Computing and Informatics, Vol. 31, 2012, 161–171

OPTIWEB: AN OPTIMIZATION APPLICATION FOR STEEL CUTTING INDUSTRIES PORTED TO THE GRID IN THE FRAMEWORK OF PIREGRID PROJECT

Jaime Ibar, Gonzalo Ruiz, Ruben Valles

BIFI: Instituto de Biocomputación y Física de Sistemas Complejos
Universidad de Zaragoza
50018 Zaragoza, Spain
e-mail: {jibar, gruiz, rvalles}@bifi.es

Alfonso Tarancón

BIFI: Instituto de Biocomputación y Física de Sistemas Complejos
Universidad de Zaragoza
50018 Zaragoza, Spain
&
Departamento de Física Teórica
Universidad de Zaragoza
50009 Zaragoza, Spain
e-mail: tarancon@unizar.es

Abstract. PireGrid [1] (Project Number EFA35/08) is an INTERREG IV A project which has two main objectives, namely the deployment of a production Grid Computing infrastructure in the regions of Aragon, Aquitaine and Midi-Pyrenees and the achievement of successful cases in execution of applications from small and medium size companies of the named regions, in order to demonstrate its usage of these emerging technologies to the companies. In the framework of this project we present Optiweb, which is the first successful porting of an application from Schnell Software [2] company, and the process followed to adapt it to a web interface and its transparent execution in a gLite grid infrastructure.

Keywords: Grid, optimization, company, python, glite

1 INTRODUCTION

In any building work, taking the rebar element lists (like beams, piles and floor structures) as a starting point, a computerized procedure is performed in order to take data, process it, cut the elements and classify them. The steel segments cutting optimization is the most demanding part in computer resources terms. In fact, Schnell Software and the University of Zaragoza have been collaborating for 9 years in the development of internationally competitive software based on techniques used to study complex systems. Optimo is the result of this close collaboration, a product commercialized in more than 20 countries.

Last year, some facts made us think about changing some important aspects of the application basis. On one hand, the product needed to take part in more steel demanding markets, more computational demanding markets like Obra Publica, or other cutting systems from Latin America. On the other hand, BIFI took part in GRID technology, which would offer some performance improvements to this kind of application executed at that moment in the clients computers. Thus, we decided to turn it into a distributed and more efficient one. This very ambitious aim called Optiweb would allow Schnell clients to optimize their steel cutting on a distributed environment, with no CPU and storage limits like before, and with no need for these clients to buy dedicated high power computers. This would reduce their costs, and also it would permit them to get better results in order to save steel, money and energy, reducing the environmental impact of this process.

This project was the first successful case of an Aragonese company using GRID technologies and was awarded with the first prize of the 2011 research transfer contest organized by the University of Zaragoza. Currently, Schnell Software has granted access to this application to its more important clients in order to show them how powerful it is, and in the coming future is planing to charge them for using it selling licenses.

In order to satisfy the company needs and give it an available product as soon as possible, this project was divided into two phases:

1. Adapting local Optimo to a web environment, Optiweb. This would permit the company clients to run their optimizations from anywhere, using any device which had a browser and an Internet connection (like desktops, notebooks, netbooks, smartphones, and so on). This would remove the computing load that their computers had before, because it would be supported by the application server. This server had not power enough to deal with a lot of clients running their optimizations at the same time, and that was the reason why we planned another phase to introduce GRID technology on it.

2. Porting the optimization process to a working GRID. For this purpose, the PireGRID platform was chosen. It has hundreds of cores available for running optimizations. This would allow Schnell Software clients to perform better optimizations using less time than in their own dedicated computers.

2 PHASE I: THE WEB APPLICATION

The original application is a command line C program developed by the researcher Alfonso Tarancon. Thus, the first study done was to look for the best way of adapting it to a web based application. Due to Schnell Software company mainly to works with Microsoft environments, we tried to make a first approach using ASP technology, but it was soon ruled out due to the program complexity and resource consumption. As BIFI has deep experience in Linux and free software, these could speed up the development process, so the technologies finally used are described below.

