AMBLE: An Awareness Model for Balancing the Load in collaborative grid Environments

Pilar Herrero[#1], Jose Luis Bosque[*2], Manuel Salvadores[#3], María S. Pérez[#4]

[#]*Facultad de Informática. Universidad Poltécnica de Madrid*
Campus de Montegancedo S/N. 28.660 Boadilla del Monte. Madrid. Spain
[1]pherrero@fi.upm.es
msalvadores@zipi.fi.upm.es
[1]mperez@fi.upm.es

[*]*Escuela Superior de Ciencias Experimentales y Tecnología*
C/ Tulipán S/N. 28.933 Móstoles, Madrid. Spain
[2]joseluis.bosque@urjc.es

***Abstract*— In this paper, we present a new extension and reinterpretation of one of the most successful models of awareness in Computer Supported Cooperative Work (CSCW), called the Spatial Model of Interaction (SMI), which manage awareness of interaction through a set of key concepts, to manage task delivery in collaborative distributed systems. This model also applies some theoretical principles and theories of multi-agents systems to create a collaborative and cooperative environment that can be able to provide an autonomous, efficient and independent management of the amount of resources available in a Grid environment. This model has been implemented using web services and some experimental results carried out over a real and heterogeneous grid are presented with the end of emphasizing the performance speedup of the system using the AMBLE model.**

I. Introduction

Grid computing intends to share heterogeneous resources in dynamic environments. The complexity of achieving this goal is caused by several factors, being the existence of different virtual organizations, the dynamism of the underlying architecture and the heterogeneity of the involved resources some of the most challenging aspects.

With the aim of providing better capabilities on a grid, it is essential to use a resource manager, which will take the suitable, and complex, decision about the allocation of processes to the resources of the system. The resource management includes other tasks, such as resources discovery, resources registration and monitoring. As expected results, the resource manager should achieve load balancing within the grid. Equilibrating the amount of work assigned to each node in a grid is a complex problem, even more than for other kinds of parallel systems. Even though load balancing has received a considerable amount of interest, it is still not definitely solved [19, 22]. Nevertheless, this problem is central for minimizing the applications' response time and optimizing the exploitation of resources, avoiding overloading some processors while others are idle. Grids present additional challenges, since they can easily become heterogeneous, requiring load distributions that take into consideration each node's computational features as well as the services that each node offers to the grid. In order to provide flexible and efficient load balancing mechanisms, new technologies could be applied. This field can take advantage of advances of other disciplines, which have been satisfyingly applied to other domains.

This is the case of the multi-agent systems, an already mature technology, which offers promising features to resource managers. The reactivity, proactivity and autonomy, as essential properties of agents, can help in the complex task of managing resources in dynamic and changing environments. Additionally, the cooperation among agents, which interchange information and resources status, allows load balancing mechanisms to be performed and efficiently deployed on a grid.

In this paper, we present a new extension and reinterpretation of the Spatial Model of Interaction (SMI), an abstract awareness model designed to manage awareness of interaction, in cooperative applications, through a set of key concepts. The SMI is based on a set of key concepts which are abstract and open enough as to be reinterpreted in many other contexts with very different meanings [12]. Thus, this paper presents a new reinterptetation of this model, and its key concepts, in the context of an asynchronous collaboration in grid environments.

This reinterpretation, open and flexible enough, merges all the OGSA [7] features with theoretical principles and theories of multi-agents systems, to create a collaborative and cooperative grid environment. Following one of the main OGSA characteristics, the use of open, standard and public interfaces, we have implemented AMBLE as a web service specification to manage awareness in collaborative grid environments, WS-AMBLE. This specification provides an open interface having the ability of managing different levels of awareness, allowing different Virtual Organizations to share computational resources based on open protocols and interfaces. As far as we know, none of the last

WS specifications offers functionalities useful enough as to create awareness models and none of the last WS specifications offers specific functionalities to manage task balancing delivery in collaborative grid environments.

II. Related work

Though the master-slave paradigm is hardly scalable and not fails tolerance, it seems to be very useful in distributed systems to achieve a good load balancing strategy. However, in grids other alternatives should be taken into account. In fact, the Global Grid Forum (GGF) [11] recommends, if needed, the use of hierarchical master-slave.

