Web Services Based Platform for the Cell Counting Problem

Juan Carlos Castillo[1], Francisco Almeida[1],
Vicente Blanco[1], and M. Carmen Ramírez[2]

[1] Departamento de Ingeniería Informática y de Sistemas, Universidad de la Laguna, España
[2] Labotario de Biología Celular y Molecular de C.M., Universidad de Castilla la Mancha, España
{jcastill,falmeida,vblanco}@ull.es, Carmen.Ramirez@uclm.es

Abstract. Cell image processing and analysis is a crucial task in any health science laboratory. Cell counting is a common task usually made by technicians with the support of a custom software tuned with experiment requirements. In this work we present a web services based platform focused on the cell counting problem. Using OpenCF, a web services development framework, we integrate in a single platform services oriented to image processing and classifying, cell counting based on a set of parameters, and data post-processing (plot generation, datasheets, etc.). A GUI added to the platform helps to launch jobs with image sets, and the execution of different tasks from a web service based client.

1 Introduction

Image pattern search and matching is a common problem in different research fields. This problem has been widely studied but it is still under research. In health sciences, pattern matching and cell or other microorganism in a sample is one of the most common tasks we can find daily in laboratories. This work is mainly developed by qualified technicians using different tools like microscopes or other analysis equipment. This process uses to be costly in terms of time and resources, and is one of the difficulties to obtain medical research results.

The method to identify and compute a number of cells in health science uses to follow a similar pattern in most of the situations. Only the different tools used in each experiment vary. In general, the method follows a set of steps. First, a target sample is obtained. Then, a set of procedures are applied in order to stand out the factors under study. And finally, the sample is processed under a visual media by a technician or a custom instrument. This last step is critical since the availability of resources implies that results can't be obtained immediately. Actually, sample processing are delegated to laboratory equipment while the technicians use to obtain the sample and supervise the process. The sample analysis process usually generates digital images with data. Image analysis uses to be hard (and sometimes impossible) to do only by research staff without using computing facilities. Some kind of software will be needed to obtain the relevant information. This software uses to be very specialized and expensive.

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part I, LNCS 8805, pp. 83–92, 2014.

In this work, we propose to implement the image processing which is done in laboratories in a remote platform using web services. The use of web services can accelerate the analysis procedure and reduce time in the obtaining of results, as has been exposed in [1]. The novelty of web services based approach in this field is the high capacity to process images in computing facilities outside the laboratories. These can lead to minimize delay in processing queues of results (with parallel HPC systems). Resources can be decentralized, dedicated laboratory equipment can be minimized, and the costs in space and maintenance can be lowered. Other advantage in this approach is the increase of different types of processing that can be done. There is a widespread types of specialized servers in image processing that could be accessible through web services.

The proposed solution to process data remotely is based in the integration of LLCECO [2], a cell image processing software, in a web services platform. LLCECO is a cell counting and classifying software developed by the High Performance Computing Group at La Laguna University in collaboration with the Biology Cellular and Molecular Laboratory at Castilla La Mancha University. The web services platform where this software was integrated is OpenCF [3]. LLCECO has been developed from scratch for this project. The LLCECO base design has been conceived as a tool to be used in the context of remote computing services.

The paper is structured as follows. Section 2 describes the state of the art in the technologies used. The software for auto-guided cell counting is described in Section 3. Section 4 presents OpenCF, the web services framework used in this project, and Section 5 describes the integration of LLCECO in the OpenCF web services framework. As an example, an use case of this platform is presented in Section 6. Finally, Section 7 shows conclusions and expressions of interests in the field of cell counting problem as well as some future work remarks.

2 Related Work

It has been growing interest in the use of web services based platforms [4]. Big software companies like Rackspace, Microsoft, Google or Amazon are promoting these kind of platforms as a technology solution for its products. Research labs and science groups are also interested [5]. Image processing is one of the most required web services by users. It can be found in facial recognition in social networks or in more complex algorithms on astronomical images [6]. Any web services platform of the companies cited above use a large amount of services related with this field [7].

In the field of medicine and biology fields we can found web services based tools like Inbiomed [8], a platform for processing image biomedical information with methods, data, and image analysis integrated in a single tool using web services, or caGrid [9]. Both are general purpose platforms, with a large set of web services supporting enterprises procedures involved in this projects. One drawback of these approaches is that the tools are designed to adapt to the specific research labs where they are used. Their integration in other research environments seems to be complex.

