Proposing and Evaluating Allocation Algorithms in a Grid Environment

Salvatore Cavalieri, Salvatore Monforte, and Fabio Scibilia

Department of Computer Science and Telecommunications Engineering,
Faculty of Engineering, University of Catania
Viale A.Doria, 6 95125 Catania, Italy
Tel.: +39 095 738 2362, Fax: +39 095 738 2397
{salvatore.cavalieri, salvatore.monforte, fabio.scibilia}@diit.unict.it

Abstract: Distributed computing addresses workload challenges by aggregating and allocating computing resources to provide for unlimited processing power. In the last ten years, it has grown from a concept that would enable organizations to simply balance workloads across heterogeneous computer resources, to an ubiquitous solution that has been embraced by some of the world's leading organizations across multiple industry sectors. And while distributed computing harnesses the full potential of existing computer resources by effectively matching the supply of processing cycles with the demand created by applications, even more importantly it has paved the way for grid computing – a more powerful, yet global approach to resource sharing. The aim of this paper is to propose new allocation algorithms for workload management in a computing grid environment. A simulation tool is used to validate and estimate performance of these algorithms. The paper is organised as follows. After a brief introduction about grid environment and on the DataGrid Project in Section 1, Section 2 presents the proposal for novel allocation policies. Finally, Section 3 provides for a performance evaluation of the algorithms proposed, comparing their performances it with those offered by the actual solution adopted in the DataGrid Project.

1 Introduction

In the last years the amount of data managed and shared by scientific communities has grown very fast. Thus, the needs to create a new platform, allowing the use of intensive computational power as well as data sharing independently on its location, became indispensable. This new concept of informatics infrastructure is known in literature as *Computing Grid*.

The term *grid* is used to indicate the power-network that provides for the distribution of electrical energy supplying any electrical device, wherever it is placed. Similarly, a computing grid is an Internet-based infrastructure that allows different users to access distributed computing and storage resources on a wide area network. The heterogeneity of machines, operating systems, communication protocols and data format representation imply the necessity to study new mechanisms in order to permit a non-ambiguous communication as well as semantic understanding.

P.M.A. Sloot et al. (Eds.): ICCS 2003, LNCS 2659, pp. 1083–1092, 2003.

As an evolution of distributed computing, grid computing allows enormous opportunities for organizations to use computing and storage capabilities from networks of computers spanning multiple geographical boundaries. Grid computing elevates clusters of desktop computers, servers or supercomputers to the next level by connecting multiple clusters over geographically dispersed areas for enhanced collaboration and resource sharing. It is clear that, Grid computing is especially beneficial to the life sciences and research communities, where enormous volumes of data are generated and analysed during any given day. In this context the most improved are physics particles, earth observation and biology applications. Next researcher's generations will be able to compute a great amount of data, from hundreds of terabyte to some petabyte, per year.

Taking into account that, different tasks, with different requirements, are expected to be executed in a computational grid, both traffic management and resource allocation policy are clearly needed in order to improve the efficiency of resources usage and to balance the resources allocation, as well.

Several activities aimed to realise a Grid environment are currently present in literature [1][2]. One of this is relevant with the European DataGrid Project [3].

The DataGrid project was born by the desire of some of the most important science communities of UE like INFN, PPARC etc. Aim of the project is to realise a new computing grid that provides informatics tools and services for different applications enabling organizations to aggregate resources within an entire IT infrastructure no matter where in the world they are located. Figure 1 summarises the structure of the DataGrid Project, highlighting the layers providing different services.

