On the Storage, Management and Analysis of (Multi) Similarity for Large Scale Protein Structure Datasets in the Grid

Gianluigi Folino[1], Azhar Ali Shah[2], and Natalio Kransnogor[2]
[1]CNR-ICAR, University of Calabria, Rende (CS), Italy
folino@icar.cnr.it

[2] School of Computer Science, University of Nottingham, NG81BB, UK
(aas,nxk)@cs.nott.ac.uk

Abstract

Assessment of the (Multi) Similarity among a set of protein structures is achieved through an ensemble of protein structure comparison methods/algorithms. This leads to the generation of a multitude of data that varies both in type and size. After passing through standardization and normalization, this data is further used in consensus development; providing domain independent and highly reliable view of the assessment of (di)similarities. This paper briefly describes some of the techniques used for the estimation of missing/invalid values resulting from the process of multi-comparison of very large scale datasets in a distributed/grid environment. This is followed by an empirical study on the storage capacity and query processing time required to cope with the results of such comparisons. In particular we investigate and compare the storage/query overhead of two commonly used database technologies such as the Hierarchical Data Format (HDF) (HDF5) and Relational Database Management System (RDBMS) (Oracle/SQL) in terms of our application deployed on the National Grid Service (NGS), UK. As the technologies explored under this investigation are quite generic in the science and engineering domain, our findings would also be beneficial for other scientific applications having related magnitude of data and functionality.

1. Introduction

As the size of commonly used scientific datasets is growing beyond tera and peta-scale bytes, the corresponding algorithmic complexity of the application programs used for their analysis is also increasing very fast. This has made it very difficult for a typical scientist to use local and ordinary resources to perform deep, systematic analysis of these very large datasets. To ameliorate this situation, many scientific domains have established *science data centers* (service stations) that provide easy and efficient access to both the data as well as related application programs needed for the analysis [6]. Furthermore, most of these science data centers also provide personal workspace to each scientist for storing the results of their analysis. Though this paradigm shift frees the end-user scientist from the burden of managing the data and applications; however, it enhances the complexity for service providers by many folds as the size of data and number of users increases. In order to cope with this situation many institutional, national and international distributed and grid computing infrastructures have been established e.g the *Biomedical Informatics Research Network* (BIRN), *National Grid Service* (NGS) in UK, *TeraGrid* in US, *Enabling Grids for E-sciencE* (EGEE) in Europe and across the world. These high-end infrastructures provide most of the computing, storage, data and software resources for a wide variety of scientific disciplines. These resources are available to both categories of scientist i.e *end-users*, who use the already deployed applications and data to perform certain analysis and *application developers*, (who also become *service providers* at the end of their application development phase in order to make their newly developed application usable by the respective community), use the computing resources to develop and test their novel applications requiring the support of that infrastructure.

From the perspective of a scientific computing engineer, the provision of the above mentioned infrastructures facilitates the development of scalable applications that could perform large scale data and compute intensive in-silico calculations such as the exhaustive all-against-all comparison, analysis and visualization of all or a set of non-redundant protein 3D structures currently available in the *Protein Data Bank* (PDB) [2]. The results of such experimentation could help in core structural proteomics activities such as automated classification of protein universe, protein structure prediction, understanding of protein's function and design

and discovery of new therapeutics [12]. Though these all activities demand for high quality of the results of structure comparison; however, each structure comparison algorithm provides different results even for the same pair of structures. This situation leads us towards the concept of protein structure *MULTI-comparison*, which uses an ensemble of most often used methods (as listed in Table 1) to provide more accurate and reliable results based on the consensus of individual results from each method [1]. The criteria beyond the selection of these particular methods was to conjugate a set of methods with maximum diversity in terms of the computational techniques (algorithms) they employ, representative features of structures they use as input, robustness of the di(similarity) measures/metrics they introduce and the popularity of their use in the community. The consensus similarity thus obtained performs at least as good as one of the community's gold standards for assessing similarities in the *Critical Assessment of Techniques for Protein Structure Prediction* (CASP), namely *GDT_TS*, but more importantly in many cases it provides better results [1].

