Workflow Management in a Protein Clustering Application*

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

*Bioinformatics is demanding more computational resources day after day. The problems proposed by this area are growing in such complexity that traditional computing systems are not able to face them. For solving complex problems which can be divided in tasks with dependencies, a workflow management system must be employed. In this paper, we introduce the use of the workflow management of the Grid*W*ay metascheduler for running a Bioinformatics application which implements a complex algorithm performing protein clustering in order to obtain non-redundant protein databases. The use of a general purpose meta-scheduling system will provide the application the fault-tolerance and advance scheduling capabilities needed to execute on a highly dynamic, heterogeneous and faulty environment. The execution results on a production Grid (the EGEE infrastructure) shows the dramatic impact of remote queue waiting times on the application performance; and the critical need of efficient re-scheduling capabilities.*

1 Introduction

As Grid Computing is consolidating and extending its application to many research areas, the complexity of tasks needed to be executed in such environments is growing. The workflow concept appears to satisfy these needs as a way to automate procedures in data transfer and job execution. Then, a workflow management system is the one which not only defines, but manages and executes workflows over a Grid [14].

In a workflow management system, the workflow should be easily designed as a dependency graph, where each node represents one or more computational tasks. Also, it must be capable to provide additional mechanisms for resource information retrieval, data transfer, task scheduling and execution, and fault tolerance. In general, the providing of these mechanisms is challenging because of the nature of the Grid itself, namely: dynamic resource availability and load, heterogeneity and a high fault rate.

The objective of this paper is to show the efficient execution of a Bioinformatics workflow by the Grid*W*ay [3]'s workflow management system, which is briefly described in Section 2. This workflow pertains to a toolset called *cd-hit* [6] which is actively used by the Spanish National Oncology Research Center (Centro Nacional de Investigaciones Oncológicas - CNIO) [1]. This toolset is composed by several sub-applications which take a protein sequence database and then eliminate redundant entries. Its algorithm is explained in Section 3.

The growing database size (increasing everyday) makes *cd-hit* infeasible to be executed on a single machine due to its memory requirements and total execution time. Also, the complexity of the algorithm implemented by its workflow is demanding the use of a high number of Grid resources for execution and data transfers. Experimental results of the execution of the *cd-hit* toolset in the EGEE testbed are discussed in Section 4.

Finally, an overview of other workflow management

*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/.

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

systems is presented in Section 5, and the conclusions can be found in Section 6.

2 GridWay's Workflow Management

Grid*W*ay is a grid metascheduler which stands on top of Globus services and that has been successfully used, not only in Bioinformatics [2], but also in other research areas [12]. In this section, we describe its workflow management according to the taxonomy proposed in [14]. Grid*W*ay natively handles Directed Acyclic Graph (DAG) based workflows, where each node is a task where its beginning is subject to (depends on) other tasks' finalization. Additionally, Grid*W*ay allows advanced flow structures like loops or branches by using its implementation of the Distributed Resource Management Application API (DRMAA), which is an Open Grid Forum [2] standard. Considering the workflow specification, Grid*W*ay uses an *abstract model* because the workflow is specified without referring to specific Grid resources for task execution. When converting abstract workflows into concrete ones, Grid*W*ay implements a *dynamic scheme* as it uses both dynamic and static information about resources. Scheduling decisions are made at run-time considering the requirements for each task and ranking expresions (a function to determining the resource assignment priority). Inside this scheme, Grid*W*ay employs *just in-time scheduling* which only makes decisions at the time of task execution. Moreover, historical information about task execution is considered. Grid*W*ay's scheduling architecture, as defined in [14], is *centralized* because one central workflow scheduler makes decisions for all tasks.

On the other hand, decision making is considered *local* because only takes each task information into account when scheduling them, as Grid*W*ay resolves task dependencies and starts working at task-level. Once a task (or graph node) is assigned to a resource, the executable and input files are staged onto the remote machine and the execution takes place. When the task finishes, the output files are staged back and Grid*W*ay checks if this event frees the dependencies of other tasks so this process starts again [3]. Grid*W*ay offers automatic staging mechanisms. In particular, a centralized approach is taken as intermediate data is transferred between resources via a checkpoint server to restart the workflow in case of failure.

