Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

SHAPE Project AmpliSIM

DemocraSIM: Enable air quality Simulation as a Commodity

Olivier Oldrini[*a], Sylvie Perdriel[a], Sylvie Therond[b], Isabelle Dupays[b]

[a] *AmpliSIM,Paris, France*
[b] *CNRS/IDRIS, Orsay, France*

Abstract

AmpliSIM, a French SME expert in numerical simulation services and project partner at IDRIS, collaborated in the SME HPC Adoption Programme in Europe (SHAPE), organised within the PRACE-4IP project. The aim of the project, DemocraSIM, is to enable air quality simulation as a commodity. To do so, AmpliSIM provides air quality modellers with a web service which allows the users to configure / launch / explore air quality numerical simulations without any requirements for software installation or an in-house calculation cluster. In the framework of this project, calculations were performed on the IDRIS supercomputer which allowed fast calculation using parallelisation. Thanks to IDRIS expertise, the AmpliSIM toolset, including numerical models and post processing tools, was improved to meet the standards of a supercomputing parallelised architecture. This led to large time reductions in simulation and post processing. Moreover, the capabilities of the web portal to connect to an HPC infrastructure were substantially improved, including improved security of the portal in order to meet IDRIS standards. Finally, the capability of the web portal to support concurrent user access was strengthened to offer excellent scalability properties.

[*] *Email address of the main author: Sylvie.Therond@idris.fr*

1. Introduction

AmpliSIM is a French start-up based in Paris. Created at the beginning of 2015, AmpliSIM provides an on-demand browser-based Numerical Simulation Service for businesses and public authorities, enabling urgent computing for operational air quality.

To be able to provide the general public with the capability to perform, share and rate simulations, AmpliSIM services needed to be upgraded with:

- Tools necessary to define and launch simulations, according to general public needs, and also to visualise and share simulation results.
- The capability to rapidly produce visualisations from large binary outputs from the numerical model.
- Scalability properties to allow a large number of users to access the service concurrently.
- **Big Data analytics necessary to obtain value from general public simulations and provide additional services to increase the ease of use: recommendations on parameterisations, estimators of simulation duration, model group averages,** and so on.

2. Presentation of the SME AmpliSIM

AmpliSIM's market is impact assessment for air quality and management of industrial hazards. The company provides private users or public authorities with the capability of performing seamless numerical simulations needed for air quality assessment as related to the impact of industrial plants.

The DemocraSIM project targets making air quality simulation available to the general public. When the Fukushima nuclear disaster occurred, it was possible to display crowd-sourced data of radioactivity measurements on a single web map (see http://safecast.org/tilemap and http://blog.safecast.org/history).

Why are people able to do their own radioactivity measurements and share them, whereas they cannot simulate an event and determine their own exposition?

The DemocraSIM SHAPE project will allow AmpliSIM to overcome technological barriers through collaboration with IDRIS engineers in the use of urgent computing, advanced visualisation and data analytics. This will enable them to launch an industrial solution on a private High Performing Computing (HPC) cloud (such as the Fortissimo platform). It will also allow IDRIS HPC experts to address these issues on "real cases" with potential re-use in academic cases of urgent computing.

3. Code porting at IDRIS

IDRIS (Institute for Development and Resources in Intensive Scientific Computing), founded in 1993, is a centre of excellence in intensive numerical calculations which serves the research branches of extreme computing, as much on the application aspects (large-scale simulations) as the research inherent to high performance calculations (calculation infrastructures, resolution methods and associated algorithms, processing of large data volumes, etc.).

IDRIS is the major centre of high performance computing for the French National Centre for Scientific Research (CNRS). Together with the two other national centres, CINES (the computing centre for the French Ministry of Higher Education and Research) and the Very Large Computing Centre (TGCC) of the French Alternative Energies and Atomic Energy Commission (CEA), and coordinated by GENCI, IDRIS participates in the installation of national computer resources for the use of government-funded research which requires extreme computing means. GENCI's role is threefold:

- Implement the national strategy which is to have the three national computing centres providing HPC resources and making the systems available for French researchers.
- Support the creation of an integrated European HPC ecosystem.
- Promote numerical simulation and HPC within the academic and industrial communities.

3.1 Description of the computing machine used

Ada is the IDRIS computer with the most wide-ranging usage. It is composed of large memory SMP nodes (IBM x3750-M4) interconnected by a high-speed InfiniBand network.