In [10], authors apply reinforcement learning to adaptive load balancing for allocating resources in a grid in an efficient way. As result of this work, it is shown that the reinforcement learning can improve the process of resource allocation in large heterogeneous systems, and more specifically, in grids. In [4], an agent-based grid management infrastructure together with a task scheduler is performed for local grid load balancing. In [5], a communication-based load balancing algorithm, named Comet, is shown. The model is based on two policies: the selection policy, that is, which agent or task is migrated and the location policy, that is, to which destination node the selected agent is migrated to.

The negotiation takes an important role in agent systems. Four different negotiation models are studied in [18] for agent-based load balancing and grid computing: contract net protocol, auction model, game theory based model and discrete optimal control model. The interaction between grids and agents challenges has been clearly defined by Foster et al in [8]. Among others, the most important lines of research identified for the overlapping of both fields are: autonomous services, dynamic and statefull services [9], negotiation and Service Level Agreements (SLA) [13], Virtual Organization Management. [15] and security. [17]. As example of the successful combination of grid and agents, a real grid system has been built by means of mobile agent technology, SMAGrid, Strong-Mobile Agent-Based Grid [21] is composed of multiple agents used for assigning resources to tasks.

.

III. The spatial model of interaction (SMI)

The Spatial Model of Interaction, defined for application to any Computer Supported Cooperative Work (CSCW) system where a spatial metric can be identified, has been driven by a number of objectives [3]:

- Scalability: It is based on the concept of *aura*. Each object has an aura for each medium in which it can interact, because the aura defines the volume of space within which this interaction is possible.
- Interactions: The SMI assumes a space populated by potentially communicating objects that may represent anything. The SMI provides a framework for objects in the environment to manage their interaction, and communication between every pair of objects.

The model itself defines five linked concepts: medium, focus, nimbus, aura and awareness.

Medium: A prerequisite for useful communication is that two objects have a compatible medium in which both objects can communicate.

Aura: In 1992, Fahlén and Bowers defined *aura* as the sub-space which effectively bounds the presence of an object within a given medium and which acts as an enabler of potential interaction [6].

In each particular medium, it is possible to delimit the observing object's interest. This idea was introduced by S. Benford in 1993, and it was called *Focus*. In the same way, it is possible to represent the observed object's projection in a particular medium, called *Nimbus*.

Awareness: It quantifies the degree, nature or quality of interaction between two objects. One object's awareness of another object quantifies the subjective importance or relevance of that object. Awareness between objects in a given medium is manipulated via *Focus* and *Nimbus*, requiring a negotiation process. Considering, for example, A's awareness of B, the negotiation process combines the observer's (A's) focus and the observed's (B's) nimbus. For a simple discrete model of focus and nimbus, there are three possible classifications of awareness values when two objects are negotiating unidirectional awareness [12]:

- *Full awareness:* object B is inside A's focus and object A is inside B's nimbus.
- *Peripheral awareness:* object B is outside A's focus but object A is inside B's nimbus, or object B is inside A's focus but object A is outside B's nimbus.
- *No awareness:* An object A has no awareness of object B in a medium M when object B is outside A's focus and object A is outside B's nimbus.

IV. AMBLE: Reinterpreting the Key Awareness Concepts

Let's consider a system containing a set of nodes $\{n_i\}$ and a task T that requires a set of processes to be solved in the system. Each of these processes necessitates some specifics requirements, being r_i the set of requirements associated to the process p_i, and therefore each of the processes will be identified by the tuple (p_i, r_i) and T could be described as $T = \sum_i \{(p_i, r_i)\}$

The AMBLE model takes into account that one of the major goals of grid computing is increase the collaboration capabilities of the system to start by a simple, abstract and preliminary interpretation of the SMI key concepts in the context of an asynchronous collaboration. Thus the AMBLE model, proposes an awareness infrastructure based on these concepts capable of managing the load management of grid

environments. This model reinterprets the SMI key concepts as follow:

Focus: It can is interpreted as the subset of the space on which the user has focused his attention with the aim of interacting with. It can be related to both tasks in grid environments. and latency in cluster computing. Regarding tasks, and given a node n_i in the system requiring the execution of a given task (T), the focus of this node would contain, at least, the subset of nodes that are composing the Virtual Organization (VO) in which this node is involved.

$$\text{Focus} : \text{Node} \rightarrow \text{System}$$
$$n_i \rightarrow \{n_j\}$$

The focus will be delimited by the *Aura* of the node in the system.

