Invited Paper:

Distributed Simulation of Large-Scale Individual Oriented Models *

Diego Mostaccio, Christianne Dalforno, Remo Suppi [1] and Emilio Luque
Department of Computer Architecture & Operating Systems
Universitat Autònoma de Barcelona, UAB
Bellaterra, 08193, Barcelona, Spain

ABSTRACT

The distributed simulation for high performance models has been carried-out into a very useful and low-cost tool. In the present work their application to an individual oriented model (Fish Schools) is analyzed. To do this, it was analyzed different alternatives for this type of simulation and their application to obtain performance and model scalability using different implementation. Thus, two distributed simulator have been developed based on PVM and MPI communication libraries. This paper resumes the advantages and drawbacks of each implementation and some conclusions about the distributed simulation for this type of models are extracted. Moreover, visualization aspects and the developed infrastructure based on OpenGl are described.

Keywords: distributed simulation, individual oriented models, high performance simulation, PVM-MPI distributed computing.

1. INTRODUCTION

Since many years ago until nowadays, the simulation has gained the attention of many areas that gave many uses to this kind of computer application. It is explained with the fact that simulation allows to imitate the behaviour of an existent system or to experience an imagined one. [1]

The importance of this tool to some areas like Astrophysics, which have to study and to do the interrelation of complexes physical process, tends to increase with the advance of the investigations.[2] Therefore, the demand for simulation tools that can give accurate results in an acceptable time will be greater in the near future.

To improve the ways of a simulation works to attend the demand of the other areas is the general aim of the Distributed Simulation studies. In this work it was choose to simulate an ecological system named Fish Schools (fish species movement simulation).

Some fish species has the characteristic of to be congregated in a big group for a long time. The peculiar behaviour of this population interests to the ecologists and biologists and can be observed in groups of another animals and even inanimate aggregations. [3, 4]

Population behaviour can be modelled in two ways:

- Some aggregated state-variables can be established to the population and expressions that will change the value of them in function of the time. This kind of model, named state-variable models, considers that "all individuals are statically similar and that each interacts uniformly with all other members of its own population and with the members of other populations." [5]
- Some simple rules can be applied to each individual and that is sufficient to describe the population behaviour. It's named Individual-oriented Model (IoM).

The IoM offers some advantages in face of the other kind of model [5], but generates a quantity of computation that makes difficult to do it in an appropriated time using a single processor with sequential algorithms.

"DES (Distributed Event Simulation) enables users to obtain efficient solutions in acceptable time periods as long as the parallelism of the simulated model allows for a reasonable degree of concurrency." [6]

Once that the distribution of the simulation is made, it inserts some problems related to the sequence of occurrence of the events. It must be guarantied that this sequence is the same as the sequence occurred in sequential simulation. To avoid this kind of problem, there are some protocols of witch the conservative protocol is the one that was chosen to be used in this work. [5]

This choice is justified by the fact that the use of the optimistic protocol presented some problems with this kind of model. [6]

The attempts to obtain better results perfecting the conservative protocol show that the gains weren't so representative. Thus this work

[1] Corresponding author: Remo.Suppi@uab.es

* Supported by the MEyC-Spain under contract TIN 2004-03388 and Generalitat de Catatalunya GRC 2001SGR-00218.

presents the results of an analyses of two distributed simulators, one using MPI and the other based on the PVM in comparison with a sequential simulator.
This paper is organized as follows: section two concentrates on Individual oriented Models (IoM). Section three focuses on Distributed Event Simulation as well as the design and implementation of the Fish-School DES simulator and visualization tools. Section four presents the experiments carried out to validate the results of the simulation and shows the performance obtained. Section five summarizes the conclusions obtained and outlines future work

2. INDIVIDUAL-ORIENTED MODEL (IoM)

The Individual-oriented Models sprouted to be a solution to the limitations imposed by the models based in a population that use, for example, differential equations to explain the behaviour of a group of individuals. [7, 8]
To do this, it's necessary to consider that all of the individuals are similar, presenting the same reactions and affecting in the same way the others individuals and the environment.
In the IoM, the unit of the system is the individual, so the heterogeneity of the group can be represented in the differentiated behaviour of each individual.
In addition, sometimes it could be very complex to model an ecological system using differential equations or stochastically process. To define the rules that guide the behaviour of one individual should be, in most of the cases, a simpler task. [5, 9]
This work uses an IoM model named Fish Schools that simulates the movement of fish species by applying simple rules to each fish in a big group of it.

