Studies on Distributed Approaches for Large Scale Multi-Criteria Protein Structure Comparison and Analysis

Azhar Ali Shah, MPhil

Thesis submitted to The University of Nottingham
for the degree of Doctor of Philosophy

September 2010

Abstract

Protein Structure Comparison (PSC) is at the core of many important structural biology problems. PSC is used to infer the evolutionary history of distantly related proteins; it can also help in the identification of the biological function of a new protein by comparing it with other proteins whose function has already been annotated; PSC is also a key step in protein structure prediction, because one needs to reliably and efficiently compare tens or hundreds of thousands of decoys (predicted structures) in evaluation of '*native-like*' candidates (e.g. *Critical Assessment of Techniques for Protein Structure Prediction* (CASP) experiment). Each of these applications, as well as many others where molecular comparison plays an important role, requires a different notion of similarity, which naturally lead to the Multi-Criteria Protein Structure Comparison (*MC-PSC*) problem. ProCKSI (www.procksi.org), was the first publicly available server to provide algorithmic solutions for the MC-PSC problem by means of an enhanced structural comparison that relies on the principled application of information fusion to similarity assessments derived from multiple comparison methods (e.g. USM, FAST, MaxCMO, DaliLite, CE and TMAlign). Current MC-PSC works well for moderately sized data sets and it is time consuming as it provides public service to multiple users. Many of the structural bioinformatics applications mentioned above would benefit from the ability to perform, for a dedicated user, thousands or tens of thousands of comparisons through multiple methods in real-time, a capacity beyond our current technology.

This research is aimed at the investigation of Grid-styled distributed computing strategies for the solution of the enormous computational challenge inherent in MC-PSC. To this aim a novel distributed algorithm has been designed, implemented and evaluated with different load balancing strategies and selection and configuration of a variety of software tools, services and technologies on different levels of infrastructures ranging from local testbeds to production level eScience infrastructures such as the *National Grid Service* (NGS). Empirical results of different experiments reporting on the scalability, speedup and efficiency of the overall system are presented and discussed along with the software engineering aspects behind the implementation of a distributed solution to the MC-PSC problem based on a local computer cluster as well as with a GRID implementation. The results lead us to conclude that the combination of better and faster parallel and distributed algorithms with more similarity comparison methods provides an unprecedented advance on protein structure comparison and analysis technology. These advances might facilitate both directed and fortuitous discovery of protein similarities, families, super-families, domains, etc, and also help pave the way to faster and better protein function inference, annotation and protein structure prediction and assessment thus empowering the structural biologist to do a science that he/she would not have done otherwise.

Acknowledgements

Doctoral studies being a journey of a discovery wouldn't be possible without the *enabling technologies* in the form of help and support of so many individuals. The first and foremost support comes from one's guide/supervisor in the form of expert advise, interest, enthusiasm and encouragement. It is a matter of fact that not all supervisors provide all this support and hence history shows that as a brilliant student as Einstein had to change his supervisor [1]. In this regard I consider myself fortunate enough to have Professor Natalio Krasnogor as my supervisor. It always leaves me wondering how Prof. Krasnogor has managed to shape his personality to have all aspects of an ideal and perfect supervisor besides being a very kind, caring, friendly and social person! I know that, a proper answer to this question might require me to write another thesis, so I would rather leave it mysterious and would like to offer my utmost gratitude and recognition for his excellent mentorship in the form of regular constructive feedback on my verbal and written ideas and drafts while sharing and discussing various issues, outcomes and future plans of my research from the very beginning to the completion of this thesis. Besides this, I am also grateful to him for acquainting me with a great team/network of co-workers and providing all the resources needed for the experimental setup as well as financial support to attend several research and professional trainings and to present my research papers at various conferences across the globe. I know that this is not the explicit listing of what I should be thankful to Prof. Krasnogor, indeed there are many and many (let me say countless) other things which I couldn't mention here but am really debited to be grateful to him. Indeed it has been an honor for me to have Prof. Krasnogor as my supervisor.

While writing several co-authored research papers I had the opportunity to leverage the expertise of many well known academicians and researchers who have their association/collaboration with ASAP group including Professor Jacek Blazewicz and Piotr Lukasiak (Poznan University, Poland), Drs. Daniel Barthel (ex-ASAPEE, now team leader in Rexroth, part of the Bosch group, Germany) and Gianluigi Folino (CNR-ICAR, University of Calabria, Italy). I am thankful for their critical comments, suggestions and help for the improvement of the publications which got accepted at targeted journals and conferences.

Had it not been the much needed technical support from *Technical Service Group* (TSG), I would had lost several days and even months to deal with all the technical problems that arose from time to time while setting-up my local Linux cluster installed with different software packages, tools and services needed for different experiments. In this regard a big thank you to TSG especially Nick Reynolds, William Armitage and Viktor Huddleston. In addition to TSG, I would also like to thank Drs. Pawel Widera, German Terrazas Angulo, James Smaldon and Amr Soghier for their help with Linux, Latex and other software related issues. I would also like to acknowledge the use of the CNR-ICAR cluster in Italy and the UK National Grid Service (NGS) in carrying out this work.

I am also thankful to all the academic and admin staff members at School of Computer Science in general and members and admin team of the ASAP group in particular for their cooperation in various activities during the complete period of my research. A similar token of gratitude goes to all the members of academic staff and admin team at Institute of Information and Communication Technology (ICT), University of Sindh (where I hold an academic position and was granted study leave and scholarship for PhD) for their help and support in various administrative aspects. Special thanks to Drs. Abdul Wahab Ansari, Imdad Ali Ismaili, Khalil Khombati and

[1] It is said that, "In 1901 Einstein transitioned his thesis supervision from "H.F. Weber" to "Alfred Kleiner"; and changed his dissertation topic from thermoelectric to molecular kinetics".

Vice-Chancellor, Mr. Mazhar-ul Haq Siddiqui. I would also like to acknowledge The University of Sindh for funding my study through scholarship (SU/PLAN/F.SCH/794). Many thanks to all my friends who in some or other way provided the social fabric needed to a foreigner living far away from his family. All the friends studying at Nottingham as well as so many other Universities in England and other countries (connect through different egroups such as ((sindhischolars,scholars_uk,hec_overseas)@yahoogroups.com) had been of great help.

Finally, thanks to those for whom thanks is a too small word to say and I would rather like to dedicate this complete thesis which is the outcome of my endeavors as well as their patience, prayers and support i.e my: late father, mother, in-laws, wife, kids, siblings (specially my elder brother Noor Muhammad Shah for his utmost love, guidance, encouragement and support) and relatives. I think, being the very first person from our family to have a PhD, I have earned something for us to be proud of!

PS: 5^{th} July 2010: at the midnight I was informed of a great loss in my life – my mother left this earth – weeping and crying I caught a flight for back home but was too late to see her – a bad side of doing PhD abroad! May God rest her soul in peace!

Azhar Ali Shah
Nottingham, England,
September 2010.

For Asfar, Maryam, Maidah, Shazia, Ada N.M Shah and my (late) parents.

Contents

List of Figures

List of Tables

Glossary of Acronyms

BIRN	Biomedical Informatics Research Network 25
BRIDGES	Biomedical Research Informatics Delivered by Grid Enabled Services . 25
DAG	Direct Acyclic Graph 47
OGSA	Open Grid Service Architecture 37
OGSA-DAI	Open Grid Service Architecture-Data Access and Integrator 24
DMS	Data Management Service 24
DQP	Distributed Query Processor 24
EBI	European Bioinformatics Institute 27
EDG	European Data Grid 38
EGEE	Enabling Grid for EscienceE vii
EMBOSS	European Molecular Biology Open Software Suite 27
EMBRACE	European Model for Bioinformatics Research and Community Education 60
GEBAF	Grid Enabled Bioinformatics Application Framework 53
GLAD	Grid Life Science Application Developer 47
GSI	Grid Security Infrastructure 60
GUI	Graphical User Interface 38
JDL	Job Description Language 38
LCG2	LHC Computing Grid 35
LRMS	Local Resource Management System 15
LSF	Load Sharing Facility 26
LSID	Life Science Identifiers 31
MIAME	Minimum Information About a Microarray Experiment 33
NCBI	National Center for Biomedical Informatics 27
NGS	National Grid Service 16
OWL	Web Ontology Language 30
PBS	Portable Batch System 15

CHAPTER 1

INTRODUCTION

This thesis presents the outcome of the exploratory and investigative *Studies on Distributed Approaches for Large Scale Multi-Criteria Protein Structure Comparison and Analysis* in ten self-contained chapters. This chapter being the first one sets the stage by introducing the research topic and explaining why this topic was chosen for study. Besides introducing the research topic, this chapter also provides a succinct overview of the research objectives, methodology and the structure of the thesis.

1.1 Introduction

> "The organic substance which is present in all constituents of the animal body, also as we shall soon see, in the plant kingdom, could be named ’protein’ from a Greek word ’proteios’ meaning ’of primary importance’" [1].

In 1839 the above words first appeared in an article authored by *Gerardus Johannes Mulder* (1802-1880), a Dutch organic chemist [1]. Mulder wrote these words after correspondence with *Jons Jakob Berzelius* (1779-1848), a Swedish chemist who originally coined the term *Protein* based on Mulder’s observations that *’all proteins have same empirical formula and are composed of a single type of molecule’*. Since this time proteins have remained among the most-actively studied molecules in biochemistry and the understanding of their structure and function has remained an es-

sential question. It was only in 1955 that a British biochemist *Frederick Sanger* (1918) introduced an experimental method to identify the sequence (*primary structure*) of a protein named *insulin'* [2]. This achievement entitled Sanger to win his first Noble Prize in 1958. This discovery was followed by another breakthrough by *Sir John Cowdery Kendrew* (1917-1997), an English biochemist and crystallographer and *Max Perutz* (1914-2002), an Austrian-British molecular biologist who discovered the three-dimensional structure (*Tertiary structure*) of two proteins named *'myoglobin'* [3] and *'hemoglobin'* [4] respectively and were co-awarded the 1962 Noble Prize in chemistry for these achievements. It was exactly at this time that *Christian Anfinsen* (1916-1995), an American biochemist, had developed his theory of protein folding, which explains the process by which a protein sequence coils (folds) into a more stable and unique three-dimensional structure (*native conformation*) [5–7]. Anfinsen's theory of protein folding provided the basis for protein structure prediction and also raised the need for structure-based comparison and analysis of proteins [8–10]. He was awarded the 1972 Noble Prize for this work. A few years later, in 1977, Sanger became successful in sequencing the complete genome of *Phage Phi X 174*, a virus (bacteriophage) that infects bacteria [11]. This provided the basis for *Human Genome Project* (HGP) [12] and hence entitled Sanger to share his second Noble Prize in chemistry in 1980 with two American biochemists named *Walter Gilbert* (1932) and *Paul Berg* (1926). The successful completion of HGP in 2003 lead to various world wide structural genomic and proteomic initiatives such as the *Structural Genomics Consortium* (SGC) [13], the *Protein Structure Initiative* (PSI) [14] , and the *Human Proteome Organization* (HUPO) [15] amongst others. These initiatives are targeted at lowering the cost and enhancing the efficiency for the experimental determination or computational prediction of novel protein 3D structures. As a consequence, there is an exponentially growing number of protein se-

quences and 3D structures being deposited in publicly available databases such as the *Universal Protein Resource* (UniProt) [16, 17] and *Protein Data Bank* (PDB) [18] respectively. In order for this data to be analyzed, understood and utilized properly, the need of automated, efficient and reliable software tools and services especially for determining the *structural similarities/dissimilarities* among all known structure becomes indispensable. The knowledge of structural (dis)similarities obtained from the comparison of protein structures is mainly used in core biomedical research activities including structure-based drug design [19], protein structure prediction/modeling [20–22], classification [23, 24], molecular docking algorithms [25] and other structural bioinformatics applications.

Several methods and tools have been developed to investigate the (dis)similarities among protein structures [26]. Not surprisingly, there is no agreement on how to optimally *define* what similarity/distance means as different definitions focus on different biological criterion such as sequence or structural relatedness, evolutionary relationships, chemical functions or biological roles etc and these are highly dependent on the task at hand. This observation calls for an explicit identification and understating of the various stages involved in the assessment of proteins' similarities. As illustrated in Figure 1.1, the first four stages, which have dominated the research in protein structure comparison so far, are: similarity conception, model building, mathematical definition and method implementation. Interestingly, the fifth stage, where one would seek to leverage the strength of a variety of methods by using appropriate consensus and ensemble mechanisms has barely been investigated. One such approach has recently been introduced by means of the *Protein (Structure) Comparison, Knowledge, Similarity and Information* (ProCKSI) web server [27]. Using a set of modern decision making techniques, ProCKSI automatically integrates the operation of a number

of the most popular comparison methods (as listed in Table 1.1) and provides an integrated consensus that can be used to obtain more reliable assessment of similarities for protein datasets. The consensus-based results obtained from ProCKSI take advantage of the *'collective wisdom'* of all the individual methods (i.e, the biases and variances of a given method are compensated by the other methods biases and variances) and minimizes the chances of falsely attributing similarity to (sets of) proteins. That is, false positives are more frequent at individual method level because usually most of globally different proteins still share some common substructures.

While trying to cope with the above mentioned *scientific challenge* (Figure 1.1) in the field of computational molecular biology, the performance of ProCKSI becomes the bottleneck for moderately large and very large instances of datasets (section 1.2 provides further details of this computational challenge). A thorough survey of the related literature reveals that in order to improve the overall performance, three routes are usually followed [28]: (a) the development of new algorithms or the redesign/modification of existing ones based on faster *heuristic* techniques [29, 30]; (b) development of special purpose ROM based hardware chips [31, 32]; and (c) the use of parallel and distributed computing. Routes (a) and (b) can only be applied in very specific cases as they require considerable in-depth knowledge of a problem or substantial economic resources respectively. The third alternative, the utilization of distributed and parallel computation is becoming a more ubiquitous approach as in some cases distributed/parallel solutions in one problem can be reused (with slight modifications) in other problems. Moreover, due to ongoing advances in processor and networking technologies, the scope of parallel computing is also extending from traditional supercomputers to massively parallel computers, clusters of workstations (COW) and even crossing the boundaries in the form of clusters of clusters i.e *grid computing* [33–35]. This paradigm shift in the

provision of parallel computing facilities afford scalability at very low cost. Furthermore, in terms of code maintenance and code portability, as compared to traditional super computers, distributed computing fares better and several successful applications to bio-sciences are discussed in [36–44].

Notwithstanding the above successes, grid computing has no magic applicability formula and many different distribution/parallelization solutions might exists for a given problem. Which of these strategies would be the best one to use will depend to a large extent not only on the specific problem structure but also on factors such as the choice/availability of particular hardware, software, interconnection types, security protocols and human resources etc. This thesis investigates several approaches that could provide the solution to the computational challenge for MC-PSC using various distributed approaches as described in chapter 4. The theoretical and empirical results of these investigations are discussed in the successive chapters as outlined in section 1.5.

1.2 Challenges of MC-PSC

As described above, *ProCKSI* is an online automated system that implements a protocol for MC-PSC. In particular, it allows the user to submit a set of protein structures and perform either *all-against-all* or *target-against-all* protein comparisons with the methods listed in Table 1.1. ProCKSI combines the results of pairwise comparisons delivered by the various available methods, normalizes them and presents a consensus form of the results through an intuitive web-based visual interface. Furthermore, it gathers information about the proteins being compared through hyper links to external sources of information e.g. *Information Hyperlinked Over Protein* (IHOP) [51], *Structural Classification of Proteins* (SCOP) [52], and *Class Architecture Topology and Hierarchy* (CATH) [53]. As demonstrated in [27], and previously suggested in [54] and [55], the ensemble

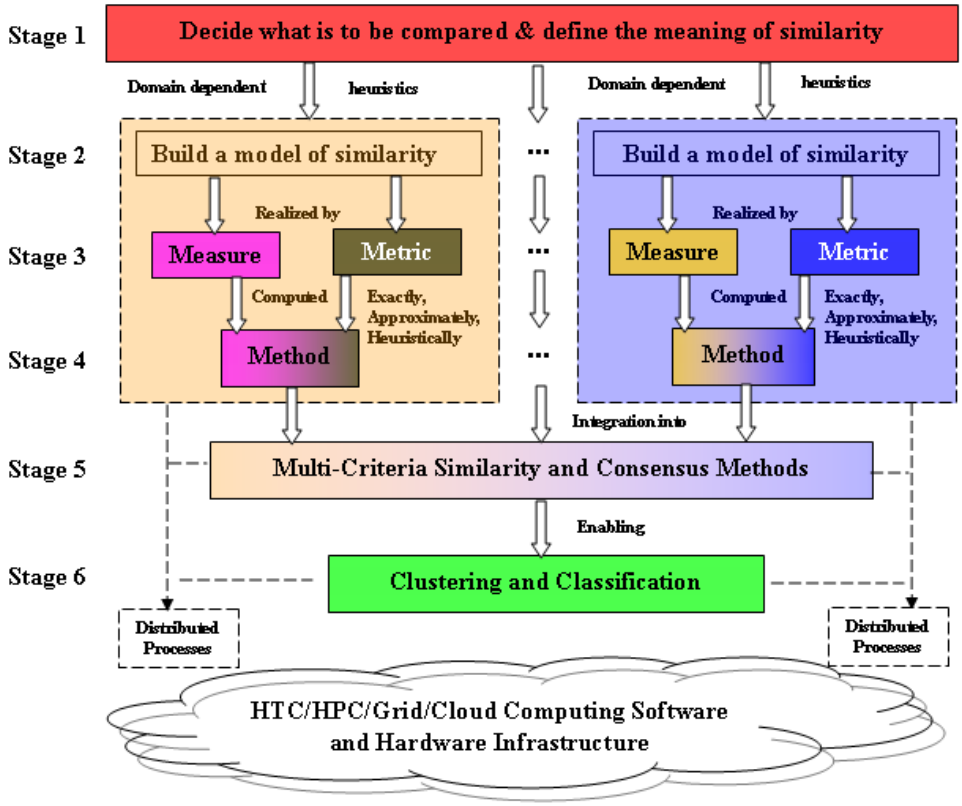


FIGURE 1.1: Stages in the derivation of a protein's classification: (1) Decide what "similarity" means, which is a declarative and problem-dependent step. (2) Heuristically build a model of similarity based on *1*. This new similarity/distance conception will have its own bias, variance and outliers. (3) Decide whether this idealized model will be instantiated as a distance/similarity measure or metric. (4) One or more algorithms are implemented in order to calculate *3*, which can be solved exactly and in polynomial time only in the simplest of cases. The more interesting similarity definitions, however, give rise to complex problems requiring heuristics/approximate algorithms for their solution. (5) Combining many different methods with different views of similarity produces a multi-competence pareto-front, from which a consensus picture might be derived. In turn, this allows the structural biologist to (6) cluster and classify proteins reliably. Furthermore, in order to provide most efficient (real-time) results based on the philosophy of (5), the need for the data and computation to be distributed and executed in a grid environment becomes indispensable.

and consensus based approach adopted by ProCKSI yields more reliable results of biological significance as compared to the results obtained with any single structure comparison method developed so far. However, the integration of multiple methods for protein structure comparison, on the one hand, coupled with a rapidly growing number of 3D structures in the Protein Data Bank (PDB), on the other hand, gives rise to a computational challenge that is far beyond the capabilities of a

TABLE 1.1: Building Blocks for Multi-Criteria Protein Structure Comparison. The name and references for each of the method is shown in column "Method", followed by the column "Notion of similarity" where the specific notions that each method uses to determine (di)similarity are mentioned. Columns "Computational techniques" and "Resulting measures/metrics" summarize how each similarity is computed and in what form it is returned. The last column gives an indication of relative computational requirements (time) for the different methods. **Key**: *AL* = Number of Alignments;*OL* = Number of Overlaps;*Z* = Z-Score; *TMS* = TM-align Score; *SN* = Normalized Score. * The average CPU time for a single pair of protein structures on a standard P4 (1.86 GHz, 2GB RAM) dual-core machine. Thus the total average execution time taken by all six methods (with a total of 15 different similarity measures/metrics) for the comparison of a single pair of protein structures is 8.54 secs plus some additional time for performing I/O.

Method	**Notion of similarity**	**Computational techniques**	**Resulting measures/metrics**	**Time* [sec]**
DaliLite [45]	intramolecular distances	distance matrices, combinatorial, simulated annealing	AL,Z, RMSD	3.33
MaxCMO [46]	overlap between contact maps	Variable Neighborhood Search	AL, OL	3.32
CE [47]	inter-residue distances rigid body superposition	heuristics, dynamic programming	AL, Z, RMSD	1.27
USM [48]	Kolmogorov complexity	compression utilities	USM-distance	0.34
TM-align [49]	inter-atomic distances	rotation matrix, dynamic programming	AL, RMSD,TMS	0.21
FAST [50]	inter-residue distances	heuristics, dynamic programming	RMSD, AL, SN	0.07

single standard workstation or a small group of workstations, specially if one would like to perform a multi-criteria comparison for very large datasets in real-time. That is, as the number of protein structures being compared increases, the corresponding number of pairwise comparison jobs, I/O files and directories, computational time and memory required for each comparison method and associated pre-processing (e.g. data extraction and contact map preparation) and post-processing (e.g. consensus generation, clustering and result visualization) methods also increases. An estimate of some of these complexities is presented in the following sections.

Job Complexity

Job complexity for protein structure comparison depends on the size (i.e number of structures) of the dataset/database in hand as well as the mode of comparison. As of writing of this dissertation, there are *64,036* protein structures in the PDB and this number grows steadily. If we compare a particular protein against all the proteins in a given dataset (e.g. PDB), this is referred to as *target-against-all* mode of comparison. While being the simplest mode, it is usually used to compare a protein of unknown function but known structure with those whose structures and functions are known. The results of comparison would provide clues regarding the function of the query protein. The number of pairwise comparison jobs in this mode is directly related to the number of structures in the target dataset. For example, given the current holdings of PDB, there will be *64,036* comparison jobs while using target-against-all mode of comparison. However, in the case of multi-criteria comparison the actual number of jobs will be the number of target structures $\times$ *the number of methods* being used for multi-comparison.

Another mode of comparison is the one in which we compare all the elements of a particular dataset among itself or with all the elements of another dataset. This mode is referred as *all-against-all* comparison and is mostly used to cluster/classify a group of structures. The resulting clustering/classification is aimed to reveal the functional and evolutionary similarities among the proteins. The number of pairwise comparison jobs in this mode is proportional to the square of the number of protein structures involved in the comparison[1] $\times$ *the number of methods*. For example, the comparison jobs for current holdings of PDB using all-against-all mode with only one method will be:

[1]Please note that some methods return different similarities for the comparison of P_i with P_j and the reverse comparison

$$N_j = n^2 = 64036^2 = 4,100,609,296$$

Where, N_j represents the number of pairwise comparison jobs, while n being the current number of protein structures available in the PDB.

As mentioned above the actual number of jobs will be *4,100,609,296* $\times$ *the number of methods* being used. Therefore, it will require an optimal way to distribute all these jobs in the form of some smaller subsets (working packages) that could be submitted for parallel/distributed execution. Needless to say, this complexity calls for a high performance computing solution. Please note that protein structure prediction methods, e.g. Robetta [56] and I-TASSER [57], often sample thousands of "decoys" that must be compared and clustered together at each iteration of the algorithm as to obtain a centroid structure. Thus comparing thousands or ten of thousands of protein structures is not limited to assessing the PDB only but actually occurs as a sub-problem in many other structural bioinformatics activities.

Time Complexity

Different protein structure comparison algorithms have different time complexities and run time profiles. Table 1.1 provides an indicative comparison between the times taken by the algorithms we used in our experiments for a typical protein pair. Arguably, depending on the length of the members of a protein pair, the times mentioned in the table would change. However, these can be use to give a rough estimate[2] of the run time profile that can be expected from these algorithms:

[2]More detailed analysis of run times is provided in later chapters

target-against-all: for a given protein structure compared against the *64,036* structures in the PDB (assuming only one chain per PDB file), a Multi-Criteria comparison with the methods available in Table 1.1 consuming the time mentioned in the fifth column, would take, on a P4 (1.86GHz, 2GB RAM) dual-core workstation *37.98* days.

all-against-all: if one were to execute this type of comparison for the entire PDB, this will result on *2,798,939,025* pairwise comparison jobs (assuming again one chain per PDB file) and it would take about *6662.71* years for all jobs to finish on a single machine.

Space Complexity

Executing potentially millions of pairwise protein structure comparison jobs has strict requirements in terms of memory and bandwidth allocation. MC-PSC jobs generate a very large number of output data files that need to be parsed and summarized in a way that enables the execution of the normalization and consensus steps but also that falls within the memory constraints of the available computational infrastructure. With the current number of proteins structures in PDB, and the total number of comparison measures/metrics for all six methods (Table 1.1) there may be as many data items in the resultant di(similarity) matrix as:

$$n^2 \times (N_{mt} + 2) = 64{,}036^2 \times 17 = 69{,}710{,}358{,}032$$

Where *n* again represents the current number of protein structures in PDB, N_{mt} represents the total number of measures/metrics (see Table 1.1) and the additional *2* accounts for the two protein IDs involved in each comparison. Using a minimum of 5 digits/characters to hold each data item it may require about 348 GB to hold the matrix. Given the size of this matrix, it becomes indispensable to compute and hold its values in a distributed environment and use some parallel I/O

techniques to assemble each distributed portion directly at an appropriate storage location.

The above back-of-the-envelope calculations point to the need for a high-performance solution to the MC-PSC problem as described in the next section.

1.3 Research Objectives

Based on the above mentioned problem description, this dissertation seeks to provide a step change in computing capabilities through a suite of grid-styled distributed computing technologies by parallelizing the existing code to bring closer the dream of real-time multi-comparisons of very large protein structures datasets. The dissertation also aims at providing a reflection on the software engineering aspects behind the implementation of a distributed solution to the MC-PSC problem based on local computer cluster as well as with a Grid implementation. The investigation of several computational strategies, approaches and techniques is aimed to substantially improve the ways to enhance the ProCKSI's functionality and interoperability with other services thus extending its applicability to a wider audience. The combination of better and faster parallel and distributed algorithms with more similarity comparison methods is deemed to represent an unprecedented advance on protein structure comparison technology. Thus, the advances that are presented through this thesis might allow both directed and fortuitous discovery of protein similarities, families, superfamilies, domains, etc, on one hand and help pave the way to faster and better protein function inference, annotation and protein structure prediction and assessment, on the other hand.

1.4 General Methodology

The work presented in this dissertation follows several methodologies, namely,

1. **Wide and deep literature survey and analysis:** there is a plethora of literature available on the use of parallel and distributed approaches for a variety of applications in the field of life sciences. The comprehensive survey and analysis of these approaches provides not only the state-of-the-art knowledge in terms of the technological development but also helps in learning the lessons from others experiences and selecting the proved approaches for further investigation.

2. **An engineering approach to distributed and Grid computing:** this type of methodology provides the insight in terms of setting-up the distributed and grid computing infrastructure based on the lessons learned from the first methodology. Starting from the infrastructure of a local cluster and testing it with a variety of options to working on a National Grid Service requires all the nitty gritty technical skills of engineering the Grid.

3. **An experimental approach for evaluating the implementations:** the implementations of the proposed parallel and distributed approaches for the solution of MC-PSC problem are evaluated based on the results of the the experiments by using standard measures and metrics being used in the community.

4. **A self-reflective commentary on the easiness or otherwise of working with the Grid:** the design, implementation, and evaluation of each approach is discussed under relative sections in terms the complexity of working with the Grid.

1.5 Thesis Organization

The design, implementation and evaluation of the newly built distributed architecture along with related case studies and background material is presented in the form of self-contained chapters as

outlined bellow:

Chapter 2: Survey of Web and Grid Technologies in Life Sciences

This chapter provides a broad survey of how the life science community as a whole has benefited from the proliferation of web and grid technologies. The reason that the study of the *world wide web* (www) or simply the *web* is carried out along with the grid technology lies in the fact that the direction of current development in both of these technologies is coalescing towards an integrated and unified *Web-based grid service* or *Grid-based web service* environment [58]. That is, proper understanding of the current state-of-the-art in grid computing requires a thorough understanding of many concepts, standards and protocols related to web technologies. Therefore, this chapter reports on the developments in both of these technologies as they pertain to the general landscape of Life Sciences. As many of the problems pertaining to different fields of life sciences usually share a common set of characteristics in terms of their computational requirements, this broad survey was essential for investigating the most common technological solution to the problem discussed in section 1.2. The major focus of this technological review was to collate up-todate information regarding the design and implementation of various bioinformatics Webs, Grids, Web-based grids or Grid-based webs in terms of their infrastructure, standards, protocols, services, applications and other tools. The review, besides surveying the current state-of-the-art, also provides a road map for future research and open questions. However, due to the flood of literature prevailing under the heading of this chapter the need for another chapter focusing specially on the narrow field of structural proteomics aroused, as described in the next section.

Chapter 3: Overview of the Grid and Distributed Computing for Structural Proteomics

This chapter aims to provide a distilled overview of some of the major projects, technologies and resources employed in the area of structural proteomics. The major emphasis is to comment on various approaches related to the gridification and parallelization of some flagship legacy applications, tools and data resources related to key structural proteomics problems such as protein structure prediction, folding and comparison. The comments are based on theoretical analysis of some interesting parameters such as performance gain after gridification, user level interaction environments, workload distribution and the choice of deployment infrastructure and technologies. Furthermore, this chapter also provides the detailed description of the ProCKSI server's existing architecture which is essential for the comprehensive understanding of the major contribution of this dissertation as presented in the following chapters.

Chapter 4: Materials and Methods

This chapter provides the description of some basic principles and procedures that have been used to carry out this research. In particular it explains the representation of the problem space and various approaches used for its partitioning. It also provides the description of different infrastructures and datasets used for the experimental analysis of the newly proposed architecture.

Chapter 5: High-throughput Distributed Framework for MC-PSC

This chapter describes the design and implementation of a high-throughput distributed re-implementation of ProCKSI for very large data sets. The core of the new framework lies in the design of an innovative distributed algorithm that runs on each compute node in a cluster/grid environment to perform

structure comparison of a given subset of input structures using some of the most popular PSC methods (e.g. USM, MaxCMO, Fast, DaliLite, CE and TMalign). This is followed by the procedure of distributed consensus building. Thus the new algorithms proposed here achieves ProCKSI's similarity assessment quality but with a fraction of the time required by it. Experimental results show that the proposed distributed method can be used efficiently to compare *a)* a particular protein against a very large protein structures data set (target-against-all comparison), *b)* a particular very large scale dataset against itself or against another very large scale dataset (all-against-all comparison). The overall speedup and efficiency of the system is further optimized with different load balancing strategies that reduce the percentage of load imbalance on each node. A comparative picture of these load balancing strategies is also described in full details along with their empirical results. Performance evaluation of the new system with different alternative Local Resource Management System (LRMS)'s and MPI implementations was also carried out in order to choose the right enabling technologies from several different alternatives as described in the next chapter.

Chapter 6: Performance Evaluation under Integrated Resource Management Environment

This chapter evaluates the effect on the performance of MC-PSC jobs when the MPI environment is integrated with a Local Resource Management System (LRMS) such as Sun Grid Engine (SGE) and Portable Batch System (PBS) using different implementations of MPI standard such as MPICH and OpenMPI. Experiments with different ways of integration provide a comparative picture of the possible approaches with the description of resource usage information for each parallel job on each processor. Understanding of different ways of integration sheds light on the most promising routes for setting up an efficient environment for very large scale protein structure comparisons.

Chapter 7: On the Scalability of the MC-PSC in the Grid

Based on the lessons learned in previous chapters by evaluating the performance of the high-throughput distributed framework for MC-PSC, the next step would be to make use of the Grid computing to overcome the limitations of a single parallel computer/cluster. It is also a fact that the use of the Grid computing also introduces additional communication overhead which needs to be taken into consideration. This chapter describes the experiments performed on the UK National Grid Service (NGS), to evaluate the scalability of the distributed algorithm across multiple sites. The results of the cross-site scalability are compared with single-site and single-machine performance to analyze the additional communication overheard in a wide-area network.

Chapter 8: On the Storage, Management and Analysis of (Multi) Similarity Data for Large Scale Protein Structure Datasets in the Grid

This chapter 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 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) is investigated and compared. These experiments were conducted on the National Grid Service (NGS), UK.

Chapter 9: Consensus based Protein Structure Similarity Clustering

This chapter compares the results of two competing paradigms for consensus development i.e., the *Total evidence* and the *Total consensus*. It uses the Protein Kinase Dataset to perform the classification with both of these approaches and discuses the pros and cons of each approach.

Chapter 10: Conclusions and Future Work

This chapter builds on the aggregation of individual conclusions from all of the chapters and provides a holistic view of the overall conclusions of this thesis. It also provides some directions for the future work.

1.6 List of Contributions

This thesis is a direct contribution to the field of bio-sciences in general and Multi-Criteria Protein Structure Comparison and Analysis in particular. It focuses on the development of a novel computational framework for MC-PSC based on grid-styled distributed computing strategies. To this end, a number of contributions were made, which are listed bellow:

- **State-of-the-art literature review**

  Technological evolution has brought upon converging effects in computing and hence a thorough understanding of one paradigm requires the knowledge and understanding of many other paradigms which are interrelated and are interweaving to form a single whole. The scope of distributed computing is broadening from parallel to Grid computing on one hand, and, on the other hand the architecture of the Grid computing is moving towards integration with the world wide web to form what is called the *World Wide Grid*. This dissertation contributes

in presenting a comprehensive survey of the scholarly articles, books, and other resources related to the evolution and application of the Grid/distributed computing in the field of biosciences in general and structural proteomics in particular. Besides identifying the areas of prior scholarship, this review also points the way forward for future research.

- **Design, implementation and evaluation of a novel distributed framework for MC-PSC**

  The solution framework for the enormous computational challenge inherent in the nature of MC-PSC is the major contribution of this dissertation. The framework is based on a novel distributed algorithm that scales to any number of available nodes in a cluster/grid environment. It uses the local storage/memory of each node to store the multi-similarity data in a distributed environment and hence tackles both data and compute intensive nature of MC-PSC and makes the processing of large scale datasets possible.

- **Empirical analysis of different load balancing strategies for MC-PSC**

  Given the fact that the size/length of protein structures varies from a few tens of amino acids to several hundreds, the execution time is hugely different for different pairwise comparisons. This variation in the execution time renders the simple decomposition based on the equal number of pairwise comparisons per node inefficient due to heavy load imbalance. This thesis presents an empirical analysis of this load imbalance and proposes an efficient load balancing strategy that distributes the load based on the number of residues.

- **Studies on the integration of parallel and local resource management environments**

  In order to reflect on the engineering aspects of the distributed environment, this dissertation provides an empirical analysis of different integrated environments for parallel and local resource management systems. The results of this analysis would significantly contribute in the

design and implementation of an optimal distributed computing infrastructure needed for the execution of distributed algorithms/applications.

- **Grid scalability evaluation for distributed MC-PSC**

  The deployment of the distributed application on the UK's National Grid Service (NGS) infrastructure and evaluation of its scalability across multiple clusters provides large scale analysis of the effect of gridification. This analysis provides insights on how to overcome the limitations of a single site/cluster and further enhance the performance of the system by leveraging the resources from multiple sites.

- **Comparison and evaluation of different database technologies for proposed MC-PSC science center**

  The proposal of a science center for MC-PSC based on the pre-computed results of multi-similarity data is another contribution of this dissertation. To this aim, a survey, analysis and evaluation of different database technologies is presented in the context of MC-PSC.

- **Comparison and evaluation of different consensus-based approaches for MC-PSC**

  The overall goal of the MC-PSC is to provide a consensus-based view of the protein structure similarity. Since, there are many different approaches for building the consensus, this dissertation contributes in providing a comparative picture of the two most widely used paradigms i.e. *Total Evidence*(TE) and *Total Consensus* (TC).

1.7 Publications

During the course of this thesis, the following peer reviewed publications were also contributed:

Peer Reviewed Conference Papers:

1. G. Folino, **A. A Shah**, and N. Krasnogor.: On the Scalability of Multi-Criteria Protein Structure Comparison in the Grid, In: Proceedings of The Euro-Par 2010 Workshop on High Performance Bioinformatics and Biomedicine (HiBB), August 31-Sep 3, 2010 , Ischia, Naples, Italy.

2. G. Folino, **A. A Shah**, and N. Krasnogor.: *On the Storage, Management and Analysis of (Multi) Similarity for Large Scale Protein Structure Datasets in the Grid*, In: Proceedings of IEEE CBMS 2009, the 22nd IEEE International Symposium on Computer-Based Medical Systems, August 3-4, 2009, Albuquerque, New Mexico, USA.

3. **A. A Shah**, G. Folino, D. Barthel and N. Krasnogor.: *Performance Evaluation of Protein (Structure) Comparison Algorithms under Integrated Resource Environment for MPI Jobs*, In: Proceedings of International Symposium on Parallel and Distributed Processing with Applications (ISPA '08), ISBN: 978-0-7695-3471-8, pp. 817-822, IEEE Computer Society, 2008.

4. **A.A. Shah**, D. Barthel and N. Krasnogor.: *Grid and Distributed Public Computing Schemes for Structural Proteomics*. In: P. Thulasiraman et. al. (Eds.): Frontiers of High Performance Computing and Networking ISPA 2007 Workshops, Lecture Notes in Computer Science, Vol. No. 4743, pp. 424-434, Springer-Verlag Berlin Heidelberg, 2007.

5. **A.A. Shah**, D. Barthel and N. Krasnogor.: Protein Structure Comparison, Clustering and Analysis: An overview of the ProCKSI decision support system. In: Proceedings of International Symposium on Biotechnology (ISB2007), University of Sindh, Pakistan, Nov 4-8, 2007

Peer Reviewed Journal Papers:

1. **A. A Shah**, G. Folino, and N. Krasnogor.: *Towards a High-Throughput, Multi-Criteria Protein Structure Comparison*, IEEE Transactions on NanoBioscience, Vol. 9(2), pp.144-155, 2010. [doi:10.1109/TNB.2010.2043851]

2. **A. A Shah**, D. Barthel1, P. Lukasiak, J. Blacewicz and N. Krasnogor.: *Web and Grid Technologies in Bioinformatics, Computational Biology and Systems Biology: A Review*. In: Current Bioinformatics, Vol 3 (1), pp.10-31(22), Bentham Publishers, 2008.

1.8 Conclusions

This chapter provided the general introduction to the thesis. It started with the background of the MC-PSC and described in detail the dimensions of the problem (computational challenge) that it faces and the main objectives of this research. The general methodology for the proposed solution has been described along with the outline and scope of the material presented in the rest of the thesis. The chapter also mentions some key contributions of this dissertation along with list of publications. The next couple of chapters provide the comprehensive survey/review of the literature which becomes foundations for the building of successive chapters.

CHAPTER 2

SURVEY OF WEB AND GRID TECHNOLOGIES IN LIFE SCIENCES

Chapter 1 provided a succinct description of the research topic and also explained the objectives and structure of the thesis in general. This chapter and the one that follows this would review the relevant literature. The review presented in these two chapters starts from a wide perspective of the field and finally converges and focuses on the specific topic of the research and summaries the potential findings that could help in conducting the research.

This chapter was published as a peer reviewed journal article in *Current Bioinformatics*, Vol. 3(1), pp.10-31, 2008. [doi:10.2174/157489308783329850].

2.1 Introduction

> "The impact of computing on biology can fairly be considered a paradigm change as biology enters the 21st century. In short, computing and information technology applied to biological problems is likely to play a role for 21st century biology that is in many ways analogous to the role that molecular biology has played across all fields of biological research for the last quarter century and computing and information technology will become embedded within biological research itself" [59].

As an example of the above referred conclusion regarding the embedding of computing

and information technology (IT) in biological research, one can look at the state-of-the-art in web and grid technologies as applied to bioinformatics, computational biology and systems biology. The World Wide Web or simply the web has revolutionized the field of IT, and related disciplines, by providing information-sharing services on top of the internet. Similarly grid technology has revolutionized the field of computing by providing location-independent resource sharing-services such as computational power, storage, databases, networks, instruments, software applications and other computer related hardware equipment. These information and resource-sharing capabilities of web and grid technologies could upgrade a single user computer into a global supercomputer with vast computational and communication power, storage capacity. Access to very large-scale datasets, application programs and tools are also some of the benefits of web and grid. The so called upgraded web and grid-enabled global super computer would make itself a potential candidate to be used in resource-hungry computing domains. For example, it could be used to efficiently solve complex calculations such as parameter sweep scenario with Monte Carlo simulation and modeling techniques, which would normally require several days, months, years or even decades of execution time on a traditional single desktop processor.

A quick look at the literature reveals that web and grid technologies are continuously being taken up by the biological community as an alternate to traditional monolithic high performance computing mainly because of the inherent nature of biological resources (distributed, heterogeneous and CPU intensive), smaller financial costs, better flexibility, scalability and efficiency offered by the web and grid-enabled environment. An important factor that provides the justification behind the ever growing use of web and grid technologies in life sciences is the continuous and rapid increase in biological data production. It is believed that *a typical gene laboratory can generate*

approximately 100 terabytes of information in a year [60], which is considered to be equivalent to about 1 million encyclopedias. The problem of managing these rapidly growing large datasets is further compounded with their heterogeneous and distributed nature (varying in terms of storage and access technologies). Currently there are no uniform standards (or at least not yet been adopted properly by the biological community as a whole) to deal with the diverse nature, type, location and storage formats of this data. On the other hand, in order to obtain the most comprehensive and competitive results, in many situations, a biologist may need to access several different types of data which are publicly available in more than 700 [61] biomolecular databases. One way to handle this situation is to convert the required databases into a single format and then store it on a single storage device with extremely large capacity. Considering the tremendous size and growth of data this solutions is infeasible, inefficient and very costly. The application of Web and Grid technology provides an opportunity to standardized the access to these data in an efficient, automatic and seamless way by providing an intermediary bridge architecture as shown in Figure 2.1. The so called intermediary bridge makes use of appropriate web and grid technologies and standards such as grid middleware specific Data Management Service (DMS), distributed storage environments such as Open Grid Service Architecture-Data Access and Integrator (OGSA-DAI) (`http://www.ogsadia.org`) with Distributed Query Processor (DQP), Storage Resource Broker (SRB) [62] and IBM DiscoveryLink [63] middleware etc.

Furthermore, it is also very common for a typical biological application that involves very complex analysis of large-scale datasets and other simulation related tasks to demand for high throughput computing power in addition to seamless access to very large biological datasets. The traditional approach towards this solution was to purchase extremely costly special-purpose super

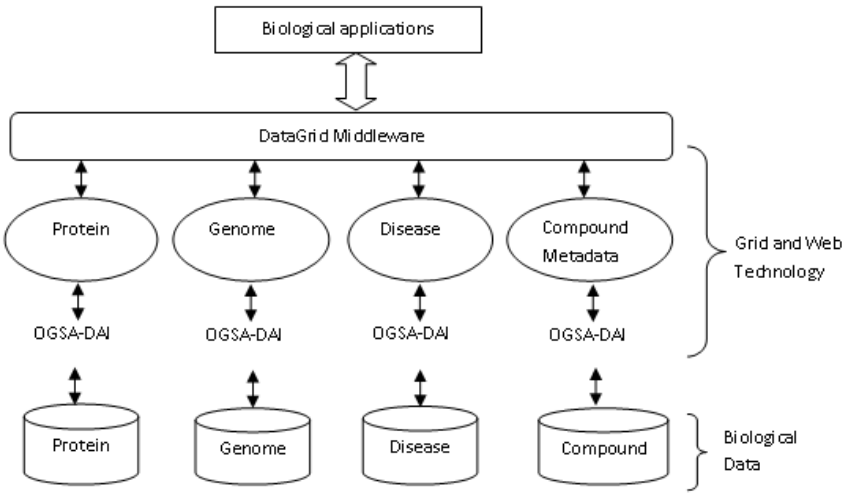


FIGURE 2.1: Major architectural components of a biological DataGrid environment (reproduced from [64] and annotated). The middle layer, which consists of grid and web technologies, provides interoperatibility between biological applications and various types of data.

computers or dedicated clusters. This type of approach is both costly somewhat limited as it locks the type of computing resources. Another problem associated with this approach would be that of poor utilization of very costly resources, i.e. if a particular application finishes its execution then the resources could remain idle.

Grid technology on the other hand provides more dynamic, scalable and economical way of achieving as much computing power as needed through computational grid infrastructures connected to a scientist's desktop machine. There are many institutional, organizational, national and international Data/Computational/Service Grid testbeds and well established production grid environments which provide these facilities free of charge to their respective scientific communities. Some of these projects include Biomedical Research Informatics Delivered by Grid Enabled Services (BRIDGES) `http://www.brc.dcs.gla.ac.uk/projects/bridges`, EGEE(http://public.eu-egee.org) [65], Biomedical Informatics Research Network (BIRN) (`http://www.nbirn.net`), National Grid Service UK [66], OpenBioGrid Japan [64], SwissBioGrid [67], Asia Pacific BioGrid

(http://www.apgrid.org), North Carolina BioGrid (http://www.ncbiotech.org), etc. All these projects consist of an internet based interconnection of a large number of pre-existing individual computers or dedicated clusters located at various distributed institutional and organizational sites that are part of the consortium. Figure 2.2 illustrates some major architectural components for such a computational setup. Each grid site in the network has usually a pool of compute elements managed by some local resource managing software such as Sun Grid Engine (SGE: http://gridengine.sunsource.net), (PBS: http://www.openpbs.org), (Load Sharing Facility (LSF): www.platform.com/Products/Platform.LSF) and Condor (www.cs.wisc.edu/condor) etc. Similarly grid storage elements are accessed via data management services and protocols such as GridFTP, Reliable File Transfer Protocol (RFTP) and OGSA-DAI etc.

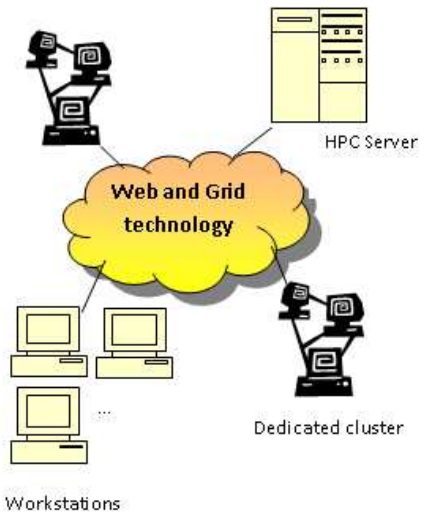


FIGURE 2.2: Computational grid architecture: internet-based interconnection of heterogeneous and distributed individual workstations, dedicated clusters, high performance computing (HPC) servers and clusters of distributed PCs (Desktop PC Grid)

Some other large scale bioinformatics grid projects have provided a platform where a biologist can design and run complex *in-silico* experiments by combing several distributed and heterogeneous resources that are wrapped as web-services. Examples of these are myGrid [68, 69],

BioMOBY [70–72], Seqhound (http://www.blueprint.org/seqhound) and Biomart (http://www.biomart.org) etc., which allow for the automatic discovery and invocation of many bioinformatics applications, tools and databases such as European Molecular Biology Open Software Suite (EMBOSS) [73] suite of bioinformatics applications and some other publicly available services from the National Center for Biomedical Informatics (NCBI) (http://www.ncbi.nlm.nih.gov) and European Bioinformatics Institute (EBI) (www.ebi.ac.uk). These projects also provide some special toolkits with necessary application programming interfaces (APIs), which can be used to transform any legacy bioinformatics application into a web-service that can be deployed on their platforms.

The availability of these BioGrid projects brought into sharp focus the need for better user interfaces as to provide the biologist with easier access to these web/grid resources. This has led to the development of various web based interfaces, portals, workflow management systems, problem solving environments, frameworks, application programming environments, middleware toolkits, data and resource management approaches along with various ways of controlling grid access and security. Figure 2.3 provides a pictorial view of these technologies in the context of the BioGrid architecture. This review attempts to provide an up-to-date coherent and curated overview of the most recent advances in web and grid technologies as they pertain to life sciences. The review aims at providing a complementary source of additional information to some previous reviews in this field such as [74, 75].

Among the many advances that the computational sciences have provided to the life sciences the proliferation of web and grid technologies is one of the most conspicuous. Driven by the demands of biological research these technologies have moved from their classical and some-

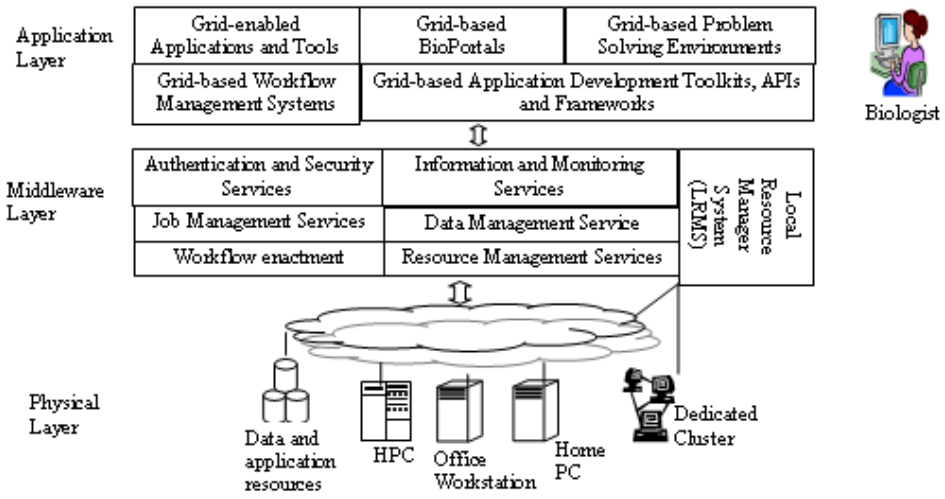


FIGURE 2.3: Major components of a generic BioGrid infrastructure: there are three main layers; the application, the middleware and the physical layer. The application layer focuses on APIs, Toolkits, Portals, Workflows etc., the middleware layer focuses on the application, data and resource management and services while the physical layer provides the actual compute and data resources.

what static architecture to more dynamic and service-oriented architecture. The direction of current development in these technologies is coalescing towards an integrated and unified Web-based grid service [58] or Grid-based web service environment as shown in Figure 2.4. Accompanying this rapid growth, a huge diversity of approaches to implementation and deployment routes have been investigated in relation to the use of various innovative web and grid technologies for the solution of problems related to life sciences. The following sections provide an overview of some of these works as per orgnaziation of Figure 2.5; sections 2.2 and 2.3 present a comprehensive review of the web and grid technologies respectively; section 2.4 describes the architecture, implementation and services provided by a selection of flagship projects. Finally, section 2.5 presents concluding remarks on the reviewed literature with a clear indication of certain key open problems with the existing technological approaches and provides a road-map and open questions for the future.

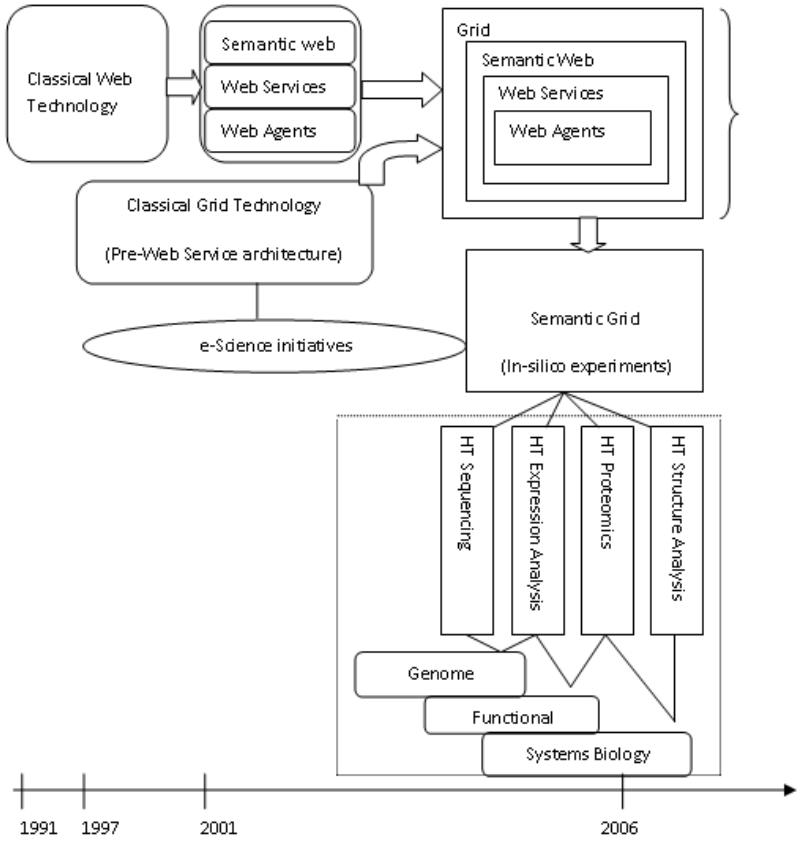


FIGURE 2.4: Review of technological infrastructure for life sciences: classical HTML based web started in 1991 and traditional Globus based grid was introduced by Ian Foster in 1997. With the introduction and development of semantic web, web-services and web agents in and after 2001, the new web and grid technologies are being converged into a single uniform platform termed as 'service-oriented autonomous semantic grid' that could satisfy the needs of HT (high throughput) experimentations in diverse fields of life sciences as depicted above.

