TOWARDS A LOW COST ETL SYSTEM

Vasco Santos[1], Rui Silva[2] and Orlando Belo[3]

[1,2]CIICESI - School of Management and Technology,
Polytechnic of Porto, Felgueiras, Portugal
[2]ALGORITMI R&D Centre, University of Minho, Portugal

ABSTRACT

Data Warehouses store integrated and consistent data in a subject-oriented data repository dedicated especially to support business intelligence processes. However, keeping these repositories updated usually involves complex and time-consuming processes, commonly denominated as Extract-Transform-Load tasks. These data intensive tasks normally execute in a limited time window and their computational requirements tend to grow in time as more data is dealt with. Therefore, we believe that a grid environment could suit rather well as support for the backbone of the technical infrastructure with the clear financial advantage of using already acquired desktop computers normally present in the organization. This article proposes a different approach to deal with the distribution of ETL processes in a grid environment, taking into account not only the processing performance of its nodes but also the existing bandwidth to estimate the grid availability in a near future and therefore optimize workflow distribution.

KEYWORDS

Data Warehousing, Grid Computing, ETL Processing, Distributing ETL tasks, ETL Performance Optimization

1. INTRODUCTION

In the last decade we have observed an exponential growth in the supply of computing devices at the same time as the increase of computational needs. Nevertheless the capabilities of such devices are often underused once they require not very demanding resources to perform their normal tasks, such as Web browsing, word processing, data analysis or social networking services, just to name a few. On the other hand, the usual approach to deal with this need for increased processing power was to purchase better devices, not considering the distributed processing capabilities that older less powerful devices could provide when cooperating in a network. Distributed and parallel processing is not a new issue to the scientific community. But the advances that occurred in network capabilities, network operating systems and portable devices have led to new fields of research. New terms have arisen like Grid computing [1] or Cloud computing [2]. Yet, both approaches have the same goals - distribution and resource sharing - with the focus shifting from a technology point of view to a service oriented philosophy. The use of grid environments in data processing intensive tasks has been studied in academic and scientific institutions already for a long time. Commercial organizations are now adopting this approach to help the mitigation of the impact of the increased amount of data gathered by their enterprise information systems that needs to be analysed, through testing grid middleware software. The basic idea is to take advantage (or to attenuate the effect) of the inactivity of their computing devices during a regular day helping them to perform those tasks. A grid based approach maximizes the investments already made and postpones some other expensive computer acquisitions [3]. One of the objectives of an enterprise information system is to support management's decision-making processes through the analysis of large amounts of data stored in them. Since data increases normally through time, the processing power needed to analyse it in a useful time also increases undermining the infrastructure available. A grid environment might be a suitable solution to this kind of problem, due to the easiness and inexpensiveness of adding new processing nodes to the infrastructure [4, 5].

In this article we studied the configuration and exploitation of a specific grid environment in order to receive and support the execution of an ETL process of a data warehousing system, giving particular emphasis to the distribution of ETL tasks, and focusing on the heterogeneous bandwidth capabilities of processing nodes interconnection. The article is organized as follows. In section 2, we briefly analyse related work in grid environments, task distribution and data warehouse applications. Next, we specialize our study on the distribution of ETL tasks over a grid environment (section 3), and present the model we conceived for ETL workflow distribution in a heterogeneous processing network, like a grid environment (section 4). Finally, we'll end the article with some brief remarks, pointing out some future developments and research lines.

2. RELATED WORK

Grid and Cloud computing environments have been the recent focus of attention of a large part of the scientific community, which was interested to achieve better performance and reliability in everyday business operations through the use of low cost computational power to data intensive operations [1]. Resource demanding operations, such as those present in the fields of health and astronomy, can benefit from this approach since they deal with large data sets requiring enormous computational resources to process them [6]. These environments are also being faced as a potential business opportunity, where resources are lent to clients that temporarily need them [7]. As a consequence of their work, recently researchers turned their attention to the complex problem of task distribution and management, relegating for a second place the grid itself (architecture, models, functionalities, etc.) [8-10]. As a natural evolution of these researching processes, the distribution and management of workflows [11-14] in grids remain a challenge and is becoming more popular in several fields of practice, in particularly in the Data Warehousing Systems (DWS) domain. DWS are known for gathering, processing and storing large amounts of data [15, 16]. Such characteristics blend well with the possibilities that any grid environment provides. There are already several studies where DWS's data is distributed in a grid environment [17-19], with particular emphasis on query distributing approaches [5, 20]. However, the use of grid environments has not been extensively studied in DWS. A DWS populating process, ETL for short, are data intensive tasks that prepare transactional data to be loaded into a data warehouse [21, 22]. Their modelling phase has been studied widely [23] in conjunction with the optimization of the workflows that they generate [24, 25]. However, it is recognized that there is a lack of research in the combination of grid environments and ETL processes because ETL tasks normally work over large data sets making bandwidth the primary concern in their distribution, thus driving away DWS researchers from this kind of approach. Recently, this last issue has been addressed by several researchers that proposed some new methods to monitor, quantify and predict bandwidth availability in grid networks [26-29].

3. SCHEDULING ETL TASKS ON GRID ENVIRONMENTS

Task distribution and management has been subject of intensive study especially over parallel architectures. Nevertheless, these architectures are very different from the normal grid environments, mainly due to the fact that these are composed of heterogeneous, autonomous and physically distributed resources. Shan [9] proposed a super scheduler architecture that manages jobs in a grid by cooperating with local schedulers, submitting, re-submitting or transferring jobs between the available resources. However, the lack of control over such local schedulers has led to several problems, since these resources are shared and their status and utilization change over time. Grid users submit jobs to the grid and the grid scheduler has to choose to which resources jobs should be sent, based on information about the grid - probably already out-dated. Keeping this in mind, Schopf [8] proposed a three-phase architecture for grid scheduling, which is based on resource discovery, system selection, and job execution. The first phase objective is to gather information about the resources that are available to the user. The second phase intends to identify which resources satisfy the minimum requirements asked by the user - operating system, software, and hardware configurations. Additionally, in this phase, some information gathering tasks are also executed, mostly concerning with dynamic updated information over resources, like CPU usage, or RAM/disk space availability. The last phase, job execution, involves a set of tasks that must be performed in order to execute the jobs, like advance reservation of resources, preparation of job submission, job monitoring and job completion tasks such as clean-up tasks.

