Escola Tècnica Superior d'Enginyeria
Departament d'Arquitectura de Computadors i Sistemes Operatius

Increasing the Scalability and the Speedup of a Fish School Distributed Simulator

Master thesis submitted by Christianne Dalforno
in fulfillment of the requirements for the degree of
Master by the Universitat Autònoma de
Barcelona.

Bellaterra

July 10, 2007

Dr. Remo Suppi Boldrito, from the Computer Architecture and Operating Systems Department at Universitat Autònoma de Barcelona,

HEREBY CERTIFIES:

That the work entitled "**Increasing the Scalability of a FishSchool Distributed Simulator**" has been written under my supervision by Christianne Dalforno, and that it constitutes the experimental work carried out within the Postgraduate Program for Computer Architecture and Parallel Processing.

Bellaterra, July 10, 2005.

Remo Suppi Boldrito

Acknowledgments

In the end of a work like that, we have to be conscious that we were not alone and recognize the direct and indirect contribution that we received. I would like to register here my acknowledgements to some of those persons that helped me to achieve this first objective. I have to first apologize me with the ones that will not appear here because I definitely think that all people that participated of my history influenced me in a way but I really can not write here my biography.

First, I would like to thanks God.

Thanks my family that suffers like or maybe more than I suffer with the distance but always encouraged me in the more difficult moments.

To Josemar Sousa that indicated me to this pos-graduation program.

Thanks to Emílio Luque to confidence in my potential and to give me the opportunity to participate of this pos-graduation program.

To Dolores Rexachs that takes care of all of us with all the attention that one can offers.

To my director Remo Suppi, thanks to all the contributions to my formation and work. Thanks for helps me in the difficult moments, to believe in my potential and to the patience.

To Diego Mostaccio to contribute with my work and to give me some open lines to choose.

To all professors from DACSO - Computer Architecture & Operating Systems Department that direct or indirect contributed to this work.

To Jordi and Dani for every thing.

To Gemma Roque for helped me so much and for the unforgettable cherries.

To my friends that are far from me. They were essentials.

To my friends that are near to me and helped me a lot in this two years with technical and emotional support: Angelo Amâncio Duarte, Genaro Costa, Guna Alexander Santos and Eduardo Argollo.

A special thanks to my great friend, imported from São Paulo, Rodrigo Godoi. I will never forget the days that he helped me to eat.

To Letícia and Raul that take care of me with so much affection.

In my academic life, I had some good examples that helped me to be what I am today the ones that are my idols: my great friend André Santanché, my godfather Thomaz Cruz, my teachers, all the investigators that I had contact in the Hospital Universitário Professor Edgard Santos.

To Leandro and Manuela to offered me a place to stay in a first moment in Spain.

To finish, I would like specially thanks all my students. With them, I learned so much interesting things.

Bellaterra, July 10, 2007

Christianne Dalforno

Prologue

The computer simulation has been used for many years as a powerful tool in the study of systems. Thus, it was applied to many knowledge areas.

In this work, we are especially interested in the application of this tool to the study of ecological systems. It is an extension of another work in which a distributed three-dimensional fish school simulator was developed.

The behavior of the fish school was modeled as an Individual-oriented Model (IoM). Thus, some rules were defined to represent the behavior of a fish. The reaction is, basically, the definition of the fish direction.

A fish can react to its neighbors in three ways: attraction, parallel orientation and repulsion. Therefore, each step of simulation consists in to select four neighbors to each fish and calculate the reaction of the fish to each one of its neighbors. So, as a result, a final reaction is calculated based in these previous calculations and, finally, the new position of each fish is defined.

The system was developed intending to simulate populations formed by thousands o fish. Thus, it was done the parallelization of the simulator to achieve an appropriated time of simulation.

Previous experimentations were done with the original simulator and the results signalized two points that could be improved:

- The communication strategy
- The data processing

Thus, this document proposes a new communication strategy and changes in the way data is processed with the aim to improve the speedup and the system scalability.

The experiments done with the new versions of the simulator showed that the simulation time reduced without loose of accuracy.

Table of Contents

List of Figures

Chapter 1

INTRODUCTION

The computer simulation is a tool that reproduces the behavior of real or imagined systems based in a predefined model. In this way, this resource has been explored in many studies aiming to better understand systems behavior and preview the consequences of a change in a system, etc.

In the available literature, we can found examples of the using of simulation by many knowledge areas:

- Climatology: simulation of the climatic variations (Sato, Kitawaki et al. 2002; Habata, Yokokawa et al. 2003).
- Prevention of fire: simulation of the advance of the front of a fire (Andrews 1986; Finney 1998; Andrews, Bevins et al. 2005; Bianchini, Cortes et al. 2006).
- Communication: simulation of systems of video under demand (VoD) (Yang, Hernández et al. 2006; Yang, Hernández et al. 2006).
- Ecology: simulation of real systems applying to models oriented to the population or the individual (Suppi, Fernández et al. 2004) (Suppi, Munt et al. 2002; Mostaccio, Suppi et al. 2004; Mostaccio, Suppi et al. 2005; Mostaccio, Suppi et al. 2005; Mostaccio, Suppi et al. 2005; Mostaccio, Suppi et al. 2006).
- Management of resources in cluster of computers: (Hanzich, Giné et al. 2006; Hanzich, Lerida et al. 2006).

In this work, we are interested in the application of the simulation in ecological systems studies. Specifically, we study the use of simulation to reproduce the behavior of fish schools.

Some kinds of fish present the characteristic behavior of being together for a long time maintaining a self-coordinated movement without the presence of leaders.

This behavior gained the attention of the biologists interested in investigating this kind of formation.

Nowadays, this kind of study gained another motivator: the consequences of the commercial fishing. The issue of April of 2007 of the National Geographic (Montaigne, Warn et al. 2007) presents an overview of the situation of the worldwide fishing. In fact, the consume of fish has been growing and, in around 50 years, the fishing went from about 30 million tons to almost 100 million tons.

One of the species that has been suffering a lot with the growing of its consume is the blue tuna, one of the preferred fish to prepare the sushi. The situation of this specie is so critical that was created the *Comisión Internacional para la Conservación del Atún Atlántico* (CICAA), responsible to administrate the reserves of the blue tuna.

Thus, the tendency is the growing of the interests in the investigations about fish behavior, reproduction, development and exploration. It has to be found a balance between commerce interests and preservation of the species. The computer simulation can be a helpful tool in this process, as a resource to test actions and theories and to make predictions about the consequences of the commercial fishing to the species of fish.

To attend the needs of this kind of studies, it will be necessary the development of more realistic and complex simulators. It will be important to represent the characteristics of the ambient that influences the development of the species. It will be important to simulate some different species to know how can they both coexist and develop, principally, if there is a relation of prey-predation between them.

There is a kind of model that establishes some rules to be applied in each simulated individual; it is called Individual-oriented Models (IoM). The IoMs will be helpful to the development of a simulator with the mentioned exigencies.

The complexity estimated of these simulators demands great computation processing power. One possible solution to this problem is the use of parallel computing.

The interests of this work goes in this way: the use of the power of the parallel computing applied to the needs of individual oriented models, in special to the simulation of the fish schools.

Parallel or distributed simulation is the use of many processor elements interconnected by a communication network to execute a simulation. The development of this kind of system involves taking care of many questions as:

- how to parallelize the application;
- to identify how the parts of the simulation interact;
- what is the dependency among this parts;
- how the time will be treated and represented, and others.

In addiction to reduction in the duration of the simulation, Fujimoto (Fujimoto 1999) listed some other benefits in to distribute a simulation as for example the execution of the simulation in a set of computers geographically distributed, the possibility of integrate simulators developed to computer from different manufacturer and the possibility of to increase the fault tolerance.

The present work is a sequence of another one in which a distributed three-dimensional Fish School simulator was developed, validated and verified (Mostaccio 2007). To develop this simulator, Mostaccio based in a two-dimensional IoM that was defined by Huth and Wissel (Huth and Wissel 1992).

The two-dimensional fish school IoM defined by Huth and Wissel successful represents the movement of groups of fish in a two dimensional space. The model achieves the behavior of the group through the application of some rules on each simulated individual.

According to this model, a fish can present one of three possible reactions to each one of its neighbors: attraction, parallel orientation or repulsion. All this reactions are mathematically expressed in two-dimensions by the original model. So, to develop a three-dimensional simulator, Mostaccio needed first to adapt it to represent the individuals, the space and the reactions in a three-dimensional way.

Other point of difference between the two works was the quantity of individuals that form the simulated population. Huth and Wissel worked with populations of 8 individuals. The objective of Mostaccio was to simulate populations with thousands of fish. This population's length generated a power of computation demand that justified the parallelization of the algorithm.

As a result, this system can simulate a big population of fish dividing the limited simulated space into fixed slices that are each one assigned to a processor in a parallel architecture. Each one of these slices corresponds to a logical process (LP)

The way the application was parallelized caused two situations that demand the exchange of messages between consecutive LPs:

- when a fish has to migrate from one LP to another LP;
- When a fish have neighbors placed in another LP.

Experimentations done with this application showed a limited scalability once that the communication strategy used demands the exchange of a great quantity of messages in some situations.

