# Usability of Scientific Workflow in Dynamically Changing Environment

Anna Bánáti[1(✉)], Eszter Kail[1], Péter Kacsuk[2,3], and Miklos Kozlovszky[1,2]

[1] John von Neumann Faculty of Informatics, Biotech Lab, Obuda University, Bécsi str. 96/b., Budapest H-1034, Hungary
{banati.anna,kozlovszky.miklos}@nik.uni-obuda.hu
[2] LPDS, MTA SZTAKI, Kende str. 13-17, Budapest H-1111, Hungary
[3] University of Westminster, 115 New Cavendish Street, London W1W 6UW, UK
kacsuk@sztaki.mta.hu

**Abstract.** Scientific workflow management systems are mainly data-flow oriented, which face several challenges due to the huge amount of data and the required computational capacity which cannot be predicted before enactment. Other problems may arise due to the dynamic access of the data storages or other data sources and the distributed nature of the scientific workflow computational infrastructures (cloud, cluster, grid, HPC), which status may change even during running of a single workflow instance. Many of these failures could be avoided with workflow management systems that provide provenance based dynamism and adaptivity to the unforeseen scenarios arising during enactment. In our work we summarize and categorize the failures that can arise in cloud environment during enactment and show the possibility of prediction and avoidance of failures with dynamic and provenance support.

**Keywords:** Scientific workflow · Dynamic workflow management system · Distributed computing · Cloud failures · Fault tolerance

## 1 Introduction

Over the last years the e-Science is gaining more and more ground. Existing e-Science experiments being (also called in silico experiments) really data and process intensive, it is inevitable to be executed in High Performance Computing (HPC) environments, such as clusters, grids and more recently clouds. Thanks to the virtualized environments, one of the main advantages of the clouds is the elasticity and the availability of resources. [1]

The enormous number of nodes, the complexity of the infrastructure and the high end computing resources needed to support scientific applications executed in the cloud, bring increased potential for failures and performance problems. On one hand due to the virtualization the resource management appears simpler from the applications' point of view and many of the system level failures are hidden from the users. However, on the other hand these failures could have significant impact on the execution of scientific workflows and because of their invisibility workflow monitoring,


L.M. Camarinha-Matos et al. (Eds.): DoCEIS 2015, IFIP AICT 450, pp. 129–136, 2015.
DOI: 10.1007/978-3-319-16766-4_14

analyzing or reproducibility is even more difficult, than in the case of other distributed systems [2]. Therefore it is even more crucial for the scientists to capture more and more parameters and data about the execution (provenance data) and about the environmental conditions since reproducibility and knowledge sharing in the scientists' community is one of the main challenges that scientific workflow management systems have to face with.

In our PhD research work we investigate provenance data analyses supporting dynamic executions of workflows and fault tolerance techniques. In this work we have summarized and classified the potential failures that may arise during the execution of scientific workflows in the cloud.

The contributions of this paper are:

1. Summarize and classify the emerging failures in the cloud during scientific workflow execution.
2. Examine the possibility of prediction and avoidance of failures with dynamic support.
3. Examine the possibility of prediction and avoidance of failures with provenance support.

The rest of the paper is organized as follows. The next section presents the contribution of our work to cloud based solutions, than we provide a short background and overview about works related to our research. Section 4 presents the dynamic requirements of scientific workflow management systems at different levels and in different phases of the workflow lifecycle. In section 5 we analyze the failures and give a solution to handle or avoid them. Finally we summarize our conclusions and reveal the potential future research directions.

## 2 Relationship to Cloud-Based Solutions

One of the main challenges in cloud systems is to ensure the reliability of job execution in the presence of failures. Cloud applications may span thousands of nodes and run for a long time before being aborted, which leads to the wastage of energy and other resources. [3, 4, 5]

In order to minimize failed execution and thus the multiple re-executions of the same workflow fault tolerance techniques must be investigated and supported. Since the numbers of failures are high and the types of them vary, general methods can hardly exist. In this work we have summarized and classified the most frequent failures that can arise during execution time on parallel and distributed environment focusing on cloud environment solely. We classified the potential failures into four different levels: cloud, workflow, task and user level. After categorizing the potential failures, we show how dynamic behavior and provenance support can give solutions for avoiding and preventing them or to recover from situations caused by failures and problems that cannot be foreseen or predicted.