Nimbus: It is defined as a tuple (Nimbus=(*NimbusState* ,*NimbusSpace*)) containing information about: (a) the load of the system in a given time (*NimbusState*); (b) the subset of the space in which a given node projects its presence (*NimbusSpace*).

As for the *NimbusState*, this concept will depend on the processor characteristics as well as on the load of the system in a given time. In this way, the *NimbusState* could have three possible values: *Null*, *Medium* or *Maximum*. If the load of a given node is not high, and this node could receive some processes, its *NimbusState* will get the *Maximum* value. If a given node n_i can accept, at least, a process, then the *NimbusState* of this node n_i would be *Medium*. Finally, if the node n_i is overload its nimbus would be *Null*. The *NimbusSpace* will determine those machines that could be taking into account in the tasks assignment process. The *NimbusSpace* will be delimited by the *Aura* of the node in the system.

Awareness of Interaction (AwareInt):

This concept will quantify the degree, nature or quality of asynchronous interaction between distributed resources. Following the awareness classification introduced by Greenhalgh in [12], this awareness could be *Full*, *Peripheral* or *Null*.

$$AwareInt(n_i, n_j) = Full$$
$$n_j \in Focus(\{n_i\}) \quad \wedge \quad n_i \in Nimbus(n_j)$$

Peripheral aware of interaction if

$$AwareInt(n_i, n_j) = Peripheral$$
$$n_j \in Focus(\{n_i\}) \quad \wedge \quad n_i \notin Nimbus(n_j)$$

or

$$n_j \notin Focus(\{n_i\}) \quad \wedge \quad n_i \in Nimbus(n_j)$$

The AMBLE model is more than a reinterpretation of the SMI, it extends the SMI to introduce some new concepts such us:

Interactive Pool:

This function returns the set of nodes { n_j }interacting with the n_i node in a given moment. Given a System and given a task T to be executed in the node n_i

Task Resolution:

This function determines if there is a service (s_i) in the node n_i, being NimbusState(n_i)/=Null, such that could be useful to execute the task T (or at least one of its processes).

$$n_i = \sum_i \{s_i\} \quad Task\ \text{Re}solution: \quad Node \times Task \rightarrow Task$$
$$n_i \times T \rightarrow \{(p_i, s)\}$$

Where s is the "score" to execute p_i in n_i node, being its value within the range [0, ∞): 0 if the node n_i fulfils the all the minimum requirements to execute the process p_i; the higher is the surplus over these requirements, the higher will be the value of this score.

Collaborative Organization:

This function will take into account the set of nodes determined by the *Interactive Pool* function and will return those nodes of the System in which it is more suitable to execute the task T. This selection will be made by means of the *TaskResolution* function.

IV. LOAD BALANCING ALGORITHM IN AMBLE)

Generally a dynamic load balancing algorithm consists of four policies: a load measurement rule, an information exchange rule, an initiation rule and a load balancing operations [20]. In this section we will introduce the load balancing algorithm as it has been introduced in the AMBLE awareness model, and how it will be applied to our distributed and collaborative multi-agent architecture in grid environments. The main characteristics of this algorithm are that it is dynamic, distributed, global and take into account the system heterogeneity [1].

A. State Measurement Rule

This local rule will be in charge of getting information about the computational capabilities of the node in the system. This information, quantified by a load index, provides aware of the NimbusState of the node. This dynamic index should be periodically and frequently measured, and should be a good estimation of a node computing capabilities. The choice of a load index has a huge impact on load balancing efficiency [14].

The load index calculation is performed by the benchmark agent. In this paper the load index is evaluated based on two parameters:

- Node computational power (P), which depends on the node computational architecture, and takes into account CPU, memory and I/O features. It is a static parameter.
- The CPU assignment which is defined as the percentage of CPU time that would be available to a newly created task, on a specific node. It will be working as a dynamic parameter.

