Procedia Computer Science

Volume 51, 2015, Pages 473–482

ICCS 2015 International Conference On Computational Science

Crowd evacuations SaaS: an ABM approach

Albert Gutierrez-Milla, Francisco Borges, Remo Suppi and Emilio Luque

Universitat Autònoma de Barcelona, Bellaterra, Barcelona, Spain
albert.gutierrez@caos.uab.cat, francisco.borges@caos.uab.cat, remo.suppi@uab.cat, emilio.luque@uab.cat

Abstract

Crowd evacuations involve thousands of persons in closed spaces. Having knowledge about where the problematic exits will be or where the disaster may occur can be crucial in emergency planning. We implemented a simulator using Agent Based Modelling able to model the behaviour of people in evacuation situations and a workflow able to run it in the cloud. The input is just a PNG image and the output are statistical results of the simulation executed on the cloud. This allows to provide the user with a system abstraction and only a map of the scenario is needed. Many events are held in main city squares, so to test our system we chose Siena and we fit about 28,000 individuals in the centre of the square. The software has special computational requirements because the results need to be statistically reliable. Because these needs we use distributed computing. In this paper we show how the simulator scales efficiently on the cloud.

Keywords: Agent Based Modeling, crowd evacuations, cloud computing, SaaS, distributed simulation.

1 Introduction

Crowd events that involve thousands of persons are becoming increasingly common. The risks that they involve are an important problem to solve. Cases as the Hillsborough Stadium disaster (Sheffield, England in 1989) causing 96 deaths, the Love Parade disaster (Duisburg, Germany in 2010) causing 21 deaths or the Kiss disaster (Santa Maria, Brazil in 2013) causing 239 deaths show us how bad decisions and planning can lead to human harm. Evacuation management and planning then become a crucial issue.

Exercises such as fire drills allow us to know the evacuation times, but not the conditions. People have stochastic behaviour that determines the variability of evacuation. It is not possible to perform as many drills as randomness. In these cases simulations becomes necessary. We will be able to know what the evacuation is like according to the simulator and with this information take decisions in evacuation cases. That way we will be able to have knowledge to carry out what is thought appropriate in evacuation planning the safest and fastest evacuation possible.

Simulators allow us to have knowledge to take informed decisions that can predict the behaviour. Decision Support Systems (DSS) have gained importance in the last years, and

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.271

the usage of simulators allows to predict the behaviour of a system according to a model and help experts to take decisions. The distribution and usage of complex software is an important problem to solve. Cloud computing will allow us to make this complexity and the infrastructure transparent to the user and provide the infrastructure on demand. In crowd evacuations there are random patterns among the public. This drives us to ask the question: is one simulation enough to predict what the behaviour of the public will be? We need statistically reliable results and one simulation allow us to see some tendencies in the evacuation, but the evacuation time can vary widely and not predict with reliability. In order to have a set of data that will allow us to have enough results, we will use a statistical method to calculate the number or runs based on the statistical reliability of the results. This approach will increase our computational needs.

In order to meet the statistical reliability computational needs we developed a distributed simulator in C using the message passing library MPI. We present a cloud tool able to execute thousands of runs in crowd events where the population can be hundreds or thousands of individuals. There are desktop tools such as Oasys MassMotion [1] or STEPS[2], but at the time of this work we were not able to find any software that implements the crowd evacuation problem in the cloud. We developed a workflow able to run the simulation on the cloud. The input of the system is the map image. With this image the application will be executed providing SaaS. The output of the system will be a reports with the evacuation information and an interactive 3D visualization of the evacuation. This systems hides the complexity of system providing a simple interface to the user.

2 Related work

By observing natural or artificial events we can extract the main rules that determine the behaviour in order to define a model. Models allow us to interact and test the system without using the real system. This will allow us the most important function of simulation: prediction. In the case of crowd evacuations, there are several approaches and models to solve it. We will comment on two important approaches: cellular automaton and social force, and finally, the approach that we used: Agent Based Modeling.

