J Supercomput (2015) 71:1869–1881
DOI 10.1007/s11227-014-1365-9

Enhancing computational efficiency on forest fire forecasting by time-aware Genetic Algorithms

Tomàs Artés · Andrés Cencerrado · Ana Cortés · Tomàs Margalef

Published online: 21 December 2014

Abstract A way to overcome data input uncertainty when simulating forest fire propagation, consists of calibrating inaccurate input data by applying computational-intensive methods. Genetic Algorithms (GA) are powerful and robust optimization techniques. However, their main drawback is their overall run time, which can easily become unacceptable, especially when dealing with natural disasters forecast. The prediction system has been parallelized using a hybrid MPI-OpenMP approach where the number of cores allocated to each GA individual is based on a priori time-aware population classification, which allows to keep bounding the optimization process bound to a predetermined deadline. In this work, an efficient time-aware GA is introduced that estimates the required number of cores to keep the calibration process under imposed time limits and also takes into account an efficient use of the computational resources.

Keywords Multi-core platforms · Forest fire spread prediction · Hybrid MPI-OpenMP scheme · Core allocation · Efficiency · Time-aware

T. Artés (✉) · A. Cencerrado · A. Cortés · T. Margalef
Computer Architecture and Operating Systems Department,
Universitat Autònoma de Barcelona, Barcelona, Spain
e-mail: tomas.artes@e-campus.uab.cat

A. Cencerrado
e-mail: andres.cencerrado@uab.cat

A. Cortés
e-mail: ana.cortes@uab.cat

T. Margalef
e-mail: tomas.margalef@uab.cat

1 Introduction

When dealing with an ongoing natural disaster such as a forest fire, a critical point to consider is the response time of the emergency systems and their ability to act in the most efficient way. Experience in fire fighting and forest fire behavior knowledge are the basic key points used to decide how to tackle an active fire. In order to help fire fighting decisions, forest fire spread simulators, such as FARSITE [1], become a relevant tools in assessing decision support systems. However, to be effective, the forecasted forest fire behavior must be delivered prior to the real fire evolution. Forest fire spread prediction system must accomplish the task with real-time constraints in order to be useful. Furthermore, and not dismissible, there exists an inherent error related to any natural hazard prediction due to, among other things, the uncertainty in the data needed to perform the forecast. For the particular case of forest fire, we can find in the literature different approaches to tackle these problems ranging from applying ensemble strategies reducing the input parameters uncertainty effects [2] to applying the Kalman filter to certain input variables to tune their values [3]. However, most of these approaches are not concerned with response time.

In this work, we focus on strategies to relieve the uncertainty effects, due to the imprecision of input parameters, by ensuring a time limit. For that purpose, we used the so-called Two-Stage prediction scheme, which is composed of a Calibration stage where the input parameters values are tuned to better reproduce the observed past behavior of the fire, and those calibrated parameters are then used in the Prediction stage to forecast the forest fire evolution [4]. As a calibration strategy, the Two-Stage prediction scheme uses a Genetic Algorithm (GA).

The algorithm starts with an initial random population of individuals, each one representing a scenario to be simulated. An individual is composed of a set of different genes that represent input variables such as dead fuel moisture, live fuels moisture, wind speed and direction, among others.

Each individual is simulated and is evaluated by comparing the predicted and real fire propagation by estimating *the symmetric difference* between predicted and real burned areas divided by actual real spread [5]. This difference takes into account the wrongly predicted burned cells (false alarms) and the real burned cells that were not predicted (misses). According to this fitness function the whole population is ranked and the genetic operators *selection*, *elitism*, *mutation* and *crossover* are performed over the population, producing an evolved population which will have, at least, the best individual of the last generation (elitism). The new population is then evaluated in the same way, and the process is repeated for a certain number of iterations.

Although GAs are powerful and robust optimization techniques, their main drawback is their overall execution time which can easily become unacceptable. Moreover, FARSITE forest fire simulator execution time can vary from minutes to hours for the same topographic area depending on the particular combination of the input parameters. Consequently, the need arises to find a trade-off between prediction accuracy, achieved thanks to the calibration strategy, and the time incurred in reaching this prediction improvement. To harmonize quality and time, we propose a multi-threaded Genetic Algorithm that exploits multi-core platform in order to accelerate the Two-Stage forest fire spread prediction system. However, the significant variability of execution time of the forest fire simulator usually provokes significant load unbalance and a inefficient use of available resources. Therefore, an approach to estimate the amount of resources (cores) required to enhance the efficiency of resource usage has been introduced.