It is due to the robustness of the measures and metrics that the consensus based results obtained from this set of methods agrees well with other methods such as *Global Distance Test Total Score* (GDT_TS) used for the assessment of similarities at *Critical Assessment of Structure Prediction* (CASP) experiments (intrested readers may refer to [1] for a case study with detailed comparison of results).

In this paper we report on our findings on the storage, management and analysis of data resulting from our newly developed distributed application for all-against-all multi-comparison of large scale protein structure datasets. Given the large volume and complexity of data involved in the experimentation the selection and use of an appropriate suite of technologies becomes quite important. We report on our experiences of using core database technologies such as *Hierarchical Data Format* (HDF) (HDF5) and *Relational Database Management System* (RDBMS) (Oracle/SQL) on the UK *National Grid Service* (NGS) infrastructure.

The organization of the rest of this paper is as follows. A brief survey of the related work is presented in section 2. This is followed by the description of the data involved in the process of multi-comparison and the techniques used for the estimation of missing/invalid values etc in section 3. Section 4, introduces the main technologies used for the data storage, management and analysis and compares their features with other related technologies such as traditional RDBMS and XML. Details of the experimental design and implementation are described in section 5, while section 6 provides the results and discussions.

2. Motivations

Due to the exponential growth in the number of protein structures in PDB, most of the online servers for protein structure comparison have started to build a database of pre-computed results for their applications and use it as a quickest way to respond to user queries which otherwise would have taken many hours or even days of processing time. Some examples of these servers which use pre-computed databases include [14, 9, 3, 16]. These servers provide both options to users i.e the user could either select a query structure that is already available in the PDB; in this case the list of similar structures will be displayed within the time of a click using the information from pre-computed database, or the user could select a novel query structure to be compared to all structures in the PDB. In the later case the existence of the pre-computed knowledge base is used to prune the search space once any strong match is found for the query structure by limiting the comparison to only the neighbors of the strong match whose list is maintained in the knowledge base. The design and implementation of such knowledge bases seems to be quite easy and straightforward as they only deal with a single structure comparison method. However, as described in the subsequent sections of this paper there are so many issues and complexities in the case of multi-comparison, and to the best of our knowledge, there exists no other solution in the literature yet.

3. Protein Structure (Multi) Similarity Data

When applying a similarity comparison algorithm to a set of proteins comparing its members in an all-against-all fashion, one obtains a matrix that describes the similarity/dissimilarity of each pair of proteins. This matrix is called a *similarity matrix* (SM) when the similarity measure is a type of scoring function (such as *number of alignments*) that starts with zero (no similarity at all) and that gives higher values for better agreement between any two proteins considered. As different algorithms apply different scoring schemes that do not necessarily correlate with any physical property (e.g. protein size), it is not possible to give an upper bound for SM values. On the other hand, there are algorithms that do not produce a scoring function but express similarity in terms of distances. Here, a value of zero means that two proteins are identical, whereas higher values indicate a higher degree of dissimilarity. The resulting matrix for an all-against-all comparison is called a *dissimilarity matrix* or *distance matrix* (DM). The *root mean square deviation* (RMSD) value is a typical example of a (poor) distance measure for protein structure comparison. More recent and more sophisticated protein structure comparison methods rely on a variety of similarity/distance definitions. Some but not all of them are:

NA: Number of Alignments (DaliLite, CE, MaxCMO, TM-Align, FAST): indicates how many elements/residues of a query protein structure are aligned to the elements/residues of the target structure. A higher value indicates more similarity.

Z-score (DaliLite, CE): indicates the statistical significance of the similarity result with respect to the random comparison of structures. Its value should be 3.5 or higher for two proteins to be similar i.e. having same fold.

RMSD: Root Mean Square Distance (DaliLite, CE, TM-Align, FAST): indicates the divergence between two protein structures. Its value should be less than 5Å for two structures to be similar i.e. belonging to same family.