Fig. 1. Folder with original and resulting images

Other important aspect in biomedical image processing is the software used in labs and research centers where privative software attached to specific instruments (citrometers, microscopes, etc) can be usually found. There are other lightweight and portable alternatives like CellC Cell Counter [10], Pixcavator [11] or GSA Image Analyser [12], usually cheaper and of general purpose. This applications implements the base image processing and counting algorithms, but restricted to one image with a fixed set of parameters given by the user. The main drawback against web service platforms is the limit imposed by the computing resources at laboratories. If there is an increase in the number of images processed by this systems, the cost of analytics process also increase in terms of time and qualified staff resources.

3 LLCECO

LLCECO (La Laguna Cell Counter) is a desktop software tool developed to identify and count different type of cells. This software is the result of a joint effort from the High Performance Computing Group at La Laguna University and the Biology Cellular and Molecular Lab at Castilla La Mancha University. LLCECO was developed in Python and C languages. It allows to process a set of input images, classify them, get optimal processing parameters automatically, identify and count patterns of cells, and generate customized documents with the resulting data. The LLCECO architecture is designed in three independent modules: preprocessing, processing and post-processing modules (see Figure 2).

The preprocessing module has the responsibility to perform tasks like getting the images to be processed, get configuration parameters for processing and post-processing modules, or manage a folder tree to save and compress images. The

folder names are automatically generated with date info from biological samples. The rest of input parameters are set to the following modules.

The processing module is the main engine of the application. The module is also organized in four sub-modules applied sequentially to the input image set:

- `Image classification`. A mean of pixels are computed for certain ranges in the RGB schema. With this task, information about background, cell colors, etc. are obtained.
- `Parameter calculation`. Once the image is classified, its aligned with a set of samples with known generic parameters. This set is composed by four types of cell images, and each of them has been obtained by the processing and classifying of sets from 25 and 50 images.
- `Counting`. The counting algorithm used in LLCECO is the Circular Hough transform [13]. Before applying the algorithm, the image is transformed to an HSV color scheme and a Gaussian filter is applied. With these two operations, noise and false circle detection are reduced.
- `Data result generation`. As a result of previous phases, a document is generated with all image data (number of cells, statistics data, etc) and original images with a mark overlay for each identified cell.

The post-processing module is the most versatile and configurable one. It manages a set of templates to generate resulting data files. It is composed by three steps: plot generation, template application and resulting data files compression. Plots are generated with the data obtained during the processing phase, aligning samples with the number of cells per image and per sets of images. To obtain the resulting data files, a set of spreadsheets templates are used, applying statistical operations, formulas, etc. The operation specified in spreadsheet templates depends on the input parameters of the preprocessing phase. Finally, a compressed file is generated with all this information.

LLCECO is a light, portable application with new features in development. It is really easy to add new templates to this software. This provides a high degree of freedom useful to adapt LLCECO to any other laboratory. We have considered the possibility to process the image set in high performance parallel computers to accelerate the results generation.

It's worth to mention that the software has a modular design, which allows to interact with each module independently. Other modules or services with new functionalities can be added, and then can also be offered as services. This features facilitate platform extensibility to other input/output devices or to other image processing algorithms.

4 OpenCF Web Services Platform

OpenCF is a web services based framework for computing that has been developed at La Laguna University. It design provides a portable framework, easy to install, with a fast setup, and smooth use. OpenCF has a modular software architecture composed by a server module and a client module (see Figure 3).

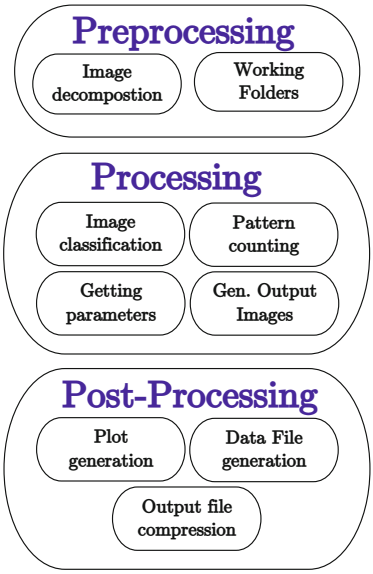

Fig. 2. LLCECO modules

Both modules can be extended independently or even substituted by other components to provide new functionalyties without modifying the remaining system components. OpenCF is oriented to HPC systems and its queue systems. The access, execution and load scheduling on HPC systems are supported by this framework.

