A Grid Approach to Efficiently Embed Information and Knowledge about Group Activity into Collaborative Learning Applications

Santi CABALLE [a, 1], Fatos XHAFA [b], Thanasis DARADOUMIS [a], Claudi PANIAGUA[a], and Joan ESTEVE [a]
[a] *Open University of Catalonia, Spain*
[b] *Polytechnic University of Catalonia, Spain*

Abstract. Constantly embedding information and knowledge about group activity into on-line collaborative learning is a challenging yet one of the latest and most attractive issues to influence learning experience in a positive manner. The possibility to enhance learning group's participation by means of providing appropriate knowledge is rapidly gaining popularity due to its great impact on group performance and outcomes. Indeed, by storing parameters of interaction such as participation behaviour and giving constant awareness and feedback of these parameters to the group may influence group's motivation and emotional state as well as enhance the learners' and groups' problem solving abilities. This implies a need to capture and structure the information generated by group activity and then to extract the relevant knowledge in order to provide learners and tutors with efficient awareness and feedback as regards group performance and collaboration. To that end, we first identify and define the main types of information generated in on-line group activity and then we propose a process for efficiently embedding this information and the knowledge extracted into collaborative learning applications. However, in order to provide learners with effective knowledge, it is necessary to process large and complex event log files from group activity in a constant manner, and thus it may require computational capacity beyond that of a single computer. To that end, in this chapter we show how a Grid approach can considerably decrease the time of processing group activity log files and thus allow group learners to receive selected knowledge even in real time.

Keywords. Grid, parallel applications, collaborative learning, interaction analysis, knowledge management, CSCL.

Introduction

The provision of effective knowledge extracted from the information collected in Computer-Supported Collaborative Learning (CSCL) environments is essential for any form of cooperation, namely coordination, communication and collaboration [1]. It allows implicit coordination of collaborative learning, opportunities for informal,

[1] Corresponding Author: Open University of Catalonia, Department of Computer Science, Multimedia and Telecommunication. Rambla Poblenou, 156, 08018 Barcelona, Spain; E-mail: scaballe@uoc.edu.

spontaneous communication, and gives users awareness [2] and feedback [3] about what is happening during collaboration. Indeed, it is crucial for group members to be aware of others' participation in the collaborative process as this may enhance the collaboration a great deal in terms of decision-making, group organization, social engagement, support, monitoring and so on [1], [2]. Moreover, providing appropriate feedback about the collaborative activities may impact positively on the motivation, emotional state, and groups' well-being in on-line collaborative learning [1], [3] by means of a steady tracking of parameters related to group functioning, task performance and scaffolding [4] and by giving a constant feedback of these parameters to the group. Note that in this context information refers to quantitative and qualitative data generated by the learning group whereas knowledge refers to the result of the treatment of this information in terms of analysis techniques and interpretations that will be presented to the same group that generated it.

Therefore, participants in a collaborative learning experience may greatly enhance their abilities by increasing their knowledge about others in terms of cognitive processes and skills of the students and the group as a whole in solving problems, individual and group effectiveness regarding participation and interaction behavior, social support and help and so on. As a result, the success of CSCL applications depends to a great extent on the capability of such applications to embed information and knowledge of group activity and use it to achieve a more effective group monitoring [1], [4] as well as constantly provide group members with as much awareness and feedback as possible. Awareness [2], [6] refers to the knowledge provided to participants about both what other participants are doing at the same time and what they did in the past, whereas feedback [3], [7] goes one step further than awareness by providing exhaustive and elaborated information and knowledge of what is going on in the group over a long period of time. Furthermore, the persistent storage of the knowledge extracted as group memory [5] is essential for both students and tutors since, on the one hand, it allows participants not to access only the latest documents and data, which are commonly stored for later retrieval, but also the context in which they were created, and, on the other hand, it allows tutors to track the collaborative learning process for several purposes such as scaffolding and assessment of the learning outcome.

In all cases, the provision of effective knowledge implies receiving knowledge simultaneously both synchronously and asynchronously since the current and history interaction data shown are continuously updated. Therefore, on the one hand, users should be aware of the current activity in the group (the contribution of other members, their location and availability, the users working on a shared document at the same time and so on) and should know what other co-participants are doing in real time (e.g. during a multi-user editor session, who is editing and what is being shown). In an asynchronous context, on the other hand, users must know the activities performed by receiving deferred information of who, when, how and where others' interactions have been performed, and also why these interactions have been performed, which implies receiving complex knowledge of the interaction history. However, the supply of efficient and transparent feedback to users in both synchronous and asynchronous modes is a significant challenge. Users are continuously interacting with the system (creating documents, reading others' contributions, etc.) thus generating a lot of events, which, once collected, they must be classified, processed, structured and analyzed [6], [7]. As a consequence of the complex knowledge provided to participants (e.g., constant and automatic learner's assessment according to quantitative and qualitative

parameters of the interaction) we need to capture all and each type of possible data that could result in a huge amount of information that is generated and gathered in data log files.

CSCL applications are characterized by a high degree of user-user and user-system interaction and hence generate a huge amount of information usually maintained in the form of event type information. In order to capture the interaction correctly, this event information can be classified into different categories such as work sessions, messages, workspaces, documents and many other objects and thus may generate large size of information, especially, in real online collaborative learning that comprise complex learning activities to be carried out during a rather long period of time and involve a considerable number of participants. This information may include a great variety of types and formats and hence tends to be large in size [7]. Indeed, at a first level of classification, we can find group activity log files of CSCL applications associated with synchronous (e.g. multi-user editors) and asynchronous collaboration (e.g. discussion forums). These applications generate different types of information depending on their specific needs and functions (e.g. a discussion forum can generate event-type information so as to capture the participants' contributions). This information can be then stored in different formats.

Our experience at the Open University of Catalonia(UOC)[2] has shown the need to monitor and evaluate real, long-term, complex, collaborative problem-solving situations through data-intensive applications that provide efficient data access, management and analysis. As a result, there is a strong need for powerful tools that record the large volume of interaction data and can be used to perform an efficient interaction analysis and knowledge extraction. Given the real needs of any online collaborative learning situation, in order to provide different types of awareness and feedback, we need to capture all and each type of possible data that could result to a huge amount of information that is generated and gathered in data log files. Moreover, the need to make the analyzed information available in real time entails that we may come across with processing requirements beyond those of a single computer.

As a matter of fact, most of the existing approaches in the literature consider a sequential approach mainly due to three reasons: (i) processing for a specific purpose (i.e. limiting the quantity of information needed for that purpose); (ii) processing the information afterwards (i.e. not in real time) and (iii) processing of small data samples, usually for research and testing purposes (i.e. not for real learning needs). Yet, the lack of sufficient computational resources is the main obstacle for processing large amounts of data log files in real time. In real situations this processing tends to be done later, after the completion of the learning activity, thus having less impact on it [8].

Recently, Grid technology is increasingly being used to reduce the overall, censored time in processing data by taking advantage of its large computing support. The concept of a computational Grid [9] has emerged as a way of capturing the vision of a networked computing system that provides broad access not only to massive information resources, but to massive computational resources as well. Thus, in this chapter, we show how a Grid approach can be used to match the time processing requirements.

[2] The Open University of Catalonia (UOC) is located in Barcelona, Spain. The UOC offers distance education through the Internet since 1994. About 40,000 students, lecturers and tutors are involved in 600 on-line official courses from 23 official degrees and other PhD and post-graduate programs. The UOC is found at http://www.uoc.edu

Several studies have been conducted at the UOC [7], [10], [11], [12], [13] to show that a Grid approach can increase the efficiency of processing a large amount of information from group activity log files [6]. These studies have involved the interaction data collected from the log files of both the BSCW system [14] used at the UOC to support Problem-Based Learning practices in small groups and the own virtual campus of the UOC. The experimental results allow us to show first the gain provided by the Grid approach in terms of relative processing time and, second, the benefits of using the inherent scalable nature of Grid while the input log files are growing up in both number and large size. In this chapter we report the different experiments carried out at the UOC that show the feasibility of the Grid approach to achieve an effective embedding of the appropriate knowledge into collaborative learning practices.