In the next section, the hybrid MPI-OpenMP prediction framework is described. Section 3 presents the time-aware core allocation scheme and introduces the efficient time-aware core allocation based on statistical data. The application of the enhanced prediction scheme to a real case is analyzed in Sect. 4, and finally, the main conclusions of this work are cited in Sect. 5.

2 Hybrid MPI-OpenMP master/worker prediction scheme

The calibration process, based on GA, allows us to find a good input parameter set, but it involves a high computational cost due to the large amount of simulations required. Therefore, it is essential to speed up the execution while maintaining the accuracy of the prediction. For this reason, the GA calibration method has been implemented using a master/worker paradigm and has been parallelized applying a hybrid MPI-OpenMP approach. So, the Two-Stage methodology has been developed to exploit two different level of parallelism.

On the one hand, since the GA fits the master/worker paradigm, an MPI implementation of this prediction framework was deployed (see Fig. 1a). At the first stage, the master node generates an initial random population which is distributed among the workers. Then, the workers simulate each individual and evaluate the fitness function. The errors obtained by the workers are sent to the master, which ranks the corresponding individuals according to their error before applying the genetic operators and producing a new population. This iterative process is repeated a fixed number of times. The last iteration (generation) contains a population from which the best individual is taken as the best solution, and is used in the Prediction stage.

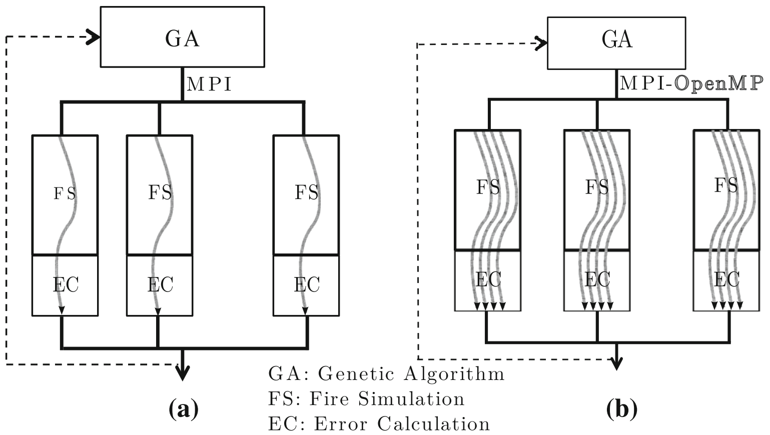

Fig. 1 MPI parallel calibration stage (**a**) and the two-level MPI-OpenMP master/worker prediction scheme (**b**)

Since every simulation can be carried out in a parallel way, the individual whose simulation takes the longest determines the elapsed time for that particular generation. So, the execution time of each GA iteration can be expressed as shown in Eq. 1, where t_{Ind} stands out from the execution time of a given individual and P_{Gen} is the set of individuals (population) at that particular generation.

$$t_{\text{iter}} = \max_{\text{Gen}}(t_{\text{Ind}}) \mid \forall \text{Ind} \in P_{\text{Gen}} \tag{1}$$

When extrapolating this fact to all GA iterations, one comes out with a total calibration execution time equal to the sum of the longest simulation time existing at each iteration. Therefore, the total execution time of the GA (t_{total}) could be evaluated using Eq. 2, where N_{Gen} is the number of generations performed of the GA, t_{Ind} stands out from the execution time of a given individual and P_{Gen} is the set of individuals (population) at the $genth$ generation.

$$t_{\text{total}} = \sum_{\text{Gen}=1}^{N_{\text{Gen}}} \max_{\text{Gen}}(t_{\text{Ind}}) \mid \forall \text{Ind} \in P_{\text{Gen}} \tag{2}$$

In order to shorten simulation times, a second level of parallelism was proposed. FARSITE was analyzed with profiling tools such as OmpP [6] and gprof [7] to determine which regions of the code could be parallelized with OpenMP. The result of such analysis determined the particular loops that could be parallelized using OpenMP pragmas. The results of such parallelization have been presented in [5].