TM-score (TM-Align): this score is based on the improvement of RMSD i.e. to be independent of protein length. Its value lies in between 1 (identical) and 0 (no similarity) with 0.5 being used as a threshold to identify fold level relationships.

SN: Normalized Score (FAST): indicates the significance of an alignment. Its value of 1.5 or greater indicates significant structural similarity.

USM distance (USM): this measure is based on the calculation of Kolmogorov complexity between two structures. The lowest the distance, the more similar the structures and vice-versa.

Usually the similarity/dissimilarity values are used for further processing steps, e.g. clustering the distance matrix in order to obtain family relationships between different proteins. Such methods require a complete and valid matrix as input and usually do not cope with missing or ill-defined data. Similarity comparison algorithms usually produce numerical values to describe the degree of similarity between two protein structures, but some methods just return a textual message that no significant similarity could be found when the similarity score falls under an internal threshold [9]. Additionally, calculation errors can occur due to invalid input structures or random flukes, or output that cannot be read or written correctly. In a cluster or GRID environment, data might not get transmitted correctly from the computation node to the storage element, or some (user-defined) preset timeout occurred that does not allow to wait longer for the arrival of the remaining results. Summing up, a similarity or distance matrix can not only contain numerical values, but also some other values such as special codes indicating for example no significant similarity (N) or errors (E), and missing data (#). When dealing with big datasets with just a few missing data points, it is often more convenient to account for them by estimation instead of submitting them for recalculation. Additionally, we must correct for invalid self-similarity *SS* values that may occur if heuristic similarity methods are used (e.g.MaxCMO [13]) that do not guarantee to find the optimal result. Hence, the SS value of a protein can be worse than any of its non-self-similarity (NSS) values obtained from comparing the protein with any other proteins in the dataset. The techniques used for the estimation of missing/invalid SS and NSS values are briefly described in the following sub sections.

3.1. Estimation of Self-Similarity Values

Estimation of SS values could exploit the fact that these values should always be better than any other NSS values for the protein considered. However, there is a slight variation of the special requirement when dealing with normalized data: we know that the best value (lower bound) for any *distance measure* must be zero, but we cannot give an upper bound for *similarity measures* (scoring functions) as it depends on the length of proteins. In the latter case, we therefore estimate any missing or invalid SS value by the best (highest) NSS value of any comparison with this protein. If no value can be found at all due to irrecoverable problems during the calculation, we adopt a worst case approach and take the worst (smallest) value of the entire similarity matrix. This makes sure that this value does not interfere with any better (higher) values in any further analysis step, but is not as drastic as setting the SS value to zero as it would make the standardization step impossible (resulting in a division by zero).

3.2. Estimation of Non-Self-Similarity Values

When estimating non-self-similarity values, we first try to exploit the symmetric nature of any similarity/dissimilarity matrix. Usually, the comparison of protein P_i with protein P_j gives the same results as when comparing P_i with P_j. So we could substitute S_{ij} by S_{ji} as they will be more similar than any other estimation can produce. However, the symmetric nature of the similarity/dissimilarity matrix might already have been exploited during the calculation procedure saving time by only calculating the NSS values in one triangle of the matrix (and the SS values). In this case, if S_{ij} contains an error and its "backup" S_{ji} was not calculated, we need to estimate S_{ij} with the worst value that can be found in the cross-section S_{im}/S_{jn} of the matrix.

4. Overview of the Core Data Storage, Management and Analysis Technologies

Most of the scientific disciplines use much simple, convenient and self-describing data models such as *Hierarchical Data Format* (HDF, HDF4 or HDF5) [7], *Flexible Image Transport System* FITS and NetCDF [17]. This is because most of the scientific data goes beyond the limits of traditional relational databases and XML documents in terms of its size, complexity and heterogeneity and is in the form of numeric arrays and images requiring more programming support (in terms of libraries, APIs and tools) for statistical analysis, manipulation, processing and visualization. All these requirements are easily accommodated by scientific data models along with additional benefits of being open source, supporting multi-object data format (i.e each data model could support different primitive data types as well as multi-dimensional arrays, tables and groups etc), data definition (in terms of metadata), efficient access (in terms of random/parallel/partial/fast IO), optimal storage (in terms of compressed and binary file format), and ease of sharing by means of their platform independent nature.

