Estructura híbrida MPI-OpenMP para predicción de incendios forestales

Tomàs Artés Vivancos[1], Ana Cortés[2], and Tomàs Margalef[3]

Departamento de Arquitectura de Computadores y Sistemas Operativos, Universitat Autònoma de Barcelona

Resumen— **Una predicción precisa de la propagación del fuego es crucial para minimizar los efectos de este. Por lo tanto, se han desarrollado distintos modelos para determinar la propagación de incendios forestales con antelación. Estos modelos requieren una cantidad importante de parámetros de entrada que habitualmente no se pueden obtener de forma precisa en caso de emergencia. En consecuencia, se ha desarrollado una metodología basada en dos etapas con el fin de calibrar aquellos parámetros gran incertidumbre en casos reales introducen error en la predicción. La calibración de estos parámetros se ha afrontado como una búsqueda de un conjunto de parámetros. Para realizar esta búsqueda se ha utilizado un Algoritmo Genético que requiere una gran cantidad simulaciones. Además, en los casos en los que el frente del fuego es grande algunos parámetros no son uniformes a lo largo de todo el frente puede ser conveniente acoplar otro modelo como ocurre con el caso del viento. En este trabajo se utiliza un modelo para obtener campos de vientos. Todo este sistema de simulación trabaja bajo estrictas condiciones de tiempo. La predicción de la propagación se debe obtener tan pronto como sea posible para ser útil. Como consecuencia se deben explotar todos medios computacionales disponibles. Con este último fin, se ha desarrollado una aplicación híbrida MPI-OpenMP cuyo objetivo es minimizar el tiempo de predicción.**

Palabras clave— **Predicción de propagación de fuegos forestales, acoplamiento de modelos, campos de vientos, sistemas multicore, MPI, OpenMP, aplicaciones híbridas.**

I. Introducción

Los desastres naturales son problemas muy significativos para la sociedad por las importantes pérdidas que causan cada año. Inundaciones[1], tsunamis[2], huracanes[3], terremotos o fuegos forestales[4] son algunos de éstos desastres naturales. Una buena predicción del comportamiento de estos desastres naturales es un punto crucial para poder luchar contra ellos y poder minimizar los costes de los daños que causan. En consecuencia, muchos investigadores de distintos ámbitos han desarrollado modelos que representan el fenómeno y son capaces de proporcionar una predicción del comportamiento futuro en caso de emergencia[5][6]. Estos modelos necesitan una gran cantidad de parámetros de entrada, en caso de emergencia no se puede disponer de todos ellos y además los valores presentan una gran incertidumbre o error. Con el objetivo de solucionar la incertidumbre en los parámetros de entrada, se ha desarrollado una metodología basada en la observación de la propagación real del fuego para poder calibrar los parámetros de entrada. Los parámetros calibrados son entonces utilizados en la siguiente etapa de predicción para mejorar los resultados. Esta metodología ha sido probada en la propagación de incendios forestales con resultados significativos[7].

La metodología utilizada utiliza el comportamiento del fuego durante el intervalo $[t_0, t_1]$ y entonces realiza una búsqueda de los valores de las entradas como la velocidad del viento, la dirección del viento, la humedad del combustible muerto, humedad del combustible vivo, parámetros de combustión para distintos tipos de combustibles vegetales, etc. Al final de esta búsqueda se obtienen los parámetros que mejor reproducen el comportamiento del fuego durante el intervalo $[t_0, t_1]$. Para realizar esta tarea de búsqueda sobre un espacio tan grande y abundante en soluciones locales se han utilizado algoritmos genéticos[8], razonamiento basado en casos y métodos estadísticos[9]. Por los resultados obtenidos el calibrado se basará en un algoritmo genético. Una vez se ha encontrado los parámetros que mejor reproducen la propagación del fuego se utilizan entonces en una segunda etapa de predicción para el intervalo $[t_1, t_2]$. Este esquema (ver Fig.1), requiere una gran cantidad de ejecuciones, con escenarios distintos y una repetición iterativa. Dicho esquema que implementa el algoritmo genético se ha implementado una aplicación con el paradigma Máster/Worker utilizando MPI[10](ver Fig.2). El proceso Máster genera distintos individuos, conjunto de parámetros para cada escenario distinto, y los distribuye en los distintos Workers. Al finalizar el trabajo los workers cada uno de ellos calcula el error cometido respecto al comportamiento real y envían un error de predicción al Máster. El Máster ordena los individuos por su error, realiza las operaciones genéticas,como la selección, el cruzamiento, la mutación y el elitismo, y genera una nueva población para la siguiente generación o iteración.