The rest of the chapter is organized as follows. In Section 2 we give the context that motivated this research and make some reference to other studies in the field. In Section 3 we show the process of creating effective knowledge and exemplify a real situation. Section 4 presents the performance problems found in processing large and complex amounts of event information while Section 5 and 6 show several Grid-based approaches that help overcome this inconvenience. We conclude in Section 7 and 8 by analysing further the results obtained and outlining ongoing work.

2. Related Work

In a real context, the Open University of Catalonia provides several on-line courses involving hundreds of undergraduate students and a dozen of tutors in a collaborative learning environment. The complexity of the learning practices entails intensive collaboration activity generating a great amount of group activity information. To implement the collaborative learning activities and capture the group interaction we use the Basic Support for Cooperative Work (BSCW) [14] as a shared workspace system which enables collaboration over the Web by supporting document upload, group management and event service among others features. BSCW event service provides awareness information to allow users to coordinate their work [15]. Events are triggered whenever a user performs an action in a workspace, such as uploading a new document, downloading (i.e. reading) an existing document, renaming a document and so on. The system records the interaction data into large daily log files and presents the recent events to each user. In addition, users can request immediate email messages whenever an event occurs and the daily activity reports which are sent to them daily and inform them about the events within the last 24 hours.

BSCW and its extension to the learning domain called Synergeia [16] provides neither log file processing nor tools for analyzing the processed information, thus it can not provide enriched awareness to users to support collaboration and learning (e.g., model the users' behavior, and track and assess the collaborative learning process). Moreover, BSCW records the interaction as line-structured text files which are scarcely commented while they produce a high degree of redundancy. As a result, the processing and analysis of this huge amount of ill-structured information requires an efficient data processing system.

There exist several approaches in the literature that deal with the processing and analysis of data for awareness purposes so that to keep track of the collaborative learning activities. In [17] a tool for processing event logs of the BSCW is presented using a social network analysis method. Other approaches [18], focus on processing

event log files for analyzing the dialogue and interaction, or for using them in query systems to increase awareness [19]. On the one hand, these approaches consider the processing of the data just for a specific purpose, limiting thus the scope of the developed tools. On the other hand, they do not address the issue of processing time requirements that might result from the huge amount of data that are to be processed, which is a common issue in collaborative learning environments.

3. The Process of Creating Effective Knowledge

Creating useful, effective, heterogeneous, yet structured knowledge is a complex process. As part of the process of embedding information and knowledge about group activity into CSCL applications (see Figure 1 and [6]), this consists of four separate, indispensable stages: collection of information, processing, analysis and presentation. The entire process fails if one of these stages is omitted.

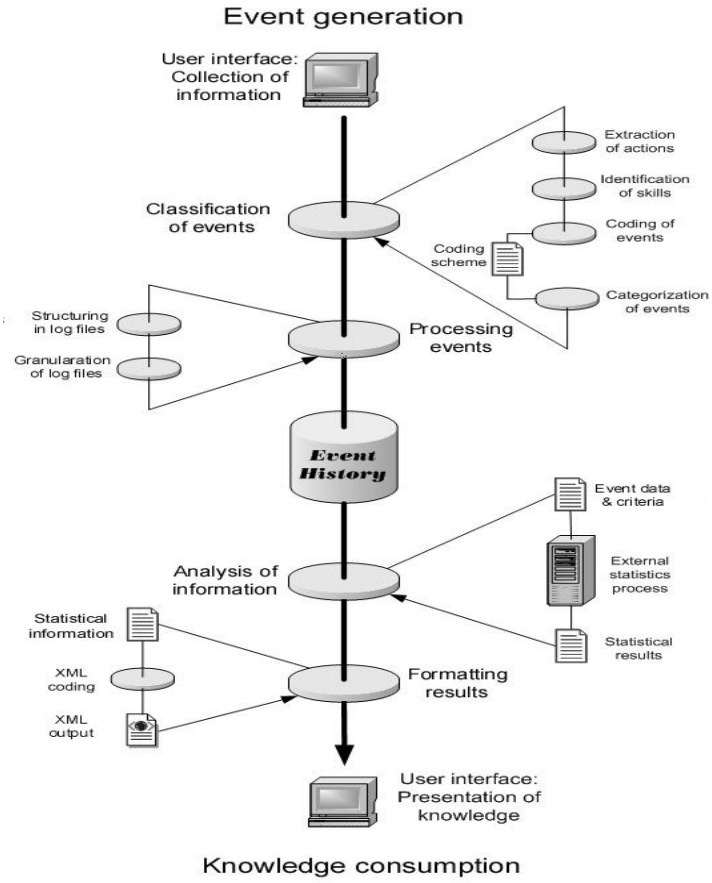

Figure 1. The embedding of information and knowledge about group activity into CSCL applications.

During the first stage, the information is collected from the collaborative actions triggered by the students and is classified in terms of specific events. Due to its conceptual importance, we now describe this stage in more detail. In particular, during the information collection, the objective is to gather as much information as possible generated during group activity to use it as a basis for the next three stages. To this end, we propose a solution consisting of classifying information by means of three generic group activity parameters, namely collaborative learning product, group functioning, and scaffolding (social support and task or group functioning help oriented services), which represent high-level collaborative learning processes [4], [20].

To measure each indicator (or skill) forming each mentioned parameter, we associate it with the actions that students perform and which represent each indicator in the best possible manner. We employ a similar terminology to the one usually used in the BSCW system [14] to refer to the actions that can be carried out in any groupware platform. Even so, they are general enough to be independent from BSCW and represent all the typical and basic actions encountered in every groupware platform.

The collaborative learning product parameter (see Table 1) features the production function and task performance of on-line groups. It is characterized by the type of actions (events) that capture and describe the functional knowledge, cognitive processes and skills of the students and the group as a whole in solving problems and producing learning outcomes in a collaborative learning practice. It is used to analyze and evaluate the individual and group effectiveness as far as task achievement concerns. It can be measured as a qualitative and quantitative parameter by the type of user task-based actions that represent contributions which express basic and supporting active learning skills as well as perception skills.

Table 1. Indicators (skills) that model *task performance*

Skills	**Sub-skills (Learning outcome contribution)**	**Actions (&objects) involved**
Basic active learning skills	Information generation	Create doc/note
Supporting active learning skills	Information refinement	Edit doc
	Information elaboration	Version/Replace doc
	Information revision	Revise/Branch doc
	Information reinforcement	Create_Noteboard doc/URL /Notes (as an attachment)
Information processing (perception) skills	Information acknowledgement	Read event

The group functioning parameter (see Table 2) is made up of the type of actions (events) that represent and are used to measure and analyze the individual and group effectiveness regarding participation and interaction behavior that facilitate the group's well-being function. As a quantitative parameter, it enables us to measure important participant contributions (in terms of specific types of user actions) which indicate skills related to: active or passive participation, well-balanced contributions and role

playing, participation quality and communication flow among group members, as well as the necessary skills that facilitate and enhance group interaction, namely active processing skills (such as task, workspace and communication processing skills). In addition, interaction behavior can also be measured as a qualitative parameter by group reflection (i.e. group and individual self-evaluation).

Table 2. Indicators (skills) that model *group functioning*

Skills	**Sub-skills (Group functioning contribution)**	**Actions (&objects) involved**
Active participation behavior and peer involve-ment skills	Participation in managing (generating, expanding and processing) information	Create Event, Change Event, Read Event
Social grounding skills	Well-balanced contributions, adequate reaction attitudes, and role playing	Create Event, Change Event, Read Event, Move Event
Task processing skills	Task planning	Create/Link Appointment Create/ChangeAccess WSCalendar
	Task (and knowledge) management	Create Folder Create Notes (as a contribution in a bulletin board)
Workspace processing skills	Workspace organisation and maintenance	Move event (cut, drop, copy, delete, forget)
Communication processing skills	Clarification	Change Description/ Change Event doc Change Description url
	Evaluation	Rate document/url
	Description (illustration)	Edit/Change Description Folder Change Description Notes
	Communication improvement	Edit Note Chvinfo/Chvno/Checkin/Checkout doc Rename Folder/Notes/doc/url/
	Meeting accommodation	ChangeDesc/ChangeDate /ChangeLocation Appointment

The scaffolding parameter (see Table 3) is specified by the type of events that refer to social support among members as well as to task- or group functioning-oriented help provided to a participant who is not quite able or ready to achieve a task on his or her own. As for the former, we look at event information that includes actions which support and promote group cohesion, such as motivational and emotional support, conflict resolution, etc. As for the latter, we focus on those specific actions designated to provide effective help to the peers when they need it during the collaborative learning activities. The participants' actions aiming at getting or providing help are classified and measured according to whether they refer to the task or group functioning.

