Procedia Computer Science

Volume 29, 2014, Pages 2137–2144

ICCS 2014. 14th International Conference on Computational Science

Enabling Global Experiments with Interactive Reconfiguration and Steering by Multiple Users

Luis Assuncao[1,2] and Jose C. Cunha[2]

[1]*Instituto Superior de Engenharia de Lisboa, Lisboa, Portugal*

[2]*CITI/DI Faculdade de Ciências e Tecnologia – Universidade Nova de Lisboa, Caparica, Portugal*

lass@isel.ipl.pt, jcc@fct.unl.pt

Abstract

In global scientific experiments with collaborative scenarios involving multinational teams there are big challenges related to data access, namely data movements are precluded to other regions or Clouds due to the constraints on latency costs, data privacy and data ownership. Furthermore, each site is processing local data sets using specialized algorithms and producing intermediate results that are helpful as inputs to applications running on remote sites. This paper shows how to model such collaborative scenarios as a scientific workflow implemented with AWARD (Autonomic Workflow Activities Reconfigurable and Dynamic), a decentralized framework offering a feasible solution to run the workflow activities on distributed data centers in different regions without the need of large data movements. The AWARD workflow activities are independently monitored and dynamically reconfigured and steering by different users, namely by hot-swapping the algorithms to enhance the computation results or by changing the workflow structure to support feedback dependencies where an activity receives feedback output from a successor activity. A real implementation of one practical scenario and its execution on multiple data centers of the Amazon Cloud is presented including experimental results with steering by multiple users.

Keywords: Scientific Workflows, Interactive Reconfigurations and Steering, Global Experiments on Cloud

1 Introduction

Workflows models and tools have increasingly been used for developing scientific applications due to their transparent support for application decomposition into multiple activities and for modeling the control-flow and data-flow interactions between those activities. Workflows are also useful for allowing the design of long-running experiments with multiple, possibly infinite iterations, as is typically required to support the scientific experimental process. However, there is currently a lack of workflow models and tools that support a flexible composition of distributed and interactive tasks without relying on a centralized enactment engine. Most of the existing approaches are based on a centralized enactment engine, although they may still allow the composition of distributed services

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.198

(Plociennik et al., 2013), (Wolstencroft & et al, 2013) or the support for allocating tasks into the appropriate distributed resources according to the users requirements (Li et al., 2012). There is also a need for approaches supporting user steering where each user is responsible for executing, monitoring and dynamically reconfiguring specific tasks without the need to restart or change the activities by other users. The above functionalities should also support Big Data applications deployments that cannot rely on a centralized approach for data storage and processing, for instance relying on a unique Cloud provider (Armbrust et al., 2009), (Yaser Mansouri & Buyya, 2013), (Vahi et al., 2013). Instead, large amounts of data are usually spread in distributed storage repositories deployed in multiple data centers and distributed computations must access those data. Furthermore, data movement between data centers raises critical issues related to the communication costs, and to data ownership and privacy. This has motivated efforts on distributed task scheduling and optimization of the data transfers between sites. On the other hand it motivated the development of solutions for in-situ processing of local data sets, for instance by relying on the processing of local data sets in each site, and only passing small amounts of intermediate data between sites, as we illustrate in this paper.

In previous work, we developed the AWARD model (Autonomic Workflow Activities, Reconfigurable and Dynamic) (Assuncao et al., 2012), to address the above concerns, and illustrated its use in distinct application scenarios, such as cloud data analytics (Goncalves et al., 2012), and fault recovery in long-running workflows (Assuncao & Cunha, 2013). In this paper we further discuss our approach, showing the flexibility of the AWARD model, for implementing such long-running workflows with support for dynamic reconfigurations according to the application demands, e.g. by dynamically changing the activity tasks for improving the performance, or for extending the functionality of the application algorithms. We illustrate this in a common application scenario where a distributed multi-site scientific experiment involves multiple interactive users, which are allowed to perform independent and autonomous modification of multiple application components. Furthermore we show how the dynamic inclusion of feedback loops into the ongoing computations is also supported, without requiring to stop and restart the entire experiment from the beginning.

In section 2 we give an overview of the AWARD framework. In section 3 we review the related work. In section 4 we describe the experimental evaluation scenarios as workflows and their execution results using the AWARD framework. In section 5 we present the conclusions.

