context of genome size, GC-content, and k-mer size of the motifs discovery.

Several methods have been used to identify SSRs including KAnalyze [14], Kmer-SSR [15]. But one of the conventional approaches is k-mer algorithm. A k-mer is a short biological sequence consisting of a fixed number (k length) of nucleotides (i.e A, T, G, C). It also introduces a sub-sequence of length k acquired from a given sequence. Counting k-mers indicates counting the number of occurrence of all possible k-mers in a sequence. Nowadays, counting k-mers in any biological sequence (DNA, RNA) has an essential step in various applications includes single-cell sequencing, genome or transcriptome assemblers and for error correction of sequence reads [16], [17].

In recent years, the MapReduce programming model has been used to provide the computational power of using clusters computing [18]. Different MapReduce based framework [19] [20] are available where large sequence data can be handled using a distributed environment. The methods MRSMRS [21] have shown the performance but did not mention the memory utilization of the execution process. k-mer counting can be implemented in various ways. One simple approach is hash tables [22], where k-mers are considered as keys and the stored values can be calculated. Although, these methods are relatively outperformed. Later the k-mer counting algorithm was improved with suffix array data structure [23]. But using suffix array is comparatively high computational operation [24]. Our study shows that MapReduce based k-mer algorithm for finding SSRs has great potential for sequence data analysis problems.

We have used Hadoop MapReduce paradigm to implement the SSR using k-mer algorithm. Section II provides background. Section III furnishes the description of the proposed method. Section IV states the Input dataset and Experimental setup. Section V presents the results and analysis. Section VI explains the conclusion and future scope of the work.

## II. Background

**K-mer:** A k-mer is a string over $\Sigma$ i.e {A, C, G, T} whose length is k. Given a small read s, s[p, q] indicates the substring of s from the pth element to the qth element. s can be broken down into n-k+1 k-mers, which we can written as s [1, k], s [2, k + 1],..., s [n - k + 1, n]. In recent times, k-mer based methods [25] have become popular mechanisms for the analysis of large DNA sequences such as chromosomes, whole genomes, parallel short read sequencing. Consider a DNA sequence S consist either of {A, C, G, T}, which are nothing but nucleobases. One of the major challenges is to count the number of occurrences in S of every substring (i.e combination of A, C, G, T ) of length K, that also known as Counting K-mers. For example, let a sequence consist of nucleotides ATTCGTAGTTCCGA. Consider the value of k = 4. So the possible k-mers for the given sequence are ATTC, TTCG, TCGT, so on and count that how many time a particular sequence occurs. We can search for k-mers of any size (I.e length of K) in many application includes preprocessing of fastq files, denovo transcript assembling [26] etc.

**MapReduce:** MapReduce is a programming prototype for distributed and parallel computing framework. It has two important divisions, Mapper and Reducer. Mapper

takes a set of data and transforms it into a different set of data i.e. in convert the data into key and value pairs. In the second step, reducer receives the key-value pairs produced by each mapper and after combining those key-value pairs it processed (reduce) into a smaller set of tuples i.e expected an ultimate result. [27]. The advantage of the MapReduce paradigm is to scale the data processing unit over multiple computing nodes whenever required. The best uses of the MapReduce programming model is Apache Hadoop [28], which is the largest open source distributed processing framework. The major component of Hadoop is MapReduce and Hadoop Distributed File System (HDFS). In the Hadoop ecosystem, MapReduce manages the data processing unit running in clustered systems and HDFS is used for storage of big data [29].

