Performances of Protein Structure Alignment Algorithms under the MapReduce Framework

Chen-En Hsien   Guan-Jie Hua   Yaw-Ling Lin
Department of Computer Science and Information Engineering
Providence University
Taichung, Taiwan
seanm7739@gmail.com
gt758215@gmail.com
yllin@pu.edu.tw

Che-Lun Hung
Department of Computer Science and Communication Engineering
Providence University
Taichung, Taiwan
clhung@pu.edu.tw

Abstract

Protein structures are essential for correct biological functions because similar structures between proteins allow molecular recognition. Identifying similar structures between proteins provide the opportunity to recognize homology that is undetectable by sequence comparison. Thus comparison and alignment of protein structures represents a powerful means of discovering functions, yielding direct insight into the molecular mechanisms. Currently, there are several techniques available in attempting to find the optimal alignment of shared structural motifs between two proteins.

This paper adopts approach using the MapReduce paradigm to parallelize tools and manage their execution. The experiment shows that the map/reduce-paralleled from the original sequential combinatorial algorithm performs well on the real-world data obtained in from the PDB data set; the computation efficiency can be effectively improved proportional to the number of processors being used.

Keywords: Cloud Computing; Map/Reduce; Protein Structures

1 Introduction

Protein structures play critical roles in vital biological functions [1]. With more than 40,000 protein structures determined by the advances in X-ray crystallography and NMR spectroscopy to date, molecular biologists these days proceed in the direction of analyzing and classifying these protein structures in order to discover the structural relationships with protein functions [2].

Detection of proteins with a similar fold can suggest a common ancestor, and often a similar function [3]. Comparison of 3D structures makes it possible to establish distant relationships, even between protein families distinct in terms of sequence comparison alone. This is why structural alignment of proteins increases our understanding of more distant evolutionary relationships [4] [5]. The link between structural classification and sequence families enables us to study functions of various folds, or whole proteins [6].

There have been several methods proposed to compare protein structures and measure the degree of structural similarity based on alignment of secondary structure elements as well as alignment of intra and inter-molecular atomic distances. The basic ideas are rapid identification of pair alignments of secondary structure elements, clustering them into groups, and scoring the best substructure alignment. For examples, the VAST system [7] is based on continuous distribution of domains in the fold space. The FSSP/DALI system [8] provides two levels of description - a coarsegrained one and one with a fine-grained resolution. The method, CATH, provides the complete PDB fold classification by domains and links to other sources of information. The two methods, CE and LGscore2 [9] focus on the local geometry rather than global features such as orientation of secondary structures and overall
topology (as in the case of VAST or DALI). VAST has been used to compare all known PDB domains to each other. The results of this computation are included in NCBI’s Molecular Modelling Database at http://www.ncbi.nlm.nih.gov/Structure/VAST/vast.html.

Note that there must be an atom-pairing scheme before one can do the structure alignment computation. The first atom of the first selection is compared to the first atom of the second selection, fifth to fifth, and so on. Incorporating with ideas of bipartite matching and 3-parameter isometric transformation, Lin et al. [10], [11] proposed methods of using parametric searching strategies with adaptive controls, and demonstrated that more accurate and similar protein structure pairings are possible comparing to previous known results like VAST [7] or CE [9].

One of the crucial steps of these algorithms is finding a good isometric transformation, which leads to the best atom-pairing alignment between two proteins. In this paper, we propose algorithms of for efficiently locating more suitable isometric transformations of one structure and aligning it to the other structure. Based upon the periodical property of the parametric settings, we propose parametric searching strategies by approximations with power series and trigonometric series. We show the eRectiveness of the proposed parametric searching strategies by a set of experiments, which leads to better alignments of structure pairing in general.

Hadoop [12] is a software framework intended to support data-intensive distributed applications. It is able to process petabytes of data with thousands of nodes. Hadoop supports MapReduce programming model [13] for writing applications that process large data set in parallel on Cloud Computing environment. The advantage of MapReduce is that it allows for distributed computing of the map and reduction operations. Each map operation is independent of the others and all maps can perform the tasks in parallel. In practice, the number of the map is limited by the data source and/or the number of CPUs near that data. Similarly, a set of reducers can perform the reduce operations. All outputs of the map operations which share the same key are presented to the same reducer, at the same time. In addition, one of the important benefits to use Hadoop to develop the applications is due to its high degree of fault tolerance. Even when running jobs on a large cluster where individual nodes or network components may experience high rates of failure, Hadoop can guide jobs toward a successful completion. Many applications of bioinformatics are often computation consuming; sometimes it needs weeks or months to complete the jobs. The traditional parallel models, such as MPI, OpenMP and Multi-thread, are not suitable to such applications, where a fault occurred in some nodes leads the entire application into total failure. In these situations, the Hadoop platform are considered as a much better solution for these real-world applications. Recently, Hadoop has been applied in various domains in bioinformatics [14], [15].

