Prediction of Protein-Protein Docking Sites Based on a Cloud-Computing Pipeline

Hui Li, Jean-Claude Tounkara, and Chunmei Liu

***Abstract*—To predict protein-protein docking sites in a massive protein dataset, we built a cloud computing based computing pipeline. This pipeline conforms to Elastic MapReduce. The implementation of this pipeline includes three components. First, the cloud computing is based on the application of the open source hadoop platform. Second, the pipeline combines several existing protein-protein docking site methods. Third, the pipeline takes advantage of network computing resource to predict protein-protein docking sites by distributed data processing services. The results show our method can highly improve the performance of protein-protein docking site prediction.**

***Index Terms*—Cloud-computing; protein docking site; pipeline.**

I. Introduction

A protein-protein docking site is a three-dimensional structure composed of known receptors and ligands. The study of protein-protein docking sites not only reveals the relationship between protein and protein, but also contributes to protein engineering such as molecular design and computer-aided drug design [1].

The computation of protein docking sites is to establish a large number of compounds of three-dimensional structures from the database, and to discover the target molecule docking sites including rigid body docking, semi-flexible docking and flexible docking. The major problems of the existing methods for predicting protein-protein docking sites are the limitation of existing computation capacity and the high cost of computing resources. 1

Some efforts have used parallelization or graphics processing unit (GPU) to accelerate the calculation, but they require access to a specific type of hardware resource. Due to the flexibility and scalability of computing models, cloud computing has become the main tool to deal with massive data analysis and processing. The virtual cloud-computing is a scalable web-based dynamic platform which manages the flexible network equipment resources of dynamic deployment of physical equipment resources including storage, computing, management and dynamical creation of files. It can help us process the large-scale computing resources.

Various molecular docking software has been widely used, such as: ZDOCK [2], RosettaDock [3] and so on. Each method has either extinguished strengths or drawbacks. Effective combination of these tools would achieve a better result. Considering the distributed computing, flexible storage and scalable management, we attempt to use the cloud computing platform to predict the protein docking sites from large scale datasets not only to save energy of the method but also to improve stability and availability of the application.

In order to improve the prediction of protein docking sites, we apply the MapReduce [4] which is a distributed computing architecture firstly proposed by Google to solve the large amount of data and to convert the result into the file system or database. We design a pipeline by combining diverse public prediction tools of protein docking sites to get the highly reliable protein docking site candidates.

Manuscript received September 25, 2012; revised November 11, 2012. This work was supported by NSF CAREER (CCF-0845888) and NSF Science & Technology Center grant CCF-0939370.

The authors are with the Department of Systems and Computer Science, Howard University, Washington, DC 20059, USA. (e-mail: hli@scs.howard.edu; jeanclaudetounkara@gmail.com; chunmei@scs.howard.edu.)

II. Method

A. *Architecture*

In this paper, we use Cloudera [5] which contributes to Hadoop [6] and related Apache projects to provide a virtual cloud platform and distributed computing platform. Hadoop is an open source software framework that supports data-intensive distributed applications licensed under the Apache v2 license [7].

We design a pipeline to combine the existing methods to improve the accuracy of protein-protein docking site prediction based on Hadoop. We facilitate the graphical management interface of Hadoop for large data processing. The architecture is shown in Fig. 1.

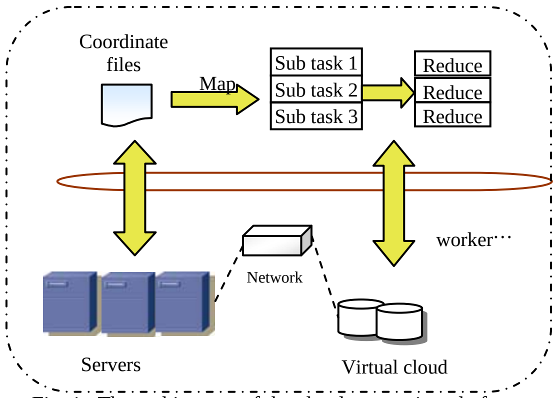

Fig. 1. The architecture of the cloud-computing platform

The distributed computing environment is based on the Map-Reduce framework to process large scale data and Cloudera Desktop to provide a web-based interface. Hadoop's file system (HDFS) is tailored to the huge sort tasks under Hadoop. All related programs are expressed as a series of maps to reduce phase’s operation on the dataset.