Fault tolerance in Grid*W*ay must be considered at task-level taking into account possible failures such as network outage or remote and local machine crash. Grid*W*ay applies the following techniques, following the notation of [14]: *retry* (tries the task execution or file transfer on the same resource in case of failure), *alternate resource* (submits a failed task to an alternate resource) and *checkpoint/restart* (failed tasks are moved transparently to other resources).

[2] http://drmaa.org/

3 The Bioinformatics Application

The Bioinformatics application considered in this work performs protein clustering in order to eliminate redundancies in a protein database by comparing its entries [6]. In UniProt[3], which is the world's most comprehensive catalog of information on proteins, *cd-hit* is used to generate the UniRef reference data sets. For parallel processing purposes, a set of separate tools which perform the database division and each singular operation of the algorithm is provided.

The algorithm is represented in Figure 1. First off, a tool called *cd-hit-div* performs the protein database division. The first division, the representative one, is passed to the *cd-hit application* so it's compared to itself (represented as the A task in Figure 1). The output is then compared to the rest of partitions through the *cd-hit-2d* tool (represented as the B tasks). The first partition which results from the last operation is then the representative one (the A' task in Figure 1) and the process starts over until there aren't any more partitions. When the last comparison is performed, all the outputs of the *cd-hit* tool are merged with the *clstr_merge.pl* tool.

With the purpose of porting the application to the Grid, tasks have been divided in two types. In the first type, the job executes both *cd-hit-2d* and *cd-hit* over the given database division. In the second type, the job executes just *cd-hit-2d*. Both the database division and final merging are performed locally. As can be understood from Figure 1, tasks from a certain level cannot start until the first type job from the previous level is not finished, so task dependencies must be managed in this workflow. In a workflow, the node path translated into a sequence of tasks, to which the completion time is subject, is called *critical path* [8]. In this case, the *critical path* is composed by the execution of the first type tasks (the shaded nodes in Figure 1).

4 Experimental Results

The input protein database is a part of the RefSeq database[4] provided by the National Center for

[3] http://www.pir.uniprot.org/database/DBDescription.shtml
[4] http://www.ncbi.nlm.nih.gov/RefSeq/

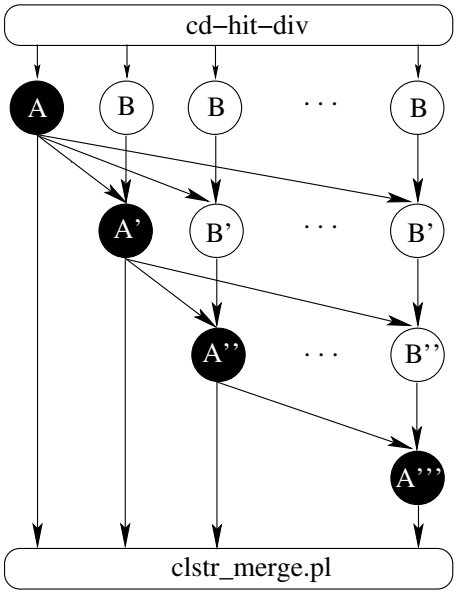

Figure 1. The cd-hit algorithm. White tasks execute *cd-hit-2d* and shaded tasks execute both *cd-hit-2d* and *cd-hit*.

Biotechnology Information (NCBI). Its size is 435MB and stores 504.876 proteins. Moreover, the input file size and the number of the resulting tasks depend on the number of database divisions. Table 1 shows these figures for the experiments performed. For the sake of completeness, the *cd-hit* and *cd-hit-2d* executable file sizes are both 1.1MB.

The experiments where run on resources (see Table 2) from the Enabling Grids for E-sciencE (EGEE) project, as Grid*W*ay can be used in this testbed [11]. The goal of this project is to build the most large production-level grid with great levels of performance and reliability. The tasks were launched, one experiment per division number, from the Universidad Complutense de Madrid at different times on different days of the week during July 2006.

Table 1. Input file sizes and number of tasks for each database division.

DB Div.	Mean Size	Tasks
10	44MB	45
12	36.5MB	66
14	31.5MB	91
16	27.5MB	120
18	24.5MB	153
20	22MB	190
22	20MB	231

Table 2. Testbed resources. All DRMS are PBS.