Hardware characteristics of Ada:

- 332 x3750-M4 compute nodes, a quadri-socket node of 4 Intel Sandy Bridge E5-4650 8-core processors at 2.7 GHz, with 32 cores per node
- 304 nodes with 128 GiB of memory (4 GiB/core)
- 28 nodes with 256 GiB of memory (8 GiB/core)
- Cumulated peak performance of 233 Tflop/s
- InfiniBand FDR10 Mellanox network (2 links per node
- GPFS parallel file system (WORKDIR) shared by Ada, Adapp and Turing, with a bandwidth of 50 GiB/s in write and in read

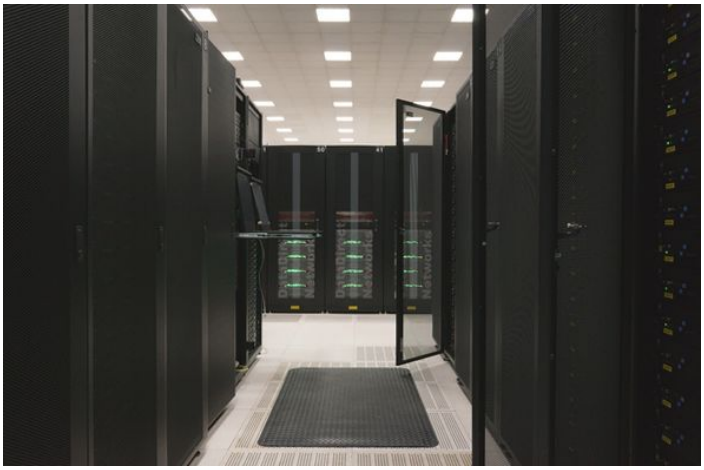

Figure 1: Ada machine

3.2 Work performed

3.2.1 Deployment

The first part of the activity was to deploy all the software and libraries that the project needs on the IDRIS supercomputers:

- Saturne (version 4.0.5 with hdf5 and MED) using the Intel Compiler and MPI Intel library
- OpenFOAM (3.0.1) using the Intel Compiler and MPI Intel library
- GDAL (2.0.1)
- Qt (5.0.6)
- Proj4 (4.8.0)
- Mercurial (4.1)
- Custom models and post processing tools provided by AmpliSIM

Compiling the Fortran/C++ codes with other compilers, such as Intel instead of GNU, highlighted some problems. We corrected them (by adding BIND(C) for the subroutine declarations and not defining a specific subroutine name for a C++ call). This solution works with all compilers.

For example, in some Fortran subroutines, we have a declaration such as:
subroutine allocate_BINFILE(BINFILE_cptr).
We replace that by:
subroutine allocate_BINFILE(BINFILE_cptr) bind(C).
It is the same action for:
subroutine deallocate_BINFILE(CBIN) replaced by *subroutine deallocate_BINFILE(CBIN) bind(C).*

In the C++ part,

```
#if _IFORTLINUX
 //void bincppwrapper_MP_allocate_binfile(ADDRESS *caddr);
 //void bincppwrapper_MP_deallocate_binfile(ADDRESS caddr);
 void allocate_binfile(ADDRESS *caddr);
 void deallocate_binfile(ADDRESS caddr);

and in the include file :
#if defined(_IFORTLINUX)
 //bincppwrapper_MP_allocate_binfile(&_data_ptr); // Allocate Fortran derived type
 allocate_binfile(&_data_ptr); // Allocate Fortran derived type

#if defined(_IFORTLINUX)
   //bincppwrapper_MP_deallocate_binfile(_data_ptr); // Deallocate Fortran derived type
   deallocate_binfile(_data_ptr); // Deallocate Fortran derived type
```

Using *#if defined(_IFORTLINUX)* provides a solution which works with all compilers.

3.2.2 Job submission

Python profiling was implemented in order to resolve any possible bottlenecks in the application.

To test simulation with a portal, we decided to create a small test case with a multi-step job. The user needed to first launch a sequential job (sequential data transfer on the supercomputer), then a second job of parallel calculation (OpenFOAM) and then, a third job which is again sequential (post processing of the previously generated files). The final job is a sequential one (transfer of the data on the local machine). It would be regrettable to reserve cores for the whole chain when they are only necessary for the parallel execution. This would not only be detrimental to the efficient utilisation of the machine but also penalise the user who would be billed for the number of cores reserved multiplied by the total Elapsed time for each job.