The benchmark agent will implement the load measurement rule, measuring periodically the needed parameters and evaluating the load-index of every node "n_i", belonging to the grid, based on the following formula:

$load\text{-}index_i = p_i / (p_{max} * np_i)$

where

* p_i: represents n_i's linpack
* p_{max} : represents the linpack of the best node
* np_i: represents the number of processes that are being executing in the node "n_i" at a given moment.

If *NinbusCal* function returns a value close to cero is because the load of the node is very high (or its performance is very low). The closer to 1 this value is, the lower is the load of the node (or the higher is its performance). The NimbusState of the node will be determined by the load index and it will depend on the node capacity at a given time. This state determines if the node could execute more (local or remotes) tasks. Its possible values would be:

- *Maximum*: The load index is low and therefore this infrautilized node will execute all the local tasks, accepting all new remote execution requests coming from other nodes.
- *Medium*: The load index has an intermediate value and therefore the node will execute all the local tasks, interfering in load balancing operations only if there are not other nodes whose NimbuState would be Maximun in the system.
- *Null*: The load index has a high value and therefore the node is overload.

B. *Information Exchange Rule*

The knowledge of the global state of the system will be determined by a policy on the information exchange. This policy should keep the information coherence without overloading the network with an excessive number of unnecessary messages.

An optimum information exchange rule for the AMBLE model should be based on events [2]. This rule only collects information when a change in the NiImbus (in the *NimbusState* or in the *NimbusSpace* or in both) of the nodes is made. If later, the node that has modified its nimbus will be in charge of notifying this modification to the rest of the nodes in the system, avoiding thus synchronisation points. The information that every node has about the *NimbusState* of the rest of the nodes is stored in a local data structure, which is updated while the node receives information messages from the others.

C. *Initiation Rule*

The initiation rule determines when to begin a new load balancing operation. As the model implements a non user interruption algorithm, the selection of the node must be made just before sending the task execution. Once the execution of the process starts in a specific node it would have to finish in the same node.

D. *Load Balancing Operation*

Once the node has made the decision of starting a new load balancing operation, this operation will be divided in another three different rules, presented in the following sections.

Localization Rule: Given a task to be executed in the node n_i, the localization rule has to determine which nodes are involved in the *CollaborativeOrganization* of the node n_i. In order to make it possible, firstly, the AMBLE model will need to determine the awareness of interaction of this node with those nodes inside its focus. Those nodes whose awareness of interaction with n_i was Full will be part of the Interactive Pool of n_i to solve the task T, and from that pre-selection the TaskResolution method will determine those nodes that are suitable to solve efficiently the task in the environment.

Selection and Distribution Rule: This algorithm joins selection and distribution rules because it determines which nodes will be in charge of executing each of the processes making up the T task. The proposed algorithm takes into account the *NimbusState* of each of the nodes as well as the *TaskResolution* to solve any of the T's processes.

The goal of this algorithm is to find the more equilibrate assignment of processes to computational nodes, based on a set of heuristics. Firstly, a complete distribution is made taking into account all the processes making up the T task as well as all the computational nodes implicated in the *CollaborativeOrganization.* If, in this first turn, it would be possible to assign all the process the algorithm would have finished. Otherwise, it would be necessary to calculate, again, the *NimbusState* of the nodes belonging to the *CollaborativeOrganization*, repeating the complete process again. The sequence of steps that implements the assignment heuristic is:

1. If any of the processes $p_i \in T$ could just be executed in one of the n_j nodes, this process would be automatically assigned to this node.
2. Among all the remaining nodes, and for each and every process, the node whose score was higher will be selected to execute the corresponding process.
3. Once a T process had been assigned to every node inside the pool, a message will be sent to each of these nodes requiring the execution of the designated processes. If the remote nodes accept the execution, the process would be definitely assigned, and, they should evaluate again its *NimbusState* returning its value to the origin node. However, if the candidate node rejects the process execution, the origin node should look for a new candidate.

