Teaching about Madrid: A Collaborative Agents-Based Distributed Learning Course

José Luis Bosque¹, Pilar Herrero², and Susana Mata³

¹ Dpto. de Electrónica y Computadores. Universidad de Cantabria
jose.luis.bosque@unican.es

² Facultad de Informática. Universidad Politécnica de Madrid, Spain
pherrero@fi.upm.es

³ Escuela Superior de Informática, Universidad Rey Juan Carlos, Spain
susana.mata@urjc.es

Abstract. Interactive art courses require a huge amount of computational resources to be running on real time. These computational resources are even bigger if the course has been designed as a Virtual Environment with which students can interact. In this paper, we present an initiative that has been developed in a close collaboration between two Spanish Universities: Universidad Politécnica de Madrid and Universidad Rey Juan Carlos with the aim of joining two previous research projects: a Collaborative Awareness Model for Task-Balancing-Deliberation (CAMT) in clusters and the “Teaching about Madrid” course, which provides a cultural interactive background of the capital of Spain.

Keywords: Cluster computing, task assignment, collaborative work.

1 Introduction

The “Teaching about Madrid” course was developed with the aim of designing a virtual tour around Madrid. This course was composed by a set of interactive images (see Figure 1) that were presented in real time by a tour-guide. Students could interact with the scenario, if need, to get more specific information about the monument, such as the year in which it was built. Students can also collaborate with each other to learn together from the projected environment. Each of the scenarios is projected on a CAVE governed by cluster of PCs. A high speed myrinet network allows processing all these operations in real time.

As for the CAMT model, it was the result of a previous collaboration between the Universidad Politécnica de Madrid and the Universidad Rey Juan Carlos. This has been designed based on the 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 manages awareness of interaction in collaborative distributed systems, through a multi-agent architecture to create a collaborative and cooperative environment. CAMT manages the interaction in the environment allowing the autonomous, efficient and independent task allocation in the environment.

Fig. 1. Real Palace of Madrid

This paper also presents how the CAMT model complements to the “Teaching about Madrid” course as it select the best processor to make the complex render task of each of the images of the course in the cluster. CAMT divide this render task of each of these images in a set of independent processes which are assigned to the more suitable nodes in the cluster. The CAMT model’s algorithms achieve very important improvements with respect to the response time and speedup.

2 Related Work

A taxonomy of load balancing methods has been defined in [3], taking into account different aspects. Three important criteria for this classification are: *Time in which workload distribution is performed* static [6] or dynamic [11]; *Control* which can be centralized [10] or distributed [6] and *System state view* global [6] or local [4].

One approach is presented in [15], which defines a generic and scalable architecture for the efficient use of resources in a cluster based on CORBA. DASH (Dynamic Agent System for Heterogeneous) [13] is an agent-based architecture for load balancing in heterogeneous clusters. The most noticeable characteristic of this proposal is the definition of a collaborative awareness model, used for providing global information that helps establish a suitable load balance. Unlike this work, our proposal (CAMT) extends and reinterprets one of the most successful models of awareness, the Spatial Model of Interaction (SMI), which manages awareness of interaction through a set of key concepts. Most of the agent-based load balancing systems use mobile agents, which makes easier the migration of tasks [7, 14].

3 Reinterpreting the SMI Key Concepts

The Spatial Model of Interaction (SMI) [2] 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. 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*: The sub-space which effectively bounds the presence of an object within a given medium and which acts as an enabler of potential interaction [5]. 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*. Finally, *Awareness* quantifies the degree, nature or quality of interaction between two objects. Awareness between objects in a given medium is manipulated via Focus and Nimbus, requiring a negotiation process.

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 specific 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) . The CAMT model reinterprets the SMI key concepts as follow:

Focus: It is interpreted as the subset of the space on which the user has focused his attention with the aim of interacting with.

Nimbus: It is 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*.

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 [8], this awareness could be *Full*, *Peripheral* or *Null*.

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

Peripheral aware of interaction if

$$AwareInt(n_i, n_j) = Peripheral \quad \text{if } \begin{array}{l} n_j \in Focus(\{n_i\}) \quad \wedge \quad n_i \notin Nimbus(n_j) \\ \text{or} \\ n_j \notin Focus(\{n_i\}) \quad \wedge \quad n_i \in Nimbus(n_j) \end{array}$$

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

Interactive Pool: This function returns the set of nodes $\{n_j\}$ interacting with the n_i node in a given moment.

