Building Scientific Workflows on the Grid: A Comparison between OpenMole and Taverna

Bojana Koteska
Ss. Cyril and Methodius University in Skopje
64 PUBLICATIONS **326** CITATIONS
SEE PROFILE

Boro Jakimovski
Ss. Cyril and Methodius University in Skopje
47 PUBLICATIONS **345** CITATIONS
SEE PROFILE

Anastas Mishev
Ss. Cyril and Methodius University in Skopje
96 PUBLICATIONS **517** CITATIONS
SEE PROFILE

Building Scientific Workflows on the Grid: A Comparison between OpenMole and Taverna

Bojana Koteska, Boro Jakimovski, and Anastas Mishev
Ss. Cyril and Methodious University,
Faculty of Information Sciences and Computer Engineering,
Rugjer Boshkovikj 16, 1000 Skopje, Macedonia
{bojana.koteska, boro.jakimovski, anastas.mishev}@finki.ukim.mk

***Abstract* — Scientific workflows are used in a number of different scientific domains. The interest for using workflows grows because of the progress of Grid computational techniques and the increased resources availability. Moreover, there are a number of scientific workflow systems that allow building workflows needed for different scientific applications and scientific experiments. In this paper, we analyze the characteristics of two open source scientific workflow systems: OpenMole and Taverna. Both systems are designed to provide environments for parallel execution of processes, to support graphical workflow design, to offer scalability and remote execution of workflows (grids, clusters, clouds). The former is relatively new, but both are open to users if they want to contribute to their development. Although these two systems have a number of common features, they differ in the way of workflow composition, execution and implementation logic. In order to show and verify the similarities and differences between these two scientific workflow systems, we designed a workflow and we implemented it in both systems.**

Keywords-Scientific workflow, OpenMole, Taverna, Grid, Java

I. INTRODUCTION

Nowadays, many scientific communities and business companies use workflow systems for designing, automating, controlling and managing the complex flow of data and processes. For example, scientific communities may use workflow systems to perform experiments with a large set of data and computations that require some remote services, while business companies can use them to automate the manufacturing process and to avoid redundant tasks.

According to the ISO 12651-2 standard, the terms *workflow, workflow engine* and *workflow management system* are defined as [1] [2]: *Workflow* - automation of a process, in whole or part, during which documents, information, or tasks are passed from one participant to another for action, according to a set of procedural rules. *Workflow engine* - software service or "engine" that provides the run time execution environment for a workflow instance. *Workflow management system* - a system that defines, creates, and manages the execution of "workflows" through the use of software running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications.

Scientific workflows are designed for modeling large-scale data-intensive and compute-intensive scientific processes [3]. The main purpose of the scientific workflow is to automate the scientific process where tasks are structured based on their control and data dependencies [4]. The goal of scientific workflows is to provide a simple and concise notation that allows parallelization of tasks and computations which is needed in the scientific domain. The scientific workflows must enable independent tasks to be run in parallel and scheduled to run in clustered or Grid environments [5].

In this paper, we analyze the characteristics of two scientific workflow systems: OpenMole and Taverna. We also design a workflow to compare the implementation details of the workflows in the two environments. We are also aiming to find out which scientific workflow enables simpler implementation for different parts of the workflow. The rest of the paper is organized as follows. Related work is given in Section 2. Section 3 presents the characteristics of the two scientific workflows and comparison between them. In Section 4, we describe the implementation of the workflow in the two systems. The conclusion and future work are specified in Section 5.

II. RELATED WORK

Barker and van Hemert [6] made a survey of existing workflow technology from the business and scientific domain and they proposed some suggestions towards the future development of scientific workflow systems. Wolstencroft et al. [7] give a description of the Taverna workflow tool. They describe Taverna as a tool designed to combine distributed Web Services and/or local tools into complex analysis pipelines. Goble and De Roure [8] present the Taverna workflow system and they emphasize that scientific workflow environments need a rich ecosystem of tools that support the experiments made by scientists. One example is myExperiment which is an initiative to create a social networking environment for workflow designers. Oinn at al. [9] give a description of Taverna and lessons learned during its development. Their goal is to show how workflows fit into the scientists' experimental context. Reuillon et al. [10] present the OpenMOLE Domain Specific Language through the example of a toy model exploration and through the automated calibration of a real-world complex system model in the field of geography. Reuillon et al. [11] also describe the OpenMOLE as a scientific framework that provides virtualized runtime environment for distributed computing.