Once the platform where the application would be executed had been chosen, different technologies were considered for the final implementation of the solution. Although a UNIX based operating system would be used, we tried to use multi-platform technologies, because it may be necessary in the future to migrate it to a Windows Server belonging to Schnell Software. The three main possibilities which arose were Java, PHP and Python [6]. The last one was finally selected because its easiness of integrating it with other programming languages, the existence of an API to access the grid resources and the BIFI previous experience in the development of all kinds of applications with this technology.

With this topic decided, the next step was to face the implementation of the web interface, including all the features that Schnell Software required during the analysis. The used server is Pylons [7], which is one of the most extended servers for Python development. The system was designed from scratch to be compliant with all the browsers of the different platforms, including mobile ones. JSON [5] has also been used, which is an increasing client-server technology. It allows to speed up the communications and make lighter systems without being necessary to reload entire web pages with each operation. With this objective the jQuery [8] framework was also selected, which facilitates all the previous tasks mentioned. In order to store all the data, a MySQL [9] database was developed ad hoc for the project. The main features grouped by sections are as follows:

Protection: a user can access the system with the user and password or access the registration web

Register: a user can create an account to use the system introducing different data, some of them are compulsory to trace the usage of the system by means of Schnell Software.

Main: a user can configure and submit optimizations of the cutting process. It has the following subsections:

- **Input:** where language can be selected (now available in Spanish and English although Italian, German, even Japanese will be available soon) as well as the metric system in which data is entered.
- **Data:** the user adds the input data, that is, the steel bars needed to form the different elements of rebar. They can be introduced both manually and by loading a file.

Machine: the user configures the cutting machine, the optimization parameters and the stock of bars in the store.

Standard cut: some advanced options can be added to optimize a special cut.

Optimization: the execution is released locally or GRID and in which the progress of the optimization with partial results of best solution found so far.

Results: final results are shown and the user can print or download files that go directly to the cutting machines.

My account: the user can edit data as well as a historical list with all optimizations, see the way they ended and download input, configuration and result files.

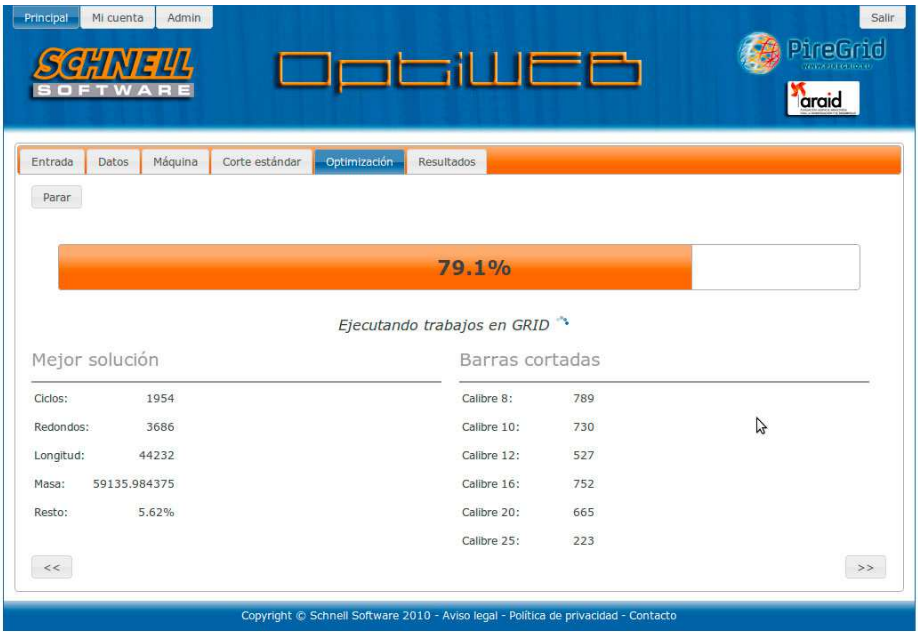
Admin: this section is only accessible for system administrators, that is, Schnell Software staff and BIFI. It is possible to get a list of all users and to change their permissions in a way that a user can be disabled in case of misuse of the application, to change the privileges only to local version, to give permissions to execute in GRID or to make the user administrator. There is also a list with all the system activities as well as the optimizations done by all the users.