The parallelized loops represents about 60 % of the FARSITE's execution time, which means that 40 % of the execution time is sequential. It implies a non-linear speedup. Figure 1b sketches the implemented hybrid MPI-OpenMP scheme. Equation 3 expresses the theoretical minimum execution time we could obtain using this OpenMP FARSITE parallelization as a function of the number of cores (N_{Cores}), where t_{s} stands for the execution time of the sequential version.

$$t_{\text{par}}(N_{\text{Cores}}) = 0.4 \times t_{\text{s}} + \frac{0.6}{N_{\text{Cores}}} \times t_{\text{s}} \tag{3}$$

However, as previously mentioned, the duration of a certain forest fire evolution simulation in a certain topographic area can take from minutes to hours depending on the particular set-up of the simulator input parameters. So, in one particular iteration it is common to have individuals that take a few minutes, while others take more than 1 h and determine the iteration execution time. In this situation, the most efficient approach to reduce iteration execution time is to dedicate more computational resources (cores) to those individuals that take longer and limit the overall iteration time. This approach is shown in Fig. 2.

So, one question that arises at that point is, what is the suitable number of cores to allocate to each given forest fire spread simulation? The next section deals with this problem by introducing a time-aware classification strategy that allows us to allocate the proper number of cores to each individual forest fire spread simulation with the aim of keeping the total prediction time bound.

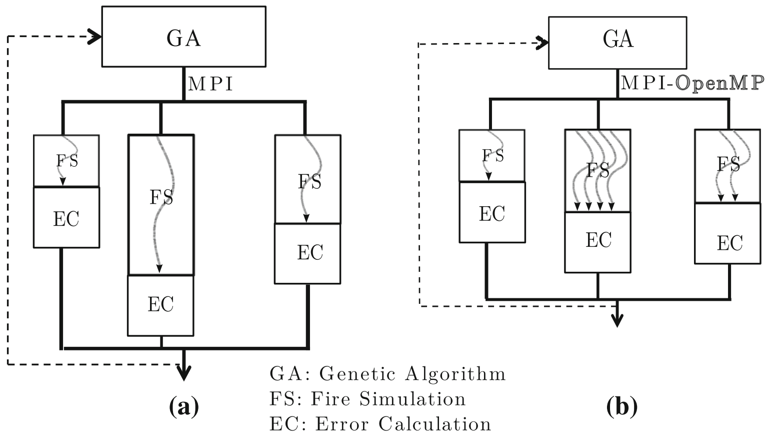

Fig. 2 Fire simulation individuals unbalance (**a**) and Core allocation depending on sequential execution time (**b**)

3 Efficient TAC-two-stage prediction scheme

The prediction scheme has been parallelized using a hybrid MPI-OpenMP approach, and a time-aware core allocation scheme has been implemented to ensure a simulation time limit for each executed GA individual. Thus, the proposed solution allows us to keep the prediction time bound to the predefined time prediction requirements, enabling the capacity of delivering forest fire behavior information useful to the wildfire analysts in charge of the fire management. Furthermore, the proposed time-aware Genetic Algorithm has also been analyzed in terms of efficiency in order to exploit to the maximum the available computational resources.

The key point of this approach is allocating more cores to the individuals that take longer, so that their simulation time can be reduced, improving the individuals unbalance and, consequently, the iteration execution time. Therefore, it is necessary to estimate the execution time of each particular individual beforehand, so that the appropriate number of cores is allocated to each individual. However, this execution time cannot be estimated from the direct analysis of the underlying input data values. So, we rely on the characterization methodology described in [8]. For a given topographic area, this methodology has the ability to assess in advance the execution time of the forest fire spread simulation associated with certain input parameter settings. This methodology takes advantage of the artificial intelligence field to generate a decision tree, which is able to classify the individuals of a new generated population into time classes. This classification is based on the serial version of the FARSITE simulator. As stated in the previous section, FARSITE has been parallelized using OpenMP pragmas. The parallelized part of FARSITE represents approximately 60 % of the total execution time, meanwhile 40 % of the execution time corresponds to a sequential part, which cannot be reduced by the implemented parallelization.

Dealing with strict real-time constraints implies setting up a time limit to the Calibration Stage and, consequently, to each GA iteration. Therefore, applying the FARSITE

Table 1 Time limit classes for a time constraint of $t_{\max_{\text{gen}}}$

Class	Cores	Time limits
A	1	$0 < t_s \leq t_{\max_{\text{Gen}}}$
B	2	$t_{\max_{\text{gen}}} < t_s \leq 1.42 \times t_{\max_{\text{Gen}}}$
C	4	$1.42 \times t_{\max_{\text{Gen}}} < t_s \leq 1.81 \times t_{\max_{\text{Gen}}}$
D	8	$1.81 \times t_{\max_{\text{Gen}}} < t_s \leq 2.1 \times t_{\max_{\text{Gen}}}$

multi-threaded version and stating that the ($t_{\text{par}}(N_{\text{Cores}})$) determines the maximum GA iteration time exploiting the parallel FARSITE version, from Eq. 3 one can state the maximum serial time permitted to accomplish a generation limit time ($t_{\text{par}}(N_{\text{Cores}})$) depending on the number of cores (see Eq. 4).

$$t_s = \frac{t_{\text{par}}(N_{\text{Cores}}) \times N_{\text{Cores}}}{0.6 + 0.4 \times N_{\text{Cores}}} \tag{4}$$

Therefore, assuming a maximum GA generation time of $t_{\max_{\text{Gen}}}$ and four core allocation configurations: 1, 2, 4 and 8 cores per FARSITE simulation, one would be able to define four time execution classes with their respective time limits as is shown in Table 1.