The Fish Schools

In the Fish Schools each fish is represented by a point in the space and its velocity. [9]
Based in the needs by the fishes about avoid predation, reproduction and avoid collisions, simple rules can be established about the behaviour of each of then in a group. [6]
The mentioned behaviour has respect with the direction to where the fish goes through and its velocity. After a certain period a fish change its parameters influenced by the rules that guarantees its survival.
The change in the course of a fish occurs like a reaction to the influence of some selected neighbours. The parameters to select this neighbours that influence in its behaviour is established by the rules:

1. The fish will choose about four (4) neighbours from all of the other fishes in the school.
2. How can be seen in the Figure 1, three radios are established. The fishes that are out of the more far radio aren't a valid candidate like the ones that are out of the vision angle of the fish (i.g the black area in the figure 1).
3. From the valid candidates, it will be chosen those that are more visible and nearest of the fish.

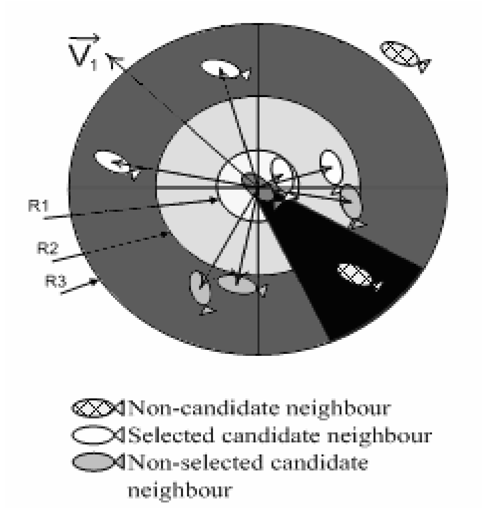
$\vec{V}_1$
R1
R2
R3
Non-candidate neighbour
Selected candidate neighbour
Non-selected candidate neighbour

Figure 1. Neighbour selection in the Fish-Schools model

After define the neighbours, it has to be calculated the change suffered by the fish. Each neighbour will influence the fish in one way in accordance to your relative position. There are three radios and then three possible reactions:

1. Repulsion: when the neighbour is in the smaller radio, the fish tends to swim to opposite direction to avoid collisions.
2. Parallel orientation: when the neighbour is between the smaller and the medium radio (R1 and R2 in the figure 1), the fish tends to swim together with it.
3. Attraction: when the neighbour is between the medium radio and the greatest radio (R2 and R3 in the figure 1), the fish tends to go in direction of it to go closer to it.

Once established the reaction suffered by the fish F_i it will be represented by the rotation of the $\boldsymbol{v_i}$ to each fish F_j. β_{ij} will be the F_i reaction with respect to F_j expressed in spherical coordinates. Each fish F_j can be found within one of three possible areas of influence with respect to F_i (R_1,R_2,R_3):

- If Dist(F_j, F_i) $\leq$ R_1, F_i then
  $\beta ij = (1,\pi,\pi)$
- If R_1<Dist(F_j, F_i) $\leq$ R_2, F_i then
  $\beta ij = (1,\theta_j,\varphi_j)$
- If R_2< Dist(F_j, F_i) $\leq$ R_3, F_i then
  $\beta ij = (1, \theta_i - \theta_j, \varphi_i-\varphi_j)$

Finally, reaction β (mean value for all β_{ij}) and $\boldsymbol{v_i}$ is rotated according to β value.