## III. Proposed Methodology

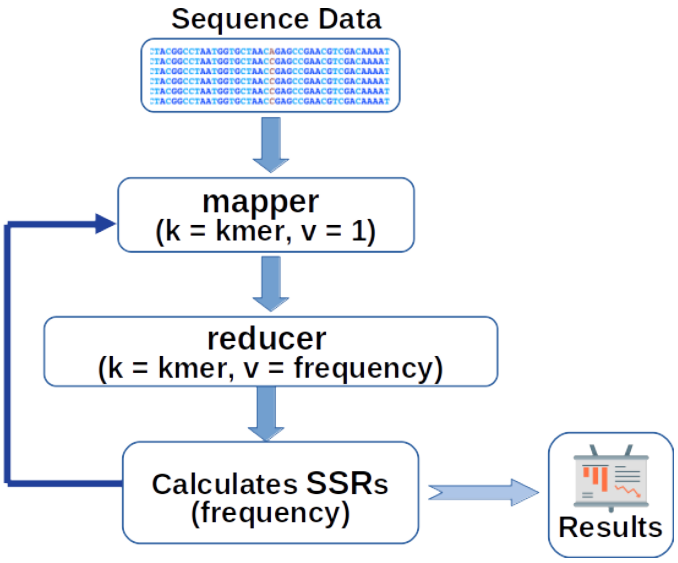


Fig. 1. Workflow diagram of the proposed method

According to the MapReduce paradigm, to find SSRs in a large sequence with the help of k-mer method, the existing algorithm needs to be revised with mapper and reducer function. The basic steps which consist of splitting the input (i.e sequence file) into several parts, where each of the splits assigned and processed by a dedicated mapper function that is executed in a parallel manner. A set of independent reducer function aggregates the final results provided by the mapper tasks. A schematic representation of the proposed method configured to run an algorithm designed for MapReduce is as described in Fig.1.

### A. Algorithm

**Algorithm 1:** MapReduce based SSR Algorithm
**Input:** A sequence S i.e $\Sigma$ = {A, C, G, T}
**Output**: $K_{SSR}$, the set of k-mer with SSR
**Step 1:** Initially the fastq/fasta file need to be stored into HDFS.
**Step 2:** Selecting sequence from HDFS as input and choose the initial seed of the first K-mer to find SSR.
**Step 3:** n $\leftarrow$ length(S) ▷ calculate length of the sequence
**Step 4:**

**procedure** MAPPER($K, n$) ▷ mapper function
  **while** $i \leq n - k + 1$ **do**
    $kmer \leftarrow Sequence.substring(i, k + i)$
    emit (kmer , 1) ▷ emit the k-mer
  **end while**
  **return** $kmer$ ▷ return the k-mer
**end procedure**

**Step 5:**

**procedure** REDUCER($K, value$) ▷ reducer function
  **while** $c \leq value$ **do**
    $sn \leftarrow sn + c$
  **end while**
  **return** $sn$ ▷ return the substring
**end procedure**

**Step 6:** Check Atomicity of SSR ▷ smallest length of each SSR is considered
**Step 7:** Check if there is any overlapping section of SSRs ▷ if SSR is belonged to a different position or may be completely contained within other SSRs
**Step 8:** Selecting new k-mer for each iteration, then find SSR for rest of the sequence repeats Step 2-7.
**Step 9:** Count and Stored the result of all SSRs into HDFS.

## IV. Input Dataset and Experimental Setup

In this experiment, we have considered human chromosomal reference sequence (hg19) downloaded ( ftp://ftp.ncbi.nlm.nih.gov/genomes/H sapiens/ ) from the National Center for Biotechnology Information (NCBI) using FTP protocol. Human reference chromosome consists of 23 pairs. Each of the chromosomes has the separate base pair in length i.e different in size. In total, the human chromosomal reference sequence is about 3.1 billion base pairs long [30] (i.e approximately 3.1 gb in size). Among all the chromosomes, chromosome 1 (248.9 Mb) is larger in size than the rest of 22 chromosomes. In our experiment chromosome 1 has been used for all the tests.

We have used all the analysis with the hardware configuration of Intel Xeon processor 12-core CPUs with speed of 2.6 GHz and 16 GB of internal memory. All experiments were performed on the Ubuntu OS and pre-installed software JDK (Java SE Development Kit)1.8 and Hadoop 2.7.2.

