BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net Software

Volume 11(5)

Compression of Large genomic datasets using COMRAD on Parallel Computing Platform

Christopher Leela Biji[1], Manu K Madhu[2], Vineetha Vishnu[3], Satheesh Kumar K[4], Vijayakumar[2] & Achuthsankar S Nair[1]*

[1]Department of Computational Biology and Bioinformatics, University of Kerala, Thiruvananthapuram; [2]School of Computer Science, Mahathma Gandhi University, Kottayam; [3]Infosys Technologies, Trivandrum; [4]Department of Future Studies, University of Kerala, Thiruvananthapuram; Achuthsankar - Email: sankar.achuth@gmail.com; *Corresponding author

Received May 04, 2015; Revised May 06, 2015; Accepted May 06, 2015; Published May 28, 2015

Abstract:
The big data storage is a challenge in a post genome era. Hence, there is a need for high performance computing solutions for managing large genomic data. Therefore, it is of interest to describe a parallel-computing approach using message-passing library for distributing the different compression stages in clusters. The genomic compression helps to reduce the on disk"foot print" of large data volumes of sequences. This supports the computational infrastructure for a more efficient archiving. The approach was shown to find utility in 21 Eukaryotic genomes using stratified sampling in this report. The method achieves an average of 6-fold disk space reduction with three times better compression time than COMRAD.

Availability: The source codes are written in C using message passing libraries and are available @https:// sourceforge.net/ projects/ comradmpi/files / COMRADMPI/

Keywords: Genome compression, Sequence analysis, Parallel Computing, Big data storage, Genome Analysis

Background:
With the advent of massively parallel computing techniques sequencing technologies generates huge chunks of genome data in peta byte scale **[1].** This in turn poses new challenge of managing, processing and analysing the generated data **[2].** Apart from this the sequencing cost is halved every 5 months where as storage cost is halved for every 14 months **[3].** The DNA specific compression tools can be mainly categorized in to reference based compression tools and reference free compression tools **[4].** Reference based compression tools require a reference genome, while reference free compression tools capture the redundancies within the dataset for a compact representation **[5].** Reference free compression methods involves many different approaches like naive encoding, statistical, dictionary, grammar and transformational methods **[6].**

Many computationally intensive problems in computational biology like ClustalW has already been well adapted to high performance computing **[7].** Though the distribution of various tasksor genome data over many different computers is difficult, genomic revolution trends demand for high performance computing solutions for data storage and management **[8].** Compression algorithm for large dataset requires a vast processing power and memory, which is rather difficult to process on desktop computers **[9]**. The compression algorithm COMRAD (Compression using Redundancy of DNA sets) is one of the best algorithm reported in the literature in terms of compression level as well as data size. The best compression achieved is of 0.25 *bpb* for *S.Cerevisiae* genomes and in this work, our aim is to reduce the computational time of COMRAD while maintaining the compression achieved using parallel computing techniques **[10].**

Methodology:
Materials
COMRAD is a sequential iterative algorithm designed for compressing DNA sequence. Through a sequential multiple passes over the input data repeat substrings are captured for

the creation of corpus-wide dictionary. This stage is further followed by the replacement, clean up and encoding of the files in a sequential manner for compressing large dataset. The outline of standard COMRAD algorithm includes the following stages **[10].**

