Giving Embodied Agents a Grid-Boost

José A. Pérez Sánchez[1], Carlos Delgado Mata[2], Antonio F. Gómez Skármeta[1], and Ruth S. Aylett[2]

[1]Department of Information and Communications Engineering, Faculty of Computing, University of Murcia, Campus de Espinardo, 30071 Murcia, Spain

[2]Centre for Virtual Environments, The University of Salford, Business House, University Road, Salford, M5 4WT, Manchester, United Kingdom

Abstract Grid technology has been widely used for large-scale computational problems, but it also provides a framework for running a big number of small-sized processes. Moreover, these processes can take advantage of the Grid facilities like security, fault tolerance, or load balancing.
This paper describes the use of Grid technology to boost an autonomous agents architecture, which simulates the behaviour of flocking mammals that communicate emotions through pheromones. The capabilities of the Grid make it possible to run simulations with thousands (or even millions) of animals in the virtual world. Further, the Grid allows agents to detach from network topology, provides them with secured communications, and migrates them to more suitable machines if necessary.

1 Introduction

In the task of creating virtual environments, a key point is making them look realistic. Including virtual animals that behave like their physical counterparts helps to achieve this aim. However, a believable behaviour needs more complex mechanisms [1,2], which results in greater computational requirements.

Autonomous agents are software entities capable of independent action in open, unpredictable environments. Using agents is a widely used method for modelling artificial life forms, as in [3,4,5]. Also, embodied agents have been used in robotics and virtual environments.

Originally, Grid computing was applied to solve large problems where the computational resources are distributed. The well-known Grid initiatives fall in this category (i.e. biomedicine, weather forecasting). Other technologies like Legion [6] or Condor [7] have been utilised also with the same objective. In addition, the Grid offers other possibilities, thus different fields are being exploited, as e-commerce or massively multiplayer games [8].

The Globus toolkit [9] provides a Grid middleware that integrates: Information services, Resource Management services, and Data Management services. All of them upon a

common security infrastructure which enables secure authentication and communication over an open network.

In this paper, we present the design of a Grid-enabled virtual environment, where agents are embodied to form virtual animals. Simulating a Virtual World is a computationally intensive task, both animating the entities and rendering the graphics. When the Virtual World is populated by agents with complex, independent, non-deterministic behaviours, the need for computionational power grows linearly with the number of agents. So grows the needed memory. Therefore, running the world in a single computer limits the amount of agents that can be executed. Our system presents a design which distributes the agents over the computers in the Grid. Thus a large number of simultaneous agents can be supported.

2 Related Work

When using agent and Grid technologies together, there are two orthogonal combinations: agent-based Grid computing and Grid-enabled agent platforms. Naturally, hybrid combinations are also possible. However, the former has been the most researched, due to the complexness of Grid management [10,11].

Many projects, like *distributed.net* [12] or *SETI@home* [13], apply the Master-Worker paradigm, which is well-known in distributed computing [14,15,16]. Large problems are solved dividing the data in several smaller problems, using an aggregation function at the completion of the tasks in order to get the final result. However, our problem data is constantly changing, and every animal has a different perception of the world at every moment.

3 System Architecture

In the following subsections, we describe the key parts of the system architecture (figure 1).

3.1 Virtual Environment

The Virtual Environment uses SGI's OpenGL Performer, which contains a scene graph. In our architecture, three types of files are used to define the contents in the VE.

- A configuration file which define the objects to be included in the world.
- A file defining the geometries of the objects.
- A file defining the repertoire of animations.

