Development of MapReduce and MPI Programs for Motif Search

Mejdl Safran[1], Saad Al-qahtani[2], Michelle Zhu[1] and Dunren Che[1]
[1]Department of Computer Science
[2]Department of Electrical Engineering
Southern Illinois University
Carbondale, IL, USA
{mejdl.safran, sa3d8a7tani}@siu.edu, {mzhu, dche}@cs.siu.edu

***Abstract* — As one of the important problems in molecular biology, motif search is computationally expensive, especially when the size of DNA sequences is large. Extended from a graduate course project in parallel and distributed computing (PDC), this paper investigates two different programming frameworks, namely MapReduce and MPI on motif finding. We implemented a serial algorithm, a MapReduce based algorithm, and a MPI program to calculate the best motif in given DNA sequences. The experimental results demonstrate that our MPI program outperformed both the MapReduce-based algorithm and the serial program with superior efficiency.**

***Keywords* —** *MPI; MapReduce; algorithm; parallel processing; motif;*

I. INTRODUCTION

One of the important problems in the field of bioinformatics is Motif search. Motifs are short, recurring patterns in DNA sequences that may have or may lead to a biological function [1]. Often they indicate sequence-specific binding sites for proteins such as nucleases and transcription factors (TF). A motif can be located anywhere within the regulatory region (RR) of genes. It is computationally expensive to find the best motifs. In this poster paper, we investigated two different PDC (parallel and distributed computing) frameworks, namely MapReduce and MPI (message passing interface), to discover motifs. MapReduce is a programming framework for processing and generating distributed large datasets [2]. Users need to implement a map function that maps the input data to key/value pairs and a reduce function that merges all intermediate values corresponding to the same intermediate key [2]. MPI has a relatively long history, stated in early 1990s [3] and was originally designed for distributed memory architectures.

Based on a graduate class project on the topic of parallel programming, we designed and implemented MapReduce-based and MPI-based algorithms to find the best motifs besides the serial motif finding algorithm. We also conducted experiments on a remote cluster to compare the performance of the three programs using a portion of DNA dataset taken from the Genome Informatics Lab at Indiana University [5]. Our experimental results showed that our MPI program outperformed both the MapReduce and the baseline serial programs. The communication cost of MapReduce seems to be the main bottleneck when the benefits of distributing the workload could not outweigh the overhead in the communication. This insight implies that suitable applications for MapReduce are those applications that involve intense computation but mild inter-data communication.

The rest of the paper is organized as follows. In section II, the serial, MapReduce, and MPI approaches to discover the best motifs are discussed. In section III, our experimental study and results are presented. Finally in section IV, we conclude this paper and point to with future work.

II. PROPOSED ALGORITHMS TO FIND MOTIFS

This section discusses and illustrates the three implementation approaches: serial, MapReduce-based and MPI-based to discover the best motif. The inputs of the programs are the dataset (DNA sequences), the number of DNA sequences (t), the size of a DNA sequence (n), and the size of the motif we want to find (L). The output is a string of size L that minimizes the total distance over all strings of that length to all DNA sequences.

A. Serial Algorithm

Algorithm 1 shows the serial algorithm to find the best motif [4]. The first thing is to generate all permutation of size L from alphabet {A, C, G, T}. Then, for each permutation s, the function TotalDistance is called to return the total distance which is the sum of all local minimum distances that each is the minimum distance of s to a corresponding DNA sequence. If the new total distance returned is better/smaller than the recorded best distance, then the best distance is updated accordingly. The corresponding permutation s as a candidate motif is saved as the best motif found so far. The local minimum distance is computed by comparing the distance between a candidate motif (L-perm) and every possible sub-sequence of the same length from the DNA sequence under consideration.

978-1-4673-6598-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CLUSTER.2015.82

Algorithm1: SerialMotifSearch(DNA, t, n, L)

```
1: bestMotif ← AAA....A
2: bestDistance ← ∞
3: for each L-perm s from AAA...A to TTT...T
4:       if TotalDistance(s,DNA) < bestDistance
5:                bestDistance ← TotalDistance(s,DNA)
6:                bestMotif ← s
7: return bestMotif
```

B. MPI Algorithm

The MPI approach is implemented by adding some collective MPI functions into the serial C program. We first distribute the permutations data using MPI_Scatter among all the processors. After each processor finishes computing the total distance, the results are combined by MPI_Reduce using the minimum operator.