2.1 Cellular Automaton & Social force

Cellular automaton (CA) are models that evolve in discrete time phases and they interact mutually. Every CA interacts locally with its neighbors and the positions of the individuals are reloaded simultaneously. To calculate the state in time T+1, we will take into account the state in time T and its neighborhood. Besides this, several models based on cellular automaton differ on this description. It has been used in crowd evacuations widely[3][4].

Social force is a continuous model presented by Helbing and Molnar [5]. It is an analytical model based on the calculus of a set of continuous time equations. Every person has an objective, which is the point where they are moving to. The variation in the individual's velocity depends on the vectorial force that is interpreted as the social force. The force is not impacting in the movement of the bodies, but in the individual motivation of the agent. Affected by repulsion, attraction or desire affects this movement.

2.2 Agent based modeling

Agent based modeling (ABM) is an approach to modeling systems comprised of individual, autonomous, interacting "agents". It is based on the representation of global behaviour from

the rules given to individuals and it enables us to see the consequences on a macro level of the interactions on a micro level. There will be a set of individuals with their behaviour and attributes and the environment. We can have a vision of how the individuals interact (homogeneous or heterogeneous agents) with the environment and the other individuals, in discrete times. Every agent is ruled by the same laws. ABM allows us to analyze how the system evolves from the rules of the individuals and their interaction with others and the environment. From the beginning, the output of the system is unknown, even with a model and a known configuration. We cannot know the output of the system analytically, and because of that we will need to simulate it. In the case of crowd evacuations it has been ussed proviously[6].

3 Crowd evacuation model

In the case of evacuation of individuals we are interested in what their state will be, in the behaviour of the individual and the evolution of their attributes as times goes by. Great importance will be given to how we model the environment with who will constantly interact with the set of individuals.

3.1 Model description

Our model is based on the model description of Helbing and Molnar[5]. It presents phenomena such as bottlenecks, clogging or arching close to the exits. Mass behaviour emerges from a common objective and is not an explicit functionality. Nevertheless the velocity of all the individuals will be a variable attribute. This means that the crowd will be slower or faster depending on the speed modeled. We describe our Crowd Model (CM) as follows:

$$CM = \langle T, C, Q, N, \beta, \delta \rangle$$

Where T is the time, divided in discretized steps; C is the global space; Q is the set of states; N is the neighbourhood space; β is the information related to the grid space N We have the Crowd Model (CM) and δ are the functions of state transition. There are three states defined: Q_1 (stopped), Q_2 (moving) and Q_3 (evacuated).

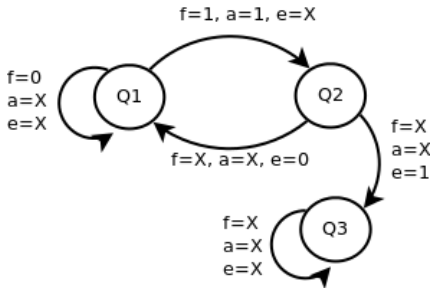

Figure 1: Model's state diagram.

The transition between states is determined by three variables. The objective function of every agent is to achieve the exit. Because of this we have the following variables that define the movement of the agent: a (approaching the exit), e (exit reached) and f (free patch). In the Fig. 1 we can see the flow diagram and the states of the individual. This logic is applied in every simulation step in the time-driven simulation. All the states are internals of every

individual. As seen previously, the rules given to the ABM are what determine the behaviour of the group. The goal of the individual is the exit and the minimum distance determines, by default, where the person is going. This step is executed befoore simulation starts running. The scenario is the main input of the simulator. It is dependent on the chosen individual model. In our case we have a grid dividing the space that will represent our scenario in a bi-dimensional space. We will name every position of the grid as "patch". Patches occupy a space of 40x40 cm, which is approximately the mean space occupied by a person. The individual analyses the 8 neighbors in random order, to avoid the individuals always moving in the same direction. It tests if the patch has a "free space" type and if there is an individual inside it. In the case that it is free and we can approach the exit, we will move to it and we will update our values. In the case that we cannot approach the exit, then the individual will wait for the people that are between them and the exit to move. Every individual fits a space of the grid, so we will assume the space necessary to hold one person is 40x40 cm. Because the rules that determine the behaviour of the individuals is the same, and also their sizes, the velocity will be the same for all of them. Every individual checks their 8 adjacent neighbors. In the space we will find different kinds of patches. They are divided as follows: walls and obstacles: they are the boundaries and the patches that cannot be crossed. Even though they are conceptually different in our model, they have the same behaviour; free space: available patch where the persons can move. It can be occupied if there is no individual there; exit: objective function of the people. The exit is achieved getting the shortest path.