1) Load files part

We use a java program to dynamically parse PDB [8] files for all the coordination files on the query protein pairs. All the coordination files are partitioned into chunks and stored on the HDFS.

2) Algorithm part of Mapper

A Mapper is a short program that runs during the mapping phase, which performs the algorithm once it receives a partition data and outputs the intermediate result. A Mapper consists of a primary key and a value. The primary key is the name of the algorithm to identify the input data. The secondary key is the index to the partition of the files. We combine three existing software to achieve a higher accuracy of protein-protein docking site prediction than each individual method from the short protein-protein docking data. Each of the three methods is transferred by the key part.

A Reducer is a short program that runs during the reducing phase, which is designed to collect all the intermediate results, converted by mapping the same key, and output the result to the HDFS.

In the platform, the pipeline automatically generates the primary and the secondary key between the mapper and reducer phases. The implementation of the mapper-reducer phase is extremely efficient that outputs billions coordination data and protein-docking sites. The major process and the roles in the architecture and the steps of the process are explained as the following.

First, the worker in the cloud-computing runs the Mapper and Reducer program. The driver script in the server detects a worker node and allocates a process to the worker. Second, the key/value pair of the Mapper is written to the local disk of the worker node, and then the worker notifies the server. The key for each sub-file is distributed to all workers in the cluster through the Hadoop's file caching facility. Third, we used HDFS method to copy all the outputs of the reducing phases from distributed file system to the master local filesystem.

B. Pipeline

Based on the platform of the cloud-computing, we design a pipeline that combines existing software, which takes advantage of all successful methods. The flow chart of the pipeline is shown in Fig. 2.

Our pipeline for protein docking site predict

An in-house java program module is used to dynamically parse the data from the protein data bank (PDB) and generate ion contains the following methods:

1) ZDOCK [2] is completed by the Boston University research team. It utilizes FFT method for rigid docking, the docking structure of the geometric complementarities, de-salvation energy and electrostatic interactions for rough scoring screening. In order to more accurately evaluate the scoring results, the subsequent development of ZRANK61 adopts more accurate scoring method.

2) RosettaDock [3] is developed by Professor Baker et al at Washington University. It uses a Monte Carlo algorithm to optimize the molecular structure. It includes the process of side-chain, rigid minimum and the final score, and evaluates the performance of the different stages using different scoring functions.

3) HADDOCK [9] is completed by Professor Bonvin et al of Utrecht University. It combines the energy optimization and the molecular dynamics simulations for molecular docking. First the conformational search is through rigid energy optimization and semi-flexible simulated annealing. Then the structure is further improved by the simulations of significant water-bearing molecular dynamics.

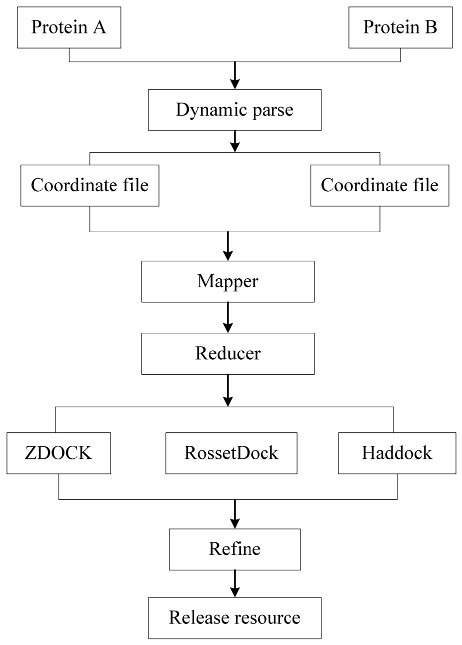

Fig. 2. The flow chart of the pipeline

The pipeline of the open-source EMR is divided into three parts: the first part is a virtual cloud environment, the second is cloud storage, and third part is the distributed computing environment.

If the server receives the coordinate files in the Hadoop, the Mapper and Reducer program is invoked. In order to send subtasks to the other worker, we need to serialize the data and assign to the mapping task. The mapping worker first reads the key/value pairs, and then performs protein-docking mapping function.

The mapping workers output a set of protein-docking sites which include a primary key and a secondary key. The primary key and the secondary key allow reducers to collect separate partitions to be processed in parallel. It also ensures that each reducer worker receive prediction of protein-protein docking sites in a sorted order. The reduce worker traverse all the sorted data and then pass the data to the protein-docking reducing function in the pipeline.