2 Overview of the AWARD Framework

The AWARD framework supports workflow applications with each workflow activity executed as an autonomic process with independent control (named AWA) with a set of inputs and a set of outputs. Multiple AWAs can run in parallel on distributed infrastructures, e. g. on Clouds (Assuncao et al., 2012), (Goncalves et al., 2012), (Assuncao et al., 2014). The AWARD model of computation follows the Process Networks (PN) model (Kahn, 1974). The AWARD Space, based on a shared tuple space (Carriero & Gelernter, 1989), is used for the coordination of AWA interactions (data-flow and/or control-flow), where data-driven tokens produced by outputs of activities are stored until other activities consume these tokens in their inputs. For each AWA the application programmer develops the activity *Task* as a Java class that implements a generic interface. The AWARD model supports a total decoupling between *Task* development and the internal details of the AWA autonomic controller. The autonomic controller for each activity loads dynamically the specified *Task* class and passes the execution to the entry point of the *Task* with a list of arguments obtained from activity inputs and a list of activity parameters. AWARD supports a set of dynamic reconfiguration operators allowing structural and behavioral workflow changes. The currently implementation of the AWARD framework relies on a Java Virtual Machine (JVM) to support workflow activities, and Jess rules engine (Friedman-Hill, 2013) to support knowledge and decisions in each AWA autonomic controller. The AWARD Space can be mapped to different implementations, including centralized, as currently based

on the IBM TSpaces API (Lehman et al., 2001), or decentralized as based on Comet (Li & Parashar, 2005). Additionally a set of AWARD Tools allows to: i) Manage the runtime configuration settings, for instance definition of working directories and the location of the AWARD Space; ii) Manage the startup of one or more AWAs, which can be launched all in the same computing node or in different ones using partitions according to application dependent heuristics. In order to establish synchronization points there are tools allowing starting the AWAs in a standby state and tools to start their execution later; iii) Monitor the workflow execution by allowing the observation of tuples in the AWARD Space, which is very useful to get log and execution information for debugging or fault detection and recovery (Assuncao & Cunha, 2013). AWARD workflows run on a diversity of infrastructures, such as standalone computers, local networks, clusters and clouds, with minimal runtime requirements. In fact any operating systems with a Java Virtual Machine (JVM) installed and supporting the execution of remote commands, e.g. through the Secure Shell (SSH) protocol or remote desktop, can be used to host the execution of AWARD workflows. Dynamic reconfigurations are allowed using a set of operators as presented in (Assuncao & Cunha, 2013), (Assuncao et al., 2014). The AWARD model is neutral regarding any global coordination involving multiple AWA activities. Then any required coordination must be provided externally, according to the application semantics. Depending on the application scenarios this can be achieved by tools enforcing the consistency of the reconfiguration, or explicitly achieved by user agreements.

A more detailed discussion of the AWARD framework can be found in (Assuncao et al., 2012), (Assuncao & Cunha, 2013) and (Assuncao et al., 2014).

3 Related Work

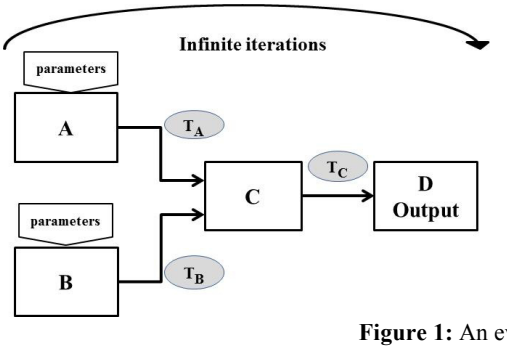
In the past decade there have been multiple proposals to enable scientific collaboration by running experiments in distributed infrastructures, for example by relying on web sites or portals (Hacker et al., 2011), (Bauer et al., 2012) and/or supporting large scale workflows on distributed infrastructures for instance on Clouds (Li et al., 2012), (Deelman et al., 2005) and tackling issues of distributed task scheduling (Plociennik et al., 2013), (Li et al., 2012) and workflow provenance (Gil et al., 2011). However among the currently open issues (Sonntag et al., 2010), (Ramakrishnan & Plale, 2010), (Mattoso et al., 2013), (Lu & Zhang, 2009) there is insufficient support for user steering of the long-running workflows, which is required to achieve a more effective collaboration life-cycle, where users should be allowed to continuously monitor the workflow activities, analyze intermediate data and perform dynamic changes in the workflow structure and behavior. In our previous work (Assuncao et al., 2012) we proposed the AWARD model and framework for supporting dynamic workflow reconfigurations, and we have shown its use for recovering from faulty Cloud services (Assuncao & Cunha, 2013) or for developing Cloud data analytics applications based on MapReduce (Goncalves et al., 2012). In this paper we illustrate the distinctive characteristics of the AWARD framework for supporting user steering of workflow activities and dynamic workflow reconfiguration in a distributed collaborative scenario. We show how AWARD allows multiple interactive users to separately launch, monitor and modify computations on local data sets, without requiring stopping and restarting the computation workflow from the beginning. Namely, we discuss how each user can change the task parameters or algorithms, and how under users agreement, they can dynamically change the workflow structure in order to include feedback loops. To the best of our knowledge these functionalities are not currently supported by some of the most widely used scientific workflow systems, including Kepler (Kepler project, 2013), Taverna (Wolstencroft & et al, 2013) or Pegasus (Deelman et al., 2005). Also, related approaches for dynamic workflow reconfigurations have less flexibility as they have been restricted to strategies for dynamic replacement of sub-workflows, which are predefined at design time, and used for instance for recovering from faults (Tolosana-Calasanz et al., 2010), (Chen & Deelman, 2012).