Though there exist many scientific data models but HDF5 is being commonly used across a wide variety of scientific domains, projects and applications (hdfgroup.org/HDF5/users5.html). HDF5 refers to a suite of open source technologies including data model, APIs, libraries, utilities and tools used for the storage, management and analysis of complex and large scale scientific datasets. Originally created by *National Center for Supercomputing Applications* (NCSA), it is now maintained by *The HDF Group* [7]. The ease of dealing with HDF5 lies in the multi-object file format that allows data of different type, nature and size to be stored in the same file with suitable data structure ranging from simple variables, multi-dimensional arrays, tables (the table object of HDF5 is also in the form of multidimensional array and hence provides much more quicker access as compared to the rows of SQL database), images to groups and pallets (it is important to note that unlike HDF5, which provides support for all data formats including images, arrays, tables etc, FITS and NetCDF only support images and array-oriented data types respectively). Each of these data items could be additionally described with related metadata, allowing further ease in terms of data discovery. Furthermore, chunking and compression along with binary file format of HDF5 provides high performance access and occupies less space and takes less time to be transmitted over the network. In order to provide Interoperability with XML (to leverage its additional benefits of working with web/grid services) HDF5 also provides some tools to translate the data from HDF5 format to XML and vice-versa.

In the following sections we present the design and implementation of a grid-enabled HDF5-based architecture for the storage, management and analysis of protein structure multi-comparison results and compare its performance with a traditional relational database using Oracle/SQL.

5. Design and Implementation of the Proposed Architecture

Figure 1 shows the architectural design for the storage, management and analysis of (multi) similarity for large scale protein structure datasets. The overall computational load of pair-wise similarity computations is uniformly distributed through a grid-enabled *Message Passing Interface* (MPI) [10] based algorithm. The use of grid-enabled MPI based model makes it easy to exchange global information needed for the computation of missing/invalid values; thereby, facilitating the process of standardization and normalization to be performed on-the-fly in a distributed environment.

The MPI based job packages are submitted for execution on a cross-site grid infrastructure provided by *National Grid Service* (NGS), UK [4]. A brief description of the NGS tools and services used for our experimentation is given bellow:

GSI-SSHTerm: An applet as well as an standalone application that provides seamless access to NGS resources i.e it offers web/command-line interface to securely connect to NGS sites from any machine having the UK e_Science digital certificate installed.

GT4: *Globus Toolkit* version 4 (GT4), a grid middleware that on the one hand enables the service providers to bind grid applications together for ease of access and management; and on the other hand it provides many services for job submission, monitoring and management. We used Globus to submit RSL scripts and perform monitoring of jobs.

MPIg: *Grid-enabled Message Passing Interface* (MPIg) is the latest version of MPICH-G2 that works with web services based version of Globus and enables an MPI-based application to spawn jobs across multiple clusters in a WAN environment. We used this library to distribute our jobs across two NGS clusters as illustrated in Figure 1.

SRB: *Storage Resource Broker* is a *Data Grid* middleware developed by *Data Intensive Cyber Environments* research group at the *San Diego Supercomputing Centre* (SDC). It enables users to access files using logical names or attributes from any location in a LAN or WAN environment without actually worrying about the physical location of each file. It achieves

this through the use of a metadata catalog (MCat), that holds information about the physical location of each file, its logical name and and attributes. As we used MPIg to run our jobs across different clusters of NGS, which have different file system hierarchies, we had to use this facility to provide uniform naming of files.

HARC: Highly-Available Robust Co-scheduler is a framework that provides scheduler based resource reservation. We used this module to make advance reservation of the number of CPUs we wanted to be used solely for our application.