The implementation of the pipeline includes the following five steps:

a big file which includes the coordinates of protein pairs. The Hadoop uses Input Data Format to divide the big file into small input files which record Key and Value.

where DA is the coordinate dataset for protein A and DB is the coordinate dataset for protein B.

All the sub-dataset is an automatic conversion. The Map Reduce library is used to copy all the three algorithms to each worker node. When a sub node returns errors, Hadoop will automatically restart the task on the cached copy of its input data.

1. Set the same parameters to the three algorithms for the prediction of docking sites.

2. The Map process defines the data structure (key, value) on the Map operation. The Map process is applied to each input dataset in parallel. Each of them has a (k2, v2) queue as Equation 1.

$$Map(k_1, v_1) \longrightarrow list(k_2, v_2) \quad (1)$$

Then the data blocks are distributed to different nodes (worker nodes). Each worker node has three methods mentioned above.

3. The Reduce process is applied to collect dataset and combine the results, and then sort the results of the same method using keys as Equation 2.

$$(k_2, \mathrm{list}(v_2)) \longrightarrow \mathrm{list}(k3, v3) \quad (2)$$

where v_3 is a dataset and k_2 includes the identifier for each method. Then the Map Reduce framework collects the output queue in the same key data, put them together, and build a different key named data collection.

4. The query protein data are executed by each method separately. Protein-protein docking sites are then exported and distributed to different workers to achieve better data reliability. The procedure is shown in Table I.

TABLE I: THE PROCEDURE OF THE MAPPER-REDUCER

5.	6.Input	7.Output
8.Map	9.<K1, V1>	10.List (<k2,V2>)
11.Reduce	12.<K2, list(v2)>	13.List (<K3,V3>)

Once output the candidate docking site from each algorithm, the in-house scripts will be used to find the overlap docking sites from each method and refine lists of docking sites from each protein-docking algorithm. The cloud computing is shown in Fig. 3.

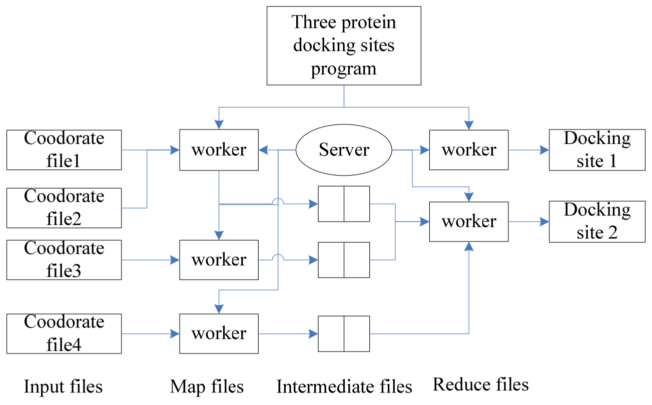

Fig. 3. The general plot of distributed computing

III. EXPERIMENTAL RESULTS

In this paper, we employ the Desktop Cloudera which is an access control interface to provide a simple interface based on the Firefox browser. Installation of Hadoop virtual server template is one of the critical points to build the EMR framework which is a good virtual cloud environment to quickly create a Hadoop virtual server node.

We installed CentOS operating system in the virtual processor, 1GB virtual memory and 60GB virtual hard disk space. We installed Cloudera platform tools that are summarized in the following major steps.

```
#./ jdk-6u16-Linux-i586-rpm.bin
# cp Cloudera-cdh2.repo  /etc/yum.repos.d/Cloudera-cdh2.repo
# yum -y install Hadoop-0.20-conf-pseudo //install Cloudera
#yum -y install Hadoop-0.20-conf-pseudo-desktop //install desktop
```

The interface of the Cloudera starts with user login. The interface after login includes the four main functional modules of the Desktop launch menu in the upper right corner of the interface: the Cluster the Health Dashboard, File Brower, the Job Browser and the Job Designer.

The File Brower will display the uploading request for a plain text file in .txt file format. Clicking Job Designer will activate the algorithm of the pipeline for data input (INPUT) and output parameters to complete the new the protein-protein docking Job tasks created.

After install the Hadoop virtual server template, each node has been installed Linux operating system and Hadoop package. Users only need to configure Hadoop distributed environment which achieves a large scale data processing. Each node can view Hadoop web tools and display the current job progress.

Running the Job task will automatically create a folder showing the progress of the task execution and results, and the final data processing results are saved to the folder. Users can also view the results of data through the result sets which are downloaded to the local system.