III. Characteristics of Scientific Workflow Systems: OpenMole and Taverna

A. *OpenMole*

OpenMole (Open MOdeL Experiment) [12] workflow engine offers parallel execution environments for naturally parallel processes. This scientific workflow system supports advanced numerical experiments on simulation models and distribution of the workflows on multi-core machines, desktop-grids, clusters and grids. Users are able to embed Java, Binary executable or NetLogo executables. The users should not care about the data and tasks sizes because OpenMole is scalable and it supports TB of data and millions of tasks.

OpenMole offers a graphical interface and a command line interface for designing experiments and their distribution. The workflow in OpenMole is called Mole. It can contain *tasks* (Groovy, Exploration, SystemExec, Mole Task, Agent based model, Sensitivity, and Stat), *transitions* (arrow that defines the precedence order between tasks), *prototypes* (variables that operate in the workflow), *samplings* (can be composed graphically - complete, shuffle, zip, combine, etc. and domains: range, multiple file, single file, uniform distribution, logarithmic range domain), *environments*, *hooks* (Mole listener, performs a particular action on the output and *sources* (reads data from a CSV file and maps it to prototypes included in the workflow). In order to be run, a mole must contain at least one task and a starting task (capsule).

B. *Taverna*

Taverna [13] is a domain-independent Workflow Management System used for designing scientific experiments. It is available as a desktop client application (Taverna Workbench), Taverna Server which enables remote execution of workflows and from a command line. Taverna is integrated with two other myGrid Tools: my experiment (workflow sharing environment) and BioCatalogue (catalogue of Web services for Life Sciences). Also, the workflow created in Taverna could be distributed on a grid, a cloud, used behind a portal or bundled with other products. It is a scalable scientific workflow system which has access to local and remote resources and grid services (more than 3500). Taverna has support for calling client libraries (in Ruby and Java) and tools and scripts on local or remote machines. One of the most interesting features is that Taverna enables rapid incorporation of new services without coding. It has a graphical workflow designer that works on the principle drag and drop. Taverna provides workflow validation and intermediate values during the execution time.

This workflow management system allows users to define how data will flow between services, but they should not care how the services will be invoked. Data also can be converted from one format to another. Taverna has a built-in support for adding: *Services*, *Lists and iterations*, *Loop*, *Control link* (used to set dependencies between services that do not directly share data), *Merges* (enable combination of outputs from more services into one single input), *Parallel service invocation*, *Component* (reusable unit of functionality which is defined by a workflow).

C. *Comparison between OpenMole and Taverna*

Both tools have many similarities in terms of functionality, but they differ slightly in the way of their implementation. Taverna offers an addition of existing Web services, the automatic conversion of the list depths in order to connect two services. On the other hand, OpenMole offers a simpler definition of composite inputs (for ex. making combinations of the elements in array). Table 1. shows the comparison of some of the important features of these tools.

IV. The Workflow

This section describes our workflow and its implementation in OpenMole and Taverna. The workflow is shown in Fig.1. Each circle represents a file containing some data. The squares a_1, a_2, a_3.., a_{64} depict file actions, in our case, merging of files' contents. We choose always 64 combinations of 3 files. In order to do that, we make two copies of the file list in the directory. Then each file list is shuffled and the first 4 files of each of the three file lists are chosen. The rectangle "folderA" represents the directory where the combinations are chosen and it is a starting point in the workflow.

The result of an each action a1, a2, a3.., a64 is a file saved in a new directory and shown in the rectangle "folderB". The actions a1, a2, a3.., a64 are performed in parallel. Also, when three files are created in "folderB", the action b is performed, which merges the contents of the 3 files into a new file. Then the files in "folderB" are deleted. The action *b* is repeated 64/3 = 21 times and it is performed in parallel with the actions a_1, a_2, a_3.., a_{64}, immediately after creating 3 new files in "folderB". When a new file is created as a result of the action *b,* it is moved to "folderA". The workflow can be executed repeatedly, depending on the user input.

