Journal of Computational Science 17 (2016) 576–590

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Hybrid application to accelerate wind field calculation

G. Sanjuan, T. Margalef*, A. Cortés

Computer Architecture and Operating Systems Department, Universitat Autònoma de Barcelona, Cerdanyola del Vallés, Spain

ARTICLE INFO

Article history:
Received 6 November 2015
Received in revised form 1 April 2016
Accepted 25 July 2016
Available online 1 August 2016

Keywords:
Forest fire propagation
Wind field
Sparse Matrix-Vector multiplication
Domain decomposition
Map partitioning

ABSTRACT

Wind field calculation is a crucial point in predicting forest fire evolution in real emergencies. Such wind field calculation involves complex models that must be solved in a very short time to accomplish the real-time constraints of forest fire propagation prediction. However, when the terrain map under consideration is large (for example, 30 km × 30 km) and the resolution is high (for example, 30 m), the time incurred by wind field calculation makes its use unfeasible in real emergencies. To accelerate the wind field calculation, several approaches have been studied (map partitioning, domain decomposition and matrix storage). The results are quite significant, but in large cases, the execution time does not meet the expected objectives. Therefore, a new approach, integrating all the previous work, has been designed, and a hybrid MPI-OpenMP version of WindNinja has been implemented. This hybrid application exploits different sources of parallelism to reduce execution time without introducing a significant loss of accuracy in wind field calculation. The results obtained with different terrain maps show that the execution time is reduced to below an established limit of 100 s.

1. Introduction

Forest fires are natural hazards that cause significant losses every year. Such hazards are particularly critical in places with hot and dry summer seasons, such as Mediterranean countries, California and Australia. To fight these hazards and use the available resources in the best possible way, it is necessary to have an accurate prediction of their evolution beforehand. So, propagation models have been developed to determine the expected propagation of a forest fire [23,2]. Such propagation models have been implemented in computer simulators [9,4,17,18,13,22,21] that receive several input parameters representing the scenario where the fire is taking place to produce the predictions of the propagation. These parameters include the digital elevation map, the vegetation map, the vegetation features, and meteorological conditions, among others. Some parameters are well known, but the values of other input parameters are obtained or estimated from indirect measurements. Such indirect estimations provoke a degree of uncertainty in forest fire propagation prediction. Therefore, the accurate measurement or estimation of the input parameters is crucial in providing useful predictions.

The parameters that most significantly affect forest fire propagation are wind speed and direction [1]. These parameters can be measured at meteorological stations or estimated from meteorological models, but, in both cases, they are obtained at a very low resolution, typically a distance of some kilometres. This fact is critical because, as is well known, the meteorological wind is modified by the topography of the terrain and, therefore, the values of the wind speed and direction at one point of the terrain are different from the values at other points, and the values at low resolution are not representative of the actual situation. This may imply that the predictions provided by forest fire simulators are not feasible and not very useful in real operation.

To estimate the wind speed and direction at each point of the terrain, it is necessary to apply a wind field model that determines those values at each point while taking the terrain topography into account. Then, the wind field generated by the wind field simulator is used as input for the forest fire propagation simulator, coupling both simulators, wind field and forest fire, to improve the accuracy of forest fire propagation predictions [6].

Moreover, meteorological wind varies dynamically over time. So, when predicting fire propagation, it is necessary to consider the meteorological forecast provided by meteorological centres at different time steps and calculate the wind field for each time step. During the forest fire simulation, the corresponding wind field must be applied at each time step. So, if the meteorological forecast provides a wind value every 15 min and the forest fire propagation is predicted for the next 6 h, it is necessary to calculate 24 wind fields (four for each hour) and introduce each wind field into the proper forest fire simulation step.

FARSITE [9] is a widely used forest fire propagation simulator that has been extensively tested on real fires and produces successful results. It is a fire behaviour and fire growth simulator that

* Corresponding author.

http://dx.doi.org/10.1016/j.jocs.2016.07.011

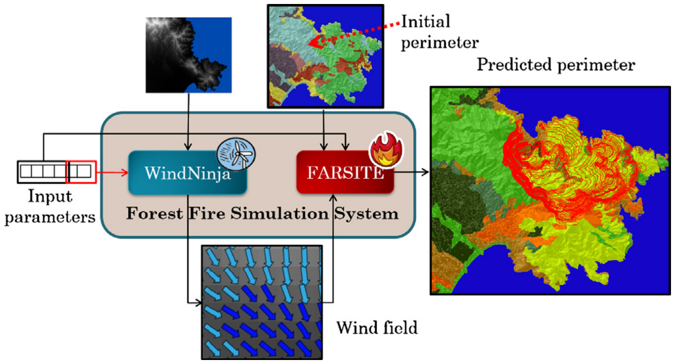

Fig. 1. Coupling wind field and forest fire propagation models.

incorporates both spatial and temporal information on topography, fuels and weather. It includes temporal variation in fire conditions. FARSITE is an elliptical wave propagation simulator and avoids a typical problem in cell-based simulators of reproducing the fire shape in heterogeneous conditions, due to their reduced number of propagation paths.

