Optimització de la simulació de camps de vents

Núria Navarro Juliana

Resum—El vent és un fenomen ambiental natural que pot afavorir positivament en molts àmbits, però a la vegada pot provocar catàstrofes en les quals es veuen perjudicades tant la naturalesa com l'ésser humà. Per aquest motiu la simulació dels camps de vents està prenent més força, ja que aquesta predicció del comportament del vent en funció del terreny, la vegetació, la direcció i velocitat del mateix pot ser decisiva a l'hora de prevenir l'extensió incontrolada d'incendis forestals. WindNinja és un programa que permet realitzar aquesta simulació. Aquest projecte tracta de l'optimització de les execucions per a obtenir la simulació, així com, l'estudi dels diferents paràmetres d'entrada que poden influir en el rendiment.

Paraules clau—*Simulació de camps de vents, Incendis forestals, WindNinja, Multiprocessing, Optimització*

Abstract—Wind is a natural environmental phenomenon that can have a positive effect on many areas, but at the same time, it can cause catastrophes in which both nature and humans are harmed. For this reason, the simulation of wind field is gaining momentum, as this prediction of wind behaviors based on terrain, vegetation, wind direction and speed can be crucial in preventing the uncontrolled spread of forest fires. WindNinja is a program that allows this simulation. This project deals with the optimization of the executions to obtain the simulation, as well as the study of the different input parameters that can influence the performance.

Index Terms—*Wind field simulation, Forest Fires, WindNinja, Multiprocessing, Optimization*

1 Introducció - Context del treball

EL vent és un fenomen ambiental que es produeix davant la diferència de pressió de l'aire, fet que provoca el seu moviment. Aquest moviment és degut a la diferència de pressió esmentada, fent que l'aire sigui accelerat des d'una pressió més gran a una de més petita [1], [2].

Aquest fenomen pot presentar avantatges com la dispersió de llavors de certes plantes així com la seva expansió sobre més terreny, afavorint la possibilitat de supervivència de l'espècie. Pot modificar el relleu gràcies a la formació de sòls o bé al contrari, provocar l'erosió d'aquests. També pot ser molt perjudicial en altres situacions, com és el cas dels incendis forestals, ja que ajuda a la seva expansió i en dificulta poder prendre el control i reduir l'impacte d'aquests.

Tal com diu Eduardo Rodríguez [3], els camps de vents proporcionen la possibilitat d'estudiar el comportament del vent en cada punt del terreny, per aquest motiu s'està convertint en una eina molt important, ja que permet resoldre problemes quotidians que fins ara tenien una complexitat de resolució molt més elevada.

2 Objectius

L'objectiu principal del treball és aconseguir optimitzar el màxim possible la simulació dels camps de vents que es realitza mitjançant WindNinja a partir del mapa del terreny a estudiar.

D'acord amb WindNinja [4], aquest és un programari informàtic que s'encarrega de la simulació de camps de vents principalment per a aplicacions d'incendis forestals, tot i que també es pot utilitzar per a qualsevol altre tipus d'aplicació que requereixi la predicció del vent. Com que l'objectiu inicial de WindNinja va dirigit a l'estudi dels incendis forestals, que és un servei d'emergència, es tracta d'una aplicació pensada perquè la seva execució sigui ràpida sense la necessitat de disposar grans requeriments quant a CPU [5].

Aquesta predicció que realitza el programa pot ser de gran ajuda en les situacions esmentades anteriorment, com ara per facilitar el control dels incendis forestals i poder tenir una possible predicció de com evolucionarà, per tal de poder prendre decisions d'estratègia en funció d'aquesta predicció. O bé, en altres situacions com ara en l'aviació és molt important saber la direcció i la velocitat del vent en cada moment, així com saber com afecta aquest en el conjunt del terreny, sobretot en els moments més crítics com ara l'enlairament i aterratge, o bé, en la navegació. Aquesta millora permetrà obtenir models més precisos del camp de vents amb un període de temps menor.

A més a més de l'optimització de la simulació de camps de vents realitzada per WindNinja, també es tractarà d'investigar si diferents configuracions d'entrada condicionen el rendiment del programari.

- *E-mail de contacte: nuria.navarroj@gmail.com*
- *Menció realitzada: Enginyeria de Computadors*
- *Treball tutoritzat per: Tomàs Manuel Margalef Burrull (Departament d'Arquitectura de Computadors i Sistemes Operatius)*
- *Curs 2021/22*

3 Estat de l'Art

Tal com defineix Protecció Civil [6], els incendis forestals són aquells que s'estenen per terreny que no estava destinat a ser cremat sense cap mena de control.

Tot i que el foc és un fenomen natural necessari, ja que ajuda a la regeneració de boscos, deixa de ser-ho en el moment en què és l'ésser humà qui els provoca convertint aquest element en una de les majors amenaces contra els ecosistemes i la població. A Espanya més del 96% dels incendis forestals són provocats per l'ésser humà [7]. Les principals causes humanes d'incendis forestals són focs iniciats intencionadament, negligència o descuits, com podria ser el llançament inadequat de burilles i residus. Així com, accidents tot i haver pres totes les mesures de prevenció necessàries [8].

Davant d'un incendi forestal hi ha diversos factors que influeixen en el comportament i l'expansió del foc, un dels que més afecten és el vent. L'aportació de WindNinja a la reducció de l'impacte d'aquests incendis és que permet fer una simulació del vent en tota classe de terreny, tenint en compte els desnivells que provoquen canvis en la direcció i velocitat del vent [9].

4 Metodologia

El primer pas a seguir ha estat l'estudi del rendiment i l'escalabilitat de WindNinja, ja que aquest està paral·lelitzat mitjançant OpenMP i, per tant, s'ha observat com varia el rendiment en funció de la mida del mapa, així com, el nombre de threads amb els quals es realitza l'execució.

Per a l'optimització de l'execució, la metodologia emprada ha estat la partició del fitxer d'entrada que conté informació del terreny del qual es vol simular el camp de vents. L'objectiu de realitzar aquesta partició és per tal de poder executar els diferents mapes generats de forma paral·lela mitjançant Python Multiprocessing, ja que tracta de paral·lelitzar el codi a nivell de processos i proporciona l'habilitat d'executar el programa de manera concurrent.

Per altra banda, per a l'estudi de les diferents configuracions, s'han tingut en compte els dos factors més importants per a la simulació del camp de vents, que són la direcció i la velocitat del vent.

5 Ecalabilitat de WindNinja

Com s'ha comentat anteriorment, el pas previ al desenvolupament és l'anàlisi del rendiment del programa.

Per a realitzar aquest estudi s'ha de tenir en compte els recursos que es disposen al dispositiu on s'han realitzat les execucions. En aquest cas, es disposava d'un processador de 2,3 GHz amb una memòria de 8 GB. I, per tant, els resultats es veuen influenciats per aquestes limitacions tant per la freqüència del processador, com per la capacitat de memòria.

S'han hagut de generar diversos mapes de mides diferents per poder veure com escala el programa a mesura que ha de realitzar els càlculs amb més dades. Els mapes generats han estat mapes quadrats de 100x100 cel·les, 200x200, 400x400, 800x800 i 1200x1200.

Com s'observa a la figura (1), quan la configuració d'execució és sense conservació del moment el programa escala correctament fins a una mida de mapa de 400x400 (160.000 cel·les), en canvi, quan el mapa és de 800x800 (640.000 cel·les) ja no escala de la manera esperada, això és degut a les limitacions de memòria, utilitzant entre un 95 i 99% d'aquesta.