4. Repeat the previous steps until all the processes had been assigned or until all the nodes in *CollaborativeOrganization* will get *NimbusState = Null*.

Once all the nodes in *CollaborativeOrganization* had achivied a *NimbusState = Null* or once it would be impossible to find a candidate in the *CollaborativeOrganization* to execute any of the T's process, the task will be waiting in n_i for a change in the state of any of the nodes.

V. AMBLE EVALUATION ARCHITECTURE

As it is possible to appreciate in the figure 1 the middleware architecture of load balancing model has been separated in three different parts:

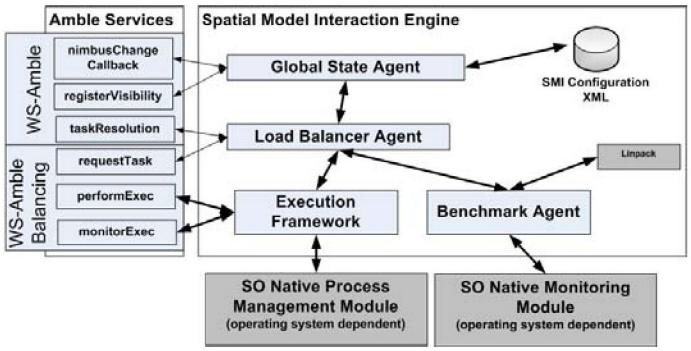

Fig. 1 The Load Balancing Model: Middleware Architecture

- **SMI-Engine (Spatial Model of Interaction Engine)**: This is the main core of the architecture and contains those components that implement all the logic of the SMI and the load balancing algorithms explained in section IV. This engine is made up by the following modules:
  - Benchmark Agent: This agent based on the Linpack benchmark [26], maintains a performance measure of the node which could be evaluated from the Load Balancer.
  - Global State Agent: This agent compiles all the information related to the two main concepts of the AMBLE, providing information about those nodes that are available in the system.
  - Load Balancer Agent: This module implements the logic for the load-balancing operation and makes the final decision of which node will execute each process.
  - Execution framework: This interface contains the modules dependent on the operating system (OS) to access to the process management APIs.
  - SO Native Process Management: It depends on the OS and uses those functionalities that the APIs of this OS offer to supply the to process execution.
  - SO Native Monitoring Module: This module also depends on the OS and uses those functionalities that the APIs of this OS offer to monitor the state of the computer.
- **AMBLE-Service:** Web service [23, 24] interface that provides those methods necessaries to establish the communication between nodes through SOAP [25] messages.
  - registerVisibility: When a node detects a new resource inside its focus, it invokes this operation. Moreover, if the observer node is also inside the observed NimbusSpace, it would be included in the awareness of interaction record with a value equal to Full.
  - nimbusChangeCallback: This operation receives the changes that a specific node has on its NimbusState.
  - requestTask: This method is invoked by a client requiring the execution of the T task composed by n processes.
  - taskResolution: This method is invoked by a node requiring "offers/scores" for the processes associated to a specific task.
  - performTask: This method is invoked to order the process execution once the process has been assigned to a particular node.
  - monitorExec: This operation is used to monitor the state of execution of a process in an identifiable node.

VI. AMBLE EXPERIMENTAL RESULTS

This section represents a set of experiments with the following objectives: corroborate if the AMBLE's tasks delivery works in a real and heterogeneous grid environment; detect the overload introduced by the AMBLE's model in a real environment; and measure the AMBLE's speedup in different scenarios.

A. The grid environment infrastructure

The tests presented in this manuscript have been evaluated in a real and heterogeneous services oriented grid environment. The system heterogeneity is reflected no just in the architecture of the computational nodes, but also in the OS utilized. The grid infrastructure was made up for 20 nodes with the following characteristics: 8 of them were Intel Centrino P4 1.5 GHZ with 0.5 GB of memory (in this paper we will refer to them as "Type A"), 11 of them were Intel P4 3.0 GHz (B) with 1GB of memory (in this paper we will refer to them as "Type B") and the last one was an Intel Xeon 3.6 GHz (C) with 4GB of memory (in this paper we will refer to it as "Type C").