Latter, Ben Segal [30] suggested that the availability and proximity of computational capacity and required data should be taken into account every time the grid's scheduler distributes and decomposes jobs for remote execution. This is very critical mainly because data is transferred to remote nodes over limited bandwidth - a normal situation in grid environments. More recently, other aspects have been taken into consideration, like fault recovery, checkpoint techniques or task replication and job migration. Also, the use of resource load prediction (usage profiling) has attracted some attention when we are interested in maximizing the performance and reliability of a grid [10]. Summarizing, whenever there is lack of server computational power to perform data intensive tasks, and exists a computer network with under-explored resources, the possibility of using a grid computing environment emerges as very viable and interesting solution. Even so, efficient task distribution and management becomes critical factors of success that real challenges such option. Therefore, in order to successfully address it, we must remember that information resource gathering is critical, job decomposition and replication are essential, to approach a maximization of the performance and reliability guarantee, particularly in environments with significant bandwidth bottlenecks.

4. Putting an ETL System on a Grid Environment

The use of grid environments to support DWS has mainly been adopted to distribute data and queries. However, since the DWS ETL component is very resource demanding and its characteristics are fit for distribution and parallelization, researching has been focused on the possible advantages of using grids, using already existing enterprises' computational resources when they are in idle time, for instance. Nevertheless, in order to take advantage of the potentialities of a grid in ETL, a very structured workflow of operations must be defined. In a previous work, we have presented an approach to model ETL workflows for grid environments, using Relational Algebra to specify each ETL task [31]. Tasks were distributed over the grid using a proposed performance prediction model [32] with the ability to categorize grid nodes according to their availability.

As a running example to help us introducing our proposal, we selected a simplified part of the Microsoft Database AdventureWorksDW[1] : a dimension table *dim_Person* that is fed from three different source tables. We present the logical model that specifies the transformation's flow needed to load the DW, using Vassiliadis notation [33] (Figure 1), and then present it oriented especially to grid environments.

[1] http://msftdbprodsamples.codeplex.com

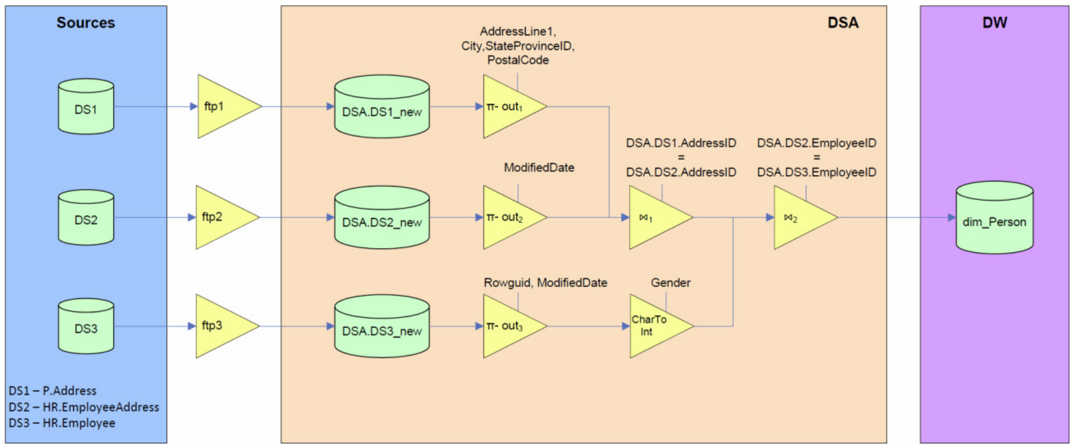
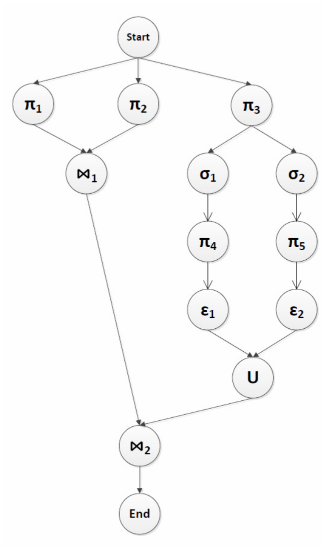


Figure 1 - ETL logical model – populating the dim_Person dimension table.

The characteristics of a grid environment led us to adopt Relational Algebra to specify each ETL task appealing to some new operators that extend relational algebra [34, 35]. Such extensions allow us to represent the most common ETL operations, and so distribute the ETL workflow over a grid (Figure 2). Data was stored in XML format for better compatibility in heterogeneous environments. Relational algebra operations were coded using JAVA.