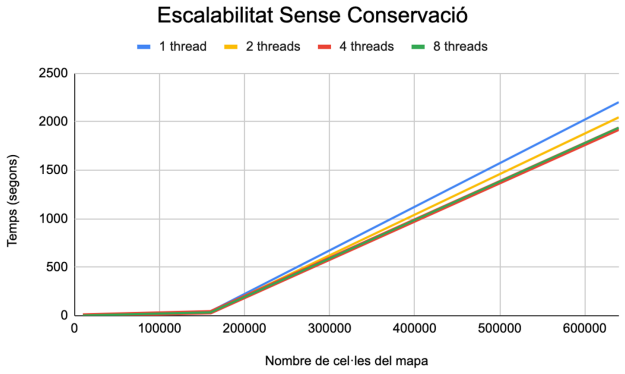

Fig. 1: Escalabilitat de WindNinja sense conservació del moment

En canvi, quan es realitza l'execució amb conservació del moment, el programa escala correctament fins a mapes de mida 800x800, com s'observa a la figura (2), on el rendiment ja comença a estar limitat per la memòria, consumint aproximadament un 90%. A causa d'aquesta limitació l'escalabilitat va empitjorant a mesura que la mida del mapa augmenta, fent que amb mapes de 1200x1200 (1.440.000 cel·les) arribi a utilitzar tota la memòria disponible impossibilitant l'execució de mapes amb més dades.

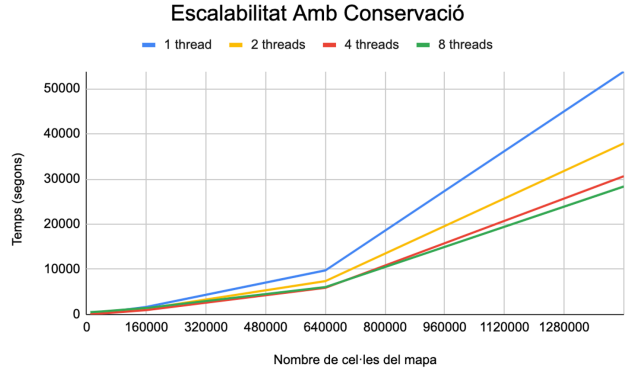

Fig. 2: Escalabilitat de WindNinja amb conservació del moment

A les següents figures, (3) i (4), es mostren les gràfiques anteriors ampliades per a poder observar amb més detall com el rendiment del programa és lineal a mesura que s'augmenta la mida del mapa, però deixa de ser-ho en el moment en el qual el programa es troba limitat per memòria, fent que presenti un impacte negatiu sobre el rendiment de la simulació de camps de vents.

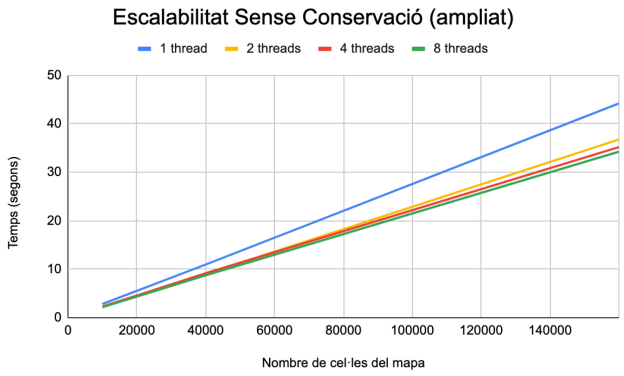

Fig. 3: Escalabilitat de WindNinja sense conservació del moment ampliat

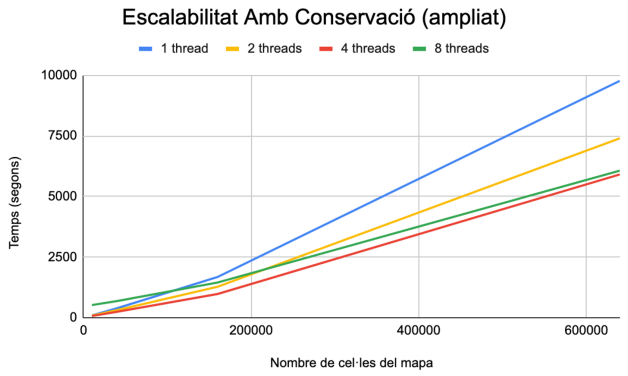

Fig. 4: Escalabilitat de WindNinja amb conservació del moment ampliat

6 DESENVOLUPAMENT

El programa rep set paràmetres d'entrada per defecte. El primer de tots és el tipus de *solver*, en aquest cas, hi ha dues opcions: una amb conservació de massa i del moment i l'altra és únicament amb conservació de massa, que ens referirem a aquest tipus d'execució com a execucions sense conservació del moment. També rep el mapa del qual es vol realitzar la simulació del camp de vents, que conté les dades d'elevació de la superfície a estudiar. La velocitat en mph i la direcció del vent en graus, la vegetació que predomina en el terreny, la resolució del mapa i, per últim, el nombre de threads amb els quals es vol executar el programa. Per a les execucions s'han establert uns valors per defecte a aquests paràmetres. La velocitat ha estat 10 mph, amb una direcció de 270 graus, una vegetació en la qual predominen els arbres i una resolució de 30. La resta de paràmetres han estat els que s'han modificat per veure el rendiment del programa.

6.1 Partició fitxer i execució amb Python Multiprocessing

La primera metodologia que s'ha aplicat és la divisió del mapa original en mapes més petits per tal d'executar aquests de manera paral·lela mitjançant Python Multiprocessing. El nombre de sub-mapes generats dependrà del nombre total de threads amb els quals volem que s'executi l'aplicació. Cada nou fitxer generat serà executat per un únic thread.

Per implementar aquesta metodologia, primerament s'ha hagut de convertir el mapa del qual volem aconseguir el camp de vents, de format TIF a ASCII mitjançant QGIS per poder-lo llegir i dividir en nous mapes.

A la figura (5) es pot observar el diagrama de flux de l'execució del mètode implementat. Com veiem, el primer pas és obtenir la mida dels nous mapes per a, posteriorment, poder dividir el fitxer. Un cop tenim els sub-mapes creats, es creen tants threads com sub-mapes mitjançant Python Multiprocessing, executant WindNinja per cada sub-mapa amb un únic thread. Un cop ha finalitzat l'execució s'han d'unir els fitxers resultants, tant el dels angles com el de la velocitat, en un d'únic per a obtenir un únic fitxer de resultats respectivament.