Considering that the parallel part of FARSITE is 60 % and the speedup is not linear, because of the significant sequential part, it is not worth increasing the number of cores to 12 or 16. The actual time reduction obtained compared to the one with 8 cores is not significant.

The above-mentioned characterization methodology based on decisions tree to assess in advance the serial execution time of a given FARSITE execution, could be redefined to classify the individuals of a GA population according to the new time limit classes, which are associated with the number of cores required to maintain the execution time of each simulation below $t_{\max_{\text{Gen}}}$. Therefore, the core allocation scheme becomes time constraint aware. That means, that the number of cores allocated to a given parallel FARSITE simulation will be sure to fit the predetermined deadlines. The calibration scheme integrating the time-aware classification (TAC) approach is called *TAC-Two-Stage Prediction Scheme*. Under this scheme, any serial FARSITE simulations whose estimated prediction time goes beyond the time limit $2.1 \times t_{\max_{\text{Gen}}}$, will discarded and will not be executed because no core allocation configuration exists that could guarantee the delivery of the corresponding simulation result on time.

To ensure that the iteration time limit is not exceeded, when this time limit is reached the executing and pending individuals are killed, and the GA is executed considering only those individuals that have completed their execution.

This approach keeps the GA generation time under the maximum iteration time. However, it is necessary to estimate the amount of resources required to execute the TAC-Two-Stage Prediction Scheme that ensures the generation time, makes an efficient use of the resources and provides quality calibrations. So, in order to use the computational resources more efficiently, we propose an efficient TAC-Two-Stage scheme (E-TAC). This scheme estimates the amount of resources required and applies an "on-demand" distribution of individuals to the available cores that maintains the following criteria:

Table 2 Number of cores required

Class	Cores per individual	Number of individuals	Total number of core
A	1	49	49
B	2	8	16
C	4	5	20
D	8	2	16
Total			101

– *Longer execution time class first* Similar to the Long Queue First (LQF) scheduling, all individuals will be sorted depending on their class. So, long class individuals will be delivered first to the workers.
– *Best fit remaining worker iteration time next* For each generation, the time-aware classification uses the current available execution time to determine the classification limits (such as Table 1) and it allows us to add the remaining time, if it exists, to the execution of the individuals of the current generation. So, E-TAC is able to adapt depending on the remaining time trying to execute as many individuals of the GA as possible.

The key point is how to determine the amount of resources required. In the worst case, all individuals in the population could belong to class D. In this case, if the population has 64 individuals, it would be necessary to use 8 cores for each individual. This makes a total number of 512 cores. However, the statistical distribution of the individuals among the classes shows that in a random population of 64 individuals the distribution of individuals among the 5 classes is 48 class A individuals, 8 class B individuals, 5 class C individuals, 2 class D individuals and 1 class E individual (that is discarded and changed to class A). In this situation the amount of cores required to execute all the individuals in parallel is 101, as shown in Table 2.

However, it must be considered that not all the class A individuals take 1,080 s, but that most of them take less time. Actually the average execution time is around 450 s. So, we could allocate more than 2 class A individuals per core. From Table 2 it can be concluded that the class B, C and D individuals require 52 cores,and the class A individuals require less than 24 cores. So, the total number of cores could be around 76 cores. But, it must be taken into account that the class B, C and D individuals running on 2, 4 and 8 cores, respectively, will not take 1,080 s and the remaining time can also be used to execute class A individuals. So, the number of 76 cores can be reduced by around 20 % which makes for a number of 61 cores, that can be rounded to 64 cores.

In the next section, the proposed *Efficient Time Aware Classification* is tested in a real case to verify the feasibility of this proposal in coping with the real needs of wildfire analysts during a real hazard, taking into account an efficient use of the available computational resources.

