TRAFAIR: Understanding traffic flows to improve air quality

INEA CEF-TELECOM Project

Agreement No INEA/CEF/ICT/A2017/1566782

D3.10 Final report on results and performance of parallelization strategies for the Lagrangian particle tracking model and code release (final version)

Code	D3.10
Date	29/02/2020
Type	Confidential
Responsible	Michela Paolucci
Participants	UNIFI, UNIZAR, CESGA
Authors	Stefano Bilotta, Paolo Nesi, Michela Paolucci, Raquel Trillo Lado, Javier Cacheiro, Cecilia Grela Llerena, Javier Cacheiro López, Lorena Marrodán, Sergio Ilarri, Javier Fabra, Alessandro Bigi, Ohad Zivan, Giorgio Veratti,
Corresponding Authors	Michela Paolucci

Co-financed by the Connecting Europe Facility of the European Union

Executive Summary 5

Parallelization strategies developed chosen and tested in the TRAFAIR smart cities 5

Particle Source Splitting 6

Particle Source Splitting-Demonstrating the viability of this approach: how to combine the final results 6

Particle Source Splitting - Two parallel computation approaches: simulating source groups in parallel and splitting sources in source groups. 7

Pre-computation of the flow fields 6

Details on code release 7

GRAL - Graz Lagrangian Model Open 7

TRAFAIR parallelization strategies: final code release 8

Particle Source Splitting-Demonstrating the viability of this approach: how to combine the final results 8

Particle Source Splitting - Two parallel computation approaches: simulating source groups in parallel and splitting sources in source groups. 8

GRAL Performance Optimization 9

Introduction 9

About GRAL 9

Infrastructure used 10

Files modified for the tests 11

Previous considerations 13

Transient Mode vs Steady-state Mode 13

What is a GFF library? 13

How does GRAL use the GFF library? 13

KeepAndReadTransientTempFiles.dat 14

Deca degrees 14

problemreport.txt 14

Number of CPU cores 14

Recommendations 18

Integration Time including Steady-state 18

Recommendations 28

Vertical Stretching Factors 28

Recommendations 29

How many GFF computations to group per calculation 29

Recommendations 30

Number of meteorological conditions to pre-compute 31

Recommendations 32

Compression Rate 32

Domain Size 32

Surface Roughness 32

Building Roughness 33

Dispersion Computation Optimizations 34

Number of CPU cores used 35

Number of Particles 42

Recommendations 42

Vertical Stretching Factors 43

Result File Compression 43

Concentration Grids 44

Surface Roughness 44

Building Roughness 44

Transient Mode vs Steady-state Mode 44

Recommendations 45

How to assess the quality of the results 45

Conclusions 47

GFF Pre-computation Optimizations recommendations summary 47

Dispersion Computation Optimizations recommendations summary 47

Results obtained by each TRAFAIR smart city applying (at least) one of the parallelization strategies 49

Modena 49

Parallelization strategy results in Modena: 49

Parallelization strategy results in Modena: 49

Florence 50

Parallelization strategy results in Florence: 50

Pisa 50

Parallelization strategy results in Pisa: 51

Livorno 51

Parallelization strategy results in Livorno: 52

Zaragoza 52

Conclusions 52

Santiago de Compostela 53

Parallelization strategy results in Santiago de Compostela: 54

Conclusions 54

Parallelization strategies comparison in the TRAFAIR cities 57

Where: 59

Executive Summary

This report describes the work performed to speed-up GRAL execution using different parallelization strategies, each strategy has been benchmarked against the original sequential code and the results of these measurements are presented. The results show that these techniques will allow TRAFAIR to exploit the potential of HPC resources to speed-up considerably GRAL execution.

In D.3.2, different speed-up strategies and parallelization paradigms have been explored and described. In this deliverable we describe the following aspects:

- Parallelization strategies developed and tested in the TRAFAIR smart cities
- Results obtained by each TRAFAIR smart city applying (at least) one of the parallelization strategies
- Study related to the GRAL input parameters

In D3.7, the first summary report about the results and the performance of parallelization strategies for GRAL which have been tested in the TRAFAIR cities is described.

This deliverable describes the final report about the results and performances of the parallelization strategies adopted by each TRAFAIR city, for the Lagrangian particle tracking model.

The HPC teams involved in the parallelization activities on the TRAFAIR cities are:

- UNIFI for the Tuscany cities (Florence, Pisa, Livorno).
- CESGA for Santiago de Compostela and Modena.
- UNIZAR for Zaragoza.

Parallelization strategies developed chosen and tested in the TRAFAIR smart cities

In D.3.2, in order to optimize the time of simulation of the GRAL-GRAMM, in accordance with the Open source simulation software potentialities, a set of speed-up strategies have been analyzed. The Speed-up identified strategies are the following:

A. Spatial Domain Splitting
B. Particle Source Splitting
C. Different execution for different heights above GRAL
D. Pre-computation of the flow fields

In the following section we describe in detail the different strategies selected. In particular, we focus on the best approaches which have been tested in the TRAFAIR smart cities: 'Particle Source splitting', the 'Pre-computation of the flow fields'.

Particle Source Splitting

The simulation of the dispersion is linear with respect to the number of source group producing emissions. Therefore, this approach could be performed by maintaining the domain and splitting the input data of emission of the sources. The computation of the emission of each source (each street in the case of line source group) is independent from the computation of each other source. After all computation (each one by considering one type of source group) finish, then the results of all of them are aggregated.

Particle Source Splitting-Demonstrating the viability of this approach: how to combine the final results

The Zaragoza team created the emission files with multiple pollution sources and developed a parallelization strategy based on using different nodes/resources of the cluster in order to execute GRAL in parallel. In more detail, each run of GRAL in a specific node/resource of the cluster considers one of the types of the multiple pollution sources considered as input. After that, the outputs of the different runs are aggregated to obtain the final result. This strategy was also suggested by GRAL Developers in the GRAL course held in March 2019. The code that the Zaragoza team developed in order to test this approach was shared with every city participating in the project in June 2019 and it is available in the TRAFAIR GitLab repository (within the folder GRAL/conmapmergerC++).

Computing source groups separately and combining the results later by adding them has been tested for GRAL 19 by considering two groups of sources separately in Zaragoza. From the results obtained, adding two source groups separately computed **are consistent** with those obtained by computing the dispersion of all the emissions as a single group; therefore, this strategy is considered valid.

If this approach is followed, then the following issues must be taken into account:

- The separation of the emissions by groups has to be balanced, in such a way that each group can be computed in approximately the same time; otherwise the internal parallelization provided by GRAL is generally a more efficient and simpler option if enough resources (i.e., time and cores) are available to execute GRAL.
- Concentration files are binary, so they cannot be merged directly unless this task and read/write operations are programmed ex profeso. Due to this reason, specific code to merge the binary files has been developed (see the GRAL/conmapmergerC++ folder in the TRAFAIR GitLab repository or follow this link: https://drive.google.com/open?id=1nfa5bVZ6cOyBaLougSLXghJuWcj18jGx). For more details about the analysis of the binary concentration files, please see the following document: https://docs.google.com/document/d/16GFAmK6cspgzrbrm2OJ6c7rcIo-Tj7yw2zG6zPWf5KU/edit?usp=sharing Annex D3.7 About GRAL .con files).

Please take into account that two consecutive computings of the same domain with the same environment do not provide identical results at the cell level but they do provide very similar values and represent the

same trend at a large scale. When the results of two consecutive runs are graphically displayed they will look like indistinguishable if they have the same settings and legend.

Experimental data with absolute and relative errors to demonstrate that this approach was viable and could be applied to all the cities involved in the project can be found at https://docs.google.com/spreadsheets/d/1GGUN16U3G5Lf9Nh5UIN-ibhZwGQgx3knG-EKVHxLVC0/edit?usp=sharing.

Particle Source Splitting - Two parallel computation approaches: simulating source groups in parallel and splitting sources in source groups.

Santiago de Compostela environmental experts created the emission files for 9 pollution source groups. After studying the files and how GRAL processes each source group, CESGA developed several parallelization strategies related to source group splitting:

1. Execute all source groups together as a starting point. This execution will use parallel optimizations done by GRAL programmers. The tests will be:
    - Execute the simulations with all source groups without a GFF library.
    - Execute the simulations with all source groups with a GFF library.
2. Particle source splitting approaches:
    - *Execute each source group separately and in parallel*: study the results and the possible problems resulting from this strategy. In the section Execute each source group separately and in parallel: Best configuration of this Splitting Source approach. a better configuration of this approach is explained.
    - *Divide one source group in smaller parts*: and execute GRAL with this configuration (1 job).

The results could be found inside "Results obtained by each TRAFAIR smart city applying (at least) one of the parallelization strategies" and then "Santiago de Compostela [CESGA]".

In these sections, it is shown how the approach has been validated by verifying that the concentration maps obtained using the sequential simulation and the parallel one are equivalent. The performance gains are also shown.

To run GRAL in parallel using this approach a modified version of GRAL is required. The changes have been incorporated in the new 2020_01 Beta4 pre-release of GRAL, so they will be generally available.

Pre-computation of the flow fields

This speed-up strategy has already been described in D.3.2. Scripts for performing this strategy are available in the Github directory: https://github.com/disit/trafair under the *Affero General Public License v3.0* and also in the TRAFAIR GitLab repository https://gitlab.com/trafair. In more detail, in the gitlab repository a docker image is available. This image contains the scripts in order to generate a gff library and also to perform a

pollutant dispersion simulation taking into account the gff library generated previously. Moreover, a set of tests, in some TRAFAIR Smart Cities, have been done changing the different GRAL input parameters, and the results on how GRAL behaves with the actual parallelization are reported in this deliverable.

This approach uses pre-computed Gradient Flow Fields (gff files) that, according to our tests, reduces the total computational cost during the real-time GRAL model simulation. More precisely, the GRAL model considers two main steps for its computation: the *flow fields computation* and the *particle dispersion computation*. By using the pre-computed gff files, the simulation time of the GRAL model converges to the time needed for the particle dispersion computation. For such a target, the creation of the flow fields library, involving the pre-processing computation of the "gff files", is needed in order to perform the simulation of the GRAL model during the real-time activity. More precisely, for each (forecast) weather situation occurring as input data in the real-time simulation of the GRAL model, the corresponding "gff file" (having the closest weather situation with respect to the occurred one) in the flow fields library is selected and considered via automatic procedure in the GRAL simulation. Then, the simulation time of the GRAL model only consists of the time needed for the particle dispersion computation, since the step related to the flow fields computation is loaded. For a more detailed description of this strategy, see D3.2.

The computational cost needed for the creation of such flow fields library (involving the pre-processing computation of the "gff files") depends on both the selected city domain in which the model GRAL is applied and the number R of weather situations which are considered in terms of wind input data. More precisely, the computations of the 'gff files' are independent from each other, so a **parallelized strategy is considered** to reduce the whole computational time needed for the definition of the flow fields library. This can be one of the aspects of the parallelization activity, since it is possible to make the computation in a parallel modality. Of course, the R weather situations can be partitioned in S different processes and S different simulations of the GRAL model can be simultaneously computed in order to split the calculus of the flow fields library in different resources. In that way, the total computational cost needed for the flow fields library having R elements converges to the computational cost of a library having only R/S elements.

The results' details involving the mentioned activity are reported in this deliverable.

Benchmarking and its results done by CESGA regarding the GFF pre-computation could be found inside the "GRAL Performance Optimization" section.

Details on code release

GRAL - Graz Langrangian Model Open

The GRAL code used as a **starting point** for our work on parallelization strategies, is available under the GNU/GPL 3 licence at the following link, starting from Jan 12, 2020:

https://github.com/GralDispersionModel/GRAL

TRAFAIR parallelization strategies: final code release

Table reporting the link to code released as result of the work performed to speed-up GRAL execution using different parallelization strategies.

Speed up/parallelization strategy	TRAFAIR city/cities in which is adopted	Development team	link to the code release	link to the code techincal guidelines
Particle Source Splitting - Two parallel computation approaches: simulating source groups in parallel and splitting sources in source groups.	ZARAGOZA	ZARAGOZA	https://drive.google.com/file/d/1nfa5bVZ6cOyBaLougSLXghJuWcj18jGx/view	https://docs.google.com/document/d/16GFAmK6cspgzrbrm2OJ6c7rcIo-Tj7yw2zG6zPWf5KU/edit?usp=sharing
Dockerize image of GRAL for flow fields precomputation and run GRAL dispersion	ZARAGOZA	ZARAGOZA	The docker image in the Trafair GitLab repository is also available as an annex to this deliverable (see Annex D3.10-GRAL_image.tar.gz available here: https://drive.google.com/file/d/1Ap8u3Wo-Z0rr1UI52QM0U1XMUBT-A1Nw/view?usp=sharing)	Readme.md available in the Docker image

Pre-computation of the flow fields	FIRENZE PISA LIVORNO	UNIFI	https://github.com/disit/trafair	inside the code and in D3.2

GRAL Performance Optimization

The objective of this point is to provide general guidelines about how to run GRAL based on the results of a set of comprehensive benchmarks created, executed and analysed by CESGA and performed using FinisTerrae II supercomputer resources at CESGA. The recommendations shown in this section are obtained taking into account benchmark results and the opinion of GRAL (http://lampz.tugraz.at/~gral/) programmers Markus Kuntner and Öttl Dietmar as experts.

With this analysis each city could prepare their GRAL operative strategies according to their cities particularities and the resources available for them.

GRAL performance analysis includes a study of several GRAL input parameters as they affect considerably the performance of the program.

Introduction

The parameters taken into account to measure performance in each execution are:

- Time elapsed.
- Number of CPUs needed.