Operation	Relational Algebra Representation
π_1	$\pi_{(\text{AddressID})}$ (P.Address)
π_2	$\pi_{(\text{EmployeeID, AddressID})}$ (HR.EmployeeAddress)
π_3	$\pi_{(\text{EmployeeID, NationalIDNumber, Title, HireDate, BirthDate, LoginID, MaritalStatus, Gender, ManagerID, ContactID})}$ (HR.Employee)
$\bowtie_1$	$R \leftarrow \pi_1 \bowtie \pi_2$
σ_1	$\sigma_{(\text{Gender='M'})}(\pi_3)$
π_4	$\pi_{(\text{EmployeeID, NationalIDNumber, Title, HireDate, BirthDate, LoginID, MaritalStatus, ManagerID, ContactID})}(\sigma_1)$
ε_1	$\varepsilon_{[\text{Gender}=1]}(\pi_4)$
σ_2	$\sigma_{(\text{Gender='F'})}(\pi_3)$
π_5	$\pi_{(\text{EmployeeID, NationalIDNumber, Title, HireDate, BirthDate, LoginID, MaritalStatus, ManagerID, ContactID})}(\sigma_2)$
ε_2	$\varepsilon_{[\text{Gender}=0]}(\pi_4)$
U	$S \leftarrow \varepsilon_1 \cup \varepsilon_2$
$\bowtie_2$	$R \bowtie S$

Figure 2 - (a) Relational algebra operations workflow; (b) Operations' specification

Once the ETL logical model is defined (Figure 2(a)), we need to represent it in a format that can be interpreted by a grid and then instantiated to its task scheduler with inputs from the Grid Information Service (GIS), such as information about the availability of the resources that meet the minimum requirements of each ETL operation. As already referred, the GIS provides valuable information about the most up to date status of the grid. Additionally, statistical information about past executions is quite important to improve task distribution, such as effective runtime, effective quantity of data transferred and received, etc. The architecture we proposed is represented in Figure 3. Nevertheless choosing the best computational node to execute a task is not an easy job. It's particularly important when the computational power of the node is not the decisive factor every time we need to choose where to delegate a task execution. Since ETL tasks tend not only to work over large data sets but also produce large ones too, the available bandwidth to transport the data is a critical factor in the evaluation of the execution node. Generalizing, the available bandwidth of each grid node, namely download and upload bandwidth, should be available to the scheduler or stored in the GIS.

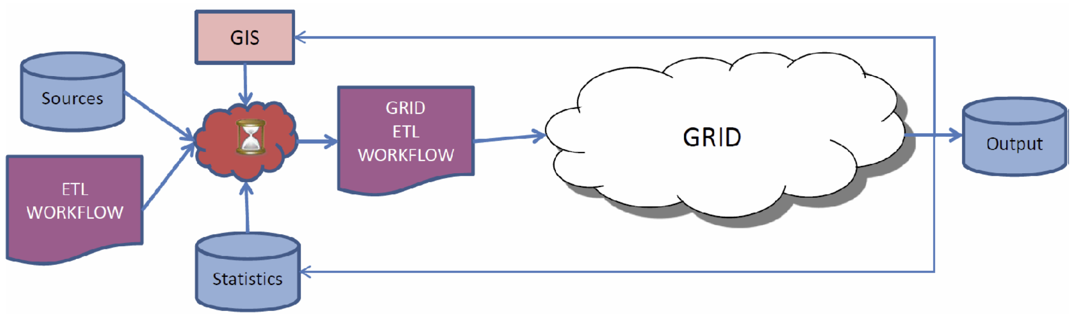


Figure 3 - A view of the grid architecture for ETL tasks.

5. Testing and Evaluating the Execution of ETL Tasks

To test and validate the way we make ETL tasks distribution involving large data sets in a grid, we used Globus Toolkit 4.2.1 Middleware [36]. In the test, we configured and used four different computational platforms as our test bed (Table 1).

Table 1 - The characteristics of the nodes.

PC	Processor	RAM	HD	OS	Network Card
debian1	Pentium 4 2.4 GHz	1.2 GB	70.3 GB	Ubuntu 10-10	100 Mb/s
debian3	Pentium 4 1.5 GHz	376.6 MB	35.6 GB	Debian 5	100 Mb/s
debian3	Pentium 4 2.4 GHz	494.9 MB	36.2 GB	Ubuntu 10-10	100 Mb/s
debian4	Pentium 4 2.8 GHz	1.5 GB	35.1 GB	Debian 5	100 Mb/s

The GIS of the Globus Middleware is provided by the MDS component [37]. Complementarily, we also used Ganglia monitoring system [38] to gather all the available information of the grid's nodes. This information is then accessed through a dedicated Globus Web Service by the scheduler.

To monitor and develop the scheduler module that deals with the defined workflow and then instantiates it to the grid we selected JAVA CoG Kit JGlobus [39]. *debian4* computer has all the grid's services, and *debian1* has the scheduler and all the source files needed for the workflow. As such, *debian1* will not accept any jobs, only the three remaining nodes will execute workflow tasks - the size of the source files is presented in Table 2.

Table 2 – Input files size.

File	Operation	Size
PersonAddress.xml	Projection_1	6.379 MB
HumanResourcesEmployAddress.xml	Projection_2	0.070 MB
HumanResourcesEmployee.xml	Projection_3	0.203 MB

5.1. Running Experiments Without Considering Bandwidth

Our first experiment was based in the work presented in [32], where each node of a grid is evaluated according to its performance availability.