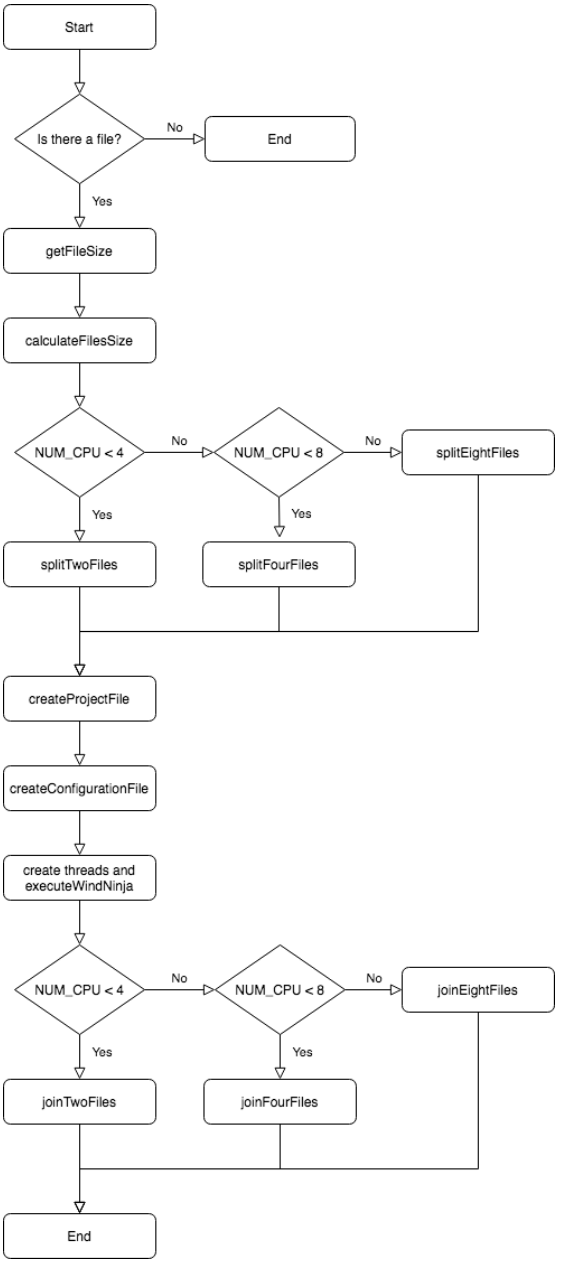

Fig. 5: Diagrama de flux del programa per a la partició del mapa original

Entrant en detall a la implementació, un cop tenim el mapa en el format correcte, necessitem saber quantes files i columnes té per poder calcular la mida dels nous mapes en funció de la mida del mapa actual i del nombre de threads amb els quals es vol executar el programa. Per a fer-ho, s'utilitza el mètode *getFileSize* que s'encarrega d'obrir el mapa en format ASCII per llegir les primeres sis línies que contenen la informació del fitxer i les guarda en una llista que serà utilitzada posteriorment. Per obtenir el nombre de columnes i files, necessitem el contingut de les dues primeres línies respectivament.

A continuació, amb aquestes dades, es procedeix a calcular la mida de cada sub-mapa en funció del nombre de fitxers que es vulguin generar, a la figura (6) s'observen les diferents maneres de dividir el mapa. En el cas que es vulgui executar el programa en dos threads, els sub-mapes creats seran a partir de la divisió del mapa original únicament en funció de les files. Pel que fa a la divisió en vuit fitxers, es realitza una partició a nivell de columnes (dos blocs), i tres a nivell de files (quatre blocs). El mètode que s'encarrega d'aquest càlcul és *calculateFilesSize*. Primer de tot, es calcula en quants blocs es dividiran les files i les columnes. Un cop sabem el nombre de divisions, calculem el nombre de files i el nombre de columnes que tindrà cada sub-mapa.

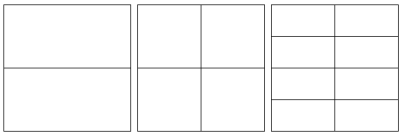

Fig. 6: Possibles divisions del mapa en funció del nombre de sub-mapes desitjats

Posteriorment, es procedeix a llegir el contingut de cada fila i columna per anar creant els fitxers corresponents amb les dades pertinents, tenint en compte que s'afegiran 20 files i columnes extres a cada nou mapa, com s'observa a les figures (7), (8) i (9), per afegir precisió i minimitzar l'error del càlcul del camp de vents.

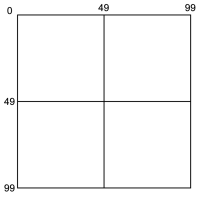

Fig. 7: Mapa original de mida 100x100 indicant les particions amb 4 sub-mapes

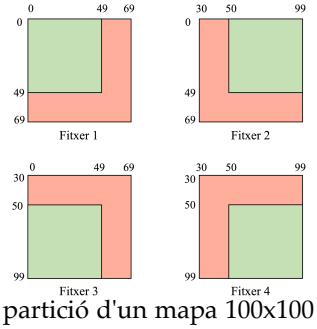

Fig. 8: Esquema partició d'un mapa 100x100 en 4 sub-mapes, amb les files i columnes extres

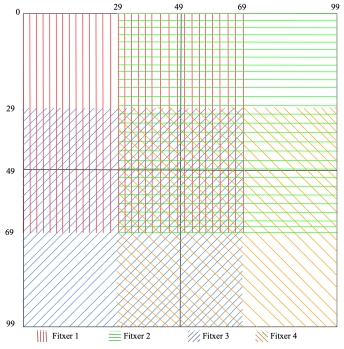

Fig. 9: Esquema solapament de divisió d'un fitxer de 100x100 en 4 sub-mapes

Els mètodes *splitTwoFiles*, *splitFourFiles* i *splitEightFiles* són els encarregats de realitzar aquesta funcionalitat, es cridarà únicament a un d'ells en funció del nombre de particions que es desitgin. El primer pas a dur a terme és crear els fitxers i escriure en ells les primeres sis línies que contenen informació, com per exemple, el nombre de files i columnes que tindrà cada sub-mapa. Seguidament, es llegeix cada cel·la del mapa original i en funció de la posició d'aquesta es calcula a quins sub-mapes s'haurà d'escriure el seu contingut.

Quan ja es tenen els sub-mapes creats s'han d'executar aquests paral·lelament mitjançant Python Multiprocessing, gràcies al mètode *executeWindNinja*; però, prèviament s'ha de crear un fitxer de configuració de l'execució per a cada nou mapa, *createConfigurationFile*, en el que s'especifica el nombre de threads, quin sub-mapa ha d'executar i el tipus d'execució, és a dir, si és amb conservació del moment o no. Així com, també s'ha de crear el fitxer de projecció del mapa, *createProjectFile*, que serà una còpia del fitxer de projecció del mapa original.

Finalment, un cop acabada l'execució dels diferents mapes, s'han d'unir els fitxers resultants en un d'únic, tant pels angles com pel fitxer de la velocitat, *joinTwoFiles*, *joinFourFiles* o *joinEightFiles*. A l'hora d'unir els diferents fitxers resultants, s'ha de tenir en compte dues situacions, la primera és que aquests fitxers contenen l'última fila i columna duplicada. Per altra banda, s'han de gestionar aquelles files i columnes que estan replicades en diversos fitxers. Per implementar aquesta funcionalitat, es comprova per cada cel·la si forma part de les vint cel·les extres per afegir precisió o no, si no és una cel·la extra es copia el contingut al fitxer final, en el cas contrari, és a dir, que es tracti d'una cel·la extra, s'ignora el seu contingut.

6.2 Ubuntu

Per a implementar la següent metodologia ha sigut necessària la instal·lació del programari WindNinja [10] a Ubuntu 16.04.

Un cop instal·lat correctament el programari, s'ha analitzat el codi mitjançant les eines *perf record* i *perf report*. A la secció (7.2) s'analitzaran els resultats obtinguts d'aquest anàlisi.

En aquest cas, no s'han pogut realitzar proves amb conservació del moment a causa d'errors durant la instal·lació d'OpenFOAM, que és un software gratuït utilitzat per a

l'execució dels càlculs necessaris per a les execucions amb conservació del moment.

Com que el codi font ja està optimitzat amb OpenMP, no s'ha pogut millorar el rendiment a partir d'aquest, per tant, no s'ha arribat a aconseguir una millora respecte al temps d'execució.

7 Resultats

Com s'ha mencionat anteriorment, les execucions s'han dut a terme amb certes limitacions quant a recursos, fet que ha provocat que les proves es veiessin limitades a aquests, sobretot a nivell de memòria.