Maximum memory consumed could be checked but it's not part of the performance optimization as the optimization refers to HPC (high performance computing) in reference to parallel computing.

Benchmarks are divided in:

- GFF pre-computation benchmarks.
- Dispersion benchmarks.

It is important to take into account that for the benchmarks exclusive resources (not shared nodes) were used so other jobs will not affect the results.

About GRAL

The Graz Lagrangian Model - GRAL is the software used to simulate the pollution concentration in this project. More information about GRAL and its creators could be found in:

http://lampz.tugraz.at/~gral/

Infrastructure used

CESGA has advanced infrastructures that are allocated to increase the research capacity of the scientific-technological community and industry. The high quality of the available infrastructures, as well as its uniqueness in the entirety of the Spanish State, have motivated the recognition of the installation as the Singular Scientific Technological Infrastructure of Spain (ICTS). The e-infrastructures managed by CESGA were financed by the Regional Government of Galicia (Xunta de Galicia), the Spanish National Research Council (CSIC), the Ministry for Economy and Competitiveness of the Government of Spain, and the European Regional Development Fund (ERDF).

CESGA has computing servers of different architectures to allow the researcher to always choose the architecture that is best adapted to needs. Computing systems installed in CESGA:

- **Finis Terrae II**: architecture used for all these tests.
- **SVG**.

Finis Terrae II (or FTII) It is the most powerful calculation server in CESGA. It is a computer system based on Intel Haswell processors and interconnected through Infiniband network with a peak performance of 328 TFlops and supported on Linpack of 213 Tflops. It is composed of the following nodes:

Number of Nodes	***Cores per node***	***Processors per node***	***Node accelerators***	***RAM per node***	***Storage per node***	***Shared Storage***	***Net***
308	**24**	**2 Haswell 2680v3**	**-**	**128GB**	**1TB**	**Lustre768TB NFS 600TB**	**InfiniBand FDR GbE**
4	24	2 Haswell 2680v3	2 NVIDIA Tesla K80	128GB	1TB	Lustre768TB NFS 600TB	InfiniBand FDR GbE
1	128	8 Haswell 8867v3	-	4096GB	28,8TB	Lustre768TB NFS 600TB	InfiniBand FDR GbE

For all the benchmarking the nodes used are the ones described in the first line.

Files modified for the tests

To configure benchmarks correctly some GRAL input files were modified:

- **GRAL.geb**: stores basic information about the GRAL grids and the model domain (see GRAL documentation for more info about this file).
- **in.dat**: this file stores the main parameters to run GRAL, such as the number of particles to be released per second, the dispersion time, etc. (see GRAL documentation for more info about this file).
- **Integrationtime.txt**: (optional) used to set the minimum and maximum number of iterations for the prognostic microscale flow field model algorithm used by GRAL. By default the values are: 0, 500.
- **Max_Proc.txt**: maximum numbers of threads when GRAL is executed using parallel computing. It can be considered equivalent to the number of CPU used.
- **meteopgt.all**: stores meteorological data. It could be equal to mettimeseries.dat (in terms of data, not structure) or be used to describe all the precomputated GFF in a GFF library (see GRAL documentation for more info about this file).
- **mettimeseries.dat**: series of meteorological data used in the simulations. During daily operative all the cities will fill this file with the daily meteorological info that will be simulated that day. Each line of this file stores a meteorological condition organized in columns (separated by commas) like this:
    - day.month, hour, Wind speed, wind direction, stability class
- **line.dat**: includes all line sources for the simulation (see GRAL documentation for more info about this file).
- **GRAL_FlowFields.txt**: see GRAL documentation for more info about this file.
- **GFF_FilePath.txt**: stores the path to the GFF library.
- **emissionsX.dat**: This is the "modulation" of emissions: it's the multiplication factor of the emissions declared in point.dat (or line.dat). For each source emission GRAL needs an emissions.dat file, for example: for source 1 an emissions001.dat file should be present in the folder.

More extended explanations for this files could be find in GRAL documentation.

GFF Pre-computation Optimizations

GRAL execution has 2 to 3 main steps:

- **Step 1**: calculation of the prognostic wind field around or through obstacles (buildings, vegetation, etc.). A parallelization of this step is difficult, because of the underlying physics. However, in practical applications the execution could be boosted: compute classified wind field one time, save the wind fields (*.gff files are generated), GRAL searches the best matching wind field for the expected meteo-situation and re-uses these wind fields.

- **Step 2 + 3**: these steps compute the dispersion of the pollutants using the wind fields calculated on step 1. In step 2 GRAL calculates the dispersion of transient particles (particles still available from earlier time steps). In step 3 new released particles are computed. These loops are parallelized in their code so part of it is executed in parallel by defect.

In step 1, for each particle and meteorological situation a prognostic wind field is calculated, this prognostic wind field will be the same between simulations if the meteorological situation doesn't change. If this prognostic wind field is stored in a file (GFF files) it could be reused in future simulations saving the time needed to calculate it again.

As a result of this, if GFF (prognostic wind field) are precomputed, step 1 will execute considerably fast (this is one of the parallelization strategies). Each precomputed GFF corresponds to a weather situation: wind direction sector, wind speed class, stability class, frequency. Meteorological experts from each city should prepare a table with the values for wind direction sectors, wind speed classes and stability classes (related to each city) to be precomputed to obtain a GFF library. The number of GFF needed could be very high so a performance optimization must be studied and this is the objective of this document section.

To study the performance optimization for GFF precomputation these values were taken into account:

- **Number of CPU cores** used in the computation.
- **Integration Time including Steady-state**.
- **Vertical Stretching Factors**: this value affects in great measure the performance of GRAL. For each simulation a grid (divides the map in sectors) is set, the stretching factors sets how the grid changes with height: the grid gets coarser with height. With this, CPU and memory needs could decrease significantly. Each city should decide which stretching factors they need to have quality results while meeting their resources availability. This parameter could be found inside GRAL.geb file.
- **How many GFF** computations to group per calculation.
- **Number of meteorological conditions** to pre-compute.
- **Compression Rate**.
- **Domain Size**.
- **Surface Roughness**.
- **Building Roughness**.

Previous considerations

Transient Mode vs Steady-state Mode

This important information about GFF precomputation should be taken into account before reviewing the benchmarks done for GFF pre-computation: for the calculation of a *.gff pool you can launch GRAL in one of these modes that affect the dispersion step of GRAL:

- Steady-state mode: GFF files are always precomputated. If a library of GFF is used but GRAL is launched in steady-state the GFF files will be rewritten.
- Transient mode: each line in “mettimeseries.dat” needs a corresponding line in “meteopgt.all”; the files must contain the same meteorological data for each line.

The configuration for Transient Mode meteopgt.all contains classified meteorological cases. The *.gff wind fields are only calculated for these classified cases (the pool). Meteopgt.all and the *.gff files belong together. If the transient propagation is calculated for a certain period of time, one needs to create a mettimeseries.dat file with the given meteorology. GRAL takes each line of mettimeseries.dat and looks for the corresponding line in meteopgt.all, then reads the *.gff file (or calculates a new one if none exists).

What is a GFF library?

A GFF library consist in:

- All the pre-computed GFF files for a specific city (also called pool of GFF). Each GFF is named with the number of its meteorological data line in meteopgt.all
- A meteopgt.all file with a line for each GFF pre-computed with the meteorological data that identified it. This file should be copied (or linked) in each folder for the daily operative.
- A GFF_FilePath.txt file with the path to the GFF pool. This file should be copied (or linked) in each folder for the daily operative. This file should only contain two lines:
    - 1st line is for the path in Windows: only needed if executing GRAL in Windows. This line could be empty but should exist.
    - 2nd line is for the path in Linux systems (like Red Hat, etc.): only needed if executing GRAL in Linux systems. This line could be empty but should exist.

How does GRAL use the GFF library?

To use a GFF library, GRAL should be executed in transient mode (flag to be set in GRAL.geb file). Then it will work like indicated in the following:

- It takes the 1st meteo situation from the file mettimeseries.dat (meteorological data to be simulated), searches the corresponding line in the meteopgt.all file (each line correspond to one GFF) and reads (or calculates if not already available) a new *.GFF file (file name corresponds to the line in meteopgt.all). If I only wanted to calculate *.GFF files, I would set the flag to 1.

KeepAndReadTransientTempFiles.dat

When calculation is performed in transient mode, it will be possible to keep the pollutant plume from a previous calculation by using the TRAFAIR option “KeepAndReadTransientTempFiles.dat”.

Deca degrees

It should be taken into account, when preparing the stability matrix to precompute GFF, that GRAL uses the degrees in sectors of 10 degrees, so you have to divide the degrees by 10.

problemreport.txt

The divergence is checked when calculating the flow field files only. The dispersion module takes the calculated windfields as they are. As long as there is no entry in the file problemreport.txt, there is no problem. Even if there is an entry, GRAL users should check the wind vectors.

Number of CPU cores

It is important to decide the optimum number of CPU cores to be used during the execution, as more CPU cores mean the computation will be more expensive: if 1 node with 24 cores is used to parallelize during 1 hour, the real time to pay for will be 24 hours.

For that we had to analyze the scaling of GRAL with the number of cores: do GRAL execute faster with more cores? How faster is the execution related to the number of cores used?

For this concrete analysis we have used the **Módena** city as input and for this a mettimeseries.dat file was created with the following line:

3.11,0,0.6,13.5,6

As explained previously this line corresponds to:

- day.month: 3.11
- hour: 0
- wind speed: 0.6

- wind direction: 13.5
- stability class: 6

Finally, it is useful to represent the speed-up with respect to the number of cores used. The speedup represents the ratio between the CPU time using a single CPU core and CPU time using multiple cores.

speedup = CPU time using 1 core / CPU Time

Ideally the speedup should be very similar to the number of cores but in real applications this is not generally the case, in this table and in the figure gives us an idea of how far GRAL scales with respect to the number of cores:

Total time to compute 1 GFF (Modena data) and speed-up with respect to the number of CPU cores used.		
Number Cores	***Elapsed (s)***	***Speedup***
1	55,05	1,00
2	26,78	2,06
3	18,70	2,94
4	14,80	3,72
5	12,64	4,35
6	10,97	5,02
7	10,07	5,47
8	9,09	6,05
9	8,36	6,58
10	7,96	6,91
11	7,57	7,28
12	7,09	7,77
13	6,99	7,88
14	6,72	8,20
15	6,52	8,45
16	6,32	8,71

17	6,33	8,70
18	6,13	8,99
19	6,00	9,18
20	5,82	9,45
21	5,84	9,42
22	5,83	9,45
23	5,78	9,52
24	5,65	9,74

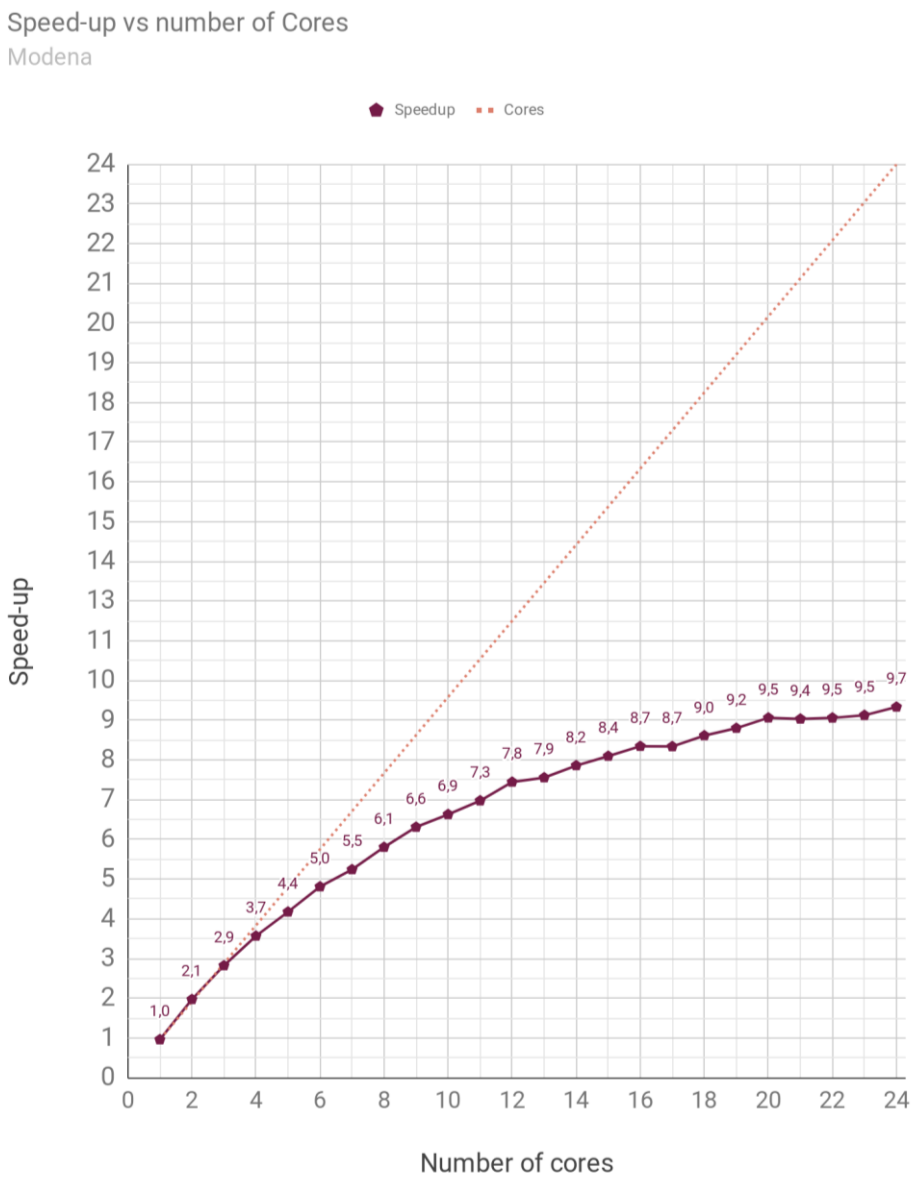

Figure: GFF computation speed-up with respect to the number of CPU cores used.