In a sequential version, this algorithm try to define the neighbours of a fish has working with the data of all other fishes been simulated. This gives to this algorithm the complexity of $O(N^2)$, where N is the number of fishes been simulated. [9]

To attend the needs of the high quantity of computation involved a solution is to use parallel or distributed computation.

3. DISTRIBUTED DISCRET EVENT SIMULATION

When someone decides to divide the simulation to be realized for many processors working together, its first task is to split it in many logical process to be executed each one in a processor.

In this work, there is a space (where are the fishes been simulated) that is divided into equal parts and each of them is assigned to a processor (figure 2). [6]

In relation to the way the simulation time is advanced and how the system state changes, the strategy used in this work was to use Discrete Event Simulation (DES). In a DES, there is an event list and each step of the simulation consists in to execute the event, that have the lowest time-stamp (the time in witch the event have to occur), and to change the simulation time to be equal to the time-stamp of the simulated event. [1]

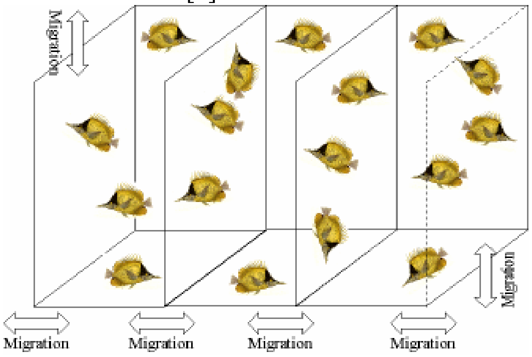

Figure 2. Model Distribution

Additionally to the fact that this simulator is a DES, it is a distributed one. Thus, sometimes, the LP's will communicate scheduling a new event, asking data or answering a received requesting. This communication is implemented by the sending of time-stamp-massages between the LP's.

Figure 3 show the influence of one fish simulated in the processor **i** that need to select neighbour candidates. The simulator process will send messages to the processor **i+1** (right), and this will answer with the position and speed of the possible candidates and this communication should not to generate causality problems. That is to say, the simulation step accomplished in the time T_i of the processor P_i must have a answer of the position/speed of the processor P_{i+1} in the time Ti. This causality condition impose that it is necessary a synchronization mechanism for the distributed simulation.

Thus, there are three kinds of message (EvRequest, EvAnsware, EvMigration) and two event types (EvNextStep and EvResumeStep), in this simulator, that are executed as follows [6]:

- EvNextStep: the new position and velocity of each fish is calculated.
- EvMigration: a fish goes from one LP to other. The possible occurrences of migration can be seen in Figure 2.
- EvRequest: sometimes, when a fish is next to the border, it needs to know if there is a neighbour-candidate in the next LP (this situation can be seen in the Figure 3). To ask for this data the LP sends an EvRequest message to the appropriate LP. When an LP sends an EvRequested it get blocked waiting for the answer.
- EvAnsware: with it an LP sends the requested data from another LP.
- EvResumeStep: restarts the simulation in the blocked LP.

The execution of the messages received by the others LP's needs some attention, because of the causality problems. The problem is that it has to be guaranteed that the sequence of the simulation of the events is the same as the sequence of execution in a sequential algorithm.

To avoid this kind of problem, in this work it was used the conservative protocol. In this method an LP only execute an event if it is sure that it will not receive any message in its past. [1]

Choose the conservative protocol is justified by the bed results obtained with the optimistic protocol. This last protocol works in a way that all LP's goes executing even if it is not sure about receive a message in its past. If this occur a rollback have to be executed and a part of the simulation have to be done again. The bed results are explained by the fact that many rollbacks are generated when there are fishes swimming in the border and this situation is difficult to control.