7.1 Partició fitxer i execució amb Python Multiprocessing

A les següents figures es mostra l'SpeedUp del temps d'execució del programa executat amb Python Multiprocessing respecte a la versió d'OpenMP en seqüencial. En funció del mapa i el nombre de sub-mapes en el que es divideix el mapa original, és a dir, el nombre de threads amb els quals s'executa WindNinja. Les execucions han estat realitzades amb la configuració predeterminada, variant el tipus d'execució sense conservació i amb conservació del moment respectivament.

Com podem veure a la figura (10), quan l'execució es realitza sense conservació del moment, la implementació realitzada amb Python Multiprocessing no presenta millora fins que els mapes són més grans, amb mapes de 200x200 ja es comença a apreciar una mica de millora, tot i que no és significativa, ja que el millor SpeedUp que s'aconsegueix és amb 4 threads amb només una millora de 1,65x. Però, en canvi, quan s'executa amb 2 i 8 threads la millora no arriba a 1,5x. Això és degut al fet que s'inverteix més temps realitzant la divisió del mapa i la creació de threads que no pas millora que s'obté. En canvi, quan els fitxers són més grans la implementació de Pyhton Multiprocessing sí que presenta una millora al temps d'execució total, ja que en aquest cas, la millora que s'obté és superior al temps que s'inverteix en la divisió del mapa i la creació de threads.

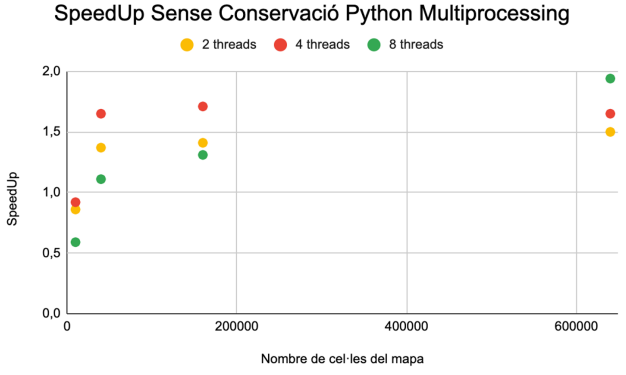

Fig. 10: SpeedUp amb tipus d'execució sense conservació del moment

Per altra banda, quan l'execució és amb conservació del moment, figura (11), sí que presenta millora inclús quan els mapes són petits. Ja que s'ha de realitzar més còmput en aquest tipus d'execució i permet reduir aquest temps en executar-se fitxers més petits de manera paral·lela. Arribant fins a un SpeedUp d'aproximadament 4,5 i 4,1 amb execucions d'un mapa de 800x800 amb 4 i 8 threads respectivament.

Amb mapes de 1200x1200 cel·les l'SpeedUp aconseguit amb aquesta versió disminueix respecte al mapa de 800 a causa de la limitació quant a la memòria.

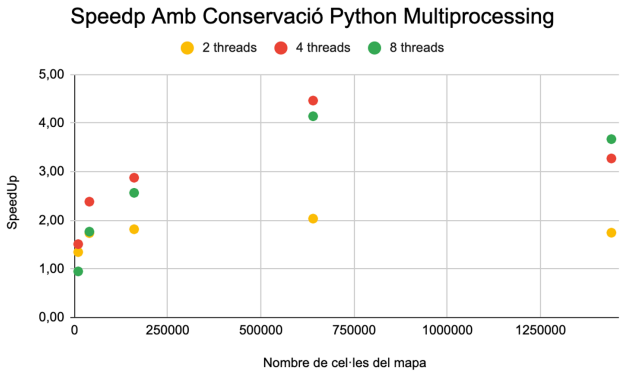

Fig. 11: SpeedUp amb tipus d'execució amb conservació del moment

Per comprovar que la implementació és correcta s'han creat mapes de 200x400 cel·les, 200 files i 400 columnes, i de 400x800. D'aquesta manera podem observar si realment s'estan executant els mapes de manera paral·lela. Per a un correcte funcionament, els resultats de les execucions de mapes de mida 400 i 800 amb 2 threads amb Python Multiprocessing, hauria de ser similar al temps d'execució de WindNinja necessari per a executar un mapa de 200x400 i 400x800 seqüencialment. Com veiem a les taules (1) i (2), les execucions realitzades seqüencialment amb mapes de 200x400 i 400x800 i les executades paral·lelament amb Python Multiprocessing presenten un rendiment molt similar, varia entre 1,1 i 1,4x. Per tant, es confirma que la versió de Python Multiprocessing està executant paral·lelament els sub-mapes que crea. Tot i que, quan l'execució és sense conservació del moment, veiem que el rendiment d'un mapa de 800 amb 2 threads amb Ptyhon Multiprocessing és 2,4x pitjor, això és degut a la limitació de memòria comentada anteriorment.

TAULA 1. Temps d'execució en funció de la mida del mapa i nombre de particions sense conservació del moment. On OMP és la versió d'OpenMP i PM de Python Multiprocessing.

Versió	Num thr.	Mida Mapa			
		200x400	400	400x800	800
OMP	1 th	22,60	-	604,23	-
PM	2 th	-	31,42	-	1467,72

A més de l'estudi del temps d'execució i l'SpeedUp, també s'ha fet una comparació dels resultats, és a dir, s'ha calculat l'error produït amb la nova implementació.

TAULA 2. Temps d'execució en funció de la mida del mapa i nombre de particions amb conservació del moment. On OMP és la versió d'OpenMP i PM de Python Multiprocessing.

Versió	Num thr.	Mida Mapa			
		200x400	400	400x800	800
OMP	1 th	768,18	-	4239,46	-
PM	2 th	-	924,6	-	4803,09

A continuació, les següents figures mostren l'error tant en la direcció del vent com en la velocitat que hi ha en cada punt d'un mapa de 400x400 executat paral·lelament amb 4 threads.

Com es pot observar a la figura (12), quan l'execució és sense conservació del moment el resultat de la direcció del vent no s'aprecia la partició del mapa, però per contres, al mapa resultant de l'error de la velocitat sí que s'observa les quatre particions.

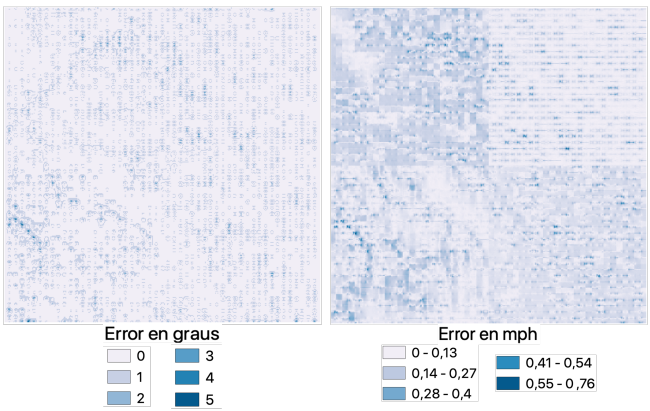

Fig. 12: Mapes d'error de la direcció i velocitat d'un mapa 400x400 d'una execució sense conservació amb 4 threads

En el cas de l'execució amb conservació del moment s'observa la partició realitzada en els dos mapes resultants, tant en la direcció com en la velocitat, figura (13). Aquest fet és degut a l'error de càlcul que es produeix als valors fronteres, ja que tot i que s'han afegit 20 cel·les extres a cada nou fitxer generat continuen sent les zones amb menys precisió.