Site		Processor	Nodes	Speed
BIFI	ES	Intel P.IV	56	3.2GHz
CESGA	ES	Intel P.III	16	500MHz
CGG	FR	Intel P.III	58	1.2GHz
CIEMAT	ES	Intel Xeon	226	3.2GHz
GRIF	FR	Intel P.IV	14	2.8GHz
JINR	RU	Intel P.D	30	2.8GHz
L.-HEP	UK	Intel P.IV	374	3GHz
PNPI	RU	Intel P.IV	60	3GHz
RAL	UK	Intel P.IV	62	2.8GHz
RALPP	UK	Intel P.III	1064	1GHz
ScotGRID	UK	Intel Xeon	6	2.8GHz
SINP	RU	Intel Xeon	94	2.8GHz

Let us first consider the Grid execution of each node of the *cd-hit* workflow. As can be expected, the behavior of the algorithm depends on the number of initial database divisions. Note that, as it increases, the parallelism level of the application is favored; although the computation to file transfer ratio gets worse. This fact is clearly shown in Figure 2, where the average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{que}) times are presented for different number of database divisions. Moreover, the time a task waits in the remote queuing system is not related to the number of divisions as it only depends on the remote resource load status, which is high because the EGEE infrastructure is at production level. Therefore, the queuing time is the most significant part of the walltime, not considering transfer times, of each task in all the experiments.

In order to analyze the impact of the above considerations in the workflow walltime, let us define the *expected walltime* (T_{exp}) without considering the task queuing times and job failures. This time can be estimated by taking into account that the completion of each level of the workflow is subject to the most significant node's execution:

$$T_{exp} = N \cdot (T^A_{cpu} + T^A_{xfr}), \quad (1)$$

where N is the number of database divisions, and T^A_{cpu} and T^A_{xfr} are the mean CPU and transfer times of the shaded tasks in Figure 1. Also, a lower bound estimation of the walltime is: $T_{min} = N \cdot T^A_{cpu}$. Finally, we can compare them with the sequential execution of the

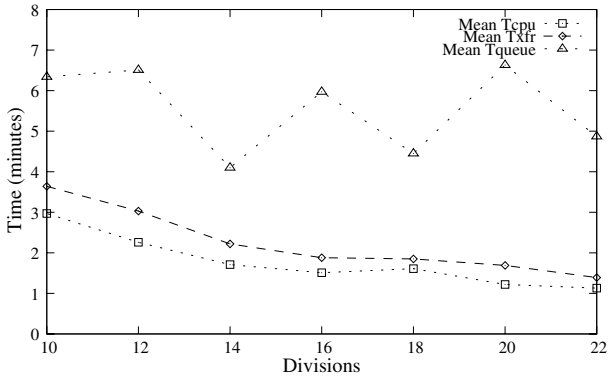

Figure 2. Average CPU (T_{cpu}), file transfer (T_{xfr}) and queuing (T_{que}) times for the workflow tasks and different number of database divisions.

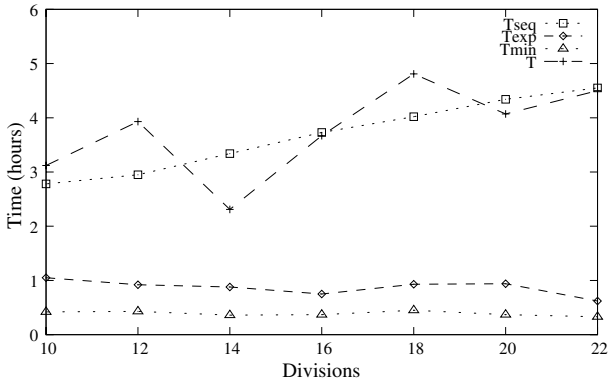

Figure 3. Workflow execution times for different number of database divisions.

workflow (T_{seq}), that can be easily estimated with:

$$T_{seq} = N \cdot T^A_{cpu} + T^B_{cpu} \cdot \sum_{n=2}^{N}(n-1), \qquad (2)$$

where T^B_{cpu} is the CPU time of the white tasks in Figure 1. The CPU times in Equation 2 are measured in the testbed's fastest machine (Intel Pentium IV 3.2GHz).

