On the Scalability of Multi-Criteria Protein Structure Comparison in the Grid

Gianluigi Folino[1], Azhar Ali Shah[2], and Natalio Krasnogor[2,⋆]

[1] Institute of High Performance Computing and Networking,
Italian National Research Council, Cosenza 87036, Italy
folino@icar.cnr.it
[2] School of Computer Science, University of Nottingham, NG81BB, UK
{azhar.shah,natalio.krasnogor}@nottingham.ac.uk

Abstract. Multi-Criteria Protein Structure Comparison (MC-PSC) is one of the Grand Challenge Applications (GCAs) in the field of structural proteomics. The solution of the MC-PSC grand challenge requires the use of distributed algorithms, architectures and environments. This paper is aimed at the analysis of the scalability of our newly developed distributed algorithm for MC-PSC in the grid environment. The scalability in the grid environment indicates the capacity of the distributed algorithm to effectively utilize an increasing number of processors across multiple sites. The results of the experiments conducted on the UK's National Grid Service (NGS) infrastructure are reported in terms of speedup, efficiency and cross-site communication overhead.

Keywords: Protein Structure Comparison, Grid, Scalability, Bioinformatics.

1 Introduction

The theoretical analysis of the scalability of the *'computation-centric'* parallel applications on the grid appears in [4] 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) [8] (please see section 3 for the details of the NGS infrastructure). The HCG model is based on the concept of *'Hierarchical Resource Manager'* [3] 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 connected using the same type of network. 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

⋆ Corresponding author.

M.R. Guarracino et al. (Eds.): Euro-Par 2010 Workshops, LNCS 6586, pp. 193–200, 2011.

overhead, i.e. the communication overhead among C CE's $Q_1(W, C)$ and the communication overhead among p processors of each CE $Q_2(W/C, p)$. With this formalism, the execution time on HCG could be defined as:

$$T_{C,p}(W) = \frac{W_p}{pC\Delta} + Q_2(W/C, p) + Q_1(W, C) \tag{1}$$

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 1 returns to the standard parallel case i.e $Q_2(W, p) = Q(W, p)$.

Equation 1 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 execution time on P processors with the sequential execution time on 1 processor. However, as suggested by [4], 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:

$$\Gamma_p^C = \frac{T_1, p(W)}{T_C, p(W)} \tag{2}$$

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 2) 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 2 we could also derive the expression for the Grid efficiency as:

$$\gamma_p^C = \frac{T_1, p(W)}{CT_C, p(W)} \tag{3}$$

where, γ_p^C is the 'Grid efficiency' and p, C, T_1 and T_C represent the same parameters as described in eq. 2.

The description of the 'Grid Efficiency' in eq. 3 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 " [1]. 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 paper performs empirical analysis of our parallel algorithm for MC-PSC as described in the following sections.

The remainder of this paper is organized as follows: section 2 describes the background related to the MC-PSC Grand Challenge, section 3 describes the

experimental setup; section 4 presents the results and discussions and finally section 5 concludes the paper.

2 The MC-PSC Grand Challenge

The problem of large scale MC-PSC could be represented as a 3D cube. The x and y axis of the cube representing the different proteins being compared, while the z axis representing different comparison methods being used such as such as the *Universal Similarity Metric* (USM), *Maximum Contact Map Overlap* (MaxCMO), *Distance Alignment Matrix* (DaliLite), *Combinatorial Extension* (CE), *Fast Alignment And Search Tool* (FAST) and TM-Align etc. 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" [11]. 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 are available in the literature [10]. 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 [2]. 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*) approach. Introduced by Ian Foster in [2], 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. Based on the philosophy of the PCAM approach, a high-throughput distributed framework for the solution of the grand challenge of MC-PSC using *Message Passing Interface* (MPI) model of parallel programming has been introduced [9]. The performance of this framework along with different load balancing strategies was evaluated on a 64-node cluster as reported in [9]. However, it was observed that for datasets having relatively large number of proteins (e.g. 1000+), even the 64-node cluster becomes a bottleneck and it takes about 11 days for the computation to complete. Hence, we tried to deploy our algorithm on the UK National Grid Service (NGS) to take advantage of greater number of cores available across multiple sites. The deployment on the NGS is reported in the next section.