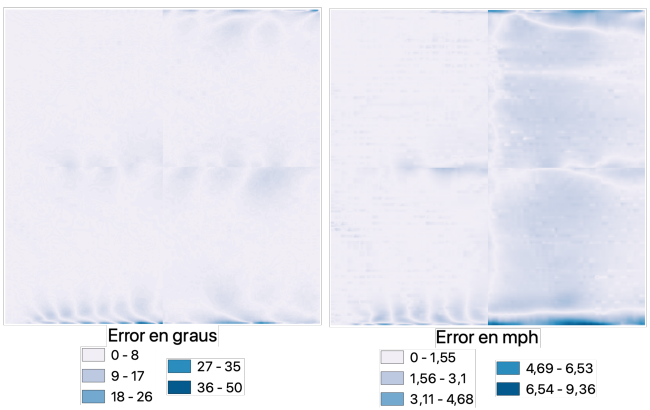

Fig. 13: Mapes d'error de la direcció i velocitat d'un mapa 400x400 d'una execució amb conservació amb 4 threads

Un altre punt a destacar respecte al càlcul de l'error és que quan l'execució es realitza amb conservació del moment, l'error produït a cada punt del mapa és més elevat.

Respecte al resultat de la direcció del vent sense conservació l'error oscil·la entre 0 i 5, en canvi, amb conservació arriba a una diferència de fins a 50 graus. El mateix succeeix amb la velocitat, el primer l'error no és superior a 1, i el segon arriba a haver-hi un error aproximadament de 9,4.

A les taules (3) i (4) es mostra l'RMSE (*Root-Mean-Square Error*) de l'execució d'un mapa de 400x400 cel·les amb 4 threads sense conservació i amb conservació respectivament, on es comprova el comentat anteriorment, quan l'execució és amb conservació del moment l'error és més elevat.

TAULA 3. RMSE per un mapa de 400x400, direcció 270, velocitat 10 mph, sense conservació del moment.

Particions	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
2	0,07	0,04	0,06	0,44
4	0,09	0,05	0,08	0,45
8	0,11	0,05	0,09	0,35

TAULA 4. RMSE per un mapa de 400x400, direcció 270, velocitat 10 mph, amb conservació del moment.

Particions	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
2	0,37	0,24	0,20	1,89
4	0,88	0,54	0,41	1,96
8	0,95	0,56	0,43	2,63

Un altre factor que es visualitza a les taules és que a major nombre de particions més elevat és l'error. Això és degut a la pèrdua de precisió per cada sub-mapa generat, com s'ha mencionat, tot i afegir 20 línies de solapament no són les necessàries perquè no es vegi reflectit en la precisió.

7.2 Ubuntu

Un cop instal·lat WindNinja a Ubuntu s'han realitzat les mateixes execucions que a Windows per veure si hi havia influència en el temps d'execució. Hi ha dos punts importants a destacar pel fet que sigui un programa precompilat.

El primer de tots és que necessita més memòria per a executar els mapes, per tant, no s'han pogut realitzar execucions amb mapes superiors a una mida de 400x400 cel·les. A la figura (14) es mostra com el rendiment, igual que amb el sistema operatiu Windows, és lineal. Per contres, en aquest cas no s'arriba a apreciar com el programa deixa d'escalar correctament a mesura que s'augmenta la mida del mapa a estudiar. Aquest fet és degut al fet que el mapa més gran que s'ha pogut executar encara no es veu limitat per memòria, sinó que és en les pròximes mides on es dóna l'afectació.

L'altre factor que es veu influït és el temps d'execució, ja que amb els mateixos valors, el temps és més elevat que les execucions realitzades a Windows. Com veiem a les figures (15) i (16), l'execució a Ubuntu empitjora entre 0,4 i 0,68 vegades. A mesura que el mapa executat és més gran el temps d'execució total empitjora a causa del fet que hi ha més restricció dels recursos a nivell de memòria.

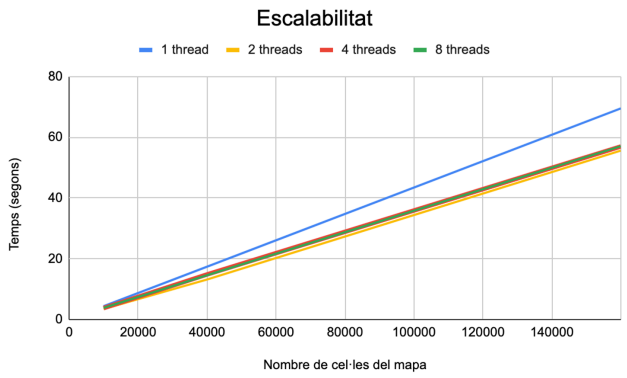

Fig. 14: Escalabilitat de WindNinja sense conservació del moment en Ubuntu

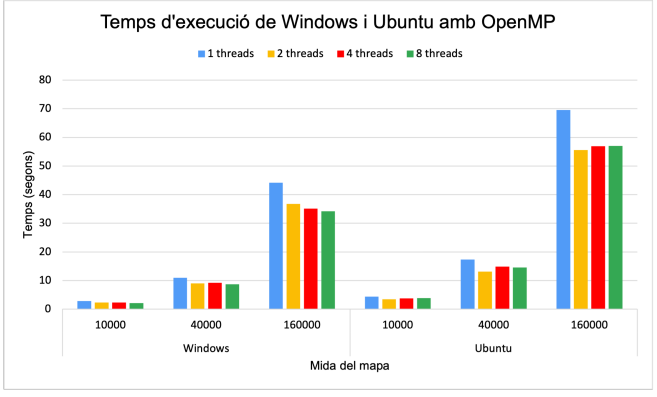

Fig. 15: Temps d'execució en segons de Windows respecte a Ubuntu amb la versió d'OpenMP

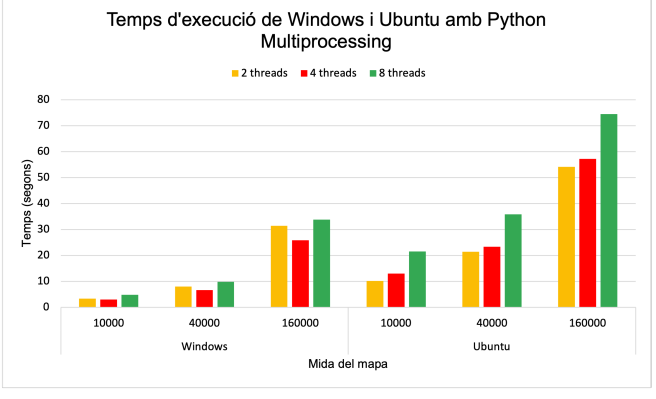

Fig. 16: Temps d'execució en segons de Windows respecte a Ubuntu amb la versió Python Multiprocessing

Com s'ha comentat a l'apartat (6.2), s'ha analitzat el rendiment del programa mitjançant *perf record*. A la figura (17) es mostra una imatge del perfil de rendiment d'una execució amb un mapa de 400x400 cel·les, sense conservació del moment i amb 4 threads. A l'apèndix (A.2) es mostra amb més detall el resultat de l'anàlisi a la figura (21).

```
24.03%  WindNinja_cli  libninja.so          [.] _ZN14Preconditioner10mkl_dcsrsv
14.06%  WindNinja_cli  libninja.so          [.] _ZN5ninja10mkl_dcsrmvEPcPiS1_Pd
11.61%  WindNinja_cli  libninja.so          [.] _ZN5ninja10mkl_dcsrmvEPcPiS1_Pd
```

Fig. 17: Resultat *perf record* d'un mapa 400x400 amb 4 threads i sense conservació del moment

Observem que la classe que utilitza més recursos és *Preconditioner*, consumint aproximadament un 24%. D'aquesta classe concretament el mètode que consumeix més recursos és *mkl_dcsrsv*. Com s'ha comentat anteriorment, tot i saber la classe i el mètode que està consumint més recursos, no s'ha aconseguit optimitzar-lo.