2.2 Web Technologies

2.2.1 Semantic Web Technologies

One of the most important limitations of the information shared through classical web technology is that it is only interpretable by human and hence it limits the automation required for more advanced and complex life science applications that may need the cascaded execution of several analytical tools with access to distributed and heterogeneous databases. The basic purpose of semantic web

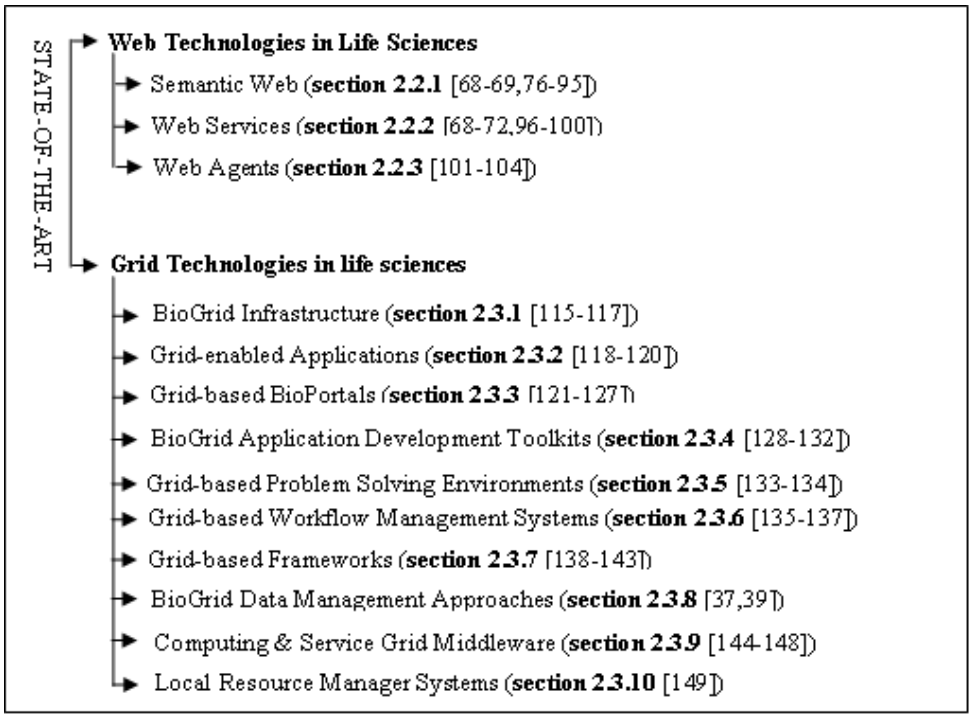


FIGURE 2.5: Hierarchical organization of the state-of-the-art overview of web and grid technologies. The number in brackets point to related references that has been reviewed.

technology is to eliminate this limitation by enabling the machine (computer) to interpret/understand the meaning (semantics) of the information and hence allow artificial intelligence based applications to carry out decisions autonomously. It does so by adding some important features to the basic information-sharing service provided by the classical web technology. These features provide a common format for interchange of data through some standard languages and data models such as eXtensible Markup Language (XML), Resource Description Framework (RDF) along with several variants of schema and semantic based markup languages such as, Web Ontology Language (OWL) and Semantic Web Rules-Language (SWRL) etc. Wang et al. [76] argues that although initially XML was used as a data standard for platform independent exchange and sharing of data, because of its basic syntactic and document-centric nature, it was found limited, especially for the rep-

resentation of rapidly increasing and diverse 'omic' data . Therefore, currently RDF along with some new variants of OWL such as OWL-Lite, OWL-DL and OWL-Full are being adopted as to implementation shown in Figure 2.6. Various attempts towards the use of semantic web for life sciences have been reported in the literature mainly focusing on data/application integration, data provenance, knowledge discovery, machine learning and mining etc. For example, Satya et al. [77] discusses the development of a semantic framework based on publicly available ontologies such as GlycO and ProPreO that could be used for modeling the structure and function of enzymes, glycans and pathways. The framework uses a sublanguage of OWL called OWL-DL [78] to integrate extremely large (500MB) and structurally diverse collection of biomolecules. One of the most important problems associated with the integration of biological databases is that of their varying degree of inconsistencies. Because of this, there have also been certain efforts for providing some external semantic-based tools for the measurement of the degree of inconsistencies between different databases. One such effort is discussed in [79], which describes an ontology-based method that uses a mathematical function to determine the compatibility between two databases based on the results of semantically matching the reference.

The autonomous and uniform integration, invocation and access to biological data and resources as provided by semantic web have also created an environment that supports the use of in-silico experiments. Proper and effective use of In-silico experiments requires the maintenance of user specific provenance data such as record of goals, hypothesis, materials, methods, results and conclusions of an experiment. For example, Zhao et al. [81] showcases the design of a RDF based provenance log for a typical in-silico experiment that performs DNA sequence analysis as a part of myGrid [68, 69] middleware services. The authors have reported the use of Life Sci-

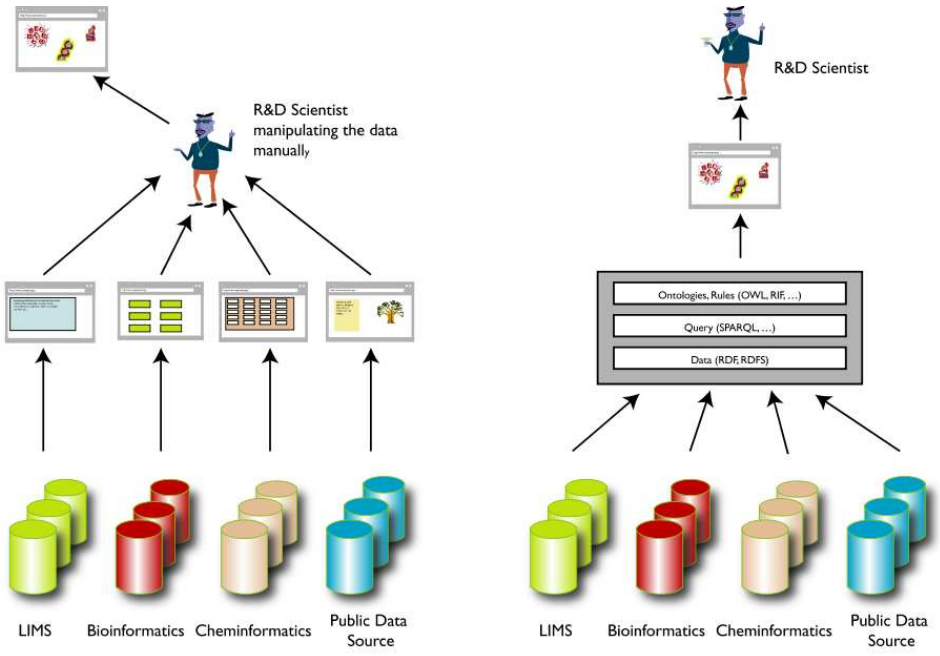


FIGURE 2.6: A biologist's view of classical and semantic web. The classical web does not provide the automated (computational) integration and interoperatibility between various sources of data/applications/services and the biologist needs to do all of this manually. The semantic web on the other hand frees the biologist from lot of manual work. (Reproduced from [80]).

ence Identifiers (LSID) for achieving location-independent access to distributed data and meta-data resources, RDF and OWL have been used for associating uniform semantic information and relationships between resources, while Haystack [82], a semantic web browser, has been used for delivering the provenance-based web pages to the end-user. The use of RDF model as compared to XML provides more flexible and graph-based resource description with location independent resource identification through URIs (Universal Resource Identifier).

There are various other significant contributions that illustrate the use of semantic web technology for the proper integration and management of data in the context of bioinformatics, computational biology and systems biology. For example, The Gene Ontology Consortium [83,84], make use of semantic web technologies to provide a central gene ontology resource for unification

of biological information. [85, 86] uses OWL-DL to develop a data exchange format that facilitates integration of biological pathway knowledge. [87, 88] uses semantic web to provide a resource for the development of tools for microarray data acquisition and query according to the concepts specified in Minimum Information About a Microarray Experiment (MIAME) a standard [89]. Further information about some semantic web and ontology based biological applications and tools for life sciences is provided in Table 2.1.

TABLE 2.1: Semantic-web and ontology based resources for life science

Semantic web based application/tool	**Usage**
Biological Pathway Exchange (BioPAX [82, 83])	Data exchange format for biological pathway data http://www.biopax.org/
Microarray for Gene Expression Data (MGED [84, 85])	Data standard for Systems Biology http://www.mged.org/
Transparent Access to Multiple Bioinformatics Information Sources (TAMBIS [86])	Biological Data Integration http://img.cs.man.ac.uk/tambis
Software Development Kit for cancer informatics (caCORE SDK [87])	Semantically integrated bioinformatics software system http://ncicb.nci.nih.gov/infrastructure/cacoresdk
AutoMatic Generation of Mediator Tools for Data Integration (AutoMed Toolkit [88])	Tools for assisting transformation and integration of distributed data http://www.doc.ic.ac.uk/automed/
Gaggle [90]	An integrated environment for systems biology http://gaggle.systemsbiology.org/docs/
Encyclopedia of Escherichia coli K-12 Genes and Metabolism (EcoCyc [91])	Molecular catalog of the E. coli cell http://ecocyc.org
Systems Biology Markup Language (SMBL [92])	Computer-readable models of biochemical reaction networks. http://sbml.org/index.psp
Cell Markup Language (CellML [93])	Storage and exchange of computer-based mathematical models for biomolecular simulations (http://www.cellml.org/)
Open Biomedical Ontology (OBO [90]) (OBO [90])	Open source controlled-vocabularies for different biomedical domains (http://obo.sourceforge.net)
Gene Ontology (GO [68]) Gene Ontology (GO [68])	Controlled-vocabulary for genes http://www.geneontology.org/
Generic Model Organism Database (GMODS [94])	An integrated organism database http://www.gmod.org/home
Proteomics Standards Initiative: Molecular Interactions (PSI-MI [95])	Data standard for proteomics http://psidev.sourceforge.net

2.2.2 Web Service Technologies

Web-service technology further extends the capabilities of classical web and semantic web by allowing information and resources to be shared among machines even in a distributed heterogeneous environment (such as a grid environment). Therefore, applications developed as web services can interoperate with peer applications without taking care of particular language, file system, operating system, processor or network. Web services are defined through Web Service Description Language (WSDL) and deployed and discovered through Universal Description, Discovery and Integration (UDDI) protocol. They can exchange XML based messages through Simple Object Access Protocol (SOAP) over different computer platforms. Furthermore, with the introduction of Web Service Resource Framework (WSRF), now web services have become more capable of storing the state information during the execution of a particular transaction. These features of web-services have made them extremely important to be applied to life science domain.

Today many life science applications are being developed as web services. For example, the National Center for Biotechnology Information (NCBI) provides a wide range of biological databases and analytical tools as web services such as all the Entrez e-utilities including EInfo, ESearch, EPost, ESummary, EFetch, ELink, MedLine, and PubMed. Similarly, the EBI provides many biological resources as web services such as SoapLab, WSDbfetch, WSfasta, WSBLast, WSInterProScan, EMBOSS amongst others. A comprehensive list of all publicly available and accessible biological web services developed by different organizations, institutions and groups can be found at myGrid website (`http://taverna.sourceforge.net`).

All these web services can be used as part of complex application specific programs. IBM provides WebSphere Information Integrator (WS II) as an easy way for developers to integrate indi-

vidual web-service components into large programs. As an example, North Carolina BioGrid [96] in collaboration with IBM uses web services to integrate several bioinformatics applications to high performance grid computing environment. This BioGrid also provides a tool (WSDL2Perl) to facilitate the wrapping of Perl based legacy bioinformatics applications as web services. Other open source projects that provide registry, discovery and use of web services for bioinformatics, computational biology and systems biology include myGrid [68, 69], BioMOBY [70–72], and caBIG(`https://cabig.nci.nih.gov/`) etc. The integration and interoperatibility of distributed and heterogeneous biological resources through web services has opened an important niche for data mining and knowledge discovery. For example, Hahn U et al. [97] introduces web based reusable text mining middleware services for bio-medical knowledge discovery. The middleware provides a Java based API for clients to call searching and mining services. Similarly, Hong et al. [98] uses web services for the implementation of a microarray data mining system for drug discovery.

Due to the success of web services to provide flexible, evolvable and scalable architectures with interoperability, between heterogeneous applications and platforms, the grid middleware is also being transformed from its pre-Web Service versions to the new ones based on Web Services. There are several initiatives in this direction such as Globus [34, 35], an open source grid middleware, has adopted web service architecture in its current version of Globus Toolkit 4, the EGEE project is also moving from its pre-web service middleware LHC Computing Grid (LCG2) to a new web service based middleware gLite (http://glite.web.cern.ch/glite/) [65], and similarly ICENI [99] and myGrid which are also adopting the web services through the use of Jini, OGSI, JXTA and other technologies [100].

2.2.3 Agent-based Semantic Web Services

Agents are described as software components that exhibit autonomous behavior and are able to communicate with their peers in a semantically defined high-level language such as FIPA-ACL (Foundation of Intelligent Physical Agents- Agents Communication Language). Since the main focus of agent technology is to enable the software components to perform certain tasks on behalf of the user, this somewhat relates and supports the goals of web-services technology and hence the two technologies have started converging towards the development of more autonomous web-services that exhibits the behavior of both web-services as well as agents.

There have been many attempts regarding the use of agents in bioinformatics, computational biology and systems biology. Merelli et al. [101], reports the use of agents for the automation of bioinformatics tasks and processes, Phylogenetic analysis of diseases, protein secondary structure prediction, stem cell analysis, and simulation among others. In their paper, the authors also highlight key open challenges in agents research: analysis of mutant proteins, laboratory information management system (LIMS), cellular process modeling, formal and semi-formal methods in bioinformatics. Similarly the use of mobile agents for the development of a decentralized, self-organizing peer-to-peer grid computing architecture for computational biology has been demonstrated in [97].

Another important use of agents in combination with semantic web and web services lies in the provision of service-oriented semantic grid middleware. For example, Luc et. al. [102], suggests the use of agents in *myGrid* [68, 69] middleware in order to best fit the ever-dynamic and open nature of biological resources. In particular they propose the use of agents for 'personalization, negotiation and communication'. The personalization agent can act on behalf of the user to automatically provide certain preferences such as the selection of preferred resources for a work-

flow based in-silico experiment and other user related information. The user-agent can store these preferences and other user related information from previously conducted user activities and thus freeing the user from tedious repetitive interactions. The user-agent could also provide a point of contact for notification and other services requiring user interaction during the execution of a particular experiment. Other experiences related to the use of agents for biological data management and annotations have been discussed in [103, 104].

2.3 Grid Technologies

Right from its inception, the main focus of grid technology has been to provide platform independent global and dynamic resource-sharing service in addition to co-ordination, manageability, and high performance. In order to best satisfy these goals, its basic architecture has undergone substantial changes to accommodate other emergent technologies. As shown in Figure 2.4, the grid has moved from its initial static and pre-web service architecture to a more dynamic Web Service Resource Framework(WSRF) based Open Grid Service Architecture (OGSA) [105] that combines existing grid standards with emerging Service Oriented Architectures(SOAs), innovative web technologies such as the semantic web, web services and web agents. This organic maturation of the grid seeks a unified technological platform that is known as service-oriented semantic grid.

The main characteristics of this service-oriented semantic grid would be to maintain intelligent agents that act as software services (grid services) capable of performing well-defined operations and communicating with peer services through uniform standard protocols such as used for web services (XML, SOAP, WSDL, UDDI, etc.). This paradigm shift in the grid's architecture is gaining relevance due to its impact on the usability of bioinformatics, computational biology and

systems biology. We find various successful demonstrations of the gridificatiton of biological resources indicating the effect and power of grid-based execution of various life science applications with different technological approaches. For example, Jacq et. al. [106] reports the deployment of various bioinformatics applications on the European Data Grid (EDG) testbed project. One of the deployed applications was PhyloJava, a Graphical User Interface (GUI) based application that calculates the polygenetic trees of a given genomic sequence using fastDNAml [107] algorithm that uses bootstrapping (a reliable albeit computationally intensive technique that calculates a consensus from a large number of repeated individual tree calculations (about 500-1000 repeats). The gridification of this application was carried out at a granularity of 50 for a total of 1000 independent sequence comparison jobs (20 independent packets of 50 jobs each) and then merging the individual job results to get the final bootstrapped tree. The selection of the appropriate value of granularity depends upon the proper consideration of the overall performance because highly parallelized jobs can be hampered by resource brokering and scheduling times whereas poorly parallelized jobs would not give significant CPU time gain [106]. The execution of this gridified application on the EDG testbed required the installation of a Globus [34,35] based EDG user interface on a Linux RedHat Machine, the use of Job Description Language (JDL) and the actual submission of the parallel jobs through Java Jobs Submission Interface (JJSI). It is reported that the gridified execution of this application provided 14 times speedup compared against a non-grid based standalone execution. The deviation in gain from ideal (speed up of 20) is considered to be the effect of network and communication overhead (latencies). Similarly, the gridification of other applications such as a grid-enabled bioinformatics portal for protein sequence analysis, grid-enabled method for securely finding unique sequences for PCR primers, and grid-enabled BLAST for orthology rules determination has also

been discussed in [106] with successful and encouraging results.

The gridification of biological databases and applications is also motivated by the fact that the number, size and diversity of these resources are continuously (and rapidly) growing. This makes it impossible for an individual biologist to store a local copy of any major databases and execute either data or computer-intensive application in a local environment even if supported with a dedicated cluster or high performance computing resources. This inability of locality demands for the grid-enablement of the resources. However, an important factor that hinders the deployment of existing biological applications, analytical tools and databases on grid-based environments is their inherent pre-grid design (legacy interface). This is so because the design suits the requirements of a local workstation environment in terms of input/output capabilities and makes it very difficult for these applications to be gridified. The gridification of such applications requires a transparent mechanism to connect local input/output with a grid-based distributed input/output through some intermediatary tools such as grid middleware specific DMS and distributed storage environments.

One such mechanism, discussed in [108], provides a transparent interface for legacy bioinformatics applications, tools and databases to be connected to computational grid infrastructures such as EGEE [65], without incurring any change in the code of these applications. Authors have reported the use of modified Parrot [109] as a tool to connect a legacy bioinformatics application to the EGEE database management system. The EGEE database management system enables location and replication of databases needed for the management of very large distributed data repositories. With Parrot-based connection, the user is freed from the overhead of performing file staging and specifying in advance an application's data need. Rather, an automated agent launched by the Parrot takes care of replicating the required data from the remote site and supplying it to the

legacy application as it would have been accessing data with local input/output capabilities. The agent resolves logical file name to the storage file name, selects the best location for replication and launching the program for execution on the downloaded data. For the purpose of demonstration, authors have reported the deployment (virtualization) of some biological databases such as Swiss-Prot and TrEMBL [110] by registering these databases with the replica management service (RMS) of EGEE. Similarly, programs for protein sequence comparison such as BLAST [111], FASTA [112], ClustalW [113] and SSearch [114] have been deployed by registering them with the experiment software management service (ESM). The deployed programs were run on a grid environment and their access to the registered databases was evaluated by two methods: replication (by copying the required database directly to the local disk) and remote input/output (attaches the local input/output stream of the program to the copied data in cache or on-the-fly mode). The evaluation of these methods show that both methods perform similarly in terms of efficiency e.g. on a database of about 500,000 protein sequences (205 MB) each method takes about 60 seconds for downloading from any grid node and about four times less than this time in case the data node is near the worker node. It is important to note, however, that the replication method creates an overhead in terms of free storage capacity on the worker node. This problem may be particularly if the size of the database to be replicated is too high or if the worker node has many CPUs sharing the same storage and each accessing a different set of databases. This is the reason why remote input/output method overweighs the replication method for accessing large distributed biological databases. Again the real selection depends on the nature of program (algorithm). For compute-intensive programs such as SSearch remote input /output is always better (as it works on copying progressive file blocks) where as for data-intensive programs such as BLAST and FASTA the replication method may work better [108].

In fact there are a very large number of contributions that report the experiences of using grid technology for bioinformatics, computational biology and systems biology. We have tried to summarize the findings of some of these contributions under the relevant categories of large projects including BioGrid infrastructure, middleware, local resource management systems, data management, application programming environments, toolkits, problem solving environments and workflow management systems and are presented in the following sections.

2.3.1 BioGird Infrastructure

Mostly BioGrid infrastructure is based on the simple idea of cluster computing and is leading towards the creation of a globally networked massively parallel supercomputing infrastructure that connects not only the computing units along with their potential hardware, software and data resources, but also expensive laboratory and industrial equipment, and ubiquitous sensor device in order to provide unlimited computing power and experimental setup required for modern day biological experiments. Moreover, this infrastructure is also being customized in a way that it becomes easily accessible by all means of an ordinary general purpose desktop/laptop machine or any type of handheld devices. Some of the major components of a generic BioGrid infrastructure has been illustrated in Figure 2.3. The overall architectural components are organized at three major levels (layers) of services. The focus of application layer services is to provide user-friendly interfaces to a biologist for carrying out the desired grid-based tasks with minimum steps of usability and interaction (enhanced automation and intelligence). Similarly the focus of grid middleware services is to provide seamless access and usability of distributed and heterogeneous physical layer resources to the application layer services. In the following sections we discuss various contributions related to the development and use of some of these services at both application and middleware level.

The design and implementation of a typical BioGrid infrastructure varies mainly in terms of the availability of resources and demands of the biological applications that are supposed to use that particular grid. There are many infrastructures starting from an institutional/organizational grids consisting of simple PC based clusters or combination of clusters [115, 116] to national and international BioGrid projects with different architectural models such as Computing Grid architecture (providing basic services for task scheduling, resource discovery, allocation and management etc), Data Grid architecture (providing services for locating, accessing, integrating and management of data), Service Grid architecture (services for advertising, registering and invocation of resources) and Knowledge Grid architecture (services for sharing collaborative scientific published or unpublished data). The infrastructure details of some major BioGrid projects are presented in Table 2.2. It may be observed that the same infrastructure may be used to serve more than one application models based on the availability of some additional service and resources. For example a compute grid with the help of some additional services and resources can be set to work as a data grid.

2.3.2 Grid-enabled Applications and Tools

As discussed in the following sections, there have been some efforts for the development of bioinformatics specific grid programming and problem solving environments, toolkits, frameworks and workflows that can help to develop grid-enabled applications easily and efficiently; however they are still, to a degree, in a prototype or demonstration state. The actual process of developing (writing the code), deploying (registering, linking and compiling), testing (checking the results and performing debugging if necessary) and executing (scheduling, coordinating and controlling) an application in a grid-based environment is far from trivial. Mainly, the difficulties faced by developers arise because of incapability of traditional software development tools and techniques to support the de-

TABLE 2.2: BioGrid infrastructure projects

BioGrid project	**Grid Infrastructure**	**Main applications**
Asia Pacific BioGrid http://www.apgrid.org	Globus1.1.4 Nimrod/G, LSF. SGE. 5 nodes, 25+ CPUs, 5 sites.	FASTA, BLAST, SSEARCH, MFOLD Virtual Lab DOCK, EMBASSY, PHYLIB and EMBOSS.
Open BioGrid Japan OBIGRID Japan [64] http://www.obigrid.org	Globus 3.2. Ipv6 for secure communication. VPN over internet for connecting multiple sites. 363 nodes, 619 CPUs, 27 sites.	Workflow based distributed bioinformatics environment. BLAST search service. Genome annotation system. Biochemical network simulator.
Swiss BioGrid [67] http://www.swissbiogrid.org	NorduGrid's ARC and GridMP. heterogeneous hardware platforms including both clusters and Desktop-PC grids	High throughput compound docking into protein structure binding sites and analysis of proteomics data
Enabling Grids for E- sciencE (EGEE) [65] http://www.eu-egee.org	gLite middleware. 30,000 CPUs and 20 Petabytes storage. 20,000 concurrent jobs on average. 90 institutions in 32 countries.	WISDOM: drug discovery. GATE: radio therapy planning. SiMRI3D: parallel MRI simulator. GPS@: Grid Protein Sequence @Analysis and other applications
North Carolina BioGrid http://www.ncbiotech.org	Avaki data grid middleware and Virtual File System across grid nodes.	Bioinformatics datasets and applications installed on native file system and shared across the grid.

velopment of some sort of virtual application or workflows, whose components can run on multiple machines within heterogeneous and distributed environment. Despite of these difficulties there are several grid-enabled applications for life sciences [75], mostly developed by using either standard languages such as Java along with message passing interfaces (e.g. MPICH/G) or web services. A brief description of some major grid-enabled applications is presented in Table 2.3. Still there might be many legacy applications that could take advantage of grid based resources; however, the migration of these applications to grid environment requires more sophisticated tools than what is currently available [117].

TABLE 2.3: Some Major Grid-enabled Applications related to life sciences with the effect of gridificatiton

Grid-enabled application Task and source	**Grid middleware tools, services and languages**	**Effect of gridification**
GADU/GNARE [118] Task: Genome Analysis and Database Update http://compbio.mcs.anl.gov	Globus Toolkit and Condor/G for distributing DAG based workflows. GriPhyN Virtual Data System for workflow management. User interface to standard databases (NCBI, JGI etc.) and analysis tools (BLAST, PFAM etc.)	Analysis of 2314886 sequences on a single 2GHz CPU can take 1061 days A grid with an average of 200 nodes took only 8 days and 16 hours for the above task.
MCell [119] Task: Computational biology simulation framework based on Monte Carlo algorithm `http://www.mcell.cnl.salk.edu/`	Globus GRAM, SSH, NetSolve, PBS for remote job starting/monitoring. GrdiFTP and scp for moving application data to grid. Java based GUI, Relational Database (Oracle), Adoptive scheduling.	A typical r_disk MCell simulation on a single 1.5 GHz CPU can take 329 days A grid with an average of 113 dual CPU nodes took only 6 days and 6 hours for the above task.
Grid Cellware [120] Task: Modeling and Simulation for systems biology `http://www.cellware.org`	Globus, Apache Axis, GridX-Meta Scheduler. GUI based jobs creation editor Jobs mapped and submitted as web services.	Different stochastic (Gillespie, Gibson etc.), deterministic (Euler Forward, Runge-Kutta) and MPI based swarm algorithms have been successfully implemented in a way to distribute their execution on grid.

2.3.3 Grid-based BioPortals

As mentioned earlier, the actual process of developing and deploying an individual application on grid requires significant level of expertise and considerable period of time. This issue hinders the usage of available grid infrastructures. Therefore, in order to enhance the use of different grid infrastructures, some individuals, groups, institution and organizations have started to provide the most frequently used and standard domain specific resources as grid-enabled services which can be accessed by any or authenticated researcher through a common browser based single-point-of-access, without the need of installing any additional software. In this context a grid portal is considered to be an extended web-based application server with the necessary software capabilities to communi-

cate with the backend grid resources and services [121]. This type of environment provides full level of abstraction and makes it easy to exploit the potential of grid seamlessly. Grid-based portals are normally developed using some publicly available grid portal construction toolkits such as GridPort Toolkit [122, 123], NinF Portal Toolkit [121], GridSphere (`http://www.gridsphere.org`), IBM WebSphere (`www-306.ibm.com/software/websphere`) etc. Most of these toolkits follow the Java portlet specification (JSR 168) standard and thus make it easy for the developer to design the portal front-end and connect it to the backend resources through middleware services.

For example, [121] enables the developer to specify the requirements of the portal front-end (e.g. authentication, user interaction fields, job management, resources etc) in terms of a XML based file which automatically generates a JSP file (through Java based XML parser), that provides an HTML based web page for front-end and a general-purpose Java Servlets that can communicate to grid-enabled backend applications and resources though Globus [34, 35] based GridRPC mechanism. The toolkit also helps the developer for the gridification of applications and resources needed at the backend. It is because of this level of ease for the creation of grid-based portals that in [124] it is claimed that portal technology has become critical for future implementation of the bioinformatics grids.

Another example is that of BRIDGES (`http://www.brc.dcs.gla.ac.uk/projects/bridges`) project which provides portal-based access to many biological resources (federated databases, analytical and visualization tools etc) distributed across all the major UK centers with appropriate level of authorization, convenience and privacy. It uses IBM WebSphere based portal technology, because of its versatility and robustness. The portal provides a separate workspace for each user that can be configured by the user as per requirements, and the configuration settings are stored using

session management techniques. This type of environment can help in many important fields of life sciences such as the field of exploratory genetics that leads towards the understanding of complex disease phenotypes such as heart disease, addiction and cancer on the basis of analysis of data from multiple sources (e.g. model organism, clinical drug trials and research studies etc). Similarly [125], presents another instance of system that is easy to use, scalable and extensible, providing among others, secure, and authenticated access to standard bioinformatics databases and analysis tools such as nucleotide and protein databases, BLAST, CLUSTAL etc.

A common portal engine was developed with the reusable components and services from Open Grid Computing Environment Toolkit (OGCE) [126] that combine the components of three individual grid portal toolkits such as CompreHensive CollaborativeE Framework (CHCEF) (`http://adsabs.harvard.edu/abs/2002AGUFMOS61C..13K`), Velocity Toolkit (`http://jakarta.apache.org/velocity`) and JetSpeed Toolkit (`http://portals.apache.org/jetspeed-1`). This common portal engine was integrated with a biological application frame work by using PISE [127] (web interface generator for molecular biology). The portal provides access to around 200 applications related to molecular biology and also provides the way to add any other application through the description of a simple XML based file.

2.3.4 BioGrid Application Development Toolkits

Although some general purpose grid toolkits such as Globus [34, 35] , COSM (`http://www.mithral.com/projects/cosm`) and GridLab (`http://www.gridlab.org`), provide certain tools (APIs and run time environments) for the development of grid-enabled applications, they are primarily aimed at the provision of low level core services needed for the implementation of a grid infrastructure. Therefore, it seems to be difficult and time consuming for an ordinary programmer

to go through the actual process of developing and testing a grid enabled application using these toolkits; instead there are some simulation based environments such as EDGSim (http://www.hep.ucl.ac.uk/~pac/EDGSim), extensible grid simulation environment [128] ,GridSim [129] and GridNet [130], that could be used at initial design and verification stage.

As different application domains require certain specific set of tools that could make the actual process of grid-enabled application development life-cycle (development, deployment, testing and execution) to be more convenient and efficient. One such proposal for the development of a Grid Life Science Application Developer (GLAD) was presented in [131]. This publicly available toolkit works on top of the ALiCE (Adaptive scaLable Internet-based Computing Engine), a light weight grid middleware, and provides a Java based grid application programming environment for life sciences. It provides a list of commonly used bioinformatics algorithms and programs as reusable library components along with other software components needed for interacting (fetching, parsing etc) with remote distributed and heterogeneous biological databases. The toolkit also assists in the implementation of task level parallelism (by providing effective parallel execution control system) for algorithms and applications ranging from those having regular computational structures (such as database searching applications) or irregular patterns (such as phylogenetic tree) [28]. Certain limitations of GLAD include the non-conformance of AliCE with OGSA standard and the use of socket based data communication which might not be good for performance critical applications.

Another grid application development toolkit for bioinformatics that provides high level user interface with a problem solving environment related to biomedical data analysis has been presented in [132]. The toolkit provides a Java based GUI that enables the user to design a Direct Acyclic Graph (DAG) based workflow selecting a variety of bioinformatics tools and data (wrapped

as java based JAX-RPC web services) with appropriate dependencies and relationships.

2.3.5 Grid-based Problem Solving Environments

The Grid-based Problem Solving Environment (PSE) is another way of providing a higher level of interface such as graphical user interface or web interface to an ordinary user so that he/she could design, deploy and execute any grid-enabled application related to a particular class of specific domain and visualize the results without knowing the underlying architectural and functional details of the backend resources and services. In fact grid-based PSE brings the grid application programming at the level of drawing, that is, instead of writing the code and worrying about the compiling and execution, the user can just use appropriate GUI components provided by the PSE to compose, compile and run the application in a grid environment. PSEs are developed using high level languages such as Java and are targeted to transforms the user designed/modeled application into an appropriate script (distributed application or web service) that could be submitted to a grid resource allocation and management service for execution and on completion of the execution the results are displayed through appropriate visualization mechanism.

There are several different grid-based PSEs available for bioinformatics applications e.g. [133] describes the design and architecture of a PSE (Proteus) that provides an integrated environment for a biomedical researchers to search, build and deploy distributed bioinformatics applications on computational grids. The PSE uses semantic based ontology (developed in DAMIL+OIL language (`http://www.daml.org`)) to associate the essential meta-data such as goals and requirements to three main classes of bioinformatics resources such as data sources (e.g SwissProt and PDB database), software components (e.g BLAST, SRS, Entrez and EMBOSS an open source suite of bioinformatics applications for sequence analysis) and tasks/processes (e.g. sequence alignment,

secondary structure prediction and similarity comparison) and stores this information in a meta-data repository. The data sources are specified on the basis of kind of biological data, its storage format and the type of the data source. Similarly, the components and tasks are modeled on the basis of the nature of tasks, steps and order in which tasks are to be performed, algorithm used, data source and type of the output etc. On the basis of this ontology the PSE provides a dictionary (knowledge-base) of data and tools locations allowing the users to compose their applications as workflows by making use of all the necessary resources without worrying about their underlying distributed and heterogeneous nature. The modeled applications are automatically translated into grid execution scripts corresponding to GRSL (Globus Resource Specification Language) and are then submitted for execution on grid through GRAM (Globus Resource Allocation Manager). The performance of this PSE was checked with a simple application that used TribeMCL (`http://www.ebi.ac.uk/research/cgg/tribe`), for clustering human protein sequences which were extracted from SwissProt database by using seqret program of the EMBOSS suite and compared all against all for similarity though BLAST program. In order to take advantage of the grid resources and enhance the performance of similarity search process the output of seqret program was split into three separate files in order to run three instances of BLAST in parallel. The individual BLAST outputs were concatenated and transformed into a Markov Matrix required as input for TribeMCL. Finally the PSE displayed the results of clustering in a opportune visualization format. It was observed that total clustering process on grid took 11h50’53” as compared to 26h48’26’ on standalone machine. It was also noted on the basis of another experimental case (taking just 30 protein sequences for clustering) that the data extraction and result visualization steps in the clustering process are nearly independent of the number of protein sequences (i.e. approximately same time was observed in the

case of all protein Vs 30 protein sequences). Another PSE for bioinformatics has been proposed in [134]. It uses Condor/G for the implementation of PSE that provides an integrated environment for developing component based workflows through commonly used bioinformatics applications and tools such as Grid-BLAST, Grid-FASTA, Grid-SWSearch, Grid-SWAlign and Ortholog-Picker etc. Condor/G is an extension to grid via Globus and it combines the inter-domain resource management protocols of Globus Toolkit with intra-domain resource management methods of Condor to provide computation management for multi-institutional grid. The choice of Condor/G is justified on the basis of its low implementation overhead as compared to other grid technologies. The implementation of a workflow based PSE is made simple by the special functionality of Condor meta-scheduler DAGMan (Directed Acyclic Graph Manager) which supports the cascaded execution of programs in a grid environment. The developed prototype model was tested by integrated (cascaded) execution of the above mentioned sequence search and alignment tools in grid environment. In order to enhance the efficiency, the sequence databases and queries were split into as much parts as the number of available nodes, where the independent tasks were executed in parallel.

2.3.6 Grid-based Workflow Management Systems

As already discussed in the context of PSE, a workflow is a process of composing an application by specifying the tasks and their order of execution. A grid-based workflow management system provides all the necessary services for the creation, execution and visualization of the status and results of the workflow in a seamless manner. These features make workflows ideal for the design and implementation of life science applications that consists of multiple steps and require the integrated access and execution of various data and application resources. Therefore one can find various domain specific efforts for the development of proper workflow management systems for

life sciences (Table 2.4). There have been several important demonstrations of different types of life science applications on grid-based workflow management systems. For example, the design and execution of a tissue-specific gene expression analysis experiment for human has been demonstrated in a grid-based workflow environment called 'WildFire' [135]. The workflow takes as an input 24 compressed GeneBank files corresponding to 24 human chromosomes and after decompression it performs exon extraction (through exonx program) from each file in parallel resulting in 24 FASTA files. In order to further increase the level of granularity each FASTA file is split into five sub-files (through dice script developed in Perl), making a total of 120 small files ready for parallel processing with BLAST against a database of transcripts ('16, 385 transcripts obtained from Mammalian Gene Collection'). The execution of this experiment on a cluster of 128 Pentium III nodes took about 1 hour and 40 minutes, which is reported to be 9 times less than the time required for the execution of its sequential version. The iteration and dynamic capabilities of WildFire has also been demonstrated through the implementation of a swarm algorithm for parameter estimation problem related to biochemical pathway model based on 36 unknowns and 8 differential equations. Similarly, the effectiveness of Taverna [136] workflow has been demonstrated by construction of a workflow that provides genetic analysis of the Graves' disease. The demonstrated workflow makes use of Sequence Retrieval System (SRS), mapping database service and other programs deployed as SoapLab services to obtain information about candidate genes which have been identified through Affymetrix U95 microarray chips as being involved in Graves' disease. The main functionality of the workflow was to map a candidate gene to an appropriate identifier corresponding to biological databases such as Swiss-Prot and EMBL in order to retrieve the sequence and published literature information about that gene through SRS and MedLine services. The result of tBLAST search

against the PDB provided identification of some related genes where as the information about the molecular weight and isoelectric point of the candidate gene was provided by the Pepstat program of EMBOSS suite. Similarly, Taverna has also been demonstrated with the successful execution of some other workflows for a diversity of in-silico experiments such as pathway map retrieval and tracking of data provenance.

TABLE 2.4: Examples of commonly used Grid-based workflow management systems for life sciences

Workflow management system	**Supported grid middleware technologies and platforms**	**Main features**
Wildfire [135] http://wildfire.bii.a-star.edu.sg/	Condor/G, SGE, PBS, LSF Workflows are mapped into Grid Execution Language (GEL) script. Open source and extensible. Runs on Windows and Linux.	GUI-based drag-and-drop environment. Workflow construction by EMBOSS. Complex operations (iteration and dynamic parallelism).
Taverna [136] http://taverna.sourceforge.net/	myGrid middleware SCULF language for workflows. Workflows mapped to web services Open source and extensible. Cross platform.	GUI-based workbench. In-silico experiments using EMBOSS, NCBI, EBI, DDBj, SoapLab, BioMoby and other web services.
ProGenGrid [137] http://www.cact.unile.it/projects/	Globus Toolkit4.1 GridFTP and DIME for data. iGrid information service for resource and web service discovery. Java Axis and gSOAP Toolkit.	UML-based editor. RASMOL for visualization. AutoDoc for drug design.

2.3.7 Grid-based Frameworks

In software development, a framework specifies the required structure of the environment needed for the development, deployment, execution and organization of a software application/project related to a particular domain in an easy, efficient, standardized, collaborative, future-proof and seamless manner. When becoming fully successful and widely accepted and used, most of these frame-

works are also made available as Toolkits. There are some general frameworks for grid-based application development such as Grid Application Development Software (GrADS) [138], Cactus [139] and IBM Grid Application Development Framework for Java (GAF4J (http://www.alphaworks.ibm.com/tech/GAF4J)). Similarly, some specific grid-based frameworks for life sciences have also been proposed and demonstrated such as Grid Enabled Bioinformatics Application Framework (GEBAF) [140], that proposes an integrated environment for grid-enabled bioinformatics application using a set of open source tools such as Bioperl Toolkit [141], Globus Toolkit [34,35], Nimrod/G [142] and Citrina (database management tool (http://www.gmod.org/citrina)). The framework provides a portal based interface that allows the user to submit a query of any number of sequences to be processed with BLAST against publicly available sequence databases. The user options are stored in a hash data structure by creating a new directory for each experiment and a script using the BioPerl::SeqIO module divides the user query into sub-queries each consisting of just a single sequence. The distributed query is then submitted through Nimrod/G plan file for parallel execution on the grid. Each grid node maintains an updated and formatted version of the sequence database through Citrina. The individual output of each sequence query is parsed and concatenated by another script that generates the summary of the experiment in the form of an XML and Comma Separated Value (CSV) files containing the number of most significant hits from each query. The contents of these files are then displayed through the result interface. A particular demonstration for BLAST was carried out with 55,000 sequences against SwissProt database.

With 55,000 parallel jobs the grid has been fully exploited within the limits of its free nodes and it has been observed that the job management overhead was low as compared to the actual search time for BLASTing of each sequence. Although summarizing thousands of results

is somewhat slow and nontrivial, its execution time remains insignificant when compared with the experimenting time itself. The developed scripts were also tested for reusability with other similar applications such as ClustalW and HMMER, with little modification. A web service interface has been proposed for future development of GEBAF in order to make use of other bioinformatics services such as Ensemble. Similarly, for better data management Storage Resource Broker (SRB) middleware is also proposed as an addition for the future.

GEBAF is not the only grid-enabling framework available. For example, Asim et al. [143] describes the benefits of using the Grid Architecture Development Software (GrADS) framework for the gridification of bioinformatics applications. Though there already existed an MPI-based master-slave version of the FASTA but it used a different approach: it kept the reference database at the master side and made the master responsible for equal distribution of database to slaves and the subsequent collection and concatenation of the results. In contrast to that, GrADS based implementation makes reference databases (as a whole or as a portion) available at some or all of the worker nodes through database replication. Thus the master at first sends a message to workers for loading their databases into memory and then it distributes the search query and collects the results back. This type of data-locality approach eliminates the communication overhead associated with the distribution of large scale databases. Furthermore, through the integration of various software development and grid middleware technologies (such as Configurable Object Program, Globus Monitoring and Discovery Service (MDS) and Network Weather Service (NWS)), GrADS framework provides all the necessary user, application and middleware services for the composition, compilation, scheduling, execution and real time monitoring of the applications on a grid infrastructure in a seamless manner.

2.3.8 BioGrid Data Management Approaches

Most of the publicly available biological data originates from different sources of information, i.e. it is heterogeneous and is acquired, stored and accessed in different ways at different locations around the world, i.e. it is distributed. The heterogeneity of data may be syntactic i.e. difference in file formats, query languages and access protocols etc, semantic i.e. genomic and proteomic data etc, or schematic i.e. difference in the names of database tables and fields etc. In order to make an integrative use of these highly heterogeneous and distributed data sources in an easy and efficient way, an end-user biologist can take advantage of some specific Data Grid infrastructure and middleware services such as BRIDGES (`http://www.brc.dcs.gla.ac.uk/projects/bridges`), BIRN (`http://www.nbirn.net`) and various other European Union Data Grid projects e.g. EU-DataGrid (`http://www.edg.org/`) and EU-DataGrid for Italy (`http://web.datagrid.cnr.it/Tutorial_Rome`) etc. These Data Grid projects make use of standard middleware technologies such as Storage Resource Broker (SRB), OGSA-DAI and IBM Discovery Link.

2.3.9 Computing and Service Grid Middleware

In the same way that a computer operating system provides a user-friendly interface between user and computer hardware similarly Grid middleware provides important services needed for easy, convenient and proper operation and functionality of grid infrastructure. These services include access, authentication, information, security and monitoring services as well as data and resource description, discovery and management services. In order to further reduce the difficulties involved in the process of installation, configuration and setting-up of grid middleware, there have also been proposals for the development of Grid Virtual Machines (GVM) and Grid Operating Systems [144–

148]. The development of specific grid operating systems or even embedding grid middleware as a part of existing operating systems would greatly boost-up the use of grid computing in all computer related domains, but this is yet to be seen in the future. The important features of currently used computing and service grid middleware are listed in Table 2.5.

TABLE 2.5: Commonly used computing and service grid middleware

Grid middleware	**Brief description of architecture and services**
Globus Toolkit GT4 [34, 35] Goal: To provide a suit of services for job, data, and resource mgt. Developer: Argonne National Laboratory, University of Chicago http://www.globus.org/	OGSA-WSRF based architecture Credential management services (MyProxy, Delegation, SimpleCA) Data management services (GridFTP, RFT, OGSA-DAI, RLS, DRS) Resource management services (RSL and GRAM) Information and monitoring services (Index, Trigger and WebMDS) Instrument management services (GTCP)
LCG-2/ gLite Goal: To provide Large scale data handling and compute power infrastructure. Developer: EGEE project in collaboration with VTD, US and partners. http://glite.web.cern.ch/glite/	LCG-2: a pre-web service middleware based on Globus 2 gLite: an advanced version of LCG-2 based on web services architecture Authentication and security services (GSI, X.509, SSL, CA) Information and monitoring services (Globus-MDS, R-GMA, GIIS, BDII) Resource management services (GUID, SURL) Data management services (WMS, SLI) Platform: Linux and Windows
UNICORE Goal: Light weight grid middleware. Developer: Fujitsu Lab EU and UniGrid. http://www.unicore.eu/	UNICORE: a pre-web service grid middleware based on OGSA standard UNICORE 6: based on web service and OGSA architecture. Security services (X5.09, CA, SSL/TLS) Execution management engine (XNJS) Platform: Unix/Linux platform Data management services (WMS, SLI) Platform: Linux and Windows

2.3.10 Local Resource Management System (LRMS)

The grid middleware interacts with different clusters of computers through Local Resource Management System (LRMS) also known as Job Management System (JMS). The LRMS (such as Sun Grid Engine, Condor/G and Nimrod/G) is responsible for submission, scheduling and monitoring of jobs in a local area network environment and providing the results and status information to the

grid middleware through appropriate wrapper interfaces. Some of the important features [149] of commonly used LRMS software are listed in Table 2.6.

TABLE 2.6: Commonly used Job Management Systems (Local Resource Management Systems (LRMS))

Local Resource Manager	**General features** **Platform, GUI and APIs**	**Job support** **description, type and MPI support**
Sun Grid Engine 6 Sun Micro Systems `gridengine.sunsource.net`	Platform: Solaris, Apple Macintosh, Linux and Windows User friendly GUI, portal and DRAMA API. Open source and extensible. Integration with globus through GE-GT Adopter	Shell scripts for job description. Standard and complex job types. Integration with MPI. 5 Million jobs on 10,000 hosts.
Condor-G University of Wisconsin `www.cs.wisc.edu/condor`	Platform: Solaris, Apple Macintosh, Linux and Windows DRAMA and Web-service interface Open source and extensible. Globus-enabled.	Job description: Classified Advertisements Standard and complex job types Integration with MPI
Nimrod-G 3.0.1 Monash University `www.csse.monash.edu.au/`	Platform: Solaris, Linux, Mac with x86 and sparc architecture web portal and API Open source and extensible. Globus-enabled.	Job description: Nimrod Agent Language GRAM interfaces to dispatch jobs to computers.

2.3.11 Fault Tolerant Aproaches in the Context of BioGrid

Like any other Grid computing infrastructure, the BioGrid environment is considered to be dynamic [150]. In the context of a dynamic BioGrid infrastructure, the availability and constraints of resources keeps changing with respect to time. The capability of the application and the system as a whole to withstand the effect of change in the state of resources and continue its normal functionality/execution is known as *fault tolerance*. Though initially fault tolerance was not addressed much in the context of BioGrid as the very idea of Grid itself was under the phase of *'proof of concept'*. However, now that the Grid has moved towards the era of robust standardization (post 2005

web-service based architecture), one could find many approaches for the implementation of fault tolerance at different levels of BioGrid infrastructure. For example, Ying Sun et. al. [151], make use of *backup task* mechanism to add fault tolerance to their application for bioinformatics computing grid (ABCGrid). The *backup task* approach takes its inspiration from Google's MapReduce Model [152], and performs monitoring of all the tasks in progress. It uses the results of monitoring to determine if any task has not finished in its normal (expected) time on any node (due to failure/poor performance of that node/resource) and assigns the same task (as a backup task) to other node. When the task returns its results (either from the primary execution or the backup execution), the task's status gets changed to 'completed' and all remaining backup executions get terminated. The authors demonstrate evaluation of their application (using bioinformatics tools such as NCBI BLAST, Hmmpfam and CE [47]) on a testbed consisting of 30 workstations connected in a local environment. They mainly report on the speedup of the application and do not present/discuss any results in terms of fault tolerance. It should also be noted that though the authors label their application with the term 'Grid' but they do not report any use of the Grid. Also, the method they use for fault tolerance could not be applied in general because because of two concerns. First this approach requires that the normal execution time of the tasks be known in advance, which might not be the case in many applications which either use heuristic based computational techniques or make use of data with varying size etc. Second, though this approach might be appropriate for fine-grained tasks but for heavily coarse-grained tasks whose normal execution may take very long, such a approach would result in very poor performance. Nevertheless, this approach might be tested in a real grid environment with different applications. Stockinger H. et. al. [153] also use similar approach but no statistics on the quality of fault tolerance is reported. For workflow based applications, the

myGrid [69] project provides checkpoint/rollback interfaces which could be implemented to develop applications/services with inherent fault tolerant support. Similarly, for applications based on parallel programming model, various MPI implementations provide mechanism for the implementation of fault tolerance e.g., MPICH-V [154], MPI-FT [155], and Berkeley Lab Checkpoint/Restart (BLCR) [156] etc. All of these implementations provide some sort of checkpointing and message logging mechanism which could be used by the applications to migrate/restart any failed process/-task. The message logging enables the application to restart the computation from the previous fault-safe state and hence not causing much overhead in terms of failure.

2.4 Some Flagship BioGrid Projects

We present here some selected flagship case studies which have elicited a positive public response from bio-scientists for their special role and contribution to the life science domain. The description of most important implementation strategies along with some main services is provided with the help of appropriate illustrations.

2.4.1 EGEE Project

The EGEE project was initially named as Enabling Grid for E-science in Europe and then it was renamed as Enabling Grid for E-scienceE) in order to enhance its scope from European to international level [65]. This project builds on its predecessor EDG project (European Data Grid) and provides an international level grid infrastructure for multi-science and industry community ranging from high-energy physics to life sciences and nanotechnology. The overall infrastructure consists of more than 30,000 CPUS with 20 petabytes of storage capacity provided by various academic institutes and other organizations and industries around the world in the form of high-

speed and high-throughput compute clusters which are being updated and interoperated through its web-service based light-weight, more dynamic and inter-disciplinary grid middleware named gLite. Like EGEE, gLite also builds on a combination of various other grid middleware projects such as LCG-2 (http://cern.ch/LCG), DataGrid (`http://www.edg.org`), DataTag (http://cern.ch/datatag), Globus Alliance (`http://www.globus.org`), GriPhyN (`http://www.griphyn.org`) and iVDGL (`http://www.ivdgl.org`). Several bio-applications have been implemented on top of EGEE platform [108] and various other resource-hungry biological projects (such as BioInfoGrid (`http://www.bioinfogrid.eu/`), Wide In Silico Docking On Malaria (WISDOM) (http://wisdom.eu-egee.fr) and European Model for Bioinformatics Research and Community Education (EMBRACE) (`http://www.embracegrid.info`)) are also continuously making use of EGEE infrastructure. In order to provide an illustration and understanding of how to make use of EGEE grid infrastructure and services for life sciences, we provide here an overview of the latest version of its grid middleware.

Authentication and Security Services:

EGEE uses Grid Security Infrastructure (GSI) for authentication (through digital X.509 certificate) and secure communication (through SSL: Secure Socket Layer protocol with enhancements for single sign-on and delegation). Therefore, in order to use the EGEE grid infrastructure resources, the user has to register first and get a digital certificate from appropriate Certificate Authority (CA). When the user signs in with the original digital certificate which is protected with a private key and a password, the system then creates another passwordless temporary certificate called proxy certificate that is then associated with every user request and activity. In order to maintain the user security at high level, the proxy certificates are kept valid for small intervals (default 12 hours), however, if user jobs require more time then appropriate age of the proxy certificate can also be set

though My Proxy Server.

Information and Monitoring Services:

The gLite 3 uses Globus MDS (Monitoring and Discovery Service) for resource discovery and status information. Additionally, it uses Relational Grid Monitoring Architecture (R-GMA) for accounting and monitoring. In order to provide more stable information services, the gLite Grid Information Indexing Server (GIIS) uses BDII (Berkeley Database Information Index Server) that stores data in more stable manner than original Globus based GIIS.

Data Management Services:

Like in traditional computing the primary unit of data management in EGEE grid is also the file. gLite provides a location independent way of accessing files on EGEE grid through use of Unix based hierarchical logical file naming mechanism. When a file is registered for the first time on the grid it is assigned a GUID (Grid Unique Identifier that is created from User Unique Identifier; MAC address and a time stamp) and it is bound with an actual physical location represented by SURL (Storage URL). Once a file is registered on the EGEE grid it cannot be modified or updated because the data management system creates several replicas of the file in order to enhance the efficiency of subsequent data access. Thus, updating any single file would create the problem of data inconsistency which has not as yet been solved in EGEE.

Workload Management System (Resource Broker):

The new gLite based work load management system (WMS or resource broker) is capable of receiving even multiple inter-dependent jobs described by Job Description Language (JDL) and it

dispatches these jobs to most appropriate grid sites (selection of appropriate grid site is based on the dynamic process of match-making) and then keeps tract of the status of the jobs and retrieves the results back when jobs are finished. While dispatching the jobs the resource broker uses Data Location Interface (DLI) service to supply input files along with job to the worker node. The flow of job is illustrated in Figure 2.7.