It is also necessary to specify the dependency directives if you want to ensure that the different steps are executed correctly, one after the other. Without these directives, all the steps begin independently as soon as the necessary resources are available. In our case, the # @dependency directive of the *submit_postprocessing*, *run_mbutil* and *run_statistics* steps indicates that these must not execute until the preceding step has terminated without error (return value equal to zero).

Submission of the whole chain also requires some information concerning the batch job consumption in order to send to the portal:

- Qdate: date and hour of job entry in the Loadleveler queue
- Bdate: date and hour of job execution start
- Edate: date and hour of job execution end
- tEse: Elapsed time consumed in seconds and in "hours, minutes, seconds"
- tCpu: CPU time consumed in seconds and in "days, hours, minutes, seconds"
- Data+Stack: Data and Stack memory requested (in megabytes)
- MAXRSS: maximum memory used by the job (in megabytes)
- #T: number of tasks or processors used
- (%): Job efficiency rate ==> tCpu*100/(tEse*#T)
- S:
    - C (completed) ==> job completed normally
    - R (removed) ==> job destroyed during execution (by using the llcancel command, for example)

The entire job file submission process can also be adapted to other parallel job schedulers like Slurm which means it is portable on other supercomputers.

3.2.3 Improvement of AmpliSIM modelling suite

The various components of the AmpliSIM modelling suite are called using the Loadleveler multistep process. Tests performed on the IDRIS supercomputer displayed bottlenecks related to the parallelism of specific steps.

The intensive computing steps are:

1. Numerical modelling
2. Tiling of numerical results (post processing with TILER code)
3. Reordering and database storage of tiles (post processing)

Step 1 was parallelized quite efficiently while steps 2 and 3 were largely improved.

Due to scaling to a larger number of cores, compared to calculations performed before the PRACE/SHAPE experiment, some limitation of steps 2 and 3 became obvious.

Since step 3 was only scalar before the project, parallelism had to be be introduced to reduce the bottleneck related to this step. Being pure Python, this step now makes use of MPI through the MPI4PY library. Database feeding can now be distributed among available cores. The database used is *SQLite*. The number of *SQLite* databases being created has now reached its limit for parallelism since *SQLite* does not support concurrent writing.

Step 2 used compiled code in C++. MPI instructions have been implemented for a long time in this step. Still, some limitations became obvious during the project. This step results in a large number of inodes since tiles are stored in directories defined by the x and y locations and also the zoom level. Creation of directories to store the images was coded without any particular care regarding concurrent access for creation of directories. Indeed, a directory was created when a tile did not yet have a directory on disk. In the framework of the project, and due to a larger number of available cores, this led to concurrent access for creation of directories: The probability was higher of having two different cores generating tiles which had to be stored in the same directory, not yet created. Due to a large number of inodes created, there were situations where a directory was created by one core just after another core detected that it was not yet created and decided to create it itself. Therefore, to prevent any overlap, the TILER code needed to be modified to centralise this task in a specific routine and distribute it among cores.

Because of these improvements, parallelism is now available throughout the intensive computing steps. Migration and deployment to other high performance computing centres is now easier.

```
#=========== Global directives ===========
# @ job_name = simmodel
# @ output   = $(job_name).$(step_name).$(jobid)
# @ error    = $(output)
#=========== Step directives ===========
# @ step_name = run_simmodel_model
# @ job_type = serial
# @ parallel_threads = 4
# @ environment = NB_TASKS=$(parallel_threads)
# @ wall_clock_limit = 0:30:00
# @ queue
#=========== Step directives ===========
# @ step_name = submit_postprocessing
# @ dependency = (run_simmodel_model == 0)
# @ job_type = serial
# @ wall_clock_limit = 0:30:00
# @ queue

SIMPATH=$WORKDIR/_SIMU/
INPUTFILE=$SIMPATH/simmodel.tgz
EXEDIR=$WORKDIR/_SIMMODEL_GDAL2/EXELINUX
MODEL=simmodel
URLUPDATE=…
TOKEN=…
export PYTHON_EGG_CACHE=$WORKDIR/.cache
export PYTHONPATH=$PYTHONPATH:$WORKDIR/PYTHON

case ${LOADL_STEP_NAME} in
  run_simmodel_model )
    set -x
    python $WORKDIR/PYTHON/ASMODELS/ASRunModelServerSide.py $INPUTFILE 10 7
$EXEDIR $SIMPATH $SMPATH $MODEL $URLUPDATE $TOKEN -nbprocs $NB_TASKS -step
1
  ;;

  submit_postprocessing )
    set -x
    llsubmit ${LOADL_STEP_INITDIR}/simmodel_job_multistep-IDRIS-mpich.ll
  ;;
esac
```