The (multi) similarity values produced by parallel jobs on each processor were stored both in HDF5 and Oracle using the following schema:

```
// HDF5 DDL SCHEMA

HDF5 "Protein_Multiverse.h5" {
GROUP "/" {
   DATASET "COMPARISON_RESULTS" {
      DATATYPE  H5T_COMPOUND {
         H5T_STRING {
            STRSIZE 32;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         } "Structure1";
         H5T_STRING {
            STRSIZE 32;
            STRPAD H5T_STR_NULLTERM;
            CSET H5T_CSET_ASCII;
            CTYPE H5T_C_S1;
         } "Structure2";
         H5T_IEEE_F32LE "F_sn";
         H5T_IEEE_F32LE "F_zscore";
         H5T_IEEE_F32LE "F_rmsd";
         H5T_IEEE_F32LE "C_zscore";
         H5T_STD_I32LE "C_align";
         H5T_IEEE_F32LE "C_rmsd";
         H5T_IEEE_F32LE "D_zscore";
         H5T_STD_I32LE "D_align";
         H5T_IEEE_F32LE "D_rmsd";
         H5T_STD_I32LE "T_align";
         H5T_IEEE_F32LE "T_tmscore";
         H5T_IEEE_F32LE "T_rmsd";
         H5T_STD_I32LE "M_align";
         H5T_STD_I32LE "M_overlap";
         H5T_IEEE_F32LE "U_usmd";
      }


//ORACLE TABLE DESCRIPTION
```

```
Name                  Type
------------------  ------------
PAIR_ID             NUMBER(8)
STR1                VARCHAR2(32)
STR2                VARCHAR2(32)
F_SN                NUMBER(5,2)
F_ZSCORE            NUMBER(5,2)
F_RMSD              NUMBER(5,2)
C_ZSCORE            NUMBER(5,2)
C_ALIGN             NUMBER(5)
C_RMSD              NUMBER(5,2)
D_ZSCORE            NUMBER(5,2)
D_ALIGN             NUMBER(5)
D_RMSD              NUMBER(5,2)
T_ALIGN             NUMBER(5)
T_TMSCORE           NUMBER(5,2)
T_RMSD              NUMBER(5,2)
M_ALIGN             NUMBER(5)
M_OVERLAP           NUMBER(5)
U_UMSD              NUMBER(5,2)
```

The integration of MPI-IO with parallel HDF5 also makes the process of result synthesis to be more efficient as each process writes concurrently to the same file. The collated results could then easily be queried through HDF5 based technologies such as *HDFView* (a java based application used to visualize and analyze the contents of HDF5 files as shown in Figure 2) and HDF-FastQuery. HDF5-FastQuery is an API that integrates *Fast Bitmap Indexing* technology with HDF5. This integration enables extremely large HDF5 datasets to be analyzed interactively. According to [5], both approximate and fixed queries performed through HDF5-FastQuery become as much faster as 10 and 2 times respectively as compared to using simple HDF5 queries.

In the following section we discuss the results of our experiments with the HDF5-based architecture and provide its comparison with Oracle/SQL.

6. Experimental Results and Discussions

Several experiments were designed to compare a varying number of protein structures with multiple methods. The datasets and methods used in our experiments are described in Table 2 and Table 1 respectively. The main objective beyond the selection of these datasets was to find out the effect of growing number of proteins on the storage capacity and query execution time using both HDF5 and Oracle/SQL database technologies. The results of such experimentation are presented in Table 3. Though the storage values for HDF5 represent the overall overhead of file structure and related metadata for the results of each dataset; however, the storage values given for Oracle only include the