3 Deployment on the NGS Infrastructure

The *National Grid Service* (NGS), provides the *eScience* infrastructure to all the UK-based scientists free of cost [8]. For our case we used the Globus-based MPIg [7] (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 and MPICH-G2), the MPIg library extends the Argonne MPICH implementation of MPI to use services provided by the Globus Toolkit 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 4x. 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) [6] 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 1, shows the overall architecture and setup of deploying the MC-PSC application on the Grid.

3.1 Dataset

The dataset used in these experiments is the one introduced by Kinjo et al [5] 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 (i.e., USM, MaxCMO, CE, DaliLite, FAST and TM-Align), 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

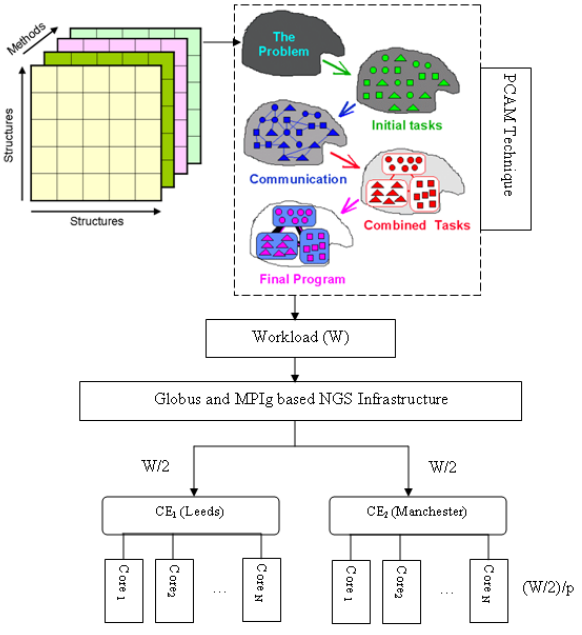


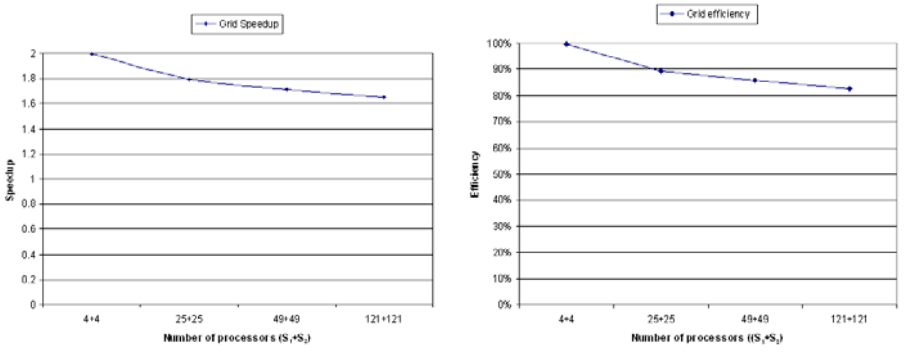
Fig. 1. Deployment of the MC-PSC application on the Grid: the application takes protein 3-D structures as input and prepares the balanced workload W to be distributed on the Grid. Half of the total workload (W/2) is assigned to each site (CE). Each site further distributes the W/2 into p number of cores.

processor and it took about 10.7 days to complete on a 64-node cluster [9]. The results on the Grid infrastructure are presented in the next section.

4 Results and Discussions

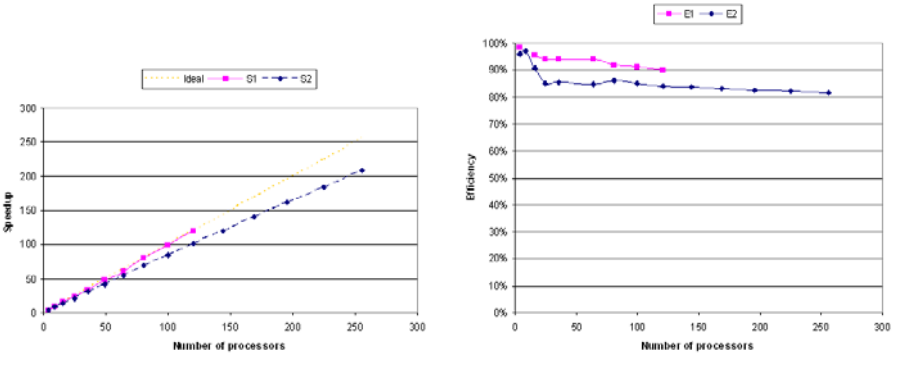
Both the single-site and cross-site experiments for MC-PSC were conducted with varying number of processors using the Kinjo et al [5] dataset. The Grid speedup and efficiency (eq. 2 and eq. 3 respectively) were calculated based on the results of these experiments and are shown in figure 2. Figure 2(a) 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 2(b).