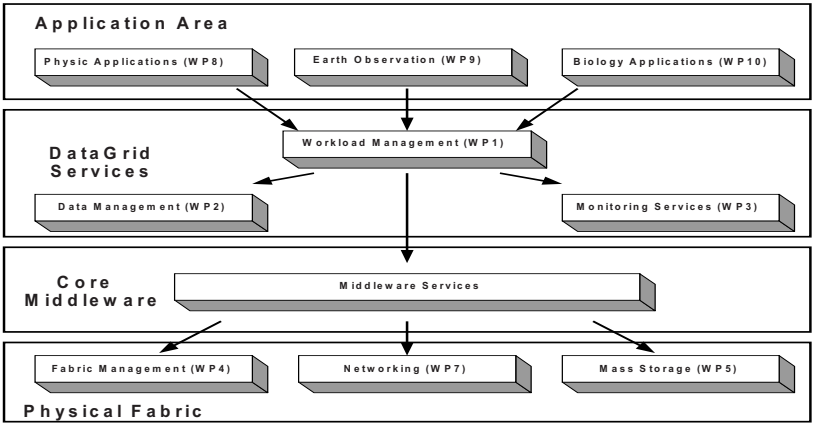

Fig. 1. Grid layers diagram

Lowest layer, *Physical Fabric Layer*, provides management of some infrastructures like computing resources (Fabric Management), mass storage management system (MSMS) and networking (management of quality of services and information collection).

Services concerning security management and information collection are mainly provided for by the *Core Middleware layer*. All these services are based on the Grid

Information Service (GIS), which is the Grid component responsible to collect and publish information about Grid status [4].

Workload and data management as well as monitoring are the most relevant services provided by the *DataGrid Services Layer*.

The *Application Area Layer* represents the upper layer of the architecture and comprises some applications that use DataGrid services for running: biology, particle physics and earth observation.

The DataGrid project is divided in some different activities called *Work Packages* (WPs) according to functional criteria, as can be seen from Figure 1. In particular, the main task of WP1 "Workload Management" concerns the definition of a suitable architecture for workload management, distributed scheduling and resource allocation for realising an optimal allocation of resources in order to improve performance of the Grid. The authors of the paper are currently involved in this WP.

The aim of this paper is to describe in detail the main feature of the Workload Management functionalities, focusing on the allocation strategy here adopted. Then new allocation algorithms will be presented and evaluated. Comparison with the current allocation strategy used in DataGrid will allow pointing out the increase in performance that can be achieved by the adoption of the strategies here proposed.

2 Brokering Policy

Brokering policy allows allocation of jobs submitted by users into resources in a grid environment, aiming to respect users' requirements and to balance the workload. Choices of computing resources where jobs has to be allocated for execution, is a complex task. Many factors can affect the allocation decision and their importance may even differ depending on the job. These factors include: location of input and output data requested by the job, authorisation to use a resource, allocation of resources to the user or to the groups the user belongs to, requirements and preferences of the user, status of the available resources, as well.

The DataGrid component which realises the brokering policy is called the Resource Broker (RB), whose development is actually the core of WP1. The RB interacts with several components. A User Interface (UI) passes all the users' requests of job submission to the RB. A Replica Catalog (RC) is used by the RB to resolve logical data set names as well as to find a preliminary set of computing resource (called Computing Elements or CEs) where the required data are stored. A Grid Information Index (GII) allows the RB to know information about Grid status. Finally, a Job Submission Service (JSS) receives from the RB the job and the name of CE where the job must be submitted.

2.1 State of the Art

The aim of this section is to provide for a description at the state-of-the-art of the matchmaking algorithm performed by the RB in order to select the most suitable Computing Element where a given job must be executed.

It is worth to identify two main different possible scenarios depending on the requirements of the user who submitted the job. It is clear that the simplest possible scenario is when the user specifies the Computing Element where the job has to be executed. In such a case the RB doesn't actually perform any matchmaking algorithm at all, but simply limits its action to the delegation of the submission request to the JSS, for the actual submission. This case will no longer be considered in the analysis undertaken in this paper.

Let's do a little step onwards and let us consider the scenario where the user specifies a job with given execution requirements, but without data files to be processed as inputs. Once the job submission request is received by the RB (through the UI, as said before), it starts the actual matchmaking algorithm to find if the current status of Grid resources matches the job requirements, i.e. to find a suitable CE where to allocate the job.