## 3 Related Works

The analyses and recovery or avoidance strategies of failures arising during scientific workflow execution is a widely dealt research area. Many research works focus on different fault tolerance techniques and fault tolerance analyses of workflow management systems. However most of these works deal only with different hardware failures, but higher level failures are not mentioned or specified, in general only the states of jobs are differentiated (finished, failed, and killed). The cause of the failures is usually not investigated.

Vishwanath and Nagappan in their work [6] investigate the number and the cause of possible hardware failures in modern day data centers, which consist of thousands of network components, such as servers, routers and switches. These components have to communicate with each other to manage tasks in order to provide highly available cloud computing services. Consequently the number of hardware failures can be surprisingly high. Researchers in this work set up a hierarchical reliability model which helps analyzing the impact of server -, networking equipment and individual component failures in order to decrease hardware costs and to design a more fault tolerant system.

Bala and Chane in their work [7] present the existing proactive and reactive fault tolerant techniques that are used in the various cloud systems. They differentiate the reactive techniques into seven categories (Checkpointing/Restart, Replication, Job migration, SGuard, Retry, Task resubmission, user defined exception handling), while proactive techniques in three categories (Software rejuvenation, Proactive Fault Tolerance using Self- Healing and Proactive Fault Tolerance using Preemptive Migration).

Plankensteiner et al. [8] investigated the fault tolerance of Grid workflow management systems. They also give an overview of the existing fault tolerant techniques in different grid systems. The detection and avoidance of failures as well as recovery methods are also discussed in the paper. They give a deep and detailed taxonomy about the failures arising during enactment. This taxonomy grounds for our research work as well. To improve fault tolerance they suggest the use of light-weight and heavy-weight checkpoints, the storing of multiple instances of data and tasks and the use of alternate tasks.

## 4 Dynamic Scientific Workflow

The Lifecycle of scientific workflows can be partitioned into disjunctive phases (hypothesis generation, design, instantiation, execution, result analyses) [9 10, 11, 12] with the help of which the development, handling and enactment steps and requirements can be clearly defined and understood.

In one of our earlier work [13] we have summarized the requirements of a dynamic workflow management system regarding three phases (design, instantiation, execution) of the workflow lifecycle. In each phase (which can be interpreted as different abstract level as well) we have differentiated additional levels in order to have a deeper insight about this topic. [table 1.]

**Table 1.** The different levels of dynamism and the various solutions and methods that can be used according to the levels

| | | | | |
|---|---|---|---|---|
| **design phase (abstract workflow level)** | **system level** | **composition level** | **task level** | |
| | - black boxes<br>- advance and late modelling technique | - language or graph structure | - modularity, reusability | |
| **instantiaton phase (concrete workflow level)** | **system level** | **task level** | **workflow level** | |
| | - incremental compilation<br>- various protocol support<br>- provenance based sched.<br>- multi instance activity | - task based sched.<br>- late binding of data | - partitioning to sub workflows<br>- parameter sweep appl.<br>- wf based sched.<br>- mapping adaptation | |
| **execution phase (execution level)** | **system level** | **task level** | **workflow level** | **user level** |
| | - exception handling<br>- breakpoints<br>- checkpoints<br>- provenance based decisions<br>- monitoring, logging<br>- dynamic resource allocation | - dynamic resource allocation | - alternate task<br>- change the model | - user intervention |

## 5 Classifying Arising Failures in the Cloud and Analyzing Dynamic Solutions

As we mentioned earlier, during the different phases of the workflow lifecycle we have to face many types of failures, which lead unfinished task or workflow execution. In these cases the users, instead of getting the appropriate results of their experiment, the workflow process aborts and in general the scientist does not have knowledge about the cause of the failure.

The arising failures are examined at four abstract levels, namely the cloud level, task level, workflow level and user level. [Table 2.], [8, 15, 16, 17] The cloud level deals on the one hand with errors and problems related to the infrastructure (hardware or network failures), on the other hand with problems related to configuration parameters, which manage the execution.

In the table after the possible failures there is a „→" sign inserted and then the potential solutions that can be carried out by a dynamic system are presented.

### 5.1 Cloud Level

Virtualized resource management or "cloud" technologies can simplify the scientific application's view of the resources, but do not remove the inherent complexity of large-scale applications. Thanks to the virtualized environments many levels of failures are hidden from the scientific application, and thus hard to monitor and analyze. [6] There are several issues related to computing, memory or storage resources, which are usual in grid, cluster or HPC systems, but thanks to the virtualization in cloud environment they become irrelevant. For example disk quota exceeded, out of memory, out of disk space and CPU time limit exceeded.

