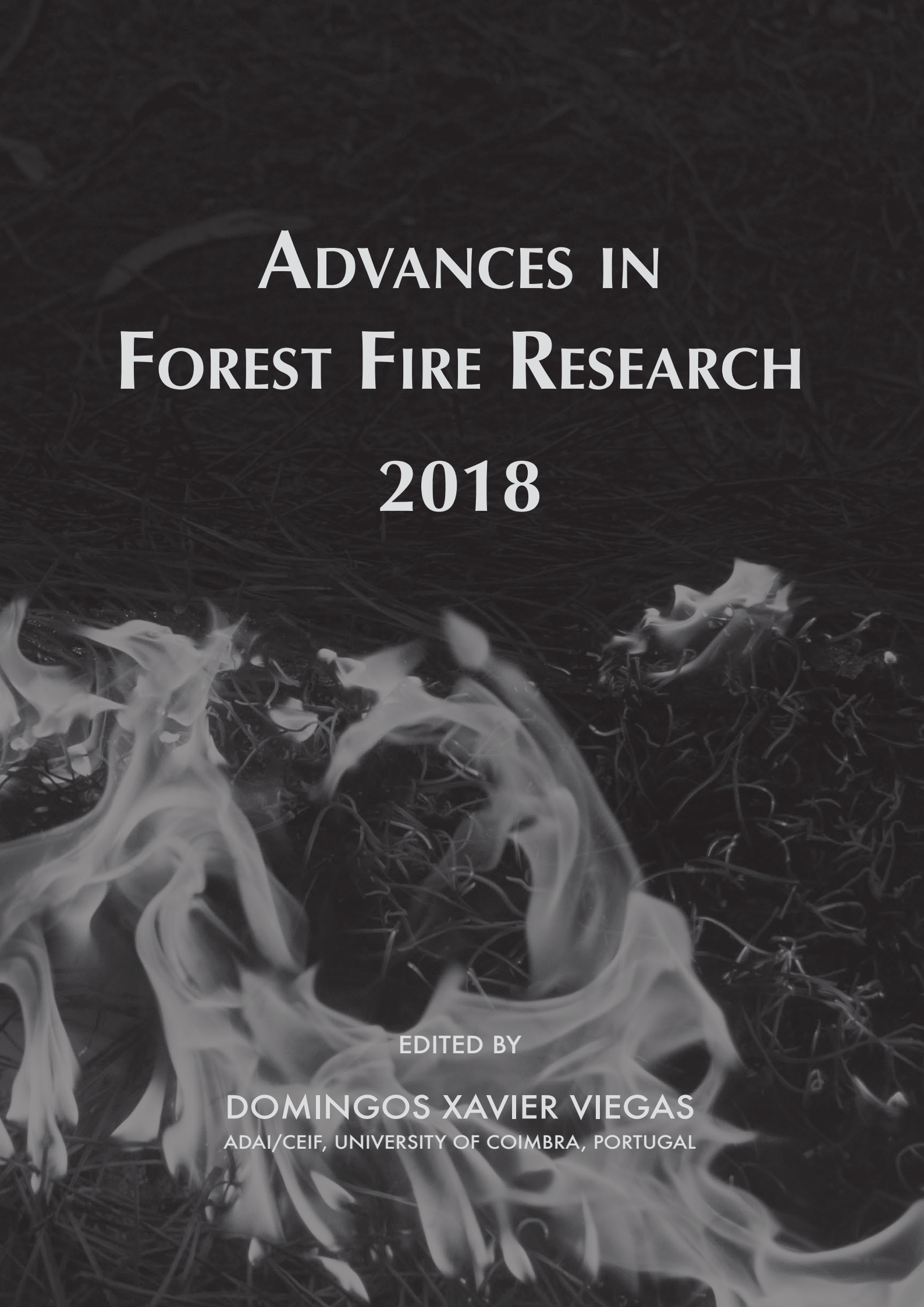
Advances in Forest Fire Research 2018



EDITED BY

DOMINGOS XAVIER VIEGAS

ADAI/CEIF, UNIVERSITY OF COIMBRA, PORTUGAL

https://doi.org/10.14195/978-989-26-16-506_100

Applying GPU Parallel Technology to Accelerate FARSITE Forest Fire Simulator

Carlos Carrillo*; Ana Cortés; Andrés Cencerrado; Antonio Espinosa; Tomàs Margalef
Computer Architecture and Operating Systems Department. Universitat Autònoma de Barcelona, Spain. {carles.carrillo@uab.cat, ana.cortes@uab.cat, andres.cencerrado@uab.cat, antoniomiguel.espinosa@uab.cat, tomas.margalef@uab.cat}*

