CD-HIT Workflow Execution on Grids using Replication Heuristics*

J. L. Vázquez-Poletti E. Huedo R. S. Montero I. M. Llorente

Departamento de Arquitectura de Computadores y Automática
Facultad de Informática, Universidad Complutense de Madrid
28040 Madrid, Spain

Abstract

Grid Computing has proven to be a solution for big workflow execution, especially in Bioinformatics. However, Grid nature itself introduces overheads that make its use in many cases an unfeasible solution if considering wall-time. Different heuristics such as list scheduling, agglomeration and replication are available for optimizing workflow execution. In particular, the replication heuristics have been previously used in heterogeneous environments with good results. In this work, we analyze their use for workflow scheduling on Grid infrastructures. In particular, we study its applications to an intree workflow, generated by the distribution of the CD-HIT application. The experiments were conducted on a testbed made of resources from two different grids and results show a significant reduction of the workflow execution time.

1 Introduction

Workflow management systems and Grid Computing are providing solutions to problems proposed by Bioinformatics. Workflow management systems [22] allow the execution of complex applications than can be divided in tasks with data dependencies. Grid Computing, on the other hand, offers the applications access to a great amount of computing resources.

*This research was supported by Consejería de Educación of Comunidad de Madrid, Fondo Europeo de Desarrollo Regional (FEDER) and Fondo Social Europeo (FSE), through BioGridNet Research Program S-0505/TIC/000101, and by Ministerio de Educación y Ciencia, through research grant TIN2006-02806. Also, this work makes use of results produced by the Enabling Grids for E-sciencE project, a project co-funded by the European Commission (under contract number INFSO-RI-031688) through the Sixth Framework Programme. EGEE brings together 91 partners in 32 countries to provide a seamless Grid infrastructure available to the European research community 24 hours a day. Full information is available at http://www.eu-egee.org/.

In a previous paper [21], we considered a Bioinformatics application, *CD-HIT* (Cluster Database at High Identity with Tolerance) [11], for its porting to the Grid. This application performs protein clustering, which consists in removing redundant sequences from a protein database in order to generate a database of only the representatives. Protein clustering can be applied in many activities such as protein family classification, domain analysis, organization of large protein databases or improving database search performance. However, the Grid version of *CD-HIT* didn't provide good performance results, even if it served to bypass memory constraints and so process large data sets. This happened because the nature of the Grid (dynamism, heterogenity and high fault rate).

As optimization is needed in this workflow, described with the previous work in Section 2, we considered well known heuristics that proved to throw good results in other heterogeneous computational infrastructures. These optimization strategies are described in Section 3. However, in Section 4 we focused in the *replication* strategy for optimizing the cited workflow and then, evaluated it through experimental results in Section 5. Finally, some conclusions and future work are shown at the end of the paper.

2 The Application

The *CD-HIT* application was successfully ported to the Grid [21] using the Grid*W*ay metascheduler [9]. However, workflow management systems such as the Directed Acyclic Graph Manager (DAGMan) [18] provided by Condor, and Pegasus [4] were considered among others. In the past, the Grid*W*ay metascheduler has been previously used with good results in many research areas, including Bioinformatics. It natively handles DAG based workflows and allows advanced flow structures like loops or branches. Grid*W*ay offers an implementation of both C and JAVA bindings

of the Distributed Resource Management Application API (DRMAA), which is an Open Grid Forum[1] standard [8].

Regarding the workflow management performed by Grid*Way*, workflow specification follows an *abstract model*, and its concretion, a *dynamic scheme* [22]. Scheduling decisions are taken considering the requirements for each task and resource ranking expressions at run-time. Additionally, historical information about task execution is considered. Grid*Way* provides fault tolerance mechanisms which include trying the task execution or file transfer on the same resource in case of failure, and submitting of a failed task to an alternate resource.

The first approach to the *gridification* of this algorithm, described in a previous publication [21], was a simple case of *list scheduling* where the workflow nodes were given a certain priority and then handled to Grid*Way*. Even if complex techniques may be used for determining task priority [13], in our case, nodes from the *critical path* [15] deserve the highest priority. This priority is given by the order in which jobs are submitted to Grid*Way* at the beginning of the process. Due to dependencies among workflow nodes, not all tasks are submitted at the first moment to remote resources, as they are put *on hold*. Summarizing, priorities are established both by the order jobs are sent to schedule and by their dependencies.