The next step was the integration of the binary program in charge of the optimization process. Some modifications were necessary in order to allow the communication between the program and the web application, and this way being able to trace the status of the execution. Each optimization runs in an independent thread on background, letting the concurrency of optimizations from different users at the same time. The major problem of this so-called local version, is the performance degradation if the number of optimizations/users increases remarkably. The input and output data are stored in different directories depending on the execution in order to avoid interferences between the optimizations and also to allow the availability of the results once the machine operator needs them.

3 PHASE II: PORTING TO THE GRID

This phase consisted in the adaptation of the application to the GRID infrastructure and its integration in the web site previously developed. The application is a stochastic optimization, which means that the results of each execution may vary independently of using the same input parameters. So, the bigger the number of execution the higher is the probability of finding an optimum solution. This is the key point of the advantage of using a grid infrastructure, because it allows to run hundreds of optimizations simultaneously, getting a better solution and in less time compared with a unique computing machine.

To achieve the porting, the first step was to test the application in a standalone version. This was possible making some modifications to the source code, compiling again, submitting it as batch jobs and checking the integrity of the obtained results.



79.1%

Ejecutando trabajos en GRID

Mejor solución	
Ciclos:	1954
Redondos:	3686
Longitud:	44232
Masa:	59135.984375
Resto:	5.62%

Barras cortadas	
Calibre 8:	789
Calibre 10:	730
Calibre 12:	527
Calibre 16:	752
Calibre 20:	665
Calibre 25:	223

Copyright © Schnell Software 2010 - Aviso legal - Política de privacidad - Contacto

Fig. 1. In this image we can see a real optimization executed in GRID. The system shows us the progress of the simulation, as well as the best solution results till the moment. We can observe that the amount of managed data is quite big because the solution found was 3 500 steel bars in a casting of 60 Tm

The following issue was to develop a Python script which would deal with all the complexity of authentication, communication, checks, etc. with the grid. In order to deploy it and have a full control of the jobs submission the gLite APIs [3, 4] has been used executed from a User Interface, which is the machine in charge of finally sending the optimization jobs to the grid. This script is, in fact, the mediator

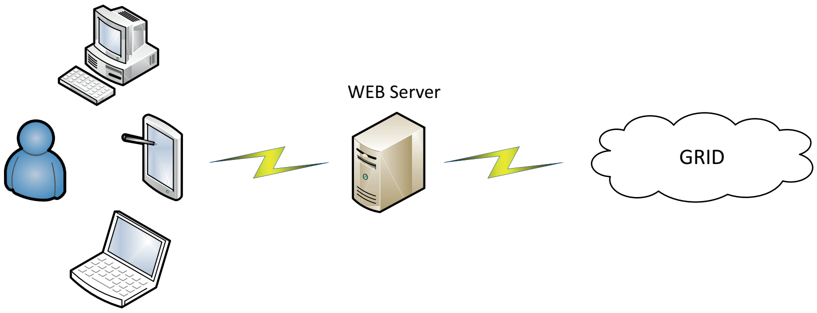


Fig. 2. Scheme of the system running with all its actors

between the web application and the supercomputing platform, which in our case is a gLite grid. From the web interface, in a transparent way for the user, the input parameters and the configuration for the job submission are sent to the script, as well as the number of job optimizations that we want to be done in a parallel way.

