Biological Experiments on the Grid: a Novel Workflow Management Platform

D. Laforenza, R. Lombardo,
M. Scarpellini, M. Serranò, F. Silvestri
ISTI - CNR
Pisa, Italy
{n.surname}@isti.cnr.it

P. Faccioli
Istituto Sperimentale Cerealicoltura (ISC)
Fiorenzuola d'Arda, Piacenza, Italy
primetta.faccioli@entecra.it

Abstract

Bioinformatics is one of the key application of this century. The attention received by all the current media is astonishing and both academics and industries are carrying on researches in many fields related to computational methods applied to life sciences. In this paper we present some results obtained within an Italian funded research project called ESCOGITARE. We present the design and implementation of a Grid-based architecture for scientific workflow management that, differently from others, allows the dynamic discovery of existing Web Services in combination to ad-hoc developed ones. The paper presents the main features of current scientific workflow management systems. Using a real-world case study (taken by the agricultural research domain), we show how we overcome the limitations of current approaches.

1 Introduction

The development of computational techniques applied to biological problems has recently received a lot of attention. In fact, there are many examples where computer science has been applied for solving common biological problems such as: similarity finding by means of gene comparison within a species or between different species; sequence analysis to automatically search for genes and regulatory sequences within a genome, and so on. All of these techniques go under the generic name of *Bioinformatics* [3].

The computational requirements for bioinformatics applications are usually very high. In fact, to effectively analyze real-world datasets high performance parallel and distributed architectures are needed. Moreover, biological experiments are usually collaborative by nature and, to effectively support such a kind of collaboration, computing platforms should enable the sharing of resources and previous experiment results. Grid Computing [5] is thus the most natural answer to this kind of needs because it represents an integrated and collaborative computational environment through which life scientists could access computing resources and data virtually unlimited.

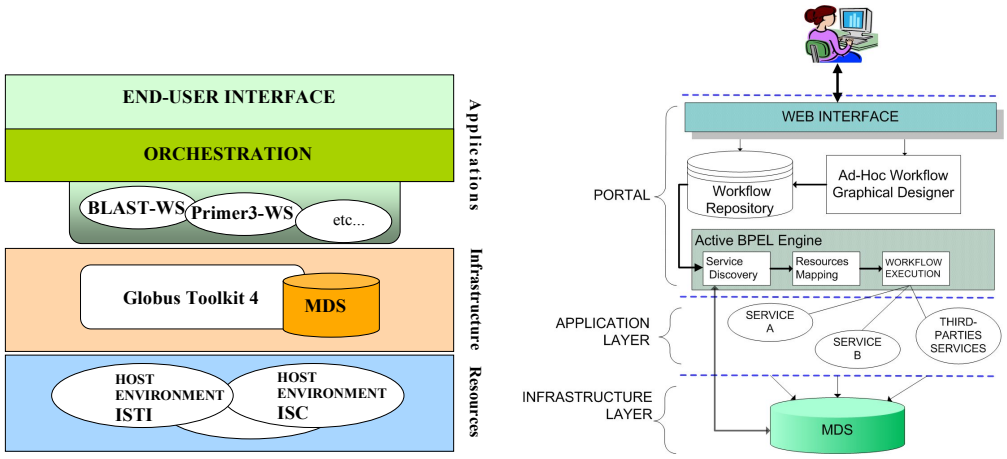
The main result presented in this paper is exploited in the Italian funded project ESCOGITARE[1]. It is a novel architecture that can be used to build workflow applications for the life sciences and for bioinformatics in particular.

ESCOGITARE is aimed at applying E-Science and the related technologies to design and build a supporting infrastructure for the research activities of the Italian Council for Agricultural Research (CRA). One of the main targets of the infrastructure developed within ESCOGITARE

[1] http://hpc.isti.cnr.it/escogitare/

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE

is to enable agriculture scientists, located in several CRA institutes spread all over Italy, to conduct bioinformatics and geoinformatics experiments using a workflow management system able to select data and instruments installed in several laboratories in a transparent and secure way. Obviously the architecture and the infrastructure are designed to be as general as possible and to support the wider spectrum of applications possible.



(a) Grid resource sharing platform (b) Workflow management platform

Figure 1. The ESCOGITARE project architecture

In Figure 1.(a) the general view of the Grid resource sharing architecture is shown (from the bottom to the top): (i) *Resources* shared by Virtual Organization's (VO) entities; (ii) *Infrastructure* to support the necessary instruments to encapsulate and virtualize the resources offered by VO members; (iii) *Applications* offered to final users. Applications (exposed as Web Services) are usually composed as workflows[2].