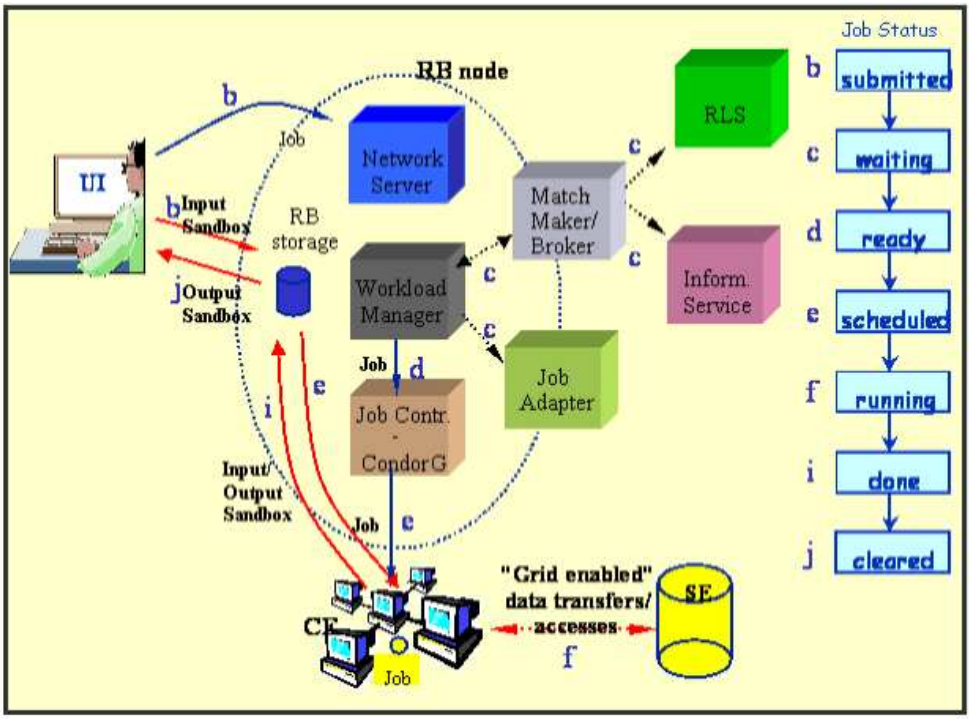


FIGURE 2.7: Job flow in the EGEE grid. The user request goes to the resource broker node (RB node) which assigns the job to a particular node based on the results of the match maker, work load manager and other status information. (reproduced from `http://glite.web.cern.ch/`

2.4.2 Organic Grid: Self Organizing Computational Biology on Desktop Grid

The idea of Organic Grid [157] is based on the decentralized functionality and behavior of self organizing, autonomous and adaptive organisms (entities) in natural complex systems. The examples of natural complex systems include functioning of biological systems and behavior of social insects

such as ant and bees. The idea of the organic grid leads towards a novel grid infrastructure that could eliminate the limitations of traditional grid computing. The main limitation of traditional grid computing lies in their centralized approach. For example, a Globus based computational grid may use a centralized meta-scheduler and thus it would be limited to smaller number of machines only. Similarly, Desktops Grid Computing based on distributed computing infrastructure such as BOINC may use centralized master/slave approach and thus would be only suitable for coarse-grained independent jobs only. The idea of Organic Grid is to provide a ubiquitous type peer-to-peer grid computing model capable of executing arbitrary computing tasks on a very large number of machines over network of any quality, by redesigning the existing desktop computing model in a way that it supports distributed adaptive scheduling through the use of mobile agents. In essence it means that, a user application submitted on such type of architecture would be encapsulated in some type of a mobile agent containing the application code along with the scheduling code. After encapsulation, the mobile agent can decide itself (based on its scheduling code and the network information) to move to any machine that has appropriate resources needed for the proper execution of the application. This type of mechanism provides the same type of user-level abstractness as provided by traditional Globus-based grid but additionally it builds on decentralized scheduling approach that enables the grid to span to very large number of machines in a more dynamic peer-to-peer computing model. The use of mobile agents (which are based on RMI mechanism that is built on top of client/server architecture) as compared to their alternate service based architecture, provides higher level of ease and abstractedness in terms of validation (experimentation) of different types of scheduling, monitoring and migration schemes. Although the project uses a scheduling scheme that builds on tree-structured overlay network, it is made adaptive based on

some value of application specific performance metric. For example, the performance metric for a data-intensive application such as BLAST would give high consideration to bandwidth capacity of the communication link before actually scheduling the job on a particular node. Similarly, it will select a high-speed node for another application that comes under the class of compute-intensive applications. Furthermore, in order to provide uninterrupted execution with dynamic and transparent migration features the project makes use of strongly mobile agents instead of traditional weakly mobile agents (Java based mobile agents that cannot access their state information). Following the common practice in grid computing research, the proof-of-concept has been demonstrated with the execution of NCBI BLAST (that falls in the class of independent task application) on a cluster of 18 machines with heterogeneous platform and ranked under the categories of fast, medium and slow through the introduction of appropriate delays in the application code. The overall task required the comparison of a 256 KB sequence against a set of 320 data chunks each of size 512 KB. This gave rise to 320 sub tasks, each responsible for matching the candidate 256 KB sequence against one specific 512 KB data chunk. The project successfully carried out the execution of these tasks and it has been observed that by adopting the scheduling according to the dynamics of the architecture greatly improves performance and quality of results. The project is being further extended to provide the support for different categories of applications and enabling the user to configure different scheduling schemes for different applications through some easy to use APIs.

2.4.3 Advancing Clinico-Genomic Trials on Cancer (ACGT)

ACGT is a Europe wide integrated biomedical grid for post-genomic research on cancer (`http://www.eu-acgt.org`) [158]. It intends to builds on the results of other biomedical grid projects such as caBIG, BIRN, MEDIGRID and myGrid. The project is based on open source and open

access architecture and provides basic tools and services required for medical knowledge discovery, analysis and visualization. The overall grid infrastructure and services are aimed to provide an environment that could help scientists to: a) Reveal the effect of genetic variations on oncogenesis b) Promote the molecular classification of cancer and development of individual therapies c) Modeling of in-silico tumor growth and therapy response. In order to create the required environment that supports the implementation of these objectives, ACGT focuses on the development of a virtual web that interconnects various cancer related centers, organizations and individual investigators across the Europe through appropriate web and grid technologies. Mainly, it uses semantic web and ontologies for data integration and knowledge discovery and Globus toolkit with its WS-GRAM, MDS and GSI services for cross organization resource sharing, job execution, monitoring and result visualization. Additionally, ACGT also uses some higher level grid services from Gridge framework developed at Poznan Supercomputing and Networking Centre (PSNC) [159]. These services include GRMS (Grid Resource Management System), GAS (Grid Authorization System) and DMS. These additional services provide the required level of dynamic and policy-based resource management; efficient and reliable data handling; and monitoring and visualization of results. Figure 2.8 provides a usage scenario of these services in the context of ACGT environment.

2.5 Conclusions

This part of the review has presented a scrupulous analysis of the state-of-the-art in web and grid technology for bioinformatics, computational biology and systems biology in a manner that provides a clear picture of currently available technological solutions for a very wide range of problems. While surveying the literature, it has been observed that there are many grid-based PSEs, Workflow

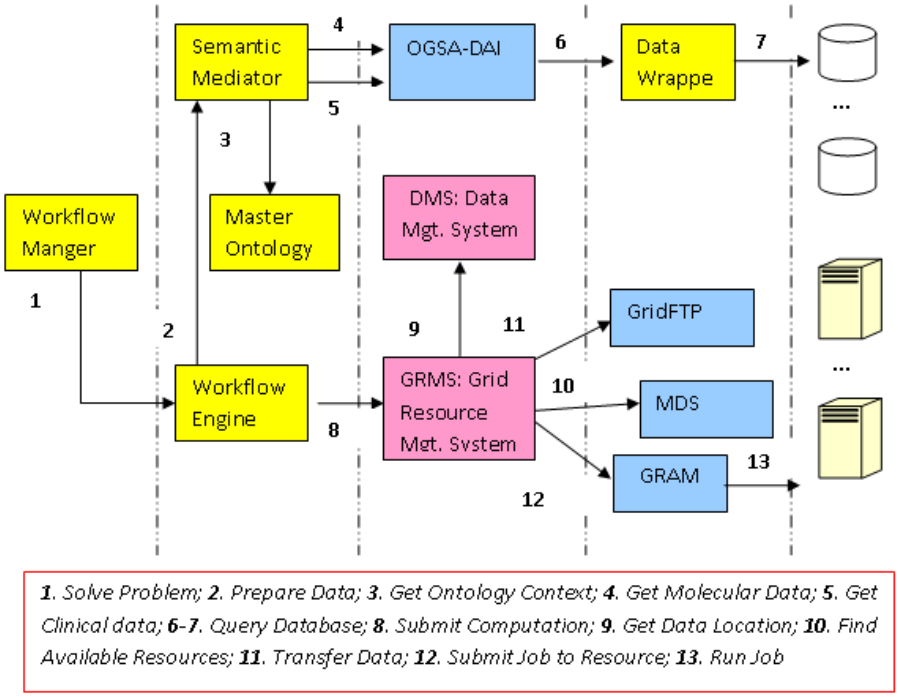


FIGURE 2.8: ACGT integrated environment usage scenario. The system goes from step 1 to 13 in order to run a particular job. (reproduced from [158])

Management Systems, Portals and Toolkits under the name of Bioinformatics but not as many for Systems or Computational Biology. However, in each case a mix of projects and applications has been found overlapping from bioinformatics to computational and systems biology. Based on the analysis of the state-of-the-art we identify bellow some key open problems can be distinguished:

- The use of semantic web technologies such as domain ontologies for life sciences is still not at its full level of maturity, perhaps because of semi-structured nature of XML and limited expressiveness of ontology languages [68].

- Biological data analysis and management is still quite a difficult job because of the lack of development and adaptation of optimized and unified data models and query engines.

- Some of the existing bioinformatics ontologies and workflow management systems are simply

in the form of Directed Acyclic Graphs (DAGs) and their descriptions are lacking expressiveness in terms of formal logic [135].

- Lack of open-source standards and tools required for the development of thesaurus and meta-thesaurus services [77].
- Need of appropriate query, visualization and authorization mechanism for the management of provenance data and meta-data in in-silico experiments [68, 135].
- Some of the BioGrid projects seem to be discontinued in terms of information updating. This might arise from funding problems or difficulties associated with their implementation.
- There is a lack of domain specific mature application programming models, toolkits and APIs for grid-enabled application development, deployment, debugging and testing.
- Still there seems to be a gap between the application layer and middleware layer of a typical BioGrid infrastructure because existing middleware services do not fully facilitate the demands of applications such as there is no proper support in any grid middleware for automatic application deployment on all grid nodes.
- It is not trivial to deploy existing bioinformatics applications on available grid testbed (such as NGS, EGEE etc), as this requires the installation and configuration of specific operating system and grid middleware toolkits, which is not at least easy from a biologist end-user point of view.
- It has been observed that there are still many issues with grid based workflow management systems in terms of their support for complex operations (such as loops), legacy bioinformatics applications and tools, use of proper ontology and web services etc. [135].

- The job submission process on existing grid infrastructures seems to be quite complex because of inappropriate maturity of resource broker services.

- Lack of appropriate implementation initiative regarding knowledge grid infrastructure for life sciences.

Some of the key lessons/findings learned from the review presented in this chapter applicable in the context of MC-PSC are:

- Identification of the availability of resources at national and international level. For example, the successful deployment of several bioinformatics related applications on the UK National Grid Service (NGS) and the European Grid for EScience-E (EGEE) provided an impetus that these resources could also be used for the case of MC-PSC. This really helped in getting access to NGS through its various training programs target at grid-based application development.

- Identification of several methods and tools for setting up a local, national or international grid infrastructure. The review provided a comprehensive picture of the available technological avenues which could be selected for further analysis with MC-PSC. In this regard different resource management and parallel environments were taken as a case study for evaluation with MC-PSC as reported in the following chapters.

- Identification of the key features/characteristics for the development, deployment and evaluation of grid-enabled application. For example, the literature presented in this chapter explains the pros and cons of the level of granularity in terms of work load distribution and corresponding communication overhead. It also helps in identifying the key performance measures

which could be used for the evaluation of distributed approaches for MC-PSC.

- Identification of appropriate web technologies which could be used in future to develop user-friendly web interfaces for the analysis and visualization of the MC-PSC similarity results.

Because of the wide scope of this review, the literature that has been reported in this chapter is not yet complete, and therefore, the next chapter provides a more focused review/survey of the literature that is closely related to the field of structural proteomics and hence informs more on the subject of this thesis.

CHAPTER 3

OVERVIEW OF GRID AND DISTRIBUTED PUBLIC COMPUTING SCHEMES FOR STRUCTURAL PROTEOMICS

As described in the previous chapter, grid and distributed computing (such as public computing schemes) has become an essential tool for many scientific fields including bioinformatics, computational biology and systems biology. The adoption of these technologies has given rise to a wide range of projects and contributions that provide various ways of setting up these environments and exploiting their potential resources and services for different domains of applications. This chapter aims to further extend the survey presented in previous chapter by specifically focusing on some of the major projects, technologies and resources employed in the area of *Structural Proteomics*. The major emphasis would be to briefly comment on various approaches related to the gridification and parallelization of some flagship legacy applications, tools and data resources related to key structural proteomics problems such as protein structure prediction, folding, and comparison. The comments are based on theoretical analysis of some interesting parameters such as performance gain after gridification, user level interaction environments, workload distribution and the choice of deployment infrastructure and technologies. The study of these parameters would provide a ba-

sis for some motivating justification needed for further research and development on the subject of this thesis i.e., the case of *Protein (Structure) Comparison, Knowledge, Similarity and Information* (ProCKSI).

Parts of this chapter were published as a peer reviewed conference paper in the *Proceedings of the Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, LNCS Vol.4743 pp.424-434, 2007. [doi:10.1007/978-3-540-74767-3_44]

3.1 Introduction

It is believed that, the rapidly evolving field of Structural Proteomics have played a role in the 1^{st} decade of 21^{st} century in terms of protein 3D structure determination and analysis that is quite equivalent to the role played by the *Human Genome Project* (HGP) in the last decade of 20^{th} century, in terms of sequence determination and analysis [160]. This is mainly because a very large number of protein primary structures (sequences) are known but the number of their corresponding 3D-structures (secondary or tertiary structures) is lagging far behind. For example, as of writing of this dissertation there are $12,347,303$ known protein sequences (UniProtKB/TrEMBL entries) as compared to just $64,036$ protein structures (PDB holdings). The reason behind this sequence-structure gap is due to the difficulties associated with experimental structure determination methods such as X-ray crystallography and NMR spectroscopy. As secondary and tertiary structures are more helpful in tracing the evolution and function of the protein as well as in rational drug design, in order to reduce the gap between known sequences and known structures, computational approaches have been proposed for the prediction of these structures from a given protein sequence. As all these approaches are based on some form of modeling (such as

ab-initio or de-novo protein modeling and comparative protein modeling techniques such as homology modeling and protein threading etc) and rely on multi-scale optimization techniques to optimize various model parameters (e.g. energy minimization), the availability of powerful computing facilities is essential. That is, structural proteomic methodologies require huge computational power and reliable access to various distributed and (often) heterogeneous biological databases and analytical tools in order to properly and accurately predict the structure from a given sequence or compare thousands of models against a target structure. Therefore, many research groups in this field such as Baker Laboratory at University of Washington (http://depts.washington.edu/bakerpg/), The Scripps Research Institute (TSRI) at California (http://www.scripps.edu/e_index.html), Pande Group at Stanford University (http://folding.stanford.edu/Pande/Main), High Throughput Computing Group at Osaka University [64] and others have started to make use of the Grid and distributed computing environments. While the grid-based projects make use of standard middleware services and institutionally-owned resources, the projects based on distributed public computing schemes build their infrastructure setup around publicly-owned unused computing resources which are voluntarily provided throughout the world such as the World Community Grid (http://www.worldcommunitygrid.org/) that supports the Human Proteome Folding Project, Folding@Home [161], Predictor@Home [162] and Rosseta@Home (http://boinc.bakerlab.org/rosetta/) etc.

This chapter focuses on some of these projects in order to find out and compare various approaches related to the gridification/parallelization of some flagship legacy applications, tools and data resources by analyzing key parameters such as job/data distribution and management, user level interaction environments, deployment technologies and infrastructures, and the effect of

gridification on overall performance of the system.

The organization of this chapter is as follows: sections 3.2 and 3.3 provide some detailed overview in the fields of protein folding and prediction respectively; while section 3.4 besides presenting the review of the literature, discusses in detail the main topic of this dissertation i.e., *Protein Structure Comparison*, its current state and future directions in terms of the "The ProCKSI Server: a decision support system for Protein (Structure) Comparison, Knowledge, Similarity and Information"; finally, section 3.5 concludes this chapter.

3.2 Protein Folding

The process of folding, first described by Anfinsen [5–7], is a thermodynamically driven process taking a few micro seconds, in which a protein adopts its native state. Failure of this process results in several lethal diseases in human and animal [163–165]. A proper understanding of this process sheds light into many issues at the core of biotechnology, such as the design of new proteins with a desired functionality, the understanding of some incurable diseases such as cancer or neurodegenerative diseases (e.g. Alzheimer's, Creutzfeldt-Jakob disease(CJD), Cystic fibrosis (CF), Huntington disease (HD) and many other practical implementations of nanotechnology. To this aim, several models have been established. These models make use of simulation based computational techniques that require extremely high computational power, far beyond the limits of any single traditional super computer or local cluster. It has been demonstrated in the Folding@Home project [161] that this requirement can be met with a world wide distributed public-resource computing network that interconnects thousands of loosely coupled heterogeneous publicly-owned and voluntarily devoted PCs. Folding@Home uses an 'ensemble dynamics' algorithm that performs

M independent simulations with the same amino acids coordinates but with different velocities on M distributed processors such that each simulation starts with a slightly different initial condition and pushes the system through a free energy minimization process. This algorithm gives an M times speedup for the simulation of folding dynamics and thus avoids the overall waiting in free energy minima. Similarly, the process can be repeated in order to effectively handle multiple free energy barriers (multiple translations for complex folding dynamics). Using a modified version of the Tinker molecular dynamics code, β -hairpin and villin were simulated and their folds successfully determined. Based on the diversity of the simulation results for a variety of molecules (from the non-biological PPA helices to the 36-residue villin headpiecea [166]) it has been observed that there is no single universal folding process and even sequences which fold to the same structure may have different folding processes. Further details on a selection of grid-enabled protein folding applications are presented in Table 3.1.

3.3 Protein Structure Prediction

Based on Anfinsen theory of protein folding [7], the aim of protein structure prediction is to predict the native conformation (tertiary structure) of a protein from its primary structure (sequence). It has remained an open problem in the field of structural proteomics [169]. New methods are being explored and investigated at various research institutes and groups throughout the world. Evaluation of the quality and performance of these methods is carried out every two years through the Critical Assessment of Techniques for Protein Structure Prediction (CASP) competition. In order to provide best predicted results for a target protein, the use of grid and distributed computing public schemes has been successfully demonstrated through various projects. For example, researchers at

TABLE 3.1: Grid-enabled applications for protein folding

Project/Application	**Grid Technologies**	**Distribution techniques and Speedup**
CHARMM [167] (Chemistry at HARvard Molecular Mechanics)	Legion grid operating system that provides process, files system, security services and resource management. Simple command line interface with basic commands for job submission, monitoring and result visualization.	400 CHARM jobs distributed with different initial conditions over 1020 grid nodes. 15% speedup in computational time.
CHARM [168]	United Devices (UD) MetaProcessor (MP) platform for DesktopGrid. Master (MP Server) controls and manages all the tasks and uses IBM DB2 for storage. Each worker runs a UD Agent with task API to run the task module and communicate with the server.	The Task: folding of src-SH3 protein with different algorithms(best-first, depth-first and breadth-first). Job distributed into 50 work units; each work-unit having 100,000 simulation steps. Experiments performed on heterogeneous platform of 45 desktop machines.

TSRI (The Scripps Research Institute) have developed a distributed public computing based protein structure prediction super computer (Predictor@Home) [162] using Berkley Open Infrastructure for Network Computing (BOINC) software. The predictor itself consists of a set of complex protocols with increasingly sophisticated models that rely on standard software tools such as BLAST, SAM-T02, PSIPRED, MFOLD simulation (for conformational sampling) and CHARMM (for molecular simulations). It is reported that during the 6th Critical Assessment of Protein Structure Prediction Methods (CASP) competition 6786 users participated in the Predictor@Home project and contributed a total compute time of about 12 billion seconds, the equivalent of about 380 years of computation on a single desktop machine, within just 3 months time. This computation power had been exploited for appropriate conformational sampling and refinement of the predicted structures

of 58 CASP6 targets.

The quality of the predicted structures utilizing the public computing infrastructure was compared with results using a dedicated local cluster (64 nodes, 2.4 GHz Pentium Xeon processors, 1GB RAM, 1GB Ethernet network). The results of the comparison indicate that the vastly larger distributed computing power afforded by the BOINC implementation resulted in far better predictions than using the dedicated cluster. A similar grid based approach that enhances the quality and performance of structure prediction has been demonstrated in [64]. It builds on the standalone web server named ROKKY (designed at Kobe University) that was ranked 2nd best prediction web server in the fold recognition category of CASP6 experiment. ROKKY uses a combination of standard analysis tools (PSI-BLAST and 3D-Jury) and the Fragment Assembly Simulated Annealing (FASA) technique using the SimFold [170] software package. In order to further enhance the quality of prediction and performance, a grid-based workflow design and control tool was added that allows the end-user to create/design a structure prediction experiment and submit it for execution on the Grid. That is, the user can modify input parameters and/or methods based on the real-time inspection/monitoring of the current predicted results. It has been reported [64] that for target T0198, the workflow-based prediction gave a faster result that was closer to to the target structure compared to employing a non-workflow based prediction, which uses simple batch files for job submission. This illustrates the importance of allowing the user to dynamically interact with the "production pipeline" even when the software is being distributed across the grid.

Another ambitious project, Encyclopedia of Life (EoL), attempts to predict structural information for all the proteins in all known organisms. The estimated computation time required for annotation of about 1.5 million sequences (as of 2003) using a pipeline of computational tools

(such as TMHMM, PSORT, SignalP, WU-BLAST, PSI-BLAST and 123D) has been approximated to be 1.8 Million CPU hours (more than 300 years!) on a single 1.8 GHz CPU. In order to facilitate this task, a grid-based workflow management systems has been proposed and demonstrated [171], which builds on the AppLeS Parameter Sweep Template (APST) technology providing an appropriate application deployment logistic and an adoptive scheduling and execution environment. The workflow was tested by running more than 54,000 proteome annotation jobs requiring 13670.5 CPU hours during the four days of the Super Computing Conference (SC03) on a grid testbed. This consisted of 215 CPU nodes managed at ten different sites having different operating systems and local resource management software. Further details of some grid-based protein structure prediction applications are presented in Table 3.2.

TABLE 3.2: Grid-based applications for protein structure prediction

Project/Application	**Grid Technologies**	**Distribution techniques and Speedup**
ProtFinder [172, 173]	Globus based GridWay Framework that uses adaptive scheduling for dynamic grids. Condor/G based GRAM. GIIS Server for resource discovery, GASS and GridFTP for data handling. User interaction with job submission agent through API or command line.	Prediction of 88 sequences was carried out in parallel by submitting an array job with 88 parallel tasks specified in a Job Template File. The entire experiment took about 43 minutes on 64 heterogeneous nodes.
PSA/GAc [174] (Parallel Simulated Annealing using Genetic Crossover)	NetSolve based client-server application model through GridRPC API. API for user interaction.	Simulated annealing distributed on NetSolve servers. GA crossover performed at the client side to reduce the communication delays.

3.4 Protein Structure Comparison

The comparison of protein three-dimensional structures based on a variety of similarity measures is a key component of the most challenging structural proteomic tasks, such as understanding the evolution of protein networks, protein function determination and, of course, protein folding and protein structure prediction. These structures are determined through experimental techniques such as X-ray crystallography or NMR spectroscopy as well as through various computational techniques involved in the process of protein structure prediction. The atomic coordinates of each structure are stored in publicly available database repositories such as Protein Data Bank (PDB) (www.pdb.org). PDB stores the information about each structure in a separate file. It uses different file formats such as PDB, PDBML/XML, mmCIF and FASTA among others. Each file is named with four lettered alphanumeric identifier and its contents consist of so many records and fields. Each record in the file may consist of single or multiple lines. With current number of structures it takes about 22 hours to download the complete database (www.pdb.org/pdb/statistics/holdings.do) on a local machine and it requires more than 36 GB of free disk space for its storage. The size (length) of each protein structure could be as simple as consisting of 40-50 residues or as complex as consisting of several thousand residues (e.g. in multi-functional proteins). Polymers having less than 40 residues are referred as peptide rather than protein. However, the average protein structure length is estimated to be around 300 amino acids (residues) per structure. In order to be comparable, protein tertiary structures as available in the PDB are usually further processed to be represented in some coordinate-independent space. The choice of suitable representation plays an important role in the development of an efficient and reliable protein structure comparison algorithm. Most commonly used coordinate-independent representations include Distance Matrix (DM), Contact

Map (CM), Absolute Contact (AC), Relative Contact (RC), and Contact Number (CN). There are several different methods (e.g., see Appendix .1), that use one of these representations to provide several measures of similarity /divergence between pairs of protein structures.

Though the process of comparison of a single pair of protein structures has been found to be solvable in polynomial time [175], but every individual method takes different time based on its individual complexity. Furthermore, as the number of known protein structures grows the size of their corresponding databases (such as PDB) also increases and hence, the process of structure comparison requires more efficient algorithms, which could exploit the power of web and grid computing technologies to provide accurate and optimal results with enhanced reliability and fault tolerance. One such approach has been demonstrated in [176], which employs a distributed grid-aware algorithm with indexing techniques based on geometric properties. It used a Globus and MPICH based Grid testbed consisting of four nodes (each with 300 MHz CPU). Experiments were performed comparing a target against 19,500 PDB structures in about 19 seconds. Another related approach is presented in [27] describing a meta-server for Protein Comparison, Knowledge, Similarity, and Information (ProCKSI), integrating multiple protein structure comparison methods such as the Universal Similarity Metric (USM), the Maximum Contact Map Overlap (MaxCMO), and an algorithm for the alignment of distance matrices (DALI), amongst others. Additionally, it produces a consensus similarity profile of all similarity measures employed. The application runs on a mini-cluster and provides a web-based interface (`http://www.procksi.net/`) for job submission and result visualization. As the study of this dissertation is based on the philosophy of ProCKSI, its further details are provided in section 3.4.1. Some other grid-enabled applications for protein structure comparison are presented in Table 3.3.

TABLE 3.3: Grid-based protein structure comparison

Project/Application	**Grid Technologies**	**Distribution techniques and Speedup**
FROG [177, 178] Fitted Rotation and Orientation of protein structure by means of real-coded Genetic algorithm.	Ninf Grid RPC based master/slave model. Asynchronous parallel programming in C language. Web based GUI through NinfCalc tool.	Master generates the initial population and then copies three non-redundant parents on each node in the grid repeatedly. Speed-up of 5.70 was achieved for the comparison of a single pair of proteins on a grid testbed of 16 nodes.
PROuST [179] Ontology and workflow based grid-enablement of PROuST application for protein structure comparison.	PROTEUS problem solving environment with Globus-based grid infrastructure. UML-based GUI for workflow composition, browsing, selection and result visualization.	The PROuST application is divided into three independent phases: pre-processing, similarity search and structural alignment. Each phase is implemented as an independent sub-workflow/component.

3.4.1 ProCKSI

ProCKSI is an online automated expert system that aims at helping an end-user biologist to compare any given set of protein structures using an ensemble of structure comparison methods (e.g. as listed in Table 1.1) and to visually analyze the consensus based relationship among the compared structures with a variety of heuristics and statistical clustering methods such as *the Unweighted Pair Group Method with Arithmetic mean* (UPGMA) [180] and *the Wards Minimum Variance* (WMV) method [181], all using an easy, intuitive and unified web interface as shown in Figure 3.1.

ProCKSI's web interface also works as a gateway and a knowledgebase of the entire protein universe as it provides hyper links to further important and most-often used sources of information needed for the exploration of specific details of any individual structure. These sources include various links to IHOP (*Information Hyperlinked Over Protein*) [51], SCOP (*Structural Classifica-*

tion of Proteins) [52], and CATH (*Class Architecture Topology and Hierarchy*) [53].

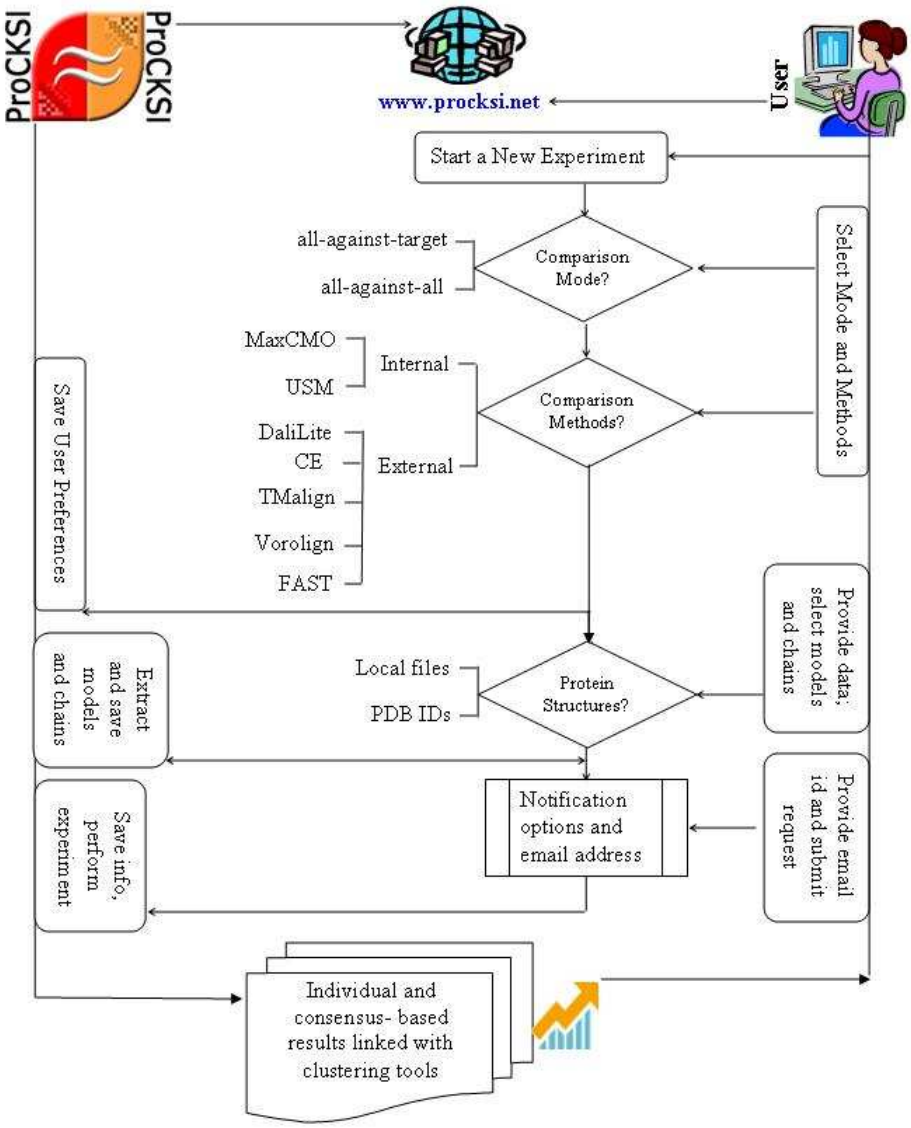


FIGURE 3.1: Flow diagram of ProCKSI's front-end. Initially the user types the URL (www.procksi.net), ProCKSI responds with a welcome page prompting for starting new experiment. The experiment requires the use to select the comparison mode and methods to be used for comparison; provide the data and select the models and chains from the list of new files extracted by ProCKSI. Finally ProCKSI asks for notification options and email address of the user for sending the notification regarding the results. The experiment is performed at ProCKSI's back-end [27] and web-based results linked with clustering and visualization tools are made available at the front-end

As demonstrated in [27], and previously suggested in [54] and [55], the ensemble and consensus based approach adopted by ProCKSI yields more reliable results of biological significance as compared to the results obtained with any single structure comparison method developed so far. This is mainly because previously developed methods used to be good in dealing with either very divergent structures or very similar structures; however, the integrated approach of ProCKSI enables it to deal well with both types of structures simultaneously. For example, to deal with the divergent structures ProCKSI uses the top level of its protocol (Figure 3.2), namely, the *Universal Similarity Metric* (USM) [48]. This method uses the contact map representation of two protein structures, say, S_1 and S_2, to heuristically approximate the Kolmogorov complexity by using any compression algorithm such as *compress, gzip, bzip etc*). It then uses the *Normalized Compression Distance* (NCD) (see equation 3.1) to express the pairwise similarities among the compared proteins. NCD, being an effective and problem-domain independent similarity metric works well particularly for divergent protein structures [48] and sequences [182].

$$NCD(s_1, s_2) = \frac{max\{K(s_1|s_2), K(s_2|s_1)\}}{max\{K(s_1), K(s_2)\}}, \tag{3.1}$$

where $K(s_i)$ is the Kolmogorov complexity of object s_i and $K(s_i|s_j)$ is the conditional complexity.

On the other hand, for more similar structures, ProCKSI uses a more refined method namely, the *Maximum Contact Map Overlap* (MaxCMO) method [46], which is able to detect the topological similarities among the compared proteins. This method employees metaheuristic to count the number of equivalent residues (alignment) and contacts (overlap) in the contact map rep-

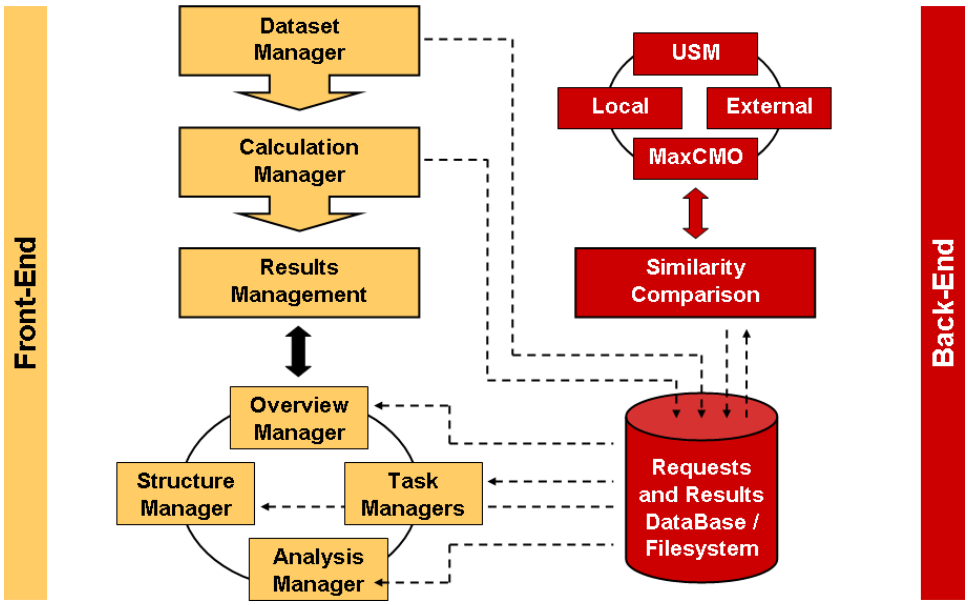


FIGURE 3.2: ProCKSI's multi-method protocol and workflow: ProCKSI with its multiple similarity comparison methods: Universal Similarity Metric (USM), Maximum Contact Map Overlap (MaxCMO), and other local and external methods. Currently, these are the DaliLite and TM-align methods, the Combinatorial Extension (CE) of the optimal path, and the FAST Align and Search Tool (FAST): extracted from: [27]

.

resentation of a given pair of structures in the following way:

> "An amino acid residue a_1 from one protein is aligned to an amino acid residue a_2 from a second protein if a contact of a_1 in the first protein $(C(a_1))$ can also be aligned to a contact of a_2 in the second protein $(C(a_2))$ closing a cycle of size 4 in the graph representation of the contact map. A further restriction for the overlaps is that they should not produce crossing edges. That is, if a_1 is aligned to a_2, $C(a_1)$ is aligned to $C(a_2)$ and, without loss of generality, $a1 < C(a_1)$ (i.e. the atom or residue a1 appears before than $C(a_1)$ in the sequence) then $a_2 < C(a_2)$. Thus, an overlap in this model is a strong indication of topological similarity between the pair of proteins as it takes into consideration the local environment of each of the aligned residues" [27].

A Fuzzy Sets based generalization of contact map has been proposed in [183] and methods of comparing proteins based on them in [184,185]. However, this work is based on only the discrete version. As each additional method complements the results of other methods, ProCKSI uses some other external methods such as the Distance Alignment (DaliLite) method [45], the Combinatorial

Extension (CE) method [47], the TM-align method [49], and the FAST method [50], in order to develop more robust and reliable consensus.

Following subsection provides an overview of major functional activities (tasks) carried out by ProCKSI for each user request in terms of its existing architectural design and infrastructure resources.This description is mainly aimed to provide an overall functional complexity of the system and hence to better understand the major challenges whose solution is explored through the rest of this dissertation.

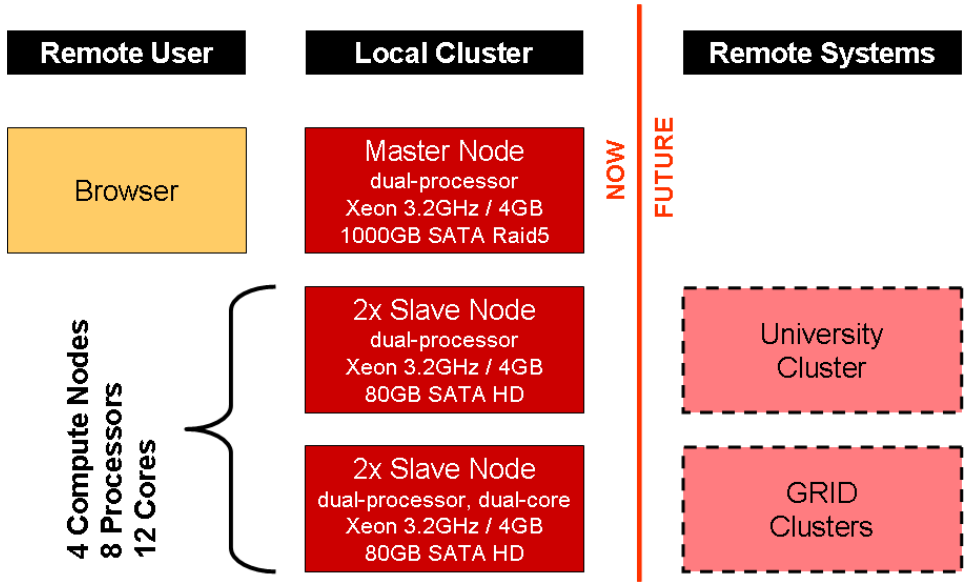
3.4.2 ProCKSI's Existing Architecture and Limitations

Currently ProCKSI runs on a PBS (Portable Batch System) based mini-cluster consisting of 5 nodes (one head and 4 compute) with 14 cpu slots in total (Figure 3.3(a)) and many software services as illustrated in Figure 3.3(b). When a new request is submitted through the simple steps as illustrated in Figure 3.4(a), the request is registered in the database and is assigned a unique ID. The registration of the request also involves the registration of tasks and structures specified by the user in his/her request (see Figure 3.4(b)). With current setup user could specify as many as 8 different tasks (methods as listed in Table 1.1) and as many as 250 protein structures (with maximum size of 100 MB) to be compared and analyzed. The specified structures need to be either uploaded from users local machine or downloaded from the PDB repository (which further limits the number of structures down to 50 so as to avoid prolonged latency involved in Internet based communication with PDB server). These structures are further extracted into models and chains as per users choice.

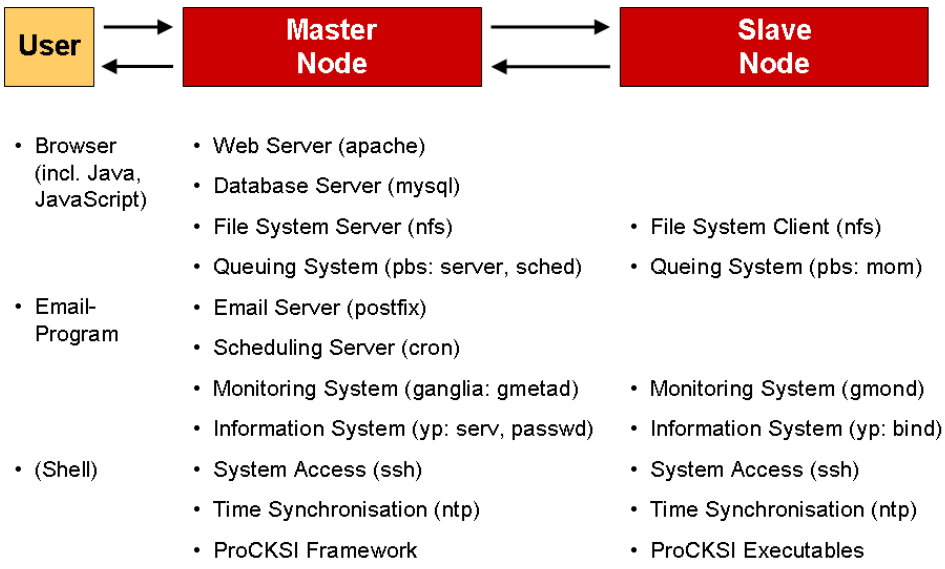
Once the request is registered it is further processed by a software component called *Task Manager* that runs on the master node of the cluster. The main role of the Task Manager is to extract the information from request and prepare individual tasks for submission to queuing system

(PBS). In its current architecture, the Task Manager of ProCKSI, prepares a separate task for each comparison method to be executed on a single processors using complete dataset as specified by the user. Depending on the load of the queuing system, the task (or job in terms of queuing system) might have to wait for some time before getting started on any slave node. Once the jobs gets started on the slave node, it might go through a long execution time depending on the size of dataset and the speed of particular structure comparison method (executable) as it runs only on a single processor.

It is therefore, the current distribution mechanism of the Task Manager needs to be scaled to some optimal fine-grained level, so as to allow more efficient comparison beyond the current limitations of 250 protein structures. The change in the distribution mechanism of the Task Manager would also have to take into account the computational load of pre/post-processing and result visualization that is currently being carried out only on the master node (see Figures 3.5(a) and (b)). The post-processing involves the complex and time consuming process of standardization (conversion of all results into a single format and normalization of all the values) and preparation of clusters to be visualized with different software for understanding the relationship among the compared set of proteins.

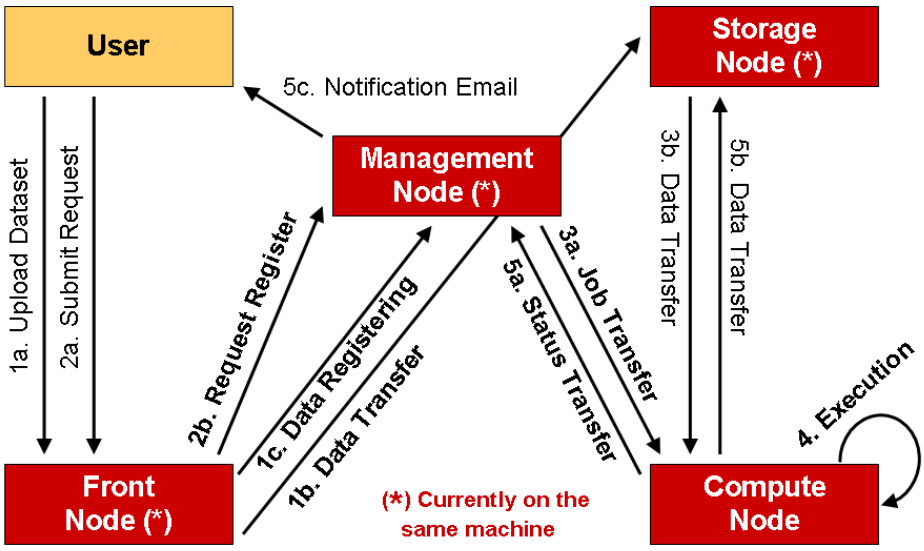


(a)

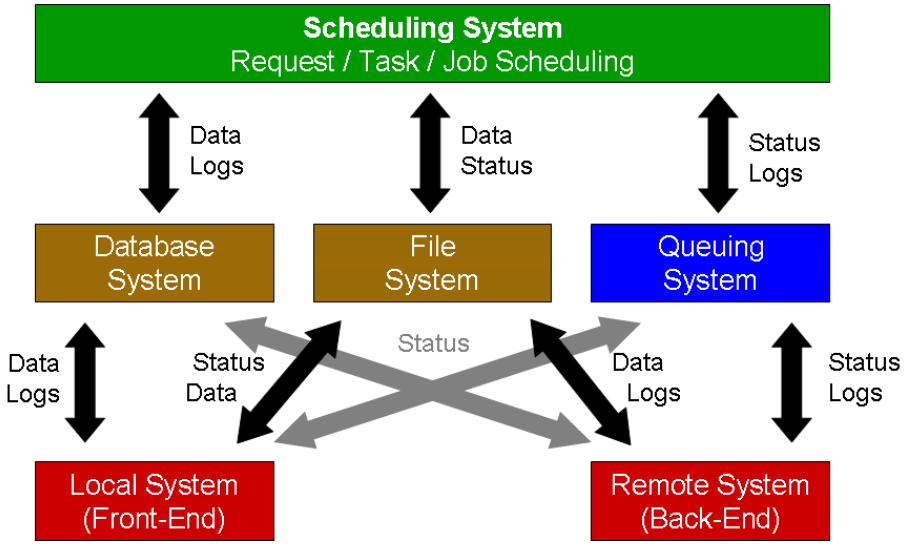


(b)

FIGURE 3.3: ProCKSI's current a) hardware infrastructure consists of 2 dual processor and 3 dual processor dual core (i.e., total of 12 cores) of which one node serves as cluster head and the rest 4 work as compute nodes) ; b) List of software services in terms of user, master and slave (compute) nodes. (*Diagram courtesy of Daniel Barthel*)

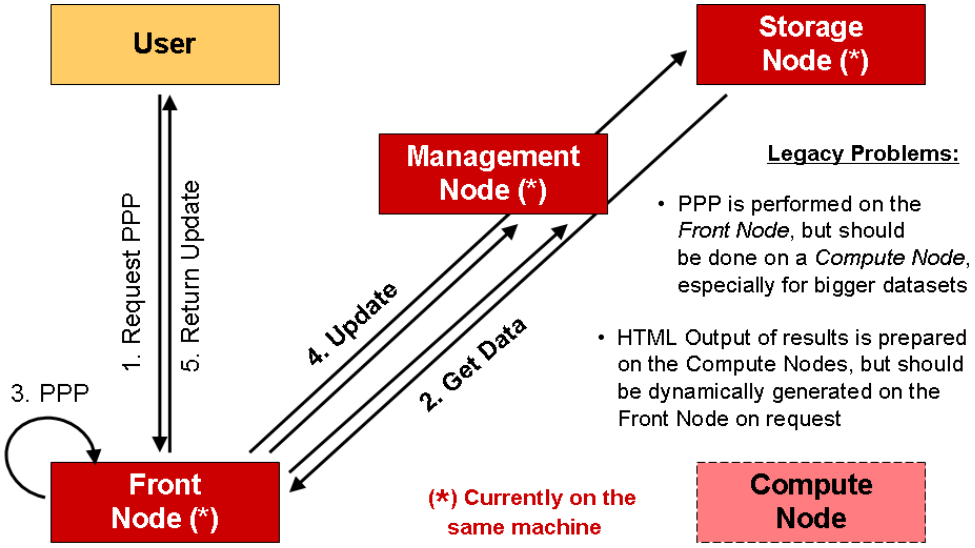


(a)

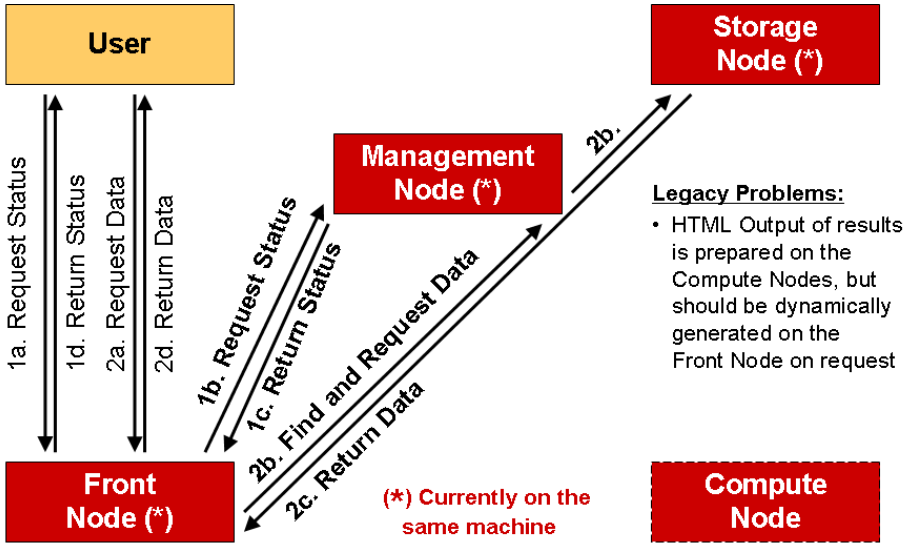


(b)

FIGURE 3.4: ProCKSI's a) request submission goes through 5 main stages for its completion. b) scheduling flowchart illustrates the main steps the data (application and status related) goes through in terms of request, task and job scheduling. (*Diagram courtesy of Daniel Barthel*)



(a)



(b)

FIGURE 3.5: ProCKSI's a) pre/post-processing (PPP) currently runs on the head-node while the data gets transferred from the storage (compute) node b) result retrieval flowchart illustrates the node interaction for sharing the request status and data. (*Diagram courtesy of Daniel Barthel*)

3.5 Conclusions

It has been observed from the reviewed literature that both Grid and distributed public computing schemes have been used successfully in the field of structural proteomics for both compute and data intensive applications. The former is powered by standard grid middleware technologies such as Globus, Legion, NetSolve, Ninf, myGrid, Condor/G, etc., whereas the latter is powered by BOINC, UD MetaProcessor etc. In fact, the diversity of enabling technologies for grid and distributed computing makes it difficult for the developer to select most appropriate technological infrastructure with proved technological standards and tools. Various demonstrations reviewed in this chapter are aimed at providing a road map in this dilemma.

It has been observed that selection of an appropriate grid/distributed computing approach mainly depends on the nature of the application. For example, applications with an independent and parallel nature of jobs are more suitable for distributed computing based on publicly-owned resources. For example, majority of the projects based on the investigation of protein folding process and protein structure prediction use this type of infrastructure as it provides huge number of resources free of cost. The availability of huge resources contribute to the better predicted results of the simulations as compared to results obtained on the limited resources available on a dedicated cluster or grid environment.

However, on the other hand, if the application involves some sort of interprocess communication along with huge amount of I/O data then organizational or cross-organizational grid infrastructure with above mentioned standard middleware would serve in a better way. This is mainly because, the interprocess communication and I/O overhead in terms of distributed public computing schemes would be very large owing to significant latencies over loosely coupled networks. In

the case of MC-PSC, though, there is no interprocess communication during the initial phase of comparison but the subsequent phase of standardization and normalization of results requires some data to be shared among all the processes. In addition to the requirement for interprocess communication, the MC-PSC also involves distribution and collection of significantly huge amount of I/O data. Therefore, in the light of the material presented in this chapter, it has been identified that the later approach (i.e., the organizational or cross-organizational grid infrastructure with standard grid middleware) would be more appropriate in the case of MC-PSC. Interestingly, though not required but almost all of the single method based protein structure comparison approaches reviewed in this chapter are also based on the use of standard grid and parallel programming environments. Building on this knowledge the next chapter provides further details of actual technology used for the solution of MC-PSC problem.

CHAPTER 4

MATERIALS AND METHODS

Chapters 2 and 3 provided the comprehensive review of the literature starting from the wide perspective of the field to more specific perspective of the research topic. The purpose of this chapter is to address the questions such as "how this research is designed?" and "which methodology will be used for the implementation and evaluation of the proposed solution?".

4.1 Introduction

The problem of large scale multi-criteria protein structure comparison (MC-PSC) and analysis could be represented as a 3D cube (Figure 4.1). The *x* and *y* axis of the cube representing the different proteins being compared, while the *z* axis representing different comparison methods being used. While processed, each cell of this 3D cube holds the output of each comparison method in terms of different measures and metrics. That is, each cell of the 3D cube represents both the processing as well as the storage perspective of the problem space while cell boundaries specify the communication overhead. Given the ever growing number of protein structure comparison methods as well as the number of protein structures being deposited in the PDB; the dimensions of this cube go on

increasing and making its computation, in our opinion, to be one of the Grand Challenge Applications (GCAs) in the field of structural biology. GCAs are defined as "fundamental problems in science and engineering with great economic and scientific impact, whose solution is intractable without the use of state-of-the-art parallel/distributed systems " [186]. Many examples of the use of parallel/distributed systems for the solution of GCAs in the field of life sciences in general and structural proteomics in particular have been showcased in previous two chapters. Based on the lessons learned from these examples, we propose a distributed framework as a solution to the grand challenge of MC-PSC. This chapter provides the description of the methodological approach that we used for the design of proposed distributed framework (section 4.2) along with the description of the programming environment used for its implementation (section 4.3), testbed for experimentation (section 4.4), datasets (section 8.1) and the performance measures/metrics used for the evaluation of the proposed system (section 4.6) .

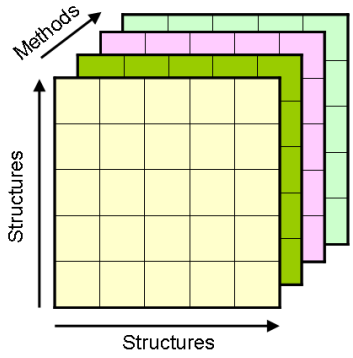


FIGURE 4.1: 3D-cube representation of the MC-PSC problem space. Each differently colored cube illustrate the comparison of a set of structures with itself (or another set of structures) using a particular algorithm (method).