Task Resolution: This function determines if there is a service in the node n_i , being $NimbusState(n_i) \neq Null$, such that could be useful to execute the task T (or at least one

of its processes). This concept would also complement the Nimbus concept, because the NimbusSpace will determine those machines that could be taking into account in the tasks assignment process because they are not overload yet.

Collaborative Organization: This function will take into account the set of nodes determined by the *InteractivePool* and will return those nodes of the System in which it is more suitable to execute the task T (or at least one of its processes p_i). This selection will be made by means of the *TaskResolution* function.

4 The Load Balancing Algorithm in CAMT

The main characteristics of this algorithm are that it is dynamic, distributed, global and take into account the system heterogeneity. This algorithm contents the following policies [12].

State Measurement Rule: It is 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. Several authors have studied their effects on the system performance [9]. However, as for the CPU utilization, we are especially interested on the computational capabilities of the node for the new task to be executed. In this research work the concept of CPU assignment is use to determine the load index. The CPU assignment, is defined as the CPU percentage that can be assigned to a new task to be executed in the node N_i . The calculation of this assignment is based on two dynamic parameters: the number of tasks N, which are ready to be executed in the CPU queue and the percentage of occupation of the CPU, U, and it would be calculated as:

$$\text{If } (U \geq \frac{1}{N}) \Rightarrow A_{CPU} = \frac{1}{N+1}$$

$$\text{If } (U < \frac{1}{N}) \Rightarrow A_{CPU} = 1 - U$$

The NimbusState of the node depends on the load index value and an increase or decrease of this index over a specific threshold will imply the corresponding modification in the NimbusState. It determines if the node could execute more, local or remote, tasks. Its possible values would be:

- *Maximum:* The load index is low, this 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, but it cannot execute remote tasks.
- *Null:* The load index has a high value and therefore the node is overload. In this situation, the node will not execute new tasks.

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 CAMT model should be based on events [1]. This rule only collects information when a change in the Nimbus 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. As this algorithm is global, this information has to be sent to all the nodes in the system.

Initiation rule: As the model implements a non user interruption algorithm, the selection of the node must be made just before sending the task execution. The decision of starting a new load balancing operation is completely local, depends on the local information storage. when a node intends to throw the execution of a new task, the initialization rule will evaluate:

If (NimbusState = Null), a new load balancing operation is started.

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: localization, distribution and selection.

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 CAMT 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 InteractivePool 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.

This algorithm joins selection and distribution rules because it determines which nodes (among all the nodes constituting the CollaborativeOrganization) will be in charge of executing each of the processes making up the T task. The goal of this algorithm is to find the more equilibrate processes assignment to the computational nodes, based on a set of heuristics. Firstly, a complete distribution of the processes making up the T task is made in the computational nodes implicated in the CollaborativeOrganization. If, in this first turn, all the process would be assigned to one of the nodes involved in the CollaborativeOrganization, the algorithm would have finished.

5 The Underlying Architecture

The load balancing multi-agent architecture is composed of four agents which are replicated for each of the nodes of the cluster.

Load Agent: The Load Agent (LA) calculates, periodically, the load index of the local node and evaluates the changes on its state. Moreover, it defines the thresholds determining the changes on its state for that node. When it detects a change on the state, this modification is notified to the local GSA and IA. The first step of the LA is to obtain the static information. Then this information is communicated to the rest of the nodes through the MPI_Reduce function, which is in charge of calculating the maximum of the computational power of all the nodes composing the cluster.

Next, the agent enters in an infinite loop where it gets dynamically information and calculates the new state. With the new state the agent determines if a node state change has occurred and communicates it to the local GSA and IA.

Global State Agent (GSA): This agent implements the exchange information rule, and therefore its main functionality is to manage the state information exchanged

among the nodes of the system, and provide LBA with this information. Firstly, it determines the current InteractivePool. Next, the agent enters in an infinite loop in which it is waiting for receiving messages from other agents. These messages are:

- **LOCAL_STATE_CHANGE:** This message comes from the local LA and this information has to be notified to all the Global State Agents that are located in a different node of the cluster to update their lists.
- **REMOTE_STATE_CHANGE:** In this case, only the local state list should be modified to update the new state of the remote node.
- **INTERACTIVE_POOL_REQUEST:** The local LBA request the Interactive-Pool to the GSA. The GSA responds to this request providing it with the required information.