It was observed that this simulator presented a good computation time reduction achieved by its parallelization. This reduction points to another possible problem: the sequential version of the simulator executes unnecessary operations. Then, another objective of this work is to analyze the data processing to identify these avoidable tasks and explore it to improve the speedup of the system.

Thus, to improve the scalability of the distributed simulator, a revision in the communication strategy was necessary and an analysis of the computation process to discover some tasks that could be excused. Then the efforts in that work were in improving the speedup and the scalability of a distributed three-dimensional fish school simulator.

Then, to achieve a better speedup and scalability of this simulator, this work presents two main improvements of previous simulator architecture:

- It proposes a new communication strategy that permits to control the number of messages exchanged by the logical processes of the system with the aim to increase the scalability of the system.
- It establishes some changes in the original algorithm to avoid unnecessary operations to reduce the simulation time.

In chapter 2 can be found an overview of the theory used in the development of this work, so, it can be read about computer simulation, individual oriented models, fish school simulation and distributed simulation. In chapter 3, we present the original simulator and the results of the experimentations done with it. In chapter 4, we explain

the new communication strategy proposed and the changes done to reduce the simulation time avoiding unnecessary data processing. Finally, in chapter 5, we talk about conclusions and future works.

Chapter 2

COMPUTER SIMULATION

"A *computer simulation* is a computation that models the behavior of some real or imagined system over time." (Fujimoto 1999)

This offer, to many fields, the possibility of to analyze the behavior of the variables involved by controlling, monitoring and setting some of them. This can be less expensive, dangerous or complex than to observe the real system or can even make possible the observation and experimentation of the imagined systems.

The simulation as analysis tool presents some characteristics that make it interesting for their use (Cores 1999):

- Great flexibility: diverse parameters of the system can be easily modified for their later study. Using simulation, the effects of certain changes in the system can be observed and analyzed by modifying the model and observing their effect on the behavior of the modelled system.
- Efficiency: the simulation can compress the time so that long intervals on the real system pass in few seconds on the simulated model. This allows saving great amount of time during the analysis of the real system.
- Isolation with respect to the physical system: it makes possible the study of a system or part of them without affecting the real system. The importance of this characteristic grows with the increase of the complexity of the iterations that evolves the analysis.
- Helps understanding the real system: the detailed observation of the system being simulated can lead to a better knowledge of it and some improvements to the real system that would be difficult to perceive can be easily observed. The simulation of complex systems helps to identify the most important variables of the system and the relation between them.

- Experimentation with physical systems: the simulation can be used to experiment new situations about which there is few or no information, verifying what happens with the real system. When new components in a system are introduced, the simulation can help to find zones of conflict or other problems that would affect in an undesired way the operation of the real system.

Fujimoto (Fujimoto 1999) pointed some interested communities that have been working in this field and have been using the simulation in your activities that are: the high-performance computing, the defense and the interactive gaming and Internet community. These communities have developed many applications so that nowadays we have the concepts of simulation applied to military applications, entertainment, education and training applications, digital logic and computer systems and others.

The system been simulated is called physical system and it has a state that evolves over time. Then a simulation consists, basically, in to represent the state of the system and make the correctly changes in it, controlling the time to reflect the way the things happen in the physical system. (Fujimoto 1999)

A simulation can be classified in continuous or discrete according to the way the time is treated. “In a continuous simulation, state changes occur continuously in time, while a discrete simulation the occurrence of an event is instantaneous and fixed to a selected point in time.” (Ferscha and Tripathi 1994)

Depending on the nature and the results that are desired to obtain a system of a class can be modeled as a model from another class. For example, a continuous system could be simulated with a discrete model. (Banks, Carson et al. 1995)

An important aspect to consider in a simulation is the form of progression of the time. The time can progress in steps of constant size (time driven) or can assume the time of occurrence of the event with smaller time stamp after its execution (event driven). (Mostaccio 2007)

In order to improve the results´ precision of a time-stepped simulator, the time of the step can be reduced but it increases the duration of the simulation.

An algorithm that implements a time stepped simulator consists of the following elements (structures of data) (Misra 1986):

- Clock: to each advance of time, the simulation's clock will also advance.
- Set of state variables: These variables stores the values that represent the different states of the system being simulated.

The discrete event driven simulation has an advantage of to bypass the time in which no events occur. In addition to the data structures mentioned before, this kind of simulator maintain an event list (EVL) with the events scheduled to occur. Each event has a time stamp and the EVL is ordered in relation to it.

Each event in the EVL is executed and, in this moment, the clock is actualized to store the value of the time stamp of the event that was just executed. So, the time interval between the clock time and the time stamp of the next event to be executed (the first in the EVL) is jumped once that there is no event to occur. This functioning helps to reduce the simulating time.

Initially, people used to develop simulators trying to establish the behavior of a population like a monolithic block. In addition to the fact that it should be very difficult to describe the behavior of a population, the biologists wasn't satisfied with this representation of the reality once that it wasn't possible to describe the differences between the individuals and the interferences that one would causes in another one. (Huston, DeAngelis et al. 1988; Lorek and Sonnenschein 1995)

Then the investigations went through the direction of to model a population based in simple rules applied to a single individual. The result of this was the developing of the Individual-oriented Models (IoM), i.e. a model where the population behavior is achieved by the definition of the rules that define the behavior of a single individual relating with the other individuals and the environment. One example of application of this kind of models, and object of study of this work, is to represent the behavior of Fish Schools, i.e. a model that represent the movement of fish species. (Huth and Wissel 1992)

When this kind of model is used to represent big populations, it generates a quantity of computation, principally if involves a graphic output with interactivity, that makes difficult to do it in an appropriated time using a single processor with sequential

algorithms (Lorek and Sonnenschein 1995). Thus, it had made necessary to use the power of the parallel and/or distributed computation. (Mostaccio 2007)

2.1 Individual-oriented Models

The Individual-oriented Models (IoM) 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 behavior of a group of individuals.

To model a system based in a population, it is 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. (Huth and Wissel 1992) However, the individuals have to adapt itself and its environment to achieve their main objective of reproduction. This adaptation is the result of the interactions of the individual with the environment and other individuals that it has contact and not as a decision based on the conscience about its population.(Grimm and Railsback 1958) The populations' properties will emerge from the actuation of those adaptive individuals.

Therefore, "(…) individuals are the building blocks of ecological system. The properties and behavior of individuals determine the properties of the systems they compose" (Grimm and Railsback 1958)

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. Grimm & Railsback, in addition, underlies the importance of this kind of model to the understanding of the interrelationship between individual traits and system dynamics.

Another problem is that, sometimes, it could be very complex to model an ecological system using differential equations or stochastically process.(Huth and Wissel 1992) To define the rules that guide the behavior of one individual should be, in most of the cases, a simpler task.

This work uses an IoM to represent and to simulate the movement of fish species by applying simple rules to each fish in a big group of it.

2.2 A Fish School Individual-oriented Model

Fish schools are big non-hierarchical groups of fish that stay together for a long time in a coordinated movement without the presence of leaders.(Huth and Wissel 1992)

This kind of group formation gained the attention of the investigators that was interested in to explain how does it works. In (Parrish, Viscido et al. 2002) can be found a comparative study of seven fish school modelling works and a new suggestion based on that previous ones.

One of the tools used in these investigations was the computer simulation that helped the scientists to prove their models. So, some models were defined and some computer simulators were developed with different aims. (Huth and Wissel 1992)

Huth and Wissel (Huth and Wissel 1992) defined a model to simulate fish species´ movement obtained with simple rules about the behaviour of an individual fish. Thus, this model is an IoM one, where the behaviour of the population is achieved by the behaviour of each individual.

Based in the needs by the fishes about avoid predation, reproduction and avoid collisions, simple rules can be established about the behavior of each of then in a group.

The mentioned behavior has respect with the direction to where the fish goes through and its position. So, in this Fish School model, each fish is represented by a point in the space and a velocity vector that defines its direction.(Huth and Wissel 1992) 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 neighbors. The parameters to select this neighbors that influence in its behavior is established by the rules:

1. The fish will choose about four (4) neighbors from all of the other fishes in the school.
2. How can be seen in the Figure 1, three radius are established. The fishes that are out of the more far radius are not a valid candidate like the ones that are out of the vision angle of the fish (e.g. the ones that are in 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.