4 Case study

The Mediterranean area is one of the European regions most affected for forest fires during the summer seasons. The selected case study corresponds to a region within

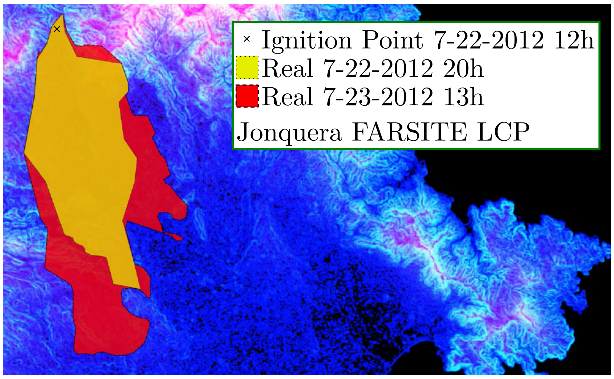

Fig. 3 La Jonquera forest fire

the Mediterranean coast that is affected by forest fires almost every year. In particular, we used a fire that occurred in *La Jonquera* (North-East of Catalonia, Spain) in July 2012. This hazard devastated nearly 13,000 ha and two people died. Figure 3 shows the burn area associated with this forest fire for two different time instants during the fire occurrence. The digital elevation map used has a resolution of 30 m.

The computing platform used to test the proposed scheme consists of two PowerEdge C6145 nodes. Each node has 4 AMD Opteron™6376 of 16 cores with 128 GB of DDR3 1,600 MHz, which means a total number of 128 cores.

In order to evaluate the Two-Stage prediction scheme including *TAC*, and to integrate it with the EFFIS (European Forest Fire Information System) [9], a maximum calibration time of 3 h must be accomplished. The MODIS Aqua and Terra satellites provide two fire perimeters every day (around 10 am and 3 pm, respectively). Once both perimeters are available, it is possible to start the calibration process. For operational reasons it is important to provide the forest fire propagation prediction for the next day around 7pm. This means that the calibration process must take less than 3 h. So, we have imposed a 3 h time limit for the calibration process.

As previously described, the Calibration stage implements a GA to reduce the prediction inaccuracy due to the input data uncertainty. The GA population size has been set up for 64 individuals. The initial population is generated randomly and the GA has been iterated ten times. Therefore, since the calibration stage has been limited to 10,800 s, the maximum execution time of one GA iteration is limited to 1,080 s. Then, assuming the best executing scenario where all GA individuals are executed in parallel, the maximum serial time that should be executed at each GA iteration should take 1,080 s. However, the unpredictable nature of GA population members could lead the system to a GA total execution time that widely exceeds the self-imposed time constraints.

The proposed Two-Stage prediction scheme with *Time Aware Classification* (*TAC-Two Stage*), described in Sect. 3, exploits the FARSITE OpenMP parallelization, to accomplish the predefined time constraints. In the case study, the time of each GA iteration has been set to 1,080 s. Then, assuming that the computational platform has enough cores to run all GA individuals (64 individuals) in parallel, independently of the

Table 3 Time limit classes for a time constraint of 1,080 s per GA generation

Class	Cores	Time limits (s)
A	1	$0 < t_s \leq 1{,}080$
B	2	$1{,}080 < t_s \leq 1{,}547$
C	4	$1{,}547 < t_s \leq 1{,}966$
D	8	$1{,}966 < t_s \leq 2{,}273$

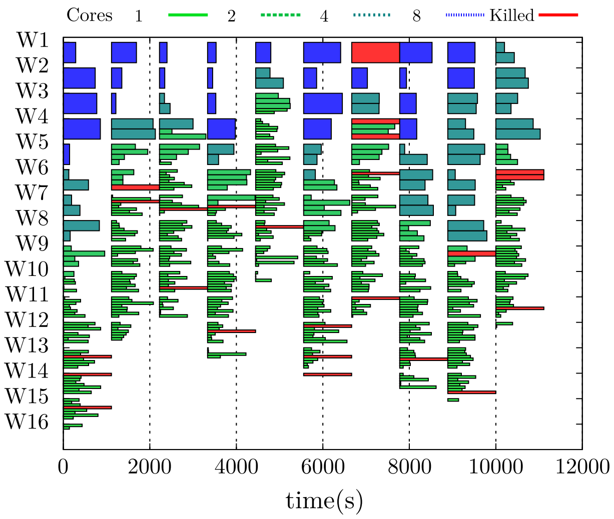

Fig. 4 Execution trace of the Calibration stage using the *Time Aware Classification* (TAC-Two-Stage prediction scheme)

number of cores allocated to each one, the maximum parallel time of one simulation is limited to 1,080 s. Then, substituting the values in Eq. 3 the resulting equation corresponds to Eq. 5 and, consequently, the resulting serial time depending on the number of cores allocated is the one stated in Eq. 6.

$$1{,}080 = 0.4 \times t_s + \frac{0.6}{N_{\text{Cores}}} \times t_s \tag{5}$$

$$t_s = \frac{N_{\text{Cores}} \times 1{,}080}{0.6 + 0.4 \times N_{\text{Cores}}} \tag{6}$$

Therefore, the resulting time limits from the serial execution time allowed to be executed depending on the number of cores used, are the ones shown in Table 3.

