Performance Improvements of Real-Time Crowd Simulations

Guillermo Vigueras
PhD Student. 4 years in PhD program
University of Valencia. Spain
guillermo.vigueras@uv.es

Juan M. Orduña, Miguel Lozano
PhD advisors. Departamento de Informática
University of Valencia. Spain
{juan.orduna,miguel.lozano}@uv.es

Abstract

The current challenge for crowd simulations is the design and development of a scalable system that is capable of simulating the individual behavior of millions of complex agents populating large scale virtual worlds with a good frame rate. In order to overcome this challenge, this thesis proposes different improvements for crowd simulations. Concretely, we propose a distributed software architecture that can take advantage of the existing distributed and multi-core architectures. In turn, the use of these distributed architectures requires partitioning strategies and workload balancing techniques for distributed crowd simulations. Also, these architectures allow the use of GPUs not only for rendering images but also for computing purposes. Finally, the design and implementation of distributed visual clients is another research topic that can help to overcome this challenge.

1. Description of the Problem

Crowd simulation has become an essential tool for many virtual environment applications, and the extensive use of high quality virtual crowds is crucial for many virtual environment applications in education, training, and entertainment [1]. Crowd simulation can be considered as a special case of Virtual Environments where the avatars are intelligent agents instead of user-driven entities. Each of these agent-based entities can have its own goals, knowledge and behavior [2].

On the one hand, crowd simulations must focus on rendering visually plausible images of the environment, requiring a high computational cost. On the other hand, complex agents must have autonomous behaviors, greatly increasing the computational cost as well. The sum of these requirements results in a computational cost that exponentially increases with the numbers of agents in the system, requiring a scalable design that can support huge amounts of agents (of different orders of magnitude) by simply adding more hardware. Thus, some proposals tackle crowd simulations as a particle system with different levels of details (eg:*impostors*) in order to reduce the computational cost [3] due to the graphical quality. Also, several proposals have been made to provide efficient and autonomous behaviors to crowd simulations [4]. The current challenge for these applications is the design and development of a scalable system that is capable of simulating the individual behavior of millions of complex agents populating large scale virtual worlds with a good frame rate.

2. The proposed approach and methodology

In order to overcome the current challenge, different improvements should be made. First, we should design a scalable system architecture that takes into account the underlying computer system that is being used for crowd simulation. This system architecture is based on a networked-server Distributed Virtual Environment (DVE) [5]. On top of this distributed computer architecture, we propose a distributed software architecture that can take advantage of the existing distributed and multi-core architectures [6], [7]. In turn, the use of distributed architectures opens other research topics, like partitioning strategies and workload balancing techniques for distributed crowd simulations. Also, the use of a distributed architecture allows the use of GPUs for computing, and not only for rendering images of the virtual world, improving the performance of crowd simulations. Finally, the design and implementation of distributed visual clients is another research topic that can help to overcome the current challenge for crowd simulations.

The research methodology consists of performing crowd simulations on real distributed systems implementing the proposed designs and techniques. The idea is to measure the performance of the systems with different number of agents. Since we are using distributed systems, the most important performance

measurements are latency and throughput [8]. Concretely, we have measured the response time provided to agents by the distributed servers controlling the virtual world when they request their actions. In this way, we can study the maximum number of agents that the system can support while providing a response time below a given threshold value. In order to define an acceptable response time, we have considered 250 ms. as the threshold value, since it is considered as the limit for providing realistic effects to users in DVEs [9].

3. Significance of the research

The proposed research will allow to use crowd simulations in a new dimension of large-scale applications and challenges. On the one hand, the realistic simulation of large-scale catastrophic events like natural disasters or terrorist attacks can help to both the design of emergency protocols and the training of emergency personnel. On the other hand, large-scale crowd simulations can be used to study social behaviors and social engineering techniques.