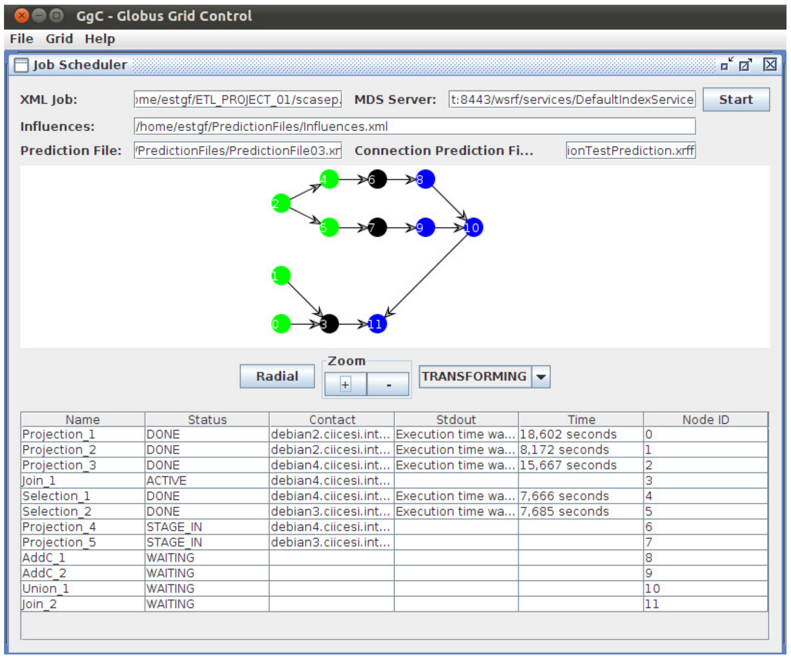
$$Perf_{AVAILABILITY} = (kCPU * kCPU_{ARCH} * CPU_{FREQ} * Free_{CPUs} + kMEM * Avail_{RAM}) * Avail_{COEF}. \quad (1)$$

In Formula 1 the performance is affected by a coefficient ($Avail_{COEF}$) that, in our case, is inversely proportional to the average CPU load of the last five minutes. We then use another proposed algorithm that classifies each node in an interval of six classes (C0-C5) based also on its performance availability. Then we define that each of these classes has an importance 20% higher than the preceding class (Table 3).

Table 3 - Probabilities assigned to the classes.

Class	Probability
C0	10%
C1	12%
C2	15%
C3	17%
C4	21%
C5	25%

Whenever a workflow is submitted to the grid, the scheduler generates a random number between 0.0 and 1.0 for each task, and calculates which node will execute the task according to its weight in the grid architecture. All workflow's tasks can be followed in the system observing the evolution of the grid's status (Figure 4). Since there is a random factor assigned to the distribution of the tasks, each experiment presented in this article is unique. The characteristics and problems observed for each approach are consistent even if we run several experiments for each approach. The computing nodes and times for each task vary slightly but the conclusions are the same.



XML Job: me/estgf/ETL_PROJECT_01/scasep.
MDS Server: t:8443/wsrf/services/DefaultIndexService
Influences: /home/estgf/PredictionFiles/Influences.xml
Prediction File: PredictionFiles/PredictionFile03.xr
Connection Prediction Fi... ionTestPrediction.xrff
TRANSFORMING

Name	Status	Contact	Stdout	Time	Node ID
Projection_1	DONE	debian2.ciicesi.int...	Execution time wa...	18,602 seconds	0
Projection_2	DONE	debian2.ciicesi.int...	Execution time wa...	8,172 seconds	1
Projection_3	DONE	debian4.ciicesi.int...	Execution time wa...	15,667 seconds	2
Join_1	ACTIVE	debian4.ciicesi.int...			3
Selection_1	DONE	debian4.ciicesi.int...	Execution time wa...	7,666 seconds	4
Selection_2	DONE	debian3.ciicesi.int...	Execution time wa...	7,685 seconds	5
Projection_4	STAGE_IN	debian4.ciicesi.int...			6
Projection_5	STAGE_IN	debian3.ciicesi.int...			7
AddC_1	WAITING				8
AddC_2	WAITING				9
Union_1	WAITING				10
Join_2	WAITING				11

Figure 4 - Showing the status of a running process.

When analysing the workflow, we see three branches of operations ending all of them with a *Join* operation. This operation can only start when all previous operations are terminated. The operation *Join_1* is very resource demanding and slows down the execution of the workflow (Figure 5).

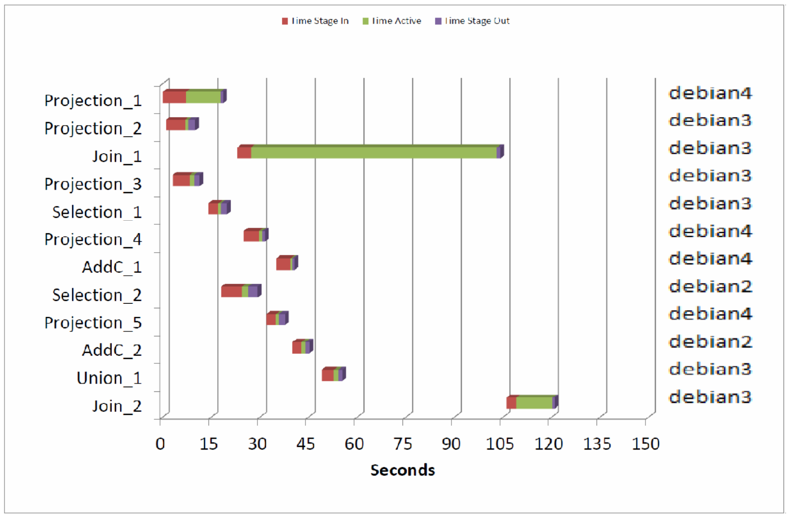


Figure 5 - Scheduling of the ETL workflow without prediction or limited bandwidth.

A second experiment involved a distribution model based on the prediction strategy presented in [32]. All nodes of the grid were evaluated taking in consideration their computing performance and memory capabilities through time. This information was stored in a statistical file and used to predict the class of each node in a time period. To help us on this specific task we used *RapidMiner*[2] decision tree algorithms. Predictions are one of the most important inputs to the scheduler. The overall worse performance of this approach, in comparison with the previous one, is justified by the fact that the grid used has a small number of working nodes (Figure 6).