Distributing the workflow nodes over the Grid has proven to bypass the memory limitations intrinsic to a single machine. Nevertheless, execution times measured on the Grid were higher than those obtained locally as Grid environment is highly dynamic, heterogeneous and faulty. In addition, the time a task waits in the remote queuing system can be the most significant part of each task's walltime, specially in infrastructures at production level. Hence the remote queue waiting time completely determines the overall efficiency obtained by the application.

The first experiments were performed with a mid-sized protein database (504,876 proteins, 435MB). However, the Spanish National Oncology Research Center (Centro Nacional de Investigaciones Oncológicas - CNIO)[2] requires the analysis of larger databases. For this purpose, the Grid approach is still valid due to the single machine restrictions mentioned above, but the model had to be revisited considering the optimization heuristics cited at the following Section.

[1] http://www.drmaa.org/

[2] http://www.cnio.es/

3 Workflow Optimization Heuristics

Among the optimization strategies that may apply to a workflow, there are different approaches that can be considered, taking into account the nature of the scheduling problem. In the first approach, called *list scheduling*, priorities are assigned to jobs either statically or dynamically [16]. In general, tasks are not scheduled regarding ulterior ones so this technique doesn't always provide an optimized solution. Algorithms pertaining to this approach are:

- Heterogeneous Earliest Finish Time (HEFT) [19], which schedules tasks minimizing their finishing time in an insertion based manner;
- Critical Path on a Processor (CPOP) [19], that detaches a machine just for critical path tasks;
- Bubble Scheduling and Allocation (BSA) [10], which firstly serializes the task graph and then inserts all the tasks to a processor;
- Dynamic Level Scheduling (DLS) [17], that delays scheduling when the given task is ready;
- Critical Nodes Parent Trees (CNPT) [5], which considers the task earliest execution time;
- Iso-Level Heterogeneous Allocation (ILHA) [14], that allocates to each processor a number of tasks proportional to its computing power.

The second approach is the *agglomeration* technique, which consists in clustering jobs in groups so communication overheads are reduced. Even if this technique is not complex in its implementation, the obtained performance by itself, is not as desiderable as expected [2].

The last approach is the *replication* strategy, that this contribution will focus on. In this technique, some jobs are replicated in order to increase the possibility of speed-up the execution due to a better resource selection. Algorithms inside this strategy are:

- Heterogeneous Critical Tasks Reverse Duplicator (HCTRD) [6], that replicates the parent-tree or some selected parents of the selected task;
- Bottom-up Top-down Duplication Heuristic (BTDH) [3], which assumes that task starting time may be reduced eventually by the replication of all the necessary task ancestors;
- Heterogeneous Critical Parents with Fast Duplicator (HCPFD) [7], that performs the replication considering the idle time left by the selected task on a given machine;

- Critical Path based Full Duplication Algorithm (CPFD) [1], which replicates all possible parents of the considered task.

A version of this last heuristic was chosen because its structure fits the studied Bioinformatics application, and its adoption is presented in the following section.

4 Applying the Replication Heuristic

In this contribution, we apply the *replication* strategy for improving the workflow's efficiency as shown at Figure 1. Again, in this technique supplementary tasks are created for given nodes of the workflow. When one of these tasks ends, the node is taken as executed and the rest of *replicated* tasks are killed. This way, the more *replicated* tasks are created, the higher is the possibility for that node to be executed shortly by reducing the effect of job failures and queue times.

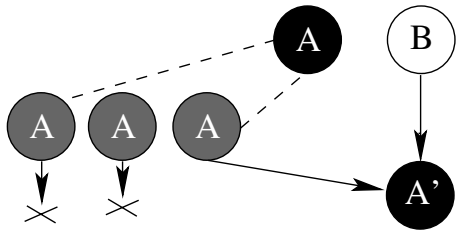

Figure 1. The *replication* technique applied to the CD-HIT algorithm.