Initiation Agent (IA): This agent is in charge of evaluating the initialisation rule and it determines, if the task can be executed locally or if a new load balancing operation has to be carried out. Its main structure contains an infinite loop and, for each of these iterations, the pending tasks in the execution queue are checked. If there is a pending task, a new assignment task loop starts:

- **LOCAL_STATE_CHANGE:** It receives a message from the local LA to notify a change on the local state.
- **EXECUTE_TASK_REQUEST:** It requests execution of a new task. As a task is composed by a set of processes, the local execution of one of these processes can change the NimbusState of that node. Therefore, when an execution request is received, the IA starts a loop to assign all the processes of the task. For the first process, the NimbusState is checked to corroborate if its value is equal to Maximum. If later, that process is executed locally. On the other hand, a new balancing operation would start and a message would be sent to the local LBA.

Load Balancer Agent (LBA): This agent is responsible of making the load balancing operation. Its structure contains an infinite loop that is waiting to receive messages from other agents, being the possible messages:

- **BALANCER EXECUTION:** This message comes from the local IA and it indicates that a new load balancing operation needs to start. For the localization rule, the LBA will follow the following sequence of steps:
 1. Request the InteractivePool and the states list to the local GSA.
 2. Determine the TaskResolution, analysing which nodes of the InteractivePool have their NimbusState different to Null.
 3. Request the score, of those processes composing the task to be executed, to the LBA of the nodes included in the TaskResolution
 4. Taking into account the TaskResolution and the requested scores, determine the Collaborative_Organization by analysing those nodes that, belonging to the TaskResolution, can execute at least one of the processes of the task.

As for the selection and distribution rule, once the CollaborativeOrganization has been made up, it is necessary to determine which processes are sent to each of the nodes of the cluster. In order to make this possible, the algorithm presented in

section 4 has been implemented. Once all the process had been assigned, they would be sent to the designated nodes.

- **REMOTE_EXECUTION:** The message received comes from the remote LBA, asking for the remote execution of a process. Once the LBA has checked its own state, it replies to the remote LBA with an acceptance or rejection message. If the process is accepted, the operation would conclude, the LBA would execute the process locally and it would update its state. The rejection could be due to a change on its nimbusState (to Null) which has not been notified yet due to the network latency.

6 AMT Evaluation on the “Teaching about Madrid Course”

The “Teaching about Madrid” course requires the render of realistic scenarios, in real time, for an immersive environment. This task entails complex processes - such as geometric transformations, collision detection, and illumination and shadowing algorithms- that require a huge amount of floating point mathematical operations. On the other hand, the CAVE has 4 projectors and each of these projectors are connected to a different PC which is in charge of rendering the corresponding images of the scenario. However, as the geometrical model and the illumination algorithms are getting more complex, the computational capacity of these PCs gets overflowed and some images are lost. If later, users perceive a gap between two consecutive images and therefore the scenario’s realism and the user’s immersion decreases considerable.

On the other hand, as the render task can be split up in several processes which can be executed independently, the CAMT model seems to be appropriated for improving the “Teaching about Madrid” performance through the execution of the render task in a high-performance cluster.

The cluster is made up of 40 PCs (nodes) connected through a 1.1 Gbps Myrinet Network. Each cluster node is a 2 GHz AMD K7 processor with 512 MB of main

Fig. 2. Puerta del Sol of Madrid

Fig. 3. Teatro Real

Fig. 4. CAMT Overhead in the Teatro Real

Fig. 5. CAMT Overhead in the Puerta del Sol

Fig. 6. CAMT Overhead in the Palacio Real

memory. The CAMT model has been developed using GNU tools and LAM/MPI 7.1.1 Library. As a way of evaluating the improvement that the CAMT model introduces on the performance of the Teaching about Madrid course, we have selected few scenarios with different level of geometrical complexity. These scenarios are: Teatro Real (Figure 3), Puerta del Sol (Figure 2) and Palacio Real (Figure 6).

In this section we present the set of results obtained evaluating the overhead introduced by CAMT while it assign the processes to the nodes of the cluster. As for the course overhead, Figures 4, 5 and 6 demonstrate that the overhead incurred by the algorithm to assign a process doesn't interfere with the frame's rate of the CAVE's projectors. The overhead remains almost constant for all of the tasks and processes even although it increases as the geometrical complexity of the scenario - and therefore the data file's size- also increases, demonstrating that the CAMT algorithm has been endowed with very strong scalability features.