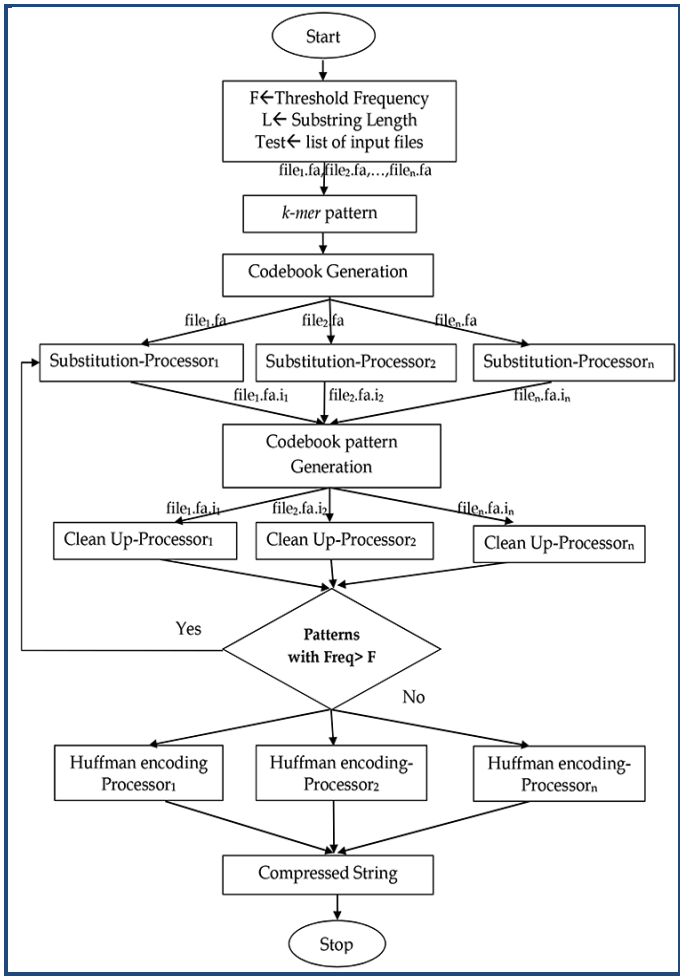

Figure 1: Flow Chart of Compression of Large Genome Dataset using COMRAD on Parallel Computing Platform

Algorithm

1: Create the frequency dictionary D1 of all L length substring, with frequency of at least F, for the input DNA sequences S0
2: Encode the input sequences S0 to get sequences S1
3: k← 2
4: while the dictionary continues to grow do
5: Create the frequency dictionary Dk of all substring matching pattern in P, with the frequency at least F, for the input sequences Sk-1
6: Encode the input sequences Sk-1 to get sequences Sk
7: k<- k+1
8: end while
9: Cleanup Dictionary D to remove infrequent non terminals and make numbering consecutive

COMRAD implementation use only a single core for processing the different compression stages which result in a long run time for compressing large DNA data set in gigabytes range. For instance, while processing *malusdomestica* genome, 75% of time is utilized for codebook creation, substitution, clean up and encoding stages. Considering the huge volume of data generated, there is a need to create frameworks for storing the large genome dataset through parallel computing approaches for saving the compression time. But, the current COMRAD compression algorithm is not adapted to high performance computing. Implementation of parallel algorithms can effectively utilize the resource and will certainly improve the compression time.

In the current study, our objective is to reduce the computational time by parallelizing the COMRAD algorithm. As a first step in this direction we introduce data parallelism by dividing equally the whole genome into n batches and each batch is processed simultaneously by a processor in the cluster computer. Further the parallelization of substitution, clean up and encoding stages were also incorporated. As the inter processor communication is meagre, the proposed algorithm can be put into embarrassingly parallel algorithm. The experimental model involves a MPI (Message Passing Interface) Communication world with 12 processors. **Fgure 1** shows the flow chart of proposed compression algorithm. Parallelism steps involved in the replacement of *k-mers* by non-terminals, clean up and Huffman encoding of each batches are carried out with Message Passing Interface (MPI) standard. This helps to turn out results more rapidly. The phases involved during iteration include

***k-mer* Pattern**

In the proposed framework, initially the large genomes files are split into batches and allocate each batch of files to each processor. The redundant features of genomes are captured using an extensive *k-mer* analysis. For space efficiency *k-mers* are stored in bit encoded form using hash table.

Code book Generation

The different processors will capture the repeated *k-mers* are added to a common dictionary at the master computer. The dictionary is further updated in recurring with the combination of DNA symbols and non-terminals until there is no more frequent *k-mers*. **Figure 2** shows the example of the code book generation.

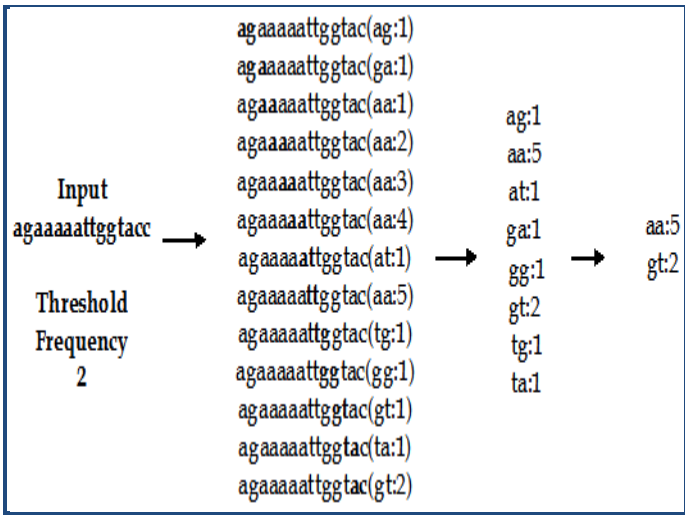

Figure 2: An example showing the code book generation for an input string with string length, L=2 and threshold frequency F=2

Code book Pattern Generation
The algorithm use the feature of COMRAD for defining a set of patterns to detect such that a given substring will match to only one of the patterns or none. On detection of a specific pattern the code book will be updated.