Different variants of this strategy can be devised depending on the nodes where *replication* is applied. The most simple variant consists in the replication of all the tasks. In other variant, just the nodes from the *critical path* are replicated. In the last variant, those target nodes above a given *blocking* threshold are replicated. We define the *blocking* value as the number of nodes of the workflow which depend on the execution of that node and it is calculated by:

$$b_{i,j} = \begin{cases} ST(N-i) & \text{if} \quad i = j \\ (i-1) + ST(N-i) & \text{if} \quad i \neq j \end{cases} \quad (1)$$

being i the column and j the level where the node is located in the workflow. N is the number of workflow levels. Finally, $ST(n)$ is the *blocking subtree* generated by a node integrating the *critical path* and can be estimated as:

$$ST(n) = \frac{n(1+n)}{2} \quad (2)$$

which is the sum of terms belonging to an arithmetic progression. An example of this is shown at Figure 1. In the present contribution, the *critical path* variant was employed (Critical Path based Full Duplication Algorithm [1]), creating 3 copies per task.

However, some studies conducted about job *replication* at the same resource state that local *backfilling* mechanisms generate mostly drawbacks [12]. In our case, not all replicated tasks are necessarily sent to the same cluster, so these mechanisms may not always affect global performance. Moreover, when a resource fails, Grid$W$ay implements an exponential linear backoff strategy at resource level, henceforth resources with persistent failures are discarded. The *replication* technique, in conjunction with this scheduling policy, allows a fast *functional* resource discovery at the beginning of the workflow execution.

5 Experimental Results

The input protein database is a compound of UniProt[3] entries and sequence fragments of the Sargasso Sea meta-genome[4], all of them provided by the National Center for Biotechnology Information (NCBI)[5]. Its size is 1.7GB and it stores 4,186,284 proteins. Focusing on job execution, input file size and job number depend on the number of divisions made to the starting protein database. For this contribution, we considered 32, 40 and 48 divisions.

For processing the proposed database, two Grid infrastructures were considered: regional and worldwide, both of them detailed at Table 1. Local and regional machines are nearer to the one where the job submission takes place so they offer less latency. On the other hand, machines pertaining to the Enabling Grids for E-siencE (EGEE)[6] infrastructure are more in number and offer more throughput. But, even with busier machines, the EGEE infrastructure guarantees exclusiveness of CPU use. As coordinated harnessing of these infrastructures was retained necessary for the processing of such a big database, the use of Grid$W$ay was still considered, due to its interoperability capabilities [20]. Experiments were conducted for each database division taken into account and results were compared depending whether the *replication* technique was used or not. Tasks were launched from Universidad Complutense de Madrid (UCM), belonging to GRIDIMadrid[7], at different times on different days of the week during April

[3] http://www.ncbi.nlm.nih.gov/RefSeq/
[4] ftp://ftp.ncbi.nih.gov/genbank/wgs/
[5] http://www.ncbi.nlm.nih.gov/
[6] http://www.eu-egee.org/
[7] http://www.gridimadrid.org/

Table 1. Grid resources from the joint infrastructure employed during the experiment.

Site	Count.	Proc.	Speed	Nodes
GRIDIMadrid Resources				
UCM	ES	P4	3216	2
CIEMAT	ES	P4	2392	22
EGEE Resources (BIOMED Virtual Organization)				
BHAM-UNI	UK	PIII	800	128
BRUNEL	UK	P4	2000	5
CGG	FR	PIII	1266	56
CIEMAT	ES	PIII	1001	220
CYF-KR	PL	P4	2800	264
GRID-ACAD	BG	P4	2400	78
HELLASGRID	GR	P4	3400	356
IFCA	ES	P4	3200	96
II	MK	P4	3300	8
IMPERIAL	UK	P4	2000	188
IN2P3	FR	PIII	1001	569
INFN	IT	P4	2400	124
IPP-ACAD	BG	P4	2800	10
JET-EFDA	UK	PIII	1098	66
KELDYSH	RU	P4	3000	14
L.-HEP	UK	P4	3000	380
LIP	PT	P4	2200	52
MAN-UNI	UK	P4	2800	844
PNPI	RU	P4	3000	112
SAVBA	SK	P4	3200	41
SRCE	HR	P4	2193	16
UAM	ES	P4	2566	14
UCL	UK	P4	2800	312
UNI-LINZ	AT	P4	3014	8