4.2 Methodological Approach

It is believed that most of the GCAs may have several parallel solutions; therefore, a methodological approach based on an exploratory nature will help in finding the best available solution [187]. An example of such approach that is widely used for the design of parallel and distributed algorithms is the PCAM (*Partitioning, Communication, Agglomeration, and Mapping*) distributed problem solving strategy as illustrated in figure 4.2.

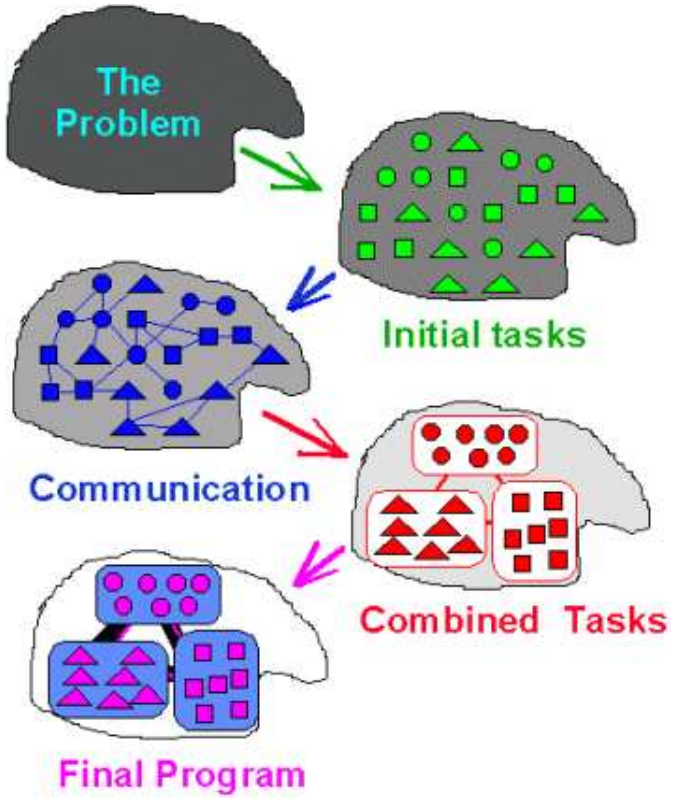


FIGURE 4.2: "PCAM: a design methodology for parallel programs. Starting with a problem specification, we develop a partition, determine communication requirements, agglomerate tasks, and finally map tasks to processors" [187]

Introduced by Ian Foster in his book "Designing and Building Parallel Programs" [187]], the beauty of this approach is that it enables the designer to consider the machine-independent issues

(e.g. concurrency, scalability and communication) first and machine-specific issues (e.g granularity and load-balancing) later in the design process. This strategy consists of four main stages which are summarized bellow:

Partitioning: The focus of this stage lies in exposing the opportunities for parallel execution in order to decompose the problem into large number of *fine-grained* tasks. The partitioning could be applied to decompose the *computation* (i.e functional decomposition) and/or the data (i.e domain decomposition). Different options were considered for both the *domain* and *functional* decomposition strategies to partition the 3D cube and analyze the pros and cons of each partition in terms of efficiency and scalability. The outcome of this stage as applied to our problem of MC-PSC is explained in section 5.3.

Communication: This stage determines the need of information to be exchanged among the parallel process resulting from the partitioning. It also specifies if there are any dependencies among the processes and if synchronization and dynamic communication strategies are needed. Appropriate communication structure is selected and theoretical cost analysis is performed in order to obtain the optimal solution. The analysis of the communication involved in sharing the input/output and other local/global data needed in the process of normalization as applied to MC-PSC problem is discussed in section5.3.

Agglomeration : The theoretical cost evaluation of the partitioning and communication stages would suggest if there is any way of grouping the fine-grained tasks in order to make the system more optimal. This stage focuses on the different options for grouping and finding out the best working model. Different methods of agglomeration for MC-PSC are are explained in section 5.3.

Mapping: This stage considers the assignment of processes/tasks for execution on each processor in a way to enhance the overall processor utilization and reduce the communication overhead. Depending upon the number of available processors this stage could suggest to adjust the agglomeration and introduce some load balancing approaches to assign same unit of work to each processor/node.

4.3 Programming Environment

Table 4.1 provides an overview of some commonly used systems for the implementation of parallel algorithms. Each of these tools serves different purpose and hence is used for a particular class of applications. We selected the *Message Passing Interface* (MPI) model of parallel programming as it is particularly used for the applications having the structure of either *Single Program Multiple Data* (SPMD) or *Multiple Program Multiple Data* (MPMD). The MPI itself is a library of standard functions for exchanging messages and performing collective operations (i.e operations which send/receive data from many nodes simultaneously e.g. *broadcast* (to send same data to all nodes), *gather* (to collect data from all nodes), and *scatter* (to divide the data into pieces and send a different piece on all node)) (see Table 4.2 for the list of commonly used MPI functions). The standards for MPI are defined and maintained by *MPI Forum*, which is an open group consisting of representatives from many organizations (`http://www.mpi-forum.org/`). MPI Forum introduced the very first standard (MPI 1.0 / MPI-1) on May 5, 1994 and the second (enhanced) standard (MPI-2) on July 18, 1997. Some of the major features introduced in MPI-2 include one-sided communications operations (e.g. Put, Get, and Accumulate etc.), Collective extensions (e.g. *MPI_Alltoallw* and *MPI_Exscan* etc.), and Dynamic process management (e.g. *MPI_Comm_spawn*, *MPI_Comm_join*,

and *MPI_Comm_accept*/*MPI_Comm_connect* etc.). Table 4.3 provides the list of both free available and vendor-supplied implementations of MPI standards. We tested the implementation of our application using three freely available implementation namely, MPICH2 [188] [189], Open MPI [190] and MPIg [191, 192].

TABLE 4.1: Overview of commonly used systems for parallel programming. Note: * indicates that this variant has become a *de-facto* standard in the community.

System	Available Variants	Purpose
Parallel C++	Compositional C++ (CC++)* [193], parallel C++ (pC++) [194], Concurrent Object-Oriented Language (COOL) [195], Mentat [196], High-Performance C++ (HPC++) [197].	All these languages and tools are extensions to C++. These extensions provide basic parallel programming features and abstractions. Used for algorithms requiring dynamic task creation and having irregular computation/communication patterns.
Fortran	Fortran M (FM)* [198] [199], Fx [200],	These languages provide all the programming constructs for 'task parallelism, deterministic execution, and modularity'.
HPF	High Performance Fortran (HPF)* [201], Connection Machine Fortran (CM Fortran) [202], Data Parallel Fortran (Fortran D) [203]	These are data parallel languages. They provide array operations to express parallelism. Mostly used for numeric algorithms.
Message Passing	Message Passing Interface (MPI)* [204], p4 [205] Portable Instrumented Communication Library [206], Parallel Virtual Machine (PVM) [207]	These systems provide standard functions for sending/receiving messages and are particularly used for algorithms having regular SPMD/MPMD structures.

4.4 Testbed and Production Level eScience Infrastructure

Experiments were run on three different levels of infrastructures: the *first* one being a simple testbed comprising of a single *Beowulf* cluster consisting of 6 nodes(with a total of 9 slots) connected through Gigabyte Ethernet. As this testbed was self-built and under our full administrative control we were able to configure and install different LRMS's e.g *Sun Grid Engine* (SGE), *Portable Batch System* (PBS) and *Load Sharing Facility* (LSF) along with different implementations of MPI such

MPI Routine	Description
MPI_Init	Initializes the MPI environment
MPI_Comm_rank	Determines the rank of the calling process within a group
MPI_Comm_size	Determines the size of the group
MPI_Bcast	Broadcasts a message from "root" process to all other processes
MPI_Send	Sends messages
MPI_Recv	Receives messages
MPI_Status	Provides information about received message in terms of error codes.
MPI_Barrier	Blocks until all process have reached this routine
MPI_Allreduce	Combines values from all processes and distribute the result back to all
MPI_Finalize	Terminates the MPI environment

TABLE 4.2: Description of MPI Routines

TABLE 4.3: Overview of commonly used MPI implementations

Implementation	Description
MPICH/MPICH2 [208, 209]	Developed by Argonne National Laboratory and Mississippi State University, MPICH/MPICH2 is the freely available and portable implementation of MPI-1/MPI-2 standard for most of the flavors of Unix and MS Windows.
SCore MPI [210]	Originally developed by Real World Computing (RWC); SCore is now taken care by PC Cluster Consortium and is freely available for many different platforms.
MPI.NET [211]	Developed at Indiana University; the MPI.NET is an open source implementation that allows parallel programming in .NET technologies such as C# and the Common Language Infrastructure (CLI).
Platform MPI [212]	Platform MPI (aka Scali MPI and HP-MP) is a commercial implementation of MPI developed by Platform Computing Inc. It claims to provide better performance as compared to open source or other commercial counterparts.
Open MPI [213]	Open MPI as the name reflects is an open source MPI developed as a result of taking best ideas from several other MPI implementations e.g. FT-MPI (University of Tennessee), LA-MPI (Los Alamos National Laboratory), LAM/MPI f (Indiana University), and PACX-MPI (University of Stuttgart). It is widely used by many TOP500supercomputers across the globe.
MacMPI [214]	MacMPI is the implementation of MPI by Dauger Research, Inc. that provides parallel programming environment for Macintosh.
MPJ Express [215]	MPJ Express is developed at the Centre for Advanced Computing and Emerging Technologies (ACET). It provides an environment to develop parallel programs in Java.

as MPICH, MPICH-G2, MPIg and OpenMPI in order to evaluate different possible alternatives. Furthermore, we also used this testbed to configure and install Globus Toolkit [34, 35], in order to

confirm the operation of our application before deploying it on production level infrastructure. The *second* level of infrastructure comprised of a production level *High Performance Computing* (HPC) Linux cluster, named *spaci* and placed at ICAR-CNR institute in Italy, with 64 dual-processors Itanium2 1.4GHz nodes each having 4GB of main memory and being connected by a Qsnet high performance network. The main purpose for using this infrastructure was to foster the collaborative relationship between Nottingham and ICAR-CNR in order to mutually share the skills and expertise needed at the interface of two complicated and inter-disciplinary subjects of Bioinformatics and Grid Computing. The *third* level of infrastructure consisted of the *eScience* infrastructure provided to all *United Kingdom* (UK) scientists free of cost by *National Grid Service* (NGS), UK [66]. In this case we used Globus-based MPIg [191, 192] (grid-based implementation of MPI) to spawn the jobs across two NGS sites; one at Leeds and the other at Manchester. Each of these sites have 256 cores (AMD Opterons) with 2.6GHz and 8GB of main memory. Succinct description of the software tools and services used for each of these infrastructures along with schematic diagrams are presented in the corresponding chapters that report on the empirical results for different cases.

4.5 Datasets

Different datasets were used to perform experiments. These datasets were previously used in the literature [27, 48, 185, 216–220] e.g. Chew-Kedem (CK34) dataset [221] , Rost and Sander dataset (RS119) [222], and Kinjo et al. [223] and because our group has experience using these datasets. The first two of these datasets i.e CK34 and RS119 were used as an example of small datasets consisting of 34 and 119 protein structures respectively, while the third dataset i.e Kinjo et al. was used as an example of large dataset consisting of 1012 protein structures. Some other datasets

were also prepared for the sake of different case studies. These include three datasets consisting of regularly increasing number of proteins i.e. SFBK-250, SFBK-500, SFBK-1000 datasets having 250, 500 and 1000 protein structures which were randomly selected and downloaded from the PDB. The main purpose for the preparation of these datasets was to investigate the effect of the increasing number of proteins on the overall execution time. Lastly, *PDB_SELECT30* dataset was prepared and used as an example of biologically significant dataset. It 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 [224]. All the PDB structures of these datasets were parsed into simple files containing single PDB chains and their contact map (CM) counterparts.

4.6 Performance Metrics

Two metrics are usually used for testing the computational scalability of a parallel/distributed system: the speedup S and the efficiency E.

Speedup:

The speedup of a parallel algorithm is the ratio between the time taken by the best sequential implementation of an application measured on one processor T_s and the execution time taken by the same application T_p running on p processors.

$$S = \frac{T_s}{T_p} \tag{4.1}$$

The optimal case is given by a linear speedup, i.e. If we run the same application on p processors, then we can expect at best a reduction in time of p, and therefore that the speedup will be at most p. In fact, this is only a theoretical condition because the parallel algorithm introduces an overhead, mainly due to the communication times among different processors. If the problem is not sufficiently complex, and the communication times are not negligible with respect to computational time, then the speedup might be noticeably smaller. Another factor that limits the speedup lies in the level of granularity of a particular piece of the program that Can't be further parallelized. This fact is known as Amdahl's law and it states that if the proportion of a program that can be parallelized is P and the proportion that can not be parallelized (i.e., remains serial) is $(1-P)$ then the speedup on N processors is limited as per following expression, no matter how much value of N is further increased:

$$\frac{1}{(1-P)+\frac{P}{N}} \tag{4.2}$$

Equation 4.2 shows that as the value of N tends towards infinity, the maximum speedup will be limited to $\frac{1}{1-P}$ and hence the benefit of further parallelization shows a saturation effect. For example, if the proportion of the program that can be parallelized is 95% then the proportion that can not be parallelized (1-P) will be 5% and hence the maximum speedup will be limited to the factor of 20 irrespective of how much value of N is further increased. This is depicted in figure 4.3.

Efficiency:

Efficiency is given by the ratio between the speedup S and the number of processors p:

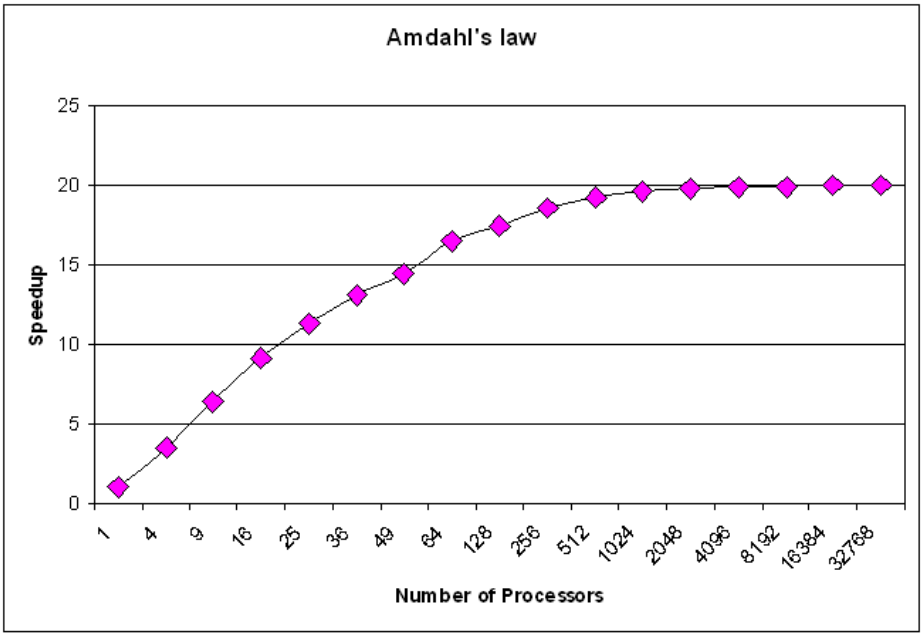


FIGURE 4.3: Illustration of Amdahl's law: the case when the proportion of the program that can be parallelized is 95% and the proportion that can not be parallelized (1-P) is 5% then the maximum speedup is limited to the factor of 20.

$$E = \frac{S}{p} \tag{4.3}$$

and it represents an index of the fraction of time usefully spent by each processor. In this case, the highest value of efficiency (equals to 1) is attained when all the processors are utilized to the maximum (communication times and other overheads equal to zero).

4.6.1 Grid speedup and efficiency

The theoretical analysis of the scalability of the *'computation-centric'* parallel applications on the grid appears in [225] with a prompt to the Grid community for the demonstration of this idea in terms of real Grid computing environments. This theoretical analysis is based on the idea of *'Homogeneous Computational Grid' (HCG)* and fits well with the real Grid computing infrastructure

provided by the UK National Grid Service (NGS) [66] that we use for MC-PSC. The HCG model is based on the concept of *'Hierarchical Resource Manager'* [226] and assumes that the Grid consists of C number of identical Computing Elements ($CE's$) and each CE (being a HPC system) has p number of identical processors along with identical means of interconnect. The workload decomposition on such a system consists of two-level hierarchy: at first the un-decomposed work (W expressed e.g. in Mflops) is equally distributed in C CE's (i.e W/C decomposition) and then within each CE the portion of the work is assigned to each of the p processors (i.e $(W/C)/p$ decomposition). Consequently, this two-level hierarchy gives rise to two sources of communication overhead i.e the communication overhead between C CE's $Q_1(W,C)$ and the communication overhead between p processors of each CE $Q_2(W/C,p)$. With this formalism, the execution time on HCG could be defined as:

$$T_{C,p}(W) = \frac{W_p}{pC\Delta} + Q_2(W/C,p) + Q_1(W,C) \tag{4.4}$$

Where Δ indicates the computing capacity of a processor e.g Mflops/s. Please note that if $C = 1$ and if $Q_1(W,1) = 0$ then the overhead of equation 4.4 returns to the standard parallel case i.e $Q_2(W,p) = Q(W,p)$.

Equation 4.4 makes it clear that running the parallel application on more than one CE's introduces an additional communication overhead in terms of $Q_1(W,C)$ which increases the execution time. However, this increase in the execution time could be masked by the value of C, which decreases the execution time by increasing the number of processors and also by reducing the communication overhead in terms of $Q_2(W/C,p)$ as compared to $Q(W,p)$ on one CE.

In order to analyze the added value of parallelism we normally compare the parallel execu-

tion time on P processors with the sequential execution time on 1 processor. However, as suggested by [225], in a Grid environment, we need to compare the parallel execution time on C CE's with the parallel execution time on 1 CE. This comparison is named as *Grid Speedup* and is mathematically defined as:

$$\Gamma_p^C = \frac{T_1, p(W)}{T_C, p(W)} \tag{4.5}$$

where, Γ_p^C is the 'Grid Speedup' (with p processors and C CE's), T_1 is the execution time on a single CE and T_C is the execution time on C CE's.

The Grid Speedup (equation 4.5) is one of the scalability metrics for the parallel applications on the Grid. Its value indicates how better a parallel application performs when decomposed on C CE's as compared to its performance on a single CE in terms of execution time. From equation 4.5 we could also derive the expression for the Grid efficiency as:

$$\gamma_p^C = \frac{T_1, p(W)}{CT_C, p(W)} \tag{4.6}$$

where, γ_p^C is the 'Grid efficiency' and p, C, T_1 and T_C represent the same parameters as described in eq. 4.5.

The description of the 'Grid Efficiency' in eq. 4.6 follows Amdahl's popular statement that "for a given instance of a particular problem, the system efficiency decreases when the number of available processors is increased " [227]. In the case of the Grid efficiency, in addition to the number of processors, it is the value of the C (number of CE's) that affects the system efficiency.

Based on these concepts of scalability, this dissertation performs empirical analysis of our parallel algorithm for MC-PSC as described in the following chapters. The above formalism could

also be extended to the heterogeneous systems as proposed in [228]. The study of the scalability in a heterogeneous environment is based on the assumption that each node in the system gets a workload related to its computational power and that the overhead time of different processors is also known. Equations 4.7 and 4.8 present the expressions for speedup and efficiency in a heterogeneous system.

$$S_{het} = \frac{T_s}{T_R} \tag{4.7}$$

where, T_s is the sequential time of the algorithm, $T_R = max_{i=1}^{N} T_i$, is the response time of the last node T_i (among a pool of N nodes) in the cluster to finish the application. That is, T_R represents the total time elapsed between the launching and termination of the application and it solely depends on the the time of the node that finishes in the last. This description of the T_R could also be applied to eq. 4.5 for achieving the Grid Speedup in the heterogeneous system.

$$E_{het} = \frac{W}{T_R \times \sum_{i=1}^{N} P_i} = \frac{W}{T_R \times P_T(N)} \tag{4.8}$$

where W represents the total workload, T_R represents the response time of the slowest node (i.e the node that finishes in the last), P_T represents the total power of the heterogeneous system (sum of the individual powers of the the nodes) and N represents the total number of processors in the system. The *power* in this case means the amount of work that a node/system can perform in a unit time while executing a particular algorithm.

4.7 Conclusions

This chapter is based on the description of the specific research methodologies which have been used for the design, implementation and evaluation of various parallel and distributed approaches

for the solution of the MC-PSC's computational challenge. Based on this methodology, the next chapter presents the design, implementation and evaluation of the distributed framework for MC-PSC and the subsequent chapters further build upon it by addressing several other issues related to this framework.

CHAPTER 5

A HIGH-THROUGHPUT DISTRIBUTED FRAMEWORK FOR MC-PSC

This chapter, presents a novel distributed framework for the efficient computation of large scale MC-PSC. The design, implementation and evaluation of this distributed framework is presented. Based on the succinct description of multi-criteria protein structure comparison (*MC-PSC*) as provided in the previous chapter through an overview of ProCKSI; and an in-depth description of the computational challenge at the core of real-time *MC-PSC* as explained in the first chapter; this chapter describes the high-throughput implementation of the entire protocol shown in Figure 1.1, whereby very large protein structure dataset comparisons are done in parallel using several methods and exploiting the intrinsic MIMD (*Multiple Instructions Multiple Data*) structure of the problem. Thus, the work presented in this chapter takes a step forward towards the ultimate goal of real-time multi-criteria similarity assessment of very large protein datasets.

This chapter was published as a peer reviewed journal paper in IEEE Transactions on NanoBioscience, Vol. 9(2), pp.144-155, 2010. [doi:10.1109/TNB.2010.2043851]

5.1 Introduction

Recent advances in high-throughput techniques have led to a data deluge in terms of the availability of biological and biomedical data such as 1D sequences (flat files), 3D structures, microscopic images, videos and motifs, etc. [28]. This has put considerable strain in the computational resources that are routinely used to store, manage, process and analyze the vast amount of data being generated. As to cope with the increase in computational demands instigated by very large data sets, many existing applications are being ported to distributed/grid environment. For example, the BLAST (Basic Local Alignment Search Tool [229]) algorithm has been parallelized/distributed through a variety of ways [38, 42, 230–236]. Some of these approaches use combinations of MPI (Message Passing Interface) [237], Grid and Public Computing based architectures to distribute either the query sequence (which could be as long as 80 billions of base pairs [238]) or the target dataset/database(which could have up to 76 million records [238]) or both. All these approaches use a simple master/slave task scheduling strategy with coarse-grained level task distribution for minimizing communication overheads [28]. Coarse-grained approaches are not always suitable: given the variable length of the sequences to be compared and the different processing power of individual nodes in a heterogeneous cluster/grid environment, deciding the actual unit of work to be assigned to a particular node is a non-trivial matter for which efficient dynamic load-balancing strategies are needed. Martino et al. [239], describe a simple, inexpensive and effective strategy that divides the target dataset/database in n buckets of fixed size (where n represents the number of available processors). The load-balancing in this case is achieved by ordering the sequences by their length (number of bases or residues) and assigning them to each bucket in a way that the longest sequence is assigned to the segment having smallest sum of sequence lengths and continuing this process in a

round-robin fashion until all sequences are assigned to buckets. This type of load-balancing strategy reduces the *percentage of work load imbalance* within homogeneous computing architectures but does not take into account the heterogeneity of cluster and grid environments. Trelles et al. [240] present another load-balancing approach based on variable size of blocks (buckets). This strategy initially distributes blocks with small sizes so as to reduce the latency time for each node to receive its first unit of work. It then increases the size of blocks (in the same way as classical *Self Guided Scheduling* (SGS) reduces their size) until the first half of dataset/database is processed and then again starts decreasing their size. The smallest size of final blocks guarantees that all n processors will terminate either at a same time (ideal case) or with a maximum time difference that depends on the size of the final block (i.e its execution time). This strategy has been tested on a cluster of 15 nodes with significant enhancement in the performance.

Proteins 3D structure comparison algorithms (e.g. those listed in Table 1.1) present a similar structure to algorithms for sequence comparison (e.g BLAST, FASTA and ClustalW etc) and hence sometimes similar parallel/distributed strategies can be used [28]. However, as compared to their sequence counterpart, there are very few instances of the application of parallel computing for 3D structure comparison methods (e.g., Ferrari et al. [176] and Park et al. [177]). None of these methods, however, deal with the much more complex issue of efficient and scalable distributed implementations for Multi-Criteria Protein Structure Comparison. This chapter, therefore, presents a novel distributed framework for the efficient computation of large scale MC-PSC. The design, implementation and evaluation of this distributed framework is presented as per following organization: sections 5.2, 5.3 and 5.4 provide the architectural design and analysis of the newly proposed framework. Experimental results and their analysis are presented and discussed in section 5.5 and

finally section 5.7 concludes the findings of this chapter.

5.2 Design and Implementation

In this section we present the algorithmic framework we use to compute in a distributed environment solutions to the MC-PSC problem. Figure 5.1 illustrates the overall architecture of the proposed system. The top module performs the distribution (through two different decomposition approaches as explained in the following sections) of pairwise comparisons and allocates them over the available nodes. Then, using the assigned (bag) proteins, each node performs, in parallel and without the need for synchronization, the pairwise comparisons required by its associated protein bag using each of the available PSC methods. That is, each compute node computes a sub-matrix from the all-against-all similarity matrices associated to each method. Afterwards, a phase of normalization and estimation of missing/invalid values is executed. This phase exchanges information among nodes, as it needs the global minimum and maximum similarities for the normalization as well as for the estimation of missing/invalid cells. All the results concerning the current node are stored on a local matrix. Note that no global and centralized matrix is maintained by the system and that all the communication among the nodes are performed using the MPI (Message Passing Interface) libraries for a cluster of computers and using the MPIg libraries [191,192] in the case of a grid-based implementation.

The pseudo-code shown in Algorithm 1 illustrates the main steps performed by each node in the distributed framework. Lines 1- 7 perform the pairwise comparison with all the methods for all of the proteins assigned to a particular node. Because the system does not maintain a global and centralized matrix, the process of finding the extrema (maximum and minimum similarity values

needed for the subsequent step of normalization) takes place in two steps. First, the local extrema are found (lines 8- 11) for all the methods. These are then shared among all the nodes to find the global extrema (line 12). Once the extrema are found, the next step (line 13) calls a subroutine that replaces all the invalid and missing values with their corresponding estimated values. Finally, line 21 calls the subroutine *normalize_diagonal* that performs the normalization of self-similarity values (across the diagonal of the matrix) and line 22 calls the subroutine *normalize_extrema* that uses the previously calculated extrema to perform the normalization of all the values.

5.3 Decomposition Strategies

The efficiency of the distributed framework strongly depends on the way in which proteins are assigned to compute nodes.

A good load balancing strategy should considerably reduce the execution time and the memory necessary to store the main matrix and other data structures necessary to the overall computation of MC-PSC.

Consider a set of resources (nodes of the clusters or machines on the grid) $N_1, N_2, \ldots, N_n$ and the main matrix ($proteins \times proteins \times methods$) storing the result of the computation and of the normalization (and estimating invalid/missing values) phases. Let p be the total number of proteins and m the total number of methods computed. Note that, indeed, M indicates the total number of **indices** computed by the different m methods; in fact, $M = \sum_{k=1}^{m} M_k$, where M_k is the number of indices computed by the method k (see Table 5.1 for complete nomenclature).

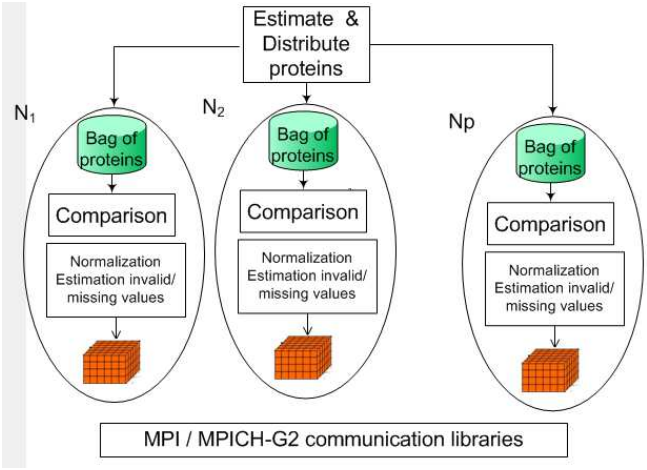


FIGURE 5.1: Software architecture of the distributed framework. The top module tries to estimate and distribute the workload equally to each node available in th pool of resources. Each node runs the distributed part of the algorithm to perform comparison with all methods followed by data post processing (standardization & normalization) and produces the sub-matrix of the results.

Algorithm 1 Pseudo-code executed from each node x concerning the multi-comparison part and the normalization/replacing invalid missing values part. Line 1 iterates for each method, with m representing the total number of methods. Lines $14-16$ scan through the results and replace the missing self-similarity (SS) and non-self similarity (NSS) values (for more details please see section 8.3). Lines $18-22$ use the extrema values to compute the normalization of the MC-PSC similarity values stored in the distributed *'matrix'*.

1: **for all** method k such that $1 \leq k \leq m$ **do**
2: **for all** protein i in row (x) **do**
3: **for all** protein j in column (x) **do**
4: compute_method k on the couple of proteins i and j {on node x}
5: **end for**
6: **end for**
7: **end for**
8: **for all** k such that $1 \leq k \leq m$ **do**
9: find_local_min
10: find_local_max
11: **end for**
12: exchange and find all global *min* and *max*
13: replace_invalid_missing_values:
14: **for all** k such that $1 \leq k \leq m$ **do**
15: missing_SS: *max* value in the cross-section
16: missing_NSS: *min* value in the *matrix*
17: **end for**
18: **for all** method k such that $1 \leq k \leq m$ **do**
19: **for all** protein i in row (x) **do**
20: **for all** protein j in column (x) **do**
21: normalize_diagonal: $matrix[i][j][k] = matrix[i][j][k]/max$
22: normalize_extrema: $matrix[i][j][k] = (matrix[i][j][k] - min)/(max - min)$
23: **end for**
24: **end for**
25: **end for**

In order to distribute the overall M among the nodes, there may be four possible partitioning schemes. Each of these schemes uses a different level of granularity for assigning the jobs as:

1. Comparison of *one pair* of proteins with *one method*. This will create $p \times p \times m$ jobs
2. Comparison of *one pair* of proteins with *all methods*. This will create $p \times p$ jobs
3. Comparison of *all pairs* of proteins with *one method*. This will create m jobs
4. Comparison of a *subset of pairs* of proteins with a *set/subset of methods*. This will create as many number of jobs as the number of available processors (or as desired).

The suitability of any of the above listed partitioning schemes could be analyzed in terms of its workload distribution. Considering the case of all-against-all comparison of PDB (64036 structures) using six comparison methods, partitioning 1 will generate ≈ 24.6 billion jobs. Obviously, all of these jobs can not be assigned to processors with one-to-one mapping and hence most of the jobs will have to remain in the queue which needs to be managed properly. Also, the major problem with this scheme would be that the processors will remain idle while reporting the output for each finished job and getting the details of the next job and fetching its related input files. This means that the partitioning 1 would be too *fine-grained* to be considered for the distributed/-grid environment. The case of partitioning 2 is also same because it has a slightly higher level of granularity than partitioning 1 and would results in as many number of jobs as ≈ 4.1 billion. Partitioning 3 on the otherhand, would generate only 6 jobs and hence would be too *coarse-grained* to be considered for the distributed/grid environment. The issues of too much fine-grained or too much coarse-grained level of granularity associated with these 3 partitioning schemes could be well

balanced with partitioning 4. This scheme suggests to create a job package consisting of many pairwise comparisons with either one or all methods to be dispatched as a single job for execution on a node. The number of pairwise comparisons in a package could be determined in an intelligent way so that equal amount of work gets assigned to each processor.

We investigate the 4^{th} partitioning scheme by applying two different approaches. The first decomposition adopted is shown in figure 5.2. The main matrix that stores the results is decomposed among the available nodes along the two proteins axis, so each comparison among two proteins for all the methods is performed on the same node, better balancing the different methods. This decomposition is the more efficient in terms of inter-jobs communication overhead, as it minimizes the number of information exchanges amongst compute nodes. Furthermore, the matrix is perfectly partitioned as each node is responsible for the computation and storage of same number of proteins $\frac{p^2 m}{n}$. In the next subsection these results will be analyzed in more detail. However, initial experiments suggested that execution times for different couples of proteins can largely fluctuate (see table 5.3), making the load among the different nodes not really balanced.

A second strategy is to balance the total execution time per compute node rather than the number of pairwise comparisons. Thus, this strategy takes into account the inhomogeneities in the size of the proteins being compared and is shown in figure 5.3. In order to setup a bag of proteins having the same overall number of residues on each node, the following largely used strategy was followed. Consider the case of proteins to be assigned to the $\sqrt{n}$ row processors (but the procedure is analogous for the column processors). First of all, proteins are sorted by the number of residues. Then, they are assigned, from the longest to the shortest one, to the node having the current lowest sum of residues. This procedure is not really time consuming, as it requires $p \log p$ for sorting the

proteins and $p(\sqrt{n})^2 = pn$ for assigning them to the correct node. The same distribution obtained for the row is also chosen for the column, so that the order of rows is not different of that of the columns and the operation of normalization and removing invalid/missing values could be performed without other overheads.

Each of the two proposed load balancing approaches result in a different CPU and memory usage. In what follows we analyze the benefits and drawbacks behind each of them. Unless otherwise stated, a given analysis/argument applies to both of the strategies. Henceforth, the first decomposition will be referred to as *even* and the second one as *uneven*.

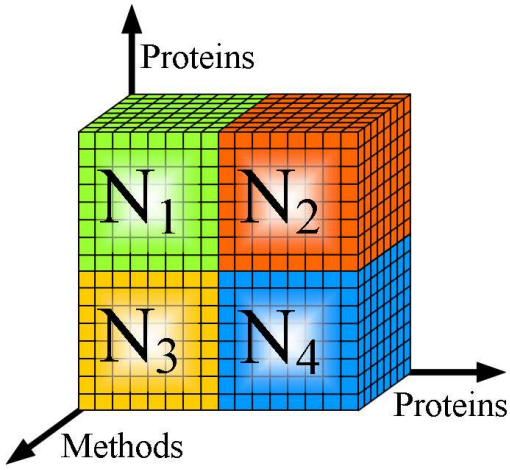


FIGURE 5.2: Even distribution of the problem space (*proteins* × *proteins* × *methods*). Each node is responsible for the same computation, i.e. same portion of the matrix)

.

5.4 Cost Analysis

5.4.1 Space Analysis

In what follows we do not take into account transient memory requirements by the different methods (e.g. internal data structures) as these have, on the one hand, been already analyzed in the original

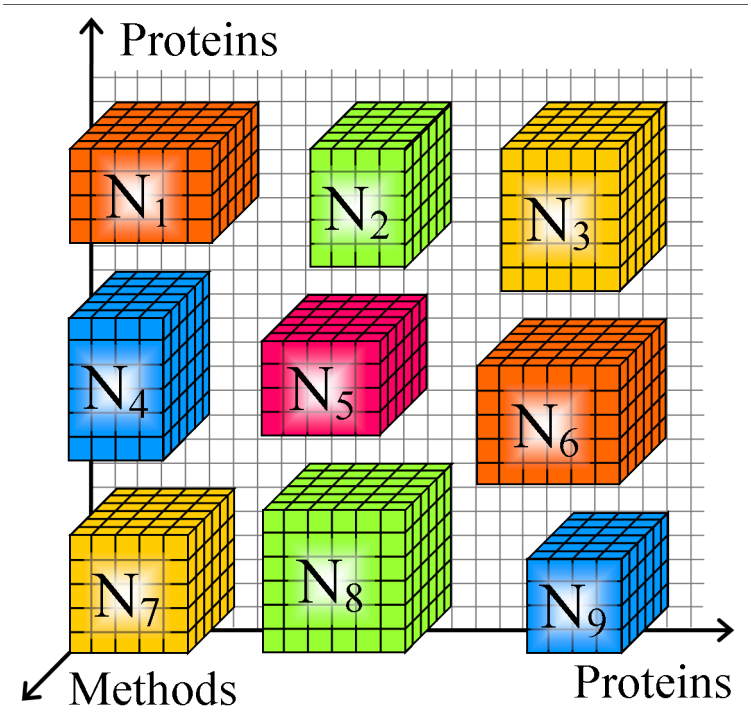


FIGURE 5.3: Uneven distribution of the problem space (*proteins* × *proteins* × *methods*). Note that the different sizes take different protein sizes into account (e.g. one node only for a few big proteins, which take quite long to calculate; and one node for many smaller proteins, which are quicker to calculate).

article where each method was originally introduced and, on the other hand, these transient space requirements are released as soon as a particular pairwise comparison is done. The nomenclature used in our analysis is summarized in Table 5.1.

The entire matrix, storing the comparison/normalization results, is decomposed along each of the two proteins axis among n nodes. So, in the case of even distribution, each node handles a matrix of size $\frac{p^2m}{n}$ and of size $= max(row\_prot_x \times col\_prot_x) \times m$ for the uneven distribution, where p is the number of proteins, n the number of nodes, m the total number of computed methods and row_prot_x and col_prot_x are respectively the proteins stored on the row and on the column of the matrix assigned to the node x. In this case the space $\rightarrow \frac{p^2m}{n}$ if $max(row\_prot_x \rightarrow \frac{p}{n}$ and $max(col\_prot_x \rightarrow \frac{p}{n}$, i.e. almost the same number of proteins is stored on each node.

	Description
p	Number of proteins
n	Number of nodes (processors)
m	Number of methods used (i.e. MaxCMO, FAST, etc..)
M_k	Number of indices computed by the method k
M	Total number of indices computed by all the methods
row_prot_x	Number of row proteins present on node x
col_prot_x	Number of col proteins present on node x
$Eval_x$	Number of evaluation conducted on node x ($row\_prot_x \times col\_prot_x$)
$\overline{S}_p$	average size of proteins
$\overline{Tm}_x$	average execution time of all the methods over all the couples of proteins on node x
$\overline{Tm}$	average execution time of all the methods over all the couples of proteins

TABLE 5.1: Nomencalutre used for the analysis.

The space necessary to store the proteins is $\frac{p^2 m\overline{S}_p}{n}$ for even distribution and $max(row\_prot_x \times col\_prot_x)m\overline{S}_p$ for uneven distribution, where $\overline{S}_p$ is the average size of proteins. This is the worst case as in many cases row proteins and column proteins are overlapped.

Obviously, in the case of the uneven distribution, the memory space is balanced only if the number of proteins stored on a node are not much different from those stored in the others.

5.4.2 Time Analysis

Let $\overline{Tm}$ be the average execution time of all the methods over one pair of proteins and $\overline{Tm}_x$ be average execution time of all the methods over one protein pair stored on node x.

So, only for the computation part of the algorithm, in a single node execution, the total execution time will be $T_s = p^2 \times \overline{Tm}$. As for the distributed case, formulation is not so simple as, depending on the distribution of the proteins, average execution times could be really different from node to node. In such case, in the even distribution the parallel execution time will be $T_p = \frac{p^2\overline{Tm}_x}{n}$ and $T_p = max(row\_prot_x \times col\_prot_x \times \overline{Tm}_x) \leq max(row\_prot_x \times col\_prot_x) \times max(\overline{Tm}_x)$ for the

uneven distribution. So, one has to balance this product as to obtain a fair computation; in the case of even distribution only if $Tm_x \leftarrow Tm$ for each node, a balanced load is achieved.

In addition to considering different execution times over different nodes, communication overheads must also be taken into consideration. This overhead happens in the first phase, when proteins are distributed over the nodes (using *MPI_Bcast* routine) and in the latter phase, when normalization and invalid/missing value replacement must be conducted (using *MPI_Allreduce* routine).

Moving the proteins to different nodes does not require an excessive time in comparison with the large computation time of the computing phase. Naturally, *even* decomposition needs slightly less overhead than *uneven* one as almost the same number of protein must be send to each node. The amount of data exchanged is, as discussed before, $\frac{p^2\overline{S}_p}{n}$ for even distribution and $max(row\_prot_x \times col\_prot_x)\overline{S}_p$ for uneven.

As for the normalization phase, we need to compute the global *minimum* and *maximum* for all the methods for a total of $2\ m$ values exchanged. For the correction of invalid or missing values we need the *minimum* or *maximum* for each row and column and method in which we got an invalid value. Thus, in the worst case, we have to exchange $2n^2m$, but typically invalid values are found only for a few methods and not for many cells.

Although the same amount of information is exchanged by the two decomposition strategies, the communication overhead is higher for the *uneven* strategy. This is mainly due to the worse efficiency for collective communication in an environment in which there are different number of rows and columns for each processor.

5.4.3 Discussion of the Theoretical Cost Analysis

The two decomposition strategies adopted present different pros and cons. Although the even decomposition better utilizes memory both in terms of cells of the matrix ($\frac{p^2m}{n}$) and proteins ($\frac{p^2\overline{S}_p}{n}$), it does not balance well the execution time on the different nodes, especially if, as usual, proteins have very different structures (or number of residues) . On the contrary, the uneven distribution, paying the cost of a larger memory requirements ($max(row\_prot_x \times col\_prot_x) \times m$ for the matrix and $max(row\_prot_x \times col\_prot_x)\overline{S}_p$ for proteins), is the only approach usable for obtaining appreciable reduction in execution times for small-medium and not well balanced datasets of proteins.

5.5 Experimental Results and Discussions

Different experiments were conducted to validate the quality of the two decomposition strategies. Two metrics are usually used for testing the computational scalability of a parallel/distributed system: the speedup S and the efficiency E. The "speedup" is a measure that indicates the improvement in the execution time of a parallel algorithm as compared to its sequential counterpart where as the "efficiency" indicates the utilization of each processor in a parallel system.

5.5.1 Datasets and Test Suite

All the experiments were performed on a Linux cluster, named *spaci* and placed at ICAR-CNR institute in Italy, with 64 dual-processors Itanium2 1.4GHz nodes each having 4GB of main memory and being connected by a Qsnet high performance network.

In our experiments, we used the first chain of the first model both for the Rost and Sander dataset (RS119) and for the Chew-Kedem (CK34) data set (see Table 5.2 for the characteristics

of these datasets). As an example of a large dataset, we used the one proposed by Kinjo et al. [223]. This dataset has been prepared by using PDB-REPRDB [223] algorithm to select 1012 non-redundant protein chains. The length of each chain in this dataset is greater than 50 with a sequence identity less than 30%. Furthermore, the dataset does not contain any chain with non-standard residues or chain breaks and all of its chains have resolution better than 2 Å and R factor better than 20%.

TABLE 5.2: Overview of the datasets used in the experiments. The hash symbol (#) is an abbreviation for *Number of*

Dataset	**# Chains per Datasets**	**# Comparisons per Datasets**	**# Residues per Datasets**
CK34 [221]	34	1,156	6,102
RS119 [222]	119	14,161	23,053
Kinjo et al. [223]	1012	1,024,144	252,569

5.5.2 Scalability of the Even Decomposition

To evaluate the quality of the even decomposition, the previously introduced metrics of scalability and efficiency were used, together with the execution time on different numbers of processors. The speed-up values obtained for the two medium datasets CK34 and RS119 are shown in figure 5.4.

For both datasets, the speed-up remains good using up to 16 processors, but using more processors does not help to speed up the total execution time to the same degree. This is due to the structural differences of the proteins, as each protein is composed by a different number of residues. Indeed, inspite of having the same number of proteins on each node, some proteins could have a large number of residues on a node and a few on another one. This consideration is confirmed by the large variance in the execution times of the different methods (Table 5.3). As for the execution time, for the RS119 (CK34) dataset, the entire execution time was reduced from about 6 days (6.2 hours),

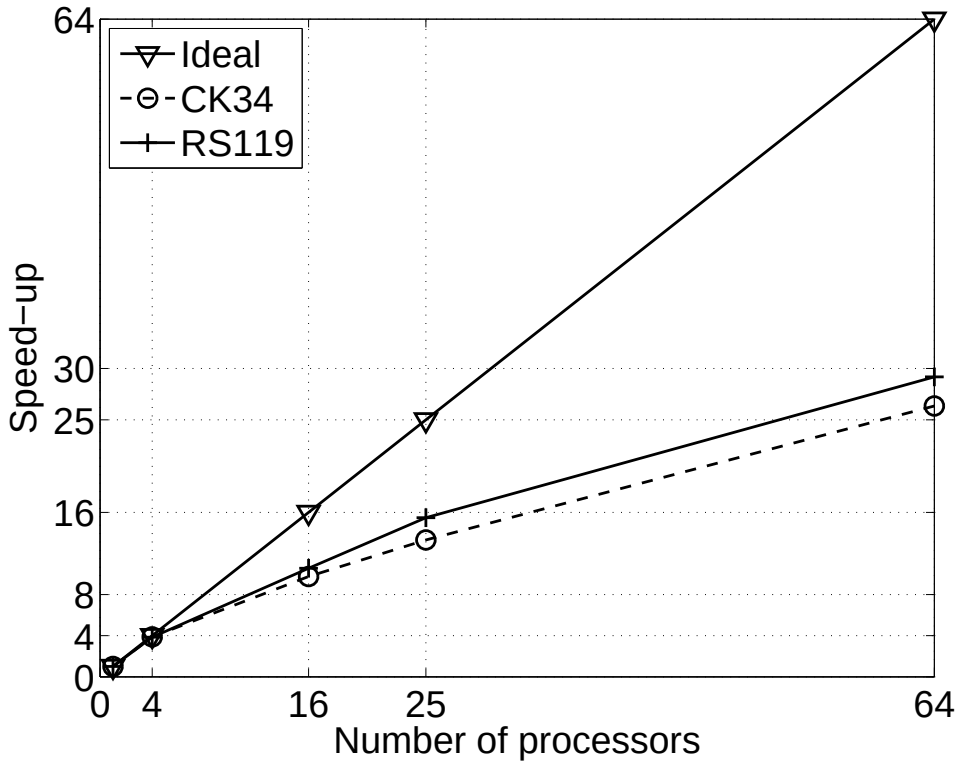


FIGURE 5.4: Speedup of the even decomposition using the CK34 and RS119 datasets on *spaci* cluster. The graphs shows that with CK34 dataset the speedup of the distributed algorithm (based on even decomposition) is around 26.2× with efficiency of 41% while the values of speedup and efficiency increase to around 30× and 46% respectively for the larger dataset i.e., RS119.

using the sequential implementation on one machine, to 4.8 hours (14.15 minutes) on 64 processors. This means that the new distributed algorithm performs 25× (with CK34 dataset) and 30× (with RS119 dataset) better than its current sequential counterpart. However, these improvements are still far from the ideal improvement of 64× and hence on 64 processors, the efficiency degrades to the values of 41% and 46% respectively for CK34 and RS119. The following sections provide analysis of this effect in detail and introduce another approach that further enhances the speedup as well as the efficiency of the system.

It is important to understand whether the execution times of the different methods described in the previous sections depends on the number of proteins, on the numbers of residues, or

TABLE 5.3: Average execution times and standard deviation (minutes) of the different methods for the CK34 and RS119 datasets averaged over 60 tries.

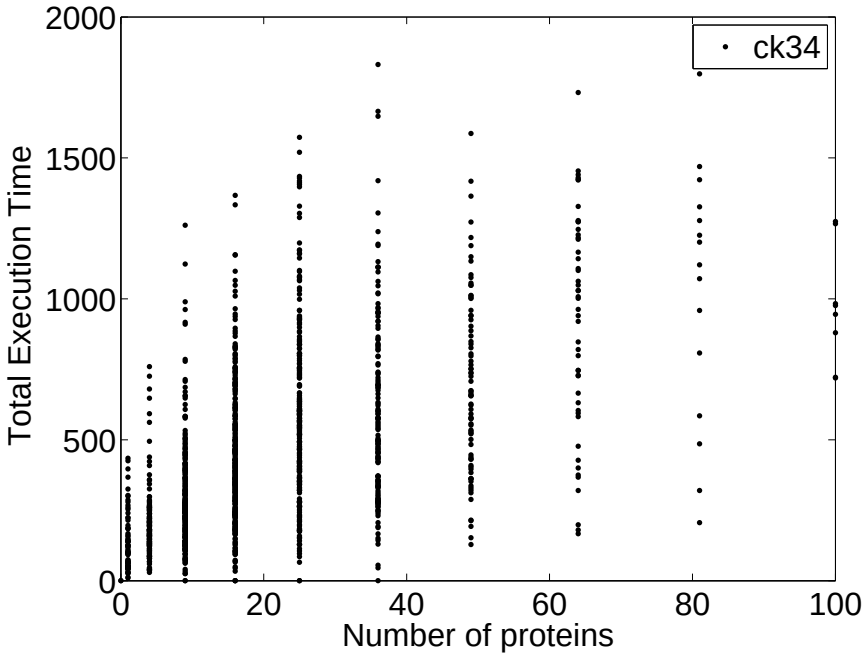
Dataset	USM	FAST	TM-ALign	Dali	CE	MaxCMO
CK34	0.52 ± 0.28	0.14 ± 0.07	0.28 ± 0.11	3.49 ± 1.53	3.20 ± 0.66	0.99 ± 0.34
RS119	3.68 ± 0.31	2.16 ± 1.05	5.78 ± 2.86	44.59 ± 20.51	41.05 ± 20.41	20.13 ± 9.69

on both of them. To this end we randomly divided the proteins composing the two datasets CK34 and RS119 among 64 nodes (a 8x8 grid of processors) and we run all the available methods and measured the execution time, the overall number of residues and of proteins present on each node. This procedure was repeated for 20 times for a total of 120 different measures of time.

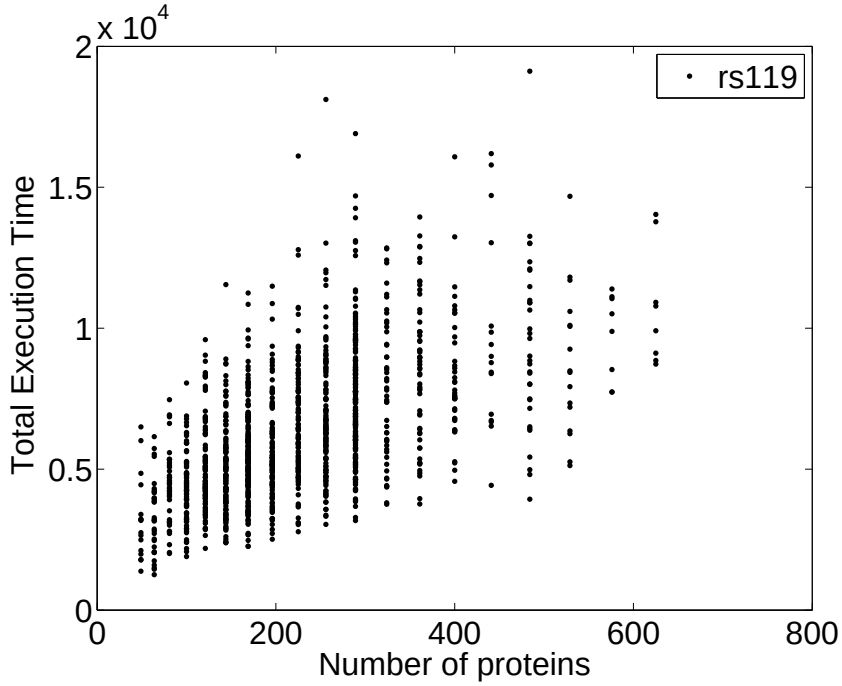
Then, we plotted the execution time vs the number of proteins (figures 5.5 a and b) and the execution time vs the overall number of residues (figures 5.6 a and b). Observing the figures, it is clear that the execution time depends mainly on the overall number of residues present on a node, i.e. the dependence of time as a function of residues number is nearly linear, while it does not exhibit a linear dependence on the number of proteins.

The largely used *Pearson product-moment correlation coefficient* (PMCC) was computed to better assess the dependency between time and residues versus the time and proteins. In the first case, we obtained a coefficient of 0.992 and 0.995 respectively for the *CK34* and *RS119* dataset, while in the latter case we obtained only 0.582 and 0.585.

Further analysis aimed to explore whether this linear dependence was influenced by one or more slowest methods or is verified for all the methods. Figures 5.7 and 5.8 show the execution time vs the number of residues for each method for CK34 and RS119. Although FAST, USM and MacCMO perform faster as compared to Dali, CE, TM-Align, but the dependence is quite evident for each of them.