Recommendations

As the figure shows, the speed-up grows steadily until 12 cores (when using 12 cores the time equals using 8 cores separately), then the growth slows down until the moment when using 24 cores equals the time for 10 cores used separately.

As a result:

- Best option will be 12 cores, this configuration will allow two executions of GRAL in one node.
- Next best option will be 24 cores, one node in exclusive. This option should only be taken into account if the problem is too big (in terms of time, memory, etc.) to be executed in half a node.

Integration Time including Steady-state

For the GFF computation it is important to decide the optimum number of iterations to perform until the gradient has converged. The computational time required will change linearly with the number of iterations, i.e. if we perform 1000 iterations the time will be double than if we perform 500 iterations.

The default values for this parameter are 100 for minimum and 500 for the maximum. The maximum value accepted by GRAL for the maximum is 2000.

This value could be set in the IntegrationTime.txt file and there are two options available:

- Fixing the maximum number of iterations allowed
- Running until a steady state is achieved

A priori, the steady-state convergence method seems the most accurate because it will automatically stop the GFF iterations if it reaches convergence. If not it will run up to 10001 iterations.

We have analyzed both approaches using different weather conditions to verify if they affect convergence.

For the analysis we have used the **Modena** city as input (as an example) and the following weather conditions:

	Wind speed (m/s)	***Wind direction (degrees)***	***Stability class***
Low wind 45º	0.5	45	F
Medium wind 45º	2	45	D
High wind 45º	7	45	F
Low wind 225º	0.5	225	F
Medium wind 225º	2	225	D

High wind 225º	7	225	F

Since the steady-state option uses up to 100001 iterations, we selected just one of the previous weather conditions to generate the GFF using this convergence mode:

	Wind speed (m/s)	***Wind direction (degrees)***	***Stability class***
Low wind 45º	0.5	45	F

In all the computations we have used a maximum number of iterations of 2500, except in the steady-state computation where the maximum is, by definition, set to the maximum allowed by the program 10001.

We have also used the following set of stretching factors in GRAL.geb:

2,1.00,30,1.05,50,1.15,100,1.25,200,1.5

which means:

- Up to 30m stretching factor 1
- From 30m up to 50m stretching factor 1.05
- From 50m up to 100m stretching factor 1.25
- From 200m stretching factor 1.5

GRAL shows us the convergence (or advection) between iteration (how different is an iteration from the previous one, if they are converging the value will go near 0). It could be interpreted as the error that should go down from each iteration group (this value is shown each 100 iterations) to the next one.

In the following table we summarize the results obtained:

	CPU Time (s)	***Convergence at Iteration***							
		100	***300***	***500***	***700***	***1000***	***1500***	***2500***	*Minimum*
Low wind 45º	1.422,30	12,397	1,490	0,546	0,173	0,068	0,041	0,037	*0,037*

Medium wind 45º	1.422,40	16,607	1,974	0,706	0,244	0,090	0,046	0,038	*0,038*
High wind 45º	1.438,60	12,398	1,492	0,546	0,173	0,067	0,042	0,039	*0,039*
Low wind 225º	1.458,20	12,416	1,482	0,596	0,183	0,061	0,044	0,037	*0,037*
Medium wind 225º	1.422,40	16,613	1,970	0,777	0,268	0,090	0,046	0,039	*0,039*
High wind 225º	1.404,00	12,416	1,482	0,597	0,182	0,063	0,042	0,038	*0,038*
Low wind 45º steady state	1.458,20	12,397	1,490	0,546	0,173	0,068	0,042	0,038	*0,038*

To analyze the convergence it is also useful to represent graphically the convergence of each weather condition:

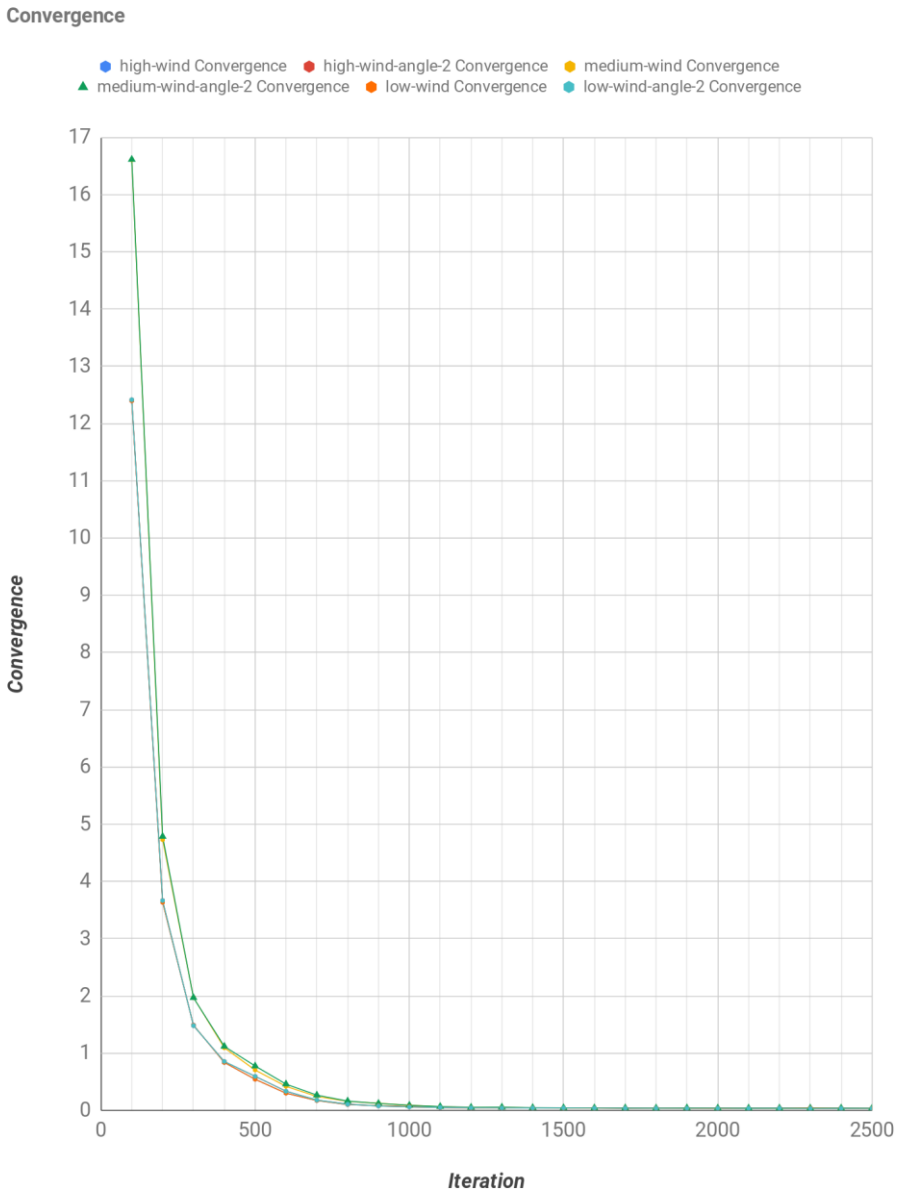

Figure: GFF convergence of the low wind 45º situation

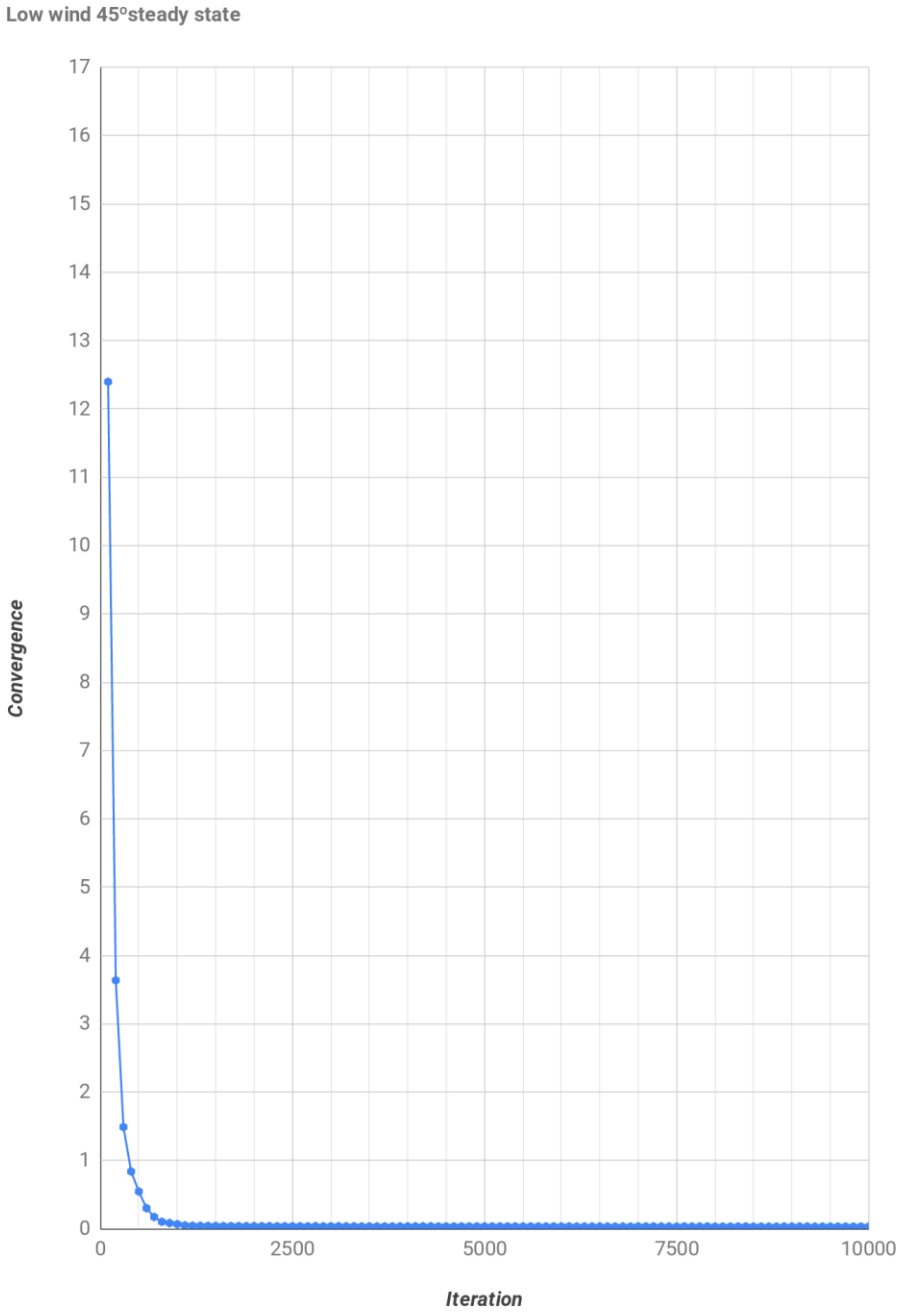

Figure: GFF convergence of the Low wind 45º steady state situation

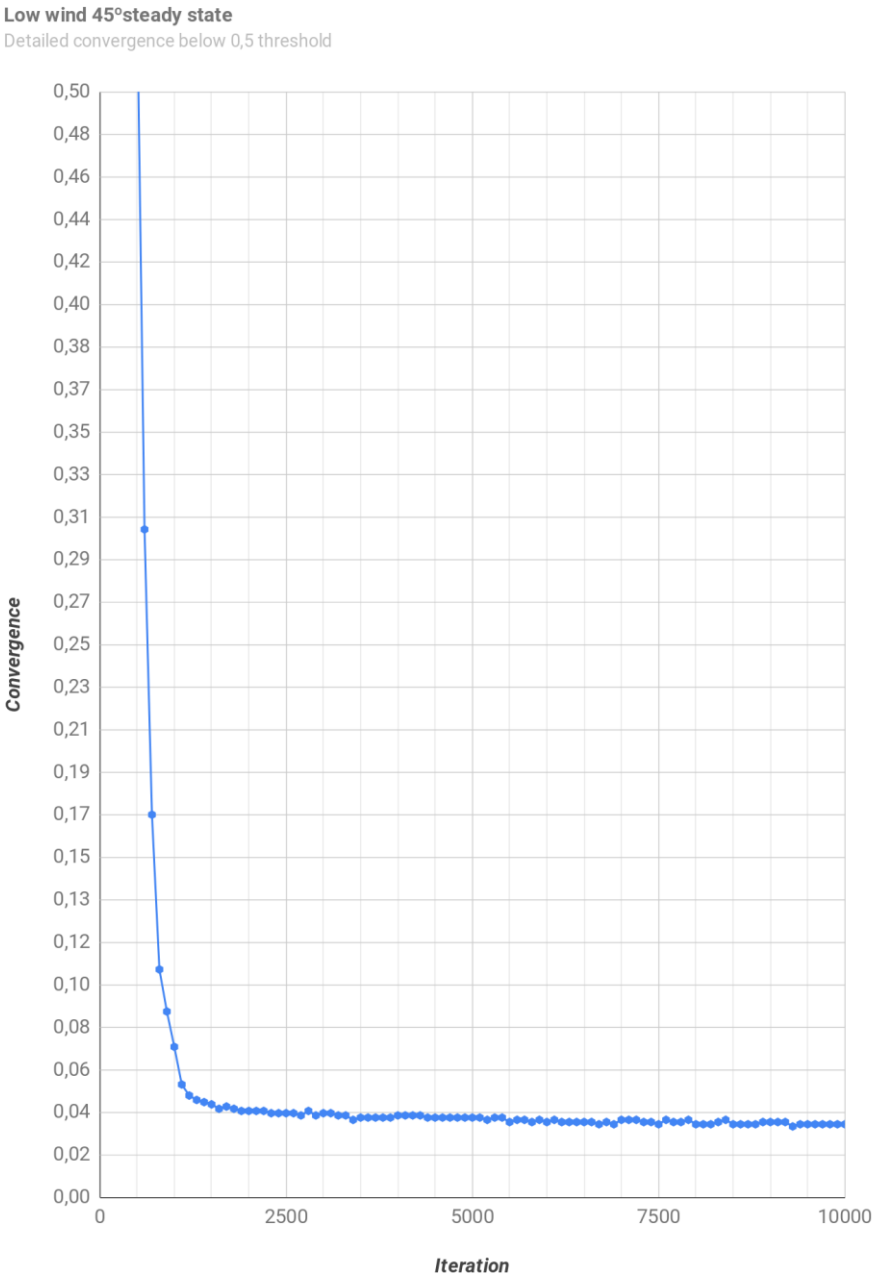

Figure: GFF convergence of the Low wind 45º steady state situation - Detailed Variation of the convergence below 0,05 threshold