Tot i això, s'ha analitzat el codi. Aquesta classe té dos precondicionadors, Jacobi i SSOR, en el que s'ha observat que quan s'executa amb el precondicionador SSOR amb OpenMP els temps d'execució són aproximadament 1,2 i 1,4 vegades millor que amb Jacobi, els valors més alts d'SpeedUp són d'aquelles execucions realitzades amb mapes petits, de 100x100 cel·les i, en canvi, quan els fitxers contenen més dades l'SpeedUp empitjora, aconseguint una millora únicament de 1,2x, figura (18).

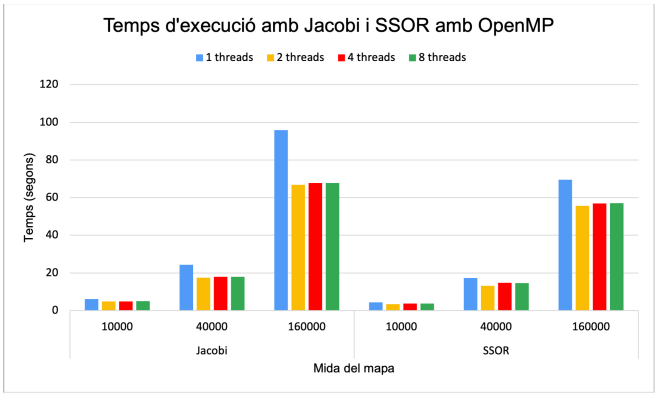

Fig. 18: Temps d'execució en segons de Jacobi respecte a SSOR amb la versió d'OpenMP

Per altra banda, l'execució de la versió optimitzada amb Python Multiprocessing, figura (19), el precondicionador Jacobi presenta millors resultats per a mapes petits de fins a 400x400. Per contres, amb mapes més grans com ara el mapa de 800x800 cel·les, tenen millor rendiment les execucions realitzades amb SSOR.

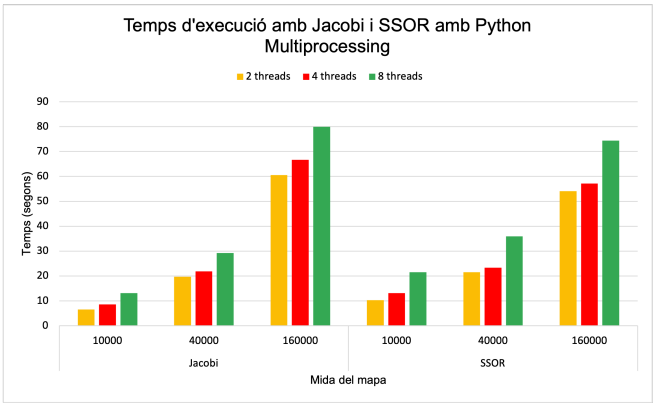

Fig. 19: Temps d'execució en segons de SSOR respecte a Jacobi amb la versió de Python Multiprocessing

Tot i això, tant a l'execució sense conservació del moment com amb conservació, es podria considerar irrellevant la diferència de rendiment entre les dues variants, ja que, si ens fixem en el temps d'execució a la taula (12) de l'apèndix (A.1), la diferència de temps respecte als dos pre-

condicionadors arriba a un màxim de 10 segons aproximadament, amb l'excepció de l'execució en seqüencial d'un mapa de 800x800, on la diferència és d'uns 26 segons. Per tant, la diferència del temps d'execució entre els dos precondicionadors és tan petita que es pot considerar insignificant.

7.3 Variació de la direcció del vent

Una altra configuració que s'ha provat ha estat la variació de la direcció del vent per comprovar si aquest factor podia influir a la velocitat d'execució del programa. Les proves s'han realitzat amb angles referents als punts cardinals; 90, 180 i 270 graus.

Com veiem a la taula (13) de l'apèndix (A.1), el fet que la direcció del vent sigui diferent no implica una variació en el rendiment destacable.

A les següents taules (5) i (6), s'ha calculat l'arrel de l'error quadràtic mitjà. Com s'observa amb execució de tipus sense conservació del moment, l'error produït en la direcció no sembla variar significativament en funció d'aquesta. Però on sí que trobem diferència és en els resultats de la velocitat. Quan la direcció del vent és vertical, 90º i 270º, presenten el mateix error, en canvi, amb 180º aquest és més elevat. Per tant, es podria dir que, tot i no veure's afectat el rendiment en funció de la direcció, sí que es pot veure influïda la precisió dels resultats.

TAULA 5. RMSE en funció de la direcció per un mapa de 400x400, velocitat 10 mph, 4 particions, sense conservació del moment.

Direcció	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
90	0,09	0,05	0,08	0,45
180	0,29	0,14	0,24	0,46
270	0,09	0,05	0,08	0,45

Per altra banda, quan l'execució és amb conservació del moment sembla que a mesura que s'augmenten els graus de la direcció, disminueix l'error produït tant en els angles com en la velocitat.

TAULA 6. RMSE en funció de la direcció per un mapa de 400x400, velocitat 10 mph, 4 particions, amb conservació del moment.

Direcció	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
90	1,10	0,57	0,49	3,03
180	1,02	0,60	0,49	2,38
270	0,88	0,54	0,41	1,96

A la figura (20) es mostren els mapes d'error de la direcció del vent amb execucions sense conservació i amb conservació del moment respectivament. Veiem que sense conservació del moment la proporció d'error és similar al llarg del mapa. En canvi, quan l'execució és amb conservació del moment, on hi ha menys precisió dels resultats és tant a les fronteres externes del mapa com en les internes del particionament.

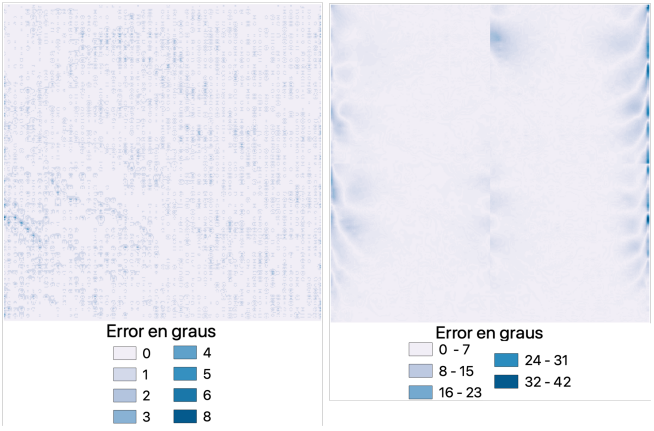

Fig. 20: Mapes d'error de la direcció d'un mapa 400x400 d'una execució sense conservació i amb conservació respectivament amb 4 threads i direcció 180 graus

7.4 Variació de la velocitat del vent

Així com s'ha comprovat si la direcció del vent influïa en el rendiment del programa, s'ha investigat si la velocitat d'aquest podria ser un factor determinant. En aquest cas, les proves s'han dut a terme amb valors de 10, 20, 30 i 40 mph.

Però, els resultats obtinguts respecte al rendiment han estat els mateixos que l'apartat anterior (7.3), com es pot comprovar a la taula (14) de l'apèndix (A.1), el rendiment no varia en funció de la velocitat.