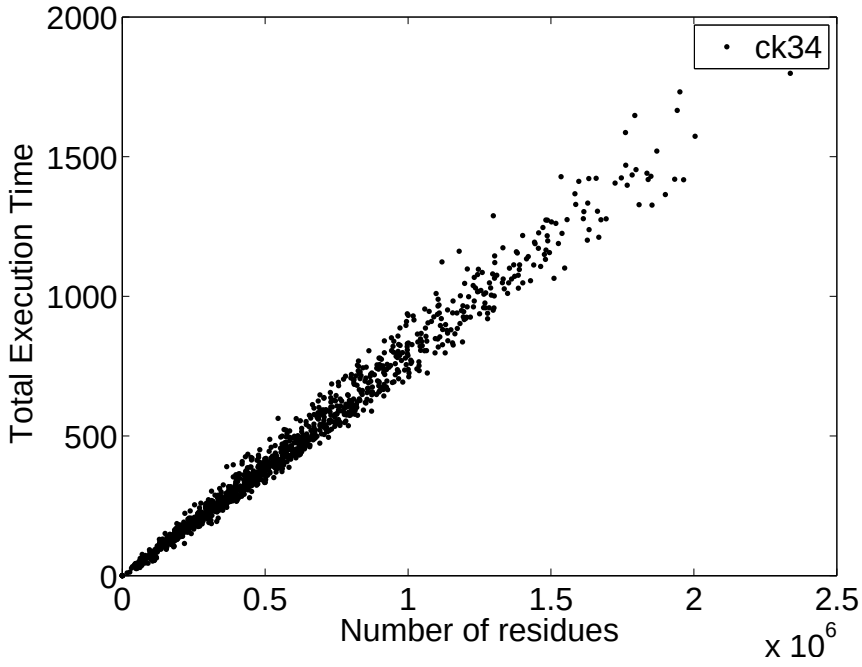


(a)

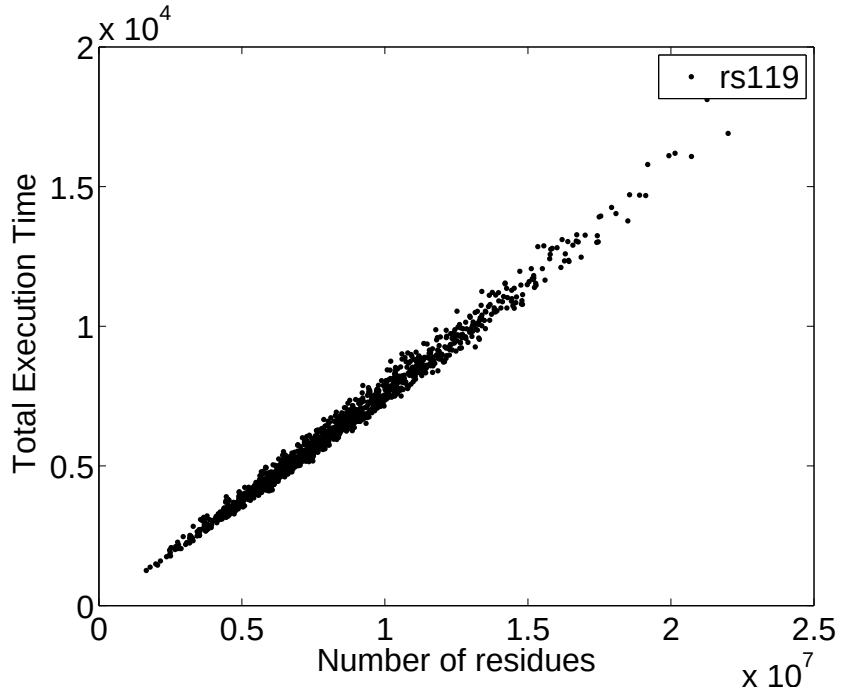


(b)

FIGURE 5.5: Execution time (s) vs number of proteins present on the node for the (a) CK34 and (b) RS119 dataset. The graph shows that the execution times exhibits a non-linear pattern in terms of the change in the number of proteins.

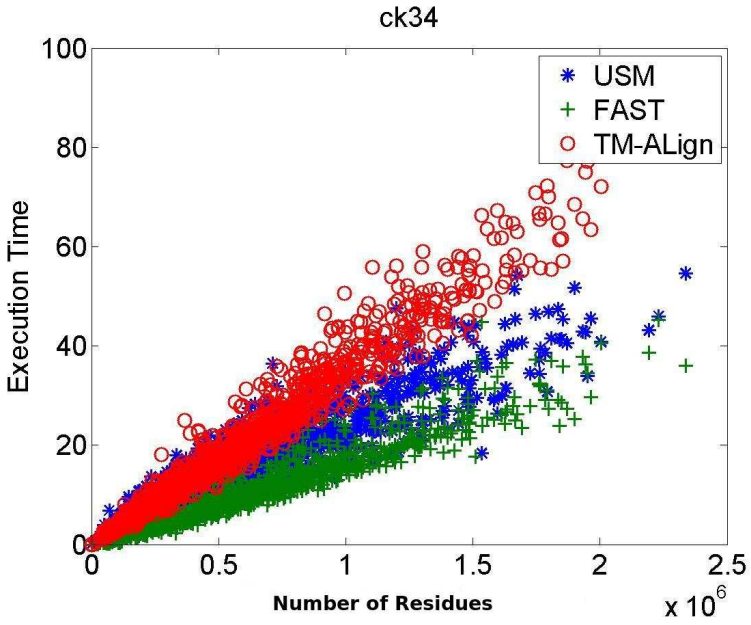


(a)

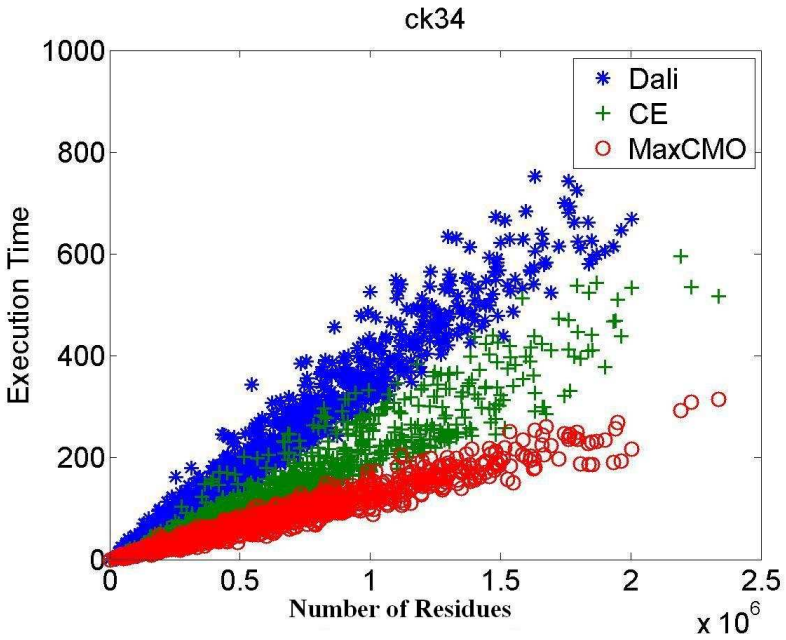


(b)

FIGURE 5.6: Execution time (s) vs number of residues present on the node for the (a) CK34 and (b) RS119 dataset. The graph shows that the execution times exhibits a linear relationship with the change in the number of residues.

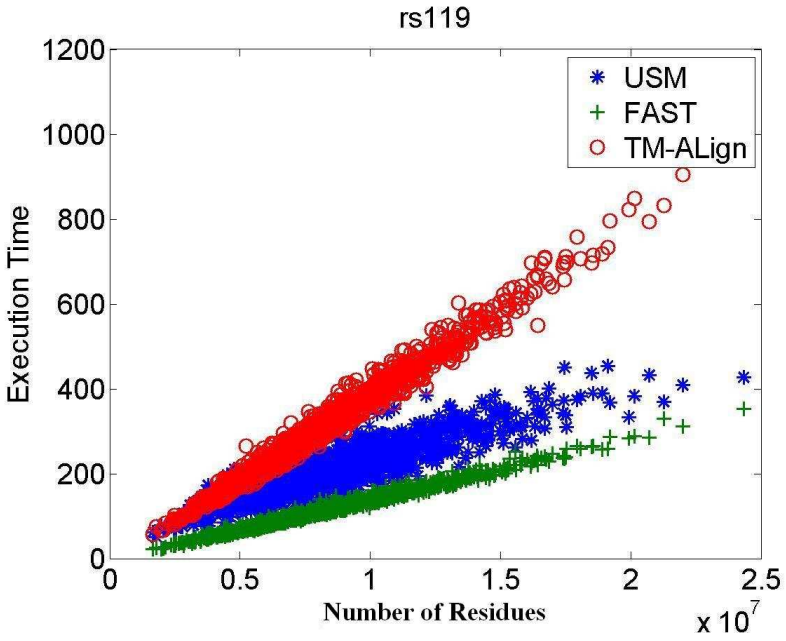


(a)

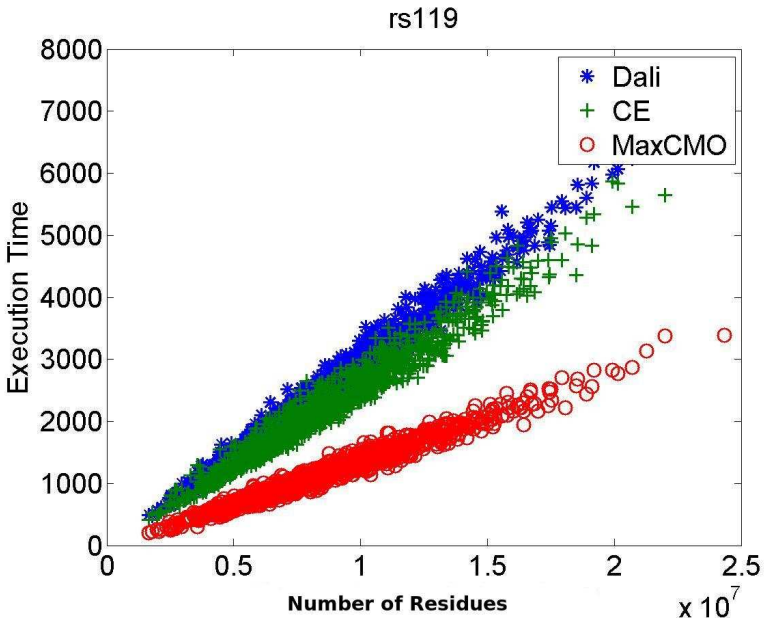


(b)

FIGURE 5.7: Execution time (s) vs number of residues present on the node for the different methods with CK34 dataset. Each method has different execution time but exhibits linear relationship with number of residues.



(a)



(b)

FIGURE 5.8: Execution time (s) vs number of residues present on the node for the different methods with RS119 dataset. Each method has different execution time but exhibits linear relationship with number of residues. Additionally, with this slightly large dataset (RS119) the linear dependence becomes even more vivid.

5.5.3 Scalability of the Uneven Decomposition

From the previous section, it is clear that the execution time strongly depends on the number of residues per node. Thus, scalability experiments for the same dataset as the *even* distribution were also conducted with *uneven* decomposition and results are reported in figure 5.9. For the RS119 (CK34) dataset, the entire execution time was reduced from about 6 days (6.2 hours), using the sequential implementation on one machine, to 3.4 hours (11.65 min.) on 64 processors. In comparison with the even strategy, we obtained an improvement on 64 processors of about 18% for the CK34 dataset and of about 29% for the RS119 dataset. Furthermore, on 64 processors, the efficiency is maintained at a quite good value of 64% for RS119. For the CHK34, we obtained a value of 50% that is not a bad result, given the small grain of the dataset.

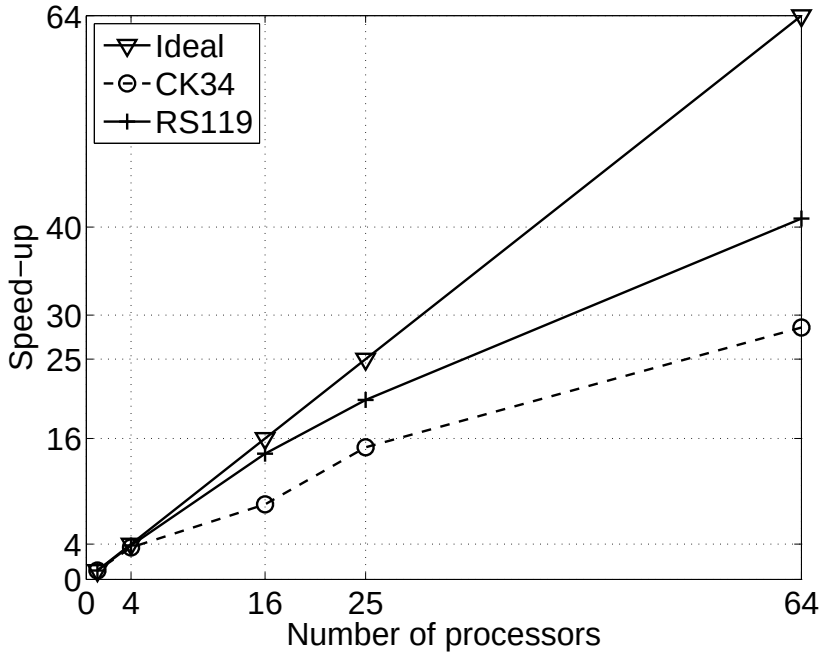


FIGURE 5.9: Speedup of the uneven decomposition using the CK34 and RS119 datasets on *spaci* cluster. The graphs shows that with CK34 dataset the speedup of the distributed algorithm (based on uneven decomposition) is around 29.9× with efficiency of 50% while the values of speedup and efficiency increase to 42.3× and 64% respectively for the larger dataset i.e., RS119.

5.6 A Further Experiment on a Large Dataset

The last experiment was performed using the the uneven strategy running on 4, 16, 25 and 64 processors applied to the Kinjo dataset, comprising of 1012 non-redundant protein chains. Using this dataset, the algorithm performed about 1 million of comparisons for all the methods.

As the execution time on a single processor is extremely large, this case was not considered, instead, scalability was measured based on an estimated base line on 4 processors running the faster of all the methods, namely, the FAST algorithm. For reference note that FAST takes approximately 11 days to execute on a single processor for such a large number of proteins. Table 5.3 shows that method USM will take approximately 2.7× the execution time of the FAST method (the factor 2.7 for USM and subsequently for other methods is based on the average of the times for CK34 and RS119 datasets as compared to method FAST). Likewise, the estimated execution time for TM-Align is 2.33×, Dali 22.78×, CE 22.93×, and for MaxCMO 8.19×, giving the aggregate factor of 58.93. As method FAST takes approximately 11 days on a single processor, the estimated time for all the methods on 1 processor would be $11 \times 58.93 \approx 648$ days or ≈ 162 days on 4 processors. The execution time of the algorithm applied to this huge dataset was reduced from 162 days on 4 processors, to 39.7 days on 16 and finally to 10.7 days on 64 processors. The scalability obtained is very close to the ideal case, as can be seen in figure 5.10. In fact, on 64 processor, respectively a scalability value of 57× and an efficiency value of 89% were measured.

5.7 Conclusions

A high-throughput/grid-aware distributed Protein structure comparison framework for very large datasets is proposed, based on an innovative distributed algorithm running both in a cluster and grid

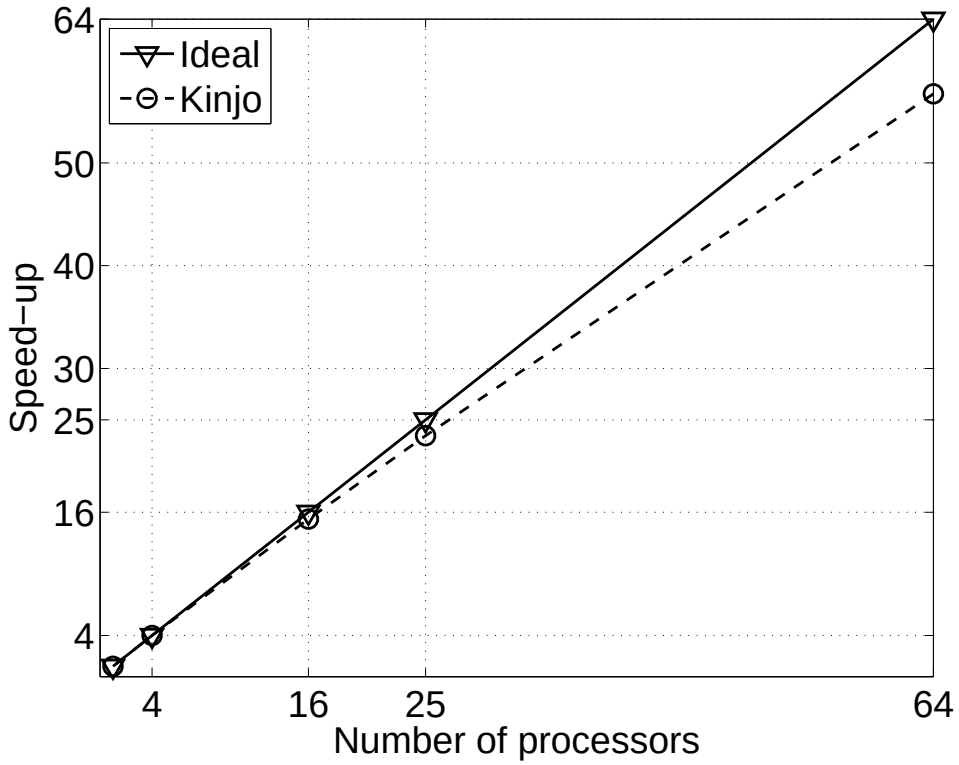


FIGURE 5.10: Speedup of the uneven decomposition using the Kinjo dataset on *spaci* cluster. The graph shows that with this large dataset the distributed algorithm (based on uneven decomposition) achieves an increased speedup of 57× and an efficiency value of 89%.

environment. This framework is able to perform structure comparison using all or a selection of the available methods. The design of this algorithm have been analyzed in terms of space, time, and communication overhead. Based on this analysis two different load balancing approaches have been used to improve the overall performance: *even* and *uneven* strategies. The former permits to obtain the best distribution in terms of memory, while the latter performs better in terms of execution time and scalability on cluster computers. Experiments conducted on medium and large real datasets prove that the algorithm permits to reduce execution time (i.e. for the RS119 dataset it was reduced from 6 days on a single processor to about 5 hours on 64 processors) and to cope with problems otherwise not tractable on a single machine as the Kinjo dataset, which took about 11 days on a

64-processors cluster.

The next chapter provides further evaluation of this distributed framework in terms of different integrated environments for MPI jobs.

CHAPTER 6

PERFORMANCE EVALUATION OF THE MC-PSC ALGORITHMS UNDER IRME FOR MPI JOBS

Chapter 5 presented the design, implementation and evaluation of the distributed framework for MC-PSC. This chapter evaluates the effect of different *Integrated Resource Management Environments* (IRMEs) on the performance of this framework in order to find the environment that could provide optimal results.

This chapter was published as a peer reviewed conference paper in *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications ISPA '08*,ISBN: 978-0-7695-3471-8, pp.817-822, 2008. [doi:10.1109/ISPA.2008.41]

6.1 Introduction

As explained in chapter 5, in order to achieve the high-end computational power needed for compute-intensive applications such as Protein Structure Comparison, the operation of these applications needs to be decomposed using some parallel programming libraries that are based on the implementations of Message Passing Interface (MPI) model. Commonly used MPI libraries include

MPICH/MPICH2 [188] [189] and LAM/MPI [241] [242] which has recently been merged into a global project under the name of OpenMPI [190]. Given that the scope of parallelism is extending from supercomputers to personal computers, workstations and networks [187]; a *converging* effect on the fields of *parallel* and *distributed* computing is being observed. It is in this perspective that the operation (submission, execution and monitoring) of MPI based jobs needs to be automated through some suitable resource management middleware (see Figure 6.1) such as Sun Grid Engine (SGE), Portable Batch System (PBS), Load Sharing Facility (LSF) etc. The specialized literature refers to the functionality provided by these software using various names for instance *Local Resource Management System* (LRMS), *Queuing System*, *Batching System*, *Workload Manager*, *Job Management System* and *Distributed Resource Manager* etc as described in chapter 2 and 3.

Normally, when MPI based parallel jobs are submitted to LRMS, it calls some external framework to launch the MPI environment for the execution of jobs [243]. If there is no appropriate coordination between the external framework and LRMS, then the launched jobs may get dispatched on somewhat different resources than what had been allocated by the LRMS, creating, on the one hand, resource overloading and conflicting problems, on the other hand, providing inaccurate or no resource usage and monitoring information. Both of these problems could be eliminated if certain means are used to integrate the operation of LRMS and MPI environment [243].

If the integration is achieved in such a way that it removes only the first problem namely the launching of parallel jobs doesn't create overloading and resource conflicts, then the integration is said to be *Loose Integration*. In Loose Integration, the LRMS only sends information to MPI environment regarding which resources a particular job should run on but the job runs in isolation from LRMS and hence no resource usage and monitoring information is obtained by the LRMS.

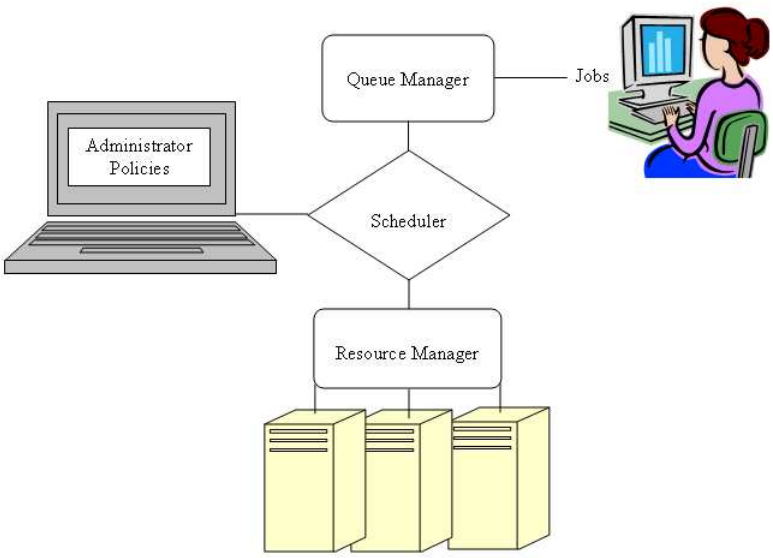


FIGURE 6.1: Three main components of a *Local Resource Management System* (LRMS). Each newly submitted request for the execution of user jobs is taken care by the *Queue Manager* (e.g. *sge_qmaster daemon of SGE*), which stores these jobs in a queue and requests the Scheduler (e.g. *sge_schedd daemon* of SGE) for their execution. The Scheduler gets job information from the Queue Manager and optimally decides when and where to execute these jobs based on the information obtained from the Resource Manger (e.g. *sge_execd/sge_shepherd daemons* of SGE) and system administrator's policies. This type of automated system frees the user from manually looking if the required resources for the execution of a job are available and to manually start the job on each node.

However, if the integration is achieved in such a way that it removes both of the problems then it is said to be a *Tight Integration*, which enables the LRMS to run the MPI jobs under the control of its *Job Controlling Agent* so as to avoid the problem of resource overloading and getting full resource usage information. This chapter aims to evaluate the performance of PSC jobs running under such an integrated architecture configured with different approaches in order to provide a scientific basis for decision making while building a large scale e-Science infrastructure that meets with the requirements of today high-impact scientific applications.

The remainder of this chapter is organized as follows: section 6.2 provides an overview of the parallel libraries and LRMS that has been used in this study; it also provides the details of

the implementation and results of the integration; section 6.3 provides the details of the testbed and datasets used for the experimentations; finally section 6.5 concludes the chapter.

6.2 Integrated Resource Management Environment (IRME) for MPI Jobs

This study is based on the use of two different implementations of MPI libraries namely MPICH2 [188] [189] and Open MPI [190]. The description of these libraries is presented in Table 4.3. The local resource management system is described in the following section.

6.2.1 Sun Grid Engine (SGE)

SGE[1] provides management solutions for *cluster grids*. SGE uses Unix as its native environment and implements most of its services and other entities as Unix daemons. The use of Unix as a native environment allows SGE to make use of already developed standard and stable services and protocols for communication and computation such as **S**ecure **Sh**ell (SSH), **R**emote **Sh**ell (RSH), Network File System (NFS), and Remote Procedure Call (RPC) etc. Normally, SGE based cluster grid consists of four types of hosts and different daemons keep running on each host. A brief description of these hosts is given bellow:

Job Server (execution name/command 'sge_qmaster'):

This daemon controls the overall operation of SGE. It interacts with other daemons to provide job management and monitoring services and information. In case of any errors it records the diagnostics messages in the directory `SGE_ROOT/<qmaster_spool_dir>/messages`.

Job Executor (execution name/cammand 'sge_execd'):

[1] SGE; the Sun Grid Engine is an open source and cross platform resource management system developed and maintained by Sun Microsystems. It also has a commercial version named 'Sun N1 Grid Engine (SN1GE)'

This daemon reads the local queue and places the job for execution in conjunction with Job Controller daemon. In case of errors, the daemon records the diagnostic messages in the directory `SGE_ROOT/<execd_spool_dir>/<hostname>/messages`.

Job Controller (execution name/cammand 'sge_shepherd'):

This daemon provides parent process functionality for a single job, and hence enables the SGE to get the resource usage information after the job finishes. It gets the information from Job Executor regarding which job to execute and starts up to 5 child processes for each job. These child processes include the prologue (if enabled) , startup of the parallel environment (if the job is parallel), startup of the job itself, shutdown of the parallel environment and finally one process for epilogue (if enabled). The prologue, epilogue, and parallel environment startup and shutdown scripts are site specific and are provided by the administrator while configuring the cluster. In case of any errors it record the messages in the Job Executor's directory.

Job Scheduler (execution name/command 'sge_schedd'):

For each new job the Job Server contacts the Job Scheduler daemon. This daemon reads the job's profile, gets the site's policy information as configured by the administrator, requests the resource information from Job Executor and finally communicates the decisions to Job Server regarding which queue this job be submitted for execution. In case of any errors the daemon records diagnostic messages in the directory `SGE_ROOT/<qmaster_spool_dir>/schedd/messages`.

In addition to different daemons, the SGE infrastructure consists of another entity called '*queue*'. A *queue* is used as a container for jobs. There may be several types of queues characterizing

different types of resources and environments. The SGE scheduler daemon performs assignment of jobs to appropriate queues based on the information from job's profile as well as the system load distribution. In order to accommodate those jobs which need to be scheduled at the same time and which require to communicate among themselves (the parallel jobs), SGE provides an interface to define parallel environments (PE). Different ways of integration of the SGE and parallel environments are further described in the following section.

According to [244], though MPICH2 provides different process management methods such as *gFoker*, *MPD*, and *SMPD*, only the later could be used for tight integration as it satisfies the requirement of a *single process group* on each node of the cluster. The SMPD method by itself could be used in two different ways i.e. *SMPD daemon-based* (SMPD-DB) and *SMPD daemon-less* (SMPD-DL). With SMPD-DB a single daemon per node is used to manage all the processes running at that node. In case of multi-processor node the resource usage information of all other processors is aggregated to the one and same daemon and hence the SGE would record the overall resource usage information per node through a single *Queue Remote Shell* (qrsh) invocation when the job finishes its execution. Whereas, with SMPD-DL each process starts on its own on each node and SGE needs to record each *qrsh* invocation for each process separately. For the sake of comparison we used both SMPD-DB and SMPD-DL to perform Loose and Tight Integration using the distinct configuration parameters as listed in Table 6.1.

6.3 Testbed and Datasets

In order to evaluate the performance of PSC jobs under both (*i.e.* Loose and Tight) ways of integration, a testbed consisting of a *heterogeneous* Linux cluster (resources shown in Table 6.2) has

TABLE 6.1: Distinct configuration parameters for Loose Vs Tight integration of SGE and MPICH2.

Parameter	Loose Integration	Tight Integration
start_proc_arg	/sge/mpich2_smpd/startmpich2.sh $pe_hostfile /local/mpich2_smpd	/sge/mpich2_smpd/startmpich2.sh $pe_hostfile -catch_rsh /local/mpich2_smpd
stop_proc_arg	/sge/mpich2_smpd/stopmpich2.sh /local/mpich2_smpd	/sge/mpich2_smpd/stopmpich2.sh -catch_rsh /local/mpich2_smpd
control_slaves	FALSE	TRUE
job_is_first_task	TRUE	FALSE

TABLE 6.2: Hardware Configuration of the Cluster Grid. All hosts communicate through 100 Mbps Ethernet cards

Host	CPU(GHz)	Cache(KB)	RAM(GB)	HD(GB)
comp1	2x P4-1.86	4096	2.0	250
comp2	2x P4- 3.0	512	1.0	80
comp3	P4-2.4	512	0.5	20
comp4	P4-1.7	256	1.0	40
comp5	2x P4-3.6	1024	2.0	80
comp6	P4-2.4	512	0.5	40

been used. The cluster consists of 6 nodes with a total of 9 slots giving an aggregated CPU power of 25.82 GHz and main memory of 7 GB.

6.3.1 Datasets

Chapter 5, worked with datasets which have already been used in the literature. Developed by different authors; these datasets do not exhibit any regularity in terms of number of structures. The regularity would however be useful in observing the continuous relationship between the execution time of PSC jobs and the size of the dataset under different software environments and configurations. To this aim, three subsets of regularly increasing number of structures have been randomly selected from the dataset of Kinjo et al. [223]. These datasets were prepared with the same algorithm and characteristics as described in section 5.2. Table 6.3 provides further details of these datasets

TABLE 6.3: Datasets used in the experiments. Hash sign (#) represents the word *'Number of'*

Dataset	# of Chains	# of comparisons	Size in MB
SFBK-250	250	62,500	33.1
SFBK-500	500	250,000	66.5
SFBK-1000	1000	1,000,000	137.5

and Figure 6.2 shows the effect of the increasing number of protein chains on the computational time taken by FAST [50], one of the structure comparison methods we used in this study.

6.4 Results and Discussions

In order to evaluate the goodness and weakness of the integrated environment described in the previous section, we run the FAST algorithm using three datasets listed in Table 6.3. The experiments were performed on 6 heterogeneous nodes (9 processors) of a Linux Cluster with the characteristics described in Table 6.2. Figure 6.3A, gives an illustration of a PSC job running FAST algorithm under loose integration, whereas Figure 6.3B illustrates the operation of the same MPI job running under tightly integrated environment.

Table 6.4 shows the computational time in terms of *wall clock* time, *user* time, *system* time and CPU time required for the execution of one of the protein structure comparison algorithms named FAST [50] with different datasets under the integrated resource management environment for MPI jobs that has been configured with different approaches. *Wall Clock* time is the real time taken by the algorithm as perceived by the user and includes CPU time, I/O time and communication time e.g. between different nodes, *User* time is part of the cpu time taken by the user application (i.e. FAST algorithm) itself; *System time* is the part of cpu time taken by the system software; and *CPU* time is the sum of *user* and *system* times. It can be seen in Table 6.4, that in terms of

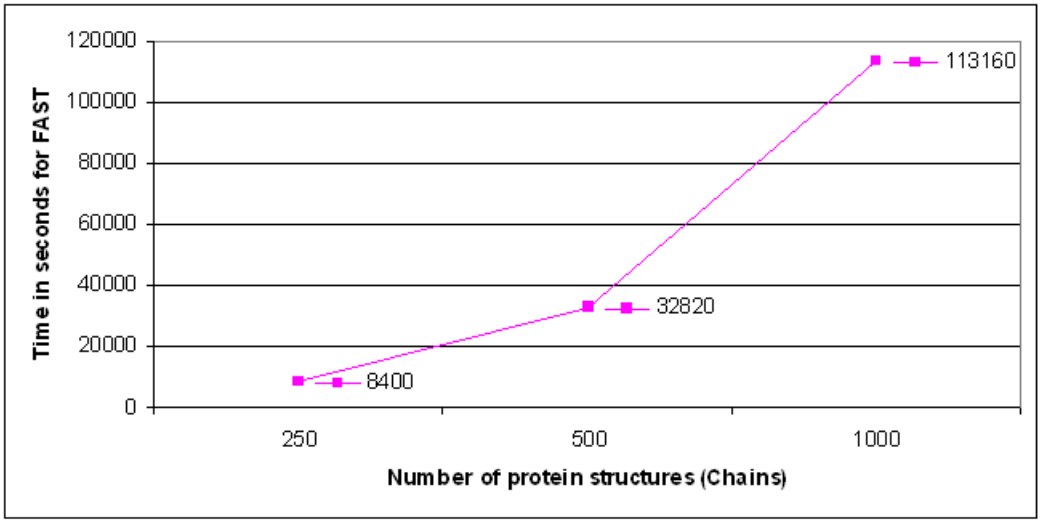


FIGURE 6.2: Computational time for PSC algorithm (FAST) on P4-2.4GHz uni-processor machine with increasing number of proteins (single chain based PDB files) in the dataset. The graph shows that the execution time increases with respect to increase in the number of proteins; however, the change is not linear because of the different number of residues i.e. length of the structures.

wall clock time, the algorithm finishes earlier under Loose integration as compared to the Tight integration. The reason beyond this large variance is the fact that with loose integration we only get the accounting information for the master process which runs on only one processor and hence it seems to be a fraction of the total wall clock time as reported in tight integration. Also, with Loose integration the *user*, *system* and *cpu* times (accounting information) can not be noted as the mpi job doesn't run under the control of job controlling agent *sge-shepherd*. Furthermore, in Loose integration the SGE is unable to get proper monitoring (state) information in case if the mpi job goes through any problems. An example of such job that SGE was unable to get information about is one suffering with an mpi related error:

> "unable to read the cmd header on the pmi context, socket connection closed, error stack: MPIDU_Socki_handle_read(607): connection closed by peer ..."

This job took far beyond its expected execution time and the SGE did not notify the user

and the job was killed manually afterward. Being able to keep track of which pairs of structures have been (un)successfully compared and by which method is an essential feature of modern robust comparison servers such as [27]. Hence, this is a potential weakness of *Loose* integration.

Referring to Table 6.4 again, it could also be observed that for a small number of proteins (e.g. *SFBK-250 dataset*) the performance of both SMPD-DB and SMPD-DL types of tight integrations is almost same, however, for larger datasets (e.g. *SFBK-500* and *SFBK-1000*), the former outperforms the later one. This is because in case of SMPD-DL, SGE needs to record each *qrsh* invocation and hence incurs an extra overhead. Tables 6.5 and 6.6 illustrate some interesting differences between *SMPD-DB* and *SMPD-DL* types of tight integrations. Table 6.5 shows that SMPD-DB integration doesn't expose the dual-processor nature of some of the nodes (e.g. *comp1*, *comp2*, and *comp4*; Table 6.2). However, it utilizes both of the processors of all dual-processor nodes implicitly as can be seen from the values of CPU times for comp2 and comp5 in Table 6.5, which are even greater than the value of their corresponding *wall clock* times. Table 6.6 shows that the overall performance of the SMPD-DL tight integration gets degraded because of the longer CPU time taken by the slowest node (*comp4*; Table 6.2). This longer CPU time could be because of the slowest speed of certain nodes in subject as well as because of the longer lengths of protein chains being compared by FAST algorithm at that node. It is because of this long running node that faster CPUs also go through long idling cycles.

TABLE 6.4: Resource usage comparison for FAST algorithm [50] with different datasets using Loose and Tight integration of SGE and MPICH2 with SMPD daemon-based (SMPD-DB) and daemon-less (SMPD-DL) startup methods.

Dataset	Time	**Loose Integration**		**Tight Integration**	
	[hh:mm:ss]	SMPD-DB	SMPD-DL	SMPD:DB	SMPD:DL
SFBK-250	Wall clock	00:10:43	00:13:51	00:59:18	00:52:27
	User	-	-	00:50:18	00:45:46
	System	-	-	00:08:05	00:06:25
	CPU	-	-	00:58:23	00:52:11
SFBK-500	Wall clock	00:51:12	01:01:29	01:38:28	02:11:48
	User	-	-	01:22:11	01:50:25
	System	-	-	00:13:18	00:18:51
	CPU	-	-	01:35:39	02:09:16
SFBK-1000	Wall clock	02:22:25	02:22:36	05:39:36	07:44:04
	User	-	-	04:34:06	06:28:56
	System	-	-	00:49:15	01:08:34
	CPU	-	-	05:23:22	07:37:30

TABLE 6.5: Time distribution for FAST algorithm with *SFBK-1000* dataset over nine processor using tight integration with SMPD daemon-based (SMPD-DB) method.

	comp1	**comp2**	**comp3**	**comp4**	**comp5**	**comp6**
Wall Clock [hh:mm:ss]	05:39:37	05:39:37	05:39:37	05:39:37	05:39:37	05:39:37
User [hh:mm:ss]	03:43:17	05:05:48	02:56:23	03:40:22	07:21:00	04:37:37
System [hh:mm:ss]	00:33:37	01:02:10	00:32:29	00:41:13	01:09:55	00:56:36
CPU [hh:mm:ss]	04:16:28	06:08:07	03:28:52	04:21:35	08:30:55	05:34:13
Mem [MB]	31.06	66.08	60.87	43.53	95.83	104.53

TABLE 6.6: Time distribution for FAST algorithm with *SFBK-1000* dataset over nine processor using tight integration with SMPD daemon-less (SMPD-DL) method.

	comp1 cpu-1	**comp1** cpu-2	**comp2** cpu-1	**comp2** cpu-2	**comp3**	**comp4**	**comp5** cpu-1	**comp5** cpu-2	**comp6**
Wall Clock [hh:mm:ss]	07:44:05	07:44:05	07:44:05	07:44:05	07:44:05	07:44:05	07:44:05	07:44:05	07:44:05
User [hh:mm:ss]	01:29:22	02:01:50	03:55:15	03:00:35	02:31:28	06:28:56	02:57:52	02:14:24	03:20:42
System [hh:mm:ss]	00:14:33	00:17:54	00:40:19	00:29:51	00:29:50	01:08:34	00:30:41	00:24:15	00:45:00
CPU [hh:mm:ss]	01:43:56	02:19:44	04:35:34	03:30:26	03:01:18	07:37:30	03:28:33	02:38:39	04:05:42
Mem [MB]	10.66	29.42	63.66	41.75	22.69	161.77	24.65	30.56	48.44

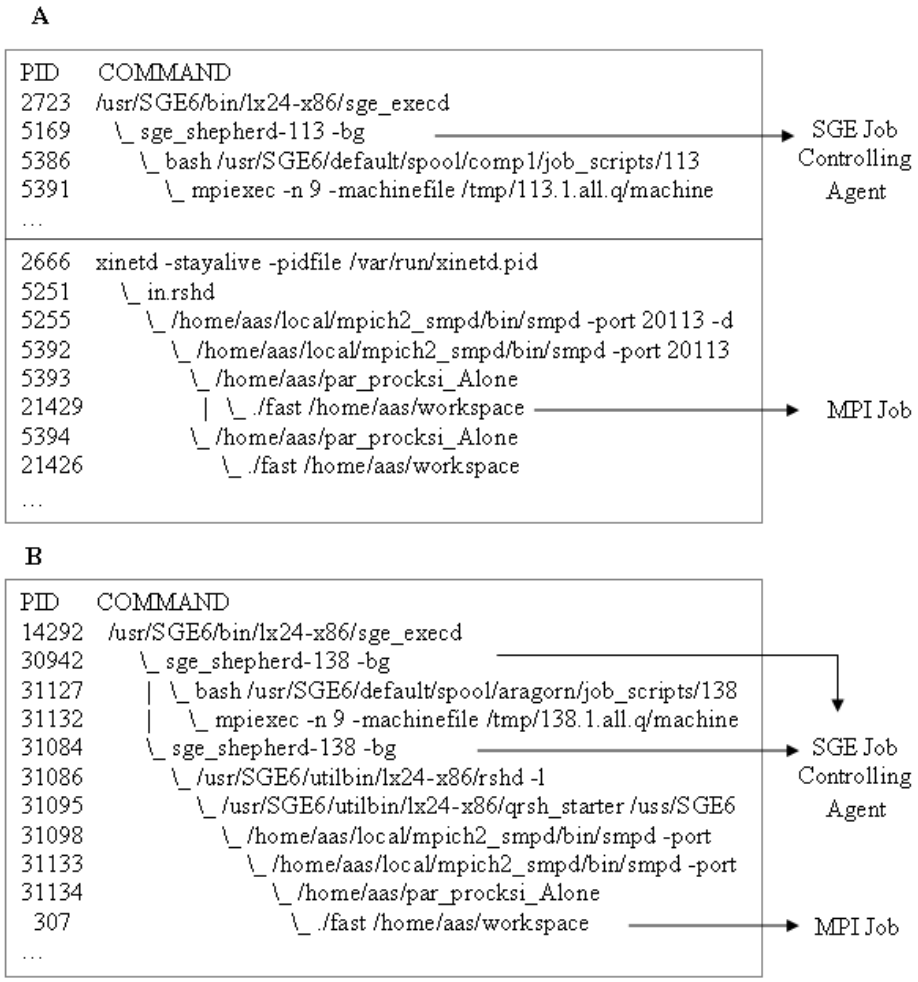


FIGURE 6.3: Parallel jobs running under integrated resource management environment for SGE and MPICH2. **A**: Loose Integration; SGE launches the parallel job through its job controlling agent *sge_shepherd* using *mpiexec* command by specifying the resources to be used (through *machine* file) and hence eliminating the problem of resource conflicting. However, the actual MPI job (FAST algorithm) doesn't run under the control of *sge_shepherd* and hence no resource usage (accounting) information could be achieved. **B**: Tight Integration; SGE uses two instances of its job controlling agent *sge_shepherd*, one for launching the parallel job by specifying the resources to be used and other for controlling (monitoring) the operation of the running MPI job (FAST algorithm) and hence providing the full resource usage (accounting) information.

6.5 Conclusions

Performance of the distributed framework for *Protein (Structure) Comparison, Knowledge, Similarity and Information* (ProCKSI) has been evaluated (using FAST algorithm as an example) under integrated resource management environment for MPI jobs. Results of the evaluation indicate that *Loose Integration* method is not much reliable in terms of accounting and monitoring information to be used for PSC jobs. Furthermore, for larger datasets, Tight Integration with SMPD daemon-based method outperforms its counterpart i.e. SMPD daemon-less method. It has also been learned that in a heterogeneous cluster where some nodes have double the performance as that of others, the slowest node could become a bottleneck for the overall performance of the system. This performance degradation could deteriorate further when that slowest node is assigned PSC jobs with longer protein chains. This type of problem could be solved with the development of some intelligent heuristics to perform better load balancing among the nodes.

Furthering the process of evaluation, the next chapter presents the results of evaluation under the Grid environment that consists of more than one clusters and uses grid-enabled version of MPI library to spawn the jobs across the sites.

CHAPTER 7

ON THE SCALABILITY OF MULTI-CRITERIA PROTEIN STRUCTURE COMPARISON IN THE GRID

The effect of different integrated environments on the performance of the distributed framework was discussed in chapter 6. This chapter, considers the process of evaluation of the distributed framework under the Grid environment consisting of more than one clusters in order to analyze the effect of inter-cluster communication overhead and other issues related to the Grid.

Parts of this chapter were published as a peer reviewed conference paper in the *Proceedings of The Euro-Par 2010 Workshop on High Performance Bioinformatics and Biomedicine (HiBB),* August 31-Sep 3, 2010 , Ischia, Naples, Italy.

7.1 Introduction

In order to achieve the consensus-driven decision making, the MC-PSC requires to execute a given set of methods on the set of protein structures to be compared. Given the computational requirements of each method and the ever growing number of entries in the PDB, the realistic computation of the MC-PSC becomes a grand challenge and hence requires the use of Grid computing to over-

come the limitations of a single parallel computer/cluster. The use of the Grid computing on the other hand also introduces additional communication overhead and hence the standard parallel computing measures and metrics such as *Speedup* and *Efficiency* need to be redefined [225]. This chapter uses the grid-based definition of the speedup (*Grid Speedup*), and efficiency (*Grid Efficiency*) introduced by Hoekstra et al. [225] (as described in chapter four, section 4.6.1), to measure the scalability of our distributed algorithm on the *UK National Grid Service* (NGS) [66] architecture. The code of the distributed algorithm is same as has been used in chapter 5. The results of cross-site scalability would be compared with single-site and single-machine performance to analyze the additional communication overheard in a wide-area network.

The remainder of this chapter is organized as under: section 2 describes the experimental setup; section 3 presents the results and discussions and finally section 4 presents the conclusion.

7.2 Deployment on the NGS Infrastructure

The *National Grid Service* (NGS), provides the *eScience* infrastructure to all the UK-based scientists free of cost [66]. For our case we used the Globus-based MPIg [191, 192] (grid-based implementation of MPI) to spawn the jobs across two NGS sites; one at Leeds and the other at Manchester. Like its predecessors (e.g MPICH-G [245] and MPICH-G2 [191]), the MPIg library extends the Argonne MPICH implementation of MPI to use services provided by the Globus Toolkit (GT) [34] for cross-site job execution using IP-based communication for inter-cluster messaging. However, being the latest implementation, the MPIg includes several performance enhancements such as in the case of inter-cluster communication it uses multiple threads as compared to the single thread communication of the previous implementations. Furthermore, besides being backward compatible with

the pre-web service Globus, the MPIg also makes use of the new web services provided by Globus version 4 [35]. By making use of the new web services, the MPIg provides much more enhanced functionality, usability and performance. The use of the MPIg for cross-site runs requires advanced resource reservation so that jobs (processes) can run simultaneously across all the sites. To facilitate this, NGS provides the *High-Available Resource Co-allocation* (HARC) [246] as a command line utility to perform automatic reservation. Each of the two NGS sites (Leeds and Manchester) consists of 256 cores (AMD Opteron with 2.6GHz and 8GB of main memory) interconnected with Myrinet M3F-PCIXD-2. However, the NGS policies allow the advance reservation of maximum of 128 cores at each site for the maximum duration of 48 hours. Once the reservation is done, then the Globus-based job submission could be achieved with the *Resource Specification Language* (RSL) scripts and other Globus services could be used for job monitoring and control. For the MPI based jobs to run on different sites, the source code of the application needs to be compiled with MPIg libraries at each site and the executable placed in the appropriate working directory under the respective local file system. The compilation of the MPI based application with MPIg does not require any change in the source code and hence from the user's perspective the deployment is as straight forward as running the parallel application on a single site/cluster with the exception that the RSL scripts specifies the resources of the additional site to be used. Figure 7.1, shows the overall architecture and setup of deploying the MC-PSC application on the Grid.

The dataset used in these experiments is the one introduced by Kinjo et al [223] consisting of 1012 non-redundant protein chains having a total of $252,569$ residues. The 1012 chains result in as many as $1,024,144$ pairwise comparisons for each method/algorithm. While using all the six methods (e.g. the *Universal Similarity Metric* (USM) [48], *Maximum Contact Map Overlap*

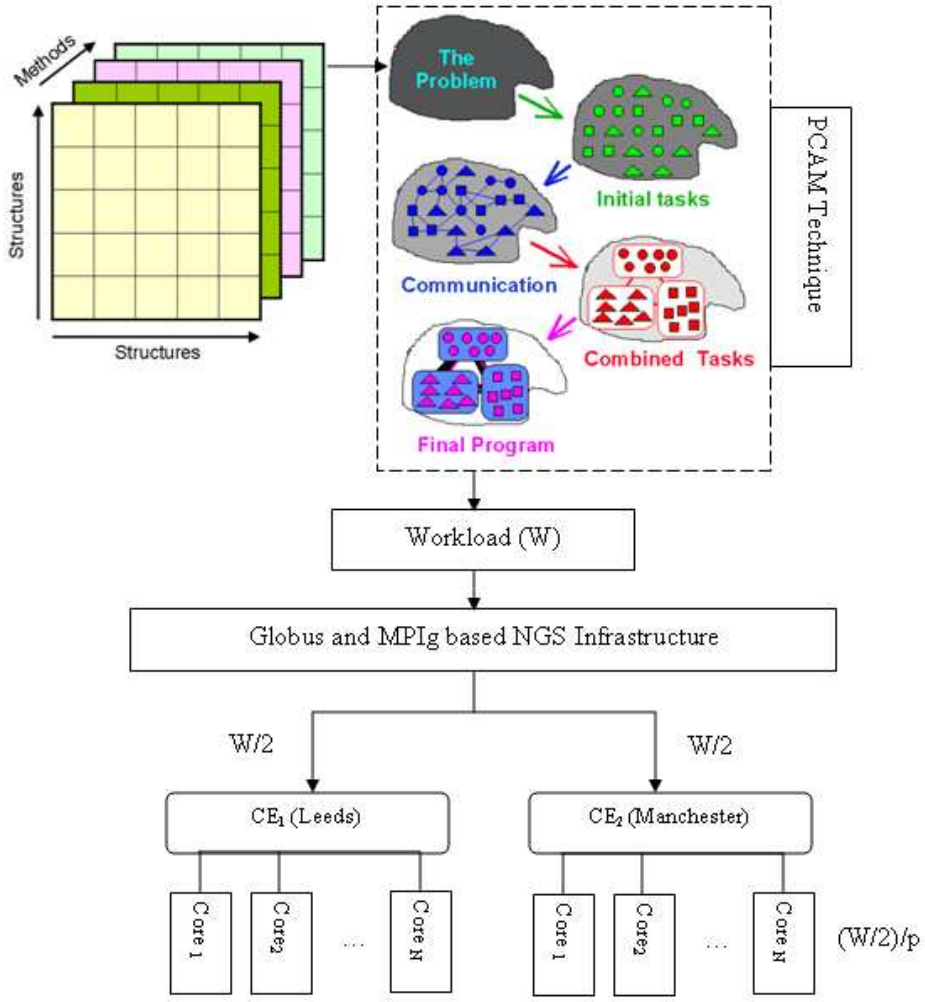


FIGURE 7.1: Deployment of the MC-PSC application on the Grid: the application takes protein 3-D structures as input and prepares the balanced workload W to be distributed on the Grid. Half of the total workload (W/2) is assigned to each site (CE). Each site further distributes the W/2 into p number of cores.

(MaxCMO) [46], *Distance Alignment Matrix* (DaliLite) [45], *Combinatorial Extension* (CE) [47], *Fast Alignment And Search Tool* (FAST) [50] and TM-Align [49]), the total number of pairwise comparisons becomes $1,024,144 \times 6 = 6,144,864$. Given that the average time for the comparison of 1 pair using all the six methods on a single processor machine is about 8 secs, this computation requires about 569 days to complete on a single processor and it took about 10.7 days to complete on a 64-node cluster [247]. The results achieved for this dataset on the Grid infrastructure are reported

in the next section.

7.3 Results and Discussions

Both the single-site and cross-site experiments for MC-PSC were conducted with varying number of processors using the Kinjo et al [223] dataset. The Grid speedup and efficiency (4.6.1) were calculated based on the results of these experiments and are shown in figure 7.2 and 7.3. Figures 7.2 shows that initially (for less number of processors), running the MC-PSC experiments across two sites almost doubles the performance to that of the single-site. However, as the number of processors increases (thereby decreasing the level of granularity and increasing the communication overhead), the speedup decreases slightly and finally reaches to about 1.65. There is also same trend in the Grid efficiency as shown in figure figure 7.3.

Figures 7.4 provides the comparison of the algorithmic speedup on a single-site (S_1, having 128 processors) and the speedup obtained while running the experiments on the two sites (S_2, having a total of 256 processors). The speedup in this case is taken as the ratio of the execution time on single-machine (single processor) (T_1) to the execution time on p processors (T_p) (i.e $S_1 = S_2 = \frac{T_1}{T_p}$). As indicated by Figure 7.4, though initially, the cross-site speedup is slightly low as compared to the single-site speedup; however, given the large number of processors available on the later, the overall speedup increases by almost a factor of 2. The total time for the computation of the given dataset on 256 cores (2.4GHz each) was reduced to 38.6 hours. Comparing this with the 569 days on the single-machine and 10.7 days required on a 64-node (though having less processor power i.e 1.4GHz each) cluster we observe a good scalability and performance of our algorithm on the Grid. The boost in the speedup and performance is two folds i.e the large number of processors

(*physical speedup*) coupled with high speed of each individual processor (*power scalability*). Figure 7.5, shows the corresponding efficiency of the algorithm on single-site and cross-site architecture. The efficiency, in this case measures the effective use of the hardware and is equal to the ratio of the speedup on p processors to p (i.e $E = \frac{S_p}{p}$). Figure 7.6 shows the cross-site communication overhead in terms of running the MC-PSC application in the Grid. The cross-site communication overhead is measured as the difference in execution time when the MC-PSC is run on single and double sites with same number of processors. For example, if T_1 is the execution time on one site with 4 processors and T_2 is the execution time on two sites with 4 processors (i.e., 2 processors per each site), then the cross-site communication overhead would be $T_1 - T_2$. Figure 7.6 shows that when a few processors are used the load of the processors and consequently the amount of data to be exchanged is high and consequently there is considerable communication overhead. However, when we use a larger number of processors, the overhead is negligible in comparison with the computation time.

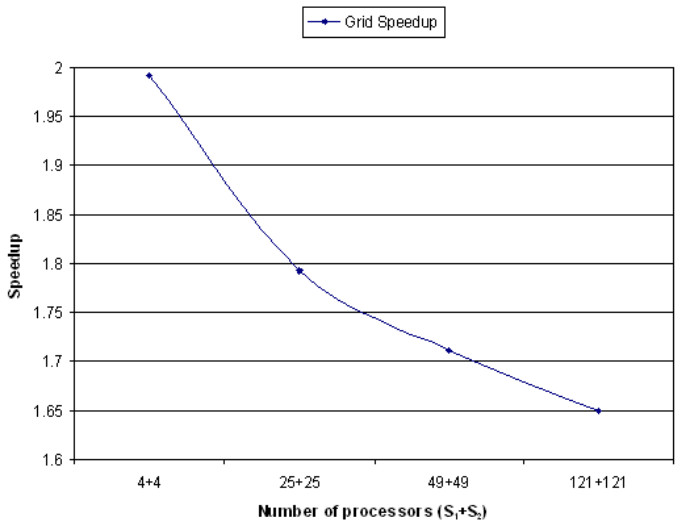


FIGURE 7.2: Performance of the MC-PSC on the Grid: grid speedup (eq 4.5); initially the speedup is almost ideal for less number of nodes but as the number of nodes increases on each site the corresponding level of granularity decreases while the the level of communication overhead increases and hence it causes the speedup to degrade slightly. Nevertheless, the overall speedup is much greater (1.6) as compared to speedup on the single site (<1).