La estructura de Máster/Worker proporciona una gran mejora de rendimiento dada la gran carga de trabajo que se distribuye en distintos nodos reduciendo así el tiempo de ejecución final. La carga de trabajo es considerable puesto que los parámetros pueden ser variables en el tiempo y en el espacio.

[1]tartes@caos.uab.es
[2]ana.cortes@uab.es
[3]tomas.margalef@uab.es

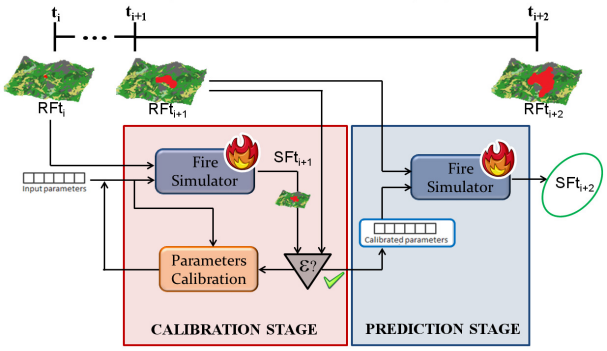

Fig. 1. Metodología basada en dos etapas

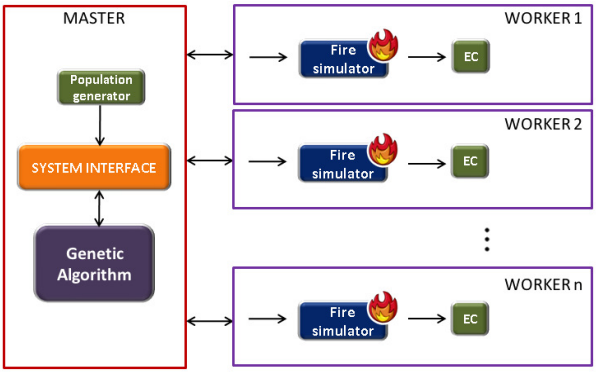

Fig. 2. Paradigma Máster/Worker

Un caso en el que es variable en tiempo y espacio, y además un factor muy importante en la propagación del fuego es el viento. Con el objetivo de conseguir los mejores valores posibles para este parámetro se ha acoplado un modelo que nos permite calcular campos de viento en función de la topografía del terreno. El siguiente apartado trata sobre el acoplamiento del modelo. El apartado 3 trata sobre la escalabilidad de cada modelo utilizando OpenMP[11]. En el apartado 4 se muestran los resultados obtenidos y por último, en el apartado 5 las conclusiones.

II. Acoplado de modelos de generación de campo de vientos y propagación de incendios

En trabajos previos[12], los parámetros se consideraban uniformes a lo largo de todo el terreno. Esto implica que se obtiene un único valor para cada parámetro para todo el espacio durante la etapa de calibración, y por lo tanto para su uso en la posterior etapa de predicción. Esta suposición es aceptable cuando se utiliza fuegos a pequeña escala, ensayos de propagación controlados, o fuegos sintéticos. Sin embargo, esta suposición en el caso de tratar con fuegos a gran escala en los cuales pueden arder cientos o incluso millares de hectáreas donde la topografía puede ser muy variada y por lo tanto también el viento en la superficie de ésta. Por lo tanto, se necesita acoplar un modelador de viento para superficies de terreno que proporcione campos de vientos en el esquema de predicción basado en dos etapas comentado anteriormente, para poder así considerar la dirección y velocidad del viento para cada punto del terreno durante cada paso de la simulación. De esta manera, el sistema de simulación podrá reflejar la influencia de la topografía del terreno sobre la dirección y velocidad del viento.