4. Related Work

The motion of crowds and other flock-like groups has been modeled as interacting particles that display different behaviors in 2D/3D scenes [10], [11]. However, when the number of agents or particles grows so does the workload generated by the crowd, making necessary the distribution of the crowd among different computers in order to keep an acceptable degree of interactivity. Typically, there are two different approaches for distributing a crowd simulation. One of them is based on the criterion of workload [12], so that different groups of agents are executed in different computers. The other approach is region-based [13], in such a way that the virtual world is split into regions (usually a 2D cell from a grid) and all the agents located at a given region are assigned to a given computer. Despite these approaches can manage up to tens of thousands of agents, the scalability of the application is limited by the synchronization scheme of agents.

Several researchers have already studied the capabilities of multi-core architectures for crowd simulations. One approach has been presented for PLAYSTATION3 which supports simulation and display of simple crowds up to 15000 individuals at 60 frames per second [14]. Another work proposes a highly parallel implementation for multi-core processors of and algorithm for crowd animation that the

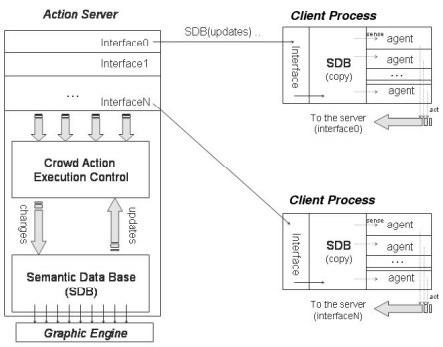

Figure 1. Software System architecture proposed for crowd simulation.

authors used for pedestrian simulation[15]. Despite the high number of agents supported by these approaches, the scalability of the system for large scale simulation is limited, since it is not designed to be distributed across different machines.

Also, there are other approaches that use graphics processor units (GPUs). One of these proposals simulates thousands of individuals using models designed for gaseous phenomena [16]. Recently, some authors have started to use GPU in an animation context (particle engine) [17], and there are also some proposals for running simple stochastic agent simulations on GPUs [18]. However, these proposals are far from displaying complex behaviors at interactive rates.

5. Results Obtained

We have proposed a distributed system architecture for crowd simulation [6]. In this scheme, a distributed computer architecture is used to implement the client-server software architecture shown in Figure 1. This software architecture is mainly composed by two elements: the action server (AS) and the client processes (CP). The AS is devoted to execute the crowd actions, while a CP handles a subset of the existing agents. Agents are implemented as threads of a single process for reducing the communication cost. Each thread manages the perception of the environment and the reasoning about the next action. Since reasoning formalisms can involve a high computational cost, each client process is hosted on a different computer, in such a way that the system can have a different number of client processes, depending on the number of agents in the system. This scheme allows to properly simulate up to tens of thousands of autonomous agents at interactive rates.

Despite of the flexibility provided by this system architecture [6], the AS represents the system bottle-

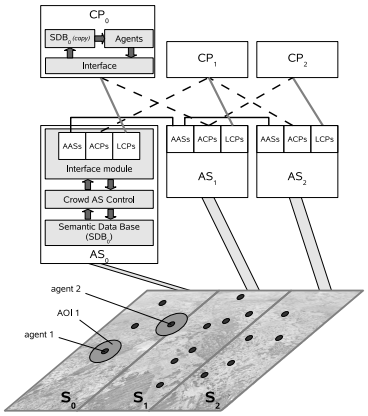

Figure 2. Scheme of the proposed distributed architecture with the Parallel Action Server.

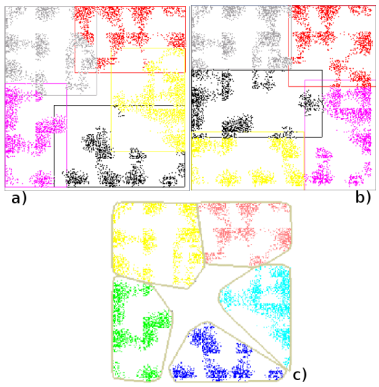

Figure 3. Snapshots of the partitions provided by a) R-Tree b) GA c) QHull methods

neck. For that reason, the previous Action Server has been divided into a set of processes so that each one can be executed in parallel in a different computer [7]. Each of these processes is denoted as an Action Server (AS) while the whole set of processes has been denoted as the Parallel Action Server (PAS).