In order to carry out the model evaluation we have selected a set of CPU-intensive tests based in iterations over the Linpack benchmark [26].

The following subsection presents three different scenarios raised to make this evaluation possible. In each of these scenarios, there is a task T, composed of 20 processes, to be

executed and a node N that receives the T execution request. Each of these scenarios also presents 4 different tests; each of them differs to the others in the size of the processes to be executed.

The table I presents the response times, in seconds, for each of the tests executed in the different grid nodes.

The metric used to have a measure of the AMBLE performance will be the response time achieved for each of the tests in the proposed scenarios. These measures will allow calculating the speedup as well as the communication overhead introduced by the AMBLE algorithm in the system [16].

TABLE I
RESPONSE TIME

	Type A	Type B	Type C
Test 1	4,95	3,86	3,48
Test 2	24,02	19,20	17,80
Test 3	47,00	38,72	34,97
Test 4	232,93	192,30	175,03

B. Experimental Results

1) Scenario A

This scenario describes the ideal conditions for the model. As it was mentioned above, the node N receives the T execution request. The N node has full awareness of interaction with the rest of the nodes making up the grid, and therefore this node throws a load balancing operation to carry out the task execution taking into account all the nodes composing the grid. Table II presents the total response times of the system using the AMBLE implementation as well as the speedup (sequential time/parallel time) of the AMBLE model related to each of the types of nodes involved in the grid

TABLE II
GLOBAL COMMUNICATION OVERHEAD AND SPEEDUP RELATED TO THE TYPES NODES FOR SCENARIO A

Amble	Communication overhead	Speed up vs A	Speed up vs B	Speed up vs C
5,9	5,65	0,84	0,65	0,59
6,964	5,76	3,45	2,76	2,56
8,233	5,88	5,71	4,70	4,25
17,69	5,92	13,17	10,87	9,89

2) Scenario B

This scenario raises the non ideal situation in which all the nodes in the grid are been infrautilised, but they are located in different auras. The grid client requests the execution of one of the task in the node N. This node has 10 more nodes inside its aura with a distance equal to 1 and the nine remaining in another aura with a distance equal to 2. The table III presents the speedup of the AMBLE model related to the local execution on each of the types of nodes as well as the communication overhead.

TABLE III
GLOBAL COMMUNICATION OVERHEAD AND SPEEDUP RELATED TO THE TYPES NODES FOR SCENARIO B

Amble	Communication overhead	Speed up vs A	Speed up vs B	Speed up vs C
6,54	6,29	0,76	0,59	0,53
7,45	6,25	3,22	2,58	2,39
9,906	7,56	4,74	3,91	3,53
19,28	7,51	12,08	9,97	9,07

3) Scenario C

This same scenario also raises the non ideal situation in which all the nodes in the grid are infrautilised but they are located in different auras. The grid client requests the execution of one of the task in the node N. This node has 10 more nodes inside its aura with a distance equal to 1 (aura1) and the nine remaining in another aura with a distance equal to 2 (aura2). However, in this situation half of the nodes that are inside the aura1 reject the execution of the processes assigned. Then, the load balancing algorithm increases the aura2 and therefore the other 9 remaining nodes could accept any of the processes that are looking for a location. The task delivery process is done among the nodes located in the aura2. While this distribution is been done, some of the nodes that are located inside the aura1 change their NimbusState and they could received new processes. The system will inform of this change and those processes that were not assigned among the nodes located in the aura2, will be assigned among all those nodes changed its NimbusState in the aura1. The table IV shows the speedup and the communication overhead.

TABLE IV
GLOBAL COMMUNICATION OVERHEAD AND SPEEDUP RELATED TO THE TYPES NODES FOR SCENARIO C

Amble	Communication overhead	Speed up vs A	Speed up vs B	Speed up vs C
10,92	10,68	-7,44	0,45	0,35
12,08	10,88	5,72	1,99	1,59
14,13	11,78	20,84	3,33	2,74
24,29	12,52	150,74	9,59	7,92