## V. Experimental Results

Identification of SSRs is an essential step for many biological sequence data analysis. It is also important to maintain the accuracy and the optimal execution time to identify SSRs. In order to determine the best performance of k-mer based SSR finding technique, we analyzed it with the performance of Kanalyze. We have used MapReduce technique on k-mer algorithm that splits the job with mapper and reducer to complete the task. MapReduce consumes lesser time to complete the k-mer counting task and also needed less memory as compared to other centralized algorithms. We performed experiments for evaluating the execution time and system memory. For our experiment, we considered a large sequence (i.e. chromosome 1) to compute SSRs. The sequence could be very large (even of size in terabyte also) and it won't affect the computational challenges as it is executed on the Hadoop framework. Further, we can easily add or remove DataNode to the cluster if required. A cluster with more DataNode eventually resolve the computational problem with the help of newly added DataNode processor and also gains the additional space in HDFS.

In our experiment, we have used a single node or Psuedo-Distributed cluster because we can easily compare the utilization of memory and the running time in both of the experiments. Table I described the detailed result of the execution time (in second) comparing normal and Hadoop single node cluster.

TABLE I
Execution Time Comparison

| K-mer length K | Normal Execution Time in Sec. | Hadoop Execution Time in Sec |
|---|---|---|
| 50 | 394 | 103 |
| 100 | 546 | 110 |
| 200 | 830 | 118 |
| 300 | 1437 | 121 |

Fig.2 and Fig.3 summarize the experimental results achieved from time and memory comparison respectively. In both of the figures, the label is named as Hadoop and Normal. Hadoop represents here the time

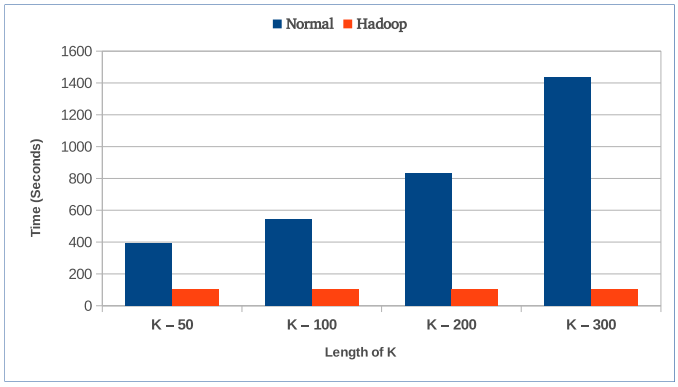


Fig. 2. Time comparison between Kanalyze and Hadoop implementation of SSRs varying different k-mer size (i.e. 50, 100, 200, 300)

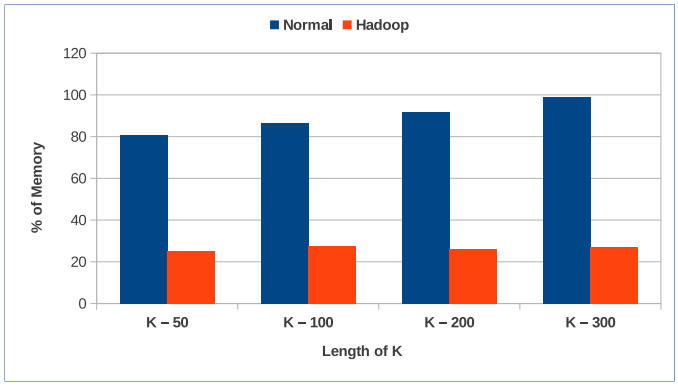


Fig. 3. Comparison of Memory usage by Kanalyze and Hadoop execution with different k-mer size (i.e. 50, 100, 200, 300)

of executing the experiments on a single node Hadoop cluster where both NameNode and DataNode run on a single machine. Furthermore, the normal execution is done using kanalyzer in a single machine.

We have used k-mer algorithm using the MapReduce programming model to find SSRs in a given sequence. Here Fig.2 shows the running time comparison of normal execution using kanalyzer and hadoop execution utilizing k-mer algorithm and also the increasing number of k-mer length (k size) in both cases. In our experiments, we are varying the length of k-mer among 50 to 300 for a comparison of two methods. And each of the length we have collected all the execution time of both cases (i.e normal and hadoop). So from Fig.2, we can see that a significant difference of time comparison between normal and hadoop execution to find SSRs using k-mer algorithm.