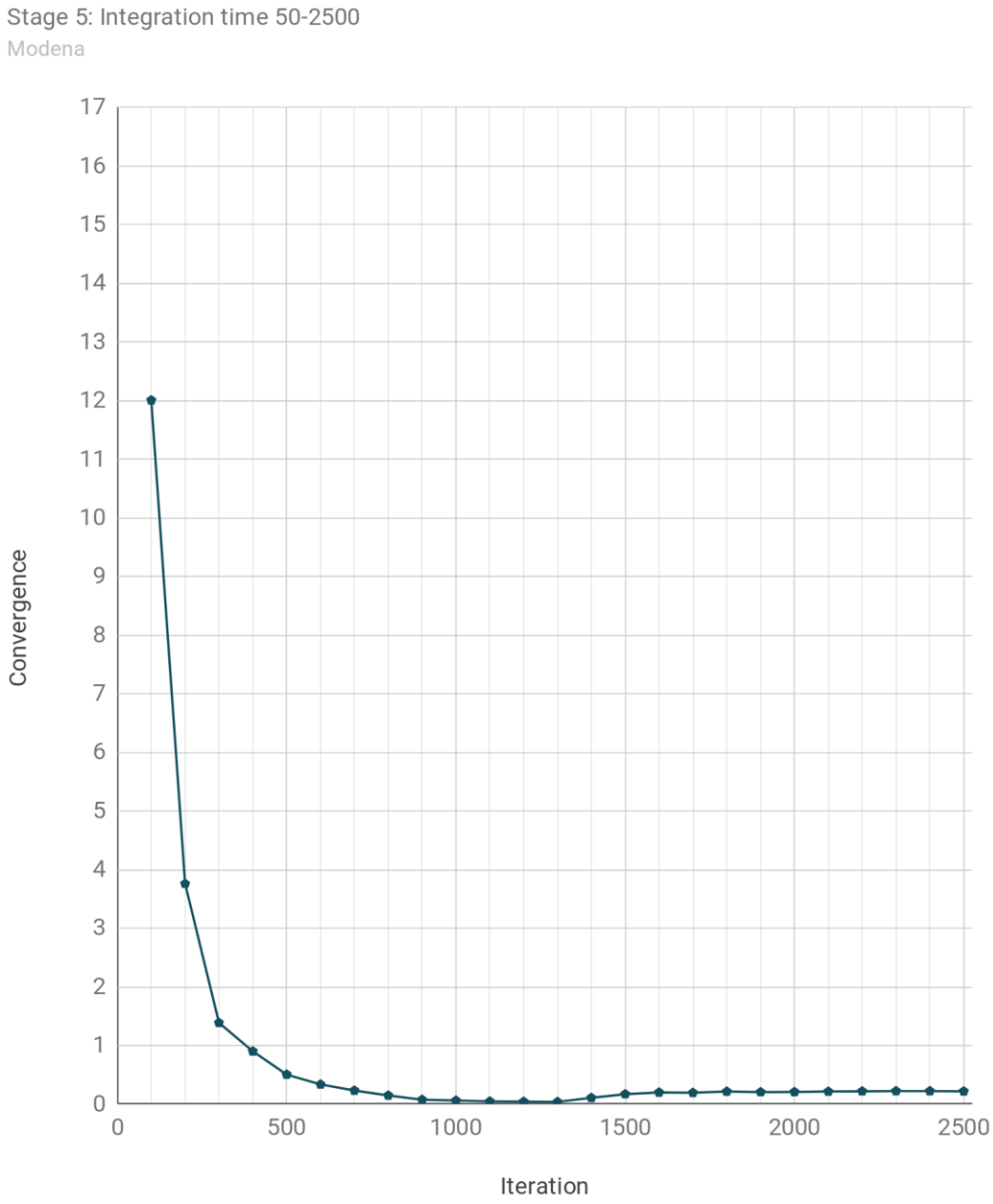

Figure: Convergence for a GFF computation for Modena.

Finally it is useful to compare the reduction of the divergence with respect to the computation time used. For that we will use an estimation of the computation time at a given iteration as the proportional time with respect to the total time used in the GFF computation part of the calculation (the dispersion time and the initial load time is not included).

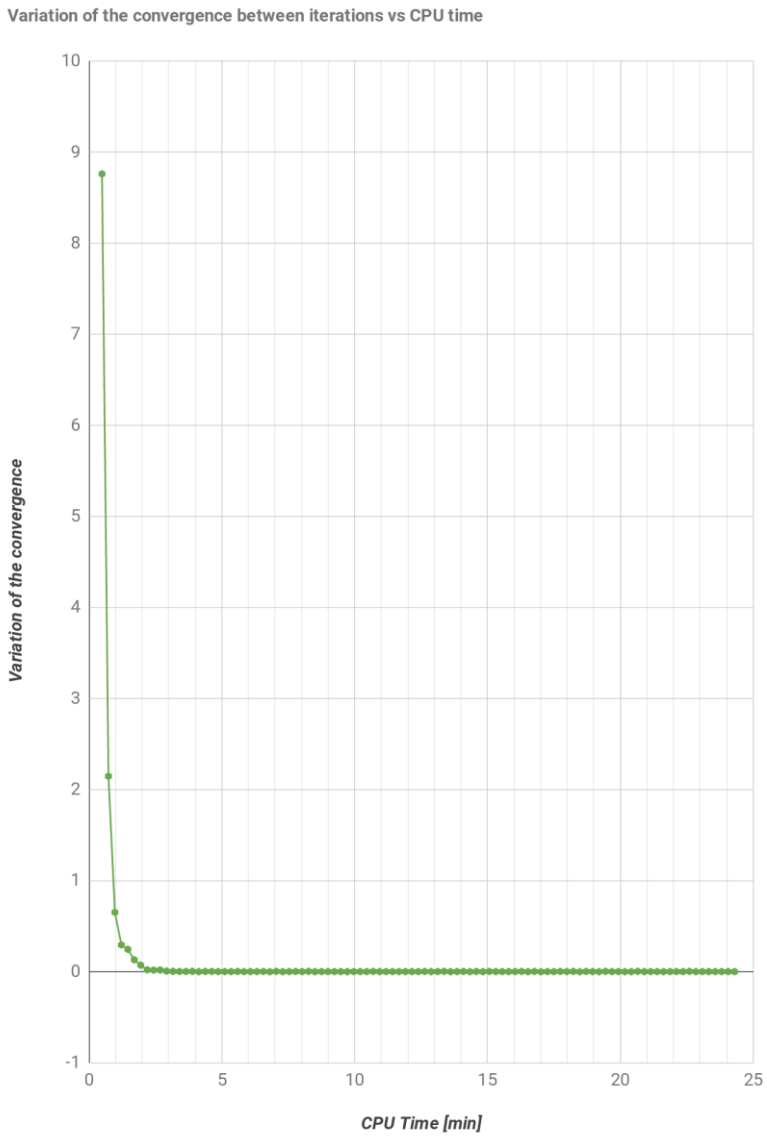

Figure: Variation of the convergence between iterations vs CPU time

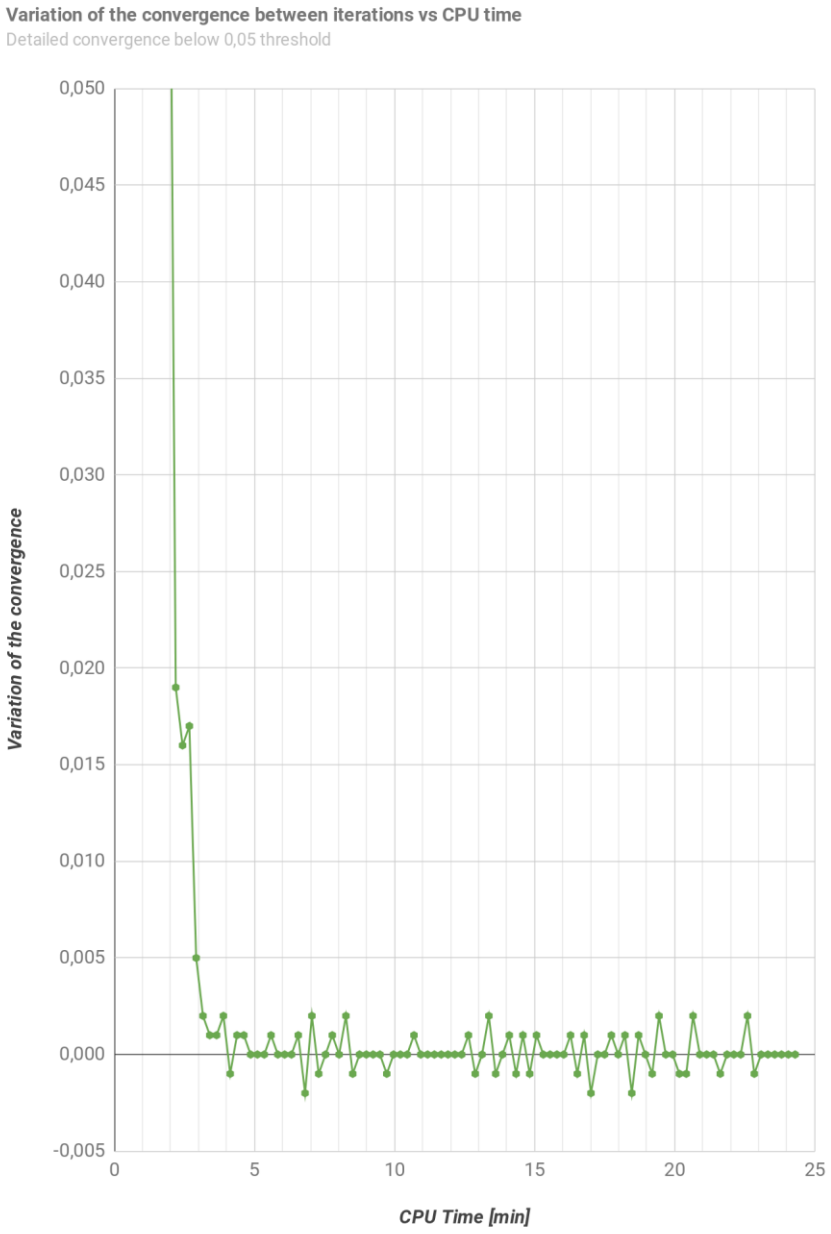

Figure: Variation of the convergence between iterations vs CPU time - Detailed Variation of the convergence below 0,05 threshold