The matchmaking algorithm consists of two different phases: requirements check and the rank computation. During the requirements check phase the RB contacts the GII in order to highlight a set of CEs compliant with a subset of user requirements, passed to the RB inside the job submission. Examples of these user requirements are the authorisation key, the operating system and the kind and/or the number of processors needed to execute the job. Taking into account that all these requirements are almost constant in time (i.e. it's improbable that a CE changes its operating system or its runtime environment in the very short term, e.g. every half an hour), it is clear that GII represents a good source for testing matches between job requirements and Computing Element features. It is clearly more efficient than contacting each Computing Element to find out the same information.

Once the set of the suitable Computing Elements where the job can be executed is selected, the RB starts performing the second phase of the matchmaking algorithm, which allows the RB to acquire information about the "quality" of the just found suitable Computing Elements: the *ranking phase*. In the ranking phase the RB contacts directly each of the Computing Elements selected in the previous phase, in order to obtain the values of those attributes that appears in a particular rank expression passed by the user to the RB inside the job submission request. Examples of these attributes are the number of free CPUs, the Estimated Traversal Time of the input queue of the CE (i.e. the estimated time interval a job must wait in the input queue before the beginning of its execution), and the amount of free memory. It should be pointed out that conversely to the previous phase, it is better to contact each suitable Computing Element, rather than using the Grid Information Index as source of information, since the rank attributes represents variables varying in time very frequently.

Once the rank expression has been evaluated by the RB, it performs a deterministic choice of the Computing Element basing its decision on the value obtained. Namely, given a list of CEs matching user requirements CE_1, CE_2, ... , CE_n (found by the RB during the requirements check phase) and their correspondent rank values R_1, R_2, ... , R_n (evaluated during the second phase of the matchmaking algorithm), the selected CE where job is really submitted is the one with higher rank value.

Finally, let's consider the scenario where the user specifies a job with given execution requirements, together with the input data files needed by the job during its execution. The main two phases of the match making algorithm performed by the RB

remain unchanged, but the RB has also to interact with the Replica Catalog in order to find out the most suitable Computing Element where the requested input data set is physically stored. If the RB isn't able to find any CE with all the files needed by the job, job submission must be delayed until all the files needed by the job have been transferred to the CE.

Actually, the Estimated Traversal Time is currently assumed by the RB as the default ranking attribute, when the user doesn't specify any rank expression ,and none network parameter, such as file locations or bandwidth, is considered. Thus, Resource Broker decisions are based on CE parameters without considering network performances.

The Job-Burst Problem

As previously said the RB uses services provided by GIS, which collects information about Grid status and maintains this information in the GII. Each Grid resource publishes, periodically, its status. Published values remain unchanged till the expiration of their time-to-live (TTL). All the Grid resources are periodically queried in order to collect all published information and store their values in the GII. It is clear that, due to hierarchical architecture described above, the higher is the destination the older is the information.

Moreover, it is clear that if the job submission rate was higher than the TTL this would lead to inconsistency between the real Grid and its representation by the GII. For the sake of understanding let's consider the scenario where a very huge set of job submission requests, featured by the same user requirements, is issued to the RB. Let's also assume that the rank expression is the default one, i.e. made up only by the Estimated Traversal Time. It is clear that inside a time interval lasting TTL, the matchmaking algorithm performed by the RB will yields the same result and thus the RB will submit the burst of jobs to the same Computing Element for all those requests issued within the same time interval. But submission of the very huge set of jobs to the same CE may cause a very strong increase in the Estimated Traversal Time parameter for that CE, but it will remain unchanged in the GII until the TTL interval expires, leading to a "wrong" allocation choice by the RB. Moreover, to submit the job the RB uses services provided by JSS, that puts the job in queue waiting for input file data transfer and than submits the job to the final CE. This behavior adds new delays to the RB-CE job migration.

2.2 Contribution to the State of the Art

In this section novel ranking policies (ranking expressions) as well as a new criteria used by the RB for choosing the Computing Elements will be presented. The main aim of the proposed ranking policy regards the minimization of job lifetime, which is defined as time spent between submission of the job and its successful completion, including queuing delays as well as input file transfer time.