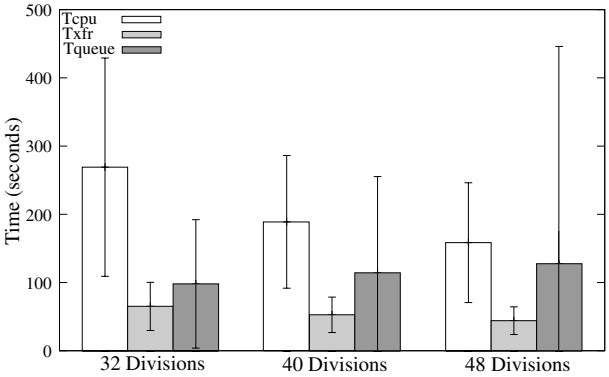

Figure 2. Consolidated CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{queue}) times for the workflow tasks and different database partition number, with and without optimization.

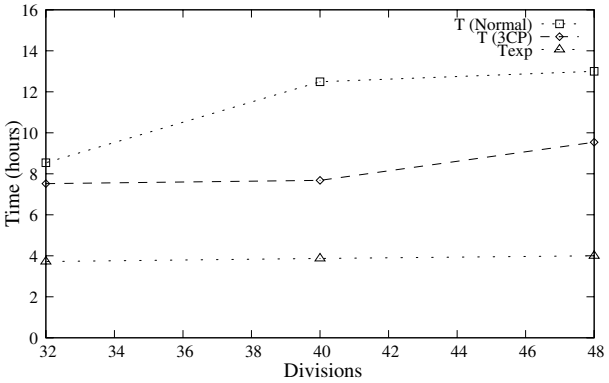

Figure 3. Workflow execution times for different number of database divisions without optimization and using the *replication* strategy (3 copies of each *critical path* task).

2007. Finally, a constrain was added to scheduling: the maximum number of tasks submitted to a site was limited to 10.

Consolidated average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{queue}) times and their standard deviations for the workflow tasks are shown in Figure 2, categorized by the different number of database partitions. Both T_{cpu} and T_{queue} present a high variability. In the case of T_{cpu}, resources with different computing capacity were available during each experiment. On the other hand, the variability of Local Resource Management Systems (LRMS) located at the resources, affected T_{queue}. T_{xfr} also presents some variability, this is due to the different network links involved.

Figure 3 shows values of different execution times: T is the experimental workflow execution time and T_{exp} is the *expected walltime*, which is explained next. The *expected walltime* (T_{exp}) does take into account only the *critical path* tasks and not considering job failures. The value of T_{exp} is the same for both cases (with and without optimization) and it is calculated as:

$$T_{exp} = N \cdot (T_{cpu} + T_{xfr} + T_{queue}), \qquad (3)$$

where N is the number of database divisions. Figure 3 shows that the obtained experimental times applying the optimization are lower than without.

Speed-up can be computed dividing the overall execution time by the workflow's *sequential time* (T_{seq}). It is possible to estimate T_{seq} with:

$$T_{seq} = N \cdot T^{A}_{cpu} + T^{B}_{cpu} \cdot \frac{N \cdot (N+1)}{2}, \qquad (4)$$

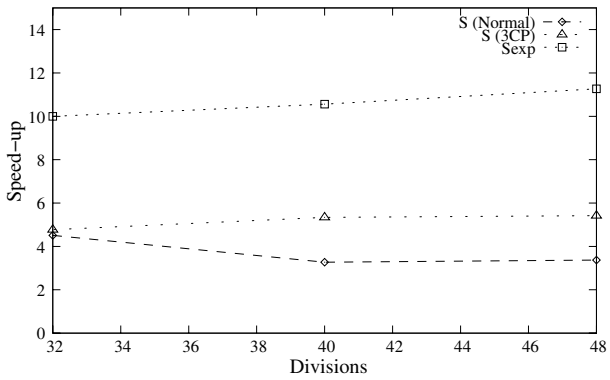

Figure 4. Speed-up of the workflow execution for different number of database divisions, with and without optimization.

where T^A_{cpu} and T^B_{cpu} are the CPU time of shaded and non-shaded tasks as shown in Figure 1. Their values correspond to the average execution times obtained during the experiments. In any case, we would like to remark that sequential processing is not possible to be done on a single CPU with the original *CD-HIT* application due to memory constraints.