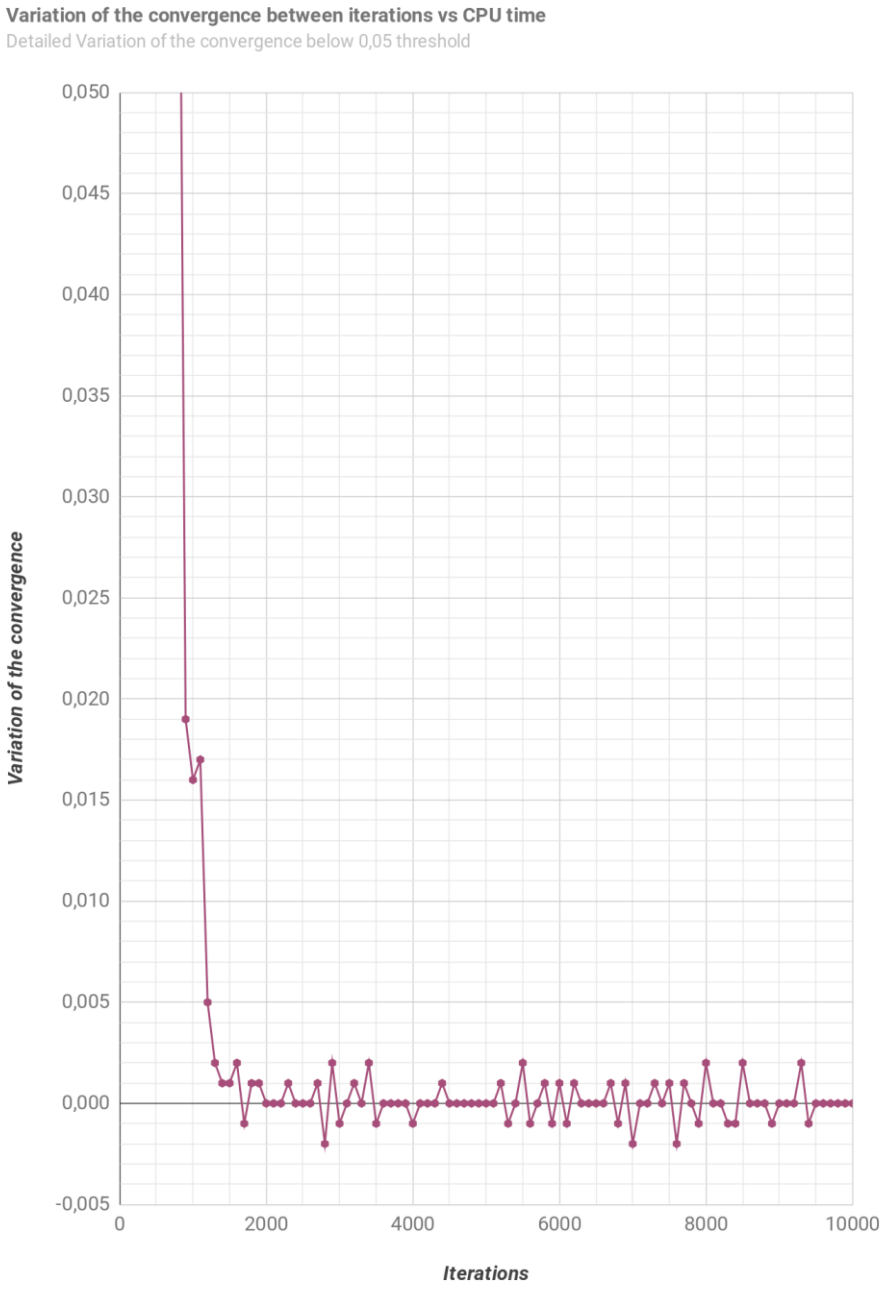

Figure: Variation of the convergence between iterations vs Iterations - Detailed Variation of the convergence below 0,05 threshold

Recommendations

As seen in the figures, the best value for Integration Time (and the one suggested by GRAL programmers) is a maximum of 500. The reason is that convergence goes under 1 when reached that number of iterations but the time spent in all the process will increase significatively if the maximum is set for more than 500.

A minimum of 100 is the best option as GRAL will execute a minimum of 100 iterations.

The contents of IntegrationTime.txt, if following this guide, will be:

100

500

Vertical Stretching Factors

For each simulation a grid (divides the map in sectors) is set, the stretching factors sets how the grid changes with height: the grid gets coarser with height. Each city should decide which stretching factors they need to have quality results while meeting their resources availability. This parameter could be found inside GRAL.geb file.

The values in this file describe a change that should be applied to the grid from a defined height: for example this line:

2,1.00,20,1.02,50,1.05,150,1.1,250,1.2
Means:

- From 2 to 20 the grid should stay as defined.
- From 20 to 50 the grid will be multiplied by 1.02
- From 50 to 150 the grid will be multiplied by 1.05
- From 150 to 250 the grid will be multiplied by 1.1
- From 250 and higher the grid will be multiplied by 1.2.

Each cell in the grid will grow with the height, affecting GRAL performance.

This configuration will affect the performance of GRAL as we suppose it will help to reduce the memory used, the documentation suggests that this configuration will also increase accuracy for low heights.

	Elapsed Time (s)	***MaxRSS (KB)***

standard	8:35	37924964
Values suggested by Modena 1.5,1.00,20,1.25,50,1.5,150,1.5,250,1.5 !cell-size for cartesian	4:57	32686016
Values suggested by CESGA	5:50	30113448
Values suggested by GRAL programmers 2,1.00,20,1.02,50,1.05,150,1.1,250,1.2 !cell-size for cartesian wind	5:44	41147228

Recommendations

The value suggested by GRAL programmers has a good performance but this parameter could only be set by the environmental experts of each city taking into account the availability of their computation resources.

How many GFF computations to group per calculation

Scaling with number of GFF per batch (using 24 cores), for this test Modena city data was used.

Num GFF	*Total time [s]*	*Total time [sec] / num GFF*
1	339,2	339,2
2	416,3	208,2
3	593,3	197,8
4	799,5	199,9

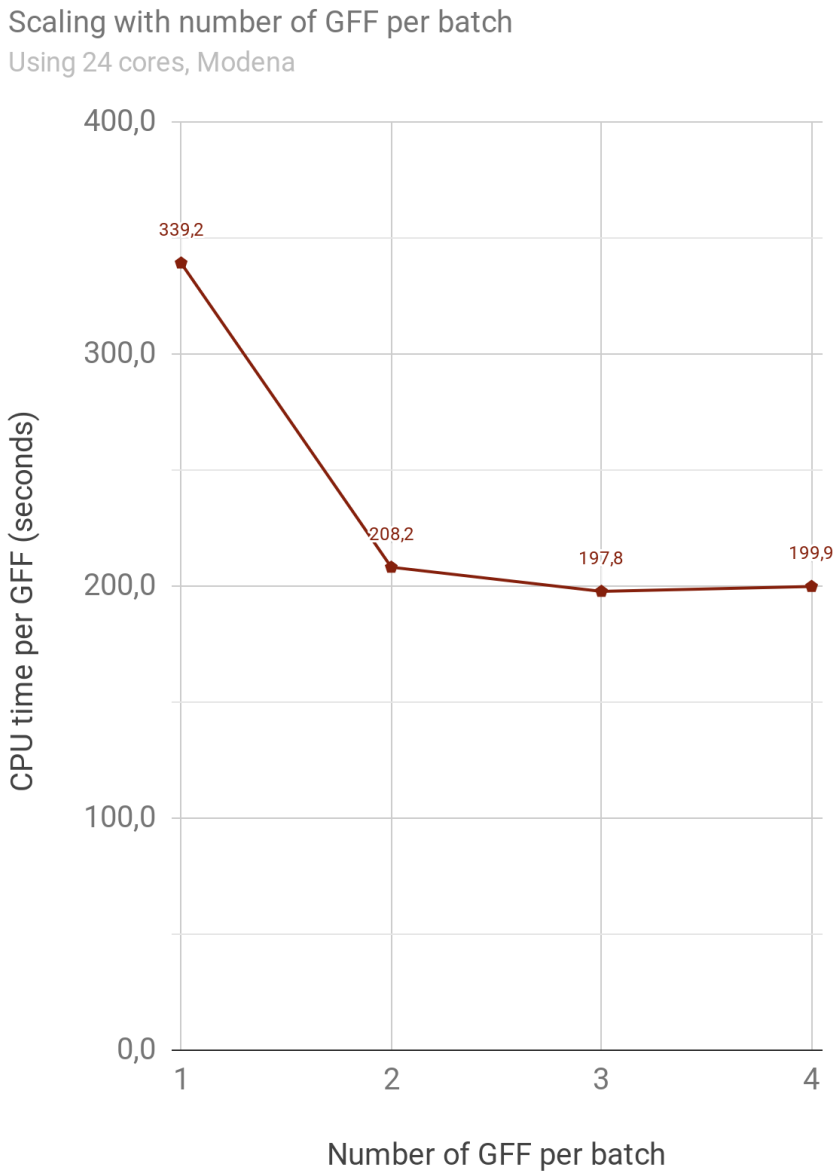

Figure: Efficiency with respect to the batch size (i.e. the number of GFF files precomputed in the same job)

This figure shows the CPU time needed to precompute one GFF in four jobs that compute 1 GFF, 2 GFF, 3 GFF and 4 GFF (batches).

Recommendations

As the figures show, from a batch of 2 to a batch of 4 the difference is not big enough to be taken into account. The number of GFF on each batch to be executed in parallel depends on the computational resources that will be used by the city to precompute GFF.

CESGA recommends the precomputation of GFF in batches of 2 as minimum and maximum as the resources allow.

Number of meteorological conditions to pre-compute

When generating the library of GFF files it is important to decide the wind speeds, directions and stability classes to pre-compute.

Take into account that all selected stability classes need to be pre-computed, because the wind profile is influenced by the stability class. The stability class does not influence the GFF so we can just reuse the same files for the other stability classes (see D3.2 A parallelized version of the GRAL atmospheric dispersion model - v1).

Even if stability does not influence the flow field computation itself, it has an effect on the calculation of the vertical wind profile, so all selected stability classes need to be considered. Please notice that this corrects the information provided in D3.2 A parallelized version of the GRAL atmospheric dispersion model - v1.

The general recommendation is that each city performs a historical analysis of the wind speed, direction and stability class to decide the meteorological conditions to pre-compute based on this matrix.

It is always possible to improve the GFF library later adding additional meteorological conditions so we recommend that you start with a small set and then if you find some additional conditions are needed, just pre-compute the corresponding GFF files and add them to the library.

As an example, we provide here the weather conditions chosen to generate the GFF library for the city of Santiago de Compostela:

- Wind speed (m/s): 0.5 1 2 4 6
- Wind direction (degrees): each 22.5º
    - 0 22.5 45 67.5 90 112.5 135 157.5 180 202.5 225 247.5 270 292.5 315 337.5
- The stability classes considered will be A-F (G class will be approximated by F):
    - 1, 2, 3, 4, 5, 6

Since microscale flow fields need to be computed for each dispersion situation as the initial vertical wind profile changes with stability we have to take into account all selected classes. It is still possible to reduce the number of combinations using the stability matrix for the given city because not all stability classes are possible for all wind speeds.

In our case this matrix was not available and since each GFF took less than 10 minutes to compute, it was decided to pre-compute all combinations.

So in the example given all combinations of 6 speeds, 16 directions and 6 stability classes are taken into account. This means that to generate the GFF library we would need to pre-compute 480 GFF files.

For that we generated a meteopgt.dat file with all the lines needed to precompute the 480 GFF files and then we precomputed them in batches so the computation is parallelized.

Recommendations

The environmental experts should take into account the creation of a stability matrix adapted to their city and based on this the library should be computed. This value depends totally on meteorological characteristics of the city.

Compression Rate

There are 2 options available for GFF file compression:

- 0: lower compression rate
- 1: higher compression rate

The time needed for the compression varely affects the total time of the simulation so this value would depend on the resources available.

Domain Size

The effect of domain size including horizontal and vertical resolution on the quality of the results and in the computation time has yet to be established. A validation method for the quality results needs to be established before performing this analysis.

Also the domain size cannot be reduced in specific cases, some cities should be taken into account as a whole and even the domain will grow bigger than the city to include motorways for example. This value will depend totally on the cities characteristics.

Surface Roughness

The surface roughness is used if different surfaces need to be taken into account (for example complex terrain without buildings or surfaces that affect the simulation), and that is not our case.

Roughness length in [m]. In case that GRAMM wind fields are used as input, and that the land-use file landuse.asc is available, the roughness length defined in the file in.dat is not used.

It is specified in the in.dat configuration file. Currently we are using 0.2m.

The effect of the surface roughness on the quality of the results and the computation time has still to be analyzed. A validation method for the quality results needs to be established before performing this analysis.

Building Roughness

This is an optional parameter used to set the surface roughness for obstacles used in the prognostic microscale flow field model of GRAL. The default value is 0.001.

It is specified in the building_roughness.txt configuration file. Currently we are using 0.001m (the default value).

The effect of the building roughness on the quality of the results and the computation time has still to be analyzed. A validation method for the quality results needs to be established before performing this analysis.

Dispersion Computation Optimizations

As it is explained previously GRAL execution has 2 to 3 main steps:

- **Step 1**: calculation of the prognostic wind field around or through obstacles (buildings, vegetation, etc.).
- **Step 2 + 3**: these steps compute the dispersion of the pollutants using the wind fields calculated on step 1. In step 2 GRAL calculates the dispersion of transient particles (particles still available from earlier time steps). In step 3 new released particles are computed. These loops are parallelized in their code so part of it is executed in parallel by defect.

When talking about "dispersion" this document refers to steps 2 and 3. These steps are what produces real results, the estimation of pollution in the streets of each city for the days simulated, and are the most important steps during daily simulations (operational simulations for each city).

Due to the importance of these steps the optimization of its computation is one of the important points in this document.

The following factors will be considered during this study, take into account that the final values for a lot of the parameters should be set by each city (depending on their resources, recommendations from environmental experts, etc.):

- **Number of CPU cores** used for the execution.
- **Number of particles used in the simulatio**n: during dispersion pollutant particles are released in the simulation environment and their dispersion is calculated. This parameter stores the total number of particles to be released during the simulation. Note that it means that during the simulation of a particular hour the number of particles inside the domain will vary as particles could disappear and new ones will be added. This parameter could be found inside in.dat file and is related to Dispersion Time parameter. Each city should decide the value for this parameter.
- **Vertical stretching factors**: this value affects in great measure the performance of GRAL. For each simulation a grid (divides the map in sectors) is set, and the stretching factors sets how the grid changes with height: the grid gets coarser with height. With this, CPU and memory needs could decrease significantly. Each city should decide which stretching factors they need to have quality results while meeting their resources availability. This parameter could be found inside GRAL.geb file.
- **Result file compression**: compress (or not) the results of the simulation.
- **Concentration Grid.**
- **Building roughness**. Each city should decide the value for this parameter.
- **Transient Mode vs Steady-state Mode**.

Number of CPU cores used

it is important to decide the optimum number of CPU cores to be used for the dispersion; this simulation will be executed each day as a foresight of the pollution on the following days. For that, we have to analyze the scaling of GRAL related to the number of cores.