En el proceso de calibración, cada individuo calcula los campos de vientos utilizando los vientos meteorológicos correspondientes a un determinado escenario. A continuación el modelo de propagación de fuego utiliza los campos de vientos generados y considerar la dirección y la velocidad del viento para cada celda del mapa. Esto implica que cada proceso Worker requiere el cálculo de varios campos de vientos para cada escenario antes de realizar la simulación de la propagación del fuego. Esta modificación del proceso que ejecutan los Workers incrementa considerablemente el cómputo a realizar. Tanto el modelo de generación de campos de vientos como el de propagación de fuego utilizan mapas de celdas georeferenciados para representar los mapas, por lo que es conveniente trabajar con una alta resolución. Trabajar con mapas topográficos de grandes dimensiones con altas resoluciones incrementa el tiempo de ejecución de los modelos.

El proceso que ejecuta el Worker, una vez finalizada la simulación con el escenario determinado por el Máster, debe calcular el error comparando el resultado con la propagación real y enviarlo al Máster. Al final, toda la carga que realiza el Worker es muy pesada y su tiempo de ejecución es considerable, entrando en conflicto con las duras restricciones de tiempo con las que se encuentran en caso de emergencia como se ha comentado anteriormente.

Para poder afrontar esas estrictas restricciones de tiempo es de gran interés reducir el tiempo de ejecución de los Workers. Con este objetivo se modifica el proceso del Worker para poder explotar los recursos que proporcionan las arquitecturas multicore. Ambos modelos han sido analizados para explotar el paralelismo multicore utilizando el paradigma de memoria compartida con OpenMP. De esta manera se puede paralelizar de manera bastante sencilla introduciendo directivas que indican cómo se realizara el paralelismo en la versión serie. Por lo tanto, el sistema total se transforma en una aplicación híbrida MPI-OpenMP basada en una estructura a nivel MPI Máster/Worker y con paralelismo multicore dentro de los modelos bajo OpenMP. En el siguiente apartado se muestra el análisis realizado para cada modelo y la escalabilidad obtenida.

III. Análisis de los modelos utilizados

La integración del modelo de generación de campos de vientos con el de propagación del fuego implica que cada Worker debe calcular varios campos de vientos antes de realizar la simulación de la propagación del fuego. Por consiguiente, el proceso que realiza el Worker se puede separar en tres partes, generación del campo de viento, simulación de la

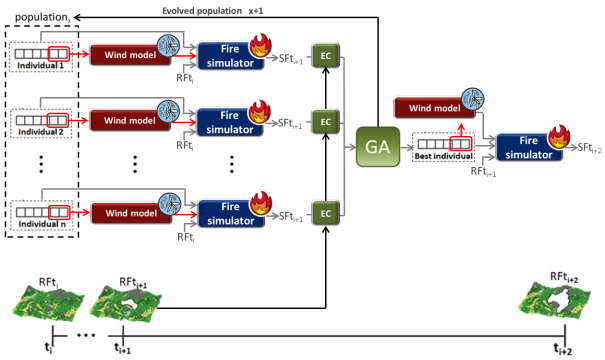

Fig. 3. Acoplado de modelo de campos de vientos y simulación de propagación de fuego

propagación del fuego y cálculo del error (ver Fig.4). Es necesario ejecutar cada Worker en el mínimo tiempo posible para alcanzar la mejor calibración posible y poder cumplir las restricciones de tiempo. En conclusión, es vital centrarse en la calidad de la predicción y en la reducción del tiempo del Worker en vez de en la eficiencia de éste. Se debe tener en cuenta que en una situación real una respuesta con buena eficiencia que no cumpla las restricciones de tiempo puede ser inútil.