A les taules (7) i (8) es mostra el càlcul de l'error. Veiem que tant les execucions sense conservació com les realitzades amb conservació del moment, a mesura que la velocitat del vent incrementa també augmenta l'error produït en calcular la velocitat. Tot i que a les execucions sense conservació de moment, l'error dels angles es manté estable independentment de la velocitat del vent.

TAULA 7. RMSE en funció de la velocitat per un mapa de 400x400, direcció 270 graus, 4 particions, sense conservació del moment.

Velocitat	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
10	0,09	0,05	0,08	0,45
20	0,19	0,14	0,12	0,45
30	0,28	0,22	0,18	0,45
40	0,38	0,27	0,25	0,45

TAULA 8. RMSE en funció de la velocitat per un mapa de 400x400, direcció 270 graus, 4 particions, amb conservació del moment.

Velocitat	Velocitat			Angles
	RMSE	RMSE cos	RMSE sin	RMSE
10	0,88	0,54	0,41	1,96
20	1,85	0,74	0,76	2,05
30	2,57	0,84	0,84	2,03
40	3,38	0,80	0,81	2,70

8 Conclusions

Com a conclusions podem dir que la versió Python Mulitprocessing presenta millores en els dos tipus de *solver*, amb conservació de massa i amb conservació de massa i del moment, però cal destacar que com les execucions sense conservació del moment tenen un temps d'execució tan reduït, fa que aquesta versió només presenti una millora quan els mapes són més grans i, per tant, quan hi ha més dades a computar. Per altra banda, les execucions amb conservació del moment, tot i mostrar una millora inclús amb mapes de mides petites, a mesura que va augmentant la mida, són millors els resultats pel que fa a millora del rendiment.

Respecte al sistema operatiu Ubuntu, com s'ha comentat anteriorment, perjudica l'execució de WindNinja tant pel temps d'execució com per la limitació de memòria que se li afegeix.

Per últim, s'ha observat que la modificació dels valors, com la direcció i la velocitat del vent, no són indicadors que influencien en el temps d'execució, per tant, el rendiment del programa es manté independentment del valor. Però, sí que pot veure's afectat l'error i, en conseqüència, la precisió dels resultats. Pel que fa a les execucions sense conservació del moment, l'error en la direcció del vent no varia gaire en funció del valor, però, en canvi, la velocitat sí. Per altra banda, en les execucions realitzades amb conservació del moment, tant els resultats de la direcció com de la velocitat presenta variació en l'error en funció dels valors.

8.1 Línies futures

Un cop finalitzat el treball es plantegen diferents línies per a continuar optimitzant el rendiment del programari.

Una possible continuació, seria a partir de Python Multiprocessing executar cada sub-mapa amb més d'un thread, ja que WindNinja està optimitzat amb OpenMP, si s'executa cada mapa independent de forma paral·lela es podrien obtenir millores pel que fa al rendiment.

Per altra banda, es podria optimitzar amb MPI, *Message Passing Interface*, ja que permet paral·lelització a nivell de memòria distribuïda i, OpenMP paral·lelitza a nivell de memòria compartida.

Agraïments

Primer de tot agrair al meu tutor del projecte, Tomàs Manuel Margalef, així com, a les dues professores que també m'han acompanyat i aconsellat durant tot el treball, Ana Cortes i Gemma Sanjuan, ja que m'han orientat i ajudat en tot moment.

També agrair a la meva família i amics per tot el suport proporcionat aquests mesos, i durant tot aquest temps.

Bibliografia

[1] «La Atmósfera en Movimiento,» 2005. [En línia]. Available: http://mct.dgf.uchile.cl/CURSOS/Clases_Atmosfera/clase6_viento.pdf

[2] «Origin of Wind,» National Weather Service, 3 Octubre 2021. [En línia]. Available: https://www.weather.gov/jetstream/wind

[3] E. R. Barrera, «Modelización y simulación numérica de campos de viento mediante elementos finitos adaptativos en 3-D,» Las Palmas de Gran Canaria, 2004.

[4] «MISSOULA FIRE SCIENCES LABORATORY,» [En línia]. Available: https://www.firelab.org/project/windninja

[5] «Wildfire Analyst,» 2014. [En línia]. Available: http://wildfireanalyst.com/help/spanish/hdwf.htm

[6] «Incendios forestales,» 2020. [En línia]. Available: https://www.proteccioncivil.es/coordinacion/gestion-de-riesgos/incendios-forestales

[7] «Incendios en España,» 2022. [En línia]. Available: https://es.greenpeace.org/es/trabajamos-en/bosques/incendios-forestales/

[8] «Principales causas de los incendios forestales,» 2021. [En línia]. Available: https://www.gobiernodecanarias.org/medioambiente/temas/biodiversidad/politica_forestal/incendios-forestales/causas_y_efectos_de_los_incendios_forestales/principales_causas/

[9] «WindNinja,» [En línia]. Available: https://www.fs.usda.gov/rmrs/tools/windninja

[10] «GitHub,» 2021. [En línia]. Available: https://github.com/firelab/windninja

APÈNDIX

A.1 Taules de resultats

TAULA 9. Temps d'execució i SpeedUp en segons amb tipus d'execució sense conservació del moment

Versió	Num threads	Mida mapa				SpeedUp			
		100	200	400	800	100	200	400	800
OpenMP (OMP)	1 th	2,82	10,97	44,19	2202,01	1,00	1,00	1,00	1,00
	2 th	2,32	9,01	36,74	2045,89	1,22	1,22	1,20	1,08
	4 th	2,29	9,18	35,17	1928,76	1,23	1,20	1,26	1,14
	8 th	2,16	8,76	34,24	1938,62	1,31	1,25	1,29	1,14
Multipro-cessing (PM)	2 th	3,30	8,02	31,42	1467,72	0,86	1,37	1,41	1,50
	4 th	3,06	6,64	25,83	1330,95	0,92	1,65	1,71	1,65
	8 th	4,80	9,86	33,74	1134,55	0,59	1,11	1,31	1,94

TAULA 10. Temps d'execució i SpeedUp en segons amb tipus d'execució amb conservació del moment

Versió	Num threads	Mida mapa					SpeedUp				
		100	200	400	800	1200	100	200	400	800	1200
Open MP (OMP)	1 th	94,15	389,44	1679,30	9772,47	53796,64	1,00	1,00	1,00	1,00	1,00
	2 th	82,23	310,77	1276,29	7397,06	37902,43	1,14	1,25	1,32	1,32	1,42
	4 th	68,21	243,01	978,62	5910,11	30619,01	1,38	1,60	1,72	1,65	1,76
	8 th	521,17	684,48	1450,41	6067,62	28358,54	0,18	0,57	1,16	1,61	1,90
Multipro-cessing (PM)	2 th	69,93	224,19	924,60	4803,09	30822,64	1,35	1,74	1,82	2,03	1,75
	4 th	62,34	163,38	583,79	2189,35	16437,56	1,51	2,38	2,88	4,46	3,27
	8 th	99,20	220,44	654,64	2359,77	14656,68	0,95	1,77	2,57	4,14	3,67

TAULA 11. Temps d'execució en segons amb tipus d'execució sense conservació del moment. SpeedUp Windows respecte a Ubuntu