Different speed-up estimations are shown at Figure 4: the *experimental speed-up* (S) and that obtained with the *expected walltime* (S_{exp}). Basically, we observe that the optimization allows to raise the level of parallelism. This doesn't happen in the non-optimized case, where S decreases with the number of database divisions. Some aspects must be considered in this study. Firstly, the maximum number of simultaneously used processors was limited to 20 due to scheduling constraints. Then, the level of parallelism of the application decreases on each level because of its in-tree shape, acquiring S_{exp} a maximum value of 12.20 with 48 database divisions.

Despite the factors explained above, the optimization represented about 50% of speed-up gain. The 20 speed-up value couldn't be reached because of the Grid's inherent nature, represented by the number of reschedules (shown at Table 2). On the other hand, the improvement was also reflected on the level completion mean times, shown at the same Table.

Even if replication heuristics provide efficiency to this workflow execution, its cost should be taken into account. Depending on the number of database divisions, we may find different average transfer and execution times associated to the two *discarded* tasks (*replicated* tasks which are killed or do finish before the framework decides to kill them). For 32 divisions, *discarded* tasks consumed an average of 45 seconds of T_{xfr} and 5 minutes 16 seconds of T_{cpu}. For 40 divisions, 30 seconds of T_{xfr} and 3 minutes 51 seconds of T_{cpu}. Finally, for 48 divisions, they consumed an average of 25 seconds of T_{xfr}, and 2 minutes 55 seconds of T_{cpu}.

Table 2. Number of jobs rescheduled in each experiment with optimization (3CP) and without (Normal) and mean times (considering T_{cpu}, T_{xfr}, T_{queue} and task reschedules) for each algorithm level completion.

Divisions	Reschedules		Mean Times	
	Norm.	3CP	Norm.	3CP
32	69	34	16.3'	14.7'
40	170	35	14.6'	11.7'
48	68	27	14.8'	12.1'

Consequently, we may define the cost of replication as the sum of the times mentioned before. The cost for 32 divisions was 8 hours 6 minutes. For 40 divisions, 7 hours 25 minutes. Finally, for 48 divisions, it was 7 hours 55 minutes. We may consider however the worst case for *discarded* tasks, which is to execute them on the slowest computing resource, employing the lowest bandwidth for data transfers. From here, an upper-bound for replication cost can be estimated. For 32 divisions, this value is 32 hours. For 40 divisions, 28 hours. At the end, for 48 divisions, it is 49 hours.

In any case, we would like to note that the additional computational effort mentioned before was performed by spare resources. Therefore, without harming the performance of other Grid applications.

6 Conclusions and Future Work

Grid's inherent nature derives in high queuing times and fault rate. This makes the Grid to be an unfeasible solution for many workflows execution. In this contribution we have reviewed three approaches to optimization techniques. Then, we focused on one of them for a specific type of workflow in order to minimize the effect of the issues addressed before.

Experimental results show that using the *replication* technique derived in a valuable speed-up. However, this speed-up was limited by different factors. Firstly and due to scheduling restrictions, the number of simultaneous running jobs was 20. Then, the algorithm's shape made the level of parallelism decrease. Finally, the Grid's nature itself derived in reschedules due to suspension timeouts and execution errors.

Studying the experimental impact of this optimization strategy is only the beginning. We intend to compare the *replication* variants and then, apply the *agglomeration* strategy. With these steps it is our idea to perform a further evaluation of optimization techniques that could be applied to this particular workflow.

7 Acknowledgments

The authors would like to thank all the institutions involved in the EGEE project and in the GRIDIMadrid infrastructure, in particular those who collaborated in the experiments.

References

[1] I. Ahmad and Y.-K. Kwok. On Exploiting Task Duplication in Parallel Program Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 9(8):872–892, 1998.

[2] R. Bajaj and D. Agrawal. Improving Scheduling of Tasks in A Heterogeneous Environment. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):107–118, 2004.