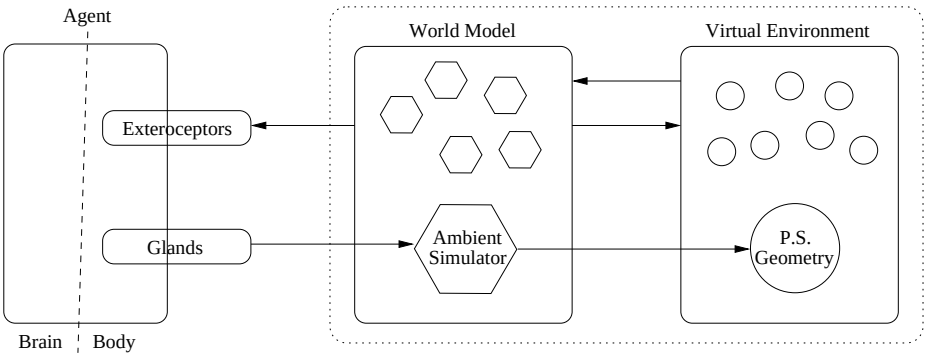

Figure 1. System overview.

There can be three types of objects in the World: static objects, moving objects and creatures (which are a specialisation of moving objects, inheriting their animation capabilities). There is a geometry file associated to each of these objects. That file can be imported into the scene-graph. Each time-step the objects' transformation matrices are updated. Then, the scene-graph is traversed in order to generate a new frame.

As its name implies, moving objects can be animated. That is, their geometries are updated to generate movements and rotations producing an "illusion of life". The animation in the moving objects is controlled by joints (like in [5]). The animation of the animal's movement was motion-captured from [17]. The resulting data were stored in a animation file, specifying the rotation of each joint in relation to its parent (another joint). There is a geometry related to each joint which is updated using matrix computation. For a in-depth explanation ethe reader is referred to [2].

3.2 Animals

As stated above, we use embodied agents for animating the virtual animals. These agents follow a two-layered design, separating the brain from the body (see figure 2).

Body The creatures' bodies belong to the virtual world's object hierarchy, therefore they are included in the simulation of the world. Each cycle, the simulation engine updates the environment. The non-intelligent entities are easily updated, but the autonomous agents need a more complex operation. Every animal is provided with a set of sensorial systems (i.e. olfaction, flight zone sensor). Those artificial senses produce numerical data which are forwarded to the brain. The brain emits commands to the motor control centre and the glands (like the *apocrine* glands). The latter emit pheronomes when the animal 'feels' fear.

Brain The creatures' brains run separately from the simulation engine. Therefore, they are not synchronised with the simulation. However, the brains depend on the sensorial data that come from their bodies. Each brain runs at its own pace, avoiding

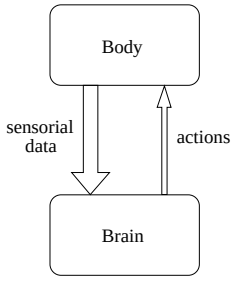

Figure 2. Agents are composed of body and brain.

the transmission of unnecessary updates. Despite the fact the brain's state is continuously updated with the sensorial data, actions are selected and sent to the body in the configured time rate (usually different from that of incoming data).

Agent's behaviour is configured in a set of XML files. For a deeper view on the internal operation of the brain and the body, the reader is referred to [1,2].

A virtual environment needs to render images at a frame rate in such a way that users perceive it like real (that is, smoothly animated). In order to achieve this, the world model has to be updated approximately from 25 to 30 times per second (the rate can be specified in the configuration file). At each update, each body sends its sensorial data to its brain. Thus, in the case of operating with more than twenty animals, there would be massive amounts of data being sent. Furthermore, there would be action data being received. This incoming traffic would not be as loaded as the former, though it would be still significant.

With these traffic characteristics, a communication solution based on streaming stands out as the most suitable, against other options like message passing or RPC (Remote Procedure Calls).

3.3 Broker

This entity plays a fundamental role in the system's operation. The broker is responsible for:

- Establishing which computers in the Grid are available and suitable.
- Sending the appropriate files to the chosen computers.
- Initiating the execution of the brains at those computers.