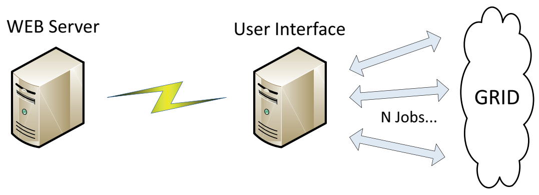


Fig. 3. Scheme of the application running in grid mode

The Python script is in charge of sending submitting the jobs to the PireGrid WMS, which selects the final execution environment depending on the workload of each node. It checks the status of the job, warning in case of any failure, and it is also responsible of automatically collecting the data of all the succeeded jobs. Once it has compiled all the outputs, it starts a local program to select the best solutions among all of the ones computed in the grid sending it back to the web server and showing it to the final user.

In this way, the final operator does not feel any difference selecting the local or the GRID process. It is just a matter of configuring the input data via web, pressing a button to launch the optimization and it is the system which deals with the issues to obtain the final result and show it to the user, provided that GRID results will be better in most of the cases due to the fact that the space of solutions is much bigger.

Obviously, the security in the approach of communication has been handled with secure protocols, always sending the information encrypted, which is one of the main requirements of data used by companies.

3.1 How the Web Server Works

The web server is the mediator between the operator using the application and the user interface. There are three main steps in the use case of the application.

- The job submission: The user inputs the data filling up the web forms or uploads an input file with all the initial parameters for the application. Then the web server prepares a tar file which is sent via ssh to the User Interface that contains the Python script, which is invoked and started.
- The execution: During execution of the jobs in the grid the user needs to know how the application is progressing, but all the complexity of the computing platform is hidden for him/her. This way, s/he is not the one who asks how everything is going on, but it is the web server which reads the status file from

the User Interface to know if the execution is yet in progress and how many jobs have already finished to update the status showing the percentage of the optimization task that has been completed.

- The results: When the status bar reaches 100 % means all grid jobs have finished properly and the final output has been copied from the UI to the Web Server, so the final results can be formatted and be shown in the web page to the operator.

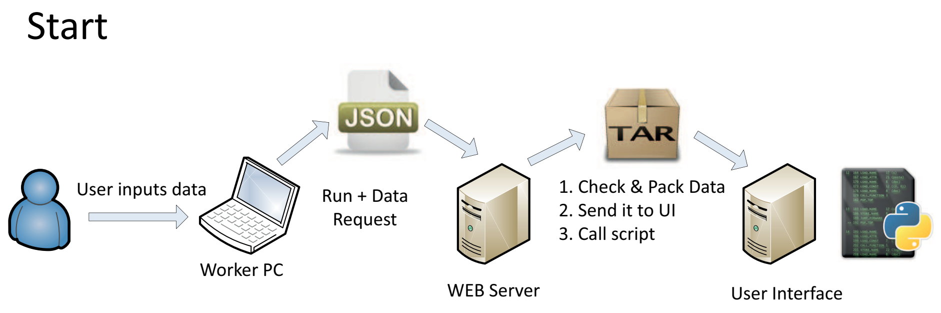


Fig. 4. Communication workflow client computer, web server and User Interface starting the optimization

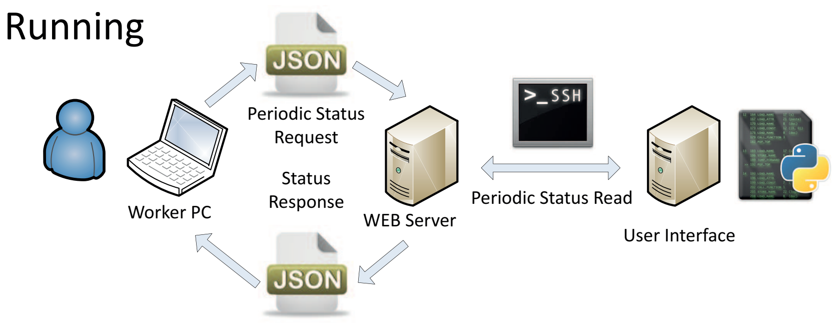


Fig. 5. Communication workflow client computer, web server and Python script while the jobs are running in the grid

3.2 How the Python Script Works

First of all, it checks whether a valid voms proxy exists, in order to be able to get authenticated in every operation with the services of the grid infrastructure. Only in case it detects there is no valid proxy enabled it creates a new one.