On the other hand Fig.3 shows the comparison of memory utilization of normal execution i.e kanalyzer and hadoop execution with the increasing number of length. Here we are varying the length of k-mer among 50 to

300 for finding SSRs and also the comparison of two methods. And each of the length we have collected all the system memory usage in both cases (i.e normal and hadoop). So from Fig.3, we can see that a significant difference in memory utilization between normal and hadoop execution in all the k-mer length we have used in this experiment.

We can conclude from figure (2) and (3) that the MapReduce based algorithm with the execution in Hadoop is more suitable for time and memory utilization in case of any sequence data analysis.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we have used distributed k-mer algorithm to improve the mechanism of finding sequence repeats (i.e. SSRs) in a sequence. In this study, we have observed that MapReduce based k-mer algorithm has great potential for finding SSR more efficiently. Several strategies are available to address the problem of finding SSRs. We provide a fast and efficient solution for solving the SSRs of the human genome sequence using k-mer in a distributed framework. The main focus is on the speed of the algorithm and optimizing the memory requirement. The k-mer algorithm with Hadoop MapReduce showed an increase in efficiency with the reference sequence. Our method shows an improvement of execution time and reduces memory footprint by approximately 33% and accurately identifies every SSRs.

In the future, we are going to implement it in Amazon AWS with the help of Elastic MapReduce (EMR) for better performance as it has an elasticity nature of the computing node. MapReduce based k-mer and SSR algorithm can be a benefit of the different major area including the SSR that can allow to finding motifs, it can help to the development of molecular markers, and using k-mer with SSR method could show regulatory functions in human genomes. Also, this work could be useful for addressing the important problem of transcriptome assembly which is a major component of RNA-seq data analysis pipeline.

## ACKNOWLEDGMENT

This research has been supported by the R&D work undertaken project under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation. This work also partially supported by University Grants Commission, Government of India and TEQIP-II, University of Calcutta. We are also thankful to Dr. Zhumur Ghosh, Bioinformatics Centre, Bose Institute for comments that greatly improved the manuscript.

## REFERENCES

[1] M. L. Metzker, "Sequencing technologiesthe next generation," *Nature reviews genetics*, vol. 11, no. 1, p. 31, 2010.

[2] M. Rho, H. Tang, and Y. Ye, "Fraggenescan: predicting genes in short and error-prone reads," *Nucleic acids research*, vol. 38, no. 20, pp. e191–e191, 2010.

[3] M. Tahir, M. Sardaraz, A. A. Ikram, and H. Bajwa, "Review of genome sequence short read error correction algorithms," *American Journal of Bioinformatics Research*, vol. 3, no. 1, pp. 1–9, 1926.

[4] S. Koren, M. C. Schatz, B. P. Walenz, J. Martin, J. T. Howard, G. Ganapathy, Z. Wang, D. A. Rasko, W. R. McCombie, E. D. Jarvis *et al.*, "Hybrid error correction and de novo assembly of single-molecule sequencing reads," *Nature biotechnology*, vol. 30, no. 7, p. 693, 2012.

[5] T. J. Treangen and S. L. Salzberg, "Repetitive dna and next-generation sequencing: computational challenges and solutions," *Nature Reviews Genetics*, vol. 13, no. 1, p. 36, 2012.

[6] J. Krishnan, F. Athar, T. S. Rani, and R. K. Mishra, "Simple sequence repeats showing length preferencehave regulatory functions in humans," *Gene*, vol. 628, pp. 156–161, 2017.

[7] R. K. Varshney, S. N. Nayak, G. D. May, and S. A. Jackson, "Next-generation sequencing technologies and their implications for crop genetics and breeding," *Trends in biotechnology*, vol. 27, no. 9, pp. 522–530, 2009.

[8] G. Levinson and G. A. Gutman, "High frequencies of short frameshifts in poly-ca/tg tandem repeats borne by bacteriophage m13 in escherichia coli k-12," *Nucleic acids research*, vol. 15, no. 13, pp. 5323–5338, 1987.

[9] T. L. Parchman, K. S. Geist, J. A. Grahnen, C. W. Benkman, and C. A. Buerkle, "Transcriptome sequencing in an ecologically important tree species: assembly, annotation, and marker discovery," *BMC genomics*, vol. 11, no. 1, p. 180, 2010.