TABLE 1 COMPARISON BETWEEN OPENMOLE AND TAVERNA

Characteristic	**OpenMole**	**Taverna**
Graphical interface for designing workflows	Y	Y
Drag and drop method	N	Y
Scalability	Y	Y
Distribution on remote machines, grids	Y	Y
Composite input - sampling (graphically)	Y	N
Invoking online Web services by URL	N	Y
Parallel execution environment	Y	Y
Automatic parallel execution of tasks when	N	Y
Support security standards	N	Y
Automatic conversion of the input/output depths	N	Y
Control links	N	Y
Different input and output depths	Y	Y
Nested wokflows	Y	Y
Workflow validation	Y	Y
Embedding Java executables	Y	Y
Running script code	Y	Y
Embedding NetLogo model	Y	N
Looping	Y	Y
Merging outputs	Y	Y
Reading data from a CSV file	Y	Y
Creating Components	N	Y
Enable plugins	Y	Y
Tracking results	Y	Y
Built-in support for sharing workflows	N	Y
Built-in support for searching workflows online	N	Y

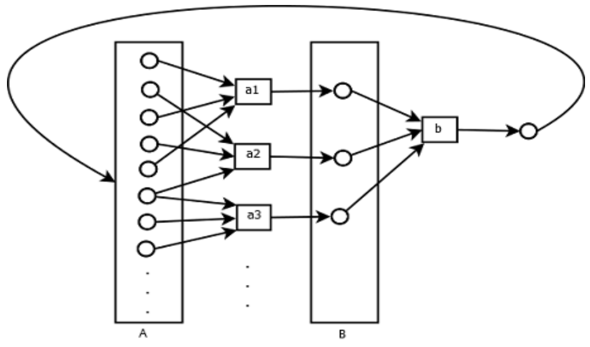

Figure 1.Our workflow diagram.

A. *The Workflow Implementation in OpenMole*

The workflow implementation in OpenMole is shown in Fig. 2. The starting "defining i" capsule is a groovy task and it is only used for setting the integer prototype *i* to 1. The prototype *i* is used as a counter for the number of workflow iterations. The output of this capsule is only *i*.

The capsule "combinations" is an exploration task and it is used for defining the sampling shown in Fig. 3. First, the three multiple file domains "in folderA" and three file prototypes "file", "file1" and "file 2" are defined. All three multiple domains have the same directory path (the directory A in the Fig. 1). The samplings: "Zip with string name", "Zip with string1 name" and "Zip with string2 name" are used for making arrays of file names in "folderA". When three arrays are made, each of them is shuffled (Shuffle sampling) and only the first 4 elements are taken (Take (4) Sampling). The four file names of each array are combined (Combine sampling). The 64 combinations (4*4*4) are made. There are 7 outputs from this task : the prototype *i* and 6 arrays (file[1], file1[1], file2[1], string[1], string1[1] and string2[1]).

The next capsule "comb. 3 files" is a groovy task and it includes a jar library. The previous and this task are connected with exploration transition which means that the same method will be executed in parallel on different combination of three files in "folderA" a_1, a_2, a_3.., a_{64} as specified in Fig. 1. We call the method from the library that has three strings as input parameters. In our case, they are file names from each combination. The three files will be merged into a new file and it will be stored in the directory "folderB". The output of this capsule is only the integer prototype *i*.

The "comb. 3 first" capsule is also a groovy task and it includes a jar file. The method called from the library is used for checking if three files exist and merging them in the directory "folderB". This task (*b*) is executed in parallel with the previous task "comb. 3 files". When three files will be created, their contents are merged into a new file and it is moved to the directory "folderA". The three files are deleted from "folderB". The output of this capsule is also only the integer prototype *i*.

The last capsule in this workflow "increase i" is a groovy task and it is used only for increasing the integer prototype *i*. It is connected with the second task only for controlling the value of the prototype *i*. The condition of this transition is set $i \leq 3$. The transition between "comb. 3 first" and this capsule is

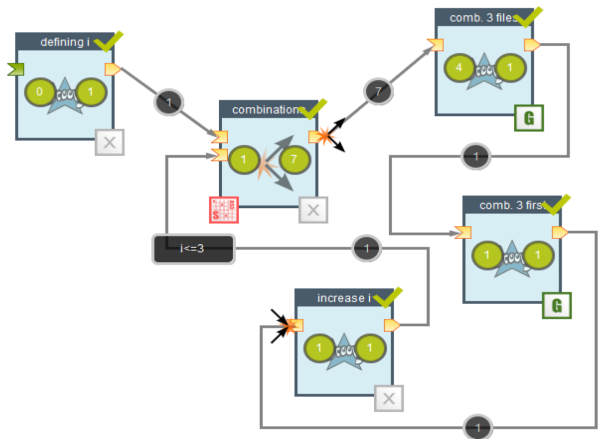

Figure 2. The workflow implementation in OpenMole.