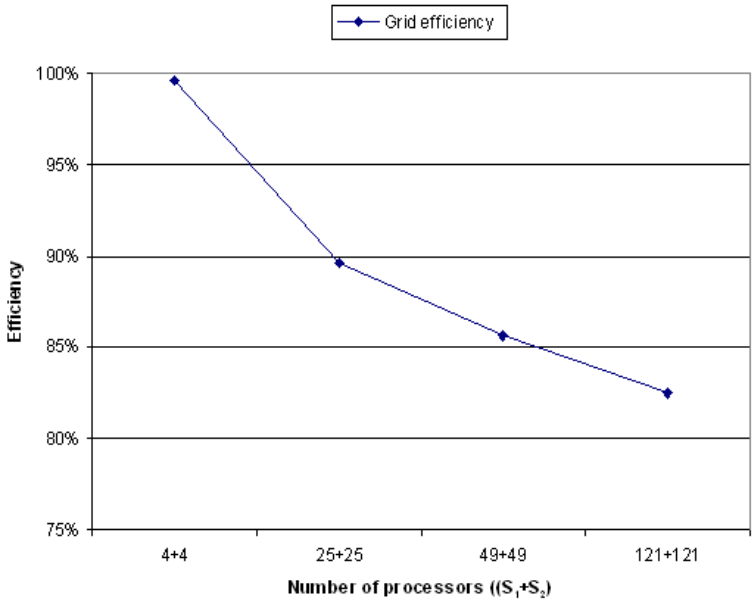


FIGURE 7.3: Performance of the MC-PSC on the Grid: grid efficiency (eq. 4.6); as expected the slight degradation of speedup causes the degradation in the efficiency of the system.

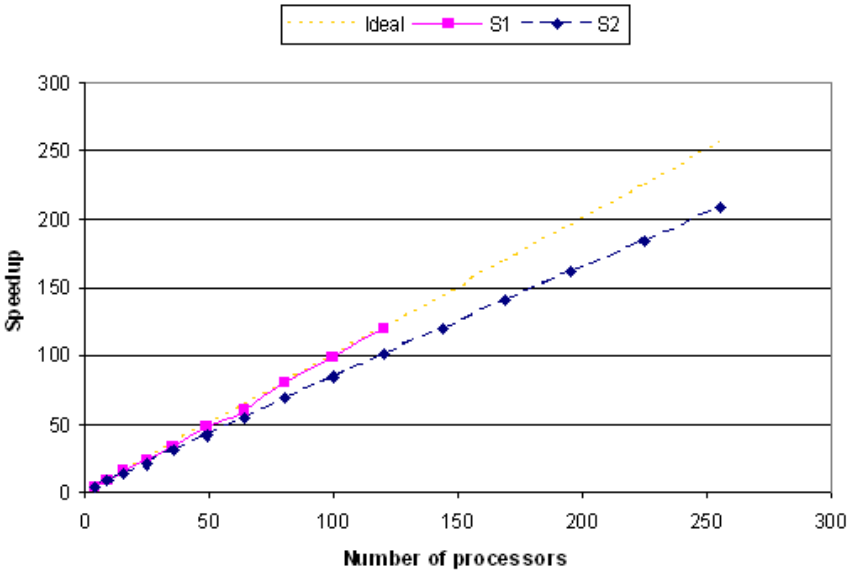


FIGURE 7.4: Single-site and cross-site speedup; the graph shows that though initially, the cross-site speedup ($S2$) is slightly low as compared to the single-site speedup ($S1$); however, given the large number of processors available on the later, the overall speedup ($S2$) increases by almost a factor of 2.

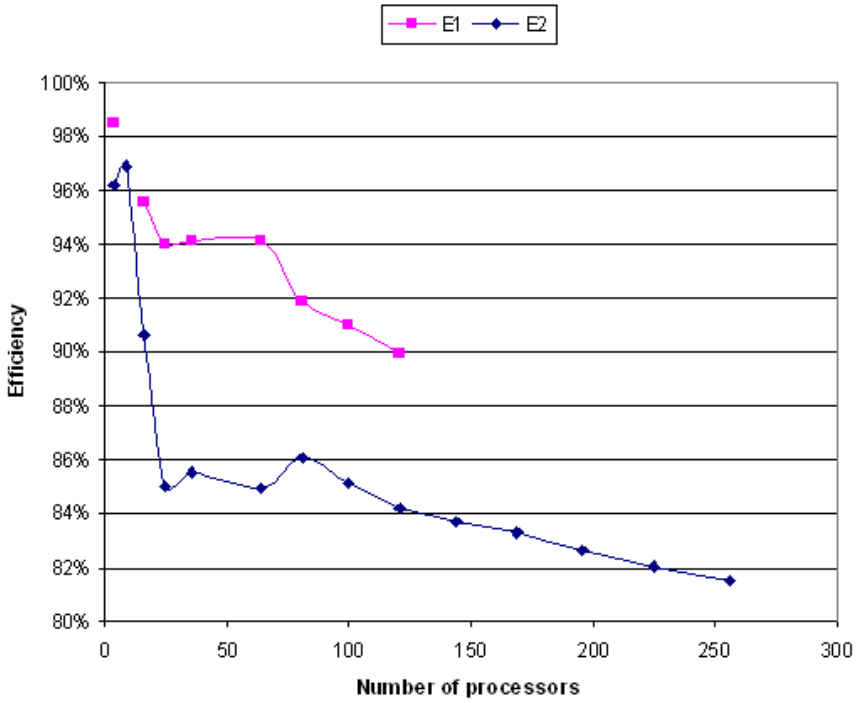


FIGURE 7.5: Single-site and cross-site efficiency; as expected the cross-site efficiency ($E2$ is slightly less as compared to the single-site efficiency ($E1$ due to extra communication overhead.

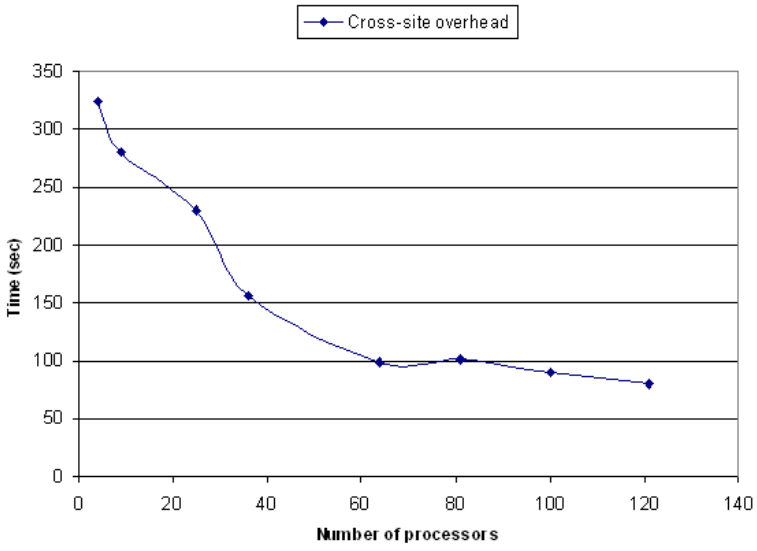


FIGURE 7.6: Cross-site communication overhead. The graph shows that when a few processors are used the load of the processors and consequently the amount of data to be exchanged is high and consequently there is considerable communication overhead. However, when we use a larger number of processors, the overhead is negligible in comparison with the computation time.

7.4 Conclusions

The quality of our parallel algorithm for MC-PSC has been measured in terms of Grid Speedup and efficiency. The results of the single-site and cross-site experiments indicate that by making use of the Grid resources, the algorithm scales well and that the cross-site communication overhead is not much significant. The current cross-site experiments were conducted on only two sites based on the HCG model of the National Grid Service (NGS), UK. As the NGS is still in the process of adding more sites, in future we would like to extend this study by increasing the number of sites as well as incorporating the heterogeneous architecture of the Grid. Because, at present the maximum time allocated for continuous execution of a job/process at NGS is limited to 48 hours and hence does not allow evaluating the performance of the application with very large datasets, hence the software developed so far could be upgraded by adding the fault tolerance mechanism in the form of checkpoint/restart. The checkpoint/restart mechanism could be added without changing the code of the application by using some libraries such as the *Berkeley Lab Checkpoint/Restart* (BLCR) [156]. With these improvements, it would be possible for the MC-PSC to perform real time computation with even large datasets and to develop a database of pre-computed results.

The next chapter is going to discuss the storage, management and analysis of (multi) similarity data resulting from the comparison of large scale protein structure datasets in the Grid.

CHAPTER 8

ON THE STORAGE, MANAGEMENT AND ANALYSIS OF (MULTI) SIMILARITY DATA FOR LARGE SCALE PROTEIN STRUCTURE DATASETS IN THE GRID

So far, in the previous chapters the design, implementation and evaluation of the distributed framework for MC-PSC has been described in terms of its execution time, communication overhead and load balancing strategies in different cluster and Grid environments. This chapter and the one that follows, present the discussion on the storage, management and analysis of (multi) similarity data resulting from the comparison of large scale protein structure datasets in the Grid. This chapter, in particular evaluates two of the most commonly used data technologies in the scientific domain and recommends the one most suitable to the requirements of the MC-PSC.

This chapter was published as a peer reviewed conference paper in *Proceedings of 22nd IEEE International Symposium on Computer-Based Medical Systems (CBMS-09,* ISBN:978-1-4244-4879-1, pp.1-8, 2009. [doi:10.1109/CBMS.2009.5255328]

8.1 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 [248]. 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 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. This chapter reports on our findings on the storage, management and analysis of data resulting from the newly developed distributed algorithm for MC-PSC (see chapter 5) for 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 chapter is as follows. A brief survey of the related work is presented in section 8.2. 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 8.3. Section 8.4, 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 8.5, while section 8.6 provides the results and discussions.

8.2 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 [249–252]. 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 chapter 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.

8.3 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. Mathematically,

$$S_{a,b} = \begin{cases} 0, & \text{No similarity at all.} \\ >0, & \text{No upper bound.} \end{cases} \tag{8.1}$$

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 (equation 8.1). On the other hand, there are algorithms that do not produce a scoring function but express similarity in terms of distances. In this case the resulting matrix for an all-against-all

comparison is called a *dissimilarity matrix* or *distance matrix* (DM). In terms of distance matrix, a value of zero means that two proteins are identical, whereas higher values indicate a higher degree of dissimilarity. Mathematically,

$$D_{a,b} = \begin{cases} 0, & \text{Identical proteins.} \\ >0, & \text{No upper bound.} \end{cases} \tag{8.2}$$

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. Though, similarity comparison algorithms usually produce numerical values to describe the degree of similarity between two protein structures, but they also return non-numerical characters in some cases as defined bellow:

- **N** Indicates that no significant similarity was found between the given pair of proteins
- **E** Indicates that an error occurred during the comparison
- # Indicates that the input data was missing or not appropriate

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 [46]) 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.

8.3.1 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). The pseudocode for this procedure is shown in Algorithm 2.

Algorithm 2 Pseudo-code for the estimation of missing/invalid self-similarity (SS) values. Since the self similarity value would always reflect high similarity, therefore, for in the case of the similarity matrix (SM) the maximum value either from the cross-section of the matrix which contains the similarity value for the current proteins i and j compared with any other protein x ($S_{ix}|S_{xj}$) is taken or (if $S_{ix}|S_{xj}$ not available then) the maximum value from the entire matrix is taken. Similarly, if the value being estimated belongs to a distance matrix (DM), then the minimum value is taken with the same approach.

```
1:  for all method k such that 1 ≤ k ≤ m do
2:    for all protein i in row (x) do
3:      for all protein j in column (x) do
4:        if matrix[i][j][k] = MISSING/INVALID && i = j then
5:          if matrix_type = SM then
6:            S_ii = max(S_ix|S_xj)|max(matrix))
7:          else if matrix_type = DM then
8:            S_ii = min(S_ix|S_xj)|min(matrix))
9:          end if
10:       end if
11:     end for
12:   end for
13: end for
```

Furthermore, similarity values can depend on the size of the proteins involved. Consider for instance the comparison of a protein with itself counting the number of alignments as the given similarity measure. When we compare identical proteins, the number of aligned residues will be the number of residues in this protein, but the number will be lower for a smaller protein than for a larger protein. We therefore have to take the size of the proteins into account and normalize a similarity value S_{ij} comparing proteins P_i and P_j by dividing it by the highest self-similarity value of both proteins [185]:

$$S_{ij,norm} = \frac{S_{ij}}{max\{S_{ii}, S_{jj}\}} \tag{8.3}$$

When applying Equation 8.3 to self-similarity values S_{ii}, one obtains normalized values $S_{ii,norm} = 1$ as $max\{S_{ii}, S_{ii}\} = S_{ii}$. Although this satisfies the first requirement for similarity matrices at the same time, the range of values will not necessarily start with zero. We require this for all matrices in order to be compared and combined in order to produce a consensus similarity, thus, another Normalisation step has to be performed for all similarity/dissimilarity matrices using the simple approach. As a result, all self-similarity values remain normalized, whereas all values of the entire matrix lie between $[0$ and $1]$.

8.3.2 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_j with P_i. 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. This is the highest value in the cross-section for dissimilarity matrices, and the lowest one for similarity matrices. Additionally, there is one special case when the comparison methods claims not to have found significant similarities between a given pair of proteins. In this case, we know that the similarity value will be very small and therefore substitute it with zero as the worst value for similarity matrices. The pseudocode for this procedure is shown in Algorithm 3.

Algorithm 3 Pseudo-code for the estimation of missing/invalid non-self-similarity (NSS) values. For similarity matrix (SM) the minimum value (i.e., the worst value) either from the cross-section or from the entire matrix is taken. Similarly, for the distance matrix (DM), the maximum value (the worst value) is taken in this case.

```
1: for all method k such that 1 ≤ k ≤ m do
2:   for all protein i in row (x) do
3:     for all protein j in column (x) do
4:       if matrix[i][j][k] = MISSING/INVALID && i != j then
5:         if matrix_type = SM then
6:           S_ij = min(S_ix|S_xj)|min(matrix))
7:         else if matrix_type = DM then
8:           S_ij = max(S_ix|S_xj)|max(matrix))
9:         end if
10:      end if
11:    end for
12:  end for
13: end for
```

8.4 Overview of the Core Data Storage, Management and Analysis Technologies

Scientific disciplines use much simple, convenient and self-describing data models such as *Hierarchical Data Format* (HDF, HDF4 or HDF5) [253], *Flexible Image Transport System* FITS and

NetCDF [254]. 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 (Figure 8.1).

Though there exist many scientific data models but HDF5 is being commonly used across a wide variety of scientific domains, projects and applications (`http://www.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* [253]. 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 (Figure 8.2. Working with all of these different formats becomes easy while using high-level interfaces such as *HDF5TB* interface, which could be used to work with tables (listing 8.1 shows a code snippet for

HDF5 table programming model).

LISTING 8.1: Code snippet for HDF5TB high level interface

```
/* Define HDF5 table dimensions  */
  #define   NFIELDS (hsize_t) 18
  #define   NRECORDS (hsize_t) 4000000
  #define   TABLE_NAME "COMPARISON_RESULTS"

/* Define field information */
  hid_t       field_type[NFIELDS];
  hid_t       string_type;
  hid_t       file_id;
  hsize_t     chunk_size = 100;
  int         *fill_data = NULL;
  int         compress  = 0;
  herr_t      status;
  int w;

 /* Initialize field_type */
  string_type = H5Tcopy( H5T_C_S1 );
  string_type = H5Tcopy( H5T_C_S1 );
  H5Tset_size(string_type, 32);
  field_type[0] = H5T_NATIVE_INT;
...

/* Create a new file using default properties. */
   file_id = H5Fcreate( "pdb_select30_2000.h5", H5F_ACC_TRUNC, H5P_DEFAULT, H5P_DEFAULT);

/* Create HDF5 table and write the data */
   status=H5TBmake_table( "Table␣Title", file_id, TABLE_NAME ,NFIELDS,NRECORDS,
                        dst_size,field_names, dst_offset, field_type,
                        chunk_size, fill_data, compress, comparison_results );

/*  Close the file   */
  H5Fclose( file_id );
```

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.

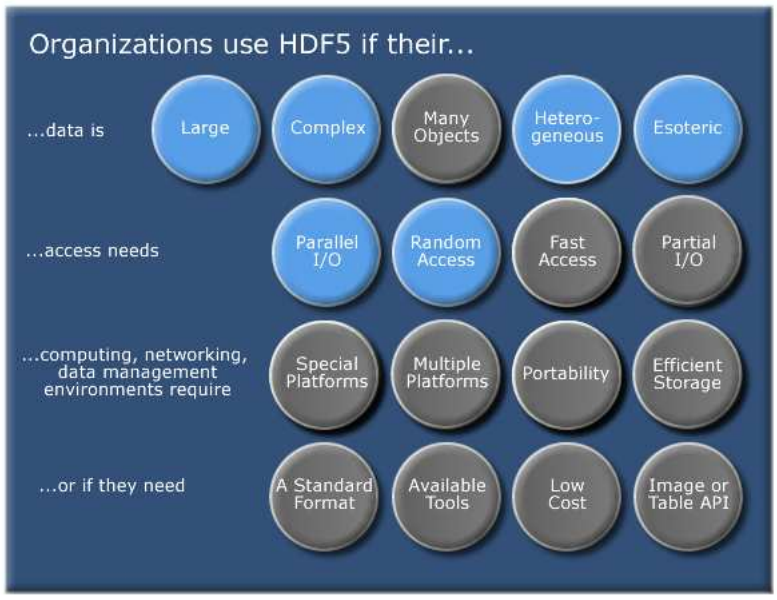


FIGURE 8.1: This figure illustrates the important reasons beyond the selection and use of the HDF i.e., to deal with the complexity, heterogeneity and size of the data in an efficient, reliable and low-cost manner. [extracted from [253]]

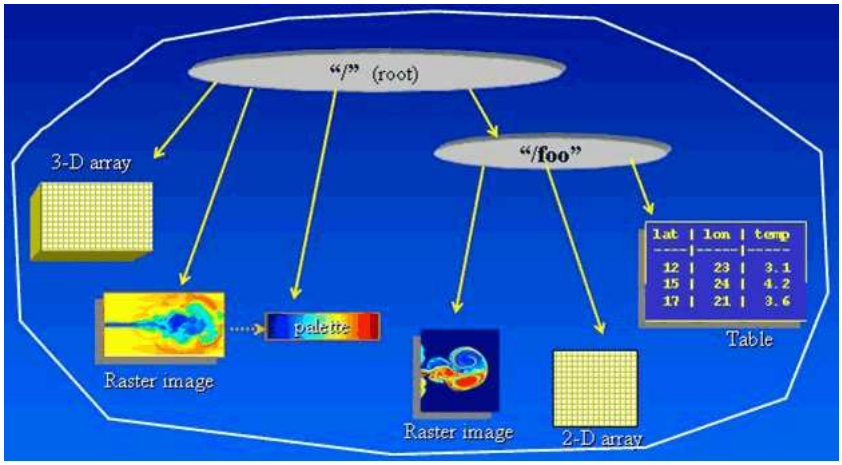


FIGURE 8.2: HDF5 file format (.hdf) could store different types of data (randing from 3d array to raster images and tables) in a single file [extracted from [253]]

8.5 Design and Implementation of the Proposed Architecture

Figure 8.3 shows the architectural design for the storage, management and analysis of (multi) similarity for large scale protein structure datasets using the infrastructure resources provided by NGS. The overall computational load of pair-wise similarity computations is uniformly distributed through a grid-enabled *Message Passing Interface* (MPI) [191] 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 [255]. A brief description of the NGS tools and services used for our experimentation is given below:

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. Digital certificates (or X.509 certificates) are being used as an alternate to user-name and password to authenticate the user in a wide range of eSciene projects. The NGS runs its own Certification Authority (CA;) which issues certificates to UK eScience community. The NGS's CA authenticates the users through its representative Registration Authorities (RAs) situated at various universities across the UK. These RAs in turn authenticate the users based on photo IDs and other personal details.

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 8.3.

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 shown in listing 8.2 and 8.3 respectively.