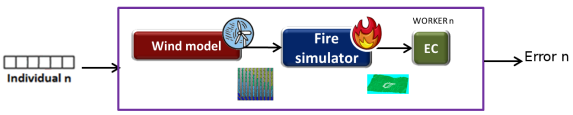

Fig. 4. Flujo de un Worker

El nodo utilizado para el análisis de escalabilidad ha sido un Dell Presicion T7500 con dos Intel XeonE5645 (6 núcleos por CPU) y 96GB de DDR3 SDRAM a 1333Mhz en 12 DIMMS.

A. Modelo de campo de vientos

La aplicación que se utiliza para el cálculo de los campos de vientos es WindNinja[13]. Este permite calcular la dirección y la velocidad del viento para cada celda de la topografía proporcionando un mapa de elevaciones y un viento meteorológico en un punto determinado del mapa. Existe una versión para Linux que utiliza OpenMP.

WindNinja ofrece dos maneras de explotar el paralelismo proporcionado por OpenMP. La primera manera de explotar el paralelismo utilizando WindNinja es proporcionar distintos vientos meteorológicos en una misma topografía. Por otro lado, la segunda manera se basa en explotar el paralelismo en el cálculo de un solo campo de viento. Esta segunda opción es la que se usa en el proceso del Worker dentro del esquema Máster/Worker para reducir el tiempo de ejecución del individuo del algoritmo genético. Esta segunda vía, presenta una escalabilidad limitada, pero aun así representa una mejora de tiempo. La Fig.5 muestra el tiempo de ejecución de WindNinja utilizando 1, 2, 4, 6 y 12 cores. Se debe destacar que el tiempo de ejecución de WindNinja utilizando un solo hilo de ejecución está cerca de los 300 segundos mientras que utilizando el paralelismo bajo OpenMP con 12 hilos de ejecución necesita menos de 150 segundos.

Se ha realizado un análisis de WindNinja para encontrar en que parte de la aplicación se utiliza más tiempo. Se ha encontrado que el programa gasta gran parte del tiempo realizando operaciones con matrices dispersas con compresión por filas en la resolución de métodos de cálculo numérico para resolución de sistemas lineales, en este caso el método SSOR (Symmetric Successive Overrelaxation Method) o el método Jacobi. El programa da preferencia al primero puesto que converge a la solución más deprisa y Jacobi no escala mucho mejor. Actualmente se está trabando en la optimización de dichos métodos.

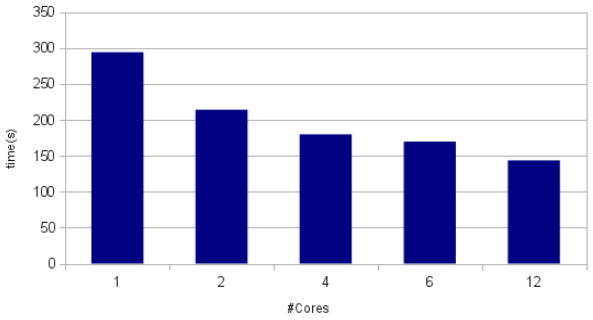

Fig. 5. Tiempo de ejecución de WindNinja

B. Simulador de propagación de incendios forestales

Por otro lado, el simulador de propagación de incendios forestales utilizado es Farsite[14] que se basa en el modelo de Rothermel[15][5]. La versión disponible para Linux es una versión no paralela. Una versión paralela del simulador de propagación de incendios forestales representaría una importante mejora reduciendo así el tiempo de ejecución total utilizando la capacidad multicore de los nodos.