Abstract

Forest fires are one of the most common natural hazards in the countries of the Mediterranean region. The forest fires spread simulators have proven to be very effective tools in the fight against these disasters. But to be able to use these simulators in an effective way it is necessary to be able to obtain realistic predictions of the behavior of the fire in a relatively short time. In the last decades, the advances in the field of computing have allowed different strategies to improve the effectiveness of simulators. With this objective, it has been tried to take advantage of the computing power of the GPUs (*Graphic Processors Units*) to accelerate the simulation of the propagation of fires. The problem with the use of GPUs is that the data transfer between the memory of the system and the GPU memory increases the execution time. For this reason, the computation operations have to compensate the transfer time of the data from the CPU (*Central Processor Unit*) to the memory of the GPU; which means, that the data volume must be sufficient to compensate this effect. The main objective of this work is to analyze the execution time of the forest fire spread simulation in a CPU and in a GPU, and to see from what data volume the execution in GPU is more efficient than in the CPU. For this study, a fire simulator has been designed based on the basic model for one point evolution in the simulator FARSITE. As study case, a synthetic fire was used where the fire progresses in a linear front and in a single direction, maintaining constant wind, terrain and vegetation conditions for all points of the fire front and throughout the simulation.

Keywords: Forest Fire Simulation, Parallel computing, GPU

1. Introduction

Forest fires are significant hazards that every year cause important damages around the world and have a deep impact on the natural environment, people and communities. This type of catastrophes is especially important in warm countries, like the countries of the Mediterranean zone, where thousands of hectares of forest burn every year, causing significant damages to the environment and economic losses. Proper prediction of the fire evolution allows to manage the firefighting equipment properly in order to mitigate the fire effects. Fire spread models have been developed and implemented in computing simulators to help command and control centers in taking the adequate decisions. To use these models and simulators effectively, it is of paramount importance to establish an appropriate measure of confidence in their performance with an acceptable execution time.

In order to improve the effectiveness of fire simulators, in the past decade different strategies have been developed to improve the accuracy and reduce the execution time using multicore architectures. In this context, most studies focus on multicore architectures based on CPU (Cencerrado et al., 2015; Brun et al., 2014). However, in the last years, the computational capacity of Graphics Processing Units (GPUs) has increased enormously (see Fig. 1). Due to their computational power, GPUs have become increasingly attractive for complex systems modelling, (like forest fire spread). The GPU architecture is ideally suited for high volume data processing applications because can process millions of operations simultaneously.

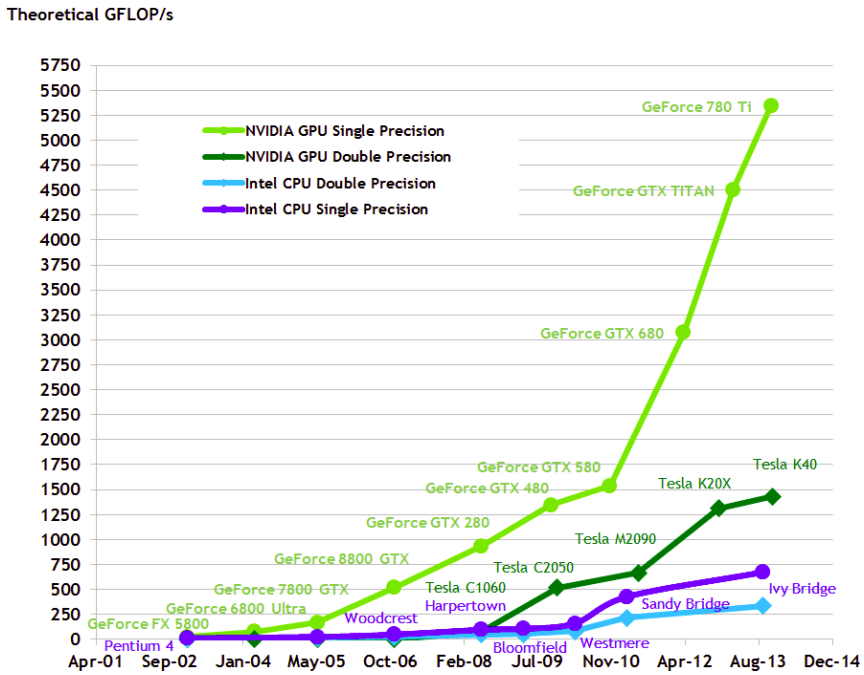


Figure 1 - Computation capacity of GPUs in the last few years in Gigaflops per second.

