CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2013)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3114

SPECIAL ISSUE PAPER

PAR-3D-BLAST: A parallel tool for searching and aligning protein structures

Ahmad Salah [1,2,3] and Kenli Li [1,3,*,†]

[1] *College of Information Science and Engineering, Hunan University, Changsha, Hunan, China*
[2] *Computer Science Department, College of Computers and Informatics, Zagazig University, Zagazig, Egypt*
[3] *The National Supercomputing Center in Changsha, Hunan University, Changsha, Hunan, China*

SUMMARY

Protein structure comparison is a vital process in several tasks like the prediction of protein structures and functions and detecting the proteins evolutionary relationships. The expansion of both the parallel computational hardware and the discovered protein structures stimulates the growth of the parallel computational tools to handle this massive data of proteome. Here, we present a parallel tool, parallel 3D-BLAST (PAR-3D-BLAST), which lists the similar structures to the query protein. Each protein in the result list has a structural similarity score and an alignment to the query structure. The presented tool is implemented to fit both the standalone multi-core computers and clusters of multi-core nodes. The achieved speedup is linear and scalable. The experimental results outline that the speedup increases as the size of the database increases. Using a cluster of 35 computing cores, the tool constructs the database of the entire structural classification of proteins dataset, 108,116 protein entries, in less than 6 min and with average query time of 1.45 s. The obtained speed up is 20 times for database construction and 17 times for searching the query. The tool is an open source and free to use, distribute, and share; it is available at http://aca.hnu.cn/par3dblast. Copyright © 2013 John Wiley & Sons, Ltd.

Received 25 February 2013; Revised 2 July 2013; Accepted 19 July 2013

KEY WORDS: protein structure; structure searching; structure alignment; parallel algorithm; multi-core

1. INTRODUCTION

Because the comparison of enzyme structures binding site is introduced in [1], there were successive advancements in the field of computational structure biology. At the core of this field, protein structure comparison, or alignment, is a vital problem. Solving this problem is a mandatory process in several tasks like drug discovery and design [2], revealing the evolutionary relationship of proteins [3] and predicting protein folding [4]. Into the bargain, the protein structure comparison is used to predict the protein functions, as the structure is linked to protein functions [5].

The protein structure consists of a chain of linked atoms in the three-dimensional (3D) space. Each atom presents an amino acid, which is the building block of the protein. The fold of this chain presents the shape of the protein that matches the functions performed by the protein.

The protein structure data is maintained by several repositories. The Protein Data Bank (PDB) is the largest repository that receives the submitted protein structural data from biologists all over the world [6]. At further level, these structural data of proteins are classified manually, using protein domains as the classification unit, at structural classification of proteins (SCOP) database

*Correspondence to: Kenli Li, College of Information Science and Engineering, Hunan University, Changsha, Hunan, China, 410082.
†E-mail: lkl@hnu.edu.cn

[7] or semi-automatically as in classification, architecture, topology, and homologous superfamily levels (CATH) [8]. These classified databases are useful to judge the accuracy of the computational tools, which perform the structure comparison, because proteins belong to the same class have similar structures.

The classification of the methods proposed to tackle this problem can be classified by different points. The first point depends on the alignment type; whether the alignment is local or global as detailed in [9]. The local structure alignment concerns on similar regions of different proteins, and it is useful for the dissimilar structures. In contrast, the global structure alignment concerns on the similarity of proteins as a whole.

Another point of classification is the similarity metrics, distances between proteins. While the majority of the pioneers' methods based on the root-mean-square-deviation, the recent statistical methods are based on the geodesic distance from a particular Riemannian metric [10].

The third point of classification is the representation of the protein structure. Several methods are based on the usage of graphs to present the protein structures as proposed in [11, 12] and [13]. In these methods, the graph contains nodes representing the secondary structure elements and the edges for their spatial relationship. Another category of methods is based on the structure geometric. The method represents the structures using the distance between C-alpha atoms as in [14] or the distance and angles between protein residues as in [15] and [16], where C-alpha is a kind of protein atoms.