In order to take advantage from the underlying system architecture, the distribution is performed at two levels. First, the virtual world is partitioned into a 2D grid, and each region of the grid is assigned to an AS process before the simulation starts. Figure 2 shows an example of the proposed architecture, and how this partitioning is performed. In this figure, the whole space is partitioned into three subregions, and each one is assigned to one AS. Once a region is assigned to a given AS, that server is responsible for checking the actions (eg. collision detection) of the agents located at that region. Once the partition has been initialized, the crowd must be also partitioned and distributed among the CPs associated to the corresponding servers. Each AS process hosts a copy of the Semantic Database. However, each AS exclusively manages the portion of the database representing the agents in its region.

Unfortunately, several problems arise when physically distributing the database. First, in order to maintain the consistency those agents near the borders of each region need to check their actions with the corresponding servers. This requires the exchanging of locking requests among the computers hosting the partition of the database. This constraint adds a significant overhead, and therefore it must be minimized. Additionally, the partition must be properly balanced, in order to avoid the saturation of the distributed system. Otherwise, one or more computers can reach saturation, greatly degrading the performance of the entire system.

We have studied different methods to solve the partitioning problem in distributed crowd simulations and we have proposed a new method that efficiently solves the problem [19]. The partitioning problem consists of finding a near optimal partition of regions (containing all the agents in the system) that minimizes the number of agents near the borders of the regions, and also that properly balances the number of agents in each region. To solve this problem we have studied three different methods. One of them based in the R-Tree data structure, other method implements a Genetic Algorithm (GA) to solve the partitioning problem and the last method uses the QuickHull algorithm to perform the partitioning based on the convex hull of each region managed by each server.

As an example, Figure 3 shows a snapshot of the different partitions provided by the considered methods during a simulation. Figure 3 a) and b) show the partitions provided by the R-Tree and the GA methods, respectively. It can be seen that both partitions use rectangular regions, although the overlapping among the regions provided by the GA method is lower than the overlapping provided by the R-Tree method. Figure 3 c) shows the partition provided by the Convex Hull method, and it can be seen that there is no significant overlapping among the regions of this partition, thus significantly improving the results provided by the GA and R-Tree methods, since the overhead resulting from the inter-server communication is reduced.

6. Remaining objectives

The advent of the multi-core era has allowed a huge increase in the computing bandwidth of current processors. However, the distributed architecture designed for crowd simulation must be adapted in

order to take advantage of the on-chip parallel computing power. New synchronization methods among the execution threads in the distributed server are being integrated and tested. Since these methods reduce the synchronization overhead, initial results show that the server throughput can be improved obtaining a good scalability with the number of processor cores.

On other hand, the huge number of cores existing in current Graphics Processor Units (GPUs) provides these devices with computing capabilities that can be exploited by crowd simulations. We have implemented a distributed server for crowd simulations using an on-board GPU [20]. Since the consistency maintained by the distributed server is the critical path in the system, we have implemented on the GPU this part of the server. Preliminary results show that the server throughput can be greatly increased by using GPUs.

Also, we are using GPUs for graphics rendering. Concretely, we are developing a distributed Visual Client Process to efficiently visualize crowd simulations in 3D, allowing different user configurations like ground walk-throughs, top view of the scene, etc.

References

[1] P. A. Kruszewski, "A game-based cots system for simulating intelligent 3d agents," in *BRIMS '05: Proceedings of the 2005 Behavior Representation in Modelling and Simulation Conference*, 2005.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1987, pp. 25–34.