Figure 3 shows the previous times along with the experimental time (T) obtained in the execution of the workflow for different number of database divisions. As can be observed the Grid execution time of the workflow is similar to that obtained in a single machine; and away from that predicted by equation 1. This is mainly due to the overhead imposed by the remote resource management systems, as shown in Figure 2.

The walltime of the workflow depends also on the

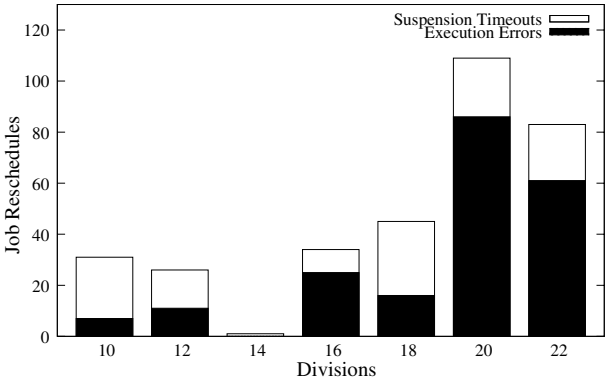

Figure 4. Number of jobs rescheduled in each experiment.

number of times a job is rescheduled to other resource. In our case, these reschedules are due to execution errors (i.e. middleware failures) or suspension timeouts (a job waits in the remote queue more than 5 minutes). The number of jobs rescheduled in each experiment are shown in Figure 4. The influence of these two factors can be clearly observed in Figure 5 for the workflow execution with 14 database divisions.

It is interesting to analyze the potential speed-up that could be obtained for this kind of applications, where the level of parallelism is not constant. Note that the number of tasks decreases in each level of the workflow. To this end, we will consider: the speed-up (S) of the workflow, that obtained without considering queue wait times or job failures (S_{exp}), and an upper bound limit (S_{max}) computed using T_{min}. These three values are represented in Figure 5. As previously discussed, the speed-up obtained by the workflow is very limited. Moreover, the file transfer times also impose a significant reduction of the expected speed-up, when compared to the upper bound limit, S_{max}.

5 Other Workflow Management Systems

Many workload management systems have been developed to execute complex jobs in Grid infrastructures. These projects are generally based on custom middleware developments which made assumptions on the underlying infrastructure. The use of a general-purpose meta-scheduling system have shown its reliability and robustness to execute workflows in a production Grid infrastructures (Globus-based). The application takes advantage of the fault-tolerance, advance scheduling and deployment features of the GridWay meta-scheduler.

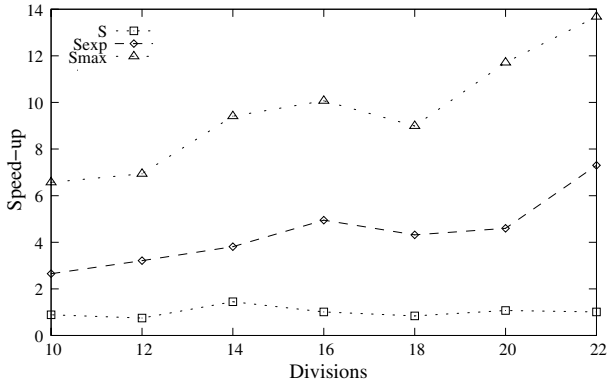

Figure 5. Speed-up of the workflow execution for different number of database divisions.

The Directed Acyclic Graph Manager (DAGMan) [10], provided by Condor, allows users to define jobs with dependencies, being Condor the one who executes each job. DAGMan offers fault tolerance with rescue DAG generation and a definable number of job execution retries.

Pegasus [1] is released as a part of the GriPhyN Virtual Data System (VDS) and extends DAGMan. It generates an executable workflow from the mapping of an abstract workflow to available Grid resources where artificial intelligence planning techniques may be used. In order to find available resources and needed data, Pegasus accesses various Grid information services such as the Globus Monitoring and Discovery Service (MDS) and the Replica Location Service (RLS), as well as the GriPhyN Transformation Catalog and Metadata Catalog Service (MCS). Then, the resource selection is performed both randomly and through a performance prediction infrastructure. Moreover, pluggable task scheduling strategies and just in-time scheduling are supported.

With Triana [9], code can run either locally or distributedly following a parallel or peer-to-peer policy. Tasks are dynamically allocated and both information retrieval and fault tolerance mechanisms are based on the Grid Application Toolkit (GAT) from GridLab [4].