Substitution
The most frequency substrings which exceeds the threshold frequency limit are replaced with non-terminal symbol as unique identifiers.

Clean Up
In the dictionary cleanup step, the algorithm replaces all non-terminals not occurring at least F times with their original substring by allotting the job to different processor which further helps to reduce the time effectively.

Huffman Encoding
The final stage involves Huffman encoding of the final string and the codebook. The most frequent symbols are replaced using fewer binary bits and less frequent symbol with higher binary bits there by helping to have substantial reduction in size.

Dataset Selection
The dataset is prepared based on a stratified sampling procedure, as the NCBI (National Center for Biotechnology Information) database support a classification of genome. The designed dataset includes input file size from 48 MB to 3 GB. The data of the speedup test comprises of the stratified sample of higher order Eukaryotic genomes of Mammals, Birds, Fishes and plants - *Bos Taurus, Felis catus, Gorilla gorilla, Homo sapiens, Mus musculus, Gallus gallus, Taeniopygia guttata, Danio rerio, Oryzias latipes, Arabidopsis thaliana, Citrus sinensis, Fragaria vesca, Malus domestica, Oryza brachyantha, Solanum lycopersicum* and *Zea mays*.

Result & Discussion:
The experiments are run over Rocks cluster (Rocks version 6.1 (Mamba) with Cent OS 6.3-64 bit version) which is an implementation of "Beowulf" cluster, running Sun Grid scheduler for job submissions. Each node is a dual six-core Intel®XeonE5645 series 2.40GHz rack server with 64GB RAM. The performance of Genome Compression using parallel computing tool is analyzed based on the Compression run time (Sec), Compression in bits per base (*bpb*) and Speedup ratio (S) **[10,7].** Compression in bits per base and Speedup ratio is defined as

$$CB = \frac{\text{Compressed size in bits}}{\text{Total No:of bases in the sequence}}$$

$$\text{Speedup ratio} = \frac{\text{Serial Compression time}}{\text{Parallel Compression time}}$$

Figure 3 shows the compression time improvement and speedup ratio for *Homospaiens* (Mammals), *Malus-domestica* (Plants), *Gallus gallus*(Bird) and *Danio rerio* (Fishes) in developed algorithm. Experiment is repeated after splitting the whole genome equally between different processors on the cluster from n=2 to 12. As the number of processors is increased, the elapsed wall time is reduced. The sequential COMRAD require 8 hours, 91 minutes , 86 minutes and 41 minutes to effectively compress the *homospaiens, danio rerio, gallus gallus and malus domestica genome* but the implemented developed algorithm could effectively compress it in just 3 hours, 30 minutes, 29 minutes and 15 minutes.

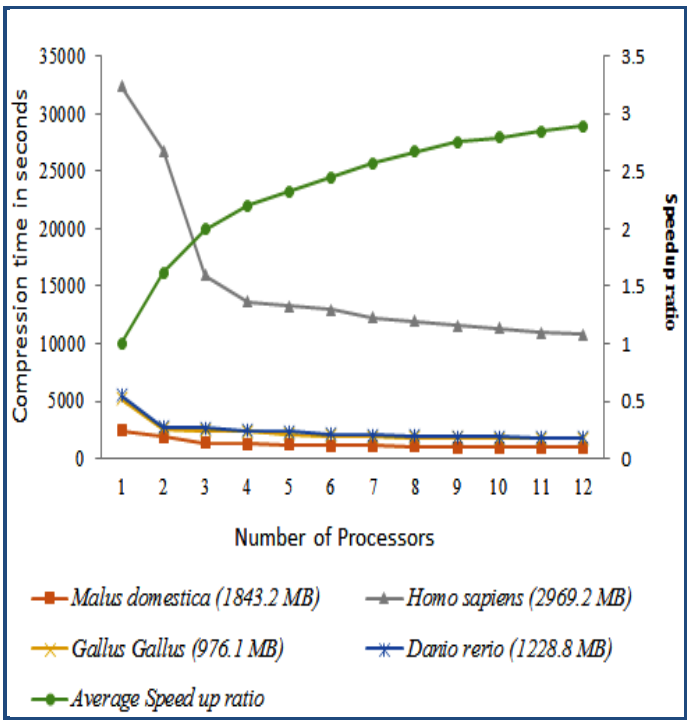

Figure 3: Compression time and speedups for Compression of Large Genomic Datasets using COMRAD on Parallel Computing Platform for selected sample data set with size in MB as a function of number of processors.