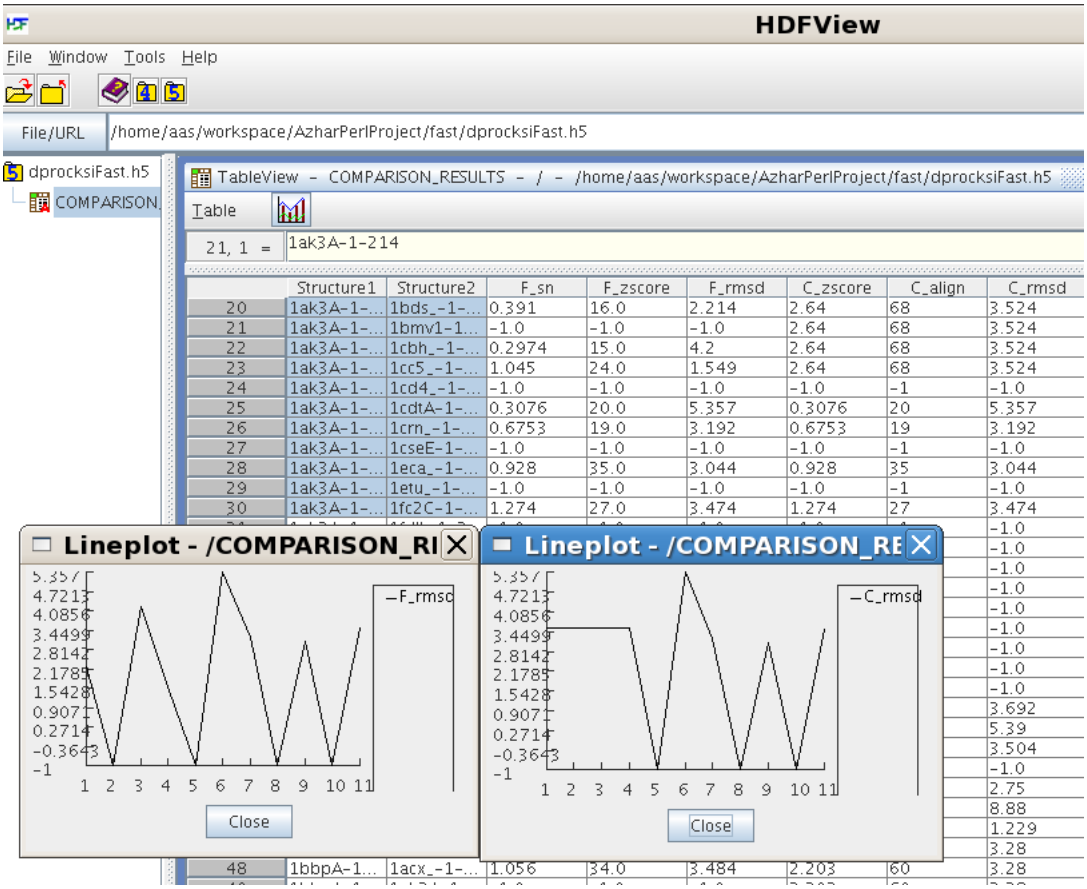

Figure 2. Visualization of the (multi) similarity data with *HDFView* (hdfgroup.org/hdf-java-html/hdfview/) tool that works as an standalone application as well as a *plug-in* in the web browsers. This tool also provides graph utilities that could be used to visualize the values of any selected measures/metrics for a desired set of proteins.

size of respective tables for the results of each dataset and the additional storage overhead needed for the base installation (an empty new oracle database, which takes about 1-2 GB) is not included. Nevertheless, for these datasets, the storage/query overhead for Oracle is quite significant as compared to HDF5. Furthermore, as we double the number of protein structures the storage requirement for both HDF5 and Oracle grow approximately with the same rate i.e by a multiple of 4. Whereas, in terms of query execution, though HDF5 maintains approximately an stable state irrespective of the size of dataset, the time for SQL query to Oracle database increases drastically with larger datasets.

The query execution time in each case (HDF5 and Oracle) is an average time taken by a simple query to retrieve the contents of a particular field from the last record (i.e the record with highest index value in the given table). The average was taken on the basis of 15 repeated queries and the standard deviation in each case is not much significant (Table 4).