In the next stage of processing, a tight structuring of the generated event information is performed and stored in log files. During the analysis stage, this information is analyzed and interpreted in order to extract knowledge according to different criteria. The final stage, presentation, is to show the students the knowledge obtained, thus providing them appropriate awareness and meta-cognition.

Table 3. Indicators that model *scaffolding*

Social support
Members' commitment toward collaboration, joint learning and accomplishment of the common group goal
Level of peer involvement and their influential contribution to the involvement of the others
Members' contribution to the achievement of mutual trust
Members' motivational and emotional support to their peers
Participation and contribution to conflict resolution
Help Services
Help is timely
Help is relevant to the student's needs
Help is qualitative
Help is understood by the student
Help can readily be applied by the student

Therefore, in order to provide users with continuous awareness, once the event information generated in group activity is correctly collected and classified, this information needs to be structured in a way that facilitates its later processing and analysis and can be addressed in a distributed environment such as a Grid [7], [8], [10].

To that end, during the processing stage, we propose the following generic steps so as to structure the event information correctly for later processing [8]: we first classify the event information collected from the collaborative actions and turn it into persistent data; then we store it in the system as structured log files. Next, we structure the files according to appropriate criteria (such as time and workspace) depending on the desired knowledge to be presented. For each of these criteria these files represent as great a degree of granularity as possible so that we can concatenate several log files and obtain the appropriate degree of granularity during data processing. This makes it possible for a distributed system such as Grid to parallelize data processing efficiently according to the characteristics of its computational resources.

Thereby, CSCL applications can take a great advantage of the inherent parallelism of a Grid environment to process several files (e.g. all the groups in a classroom) at the same time and thus considerably reduce the overall computational time. As a result, it is possible for these applications to process a large volume of collaboration activity data and make the results available in real time. The aim is to process large amounts of information efficiently enabling the constant presentation of both single awareness information in (almost) real-time and complex, in-depth deferred awareness.

At this point, in order to clarify the process of creating complex and useful awareness in a real situation, we briefly exemplify the case of monitoring the users' behavior by processing BSCW log file information involving time and workspaces. Note that the case of the UOC virtual campus and other collaborative learning systems are very similar.

We first take all the daily single BSCW log files as a starting point within the desired period of time that are to be analyzed. Each of these log files records all the event information generated each day and thus they collect all the users' activity in the system. This information is first classified according to users, time and workspaces so that it is only related to the user who generated events, the time when they occurred and the workspace where they took place. Then, the resulting log file is partitioned into multiple log files of a finer grain, each one storing all the workspaces that a certain user

has visited during the shortest time interval. Consequently, several of these log files will be concatenated so as to obtain the appropriate degree of granularity and thus fit their size to the characteristics of the computational resources of Grid's nodes. The aim is to distribute the workload among the Grid's nodes correctly and as a result to parallelize the data processing efficiently. Finally, the processed results are stored in a database in a way that they can be easily analyzed by statistical packages so as to extract knowledge about the user's behavior: usual time to enter and leave a certain workspace, time average staying in each workspace and number of times per day entering the same workspace are, among others, some analysis needs. At the end of the process, this obtained knowledge can be presented to the user as structured data.

4. A Sequential Approach

In this section, we report how to deal with the problem of extracting useful information from the event logs generated by both the BSCW system and the UOC's virtual campus.

4.1. The Problem of Processing log files of the BSCW

For the case of the BSCW, we developed a simple application in Java, called *EventExtractor*. This application runs offline on the same machine as the BSCW server and uses the daily log files generated by the BSCW server as input so as to: (i) identify the event boundaries inside the log file, (ii) map specific information contained in these events about users, objects, sessions, etc. to typed data structures, and (iii) store these data structures in a persistent support. Note that as the processing is done offline there is no need for a sensor component. In order to analyze the performance of this sequential application, we designed a specific test battery in which we used both large amounts of event information and well-stratified short samples consisting of all the existing daily log files making up the whole group activity generated during an academic term in the computer science subject "Software Development Techniques" at the Open University of Catalonia. This course involved two classrooms, with a total of 140 students arranged in groups of 5 students and 2 tutors. On the other hand, other tests involved a few log files with selected file size and event complexity forming a sample of each representative stratum. This allowed us to obtain reliable statistical results using an input data size easy to use.

All our test batteries were processed by this application on single-processor machines involving usual configurations. The test batteries were executed several times with different workload in order to have more reliable results in statistical terms involving file size, number of events processed and execution time along with other basic statistics. As an example, the experimental results from the sequential processing of the *EventExtractor* application are summarized in Figure 2, where for each event log file we show the relative comparison scale for the file size, number of events and the processing time. In a similar way, Figure 3 presents the processing results of over one hundred event log files involving file size and processing time showing that the processing time is linear on the size of the log file processed.

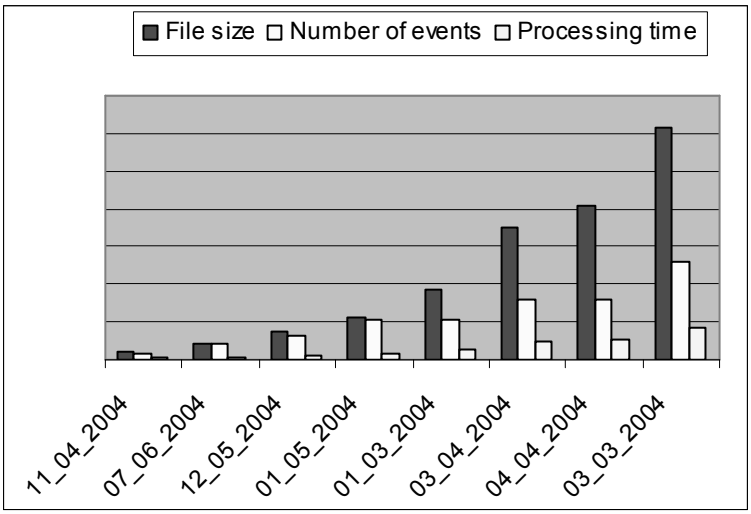

Figure 2. Relative comparison scale of a sample of selected log files with the group activity occurring in 8 certain days of the spring academic term of 2004. Note that due to the different event complexity, the number of events does not increase linearly with the file size.

However, when executing sequentially in a single machine, this application needs a lot of time and resources to process such amount of information and hence it is not feasible to constantly process data log files in real time. Therefore, these statistical results make evident the lack of computational resources as the main obstacle to constantly present knowledge to users in terms of awareness and feedback in real time.

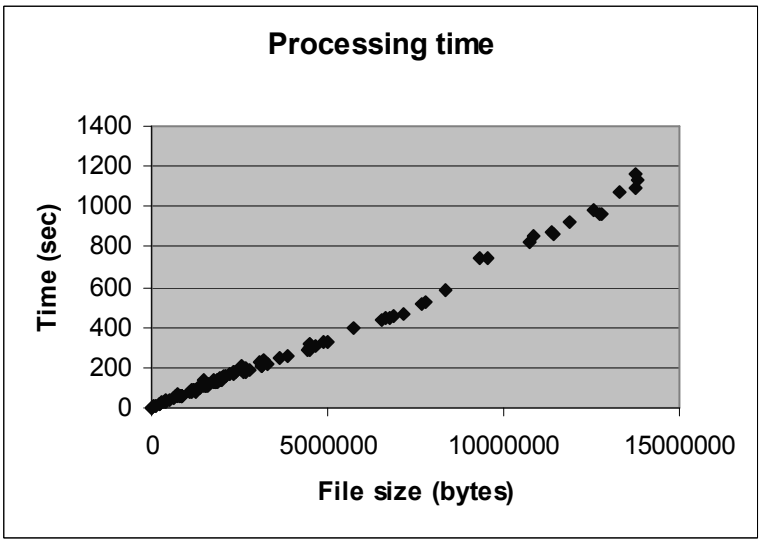

Figure 3. Sequential processing time for every event log file size.