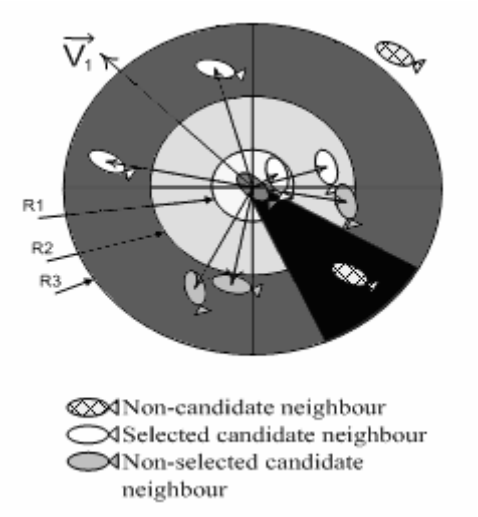

Figure 1: Neighbors selection in the Fish-Schools model

After define the neighbors, it has to be calculated the change suffered by the fish. Each neighbor will influence the fish in one way in accordance to your relative position. There are three radius and then three possible reactions(Huth and Wissel 1992):

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

In a sequential version, this algorithm try to define the neighbors of a fish 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. (Lorek and Sonnenschein 1995)

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

2.3 Distributed Simulation

"Parallel simulation and distributed simulation refer to technologies that enable a simulation program to execute on a computing system containing multiple processors, such as personal computers, interconnected by a communication network." (Fujimoto 1999)

A simulation can be parallelized or distributed in different levels (Ferscha and Tripathi 1994):

1. Level of application: This happens when independent instances of the simulation are assigned to the available processors with distinct input values.
2. Level of subroutine: copies of the subroutines that constitute the program are distributed to the processors to accelerate the event or the data processing.
3. Level of component (physical system): none of the two mentioned distributions makes use of the possible parallelism available in the modeled physical system. With the purpose of obtaining a greater benefit of existing parallelism, the simulated model is decomposed into models of components or sub models. This decomposition directly reflects the inherent parallelism of the model or at least conserves the possibility of some gain during the simulation.
4. Level of events:
    a. Centralized list of events: in this scheme the list of events is one structure of data centralized and administered by a master processor. The acceleration can be obtained distributing concurrent events to a pool of slave processors dedicated to execute them.
    b. Decentralized list of events: the events from different points of the plan space-time are assigned to different processors in a regular or structured way. A greater level of parallelism can be achieved if the simulation strategies allow concurrent simulation of events with different times from occurrence. Schemes that follow this idea require protocols for the local synchronization that cause the increase of the communication.

When a simulation is distributed in the level of events, it is common to divide the simulation into logical process (LP) (Ferscha and Tripathi 1994). This strategy

permits to exploit the parallelism inherent of the model by the concurrent execution of these LPs.

So, the called Logical Process Simulation (LP simulation) is achieved by the cooperation of a group of interactive LPs. In that way, the architecture of an LP simulation has to define a communication system to provide a canal of interaction among the LPs.

Another important detail is the clock representation and the time advance. In a time-stepped LP simulation, there are two ways of implementing the simulated time: as a global clock with a centralized data structure or using a local clock to each LP. In the last case, all local clocks have the same value, as it was a copy of the global clock.

In the event driven simulation, every LP has a local clock and the value of each one of them is not necessarily equal to the global clock. As the LPs can interact generating, sometimes, events to occur in another one, a causality problem can occur. A causality problem is a situation in which some events occur in an order different from the physical system. This can be avoiding executing always the event with smaller timestamp. Nevertheless, to solve definitely the problem it was developed some methods that can be classified as conservative or optimistic.

The conservative methods establish that an event will only be executed when there is no possibility of to receive a message from another LP with an event which the time stamp is smaller than the local virtual time.

The optimistic methods establish that the events will be executed and if a message with a timestamp in the past of the LP arrives it will cause a roll back to recover the stable state of the LP.

Parallelizing a simulation can be done in two ways. The first form is to execute as many simulations instances as available processors. Using this methodology, it can be analyzed the behavior of the system with different values to the input parameters. However, each one of the processing elements has to present capacity to execute the whole simulator.

An alternative to this way is to try to accelerate a single simulation through the cooperation of some processor working together. This form demands the distribution of the algorithm to the available processors. (Ferscha and Tripathi 1994)

It is important to distinguish the operational principles of the parallel machines. One of the available architectures is called SIMD (Single Instruction Multiple Data) where a set of processors execute identical instructions on different data. Each processor has its own local memory to store its programs and private data.

The SIMD machines are physically implemented in shared or distributed memories architectures. The static network interconnection attends the needs of message exchange. The parallel simulation occurs when the synchronism of the SIMD is used to conduce the simulation with some processors.

Another architecture is called MIMD (Multiple Instruction Multiple Data). A set of process are assigned to some processors that work asynchronously. In this architecture, the synchronization is achieved exchanging messages. The called distributed simulator is an application to this kind of architecture with an explicitly codification to the synchronization strategy.

Chapter 3

A DISTRIBUTED THREE-DIMENSIONAL FISH SCHOOL SIMULATOR

How it was mentioned before, this work is an extension of another one in which a distributed Fish School simulator was developed.(Mostaccio 2007)

The developed simulator is based on the Huth and Weissel´s two-dimensional model (Huth and Wissel 1992) that represents the behavior of fish schools in a two-dimensional space. The original model, an Individual-oriented Model (IoM), consists in a definition of some rules to be applied to the individual of a group of fish.

The rules consist in the definition of the reaction of each fish to its neighbors based in the distance between them. There are three possible reactions: attraction, parallel orientation and repulsion

Each one of these reactions is mathematically expressed to achieve the two-dimensional final movement of each simulated fish.

Although the simulator developed by Mostaccio was based in this model, it simulates a three-dimensional space. Thus, some adaptations were done to change this model to simulate the fish movement in a three-dimensional space. All data representation had to be changed to reflect the three dimensions of the space. All the calculations that reflect the reactions of the fish had to be redefined to produce the same behavior but as a three-dimensional movement

Huth and Wissel worked with populations of 8 individuals which was sufficient to attend the needs of their studies. However, Mostaccio (Mostaccio, Dalforno et al. 2006; Mostaccio 2007) directed his efforts to simulate big groups formed by thousands

of fish. This population's length generated a power of computation demand that justified the parallelization of the algorithm.

As a result, a distributed three-dimensional fish school simulator was developed and validated. The results presented by the simulator proved that the three-dimensional mathematical model proposed represents a similar behavior presented by the Huth and Wissels' two-dimensional model.

Experiments were done with the application to simulate a population of 640.000 individuals. The experiments showed problems with the scalability caused by the communication strategy. The reduction of the complexity achieved by the parallelization of the simulator suggests that analyzing the data processing it should be find tasks that can be avoid without loosing accuracy.

3.1 Mathematical Model

The first effort of the Mostaccio's work (Mostaccio 2007) was to extend the Huth and Wissel's two-dimensional model (Huth and Wissel 1992), in a way to simulate a three-dimensional space. To do that, it was necessary to revise the mathematical model starting by the definition of a fish that is represented, now, by a 3D coordinate that defines its position, and a velocity vector that defines its direction. So, the i^{th} fish should have:

- Position, P_i (x_i, y_i, z_i)
- Velocity, Vi (v_x, v_y, v_z).

One of the changes is in respect to the influence radius. Now those don't represent circumferences any more, but spheres and the blind area is, now, delimited by a cone. The areas defined by these spheres and the cone will be used to choose the neighbours of a fish.

Once that the neighbours are selected, instead of a rotation angle to each of them, it will be calculated a vector V_{ij} that represents the reaction of the i^{th} fish to the j^{th} neighbour. All the reactions calculations had to be redefined to produce the same behavior in the fish but as a three-dimensional movement. Following, in this topic, we describe each one of them.

The defined reactions have to be combined in a way to establish the new velocity vector of the i^{th} fish. The new velocity vector (V_i) will be the sum of the previous calculated vectors. So,

$$\vec{V}_i = \sum_{j=1}^{4} \vec{V}_{ij} \quad (1)$$

Attraction

The attraction causes a change in the velocity vector that the fish starts to swimming in direction to its neighbour. In that way the fish tend to stay together in a group. To produce this effect, the new velocity vector is defined by the difference between the positions of the two fish involved.

So, the velocity vector to the interaction between the i^{th} and the j^{th} fish is:

$\vec{V}_{ij} = \vec{P}_j - \vec{P}_i$ *, where P_j and P_i are the 3D-coordinates of the j^{th} and the i^{th} fish.* (2)

Parallel orientation

In this case the i^{th} fish will change its velocity vector to be equal to the velocity vector of its neighbour. This reactions in addiction to the first one, maintain the group together with its individuals swimming parallel.

So,

$$\vec{V}_{ij} = \vec{V}_j \quad (3)$$

Repulsion

The repulsion does the i^{th} fish goes throw in a direction perpendicular to the way that the j^{th} fish goes.

Mostaccio defined that to find the vector perpendicular to V_j is necessary to put the vectors V_i and V_j in the origin of the coordinated system. After that it is necessary to

find the plan that contains the origin of the coordinated system and is perpendicular to V_j. Once that the plan is choose, it has to be found the vector V_{ij} that:

- is in the plan formed by the vectors V_i and V_j
- is perpendicular to V_j
- and is in the plan perpendicular to V_j.

These definitions will guarantee that the angle of rotation is the smaller one.

All of that can be mathematically expressed by:

$$\begin{cases} x_{ij} = x_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot x_j \\ y_{ij} = y_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot y_j \\ z_{ij} = z_i - \dfrac{v_{x_i} \cdot v_{x_j} + v_{y_i} \cdot v_{y_j} + v_{z_i} \cdot v_{z_j}}{v_{x_j}^2 + v_{y_j}^2 + v_{z_j}^2} \cdot z_j \end{cases} \quad (4)$$

3.2 The Distributed Simulator

Once that the mathematical model was established, the challenge was to parallelize the application.

Lorek and Soneschein (Lorek and Sonnenschein 1995) pointed two ways of to distribute a fish school simulation: static or dynamic. The static distribution consists in randomly assigning the fish to processors. The distribution will not change during the simulation. Dynamically distribute the fish mean assign each fish to a processor based in its position. Therefore, each processor simulates a part of the total space. Finally, the author indicates the dynamically distribution to simulate big populations once that the neighbors of a fish will stay together in the same processor or a neighbor one.