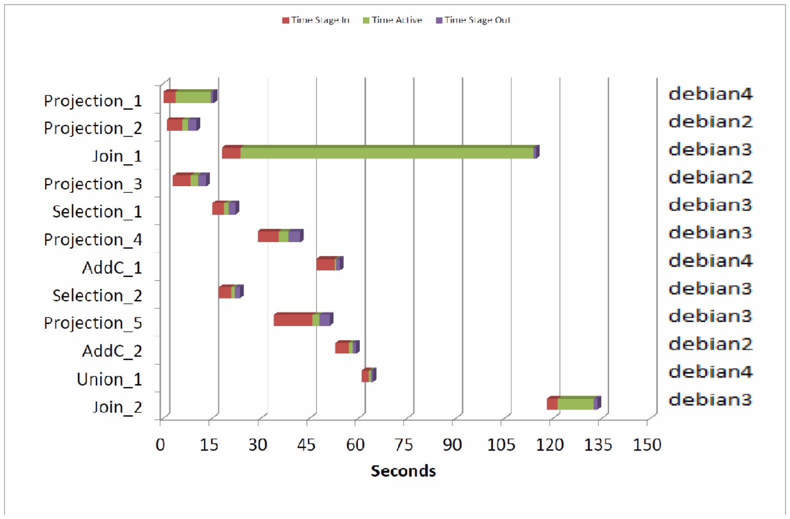


Figure 6 - Scheduling of the ETL workflow with prediction and without limited bandwidth.

The prevision model distinguishes the best nodes, but the random factor used to select the node and the reduced difference of importance in the class nodes, contribute to a slower outcome. In fact, a complex task may be assigned to a slower node (as was the case in *Join_1*) increasing the

[2] http://www.rapidminer.org

amount of time needed for completion. In addition, since we are using prediction to evaluate performance in a given time, the real availability of the nodes could be different.

In order to reduce the *Stage Out* time of the tasks and network communications when returning results, we decided to distribute branches of tasks to a same node. In this experiment case we intended to reduce communication bandwidth with the scheduler node as well as the *Stage In* and *Stage Out* time (Figure 7). To achieve this goal, we started to change the scheduler behaviour and remove the *Stage Out* phase. So, succeeding tasks could get directly the input files that they need from the preceding node. When a task is assigned to a node, the grid scheduler verifies if the preceding task has only one child. If so, it assigns the task to the same node that has the parent task in order to save some time, once the input file that it requires is already in the node. If the parent node has more than one child, the scheduler behaves in the same mode as already described. The only difference is that the input files are not located in the scheduler node but in the node where the parent task was executed. Therefore, the transference of the input files occurs between those two nodes.

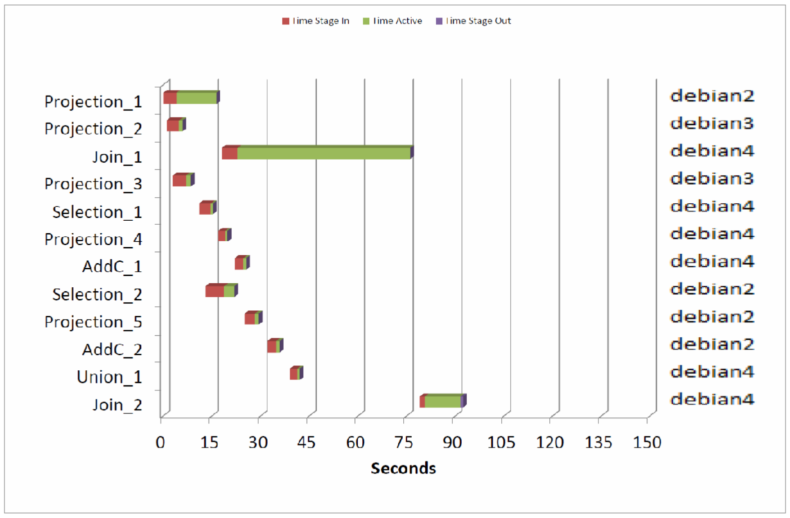


Figure 7 - Scheduling of the ETL Workflow, with prediction, branch optimization and without limited bandwidth.

5.2. Running Experiments with Limited Bandwidth

The next batch of experiments was based on the premises that grid architectures have usually a lot of nodes with disparate characteristics in terms of performance capabilities and bandwidth connections. The latter, is one of the more relevant characteristic when we are worried about the transference of large files, undermining the scheduler effort of selecting the best computational node. In the grid we configured, we limited the bandwidth of the node *debian3* to 20Mbits/s instead of the original 100Mbits/s. Then, we resubmitted the tasks workflow accordingly to the guidelines presented in Figure 6 and 7, i.e., with prediction and branch optimization, respectively. In this experiment, the first task was assigned to the limited node and the *Stage In* phase took, in average, two times more than when we used normal bandwidth. This result is justified mainly because on this phase we considered authentication between the nodes, independently of the bandwidth available, which consumes, as we know, a precious slice of time. We also had other tasks assigned to the limited node. However, the dimension of the data transference involved with was not relevant to the outcome.

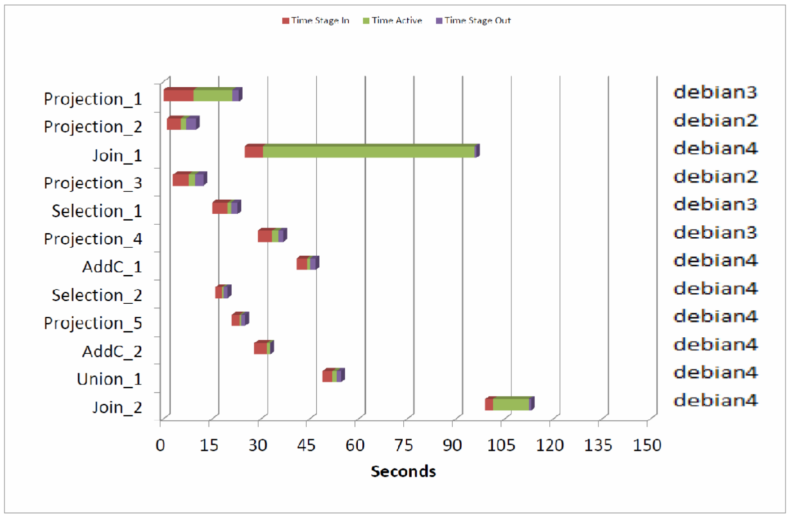


Figure 8 - Scheduling of the ETL Workflow, considering prediction, and limited bandwidth on *debian3*.