The steady growth of the computational power like multi-core, clusters, and cloud computing systems, beside, the exponential increase of the proteomic data stimulates the development of parallel version of the previous sequential tools. These tools should be able to work on the various parallel environments, from multi-core standalone workstations to a cluster of nodes.

In the rest of the paper, we discuss the main available protein structure databases and methods to tackle the protein structure comparison in Section 2. In Section 3, we explain the methodology of the proposed tool. We discuss the experiments and analyze the results in Section 4. Finally, we conclude the work in Section 5.

2. RELATED WORK

In this section, we discuss the main repositories and algorithms, which side by side tackle the protein structure comparison problem. While there are several repositories and algorithms, we focus only on the ones with significant importance in the meantime.

The Protein Data Bank, managed and maintained by research collaboratory for structural bioinformatics, is a widely used database for protein secondary structure, which store the records of the 3D structure of the atoms and amino acids that build the protein. Each protein is presented by a file of data for both sequence and structure [6]. Figure 1 shows the exponential growth of the number of available proteins in PDB. Structural classification of proteins database is a classified version of PDB. Scientists manually, by visual inspection, classify the data in the PDB files. Scientists hierarchically classify protein structures into four levels; these levels, from the lowest to the highest, are domains, families, super-families and folds [7]. SCOP utilizes four levels of hierarchical classification of structures, which are the following:

(i) Class: General "structural architecture" of the domain.
(ii) Fold: Similar arrangement of regular secondary structures but without evidence of evolutionary relatedness.
(iii) Super-family: Similar in structure and function but not necessarily similar in sequence. It is used to infer the evolutionary relationship.
(iv) Family: Some sequence similarity can be detected.

Figure 2 shows the growth of the last three levels of SCOP per release, the first level is constant and it currently contains 7 items.

The CATH, for class, architecture, topology and homologous, database presents a classification scheme similar to that of SCOP. In CATH, proteins with highly similar structures, sequences, and functions are grouped into sequence families. A homologous super-family contains proteins

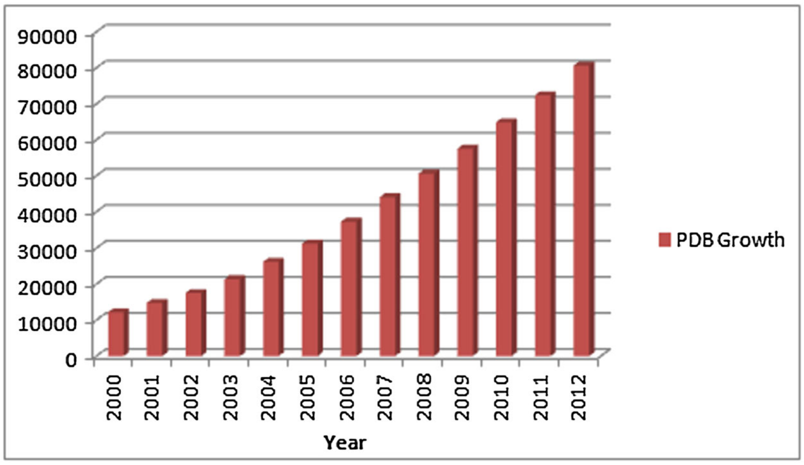

Figure 1. Protein Data Bank growth since year 2000.

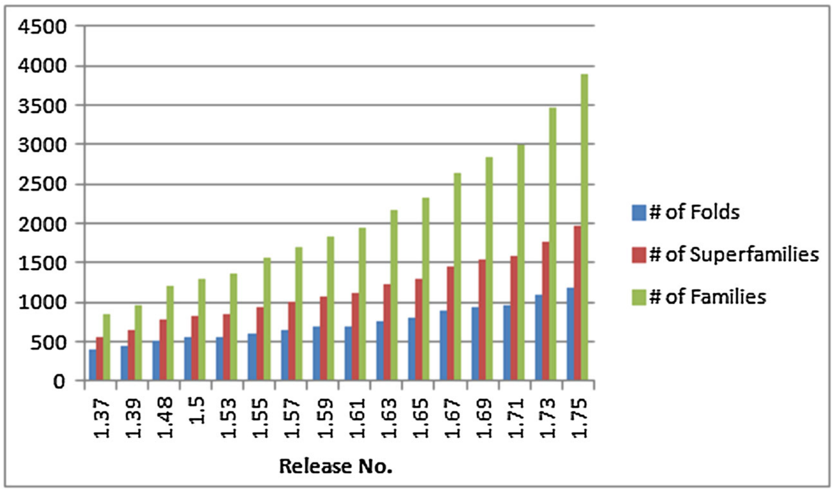

Figure 2. Structural classification of proteins growth by release.