Let us consider the following ranking expression:

$$\text{rank}(CE, \text{filelist}) = \alpha\ \text{rank\_expr}(CE) + (1-\alpha)\ \text{file\_rank}(CE, \text{filelist}) \qquad (1)$$

where *CE* is the Computing Element candidate for executing the job, *filelist* is the list of input files specified in the submission request and $\alpha \in [0,1]$.

The *rank_expression(CE)* is the rank expression already described. As said before, its deafult value is given by the Estimated Traversal Time, i.e.:

$$\text{rank\_expr(CE)=- EstimatedTraversalTime(CE)} \quad (2)$$

The *file_rank(CE,filelist)* function indicates goodness of the Computing Element CE with respect to both file access and transfer time, and can be expressed as:

$$\text{file\_rank(CE,filelist)=-[} \beta \text{ file\_access\_time + (1-}\beta\text{) file\_transfer\_time]} \quad (3)$$

where $\beta \in [0,1]$.

Values of α e β parameters could be or not specified by user in the classAd during submission phase. In not specified by user default values are used. Default values of α e β parameters will be find by using offline simulation tools in which configuration files describe, with most possible adherence to the reality, current Grid architecture and workload conditions. Their values could be updated by novel simulations periodically (every month, every year ..), or when Grid architecture and/or workload conditions change in a significantly manner.

Let us consider a set of *n* suitable Computing Elements CE_1, CE_2, ... , CE_n and define R_i as the rank for the *i-th* Computing Element. Two different allocation strategies are here proposed.

The first one is very simple and features the choice of the CE with the lower value of the rank given by (1). This strategy will be called *deterministic* one.

Let's define K_i as inverse of absolute value of rank R_i , i.e. $K_i = 1/|R_i|$. According to the second allocation strategy here proposed, the probability that a CE_i is selected for job execution can be given by $P_i=K_i/K_{tot}$ where $K_{tot} = \sum_{i=1}^{n} K_i$. This second strategy will be called *stochastic* one. It associates different probabilities to all CEs that are proportional to their rank values. In this mode is possible to minimise problems derived by presence of *job bursts.* Let us image that a burst of 100 jobs is submitted to Resource Broker. With deterministic choice criteria all job submission requests issued within a time interval featured by a duration equal to TTL, would be submitted to the same ComputingElement. On the other hand, using stochastic choice criteria, jobs would be distributed to all the suitable CEs, according to their probability that the value given by the evaluated rank expression yields.

2.3 Evaluation of Brokering Policy

In order to evaluate the proposed brokering policies described above, a simulation tool was purposely developed in C++, which models in a very detailed manner Grid components and their interactions. The simulator engine is event driven. Each event is characterized by a timestamp, which is used to sort event list with growing time

criteria, by a message, which identifies the kind of message and the actions to do, and by an owner, which is the generator and handler of the event.

The network infrastructure of the Grid is the GARR-B net used by Italian Science Communities [5]. Network parameters used as input for simulation were measured directly on the real networks using services provided by GARR-B routers, and thus they are fixed for all simulations.

Life cycle of a job is evaluated by the simulator. As said before, life cycle begins in a User Interface that issues job submission requests to a Resource Broker as shown in Figure 2.

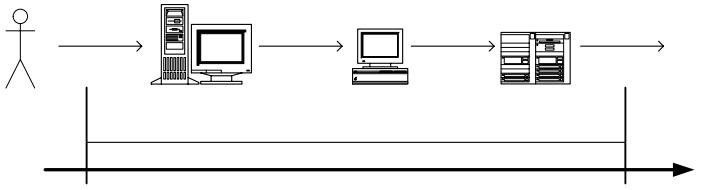

Fig. 2. Job life cycle

Interarrival time between two consecutive job submissions follows the negative exponential distribution. Moreover, the User Interface submits continuously several instances of jobs of the same kind. Therefore, a simulation is both related to a given workload, which is specified by number of average submissions per second, and characterized by a unique kind of job submissions.

In the following sub sections results obtained by several simulations will be presented.