Figure 2: Simple example of job submission for the whole process

```
#========== Global directives ==========
# @ job_name = calx
# @ output   = $(job_name).$(step_name).$(jobid)
# @ error    = $(output)
#========== Step directives ==========
# @ step_name = run_tiler
# @ job_type = mpich
# @ total_tasks = 32
# @ environment = NB_TASKS=$(total_tasks)
# @ wall_clock_limit = 0:30:00
# @ queue
#========== Step directives ==========
# @ step_name = run_mbutil
# @ dependency = (run_tiler  == 0)
# @ job_type = mpich
# @ total_tasks = 16
# @ environment = NB_TASKS=$(total_tasks)
# @ wall_clock_limit = 10:00:00
# @ queue
#========== Step directives ==========
# @ step_name = run_statistics
# @ dependency = (run_mbutil == 0)
# @ job_type = serial
# @ wall_clock_limit = 0:30:00
# @ queue
SIMPATH=$WORKDIR/_SIMU/
INPUTFILE=$SIMPATH/calx.tgz
EXEDIR=$WORKDIR/_CALX_GDAL2/EXELINUX
MODEL=calx
URLUPDATE=url
TOKEN=token
export PYTHON_EGG_CACHE=$WORKDIR/.cache
export PYTHONPATH=$PYTHONPATH:$WORKDIR/PYTHON

case ${LOADL_STEP_NAME} in
 run_tiler )
   set -x
   python $WORKDIR/PYTHON/ASMODELS/ASRunModelServerSide.py $INPUTFILE 10 7
$EXEDIR $SIMPATH $S
IMPATH $MODEL $URLUPDATE $TOKEN -nbprocs $NB_TASKS -step 2
 ;;
 run_mbutil )
   set -x
   python $WORKDIR/PYTHON/ASMODELS/ASRunModelServerSide.py $INPUTFILE 10 7
$EXEDIR $SIMPATH $S
IMPATH $MODEL $URLUPDATE $TOKEN -nbprocs $NB_TASKS -step 3
 ;;
 run_statistics )
   set -x
   ls
 ;;
esac
```

Figure 3 : Simmodel_job_multistep-IDRIS-mpich.ll

4 Benefits for the SME AmpliSIM

4.1 Benefits for AmpliSIM modelling tools

The post processing of the numerical models binary outputs for visualisation is a current challenge for our web service. The obligation is to be able to rapidly visualise large amounts of data. To meet the requirements of both the size of the data and the rapidity, we have developed our proprietary post-processing suite.

This suite, called *TILER*, relies on a tiling process, as used in the web geographical information system: Geographical data are defined at different zoom levels and sliced in tiles. These tiles are then loaded for visualisation according to the location and the zoom level being explored by the user. *TILER* enables AmpliSIM to visualise very large computational domains.

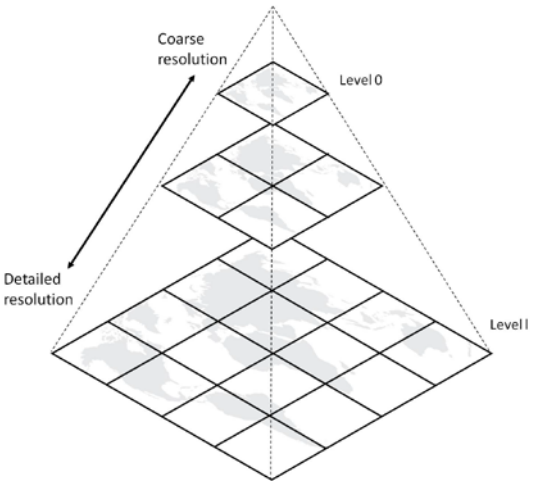

Figure 4: Example of the tiling and pyramidal process for data visualisation

TILER is parallelised using MPI to enable AmpliSIM to rapidly produce images from binary model outputs. With the help of the IDRIS team, we were able to optimise the *TILER* suite which included improving compilation options and solving parallel paradigm issues related to file access and directory creation (see section 3.2.3, § *Improvement of AmpliSIM modelling suite*). We also received essential help in successfully increasing the portability of the TILER suite.