in the form of sequence and structure. A topology or fold family comprises sets of homologous super-families that share the spatial arrangement and connectivity of helices and strands of sheet. Architecture is a classification of proteins based on the similar arrangements of helices and sheets but with dissimilar connectivity [8].

In the following, we discuss the main methods that used the presented databases. Distance matrix alignment represents each structure by a two-dimensional matrix of inter-residual, C-alpha-C-alpha, distances [17]. Similar structures should have similar distance matrices, which are used to find the optimal superposition of the matrices. A parallel version is proposed in [14].

Combinatorial extension (CE)represents proteins as a vector of C-alpha distances for each eight consecutive residues in the structure. Each pair of vectors fragments with distance below a predefined threshold is considered an aligned fragment pair, and then CE employs the CE algorithm to identify and combine the closest aligned fragment pairs between the compared structures [18].

Sequential structure alignment program is based on double dynamic programming to find the optimal structure alignment. It consists of two phases. The first phase constructs a matrix of distances for the entire residues of the protein. This matrix is constructed by using dynamic programming for each possible pair of the compared structures. The second phase uses the dynamic programming to form the alignment by aggregating the matrices obtained in the previous phase [19].

Protein structure indexing using suffix trees is a fast and scalable method proposed in [15]. The method represents the protein structure as a vector of both the distance and angle, torsion angles, of residues. This representation is the rotation and translation invariants. The obtained vectors are then indexed using the suffix tree structure [20]. The process of comparing structures is reduced to searching the suffix tree, which is a linear and simple process. Matching molecular models obtained from theory is an accurate, recent method [21]. It represents the structure as a set of vector; each of size six. The proposed method consists of three steps. In the first step, the similarity matrix is established by using the optimal superposing of the unit-vectors for the compared structures, using the unit-vector root mean square. In a second step, the dynamic programming algorithm finds the optimal path in the similarity matrix. In the last step, it uses a heuristic to identify the largest local structure similarity for a given root-mean-square-deviation threshold.

Laplacian norm alignment is a recent method [22]. The representation of protein is performed over two phases. In the first phase, a graph corresponding to the residues adjacency matrix is represented. In the second phase, the feature values are evaluated by performing the Laplacian operator on the obtained graph at the previous phase. At the comparison step, the authors present two algorithms; one is based on the local similarity meanwhile the other is based on the global.

The SALIGN is a unique and flexible method, because it permits the user to define the properties that represents the structure [23]. First, the method computes the dissimilarity matrix of the compared structures. The dissimilarity score is computed by a weighted sum of the properties representing the proteins. Second and similar to matching molecular models obtained from theory, dynamic programming is used to find the optimal alignment that is presented by the optimal path in the matrix.

The 3D-BLAST is a fast and accurate method; it is as fast as BLAST. Firstly, the method defines 23 states of the structural alphabet. This alphabet represents the protein backbone fragments pattern profiles. The representation of the structure is formed from these predefined states. The database of known protein structures is constructed using this representation, which is finally considered structural alphabet sequence databases (SADB). Finally, for any given structure, the SADB is searched, using BLAST, in order to find the longest common substructures. The method provides the statistical significance (E-value) of the alignment to measure the quality of the output [24].

In this work, we propose a massively parallel tool based on the 3D-BLAST algorithm, which is presented in [24]. The selection of the 3D-BLAST is built upon a compromise between how fast and how accurate is the selected algorithm. 3D-BLAST is as fast as the famous BLAST for protein sequence comparison [25]. The proposed tool implements two different algorithms for two different levels of parallelism. The first level targets the multi-core parallel architectures, and the second level targets the multi-node parallel architectures.