3.1.1 Simulation algorithm

Every simulation step determines the movement of the individuals and the transition of simulation time $\Delta \rightarrow \Delta + 1$. Because the space is divided in a grid, we have matrices that allow us to directly access the individual position. Locally the individuals will move to another path that fills their objective function: approaching the exit. In this case we will only consider single-story spaces.

Algorithm 1 Simulation algorithm

```
1: procedure SimulationStep
2:     readMap()
3:     readExits()
4:     generatePotentialField()
5:     e ← targetExit()
6:     for all p ∈ People do
7:         for all n ∈ neighbours(p) do
8:             if n = EXIT then
9:                 releaseAgent(p)
10:            if n = FREE ∧ range(n, e) < range(p, e) then
11:                p ← n
```

3.1.2 Path finding

To get to the exit, we used a minimum distance approach with obstacle avoidance. We implemented an algorithm to know what the minimum distance is. Potential field allows us to provide individuals with a way to be informed if they are approaching the exit. There are as

many flood fills as exits. The data structure is a list of matrices where every matrix is the flood fill values. There are several potential field algorithms and we chose a mix of the Manhattan and Chessboard potential field calculation[7]. Every individual will have knowledge of the potential field of their exit and will use it to take decisions between steps.

3.2 Statistical analysis of the model

We can consider simulation as terminating and nonterminating. It depends on concepts such as convergence, and if there is a final point of the simulation. In our case we have a terminating simulation where the end of it is when all the public has evacuated the simulated space.

In the evacuation there are two main phases. In the first phase, the public are distributed randomly around the space and there is no clogging. When the evacuation starts, they move to the exits. The second phase is the congestion one. The public have moved to the exits and now clogging, arches, and other phenomena emerge as a problem. We need to distinguish these two phases when calculating the number of runs that we need. To calculate the number of runs necessary to have statistically significant results, we used the Diaz-Emparanza[8] model. He proposes a formula where values of probability interval and accuracy are parametrized to get analytically the minimum number of runs necessary.

$$T = \frac{t^2_{\frac{\alpha}{2}} p_H(1-p_H)}{A^2} \tag{1}$$

Where T is the number of simulations, in our case we use the following values with which we get the minimum number of simulations. Our probability interval is [0.045 0.055]. We selected the following values:probability interval (p_H): 0.05; and accuracy (A): 0.002; level of confidence $(1-\alpha)$ where we get Student's t-distribution value.

$$T \leq \frac{2.3^2 0.05(1-0.05)}{0.002^2} = 62,818.75$$

This value will be the minimum number of runs that we need to meet the previous criteria.

3.3 Time calculation

The mean walking speed is 1,4m/s[9], but in evacuation cases the crowd speed depends on the density. We used the velocity/density relation test. The total time is calculated among the time transitions. The simulator calculates in execution time the density of the crowd to determine the public speed. Density is checked around neighbours. We check how neighbours can have, being 8 the maximum. According to this we divide and get the density. We get the mean number of neighbours and then divide by 8 to get the percentage. Because every agent needs a space of 40x40 cm in our model the maximum density is $6,25 persons/m^2$. We selected the linear model being 1.4 m/s the maximum and 0,2 m/s the minimum speed, based on previous studies[10]. The value of a simulation steps varies over the time, where et is evacuation time.

$$v_{crowd} = freePatches * 1,2 + 0,2 \tag{2}$$

$$\Delta t = \frac{d_{patch}}{v_{crowd}} \tag{3}$$

Where *freePatches* is the mean of the percentage of free patches surrounding the agents, and Δt is the elapsed time in the current simulation time.