In recent years, different works have been carried out to apply the computational capacity of GPUs to the simulation of forest fires behavior and to accelerate these simulations (Gregoria et al., 2013; Ntinas et al., 2017; Sousa et al., 2012). These works have been focused on the application in simulators based on cellular automata (CA) that are relatively easy to parallelize. These studies have demonstrated that the use of GPU can reduce the execution time significantly without losing resolution. However, the main limitation is the intrinsic low accuracy of the simulators based on CA approach.

In order to effectively use a GPU, programmers need to build scalable programs, typically in CUDA programming model, that are able to use all GPU resources. For that, they have to deal with memory management, load balance of the execution threads, data races and data transfer cost. A GPU program workflow follows the general idea shown in Fig. 2. The necessary data is downloaded to the GPU (*Device*) from the CPU (*Host*). The operations are executed in parallel in the GPU and the result obtained is returned to the CPU.

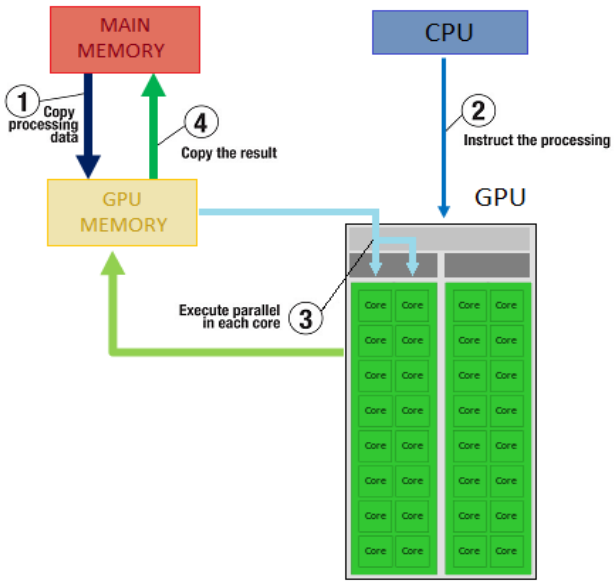


Figure 2 - General workflow of a GPU program

This data flow between the CPU and the GPU means that part of the program's execution time is wasted in the data copy, which implies that the GPU utilization only makes sense if the size of our problem is large enough, so that the data copying time is compensated by the computation time gain.

The main objective of this work is to analyze both the CPU and GPU execution time of a given forest fire spread simulator in order to determine the data volume requirements that implies higher efficiency using GPU than CPU.

In particular, in this work, we apply the GPUs parallel technology to *FARSITE* (Fire Area Simulator) (Finney, 2004), which is a forest fire simulator based on the Huygens principle and on the Rothermel's model as forest fire propagation (Rothermel, 1972). Simulators based on the Huygens principle, or elliptical wave propagation, have a higher precision than those based on cellular automata, however their execution time is higher. To carry out our study, we have extracted the FARSITE simulation kernel and we have implemented it in CUDA (*Compute Unified Device Architecture*) to re-incorporate it into the FARSITE body, and thus compare the two implementations. As first approach to the problem, a synthetic fire is used, which consists of a linear front in a flat terrain, maintaining constant wind speed, wind direction and the vegetation conditions throughout the simulation. To be able to compare the two executions (GPU and CPU) the simulations have been limited to one hour propagation, consequently, the execution time will be mainly linked to the volume of data to be treated and not to the propagation time of the fire to be simulated.

This paper is organized as follows. Section 2 describes different types of forest fires spread simulators. Section 3 includes a brief introduction to the GPU architecture and the methodology used to perform the spread simulation of the fire in this device. Section 4 presents the experimental procedure and the obtained results and finally, Section 5 summarizes the main conclusions and future work.

1. Forest Fires Spread Simulators

In the last decades, a large number of forest fires spread simulators have been developed. Most of these simulators are based on the Rothermel's fire spread model (Rothermel, 1972)**,** which is formulated in the following way:

$$\boldsymbol{R = \frac{I_R \xi \cdot (\vec{n} + \vec{\phi}_w + \vec{\phi}_s)}{\rho_b \varepsilon Q_{ig}}} \quad (1)$$

where R represents the rate of spread in a particular point, I_R is the reaction intensity, ξ is the propagation flux ratio, ρ_b is the ovendry bulk density, ε is the effective heating number, Qig is the heat of pre-ignition, $\vec{n}$ is the normal direction to the fire perimeter on that particular point $\vec{\phi}_W$, is the wind factor and $\vec{\phi}_S$ is the slope factor. Except, $\vec{\phi}_W$ and $\vec{\phi}_S$, all the factors only depend of the vegetation of the cell. Equation 1 can be rewritten in the following way:

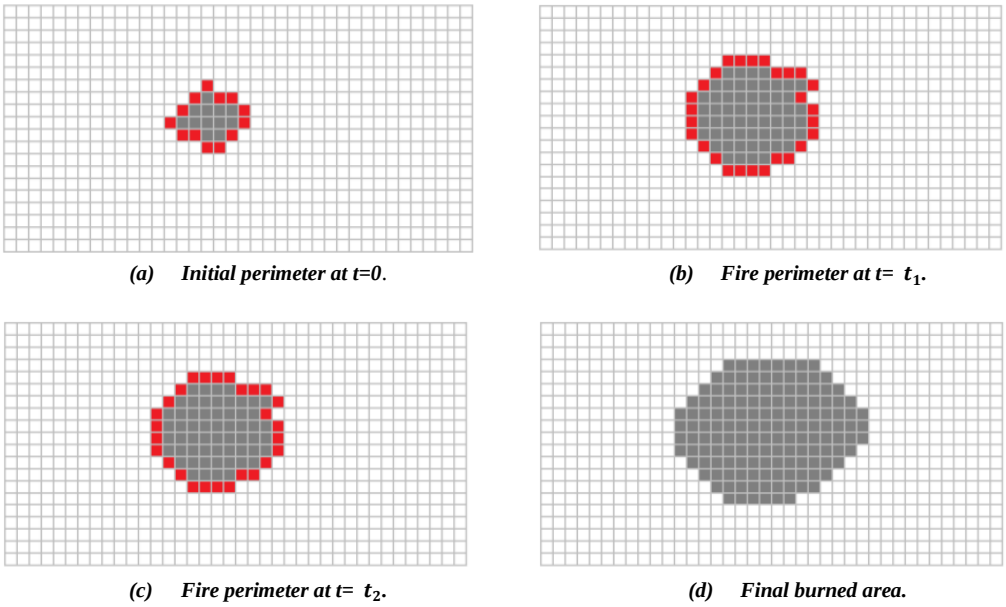
$$\boldsymbol{R = R_0 \cdot (\vec{n} + \vec{\phi}_W + \vec{\phi}_S)} \quad (2)$$

where R_0 represents the rate of spread in a particular point with no wind and no slope.

If we make a classification of the forest fire spread simulators according to the method used to propagate the fire front, three large groups can be defined:

- **Raster-based Approaches or Cellular Automata**: In the Raster-based or Cellular Automata simulators, the area where the fire takes place is divided into cells forming a network of cells (Alexandridis et al., 2008). The fire advances from one cell to the next through a set of rules that describe the spread of fire (Fig. 3). These rules are determined by the fire model used by the simulator. While faster and simpler to implement, they lack

precision when compared to other approaches (Pastor et al., 2003), but its parallelization is simpler than the Vector-based Approaches parallelization which is subsequently described. Fire simulators that use this approach are *FireLib*, *FireMap* and *Cardin*.



(a) Initial perimeter at t=0. ***(b) Fire perimeter at t= t_1.***

(c) Fire perimeter at t= t_2. ***(d) Final burned area.***

Figure 3 - Simulators based on cellular automata. In this case, the white color indicates unburned cells, the red indicates the cells burning at that instant and the gray cells are those completely burned.

- **Vector-based Approaches or Elliptical wave propagation:** In this type of simulators the perimeter of the fire is divided into series of points, where the distance between the different points will be determined by the resolution of the simulator (Knight et al., 2003). To obtain the evolution of the perimeter of the fire, for each point a ellipse is generated. The shape of the ellipses is determined by the local characteristics at each point. In this way, the new perimeter is obtained by joining the obtained ellipses (Fig. 4). In this type of simulators, the propagation of each point of the perimeter is done in a continuous space. The accuracy of this type of simulators is greater than the CA approaches; however, their complexity and time requirements are higher. *SiroFire*, *Prometheus*, and *FARSITE* are relevant examples of this kind of forest fire spread simulators.