In this case, it was decided to split the application based in the space been simulated. Thus, the three-dimensional space in which the fish are been simulated is divided and each one of the resultant slices are sent to a processor as can be seen in the figure 2. (Mostaccio, Dalforno et al. 2006; Mostaccio 2007)

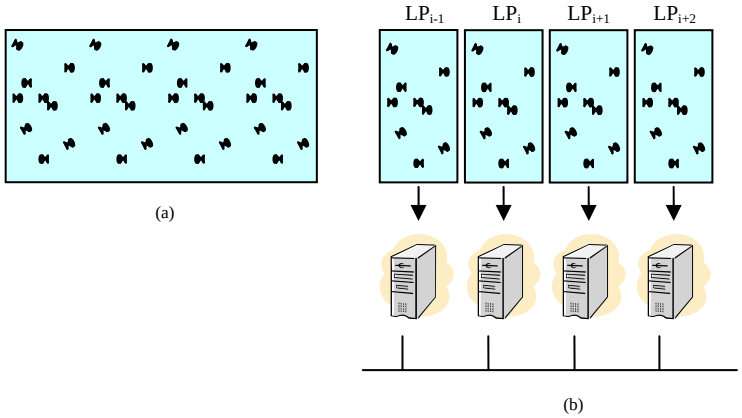

Figure 2: Model distribution. (a) All simulated space. (b) The simulated space divided into slices that will be processed each one by a processor.

This form to divide the space will possibility the occurrence of a situation where a logical process (LP) will need data from the next LP. This will be produced when one or more fish are next to the border of the LP as can be seen in figure 3. (Mostaccio, Dalforno et al. 2006) In this situation, the fish that is in the border needs data from the next LP that consists in the position and velocity vectors of its possible neighbor fish that are outside its own slice.

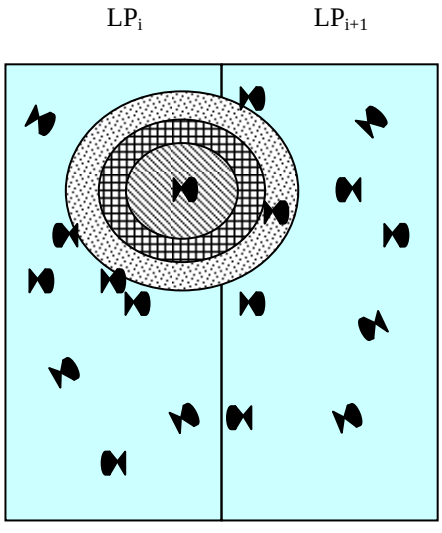

Figure 3: Fish with a neighbor in the next LP.

The solution gave to it was to ask for the needed data always that this situation is detected. Thus for each fish that is next to the border and, because of that it can have neighbors in the next LP, a message is sent asking for the data. To each message asking for data, it will be generated an answering message with the data of the neighbor candidates of the fish.

Another situation, caused by the way of the computation was distributed, is the migration of some fish. When a fish swim to a space that is simulated by other LP, the data referent to it has to be sent to the appropriated LP and erased of the LP where it was.

If a fish swim out of the simulated space, the simulator replaces it in the opposite place that it went out vertically or horizontally. An example of this kind of situation can be seen in the figure 4 where a fish swims out to the simulated space through the right border of the last LP (LP_n). The fish is replaced in the simulated space next to the left border of the first LP. The idea is to consider that the LP_1 is the right neighbor of the LP_n. This was established to maintain the number of individuals being simulated (Mostaccio 2007).

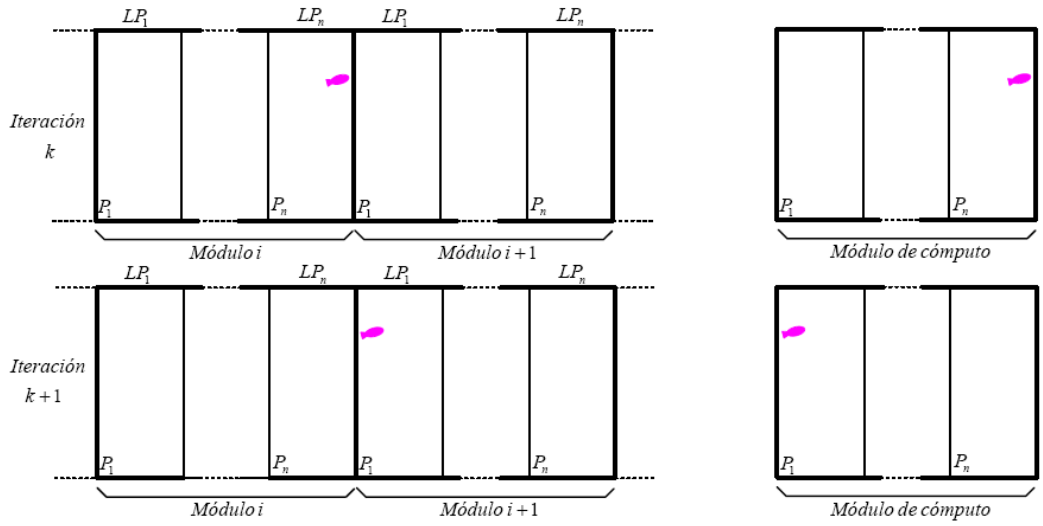

Figure 4: Example of a fish that swims out of the simulated space and is reinserted.

Thus, the communication only occurs between neighbors LPs, i.e. the LP_i only communicate with LP_{i-1} and LP_{i+1}. However, it will be necessary a communication between LP_1 and LP_n as it can be seen in figure 5 to attend the cases in which the fish has to be reinserted in the simulated space.

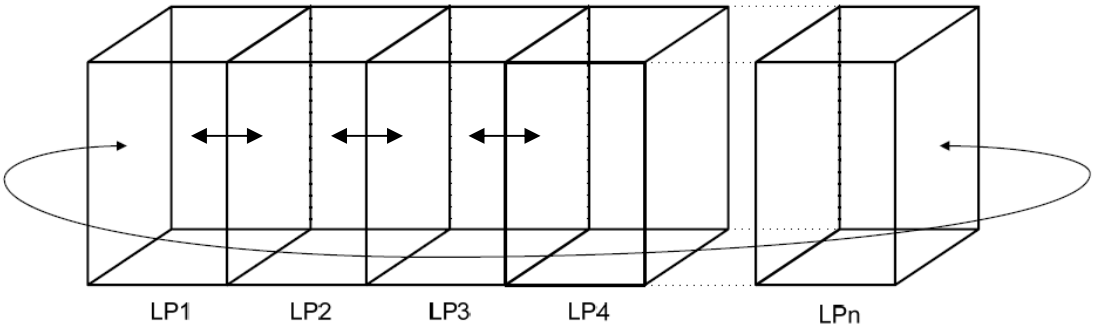

Figure 5: Simulated space divided into slices and the indication of the possibilities of communication between neighbors ones.

3.3 The Algorithm

The task to be executed in the simulation is to define the initial position and velocity vector of all fish. There are two ways to initialize the fish data:

- The user can define the initial position and velocity of each one of the fish using external files.
- The simulator defines the initial position and velocity of the fish by chance. However, it is guaranteed that all LPs will have the same number of individuals. The fish are distributed uniformly in the simulated space.

Once the LPs are initialized, the simulation can start. To each step of the simulation, it have to be calculated the new position and velocity vector of the individuals.

In each interaction of the simulation, some tasks are executed to each one of the fish:

- Choose the neighbor fish candidates. The valid candidates are the individuals inside the bigger sphere defined by the third influence radius and out of the blind vision cone.
- Neighbors fish selection. Four fish have to be selected using the front priority algorithm (Huth and Wissel 1992). This algorithm will choose the fish that are more in the front of the i^{th} fish. If there are less than four candidates, this selection is not necessary and all candidates are considered neighbors.

- Calculation of the influence. The fish will change its velocity vector as a reaction to its neighbors. So, it has to be calculated the change that has to be done referent to each neighbor and after that determines the final influence.
- The velocity vector is changed and the new position is defined.

To execute these tasks, some communication between the neighbor LPs are necessary and some synchronization is needed too. So the algorithm can be represented by the flowchart that can be seen in the figure 6 (Mostaccio 2007).