After that, the jdl file with the description of the job is generated, according to the specification of the parameters that the web server has sent to the script as

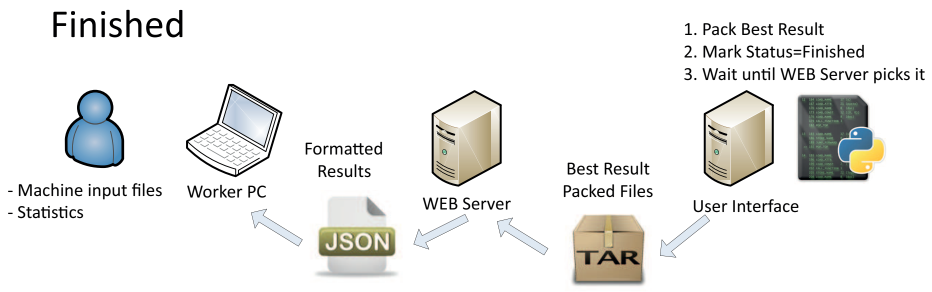


Fig. 6. Finished status: retrieving the output results from the grid to the web server

input parameters. In the case of this application, also an optimization of the way the jobs are sent to the grid has been performed. It consists in generating a jdl file of type collection, which lowers the load of the WMS service, because it only receives one big job composed of several ones and the input data is only transferred once. This approach is possible because, once the parameters are fixed in the web server interface, they are the same for all the jobs sent to the grid.

Having created the jdl file there are two options depending on the InputSandbox:

- If there are no files in the InputSandbox, only a jobStart is required, so the job will be directly submitted to execution.
- In our case, there are input files, so there is a previous step to be done. First we have to do a jobRegister to obtain the id of the job collection and the url where the input files will be stored in the WMS (of the type `gsiftp:///path_to_input_files`).

This URL is then used to uploaded the input file to the WMS using the lcg_util API. As they are quite small and as using the job collection approach they are transmitted only once, the usage of the Storage Element Service is not required.

When the input files have been properly uploaded to the WMS, the jobStart can be done in order to submit the job to the WMS. The later is obviously the one in charge of deciding in base of his algorithms to decide the most suitable site to execute the jobs.

The jobStart command returns the list of the unique ids of the jobs which conform the collection. This list is constantly used in the main loop of the script, because once they have been submitted they are treated as independent jobs and not as a collection.

The main sequence of tracing the job execution is checking the status of all the job ids. In case of its termination, the output is retrieved from the WMS and it is deleted from the list to avoid future checks. At the same time, in each iteration the status file where the execution profile is stored is updated with the correct number of jobs sent, running, done, etc.

When the number of finished jobs equals the input parameter received from the server, the execution of the Python script finishes and the control is delegated to the web server.

One of the most interesting advantages that we have using the Python API is the abstraction of the command line, the simplicity and speed to develop this kind of script, but above all the complete control that you have at low level to interact with the different services and steps of the workflow of submitting jobs and handling their data.