C. MapReduce Algorithm

We design the map and the reduce functions as shown in Algorithm 2. The map function sets the key to be a permutation of size L, and the value to be the associated minimum local distance in each DNA sequence. The reduce function merges all local minimum distances that have the same key to produce global distance for each unique key. The final step returns the key with the minimum global distance to be the best motif.

Algorithm2: MapReduceMotifSearch(DNA, t, n, L)

```
MapFunction(key, value, context)
 1: for each L-perm s from AAA...A to TTT...T
 2:   for each L-string z from index 0 to value.length()-L+1
 3:       if LocalDistance(s,z) < bestLocalDistance
 4:                bestLocalDistance ← LocalDistance(s,z)
 5:   EndFor
 6:   context.write(s, bestLocalDistance)
ReduceFunction(key, values, context)
 1: for each value v in values
 2:   GlobalDistance = GlobalDistance + v
 3: EndFor
 4: if GlobalDistance < bestGlobalDistance
 5:       bestGlobalDistance ← GlobalDistance
 6:     bestMotif ← key
 7: context.write(bestMotif, BestGlobalDistance)
```

III. Experimental Results

Two experiments are performed on two real datasets with 1000 and 5000 DNA sequences, respectively, taken from the Genome Informatics Research Lab [5]. Each DNA sequence has a size of 61 nucleotides. We choose three different motif sizes: 3, 5, and 8. The serial, MPI and MapReduce programs are implemented using C and Java programming languages. For MPI setup, we use 4 nodes running on Cheetah Cluster Server at Georgia State University. For MapReduce setup, we use 6 nodes running on SIU-CS Hadoop cluster – 6 mappers and 6 reducers. We use the speedup rate and parallelism efficiency as performance metrics to evaluate the three approaches. Table I and Table II show the performance in seconds for running the three programs with 1000 and 5000 DNA sequences, respectively. Table III illustrates the speedup rate and the parallelism efficiency of the MPI program compared to the serial program. We can observe that the speedup and efficiency values increase as the problem size goes up. Note that all programs generate the same final best motif. The MPI algorithm is the best in finding the best motif regardless of the size of both the dataset and the motif. Table III shows the good speedup and efficiency of the MPI program over the serial program. Surprisingly, the serial algorithm outperforms the MapReduce algorithm since MapReduce has much communication cost consisting of splitting the data, communication between the mappers and reducers, as well as the sorting of the keys in the implied middle stage. The startup of MapReduce also takes time and contributes to its slow execution. It may also be due to the slow network conncection between the MapReduce nodes.

TABLE I. Experimental Results (1000 DNA)

Motif size	Program		
	Serial (sec)	*MPI (sec)*	*MapReduce (sec)*
3	0.13	0.066	28.410
5	3.21	0.884	33.564
8	357.03	33.564	593.63

TABLE II. Experimental Results (5000 DNA)

Motif size	Program		
	Serial (sec)	*MPI (sec)*	*MapReduce (sec)*
3	0.42	0.147	44.122
5	9.57	2.497	72.105
8	1038.43	263.689	3109.026

TABLE III. Speedup and Efficiency (5000 DNA)

Motif size	MPI	
	Speedup ($=T_{SERIAL}/T_{PARALLEL}$)	*Efficiency (=Speedup/#Processors)*
3	2.85	0.71
5	3.83	0.95
8	3.93	0.98

IV. Conclusion

Motif finding is computationally expensive. In this paper, we developed MapReduce and MPI motif searching programs and compared with the original serial program. The experimental results show that the MPI approach achieved the best results compared to both the MapReduce and the serial approaches.

References

[1] P. D'Haeseleer, "What are DNA seuqence motif? Nat Biotechnol, vol. 24, pp. 423-425, 2006.

[2] J. Dean, S. Ghemawat, "MapReduce: simplified data processing on large clusters," Communciation of the ACM, vol. 51, no. 1, 2008.

[3] B. Barney, "Massage passing interface (MPI)," 2011, from https://computing.llnl.gov/tutorials/mpi/

[4] N.C. Jones, P.A. Pevzner, An introduction to bioinformatics algorithms, MIT Press, 2004.

[5] Genome Informatics Research Lab: http://iubio.bio.indiana.edu/gil/