The broker accomplishes the first task by querying the Globus Monitoring and Discovery Service (MDS) [18], which provides information about the Grid. Two kinds of

information can be obtained: static and dynamic. Static information includes number of CPUs and type, real memory, virtual memory, filesystems and networks; while dynamic information includes CPU, memory, filesystems and bandwidth usage, or information about the current jobs. After querying the information, the broker asks the MDS for a set of computers that meet the requirements for running the brains: the CPU type is compatible, the CPU is idle enough, and it has enough network bandwidth, among others. If the set of available computers is not big enough for the number of animals, the user must choose between reducing the number of animals or running the simulation as it is, bearing in mind that some animals will suffer from *slow thinking* (their brains would send actions based on old sensorial data, thus reacting too late to look real).

For the second task, the broker sends the binary and configuration files needed for running the brains, through the Globus Access to Secondary Storage (GASS). Because of the Grid's inherent heterogeneousness, the binaries are compiled for different platforms. In this way, more computers will be able to participate in the simulation.

Finally, the broker contacts the Globus Resource Allocation Manager (GRAM) [19], and requests the execution of the brains. The bodies are then connected to the brains, and the system starts to work.

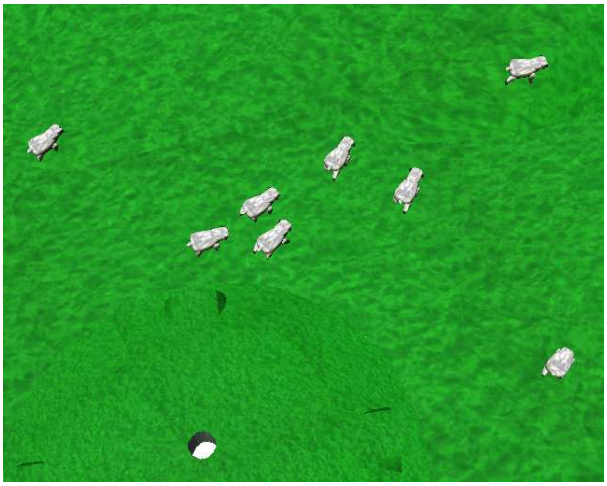

Figure 3. Sheeps grazing. A snapshot of the current system.

4 Current implementation

A non-Grid version is already developed and working [2]. The above design improves the performance of the existing version. Network communication and deployment is also enhanced.

The Virtual Environment runs on a graphics workstation (in our case twelve processors SGI Onyx that support a CAVE, an enclosure with four screens of 3m$\times$3m), on a O_2 and even on the less-expensive Linux workstations. In figure 3, some virtual sheeps are grazing in a virtual rangeland.

The current system needs manual deployment and configuration. Users have to copy the brains' executables to the computers, write the configuration files (e.g. selecting the ports where they brains shall be listening), and launch the programs. Brains and bodies intercommunicate using plain UDP packets, vulnerable to mischievous hackers.

5 Conclusions and Future Work

This work has presented the design for a Grid-enabled autonomous agents system, where the agents' brains are distributed for performance boost. The Globus toolkit provides the framework for developing it.

Further work will be continued on the following issues:

5.1 Fault tolerance

In case of computer or network failure, some of the animals would stop moving. To an observer, they would look like mesmerized, or stuck forever in the same action, spoiling the illusion of reality.

Heartbeat is a system keep-alive function widely used for monitoring the health of the nodes in a cluster. With this facility, in case a computer stops answering, the broker will be able to detect the failure. Then, the broker would find a new host (or hosts) for running the substituting brains. It would also re-route the sensorial data streams to them. Although the new brains would be in an initial state, they would reach a normal behaviour soon.

5.2 Load balancing

This issue is addressed in a similar way to fault tolerance. Contrary to what happened above, the broker must decide whether it launches the mechanism (although there can be external influences, like the need for sharing some computers with another applications). Unlike with fault tolerance mechanism, brains' states will not be lost. Hence, the newly created brains will be initiated with the saved states, and the system will continue to run smoothly.