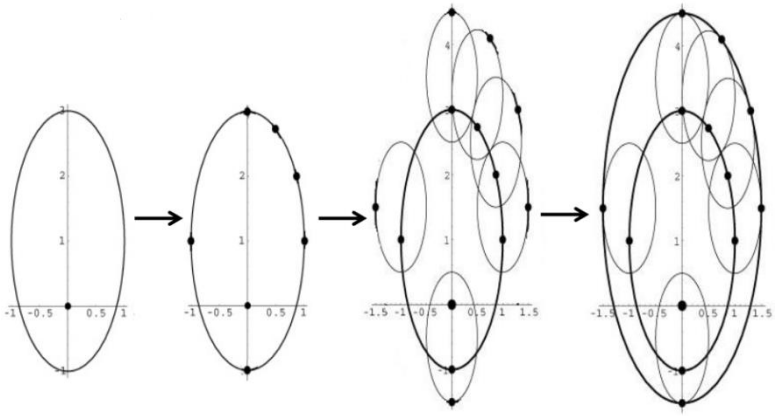


Figure 4 - Elliptical wave propagation from t1 to t2

- **Level Set Method:** In this kind of fire simulators, the burned region is defined as a level set function $\psi = \psi(\vec{x}, t)$ whose values are:

$$\begin{cases} \psi(\vec{x}, t) \leq 0 \ \ Burned\ Area \\ \psi(\vec{x}, t) = 0 \ \ Fire\ Front \\ \psi(\vec{x}, t) > 0 \ \ Saved\ Area \end{cases} \quad (3)$$

and it evolves through the partial differential equation

$$\partial\psi/\partial t = R\|\nabla\psi\| \quad (4)$$

where R is the rate of spread (Farguell et al., 2017). Equation 4 can be solved numerically given initial and boundary values for ψ, as well as, a model to compute the rate of speed. An advantage of this method is that the behavior of fire fronts arises naturally from the underlying mathematics and thus does not require the special handling that is needed when an Elliptical wave propagation method is used. *WRF-SFIRE* uses this approach (Fig. 5).

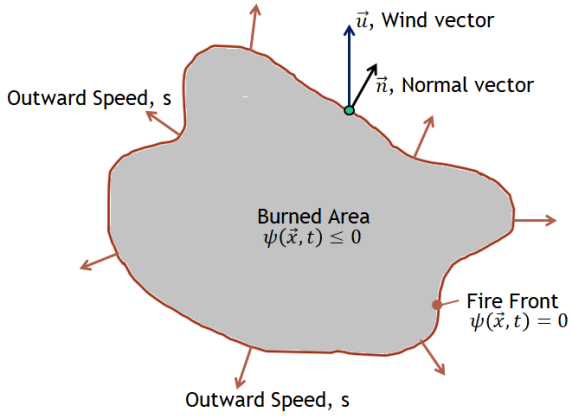


Figure 5 - Simulator based on Level Set Method.

2. Forest Fires Simulator Based on GPUs

As previously mentioned, in this study we focused on FARSITE simulator, which is based on elliptical wave propagation. As we have seen, in the simulators based on the Huygens principle, the expansion of a point only depends on the local conditions of this point and, therefore, it does not depend on the expansion of the other points. When a simulation in FARSITE is started, the resolution of the perimeter must be indicated. This perimeter resolution defines the distance between points of the perimeter of the fire. Higher perimeter resolution provides more accurate simulations. But, on the other hand, a higher resolution implies a greater number of points to compute, and therefore, a longer execution time.