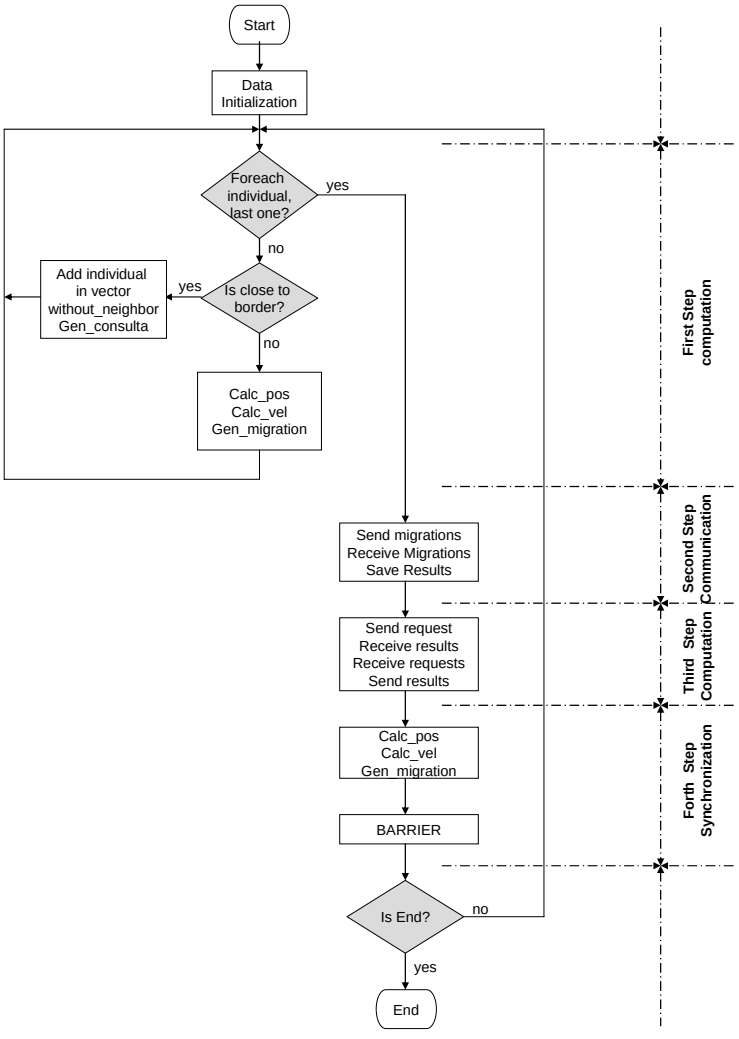

Figure 6: Original simulator's flowchart

Based on the simulator algorithm, it was developed an analytical model to predict:

- The time of simulation based on the number of processors being used and the quantity of individuals.
- The number of processors needed to simulate a quantity of individual in a determined time.
- The number of individuals that can be simulated in an established time using a defined number of processors.

3.4 Experimentation

Here we will show the results of the experiments done with the original simulator. It was done five experiments simulating 640.000 individuals in a space with the dimensions x = 25600, y = 500 and z = 500 to each quantity of LPs. It was measured the time needed to simulate a frame that consists in to calculate the new position and velocity vector of each fish. So, the values showed in the table 1 are the average of these experiments.

Table 1. Experimental results with the original simulator.

LPs	Computation time (seconds)	Communication time (seconds)	Total time (seconds)
1	30785.500	0.000	30785.500
8	489.769	5.176	517.417
16	122.845	3.674	131.548
32	31.293	2.421	36.005
64	8.348	2.492	11.598
128	2.394	2.346	5.255
256	0.758	2.439	3.652

Analyzing the results presented in table 1, it can be observed that it is not advisable to scale the system with more than 128 LPs. The values in the table show that using more than 128 LPs, the communication spends the best part of the time of simulation. This can be explained by the fact that the communication strategy used should, in this cases, to demand the exchange of a great number of messages between neighbors LPs.

When the number of fish been simulated or the number of LPs increases, the number of messages generated can grow in an undesirable way or can stay almost inalterably as can be seen in the table 1. When we used 32, 64 or 128 processors the time of communication measured was around 2.4 seconds in all the cases. The problem is that it harms the scalability of the system because communication time does not decrease in the same way that the processing time decreases.

The great decrease in the simulation time achieved with the parallelization of the application can be explained by the change in the complexity of the algorithm. In the sequential version, the complexity is $O(N^2)$, where N is the number of fish been simulated. This complexity is justified by the fact that to find the neighbors of each fish, its position is compared with all other ones been simulated.

Parallelizing the application, the fish was divided, i.e. according to its position, a fish is placed in a determined LP. Thus, some needless tasks are not realized. It means that a fish that is in an LP will not verify another one that isn't in its own LP or in a consecutive one because the distance between them is grater then the third radius. So, the complexity of the parallel version of the simulator to each processor is $O((N/P)^2)$, where N is the number of the fish been simulated and P is the number of processors. This points to a necessity of to revise the sequential version to find a way to avoid realizing unnecessary tasks.

Chapter 4

IMPROVEMENTS IN A DISTRIBUTED FISH SCHOOL SIMULATOR

As we mentioned before, the principal aim of this work is to improve the speedup and the scalability of a distributed three-dimensional fish school simulator.

To identify possible ways to achieve this objective some experimentations was done with this system. The experiments consisted in to simulate a population of 640.000 individuals in a space with dimensions x = 25600, y = 500 and z = 500.

The results founded with the experiments using the original simulator signalize two points that harms the scalability and the speedup of the system:

- The communication strategy used suggests the exchange of an undesirable number of messages in some situations.
- The simulator executes some operations that could be avoided.

Thus in this work, we propose a new communication strategy intending to limit the quantity of messages exchanged between consecutives logical process (LPs). Limiting the quantity of messages exchanged must reduce the time spent with communication and increase the speedup of the simulator. A better control of the communication time must provide a better scalability of the system.

After realize a better control of the communications, we will analyze the data processing to found a solution that explores the avoidance of unnecessary tasks achieved by the parallelization of the algorithm. This second action intends to reduce the duration of the simulation.

In this chapter, we explain the new communication strategy proposed and the changes done in the simulator to implement it. In addition, we talk about the modifications done to avoid the needless work. It will be shown here the results of the experimentations done after these changes.

4.1 The communication problem

Originally, the communication occurred in some situations:

- When there were fish to migrate to another LP. The strategy used for this communication situation was to send the data of all fish that has to migrate to another LP together in one only message.
- When a fish is next to the LP's border (in a distance minor than the third influence radius) and so needs data about neighbors in the next LP. In this case, it is generated a message asking for data and a message answering the previous one to each fish that is next to the border.

The first presented situation do not causes problems because independent of the number of fish that have migrate, it will be generated only one message to each one of the consecutives LPs (the one in the left and the one in the right) and only one message will be received from them.

In the other situation, the defined model of communication can generate so much messages to be exchanged if the number of fish in the border is great. It occurs because as greater is the number of fish in the border, as greater will be the quantity of messages exchanged by the neighbors LPs.

The challenge to the new communication strategy proposed in this work is to avoid the increasing in the number of messages exchanged by two LPs with the growing of the fish in the border of the subspace simulated by them.

4.1.1 The new communication strategy

As the analysis of the results pointed to a limitation of the scalability caused by the communication, we started to analyze the original communication strategy to find a new solution.

The communications was implemented using MPI. Each message sent using MPI has an additional fixed time that is independent of the quantity of data. If the contents of two messages are sent as one, it will spend less time than if it was sent separately. Therefore, a way to reduce the communication time is trying to reduce the number of messages sent.

The original communication strategy defines that to each fish in a distance from the border of the slice smaller than the third radius, two messages should be exchanged between two consecutive LPs:

- A message asking for the necessary data (the fish's neighbor candidates that are outside of its own slice).
- A message answering the first one.

As the fish are uniformly placed in the simulated space, we consider that the quantity of fish next to one of the borders of the LP is

$$\frac{NR_3}{X} \quad \textit{,where N is the number of fish in the LP, } R_3 \textit{ is the third influence radius and X is the dimension of the LP in relation to the X-axis.} \tag{5}$$

So, messages will be exchanged between two LPs to attend the data demand of the fish that are in one border of one of them.

Thus, an LP will:

- Send $2 \cdot \left(\frac{NR_3}{X} \right)$ asking messages
- Receive $2 \cdot \left(\frac{NR_3}{X} \right)$ answering messages
- Receive $2 \cdot \left(\frac{NR_3}{X} \right)$ asking messages
- Send $2 \cdot \left(\frac{NR_3}{X} \right)$ answering messages

Hence, except to the first and the last LPs, that consider only one border, all of the others will send and receive $8 \cdot \left(\frac{NR_3}{X} \right)$ messages.

The challenge is to reduce the quantity of messages exchanged between the consecutives LPs without avoid to attend the data needs of the fish in the border.

There is a well know technique called halo exchange that suggests a solution to this problem. Bertacchini and Benabén (Bertacchini and Benabén 2005) give an example to explain this technique that consists in updating a data array in each iteration

of a system execution. To change each position of this array it is necessary to use the data of the neighbor positions. If the array is divided to be processed in parallel, as can be seen in figure 7, what happens is that each process needs data that is placed in another processor.

Therefore, the boundary cells of each sub-array have to be sent to the neighbors to be used. This data is called hallo cell or ghost cell.

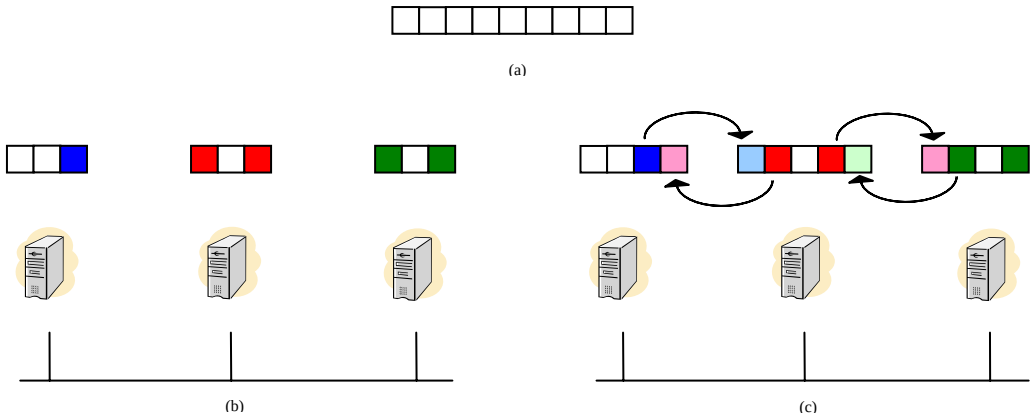

Figure 7: Example of halo exchange. (a) A vector to be actualized. (b) The same vector distributed into 3 processors. (c) Example of the halo cells exchange.