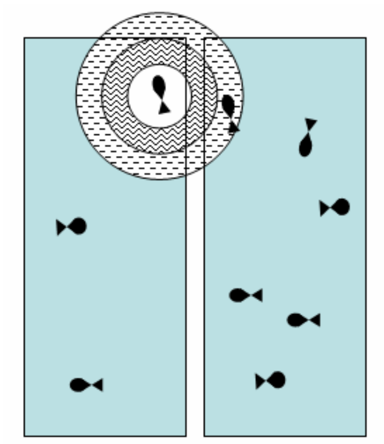

Figure 3. Fish influenced by a neighbour in the next LP

Distributed Simulation and Animation Infrastructure

The Fish Schools application has been split into two computing parts (Fig. 4):

- **Beowulf Simulation cluster** (based on Linux & MPI): where the distributed simulator is executed and the frame animation data is generated. In this cluster, there is a machine that reorganizes the data and prepares them in order to be animated (Data Collector). This data collector can be connected on line with the animation application through Tcp/Ip as well as generating trace files to be visualized in a subsequent animation.
- **Animation tool:** This tool animates the data sent by the simulator and controls all the animation parameters such as cameras, lights, scenes, point of view, etc.

The user interaction is made through a simple GUI based on a web form where the user indicates the parameters and the characteristics of the simulation. The web server, which is executed in the data collector, generates the initial conditions and runs the distributed simulator that will send data through a Fast Ethernet to the animation tool, or else will generate trace files for post-animation. Figure 4 shows the whole system and its interaction.

An animation tool has been designed as a multithreading application communicated by sockets with a distributed simulator process (data collector), in order to obtain the real-time data of the fish movement. OpenGL library was used to obtain a high-quality animation in 3D with the possibility to provide the user with an interactive interface with the animation engine changing points of view, light effects, shades, textures etc.

The application of animation is a critical program whose structure has been carefully designed to represent 3D objects (fish) in a virtual environment in real time. The position and the speed of the fish are produced by the distributed simulator and are collected into the animation server orstored into trace files).

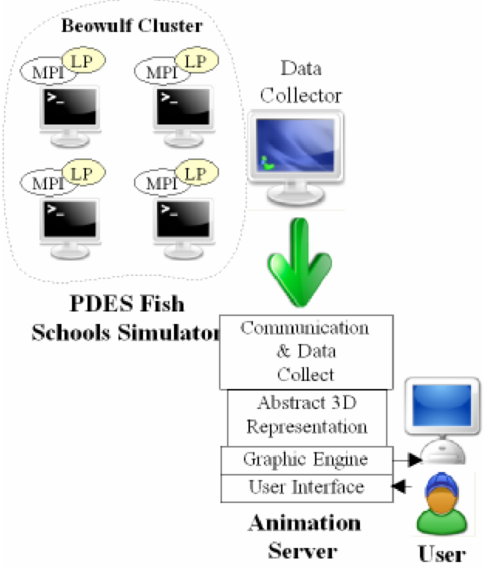

Figure 4. Simulation & Animation Infrastructure

The animation tool is organized in four modules [12]:

- **User Interface:** this is a thread that attends to the user during the animation.
- **Communication and Data Collect**: this is the thread responsible for establishing communication with the simulator to obtain the data and to create and control the hash buffers used to accomplish the animation.
- **Abstract 3D Representation**: this module manages the virtual world (three-dimensional space and objects) using 3D models of the objects.
- **Graphic Engine**: the main functions of this module are: initialization of the graphics environment and 3D models subsystems, setup and control of the cameras, lights, textures, scene, etc. and repainting of the screen.

Figures 5 shows two types of animation frames obtained with the DES simulator for 200 fish (with and without texture and decorations).

4. EXPERIMENTAL RESULTS

The verification and validation of the model and simulator were made using a Beowulf cluster with Linux and Fast Ethernet. A sequential simulator was developed to analyse the scalability and performance of the DES simulator. This simulator has only one event list and runs on a single computer (one processor). As a performance measure, frame time generation was selected. This measurement is the time required to compute the following position and speed for all individuals in the simulated world. Figure 6 shows the time per frame for different numbers of fish (from 100 to 25,000).