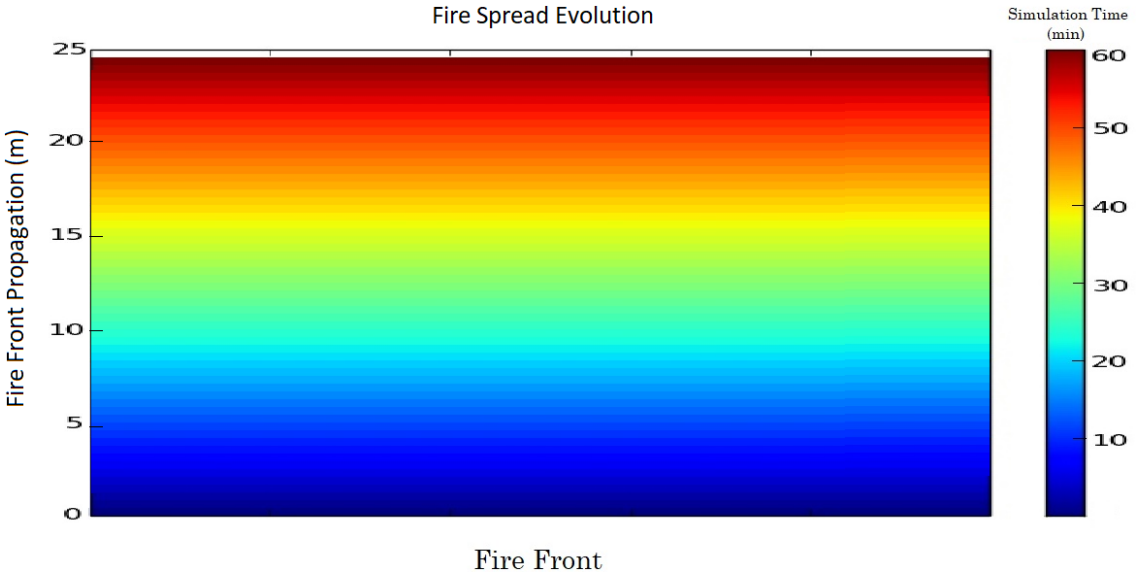


Figure 5 - Evolution of the fire front. The y-axis shows the propagation distance of the fire in meters. The color scale indicates the time step of the simulation in minutes.

As a first approximation, a synthetic fire that advances on a linear front has been used. As a simplification, the boundary problems at the front ends are ignored. In this particular case, it has been considered a flat terrain, with a homogeneous vegetation and constant wind speed and wind direction during the whole simulation. Under these conditions the propagation will be the same for all points of the fire front. Fig. 5 shows the evolution of the fire front, the color scale indicates the evolution time of the fire, while the ordinates axis shows the distance advanced by the front.

When executing the simulation on CPU, the point's evolution is computed sequentially, see Fig. 6. Therefore, if the number of points to expand is increased, the execution time will proportionally increase. For that reason, simulations with high resolutions provide high execution time, which limits their use in real situations. On the other hand, reducing the resolution implies that the execution time will be reduced, but the accuracy of the simulation will be penalized and therefore, the simulation will not properly describe the behavior of the fire.

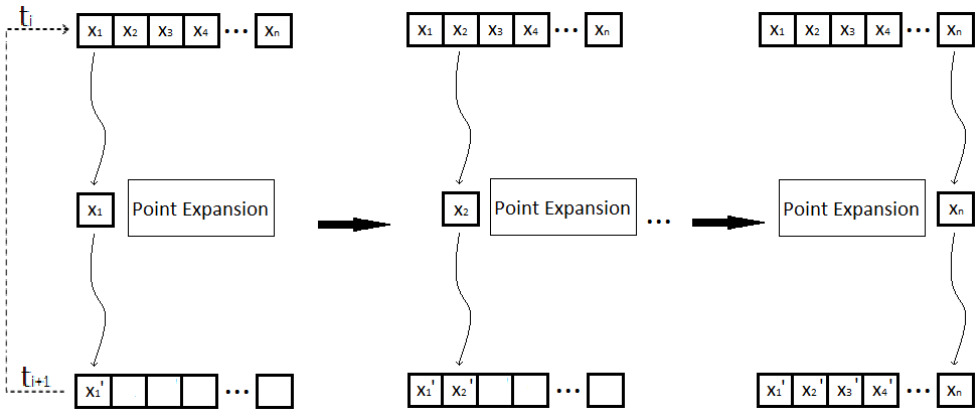


Figure 6 - Workflow execution of point's expansion in the CPU.

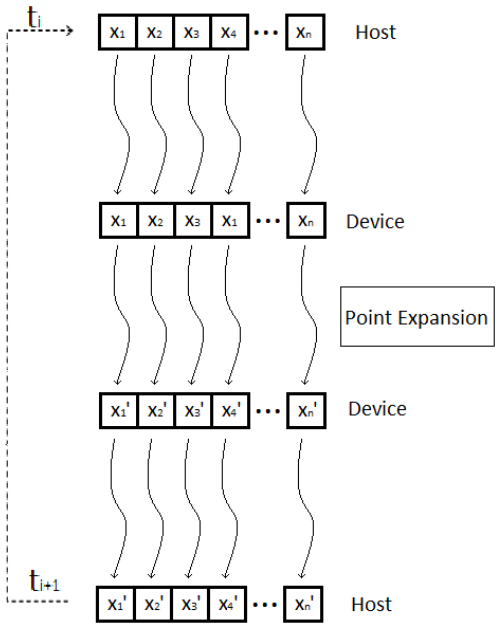


Figure 7- Workflow execution of point's expansion in the GPU.

One way to manage a simulation with high resolution is to be able to compute as much points of the fire front as possible in parallel. Computational accelerators like GPUs, provide the computer characteristics that allow to exploit this parallelization requirement. Therefore, the previous described point propagation scheme has been modified to fit the parallel scheme required by a GPU (Fig. 7). In this implementation, the number of fire points can be increased without increasing the execution time excessively.