5.3 World distribution

While brains are distributed all over the Grid, the world simulation is managed in one computer. As the world size grows, a number of problems arise. On the one hand, the simulation needs more computational power because the world is bigger. Furthermore, the world will contain more objects and agents. On the other hand, more bandwidth will be used to communicate with the larger number of brains. A solution for these issues is segmenting the world [8].

References

1. Delgado-Mata, C., Ibanez, J., Aylett, R.: Let's run for it: Conspecific emotional flocking triggered via virtual pheromones. In Butz, A., Krüger, A., Oliver, P., eds.: LNCS Smart Graphics 2003, Heidelberg, Germany, Springer-Verlag (2003) 131–140
2. Delgado-Mata, C.: A Mechanism for Emotion Signalling in Multiple Intelligent Virtual Agents. PhD thesis, University of Salford, Salford, United Kingdom (2004) In preparation.
3. Tu, X., Terzopoulos, D.: Artificial fishes: Physics, locomotion, perception, behavior. In: In Proceedings of SIGGRAPH'94, ACM Computer Graphics (1994) 43–50
4. Tyrrell, T.: Computational Mechanisms for Action Selection. PhD thesis, University of Edinburgh, Edinburgh, Scotland (1993)
5. Blumberg, B.: Old Tricks, New Dogs: Ethology and Interactive Creatures. PhD thesis, Massachusetts Institute of Technology, MIT, Cambridge, MA (1996)
6. Grimshaw, A.S., Wulf, W.A.: The Legion Vision of a Worldwide Virtual Computer. Communications of the ACM **40** (1997)
7. Litzkow, M., Livny, M., Mutka, M.: Condor - A Hunter of Idle Workstations. In: Proceedings of the 8th International Conference of Distributed Computing Systems. (1988)
8. Butterfly.net, Inc.: Butterfly Grid Solution for Online Games. `http://www.butterfly.net` (2004)
9. University of Chicago: The Globus Project Home Page. `http://www.globus.org` (2004)
10. Overeinder, B.J., W.N.S.M.v.B.F.: Multi-Agent Support for Internet-Scale Grid Management. In Rana, O., Schroeder, M., eds.: Proceedings of the AISB'02 Symposium on AI and Grid Computing, London, UK (2002) 18–22
11. Rana, O.F., Moreau, L.: Issues in Building Agent-Based Computational Grids. In: UK Multi-Agent Systems Workshop, Oxford, UK (2000)
12. distributed.net: `http://www.distributed.net` (2004)
13. Sullivan, W.T., Werthimer, D., Bowyer, S., Cobb, J., Gedye, D., Anderson, D.: Astronomical and biochemical origins and the search for life in the universe. In C.B. Cosmovici, S.B., Werthimer, D., eds.: Proc. of the Fifth Intl. Conf. on Bioastronomy, Bologna, Italy, Editrice Compositori (1997) IAU Colloq. No. 161
14. Foster, I., Kesselman, C., eds.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann (1999)
15. Shao, G., Berman, F., Wolski, R.: Master/Slave Computing on the Grid. In: Proceedings of the 2000 Heterogeneous Computing Workshop. (2000) `http://gcl.ucsd.edu/amwat/`.
16. Linderoth, J., Kulkarni, S., Goux, J.P., Yoder, M.: An Enabling Framework for Master-Worker Applications on the Computational Grid. In: Proceedings of the Ninth IEEE Symposium on High Performance Distributed Computing (HPDC9), Pittsburgh, PN (2000) 43–50 `http://www.cs.wisc.edu/condor/mw/`.

17. Muybridge, E.: Animals in Motion. Dover, New York (1957)
18. Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., Tuecke, S.: A directory service for confi guring high-performance distributed computations. In: Proc. 6th IEEE Symp. on High Performance Distributed Computing, IEEE Computer Society Press (1997) 365–375
19. Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., Tuecke, S.: A resource management architecture for metacomputing systems. In: The 4th Workshop on Job Scheduling Strategies for Parallel Processing, Springer-Verlag LNCS 1459 (1998) 62–82