For the analysis, we have used the Santiago de Compostela city as input and the following weather conditions (mettimeseries.dat):

04.10,0,1,31.5,6
04.10,1,0.5,33.75,6
04.10,2,0.5,33.75,6
04.10,3,1,33.75,6
04.10,4,0.5,2.25,6
04.10,5,0.5,4.5,6
04.10,6,0.5,6.75,6
04.10,7,0.5,29.25,6
04.10,8,0.5,33.75,6
04.10,9,0.5,0.0,6
04.10,10,0.5,29.25,6
04.10,11,2,27.0,5
04.10,12,2,27.0,5
04.10,13,4,24.75,4
04.10,14,4,24.75,4
04.10,15,6,27.0,4
04.10,16,6,27.0,4
04.10,17,4,29.25,4
04.10,18,2,29.25,4
04.10,19,2,31.5,4
04.10,20,4,33.75,4
04.10,21,2,33.75,4
04.10,22,2,33.75,4
04.10,23,2,33.75,4
05.10,0,2,33.75,6
05.10,1,1,2.25,6
05.10,2,1,0.0,6

05.10,3,1,2.25,6
05.10,4,0.5,6.75,6
05.10,5,0.5,11.25,6
05.10,6,0.5,9.0,6
05.10,7,0.5,9.0,6
05.10,8,1,9.0,6
05.10,9,0.5,11.25,6
05.10,10,0.5,22.5,6
05.10,11,2,29.25,6
05.10,12,2,29.25,4
05.10,13,4,29.25,4
05.10,14,4,27.0,4
05.10,15,4,27.0,4
05.10,16,4,27.0,4
05.10,17,4,29.25,4
05.10,18,4,29.25,4
05.10,19,2,29.25,4
05.10,20,2,31.5,4
05.10,21,1,33.75,6
05.10,22,1,2.25,6
05.10,23,0.5,31.5,6

As explained previously this line corresponds to (example with the last line):

- day.month: 5.10
- hour: 23
- wind speed: 0.5
- wind direction: 31.5
- stability class: 6

Finally, it is useful to represent the speed-up with respect to the number of cores used. The speedup represents the ratio between the CPU time using a single CPU core and CPU time using multiple cores.

$$speedup = CPU\ time\ using\ 1\ core\ /\ CPU\ Time$$

Ideally the speedup should be very similar to the number of cores but in real applications this is not generally the case, so the figure following the table gives us an idea of how far GRAL scales with respect to the number of cores:

Total time for dispersion computation using pre-computed GFF for 2000 particles Santiago de Compostela		
Number Cores	***Elapsed (s)***	***Speedup***
24	325,3	17,01
23	337,8	16,38
22	345,8	16,00
21	386,4	14,32
20	378,1	14,63
19	389,8	14,19
18	412,5	13,41
17	428	12,93
16	457,3	12,10
15	483,1	11,45
14	511,6	10,81
13	549,5	10,07
12	586,3	9,44
11	633,9	8,73
10	691,1	8,01
9	756,5	7,31
8	831,2	6,66
7	931,1	5,94
6	1060,9	5,22
5	1237,6	4,47
4	1468,7	3,77
3	1923,5	2,88

2	2811,5	1,97
1	5532,9	1,00

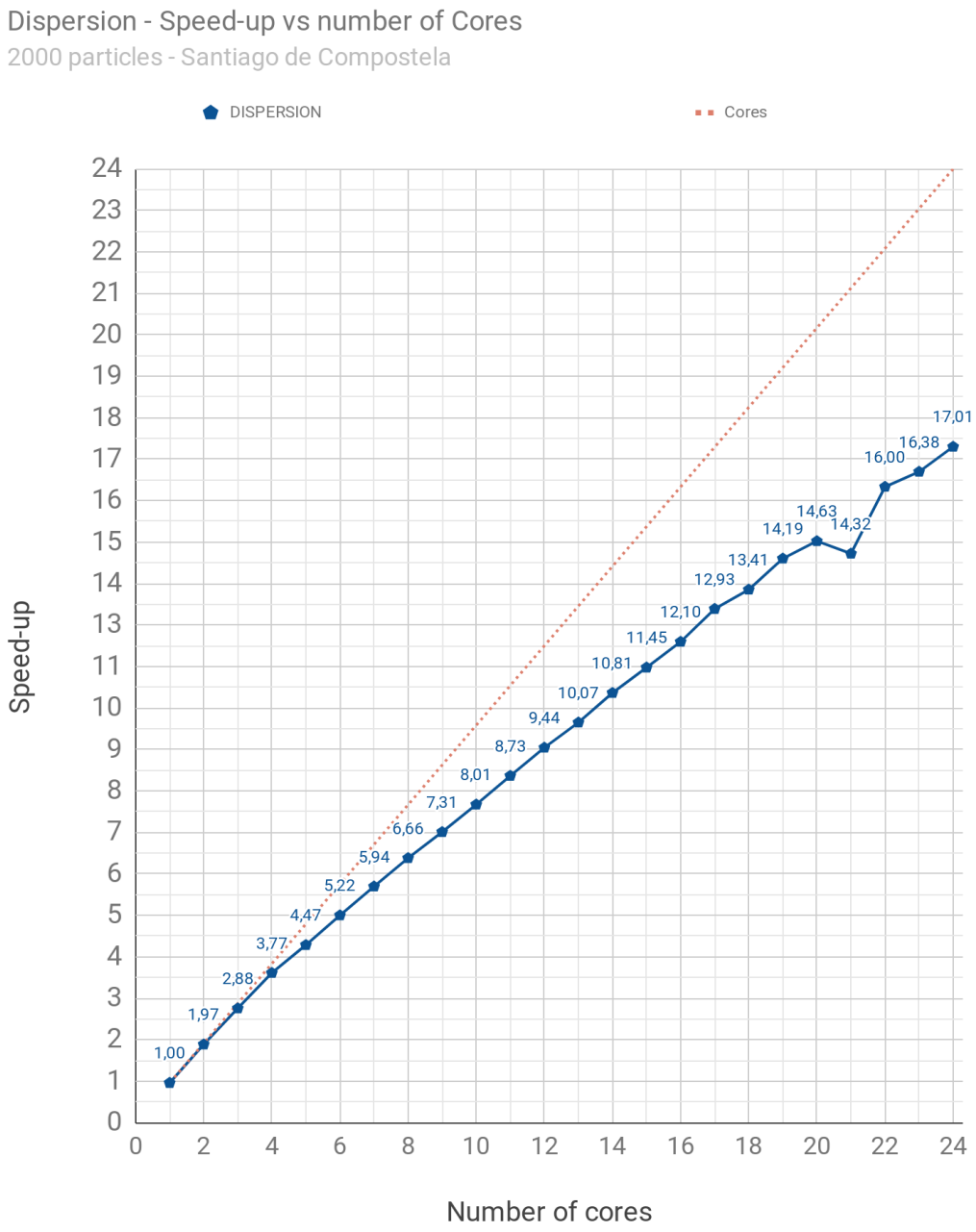

Figure: Speed-up with respect to number of cores for Santiago de Compostela using pre-computed GFF and 2000 particles.

Total time for dispersion computation using pre-computed GFF for 100 particles Santiago de Compostela		
Number Cores	**Elapsed (s)**	**Speedup**
24	66,70	5,15
23	73,30	4,68
22	67,60	5,08
21	66,60	5,15
20	71,90	4,77
19	65,70	5,22
18	67,00	5,12
17	68,30	5,02
16	70,40	4,88
15	72,90	4,71
14	74,80	4,59
13	76,30	4,50
12	74,60	4,60
11	80,90	4,24
10	84,60	4,06
9	83,90	4,09
8	87,50	3,92
7	97,20	3,53
6	104,20	3,29
5	113,80	3,02
4	129,60	2,65
3	152,40	2,25

2	200,00	1,72
1	343,20	1,00

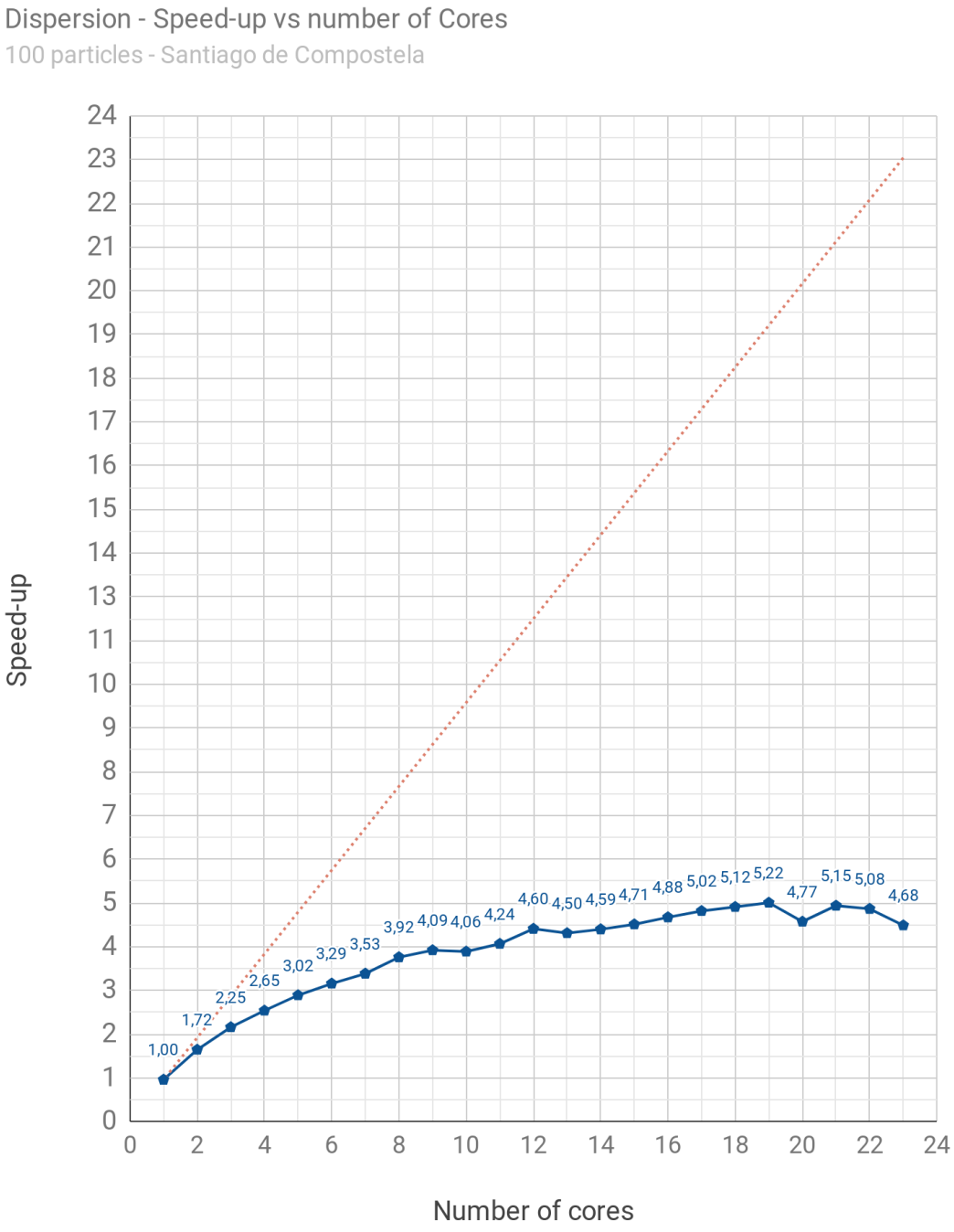

Figure: Speed-up with respect to number of cores for Santiago de Compostela using pre-computed GFF and 100 particles.

Recommendations

The tables and the figures show that the efficiency of GRAL grows from using 100 particles to using 2000 so this value greatly affects the performance of GRAL.

Taking into account this and other benchmarks, the recommendations are (the recommendations for this value should be read in conjunction with the ones for "Number of particles"):

- Best option will be 12 cores, this configuration will allow two executions of GRAL in one node.
- Next best option will be 24 cores, one node in exclusive. This option should only be taken into account if the problem is too big (in terms of time, memory, etc.) to be executed in half a node.

Number of Particles

The effect of the number of particles in the results (Santiago de Compostela)

Number of Particles	Elapsed Time (s)	Elapsed Time (s) Comparison with time for 100 particles	MaxRSS (KB)	MaxRSS (KB) Difference with time for 100 particles
100	73		25893904	
500	128	2 x time for 100	29201356	3307452
1000	189	2.5 x time for 100	28240192	2346288
2000	332	4.5 x time for 100	30004672	4110768
3000	470	6.4 x time for 100	30099564	4205660

Recommendations

The table shows that even if the elapsed time for 2000 particles is 4.5 the elapsed time for 100 particles, this growth in the elapsed time is very small, taking into account that 2000 particles is 20 times 100 particles and that the time did not grow 20 times.

The memory use is only 4 Gigabytes, which for a total of 30 Gigabytes is not important (it is a 1,16% of the memory used by the simulation with 100 particles).

The tables and the figures in "Number of CPU cores used" point show that the efficiency of GRAL grows from using 100 particles to 2000:

- With 2000 particles the speed-up for 24 cores is 17. The speed-up grows almost steadily from 1 core to 24 cores.
- With 100 particles the speed-up for 24 cores is 5 cores. From 9 to 24 cores the speed-up stays between 4 and 6 cores showing a poor performance with this configuration.

Taking all of this into account, the recommendations is 2000 particles from the point of view of time and efficiency when using a node (or even half a node) when executing a simulation in GRAL.

Vertical Stretching Factors

In this case the precomputed GFF are being used for all the calculations for us to be able to value the dispersion performance only.