3. THE METHODOLOGY

In this section, we present two algorithms that target two different levels of parallelism. While the first algorithm is designed to benefit from the multi-core architecture, the second can be used on a cluster or on grid architectures. The following two sub-sections are dedicated for explaining these two algorithms.

3.1. Multi-core 3D-BLAST algorithm

The main goal of this algorithm is to speed up the 3D-BLAST using the available CPU(s) cores. The 3D-BLAST treats each database entry separately; it compares the query entry against the database entry and assigns a similarity score. Finally, it displays out the top N scores. In the proposed algorithm, we realize the massive database of the entire protein entries into p small databases sum up to the complete database. Then, we apply the sequential 3D-BLAST tool for each portion using a single core. In the last step, we merge the output results of the entire portions and sort the results by score. We just need to divide the database, launch the p instances of the 3D-BLAST and merge the output files; there is no need to modify the code of the 3D-BLAST. The algorithm is listed in algorithm 1.

Algorithm 1 Multi-core 3D-BLAST

```
mc3D-BLAST($D$, $p$, $Q$)
$D$ is the protein database
$p$ is the number of cores
$Q$ is a set of query protein(s)
  1. Split $D$ into $p$ portions of equal number of proteins.
  2. In parallel, each core execute the sequential 3D-BLAST given one portion of $D$ and the entire $Q$.
  3. Collect and merge the output files of all the $p$ cores.
  4. Sort and display the highest score matched proteins for each query.
```

The first step is sequential, and it can be performed in linear time. The second step depends on the 3D-BLAST complexity time. The third and fourth steps are computational trivial tasks with almost constant time for a small number of p parts to be merged and sorted. The proposed algorithm time complexity is as 3D-BLAST, because it is the main bulk of the algorithm.

3.2. Multi-node 3D-BLAST algorithm

Because the harness of the multi-core architecture is not enough to handle the large-scale proteomic data, so we extend the tool's abilities to be able to work on any massively parallel environment. We design the tool based on the client server architecture. The 3D-BLAST is configured on the entire nodes of the used cluster or grid, and one node will operate as the server to collect, merge, and sort the results. Finally, the server node will forward the top similar protein structures to the end user. Algorithm 2, beside Figure 3, explains the steps of the proposed tool.

First, algorithm 2 divides, distributes, and constructs the database for set D of the entire known proteins as shown in step 1. Each node is assigned m/p proteins, where m is the number of the known proteins in the database, and p is the number of client nodes. The construction of the distributed database consists in the following three steps:

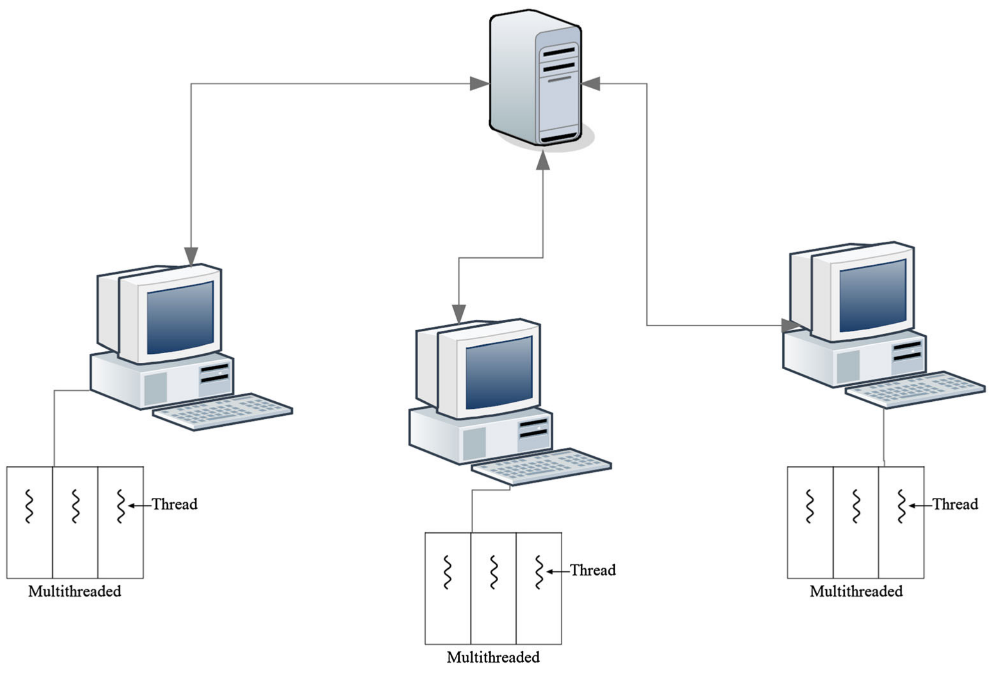

Figure 3. Client-server architecture for the inter-workstation parallel strategy.

Algorithm 2 Multi-node 3D-BLAST

```
mn3D-BLAST($D$, $N$, $p$, m)
$D$ is the set of all known proteins
$N$ is the set of all nodes
$p$ is the number of nodes
$m$ is the number of proteins in set $D$

 1. Server load the list of nodes
 2. In parallel, For $i = 0$ to $p - 1$ do
      i. Node $N(i+1)$: $d_{i+1} = d_{(i \times m/p)+1}, \ldots, d_{((i+1)\times m/p)} \in D$
     ii. Node $N(i+1)$ divides the received $d_{i+1}$ into $k$ portions, where $k$ represents the number of
         cores at that node.
    iii. Use the protein files to construct the SADB database file for each core.
 3. Repeat
 4. Server receives the query protein(s) $q$ from the user.
 5. Server sends $q$ $\forall n \in N$ .
 6. $\forall n \in N$ performs algorithm 1.
 7. $\forall n \in N$ this node has set $R$ which contains the output of algorithm 2.
 8. $\forall n \in N$ send set $R$ to the server.
 9. Server merges all sets $R$ received from all node into $GR$ set.
10. Server sorts $GR$ set elements by their scores.
11. Server forward the top scored proteins , out of $GR$ set, to the user.
12. Until (true)
```

(i) In the first step, the database entries are divided over all the available nodes.
(ii) In the second step, each node divides the assigned portion of proteins database to smaller portions in size, and then assigns the obtained smaller portions one for each core. The number of portions depends on the number of available cores.
(iii) In the last step, the algorithm builds the SADB database for the given proteins.

The algorithm repeats steps 3 through 10 each time the system received a new query. In step 3, the server is ready to receive a protein, or set of proteins, to serve as the query protein(s). The server broadcast the received protein(s) to the client nodes as listed in step 4. In step 5, all of the nodes perform the actual protein structure comparisons between the query protein(s) and the portion of protein stored in the SADB database file using algorithm 2. In step 6, each node should return a set of proteins with the highest structural similarity score along with their scores, where the number of retrieved proteins is user defined. In step 8, the server merges the received sets in on set GR, and then sorts the element of this grand set by proteins scores, as in step 9. Finally, in step 10, the server sends the top scored proteins of set GR, which represents the most structurally similar proteins to the query protein.

3.3. *PAR-3D-BLAST usage*

The proposed tool follows 3D-BLAST license. It is an open source, free to use, distribute and modify. The PAR-3D-BLAST tool uses the original 3D-BLAST as a black-box without any code modification. We harnessed the strategy used in [26]. It, in parallel, runs any sequential program for multiple sequence alignment as a black-box on a multi-core architecture. We extended this strategy to additionally support a cluster of computers.

In contrast with 3D-BLAST, PAR-3D-BLAST has a GUI for ease of use. The user can use the tool in two modes multi-core or local area network. The former is used on a standalone machine with multi-core CPU, with the ability to set the number of the used cores. The latter is used on the local area network of nodes; user has to set the Internet Protocol addresses (IPs) and the number of used cores at each node. After describing the computational resources and for the first use, user has to set the database to operate locally for the former or to distribute the database for the latter. After setting the database, user can select the query to perform the actual comparison. Finally, user has to set the command to run the original 3D-BLAST; thus, the tool is independent of the 3D-BLAST tool and user can tune and use all the available options of the original sequential 3D-BLAST. Database and query files should be in the format of PDB, SCOP (.ent), or Definition of Secondary Structure of Proteins (DSSP) files.