The main innovation we introduce is the capability of: (i) combining the use of external (w.r.t. the organization's Grid infrastructure) off-the-shelf Web Services, in conjunction with ad-hoc – i.e. internally available – ones; (ii) dynamically binding workflow's tasks to Web Services during its execution.

2 Our Architecture

In this section we describe in deep the architecture of the workflow management system of the ESCOGITARE project. Our architecture is based upon the "de-facto standard" Grid middleware Globus Toolkit 4 (GT4)[3]. The choice of GT4 enables us to build the underneath Grid infrastructure, also exploiting WSRF-based[4] services. The selection of suitable technologies for the implementation of our workflow management system, built on top of GT4, was strictly guided by the real needs coming from the applicative domain. After a comprehensive analysis of the previously mentioned approaches and tools and of the state of the art in workflow management, our final decision was the adoption of BPEL [5] (supported by the open-source ActiveBPEL 2.0 engine [6]) as the workflow definition language. BPEL is a "de-facto standard"

[2] http://www.w3.org/TR/ws-arch/
[3] http://www.globus.org
[4] http://www.globus.org/wsrf
[5] http://www-128.ibm.com/developerworks/library/ws-bpel
[6] http://www.activebpel.org

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE

for the definition of business processes. We believe that BPEL is more adaptable than the other technologies to a highly dynamic environment that exploits Web Services compliant to the WSRF specifications. Moreover, BPEL enables the workflow designer to take into account particular reactions to faults introducing fault tolerance in the infrastructure. Another remarkable advantage is that several stable tools support BPEL. Anyway, due to its business-oriented nature, some changes are required before adopting BPEL in a scientific domain that exploits WSRF-compliant services. These changes deal with all the life-cycle phases of a BPEL process: (i) its definition through a GUI; (ii) its deployment on an engine; (iii) its execution. The philosophy we adopt is to simplify all these phases and to require as less user-interaction as possible. The result is the architecture shown in Figure 1.(b).

BPEL process definition through a GUI Different professional workflow designers exist (e.g. ActiveBPEL Designer) that enable BPEL workflow graphical definition by the user, but all require the learning of the technicalities related to such powerful and comprehensive tools. To avoid this, we enriched the ESCOGITARE portal with some useful graphical features, providing an on-line simple designer. Our idea is to allow the user to graphically create a workflow starting from existing Web Service descriptions, i.e. taken by a WSDL palette included in the "workflow editing" section of the ESCOGITARE portal. Those WSDL will be contained in an archive and grouped by their application domain. They describe only the operations a service exposes (e.g. sequence alignment using ClustalW[7]), without specifying any binding information, enabling users to compose services without any knowledge on the actual services instance. The designer, besides permitting the composition of domain specific services, allows the user to insert in the workflow some cross-domain operations like storing data, requiring the user approval for intermediate output, etc. Researchers with particular knowledge on Web Services can still build workflows using professional tools exploiting all the power of the related definition language.

BPEL process deployment BPEL processes, defined through the on-line designer, are deployed on our modified ActiveBPEL 2.0 engine (in order to be used by the researchers). The deployment of a process on an ActiveBPEL engine requires the definition of a particular file, called *Process Deployment Descriptor* (PDD). The PDD file gives the engine several information about the process (e.g. how to address partner services, etc.) that will be exploited during process execution. To create the PDD some information (e.g. service namespaces) are needed; unfortunately this operation requires to be conducted by people very well skilled on WSDL technology. In order to hide the complexity to the typical end-user (e.g. an agriculture scientist), we automatically create the PDD file using information contained in the WSDL archive. In this way we decouple experiment logic from technical details.

BPEL process execution and monitoring In order to introduce some improvements in the last phase of the BPEL process life cycle, we modified the ActiveBPEL engine. The improvements are a late binding mechanism to select the most suitable service instance and, most important, a mechanism to enable interoperation with services compliant to the WSRF specifications. Both operations are based on service information contained in the MDS[8]. The late binding is performed using the load information of the computational resources provided by a monitoring service and stored in the MDS. The *Uniform Resource Name* (URN) associated to the service is used to query the MDS in order to obtain this information for every computational resource hosting a service instance. Once we gather the information for all the computational

[7] http://www.ebi.ac.uk/clustalw/
[8] http://www.globus.org/mds/

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

resources, we apply a mapping heuristic to locate the most suitable resource for service invocation. Due to the growing importance of WSRF, we believe it is important to allow the BPEL process to invoke operations exposed by Web Services compliant with the WSRF specification. Information contained in the MDS (e.g. resource keys) are used to allow the invocation of WSRF services. A useful feature of the ActiveBPEL engine is the associated web interface that shows the active processes and allow the user to follow step-by-step the execution of his experiment and to inspect the intermediate results through a simple browser.