4 Architecture overview

The model described previously has computational requirements that make it depend on an HPC architecture. To hide all this complexity we will move the simulator to the cloud, and particularly to the Amazon infrastructure. Under the need of having an MPI cluster on demand we used the project named StarCluster. In order to execute it on the cloud and make it SaaS, we create a system execution flow from a web interface until the output of the system. The cloud resources intercommunication is hided by the Star Cloud. It stablishes SSH connections between nodes and MPI uses it for the message passing.

4.1 HPC tool

The first decision for moving an application to the cloud has to be which platform will be used. We chose Amazon because it allows us to execute MPI jobs, and the pricing is one of the cheapest on the market. To execute the job, we needed to configure the cluster. To do this task we selected StartCluster, a free software project from MIT[12]. It configures the cluster and installs necessary software as an NFS parallel file system, OpenMPI libraries or SGE.

4.2 System workflow

In order to execute the simulator using a web interface from the cloud, we created a whole execution flow to run the simulator on the cloud. The web page is responsible for collecting the input data and showing the output. The necessary input for the system will be: bi-dimensional map and public (optional). We created a web page that uploads the map and the rest are introduced using a form. Once all this information considered mandatory to run the model is set, then we assert all the input is correct. As input of the simulator, a lossless image compression format should be used. These image can be created using satellite images or city/building maps . The image needs to be saved in a lossless format. In our case we chose PNG. every pixel represent information about the bidimensional space and the scale of every pixel is 40x40 cm^2. The information is: black (obstacles), white (free space), red (exit center), green (exit) and blue (initialized individual).

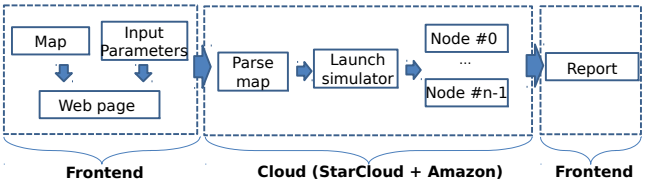

Figure 2: Execution workflow.

StarCluster is automatically configured and the number of nodes are transparent for the user, so they must be configured beforehand. Once it has been checked, a Python script launches the OpenMPI job along all the configured nodes.

To distribute the simulations among all the nodes, a static scheduler is used. All the nodes get a fixed number of simulations. The nodes save the information locally in a NFS file system. This minimizes the communication among nodes and makes an independent approach where there is no dependency.

Once the simulations have finished, the user will have feedback with a report that will be sent to its personal e-mail. There will be a report showing information such as: mean, minimum and maximum evacuation time, people evacuated per gate, and a 3D visualization of the evacuation. This interface will be a tridimensional interaction with the bidimensional execution, allowing another axe to navigate. The output and interaction of the system are crucial. The user experience can determine if a system is used as well as any technical quality. For the visualization, we used WebGL and we reproduce a simulation on the web browser. The user can browse time and space, going backwards and forward.

5 Experimental results

To implement the simulator, we used the C programming language compiling the code with GCC 4.7, and the message passing library OpenMPI 1.6.4. For all the simulations, we used the previously calculated number of runs in section 3.2. The results shown in the experimental part have been executed in the instance m1.small with 0,613GB RAM, 1 processor and Ubuntu OS.

We used the *perf* tool to analyze the memory usage with the case that will be used in the experimental part: *Siena fiesta del Palio*. The execution is waiting 27.1 % frontend idle, 12.04% back-end idle. In the Siena case, the program mallocs dynamically 6.6MB of data. In this case, we have a mean of 1,950 page faults, where every memory page is 4 KB. This means an average access of 8MB to main memory and that our problem is limited by memory. Choosing a more powerful instance will waste resources and money. That is why we are working with the previous instances.

5.1 Model flows validation

There are several approaches solving the validation problem. One is video tracking, where all the individuals in an evacuation drill are set in our system and we try to do the same. Another one is to track almost all of them and predict the individuals that are not being tracked. In our case, because we are interested in the evacuation times. To validate this model these requirements, we will carry out a functional validation. We have a set of flows with several scenarios and several velocities and evacuation times. The simulator has been tested with these configurations, setting a random position to the agents and running the simulations calculated in the statistical model.