	Elapsed Time (s)	***MaxRSS (KB)***
standard	79	31769164
Values suggested by Modena 1.5,1.00,20,1.25,50,1.5,150,1.5,250,1.5 !cell-size for cartesian	74	29710804
Values suggested by CESGA	73	25893904

The table shows the elapsed time in seconds; it doesn't grow significantly between different stretching factors used, so the recommendations should be the same as the ones for GFF Precomputation.

Result File Compression

We could assume that in total computing time the type of file compression selected should not have a noticeable influence on the computing time (even the time used in writing and reading the files is very low) but it should be tested in case it affects performance.

The compression options could be set inside in.dat file and are: uncompressed, v1 (deprecated, unused), v2, v3. Developers suggest that V2 should produce the smallest files and that it will be the fastest option due to the reduction in the writing time.

Recommendations

The set for this parameter is recommended to be compression v2 or uncompressed if there is no need of saving space.

Concentration Grids

To calculate the concentration grids there are three main options to tune:

- Horizontal grid resolution: currently using 4m.
- Vertical dimension of concentration layers: currently using 3m.
- Heights above ground where to calculate the concentration grids: currently using just one at 3.5m.

It is interesting to note that in several GRAL examples in the documentation they use 4m horizontal, 0.5m vertical extension and 1m above ground. This parameter depends on the size of the problem, the resolution wanted for the results and other parameters and should be decided between the environmental experts and the IT experts taking into account the resources available.

Surface Roughness

Roughness length in [m]. In case that GRAMM wind fields are used as input, and that the land-use file landuse.asc is available, the roughness length defined here is not used.

It is specified in the in.dat configuration file. Currently we are using 0.2m.

The effect of the surface roughness on the quality of the results and the computation time has still to be analyzed. A validation method for the quality results needs to be established before performing this analysis.

Building Roughness

This is an optional parameter used to set the surface roughness for obstacles used in the prognostic microscale flow field model of GRAL. The default value is 0.001.

It is specified in the building_roughness.txt configuration file. Currently we are using 0.001m (the default value).

The effect of the building roughness on the quality of the results and the computation time has still to be analyzed. A validation method for the quality results needs to be established before performing this analysis.

Transient Mode vs Steady-state Mode

In steady-state mode GFF files are always computed so we can not reuse them.

According to GRAL documentation, GRAL provides two options for the time series computation:

Steady-state mode:

Computation of steady-state concentration fields: In this case particles are tracked until they leave the model domain regardless the time they need to do so. There is no dependence of concentrations on the selected dispersion time.

Transient mode:

Computation of concentrations fields, which are dependent on the averaging time chosen: In this case particles are only tracked until the dispersion time is elapsed. Moreover, the last particle's position is rendered into a three-dimensional concentration field, which is stored for the following weather situation, where this concentration field is again transformed into a particle mass. For each grid element of the three-dimensional concentration field one single particle is released. The concentration grid is based on the Cartesian grid used for the microscale flow-field simulations in the horizontal direction. In vertical direction it uses the height of the first grid cell of the flow-field grid, but has an independent vertical stretching algorithm, which is not adjustable by the user. The grid itself is terrain-following. All these secondary particles share the same properties with regard to deposition and sedimentation velocities for each user defined source group. Emissions can be modulated for each weather situation and source group using the input file "emissions_timeseries.txt".

The transient mode was introduced with GRAL version 19.01.

A validation method for the quality results needs to be established before performing this analysis.

Recommendations

The environmental experts should use the mode needed in each experiment. For daily operatives transient mode is the recommended mode.

How to assess the quality of the results

CESGA could only assess the quality or the results in terms of computational behaviour, efficiency using computational resources and performance. The quality of the results of all simulations in terms of quality of the prediction of pollutant dispersion, the quality of GFF library, etc. should be assessed by environmental experts. This could be done taking into account these values:

- Comparison between simulation results and sensor values. Also if possible the could should be compared with reference stations if environmental experts found it useful.

- When different configurations are applied to GRAL, i.e. different stretching factors, a comparison between results could help assess the quality of the results by the environmental experts.
- If PMSS is being executed a comparison between it's results and GRAL simulations could help assess the quality of the results by the environmental experts.

Conclusions

This study leads to some GRAL's configuration recommendations; these recommendations will be separated in two categoríes: GFF Pre-computation Optimizations and Dispersion Computation Optimizations, as both methods should be executed separately and because of that will have different configurations.

The following recommendations will lead to the best performance when executing GRAL in an environment similar to FTII in CESGA.

GFF Pre-computation Optimizations recommendations summary

- **Number of CPU cores**:
  - Best option will be 12 cores, the reason is that over 12 cores the speed-up will grow very little. This will allow two executions of GRAL in one node.
  - Next best option will be 24 cores, one node in exclusive. This option should only be taken into account if the problem is too big (in terms of time, memory, etc.) to be executed in half a node.
- **Integration Time including Steady-state**:
  - As seen in the figures, the best value for Integration Time (and the one suggested by GRAL programmers) is a maximum of 500. The reason is that convergence goes under 1 when reached that number of iterations but the time spent in all the process will increase significatively if the maximum is set for more than 500.
  - A minimum of 100 is the best option as GRAL will execute a minimum of 100 iterations.
- **Vertical Stretching Factors**: The value suggested by GRAL programmers has a good performance but this parameter could only be set by the environmental experts of each city taking into account the availability of their computation resources.
- **How many GFF computations to group per calculation**: recommends the precomputation of GFF in batches of 2 as minimum and maximum as the resources allow.
- **Number of meteorological conditions to pre-compute**: The environmental experts should take into account the creation of a stability matrix adapted to their city and based on this library should be computed. This value depends totally on meteorological characteristics of the city.
- **Compression Rate**: The time needed for the compression varely affects the total time of the simulation so this value would depend on the resources available.
- **Domain Size**: This value will depend totally on the cities characteristics.

Dispersion Computation Optimizations recommendations summary

- **Number of CPU cores used**: the recommendations for this value should be read in conjunction with the ones for "Number of particles":
  - Best option will be 12 cores, this configuration will allow two executions of GRAL in one node.

    - Next best option will be 24 cores, one node in exclusive. This option should only be taken into account if the problem is too big (in terms of time, memory, etc.) to be executed in half a node.
- **Number of Particles**: Taking all of the studies into account the recommendations is 2000 particles from the point of view of time and efficiency when using a node (or even half a node) when executing a simulation in GRAL.
- **Vertical Stretching Factors**: the recommendations should be the same as the ones for GFF Precomputation.
- **Result File Compression**: The set for this parameter is recommended to be compression v2 or uncompressed if there is no need of saving space.
- **Concentration Grid:** It is interesting to note that in several GRAL examples in the documentation they use 4m horizontal, 0.5m vertical extension and 1m above ground. This parameter depends on the size of the problem, the resolution wanted for the results and other parameters and should be decided between the environmental experts and the IT experts taking into account the resources available.
- **Building roughness**. Each city should decide the value for this parameter.
- **Transient Mode vs Steady-state Mode**: the environmental experts should use the mode needed in each experiment. For daily operatives transient mode is the recommended mode.

Results obtained by each TRAFAIR smart city applying (at least) one of the parallelization strategies

For each TRAFAIR Smart City, the results obtained in terms of timing and computation are detailed. Moreover, for each city will be reported the following information:

- Parallelization strategy adopted
- HPC infrastructure used
- Domain features
- Date from which the parallelized version is running in real time or as a simulation
- Results in terms of timing

Modena

Parallelization strategy adopted: Precomputation of the flow fields
HPC infrastructure used: Centro Tecnolóxico de Supercomputación de Galicia (CESGA)

Parallelization strategy results in Modena:

In the city of Modena, the pre-computation strategy is active from 03-01-2020 to now, in a real time modality. The pre-computation of the "gff files" has been performed by considering 820 different weather situations combining possible wind speed, wind direction and stability class. The pre-computation of the "gff files" has been realized in about 1 week in December 2019 using 4 parallel processes in 2 physical nodes having 24 cores each and 128 GB Ram each.

The following tests here presented, are the same as presented in D3.7 and were conducted in order to validate the speed up of the GRAL simulation considering the pre-processed gff files.

The test was based on the pre-computation of 144 weather situations. The pre-computation of the "gff files" was realized using 16 cores at FinisTerraeII HPC of CESGA using 923:58:24 cores hours of Xeon(R) CPU E5-2680 v3 @ 2.50GHz.

The test was conducted in the area of Modena, as described in the section 'Parallelization strategies comparison in the TRAFAIR cities'- Table1.

Parallelization strategy results in Modena:

As it can be seen in Table 1, **using the pre-computed gff fields reduces the total time of the simulation to about 20% of the original time.**

Florence

Parallelization strategy adopted: Pre-computation of the flow fields
HPC infrastructure used: DISIT

Description of the Disit Infrastructure resources
Cluster of 26 hosts based on multi CPU and multicore XEON on cloud with 10Gbps multiple connections. 4 TB of RAM, 300 TByte of online storage in RAID, a part in SSD and a part in 15Krps, 10Krps. Presently we have also 3800 GPU and we are upgrading at 15.000 GPU in short time.

In the city of Florence, the pre-computation strategy is active from 20-05-2019 to now, in a real time modality. While from 25-02-2019 to 20-05-2019 GRAL was running in the real time modality with the original Open Source code (without the pre-computation strategy adopted). The Florence GRAL map (NOX, 3m and 6m) is visible here:
https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTUzMg==

The pre-computation of the "gff files" has been performed by considering 2376 different weather situations coming from the cartesian product R = V x G, where V is the set of wind speeds and G is the set of the wind directions. More precisely, V= {0.1,0.3, …, 10.1} U {10.5, 10.9, …, 12.1} U {12.9, 13.7, …, 20.1} with |V|=66 and G= {5º, 15º, …, 355º} with |G|=36. The pre-computation of the "gff files" has been realized in about one month from 19-04-2019 to 20-05-2019, using 4 parallel processes in 2 virtual machines having 24 cores at 2.30 GHz and 248 GB Ram each (reducing the original computational cost by 4 times).

The following tests have been conducted in order to validate the speed up of the GRAL simulation considering the pre-processed gff files. The experiments are conducted in the same machine having 80 cores at 2.00 GHz and 251 GB Ram (2 X Xeon 40 COREs).

- **Florence_24h**

The test is conducted in the area of Florence with the characteristics available in the section 'Parallelization strategies comparison in the TRAFAIR cities'- Table1.

Parallelization strategy results in Florence:

As it can be seen in Table 1, **using the pre-computed gff fields reduces the total time of the simulation to just a 33% of the original time.**

Pisa

Parallelization strategy adopted: Pre-computation of the flow fields

HPC infrastructure used: DISIT

Description of the Disit Infrastructure resources
Cluster of 26 hosts based on multi CPU and multicore XEON on cloud with 10Gbps multiple connections. 4 TB of RAM, 300 TByte of online storage in RAID, a part in SSD and a part in 15Krps, 10Krps. Presently we have also 3800 GPU and we are upgrading at 15.000 GPU in short time.

In the city of Pisa, the pre-computation strategy is active from 02-07-2019 to now, in a real time modality. While from 14-05-2019 to 20-06-2019, GRAL was running in the real time modality with the Original Open Source code (without the precomputation strategy adopted). The Florence GRAL map (NOX, 3m and 6m) is visible here:
https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTc2Nw==

The pre-computation of the "gff files" has been performed by considering 2376 different weather situations coming from the cartesian product R = V x G, where V is the set of wind speeds and G is the set of wind directions. More precisely, V= {0.1,0.3, …, 10.1} U {10.5, 10.9, …, 12.1} U {12.9, 13.7, …, 20.1} with |V|=66 and G= {5º, 15º, …, 355º} with |G|=36. The pre-computation of the "gff files" has been realized in about two weeks, starting from 07-06-2019 to 20-06-2019, using 4 parallel processes in 2 virtual machines having 24 cores at 2.30 GHz and 248 GB Ram each (reducing the original computational cost by 4 times).

In the city of Pisa, the following test has been conducted in order to validate the speed up of the GRAL simulation considering the pre-processed gff files. The experiments are conducted in the same machine having 80 cores at 2.00 GHz and 251 GB Ram (2 X Xeon 40 COREs).

- **Pisa_24h**

The test is conducted in the area of Pisa with the characteristics available in the section 'Parallelization strategies comparison in the TRAFAIR cities'- Table1.

Parallelization strategy results in Pisa:

In Table 1, results in terms of timing in the city of Pisa are reported.

As it can be seen in Table 1, **using the pre-computed gff fields reduces the total time of the simulation to just a 47.7% of the original time.**

Livorno

Parallelization strategy adopted: Pre-computation of the flow fields

HPC infrastructure used: DISIT

Description of the Disit Infrastructure resources
Cluster of 26 hosts based on multi CPU and multicore XEON on cloud with 10Gbps multiple connections. 4 TB of RAM, 300 TByte of online storage in RAID, a part in SSD and a part in 15Krps, 10Krps. Presently we have also 3800 GPU and we are upgrading at 15.000 GPU in short time.

In the city of Livorno, the pre-computation strategy is active from 22-07-2019 to now, in a real time modality. While from 03-07-2019 to 22-07-2019 the GRAL was running, in the real time modality, with the Original Open Source code (without the precomputation strategy adopted). The Florence GRAL map (NOX, 3m and 6m) is visible here:
https://www.snap4city.org/dashboardSmartCity/view/index.php?iddasboard=MTgzMw==

The pre-computation of the "gff files" has been performed by considering 2376 different weather situations coming from the cartesian product R = V x G, where V is the set of wind speeds and G is the set of the wind directions. More precisely, V= {0.1,0.3, …, 10.1} U {10.5, 10.9, …, 12.1} U {12.9, 13.7, …, 20.1} with |V|=66 and G= {5º, 15º, …, 355º} with |G|=36. The pre-computation of the "gff files" has been realized in about two weeks, starting from 04-07-2019 to 17-07-2019, using 4 parallel processes in 1 machine having 80 cores at 2.00 GHz and 251 GB Ram (reducing the original computational cost by 4 times).