Farsite ha sido analizado utilizando gprof[16] bajo distintas cargas con el fin de encontrar las funciones responsables de gran parte del tiempo de ejecución. Este análisis muestra que el tiempo de ejecución inclusivo está bastante distribuido por una gran cantidad de funciones excepto en el caso de dos funciones. La función CrossThread::Cross() consume sobre un 21% y la función GetPerimeterValue1() sobre un 14% de tiempo inclusivo respecto al total del tiempo de ejecución. Se podría intentar paralelizar estas funciones utilizando OpenMP pero con una mejora bastante limitada puesto que ya sabemos que tanto por ciento del tiempo de ejecución que se ha paralelizado. Por lo tanto, se ha realizado una búsqueda por el árbol de llamadas buscando una función con un bucle paralelizable utilizando OpenMP que incluya las llamadas a las dos funciones comentadas anteriormente. Como resultado de la búsqueda se encuentra una función, Intersecction::CrossCpare(), que tiene sobre un 60% de tiempo exclusivo y está situada más cerca de la raíz aunque en la misma rama que las dos

funciones comentadas que contienen un tiempo inclusivo considerable.

La mejora máxima orientativa a la que se puede llegar viene determinada por la proporción del tiempo de ejecución del programa que se paraleliza. Al saber que se paraleliza sobre un 60% del tiempo total de ejecución, la reducción máxima que podemos obtener es del 40% del tiempo total de ejecución. En la Fig.6 se puede observar como los resultados obtenidos están muy cerca de los que se preveían. Después de haber paralelizado Farsite, la herramienta de perfil OmpP[17], se ha utilizado para saber que parte del tiempo de ejecución total se realiza en serie o en paralelo. Se puede observar como la parte serie de Farsite es de 140 segundos y la parte paralelizada tarda sobre unos 220 segundos en un solo core. En dos cores la parte paralelizada se reduce a 120 segundos, con cuatro cores se reduce a 60 segundos y utilizando 12 cores se queda en menos de 20 segundos. El tiempo total de ejecución se reduce de 360 segundos a menos de 160 segundos. Este resultado es muy bueno sabiendo el grado de paralelización del programa. La mejora de la parte paralela se puede observar en la Fig.7.

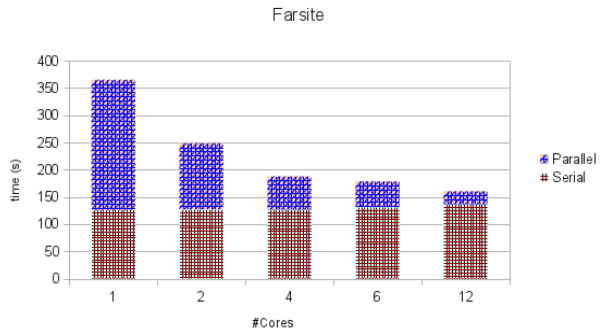

Fig. 6. Tiempo de ejecución serie y paralelo de Fariste

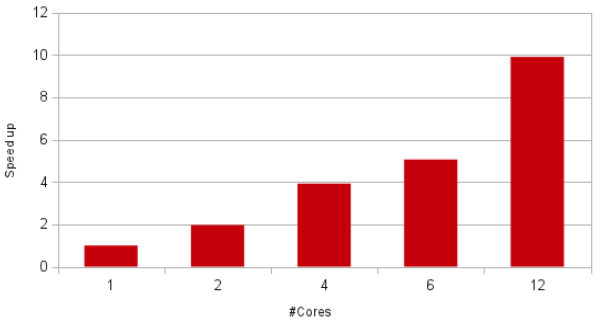

Fig. 7. Speed up del tiempo de ejecución de la parte paralela de Farsite

IV. Tiempo de ejecución de un Worker

Se han analizado las distintas partes del flujo de ejecución del Worker excepto el cálculo del error. El tiempo utilizado para el cómputo de este error es despreciable comparado con el resto del tiempo de ejecución del Worker. Como se puede observar en le Fig.8, el tiempo total de ejecución se ha reducido de 650 segundos a 300 segundos. Las cargas de trabajo utilizadas en la etapa de obtención de los perfiles de las aplicaciones se basan en casos reales (sobre las 3013hm$_2$), se usan mapas de alta resolución, simulaciones de propagación con frentes cada 1h, y una duración de incendio de varios días.