4 Evaluation Scenario

A scenario of a multinational distributed scientific experiment can be modeled as a generic workflow template (Figure 1) with the following characteristics: i) On each site A,B,C,D different users manage large data sets, such that each user uA, uB, uC, uD knows about their data models and develops/uses software components with appropriate algorithms to process the local data sets; ii) Data cannot be moved between sites due to data ownership, privacy and communication costs; iii) Users establish a multi-site cooperation project to conduct multinational experiments, where sites A and B continuously produce small data sets with the results obtained from their local experiments. Then site C uses the information received from sites A and B as inputs to new algorithms, while site D receives the results from site C.

These continuous experiments generate long-running iterative computations with infinite iterations, which can be functionally defined as a pipeline function $fexperiment^i = f^iD(f^iC(f^iA(),f^iB()))$ where f^ix represents the algorithm functionality at site $x \in \{A,B,C,D\}$ at iteration i.

In order to improve the results, each user should be allowed to change independently their algorithms without the need to stop or restart the global experiment, thus provoking only side effects on the results supplied as inputs to the other sites along the experiment chain. For instance, at iteration n a user should be able to dynamically reconfigure the local algorithm f^nx in order to enhance the results. Then on site C at iterations i and j with $i<j$ it is possible that $f^iC \neq f^jC$ meaning that the user uC, has changed the corresponding algorithm. Furthermore, as shown in Figure 2, at certain iteration the users in sites C and D can agree to introduce enhancements on their algorithms, using feedback information meaning that the function in site C shall be replaced to receive one more argument with the information from activity D. In this case and assuming this dynamic reconfiguration occurs at iteration k the functionality of the experiment at iteration $k+1$ can be described as $fexperiment^{k+1} = f^{k+1}D(f^{k+1}C(f^kDfeedback(),f^{k+1}A(),f^{k+1}B()))$ where $f^kDfeedback()$ represents the feedback information used on the enhancement of the algorithm of activity C. The discussion of global collaborative coordination in this scenario is out of scope of this paper, concerning how different users have achieved the agreement to apply the reconfiguration after iteration k and what kind of feedback data is supplied.



$T_A = (TaskA([parameter\ list]), \Delta_{tA})$ where Δ_{tA} is the elapsed execution time of activity A

$T_B = (TaskB([parameter\ list]), \Delta_{tB})$ where Δ_{tB} is the elapsed execution time of activity B

$T_C = (TaskC(T_A,T_B), \Delta_{tC})[\Delta_\varepsilon]$ where Δ_{tC} is the elapsed execution time of activity C and Δ_ε is the accumulated elapsed execution time, i.e., $\Delta_\varepsilon = \max(\Delta_{tA},\Delta_{tB}) + \Delta_{tC}$

Figure 1: An evaluation scenario

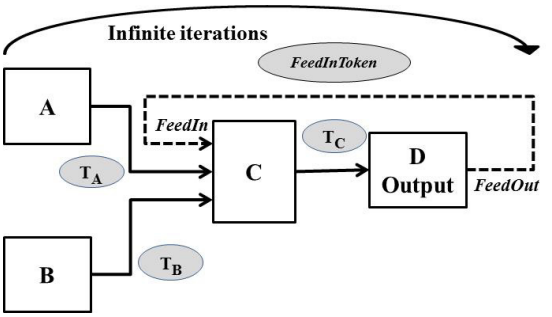
The main characteristics of this scenario are: i) The large data sets remain on their local sites and there are not large amounts of data moving between sites. Only small amounts of intermediate data are moved, typically to enable the next activities on the experiment chain; ii) The local activities are

independently launched and monitored by different users; iii) The experiment is a long-running workflow with infinite iterations, where the activities can be executed at different paces; iv) Users should be able to steering local activities by dynamic changes on algorithms and their parameters.