4.2. The Problem of Processing log files of the Virtual Campus of the UOC

The on-line web-based campus of the UOC is made up of individual and community virtual areas such as mailbox, agenda, classrooms, library, secretary's office, and so on. Students and other users (lecturers, tutors, administrative staff, etc.) continuously browse these areas where they request for services to satisfy their particular needs and interests. For instance, students make strong use of email service so as to communicate with other students and lecturers as part of their learning process.

All users' requests are chiefly processed by a collection of Apache [21] web servers as well as database servers and other secondary applications, all of which provide service to the whole community and thus satisfy a great deal of users' requests. For load balance purposes, all HTTP traffic is smartly distributed among the different Apache web servers available. Each web server stores in a log file all users' requests received in this specific server and the information generated from processing the requests. Once a day (namely, at 01:00 a.m.), all web servers in a daily rotation merge their logs producing a single very large log file containing the whole user interaction with the campus performed in the last 24 hours.

A typical daily log file size may be up to 10 GB. This great amount of information is first pre-processed using filtering techniques in order to remove a lot of futile, non relevant information (e.g. information coming from automatic control processes, the uploading of graphical and format elements, etc.). However, after this pre-processing, about 1.8 GB of potentially useful information corresponding to 3,500,000 of log entries in average still remains [22].

Log file entries are structured following a type of format known as Common Log Format (CLF) [23] which is produced by most of web servers including Apache and is fairly configurable. For the purpose of registering the campus activity, log files entries were set up with the purpose of capturing the following information: who performed a request (i.e. user's IP address along with a session key that uniquely identifies a user session); when the request was processed (i.e. timestamp); what type of service was requested (a URL string format description of the server application providing the service requested along with the input values) and where (i.e. an absolute URL containing the full path to the server application providing the service requested).

At this point, we point out some problems which arise when dealing with these log files. Each explicit user request generates at least an entry in the log file and after being processed by a web server, other log entries are generated from the response of this user request; certain non-trivial requests (e.g. user login) involve in turn requesting others and hence they may implicitly trigger new log entries; the what and where fields contain very similar information regarding the URL strings that describe the service requested and the parameters with the input values; certain information is found in a very primitive form and is represented as long text strings (e.g. user session key is a long 128-character string). Therefore, there is a high degree of redundancy, tedious and ill-formatted information as well as incomplete as at some cases certain user actions do not generate any log entry (e.g. user may leave the campus by either closing or readdressing the browser) and have to be inferred. As a consequence, treating this information is very costly in terms of time and space needing a great processing effort.

In order to deal with the above mentioned problems and inconveniences, we developed a simple application in Java, called *UOCLogsProcessing* that processes log files of the UOC. In particular, this application runs offline on the same machine as the logging application server. It uses, as an input, the daily log files obtained as a result of

merging all web servers' log files. The following process is run: (i) identify the log entries boundaries and extract the fields that make up each entry, (ii) capture the specific information contained in the fields about users, time, sessions, areas, etc., (iii) infer the missing information, (iv) map the information obtained to typed data structures, and (v) store these data structures in a persistent support.

However, after running similar test batteries as those for the BDCW case, the repercussions of processing UOC's log file data sequentially are very similar to those mentioned for the case of BSCW log file processing. Thus, the sequential processing of the *UOCLogsProcessing* also takes too long to complete the work and it has to be done after the completion of the learning activity, which impedes from providing complex feedback to users in real time.

In the next section, we present an improved version of the processing of log files to parallelize the processing of information and the main experimental results achieved. Although our approach is generic and is valid for both types of log files (i.e., the BSCW system and the UOC virtual campus), for testing purposes we use the type of log file which most fits and can take advantage of characteristics of each resulting prototype.

5. Parallel Grid-based Approaches

In this section, we present improved versions of the sequential-execution applications to process log files. To this end, we first describe a general Grid-based approach to process log files. Then, we will show different Grid middleware approaches to realize our approach in order to efficiently parallelize the processing of log files. The next section will show the main experimental results achieved.

5.1. A Grid-based approach to process log files

Over the last few years, Grid computing has become a real alternative for developing parallel applications that employ its great computational power. However, due to the complexity of the Computational Grid, the difficulty encountered in developing parallel applications is higher than in traditional parallel computing environments. Thus, in order to simplify the development of Grid-aware applications several high-level programming frameworks have been proposed, among which is the Master-Worker Framework (MWF) [24].

The Master-Worker (MW) [25] model (also known as Master-Slave or Task Farming model) has been widely used for developing parallel applications in traditional supercomputing environments such as parallel machines and clusters of machines. In the MW model there are two distinct types of processors: master and workers. The master processor performs the control and coordination and assigns tasks to the workers. It also decides what data will be sent to the workers. The workers typically perform most of the computational work. The MW model has proved to be efficient in developing applications using different degrees of granularity of parallelism. Indeed, it has several advantages such as flexibility and scalability (the worker processors can be implemented in many different ways and they can be easily added if needed) as well as separation of concerns (the master performs coordination tasks and the worker processors carry out specific tasks). This paradigm is particularly useful when the

definition of the tasks to be completed by the workers can be done easily and the communication load between the master and workers is low.

MWF allows users to easily parallelize scientific computations through the master-worker paradigm on the computational Grid. On the one hand, MWF provides a top level interface that helps the programming tasks to distribute large computations in a Grid computing environment; on the other hand, it offers a bottom level interface to existing Grid computing toolkits, for instance, using the Condor system to provide Grid services. The target applications of MWF are parallel applications with weak synchronization and reasonably large granularity. As we show next, this framework is appropriate for processing log files of group activity since we have different degrees of granularity available so as to guarantee efficiency and, furthermore, there is no need for synchronization or communication between the worker processors. Moreover, in our application, the communication load between the master and workers is very low.

The architecture of the application (see Figure 4) is made up of three parts: (1) the Collaborative Learning Application Server, which is in charge of maintaining the log files and storing them in specified locations; (2) the MW application for processing log files and, (3) the application that uses the resulting information in the data bases to compute statistical results and present them to the final user.

Next subsection introduces two different realizations based on this architecture in the form of Grid middleware to efficiently parallelize the processing of logs files from both the *EventExtractor* and *UOCLogsProcessing* routines described in Section 4.