To evaluate the performance of the proposed parallelization, five different simulations have been performed on CPU and GPU for each number of points between 1,000 and 50,000, with intervals of 1,000 points. The obtained results are presented in the next section.

3. Experimental Study and Results

As execution platform, we have used an Intel (R) Xeon (R) CPU E5-2603 v2 @ 1.80GHz, with 4 cores and for the GPU simulations, a GeForce GTX TITAN X with 3072 CUDA cores was used. As it was previously mentioned, the fire propagation time has been limited to one hour, so the execution time of each simulation will be mainly determined by the volume of data to be treated, not by the fire evolution time.

Fig. 8 shows the execution time when the simulation is done in CPU (Red) and in GPU (Blue) depending on the number of points of the fire front. As it can be seen, the execution time in the CPU increases significantly when the number of points increases. We can observe that for a small number of points, the execution in GPU is slower than executing the same number of points in CPU. This is because data transfer from CPU memory to GPU memory consumes most of the execution time. From 15,000 points the execution in GPU is faster than the execution in CPU.

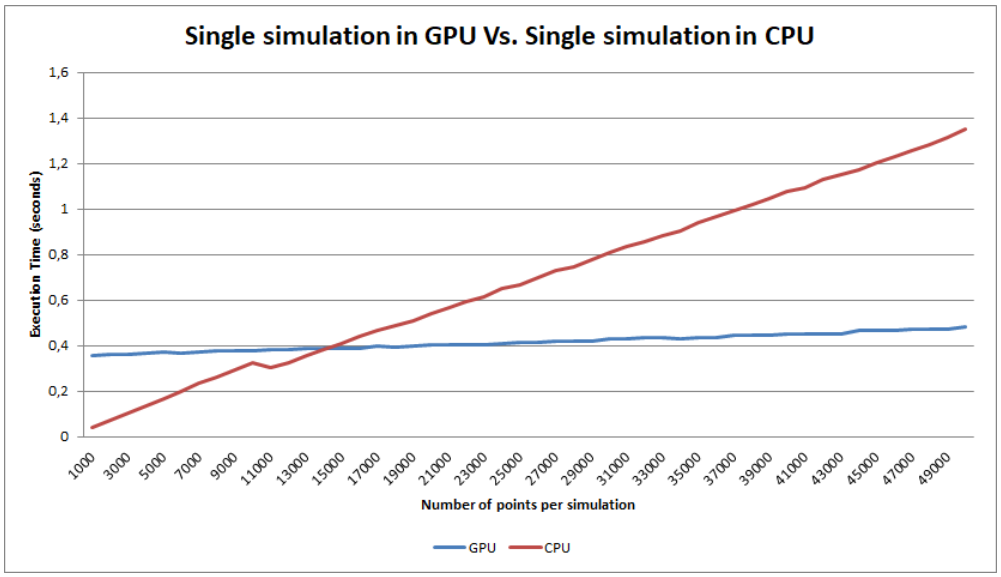


Figure 8 - Average execution time in CPU (Red) and GPU (blue) depending on the number of simulated points.

The Fig. 9 shows the speed-up of the simulator when it is running on the GPU. As it can be seen, when the number of points are above 15,000, the speed-up is higher than 1, which indicates that the execution in the GPU is more efficient than on the CPU.