4.2 Benefits for the AmpliSIM connection backend

The workflow of AmpliSIM services consists of several steps:

- Case setup for the numerical model
- Uploading of input data on the IDRIS cluster
- Launching the simulation
- Retrieval of post processed data from the IDRIS cluster

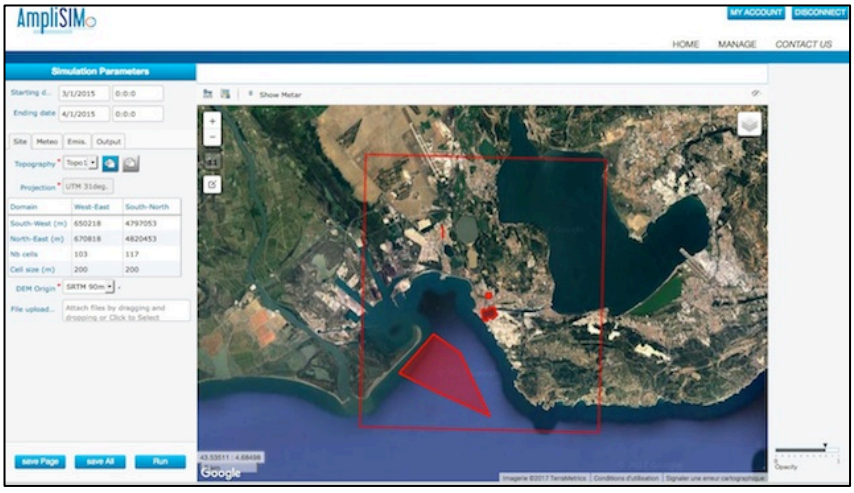

Figure 5: Example of case setup using AmpliSIM web service

The connection on the IDRIS infrastructure within the framework of the DemocraSIM project allowed us to upgrade our connection backend. This backend allowed the web portal to drop input files on the cluster, launch simulations and retrieve the output.

The AmpliSIM Connection Backend (ACB) relied on an *ssh* connect to the IDRIS cluster. We had to improve the modularity and security of the ACB in order to meet IDRIS standards.

Regarding modularity, we had to generalise ACB launching scripts and introduce the IBM LoadLeveler (LL) process. In addition to being able to link launching steps thanks to the LL step properties, AmpliSIM is now able to connect to other LL type clusters. IDRIS security constraints forced us to strengthen SSH options used by the ACB. This is particularly important for our business: Data handled by AmpliSIM web services remain the property of AmpliSIM customers and are often considered as sensitive. For example, pollution levels of an industrial facility are considered to be very sensitive for our customers and could lead to distress among the general public if the levels were communicated without the necessary context of the study. The ability to assure our customers of the highest level of security for their data, even when moving them outside of our web servers to the calculation backend, is mandatory for our business.

4.3 Benefits for AmpliSIM web portal

We also adapted our platform to meet the project obligations: In order to allow for scaling of different services provided by the platform, we had to increase modularity in the component. To do this, we introduced Docker containers to split the portal into different components which can be scaled independently.

Security of the web services had yet to be defined in a manner which would allow for a Single Sign On process: A user of the portal is prohibited from identifying himself more than once. To achieve the Single Sign On process, we had to introduce Central Access Service (CAS) within the portal. The user, once signed in, has the ability to access the different web services provided by the portal without any additional authentication while at the same time assuring protected access to the data.

5. Conclusion

The IDRIS team helped us to plug our web services into the IDRIS supercomputer and perform simulations from the AmpliSIM web service:

- Define the test case and prepare the necessary input files, including the submission scripts.
- Upload the files on the service.
- Run the calculation and post process the output.
- Download the processed output.