We can apply this solution to the communication problem of the fish school simulator. The idea was in instead of wait for a message asking for the data related to one fish that is next to the border, it could be sent previously to it consecutive LP all data that could be used to a fish in that situation. It means that the data about all fish that is in a distance smaller than the third radius has to be sent to the appropriated consecutive LP.

The idea is to send the data about the fish that are in a distance of the border that can interest to a fish in a neighbor LP. Thus, if a fish is next to the border it will not need to send a message asking for data because this data was just received by the LP.

As can be seen in figure 8 the received data will be used as an extension of the slice and thus the neighbors candidates of the fish next to the border will be find locally. This new strategy guarantees that the maximum messages sent by an LP to be used for choose the neighbors of the fish in the border will be two and thus each LP will receive one or two messages with data that extend its simulated subspace to the right and to the left.

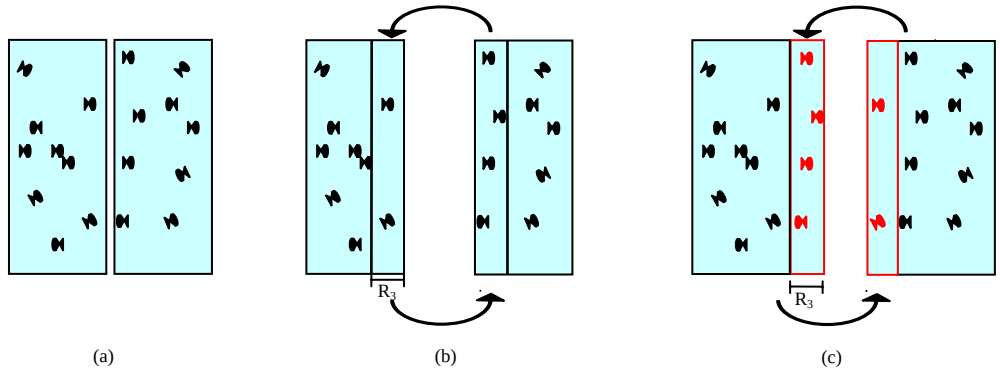

Figure 8: Example of a slice extension. (a) Two consecutive slices. (b) The area into the slices that contains the fish that can be used by the neighbor slice (i.e. the fish that are in a distance smaller than R_3). This area defines what data will be sent to the neighbor LP. (c) The slices extended with the received data.

4.1.2 The new algorithm

The implementation of this new strategy demanded some changes in the algorithm.

Originally, when the first stage of simulation was been executed, for each fish in the border that was identified, it was sent a message to the appropriated LP asking for the needed data.

Each one of those messages was received and processed to the destined LP in the second stage that is a communication stage. In this stage, the LPs exchange the answers to the previous messages.

Another computation stage occurs and if there were identified migration cases, it will be sent in the last stage that is a stage of synchronization. The synchronization is explicitly done using barrier.

The new communication strategy offered a simpler flowchart to the algorithm that can be seen in figure 9.

The first stage is a communication one. In that moment all data about the fish that are in the border, are sent to and received by the appropriated LP.

Once that all data needed to the computation can be found locally by all LPs, the data computation stage starts. While the computation is done, the cases of migration are identified.

After change the values of position and velocity of all fish, the data related with the migration is sent to the appropriated LPs. It is the last stage.

There is an implicit synchronization executed by the communication execution. So, the barrier is not required any more.

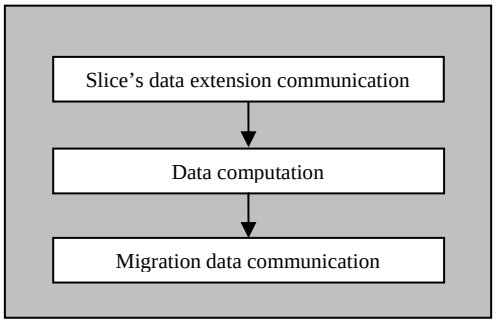

Figure 9: Simple flowchart of a step of simulation using the new communication strategy.

4.1.3 Experimentation

Once that the new communication strategy was implemented, we did some experiments to compare with the previous ones done with the original simulator.

So, we simulated a fish school with the same quantity of individuals of the group simulated before (640.000 individuals). The dimensions defined to the simulated space were the same used before. These were done to guarantee that the LPs would have the same load that in the experimentations done with the original simulator.

Comparing the new data in table 2 with that presented in table 1, we can see that the communication time reduced with the use of the new strategy.

Table 2. Time mesurement done during the execution of the simulation of a frame.

LPs	Computation time (seconds)	Communication time (seconds)	Total time (seconds)
1	22272.500	0.000	22285.800
8	357.257	14.401	374.427
16	90.469	0.830	92.638
32	23.118	0.272	24.191
64	6.238	0.185	6.857
128	1.805	0.174	2.182
256	0.592	0.101	0.807

So, the proposed communication strategy improved the speedup of the presented distributed application using simple resources of communication offered by MPI.

When one has to establish a communication strategy, it has to taking in account that trying to adjust the system to send only useful data can generate an additional message demand to define what is truly necessary. Another problem is that the data will be sent in small messages instead of all together in an only message what increases the communication time necessary to send the same data. As the data is sent in the moment that it is required, some data will, probably, be sent more than once.

Thus, anticipating the data sending will causes that some needless data will be sent. But the time spent to send this extra-data will be, normally, smaller than the time spent in asking for the data when it is necessary.

This changing in the strategy can be applied in systems that have process that needs data from other processors. In some kind of systems, it will be possible to send the data previously. It reduces the communication because it will not be necessary to ask for information, but in some cases, some sent data would not be used.

The new communication strategy defined in this work will be very useful in future works in a way that it makes more feasible to increasing the parallelism of the application by doing in the same time data processing and communication. It is possible because only a limited number of fish needs the data that come from another LPs. So, the communication can be done at the same moment that the fish that are in the centre of the slice (in a distance greater than the third radius from the borders) are being processed.

4.2 Excusing unnecessary work by the space subdivision

The great reduction in the complexity, from N^2 to $(N/P)^2$, suggests loose of results accuracy. However, what really occurs is that the way the application was parallelized helped avoiding the execution of unnecessary tasks.

To take more advantage of this, we start to analyze the data computation stage to identify the tasks in which the simulator spends more time.

Each step of the simulation consists in to calculate the new position and velocity vector of each fish. Therefore, in each step we will have the execution of the flowchart showed in figure 10 to each fish been simulated.

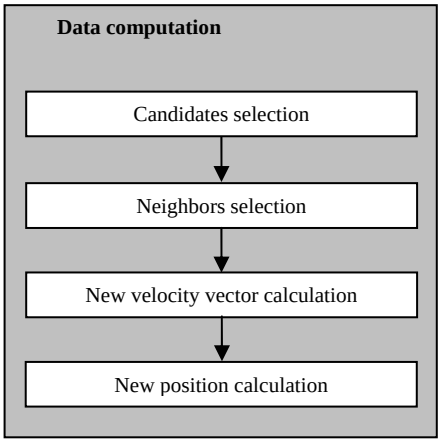

Figure 10: Flowchart of the tasks realized to each fish simulated in one-step of simulation.

The idea is to identify what part of the computation is not executed, in the parallel version, to use this to avoid unnecessary work.

4.2.1 The space subdivision and the avoidance of unnecessary work

How can be seen in table 2 (presented in topic 4.1.3), the computation time represents, generally, more than 90% of the total time of the simulation. It can be observed that increasing the number of processors, this percentage decreases.

The computation time is almost all spent executing the first task (the neighbor candidates selection) realized by each fish in a step of simulation (figure 10). This can be observed in table 3, where is shown the time spent in each computation task.

Table 3. Time (in seconds) spent in each one of the computation tasks realized in one-step of simulation.