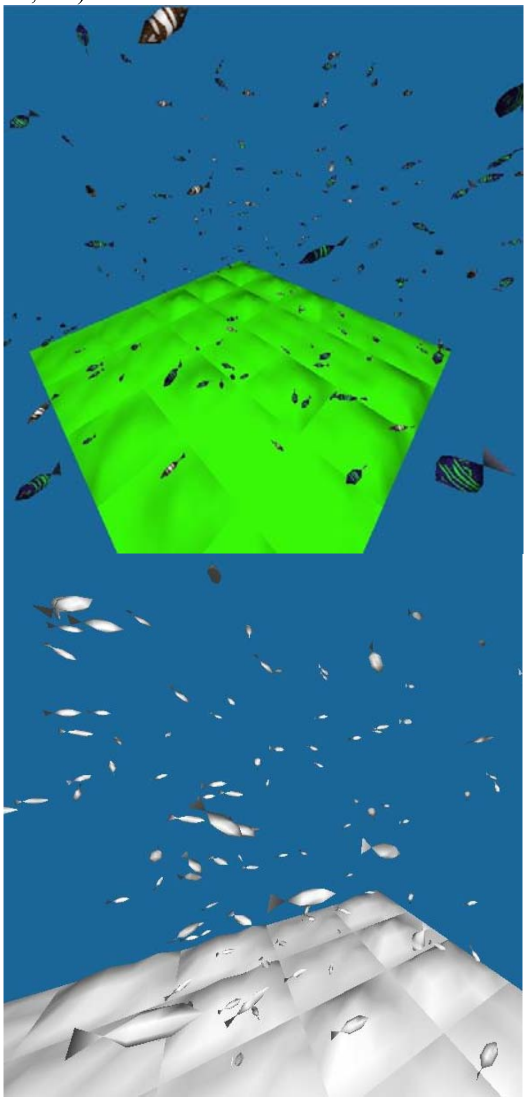
Figure 5. Animations frames obtained from Animation Server

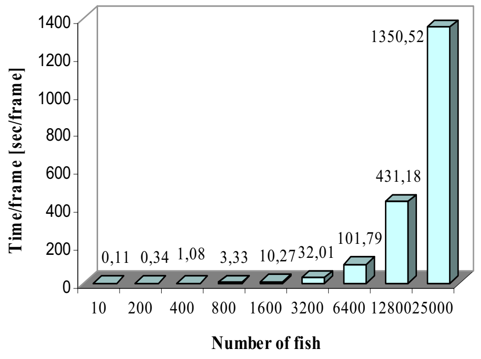

Figure 6: Time per frame in the sequential simulator

The experiments accomplished for the distributed simulator with PVM have been limited to 1,600 individual and 8 processing nodes. These restrictions have been conditioned by scalability problems in the PVM communications model. The MPI distributed simulator uses a different communications model and we have been able to perform different tests by varying the number of individuals (from 100 to 25,000) as well as the number of processors (from 2 to 32). Figure 7 shows the times per frame for two distributed simulators for populations between 100 and 1,600 fish and 8 computing nodes.

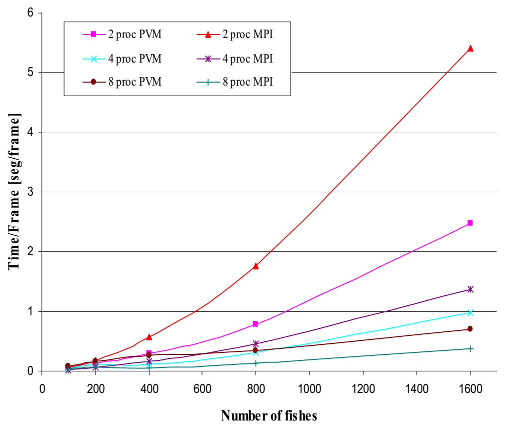

Figure 7: Time per frame in the distributed simulators

The first development of the FishSchools simulator was accomplished using the PVM [10] communication library. This development facilitated the analysis and in deep study of new distributed simulation algorithms. In this simulator was verified that the optimistic simulation policies generate rollbacks chains in the edge or migration zones. This simulator was fundamental to decide the application of conservative simulation policies on this type of models.