2 Method and Materials

A. Protein structure Analysis

The main idea of our local refinement algorithm for finding a suitable matching between two sets of points before utilizing the RMSD procedure to fine-tune the final result is by adjusting the suitable parameter sets by ways of searching the underlying parametric space.

Root mean squared deviation

The smallest root mean squared deviation (rmsd) is a least-squares fitting method for two sequences of points [8]. The idea is to align atom vectors of the two given (molecular) structures, and use the common least averaged squared errors as a measurement of differences between these two (paired) sequences. Formally, let $P = \{p_1, ..., p_n\}$ and $Q = \{q_1, ..., q_n\}$ be two sequences of points. We assume that $P$ is translated so that its centroid $\left(\frac{1}{n} \sum_{i=1}^{n} p_i\right)$ is at the origin. We also assume that $Q$ is translated in the same way. For each point or vector $x$, let $(x) (i = 1, 2, 3)$ denote the $i$-th $(X, Y, Z)$ coordinate value of $x$ and $\|x\|$ denote the length of $x$. Let

$$\text{RMSD}(P, Q, R, a) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \|Rp_i + a - q_i\|^2},$$

where “$R$” is a rotation matrix and “$a$” is a translation vector. Then, the $\text{rmsd}$ value $d(P, Q)$ between $P$ and $Q$ is defined by $d(P, Q) = \min_{R, a} \text{RMSD}(P, Q, R, a)$. We refer to Martin’s ProFit package (standing for protein fitting system) [16] and write a program to calculate the $\text{rmsd}$ between C-$\alpha$ atoms of paired protein backbones with C language. Fitting was performed using the McLachlan algorithm [17].

Isometric rotation transformation
According to Euler’s rotation theorem [18], any rotation about the origin point can be described by using three angular parameters. The rotation is determined by 3 consecutive rotations with 3 Euler angles \((\alpha, \beta, \gamma)\). The first rotation is done by the angle \(\alpha\) around the \(z\)-axis, the second is done by the angle \(\beta\) around the \(x\)-axis, and the third rotation is done by the angle \(\gamma\) around the \(z\)-axis. See [19] for more detailed discussions about the transformation.

Analog to Euler’s setting, we can also consider the point of north-pole \(n = (0, 0, 1)^T\) on the unit sphere. After the rotation, \(R\), say \(n\) is rotated to another point \(p = (x, y, z)^T\); i.e., \(p = Rn\). Let \(\alpha\) denote the angle \(\angle nOp\). Note that \(\alpha\) determines the \(z\)-coordinate of \(p\). To determine \(x\)-coordinate and \(y\)-coordinate of \(p\), the point is rotated around the \(z\)-axis for the angle \(\beta\) on the unit sphere. Note that there are infinitely numbers of rotation that transform \(n\) to \(p\). The particular rotation \(R\) can be decided by rotating all other points around the vector \(p\) by the angle \(\gamma\). It is not hard to verified that, in such a way, any rigid rotation transformation can be parameterized by the three-tuple \((\alpha, \beta, \gamma)\). Thus, we call a vector \(p = (x, y, z)^T\) on the surface of the unit sphere a probe. Note that the movement of each probe is started from the northpole \((0, 0, 1)^T\) to other points in the sphere. The position of \(p\) is decided by the parameters \((\alpha, \beta)\), and exact rotation is fixed by the self-rotation angle \(\gamma\).

As a result, we reduce the problem of finding the good rotation matrix to the new problem of finding a good 3-parameter setting. The rotation matrix is thus characterized by just adjusting the 3 uniformly distributed parameters.

**Minimum bipartite matching**

We use the minimum bipartite matching algorithm to find the best matching between two sets of points to decide the best matching for the \(rmsd\) procedure. We adopted the Munkres [20], [21] algorithm. In order to improve the efficiency of computation, we rewrite the Munkres algorithm, which is implemented on a public available perl module, by C language.

**VAST**

The structure alignment algorithm of VAST is based on aligning secondary structure elements using an algorithm from the field of graph theory. The output is a neighbors D list. It also contains the complete PDB representative structure comparison structure alignments and a structure superposition tool. The search space for alternative secondary structure elements depends on the length of proteins. All pairs of secondary structure elements (one from each structure) that have the same type are represented as nodes of a graph. This correspondence graph is then searched to find the maximal subgraph such that every node in the subgraph is connected to every other node in the subgraph and is not contained in any larger subgraph with this property. VAST has been used to compare all known PDB domains to each other. The results of this computation are included in NCBI’s Molecular Modeling Database at http://www.ncbi.nlm.nih.gov/Structure/VAST/vast.html.