The client module (front–end) and the server module (back–end) implement the three lower layers of the web services stack: Service Description, XML messaging, and Transport. The fourth level, Service Discovering, has not been implemented for the sake of security. The user access to web portal is implemented on the client side. Communication between client and server modules is verified by traditional authentications techniques.

The server module manages jobs executions on HPC system, offering them as web services and taking care of excution status. Each HTTP request received by server module from OpenCF client is assigned to an independent execution thread with an independent instance of the server module.

The client module provides an user interface and traslates user request to server requests. The server module receives this requests from authenticated clients and translate them to jobs for the batch system. These modules are also decomposed in smaller parts: Control Access modules, Request Manager, and Collector are in both server and client sides. The client module acts as a front–end and manages a database with information generated by the system. The server, as a back–end, includes modules for batch script generation and job submission to batch system.

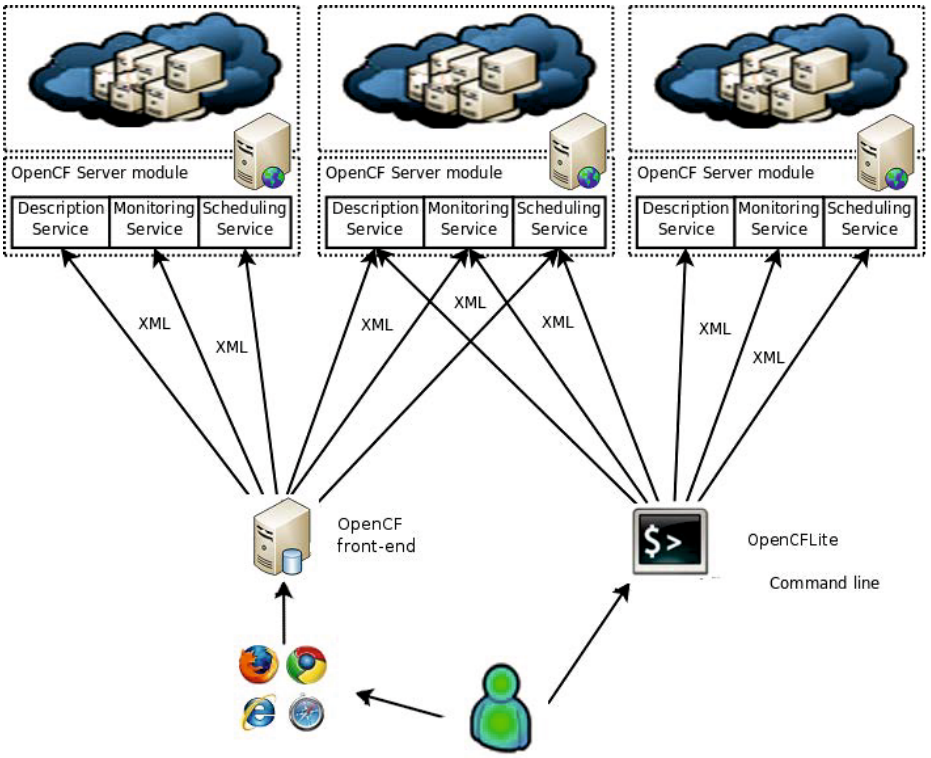

Fig. 3. OpenCF software architecture

The OpenCF server allows the easy add of services from available applications or libraries. Only requires a XML file describing tasks (name, description and arguments) for each application/routine we wish to add as a web service. The new services are offered to OpenCF clients configured at the server automatically. The client receives the service information showing users the service name, arguments, short description, and the OpenCF server offering the service.

OpenCF also implements monitoring and scheduling services [14]. The client shows common services to different servers grouped as a single service. Using an scheduling policy (for example, sorting servers by a metric like "normalized computational power"), the system will send a request to the best candidate server.

5 LLCECO Integration in OpenCF

The framework we have developed integrates the OpenCF web service platform with the LLCECO application. The synergy generated by this effort provides a set of advantages. With LLCECO offered as a web service in OpenCF, we can

Fig. 4. LLCECO Web service in OpenCF

process images remotely and efficiently with a simple interface. We have used the OpenCF platform over other similar tools (like OpenNebula or Eucalyptus [15]) because of its easy integration of external modules (LLCECO) and it's support for parallel systems. The integration has been developed in two steps: first, an application wrapper for LLCECO is developed and then we describe it as a service in OpenCF.