Method	Measures/Metrics
MaxCMO [13]	NA, NO
DaliLite [9]	NA, Z-score, RMSD
CE [15]	NA, Z-score, RMSD
TM-align [18]	NA, TM-score, RMSD
USM [11]	USM-distance
FAST [19]	NA, SN, RMSD

Table 1. Methods used to detect (Multi) Similarity among Protein Structure Datasets. *Note*: For full measure/metric names please refer to section 3.

Dataset	HDF5 Storage (MB)	Oracle Storage (MB)	HDF5 Avg. Query (sec)	Oracle (SQL) Avg. Query (sec)
pdd_select30_250	0.26	6.84	0.00070	0.01467
pdb_select30_500	1.00	26.37	0.00071	0.04300
pdb_select30_1000	4.01	109.37	0.00072	1.13560
pdb_select30_2000	16.04	421.87	0.00072	10.45620
pdb_select30_4000	67.23	1687.50	0.00075	32.89250

Table 3. HDF5 Vs Oracle Storage/Query benchmarking with different datasets. The total number of records (rows) in each case is equal to the number of pairwise comparisons for each dataset *(Table 2)* and the total number of fields is equal to the number of measures/metrics for all the methods (i.e 15,*Table 1*) plus 3 additional fields of which 2 store protein IDs and the remaining 1 serves as the record identifier.

Dataset	# of Chains	# of comparisons
PDB_SELECT30_250	250	62,500
PDB_SELECT30_500	500	250,000
PDB_SELECT30_1000	1000	1,000,000
PDB_SELECT30_2000	2000	4,000,000
PDB_SELECT30_4000	4000	16,000,000

Table 2. Datasets used in the experiments. *PDB_SELECT30* is a representative dataset (subset of PDB) consisting of all non-redundant protein structures having chain length greater than 50 and sequence identity less than 30%. This dataset has been prepared by using the *PDB_SELECT* algorithm designed by Uwe Hobohm and Chris Sander [8]. The overall dataset consists of 7183 structures and was partitioned into selected groups as listed in the first column of this table. Hash sign (#) represents the word *'Number of'*

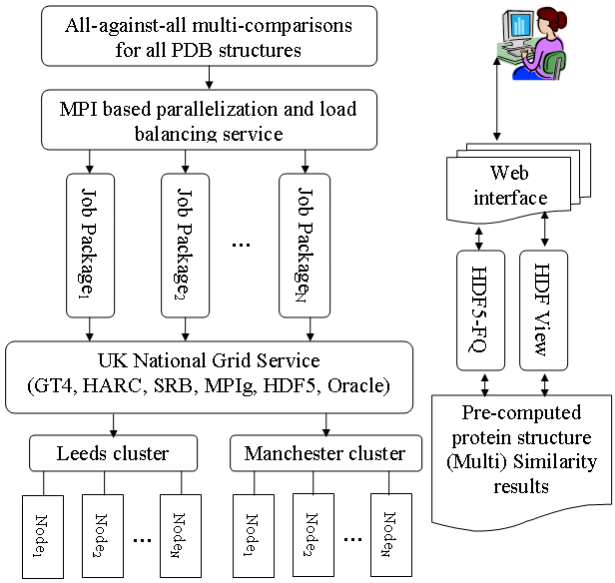

Figure 1. Architectural Design of the Storage, Management and Analysis for Protein Structure Multi-comparison Data in the Grid Environment

Dataset	HDF5 Query (STDEV)	Oracle Query (STDEV)
pdd_select30_250	3.23E-05	0.0019
pdb_select30_500	2.45E-05	0.0057
pdb_select30_1000	2.30E-05	0.464
pdb_select30_2000	3.33E-05	1.332
pdb_select30_4000	4.25E-05	3.929

Table 4. Standard Deviations Calculated on 15 Query Times for HDF5 and Oracle (SQL)

6.1. Concluding Remarks and Future Directions

The HDF5 has been used as a scientific data format for the storage, management and analysis of (multi) similarity of large scale protein structure datasets in a distributed/grid environment provided by National Grid Service (NGS), UK. Mathematical techniques used for the estimation of missing/invalid values have been described. The storage/query overhead of HDF5 was compared with that of Oracle/SQL. The results show significant performance benefit of HDF5 over Oracle/SQL. Based on this initial benchmarking, we plan to compute the (multi) similarities of all available PDB structures; thereby, creating an efficient sci-

entific knowledgebase of pre-computed results. The development of such knowledge base is aimed at providing an easy to use interface, so the biologist can perform scientic queries using a vast variety of criteria and options leading to far better and reliable understanding of the protein structural universe.