The only one PVM drawback was the performance reduction for large individual's quantities. The main problems was derived from PVM implementation (pvmd daemon), the memory management and the tail-drop policy of messages buffers for large quantities of individuals and processors (more than 2400 individual and more than 8 processors).

In order to compare the distributed and serial simulator, the speedup was obtained. The figure 8 shows the PVM speedup until 1600 individual and 8 processors.

As can be observed, there are values higher than the linearity. This is due to calculation of the neighbour position. In the sequential algorithm, there is only one block and all individuals are computed, but in the DES only the individuals in the same block and the contiguous blocks are computed. In the first case, the algorithm complexity is $O(N^2)$ and in the second it is $O(N^2/n)$ where n is the number of LPs.

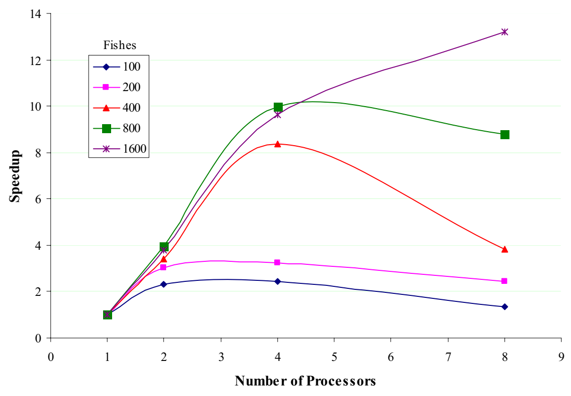

Figure 8: Speedup PVM

As can be observed, the distributed simulation only extracts benefits when the relationship between computing time and communication time is a high value. For small quantities of individuals, the messages sent to discover the neighbours in the processors that have the contiguous zone of the model affect more than the distributed computing and we can define a lower limit in (minimal) 100 individual by processing element.
This conclusion permits to generate large expectative for large quantities of individuals because the biologists have to study the large colonies behaviour of fish (between 20000 and 10^5 individual).
The second developed simulator was based on the MPI [11] communications library. This simulator allows us to simulate large quantities of individuals (more than 10^5) and with a high stability on a 32 nodes cluster.
The figure 9 shows the speedup until 25000 individual and 32 computing nodes. As can be observed, the results are very good and permits to conclude that the distributed simulation is a useful and versatile tool for high performance simulation and where it is necessary to manage a high data volume.

5. CONCLUSIONS

Ecological system simulation is a field that requires large computing capacity powers when individual oriented models are used. Distributed Event Simulation (DES) is an excellent tool for solving this kind of problem.