[3] S. Dobbyn, J. Hamill, K. O'Conor, and C. O'Sullivan, "Geopostors: a real-time geometry/impostor crowd rendering system," *ACM Trans. Graph.*, vol. 24, no. 3, pp. 933–933, 2005.

[4] A. Iglesias and F. Luengo, "New goal selection scheme for behavioral animation of intelligent virtual agents," *IEICE Transactions on Information and Systems, Special Issue on 'CyberWorlds'*, vol. E88-D, no. 5, pp. 865–871, 2005.

[5] S. Singhal and M. Zyda, *Networked Virtual Environments*. ACM Press, 1999.

[6] M. Lozano, P. Morillo, J. M. Orduña, V. Cavero, and G. Vigueras, "A new system architecture for crowd simulation," *J. Netw. Comput. Appl.*, vol. 32, no. 2, pp. 474–482, 2009.

[7] G. Vigueras, M. Lozano, C. Perez, and J. Orduña, "A scalable architecture for crowd simulation: Implementing a parallel action server," in *Proc. of 37th Int. Conference on Parallel Processing (ICPP-08)*, Sept. 2008, pp. 430–437.

[8] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. IEEE Computer Society Press, 1997.

[9] T. Henderson and S. Bhatti, "Networked games: a qos-sensitive application for qos-insensitive users?" in *Proceedings of the ACM SIGCOMM 2003*. ACM Press / ACM SIGCOMM, 2003, pp. 141–147.

[10] K. Sims, "Particle animation and rendering using data parallel computation," in *SIGGRAPH '90: Proc. of 17th conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1990, pp. 405–413.

[11] T. Frank, K. Bernert, and K. Pachler, "Dynamic load balancing for lagrangian particle tracking algorithms on mimd cluster computers," in *PARCO'2001 - International Conference on Parallel Computing 2001.*, 2001.

[12] A. Steed and R. Abou-Haidar, "Partitioning crowded virtual environments," in *VRST '03: Proceedings of the ACM symposium on Virtual reality software and technology*. New York, NY, USA: ACM, 2003, pp. 7–14.

[13] M. J. Quinn, R. A. Metoyer, and K. Hunter-Zaworski, "Parallel implementation of the social forces model," in *In Proceedings of the Second International Conference in Pedestrian and Evacuation Dynamics*, 2003, pp. 63–74.

[14] C. Reynolds, "Big fast crowds on ps3," in *Proceedings of the ACM SIGGRAPH symposium on Videogames*. New York, NY, USA: ACM, 2006, pp. 113–121.

[15] S. J. Guy, J. Chhugani, C. Kim, N. Satish, M. Lin, D. Manocha, and P. Dubey, "Clearpath: highly parallel collision avoidance for multi-agent simulation," in *SCA '09: 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 2009, pp. 177–187.

[16] N. Courty and S. R. Musse, "Simulation of large crowds in emergency situations including gaseous phenomena," in *CGI '05: Proceedings of the Computer Graphics International 2005*. IEEE Computer Society, 2005, pp. 206–212.

[17] K. Peter, S. Mark, and W. Rudiger, "Uberflow: a gpu-based particle engine," in *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*. ACM, 2004.

[18] M. Lysenko and R. M. D'Souza, "A framework for megascale agent based model simulations on graphics processing units," *Journal of Artificial Societies and Social Simulation*, vol. 11, no. 4, p. 10, 2008.

[19] G. Vigueras, M. Lozano, J. O. na, and F. Grimaldo, "A comparative study of partitioning methods for crowd simulations," *Journal of Applied Soft Computing*, vol. 10, no. 1, pp. 225 – 235, 2010.

[20] G. Vigueras, J. Orduña, and M. Lozano, "A gpu-based multi-agent system for real-time simulations," in *Proceedings of the 8th International Conference on Practical Applications of Agents and Multi-Agent Systems (PAAMS 2010)*, April 2010.