Any execution of the forest fire simulator, whose estimated serial execution time takes longer than 2,273 s will be discarded and, consequently, eliminated from the calibration process. In order to keep the population size constant, those killed individuals are replaced by new individuals at run time so as not to penalize the GA search.

Considering that the test platform has 128 cores, the prediction system uses a master/worker scheme with 16 workers, each one with eight assigned cores. So, the total number of cores used is 128 cores. In Fig. 4, we can see the GA evolution and the execution of the individuals for all generations taking into account the number of cores allocated to each one. The wider the time line is, the more cores are used for the corresponding FARSITE simulation. For example, at the first iteration, the worker labeled *W1* has one single FARSITE simulation assigned using 8 cores, while worker *W9* has 4 FARSITE simulations assigned, running on 2 cores each. As can be observed, the self-imposed limit of 10,800 s for the Calibration stage has been accomplished.

One concern that could arise from this approach is directly related to the error achieved at the end of the calibration stage. Since individuals that are classified as too long (more than 2,273 s for this particular case) are discarded and replaced by new individuals, one can be concerned about the loss of diversity in the GA population and how it affects the final result. However, previous works [10] stated that discarding long individuals in the prediction scheme, does not affect the final result in terms of calibration accuracy. Therefore, being able to assess in advance an interval execution time (class) for any FARSITE simulation, enables the Two-Stage prediction system with the capacity of classifying the GA individual according to their estimated elapsed time.

However, as can be observed in Fig. 4, maximizing the available parallelism to execute all individuals in parallel, directly affects the system efficiency. Figure 4 has an efficiency of 27.07 %, which is very poor. This low efficiency is clearly caused by the workers' idle times. As can be observed, as the GA evolves, the number of workers with individuals allocated varies due to the unpredictable nature of the GA population evolution. We have reserved 128 cores to be used during the calibration process. However, in this case, the GA drives the solution to individuals classified as C. The class C individual will be allocated 4 cores each, but also due to the GA behavior, not all new individuals will match this class as can be observed at the 5th generation. In this generation, the number of class D individuals is only 1, the number of individual classified as class C is 2, there are 4 individuals of class B and the rest of the individuals are class A members. Consequently, workers 11 to 16 are idle workers because the number of resources has been overestimated. This fact clearly impacts the system efficiency. Therefore, in order to use the computational resources more efficiently, the *Efficient TAC-Two-Stage* scheme is applied. To properly analyze the efficiency variation when dealing with less computational resources, we have set up the execution platform to have 64 cores, as stated in the previous section. It corresponds to using 8 workers with 8 cores assigned to each. Figure 5 shows the results of applying this strategy to the *La Jonquera* case.

In order to obtain results comparable to not applying the efficient strategy, the initial population of the GA has been set to the same as the one used in the experiment shown in Fig. 4. This population has 5 class D individuals, 6 class C individuals, 5 class B individuals and 48 class A individuals. This population fits in the statistical distribution shown in Table 2. There are more D and C individuals and the number of B individuals is a little bit lower than the average value for that class, but these variations correspond to the statistical behavior. Actually, the initial population would require more cores than the estimated ones.

As we can observe, the total calibration time of 10,800 s is achieved and the idle worker times are considerably reduced. The efficiency of the obtained scheme is in this case 53.61 %. It implies a significant improvement with respect to running all

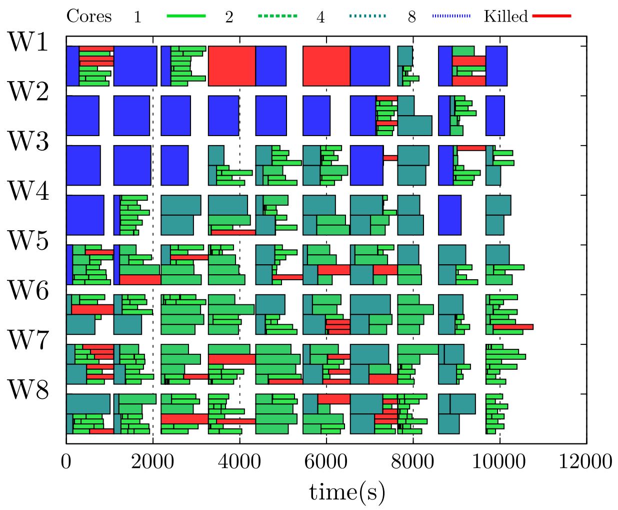

Fig. 5 Execution trace of the Calibration stage using the *Efficient TAC-Two-Stage prediction scheme*