In ICENI [7], several scheduling algorithms are provided such as random, best of n random, simulated annealing and game theory. Moreover, new algorithms can be plugged. Historical data can be used in scheduling as performance is being monitored for each resource and the user may specify metrics. Two scheduling schemes can be found in ICENI: lazy and advanced reservation using WS-Agreement.

GridAnt [5] extends the Ant deploy tool with new components and vocabulary. The information retrieval is performed through Globus MDS and the user defines fault tolerance mechanisms as the architecture of GridAnt is designed for user extension.

The hierarchical scheduling in Gridbus [13] uses the tuple-space model [13]. Gridbus workflow accesses the Grid Market Directory (GMD) in order to retrieve resource information including its access cost. On the other hand, for accounting and billing purposes, the Grid Bank (GB) service can be accessed as well. Failed tasks can be rescheduled to alternative resources. Finally, users can define quality of service constraints such as deadline and cost budget.

6 Conclusions

Grid computing is a matured technology that allows users to run embarrassingly distributed applications. As long as Grid computing reached these first needs, problems that must be faced are gaining more complexity. In this paper we have analyzed the porting of a bioinformatics workflow to a production Grid environment. This kind of workflow computations can not be performed for large protein databases in a single computer due to memory restrictions.

Grid Computing served to process these database divisions separately. However, the efficiency that could be expected is dramatically limited by the nature of the Grid itself: dynamism (queue times), heterogenity and high fault rate. Among the failures we found in our experiments, there were authentication errors, and the loose of callbacks and exit codes. In this study, the GridWay workflow engine has shown to be robust and reliable, as the computation of mid-size databases could be successfully carried out.

Acknowledgements

We would like to end this contribution thanking all the sites belonging to EGEE, in particular those whose machines participated in the experiments.

References

[1] 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. Second European AcrossGrids Conference (AxGrids 2004)*, volume 3165 of *Lecture Notes in Computer Science*, pages 11–20, 2004.

[2] E. Huedo, U. Bastolla, R. Montero, and I. Llorente. Computational Proteomics on the Grid. *New Generation Computing, special issue on Grid Systems for Life Sciences*, 22:191–192, 2004.

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

[4] K. Kurowski, B. Ludwiczak, J. Nabrzyski, A. Oleksiak, and J. Pukacki. Dynamic Grid Scheduling with Job Migration and Rescheduling in the GridLab Resource Management System. *Scientific Programming (AxGrids 2004 Special Issue)*, 12(4):263–273, 2004.

[5] G. V. Laszewski, K. Amin, M. Hategan, N. Zaluzec, S. Hampton, and A. Rossi. Gridant: A Client-Controllable Grid Workflow System. In *Proc. 37th Annual Hawaii International Conference on System Sciences (HICSS'04)*, 2004.

[6] W. Li, L. Jaroszewski, and A. Godzik. Clustering of Highly Homologous Sequences to Reduce the Size of Large Protein Databases. *Bioinformatics*, 17:282–283, 2001.

[7] S. McGough, L. Young, A. Afzal, S. Newhouse, and J. Darlington. Workflow Enactment in ICENI. In *Proc. UK e-Science All Hands Meeting*, pages 894–900, 2004.

[8] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, New Jersey, NJ, second edition, 2002.

[9] I. Taylor, M. Shields, and I. Wang. Resource Management for the Triana Peer-to-Peer Services. In J. Nabrzyski, J. M. Schopf, and J. Węglarz, editors, *Grid Resource Management*, pages 451–462. Kluwer Academic Publishers, 2004.

[10] D. Thain, T. Tannenbaum, and M. Livny. *Grid Computing*, chapter Condor and the Grid, pages 299–335. John Wiley & Sons, Inc., 2003.

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

[12] J. L. Vázquez-Poletti, E. Huedo, R. S. Montero, and I. M. Llorente. Massive Ray Tracing in Fusion Plasmas on EGEE. In *Proc. EGEE (Enabling Grids for E-sciencE) User Forum 2006*, 2006.

[13] J. Yu and B. Buyya. A Novel Architecture for Realizing Grid Workflow using Tuple Spaces. In *Proc. 5th IEEE/ACM International Workshop on Grid Computing (Grid 2004)*, 2004.

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