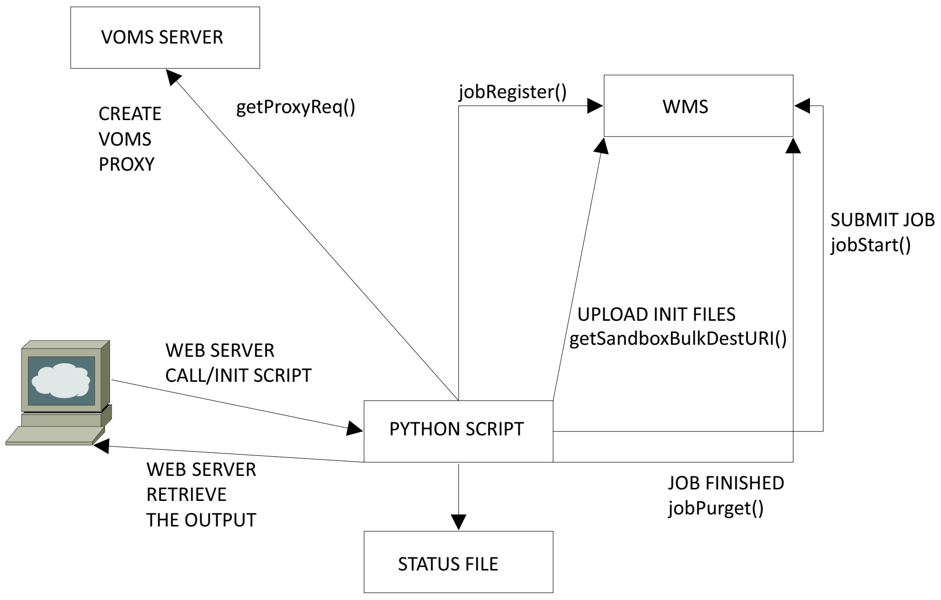


Fig. 7. Scheme of the communication of the Python script with the different services of the grid infrastructure and the web server

4 ANALYSIS OF RESULTS

Below we show two graphs where the improvement in the performance of the implemented system solution can be observed. In the first one we can see the execution of 1 000 optimization jobs in the grid for a casting of 60 Tm. In the y axis we observe the percentage of remains, and in the x axis the number of jobs. The lower the point the better the solution. This example would be equivalent to 1 000 local executions, which would last for a very long period. If we only executed one local solution the probability says that it would be very close to the mean, represented with a blue line. Thanks to the power of the grid we can observe a 2.5 % of improvement, which in our example would mean a saving of 2 Tm of steel, storage, transport, time, etc.