LPs	Candidate Selection Time	Neighbor Selection Time	New position Calculation time	New Velocity Vector Calculation time	Computation Time
8	353.792	1.932	0.167	0.312	357.257
16	88.836	0.892	0.083	0.150	90.469
32	22.395	0.392	0.038	0.060	23.118
64	5.889	0.189	0.019	0.028	6.238
128	1.631	0.093	0.009	0.014	1.805
256	0.504	0.046	0.005	0.007	0.592

The Original Neighbor Candidates Selection Algorithm

In the original neighbor Candidates Selection Algorithm, to select the neighbor candidates of a fish it is necessary first to calculate the distance between each fish and the others to select all the ones that are in a distance lower than the third radius excluding the individual in the dead area. (Mostaccio 2007)

In a sequential version, this algorithm will have the complexity $O(N^2)$.This contrasts with the parallel version that has approximately the complexity of $O((N/P)^2)$. (Mostaccio 2007)

This complexity gives to the parallel simulator a super scalability without loose exactitude. What really occur is that, in the parallel version, the fish are divided by the processors based in its location in space. So, it can restrict the neighbor candidates selection to the group of fish assigned to an LP and the ones received before the

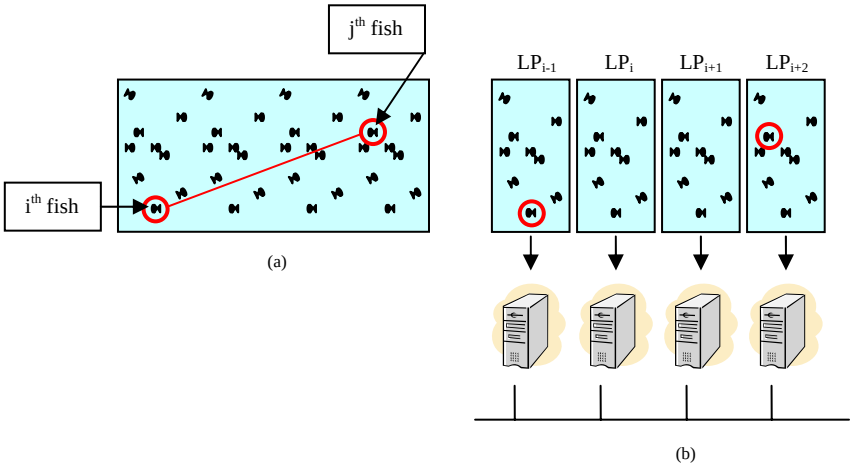

Figure 11: Unnecessary work avoidance example.

execution of the task.

Figure 11 shows an example of this kind of situation. The i^{th} fish is the fish been calculated. To define its neighbor candidates, all others fish that can be seen in the simulated space (a) has to be analyzed in relation to the i^{th} fish. So the j^{th} fish that is marked will be consulted too.

In the parallel version, the j^{th} fish will not be analyzed because the neighbors of a fish that are in a LP can be found locally, or in a consecutive LP. The way the

parallelization was done, subdividing the simulated space, pre-established subspaces where a fish will not found a neighbor.

The observation that in the parallel version it wasn't necessary to verify all simulated fish to select the neighbors of one of them gave us the idea of to try avoiding execution of those unnecessary tasks in the sequential version and after that to apply this solution to each LP of the parallel version.

The New Neighbor Candidates Selection Algorithm

To achieve the new propose of to avoid consult all fish to determine the neighbors candidates of one of them, it was necessary to reflect the position of the fish in the simulated space in the position of them into the fish vector.

Thus, the strategy used was to sort the fish's data. The ordering is done in function of the x coordinate of the position of the fish. To equal values of x, it will be used the y coordinate to ordering the fish's data and to equal values of y it will be used the z coordinate.

To sort the data into the vector it was used the sort algorithm of the Standard Template Library (STL).

Once the data are organized, a more efficient algorithm of candidates choose can be developed. In our new propose we will only avoid to consult the fish that are in a distance greater than the third radius in relation to the x coordinate.

So, now, the neighbors candidates of a fish will be looked for since the vector's position of the fish to the vector's beginning until find a fish that is in a distance grater than the third radius or until arrive the vector's beginning. And the same thing will be done since the fish vector's position to the vector's end.

Considering that the fish are uniformly distributed in the space, we can assume that the quantity of fish that exist in a space with dimensions x = R_3, y = Y and z = Z (with Y and Z been the dimensions of the total simulated space) is equal to:

$$N_{R_3} = R_3 \cdot \left(\frac{N}{X} \right) \quad \text{, where } N \text{ is the total of the simulated fish, } R_3 \text{ is the third radius and } X \text{ is the } x\text{'s dimension of the total simulated space.} \tag{6}$$

Thus, the new algorithm to choose neighbor candidates has complexity $O((N^2 2R_3)/X)$, to the sequential version.

In the parallel version, this algorithm will have, to each LP, complexity $O((N^2 2R_3)/XP)$, where P is the number o LPs.

4.2.2 Experiments

In experiments realized using this new algorithm, we found the times show in table 4.

Table 4. Time (in seconds) spent with communication, computation and in the task of selection of neighbor candidates

LPS	Communication Time	Computation Time	Candidate Selection Time	Total Time
1	0.00	57.10	35.50	57.10
8	1.15	11.92	9.20	16.21
16	0.73	5.94	4.57	8.39
32	0.21	2.99	2.30	4.13
64	0.40	1.37	1.02	2.29
128	0.12	0.76	0.59	1.20
256	0.09	0.41	0.32	0.63

The results show that the use of the new neighbor candidates selection algorithm reduced a lot the simulation time.

It is important to know that this great reduction is in function of the quantity of individuals and the dimension x of the simulated space. A simulation of a reduced number of individuals would not give so different result between the original algorithm and the new one. In the same way, a space with a small value to the X dimension would not have the same reduction in the time of simulation.

In future works we intend to limit the selection of neighbor candidates in relation to the dimensions Y and Z.

Figure 12 shows the speedup of the three versions of the simulator calculated as:

$$\text{Speedup} = \frac{\text{Sequential Time}}{\text{Parallel Time}}$$

To calculate the speedup we used as sequential time the time of the third version of the simulator (i.e. using the new communication strategy and avoiding the execution of unnecessary tasks) executed by only one processor.

The graphic shows that the speedup of the system is not equal to the linear speedup, but it was improved by each new version.

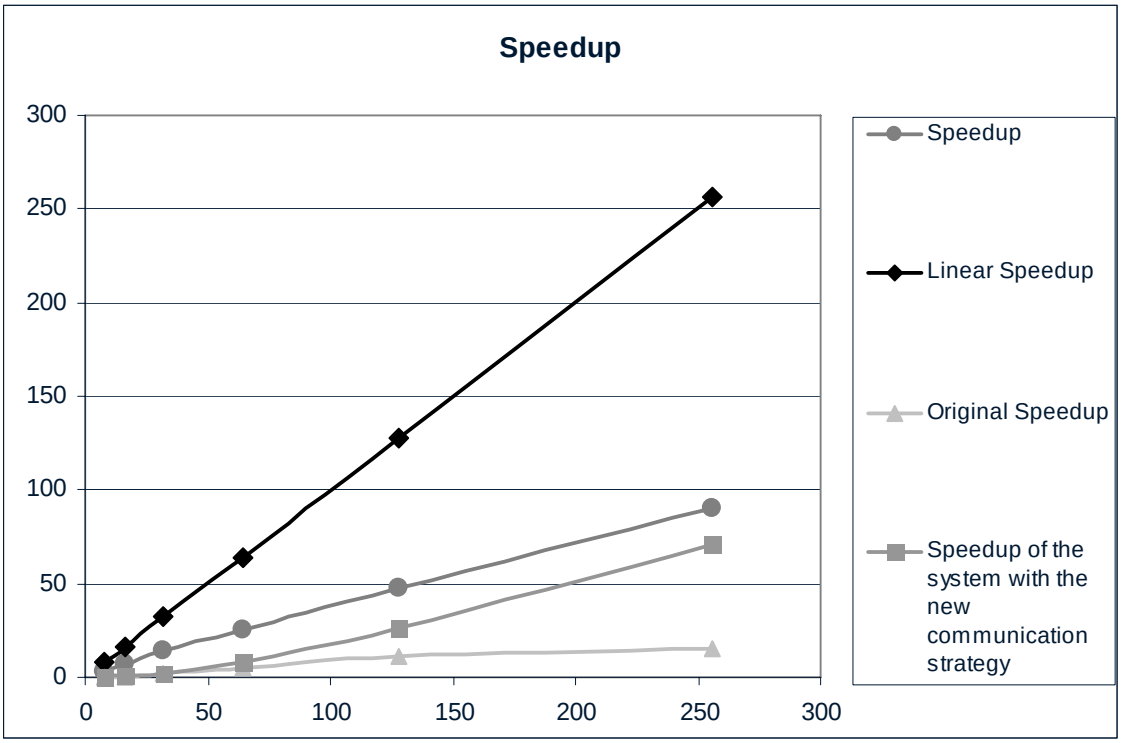

Figure 12: Comparison of the speedup of the different versions of the simulator and the linear speedup.

In the figure 12 we call original speedup the speedup of the original simulator. Applying the new communication strategy, we obtained a better speedup curve. Finally, it is identified as only speedup the curve obtained with the third version of the simulator, the one that avoid the unnecessary work.

The graphic shows that the objectives of this work was achieved, once that the new communication strategy and the new way of neighbors candidates selection helped to reduce the simulation time and so, improved the speedup

The use of the new communication strategy increased the scalability of the system. In the first version, it was not recommendable to use more than 64 processors because the time of communication to more than that number of processors was greater than the computation time. The results founded with the use of the new communication strategy showed that this problem does not occur any more, because the quantity of messages exchanged is better controlled.

Chapter 5

CONCLUSIONS AND FUTURE WORK

In this work we did some experimentations with a distributed fish school simulator (Mostaccio, Dalforno et al. 2006; Mostaccio 2007) with the aim to found a way to improve its speedup and scalability.