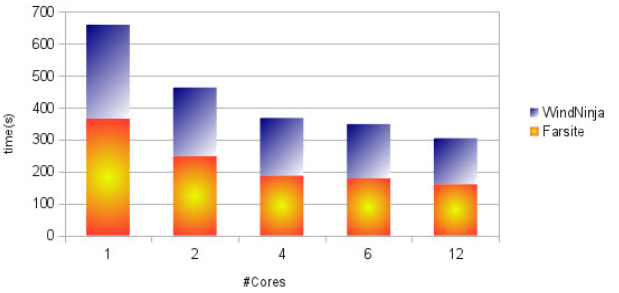

Fig. 8. Tiempo de ejecución del Worker

En la Fig.8 se puede observar que podría llegar a ser más rápido ejecutar dos individuos distintos del algoritmo genético en dos núcleos distintos en lugar de ejecutar dos ejecuciones secuencialmente donde cada una de ellas utiliza dos núcleos. Es decir, un individuo ejecutado en dos cores utiliza algo más 460, que ejecutadas secuencialmente tardaría sobre 900 segundos. Mientras, que dos ejecuciones serie ejecutadas de manera concurrente en dos cores separados tardarían 650 cores. Sin embargo, en el caso de utilizar más recursos que individuos la paralización representa una gran mejora.

Se ha realizado una prueba de la estructura híbrida MPI-OpenMP bajo un cluster formado por nodos IBM X3550 Nodes con dos Dual-Core Intel(R) Xeon(R) 5160 a 3.00GHz con 4MB L2 (2x2) y 12 GB DIMM 667 MHz. Se ha lanzado una ejecución basada en mapas reales con 10 nodos. Se ha obtenido el tiempo de una generación en la que se ejecutan 9 individuos en 9 nodos. Los resultados obtenidos se pueden ver la tabla I.

TABLA I

Mejora respecto a 9 nodos con un core por nodo

Cores/Nodo.	tiempo(s)	Reducción(%)
2	581	31
4	449	46

Se debe tener en cuenta que en el algoritmo genético el hecho de poder realizar más generaciones en el mismo tiempo permite converger a una solución más cercana a la óptima mientras que aumentar la cantidad de individuos permite evitar los mínimos locales.

V. Conclusión

En el contexto de los fuegos forestales, cuando un fuego empieza, cada segundo cuenta. Utilizando un algoritmo genético con el paradigma Máster/Worker implementado con MPI se puede obtener unos buenos resultados de predicción, gracias al calibrado de parámetros de los que no se dispone en casos

reales de emergencia. Como el método de calibración tiene un alto coste de cómputo y, además, los resultados de la predicción deben proporcionarse bajo duras restricciones temporales. Actualmente, las arquitecturas multicore pueden representar una gran mejora en aplicaciones existentes diseñadas para ejecutarse en serie, simplemente, añadiendo directivas OpenMP en lugares estratégicos. Las herramientas de obtención de perfiles de aplicaciones son una utilidad clave para saber localizar los lugares estratégicos en programas cuya magnitud de cantidad de líneas es considerable.

La manera que se presenta de optimización del método de calibrado está basado en añadir paralelismo multicore en los procesos Workers utilizando OpenMP. Además, el tiempo que necesita cada Worker puede ser reducido significativamente. Al poder reducir el tiempo de ejecución de los Workers, podemos reducir el Worker que más tarda en una ejecución de una generación del algoritmo genético, y por lo tanto, reducir el tiempo que se tarda por generación y en definitiva todo el tiempo genético y la etapa de calibración. Esta mejora también permite realizar más iteraciones o generaciones en el algoritmo genético en el mismo tiempo, reduciendo así el error de calibración o mejorando los resultados en el caso particular que el algoritmo tenga más dificultad para converger a una solución.