an aggregation because this task does not need to be executed more times, but only once in each workflow iteration. In order to run the tasks on the Grid, the Grid environment should be created and added as an execution environment of the capsules "comb. 3 files" and "comb. 3 first".

B. *The Workflow Implementation in Taverna*

The workflow implementation in Taverna is shown in Fig. 4. The workflow "Workflow 14" is a nested workflow. It enables the workflow to be executed *n* times. The input *n* is a list of integer numbers. This is the starting point and the only important thing is the number of elements in the list.There is also an integer input port n_i at the beginning of the nested workflow which is increasing in each workflow iteration (iterate through the elements in the list).

The service "DirectoryPath" is a beanshell script where the path of the directory "folderA" is specified as a string "directory". That is the only output of this beanshell script. Next, there are 3 services "List_Files", "List_Files_1", "List_Files_2" which are identical, but they are used for making the same array of files in the input directory. Then, each of the three arrays of files is shuffled and only the first four elements are taken. This is done by writing a script (Java code). These three services will create 64 random combinations, each of the three files. The output of each of these three services is a list of four files.

The next service is called "concatenate_and_write_in_files_by_3_Files" and it has inputs with length 0, which means that it takes one by one element of each of the three lists of four files. That is how all the combinations are made. This service is also a beanshell script and it merges the contents of each combination of three files. This action will be repeated 64 times a_1, a_2, a_3.., a_{64}. Each new created file will be stored in the directory "folderB".

The "combine_first_three" is a service used for checking if three files exist and merging them in the directory "folderB". This service (*b*) is executed in parallel with the previous service "concatenate_and_write_in_files_by_3_Files". When three files will be created, their contents are merged into a new file and it is moved to the directory "folderA".

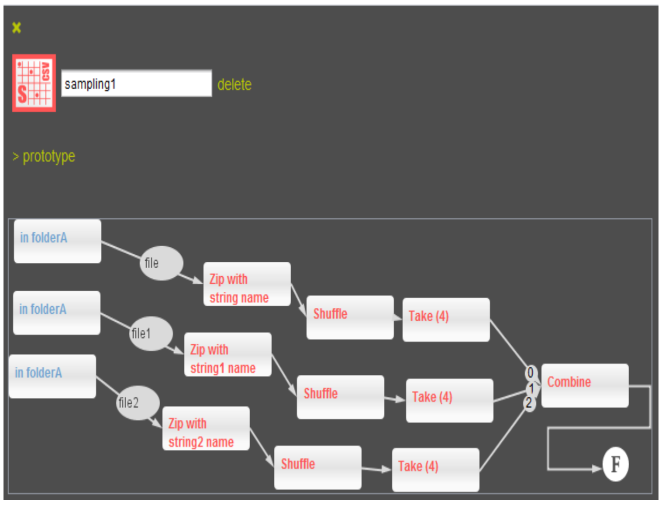

Figure 3. Designing Sampling in OpenMole.

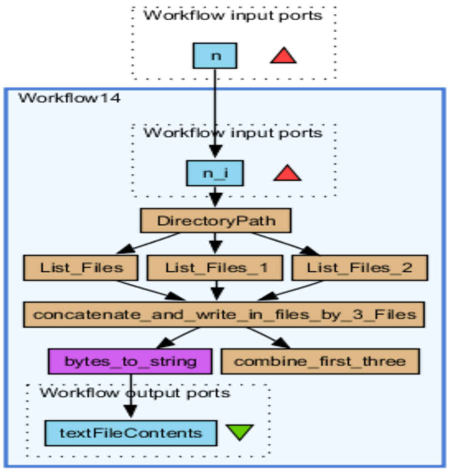

Figure 4. The workflow implementation in Taverna

The three files are deleted from "folderB".