LISTING 8.2: 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";
      }
```

LISTING 8.3: 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 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 [256], 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.

8.6 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 8.1 and Table 8.2 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 8.3. All the results were stored directly on the NGS infrastructure. 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 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

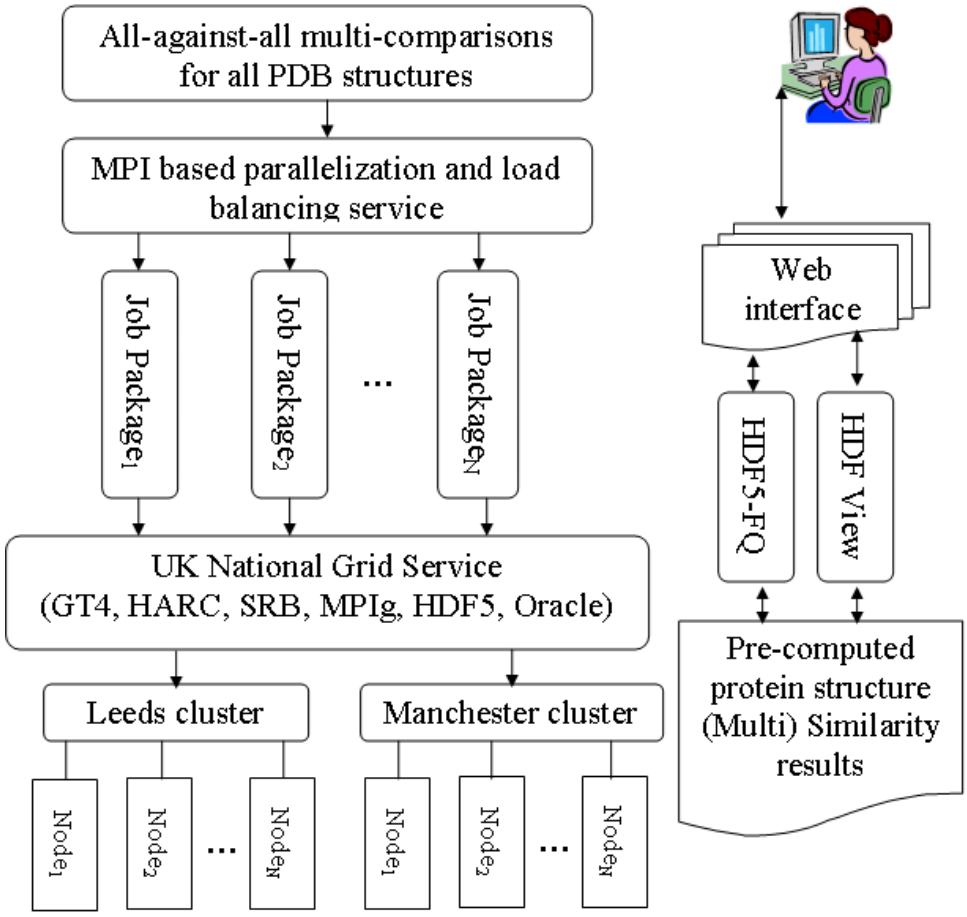


FIGURE 8.3: Architectural design of the storage, management and analysis for MC-PSC data in the grid environment. The multi-similarity results are stored by each node in a single HDF5 file and oracle database. The results from the HDF5 file could be accessed and analyzed through HDFView and HDF5-FQ as explained in the the text.

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

TABLE 8.1: 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 [224]. 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'*

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 8.2: Methods used to detect (Multi) Similarity among Protein Structure Datasets. *Note*: For full measure/metric names please refer to section 8.3

Method	Measures/Metrics
MaxCMO [46]	NA, NO
DaliLite [250]	NA, Z-score, RMSD
CE [47]	NA, Z-score, RMSD
TM-align [49]	NA, TM-score, RMSD
USM [48]	USM-distance
FAST [50]	NA, SN, RMSD

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 8.4).

TABLE 8.3: 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 8.1)* and the total number of fields is equal to the number of measures/metrics for all the methods (i.e 15,*Table 8.2*) plus 3 additional fields of which 2 store protein IDs and the remaining 1 serves as the record identifier.

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 8.4: Standard Deviations Calculated on 15 Query Times for HDF5 and Oracle (SQL)

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

8.7 Conclusions

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. It should be noted that the performance evaluation of the Oracle database is based on non-indexed queries. It is known from the Oracle documentation that the indexed keys introduce performance loss in terms of INSERTs, UPDATEs, and DELETEs. Therefore, in future, it would be interesting to compare the performance gain obtained with indexed keys with the one we have currently obtained (without indexing).

Based on this initial benchmarking, we plan to compute the (multi) similarities of all available PDB structures; thereby, creating an efficient scientific 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 scientific queries using a vast variety of criteria and options leading to far better and reliable understanding of the protein structural universe.

The next chapter provides an overview of the consensus-based Protein Structure Similarity Clustering and compares the results by using two different consensus clustering approaches.

CHAPTER 9

CONSENSUS-BASED PROTEIN STRUCTURE SIMILARITY CLUSTERING

The previous chapter discussed the storage, management and analysis of protein structure (multi) similarity data in terms of the comparison of two data technologies i.e., Oracle and HDF5. The analysis part discussed there was based on the query/visualization of the results obtained from individual methods. This chapter discusses in details the process of consensus and compares the results of single methods with that of the consensus obtained from two different approaches i.e., total evidence and total consensus.

9.1 Introduction

Protein Structure Similarity Clustering (PSSC) is the process of analyzing the results obtained from the comparison of protein structures [257–259]. Because MC-PSC involves multiple methods, the results from each method need to be combined/unified based on the process of consensus. As illustrated in Figure 9.1, there are different approaches for deriving the consensus [260]. Depending on whether we combine the data (i.e, character congruence) or the trees (i.e taxonomic congruence)

these approaches are termed as *Total evidence* and *Total consensus* respectively [261]. It had already been demonstrated in [27] that the TE based results obtained from MC-PSC are more robust and accurate as compared to the individual results from each single method. Based on the initial results of [27], this chapter presents an extended study on the case of '*Structural Comparison and Clustering of Protein Kinases*'. The extension involves the incorporation of more comparison methods in the process of consensus as well as driving the consensus with a new approach (i.e *'Total consensus'*). In what follows, a succinct description of the clustering algorithms, the approaches used for the consensus and the characteristics of the *Protein Kinase Dataset* is presented along with the results, discussions and conclusions.

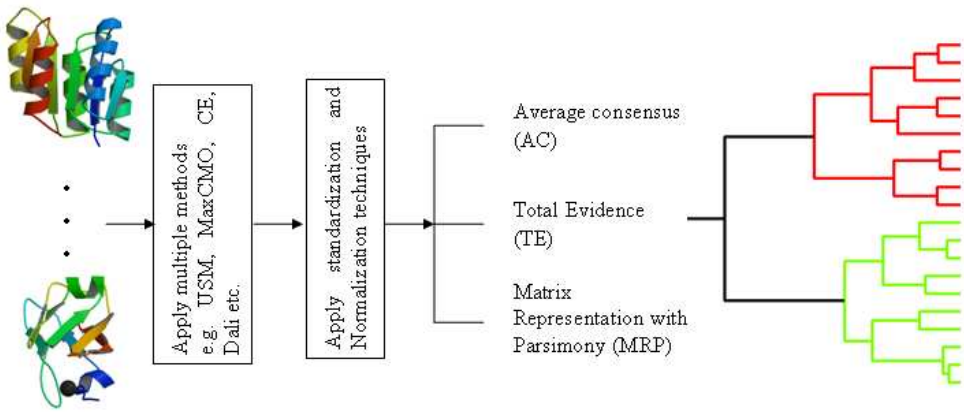


FIGURE 9.1: Simple MC-PSC process: takes protein 3-D structures as input; applies popularly used algorithms for the comparison of these structures; makes the results homogeneous and finally develops consensus based clustering. Since, there are also different methods for consensus [260], the MC-PSC process also needs to investigate the most reliable of all of these.

9.2 Clustering Algorithms

Protein structure similarity results (usually a square similarity matrix or a square distance matrix) are applied to a clustering algorithm which groups the proteins into a set of disjoint classes so

that proteins within a class have high *similarity*, while proteins in separate classes are more *dissimilar* [262]. There are different algorithms available in different software packages. The results presented in this chapter are based on the localized version of the *Clustering Calculator* [263]. The *Clustering Calculator* provides a variety of hierarchical clustering algorithms, including e.g. the *Unweighted Pair Group Method with Arithmetic mean* (UPGMA) [180] and the *Ward's Minimum Variance* (WMV) method [181]. The results of the clustering are produced as a tree in the *PHYLIP*-format [264], which could be visualized in a variety of ways using tree visualization software such as *HyperTree* [265] and *Dendroscope* [266].

9.3 Consensus Approaches

Two of the most competing paradigms in the field of phylogenetic inference (systematics) are: *'total evidence'* (character congruence) and *'total consensus'* (taxonomic congruence) [267]. A brief description of these paradigms is given in the following sections.

9.3.1 Total evidence

The term total-evidence was originally introduced by Rudolf Carnap in the context of *inductive logic* [268]. Based on this concept, in 1989, Arnold G. Kluge introduced this term in the field of systematics [269]. As per Kluge's definition, the idea behind the *'total evidence'* is to consider all evidence (e.g. distances) simultaneously and combine them together in order to achieve best phylogenetic estimate. Figure 9.2 illustrates the steps involved in the process of total evidence. At top there are several different distance matrices (produced by different methods in the case of MC-PSC), which are combined (by taking average, mean or median) into a single matrix called 'total evidence matrix'. This matrix is then used to produce the tree for the final analysis.

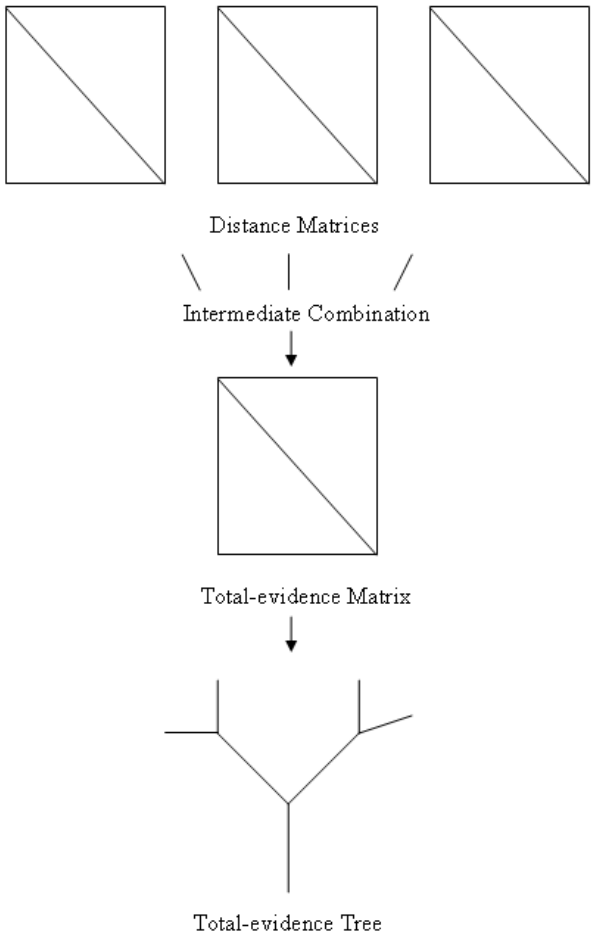


FIGURE 9.2: Total-evidence: the distance matrices (produced by different methods in the case of MC-PSC are) combined (by taking average, mean or median) into a single matrix (total evidence matrix). The resultant matrix produces the total evidence based tree to depict the phylogenetic relationship among the elements of the dataset.

9.3.2 Total consensus

The idea of *'total consensus'* is based on Karl Popper's *Logik der Forschung* (The Logic of Scientific Discovery) [270] which says that the mutual confirmation of the independent lines of evidence gives strongest support for conclusions. Hence, this approach focuses on the separate analysis of the data and then combining the analyzed data (trees) to form a consensus tree (super tree) [267]. Figure

9.3 illustrates the steps involved in the process of total consensus. At top there are several different distance matrices (produced by different methods in the case of MC-PSC), each of these matrices is separately analyzed (by producing a tree) and then these individual trees are combined together to form the consensus tree for the final analysis. There are several approaches and software packages for the combination of individual trees to form the consensus [271]. This study uses the *Total consensus* (TC) method of the *Clann* (the Irish word for "family") software [271].

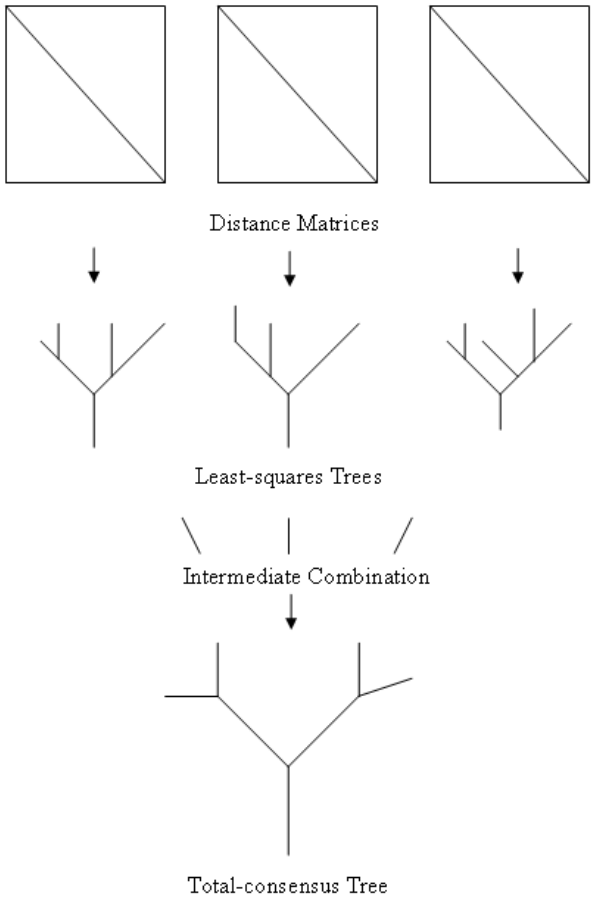


FIGURE 9.3: Total-consensus: the distance matrices (produced by different methods in the case of MC-PSC) are applied to clustering algorithm to generate trees for separate analysis. The individual trees are then combined to produce a total consensus based super tree which depicts the phylogenetic relationship among the elements of the dataset.

9.4 Protein Kinase Dataset

Protein Kinase Dataset consists of 45 structures from 9 different groups (super-families), namely 1) cAMP Dependent Kinases, 2) Protein Kinases C, 3) Phosphorylase Kinases, 4) Calmodulin Kinases, 5) Casein Kinases, 6) Cyclin Dependent Kinases, 7) Tyrosine Kinases, 8) Mitogen Activated Kinases, and 9) Twitchen Kinases [272–275] as shown in Table 9.1.

Table 9.1 - Protein Kinase Dataset: Protein Kinase Dataset. Detailed SCOP classification for each protein domain in the Protein Kinase (PK) dataset, grouped according to Hanks' and Hunters' (HH) original classification scheme. IG = Immunoglobulin, PhK = Tyrosine Phosphorylase Kinase, S/TK = Serine/Threonin Kinase, TK = Tyrosine Kinase, c.s. = catalytic subunit, c.-r. = cysteine-rich., dep. = dependent. Reproduced from [27]

HH	Sub	Protein	SCOP Classification Level					
Cluster	Cluster	Domain	Class	Fold	Superfamily	Family	Domain	Species
1	A	d1apme_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	mouse
		d1atpe_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	mouse
		d1bkxa_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	mouse
		d1fmoe_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	mouse
		d2cpke_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	mouse
	B	d1cdka_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	pig
		d1cmke_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	pig
		d1ctpe_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	pig
	C	d1stce_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	cow
		d1ydre_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	cow
		d1ydse_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	cow
		d1ydte_	α+β	PK-like	PK-like	PK c.s. (S/TK)	cAMP-dep. PK, c.s.	cow
2	-	d1ptq__	small	PK c.-r. domain	PK c.-r. domain	PK c.-r. domain	PK C-delta (PKCdelta)	mouse
		d1ptr__	small	PK c.-r. domain	PK c.-r. domain	PK c.-r. domain	PK C-delta (PKCdelta)	mouse
3	-	d1phk__	α+β	PK-like	PK-like	PK c.s. (S/TK)	γ-subunit glycogen Phk	rabbit
4	A	d1a06__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Calmodulin-dep. PK	rat
	B	d1cdma_	α	EF Hand-like	EF-hand	Calmodulin-like	Calmodulin	cow
		d1cm1a_	α	EF Hand-like	EF-hand	Calmodulin-like	Calmodulin	cow
		d1cm4a_	α	EF Hand-like	EF-hand	Calmodulin-like	Calmodulin	cow
5	A	d1lr4a_	α+β	PK-like	PK-like	PK c.s. (S/TK)	Casein kinase-2, CK2	maize
	B	d1csn__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Casein kinase-1, CK1	fission yeast
		d2csn__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Casein kinase-1, CK1	fission yeast
6	-	d1aq1__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Cyclin-dep. PK, CDK2	human
		d1fina_	α+β	PK-like	PK-like	PK c.s. (S/TK)	Cyclin-dep. PK, CDK2	human
		d1hck__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Cyclin-dep. PK, CDK2	human
		d1hcl__	α+β	PK-like	PK-like	PK c.s. (S/TK)	Cyclin-dep. PK, CDK2	human
		d1jsua_	α+β	PK-like	PK-like	PK c.s. (S/TK)	Cyclin-dep. PK, CDK2	human
7	A	d1ad5a1	β	SH3-like barrel	SH3-domain	SH3-domain	Hemapoetic cell kinase Hck	human
		d1ad5a2	α+β	SH2-like	SH2 domain	SH2 domain	Hemopoetic cell kinase Hck	human
		d1ad5a3	α+β	PK-like	PK-like	PK c.s. (TK)	Hemopoetic cell kinase Hck	human
		d1fmk_1	β	SH3-like barrel	SH3-domain	SH3-domain	c-src protein TK	human
		d1fmk_2	α+β	SH2-like	SH2 domain	SH2 domain	c-src TK	human
		d1fmk_3	α+β	PK-like	PK-like	PK c.s. (TK)	c-src TK	human
		d2hcka1	β	SH3-like barrel	SH3-domain	SH3-domain	Hemapoetic cell kinase Hck	human
		d2hcka2	α+β	SH2-like	SH2 domain	SH2 domain	Hemopoetic cell kinase Hck	human
		d2hcka3	α+β	PK-like	PK-like	PK c.s. (TK)	Haemopoetic cell kinase Hck	human
		d2ptk_1	b	SH3-like barrel	SH3-domain	SH3-domain	c-src protein TK	chicken
		d2ptk_2	α+β	SH2-like	SH2 domain	SH2 domain	c-src TK	chicken
		d2ptk_3	α+β	PK-like	PK-like	PK c.s. (TK)	c-src TK	chicken
	B	d1aotf_	α+β	SH2-like	SH2 domain	SH2 domain	TK Fyn	human
		d1blj__	α+β	SH2-like	SH2 domain	SH2 domain	P55 Blk protein TK	mouse
		d1csya_	α+β	SH2-like	SH2 domain	SH2 domain	Syk TK	human
		d1cwea_	α+β	SH2-like	SH2 domain	SH2 domain	p56-lck TK	human
	C	d1fgka_	α+β	PK-like	PK-like	PK c.s. (TK)	Fibroblast growth factor receptor 1	human
		d1ir3a_	α+β	PK-like	PK-like	PK c.s. (TK)	Insulin receptor	human
		d1irk__	α+β	PK-like	PK-like	PK c.s. (TK)	Insulin receptor	human
		d3lck__	α+β	PK-like	PK-like	PK c.s. (TK)	Lymphocyte kinase (lck)	human
8	A	d1erk__	α+β	PK-like	PK-like	PK c.s. (S/TK)	MAP kinase Erk2	rat
	B	d1ian__	α+β	PK-like	PK-like	PK c.s. (S/TK)	MAP kinase p38	human
		d1p38__	α+β	PK-like	PK-like	PK c.s. (S/TK)	MAP kinase p38	mouse
		d1wfc__	α+β	PK-like	PK-like	PK c.s. (S/TK)	MAP kinase p38	human
9	A	d1koa_1	β	IG-like β-sandwich	IG	I set domains	Twitchin	nematode
		d1koa_2	B α+β	PK-like	PK-like	PK c.s. (S/TK)	Twitchin, kinase domain	Caenorhabditis elegans
		d1koba_	α+β	PK-like	PK-like	PK c.s. (S/TK)	Twitchin, kinase domain	California sea hare

9.5 Results and Discussions

Figure 9.4(a) shows the classification of the protein kinase dataset with USM method. This method distinguishes the kinases at class/fold level and groups them into two main clusters i.e *alpha+beta*/PK-like proteins and others (i.e., all small proteins (2), all alpha proteins with EF Hand-like fold (4B), and such from the alpha+beta class but with SH2-like fold (7B)). All of the kinases are correctly identified and clustered (36 in cluster 1 and 9 in cluster 2 (*green box*)). The *blue box* in cluster 1 indicates that the classification with USM also detects the similarities up to the species level i.e., similar kinases from HH cluster 1 (pigs, cows and mice) are clustered appropriately. The exceptions (errors/wrong clustering of kinases at species level) are indicated in blue (inside the blue box), green (inside the green box) and red (for the rest of the kinases). Figures 9.4(b), to 9.11 show the similar clustering (based on single method/measure) for the rest of the methods. Whether a particular method/measure is able to group the kinases properly or not is discussed in the caption of each figure. It seems that the poorest performing measure amongst all is the *RMSD* of each method (Figures 9.6(a), 9.7(b), 9.9(a) and 9.10(a)). Hence, studies such as [257–259], which are based on the clustering of RMSDs might not have produced proper results, and, therefore, the use of consensus mechanism to combine the results from different methods/measures becomes indispensable.

Figure 9.12 shows the classification of the protein kinase dataset with *Total evidence* (a) and *Total consensus* (b) when only Z-scores are combined from each method. As discussed in the caption of the figure, the total evidence based consensus produces better classification as compared to the total consensus. The total consensus fails to separate the kinases in two main clusters as well

as the total number of wrongly clustered kinases at species level (i.e., Red, Green and Blue kinases) is much larger (11) as compared to the total evidence (7). This trend could also be observed in the case of figures 9.13 and 9.14, which are based on the consensus of *Number of Alignments* and *RMSDs* respectively.

Finally, figure 9.15 shows the results of the classification when the consensus is derived from the combination of all the method/measures. Here, again the total evidence approach provides much better classification as compared to the total consensus. The total consensus fails to separate the kinases in two main clusters as well as the total number of wrongly clustered kinases at species level (i.e., Red, Green and Blue kinases) is much larger (10) as compared to the total evidence (3). Table 9.2, summarizes the performance of all the methods individually, with total consensus and total evidence.

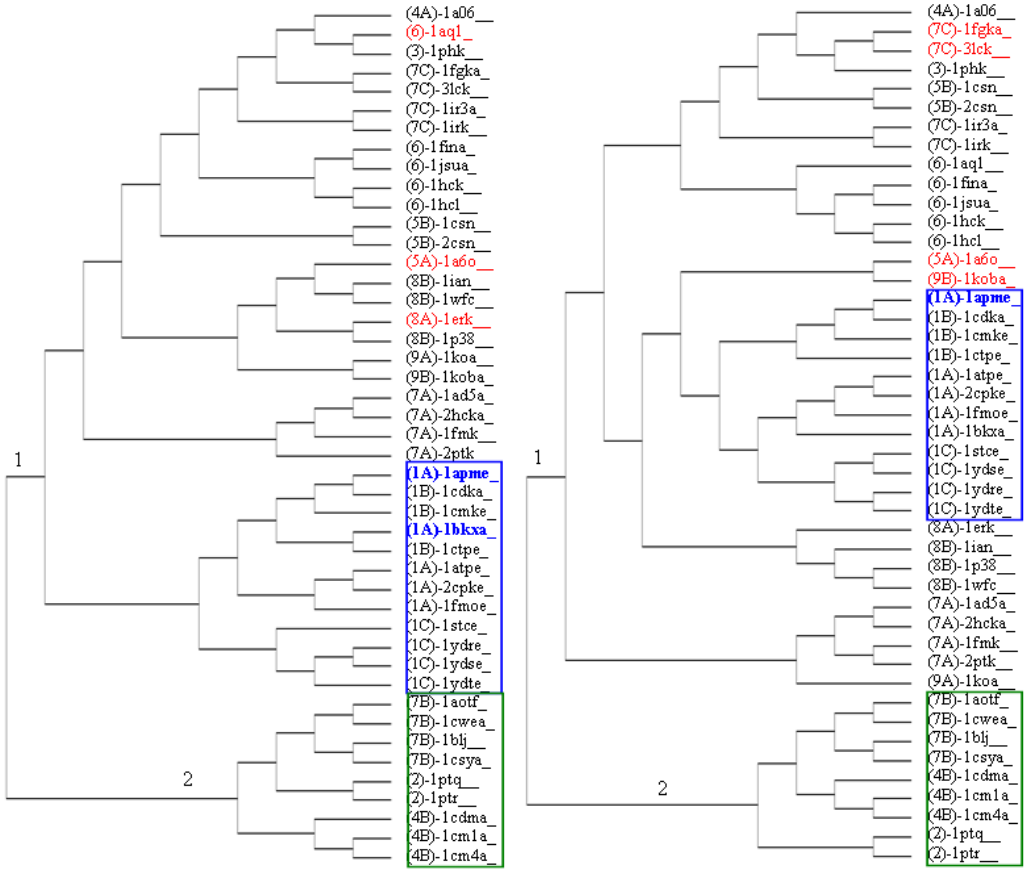
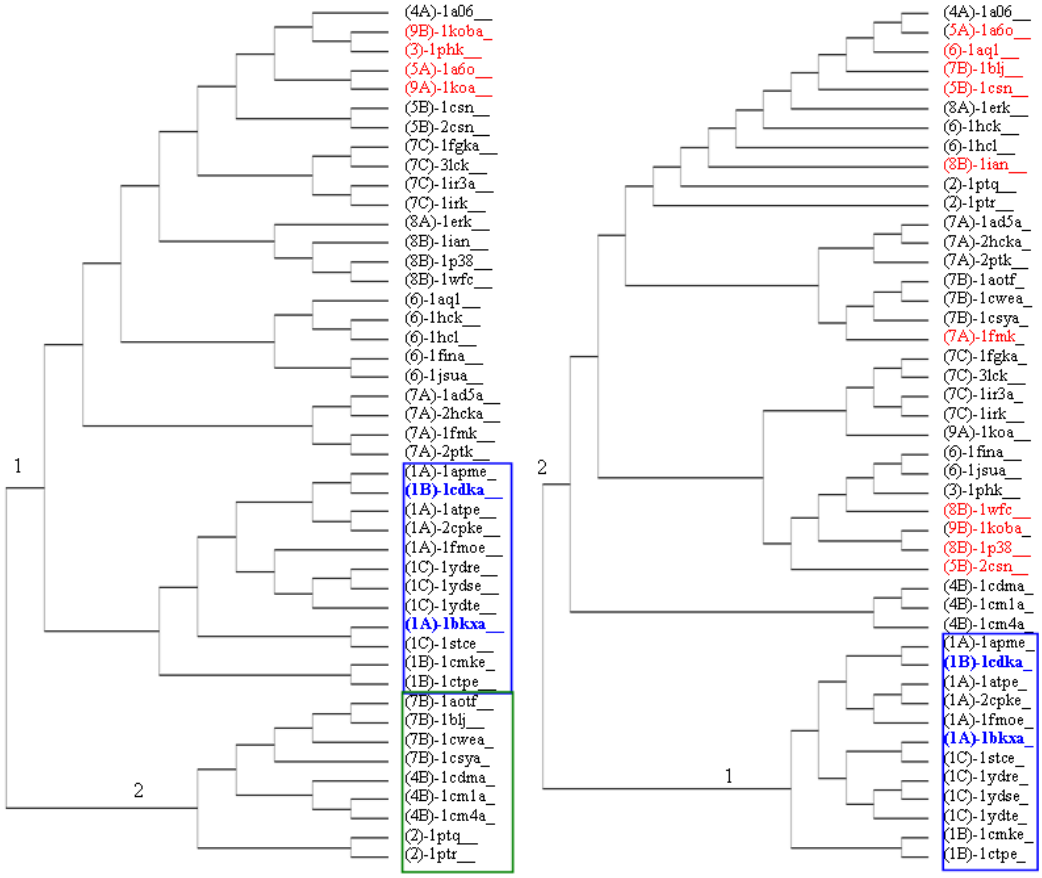
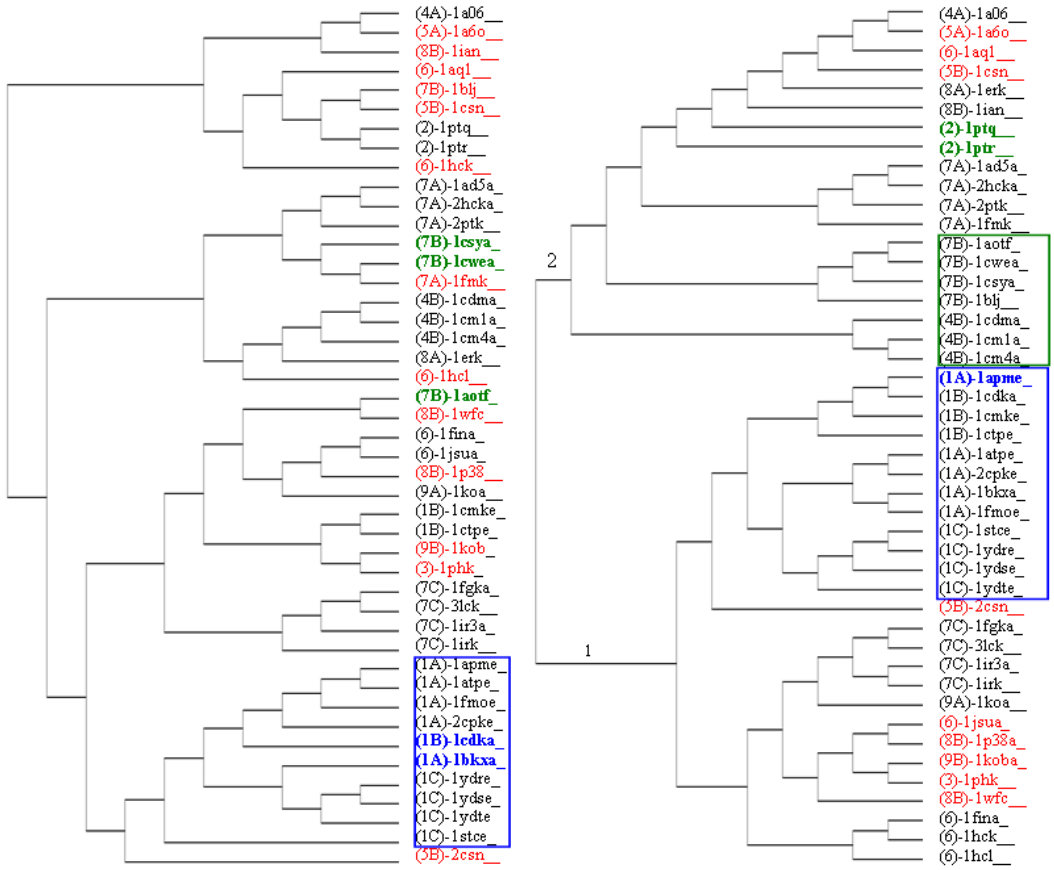


FIGURE 9.4: Single method clustering **(a)** USM **(b)** MaxCMO/Align. *Note*: Red, Green and Blue kinases indicate errors at species level.



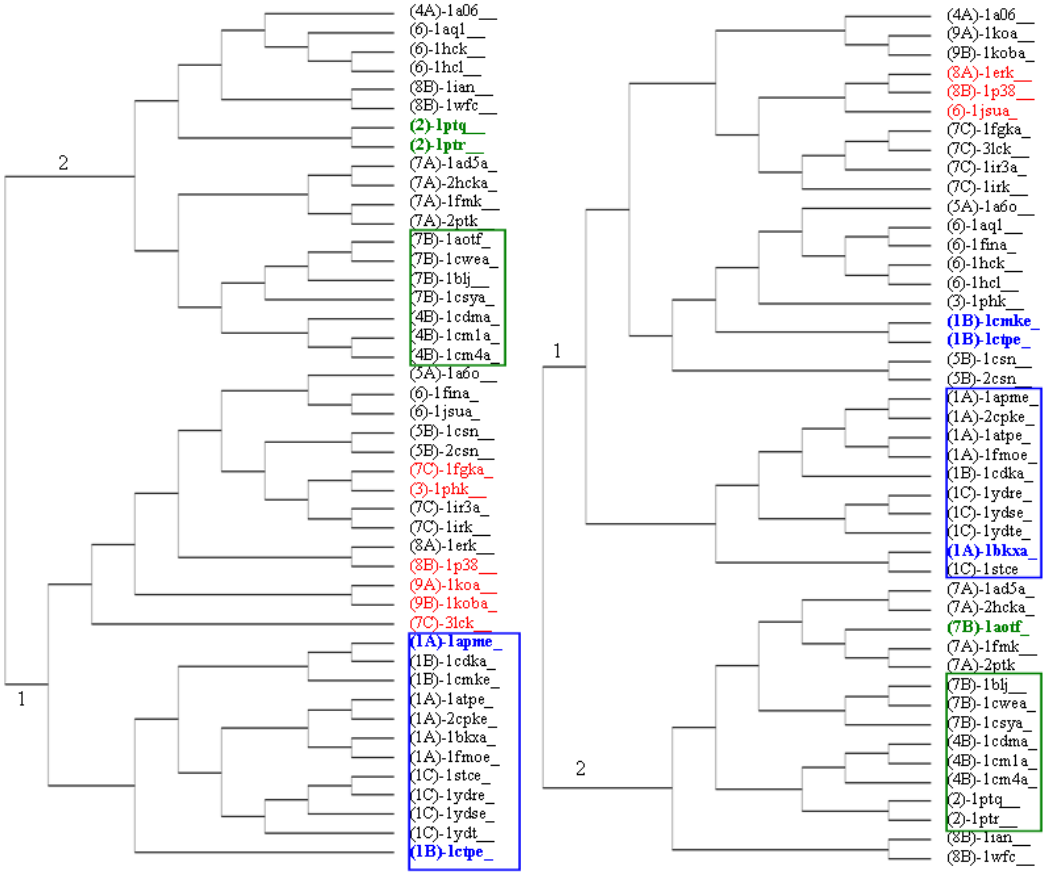
(a) (b)

FIGURE 9.5: Single method clustering **(a)** MaxCMO/OverLap **(b)** Dali/Z: Kinases are not grouped into two main clusters, i.e., kinases from 1st cluster are mixed with 2nd; kinases in the 2nd cluster do not group into the green box. *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.6: Single method clustering **(a)** Dali/RMSD: Kinases are not grouped into two main clusters; kinases in the 2nd cluster do not group into the green box. **(b)** Dali/Align: Kinases are not separated into two main clusters, i.e., kinases from 1st cluster are mixed with 2nd cluster; the green box lacks its two kinases (1ptq,1ptr). *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.7: Single method clustering **(a)** CE/Align: Kinases are not separated into two main clusters properly, i.e., kinases belonging to 1st cluster are mixed with the 2nd; also the 2nd cluster (green box) lacks two of its kinases (1ptq,1ptr). **(b)** CE/RMSD: Kinases are not grouped into two main clusters properly, i.e., some kinases from 1st cluster are mixed with 2nd cluster; on the hand the blue box lacks two kinases (1cmke,1ctpe) and the green box lacks one kinase (1aotf). *Note*: Red, Green and Blue kinases indicate errors at species level.

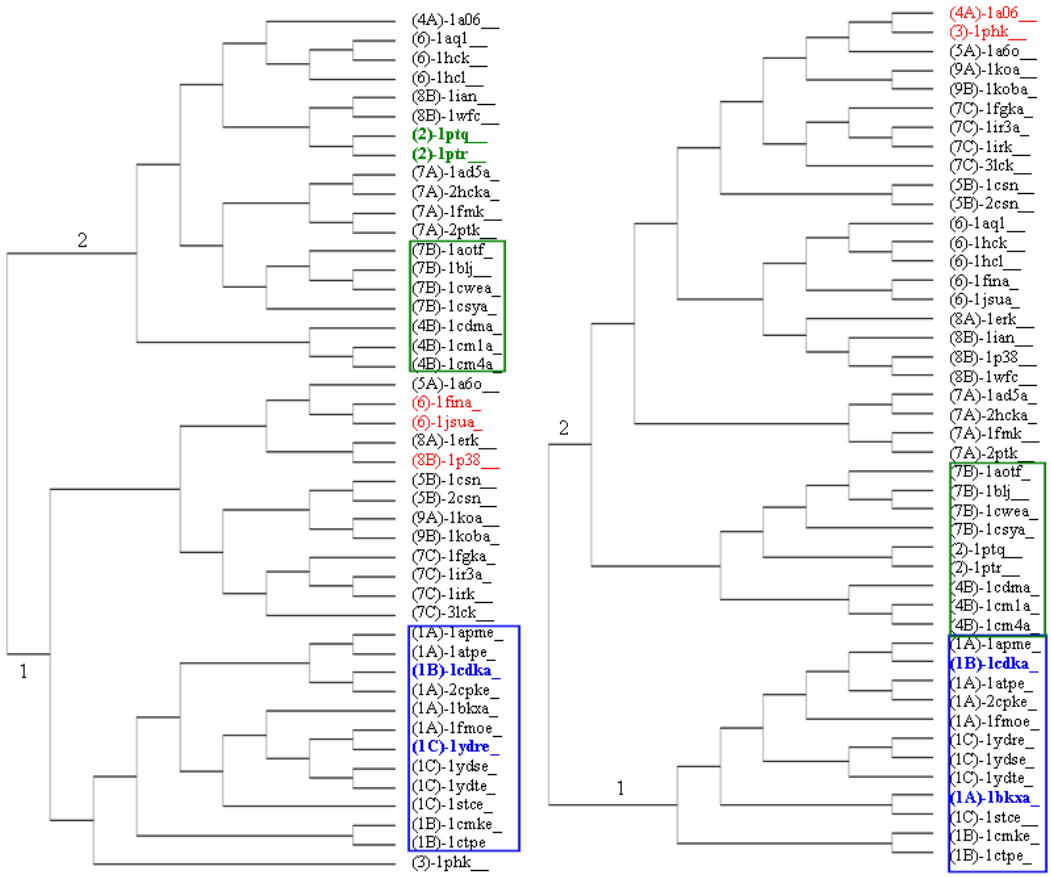
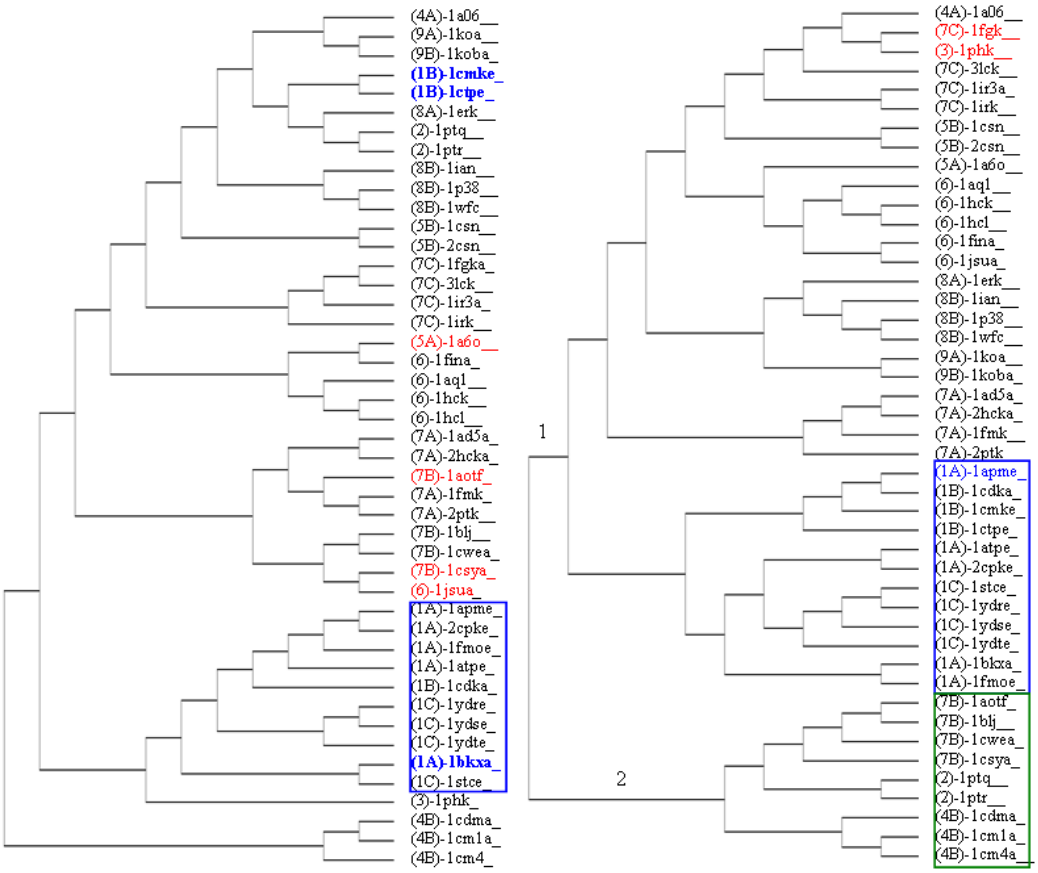
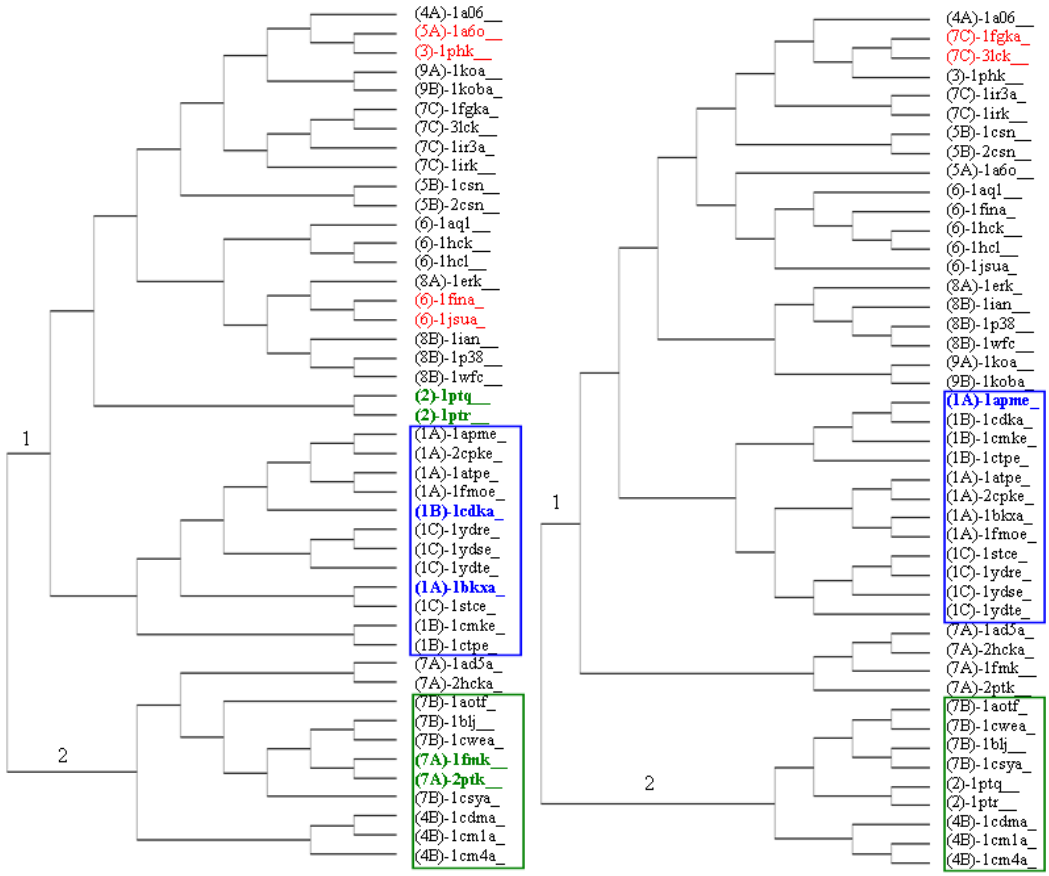


FIGURE 9.8: Single method clustering **(a)** CE/Z: kinases are not separated into two main clusters properly i.e kinases belonging to 1st cluster are mixed with the 2nd. Also the 2nd cluster (green box) lacks two of its kinases (1ptq,1ptr). **(b)** FAST/SN: kinases are not grouped into two main clusters properly, i.e., kinases belonging to 1st cluster are mixed with 2nd. *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.9: Single method clustering **(a)** FAST/RMSD: kinases are not grouped into two main clusters properly. The 2nd cluster (green box; missing here) gets mixed with kinases from 1st. **(b)** FAST/Align. *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.10: Single method clustering **(a)** TMAlign/RMSD: The 2nd cluster (green box) lacks its 2 kinases (1ptq, 1ptr) and contains two extra/wrong kinases (1fmk,2ptk). **(b)** TMAlign/Align. *Note*: Red, Green and Blue kinases indicate errors at species level.

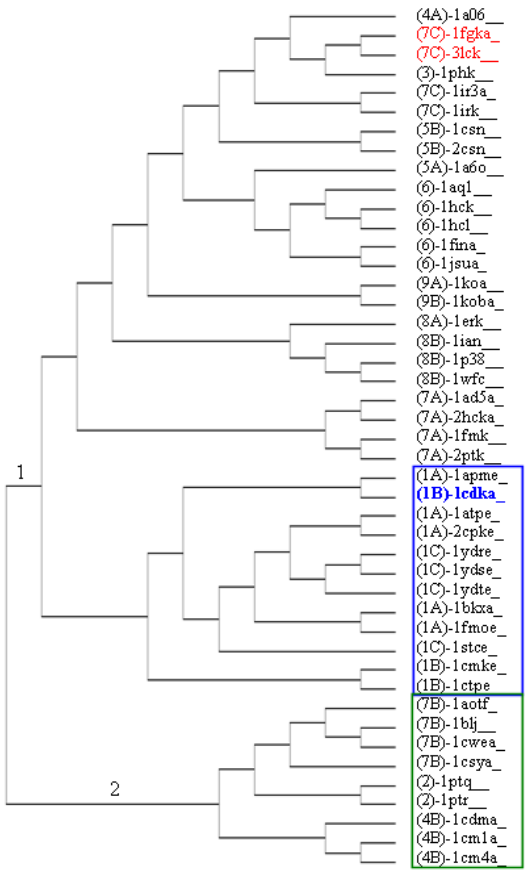
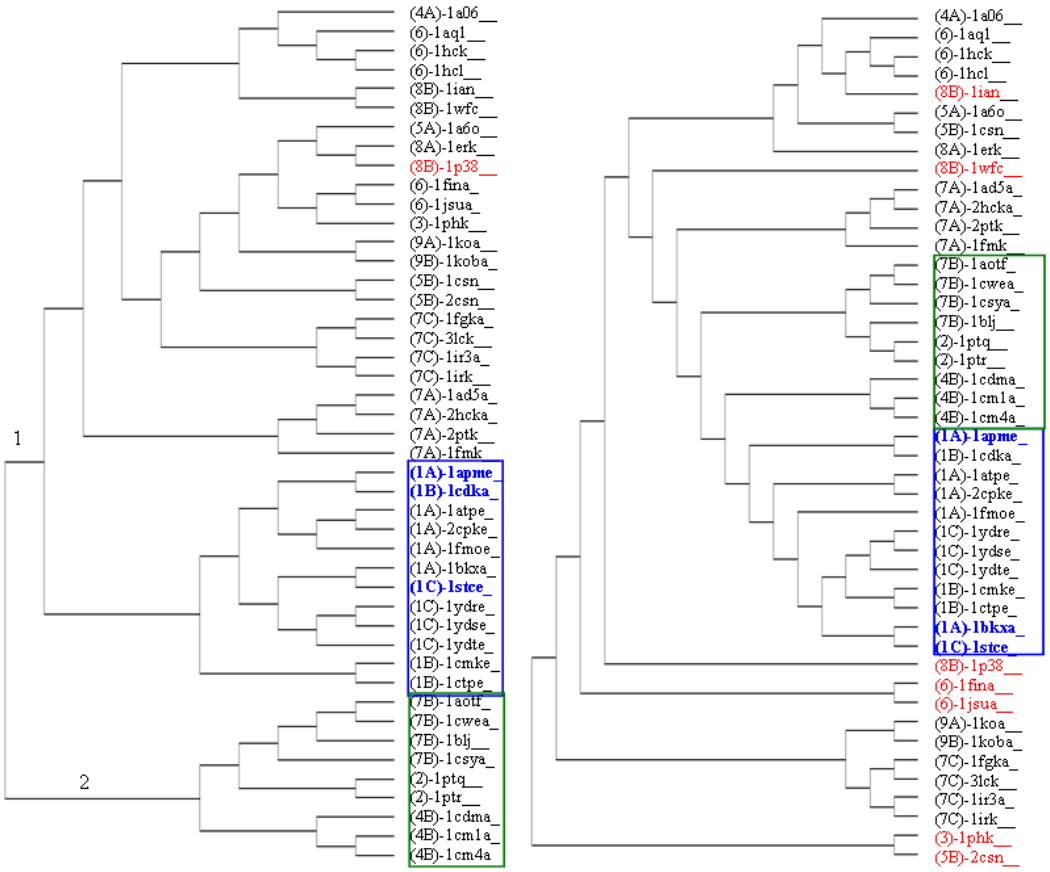
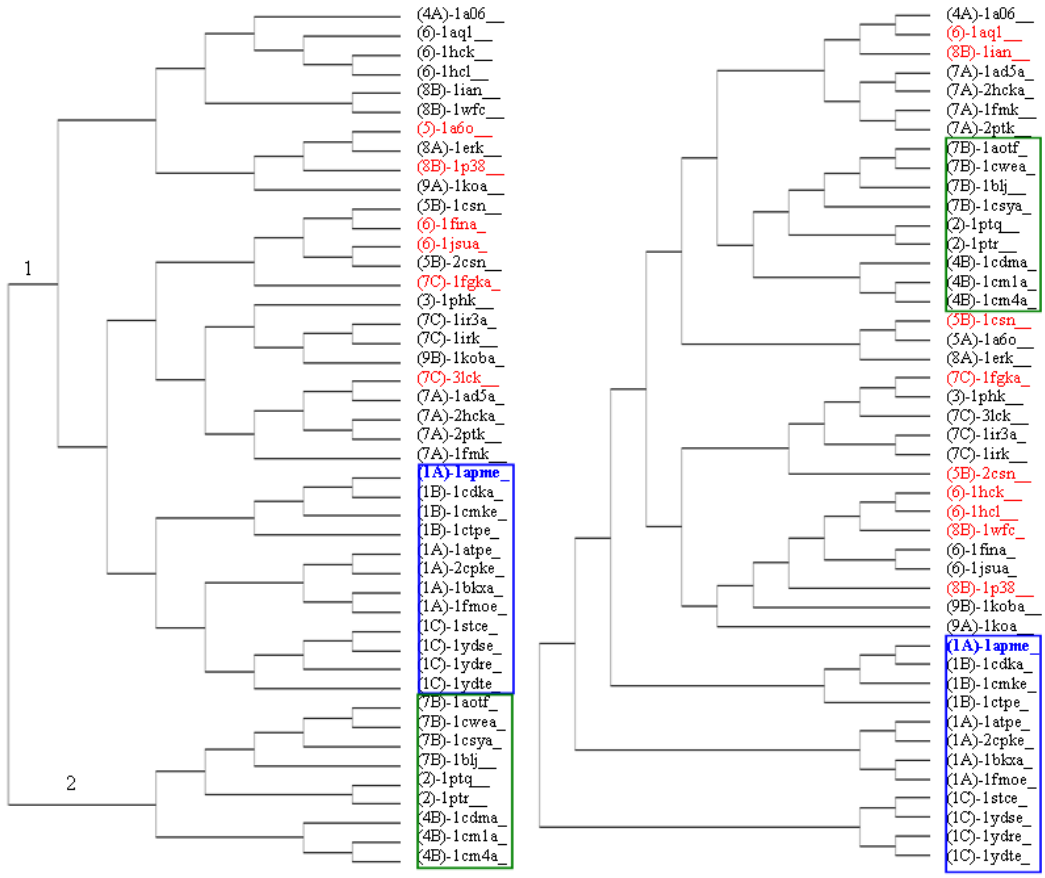


FIGURE 9.11: Single method clustering: TMAlign/TM-Score. *Note*: Red, Green and Blue kinases indicate errors at species level.



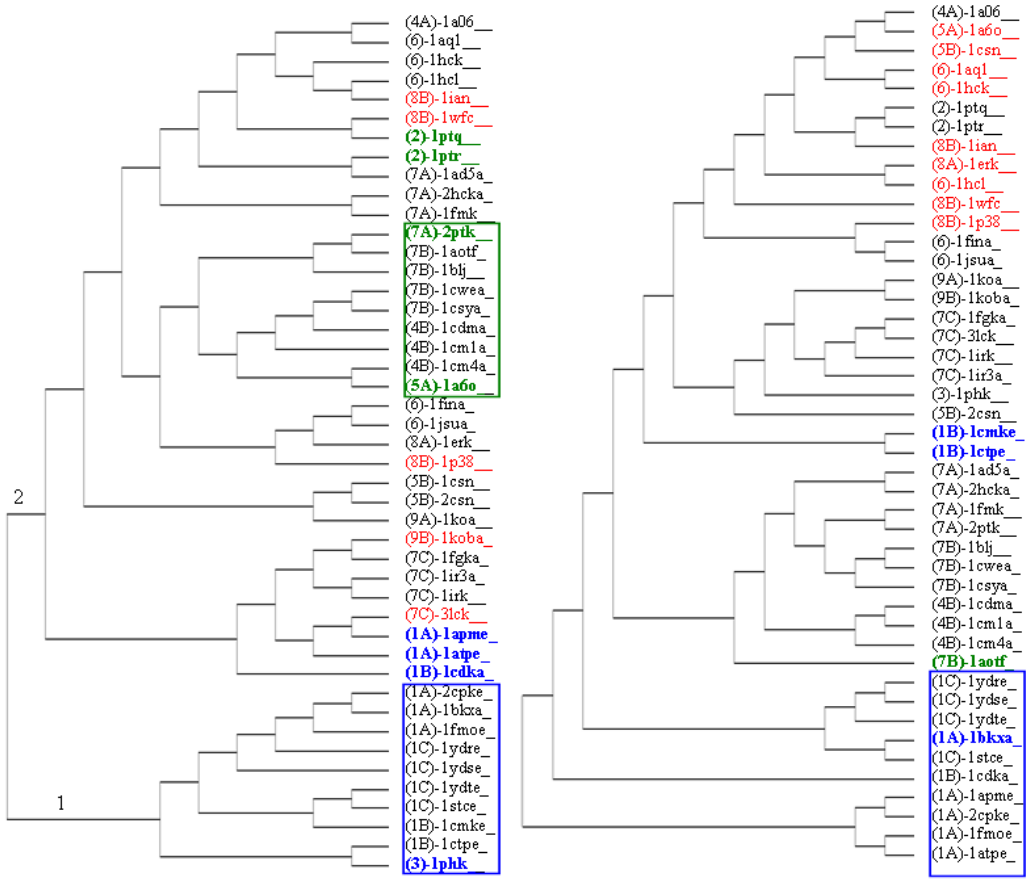
(a) (b)

FIGURE 9.12: Total evidence (TE) vs Total consensus (TC) **(a)** TE/All/Z: kinases are not grouped into two main clusters, i.e., kinases from the 1st cluster are mixed with the 2nd. The blue box lacks its three kinases (1apme,atpe,1cdka) and contains one extra/wrong kinase (1phk). The green box lacks its two kinases (1ptq, 1ptr) and contains two extra/wrong kinases (1a6o,2ptk). **(b)** TC/All/Z: kinases are not separated into two main clusters, rather they follow complicated hierarchy. *Note*: Red, Green and Blue kinases indicate errors at species level.



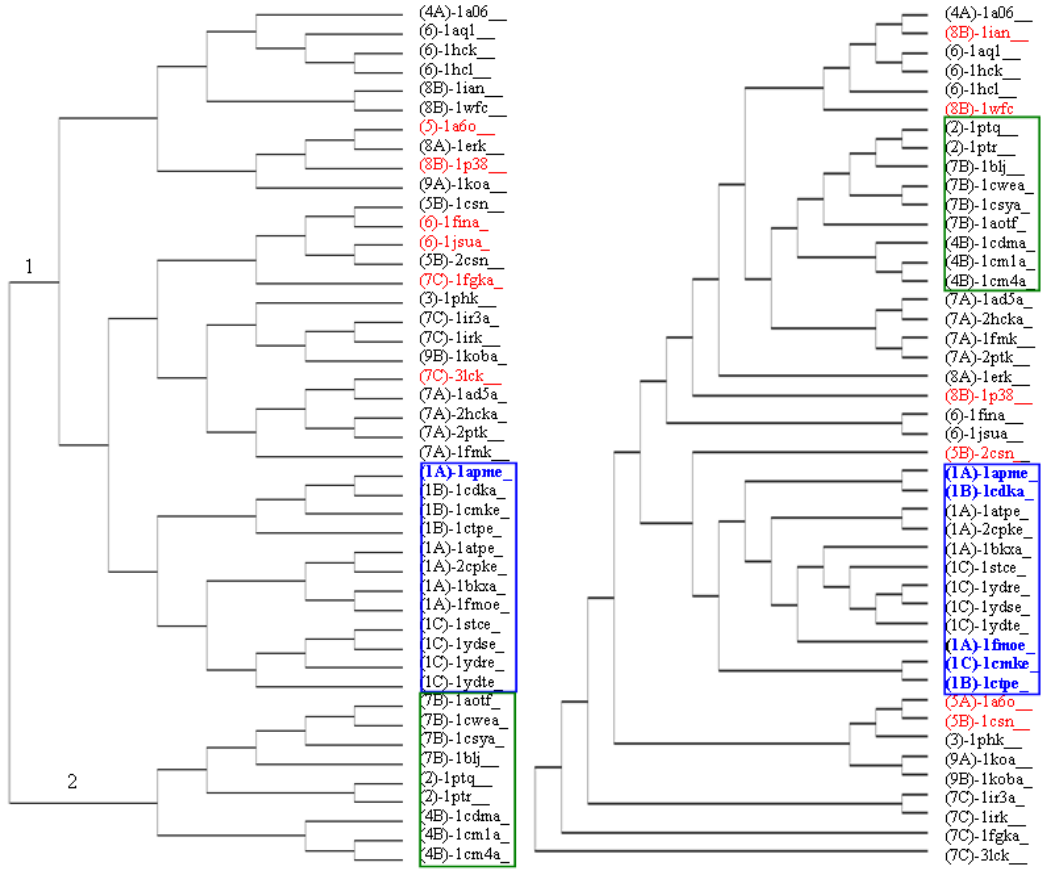
(a) (b)

FIGURE 9.13: Total evidence (TE) vs Total consensus (TC) **(a)** TE/All/Align. **(b)** TC/All/Align: kinases are not grouped into two main clusters. kinases are not separated into two main clusters. *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.14: Total evidence (TE) vs Total consensus (TC) **(a)** TE/All/RMSD: kinases are not grouped into two main clusters, i.e., kinases from the 1st cluster are mixed with the 2nd. The blue box lacks its three kinases (1apme,atpe,1cdka) and contains one extra/wrong kinase (1phk). The green box lacks its two kinases (1ptq, 1ptr) and contains two extra/wrong kinases (1a6o,2ptk). **(b)** TC/All/RMSD: kinases are not grouped into two main clusters, rather they follow complicated hierarchy. The blue box lacks its two kinases (1cmke,1ctpe). The green box is missing (all kinases dispersed). *Note*: Red, Green and Blue kinases indicate errors at species level.



(a) (b)

FIGURE 9.15: Total evidence (TE) vs Total consensus (TC) **(a)** TE/All/All. **(b)** TC/All/All: kinases are not separated into two main clusters, rather they follow complicated hierarchy. *Note*: Red, Green and Blue kinases indicate errors at species level.

TABLE 9.2: Single Method Vs Total consensus (TC) Vs Total Evidence (TE))

Method/Measure	**kinases grouped in 2 main Clusters?**	**Number of wrongly clustered kinases:**			
		Red	**Green**	**Blue**	**Total**
USM	Yes	3	-	2	5
MaxCMO/Align	Yes	4	-	1	5
MaxCMO/OL	Yes	4	-	2	6
Dali/Align	No	9	2	1	12
Dali/RMSD	No	13	3	2	18
Dali/Z	No	10	-	2	12
CE/Align	No	6	2	2	10
CE/RMSD	No	3	1	3	7
CE/Z	No	3	2	2	7
TMAlign/Align	Yes	2	-	1	3
TMAlign/RMSD	No	4	4	2	10
TMAlign/Z	Yes	2	-	1	3
FAST/SN	No	2	-	2	4
FAST/Align	Yes	2	-	1	3
FAST/RMSD	No	4	-	3	7
TC/RMSD	No	9	1	3	13
TC/Align	No	9	1	-	10
TC/Z	No	7	-	3	10
TC/All	No	6	-	5	11
TE/RMSD	No	5	4	4	13
TE/Align	Yes	6	-	1	7
TE/Z	Yes	1	-	3	4
TE/All	Yes	6	-	1	7

9.6 Conclusions

Protein Structure Similarity Clustering has been performed for the *Protein Kinase Dataset* [272–275]. It has been observed that the results of the clustering for each single method differ significantly from each other. Some methods (e.g., CE/All, Dali/All, TMAlign/RMSD, FAST/SN/RMSD) even fail to classify the proteins into two main clusters, which is indeed a significant error in terms of accuracy of classification because they mix the proteins belonging to different class/fold. The total consensus approach (using all methods/measures i.e., TC/ALL) also exhibits same mistake, however; the total evidence approach (TE/ALL) rightly classifies the proteins into two main clusters and hence it performs more accurate and reliable classification as compared to the results of individual methods or the total consensus approach. Furthermore, the total number of less significant errors (miss-classification errors at specie level (Red, Green, Blue) in table 9.2) is worst for some single method based results (e.g., Dali/RMSD has 18 errors, TE/RMSD has 13 errors, Dali/Align/Z have 12 errors) as well as for the total consensus method (i.e., TC/ALL has 11 errors) but it is significantly less for the results of total evidence (i.e., TE/ALL has only 7 errors). This proves that in the case of the protein structure similarity clustering the total evidence based consensus approach performs much better than many single methods as well as the total consensus based approach.

The next chapter summarizes the overall work of this dissertation and provides some directions for the possible future work.

CHAPTER 10

CONCLUSIONS AND FUTURE WORK

This chapter summarizes the overall work of this dissertation and provides some directions for the possible future work.

10.1 Introduction

This thesis is based on studies of the use of grid-styled parallel/distributed approaches for one of the *Grand Challenge Applications* (GCAs) in the field of structural proteomics namely '*Multi-criteria Protein Structure Comparison and Analysis* (MC-PSC)'. The thesis builds on the critical review of the related literature and uses the knowledge gained from this review to design and implement an optimal distributed framework to achieve the best solution for the problem of MC-PSC. The efficiency and performance of the proposed framework has been evaluated and different strategies have been used for the optimization of the overall system. The contributions and findings of this work have been presented coherently in separate chapters and are summarized in the following section. Section 10.3 also provides some guidelines for possible future enhancements.

10.2 Contributions

Chapter 1 sets the stage by providing the historical background on protein structure comparison, the need for multi-criteria protein structure comparison (MC-PSC) and the problem it faces in terms of computational challenge. This chapter also discusses the proposed solution along with general methodology and organization of the dissertation.

The material presented in chapter 2 contributes in two ways: *firstly*, it provides a synthetic overview of several major contributions in a way that could help and serve a wide range of individuals and organizations interested in:

1. Setting up a local, enterprise or global IT infrastructure for life sciences,
2. Solving a particular life science related problem by selecting the most appropriate technological options that have been successfully demonstrated and reported herein,
3. Migrating already existing bioinformatics legacy applications, tools and services to a grid-enabled environment in a way that requires less effort and is motivated with previous related studies provided herein,
4. comparing a newly developed application/ service features with those currently available, and
5. Starting a research and development career related to the use of web and grid technology for biosciences.

Secondly, it identifies some key open problems such as:

1. Biological data analysis and management is still quite a difficult job because of the lack of development and adaptation of optimized and unified data models and query engines,

2. Some of the existing bioinformatics ontologies and workflow management systems are simply in the form of Directed Acyclic Graphs (DAGs) and their descriptions are lacking expressiveness in terms of formal logic [135],

3. Lack of open-source standards and tools required for the development of thesaurus and meta-thesaurus services [77],

4. Need of appropriate query, visualization and authorization mechanism for the management of provenance data and meta-data in in-silico experiments [68, 135],

5. Some of the BioGrid projects seem to be discontinued in terms of information updating. This might arise from funding problems or difficulties associated with their implementation,

6. There is a lack of domain specific mature application programming models, toolkits and APIs for grid-enabled application development, deployment, debugging and testing,

7. Still there seems to be a gap between the application layer and middleware layer of a typical BioGrid infrastructure because existing middleware services do not fully facilitate the demands of applications such as there is no proper support in any grid middleware for automatic application deployment on all grid nodes,

8. It is not trivial to deploy existing bioinformatics applications on available grid testbed (such as NGS, EGEE etc), as this requires the installation and configuration of specific operating system and grid middleware toolkits, which is not at least easy from a biologist end-user point of view,

9. It has been observed that there are still many issues with grid based workflow management

systems in terms of their support for complex operations (such as loops), legacy bioinformatics applications and tools, use of proper ontology and web services etc. [135],

10. The job submission process on existing grid infrastructures seems to be quite complex because of inappropriate maturity of resource broker services, and

11. Lack of appropriate implementation initiative regarding knowledge grid infrastructure for life sciences.

This chapter was published as a peer reviewed journal article in *Current Bioinformatics*, Vol. 3(1), pp.10-31, 2008. [doi:10.2174/157489308783329850].

Chapter 3, builds on the assumption that the diversity of enabling technologies for grid and distributed computing makes it difficult for the developer to select most appropriate technological infrastructure with proved technological standards and tools. It therefore, performs a more focused review of various demonstrations related to the use of grid and distributed public computing schemes for structural proteomics in order to provide a road-map in this dilemma. This chapter identifies that the selection of an appropriate grid/distributed computing approach mainly depends on the nature of the application. For example, applications with an independent and parallel nature of jobs are more suitable for distributed computing based on publicly-owned resources using middleware such as BOINC, UD MetaProcessor etc. On the other hand, if the application requires some pre-determined and controlled quality of service in terms of data and process management with enhanced reliability and security, then organizational or cross-organizational grid infrastructure with standard middleware (such as Globus, Legion, NetSolve, Ninf, myGrid, Condor/G, etc.) would serve in a better way. Additionally, this chapter also provides the background of multi-criteria

protein structure comparison in terms of the ProCKSI's architecture and functionality.

Parts of this chapter were published as a peer reviewed conference paper in the *Proceedings of the Frontiers of High Performance Computing and Networking ISPA 2007 Workshops,* LNCS Vol.4743 pp.424-434, 2007. [doi:10.1007/978-3-540-74767-3_44]

Based on the comprehensive understanding of the problem at hand, as described in the earlier chapters, the proposed methodology for the solution has been presented in chapter 4. This chapter also explains the programming environment that has been used for the implementation of the solution as well as the description of the test datasets along with performance measures and metrics that has been used for the evaluation of the implemented solutions.

The design and implementation of an innovative distributed algorithm for MC-PSC has been introduced in Chapter 5. The design of this algorithm have been analyzed in terms of space, time, and communication overhead. Based on this analysis two different load balancing approaches have been used to improve the overall performance: *even* and *uneven* strategies. The *even* strategy considers the total number of protein structures/pairs to be compared and distributes them equally on each of the available processors; while the *uneven* strategy tries to consider the length of each protein (in terms of number of residues) and assigns the protein structures/pairs to each of the available processors in a way that each processor gets different number of protein structures/pairs but nearly equal number of overall residues. The former permits to obtain the best distribution in terms of memory, while the latter performs better in terms of execution time and scalability on the cluster/grid computers. Experiments conducted on medium and large real datasets prove that

the algorithm permits to reduce execution time (i.e. for the RS119 dataset it was reduced from 6 days on a single processor to about 5 hours on 64 processors) and to cope with problems otherwise not tractable on a single machine as the Kinjo dataset, which took about 11 days on a 64-processors cluster. In the future, we intend to investigate in more depth the use of grid computing environments in order to cope with very large proteomics datasets.

This chapter was published as a peer reviewed journal paper in IEEE Transactions on NanoBioscience, Vol. 9(2), pp.144-155, 2010. [doi:10.1109/TNB.2010.2043851]

Chapter 6 studied the effect of different integrated environments on the performance of the newly developed algorithm. The aim of this study was to provide a scientific basis for decision making while building a large scale e-Science infrastructure that meets with the requirements of today's high-impact scientific applications. The study makes use of the Sun Grid Engine (SGE) as a distributed local resource management system and tries to integrate the support for parallel environments using different MPI implementations. The integration has been achieved in two ways i.e *Loose Integration* and *Tight Integration* along with MPI's different methods for starting the jobs i.e *SMPD daemon-based* and *SMPD daemon-less*. The results of the evaluation indicate that the *Loose Integration* method is not much reliable in terms of accounting and monitoring information to be used for PSC jobs. Furthermore, for larger datasets, *Tight Integration* with SMPD daemon-based method outperforms its counterpart i.e. SMPD daemon-less method. It has also been learned that in a heterogeneous cluster where some nodes have double the performance as that of others, the slowest node could become a bottleneck for the overall performance of the system.

This chapter was published as a peer reviewed conference paper in *Proceedings of the*

IEEE International Symposium on Parallel and Distributed Processing with Applications ISPA '08,ISBN: 978-0-7695-3471-8, pp.817-822, 2008. [doi:10.1109/ISPA.2008.41]

The Scalability of the high-throughput distributed framework for MC-PSC in the grid environment was presented in chapter 7. Though, the grand challenge nature of the MC-PSC problem requires the use of Grid computing to overcome the limitations of a single parallel computer/cluster, however; the use of the Grid computing also introduces additional communication overhead and hence the standard parallel computing measures and metrics such as *Speedup* and *Efficiency* need to be redefined [225]. This chapter introduced the grid-based definition of the speedup (*Grid Speedup*), and efficiency (*Grid Efficiency*) by taking ideas from Hoekstra et al. [225]. These Grid metrics were then used to measure the scalability of our distributed algorithm on the *UK National Grid Service* (NGS) [66] architecture using the latest implementation of grid-enabled MPI i.e. MPIg [276]. The results of cross-site scalability were compared with single-site and single-machine performance to analyze the additional communication overheard in a wide-area network. It has been observed that the algorithm performs well on the combined resources of the two sites.

This chapter was published as a peer reviewed conference paper in *Proceedings of The Euro-Par 2010 Workshop on High Performance Bioinformatics and Biomedicine (HiBB)*, August 31-Sep 3, 2010 , Ischia, Naples, Italy.

Chapter 8 performed the study to identify appropriate technology which could be used for the storage, management and analysis of (multi)Similarity data resulting from the comparison of very large datasets. For this case, different experiments were performed to evaluate the per-

formance/efficiency of HDF5 and Oracle. In particular, the storage/query overhead of HDF5 was compared with that of Oracle/SQL. The results showed significant performance benefit of HDF5 over Oracle/SQL. This initial benchmarking, provides a road-map to compute the (multi) similarities of all available PDB structures; thereby, creating an efficient scientific knowledgebase of pre-computed results. This knowledgebase could be used to provide an easy to use interface, so the biologist can perform scientific queries using a vast variety of criteria and options leading to far better and reliable understanding of the protein structural universe.

This chapter was published as a peer reviewed conference paper in *Proceedings of 22nd IEEE International Symposium on Computer-Based Medical Systems (CBMS-09,* ISBN:978-1-4244-4879-1, pp.1-8, 2009. [doi:10.1109/CBMS.2009.5255328]

Finally, chapter 9 discusses the concept of consensus based protein structure similarity clustering. It uses the *Protein Kinase Dataset* to compare the results of two competing paradigms in the field of systematics i.e., total evidence and total consensus. The results of this work indicate that for the case of protein structure similarity clustering the *total evidence* approach gives better results as compared to total consensus. Hence, this finding strengthens the current protocol being used in the ProCKSI.

Needless to say that, in making all this work, there were tremendous problems involved in the configuration, installation and operation of the Grid computing related hardware and software which consumed too much time for their fixing. Some of these problems are reported in Appendix .2. Nevertheless, this thesis resulted in 5 peer reviewed conference papers (which needed traveling

to be presented at as far countries as Canada, Australia and USA) and 2 peer reviewed journal papers, 2340 lines of code and several giga bytes of data.

10.3 Future Directions

The work presented in this dissertation sets the foundation for the real time multi-criteria protein structure comparison and analysis. The work opens many directions for future research including:

- The developed software could be deployed on the high-end resources such as the *UK's High End Computing Terascale Resource* (HECToR) and thereby, compute the all-against-all similarities for all the structures available in the PDB. However, the access to HECToR is not free and requires the applicant to prepare a full research grant proposal for submission to EPSRC or BBSRC. Furthermore, in order to make such a deployment, the software developed so far could be upgraded by adding the fault tolerance mechanism in the form of checkpoint/restart. The checkpoint/restart support could be added without changing the code of the application by using some libraries such as the *Berkeley Lab Checkpoint/Restart* (BLCR) [156].

- Another important direction would be to investigate the proper visualization of the clusters resulting from the multi-verse of similarities. This large scale clustering and visualization could be achieved by making use of the newly introduced open-source algorithm *memory-constrained UPGMA* (MC-UPGMA) and its related visualization interface that is being used for single-similarities in the domain of sequence comparison/alignment.

- It would be interesting to exhaustively and systematically explore the utilization of fuzzy decision making for MC-PSC. In [183–185] the authors took first steps into that direction but more work remains to be done.

References

[1] G. J. Mulder, “ On the composition of some animal substances ,” *Journal fuktische Chemie*, vol. 16, pp. 129–, 1839.

[2] F. Sanger, E. Thompson, and R. Kitai, “ The amide groups of insulin ,” *Biochem J.*, vol. 59, no. 3, pp. 509–518, 1955.

[3] J. Kendrew, G. Bodo, H. Dintzis, R. Parrish, H. Wyckoff, and D. Phillips, “ A three-dimensional model of the myoglobin molecule obtained by x-ray analysis ,” *Nature*, vol. 181, no. 4610, pp. 662–666, 1958.

[4] H. Muirhead and M. Perutz, “ Structure of hemoglobin. A three-dimensional fourier synthesis of reduced human hemoglobin at 5.5 Angstrom resolution ,” *Nature*, vol. 199, no. 4894, pp. 633–638, 1963.

[5] C. B. Anfinsen, R. R. Redfield, W. L. Choate, J. Page, and W. R. Carroll, “Studies on the gross structure, cross-linkages, and terminal sequences in ribonuclease.” *The Journal of biological chemistry*, vol. 207, no. 1, pp. 201–210, March 1954.

[6] C. B. Anfinsen, E. Haber, M. Sela, and F. H. White, “The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain.” *Proceedings of the National*

Academy of Sciences of the United States of America, vol. 47, pp. 1309–1314, September 1961.

[7] C. B. Anfinsen, "Principles that govern the folding of protein chains." *Science*, vol. 181, no. 96, pp. 223–230, July 1973.

[8] J. C. Wooley and Y. Ye, *A Historical Perspective and Overview of Protein Structure Prediction*. Springer New York, 2007, pp. 1–43.

[9] B. Rost, "Twilight zone of protein sequence alignments," *Protein Eng*, vol. 12, no. 2, pp. 85–94, February 1999. [Online]. Available: http://dx.doi.org/10.1093/protein/12.2.85

[10] J. M. Berg, J. L. Tymoczko, and L. Stryer, *Protein Structure and Function*. San Francisco: W.H. Freeman, 2002.

[11] F. Sanger, G. Air, B. Barrell, N. Brown, A. Coulson, C. Fiddes, C. Hutchison, P. Slocombe, and M. Smith, "Nucleotide sequence of bacteriophage phi X174 DNA," *Nature*, vol. 265, no. 5596, pp. 687–695, 1977.

[12] E. S. Lander, L. M. Linton, B. Birren, and et al., "Initial sequencing and analysis of the human genome ," *Nature*, vol. 409, pp. 860–921, 2001.

[13] A. Williamson, "Creating a structural genomics consortium," *Nat Struct Biol*, vol. 7, 2000.

[14] B. Matthews, "Protein structure initiative: getting into gear," *Nat Struct Mol Biol*, vol. 14, pp. 459–60, 2007.

[15] Editorial, "Proteomics' new order," *Nature*, vol. 437, pp. 169–70, 2005.

[16] The-UniProt-Consortium, “The Universal Protein Resource (UniProt),” *Nucl. Acids Res.*, vol. 36, no. 1, pp. D190–195, 2008. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/36/suppl_1/D190

[17] E. Jain, A. Bairoch, S. Duvaud, I. Phan, N. Redaschi, B. Suzek, M. Martin, P. McGarvey, and E. Gasteiger, “Infrastructure for the life sciences: design and implementation of the uniprot website,” *BMC Bioinformatics*, vol. 10, no. 1, p. 136, 2009. [Online]. Available: http://www.biomedcentral.com/1471-2105/10/136

[18] H. Berman, K. Henrick, and H. Nakamura, “Announcing the worldwide protein data bank,” *Nat Struct Biol*, vol. 10, p. 980, 2003.

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

[20] A. Zemla, C. Venclovas, J. Moult, and K. Fidelis, “Processing and analysis of casp3 protein structure predictions,” *Proteins Struct Funct Genet*, vol. 3, pp. 22–29, 1999.

[21] D. Kihara, Y. Zhang, H. Lu, A. Kolinski, and J. Skolnick, “Ab initio protein structure prediction on a genomic scale: Application to the mycoplasma genitalium genome,” *PNAS*, vol. 99, pp. 5993–5998, 2002.

[22] Y. Zhang and J. Skolnick, “Automated structure prediction of weakly homologous proteins on a genomic scale,” *PNAS*, vol. 101, p. 7594âĂŞ7599, 2004.

[23] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, “Scop: a structural classification

of proteins database for the investigation of sequences and structures," *J Mol Biol*, vol. 247, pp. 536–540, 1995.

[24] C. A. Orengo, A. D. Michie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thornton, "Cath - a hierarchic classification of protein domain structures," *Structure*, vol. 5, pp. 1093–1108, 1997.

[25] S.-Y. Huang and X. Zou, "Efficient molecular docking of nmr structures: Application to hiv-1 protease," *Protein Sci.*, vol. 16, pp. 43–51, 2007.

[26] R. Kolodny, P. Koehl, and M. Levitt, "Comprehensive evaluation of protein structure alignment methods: Scoreing by geometric measures," *J Mol Biol*, vol. 346, pp. 1173–1188, 2005.

[27] 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*, vol. 8, p. 416, 2007.

[28] O. Trelles, "On the parallelisation of bioinformatics applications," *Briefings in Bioinformatics*, vol. 2, pp. 181–194, 2001.

[29] S. F. Altschul, T. L. Madden, and A. A. Schaffer, "Gapped blast and psi-blast: A new generation of protein db search programs," *Nucleic Acids Res.*, vol. 25, pp. 3389–3402, 1997.

[30] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *Proc. Natl Acad. Sci.*, vol. 85, pp. 2444–2448, 1988.

[31] "The bio-accelerator." [Online]. Available: http://sgbcd//weizmann.ac.il/

[32] R. K. Singh, W. D. Dettloff, V. L. Chi, D. L. Hoffman, S. G. Tell, C. T. White, S. F. Altschul, and B. W. Erickson, “Bioscan: A dynamically reconïňĄgurable systolic array for biosequence analysis,” in *Proc. of CERCS96, National Science Foundation*, 1996.

[33] F. Berman, G. Fox, and A. J. G. Hey, *Grid Computing: Making the Global Infrastructure a Reality*, T. Hey, Ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[34] I. Foster and C. Kesselman, “Globus: A metacomputing infrastructure toolkit,” *International Journal of Supercomputer Applications*, vol. 11, pp. 115–128, 1996.

[35] I. Foster, “Globus toolkit version 4: Software for service-oriented systems,” in *IFIP International Conference on Network and Parallel Computing*, ser. LNCS 3779, 2005, pp. 2–13.

[36] A. Shah, D. Barthel1, P. Lukasiak, J. Blacewicz, and N. Krasnogor, “Web and grid technologies in bioinformatics, computational biology and systems biology: A review,” *Current Bioinformatics*, vol. 3, no. 1, pp. 10–31, 2008.

[37] A. Shah, D. Barthel, and N. Krasnogor, “Grid and distributed public computing schemes for structural proteomics,” in *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, ser. LNCS 4743. Berlin:Springer, 2007, pp. 424–434.

[38] S. Pellicer, G. Chen, K. C. C. Chan, and Y. Pan, “Distributed sequence alignment applications for the public computing architecture,” *IEEE Transactions on NanoBioscience*, vol. 7, pp. 35–43, 2008.

[39] W.-L. Chang, “Fast parallel dna-based algorithms for molecular computation: The set-partition problem,” *IEEE Transactions on NanoBioscience*, vol. 6, pp. 346–353, 2007.

[40] I. Merelli, G. Morra, and L. Milanesi, "Evaluation of a grid based molecular dynamics approach for polypeptide simulations," *IEEE Transactions on NanoBioscience*, vol. 6, pp. 229–234, 2007.

[41] M. Mirto, M. Cafaro, S. L. Fiore, D. Tartarini, and G. Aloisio, "Evaluation of a grid based molecular dynamics approach for polypeptide simulations," *IEEE Transactions on NanoBioscience*, vol. 6, pp. 124–130, 2007.

[42] A. Arbona, S. Benkner, G. Engelbrecht, J. Fingberg, M. Hofmann, K. Kumpf, G. Lonsdale, and A. Woehrer, "A service-oriented grid infrastructure for biomedical data and compute services," *IEEE Transactions on NanoBioscience*, vol. 6, pp. 136–141, 2007.

[43] M. Cannataro, A. Barla, R. Flor, G. Jurman, S. Merler, S. Paoli, G. Tradigo, P. Veltri, and C. Furlanello, "A grid environment for high-throughput proteomics," *IEEE Transactions on NanoBioscience*, vol. 6, pp. 117–123, 2007.

[44] A. Boccia, G. Busiello, L. Milanesi, and G. Paolella, "A fast job scheduling system for a wide range of bioinformatic applications," *IEEE Transactions on NanoBioscience*, vol. 6, pp. 149–154, 2007.

[45] L. Holm and J. Park, "Dalilite workbench for protein structure comparison," *Bioinformatics*, vol. 16, pp. 566–567, 2000.

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

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

[48] N. Krasnogor and D. A. Pelta, “Measuring the similarity of protein structures by means of the universal similarity metric,” *Bioinformatics*, vol. 20, pp. 1015–1021, 2004.

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

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

[51] “iHOP: Information hyperlinked over proteins.” [Online]. Available: "http://www.ihop-net.org"

[52] “SCOP: Structural classification of proteins.” [Online]. Available: http://scop.mrc-lmb.cam.ac.uk/scop

[53] F. M. G. Pearl, C. F. Bennett, J. E. Bray, A. P. Harrison, N. Martin, A. Shepherd, I. Sillitoe, J. Thornton, and C. A. Orengo, “The cath database: an extended protein family resource for structural and functional genomics,” *Nucleic Acids Res*, vol. 31, pp. 452–455, 2003.

[54] O. Camoglu, T. Can, and A. Singh, “Integrating multi-attribute similarity networks for robust representation of the protein space,” *Bioinformatics*, vol. 22, pp. 1585–1592, 2006.

[55] V. Filkov and S. Skiena, “Heterogeneous data integration with the consensus clustering formalism,” in *1st International Workshop on Data Integration in the Life Science (DILS)*, ser. LNCS 2994. Berlin:Springer, 2004, pp. 110–123.

[56] C. A. Rohl, C. E. M. Strauss, K. M. S. Misura, and D. Baker, “Protein Structure Predic-

tion Using Rosetta," in *Numerical Computer Methods, Part D*, ser. Methods in Enzymology, L. Brand and M. L. Johnson, Eds. Academic Press, Jan. 2004, vol. Volume 383, pp. 66–93.

[57] S. Wu, J. Skolnick, and Y. Zhang, "Ab initio modeling of small proteins by iterative TASSER simulations." *BMC Biol*, vol. 5, no. 1, p. 17, May 2007.

[58] A. Naseer and L. K. Stergioulas, "Integrating grid and web services: A critical assessment of methods and implications to resource discovery," in *Proceedings of the 2nd Workshop on Innovations in Web Infrastructure Edinburgh, UK*, 2005.

[59] J. C. Wooley and H. S. Lin, *Catalyzing Inquiry at the Interface of Computing and Biology*, 1st ed. The National Academies Press, Washington, DC, 2005.

[60] D. Malakoff, "Nih urged to fund centers to merge computing and biology," *Science*, vol. 284, no. 5421, pp. 1742–1743, 1999.

[61] M. Galperin, "The molecular biology database collection: 2006 update," *Nucleic Acids Res*, vol. 34, pp. D3–D5, 2006.

[62] C. Baru, R. Moore, A. Rajasekar, and M. Wan, "The sdsc storage resource broker," in *In Proceedings of CASCON*, 1998.

[63] L. M. Haas, P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope, "Discoverylink: A system for integrated access to life sciences data sources," *IBM Systems Journal*, vol. 40, no. 2, 2001.

[64] S. Date, K. Fujikawa, H. Matsuda, H. Nakamura, and S. Shimojo, "An empirical study of

grid applications in life science - lesson learnt from biogrid project in japan -," *International Journal of Information Technology .*, vol. 11, no. 3, pp. 16–28, 2005.

[65] F. Gagliardi, B. Jones, F. Grey, M.-E. Begn, and M. Heikkurinen, "Building an infrastructure for scientific Grid computing: status and goals of the EGEE project," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 363, no. 1833, pp. 1729–1742, 2005.