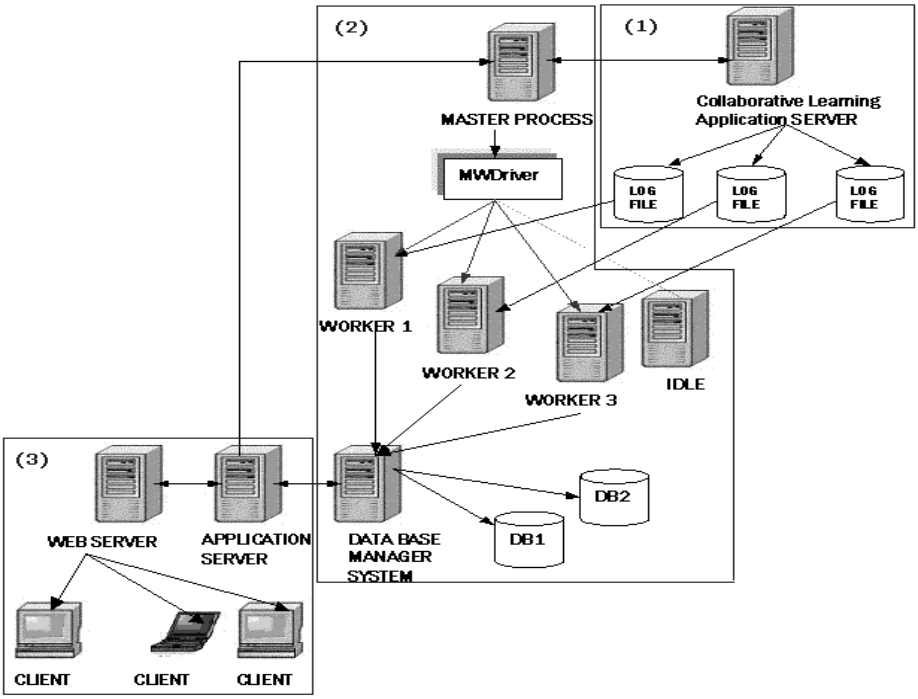

Figure 4. Generic architecture of the application for processing log files

5.2. Using Grid infrastructure to Parallelize theProcessing of Log Files

We show here how the MW paradigm is appropriate for processing log files of group activity in a Grid environment, since we have different degrees of granularity available and, moreover, there is no need for synchronization between the worker processors as tasks are completely independent from one another. To this end, we have written a minimal Grid implementation prototype using both the standard Globus Toolkit (GT) [26] middleware and an ad hoc middleware called Juxta-CAT [28] and have deployed it on the Planetlab [29] platform. The latter is first described next.

Planetlab [10], [29] is an open platform for developing, deploying and accessing planetary-scale services. It is, at the time of this writing, composed up of 825 nodes hosted in 406 different sites. Each Planetlab node runs the same base software, basically a modified Linux operating system offering services to create virtual isolated partitions in the node, which look to users as the real machine. The next subsection introduces two different realizations based on this architecture in the form of Grid middleware to efficiently parallelize the processing of logs files.

5.2.1. Using standard Grid middleware

The Globus Toolkit (GT) [6], [10], [26] is the actual de facto Grid middleware standard. Version 3 of GT (GT3) is a refactoring of version 2 in which every functionality is exposed to the world via a Grid service (i.e. basically, stateful web services). The core of the GT is a Grid service container implemented in Java that leverages and extends the Apache's AXIS [30] web services container.

In order to test this Grid prototype we used log files of the BSCW system due to their relatively small size and relatively low occurrence of complex events but with high variability of file size, which fits well in this case. To this end, we turned Planetlab into a Grid fabric by installing the GT3's Grid service container. Moreover, we implemented the worker as a simple Grid service that we deployed on the GT3's container. Then, we wrote a simple Java client that plays the role of the master by dispatching tasks just by calling the operations exposed by the worker Grid services, as follows:

- The worker Grid service publishes an interface with only one operation that the master calls in order to dispatch a task to the worker. This operation, which is implemented by wrapping the Java code of the mentioned *EventExtractor* routine (see Section 4), passes as an input a textual representation of the events to be processed by that task and returns a data structure containing performance information about the task executed (i.e. elapsed time, number of events processed and number of bytes processed).
- The master is just a simple Java application that reads from a configuration file (1) the folder that contains the event log files to process, (2) the available workers, (3) the number of workers to use, and (4) the size of the task to be dispatched to each worker expressed in number of events. The master then proceeds as follows: it picks as much workers as needed from the configuration file and puts them all in a queue of idle workers. Then it enters a loop reading the events from the event log files and, each time it has read a number of events, it either waits for a worker if the queue is empty or calls the worker's operation. Once the call to the worker returns, the worker is put back

into the queue of idle workers. The master exits the loop when all events in the event log files have been read and all the tasks that were dispatched have finalized.

As we have stated, this is not a real GT3 Grid implementation of the MW paradigm but a proof-of-concept prototype, thus important features in a real environment such as fault-tolerance and dynamic discovery of available workers, are missing. Finally, the experimental results of processing log files using GT middleware are shown in Section 6.

5.2.2. Using ad hoc JXTA-based Grid middleware

In this section, we briefly introduce the main aspects of the Grid platform, called Juxta-CAT [28], which we have used for the processing of log files. The Juxta-CAT platform has been developed using the JXTA protocols [31] and offers a shared Grid where client peers can submit their tasks in the form of java programs stored on signed jar files and are remotely solved on the nodes of the platform. Juxta-CAT Project and its official web site have been hosted in Java.NET community [28]. In order to test this Grid prototype we used the very large and ill-structured log files of the UOC virtual campus due to the great flexibility provided by the JXTA protocols, which allowed us to split the large log files into many short samples consisting of representative daily periods with different activity degrees.

The architecture of Juxta-CAT platform is made up of two types of peers: common *client peers* and *broker peers*.

- Client peers create and submit their requests using a GUI-based application (see Figure 5) and are the end users of the Juxta-CAT, which are obtained by downloading and installing the application from the official page of Juxta-CAT. Once the machine is converted into a client peer, the user will connect to the peer-to-peer network and can submit execution requests to their peer group nodes. Also, client peers will be able to process received requests sent to them by other nodes through the brokering and notify them the result of the requests, once they are completed.
- Broker peers are the administrators of the Grid, which are in charge of efficiently assigning client requests to the Grid nodes and notify the results to the owner's requests. Whenever a broker receives a request, it explores the state of the rest of nodes currently connected to the network, examining their working and connection statistics. Then, it uses this historical/statistical data to select, according to a price-based economic model, the best candidate peer for processing that request. To assure an efficient use of resources, brokers use an allocation algorithm, which can be viewed as a price-based economic model, to determine the best candidate node to process each new received petition.

The implementation and design of peers, groups, job and presence discovery, pipe-based messaging, etc. are developed using the latest updated JXTA libraries (currently release 2.3.7) and JDK 1.5 version.

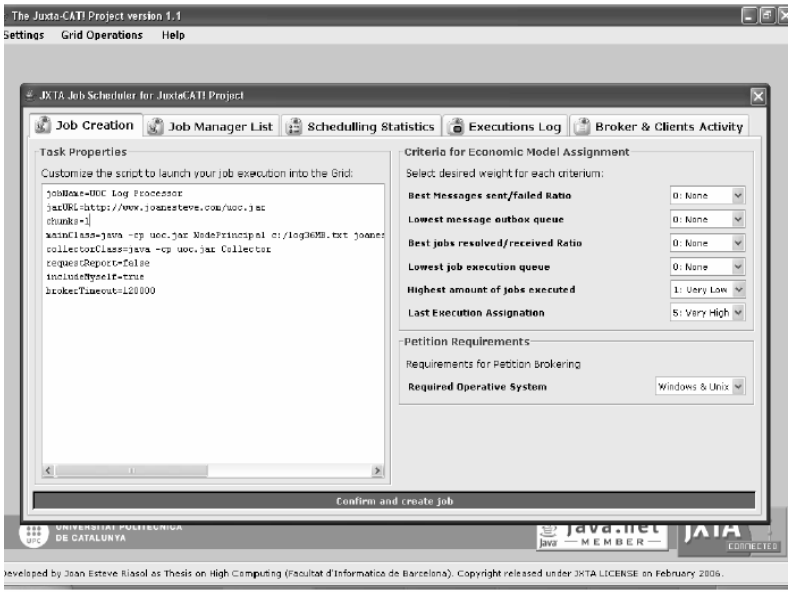

Figure 5: GUI view of submitting the processing of a log file to the Juxta-CAT.