To demonstrate the execution of applications similar to the above scenario the activities A, B and C should be deployed and executed independently, for example at Amazon cloud infrastructure. The activity D should be executed on a scientist desktop machine in order to output the iteration results. The *Tasks* of activities A, B and C emit for each iteration respectively the tokens T_A,T_B,T_C while *TaskD* receives the tokens T_C showing them through a graphical user interface component allowing a user to monitor the intermediate results. Furthermore different users on different computers should be able to monitor, change parameters and change algorithms on other activities. To illustrate the case of feedback dependencies we use four dynamic reconfiguration operations (Figure 2a) leading to the workflow depicted in Figure 2b.

- ***CreateOutput*** *FeedOut* on activity D;
- ***ChangeTask*** on activity D to produce tokens to output *FeedOut* ;
- ***CreateInput*** *FeedIn* on activity C;
- ***ChangeTask*** on activity C to receive one more argument. Then *TaskC* is changed to emit tokens

$T_C=(TaskC(FeedInToken,T_A,T_B), \Delta_{tC})[\Delta_\varepsilon]$ where *FeedInToken* may be the date and time of the computer where activity D runs.



a) The reconfiguration operations **b)** The workflow after the reconfiguration

Figure 2: The workflow after performing a dynamic reconfiguration to introduce feedback dependencies

As a proof of concept we demonstrate how the above scenario requirements are met by the AWARD framework with the following mappings for executing the workflows of Figure 1 and Figure 2: Activities A and B respectively with algorithms *TaskA* and *TaskB* were launched on Amazon EC2 Linux virtual machines in US data centers; Activity C on the Amazon EC2 Linux virtual machine in Ireland data center; Activity D (output) is launched with a graphical interface (GUI) on a local desktop computer allowing a user constantly monitoring the results. To experiment we developed the *TaskA* to emit periodically, on each 2000 milliseconds the token $T_A=(TaskA([parameter\ list]), \Delta_{tA})$ and *TaskB* to emit periodically, on each 1000 milliseconds the token $T_B=(TaskB([parameter\ list]), \Delta_{tB})$. In activity C with the algorithm *TaskC* reads tokens from its two inputs, imposes a delay of 4000 milliseconds and then emits the token $T_C=(TaskC(T_A,T_B), \Delta_{tC})[\Delta_\varepsilon]$ converted as a string. Figure 3 presents the graphical user interface of activity D (output) where we can see that after iteration 25 the user of activity B changed the value *b1* of *TaskB* parameter to a new value *b1new*. At iteration 30 the user of activity C changed to a new task *TaskCnew* and at iteration 35 the user of activity A changed the task to *TaskAnew* leading to a decreased elapsed execution time. Firstly the new task of activity C has decreased from 4000 milliseconds to 2000 milliseconds and secondly the task of activity A has decreased from 2000 milliseconds to 1000 milliseconds, illustrating how algorithm enhancements can be achieved by dynamic reconfigurations. For these simple tasks we get small overheads, few tens of milliseconds, for loading and instantiating the new tasks. A discussion of overheads is out the scope of this paper.

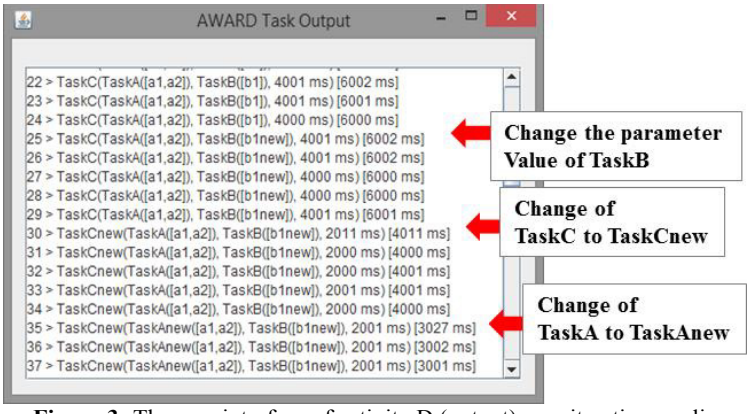


Figure 3: The user interface of activity D (output), one iteration per line

To evaluate the scenario of Figure 2, we submitted the reconfiguration (Figure 2a) at iteration 50. We assume an agreement between users on activities C and D. Then at iteration 50 the activity D emits a token with the current time on the new port *FeedOut* and as shown in Figure 4 the new task of C (*TaskCwithFeedback)* at iteration 51 receives three arguments where the first one is the token (current date and time) sent by activity D.