3 Use Case

The use case is a bioinformatics application devoted to cereal species traceability in food (ISC has provided this use case). The composition of a specific food is a key factor in determining the quality and the safety of the final product. The reliable identification of plant, animal and microbial species is therefore essential for the handling, marketing and processing of grain and derived products. Usually, the traceability lab working flow starts with the identification of a set of DNA sequences publicly available on reference databases (e.g. Genebank, TIGR) and selected via a functional annotation-based query (i.e. seed storage proteins coding genes). On the basis of the researcher knowledge, a deeper selection of the hypothetical optimal sequences is then obtained. The chosen sequences are subsequently aligned by means of a dedicated, locally installed or web-based, software (i.e. ClustalW). In this way, common DNA regions are underlined and a more restricted set of sequences is selected for the lab analysis via a wet lab technique named PCR (Polymerase Chain Reaction). This technique produces enzymatically replicated DNA and the specificity of the amplification process is ensured by the correct identification of boundaries defining primer sequences. The optimal primer sequences can be found running specialized software packages (i.e. "Primer3[9]"). The species specificity of the selected primers are verified by BLAST[10] (Basic Local Alignment Search) search to ensure that the PCR reaction will be positive just in case of the presence of the cereal species of interest (i.e. presence of wheat flour in durum pasta). BLAST compares nucleotide or protein sequences to specific databases and a statistical significance of the matches is calculated. So, for the case study, the tool should find a match just between primer sequences and the specific cereal sequences we are interested to outline. At this point the researcher activity moves from the dry (or *in-silico*) biology to the lab bench biology to *in-vitro* test the effectiveness of the traceability protocol.

In order to prove the usability of our architecture we proceeded as follow: first we installed mpiBlast 1.4.0 (a MPI version of BLAST) on our Apple XServe cluster along with the NCBI's "nucleotide sequence[11]" database while the "Primer3" and the "ClustalW" executables have been installed on a Linux machine.

We then provide a set of Web Services wrapping those executables. The Web Services require structured XML representation for the input and output data of the operations exposed. Unfortunately, only BLAST is able to generate XML output and no standard exists to represent the other kind of data used in the experiment (e.g. sequence or alignment representation). To solve this problem we adopt some XML schemas defined by the BioSchemas project[12] (SequenceML and AlignmentML) that fit exactly our needs, and we develop a simple XML representation for the Primer3 output. These XML representations are converted in the format accepted by the tools and written on files before executing the programs. We then apply the same operation to the output generated by the executables to obtain an XML representation of such data.

[9] http://frodo.wi.mit.edu/primer3/input.htm
[10] http://www.ncbi.nlm.nih.gov/blast/
[11] ftp://ftp.ncbi.nih.gov/blast/db/blastdb.html
[12] http://bioschemas.sourceforge.net

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

We also provide a "human-interaction" service to handle the interaction between the user and the workflow for the validation of intermediate results. The service sends an e-mail to the user redirecting him to a dynamically-generated web page where the intermediate results are located. The user can then inspect the results and decide whether the workflow can continue or have to be stopped.

We then use the designer to compose the services in a workflow that reflects the structure of the experiment. We decide to start the workflow after the query to the sequence database, so our workflow starts with the BLAST analysis of the selected sequences. It is worth noticing that, in this phase, we are able to compose the workflow without any knowledge of the actual location of the services. The tasks of the workflow are just tagged with a keyword (e.g. "clustalw") that will be used by the modified workflow engine to retrieve the location of a suitable service exploiting the Grid infrastructure.

In order to explore all the possible paths we performed experiments both on a single sequence and on a family of sequence.

Note that, instead of invoking a local copy of the BLAST service we could have opted for the invocation of the NCBI's provided service. Better to say, we can let the mapping heuristic choose among the local or the remote copies of the services.

4 Related Work

Enabling the execution of workflows for complex and long-running scientific experiments or business processes, has been a hot research topic in the past years. Table 1 shows some of the most representative ones compared to our proposal.