C. Analysis of results

Generally speaking, it would be possible to conclude that the experimental results obtained in the tests above described are very successful and corroborate the usefulness of the AMBLE model as to be applied to work-load balancing

operations in real heterogeneous grid environments. As for some more specific details (see tables II, III and IV) the performance improvements obtained by using this model are excellent in all the scenarios and in, almost, all the tests. It is worthy to highlight that the results achieved in the test1 in which the model get worse experimental results. These results are consequence to the small size of the task to be executed, which provokes a considerable communication overhead in this delivery operation, increasing the response times and making that these times were higher that the one associated to the local execution. Due to this fact, the speedup of these experiments are lower than 1. An important conclusion could be taken out from these results: if the processes to be process have a response time lower than the communication overhead), it would be preferable to carry out the local execution of the task instead of distributing the processes along the grid.

The results corresponding to the other tests show that the speedup experiment an important improvement and, in general, it would be possible to conclude that, the bigger is the size of the problem to be solved the better are the results achieved. The communication overhead is the factor limiting the performance increase. This overhead is independent of the problem size. In this way, when the problem size increases, the parallelizable portion of the task also increases and therefore the speedup experiment a considerable improvement.As for the scenarios presented in this paper, the scenario A gets the best speedup results, related to the local execution. This is a consequence of the ideal conditions, for the execution of the AMBLE model, in which this scenario takes place. The scenario B presents an increment in the aura, and in scenario C there are some modifications on the NimbusState of some of the nodes. These situations imply that the load balancing model will require a set of messages to carry out the delivery operation and this communication overhead is reflected in the speedup results. However, in spite of this additional communication overhead the results are still very successful.

VII. CONCLUSIONS

Equilibrating the amount of work assigned to each of the nodes in a grid environment is a complex problem, even more than for other kinds of parallel systems. This balance has to take into account not just the computational capabilities of each of nodes involved in the balancing operation but also the users' confidence in the other nodes and users as well as its collaborative and cooperative intentions. Context should define a set of common information about the current status of each of the nodes in the grid and its capability of collaborating with peers, and this context could come from an awareness model.

This paper presents an awareness model for balancing the load in collaborative grid environments, AMBLE, in a collaborative multi-agent system. This model extends and reinterprets some of the key concepts of the most successful models of awareness in Computer Supported Cooperative Work (CSCW), called the Spatial Model of Interaction (SMI). AMBLE manages the interaction in the environment allowing the autonomous, efficient and independent task allocation in the environment. The AMBLE implementation is based on the Web Services specifications and follows one of the key principles in the grid theory: the use of open, standard and public interfaces.

This model has been evaluated in a real and heterogeneous grid infrastructure. Different scenarios were designed for this purpose. Each of these scenarios was also composed of a set of different computational-intensive tests based in iterations over the Linpack benchmark. These scenarios were designed in such way that each of them introduced some additional handicaps to the previous one. The most important conclusions that could be extracted from the experimental results presented in this paper are: Firstly, the AMBLE model can contribute to the performance of heterogeneous systems by distributing the work-load in an equilibrating way among all the nodes composing the grid; Secondly, the communication overhead in a grid environment is a factor to be considered due to the remarkable limitations in the performance improvements. This overhead doesn't depend on the problem size, it mainly depends on the dynamism of the grid system, in each and every moment, and therefore it can not be predict beforehand. Finally, it is important to highlight that the size of the processes, to be distributed in the grid, has a fundamental impact in the global performance of the system. Those processes whose response time is low, are not suitable to be distributed in the grid because the communication overhead could be bigger that the local response time, entailing a worsening of the system.

ACKNOWLEDGMENT

This work has been partially founded by the Spanish Ministry of Education and Science (grant TIC2003-08933-C02) and Government of the Community of Madrid (grants GR/SAL/0940/2004 and S-0505/DPI/0235).

REFERENCES

[1] M. Beltrán, A. Guzman, J. L. Bosque. *Dynamic tasks assignment for real heterogeneous clusters*. Parallel Processing and Applied Mathematics: 5th International Conference, PPAM 2003. Lectures Notes in Computer Science. Springer-Verlag. Vol: 3019 / 2004. pp 888 – 895. Abril 2004.