Jobs with Light Execution Time

A first simulation set regards job submission requests featured by a light execution time, which is comparable with both file access and file transfer time. In particular, it's assumed that each job is featured by an estimated execution time given by $10^5 / CE_{MIPS}$ sec (CE_{MIPS} is the number of MIPS featured by the CE) and requires, for execution, two files as input data set whose size are 1GB and 50MB, respectively. The overall computing power of the grid consists of sixteen CEs spread with different computational power: 300 MIPS (high), 200 MIPS (medium) and 100 MIPS (low).

The first scenario is characterised by a computational power uniformly distributed all over the Grid. The same apply for the sizes of files. Thus, this scenario can be considered as a balanced case.

Figure 3a shows a direct comparison between current brokering policy and the proposed brokering policy with both deterministic and stochastic choice criteria, varying workload conditions. Values of parameter α and β are, respectively, 0.75 (high importance to EstimatedTraversalTime of the candidate CE) and 0.25 (low importance to file_access_time respect with file_transfer_time). These values have been selected after several simulations, as they are able to give the best performances.

On X-axis average interarrival time (seconds) between two job submissions is represented, thus the left side of the diagram presents values relevant to the highest workload conditions. On Y-axis average job life time of submitted jobs is represented.

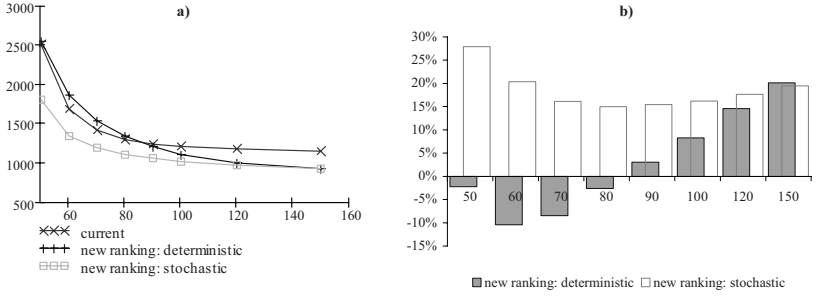

Fig. 3. Case of light jobs in a balanced scenario

Figure 3b depicts the percentage improvement of job-lifetime obtained using the two proposed policies varying workload conditions, with respect to current scheduling policy. New ranking policy with deterministic choice criteria improves performances, respect with current scheduling policy, for interarrival submission time greater than 80 seconds (lower workload conditions). New ranking policy with stochastic choice criteria improves performance for any workload conditions, especially for high workload conditions where the job burst problem occurs. For lower workload conditions new deterministic policy gives better results than current policy, cause the first considers network parameters, such as both file transfer and file access time, that are not used in the latter one. Thus, Grid performances are sensitive to network conditions.

Additionally, a second set of simulations was carried out. The characteristics of job submission requests are the same of the first set of simulations described above. The two sets of simulations only differ in the distribution of computational and storage resources. Namely, the higher the computational power of a site, the larger the size of the file neighbouring to such a site. Thus, this scenario could be considered such as a worst case just to evaluate behavior of novel scheduling policies.

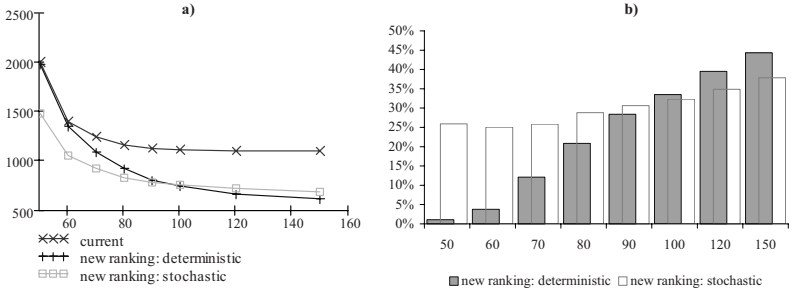

Fig. 4. Case of light jobs in an unbalanced scenario

Figure 4a shows a direct comparison between the three brokering policies in this case.

Referring to the previous figure, it is possible to notice that again both proposed brokering policies, the one with deterministic choice criteria and the one with stochastic choice criteria, give results that are better than current brokering policy for all workload conditions. Lower is grid workload higher is improvement of both new scheduling policies.