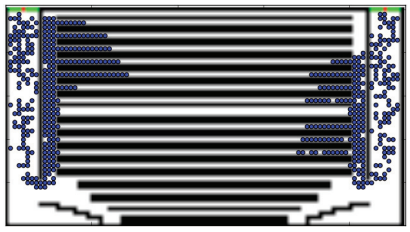

Figure 3: Tampere theater map.

In 1995 an experiment was carried out in the Tampere Theater[11] where about 612 people where evacuated. The distribution of the theater can be seen in Fig. 3 with agents evacuating. The evacuation time was 3 minutes 37 seconds (217 sec) and the people started the evacuation

in a time between 25 and 47 seconds. Here we validate the evacuation times. In table 1 the results of the execution using the simulator can be seen.

Table 1: Table with Tampere times.

Place	Evacuation time
Real case	217 seconds
Model	205 - 220 seconds

The mean walking speed is 1,4m/s, but in evacuation cases the crowd speed depends on the density. We used the velocity/density relation based on the linear model previusly presented[10]. The simulator calculates in execution time the density of the crowd to determinae the public speed.

Figure 4: (a) Festa del Palio di Siena. (b) System input of Palio di Siena.

5.2 Scenario

We chose two different scenarios to test our simulator. The Tampere theater, which was tested on evacuation. The *festa del Palio* of Siena[13], where around twenty thousand persons meet to see an event: the horse race. Here we are interested in a crowd event in a closed space. In the Fig. 4 (a) we can see the crowd in the center of the square. We chose these scenarios for validation and to have a real scenario as a proof of concept of our simulator.

We introduced by default a set of individuals: 28,378 persons filling the center of the square. In Fig. 4 (b) we can see the system input. The black space represent obstacles (in the center we can see the Fonte di Gaia); the white spaces, which represent the free space; and the blue patches, which are free spaces filled by people.

5.3 Evacuation performance

Because the simulations are independent, we have a parallel model using the embarrassing parallel computing paradigm. The simulations are divided in chunks by an static scheduler and are delivered to the computing nodes. The distribution of workload between MPI tasks is done using a static scheduler. The number of simulaitons is divided among the total number of nodes and then distibuted. We selected several instances and numbers of nodes (4, 8 and 12) to test the performance of this parallelization. In this case, we do not use the parallel file system to save partial results, because it will penalize the performance. We use the local file system to store temporary results. As we see in Fig. 5 the speed-up show to be almost linear for all kinds of instances, and this means the application scales in the infrastructure. This is due to the almost total lack of communication. As seen previously, the simulator scales efficiently. This is

because our application is not limited by compute, but by memory, because the matrices that hold are constantly accessed.

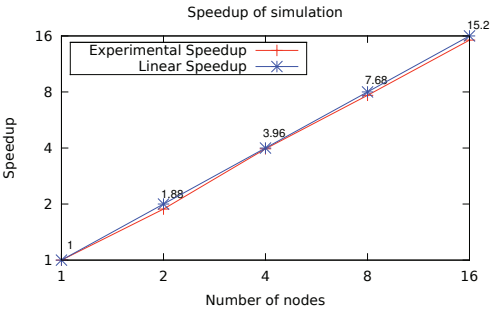

Figure 5: Speedup.

5.4 Crowd evacuations cloud tool

To implement the workflow, we implemented the front-end using the Apache web server with a Python module to program the web scripts. The web page allows the user to upload the PNG image. Once the file has been uploaded, it parses the image and runs the simulator.

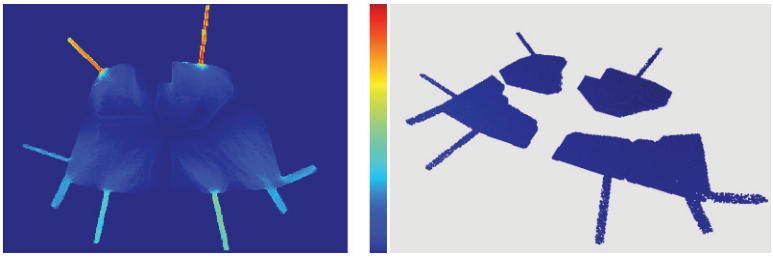

Figure 6: (a) Piazza del campo evacuation heat map. (b) Offline visualization of the evacuation.