In the city of Livorno, the following test has been conducted in order to validate the speed up of the GRAL simulation considering the pre-processed gff files. The experiments are conducted in the same machine having 80 cores at 2.00 GHz and 251 GB Ram (2 X Xeon 40 COREs).

- **Livorno_24h**

The test is conducted in the area of Livorno with the characteristics available in the section 'Parallelization strategies comparison in the TRAFAIR cities'- Table1.

Parallelization strategy results in Livorno:

As it can be seen **in Table 1, using the pre-computed gff fields reduces the total time of the simulation to just a 36.6% of the original time.**

Zaragoza

HPC infrastructure used: cluster HERMES, in particular an AMD Opteron (6376) machine of 256 GB RAM, 32 cores and Scientific Linux 5.5

Parallelization strategy adopted: Pre-computation of the flow fields.

In the city of Zaragoza, the pre-computation of the Gral Flow Fields GFF has been adopted. As a first approach, and after an analysis of the historical most-common wind speeds in the city of Zaragoza, the precomputation of the "gff-files" was applied by considering 216 different weather situations. It corresponds to the product of 4 wind speeds (1, 3, 7 and 10 m/s), 18 different wind directions (every 20°, starting from 5°) and 3 different stability classes (3, 4 and 6). In every pollutant dispersion simulation, an output file reports the error/deviation from the real weather conditions. If the error is too large, a new precomputation of "gff-files" will be carried out for these cases.

The city domain selected for the computation of the GRAL simulation has been reduced with respect to the one reported in D3.7 in order to speed up the computations and to be able to use more nodes in the HERMES cluster. In more detail, the current domain for the city of Zaragoza (since December 2019) is the one indicated by the following points: West 671848, East 679476, South 4609612 and North 4617648.

Parallelization strategy results in Zaragoza:
As it can be seen in Table 1, the use of precomputed GFF allowed to reduce the computation time by 40%, approximately.

Santiago de Compostela

CESGA, with the collaboration of the environment experts from Santiago de Compostela, applied the following parallelization strategies to Santiago de Compostela city's simulations using GRAL.

For the city of Santiago de Compostela the following parallelization strategies were applied:

- **GFF Precomputation**: GFF were calculated in parallel in batches
- **Split sources strategy**: a modified version of GRAL was used to run in parallel

In the following table the performance gains of each strategy are shown:

Strategy used	Time (hours)
Normal execution (sequential execution)	10.35
Using pre-computed GFF (sequential execution)	6.41
Split sources (parallel execution)	1.32

In all cases 2000 particles were used. If more particles are used, as recommended by the GRAL developers for large domains, then the simulation times will increase and in this case the need for the split sources parallel approach will be obvious.

The times provided for the split sources approach are for the case where we split the original computation in 9 source groups. If more subgroups are created then the level of parallelism could be increased and the simulation time will be reduced.

Parallelization strategy/strategies adopted: precomputation of the GFF (was done once), the operative executes daily in our supercomputer FTII: one job (one execution of GRAL) with emission sources splitted.

Parallelization strategy results in Santiago de Compostela:

In the "GRAL Performance Optimization" section, CESGA explains all the strategies applied, all the benchmarks are done taking into account that they should be executed in a parallelized system (FTII in CESGA) therefore all of the results shown in the tables and graphics are also parallelization strategy results.

CESGA could only assess the quality or the results in terms of computational behaviour, efficiency using computational resources and performance. The quality of the results of all simulations in terms of quality of the prediction of pollutant dispersion, the quality of GFF library, etc. should be assessed by environmental experts.

D3.7 could be consulted for more information about the results of different parallelization strategies for Santiago de Compostela: "Parallelization strategy results: GFF Precomputation & Dispersion computation", "Parallelization strategy results: Split source group strategy with Santiago de Compostela data", "Divide one source group in smaller parts: and execute GRAL with this configuration (1 job)", "Execute each source group separately and in parallel: Best configuration of this Splitting Source approach".

Conclusions

After the benchmarks, performance optimizations and tests were done all the results were revised and for Santiago de Compostela the best approach was an strategy that uses several parallelization strategies at the same time:

- GRAL's configuration was adapted to offer it's best when parallelized based on CESGA's recommendations.

- Pre-computation of the flow fields: Santiago de Compostela meteorological expertes studied the parameters needed for the computation of flow fields (gff files) and following their advice CESGA precomputed the flow fields suitable for this city.
- Particle Source Splitting: the sources of the particles (emission fonts) were splitted in groups. Currently the results obtained from each group are being studied by Santiago de Compostela meteorological expertes to check if it is necessary to adapt them based on the quality of the results and future needs for the city.

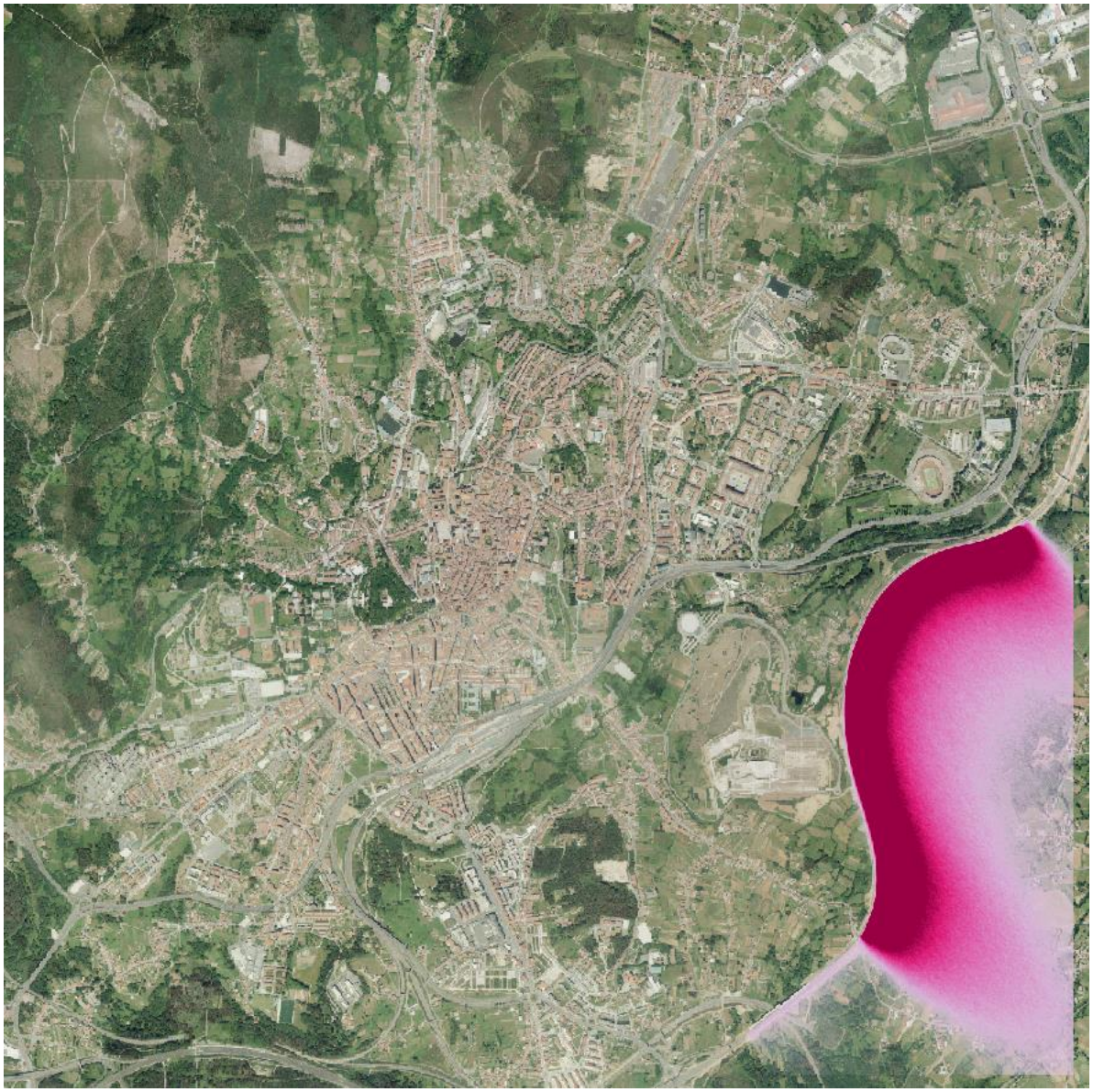

Figure: GFF Source group 14 - Results obtained from a full run.

Parallelization strategies comparison in the TRAFAIR cities

In the following table the results related to the test done for each city is visible.

City		Modena	Firenze	Pisa	Livorno	Santiago de Compostela	Zaragoza
GRAL domain area	West (abs)	647608	676400	611320	604476	534680	671848
	East (abs)	656752	684640	613996	607816	540232	679476
	North (abs)	4942132	4852854	4842794	4825700	4744944	4617648
	South (abs)	4948816	4847830	4839370	4820156	4750248	4609612
	Latitude [°]	44	43	43	43	43	47
Grid dimension	CGCx	2286	2060	669	835	1388	1907
	CGCy	1671	1256	856	1386	1326	2009
	CGSx [m]	4	4	4	4	5542	4
	CGSy [m]	4	4	4	4	5304	4
	CGSz [m]	1.5 + stretch	1	1	1	2,1.00,30,1.05,50,1.15,100,1.25,200,1.5 (stretching factors)	2, 1.00, 30, 1.05, 50, 1.15, 100, 1.25, 200, 1.5 (stretching factor)
	TnHS	1	2	2	2	1	1
	Slice height AG [m]	3.5	• 3 • 6	• 3 • 6	• 3 • 6	3.5	3.5
Settings	SGc	6	1	1	1	9	1
	TnBC	486282	612580	121695	208303	151284	690620
	TnLSS	5340	9883	3132	4073	1743	33191
	TE [kg/h]	28.678	14.177	2.870	15.559	147,227	108.2
	SG 1[kg/h]	5.179	14.177	2.870	15.559	52,605	108.2
	SG 2[kg/h]	7.823				37,949	
	SG 3[kg/h]	2.692				0,621	
	SG 4[kg/h]	7.061				0,067	
	SG 5[kg/h]	4.160				0,007	
	SG 6[kg/h]	1.762				0,017	
	SG 7[kg/h]					0,000	
	SG 8[kg/h]					55,816	

	SG 9[kg/h]					0,144	
	TPU	7200553 (2000 part/s)	3600482 (1000 part/s)	3600088 (1000 part/s)	3600274 (1000 part/s)	7210717 (2000 particles)	18067758 (5000 part/s)
	Gral mode	Transient GRAL mode	Transient GRAL mode	Transient GRAL mode	Transient GRAL mode	Transient GRAL mode (modified 20.01 Beta 2 version)	Transient GRAL mode
	NWS	144	24	24	24	48	24
	PsWS	2000	1000	1000	1000	2000	5000
	DT [s]	3600	3600	3600	3600	3600	3600
GRAL simulation via precomputed gff file	TST [s]	29115.4	13780.7	6002	7762.3	4767	69913
	Dispersion [s]	27451.9	13496	5932.7	7630.8	4063	69384.5
	Flow field [s]	1542.5	284.7	69.3	131.5	132	367.2
	Used cores	16	20	20	20	24	32
	CPU CK (GHz)	2.5	2.00	2.00	2.00	2.5	1.8
	Used Ram (GB)	59.7	40	13	20	32	40
GRAL simulation via none gff file precomputation	TST [s]	208346.0	41680	12571.9	21207.3	37274	114482.4
	Dispersion [s]	354.0	13394.2	5840.2	7734.1	22978	68436
	Flow field [s]	207894.0	28285.8	6731.7	13473.2	13775	45916.8
	Used cores	16	20	20	20	24	32
	CPU CK (GHz)	2.5	2.00	2.00	2.00	2.5	1.8
	Used Ram (GB)	40.9	40	13	20	32	50
GRAL parallel simulation via splitting sources	TST [s]					4778	
	Dispersion [s]					4063	
	Flow field [s]					137	
	Used cores					216	
	CPU CK (GHz)					2.5	
	Used Ram (GB)					32	

HCP Used for the simulations (not all the infrastructure)	2 Haswell 2680 v3 (24 cores) 128GB Ram	2 X Xeon 40 COREs, 251GB RAM	2 X Xeon 40 COREs, 251GB RAM	2 X Xeon 40 COREs, 251GB RAM	2 Haswell 2680v3 (24 cores)	AMD Opteron (6376) 32 cores, 256GB RAM

Where:

- CGCx = Concentration Grid Cell Concentration grid cell nr. in x-direction
- CGCy = Concentration Grid Cell Concentration grid cell nr. in y-direction
- CGSx = Concentration Grid Size in x direction
- CGSy = Concentration Grid Size in x direction
- CGSz = Concentration Grid Size in x direction
- TnHS = Total number of Horizontal Slices for the concentration grid
- Slice height AG = Slice height above ground
- SGc = Source group count
- TnBC = Total number of blocked cells in 2D
- TnLSS = Total number of line source segments
- TE = Total emission
- TPU = Total of particles used within the model domain
- NWS = Number of weather situations
- PsWS = Particles per second per weather situation
- DT = Dispersion time
- TST= Total simulation time