Intensive Computational Jobs

Another set of simulations regards job submission requests featured by a heavy execution time, in which both file access and transfer time are negligible when compared with job execution time. Therefore, the meaningful parameter for brokering phase is the computational power of the eligible CEs. Each job is featured by an estimated execution time given by $10^7 / CE_{MIPS}$ sec and requires, for execution, two files as input data set whose size are 1MB and 300KB, respectively. The overall computing power of the simulated grid consists of five sites having a number of CPUs between 20 to 30 and featured by a computational power uniformly distributed in the interval [100, 300] MIPS.

Graphics shown in this section refer to simulation having α=0,25 and β=0,25. Again these values have been chosen after several simulations, as they allow achieving the best performance.

Figure 5a shows a direct comparison between the different policies in the scenario described above. Grid performances are improved by both proposed brokering policy. In this case improvement is less than other cases described in previous sections cause the job burst problem doesn't occur for two different considerations described below.

Average job execution time is higher than previous cases. Thus, Grid resources reach saturation with an higher average interarrival job submission time (720 seconds vs 50 seconds of previous cases). Therefore the rate of job bursts is very low.

JSS queuing time is lower than previous cases due to a lower input file transfer. Thus, time spent between the election of the destination CE and the following job submission through JSS is lower than previous cases.

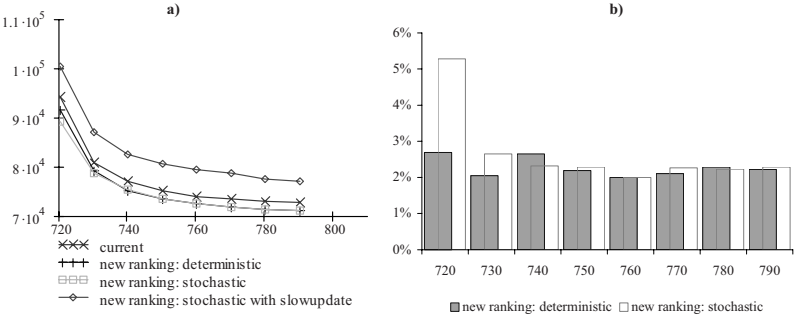

Fig. 5. Case of intensive computational jobs

Figure 5b, as said before, depicts the percentage improvement of job-lifetime obtained using the proposed policy varying workload conditions, with respect to current scheduling policy.

In this scenario, where no overhead due to file transfer is present, the job-burst problem, highlighted in a previous sub-section, plays a very heavy impact in the performance of the system. In fact, in this case, when a job submission request is received by the RB, once this last has chosen the best CE for the job, it can immediately pass the job to the JSS in order to be submitted to the CE. So, the submission of the job is immediate because no file transfer is needed (it's important to remember that file transfer requires a lot of time when files are very huge and the number of files needed is great, too). This means that a very high job submission rate to a particular CE may cause a very sudden increase in the Estimated Traversal Time. If the updating rate of the GII is not enough high, the job-burst problem described before arises. In order to evaluate the impact of the presence of this problem onto the performance of the system, a set of simulations has been carried out. Two different refresh rates of the Grid Information Index were considered. The first one was fixed equal to the job submission frequency. The other was set equal to 1/10 of this last frequency. In Figure 5a a new curve labelled "stochastic with slow update" refers to the stochastic allocation strategy used by the RB in the case the frequency of job submission is ten times greater that the updating frequency of the GIS. As expected, this figure confirms that, due to the inconsistencies arising, the slower the GII is refreshed the worst the average job life time is.

4 Conclusions

This paper has given a state of the art of one the current project aimed to define a Computational Grid, the DataGrid European Project. After a detailed description of the allocation strategy here adopted, new allocation algorithms were proposed and studied. It was pointed out that an allocation policy based on stochastic choices could improve grid performances.

Reference

[1] http://www.globus.org/
[2] http://www.cs.wisc.edu/condor/
[3] http://eu-datagrid.web.cern.ch/eu-datagrid/
[4] Grid Information Services for Distributed Resource Sharing. K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman. Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press, August 2001.
[5] http://www.garr.net/