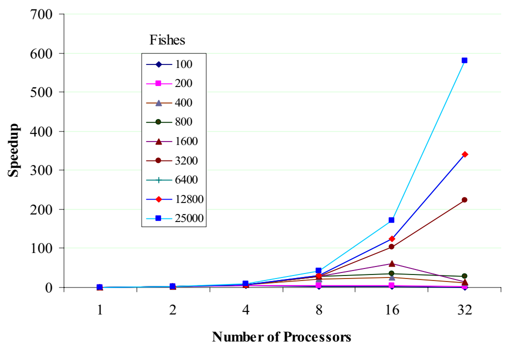

Figure 9: Speedup MPI

The present work analyses the DES in Individual oriented models (specifically fish movement) using a distributed simulation based on conservative algorithms. The results obtained demonstrate that it is a viable option but that conservative algorithms impose a strong limitation on the method possibilities.
In order to measure the benefits of DES, a set of experiments has been performed. These experiments were performed using a sequential simulator and the results were compared with the distributed simulators. The conclusion with respect to the distributed simulation is that the one based on PVM obtains better results than the simulation based on MPI for a lower number of individuals but it is not viable for a huge number of individuals. It is also demonstrated that for these values, the optimum number of individuals is from 100 to 200 per computing node. On the other hand, when the number of individuals is increased, MPI presents better behaviour for distributed simulators based on individual oriented models.
In conclusion, it can be observed that DES is recommendable for large simulation environments (7,000 individuals or more) but there are also acceptable results for a smaller number of individuals. The main restriction of DES conservative simulation is its communication and blocking time but even with these times, the simulation speedup is acceptable.
The future works will be aimed to obtain more performance from the simulation model and physical model implementation. It can be observed that in determined points, the addition of processors doesn’t influence the speedup in a positive way. This can be explained with the fact that the quantity of communications generated between the LP’s tend to grow.
Thus it suggests that to obtain a significant reduction in the execution time it is needed to reduce the communications making model adaptations or using allocation policies to better distribute fish in the processors.

Furthermore, other future elements that will have to be included in the Fish Schools simulator are:

- Improving the conservative distributed algorithm in order to reduce the restrictions imposed by the communication and blocking time.
- Improving the simulation environment in order to enhance the interactive DES simulation with real time animation.
- Improving the Fish School model in order to include predators, obstacles, more than one specie, energy control, and individual ages in order to be able to atend the needs of biologists and ecologists.

6. REFERENCES

[1] Fujimoto, Richard M. Parallel and Distributed Simulation Systems. New York: John Wiley & Sons, 2000.

[2] V. Springel, "The cosmological simulation code gadget-2," in Monthly Notices of the Royal Astronomical Society. vol. 364, pp. 1105-1134, 2005.

[3] Parrish, J.K.; Viscido, S.V. & Grunbaum, D. Self-Organized Fish Schools: An Examination of Emergent Properties *Biol Bull,* 296-305, 2002.

[4] Suppi, R., Cores, F, Luque, E., Improving Optimistic PDES in PVM Environments, Lecture Notes in Computer Science ISSN 0302-9743, Springer-Verlag, Vol. 2329(1), pp. 107-116, 2002.

[5] Huston, Michael; DeAngelis, Donald & Post, Wilfred. New Computer Models Unify Ecological Theory. In: *BioScience*, Vol. 38, No. 10, pp. 682-691, 1988.

[6] Mostaccio, Diego & Suppi, Remo. Simulation of Ecologic Systems Using MPI. Lecture notes in computer science. **ISSN** 03029743. Springer Verlag, Vol. 3666, pp. 449-456, 2005.

[7] Smith, M. J. Models in Ecology, Cambridge University Press, 1994.

[8] Kreft, J. Booth, G, Wimpenny, W. T. J. BacSim, a simulator for individual-based modelling of bacterial colony growth, Microbiology Journal, 144, pages 3275-3287, 1998.

[9] Lorek, Helmut and Sonnenschein, Michael. Using parallel computers to simulate individual-oriented models in ecology: A case study. In Proceedings of the 1995 European Simulation Multiconference (ESM), pp 526-531, June 1995. http://citeseer.ist.psu.edu /lorek95using.html.

[10] Geist, A., Beguelin, A., Dongarra, J, Jiang, W., Manchek, R., Sunderman, V. Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing. The MIT Press, 2000.

[11] Message Passing Interface Forum. MPI: A Message-Passing Interface standard. Technical report, 2004. http://www.mpi-forum.org.

[12] Remo Suppi, Daniel Fernández, Emilio Luque. Fish Schools: PDES Simulation and Real Time 3D Animation. Lecture Notes in Computer Science 3-540-21946-3, ISSN 0302-9743 Springer-Verlag EC. Vol.: 3019, pp.. 505-512, 2004.