[3] Y. Chung and S. Ranka. Application and Performance Analysis of a Compile-Time Optimization Approach for List Scheduling Algorithms on Distributed-Memory Multiprocessors. In *Proc. Supercomputing '92*, pages 512–521. IEEE CS, 1992.

[4] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus: Mapping Scientific Workflows onto the Grid. In *Proc. 2nd European AcrossGrids Conference (AxGrids 2004)*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20, 2004.

[5] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Job Scheduling in Homogeneous Computing Environments. In *Proc. Intl. Conf. on Parallel Processing Workshops (ICPPW'03)*, pages 149–155. IEEE CS, 2003.

[6] T. Hagras and J. Janecek. A Near Lower-Bound Complexity Algorithm for Compile-Time Task-Scheduling in Heterogeneous Computing Systems. In *Proc. 3rd Intl. Symp. Parallel and Distributed Computing, 3rd Intl. Work. Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, pages 106–113. IEEE CS, 2004.

[7] T. Hagras and J. Janecek. A High Performance, Low Complexity Algorithm for Compile-Time Task Scheduling in Heterogeneous Systems. *J. Parallel and Distributed Computing*, 65(4):479–491, 2005.

[8] J. Herrera, E. Huedo, R. S. Montero, and I. M. Llorente. Developing Grid-Aware Applications with DRMAA on Globus-based Grids. In *Proc. 10th Intl. Conf. Parallel Processing (Euro-Par 2004)*, volume 3149 of *Lecture Notes in Computer Science*, pages 429–435, 2004.

[9] E. Huedo, R. S. Montero, and I. M. Llorente. A Framework for Adaptive Execution on Grids. *Software – Practice and Experience*, 34(7):631–651, 2004.

[10] Y.-K. Kwok and I. Ahmad. Link Contention-Constrained Scheduling and Mapping of Tasks and Messages to A Network of Heterogeneous Processors. *J. Cluster Computing*, 3(2):113–124, 2000.

[11] W. Li and A. Godzik. CD-HIT: A Fast Program for Clustering and Comparing Large Sets of Protein or Nucleotide Sequences. *Bioinformatics*, 22:1658–1659, 2006.

[12] G. Malewicz, I. Foster, A. L. Rosemberg, and M. Wilde. Benefits and Drawbacks of Redundant Batch Requests. *J. Grid Computing*, 5(2):197–212, 2007.

[13] G. Malewicz, I. Foster, A. L. Rosemberg, and M. Wilde. A Tool for Prioritizing DAGMan Jobs and Its Evaluation. *J. Grid Computing*, 5(2):197–212, 2007.

[14] B. Olivier, B. Vincent, and R. Yves. The Iso-Level Scheduling Heuristic for Heterogeneous Processors. In *Proc. 10th Euromicro Conf. Parallel, Distributed and Network-based Processing (PDP 2002)*, pages 335–350. IEEE CS, 2002.

[15] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, second edition, 2002.

[16] S. Selvakumar and C. Siva Ram Murthy. Scheduling Precedence Constrained Task Graphs with Non-Negligible Intertask Communication onto Multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 5(4):328–336, 1994.

[17] G. Sih and E. Lee. A Compiler-Time Scheduling Heuristic for Interconnection-Constrained Heterogeneous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):175–186, 1993.

[18] D. Thain, T. Tannenbaum, and M. Livny. *J. Grid Computing*, chapter Condor and the Grid, pages 299–335. 2003.

[19] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-Effective and Low-complexity Task Scheduling for Heterogeneous Computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274, 2002.

[20] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Coordinated Harnessing of the IRISGrid and EGEE Testbeds with GridWay. *J. Parallel and Distributed Computing*, 66(5):763–771, 2006.

[21] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Workflow Management in a Protein Clustering Application. In *Proc. 5th Intl. Work. Biomedical Computations on the Grid (BioGrid'07). 7th IEEE Intl. Symp. Cluster Computing and the Grid (CCGrid 2007)*, pages 679–684. IEEE CS, 2007.

[22] J. Yu and R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. *J. Grid Computing*, 3(3–4):171–200, 2005.