The performance of the tool is one of the most important factors, due the fact that the results of the simulator is the aim of the present research. We also calculated the heat map in order to analyze the zones where there is more concentration of the public, and also more possibilities of accidents. In Fig. 6 (a) we can see an example of a heat map with the viewer of the system. In the Fig. 6 we can see an example of the crowd simulation viewer. In this case we have a simulation step where we can analyze the agents positions and the distribution among the space.

6 Conclusions

Crowd evacuations is an important problem in evacuation planning, and in taking decisions to maintain the spaces safe. In this paper we showed how we developed a computational model and implemented a simulator to be able to evacuate individuals in bi-dimensional spaces. Afterwards we created a methodology to hide the complexity of the model to the user. The input of the

system is a PNG image that holds the scenario information. This allows the system to have the necessary information to provide the user with the statistical results and an interactive visualization of the simulation. The simulation shown to scale on HPC systems and on the cloud. The cloud allows us to use IaaS to provide SaaS. StarCluster creates the MPI cluster automatically, distribute the load and finish the work in Nth times faster, in our experiments, were N is the number of nodes. Because our simulator is memory limited, we do not need powerful instances and small work is fine. This allows us to spend the same amount of money in computation saving time and provides us with faster results. To validate the model we used a real event in the Tampere theater and we compared the output of the simulator with the drill.

Acknowledgment

This research has been supported by the MINECO (MICINN) Spain under contract TIN2011-24384.

References

[1] Oassys MassMotion software, http://www.oasys-software.com/products/ engineering/ massmotion.html.

[2] STEPS software, http://www.steps.mottmac.com/.

[3] A. U. K. Wagoum, M. Chraibi, J. Mehlich, A. Seyfried and A. Schadschneider "Efficient and validated simulation of crowds for an evacuation assistant", *Computer Animation and Virtual Worlds.*, 2012, vol. 23(1), pp. 3-15.

[4] P. C. Tissera, A. M. Printista and E. Luque, "A hybrid simulation model to test behaviour designs in an emergency evacuation". *Proc. 27th International Conference on Computational Science.*, 2012, pp. 266-275.

[5] D. Helbing, I. Farkas, T. Vicsek, "Simulating dynamical features of escape panic" in *Nature.*, 2000, vol. 407, pp. 487-490.

[6] A. Gutierrez-Milla, F. Borges, R. Suppi and E. Luque, "Individual-Oriented Model Crowd Evacuations Distributed Simulation", *Proc. 29th International Conference on Computational Science.*, 2014, pp. 1600-1609.

[7] T. Kretz, C. Bnisch and P. Vortisch, "Comparison of various methods for the calculation of the distance potential field", *Proc. 4th. International Conf. Pedestrian and evacuation dynamics Conf.*, 2008, pp. 335-346.

[8] I. Diaz-Emparanza, "Seleccin del nmero de replicaciones en un estudio de simulacin", *Estadstica espaola.*, 1995, vol. 37(140), pp. 497-509.

[9] B. J. Mohler, W. B. Thompson, S. H. Creem-Regehr, H. L. Pick Jr and W. H. Warren Jr, "Visual flow influences gait transition speed and preferred walking speed", *Experimental Brain Research.*, 2007, vol. 181(2), pp. 221-228.

[10] R. Lubas, M. Mycek, J. Porzycki and J. Was, "Verification and Validation of Evacuation ModelsMethodology Expansion Proposition", *Transportation Research Procedia.*, 2014, vol. 2, pp. 715-723.

[11] B. Mandt, T. Meyer-Knig, "Evacuation AnalysisTheatre in Tampere Finnland", *TraffGo GmbH*, 2002

[12] STAR - Software Tools for Academics and Researchers: Cluster, http://star.mit.edu/cluster/.

[13] The Palio of Siena, http://www.aboutsiena.com/palio-of-Siena.html.