[10] N. Joy, Y. M. Beevi, and E. Soniya, "A deeper view into the significance of simple sequence repeats in pre-mirnas provides clues for its possible roles in determining the function of micrornas," *BMC genetics*, vol. 19, no. 1, p. 29, 2018.

[11] X. Wang, S. Yang, Y. Chen, S. Zhang, Q. Zhao, M. Li, Y. Gao, L. Yang, and J. L. Bennetzen, "Comparative genome-wide characterization leading to simple sequence repeat marker development for nicotiana," *BMC genomics*, vol. 19, no. 1, p. 500, 2018.

[12] W.-H. Qi, X.-M. Jiang, C.-C. Yan, W.-Q. Zhang, G.-S. Xiao, B.-S. Yue, and C.-Q. Zhou, "Distribution patterns and variation analysis of simple sequence repeats in different genomic regions of bovid genomes," *Scientific reports*, vol. 8, no. 1, p. 14407, 2018.

[13] S. Surabhi, A. K. Avvaru, D. T. Sowpati, and R. K. Mishra, "Patterns of microsatellite distribution reflect the evolution of biological complexity," *bioRxiv*, p. 253930, 2018.

[14] P. Audano and F. Vannberg, "Kanalyze: a fast versatile pipelined k-mer toolkit," *Bioinformatics*, vol. 30, no. 14, pp. 2070–2072, 2014.

[15] B. D. Pickett, J. B. Miller, and P. G. Ridge, "Kmer-ssr: a fast and exhaustive ssr search algorithm," *Bioinformatics*, vol. 33, no. 24, pp. 3922–3928, 2017.

[16] J. R. Miller, S. Koren, and G. Sutton, "Assembly algorithms for next-generation sequencing data," *Genomics*, vol. 95, no. 6, pp. 315–327, 2010.

[17] A. Bankevich, S. Nurk, D. Antipov, A. A. Gurevich, M. Dvorkin, A. S. Kulikov, V. M. Lesin, S. I. Nikolenko, S. Pham, A. D. Prjibelski *et al.*, "Spades: a new genome assembly algorithm and its applications to single-cell sequencing," *Journal of computational biology*, vol. 19, no. 5, pp. 455–477, 2012.

[18] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[19] R. C. Taylor, "An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics," in *BMC bioinformatics*, vol. 11, no. 12. BioMed Central, 2010, p. S1.

[20] S. Mondal and S. Khatua, "Accelerating pairwise sequence alignment algorithm by mapreduce technique for next-generation sequencing (ngs) data analysis," in *Emerging Technologies in Data Mining and Information Security*. Springer, 2019, pp. 213–220.

[21] H. Cao, M. Phinney, D. Petersohn, B. Merideth, and C.-R. Shyu, "Mrsmrs: Mining repetitive sequences in a mapreduce setting," in *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 463–470.

[22] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and I. Birol, "Abyss: a parallel assembler for short read sequence data," *Genome research*, pp. gr–089 532, 2009.

[23] G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," *Bioinformatics*, vol. 27, no. 6, pp. 764–770, 2011.

[24] S. Sonnenburg and V. Franc, "Coffin: A computational framework for linear svms." in *ICML*, 2010, pp. 999–1006.

[25] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen *et al.*, "De novo assembly of human genomes with massively parallel short read sequencing," *Genome research*, vol. 20, no. 2, pp. 265–272, 2010.

[26] M. G. Grabherr, B. J. Haas, M. Yassour, J. Z. Levin, D. A. Thompson, I. Amit, X. Adiconis, L. Fan, R. Raychowdhury, Q. Zeng *et al.*, "Full-length transcriptome assembly from rna-seq data without a reference genome," *Nature biotechnology*, vol. 29, no. 7, p. 644, 2011.

[27] H. Karloff, S. Suri, and S. Vassilvitskii, "A model of computation for mapreduce," in *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 2010, pp. 938–948.

[28] T. White, *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[29] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The hadoop distributed file system," in *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*. Ieee, 2010, pp. 1–10.

[30] I. H. Consortium *et al.*, "A second generation human haplotype map of over 3.1 million snps," *Nature*, vol. 449, no. 7164, p. 851, 2007.