The results of the example are shown in Figure 5. The above results were computed on a Hadoop 0.20 cluster with 7 worker nodes located in our laboratory. Centos 5.5 with 4 GB of physical memory and 70 GB of local storage are available for the Hadoop Distributed File system (HDFS) and connected via gigabit Ethernet. The data set we used is shown in Fig. 4.

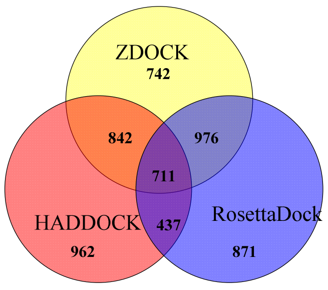

Fig. 4. The candidate data set of the three methods

In order to illustrate the accuracy and efficiency of our pipeline over each of the three methods, we docked the PDB structures on 117 cases and the results are shown in Table II.

TABLE II: THE COMPARISON OF CORRECT INTERFACE RESIDUE CONTACT PAIRS PREDICTED

	Contact %
Pipeline	87
ZDOCK	74
HADOCK	72
RosettaDock	70

IV. CONCLUSION

In this paper we present a cloud computing environment setup by MapReduce and Hadoop which are leveraged to efficiently parallelize the existing protein-protein docking algorithms.

Our method achieves a better accuracy on the datasets than each individual method. We analyze the refine list based on three methods. Our method is flexible to increase or decrease the number of the clusters for rapid completion of the prediction of the protein-protein docking sites. When the task is completed, the computing resources will recover and be assigned to other applications.

REFERENCES

[1] W. Wang, X. Zhou, W. He, Y. Fan, Y. Chen, and X. Chen, "The interprotein scoring noises in glide docking scores, " *Proteins*, vol. 80, pp. 169-83, 2010.

[2] R. Chen, L. Li, and Z. Weng, "ZDOCK: an initial-stage protein-docking algorithm, " *Proteins*, vol. 52, pp. 80-7, 2003.

[3] S. Lyskov and J. J. Gray, "The RosettaDock server for local protein-protein docking, " *Nucleic Acids Res*, vol. 36, pp. W233-8, 2008.

[4] S. G. Jeffrey Dean, "MapReduce: simplified data processing on large clusters," in *Communications of the ACM - 50th anniversary issue:*, vol. 51, 2008, pp. 107-113.

[5] A. Vance, "Bottling the Magic Behind Google and Facebook," *The New York Times.* , vol. 16, 2009.

[6] D. cutting, "Apache Hadoop is a new way for enterprises to store and analyze data., " Cloudera, 2010.

[7] S. Y. X. R. Z. D. Y. T. M. Jiong Xie, J. Manzanares, and A. Xiao Qin, "Improving MapReduce performance through data placement in heterogeneous Hadoop clusters," presented at Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, Atlanta, GA, 2010.

[8] H. M. Berman, T. Battistuz, T. N. Bhat, W. F. Bluhm, P. E. Bourne, K. Burkhardt, Z. Feng, G. L. Gilliland, L. Iype, S. Jain, P. Fagan, J. Marvin, D. Padilla, V. Ravichandran, B. Schneider, N. Thanki, H. Weissig, J. D. Westbrook, and C. Zardecki, "The Protein Data Bank, " *Acta Crystallogr D Biol Crystallogr*, vol. 58, pp. 899-907, 2002.

[9] S. J. de Vries, M. van Dijk, and A. M. Bonvin, "The HADDOCK web server for data-driven biomolecular docking," *Nat Protoc*, vol. 5, pp. 883-97, 2010.

Hui Li is currently a Postdoctoral Fellow at the Department of Systems and Computer Science in Howard University. He received his PhD in Computer Science from Beijing School of Computer Science, University of Technology, Beijing in 2009. His research interests include computational biology, bioinformatics, pattern recognition and algorithm.

Jean-claudeToutara is currently a master student of Department of Systems and Computer Science at Howard University. He will get his M.S. in Computer Science at Howard University in 2012. He is interested in Artificial Intelligence of Experts Systems and Computational Biology.

Chunmei Liu is currently an Associate Professor of the Department of Systems and Comp uter Science at Howard University. She got her Ph.D.. in Computer Science from the University of Georgia; Athens, GA in 2006. Her research interests include Graph Theory, Computational Biology, Algorithms, and Complexity.