The simulator was based in a two-dimensional IoM model that defines mathematically the possible reactions of a fish. Although the original model were two-dimensional, some adaptations were done in the way to represent an individual, the space and the reactions to obtain a three-dimensional model. Thus, the original distributed simulator used in this work was a three-dimensional one.

Experimentations were done with the original simulator to found possible ways to improve it. The experiments consisted in to simulate a population of 640.000 individuals in a space with dimensions x = 25600, y = 500 and z = 500.

The results founded with the experiments using the original simulator signalized two points that harms the scalability and the speedup of the system:

- The communication strategy that suggests the exchange of an undesirable number of messages in some situations.
- The execution of unnecessary operations.

First it was revised the original communication strategy to identify the causes of the generation of the great quantity of messages. As a result, a new communication strategy was proposed with the aim to limit the quantity of messages without loosing the result's accuracy.

The experiments done, using the new strategy, showed that the communication time reduced. This was explained by the fact that with this new strategy the quantity of messages exchanged by one LP with its neighbors is in maximum 8. Using the original strategy, the number of messages increases in function of the quantity of fish in the

border of the LP. Thus it was expected a maximum of $\left(4+\left(8\cdot\frac{(NR_3)}{X}\right)\right)$ messages exchanged by an LP, where N is the quantity of fish been simulated, R_3 is the third radius and X is the dimension of the simulated space in relation to the x-axis.

The implementation of this strategy changed the algorithm in a way that the synchronization is now implicitly done.

When the data computation stage of one-step of simulation starts, it means that all data needed to realize it can be found locally and, so, there will be no fish waiting to be processed later.

This new strategy avoids the exchange of the same data repeatedly and its use offers a facility to achieve a better parallelization of the application doing in the same time computation and communication.

The second improvement achieved by this work to the application was the reduction of the time spent with the data computation stage.

This was done avoiding the execution of unnecessary work. The great reduction in the complexity of the algorithm pointed to a probably execution of tasks that was not really needed. Analyzing the application, we found that the subdivision of the space, done to parallelize the application, was the key to avoid these tasks.

Based on that, it was developed a new neighbors candidates algorithm that only verify the fish that are in a distance lower than R_3 (in relation to x-axis) from the fish been calculated.

The founded results showed a great reduction in the data processing time. As fewer processors are used, as bigger is the reduction of time. This is explained by the subdivision of the space that just was done by the parallelization.

Thus, the parallelization of the application suggested improvement to the sequential version in this case.

Other advantage is that we can obtain better results using fewer resources. Now, with 8 processors we achieved a better time than we achieved with 32 processors.

5.1 Future Works

In future works we intend to improve the parallelism of this distributed fish school simulator doing at the same time computation and communication. This is possible because only a part of the fish simulated by an LP need data from fish outside its own slice.

Specifically the fish that need this extra data are the ones that are in a distance lower than R_3 from the LP's right and left borders, i.e. $\left(2\left(\frac{NR_3}{X}\right)\right)$ in maximum. All other fish in the LP $\left(N-\left(2\left(\frac{NR_3}{X}\right)\right)\right)$ can be processed before the arrival of the data of the fish in the neighbors LPs.

Another improvement should be to restrict the choose of neighbors candidates in relation to the y-axis and the z-axis reducing a little more the time spent with this task.

The presented simulator was based in the Huth and Wissel's s model (Huth and Wissel 1992) that defined the fish school behavior based in the fish relationship. A future aim of this project is to study others models that defines the behavior of the fish school in presence of a predator, obstacles and other fish species.

The improvement of the model's complexity can demand the control of the load balance of the system. The presence of a predator can for example subdivide the group and direct the best part of it to specifics LPs.

An idea that can be useful to maintain the load balance of the system is to map the space in the used LPs. It means to create a mechanism to simulate in an LP more then one part of the simulated space. Instead of to help to maintain the load balance this mechanism will possibility the simulation of a bigger space, or a space that can grow in the moment of the simulation without changing the dimensions of each slice.

References

Andrews, P. L. (1986). BEHAVE: Fire Behavior prediction and modeling systems - Burn subsystem, part 1, Odgen, UT, US Department of Agriculture, Forest Service, Intermountain Research Station.

Andrews, P. L., C. D. Bevins, et al. (2005). BehavePlus fire modeling system, version 3.0: User's Guide, Ogden, UT:

Department of Agriculture, Forest Service, Rocky Mountain Research Station.

Banks, J., J. S. Carson, et al. (1995). Discrete-event system simulation. New Jersey, Prentice Hall.

Bertacchini, M. and A. Benabén (2005). Reducing MPI communication latency with GAMMA. Encuentro Científico Internacional (ECI). Lima.

Bianchini, G., A. Cortes, et al. (2006). Improved prediction methods for Wildfires using High Performance Computing: A comparison. ICCS 2006 - International Conference on Computational Science University of Reading, UK.

Cores, F. (1999). Switch Time Warp: Un método para el control del optimismo en el protocolo de simulacion distribuida Time Warp. Departamento de Arquitectura de Computadores y Sistemas Operativos (DACSO), Universidad Autónoma de Barcelona (UAB).

Ferscha, A. F. and S. K. Tripathi (1994). Parallel and Distributed Simulation of Discrete Event Systems.

Finney, M. A. (1998). FARSITE: Fire Area Simulator-model development and evaluation, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station.

Fujimoto, R., M. (1999). Parallel and Distribution Simulation Systems, John Wiley \& Sons, Inc.

Grimm, V. and S. F. Railsback (1958). Individual-based Modeling and Ecology. New Jersey, Princeton University Press.

Habata, S., M. Yokokawa, et al. (2003). "The earth simulator system." NEC Res. & Develop. **44**(1).

Hanzich, M., F. Giné, et al. (2006). Using On-The-Fly Simulation For Estimating the Turnaround Time on Non-Dedicated Clusters. EuroPar 2006.

Hanzich, M., J. Lerida, et al. (2006). Using Simulation, Historical and Hybrid Estimation for Enhancing Job Scheduling on NOWs. IEEE International Conference on Cluster Computing.

Huston, M., D. DeAngelis, et al. (1988). "New Computer Models Unify Ecological Theory " BioScience **38**(10): 682-691.

Huth, A. and C. Wissel (1992). "The simulation of the movement of fish schools." Journal of Theoretical Biology **156**(3): 365-385.

Lorek, H. and M. Sonnenschein (1995). Using Parallel Computers To Simulate Individual-Oriented Models In Ecology: A Case Study. 1995 European Simulation Multiconference (ESM).

Misra, J. (1986). "Distributed discrete-event simulation." ACM Computing Surveys (CSUR) **18**: 39-65.

Montaigne, F., K. Warn, et al. (2007). Crirsis Mundial de la Pesca. National Geographic - España. **20:** 3-69.

Mostaccio, D. (2007). Simulación de Altas Prestaciones para Modelos Orientados al Individuo. Computer Architecture & Operating Systems Department. Barcelona, Universitat Autònoma de Barcelona.

Mostaccio, D., C. Dalforno, et al. (2006). "Distributed Simulation of Large-Scale Individual Oriented Models." Journal of Computer Science & Technology **6**(2): 59-65.

Mostaccio, D., R. Suppi, et al. (2004). Simulación Distribuida de Modelo Orientados al Individuo utilizando MPI. CACIC 2004 - X Congreso Argentino de Ciencia de la Computación. La Matanza - Argentina.

Mostaccio, D., R. Suppi, et al. (2005). Distributed Events Simulation for Individual Oriented Models. EMSS 2005 - European Modeling Simulation Symposium in IM3 - International Mediterranean Multimodeling Multiconference, Marsella - Francia.

Mostaccio, D., R. Suppi, et al. (2005). Distributed Simulation of Ecologic Systems. XVI Jornadas de Paralelismo, Granada - España, Thomson.

Mostaccio, D., R. Suppi, et al. (2005). Simulation of Ecologic Systems Using MPI. EuroPVM/MPI 2005, Sorrento - Italia.

Mostaccio, D., R. Suppi, et al. (2006). "Distributed Simulation of Large-Scale Individual Oriented Models." Journal of Computer Science & Technology **6**(2): 59-65.

Parrish, J. K., S. V. Viscido, et al. (2002). "Self-Organized Fish Schools: An Examination of Emergent Properties." Bio. Bull. **202**: 296-305.

Sato, T., S. Kitawaki, et al. (2002). Earth Simulator Running. ISC2002 - International Supercomputing Conference. Heidelberg.

Suppi, R., D. Fernández, et al. (2004). Fish Schools: PDES Simulation and Real Time 3D Animation. Fifth International Conference on Parallel Processing and Applied Mathematics, Berlin / Heidelberg.

Suppi, R., P. Munt, et al. (2002). Using PDES to simulate Individual-Oriented Models in Ecology: A case study. ICCS 2002 - Proceedings of the international Conference on Computational Science-Part I, Amsterdam, The Netherlands.

Yang, X. Y., P. Hernández, et al. (2006). Providing VCR in a Distributed Client Collaborative Multicast Video Delivery Scheme. Euro-Par 2006, Dresden, Germany.

Yang, X. Y., P. Hernández, et al. (2006). DVoDP2P: Distributed p2p assisted multicast vod architecture. IEEE International Parallel & Distributed Processing Symposium (IPDPS'06). Rhodes Island, Greece.