After feeding all the required inputs, the tool launches and manages a number of instances of the sequential 3D-BLAST equals to the overall used number of cores. Then, the tool receives the output files of each instance, merge the results, and displays the top N similar structures as the user set the maximum number of hits. The outputs are text files, which contain a list of database proteins with the top structural similarity to the query protein. Each protein in the list contains a similarity score and an alignment to the query protein.

4. EXPERIMENTS DESIGN AND ANALYSIS

We evaluate the proposed tool by two types of experiments; each part is dedicated to one of the discussed algorithms. In our experiments, we use a two different size datasets based on the SCOP database. Proteins share structural relationships are classified into four different classes. The SCOP dataset provides classified data that can be used to evaluate the accuracy of the proposed algorithms.

4.1. Multi-core speedup evaluation

The experiments at the multi-core level are performed on 2.3 GHz AMD Opteron processor 6134 (AMD, Sunnyvale, California, USA), 8 GB random access memory and two CPUs each contains eight cores. The running operating system is Linux kernel version 3.4.6-2.10-desktop, 64 bits. All the algorithms are written in Java language, using Java development kit version 1.6.0 33.

In all of the proposed test cases, we used the standalone 3D-BLAST Linux beta version 102. We used the default parameter values, which are 50 for the maximum number of hit structures, and 10 for the E-value parameters. These parameters values have no effect on the obtained speedup. The speedup comes from simultaneously comparing different portions of the database instead of sequentially comparing the database as a whole. Thus, we used fixed values for the parameters.

In order to test the gained speedup of the multi-core algorithm, we used a well known test and a newly designed test. Firstly, we have used the classic dataset proposed in [27]; we refer to this dataset by protdex and it contains 34,055 entries. For the second experiment, we used the entire latest release, to the date it is 1.75, of SCOP dataset. We constructed the query list by searching the whole SCOP dataset for super-families, which contains at least ten members, and selected one member of each super-family to create the query list. The dataset contains 108,116 entries, where the original SCOP 110,799, but 3D-BLAST cannot handle 2,683 entries because of several conditions (i.e., file with less than five residues).

Figure 4 shows the gained speedup of constructing the two databases. In SCOP database, the number of entries for 14 cores is 7,722 entries for each core, which results in a speedup of 13 times. On the other hand, we gain speedup of ten times for 2,432 entries in the protdex dataset.

Figure 5 shows the gained speedup of the query time for the two databases. The result expresses an obvious fact that the more the size of dataset and used cores, the more the speedup we can gain. SCOP dataset is almost three times in size as protdex size. The speed of SCOP is almost optimal, while the speedup of protdex is linear but not close of the optimal due to the small size of the dataset. Figure 6 shows the average query time for the tested datasets. The entire SCOP database can be searched within less than 2 s using a standalone machine with 14 cores.

4.2. Multi-node speedup evaluation

The experiments of multi-node level are performed on a cluster of one server and five workstations, all of the same specifications like the one mentioned in the multi-core test, except each workstation contains one CPU of eight cores. We use seven cores out of each workstation and keep one core for operating system purposes. We tested only the SCOP dataset.

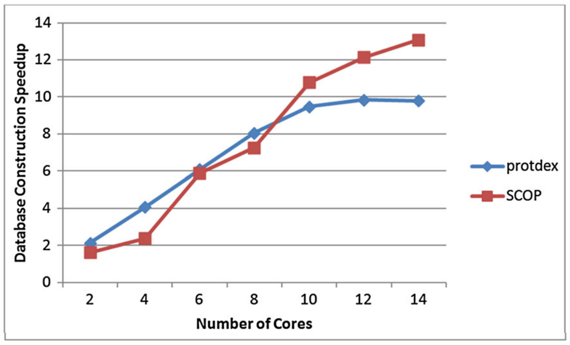

Figure 4. Database construction speedup.