Distributing the workflow with branch optimization on a limited bandwidth environment is also problematic. However, since a branch is assigned to a node, data is only transferred in the beginning and at the end of the branch simplifying the effect of limited bandwidth on the overall result. Nevertheless, for each task, the grid scheduler keeps the transference of the task (and some additional information) to the execution node. In addition, if a limited node is chosen to execute a task that uses (or produces) a large data set, the impact of the limited bandwidth will be exponential. When comparing this last experiment with the ones that we showed previously in Figure 7, we quickly see that we got worst results. This last experiment is approximately 20% slower than the one presented in Figure 7.

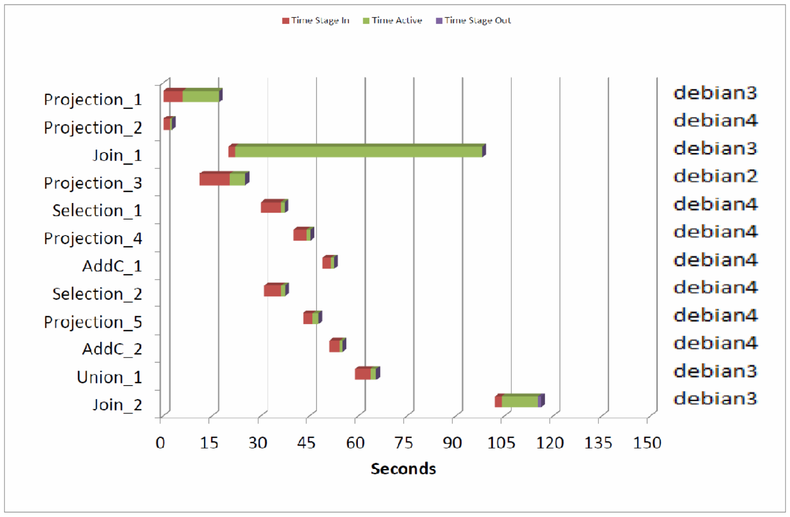


Figure 9 - Scheduling of the ETL Workflow, considering prediction, branch optimization, and limited bandwidth on *debian3*.

To finish our experiments, we adopted the same approach as followed in [32] but applying it not only to node performance but also to the bandwidth availability. We store statistical information over bandwidth availability of each grid node and use *RapidMiner* to predict bandwidth availability for a certain point of time. Then we average the performance and bandwidth classes of each node in order to define the overall class.

The result was an improvement if we consider the overall time, mainly because the limited node, although good in performance wise, has an overall reduced class that reduces the chance of being used to execute a task. Nevertheless, one branch was assigned to it. Improving the balance between bandwidth and performance and distinguishing the weight of each class used on nodes evaluation, would contribute to a better outcome. Eventually, each task of the workflow might have a ratio to balance bandwidth and performance according to its inputs, outputs and computational tasks.

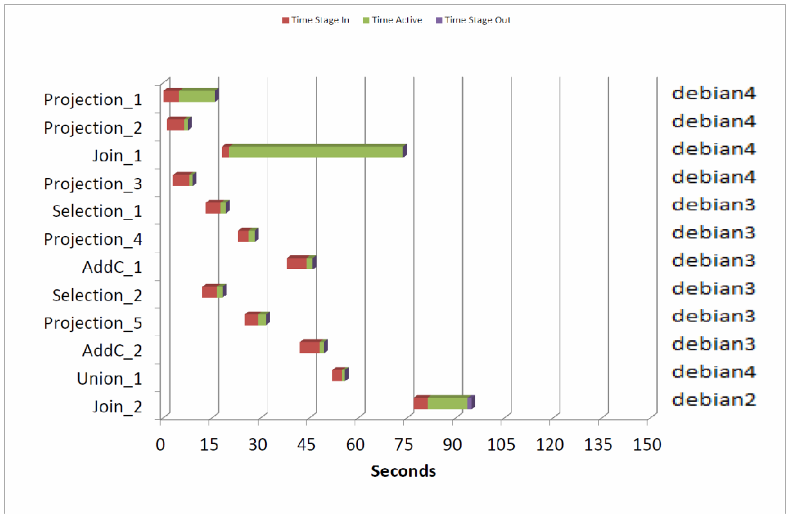


Figure 10 - Scheduling of the ETL workflow, considering prediction based on performance, available bandwidth, and branch optimization.

6. Conclusions and Future Work

In most cases, DWS are extremely demanding in terms of computational resources, requiring significant processing and storing capabilities. They use to deal with very large data sets during a limited time window, in a daily basis. The architectures that are defined and installed to support their activities are not available to all companies, especially the ones with scarce financial resources. Besides, as a DWS grows its architecture also needs to evolve, frequently implying strong restructurings to receive new requirements and services. These are, as we know, very expensive processes. In this article we developed and tested an approach based on grid environments, as the supporting infrastructure exclusively for ETL systems. Our goal was to prove that a grid infrastructure could be an efficient alternative to implement such kind of processes, taking advantage, of course, of the existence of computational resources with low utilization. Thus, it's possible to reduce significantly the costs of a traditional ETL system implementation. To configure and run an ETL system (or a set of specific ETL tasks) we decompose the ETL workflow in a set of relational algebra operations, which were distributed and then executed, using JAVA and XML as support.