**Parametric adjustment with trigonometric series**

In the trigonometric series estimation algorithm [22], the three parameters (rotational angles) are assumed to be independent. Three parameters are adjusted one by one, and the estimation function corresponding to the affected \(rmsd\) values is subsequently better estimated by increasing the curvature degree of the estimated function. The trigonometric series function is described as the following:

\[
\begin{align*}
    f(\theta) &= C_1 + C_2 \cos \pi \theta + C_3 \sin \pi \theta \\
    &+ C_4 \cos 2\pi \theta + C_5 \sin 2\pi \theta \\
    &+ C_6 \cos 3\pi \theta + C_7 \sin 3\pi \theta + \ldots \\
    &+ C_{2k} \cos k\pi \theta + C_{2k+1} \sin k\pi \theta.
\end{align*}
\]

where \(f(\theta)\) denote the corresponding \(rmsd\) value with respect of one of the three parameters, \((\alpha, \beta, \gamma)\). The \(k\) usually reflects the numbers of local maximal points in the approximated curve. We propose algorithms to improve the \(rmsd\) value between a protein structure pair by finding better alignment list. A set of experiments is designed to test the parameters; these adjusted parameters are set to perform the experiments over a thousand of protein pairs, which are uniformly random sampled from the PDB database. As the results shown our methods improve the alignment computed by the VAST by an averaged improvement ratios about 11 %. The results of figure 1 demonstrate that the method of using 3D Euclidean distance minimum bipartite matching with trigonometric series estimated parametric searching scheme indeed improves existed known system like the VAST.[23]
since the structure comparison problem, like many scientific computation/simulation problem, is very time-consuming under cases of large structures and large number of paired structures, it is desirable to implement the system under massive parallel machines cluster, e.g., the grid-environment, to increase the throughput of the system.

B. Hadoop MapReduce framework

Hadoop is a software framework for coordinating computing nodes to process distributed data in parallel. Hadoop adopts the map/reduce parallel programming model, to develop parallel computing applications. The standard map/reduce mechanism has been applied in many successful Cloud computing service providers, such as Yahoo, Amazon EC2, IBM, Google and so on. An application developed by Map/Reduce is composed of Map stage and Reduce stage (optionally). Figure 2 illustrates the Map/Reduce framework. Input data will be split into smaller chunks corresponding to the number of Maps. Output of Map stage has the format of \(<key, value>\) pairs. Output from all Map nodes, \(<key, value>\) pairs, are classified by key before being distributed to Reduce stage. Reduce stage combines value by key. Output of Reduce stage are \(<key, value>\) pairs where each key is unique.

Hadoop cluster includes a single master and multiple slave nodes. The master node consists of a jobtracker, tasktracker, namenode, and datanode. A slave node, as computing node, consists of a datanode and tasktracker. The jobtracker is the service within Hadoop that farms out Map/Reduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack. A tasktracker is a node in the cluster that accepts tasks; Map, Reduce and Shuffle operations from a jobtracker.

Hadoop Distributed File System (HDFS) is the primary file system used by Hadoop framework. Each input file is split into data blocks that are distributed on datanodes. Hadoop also creates multiple replicas of data blocks and distributes them on datanodes throughout a cluster to enable reliable, extremely rapid computations. The namenode serves as both a directory namespace manager and a node metadata manager for the HDFS. There is a single namenode running in the HDFS architecture.

C. Protein structure analysis on Map/Reduce framework

Figure 3 illustrates the MapReduce framework for the protein structure analysis. Assume that the number of compute node is \(N\) and the number of protein pairs lines is \(P\) in the protein pairs list file, the \(P\) lines will become \(P\) map tasks and send to hadoop by streaming program. Each computing node receives a map and then performs the analysis work. When a node completes the map task, the node pass the score to the Reduce and receive next map task to compute unless the total map are finished. Generally, each node will be assigned to deal with \(P/N\) maps.

Therefore, Reduce will have \(P\) RMSD scores. The RMSD score is described in the previous section 2.A. The reduce operation writes the protein structure pair analysis scores line by line to the output file on HDFS.

4 Experiments

A. Experimental environment and data source

All of the experiments were performed on one NFS server and four IBM blade server within our Cloud Computation laboratory. Each server is equipped with two Quad-Core Intel Xeon
Figure 2. Example of counting words on MapReduce framework. The input data set \{ (A, B), (A, B, B, A), (B, B, A) \} are split into three maps, and each map process its data. Reduce process the data with same key. The output is number of "A" and "B".

Figure 3. Protein structure analysis use hadoop platform use Map/reduce framework.