7 Conclusions

This paper presents the integration of two previous research works. The first of these two projects was a guided course, named "Teaching about Madrid" which intended to provide students with a cultural interactive background of Madrid. The second one, CAMT, manages awareness of interaction in collaborative distributed systems, through a multi-agent architecture to allow the autonomous, efficient and independent task allocation in the environment. The CAMT model complements to the "Teaching about Madrid" course as it select the best processor to make the complex render task of each of the images of the course in the cluster. CAMT divide this render task of each of these images in a set of independent processes which are assigned to the more suitable nodes in the cluster. Thus, even although the geometrical model and the illumination algorithms are complex, practically none of the images are lost, and users never perceive a gap between two consecutives images, feeling a high degree of realism and immersion. Moreover, the experimental results presented in this paper demonstrate that the overhead incurred by the algorithm to assign a process doesn't

interfere with the frame's rate of the CAVE's projectors, and therefore we can conclude that CAMT complements successfully the teaching course.

References

- [1] Beltrán, M., Bosque, J.L., Guzmán, A.: Resource Dissemination policies on Grids. In: *On the Move to Meaningful Internet Systems 2004: OTM 2004*. LNCS, pp. 135–144. Springer, Heidelberg (2004)
- [2] Benford, S.D., Fahlén, L.E.: A Spatial Model of Interaction in Large Virtual Environments. In: *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW 1993)*, Milano, Italy, pp. 109–124 (1993)
- [3] Casavant, T.L., Kuhl, J.G.: A taxonomy of scheduling in general-purpose distributed computing systems. In: *Readings and Distributed Computing Systems*, pp. 31–51 (1994)
- [4] Corradi, A., Leonardi, L., Zambonelli, F.: Diffusive load-balancing policies for dynamic applications. *IEEE Concurrency* 7(1), 22–31 (1999)
- [5] Darbha, S., Agrawal, D.P.: Optimal scheduling algorithm for distributed-memory machines. *IEEE Trans. on Parallel and Distributed Systems* 9(1), 87–95 (1998)
- [6] Das, S.K., Harvey, D.J., Biswas, R.: Parallel processing of adaptive meshes with load balancing. *IEEE Trans. on Parallel and Distributed Systems* (12), 1269–1280 (2001)
- [7] Desic, S., Huljenic, D.: Agents based load balancing with component distribution capability. In: *Proceedings of the 2nd International Symposium on Cluster Computing and the Grid (CCGRID 2002)* (2002)
- [8] Greenhalgh, C.: *Large Scale Collaborative Virtual Environments*, Doctoral Thesis. University of Nottingham (October 1997)
- [9] Kunz, T.: The influence of different workload descriptions on a heuristic load balancing scheme. *IEEE Transactions on Software Engineering* 17(7), 725–730 (1991)
- [10] Schnor, B., Petri, S., Oleyniczak, R., Langendorfer, H.: Scheduling of Parallel Applications on Heterogeneous Workstation Clusters. In: *Proc. of the 9th Int. Conf. on Parallel and Distributed Computing Systems*, vol. 1, pp. 330–337 (September 1996)
- [11] Xiao, L., Chen, S., Zhang, X.: Dynamic cluster resource allocations for jobs with known and unknown memory demands. *IEEE Trans. on Parallel and Distributed Systems* 13(3), 223–240 (2002)
- [12] Xu, C., Lau, F.: *Load balancing in parallel computers: theory and practice*. Kluwer Academic Publishers, Dordrecht (1997)
- [13] Rajagopalan, A., Hariri, S.: An Agent Based Dynamic Load Balancing System. In: *Proc. of the Int.l. Workshop on Autonomous Decentralized Systems*, pp. 164–171 (2000)
- [14] Suri, N., Groth, P.T., Bradshaw, J.M.: While You're Away: A System for Load-Balancing and Resource Sharing Based on Mobile Agents. In: *1st Int. Sym. on Cluster Computing and the Grid*, p. 470 (2001)
- [15] Vanhastel, S., De Turck, F., Demeester, P.: Design of a generic platform for efficient and scalable cluster computing based on middleware technology. In: *Proceedings of the CCGRID 2001*, pp. 40–47 (2001)