The scheduling of workflows that deal with large data sets is not only a computational problem but also a communications problem, since a grid environment might contain nodes with limited download and upload bandwidth. We studied the impact of scheduling an ETL workflow using performance prediction, branch optimization and finally bandwidth prediction. As we have seen, the first three tests we have made do not take into account the available bandwidth, which makes them impractical to be implemented, since it does not provide a credible solution to accommodate a conventional ETL system - we assumed that a grid normally does not comprises local nodes only. However, the following tests have considered this problem. The prediction of performance and bandwidth, although they may produce worse results than those who do not use prediction, allow reducing the work of the grid's scheduler. Otherwise it would still have to track the performance and the available bandwidth of each node before submitting the work. Additionally, we found that the submission of branches of tasks, significantly reduces the communication time between jobs, simply because the results are already stored at the execution node rather than having the scheduler receive and transmit those results. In spite of all these advantages, we know clearly that this kind of solution it's not viable for (near) real-time ETL processes or for ETL processes that demands exclusive resources for task execution. However, we believe for a small-medium class 3 ETL process that a grid environment could be a good infrastructure solution.

In a near future, several improvements and optimizations might be considered. Some of them will be related to the application of different weights to the classes of performance used to select the nodes, or to use different bandwidth and performance ratios in the evaluation of the nodes, particularly applying them according to each task characteristics.

REFERENCES

[1] I. Foster, C. Kesselman, and S. Tuecke. (2001). The Anatomy of the Grid : Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications, 15*(3), 200-222.

[2] M. A. Vouk. (2008). Cloud Computing – Issues, Research and Implementations. *Journal of Computing and Information Technology, 16*(4), 235-246.

[3] T. Demiya, T. Yoshihisa, and M. Kanazawa. (2008). Compact grid : a grid computing system using low resource compact computers. *Int. J. Commun. Netw. Distrib. Syst., 1*(2), 17.

[4] M. Poess and R. O. Nambiar. (2005). *Large scale data warehouses on grid: Oracle database 10g and HP proliant servers*. Paper presented at the Proceedings of the 31st international conference on Very large data bases, Trondheim, Norway.

[5] P. Wehrle, M. Miquel, and A. Tchounikine. (2005, 20-22 July). *A Model for Distributing and Querying a Data Warehouse on a Computing Grid.* Paper presented at the Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on.

[6] B. T. Beshah, J. Welter, and K. Morin. (2004). *Distributed Computing in Ground Processing.* Paper presented at the XXth ISPRS Congress, Istambul, Turkey.

[7] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger. (2002). Economic models for resource management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience, 14*(13-15), 1507-1542. doi: 10.1002/cpe.690

[8] J. M. Schopf. (2002). A General Architecture for Scheduling on the Grid. *Journal of Parallel and Distributed Computing*, 17.

[9] H. Shan, L. Oliker, and R. Biswas. (2003). *Job Superscheduler Architecture and Performance in Computational Grid Environments*. Paper presented at the Proceedings of the 2003 ACM/IEEE conference on Supercomputing.

[10] A. R. Mury, B. Schulze, and A. T. A. Gomes. (2010). Task distribution models in grids: towards a profile-based approach. *Concurrency and Computation: Practice and Experience, 22*(3), 358-374. doi: 10.1002/cpe.1474

[11] C. Junwei. (2003). *GridFlow: Workflow Management for Grid Computing*.

[12] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil*, et al.* (2004). Pegasus: Mapping Scientific Workflows onto the Grid. In M. Dikaiakos (Ed.), *Grid Computing* (Vol. 3165, pp. 131-140): Springer Berlin / Heidelberg.

[13] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal*, et al.* (2005, 9-12 May). *Task scheduling strategies for workflow-based applications in grids.* Paper presented at the IEEE International Symposium on Cluster Computing and the Grid, 2005.

[14] J. Yu and R. Buyya. (2005). A Taxonomy of Workflow Management Systems for Grid Computing. *Journal of Grid Computing, 3*(3), 171-200. doi: 10.1007/s10723-005-9010-8

[15] W. H. Inmon. (2002). *Building the Data Warehouse* (3rd ed.): John Wiley & Sons.

[16] R. Kimball, M. Ross, W. Thornthwaite, J. Mundy, and B. Becker. (2008). *The Data Warehouse Lifecycle Toolkit - Practical Techniques for Building Data Warehouse and Business Intelligence Systems* (Second Edition ed.). : Wiley Publishing, Inc.

[17] S. Iqbal, J. J. Bunn, and H. B. Newman. (2003). *Distributed Heterogeneous Relational Data Warehouse In A Grid Environment.* Paper presented at the Computing in High Energy and Nuclear Physics, La Jolla, California.

[18] W. Dubitzky, D. McCourt, M. Galushka, M. Romberg, and B. Schuller. (2004). Grid-enabled data warehousing for molecular engineering. *Parallel Computing, 30*(9-10), 1019-1035. doi: 10.1016/j.parco.2004.07.009

[19] R. L. d. C. Costa and P. Furtado. (2007). *An SLA-Enabled Grid DataWarehouse.* Paper presented at the 11th International Database Engineering and Applications Symposium (IDEAS 2007), Banff, Alberta, Canada.

[20] P. Wehrle, M. Miquel, and A. Tchounikine. (2007, 21-23 May). *A Grid Services-Oriented Architecture for Efficient Operation of Distributed Data Warehouses on Globus.* Paper presented at the Advanced Information Networking and Applications, 2007.

[21] R. Kimball and J. Caserta. (2004). *The Data Warehouse ETL Toolkit - Pratical Techniques for Extracting, Cleaning, Conforming, and Delivering Data.* : Wiley Publishing, Inc.