We explain now how the processing of log files is done in the Juxta-CAT platform. The implementation follows the well-known Master-Worker paradigm. We note first that the sequential java class of the *UOCLogsProcessing* routine described in Section 4 also encapsulates functionalities to provide the division of the log file into as many equal parts as Grid nodes will be used for processing them; these parts will be later on submitted for processing to the Juxta-CAT. The main steps that would follow the user (the master node) to process a log file in the Juxta-CAT are as follows:

1. [**Preparation phase**]: Provide the necessary information (to the Master) for the preparation of the petitions to submit to the Juxta-CAT:
    a. Indicate the path to the log file and its name and the number of nodes participating in the processing. Log processor routines count the total number of lines of the log file, totalNbLines, and knowing the number of Grid nodes to be used, nbNodes, each node will read and process a totalNbLines/nbNodes of lines from the file.
    b. Indicate an FTP server, a user name and a password as well as a public address where the parts of the file will be uploaded. The implementation of FTP for Java, known as PureFTP, is included in the Jakarta Apache *commons-net-1.4.1.jar* library [32].
2. [**Master Loop**]: Repeat
    a. Read totalNbLines/nbNodes lines
    b. Upload the file to the indicated public address via FTP
    c. Create a petition and submit it to Juxta-CAT

    Until the original log file has been completely scanned.
3. [**Juxta-cat processing**]:

Figure 6. State information of a petition once it is processed.

a. Each time a petition is received by brokers of Juxta-CAT, it is assigned to a peer node of the platform.
b. The peer node, upon receiving the petition, reads according to the petition's description, the part of the file it has to read via HTTP. The peer runs *UOCLogProcessing* functionality for processing the lines of the file, one at a time, and stores the results of the processing in a buffer.
c. The peer node, once the processing of the petition is done, sends back to the master node the content of the buffer.

4. [**Master's final phase**]: Receive messages from peers and append the new received resulting file to the final file containing the information extracted from the original log file.

The *UOCLogsProcessing* routine is compiled in a unique java jar packages, which includes the library developed by Jakarta Apache needed for the FTP transfer. The code has been optimized using Java Proguard 3.5 so that the final jar file size is 28.7 KB. Figures 5 and 6 show the submission of a petition to Juxta-CAT and the state information once it is processed. Note that the user has to just provide the in formation needed in Step 1 (see above); the rest is automatically done by Juxta-CAT. The experimental results using Juxta-CAT are shown in Sub-Section 6.2.

6. Experimental Results

In order to carry out a comparative study between the sequential and Grid approaches, we designed a specific test battery in which we used both large amounts of event information and well-stratified short samples. In this section, we present the

experimental results achieved of our Grid prototypes while next section analyzes certain important aspects of these results to be considered.

6.1. Results from Paralleizing the Processing of BSCW Log Files

In order to test our GT-based prototype and compare the results to the sequential approach, we run the *EventExtractor* routine for processing log files from the BSCW system on this Grid platform in the Planetlab nodes. To this end, as mentioned in Section 4, we used the existing daily log files making up the whole group activity generated during a whole academic term in the course "Software Development Techniques" at the Open University of Catalonia. Other tests involved a few log files with selected file size and event complexity forming a sample of each representative stratum.

The linearity found in processing time in the sequential approach (see Figure 3) allowed us to greatly simplify the experiment by using the same event log file as input for all the Grid tests in the experiment. Then, we left to vary the parameters regarding both the number of workers and the size of the tasks (expressed in number of events) which were to be executed by the workers. We ran tests for a different number of workers with different task sizes.

Parallel speed-up is used to measure the performance gain from a parallelized execution over its serial execution and defined as `S(s,p) = Ts (s) / Tp(s,p)`, where s is the size of the log file, `Ts(s)` is the total running time of the sequential execution for a log file of size s and `Tp(s,p)` is the total running time of the parallel execution for a log file of size s, using p processors.

Figure 7 shows the maximum speed-ups achieved for the observed bandwidth between our master processor and the Planetlab nodes at the time of running the experiment and for the different number of workers we tested.

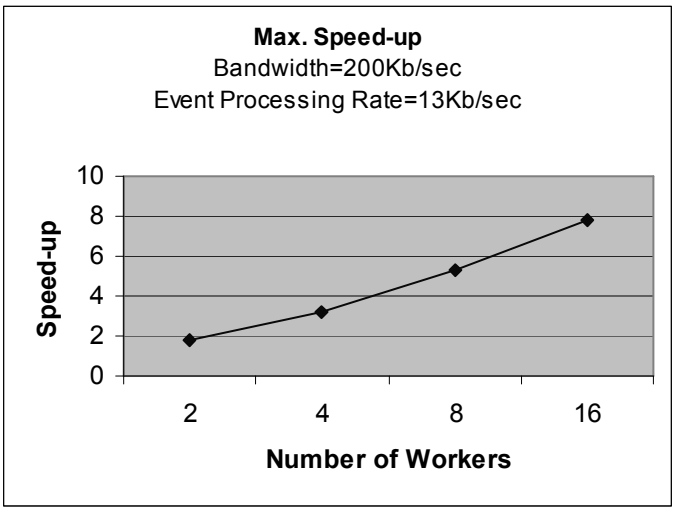

Figure 7. Maximum speed-up vs. number of workers

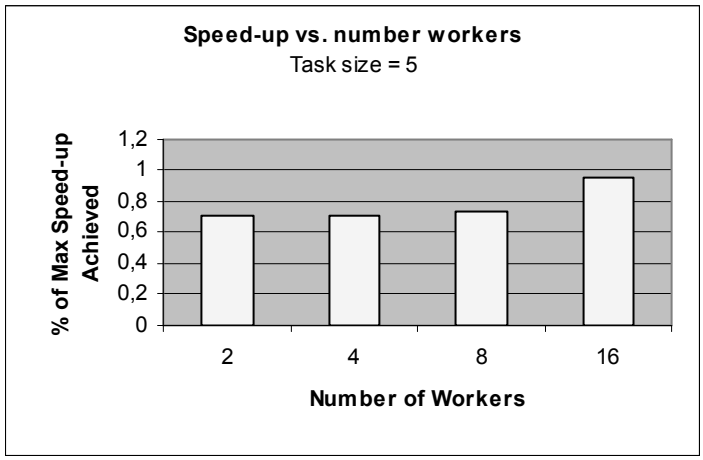

Figure 8. Relative speed-up vs. number of workers for a task size of 5 events

As mentioned before, the worker returns the elapsed time of its execution, whereas the master executes all the events found up to the input event log files have been completely parsed and all dispatched tasks have been completed. We computed the observed speed-up for the test by dividing (1) the sum of all the elapsed times returned by each invocation of the worker into (2) the elapsed time the master run multiplied by a normalization factor to compensate the different speed between the machine running the master and the Planetlab nodes running the workers.

Therefore, the main experimental results from the parallel processing of log files are given in terms of how much close each set of workers is to achieve its theoretic maximum speed-up (see Figure 7) for different task size processed and, thus, providing the best processing time possible while parallelizing the data processing. To this end, Figure 8 shows the graphical representation of an extract of these results in relative terms for a sample of a specific 5-event size task.

6.2. Results from Paralleizing the Processing of UOC Log Files

In this sub-section, we present the experimental results obtained after running our JXTA-based Grid platform Juxta-CAT in the Planetlab nodes on a test battery made up of the log files from the UOC virtual campus showing the speedup achieved.

This test battery uses both large amounts of log information (i.e. daily log files) and well-stratified short samples consisting of representative daily periods with different activity degrees (e.g. from 7 p.m. to 1 a.m. as the most active lecturing period and from 1 a.m. to 7 a.m. as the period with least activity in the campus). In addition, other tests involved a few log files with selected file size forming a sample of each representative stratum. This allowed us to obtain reliable statistical results using an input data size easy to use.

Table 4. PlanetLab nodes used to run the experiment.

Host	Description
planet1.manchester.ac.uk	University of Manchester
lsirextpc01.epfl.ch	École Fédérale de Lausanne
planetlab1.polito.it	Politecnico di Torino
planetlab1.info.ucl.ac.be	University of Louivain
planetlab2.upc.es	Universitat Politècnica de Catalunya
planetlab1.sics.se	Swedish Institute of Computer Sci.
planetlab1.ifi.uio.no	University of Oslo
planetlab3.upc.es	Universitat Politècnica de Catalunya
planetlab1.ls.fi.upm.es	Universidad Politécnica de Madrid
planetlab1.hiit.fi	Technology Institute of Helsinki
planetlab-1.cs.ucy.ac.cy	University of Cyprus
planetlab1.ru.is	University of Reykjavik
planetlab2.sics.se	Swedish Institute of Computer Sci.
planetlab1.mini.pw.edu.pl	Telekomunikacja Polska Warsaw
planetlab1.cs.uit.no	University of Tromso
planetlab-02.ipv6.lip6.fr	Laboratoire d'Informatique de Paris

The battery test was processed by the *UOCLogsProcessing* application described in Sub-Section 4.2 and executed several times first on single-processor machines involving usual configurations and with different workload in order to have more reliable results in statistical terms involving file size, number of log entries processed and execution time along with other basic statistics. Then, the same battery test was processed in a parallel fashion by Juxta-CAT using different number of nodes, specifically, 2, 4, 8, and 16 nodes, using the PlanetLab nodes appearing in Table 4.