Figure 3 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



(a) (b)

Fig. 2. Performance of the MC-PSC on the Grid: (a) Grid Speedup (eq 2); 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). (b) Grid efficiency (eq. 3); as expected the slight degradation of speedup causes the degradation in the efficiency of the system.



(a) (b)

Fig. 3. Single-Site and Cross-Site: (a) 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. (b) Efficiency; as expected the cross-site efficiency ($E2$ is slightly less as compared to the single-site efficiency ($E1$ due to extra communication overhead.

(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 3(a), 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 boast 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 3(b), 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 4 shows the cross-site communication overhead in terms of running the MC-PSC application in the Grid. It shows that, when a few processors are used, the load of the processors and the amount of data to be exchanged is high and consequently there is a considerable communication overhead. However, when we use a larger number of processors, the overhead is negligible in comparison with the computation time.

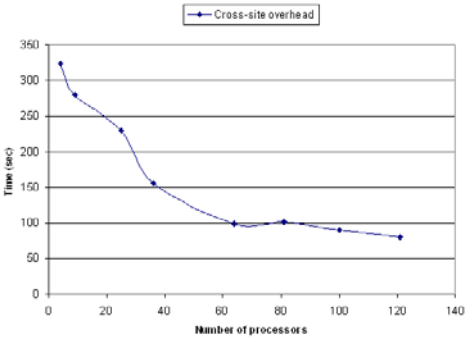


Fig. 4. Cross-site communication overhead. The graph shows that when a few processors are used the load of the processors and consequently the amount of data to be exchanged is high and consequently there is considerable communication overhead. However, when we use a larger number of processors, the overhead is negligible in comparison with the computation time.

4.1 Concluding Remarks and Future Directions

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 does not cause performance degradation. The current cross-site experiments were conducted on 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, it does not allow evaluating the performance of the application with larger datasets; hence the software developed so far could be upgraded by adding fault tolerance mechanisms 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). 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.

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

References

1. Amdahl, G.: Validity of the single processor approach to achieving large-scale computing capabilities. In: Proceedings of AFIPS Conference, (30), pp. 483–485 (1967)
2. Foster, I.: Parallel computers and computation. In: Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering (1995)
3. Halderen, A.W., Overeinder, B.J., Sloot, P.M.A., van Dantzig, R., Epema, D.H.J., Livny, M.: Hierarchical resource management in the polder metacomputing initiative. Parallel Computing 24(12-13), 1807–1825 (1998)
4. Hoekstra, A.G., Sloot, P.M.A.: Introducing grid speedup g: A scalability metric for parallel applications on the grid. In: Sloot, P.M.A., Hoekstra, A.G., Priol, T., Reinefeld, A., Bubak, M. (eds.) EGC 2005. LNCS, vol. 3470, pp. 245–254. Springer, Heidelberg (2005)
5. Kinjo, A.R., Horimoto, K., Nishikawa, K.: Predicting absolute contact numbers of native protein structure from amino acid sequence. Proteins Struct. Funct. Bioinf. 58, 158–165 (2005)
6. MacLaren, J., Keown, M.M., Pickles, S.: Co-allocation fault tolerance and grid computing. In: UK e-Science AHM (2006)
7. Manos, S., Mazzeo, M., Kenway, O., Coveney, P.V., Karonis, N.T., Toonen, B.: Distributed mpi cross-site run performance using mpig. In: HPDC 2008: Proceedings of the 17th International Symposium on High Performance Distributed Computing, pp. 229–230. ACM, New York (2008)
8. Richards, A., Sinclair, G.M.: UK National Grid Service. CRC Press, Boca Raton (2009)
9. Shah, A.A., Folino, G., Krasnogor, N.: Towards high-throughput, multi-criteria protein structure comparison and analysis. IEEE Transactions on NanoBioscience 9, 1–12 (2010)
10. Shah, A., Barthel, D., Lukasiak, P., Blacewicz, J., Krasnogor, N.: Web and grid technologies in bioinformatics, computational biology and systems biology: A review. Current Bioinformatics 3(1), 10–31 (2008)
11. Silva, L., Buyya, R.: Parallel programming models and paradigms, pp. 27–42. Prentice Hall PTR, NJ (1999)