7. Acknowledgments

All the authors would like to acknowledge the use of the UK National Grid Service in carrying out this work; Azhar A. Shah acknowledges The University of Sindh for providing scholarship SU/PLAN/F.SCH/794.

References

[1] D. Barthel, J. Hirst, J. Blazewicz, E. K. Burke, and N. Krasnogor. The ProCKSI server: a decision support system for protein (structure) comparison, knowledge, similarity and information. *BMC Bioinformatics*, 8:416, 2007.

[2] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and P. Bourne. The protein data bank. *Nucleic Acids Res*, 28:235–242, 2000.

[3] G. Chittibabu, R. P. Lipika, and N. S. Ilya. Dmaps: a database of multiple alignments for protein structures. *Nucleic Acids Res.*, 34:D273D276, 2006.

[4] N. Geddes. The national grid service of the uk. In *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, 2006.

[5] L. Gosink, J. Shalf, K. Stockinger, K. Wu, and W. Bethel. Hdf5-fastquery: Accelerating complex queries on hdf datasets using fast bitmap indices. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*. IEEE Computer Society Press, 2006.

[6] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber. Scientific data management in the coming decade. *SIGMOD Rec.*, 34(4):34–41, 2005.

[7] Hierarchical data format, hdfgroup.org/hdf5/.

[8] U. Hobohm and C. Sander. Enlarged representative set of protein. *Protein Science*, 3:522–524, 1994.

[9] L. Holm, S. Kaariainen, P. Rosenstrom, and A. Schenkel. Searching protein structure databases with dalilite v.3. *Bioinformatics*, 24:2780–2781, 2008.

[10] N. T. Karonis, B. Toonen, and I. Foster. Mpich-g2: a grid-enabled implementation of the message passing interface. *Journal of Parallel and Distributed Computing*, 63(5):551–563, 2003.

[11] N. Krasnogor and D. A. Pelta. Measuring the similarity of protein structures by means of the universal similarity metric. *Bioinformatics*, 20:1015–1021, 2004.

[12] T.-S. Mayuko, T. Daisuke, C. Chieko, T. Hirokazu, and U. Hideaki. Protein structure prediction in structure based drug design. *Current Medicinal Chemistry*, 11:551–558, 2004.

[13] D. A. Pelta, J. R. Gonzalez, and M. V. M. A simple and fast heuristic for protein structure comparison. *BMC Bioinformatics*, 9:161, 2008.

[14] Y. Sato, A. Nakaya, K. Shiraishi, S. Kawashima, S. Goto, and M. Kanehisa. Ssdb: Sequence similarity database in kegg. *Genome Informatics*, 12:230–231, 2001.

[15] I. Shindyalov and P. Bourne. Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Eng*, 11:739–747, 1998.

[16] I. Shindyalov and P. Bourne. a database and tool for 3-d protein structure comparison and alignment. *Nucleic Acids Res.*, 29:228–229, 2001.

[17] D. Wells, E. Greisen, and R. Harten. Fits: A flexible image transport system. *Astronomy and Astrophysics Supplement Series*, 44:363–370, 1981.

[18] Y. Zhang and J. Skolnick. Tm-align: A protein structure alignment algorithm based on tm-score. *Nucleic Acids Res*, 33:2302–2309, 2005.

[19] J. Zhu and Z. Weng. Fast: A novel protein structure alignment algorithm. *Proteins Struct Funct Bioinf*, 58:618–627, 2005.