Agradecimientos

El presente trabajo ha sido financiado por el MICINN-Spain bajo contrato TIN2011-28689-C02-01.

Referencias

[1] P.D Bates and A.P.J De Roo, "A simple raster-based model for flood inundation simulation," *Journal of Hydrology*, vol. 236, no. 1–2, pp. 54–77, 2000.

[2] V. V. Titov, *Implementation and testing of the method of splitting tsunami (MOST) model*, 1997.

[3] P. J. Vickery, *HAZUS-MH hurricane model methodology. I: hurricane hazard, terrain, and wind load modeling.*

[4] D.X. Viegas, LM Ribeiro, MT Viegas, LP Pita, and C. Rossa, "Impacts of fire on society: Extreme fire propagation issues," *Earth Observation of Wildland Fires in Mediterranean Ecosystems*, p. 97, 2009.

[5] R.C. Rothermel, *How to predict the spread and intensity of forest and range fires*, Intermountain Forest and Range Experiment Station Ogden, 1983.

[6] F.A. Albini, Intermountain Forest, Utah) Range Experiment Station (Ogden, and United States. Forest Service, *Estimating wildfire behavior and effects*, General technical report INT. Dept. of Agriculture, Forest Service, Intermountain Forest and Range Experiment Station, 1976.

[7] Baker Abdalhaq, Ana Cortés, Tomàs Margalef, and Emilio Luque, "Enhancing wildland fire prediction on cluster systems applying evolutionary optimization techniques," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 61–67, 2005.

[8] M. Denham, A. Cortés, and T. Margalef, "Computational steering strategy to calibrate input variables in a dynamic data driven genetic algorithm for forest fire spread prediction," *Computational Science–ICCS 2009*, pp. 479–488, 2009.

[9] G. Bianchini, M. Denham, A. Cortés, T. Margalef, and E. Luque, "Wildland fire growth prediction method based on multiple overlapping solution," *Journal of Computational Science*, vol. 1, no. 4, pp. 229–237, 2010.

[10] William Gropp, "Mpich2: A new start for mpi implementations," in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Dieter Kranzlmüller, Jens Volkert, Peter Kacsuk, and Jack Dongarra, Eds., vol. 2474 of *Lecture Notes in Computer Science*, pp. 37–42. Springer Berlin / Heidelberg, 2002.

[11] B. Chapman, G. Jost, and R. Van Der Pas, *Using OpenMP: portable shared memory parallel programming*, vol. 10, The MIT Press, 2007.

[12] Mónica Denham, Ana Cortés, Tomàs Margalef, and Emilio Luque, "Applying a dynamic data driven genetic algorithm to improve forest fire spread prediction," in *Computational Science – ICCS 2008*, Marian Bubak, Geert van Albada, Jack Dongarra, and Peter Sloot, Eds., vol. 5103 of *Lecture Notes in Computer Science*, pp. 36–45. Springer Berlin / Heidelberg, 2008, 10.1007/978-3-540-69389-5_6.

[13] Brian Potter and Bret Butler, "Using wind models to more effectively manage wildfire," *Managementtoday*, p. 40, 2009.

[14] M. A. Finney, *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. 1998.

[15] R C Rothermel, "A mathematical model for predicting fire spread in wildland fuels," *Director*, , no. INT-115, pp. 40, 1972.

[16] Susan L. Graham, Peter B. Kessler, and Marshall K. McKusick, "gprof: a call graph execution profiler," *SIGPLAN Not.*, vol. 39, no. 4, pp. 49–57, Apr. 2004.

[17] K. Fürlinger and M. Gerndt, ": A profiling tool for openmp," *OpenMP Shared Memory Parallel Programming*, pp. 15–23, 2008.