Sistema Operatiu	Versió	Num theads	Temps d'execució			SpeedUp		
			100	200	400	100	200	400
Windows	OpenMP (OMP)	1 th	2,82	10,97	44,19	1,00	1,00	1,00
		2 th	2,32	9,01	36,74	1,00	1,00	1,00
		4 th	2,29	9,18	35,19	1,00	1,00	1,00
		8 th	2,16	8,76	34,24	1,00	1,00	1,00
	Multipro-cessing (PM)	2 th	3,30	8,02	31,42	1,00	1,00	1,00
		4 th	3,06	6,64	25,83	1,00	1,00	1,00
		8 th	4,80	9,86	33,74	1,00	1,00	1,00
Ubuntu	OpenMP (OMP)	1 th	4,39	17,39	69,52	0,64	0,63	0,64
		2 th	3,48	13,17	55,61	0,67	0,68	0,66
		4 th	3,78	14,83	56,95	0,61	0,62	0,62
		8 th	3,84	14,59	57,04	0,56	0,60	0,60
	Mulitpro-cessing (PM)	2 th	10,22	21,45	54,06	0,60	0,54	0,58
		4 th	13,05	23,36	57,13	0,57	0,47	0,45
		8 th	21,47	35,87	74,47	0,43	0,40	0,45

TAULA 12. Temps d'execució en segons i SpeedUp en funció dels precondicionadors a Ubuntu

Versió	Num threads	Temps d'execució Jacobi			Temps d'execució SSOR			SpeedUp		
		100	200	400	100	200	400	100	200	400
OpenMP (OMP)	1 th	6,21	24,30	95,87	4,39	17,39	69,52	1,41	1,40	1,38
	2 th	4,85	17,56	66,80	3,48	13,17	55,61	1,40	1,33	1,20
	4 th	4,92	17,92	67,83	3,78	14,83	56,95	1,30	1,21	1,19
	8 th	5,01	18,00	67,84	3,84	14,59	57,04	1,31	1,23	1,19
Multipro-cessing (PM)	2 th	6,53	19,65	60,57	10,22	21,45	54,06	0,64	0,92	1,12
	4 th	8,57	21,82	66,68	13,05	23,36	57,13	0,66	0,93	1,17
	8 th	13,11	29,26	80,01	21,47	35,87	74,47	0,61	0,82	1,07

TAULA 13. Temps d'execució per a diferents direccions del vent amb un mapa de 400x400

Num threads	Direcció vent	Temps d'execució OpenMP		Temps d'execució Multiprocessing	
		Sense Conservació	Amb Conservació	Sense Conservació	Amb Conservació
1	90	43,92	1746,52	-	-
	180	44,89	1623,68	-	-
	270	44,19	1679,30	-	-
2	90	40,20	1305,52	31,69	915,53
	180	41,80	1293,44	32,13	900,56
	270	36,74	1276,29	31,42	924,60
4	90	42,18	969,03	25,49	556,69
	180	39,94	961,76	24,42	550,83
	270	35,17	978,62	25,83	583,79
8	90	35,87	1488,90	32,04	662,48
	180	34,76	1471,75	31,85	656,21
	270	34,24	1450,41	33,74	654,64

TAULA 14. Temps d'execució per a diferents velocitats del vent amb un mapa de 400x400

Num threads	Velocitat vent	Temps d'execució OpenMP		Temps d'execució Multiprocessing	
		Sense Conservació	Amb Conservació	Sense Conservació	Amb Conservació
1	10	44,19	1679,30	-	-
	20	44,76	1662,55	-	-
	30	44,96	1624,40	-	-
	40	44,91	1613,12	-	-
2	10	36,74	1276,29	31,42	924,60
	20	36,88	1276,26	27,23	882,97
	30	36,28	1279,19	27,46	880,81
	40	36,30	1272,10	30,72	881,92
4	10	35,17	978,62	25,83	583,79
	20	34,54	961,13	20,45	538,00
	30	34,79	954,65	20,66	539,40
	40	34,35	954,65	20,76	544,83
8	10	34,24	1450,41	33,74	654,64
	20	34,04	1439,44	28,61	653,02
	30	36,29	1437,82	27,31	653,72
	40	35,46	1507,14	27,21	660,73

A.2 perf record

```
Samples: 299K of event 'cpu-clock', Event count (approx.): 74992500000
Overhead  Command        Shared Object        Symbol
  24.03%  WindNinja_cli  libninja.so          [.] _ZN14Preconditioner10mkl_dcsrsvEPcPiPdS0_S2_S1_S1_S1_S2_S2_
  14.66%  WindNinja_cli  libninja.so          [.] _ZN5ninja10mkl_dcsrmvEPcPiS1_PdS0_S2_S1_S1_S1_S2_S2_S2_._omp_fn.3
  11.61%  WindNinja_cli  libninja.so          [.] _ZN5ninja10mkl_dcsrmvEPcPiS1_PdS0_S2_S1_S1_S1_S2_S2_S2_
   8.36%  WindNinja_cli  libninja.so          [.] _ZN5ninja10discretizeEv._omp_fn.8
   7.46%  WindNinja_cli  libninja.so          [.] _ZNK4Mesh9get_node0ERKi
   2.91%  WindNinja_cli  libninja.so          [.] _ZN7element30computeJacobianQuadraturePointERKiS1_RdS2_S2_
   2.48%  WindNinja_cli  libninja.so          [.] _ZN7element30computeJacobianQuadraturePointERKiS1_
   2.33%  WindNinja_cli  libninja.so          [.] _ZNK4Mesh15get_global_nodeERKiS1_
   1.39%  WindNinja_cli  libninja.so          [.] _ZN5ninja11cblas_daxpyEidPKdiPdi._omp_fn.2
   1.36%  WindNinja_cli  libninja.so          [.] _ZN5ninja15computeUVWFieldEv._omp_fn.10
   1.35%  WindNinja_cli  libninja.so          [.] _ZN5ninja10cblas_ddotEiPKdiS1_i._omp_fn.1
   1.26%  WindNinja_cli  libm-2.23.so         [.] __ieee754_exp
   1.07%  WindNinja_cli  libninja.so          [.] _ZN5ninja10discretizeEv
   1.00%  WindNinja_cli  libninja.so          [.] __x86.get_pc_thunk.ax
   0.93%  WindNinja_cli  libgcc_s.so.1        [.] __powidf2
   0.91%  WindNinja_cli  libninja.so          [.] _ZNK10wn_3dArrayclEi
   0.76%  WindNinja_cli  libninja.so          [.] _ZN5ninja5solveEPdS0_S0_PiS1_iiid._omp_fn.0
   0.75%  WindNinja_cli  libninja.so          [.] _ZN5ninja11cblas_dnrm2EiPKdi
   0.70%  WindNinja_cli  libninja.so          [.] _ZN14Preconditioner10initializeEiPdPiS1_iPc
   0.54%  WindNinja_cli  libninja.so          [.] _ZNK9AsciiGridIdE9get_nColsEv
   0.53%  WindNinja_cli  libc-2.23.so         [.] __GI___printf_fp_l
   0.51%  WindNinja_cli  libm-2.23.so         [.] __ieee754_log
   0.49%  WindNinja_cli  libninja.so          [.] _ZN16wn_3dScalarFieldclEi
   0.49%  WindNinja_cli  libninja.so          [.] _ZNK7Array2DIdE11get_numColsEv
   0.41%  WindNinja_cli  libm-2.23.so         [.] __sqrt_finite
   0.34%  WindNinja_cli  libninja.so          [.] _ZN11windProfile12monin_obukovEdRKdS1_S1_S1_
   0.34%  WindNinja_cli  libc-2.23.so         [.] vfprintf
   0.32%  WindNinja_cli  libninja.so          [.] _ZN10wn_3dArrayclEi
```

Fig. 21: Resultat detallat *perf record* d'un mapa 400x400 amb 4 threads i sense conservació del moment