[66] A. Richards and G. M. Sinclair, *UK National Grid Service*. CRC Press, 2009.

[67] M. Podvinec, T. Schwede, L. Cerutti, P. Kunszt, B. Nyffeler, H. Stockinger, S. Maffioletti, A. Thomas, and C. Turker, "The swissbiogrid project: Objectives, preliminary results and lessons learned," in *Proceedings of the 2nd IEEE International Conference on e-Science and Grid Computing, Amsterdam, Netherlands*, 2007.

[68] R. D. Stevens, A. J. Robinson, and C. A. Goble, "myGrid: personalised bioinformatics on the information grid," *Bioinformatics*, vol. 19, no. 1, pp. i302–i304, 2003.

[69] C. G. Chris, C. Wroe, and R. Stevens, "myGrid Project: Services, architecture and demonstrator," in *Proceedings of UK e-Science All Hands Meeting Nottingham*, 2003.

[70] M. D. Wilkinson and M. Links, "myGrid: personalised bioinformatics on the information grid," *Briefings in Bioinformatics*, vol. 3, no. 4, pp. 331–341, 2002.

[71] M. Wilkinson, H. Schoof, R. Ernst, and D. Haase, "Biomoby successfully integrates distributed heterogeneous bioinformatics web services. the planet exemplar case," *Plant Physiology*, vol. 138, pp. 5–17, 2005.

[72] A. Kerhornou and R. Guigo, “Biomoby web services to support clustering of co-regulated genes based on similarity of promoter configurations,” *Bioinformatics*, vol. 23, no. 14, pp. 1831–1833, 2007.

[73] P. Rice, I. Longden, and A. Bleasby, “Emboss: The european molecular biology open software suite,” *Trends in Genetics*, vol. 16, no. 6, pp. 276–277, 2000.

[74] G. P. Trabado, *A Survey on Grid Architectures for Bioinformatics*. American Scientific Publishers, USA, 2006, pp. 109–112.

[75] A. Krishnan, “A survey of life sciences applications on the grid,” *New Generation Computing*, vol. 22, no. 2, pp. 111–125, 2004. [Online]. Available: http://dx.doi.org/10.1007/BF03040950

[76] X. Wang, R. Gorlitsky, and J. S. Almeida, “From xml to rdf: how semantic web technologies will change the design of ’omic’ standards,” *Nature Biotechnology*, vol. 23, no. 9, pp. 1099–1103, September 2005. [Online]. Available: http://dx.doi.org/10.1038/nbt1139

[77] S. S. Sahoo, C. Thomas, A. Sheth, W. S. York, and S. Tartir, “Knowledge modeling and its application in life sciences: a tale of two ontologies,” in *WWW ’06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA: ACM, 2006, pp. 317–326.

[78] I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen, “From shiq and rdf to owl: The making of a web ontology language,” *Journal of Web Semantics*, vol. 1, p. 2003, 2003.

[79] Y.-P. P. Chen and Q. Chen, “Analyzing inconsistency toward enhancing integration of biolog-

ical molecular databases," in *Proceedings of the 4th Asia-Pacific Bioinformatics Conference*. London, England: Imperial College Press, 2006, pp. 197–206.

[80] K. Cheung, A. Smith, K. Yip, C. Baker, and M. Gerstein, *Semantic Web Approach to Database Integration in the Life Sciences*. Springer, NY, 2007, pp. 11–30.

[81] J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, , and M. Greenwood, *Using Semantic Web Technologies for Representing E-science Provenance*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3298/2004, pp. 92–106.

[82] D. A. Quan and R. Karger, "How to make a semantic web browser," in *WWW '04: Proceedings of the 13th international conference on World Wide Web*. New York, NY, USA: ACM Press, 2004, pp. 255–265. [Online]. Available: http://dx.doi.org/10.1145/988672.988707

[83] C. Geneontology, "Creating the gene ontology resource: design and implementation." *Genome Res*, vol. 11, no. 8, pp. 1425–1433, August 2001. [Online]. Available: http://dx.doi.org/10.1101/gr.180801

[84] G. O. Consortium, "The Gene Ontology (GO) project in 2006," *Nucl. Acids Res.*, vol. 34, no. 1, pp. 322–326, 2006.

[85] A. R. ttenberg, J. A. Rees, and J. S. L. ciano, "Experience using OWL-DL for the exchange of biological pathway information," in *Workshop on OWL Experiences and Directions*, 2005.

[86] A. Ruttenberg, J. Rees, and J. Zucker, "What biopax communicates and how to extend owl to help it," in *OWL: Experiences and Directions Workshop Series*, 2006.

[87] P. L. Whetzel, H. Parkinson, H. C. Causton, L. Fan, J. Fostel, G. Fragoso, L. Game, M. Heiskanen, N. Morrison, P. Rocca-Serra, S.-A. Sansone, C. Taylor, J. White, and J. Stoeckert, Christian J., "The MGED Ontology: a resource for semantics-based description of microarray experiments," *Bioinformatics*, vol. 22, no. 7, pp. 866–873, 2006. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/22/7/866

[88] M. Navarange, L. Game, D. Fowler, V. Wadekar, H. Banks, N. Cooley, F. Rahman, J. Hinshelwood, P. Broderick, and H. Causton, "Mimir: a comprehensive solution for storage, annotation and exchange of microarray data," *BMC Bioinformatics*, vol. 6, no. 1, p. 268, 2005. [Online]. Available: http://www.biomedcentral.com/1471-2105/6/268

[89] A. Brazma, P. Hingamp, J. Quackenbush, G. Sherlock, P. Spellman, C. Stoeckert, J. Aach, W. Ansorge, C. A. Ball, H. C. Causton, T. Gaasterland, P. Glenisson, F. C. Holstege, I. F. Kim, V. Markowitz, J. C. Matese, H. Parkinson, A. Robinson, U. Sarkans, S. Schulze-Kremer, J. Stewart, R. Taylor, J. Vilo, and M. Vingron, "Minimum information about a microarray experiment (miame)-toward standards for microarray data." *Nat Genet*, vol. 29, no. 4, pp. 365–371, December 2001. [Online]. Available: http://dx.doi.org/10.1038/ng1201-365

[90] A. Brazma, "On the importance of standardisation in life sciences." *Bioinformatics*, vol. 17, no. 2, pp. 113–114, February 2001. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/11238066

[91] J. S. Luciano, "Pax of mind for pathway researchers." *Drug Discov Today*, vol. 10, no. 13, pp. 937–942, July 2005. [Online]. Available: http://dx.doi.org/10.1016/S1359-6446(05)03501-4

[92] G. Joshi-Tope, M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G. Gopinath, G. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein, "Reactome: a knowledgebase of biological pathways," *Nucl. Acids Res.*, vol. 33, no. 1, pp. 428–432, 2005. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/33/suppl_1/D428

[93] A. Brazma, A. Robinson, G. Cameron, and M. Ashburner, "One-stop shop for microarray data." *Nature*, vol. 403, no. 6771, pp. 699–700, February 2000. [Online]. Available: http://dx.doi.org/10.1038/35001676

[94] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass, "TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources ," *Bioinformatics*, vol. 16, no. 2, pp. 184–186, 2000. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/16/2/184

[95] G. A. Komatsoulis, D. B. Warzel, F. W. Hartel, K. Shanbhag, R. Chilukuri, G. Fragoso, S. de Coronado, D. M. Reeves, J. B. Hadfield, C. Ludet, and P. A. Covitz, "cacore version 3: Implementation of a model driven, service-oriented architecture for semantic interoperability," *Journal of Biomedical Informatics*, vol. 41, no. 1, pp. 106 – 123, 2008.

[96] M. Altunay, D. Colonnese, and C. Warade, "High throughput web services for life sciences," in *ITCC '05: Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 329–334.

[97] U. Hahn, J. Wermter, R. Blasczyk, and P. A. Horn, "Text mining: powering the database

revolution," *Nature*, vol. 448, no. 7150, p. 130, July 2007. [Online]. Available: http://dx.doi.org/10.1038/448130b

[98] H. T. Gao, J. H. Hayes, and H. Cai, "Integrating biological research through web services," *Computer*, vol. 38, no. 3, pp. 26–31, 2005.

[99] N. Furmento, J. Hau, W. Lee, S. Newhouse, and J. Darlington, "Implementations of a service-oriented architecture on top of jini, jxta and ogsi," in *AxGrids 2004, LNCS 3165*, 2004, pp. 90–99.

[100] M. Radenkovic and B. Wietrzyk, "Life science grid middleware in a more dynamic environment," in *On the Move to Meaningful Internet Systems 2005: OTM Workshops, LNCS 3762*, 2005, pp. 264–273.

[101] E. Merelli, G. Armano, N. Cannata, F. Corradini, M. d'Inverno, A. Doms, P. Lord, A. Martin, L. Milanesi, S. Moller, M. Schroeder, and M. Luck, "Agents in bioinformatics, computational and systems biology," *Brief Bioinform*, vol. 8, no. 1, pp. 45–59, 2007. [Online]. Available: http://bib.oxfordjournals.org/cgi/content/abstract/8/1/45

[102] L. Moreau, S. Miles, C. Goble, M. Greenwood, V. Dialani, M. Addis, N. Alpdemir, R. Cawley, D. D. Roure, J. Ferris, R. Gaizauskas, K. Glover, Chris, C. Greenhalgh, P. Li, X. Liu, P. Lord, M. Luck, D. Marvin, T. Oinn, N. Paton, S. Pettifer, V. Radenkovic, A. Roberts, A. Robinson, T. Rodden, M. Senger, N. Sharman, R. Stevens, B. Warboys, A. Wipat, and C. Wroe, "On the use of agents in a bioinformatics grid," in *Proc. of the 3rd IEEE/ACM CCGRID-2003 Workshop on Agent Based Cluster and Grid Computing*, 2003, pp. 653–661.

[103] K. Bryson, M. Luck, M. Joy, and D. T. Jones, "Agent interaction for bioinformatics data management," *Applied Artificial Intelligence*, vol. 15, pp. 200–1, 2002.

[104] K. Decker, S. Khan, C. Schmidt, and D. Michaud, "Extending a multi-agent system for genomic annotation," in *Cooperative Information Agents IV*. Springer-Verlag, 2001, pp. 106–117.

[105] I. Foster, "Service-Oriented Science," *Science*, vol. 308, no. 5723, pp. 814–817, 2005. [Online]. Available: http://www.sciencemag.org/cgi/content/abstract/308/5723/814

[106] N. Jacq, C. Blanchet, C. Combet, E. Cornillot, L. Duret, K. Kurata, H. Nakamura, T. Silvestre, and V. Breton, "Grid as a bioinformatic tool," *Parallel Comput.*, vol. 30, no. 9-10, pp. 1093–1107, 2004.

[107] J. Felsenstein, "Evolutionary trees from dna sequences: a maximum likelihood approach." *J Mol Evol*, vol. 17, no. 6, pp. 368–376, 1981. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/7288891

[108] C. Blanchet, R. Mollon, D. Thain, and G. Deleage, "Grid deployment of legacy bioinformatics applications with transparent data access," in *GRID '06: Proceedings of the 7th IEEE/ACM International Conference on Grid Computing*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 120–127.

[109] D. Thain and M. Livny, "Parrot: Transparent user-level middleware for data-intensive computing," in *In Workshop on Adaptive Grid Middleware*, 2003.

[110] A. Bairoch and R. Apweiler, "The SWISS-PROT protein sequence database and its

supplement TrEMBL in 2000," *Nucl. Acids Res.*, vol. 28, no. 1, pp. 45–48, 2000. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/28/1/45

[111] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool." *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403–410, October 1990. [Online]. Available: http://dx.doi.org/10.1006/jmbi.1990.9999

[112] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences." *J Mol Biol*, vol. 147, no. 1, pp. 195–197, March 1981. [Online]. Available: http://view.ncbi.nlm.nih.gov/pubmed/7265238

[113] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, pp. 4673–4680, 1994.

[114] W. R. Pearson, "Searching protein sequence libraries: Comparison of the sensitivity and selectivity of the smith-waterman and fasta algorithms," *Genomics*, vol. 11, no. 3, pp. 635–650, 1991.

[115] C.-T. Yang, Y.-L. Kuo1, K.-C. Li, and J.-L. Gaudiot, "On design of cluster and grid computing environment toolkit for bioinformatics applications," in *Distributed Computing - IWDC 2004, LNCS 3326/2005*. Springer Berlin / Heidelberg, 2006, pp. 82–87.

[116] C.-T. Yang, Y.-C. Hsiung, and H.-C. Kan, "Implementation and evaluation of a java based computational grid for bioinformatics applications," *Advanced Information Networking and Applications, International Conference on*, vol. 1, pp. 298–303, 2005.

[117] J. Kommineni and D. Abramson, “Griddles enhancements and building virtual applications for the grid with legacy components,” in *Advances in Grid Computing - EGC 2005, LNCS 3470/2005*. Springer Berlin / Heidelberg, 2005, pp. 961–971.

[118] D. Sulakhe, A. Rodriguez, M. DSouza, M. Wilde, V. Nefedova, I. Foster, and N. Maltsev, “Gnare: Automated system for high-throughput genome analysis with grid computational backend,” *The Journal of Clinical Monitoring and Computing*, vol. 19, pp. 361–369(9), 2005. [Online]. Available: http://www.ingentaconnect.com/content/klu/jocm/2005/00000019/F0020004/00003463

[119] H. Casanova, F. Berman, T. Bartol, E. Gokcay, T. Sejnowski, A. Birnbaum, J. Dongarra, M. Miller, M. Ellisman, M. Faerman, G. Obertelli, R. Wolski, S. Pomerantz, and J. Stiles, “The virtual instrument: Support for grid-enabled mcell simulations,” *Int. J. High Perform. Comput. Appl.*, vol. 18, no. 1, pp. 3–17, 2004.

[120] P. K. Dhar, T. C. Meng, S. Somani, L. Ye, K. Sakharkar, A. Krishnan, A. B. Ridwan, S. H. K. Wah, M. Chitre, and Z. Hao, “Grid Cellware: the first grid-enabled tool for modelling and simulating cellular processes,” *Bioinformatics*, vol. 21, no. 7, pp. 1284–1287, 2005. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/21/7/1284

[121] H. Nakada, M. Sato, and S. Sekiguchi, “Design and implementations of ninf: towards a global computing infrastructure,” *Future Gener. Comput. Syst.*, vol. 15, no. 5-6, pp. 649–658, 1999.

[122] M. Dahan and J. R. Boisseau, “The gridport toolkit: A system for building grid portals,” in

HPDC '01: Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing. Washington, DC, USA: IEEE Computer Society, 2001, p. 216.

[123] J. Novotny, "The grid portal development kit," in *Grid Computing: Making the Global Infrastructure a Reality*, 2001.

[124] Z. Chongjie, "Grid portal toolkits," in *Portals Portlets and GridSphere Workshop*, 2005.

[125] L. Ramakrishnan, S. Reed, J. L. Tilson, and D. A. Reed, "Grid portals for bioinformatics," in *Second International Workshop on Grid Computing Environments*, 2006.

[126] J. Alameda, M. Christie, G. Fox, J. Futrelle, D. Gannon, M. Hategan, G. Kandaswamy, G. von Laszewski, M. A. Nacar, M. Pierce, E. Roberts, C. Severance, and M. Thomas, "The open grid computing environments collaboration: portlets and services for science gateways: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 19, no. 6, pp. 921–942, 2007.

[127] C. Letondal, "A Web interface generator for molecular biology programs in Unix ," *Bioinformatics*, vol. 17, no. 1, pp. 73–82, 2001. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/17/1/73

[128] E. Lu, Z. Xu, and J. Sun, "An extendable grid simulation environment based on gridsim," in *Grid and Cooperative Computing, LNCS 3032/2004*. Springer Berlin / Heidelberg, 2004, pp. 205–208.

[129] R. Buyya, M. Murshed, and D. Abramson, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," in *Journal of Concurrency and Computation: Practice and Experience (CCPE*. Wiley Press, 2002, pp. 1175–1220.

[130] H. Lamehamedi, Z. Shentu, B. Szymanski, and E. Deelman, “Simulation of dynamic data replication strategies in data grids,” in *In Proc. 12th Heterogeneous Computing Workshop (HCW2003*. IEEE CS Press, 2003.

[131] Y.-M. Teo, X. Wang, and Y.-K. Ng, “GLAD: a system for developing and deploying large-scale bioinformatics grid,” *Bioinformatics*, vol. 21, no. 6, pp. 794–802, 2005. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/21/6/794

[132] R. Nobrega, J. Barbosa, and A. Monteiro, “Biogrid application toolkit: a grid-based problem solving environment tool for biomedical data analysis,” in *VECPAR-7th International Meeting on High Performance Computing for Computational Science*, 2006.

[133] M. Cannataro, C. Comito, F. Schiavo, and P. Veltri, “PROTEUS: a Grid Based Problem Solving Environment for Bionformatics,” 2004. [Online]. Available: citeseer.ist.psu.edu/660273.html

[134] C.-H. Sun, B.-J. Kim, G.-S. Yi, and H. Park, “A model of problem solving environment for integrated bioinformatics solution on grid by using condor,” in *Grid and Cooperative Computing âĂŞ GCC 2004, LNCS 3251/2004*. Springer Berlin / Heidelberg, 2004, pp. 935–938.

[135] F. Tang, C. L. Chua, L.-Y. Ho, Y. P. Lim, P. Issac, and A. Krishnan, “Wildfire: distributed, grid-enabled workflow construction and execution,” *BMC Bioinformatics*, vol. 6, no. 1, p. 69, 2005. [Online]. Available: http://www.biomedcentral.com/1471-2105/6/69

[136] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn, “Taverna: a tool for building and running workflows of services,” *Nucl. Acids Res.*, vol. 34, no. 2,

pp. 729–732, 2006. [Online]. Available: http://nar.oxfordjournals.org/cgi/content/abstract/34/suppl_2/W729

[137] G. Aloisio, M. Cafaro, S. Fiore, and M. Mirto, "A workflow management system for bioinformatics grid," in *Proceedings of the Network Tools and Applications in Biology (NETTAB) Workshops*, 2005.

[138] H. Dail, H. Casanova, and F. Berman, "A decoupled scheduling approach for the grads program development environment," in *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2002, pp. 1–14.

[139] T. Goodale, G. Allen, G. Lanfermann, J. MassÃş, T. Radke7, E. Seidel7, and J. Shalf9, "The cactus framework and toolkit: Design and applications," in *High Performance Computing for Computational Science - VECPAR 2002, LNCS 2565/2003*. Springer Berlin / Heidelberg, 2003, pp. 15–36.

[140] S. Moskwa, "A Grid-enabled Bioinformatics Applications Framework (GEBAF)," Master's thesis, School of Computer Science, The University of Adelaide, Australia, November 2005.

[141] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. Gilbert, I. Korf, H. Lapp, H. Lehvaslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, and E. Birney, "The bioperl toolkit: Perl modules for the life sciences," *Genome Res.*, vol. 12, no. 10, pp. 1611–1618, October 2002. [Online]. Available: http://dx.doi.org/10.1101/gr.361602

[142] D. Abramson, R. Buyya, and J. Giddy, "A computational economy for grid computing and its implementation in the nimrod-g resource broker," *Future Gener. Comput. Syst.*, vol. 18, no. 8, pp. 1061–1074, 2002.

[143] A. YarKhan and J. J. Dongarra, "Biological sequence alignment on the computational grid using the grads framework," *Future Gener. Comput. Syst.*, vol. 21, no. 6, pp. 980–986, 2005.

[144] K. Krauter and M. Maheswaran, "Architecture for a grid operating system," in *GRID '00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. London, UK: Springer-Verlag, 2000, pp. 65–76.

[145] P. Padala and J. N. Wilson, "Gridos: Operating system services for grid architectures," in *In Proceedings of International Conference On High Performance Computing, LNCS 2913/2003*, 2003, pp. 353–362.

[146] E.-N. Huh and Y. Mun, "Performance analysis for real-time grid systems on cots operating systems," in *Computational Science - ICCS 2003, LNCS 2660/2003*. Springer-Verlag, 2003, pp. 715–.

[147] D. Talia, "Towards grid operating systems: from glinux to a gvm," in *Proc. Workshop on Network Centric Operating Systems, Brussels*, 2005, pp. 16–17.

[148] A. Grimshaw and A. Natrajan, "Legion: Lessons Learned Building a Grid Operating System," *Proceedings of the IEEE*, vol. 93, no. 3, pp. 589–603, 2005.

[149] I. E, R. B, and D. D, "Job management systems analysis," in *6th CARNET Users Conference: Hungary*, 2004.

[150] E.-G. Talbi and A. Y. Zomaya, Eds. Wiley, 2008.

[151] Y. Sun, S. Zhao, H. Yu, G. Gao, and J. Luo, "ABCGrid: Application for Bioinformatics Computing Grid," *Bioinformatics*, vol. 23, no. 9, pp. 1175–1177, 2007.

[152] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, January 2008.

[153] H. Stockinger, M. Pagni, L. Cerutti, and L. Falquet, "Grid approach to embarrassingly parallel cpu-intensive bioinformatics problems," in *e-Science and Grid Computing, 2006. e-Science '06. Second IEEE International Conference on*, 2006, pp. 58 –58.

[154] G. Bosilca, A. Bouteiller, F. Cappello, S. Djilali, G. Fedak, C. Germain, T. Herault, P. Lemarinier, O. Lodygensky, F. Magniette, V. Neri, and A. Selikhov, "Mpich-v: Toward a scalable fault tolerant mpi for volatile nodes," in *Supercomputing, ACM/IEEE 2002 Conference*, 2002, p. 29.

[155] R. Batchu, Y. S. Dandass, A. Skjellum, and M. Beddhu, "Mpi/ft: A model-based approach to low-overhead fault tolerant message-passing middleware," *Cluster Computing*, vol. 7, pp. 303–315, 2004, 10.1023/B:CLUS.0000039491.64560.8a. [Online]. Available: http://dx.doi.org/10.1023/B:CLUS.0000039491.64560.8a

[156] S. Sriram, S. Jeffrey M., B. Brian, L. Andrew, D. Jason, H. Paul, and R. Eric, "The lam/mpi checkpoint/restart framework: System-initiated checkpointing," in *LACSI Symposium*, 2003.

[157] A. J. Chakravarti, *The Organic Grid: Self-Organizing Computational Biology on Desktop Grids*. John Wiley & Sons, Inc., 2006.

[158] M. Tsiknakis, M. Brochhausen, J. Nabrzyski, J. Pucacki, S. Sfakianakis, G. Potamias, C. Desmedt, and D. Kafetzopoulos, “A Semantic Grid Infrastructure Enabling Integrated Access and Analysis of Multilevel Biomedical Data in Support of Postgenomic Clinical Trials on Cancer,” *Information Technology in Biomedicine, IEEE Transactions on*, vol. 12, no. 2, pp. 205–217, 2008.

[159] J. Pukacki, M. Kosiedowski, R. Mikolajczak, M. Adamski, P. Grabowski, M. Jankowski, M. Kupczyk, C. Mazurek, N. Meyer, J. Nabrzyski, T. Piontek, M. Russell, M. Stroinski, and M. Wolski, “Programming Grid Applications with Gridge,” *Computational Methods in Life Sciences and Technology*, vol. 12, no. 1, pp. 10–20, 2006.

[160] I. Silman, *Structural Proteomics and Its Impact on the Life Sciences*, 1st ed. World Scientific, 2008.

[161] V. Pande, I. Baker, J. Chapman, S. Elmer, S. Khaliq, S. Larson, Y. Rhee, M. Shirts, C. Snow, E. Sorin, and B. Zagrovic, “Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing.” *Biopolymers*, vol. 68, no. 1, pp. 91–109, 2003.

[162] C. An, “Predictor@home: A "protein structure prediction supercomputer’ based on global computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 8, pp. 786–796, 2006, member-Taufer, Michela and Member-Kerstens, Andreas and Member-Brooks III, Charles L.

[163] C. Masters and K. Beyreuther, “Spongiform encephalopathies: tracking turncoat prion proteins,” *Nature*, vol. 388, pp. 228–229, 1997.

[164] R. Carrell and B. Gooptu, “Conformational changes and disease - serpins, prions and

alzheimer's," *Current Opinion in Structural Biology*, vol. 8, pp. 799–809(11), December 1998.

[165] S. Radford and C. Dobson, "From computer simulations to human disease: emerging themes in protein folding," *Cell*, vol. 97, pp. 291–298, 1999.

[166] S. Murdock, K. T. K, and M. Ng, "Biosimgrid: A distributed environment for archiving and the analysis of biomolecular simulations," *Abstracts of Papers of the American Chemical Society*, vol. 230, pp. U1309–U1310, 2005.

[167] A. Natrajan, M. Crowley, N. Wilkins-Diehr, M. A. Humphrey, A. D. Fox, A. S. Grimshaw, and C. L. Brooks, III, "Studying protein folding on the grid: experiences using CHARMM on NPACI resources under legion: Research articles," *Concurr. Comput. : Pract. Exper.*, vol. 16, no. 4, pp. 385–397, 2004.

[168] B. Uk, M. Taufer, T. Stricker, G. Settanni, and A. Cavalli, "Implementation and characterization of protein folding on a desktop computational grid " is charmm a suitable candidate for the united devices metaprocessor?" in *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2003, p. 50.2.

[169] Y. Zhang and J. Skolnick, "The protein structure prediction problem could be solved using the current PDB library," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 4, pp. 1029–1034, 2005.

[170] C. George, F. Yoshimi, and T. Shoji, " A reversible fragment assembly method for de novo

protein structure prediction ," *The Journal of chemical physics*, vol. 119, no. 13, pp. 6895–6903, 2003.

[171] A. Birnbaum, J. Hayes, W. W. Li, M. A. Miller, P. W. Arzberger, P. E. Bourne, and H. Casanova, "Grid workflow software for high-throughput proteome annotation pipeline," in *Grid Computing in Life Science, LNCS 3370/2005*. Springer Berlin / Heidelberg, 2005, pp. 68–81.

[172] J. Herrera, E. Huedo, R. Montero, and I. Llorente, "Benchmarking of a joint irisgrid/egee testbed with a bioinformatics application," *Scalable Computing: Practice and Experience*, vol. 7, no. 2, pp. 25–32, 2006.

[173] E. Huedo, R. S. Montero, and I. M. Llorente, "Adaptive grid scheduling of a high-throughput bioinformatics application," in *Parallel Processing and Applied Mathematics, LNCS 3019/2004*. Springer Berlin / Heidelberg, 2004, pp. 840–847.

[174] Y. Tanimura, K. Aoi, T. Hiroyasu, M. Miki, Y. Okamoto, and J. Dongarra, "Implementation of protein tertiary structure prediction system with netsolve," in *HPCASIA '04: Proceedings of the High Performance Computing and Grid in Asia Pacific Region, Seventh International Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 320–327.

[175] R. Kolodny and N. Linial, "Approximate protein structural alignment in polynomial time," *Protein Sci.*, vol. 101, pp. 12 201–12 206, 2004.

[176] C. Ferrari, C. Guerra, and G. Zanottib, "A grid-aware approach to protein structure comparison," *J. Parallel Distrib. Comput.*, vol. 63, pp. 728–737, 2003.

[177] S.-J. Park and M. Yamamura, "Frog (fitted rotation and orientation of protein structure by means of real-coded genetic algorithm) : Asynchronous parallelizing for protein structure-based comparison on the basis of geometrical similarity," *Genome Informatics*, vol. 13, pp. 344–345, 2002.

[178] S. Park and M. Yamamura, "Two-layer protein structure comparison," in *ICTAI '03: Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*. Washington, DC, USA: IEEE Computer Society, 2003, p. 435.

[179] M. Cannataro, M. Comin, C. Ferrari, C. Guerra, A. Guzzo, and P. Veltri, "Modeling a protein structure comparison application on the grid using proteus," in *Scientific Applications of Grid computing (SAG2004)*, ser. LNCS 3458. Berlin:Springer, 2004, pp. 75–85.

[180] R. R. Sokal and C. D. Michener, "A statistical method for evaluating systematic relationships," *Univ Kansas Sci Bull*, vol. 38, pp. 1409–1438, 1958.

[181] J. Ward, Joe H., "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 236–244, 1963. [Online]. Available: http://www.jstor.org/stable/2282967

[182] A. Kocsor, A. Kertesz-Farkas, L. Kajan, and S. Pongor, "Application of compression-based distance measures to protein sequence classification: a methodological study," *Bioinformatics*, vol. 22, pp. 407–412, 2006.

[183] D. Pelta, N. Krasnogor, C. Bousono-Calzon, J. Verdegay, J. Hirst, and E. Burke, "A fuzzy sets based generalization of contact maps for the overlap of protein structures," *Journal of*

Fuzzy Sets and Systems, vol. 152, no. 2, pp. 103–123, 2005. [Online]. Available: http://dx.doi.org/10.1016/j.fss.2004.10.017

[184] C. Bousono-Calzon, A. Oropesa-Garcia, and N. Krasnogor, "Selecting protein fuzzy contact maps through information and structure measures," in *Fourth Conference of the European Society for Fuzzy Logic and Technology and 11 Recontres Francophones sur la Logique Floue et ses Applications*. Barcelona, Spain: EUSFLAT-LFA, September 2005, pp. 1106–1111. [Online]. Available: http://www.cs.nott.ac.uk/~nxk/PAPERS/s112-01.pdf

[185] D. Pelta, J. Gonzales, and N. Krasnogor, "Protein structure comparison through fuzzy contact maps and the universal similarity metric," in *Fourth Conference of the European Society for Fuzzy Logic and Technology and 11 Recontres Francophones sur la Logique Floue et ses Applications*. Barcelona, Spain: EUSFLAT-LFA, September 2005, pp. 1124–1129. [Online]. Available: http://www.cs.nott.ac.uk/~nxk/PAPERS/s112-04.pdf

[186] L. Silva and R. Buyya, *Parallel programming models and paradigms*. NJ, USA: Prentice Hall PTR, 1999, pp. 27–42.

[187] I. Foster, "Parallel computers and computation," in *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, 1995.

[188] W. D. Gropp and E. Lusk, *User's Guide for* `mpich`*, a Portable Implementation of MPI*, Mathematics and Computer Science Division, Argonne National Laboratory, 1996, aNL-96/6.

[189] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the MPI message passing interface standard," *Parallel Computing*, vol. 22, no. 6, pp. 789–828, 1996.

[190] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, 2004, pp. 97–104.

[191] 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*, vol. 63, no. 5, pp. 551–563, 2003.

[192] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. Toonen, "Distributed mpi cross-site run performance using mpig," in *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*. New York, NY, USA: ACM, 2008, pp. 229–230.

[193] K. M. Chandy and C. Kesselman, "Cc++: A declarative concurrent object oriented programming notation," Caltech Computer Science Technical Reports [http://caltechcstr.library.caltech.edu/perl/oai2] (United States), Tech. Rep., 1993. [Online]. Available: http://resolver.caltech.edu/CaltechCSTR:1993.cs-tr-92-01

[194] F. Bodin, P. Beckman, D. Gannon, S. Narayana, and S. X. Yang, "Distributed pc++ basic ideas for an object parallel language," *Scientific Programming*, vol. 2, no. 3, pp. 7–22, 1993.

[195] R. Chandra, A. Gupta, and J. Hennessy, "Cool: An object-based language for parallel programming," *Computer*, vol. 27, no. 8, pp. 14–26, 1994.

[196] A. S. Grimshaw, “An introduction to parallel object-oriented programming with mentat,” University of Virginia, Charlottesville, VA, USA, Tech. Rep., 1991.

[197] E. Johnson, D. Gannon, and P. Beckman, “Hpc++: Experiments with the parallel standard template library,” in *In Proceedings of the 11th International Conference on Supercomputing (ICS-97*. ACM Press, 1997, pp. 124–131.

[198] I. T. Foster and K. M. Chandy, “Fortran m: A language for modular parallel programming,” *Journal of Parallel and Distributed Computing*, vol. 26, 1992.

[199] I. Foster, R. Olson, and S. Tuecke, “Programming in fortran m,” Technical Report ANL-93/26, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., USA, Tech. Rep., 1993.

[200] D. C. DiNucci, *Alliant FX/8*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1987, pp. 27–42.

[201] C. Koelbel, D. Loveman, R. Schreiber, G. Steele, and M. Zosel, *The High Performance Fortran Handbook*. MIT Press, 1994.

[202] G. W. Sabot, “Optimized cm fortran compiler for the connection machine computer,” in *In Proceedings of Hawaii International Conference on System Sciences*. IEEE Computer Society, 1992, pp. 161–172.

[203] C. Koelbel, U. Kremer, C. wen Tseng, M.-Y. Wu, G. Fox, G. Fox, S. Hiranandani, S. Hiranandani, K. Kennedy, and K. Kennedy, “Fortran d language specification,” Technical Report TR90-141, Dept. of Computer Science, Rice University, Tech. Rep., 1990.

[204] M. P. I. Forum, “MPI: A message passing interface,” in *In Proc. Supercomputing ’93*. IEEE Computer Society, 1993, pp. 878–883.

[205] R. Butler and E. Lusk, “Monitors, message, and clusters: The p4 parallel programming system,” *Parallel Computing*, vol. 20, pp. 547–564, 1994.

[206] A. Grant and R. Dickens, “An implementation of a portable instrumented communication library using cs tools,” *Future Gener. Comput. Syst.*, vol. 9, no. 1, pp. 53–61, 1993.

[207] V. S. Sunderam, “PVM: A framework for parallel distributed computing,” *Concurrency: Practice and Experience*, vol. 2, pp. 315–339, 1990.

[208] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1999.

[209] W. Gropp, E. Lusk, and R. Thakur, *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press, 1999.

[210] A. Hori, “SCore: An integrated cluster system software package for high performance cluster computing,” *Cluster Computing, IEEE International Conference on*, vol. 0, p. 449, 2001.

[211] D. Gregor and A. Lumsdaine, “Design and implementation of a high-performance mpi for c# and the common language infrastructure,” in *PPoPP ’08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*. New York, NY, USA: ACM, 2008, pp. 133–142.

[212] “Platform mpi, http://www.platform.com/products/platform-mpi.” [Online]. Available: http://www.platform.com/Products/platform-mpi

[213] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, Budapest, Hungary, September 2004, pp. 97–104.

[214] "MacMPI, http://exodus.physics.ucla.edu/appleseed/dev/developer.html." [Online]. Available: http://exodus.physics.ucla.edu/appleseed/dev/developer.html

[215] M. Baker, B. Carpenter, and A. Shafi, "MPJ Express: Towards thread safe java hpc," in *IEEE International Conference on Cluster Computing (Cluster 2006)*, 2006.

[216] M. Stout, J. Bacardit, J. Hirst, R. Smith, and N. Krasnogor, "Prediction of topological contacts in proteins using learning classifier systems," *Journal Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 3, pp. 245–258, 2008, (for the latest version of this paper please refer to the journal website). [Online]. Available: http://www.springerlink.com/content/f437410653r73vj8/fulltext.pdf

[217] M. Stout, J. Bacardit, J. Hirst, and N. Krasnogor, "Prediction of recursive convex hull class assignment for protein residues," *Bioinformatics*, vol. 24(7), pp. 916–923, 2008. [Online]. Available: http://bioinformatics.oxfordjournals.org/cgi/content/abstract/btn050v1?ct=ct

[218] J. Bacardit, M. Stout, J. Hirst, A. Valencia, R. Smith, and N. Krasnogor, "Automated alphabet reduction for protein datasets," *BMC Bioinformatics*, vol. 10(6), 2009, (for the latest version of this paper please refer to the journal website). [Online]. Available: http://www.biomedcentral.com/1471-2105/10/6

[219] N. Krasnogor and S. Gustafson, "A study on the use of "self-generation" in memetic algorithms," *Natural Computing*, vol. 3, no. 1, pp. 53 – 76, 2004. [Online]. Available: http://www.cs.nott.ac.uk/~nxk/PAPERS/KrasnogorGustafson.pdf

[220] N. Krasnogor, "Self-generating metaheuristics in bioinformatics: The protein structure comparison case," *Genetic Programming and Evolvable Machines*, vol. 5, pp. 181–201, 2004.

[221] L. P. Chew and K. Kedem, "Finding the consensus shape for a protein family," in *Proceedings of the 18th Annual Symposium on Computational Geometry (SCG)*. New York: Springer, 2002, pp. 64–73.

[222] B. Rost and C. Sander, "Prediction of protein secondary structure at better than 70% accuracy," *J Mol Biol*, vol. 232, pp. 584–599, 1993.

[223] A. R. Kinjo, K. Horimoto, and K. Nishikawa, "Predicting absolute contact numbers of native protein structure from amino acid sequence," *Proteins Struct Funct Bioinf*, vol. 58, pp. 158–165, 2005.

[224] U. Hobohm and C. Sander, "Enlarged representative set of protein," *Protein Science*, vol. 3, pp. 522–524, 1994.

[225] A. G. Hoekstra and P. M. A. Sloot, "Introducing grid speedup g: A scalability metric for parallel applications on the grid," in *EGC*, 2005, pp. 245–254.

[226] A. W. van Halderen, B. J. Overeinder, P. M. A. Sloot, R. van Dantzig, D. H. J. Epema, and M. Livny, "Hierarchical resource management in the polder metacomputing initiative," *Parallel Computing*, vol. 24, no. 12-13, pp. 1807 – 1825, 1998.

[Online]. Available: http://www.sciencedirect.com/science/article/B6V12-3WN74RX-T/2/d72d9233c0310c7cd71df5b3c0409d7e

[227] G. Amdahl, "Validity of the single processor approach to achieving large-scale computing capabilities," in *Proceedings of AFIPS Conference (30)*, 1967, p. 483âĂŞ485.

[228] J. Bosque and L. Perez, "Theoretical scalability analysis for heterogeneous clusters," *Cluster Computing and the Grid, IEEE International Symposium on*, vol. 0, pp. 285–292, 2004.

[229] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, p. 403âĂŞ410, 1990.

[230] H. Lin, P. Balaji, R. Poole, C. Sosa, X. Ma, and W. chun Feng, "Massively parallel genomic sequence search on the blue gene/p architecture," in *SC '08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. Piscataway, NJ, USA: IEEE Press, 2008, pp. 1–11.

[231] J. Archuleta, F. Wu-chun, and E. Tilevich, "A pluggable framework for parallel pairwise sequence search," in *29th Annual International Conference of the IEEE*, 2007, pp. 127–130.

[232] C. Oehmen and J. Nieplocha, "ScalaBLAST: A scalable implementation of blast for high-performance data-intensive bioinformatics analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, pp. 740– 749, 2006.

[233] A. Krishnan, "GridBLAST: a globus-based high-throughput implementation of BLAST in a grid computing framework," *Concurrency and Computation: Practice and Experience*, vol. 17, pp. 1607–1623, 2005.

[234] A. E. Darling, L. Carey, and W. chun Feng, “The design, implementation, and evaluation of mpiBLAST,” in *In Proceedings of ClusterWorld 2003*, 2003.

[235] R. L. D. C. Costa and S. Lifschitz, “Database allocation strategies for parallel blast evaluation on clusters,” *Distrib. Parallel Databases*, vol. 13, no. 1, pp. 99–127, 2003.

[236] R. Braun, K. Pedretti, T. Casavant, T. Scheetz, C. Birkett, and C. Roberts, “Parallelization of local BLAST service on workstation clusters,” *Future Generation Computer Systems*, vol. 17, pp. 745–754, 2001.

[237] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, “A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard,” *Parallel Computing*, vol. 22, no. 6, pp. 789–828, Sep. 1996.

[238] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler, “ GenBank ,” *Nucleic Acids Res.*, vol. 17, p. D25âĂŞD30, 2008.

[239] R. Martino, C. Johnson, E. Suh, B. Trus, and T. Yap, “Parallel computing in biomedical research,” *Science*, vol. 265, pp. 902–908, 1994.

[240] O. Trelles-Salazar, E. Zapata, and J. Carazo, “On an efficient parallelization of exhaustive sequence comparison algorithms on message passing architectures,” *Bioinformatics*, vol. 10, pp. 509–511, 1994.

[241] G. Burns, R. Daoud, and J. Vaigl, “LAM: An Open Cluster Environment for MPI,” in *Proceedings of Supercomputing Symposium*, 1994, pp. 379–386.

[242] J. M. Squyres and A. Lumsdaine, “A Component Architecture for LAM/MPI,” in *Proceed-*

ings, 10th European PVM/MPI Users' Group Meeting, ser. Lecture Notes in Computer Science, no. 2840. Venice, Italy: Springer-Verlag, 2003, pp. 379–387.

[243] S. Sistare, J. Test, and D. Plauger, "An architecture for integrated resource management of mpi jobs," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER'02)*. IEEE Computer Society, 2002, pp. 370–377.

[244] "Tight integration of the mpich2 library into sge." [Online]. Available: http://gridengine.sunsource.net/howto/mpich2-integration/mpich2-integration.html

[245] I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, S. Tuecke, and N. K. E. Lusk, "Wide-Area implementation of the message passing interface," *Parallel Computing*, vol. 24, pp. 1735–1749, 1998.

[246] J. MacLaren, M. M. Keown, and S. Pickles, "Co-allocation fault tolerance and grid computing," in *UK e-Science AHM*, 2006.

[247] A. A. Shah, G. Folino, and N. Krasnogor, "Towards high-throughput, multi-criteria protein structure comparison and analysis," *IEEE Transactions on NanoBioscience*, vol. 9, pp. 1–12, 2010.

[248] 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.*, vol. 34, no. 4, pp. 34–41, 2005.

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

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

[251] G. Chittibabu, R. P. Lipika, and N. S. Ilya, “Dmaps: a database of multiple alignments for protein structures,” *Nucleic Acids Res.*, vol. 34, p. D273âĂŞD276, 2006.

[252] I. Shindyalov and P. Bourne, “A database and tool for 3-d protein structure comparison and alignment,” *Nucleic Acids Res.*, vol. 29, pp. 228–229, 2001.

[253] “Hierarchical data format.” [Online]. Available: http://www.hdfgroup.org/HDF5/

[254] D. Wells, E. Greisen, and R. Harten, “FITS: A flexible image transport system,” *Astronomy and Astrophysics Supplement Series*, vol. 44, pp. 363–370, 1981.

[255] 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.

[256] 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.

[257] M. A. Koch and H. Waldmann, “Protein structure similarity clustering and natural product structure as guiding principles in drug discovery,” *Drug Discovery Today*, vol. 10, no. 7, pp. 471 – 483, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/B6T64-4FVHYS2-9/2/5ab89893bf679be7430b04309872f612

[258] F. J. Dekker, M. A. Koch, and H. Waldmann, “Protein structure similarity clustering (PSSC)

and natural product structure as inspiration sources for drug development and chemical genomics," *Current Opinion in Chemical Biology*, vol. 9, no. 3, pp. 232 – 239, 2005, combinatorial chemistry. [Online]. Available: http://www.sciencedirect.com/science/article/B6VRX-4FWT431-1/2/3ef564daf2fc9a09a67dfca3c75423a4

[259] B. D. Charette, R. G. MacDonald, S. Wetzel, D. B. Berkowitz, and H. Waldmann, "Protein Structure Similarity Clustering: dynamic treatment of PDB structures facilitates clustering," *Angewandte Chemie International Edition*, vol. 45, no. 46, pp. 7766–7770, 2006.

[260] C. Levasseur and F.-J. Lapointe, "Total evidence, average consensus and matrix representation with parsimony: What a difference distances make," *Evol Bioinf Online*, vol. 2, pp. 249–253, 2006.

[261] F.-J. Lapointe, J. Kirsch, and J. Hutcheon, "Total evidence, consensus, and bat phylogeny: A distance-based approach," *Mol Phylogenet Evol*, vol. 11, no. 1, pp. 55–66, 1999.

[262] V. Di Gesù, "Data analysis and bioinformatics," in *PReMI'07: Proceedings of the 2nd international conference on Pattern recognition and machine intelligence*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 373–388.

[263] "Clustering calculator." [Online]. Available: http://www2.biology.ualberta.ca/jbrzusto/cluster.php

[264] J. Felsenstein, "Phylip - phylogeny inference package (version 3.2)," *Cladistics*, vol. 5, pp. 164–166, 1989.

[265] J. Bingham and S. Sudarsanam, "Visualizing large hierarchical clusters in hyperbolic space," *Bioinformatics*, vol. 16, pp. 660–661, 2000.

[266] D. Huson, D. Richter, C. Rausch, T. Dezulian, M. Franz, and R. Rupp, "Dendroscope: An interactive viewer for large phylogenetic trees," *BMC Bioinformatics*, vol. 8, no. 1, p. 460, 2007. [Online]. Available: http://www.biomedcentral.com/1471-2105/8/460

[267] D. Eernisse and A. Kluge, "Taxonomic congruence versus total evidence, and amniote phylogeny inferred from fossils, molecules, and morphology," *Mol Biol Evol*, vol. 10, pp. 1170–1195, 1993.

[268] R. Carnap, *Logical foundations of probability*. University of Chicago Press, 1950.

[269] A. G. Kluge, "A concern for evidence and a phylogenetic hypothesis of relationships among epicrates (boidae, serpentes)," *Systematic Zoology*, vol. 38, no. 1, pp. 7–25, 1989. [Online]. Available: http://www.jstor.org/stable/2992432

[270] K. R. Popper, *Logik der Forschung*. Mohr Siebeck, 1934.

[271] C. J. Creevey and J. O. McInerney, "Trees from Trees: Construction of phylogenetic supertrees using Clann," in *Bioinformatics for DNA Sequence Analysis*, ser. Methods in Molecular Biology, D. Posada, Ed. Humana Press, 2009, vol. 537, pp. 139–161.

[272] "Mirror of the Protein Kinase Resourse (PKR)."

[273] C. Smith, I. Shindyalov, V. S., M. Gribskov, S. Taylor, L. Ten Eyok, and B. P., "The protein kinase resource," *Trends Biochem Sci*, vol. 11, pp. 444–446, 1997.

[274] C. Smith, "The protein kinase resource and other bioinformation resources," *Prog Biophys Mol Biol*, vol. 71, pp. 525–533, 1999.

[275] C. Petretti and C. Prigent, “The protein kinase resource: everything you always wanted o know about protein kinases but were afraid to ask,” *Biol Cell*, vol. 97, pp. 113–118, 2005.

[276] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. Toonen, “Distributed mpi cross-site run performance using mpig.” in *HPDC*, M. Parashar, K. Schwan, J. B. Weissman, and D. Laforenza, Eds. ACM, 2008, pp. 229–230. [Online]. Available: http://dblp.uni-trier.de/db/conf/hpdc/hpdc2008.html#ManosMKCKT08

Appendices

Appendices

.1 List of Protein Structure Comparison Methods

Taken from: http://en.wikipedia.org/wiki/Structural_alignment_software

NAME	Description
MAMMOTH	MAtching Molecular Models Obtained from Theory
CE/CE-MC	Combinatorial Extension – Monte Carlo
DaliLite	Distance Matrix Alignment
VAST	Vector Alignment Search Tool
PrISM	Protein Informatics Systems for Modeling
SSAP	Sequential Structure Alignment Program
SARF2	Spatial ARrangements of Backbone Fragments
STAMP	STructural Alignment of Multiple Proteins
MASS	Multiple Alignment by Secondary Structure
SCALI	Structural Core ALIgnment of proteins
SSM	Secondary Structure Matching
SHEBA	Structural Homology by Environment-Based Alignment
LGA	Local-Global Alignment
POSA	Partial Order Structure Alignment
PyMOL	"super" command does sequence-independent 3D alignment
FATCAT	Flexible Structure AlignmenT by Chaining Aligned Fragment Pairs Allowing Twists

Matras	MArkovian TRAnsition of protein Structure
MAMMOTH-mult	MAMMOTH-based multiple structure alignment
PRIDE	PRobaility of IDEntity
FAST	FAST Alignment and Search Tool
C-BOP	Coordinate-Based Organization of Proteins
ProFit	Protein least-squares Fitting
TOPOFIT	Alignment as a superimposition of common volumes ...
MUSTANG	MUltiple STructural AligNment AlGorithm
URMS	Unit-vector RMSD
LOCK	Hierarchical protein structure superposition
LOCK 2	Improvements over LOCK
CBA	Consistency Based Alignment
TetraDA	Tetrahedral Decomposition Alignment
STRAP	STRucture based Alignment Program
LOVOALIGN	Low Order Value Optimization methods for Structural Alignment
GANGSTA	Genetic Algorithm for Non-sequential, Gapped protein STructure
GANGSTA+	Combinatorial algorithm for nonsequential and gapped ...
TM-align	TM-score based protein structure alignment
MatAlign	Protein Structure Comparison by Matrix Alignment
Vorolign	Fast structure alignment using Voronoi contacts
EXPRESSO	Fast Multiple Structural Alignment using T-Coffee and Sap
YAKUSA	Internal Coordinates and BLAST type algorithm

BLOMAPS	Conformation-based alphabet alignments
CLEPAPS	Conformation-based alphabet alignments
TALI	Torsion Angle ALIgnment
FlexProt	Flexible Alignment of Protein Structures
MultiProt	Multiple Alignment of Protein Structures
CTSS	Protein Structure Alignment Using Local Geometrical Features
Matt	Multiple Alignment with Translations and Twists
TopMatch	Protein structure alignment and visualization of structural
SSGS	Secondary Structure Guided Superimposition
Matchprot	Comparison of protein structures by growing neighborhood
UCSF Chimera	see MatchMaker tool and "matchmaker" command
FLASH	Fast aLignment Algorithm for finding Structural Homology
RAPIDO	Rapid Alignment of Protein structures with Domain
ComSubstruct	Structural Alignment based on Differential Geometrical Encoding
ProCKSI	Protein (Structure) Comparison, Knowledge, Similarity and Info
SARST	Structure similarity search Aided by Ramachandran Sequential Trans.
Fr-TM-align	Fragment-TM-score based protein structure alignment
TOPS+ COMPARISON	Comparing topological models of protein structures
TOPS++FATCAT	Flexible Structure AlignmenT by Chaining Aligned Fragment...
MolLoc	Molecular Local Surface Alignment
FASE	Flexible Alignment of Secondary Structure Elements
SABERTOOTH	Protein Structural Alignment based on a vectorial Structure ...

.2 Software Related Technical Probelms and Solutions

Sun Grid Engine 6.1u2 related issues

Problem 1: "Unsupported local hostname"

Run ’ifconfig -a’ and obtain the IP address of the interface which is

used to connect to exec hosts. Then edit the file ’/etc/hosts’ and make an entry

127.0.0.1 localhost

128.243.18.20 taramel taramel.cs.nott.ac.uk

Problem 2:

"mount: mount to NFS server ’taramel’ failed: System Error: No route to host"

It indicates there is some thing wrong with the Firewall.

Add the following rules to /etc/sysconfig/iptables on master node:

-A RH-Firewall-1-INPUT -s 128.243.24.xx -j ACCEPT

-A RH-Firewall-1-INPUT -s 128.243.24.xx -j ACCEPT

-A RH-Firewall-1-INPUT -s 128.243.24.xxx -j ACCEPT

Restart the firewall (/etc/init.d/iptables restart).

Problem 3: Cannot contact qmaster. The command failed:

Edit /etc/services on each execution host for entries sge_qmater 536/tcp

and sge_execd 537/tcp and it should work.

Problem 4: "-bash: qconf: command not found"

run ./usr/SGE6/defualt/common/settings.sh or

cp /usr/SGE6/defualt/common/settings.sh /etc/profile.d/gridengine.sh

MPICH2-1.0.7rc2 + SGE 6.1u2 on CentOS-5

Problem 5: "Connect to address 128.243.: Connection refused"

Check if rsh server is installed. Otherwise install it:

yum install rsh-server

and restart xinetd

Also, the integration of MPICH2 with SGE requires passwordless SSH/RSH on all nodes:

1. local$ $ssh - keygen - tdsa$

2. local$ $scp /.ssh/id_dsa.pubremote$

3. local$ *sshusername@remote*

4. remote$ $cat /id_dsa.pub >> /.ssh/authorized_keys$

5. remote$ *chmod*644 $/.ssh/authorized_keys$ 6. remote$ *exit*

7. local$ *sshusername@remote*

use: grep rsh /var/log/messages

In /etc/xinetd.d/rsh change "disable" to

disable = no

restart xinetd with

/etc/init.d/xinetd reload

look carefully at the contents of

/.rhosts, /root/.rhosts and /etc/hosts.equiv

Problem 6: smpd -s: Floating point exception

Check if all machines have same version of OS. Otherwise

instal MPICH2 locally on each machine.

Problem 7: "export: Command not found"

Specify an interpreting shell in the job script with syntax such as:

#$ – *S*/*bin*/*bash*

Globus related issues

Problem 7:

GRAM Job submission failed because data transfer to the server failed (error code 10)

Install the advisory from:

`www.mcs.anl.gov/~bester/patches/globus_gram_protocol-7.5.tar.gz`

Run *gpt* – *buildglobus*$_g$*ram*$_p$*rotocol* – 7.5.*tar.gzgcc*32*dbggcc*32*dbgpthr*

Problems on National Grid Service

Problem 8: "mpicc with hdf5 fails"

use: module load 64-bit gnu-compilers mpi hdf5

and then use proper wrappers for mpicc, gcc and h5cc

MPICC wrapping H5CC wrapping GCC

mpicc -cc=h5cc parallel.c procksi_Alone.c -o par_procksi_Alone

-DDEBUG -DTIMING -DPARALLEL

OR

HD55 wrapping MPICC wrapping GCC

env HDF5_CC=mpicc HDF5_CLINKER=mpicc h5cc parallel.c procksi_Alone.c

-o par_procksi_Alone -DDEBUG -DTIMING -DPARALLEL

Problem 9: "Pro*C code fails to compile"

Use proper scripts for compilation as:

[*ngs*0933@*ngs*]$*moduleloadoracle*

[*ngs*0933@*ngs*]$*gcc* − *I*/*apps*/*oracle*/10.2.0.3/*client*/*precomp*/*public*/

−*L*/*apps*/*oracle*/10.2.0.3/*client*/*lib* − *lsqlplus*$ − *osample*1.*newsample*1 − *pro.c*

[*ngs*0933@*ngs*]$*readel f* − *hsample*1.*new*

[*ngs*0933@*ngs*]$./*sample*1.*new*

Connected to the database as user: ngsdb0053