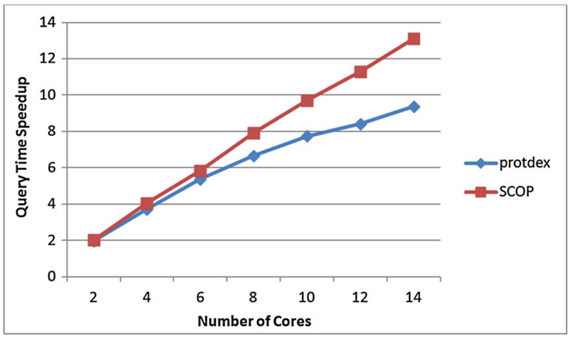

Figure 5. Query time speedup.

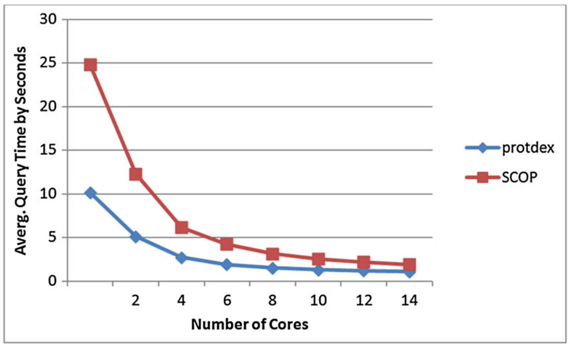

Figure 6. Average query time per second for different number of cores.

Using five nodes (clients) and the entire SCOP dataset, the algorithm can construct the database in 305 s on average, and the average query time is 1.45 s. The speed up of the query searching is less than the database construction because the former includes some network traffic.

In the sake of accuracy, we compared the results of the multi-core and multi-node algorithms, and we found they have the same output. Into the bargain, we obtained the precision-and-recall values of 3D-BLAST for our suggested query list of SCOP, which is listed in Table I. The table contains the precision for the family and super-family SCOP classification for different recall values. For example, if the results contain 100 structures, then the first ten structures have an average of 8.8 correct and 1.2 fault structures for the super-family dataset.

Table I. Precision-recall values for the structural classification of proteins query list.

Recall (%)	Super-family average precision (%)	Family average precision (%)
10	88	91
20	78	82
30	68	72
40	60	65
50	54	58
60	47	51
70	41	44
80	37	40
90	31	33
100	24	27

5. CONCLUSIONS AND FUTURE DIRECTIONS

The steady growth of the proteomic data requires the development of faster algorithms for protein structure comparison. We have presented a parallel tool PAR-3D-BLAST based on the accurate 3D-BLAST protein structure comparison algorithm. The tool can benefit from different types of parallelism from standalone machine with multi-core to clusters or clouds of heterogeneous computers. The tool can search the entire SCOP dataset in less than 2 s for standalone machine using 14 cores, achieving 13 times speedup. In addition, the tool can process a query on average time in less than 1.5 s using a cluster of five nodes with 35 computing nodes, achieving 17 times speedup. For the mentioned standalone machine and cluster, the tool speed up the database construction by 13 and 20 times, respectively.

For future work, we plan to test the proposed tool on Tianhe-1 supercomputer of heterogeneous CPUs–graphics processing unit(s), it has a theoretical peak speed of 1.37 petaflops, and the cloud systems. In addition, we will test other databases besides the SCOP database.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous reviewers for the constructive comments, which improved the quality of this manuscript. This research is funded by the Key Program of National Natural Science Foundation of China (Grant No. 61133005), the Natural Science Foundation of China (Grant Nos. 61070057,61103047), the Key Projects in the National Science & Technology Pillar Program, the Cultivation Fund of the Key Scientific and Technical Innovation Project(2012BAH09B02), the Ministry of Education of China (Grant No. 708066), the PhD Programs Foundation of Ministry of Education of China (20100161110019), the National Science Foundation for Distinguished Young Scholars of Hunan (12JJ1011), and the Ministry of Higher Education of Egypt.