For the wind field simulator, WindNinja [10] is a mass conservation wind field simulator that, given a meteorological wind and a digital elevation map, generates the wind field at the needed resolution (30 m). One of the main advantages of WindNinja is that the output generated can be directly used as input by FARSITE. This simplifies the coupling of both models. Fig. 1 shows the coupling of WindNinja and FARSITE. WindNinja has also been used in other forest fire simulation environments and coupled with simulators such as WildFireAnalyst [22], Phoenix RapidFire [30] and WIFIRE [3].

Coupling a wind field simulator and a forest fire propagation simulator improves the accuracy of the forest fire propagation prediction, but it also significantly increases the computation time. For a large map (for example, 45 km × 45 km) with a high resolution (for example, 30 m), the execution time of the WindNinja wind field simulator on a single processor is around 3600 sconds, which makes it completely useless in real operation. Therefore, the main goal of this work is to reduce the WindNinja execution time for large terrain maps, such as 45 km × 45 km and 1500 × 1500 cells, in order to make it operational in real scenarios while accomplishing real-time constraints. Actually, an operational limit of 100 s has been established as a maximum execution time for the wind field simulator. In this way, the forest fire propagation prediction can be provided allowing the coordination centres to take the appropriate decisions and communicate the actions to be taken to the field firefighters.

In previous works, three approaches were proposed to exploit parallelism at different levels of WindNinja. These three approaches were the following ones:

- Sparse matrix storage format and Sparse Matrix-Vector multiplication (SpMV) [29].
- Domain decomposition with overlapping: Schwarz alternating method [28].
- Map partitioning [27,26].

The results obtained by the three approaches were quite significant. However, the scalability was limited and, in most cases, for large maps, the limit of 100 s was not met. This paper focuses on combining the three parallelization approaches and integrating them into a hybrid MPI-OpenMP application. This hybrid application exploits data parallelism in an MPI Master/Worker scheme by partitioning the terrain map and distributing the different parts among workers. Then, each part is solved by applying a new level of Master/Worker parallelism based on domain decomposition. Finally, the sparse matrix storage system is modified to avoid cache misses and the Sparse Matrix-Vector multiplication (SpMV) is parallelized applying OpenMP.

The rest of the paper is organized as follows. Section 2 describes the main features of WindNinja and presents its internal structure. Section 3 introduces the three approaches to parallelize WindNinja. Section 4 presents the integration of the three approaches into two hybrid MPI-OpenMP applications and shows the results obtained with these applications. Finally, Section 5 shows the main conclusions of this work.

2. WindNinja

WindNinja [10] is a wind field simulator that requires the elevation map and the meteorological wind speed (*ws*) and direction (*wd*) to determine the wind parameters at each cell of the terrain. The number of the cells of the terrain actually depends on the map size and resolution.

WindNinja is based on the equations that describe air flow variation in the atmosphere. Specifically, it is a mass-consistent model initialized by boundary conditions. The function to minimize is constructed using the square of the difference between the adjusted and observed values as shown in Eq. (1),

$$E(u, v, w) = \int_{\Omega} [\alpha_1^2(u-u_0)^2 + \alpha_1^2(v-v_0)^2 + \alpha_2^2(w-w_0)^2] d\Omega \quad (1)$$

where u, v, w are the velocity components in the x (positive to East), y (positive to North), and z (positive upward) directions, respectively; u_0, v_0, w_0 are initial values of velocity, and α_i is the Gauss precision moduli that can be used to control the relative amount of change induced by the model to the horizontal and vertical directions.

The minimization of Eq. (1) is subject to the strong constraint of conservation of mass that can be expressed as shown in Eq. (2).

$$H(u_x, v_y, w_z) = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) = 0 \quad (2)$$

This constraint can be expressed applying Lagrange multiplier theory. So, the function can be expressed as shown in Eq. (3),

$$F(u, v, w, \lambda) = \int_{\Omega} [\alpha_1^2(u-u_0)^2 + \alpha_1^2(v-v_0)^2 + \alpha_2^2(w-w_0)^2 + \lambda \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right)] d\Omega \quad (3)$$

where $\lambda(x, y, z)$ is a Lagrange multiplier.

This implies that terrain slope variation generates wind changes and, because of boundary conditions, the obtained results in regions close to the borders of the map will not be correct until the system is stabilized. Consequently, several external map cells have an unreliable value and, therefore, a set of cells around the evaluated map must be dismissed as a final result.

The internal functioning of WindNinja can be summarized as follows:

1. WindNinja takes the digital elevation map and the meteorological wind parameters and generates the mesh.
2. Then it applies the mass conservation equations at each point of the mesh to generate the linear system $Ax = b$.
3. In the linear system $Ax = b$, the A matrix is a sparse matrix which is stored in CRS (Compressed Row Storage) format. The matrix A has a low density and a diagonal pattern.