Communication between OpenCF and LLCECO is carried out through a wrapper layer. The wrapper translates server requests to application commands for both input arguments passing and data results collecting. This layer is implemented as a Python script that reads arguments from command line and talks with LLCECO application. OpenCF platform gathers input parameters through its web interface front-end (Fig. 4). A request to LLCECO service is sent to OpenCF server. After processing the service, a call to the LLCECO wrapper is made, passing input parameters as arguments. Once LLCECO application has ended, the wrapper gathers results and compress all the information in a file. The file is assigned to the results in web service with the ID task, allowing the user to download it through OpenCF client interface. The service also sends an email to final user once the taks is completed.

We have described LLCECO service in OpenCF as usual with an XML file. We have set service name, a short description, and the set of input and output parameters (name, type of argument and description, etc. See Listing 1.1).

The name of service is set to *cell_count*, input arguments to *images_zip* and *factor*, and output argument to *output_zip*). This specification must be the same in all servers were LLCECO service is installed. Thus, scheduling in services on OpenCF clients will be performed.

Once the script wrapper and XML describing service are developed, LLCECO application has to be installed in server. Then, wrapper script is configured with application path, number of threads to execute and results folder on OpenCF server. The XML document is also stored in corresponding services folder and the service is added to OpenCF platform with *add_job.sh* script, passing as argument the name of XML service description file.