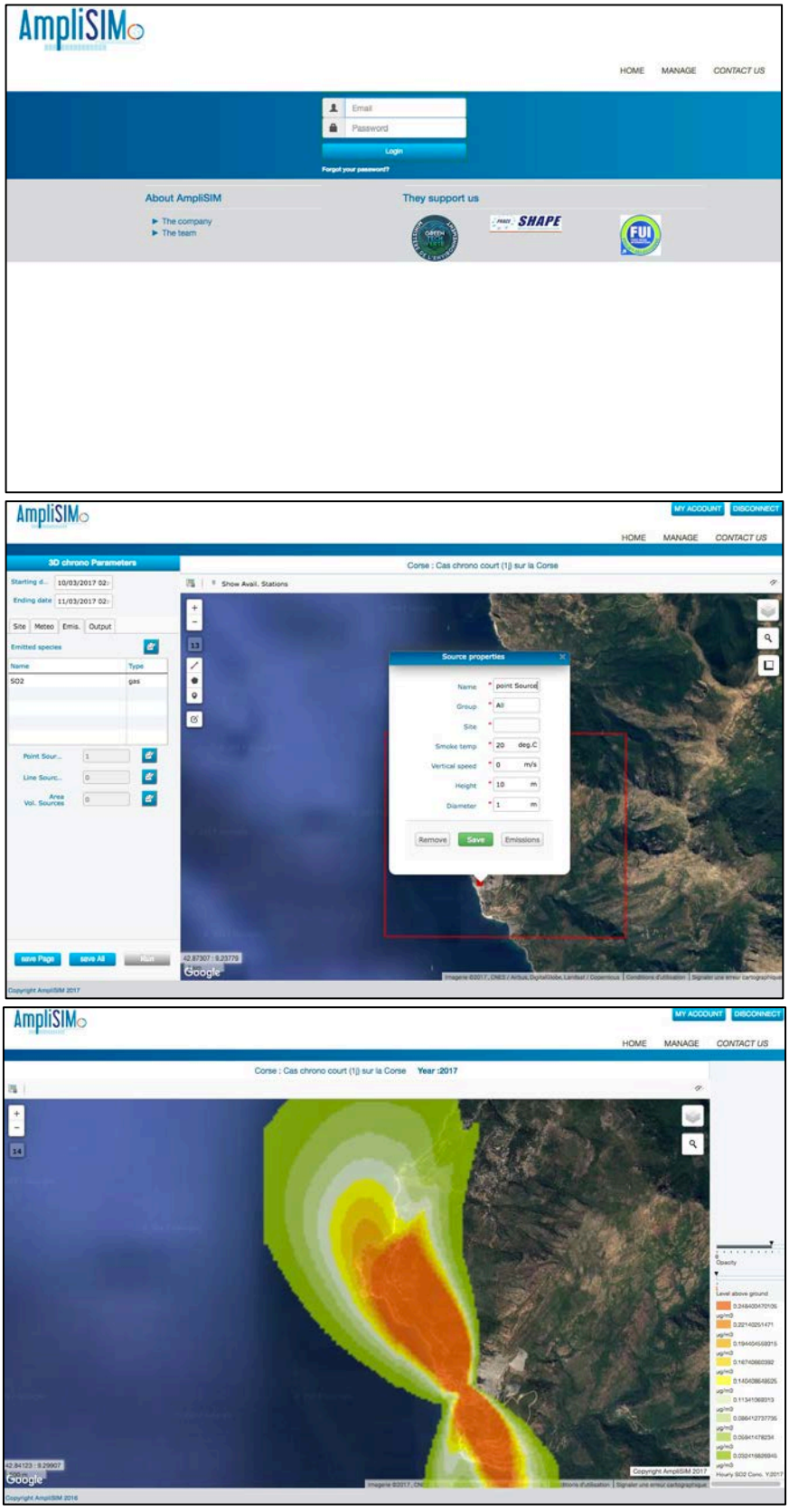

Figure 6: Example of a case run on IDRIS supercomputer. From top to bottom: login on AmpliSIM service, definition of the setup and visualisation of the simulation, visualisations retrieved from the supercomputer.

Thanks to the expertise of the IDRIS team, we were able to:

- Increase the parallel capabilities, and hence the speed, of our post processing suite.
- **Improve the security and the scalability of our web service.**

Due to time constraints, we were, unfortunately, unable to achieve the last objective of the project. This objective, big data analytics on simulation results to offer additional services to the users, would require an additional follow-up to the project. Indeed, capabilities offered by big data analytics would allow non-expert users, such as general public users, to define a setup more easily.

Acknowledgements

This work was financially supported by the PRACE project, funded in part by the EU's Horizon 2020 research and innovation program (2014-2020), under grant agreement 653838.