2.26GHz CPU, 24G RAM, and 296G disk running under the operation system Ubuntu version 10.4 with KVM and QEMU platform. Under the current system environment, we create 8 virtual machine by KVM, each virtual machine is set one core CPU, 2G RAM, and 30G disk running under the operation system Ubuntu version 10.4 with Hadoop version 0.2 MapReduce platform, and we control each virtual machine execution processes 1 map operation and 1 reduce operation. The total number of the map/reduce operations are up to 8 respectively.

Hadoop version 0.2 MapReduce platform, and we control each virtual machine execution processes 1 map operation and 1 reduce operation. The total number of the map/reduce operations are up to 8 respectively.

The Protein structure data sources are gathered from the World Wide Protein Data Bank (http://www.wwpdb.org), which is a PDB format content with 4-latter entry name. This Data Bank maintains this single archive of macromolecular structural data that is freely and publicly available to the global community. We downloaded the protein structure data from the wwPDB's ftp server (ftp://ftp.wwpdb.org/) and the number of protein structure data is 79,180. These data are treated as the input data for our experiments.

B. Experimental result

To assess the performance of the proposed Hadoop MapReduce algorithm, we compare the computational time between various number of structure data and various number of map/reduce operations. Two factors, the number of protein
structure data and the number of protein structure atoms, affect the performance of sequential algorithm and the proposed algorithm. The RMSD scores are calculated with protein ID pairs. Figure 4 and 5 illustrate the comparisons between sequential algorithm and our proposed algorithm under the MapReduce framework. In Fig. 4 and Fig. 5, the maximum atom numbers of protein pairs are between 1-100 and 101-200, respectively.

The experimental result reveals that the computational time is effectively reduced when more map operations are deployed. Comparing to the original sequential algorithm, two, four and eight map operations improve the computation time by factor of two, four and eight times. The computation efficiency can be effectively improved proportional to the number of processors being used.

In Fig. 4 and Fig. 5, it is observed that computational time increases corresponding to number of protein pair and number of protein atom. The computation time with 1-100 atoms is less than that with 101-200 atoms. More number of atom lead to higher computational cost. These experimental results are corresponding to the algorithm analysis in previous section.

C. Web System

We make a web for user who is interesting in our protein structures alignment system at https://sites.google.com/site/pucspasas/.

The user of our web service usually provides two protein PDB IDs, and the service will search protein structure data from wwPDB ftp and perform the desired protein structure comparison using the chosen set of algorithms.

To avoid the time delay, our web service also provides user access keys for user to check the result later on after they submit the desired query. User can come back and check the result after operation finished by our local server.

5 Concluding Remarks

Comparison of 3D structures is useful in analyzing and classifying these protein structures in order to discover the structural relationships with protein functions. It is possible to establish distant relationships, even between protein families distinct in terms of sequence comparison alone.

Protein structure comparison methods have been mature enough in bioinformatics, but in the implementation of performance above a single computer has been unable to have a better breakthrough.

A good solution is to use parallel computing platform, like Hadoop, and integration into the algorithm, so you can easily enhance the performance of our implementation.

In the sequential version, the algorithm takes a huge time to compute the score between two protein structure, it always spends hours even days. However, in the parallelized-map/reduce version, it's performance can improve the computation time by the factor that the maps/reduces we have. It was greatly speed up the efficiency of this protein structure computation algorithm.

In this paper, we have compared the performances between the sequential version and the parallelized-map/reduce version of our proposed protein structure alignment algorithm. In the future, we will apply more bioinformatics functions to the parallel algorithms to provide more perspectives for biologists to improve the performance of bioinformatic algorithms on for analyzing the increasingly huge bioinformatics data.

Acknowledgment

This research was partially supported by the National Science Council under the Grants NSC-99-2632-E-126-001-MY3

References


Figure 4. Performance comparison between sequential compute RMSD of protein structure pair and MapReduce compute RMSD of protein structure pair using various number of map. The Number of Protein atom has been split up into two types, 1-100 and 101-200.

<table>
<thead>
<tr>
<th></th>
<th>1-100 Pairs</th>
<th>101-200 Pairs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential</td>
<td>3037</td>
<td>13587</td>
</tr>
<tr>
<td>2map</td>
<td>1745</td>
<td>6755</td>
</tr>
<tr>
<td>4map</td>
<td>892</td>
<td>3356</td>
</tr>
<tr>
<td>8map</td>
<td>585</td>
<td>2010</td>
</tr>
</tbody>
</table>

Figure 5. Computing time and the percentage relationship of sequential algorithm and using MapReduce algorithm. The pairs is number of protein structure data pair and be separated form number of protein atom by 1-100 and 101-200.