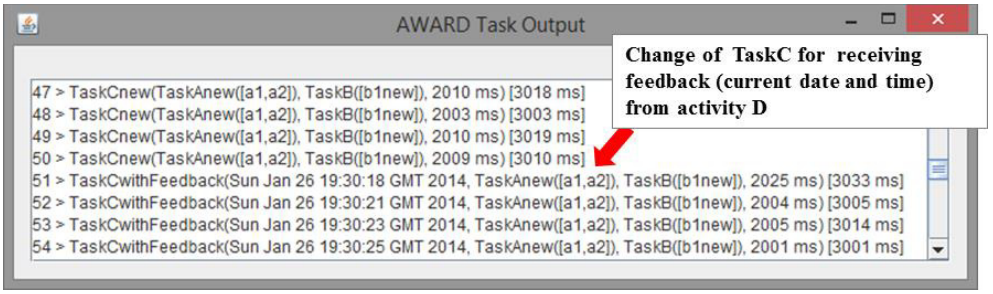


Figure 4: The user interface of activity D (output) showing the result after feedback reconfiguration

Despite the apparent simplicity of these functionalities, the execution of this workflow enables the following main characteristics: i) The execution is distributed by different data centers; ii) Each activity is launched, monitored and dynamically reconfigured by different users.

These experimental results clearly demonstrate the feasibility of executing global workflow experiments allowing users on different sites to dynamically reconfigure their activities e.g. changing tasks to new algorithms or the structure of the workflow to support feedback loops.

5 Conclusions

This paper discusses how the AWARD model and framework was used for the modeling and implementation of a common collaborative scenario where geographically distributed activities interact in a long-running computation towards achieving scientific results. We modeled this scenario as a distributed workflow such that each activity produces small intermediate data that is passed on to other sites. Furthermore in each site different users are able to independently develop and change the activities that consume those intermediate data as their inputs for their own computations. The feasibility of this approach was confirmed experimentally by running an AWARD workflow with distributed activities deployed on data centers in different countries of the Amazon Cloud, and by

discussing how this experiment enabled the different users on different sites to dynamically and independently reconfigure their activities. For example, users can dynamically change the task algorithms for instance to perform local optimizations, and the structure of the workflow can be reconfigured in order to include feedback dependencies between workflow activities without having to stop and restart the entire workflow from the beginning. These capabilities are innovative and to the best of our knowledge they are not currently supported by other workflow tools. From a practical perspective, the developed experiment is relevant because it can be instantiated to represent real application scenarios modeled as workflows with geographically distributed activities.

Acknowledgements

Thanks are due to PEst-OE/EEI/UI0527/2011, CITI/FCT/UNL-2011-2012 and to GIATSI-MIELE project, ISEL/IPL.

References

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., & Zaharia, M. (2009). *Above the Clouds: A Berkeley View of Cloud Computing*. Technical report.

Assuncao, L. & Cunha, J. C. (2013). Dynamic Workflow Reconfigurations for Recovering from Faulty Cloud Services. In *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 88–95).

Assuncao, L., Goncalves, C., & Cunha, J. C. (2012). Autonomic Activities in the Execution of Scientific Workflows: Evaluation of the AWARD Framework. In *9th International Conference on Autonomic Trusted Computing (ATC)* (pp. 423–430).

Assuncao, L., Goncalves, C., & Cunha, J. C. (2014). Autonomic Workflow Activities: The AWARD Framework. *To appear in Int. Journal of Adaptive, Resilient, and Autonomic Systems*, 5(2).

Bauer, M., McIntyre, S., Sherry, N., Qin, J., Maxwell, D., Liu, D., Matias, E., Fuller, M., Xie, Y., & Mola, O. (2012). Experimenters Portal: The Collection, Management and Analysis of Scientific Data from Remote Sites. In *10th International Workshop on Middleware for Grids, Clouds and e-Science* (pp. 7:1–7:6).