Listing 1.1. imgs/code.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="job.xsl"?>
<job xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNameSpaceSchemaLocation="job.xsd">
   <name>Cell count</name>
   <service_name>cell_count</service_name>
   <wclient_id>opencf-dev_pcg_ull_es</wclient_id>
   <user_id>test</user_id>
   <binary>bin/cell_process.py</binary>
   <description>
      Example of a matrix invert operation in R
   </description>
   <argument type="base64Binary">
      <name>images_zip</name>
      <sdesc>Zip file with cells image</sdesc>
      <ldesc>
         Zip containing a directory with images of cells
      </ldesc>
      <fname_in_server>images.zip</fname_in_server>
   </argument>
   <argument type="integer">
      <name>factor</name>
      <sdesc>Factor to multiply cell numbers</sdesc>
   </argument>
   <output_file>
      <name>output_zip</name>
      <description>
        All the input and results files of the execution
        of the service.
      </description>
   </output_file>
</job>
```

6 Use Case

To show the usefulness of this platform, we have processed a set of samples from Cellular and Molecular Biology Lab at CRIB, Castilla La Mancha University. The data are stored in two folders with images from two cell cultures with 25 and 40 images respectively. Only one request is registered on the platform where both folders are included in a compressed file (approximately 6Mb, with an upload time around a couple of seconds in a local research network and less than a minute from a home adsl line). The system returns a compressed file with a folder including processed images and a datasheet with resulting data and plots.

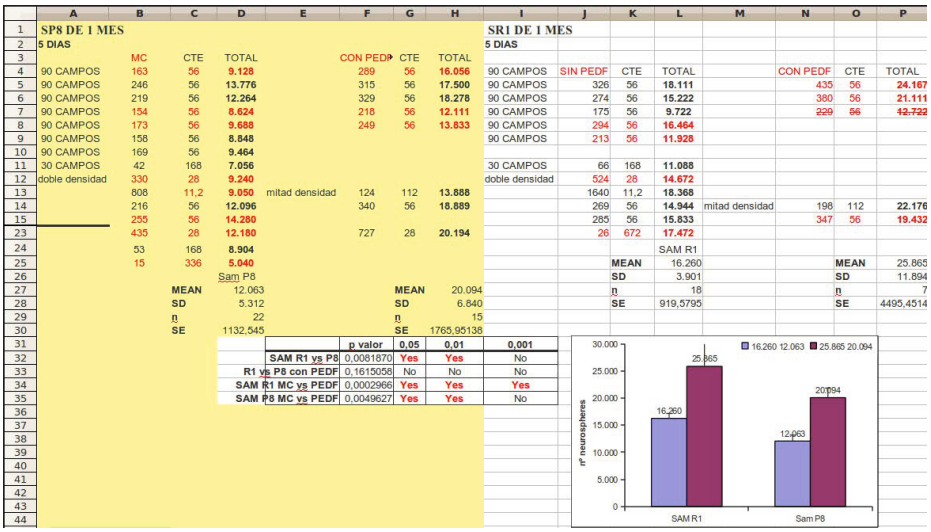

Fig. 5. Results spreadsheet with data, statistical analysis, and plots

The whole image processing and computed data takes only a few seconds. The bottleneck in the applications is located in sending and receiving data (with bigger archive sizes, files can be split to optimize the process).

If the number of requests to the platform increases, we can use an HPC cluster to keep the execution times in this range. The service could scale to a high number of requests easily. Figure 1 shows the output corresponding to one of the image folders. Figure 5 shows the spreadsheet resulting for an experiment with an ANOVA statistical analysis.

7 Conclusions and Future Work

In this work we present a general web services based computing platform for the cell counting problem on biomedical images. This tool is composed by the integration of a web services framework (OpenCF) with an application to process and count cells (LLCECO). The platform offers a web interface where the user can process the cell counting on a set of images. The user must provide a compressed file with target images stored in folders. A set of input parameters will also be provided to specify process configuration. As a result, a set of files with data analysis and output images is provided. Our goal with this platform is to improve the working pace in labs and research centers than needs the analysis and counting of cells in their experiments, reducing costs and time to obtain results.

This platform differs from others in flexibility, portability, general services and computing power. By using web services we can access to a cloud of servers to process images without intervention by the user and the computing power is higher than those systems we can find in laboratories.

Actual development lines for the platform focuses on auto–guided learning for the classification process and optimal parameters gathering for image analysis. In this way, the execution of tasks in computing resources can be done totally autonomous. We have also researched the use of the tool with other image experiments like topographical images, other type of cultures or spermiograms.

As improvements to LLCECO, we propose a lightweight multi–platform client that can be installed in lab equipment where the images are obtained. So the system can execute the task with corresponding input parameters automatically, without intervention of the technical staff.

Acknowledgements. This work has been partially supported by EC (FEDER) and Spanish MICINN (Plan Nacional de I+D+I) under contracts TIN2008-06570-C04-03 and TIN2011-24598.

References

1. Solomonides, T., McClatchey, R., Breton, V.: From grid to HealthGrid. In: Technology and Informatics, Amsterdam, Netherlands (September 2005)
2. Castillo, J.: Llceco: La laguna cell counter (2012), `http://opencf.pcg.ull.es/redmine/projects/opencfcc`
3. Santos, A., Almeida, F., Blanco, V.: Lightweight web services for high performace computing. In: Oquendo, F. (ed.) ECSA 2007. LNCS, vol. 4758, pp. 225–236. Springer, Heidelberg (2007)
4. Darrow, B.: Amazon is no.1 who's next in cloud computing (2012)
5. Evangelinos, C.H.C.: Cloud computing for parallel scientific hpc applications: Feasibility of running coupled atmosphere-ocean climate models on amazons ec2. In: First Workshop on Cloud Computing and its Applications, Chicago, USA (2008)
6. Almeida, F., Blanco, V., Delgado, C., de Sande, F., Santos, A.: Idewep: Web service for astronomical parallel image deconvolution. J. Netw. Comput. Appl. 32(1), 293–313 (2009)
7. Fronckowiak, J.: Processing images with amazon web services (2008)
8. Perez, D., Crespo, J., Biomedical, A.A.: image processing integration through inbiomed: A web services-based platform. Biological and Medical Data Analysis, 34–43 (2005)
9. Zhang, J., Madduri, R., Tan, W., Deichl, K., Alexander, J., Foster, I.: Toward semantics empowered biomedical web services. In: 2011 IEEE International Conference on Web Services (ICWS), pp. 371–378 (July 2011)
10. Selinummi, J., Jenni Seppälä, O.Y.H., Puhakka, J.A.: Software for quantification of labeled bacteria from digital microscope images by automated image analysis. BioTechniques 39(6), 859–863 (2005)
11. Saveliev, P.: Pixcavator (2011), `http://inperc.com/wiki/index.php?title=Cellcounting`
12. GSA: Gsa image analyser (2012), `http://image.analyser.gsa-online.de/`
13. OpenCV: Hough circle transform (2012), `http://opencv.itseez.com/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html`.
14. Santos, A., Almeida, F., Blanco, V., Castillo, J.C.: Web services based scheduling in opencf. Journal of SuperComputing, 88–114 (2010)
15. Sempolinski, P., Thain, D.: A comparison and critique of eucalyptus, opennebula and nimbus. Critique, 417–426 (November 2010)