[22] A. Albrecht and F. Naumann. (2008). *Managing ETL Processes.* Paper presented at the Proceedings of the International Workshop on New Trends in Information Integration, NTII 2008, Auckland, New Zealand.

[23] A. Simitsis. (2003). *Modeling and managing ETL processes.* Paper presented at the 29th International Conference on Very Large Data Bases, Berlin.

[24] P. Vassiliadis, A. Simitsis, M. Terrovitis, and S. Skiadopoulos. (2005). Blueprints and Measures for ETL Workflows (pp. 385-400).

[25] A. Simitsis, K. Wilkinson, U. Dayal, and M. Castellanos. (2010, 1-6 March). *Optimizing ETL workflows for fault-tolerance.* Paper presented at the Proceedings of the 26th International Conference on Data Engineering, Long Beach, California.

[26] H. Ningning and P. Steenkiste. (2003). Evaluation and characterization of available bandwidth probing techniques. *Selected Areas in Communications, IEEE Journal on, 21*(6), 879-894.

[27] M. M. Yousaf and M. Welzl. (2005). A Reliable Network Measurement and Prediction Architecture for Grid Scheduling.

[28] L. Marchal, P. V. B. Primet, Y. Robert, and J. Zeng. (2006). *Optimal Bandwidth Sharing in Grid Environments.* Paper presented at the High Performance Distributed Computing, 2006 15th IEEE International Symposium on.

[29] N. W. Keat, A. T. Fong, L. T. Chaw, and L. C. Sun. (2006). Scheduling Framework for Bandwidth-Aware Job Grouping-Based Scheduling in Grid Computing. *Malaysian Journal of Computer Science, 19*(2), 117--125.

[30] B. Segal, L. Robertson, F. Gagliardi, and F. Carminati. (2000). *Grid computing: the European Data Grid Project.* Paper presented at the 47th IEEE Nuclear Science Symposium and Medical Imaging Conference, Lyons, France.

[31] V. Santos, B. Oliveira, R. Silva, and O. Belo. (2012). Configuring and Executing ETL Tasks on GRID Environments - Requirements and Specificities. *Procedia Technology, 1*, 112--117.

[32] N. Guerreiro and O. Belo. (2009). Predicting the Performance of a GRID Environment: An Initial Effort to Increase Scheduling Efficiency. In Y.-h. Lee, T.-h. Kim, W.-c. Fang, and D. Slezak (Eds.), *Future Generation Information Technology* (Vol. 5899, pp. 112-119): Springer Berlin / Heidelberg.

[33] P. Vassiliadis, A. Simitsis, and S. Skiadopoulos. (2002). *Conceptual modeling for ETL processes.* Paper presented at the Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP, McLean, Virginia, USA.

[34] P. W. P. J. Grefen and R. A. de By. (1994, 14-18 February). *A multi-set extended relational algebra: a formal approach to a practical issue.* Paper presented at the Data Engineering, 1994. Proceedings.10th International Conference.

[35] J. Albert. (1991). *Algebraic Properties of Bag Data Types.* Paper presented at the Proceedings of the 17th International Conference on Very Large Data Bases.

[36] I. Foster. (2006). *Globus Toolkit Version 4: Software for Service-Oriented Systems.* Paper presented at the IFIP International Conference on Network and Parallel Computing.

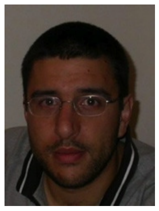
[37] Z. Xuehai and J. M. Schopf. (2004). *Performance analysis of the Globus Toolkit Monitoring and Discovery Service, MDS2.* Paper presented at the IEEE International Conference on Performance, Computing, and Communications.

[38] M. L. Massie, B. N. Chun, and D. E. Culler. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing, 30*(7), 817-840. doi: 10.1016/j.parco.2004.04.001

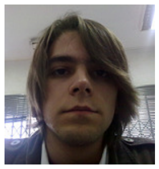
[39] G. von Laszewski, J. Gawor, P. Lane, N. Rehn, and M. Russell. (2002). Features of the Java Commodity Grid Kit. *Concurrency and Computation: Practice and Experience, 14*(13-15), 1045-1055. doi: 10.1002/cpe.674

Authors

Vasco Santos (http://www.estgf.ipp.pt) is an auxiliary professor at Polytechnic of Porto – School of Management and Technology of Felgueiras (IPP.ESTGF), in Portugal. He obtained is graduate degree in 1997 at University of Minho and is MSc. in 2004 in the same University. He is currently obtaining is PhD under the supervision of professor Orlando Belo. He is member of CIICESI research center in the IPP.ESTGF school and is currently developing is PhD studies in the area of data warehouse design methodologies in particular ETL conceptual and logical design.



Rui Silva (http://www.estgf.ipp.pt) is a guest assistant professor at Polytechnic of Porto – School of Management and Technology of Felgueiras (IPP.ESTGF), in Portugal. He obtained is graduate degree in 2009 at IPP.ESTGF and is MSc. in 2012 in the same University. He is member of CIICESI research center in the IPP.ESTGF school. His main research topic is related with ETL process scheduling in grid environments.



Orlando Belo (http://www.di.uminho.pt) is an associate professor, with habilitation, at the Department of Informatics of Minho University, in Portugal. He is also a member of the ALGORITMI R&D Centre in the same university, working in Business Intelligence, with particular emphasis in Data Warehousing Systems, OLAP, and Data Mining. His main research topics are related with data warehouse design, implementation and tuning, ETL services, and distributed multidimensional structures processing. During the last few years he was involved with several projects in the decision support systems area designing and implementing computational platforms for specific applications like fraud detection and control in telecommunication systems, data quality evaluation, and ETL systems for industrial data warehousing systems. More recently, he was developing some research work establishing OLAP usage profiles and optimizing OLAP selection methods incorporating knowledge acquired over OLAP user sessions and users' preferences.