Parallel efficiency measures the degree of utilization of the computing resources involved in the parallel computation and is defined as the parallel speed up (defined in Sub-Section 6.1) divided by the number of computing resources (i.e. processors): `E(s,p) = S(s) / p`.

From the execution times (see Figure 9) and the formulas previously introduced, we show in Table 5 the gain in terms of parallel speed-up and efficiency we achieved.

7. Analysis of the results

Analyzing the experimental results obtained from our Grid prototypes, we found that, on the one hand, from certain values of the task size, the speed-up observed was very close to the theoretic maximum achievable. This allows us to conclude that only for a very small value of the task size the impact on the speed-up can be great due to the cost of the transmission overhead. However, we also observed that the more workers we used in our tests the closer to the theoretic maximum was the speed-up achieved by the small tasks, and this increased quickly up to the point that, given a sufficient number of workers, even the smallest tasks (i.e. one-event task size) achieved considerable speed-up.

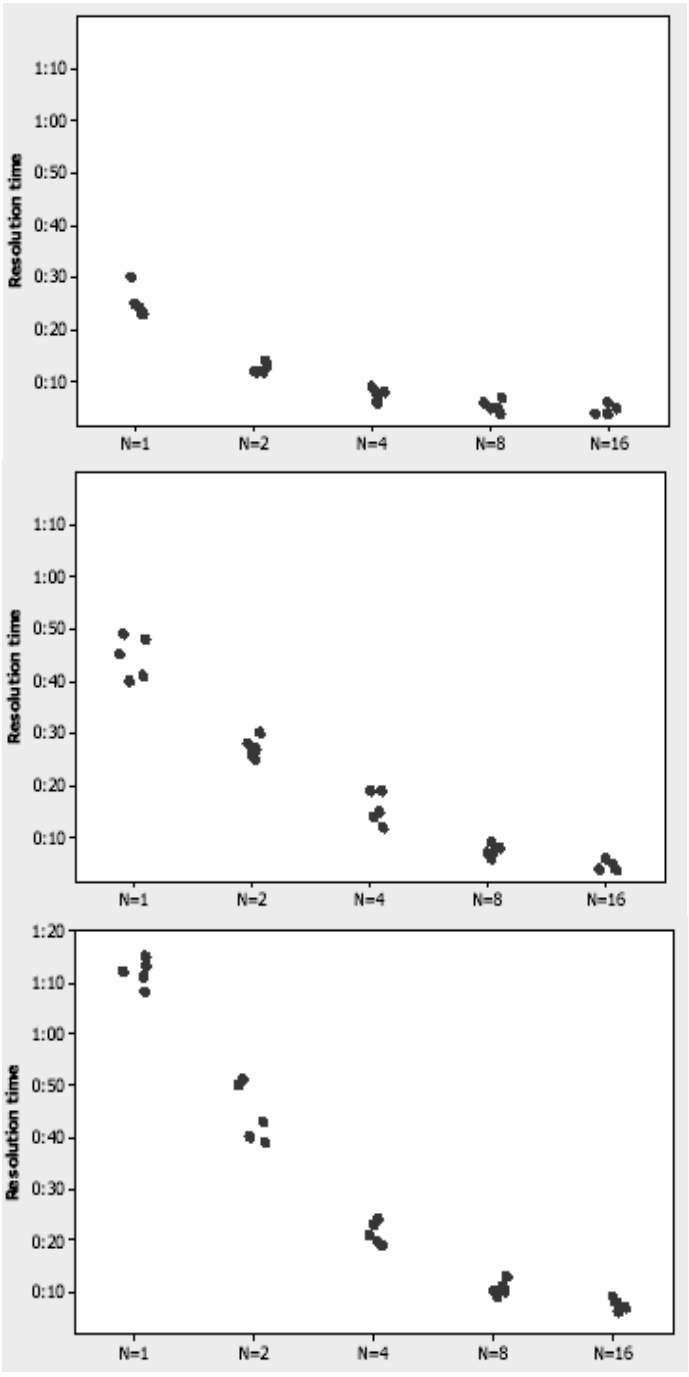

Figure 9. Three execution time results for log files with sizes of 12MB, 24MB and 36MB respectively; x-axis indicates the number of processors and y-axis the processing time (mm:ss).

Table 5. Parallel speed-up and efficiency.

Log file size	Speed-up	Efficiency
12 MB	6.1	38.2 %
24 MB	7.4	46,2 %
36 MB	9.1	56.8 %

On the other hand, the homogenous behavior observed in Planetlab nodes justified our decision of testing with the same task size for all workers. However, in a real Grid environment, task sizes should be adjusted per worker node case to fit the dynamically changing workloads the nodes may be experimenting and to account for different machine speeds.

We note, however, that although the results of this experiment are promising, a deeper and more precise analysis on both the primary interaction occurring between participants in the virtual classrooms and the real collaborative learning activity based on complex parameters of the collaboration, such as the above-mentioned task performance, group functioning and scaffolding, it is expected to generate a much larger amount and more complex events than those used for our experiments . This scenario will take much more advantage of the benefits provided by a Grid environment and will provide a more useful knowledge about the actual performance occurring in the on-line learning activity and will help monitor and support learning participants more conveniently.

8. Conclusions and Future Work

In this chapter, we have first argued how the provision of continuous knowledge to on-line teams in CSCL environments can greatly improve the group activity in terms of decision-making, group organization, social engagement, support, monitoring and so on. As a result, large amounts of log information generated from the collaborative interaction need to be efficiently processed and analyzed. Moreover, in order to make the knowledge extracted from the analysis be useful for awareness and feedback purposes (see Figure 10), users should be provided with both single information delivered fast in (almost) real-time and complex, exhaustive, yet structured deferred information thus stressing even more the processing requirements beyond those of a single computer.

We proposed then several Grid approaches to overcome these demanding requirements by improving the processing time of a large amount of log files storing complex event information from the group activity. The results obtained and the experience achieved in our studies allows us to eventually conclude that the question whether the Grid is beneficial or not will heavily depend on the volume and structure of information being processed. Therefore, these results encourage us to keep up working on the development of a real working Grid implementation to address the problem of processing group activity event log files.

We proposed then several Grid approaches to overcome these demanding requirements by improving the processing time of a large amount of log files storing complex event information from the group activity. The results obtained and the experience achieved in our studies allows us to eventually conclude that the question whether the Grid is beneficial or not will heavily depend on the volume and structure of

information being processed. Therefore, these results encourage us to keep up working on the development of a real working Grid implementation to address the problem of processing group activity event log files.