Project	Workflow Language	Engine	Middleware	Flow Element
Progengrid	UML + XML	GRB	GT 4	WSs
Pegasus	Chimera	DAGMan	GT 4	EXEs
Taverna	SCUFL	FREEFLUO	WS-Based	WSs
Proteus	XML	GRAM	GT 4	EXEs
ESCOGITARE	BPEL	Active BPEL	GT 4	WSs

Table 1. Main features of current scientific workflow management systems.

ProGenGRID [1] is a system, which aims at providing a virtual laboratory where e-scientists can design biological workflows composed using an UML dialect. We differentiate from them since we adopt a service-oriented workflow definition language (BPEL) often preferred in a SOA environment.

Proteus [2] is a Grid-based Problem Solving Environment for bioinformatics applications. It uses ontologies to describe the semantics of data sources and to build a knowledge base about bioinformatics resources and processes. User can design workflows selecting ontology-based resources and composing them in a graphical model. The abstract workflow, defined by the user, is then translated into a concrete workflow script that is submitted to Globus' GRAM for execution. Proteus, therefore, can invoke only executable programs (EXEs). Since our solution is WSDL-compliant we can invoke Web Services (WSs) directly.

Taverna [6] is a tool, developed within the *my*Grid[13] project, that enables users to graphically compose bioinformatics workflows exploiting several web services and tools spread all over the World. The workflows are written in *SCUFL* and follow a particular XML schema. The FreeFluo

[13] http://www.mygrid.org

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE

IEEE COMPUTER SOCIETY

workflow engine, which is coupled with Taverna, manages the execution of the workflows. Our architecture allows the composition of a workflow without worrying about where each composing piece is located. In other words, it allows the dynamic binding of workflow's components at run-time. Taverna, on the other hand, only allows static binding. Furthermore, the expressiveness of BPEL is better than Taverna's SCUFL. This, however, at a price of a higher complexity that is however hidden by our designer interface.

Pegasus [4] is a configurable system that can map and execute complex workflows on the Grid. Pegasus receives an abstract workflow description from Chimera, produces a concrete workflow, and submits it to Condor's DAGMan for execution. The abstract workflow describes the transformations and data in terms of their logical names. The concrete workflow, which specifies the location of the data and the execution platforms, is optimized by Pegasus from the point of view of Virtual Data. If data products described within abstract workflows are found to be already materialized, Pegasus reuses them and thus reduces the complexity of the concrete workflow. The Pegasus engine, as in the case of Proteus, allows the execution of EXEs thus preventing from directly invoking WSs.

To resume, our environment is built using SOA's principles and allows us to dynamically bind and to directly call WSDL-based services. Advantages are thus two-fold: application level designer can take advantage of services developed ad-hoc for our infrastructure and even of those developed by third-parties but still WSDL-compliant; at the infrastructure level we can remotely deploy services at run-time. This allows us to manage in an optimal way the problem of service migration.

5 Conclusion and Future Work

In this paper we have shown the design and the result of a validation phase of our architecture on a real-world bioinformatics use case. We have shown the main features and the details on how we implemented them. We have validated the design with a pilot application in the domain of bioinformatics applied to agricultural research. In the near future we are going to introduce some novel features such as the possibility of exchanging very large datasets between services. Currently, orchestration is centralized, this means that results are returned back to, and forwarded by, the coordinator. For large datasets this approach is inefficient, we will design some decentralized orchestration mechanisms enabling the direct forwarding of results to the next service in the workflow (e.g. extending the BPEL language with data handling constructs).

References

[1] Giovanni Aloisio and Maria Mirto Massimo Cafaro, Sandro Fiore. Progengrid: A workflow service infrastructure for composing and executing bioinformatics grid services. In *CBMS '05: Proceedings of the 18th IEEE Symposium on Computer-Based Medical Systems (CBMS'05)*, pages 555–560, Washington, DC, USA, 2005. IEEE Computer Society.

[2] Mario Cannataro, Carmela Comito, Filippo Lo Schiavo, and Pierangelo Veltri. Proteus: a grid based problem solving environment for bioinformatics. *IEEE Computational Intelligence Bullettin*, 3(1):7–18, 2004.

[3] Jacques Cohen. Computer science and bioinformatics. *Commun. ACM*, 48(3):72–78, 2005.

[4] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G. Bruce Berriman, John Good, Anastasia Laity, Joseph C. Jacob, and Daniel S. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(3):219–237, 2005.

[5] Ian Foster, Carl Kesselman, and Steve Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.

[6] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R. Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS'07)
0-7695-2905-4/07 $20.00 © 2007 IEEE