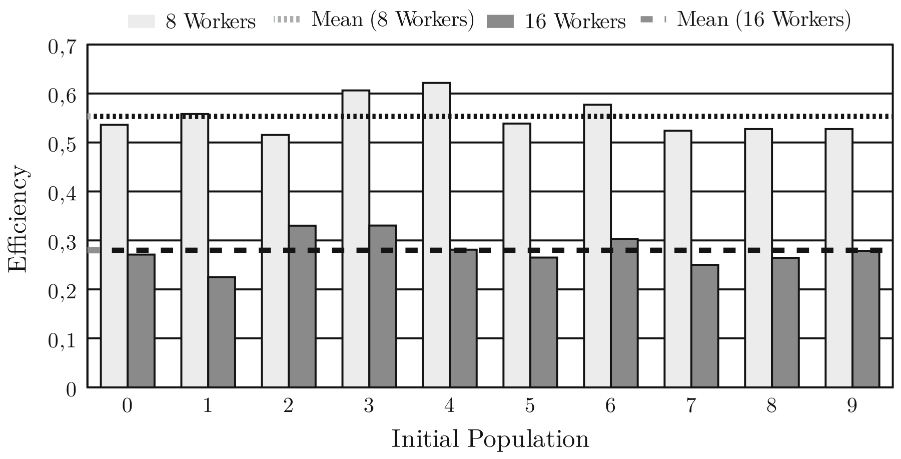

Fig. 6 Computational efficiency when applying the *Efficient TAC-Two-Stage* scheme using 8 (64 cores) and 16 workers (128 cores)

individuals in parallel. In order to consider the non-deterministic aspect of the GA, the above described experiments have been repeated 10 times with different initial populations and the obtained results are shown in Fig. 6. In this figure, the efficiency values for all experiments when using 64 and 128 cores are depicted. As can be observed, the mean value when using 8 workers is twice the mean value when 16 workers are applied. Therefore, using the proposed prediction scheme, one can release computational resources that could be used for different purposes such as running complementary models to enhance accuracy results [11].

Although we are applying the *Efficient TAC-Two-Stage* approach, that keeps track of the remaining iteration time for each core assigned to each worker, one cannot dis-

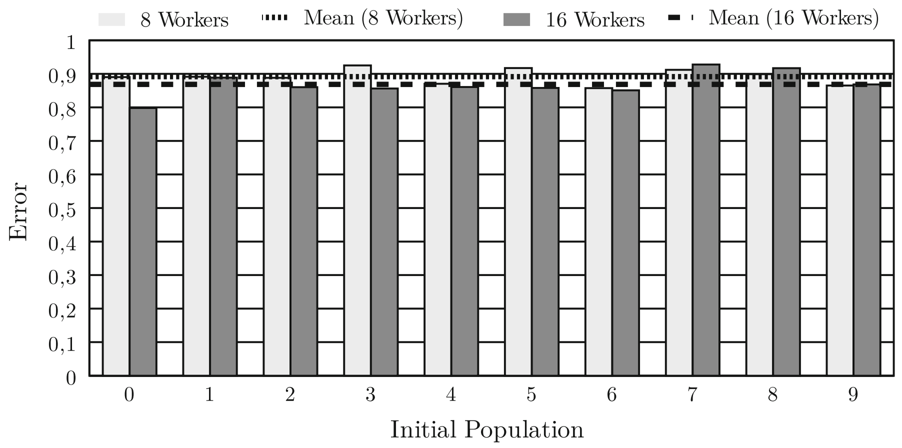

Fig. 7 Final calibration error when applying the *Efficient TAC-Two-Stage* scheme using 8 (64 cores) and 16 workers (128 cores)