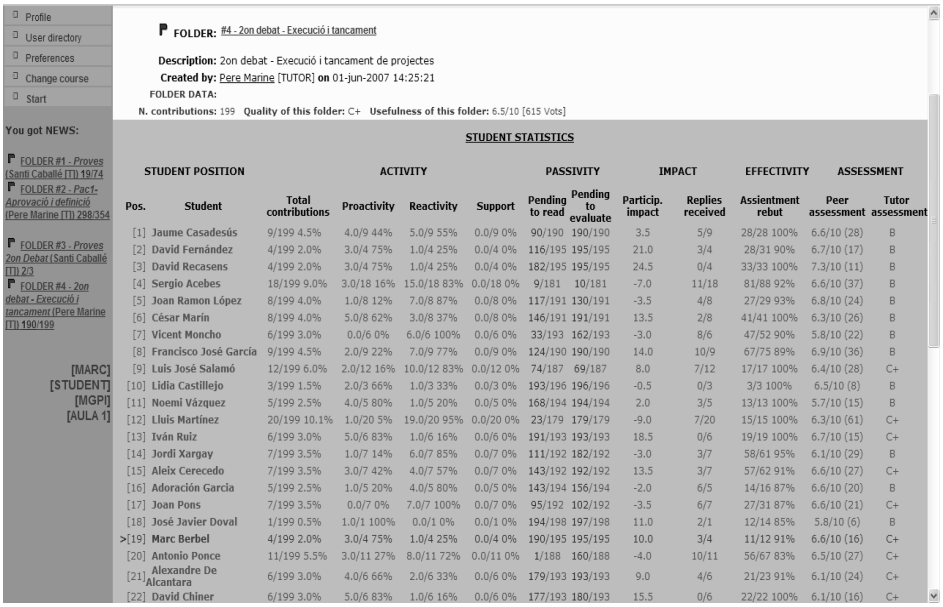

Figure 10. An example of awareness (in the form of flags) and complex feedback (statistics information) constantly presented to students from the interaction information collected, processed and analyzed during a discussion process.

As ongoing work, we plan to improve the implementation of our Grid prototypes by including a more thorough mining process that provides detailed information and deeper knowledge about the learning activity. This, in turn, will require more processing time in comparison to the log processors used and will take more advantage of the high-throughput performance offered by our prototypes.

Acknowledgements

This work has been partially supported by the Spanish MCYT project TSI2005-08225-C07-05.

References

[1] Dillenbourg, P. (ed.) (1999): Collaborative Learning. Cognitive and Computational Approaches. Elsevier Science Ltd. 1-19.

[2] Gutwin, C., Stark, G. and Greenberg, S., Support for Workspace Awareness in Educational Groupware. in Proceedings of the ACM Conference on Computer Supported Collaborative Learning, Bloomington, Indiana, USA October 17-20, 1995.

[3] Zumbach, J., Hillers, A. & Reimann, P. (2003). Supporting Distributed Problem-Based Learning: The Use of Feedback in Online Learning. In T. Roberts (Ed.), Online Collaborative Learning: Theory and Practice pp. 86-103. Hershey, PA: Idea.

[4] Daradoumis, T., Martínez, A. & Xhafa, F. (2006). A Layered Framework for Evaluating Online Collaborative Learning Interactions. International Journal of Human-Computer Studies. Special Issue on "Theoretical and Empirical Advances in Groupware Research", 64 (7), 622-635.

[5] Conklin, J. "Capturing Organization Memory". Groupware 92. David D. Coleman (editor). San Mateo, CA: Morgan Kaufmann Pub. 1992.

[6] Caballé, S., Daradoumis, T., Paniagua, C. and Xhafa, F. (2005) A Grid Approach to Provide Effective Awareness to On-line Collaborative Learning Teams. In: Proc. of the 1st International Kaleidoscope Learning GRID Special Interest Group Workshop on Distributed e-Learning Environments (DLE'05). Napoli (Italy).

[7] Caballé, S., Paniagua, C., Xhafa, F., and Daradoumis, Th., 2005. A Grid-aware Implementation for Providing Effective Feedback to On-line Learning Groups. In: proc. of the GADA'05, Cyprus.

[8] Xhafa, F., Caballé, S., Daradoumis, Th. and Zhou, N. (2004). A Grid-Based Approach for Processing Group Activity Log Files. In: proc. of the GADA'04, Agya Napa, Cyprus.

[9] Foster, I. and Kesselman, C. The Grid: Blueprint for a Future Computing Infrastructure. pp. 15-52. Morgan Kaufmann, San Francisco, CA, 1998.

[10] Paniagua, C., Xhafa, F., Caballé, S., & Daradoumis, T. (2005). A Grid Prototype Implementation for Real Time Processing of Group Activity Log Data in Collaborative Applications. In: proceedings of the 2005 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA (pp. 1177-1183). CSREA Press. ISBN: 1-932415-60-2.

[11] Caballé, S., Daradoumis, Th., Xhafa, F., Esteve J. (2007). Supporting Effective and Useful Web-based Distance Learning. In: proceedings of the Third International Conference on Web Information Systems and Technologies (WEBIST 2007), Barcelona, Spain. INSTICC Press. ISBN: 978-972-8865-79-5. pp. 536-539.

[12] Caballé, S., Daradoumis, Th., Xhafa, F., Esteve J., Barolli, L., Durresi, A., (2007). Using a Grid Platform for Enabling Real Time User Modeling in On-line Campus Proceedings of the CISIS 2007. IEEE Computer Society.

[13] Caballé, S., Xhafa, F., Fernández, R., Daradoumis, Th., (2007). Efficient Enabling of Real Time User Modeling in On-line Campus Proceedings of the UM 2007.LNCS.

[14] Bentley, R., Appelt, W., Busbach. U., Hinrichs, E., Kerr, D., Sikkel, S., Trevor, J. and Woetzel, G. (1997) Basic Support for Cooperative Work on the World Wide Web. Int. J. of Human-Computer Studies 46(6) 827-846.

[15] Appelt, W. (2001): What Groupware Functionality do Users Really Use? In Proceedings of the 9th Euromicro Workshop on PDP 2001, Mantua, February 7-9, 2001. IEEE Computer Society, LosAlamitos.

[16] Stahl, G. (2002) Groupware Goes to School, 8th Int. Workshop on Groupware, CRIWG'02, La Sirena Chile, LNCS, Vol. 2440, pp. 7-24. ISBN: 3-540-44112-3.

[17] Martínez, A., Dimitriadis, Y., Rubia, B., Gómez, E., Garrachón, I. and Marcos, J. A. (2003) Combining qualitative evaluation and social network analysis for the study of classroom social interactions. Workshop "Documenting collaborative interactions" in Computers and Education.

[18] Avouris, N., Komis, V., Fiotakis, F., Margaritis, M., Tselios, N. (2003) On tools for analysis of collaborative problem solving. Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT'03).

[19] Hardings, J. (2003) An XML-based Query Mechanism to Augment Awareness in Computer-integrated classrooms. 11th International Conference on Artificial Intelligence in Education, Australia.

[20] T. Daradoumis, A. Martinez and F. Xhafa: An Integrated Approach for Analysing and Assessing the Performance of Virtual Learning Groups, 10th Int. Workshop on Groupware, CRIWG'04, San Carlos, Costa Rica. Lecture Notes in Computer Science, Vol. 3198, pp. 289-304.
[21] Apache HTTP Server Project: http://httpd.apache.org/ (web page as of November 2007).
[22] Carbó, JM., Mor, E., Minguillón, J. (2005). User Navigational Behavior in e-Learning Virtual Environments. The 2005 IEEE/WIC/ACM International Conference on Web Intel-ligence (WI'05), pp. 243-249.
[23] Common Log Format: http://httpd.apache.org/docs/1.3/logs.html#common (web page as of November 2007).
[24] Goux, J.P., Kulkarni, S., Linderoth, J. and Yoder, M. (2000): An enabling framework for master-worker applications on the computational Grid. In 9th IEEE International Symposium on High Performance Distributed Computing (HPDC'00). IEEE Computer Society.
[25] Master-Worker: http://www.cs.wisc.edu/condor/mw/ (web page as of November 2007).
[26] Globus: http://www.globus.org (web page as of November 2007).
[27] Esteve J., Xhafa F. (2006). Juxta-CAT: A JXTA-based Platform for Distributed Computing. The ACM International Conference on Principles and Practice of Programming in Java'06.
[28] Juxta-CAT: https://juxtacat.dev.java.net/ (web page as of November 2007).
[29] Planetlab: http://www.planet-lab.org (web page as of November 2007).
[30] Apache Axis: http://ws.apache.org/axis/ (web page as of November 2007).
[31] JXTA: http://www.jxta.org/ (web page as of November 2007).
[32] Jakarta Project: http://jakarta.apache.org/ (web page as of November 2007).