REFERENCES

1. Rossmann MG, Patrick A. The taxonomy of binding sites in proteins. *Molecular and Cellular Biochemistry* 1978; **21**(3):161–182.
2. Zhang C, Lai L. Towards structure-based protein drug design. *Biochemical Society Transactions* 2011; **39**(5): 1382–1386.
3. Orengo CA, Sillitoe I, Reeves G, Pearl FMG. Review: what can structural classifications reveal about protein evolution? *Journal of Structural Biology* 2001; **134**(2):145–165.
4. Thomas S, Tanase G, Dale LK, Moreira JM, Rauchwerger L, Amato NM. Parallel protein folding with STAPL. *Concurrency and Computation: Practice and Experience* 2005; **17**(14):1643–1656.
5. Hegyi H, Gerstein M. The relationship between protein structure and function: a comprehensive survey with application to the yeast genome. *Journal of Molecular Biology* 1999; **288**(1):147–164.
6. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE. The protein data bank. *Nucleic Acids Research* 2000; **28**(1):235–242.
7. Conte LL, Ailey B, Hubbard TJ, Brenner SE, Murzin AG, Chothia C. SCOP: a structural classification of proteins database. *Nucleic Acids Research* 2000; **28**(1):257–259.

8. Orengo CA, Pearl FM, Thornton JM. The CATH domain structure database. *Structural Bioinformatics* 2005; **44**:249–271.
9. Gherardini PF, Helmer-Citterich M. Structure-based function prediction: approaches and applications. *Briefings in Functional Genomics & Proteomics* 2008; **7**(4):291–302.
10. Liu W, Srivastava A, Zhang J. A mathematical framework for protein structure comparison. *PLoS Computational Biology* 2011; **7**(2):e1001075.
11. Krissinel E, Henrick K. Secondary-structure matching (SSM), a new tool for fast protein structure alignment in three dimensions. *Acta Crystallographica Section D: Biological Crystallography* 2004; **60**(12):2256–2268.
12. Harrison A, Pearl F, Sillitoe I, Slidel T, Mott R, Thornton J, Orengo C. Recognizing the fold of a protein structure. *Bioinformatics* 2003; **19**(14):1748–1759.
13. Madej T, Gibrat JF, Bryant SH. Threading a database of protein cores. *Proteins: Structure, Function, and Bioinformatics* 2004; **23**(3):356–369.
14. Holm L, Kääriäinen S, Rosenström P, Schenkel A. Searching protein structure databases with DaliLite v. 3. *Bioinformatics* 2008; **24**(23):2780–2781.
15. Gao F, Zaki MJ. PSIST: A scalable approach to indexing protein structures using suffix trees. *Journal of Parallel and Distributed Computing* 2008; **68**(1):54–63.
16. Gharib TF, Salah A, Salem ABM. PSISA: an algorithm for indexing and searching protein structure using suffix arrays. *Proceedings of the 12th WSEAS International Conference on Computers*, Heraklion, Greece, 2008; 775–780.
17. Holm L, Sander C. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology* 1993; **233**(1):123–138.
18. Shindyalov IN, Bourne PE. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering* 1998; **11**(9):739–747.
19. Orengo CA, Taylor WR. SSAP: sequential structure alignment program for protein structure comparison. *Methods in Enzymology* 1996; **266**:617–635.
20. Ukkonen E. On-line construction of suffix trees. *Algorithmica* 1995; **14**(3):249–260.
21. Ortiz AR, Strauss CE, Olmea O. MAMMOTH (matching molecular models obtained from theory): an automated method for model comparison. *Protein Science* 2009; **11**(11):2606–2621.
22. Bonnel N, Marteau PF. LNA: Fast protein structural comparison using a Laplacian characterization of tertiary structure. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2012; **9**(5):1451–1458.
23. Braberg H, Webb BM, Tjioe E, Pieper U, Sali A, Madhusudhan MS. SALIGN: a web server for alignment of multiple protein sequences and structures. *Bioinformatics* 2012; **28**(15):2072–2073.
24. Yang JM, Tung CH. Protein structure database search and evolutionary classification. *Nucleic Acids Research* 2006; **34**(13):3646–3659.
25. Altschul SF, Madden TL, Schäffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* 1997; **25**(17):3389–3402.
26. Xiangyuan Z, Li K, Salah A. A data parallel strategy for aligning multiple biological sequences on multi-core computers. *Computers in Biology and Medicine* 2013; **43**(4):350–361.
27. Aung Z, Tan KL. Rapid 3D protein structure database searching using information retrieval techniques. *Bioinformatics* 2004; **20**(7):1045–1052.