miss the classification error that could be made when applying the *TAC* classification. Despite the low classification error provided by *TAC*, the system must include a safety feature to be sure that the preset deadline will not be passed. This could happen when an individual whose real execution time would correspond to class, for example D, is eventually classified by error to belong to class C. Under these circumstances the number of cores allocated to that individual will be 4 instead of 8. This could lead to its execution time taking longer than the available iteration time. For that reason, those anomalous cases are detected on-line and killed when the auto-imposed iteration time has been reached. The individuals depicted in red in Fig. 5 correspond to killed individuals. So, a question that arises is how this individual's elimination affects the final error. Figure 7 shows the calibration error achieved for ten different runs of the *Efficient TAC-Two-Stage*, as well as the corresponding mean values for 8 and 16 workers, respectively. As one can observe, the error difference is not relevant remaining in all cases very close to 0.9. As a consequence, one can state that applying the efficient TAC-Two-Stage prediction approach will have no effect on the final calibration error compared with not using the efficient scheme. There are even three runs with 8 workers and 64 cores which achieve slightly better error than with 16 workers and 128 cores (7, 8 and 9 in Fig. 7). Nevertheless, in terms of mean, error is still better with 128 cores, although the difference in mean error between 8 workers and 16 workers is not significant.

5 Conclusions

The TAC-Two-Stage prediction approach exploits the multi-thread FARSITE version to fit the prediction time within a current time limit. To achieve such a goal, a time-aware core allocation strategy was included in the basic Two-Stage prediction scheme, which is able to estimate in advance the maximum execution time of a given simulation and, consequently, determine at run time, how many cores it allocates to each simulation to accomplish the desired time constraints. When strictly using TAC we are able to provide a forest fire spread prediction on time but it can incur very poor efficiency in the MPI-OpenMP parallel calibration stage. For that reason, a scheduler module has been designed and implemented to improve system efficiency. In this work, we describe an efficient time-aware parallel forest fire spread prediction scheme (*Efficient TAC-Two-Stage*). This improvement estimates from statistical data the required number of cores and integrates a scheduling policy that enables the system to deliver prediction results under real-time constraints provided by the forest fire management services. This approach is able to reach the same calibration quality with less computational resources that releases enough cores that can be used to run complementary models and enrich the input data quality. The system has been tested with a real forest fire that took place in Catalonia in July 2012. The results show that, using the estimated number of cores, the prediction scheme has no penalization in terms of calibration quality and significantly increases the efficiency. In turn, the available free resources allows us to run complementary models such as wind field models improving the final prediction results.

Acknowledgments This work has been supported by MICINN-Spain under contract TIN2011-28689-C02-01 and by the Catalan government under grant 2014-SGR-576.

References

1. Finney MA (1998) FARSITE, Fire Area Simulator-model development and evaluation. Res. Pap. RMRS-RP-4, Ogden, UT: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station
2. Rodriguez-Aseretto D, de Rigo D, Di Leo M, Cortés A, San-Miguel-Ayanz J (2013) A data-driven model for large wildfire behaviour prediction in Europe. Proc Comput Sci 18:1861–1870
3. Mandel J, Bergou E, Gratton S (2013) 4dvar by ensemble kalman smoother. arXiv preprint arXiv:1304.5271
4. Abdalhaq B, Cortés A, Margalef T, Luque E (2005) Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques. Future Gener Comput Syst 21(1):61–67
5. Artés T, Cencerrado A, Cortés A, Margalef T (2013) Relieving the effects of uncertainty in forest fire spread prediction by hybrid mpi-openmp parallel strategies. Proc Comput Sci 18:2278–2287
6. Fürlinger K, Gerndt M (2008) A Profiling Tool for OpenMP. OpenMP Shared Memory Parallel Programming, pp 15–23
7. Graham SL, Kessler PB, McKusick MK (2004) gprof: a call graph execution profiler. SIGPLAN Not 39(4):49–57
8. Cencerrado A, Cortés A, Margalef T (2014) Response time assessment in forest fire spread simulation: an integrated methodology for efficient exploitation of available prediction time. Environ Model Softw 54:153–164
9. San-Miguel-Ayanz J, Barbosa P, Schmuck G, Libertà G, Meyer-Roux J (2003) The european forest fire information system. In: AGILE 2003: 6th AGILE Conference on Geographic Information Science, p 27. PPUR presses polytechniques. http://forest.jrc.ec.europa.eu/effis/about-effis/
10. Cencerrado A, Cortés A, Margalef T (2012) Genetic algorithm characterization for the quality assessment of forest fire spread prediction. In: Proceedings of the International Conference on Computational Science, ICCS 2012 Procedia Computer Science, vol 9 (0), pp 312–320
11. Brun C, Margalef T, Cortés A, Sikora A (2014) Enhancing multi-model forest fire spread prediction by exploiting multi-core parallelism. J Supercomput 70(2):721–732