Carriero, N. & Gelernter, D. (1989). Linda in Context. *Communication of the ACM*, 32(4).

Chen, W. & Deelman, E. (2012). Fault Tolerant Clustering in Scientific Workflows. In *IEEE World Congress on Services* (pp. 9–16).

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G., Good, J., Laity, A., Jacob, J., & Katz, D. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Prog.*, 13, 219–237.

Friedman-Hill, E. (2013). Jess: the Rule Engine for the Java Platform, Online: http://www.jessrules.com/jess/.

Gil, Y., Ratnakar, V., Kim, J., Moody, J., Deelman, E., Calero, P., & Groth, P. (2011). Wings: Intelligent Workflow-Based Design of Computational Experiments. *Intellig. Syst.*, (pp. 62–72).

Goncalves, C., Assuncao, L., & Cunha, J. C. (2012). Data Analytics in the Cloud with Flexible MapReduce Workflows. In *IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)* (pp. 427 –434).

Hacker, T., Eigenmann, R., Bagchi, S., Irfanoglu, A., Pujol, S., Catlin, A., & E., R. (2011). The NEEShub Cyberinfrastructure for Earthquake Engineering. *Computing in Science Engineering*, 13(4), 67–78.

Kahn, G. (1974). The Semantics of a Simple Language for Parallel Programming. In *Information Processing* (pp. 471–475).

Kepler project (2013). Kepler web site, Online: https://kepler-project.org/.

Lehman, T. J., Cozzi, A., Xiong, Y., Gottschalk, J., Vasudevan, V., Landis, S., Davis, P., Khavar, B., & Bowman, P. (2001). Hitting the Distributed Computing Sweet Spot with TSpaces. *Computer Networks*, 35(4), 457–472.

Li, X., Calheiros, R., Lu, S., Wang, L., Palit, H., Zheng, Q., & Buyya, R. (2012). Design and Development of an Adaptive Workflow-Enabled Spatial-Temporal Analytics Framework. In *IEEE International Conference on Parallel and Distributed Systems* (pp. 862–867).

Li, Z. & Parashar, M. (2005). Comet: A Scalable Coordination Space for Decentralized Distributed Environments. In *2nd Int. Workshop on Hot Topics in Peer-to-Peer Systems* (pp. 104–112).

Lu, S. & Zhang, J. (2009). Collaborative Scientific Workflows. In *IEEE International Conference on Web Services* (pp. 527–534).

Mattoso, M., Ocana, K., Horta, F., Dias, J., Ogasawara, E., Silva, V., Oliveira, D., Costa, F., & Igor, A. (2013). User-steering of HPC Workflows: State-of-the-art and Future Directions. In *2nd ACM Workshop on Scalable Workflow Execution Engines and Technologies* (pp. 4:1–4:6).

Plociennik, M., Zok, T., & Altintas, I. (2013). Approaches to Distributed Execution of Scientific Workflows in Kepler. *Fundamenta Informaticae*, 128, 281–302.

Ramakrishnan, L. & Plale, B. (2010). A Multi-dimensional Classification Model for Scientific Workflow Characteristics. In *1st International Workshop on Workflow Approaches to New Data-centric Science* (pp. 4:1–4:12).

Sonntag, M., Karastoyanova, D., & Deelman, E. (2010). Bridging the Gap between Business and Scientific Workflows: Humans in the Loop of Scientific Workflows. In *IEEE Sixth International Conference on e-Science* (pp. 206–213).

Tolosana-Calasanz, R., Banares, J. A., Rana, O. F., Alvarez, P., Ezpeleta, J., & Hoheisel, A. (2010). Adaptive exception handling for scientific workflows. *Concurrency Computation: Practice Experience*, 22(5), 617–642.

Vahi, K., Rynge, M., Juve, G., Mayani, R., & Deelman, E. (2013). Rethinking Data Management for Big Data Scientific Workflows. In *IEEE International Conf. on Big Data* (pp. 27–35).

Wolstencroft, K. & et al (2013). The Taverna Workflow Suite: Designing and Executing Workflows of Web Services on the Desktop, Web or in the Cloud. *Nucleic Acids Research, Online: http://nar.oxfordjournals.org/content/early/2013/05/02/nar.gkt328.abstract*.

Yaser Mansouri, A. N. T. & Buyya, R. (2013). Brokering Algorithms for Optimizing the Availability and Cost of Cloud Storage Services. In *IEEE 5th International Conference on Cloud Computing Technology and Science* (pp. 581–589).