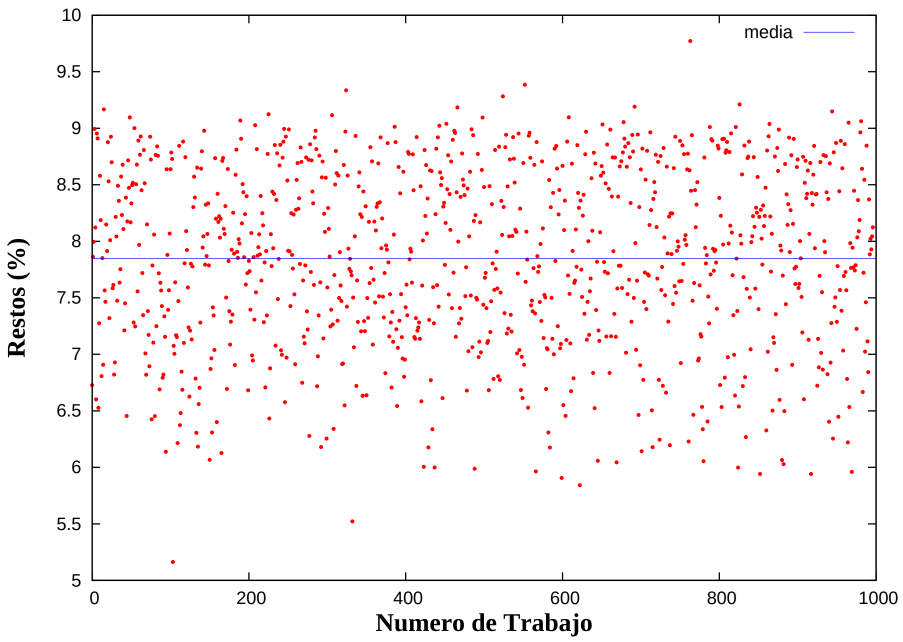


Fig. 8. Execution in GRID of 1 000 optimizations for a casting of 60 Tm

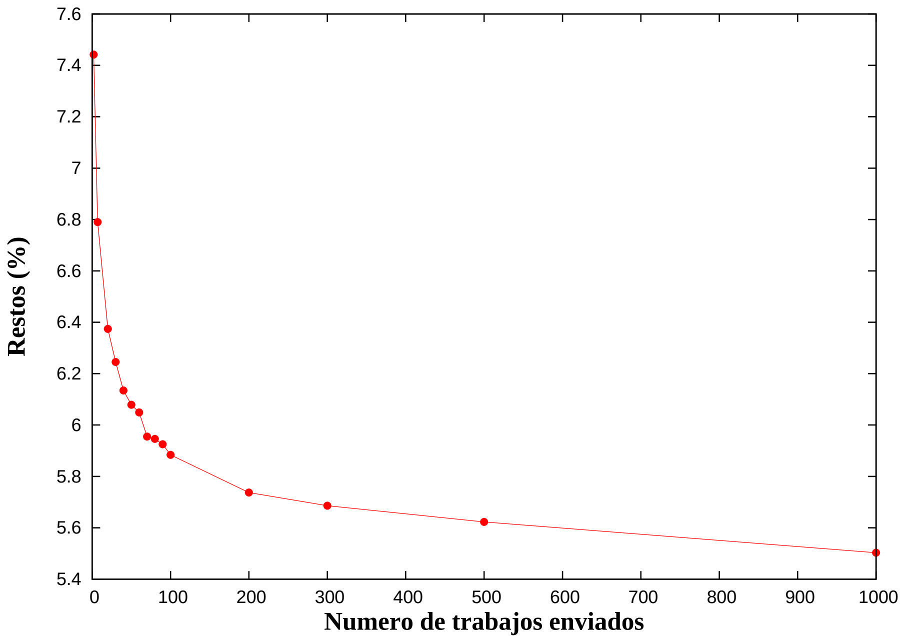


Fig. 9. Representation of the improvement while the number of jobs/optimizations increases

In the second graph, we can observe the improvement of the solutions with the growth of the jobs. If the client would execute it only on his/her PC, the obtained solution would have been in point 1 of the abscissas and would have a little bit more than a 7.4 % of remains. In the grid approach, when executing 100 jobs the remains would be reduced to 5.8 %.

Acknowledgements

A special acknowledgement goes to the Schnell Software colleagues (Miguel Caro and Mari Mar Lahuerta) who have collaborated with all the technical issues related to analysis requirements of the web application. Thanks also to all the PireGrid partners and all the people collaborating in deploying and monitoring the PireGrid infrastructure.

REFERENCES

[1] PireGrid web page: `http://www.piregrid.eu/`.
[2] Schnell Software web page: `http://web.schnellsoftware.net`.
[3] Python API Web page: `http://trinity.datamat.it/projects/EGEE/wiki/3.1/htmlpython/wmproxyapipython.html`.
[4] Python API web page alternative: `http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/api_docwmproxy_python.html`.
[5] Json web page: `http://www.json.org/`.
[6] Python: `http://www.python.org/`.
[7] Pylons: `http://pylonshq.com/`.
[8] jQuery: `http://www.jquery.com/`.
[9] MySQL: `http://www.mysql.com/`.

Ruben Valles studied computer engineering at CPS (Centro Politecnico Superior de Ingenieros) in Zaragoza. He started to work at BIFI in 2005 collaborating in different projects related to software development and at the beginning of 2006 he joined the cluster and grid computing research group. He is currently the responsible of the Distributed Computing area in which he manage both grid infrastructures and the participation of BIFI in national and international grid projects like AraGrid, PireGrid, Ibergrid, EGI-InSPIRE, etc.