[2] M. Beltrán, J. L. Bosque, A. Guzmán. *Resource Disseminatioin policies on Grids*. On the Move to Meaningful Internet Systems 2004: OTM 2004. Lectures Notes in Computer Science. Springer-Verlag. pp 135 – 144. October 25-29, 2004

[3] Benford S.D., and Fahlén L.E. A Spatial Model of Interaction in Large Virtual Environments. Published in Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93). Milano. Italy. Kluwer Academic Publishers, pp. 109-124, 1993.

[4] J. Cao et al., "Agent-Based Grid Load Balancing using Performance-Driven Task Scheduling", Proc. of the International Parallel and Distributed Processing Symposium 2003.

[5] K. Chow and Y. Kwok, "On Load Balancing for Distributed Multiagent Computing", IEEE Transactions on Parallel and Distributed Systems 13(8), 787-801, Aug. 2002.

[6] Fahlén, L. E. and Brown, C.G., *The Use of a 3D Aura Metaphor for Compter Based Conferencing and Teleworking*, Published in Proceedings of the 4th Multi-G Workshop, Stockholm-Kista, pp. 69-74, 1992.

[7] Foster and C. Kesselman and J. Nick and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Globus Project, 2002.

[8] I. Foster, N. R. Jennings and C. Kesselman, "Brain Meets Brawn: Why Grid and Agents Need Each Other", Proceedings 3rd Int. Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2004), New York, USA, 2004.

[9] Foster et al., "Modeling Stateful Resources with Web Services", Globus Alliance, 2004.

[10] A. Galstyan, K. Czajkowski and K. Lerman, "Resource Allocation in the Grid using Reinforcement Learning", International Conference on Autonomous Agents and Multiagent Systems, 2004.

[11] Global Grid Forum. http://www.gridforum.org/ Consulted 2006.

[12] Greenhalgh, C., *Large Scale Collaborative Virtual Environments*, Doctoral Thesis. University of Nottingham. October 1997.

[13] R. Jennings et al., "Automated Negotiation: Prospects, Methods and Challenges", International Journal of Group Decision and Negotiation 10(2), 199-213, 2001.

[14] T. Kunz, "*The influence of different workload descriptions on a heuristic load balancing scheme*," IEEE Transactions on Software Engineering, vol. 17, no. 7, pp. 725–730, July 1991.

[15] M. Luck, P. McBurney and C. Preist, "Agent Technology: Enabling Next Generation Computing", AgentLink, 2003.

[16] L. Pastor y J. L. Bosque. An Efficiency and Scalability Model for Heterogeneous Clusters. Proceedings of the 3° IEEE International Conference on Cluster Computing. Editorial: IEEE Computer Society. California. Octubre 2001. pp 427 – 434. ISBN: 0-7695-1116-3.

[17] S. D. Ramchurn, D. Huynh and N. R. Jennings, "Trust in Multiagent Systems", The Knowledge Engineering Review, 2004.

[18] W. Shen et al., "Adaptive Negotiation for Agent-Based Grid Computing", Journal of the American Statistical Association, 2002.

[19] L. Xiao, S. Chen, and X. Zhang. *Dynamic cluster resource allocations for jobs with known and unknown memory demands.* IEEE Trans. on Parallel and Distributed Systems, 13(3):223–240, March 2002.

[20] C. Xu and F. Lau, *Load balancing in parallel computers: theory and pra*ctice. Kluwer Academic Publishers, 1997.

[21] Z. Zhang and S. Luo, "Constructing Grid System with Mobile Multiagent", Proc. of the Second Int. Conference on Machine Learning and Cybernetics, Xi'an, Nov. 2003.

[22] Albert Y. Zomaya and Yee-Hwei Teh. *Observations on using genetic algorithms for dynamic load-balancing.* IEEE Trans. on Parallel and Distributed Systems, 12(9):899–911, 2001.

[23] http://www.w3.org/. Consulted in 2006.

[24] http://www.w3.org/2002/ws/. Consulted in 2006.

[25] http://www.w3.org/TR/soap/. Consulted in 2006

[26] http://www.netlib.org/benchmark/performance.pdf Performance of Various Computers Using Standard Linear Equations Software. Technical Report CS-89-85, University of Tennessee, 2006.