Cloud level is divided into two sublevels, namely the hardware level and the configuration level.

**Hardware Level**

The infrastructure is constructed from thousands of servers, routers and switches connected to each other. They communicate with each other to process the jobs. The servers consist of multiple hardware elements for example hard disks, memory modules, network cards, and processors etc., which are capable of failing. While the probability of seeing any such failure in the lifetime (typically 3-5 years in industry) of a server can be very small, the probability to meet failures in the datacenter, the number of components that could fail at any given instant is can be very high. At such a large scale, hardware component failure is rather normal than an exception. [6]

Cloud providers apply hardware redundancy and fault tolerance techniques to handle them transparently. The most popular fault tolerance technique is checkpointing and job migration or replication in order to prevent or to recover from failures. Some systems provide automatic and periodic checkpointing but a dynamic system should give the possibility of dynamic and user defined checkpointing techniques as well.

**Configuration Level**

At cloud level we have to face with not only hardware failures, but also task submission -, authentication -, service unreachable and file staging failures. At configuration level it is also true, that the checkponting technique can decrease the waste derived from any failures. In addition, the execution of a task can be failed, if some configuration policy is not suitable. For example the system should periodically check the queues and guarantee, that all the jobs in the queues will be processed in a limited time. Another example can be the number of job resubmissions, which can influence the success of job executions, when it is a constant value. In order to be able to prevent these types of failures we can adjust the configuration parameters based on provenance information, and we can dynamically change the settings during workflow execution. [18]

### 5.2 Workflow Level

At workflow level we mention those failures, that have impact on the whole workflow and can corrupt the whole execution of the workflow. Independently from the type of failures, the scientists can reduce the severity of waste, if they can or they have the possibility to build the abstract workflow model from smaller modules or sub-workflows. In this way the effect of failures is isolated to a small piece of workflow.

At this level, unavailable input data, invalid input data and failed data transfer can lead to errors. To handle these faults, the common fault tolerance techniques, namely data and file replication is one of the best solutions.

**Table 2.** The failures at the different levels and their possible solution or optimization

| | | | | |
|---|---|---|---|---|
| **design phase (abstract wf level)** | | | **user level** | |
| | | | - infinite loop<br>**→advanced language and modeling support** | |
| **instantiation level** | **cloud level failures** | **task level failures** | **workflow level failures** | |
| | - HW failures<br>- network failures<br>- file not found<br>- Network congestion<br>- task submission failure<br>**→checkpoint**<br>- authentication failed<br>**→user intervention**<br>- file staging<br>- Service not reachable | - Incorrect output data<br>- Missing shared libraries | - Input data not available<br>**→data and file replication**<br>- Input error<br>**→data and file replication**<br>- Data movement failed<br>**→checkpoint** | |
| **Execution level** | **cloud level failures** | **task level failures** | **workflow level failures** | **user level failures** |
| | - HW failures<br>- network failures.<br>- file not found<br>- Job hanging in the queue of the local resource manager**→dynamic resource brokering**<br>- Job lost before reaching the local resource manager **→ dynamic resource brokering + user intervention** | - job crashed<br>**→user intervention, alternate task, checkpoint**<br>- deadlock/livelock<br>**→dynamic resource allocations, checkpoint**<br>- uncaught exception (numerical)<br>**→exeption handling + user intervention** | - Data movement failed<br>**→checkpoint** | – User-definable exception |

### 5.3 Task Level

The task level failures can influence the execution of only one task, and the impact of any failures does not cover the whole workflow. In generally it is true at this level (and at workflow level, too), that in the bulk of the cases the possibility of user intervention is the most helpful and efficient tools to handle the arising failures. On the one hand the user intervention can occur on the fly during execution, if it concerns to only one or a few threads of the whole workflow. On the other hand when a problem affects the entire execution than it has to be suspended and a checkpoint has to be inserted. If user intervention is supported, at these points [19] the user has the possibility to solve certain failures. In addition the scientist can make some changes, for example modify filtering criteria, change parameters or input data, restart a given task or a whole workflow or even do some time management actions.

In the instantiation phase incorrect output data or missing shared libraries can abort the execution.