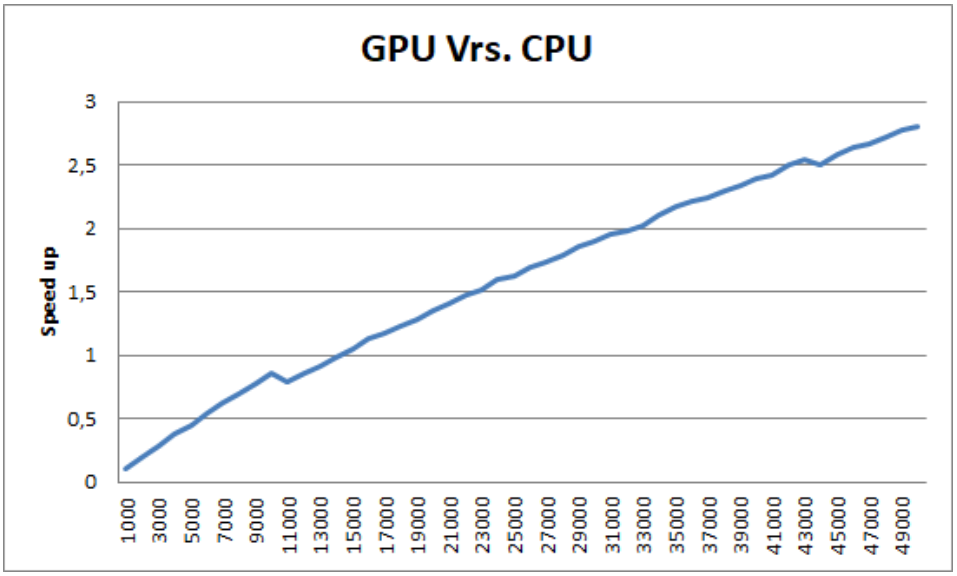


Figure 9: Average speed-up of the simulator in GPU depending on the number of front points.

4. Conclusions and Future work

In recent years, we have witnessed a significant increase in the computational capabilities of GPUs. This computing power makes GPUs ideal for those problems that require the processing of a large amount of data. In our case, we have focused on the study of forest fire propagation simulators based on the Huygens principle, in particular FARSITE. In this work, a synthetic fire has been used, with a linear fire front and constant wind and vegetation conditions throughout the simulation.

According to the study carried out, when the simulator is executed in the GPU most of the execution time is invested in the data copy between memories. For this reason, the volume of data to be computed must be large enough to be able to justify this copying time. It has been shown that, with a number of points greater than 15,000 points, the copying time is compensated by the computation time, so it is advisable to perform the execution on the GPU. On the other hand, when the number of

points is lower, the copying time is not compensated by the execution time, so the execution in CPU is more efficient than in GPU.

That preliminary work open a new way of approaching forest fire spread simulation in the sense that we expect to obtain more accurate results and, at the same time, faster and, therefore operationally simulation time

5. Acknowledgments

This research has been supported by MINECO-Spain under contract TIN2014-53234-C2-1-R and TIN2017-84553-C2-1-R, and by the Catalan government under grant 2017-SGR-313.

6. References

A. Cencerrado, T. Artés, A. Cortés, and T. Margalef, "Relieving uncertainty in forest fire spread prediction by exploiting multicore architectures," in *Proceedings of the International Conference on Computational Science, ICCS 2015, Computational Science at the Gates of Nature, Reykjavík, Iceland, 1-3, June, 2015, 2014, 2015, pp. 1752-1761.*

C. Brun, T. Margalef, A. Cortes, and A.a Sikora, "Enhancing multi-model forest fire spread prediction by exploiting multi-core parallelism," *The Journal of Supercomputing, vol. 70, no. 2, pp. 721-732, 2014.*

Finney, M.A. "FARSITE: Fire Area Simulator–model development and evaluation". *Research Paper RMRS-RP-4 Revised. Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station, 2004.*

Gregorio S.D., Filippone G., Spataro W., Trunfio G.A. "Accelerating wildfire susceptibility mapping through GPU". *Journal of Parallel and Distributed Computing 2013; 73(8):1183 –1194.*

Ntinas V.G., Moutafis B.E., Trunfio G.A., Sirakoulis G.C. "Parallel fuzzy cellular automata for data-driven simulation of wildfire spreading". *Journal of Computational Science 2017; 21:469 – 485.*

Sousa F.A., dos Reis R.J.N., Pereira J.C.F." Simulation of surface fire fronts using firelib and GPUs". *Environmental Modelling and Software 2012; 38:167–77.*

Rothermel R. "A mathematical model for predicting fire spread in wildland fuels"; *1972.*

A. Alexandridis, D. Vakalis, Constantinos I. Siettos, and George V. Bafas, "A cellular automata model for forest fire spread prediction: The case of the wildfire that swept through spetses island in 1990," *Applied Mathematics and Computation, vol. 204, no. 1, pp. 191-201, 2008.*

E. Pastor, L. Zárate, E. Planas, J. Arnaldos. "Mathematical models and calculation systems for the study of wildland fire behaviour". *Progress in Energy and Combustion Science*, 29(2):139-153, 2003.

I. Knight and J. Coleman, "A fire perimeter expansion algorithm-based on Huygens wavelet propagation," *International Journal of Wildland Fire* **3**, 73-84. 1993.

À. Farguell, A. Cortés, T. Margalef, J.R. Miro, and J. Mercader. Data resolution effects on a coupled data driven system for forest fire propagation prediction. *Procedia Computer Science, 108:1562 – 1571, 2017. International Conference on Computational Science, ICCS 2017*, 12-14 June 2017, Zurich, Switzerland.