The proposed method is competitive in terms of compression time to the sequential compression COMRAD algorithm. We observe that an average compression run time of three times better than sequential COMRAD. We further extended the experiemental analysis by appending more dataset together. While adding more dataset, redundancy with in the dataset is increased and the developed algorithm was able to compress multliple files relatively faster than COMRAD while maintaining the same compression. Among the list of land *plants Arabidopsis thaliana, Fragaria vesca, Oryza brachyantha* have relatively less repeat patterns with the index of repetitiveness 0.14, 0.03, 0.05 and 0.06 as depicted in **Table 1 (see supplementary material) [11].** Hence individually the sequences are hard to compress. The individual genome reported a compression of 2.13 *bpb*, 1.99 *bpb*, 2.02 *bpb* and1.92bpb respectively using COMRAD. But when all the plants genomes in the sample dataset are pooled up the overall compression improved to 1.34 *bpb* both in COMRAD and developed algorithm using parallel computing plat form. But the elaspsed time for COMRAD is 6 hours for compressing the plant genomes of size 5.2 GB. While COMRAD using parallel computing technique took just 2 hours for compressing the plant genomes, which shows the advantage of newely developed algorithm. Yet another advance of pooling the dataset is that dictionary size almost remain same at 155 MB, even though input dataset size varied for each experiment. So even if we further append the dataset, the average

compression achieved will be maintained with the advantage of compression time.

Conclusion:

We have shown a parallel computing approach for compressing large genome dataset. The results of our experimental studies demonstrated that parallel computing algorithms are worthy alternatives which can pave a new direction for effective genome data storage and management system. We have scaled the COMRAD algorithm tobe adaptable in a high performance computing multicore processors. There is approximately 65% of compression time improvement with the parallelization of substitution stage, clean up and Huffman encoding stage.

Acknowledgement:

We thank the Campus Computing facility, University of Kerala especially Mr Deelip Kumar, Systems Manager for all the technical support.

Funding: This work was funded by State Inter University Centre of Excellence in Bioinformatics, University of Kerala, Thiruvananthapuram, India.

Reference:

[1] O'Driscoll A *et al. J Biomed Inform*. 2013 **46**: 774 [PMID: 23872175]
[2] Marx V, *Nature* 2013 **498** : 255 [PMID: 23765498]
[3] Koboldt DC *et al. Brief Bioinform*. 2010 **11**: 484 [PMID: 20519329]
[4] Rozov R *et al. BMC Bioinformatics* 2014 **15**: S7 [PMID: 25252952]
[5] Hsi-Yang Fritz M *et al. Genome Res.* 2011 **21**: 734 [PMID: 21245279]
[6] Zhu Z *et al. Brief Bioinform* 2015 **16**: 1 [PMID: 24300111]
[7] Li KB, *Bioinformatics* 2013 **19**: 1585 [PMID: 12912844]
[8] Schadt EE *et al. Nat Rev Genet*. 2010 **11**: 647 [PMID: 20717155]
[9] Pinho AJ & Pratas D, *Bioinformatics* 2014 **30**: 117 [PMID: 24132931]
[10] Kuruppu S *et al. IEEE/ACM Trans Comput Biol Bioinfor*. 2012 **9**: 137 [PMID: 21576758]
[11] Haubold B & Wiehe T, *BMC Bioinformatics* 2006 **7:** 541 [PMID: 17187668]

Edited by P Kangueane
Citation: **Biji *et al,*** Bioinformation 11(5): 267-271 (2015)

BIOINFORMATION

open access

Supplementary material:

Table 1: Index of repetitiveness for the Genome

No	Genome dataset	Genome size in MB	Indexof repetitiveness
1	*Arabidopsis thaliana-* (ATL)	115.3	0.14
2	*Citrus sinensis*(CSL-)	231.3	0.20
3	*Fragariavesca* (FVL)	191.8	0.03
4	*Malusdomestica*(MDL)	1843.2	0.32
5	*Oryzabrachyantha*(OBL)	242.7	0.05
6	*Solanumlycopersicum*(SLL)	735.2	0.08
7	*Zea mays*(ZML)	1945.6	0.30
8	*Daniorerio*(DRF)	1228.8	0.21
9	*Oryzias_latipes*(OLF)	661.6	0.09
10	*Gallus gallus*(GGB)	976.1	0.01
11	*Taeniopygiaguttata* (TGB)	949.9	0.02
12	*Apis_mellifera* (AMI)	196.8	0.02
13	*Bombus_terrestris*(BTI)	206.4	0.02
14	*Drosophila melanogaster* (DMI)	116.4	0.14
15	*Drosophila_pseudoobscura*(DPI)	48.9	0.02
16	*Tribolium_castaneum* (TCI)	133.2	0.04
17	*Bostaurus*(BTM)	2560	0.27
18	*Feliscatus*(FCM)	2252.8	0.03
19	*Gorilla gorilla* (GGM)	2867.2	0.22
20	*Homosapiens* (HSM)	2969.2	0.23
21	*M musculus* (MSM)	2355.2	0.04