## 6 Conclusion and Future Work

In order to minimize the effects of failures on the execution of scientific workflows, the failures should be discovered, handled or even predicted. In our work we have investigated the arising failures of scientific workflow execution in cloud environment at the different operation levels. We analyzed the possible solutions to prevent or to handle these failures transparently from the user. We have showed the methods and tools with which most of the problems can be solved. We have also highlighted that with provenance support the problems can be even more effectively handled or prevented. In our future work we would like to prove the effectiveness of the dynamic system and the provenance support with a mathematical model. and based on this model we would like to develop new proactive fault tolerance techniques.

## References

1. da Cruz, S.M.S., Paulino, C.E., de Oliveira, D., Campos, M.L.M., Mattoso, M.: Capturing distributed provenance metadata from cloud-based scientific workflows. Journal of Information and Data Management **2**(1), 43 (2011)
2. Samak, T., Gunter, D., Goode, M., Deelman, E., Juve, G., Silva, F., Vahi, K.: Failure analysis of distributed scientific workflows executing in the cloud. In: 2012 8th International Conference on Network and Service Management (CNSM) and 2012 Workshop on Systems Virtualiztion Management (SVM), pp. 46–54. IEEE (2012)
3. Liang, Y., Zhang, Y., Jette, M., Sivasubramaniam, A., Sahoo, R.: BlueGene/L failure analysis and prediction models. In: International Conference on Dependable Systems and Networks (DSN), pp. 425–434 (2006)
4. Pham, C., Cao, P., Kalbarczyk, Z., Iyer, R.K.: Toward a high availability cloud: Techniques and challenges. In: IEEE/IFIP 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W), pp. 1–6. IEEE (2012)

5. Chen, X., Lu, C. D., Pattabiraman, K.: Failure analysis of jobs in compute clouds: A google cluster case study. In: The International Symposium on Software Reliability Engineering (ISSRE). IEEE (2014)
6. Vishwanath, K.W., Nachiappan, N.: Characterizing cloud computing hardware reliability. In: 1st ACM Symposium on Cloud Computing. ACM (2010)
7. Bala, A., Chana, I.: Fault tolerance-challenges, techniques and implementation in cloud computing. IJCSI International Journal of Computer Science Issues **9**(1) (2012) ISSN (Online): 1694-0814
8. Plankensteiner, K., Prodan, R, Fahringer, T., Kertesz, A., Kacsuk, P.: Fault-tolerant behavior in state-of-the-art grid workflow management systems (2007)
9. Bahsi, E.M.: Dynamic Workflow Management For Large Scale Scientific Applications, PhD Thesis, B.S., Fatih University (2006)
10. Deelman, E., Gil, Y., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J.: Examining the challenges of scientific workflows. IEEE Computer **40**(12), 26–34 (2007)
11. Deelman, E., Gil, Y.: Managing large-scale scientific workflows in distributed environments: Experiences and challenges. In: e-Science, p. 144 (2006)
12. Ludäscher, B., Altintas, I., Bowers, S., Cummings, J., Critchlow, T., Deelman, E., Vouk, M.: Scientific process automation and workflow management. In: Scientific Data Management: Challenges, Existing Technology, and Deployment. Computational Science Series, pp. 476–508 (2009)
13. Kail, E., Bánáti, A., Karóczkai, K., Kacsuk, P., Kozlovszky, M.: Dynamic workflow support in gUSE. In: Proceedings of the 37th International Convention, MIPRO (2014)
14. Kail, E., Bánáti, A., Kacsuk, P., Kozlovszky, M.: Provenance based adaptive and dynamic workflows. In: 15th IEEE International Symposium on Computational Intelligence and Informatics, pp 215–219. IEEE Press, Budapest (2014)
15. Das, A.: On Fault Tolerance of Resources in Computational Grids. International Journal of Grid Computing & Applications **3**, 1–10 (2012)
16. Mouallem, P.A., Vouk, M.: A fault tolerance framework for kepler-based distributed scientific workflows. North Carolina State University (2011)
17. Alsoghayer, R.A.: Risk assessment models for resource failure in grid computing. Thesis, University of Leeds (2011)
18. Bánáti, A., Kacsuk, P., Kozlovszky, M.: Towards flexible provenance and workflow manipulation in scientific workflows. In: Proceedings of CGW 2014 (2014)
19. Kail, E., Kacsuk, P., Kozlovszky, M.: A novel approach to user-steering in scientific workflow. In: Proceedings of CGW 2014 (2014)