The "bytes_to_string" service is a local text service which converts an array of bytes into a string. This is only used for showing the output of each new created file (total 64 of each workflow iteration). In order to be run on the Grid the workflow, the workflow must be installed as an application on the grid node and then to be executed remotely by using the Taverna Command Line Tool or the Taverna Server.

V. Conclusion and Future Work

In this paper we analyzed the most important characteristics of the workflow systems. Two scientific workflow systems were described and also the comparison between them was presented. We designed a workflow and it was implemented in OpenMole and Taverna. Also the implementation details and steps were given. The implementation logic is different. Taverna offers natural parallelism of processes, while in OpenMole the user should explicitly define that. The implementation of our workflow in Taverna was easier because we should not care about the parallel execution of the tasks. On the other hand, the visual interface for creating composite inputs in OpenMole is more user friendly and easier to be understood. Taverna is more suitable for designing complex workflows because it offers plenty of predefined services which can be used or modified. One of the greatest options in Taverna is that a service could be found and used only by specifying its URL. OpenMole provides simple and interactive interface which could be very useful for people that do not have so much experience in designing workflows. Also, the workflow execution on the Grid is much simpler in OpenMole. In OpenMole, only some basic parameters for the Grid certificate should be inserted, while in Taverna, it is necessary the workflow to be installed on any Grid node.

Our further research is oriented to analyzing and measuring the execution times of more complex workflows in both workflow systems. We will analyze where and which tasks are executed quickly and if possible how the workflows could be changed in order to provide better execution times.

References

[1] 12651-2, ISO/DIS 12651-2, Electronic imaging-Vocabulary-Part 2: Document workflow, http://www.iso.org/iso/catalogue_detail.htm?csnumber=42673, 2013.

[2] 1. Assoc for Info and Image Mgmt Intl, The Global Community of Information Professionals, http://www.aiim.org/community/wiki/view/Index-W, July 2013.

[3] X. Fei and S. Lu, “A dataflow-based scientific workfloow composition framework”, 1st ed., vol.5, IEEE Trans. Serv. Comput., 2012, pp. 45-58.

[4] J. Yu and R. Buyya, “A taxonomy of scientific work flow systems for grid computing”, 3rd ed., vol. 34, SIGMOD Rec., 2005, pp. 44-49.

[5] Y. Zhao, I. Raicu, and I. Foster, “Scientific workflow systems for 21st century, new bottle or new wine?”, in Proceedings of the 2008 IEEE Congress on Services - Part I. pp. 467-47, 2008.

[6] A. Barker and J. Van Hemert, “Scientific workflow: a survey and research directions”, in Proceedings of the 7th international conference on Parallel processing and applied mathematics, pp. 746-753, 2008.

[7] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, D. Withers, S, Owen, S. Soiland-Reyes, I. Dunlop, A. Nenadic, P. Fisher, J. Bhagat, K. Belhajjame, F. Bacall, A. Hardisty, A. Nieva de la Hidalga, M.P. Balcazar Vargas, S. Sufi, and C. Goble, “The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud”, Nucleic Acids Res, 2013.

[8] C.A. Goble and D.C. De Roure, “myexperiment: social networking for workflow using e-scientists”, in Proceedings of the 2nd workshop on Workows in support of large-scale science, pp. 1-2, 2007.

[9] T. Oinn, M. Greenwood, M. Addis, M.N. Alpdemir, J. Ferris, K. Glover, C. Goble, A Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M.R. Pocock, M. Senger, R. Stevens, A. Wipat and C. Wroe, “Taverna: lessons in creating a workflow environment for the life sciences: Research articles”, 10th ed., vol.18, Concurr. Comput. : Pract. Exper, 2006, pp. 1067-1100.

[10] R. Reuillon, M. Leclaire and S. Rey-Coyrehourcq, “Openmole, a workflow engine specifically tailored for the distributed exploration of simulation models”, 8th ed., vol. 29, Future Generation Computer Systems, 2013, pp. 1981-1990.

[11] R. Reuillon, F. Chuart, M. Leclaire, T. Faure, N. Dumoulin and D. Hill, “Declarative task delegation in openmole”, in International Conference on High Performance Computing and Simulation (HPCS), pp. 55-62, 2010.

[12] Free Software Community: Openmole, http://www.openmole.org/, July 2013.

[13] myGrid: Taverna, http://www.taverna.org.uk/, July 2013.