Towards the Scientific Cloud Workflow Architecture*

Rafael Tolosana Calasanz, José A Bañares, José-Manuel Colom, Mustapha Ait-Idir, Nazim Agoulmine

Towards the Scientific Cloud Workflow Architecture*

Rafael Tolosana Calasanz,[1], José A. Bañares,[1], José-Manuel Colom,[1], Mustapha Ait-Idir,[2], and Nazim Agoulmine[2]

[1] Universidad de Zaragoza, Zaragoza, Spain
[rafaelt, banares, jm] @unizar.es
[2] IBISC Lab, University of Evry Val d'Essonne, Evry Val d'Esssonne, France
[maitidir, nazim.agoulmine] @ibisc.univ-evry.fr

Abstract

Scientific workflows emerged as a technology that enables scientists to undertake computational scientific experiments. Workflow enactors map workflow tasks onto distributed resources, hiding the inherent complexity of distributed infrastructures to the users. In the past, while the emphasis has been focused in adapting the workflow structure onto the resources, today the emergence of the cloud computing paradigm enables us to adapt the resources to the workflow tasks based on their characteristics. In this paper, we examine the concept of Cloud Scientific Workflow, and propose a new architectural approach based on autonomic principles, guided by a combination of high-level and low-level policies. High-level policies enable the workflow enactor to choose among a number of workflow structure transformations that better suit the underlying resources dynamically depending on the context, whereas low-level policies enable the autonomic resource manager to adjust the required computational power to the workload derived from a scientific workflow specification, exploiting the cloud elasticity property, and to cope with performance fluctuations or unexpected events in the cloud infrastructure. The novelty of our approach is the combination of both policies that can lead to higher degrees of dynamism. The key enabling architectural components for such a dynamism are the petri-net based performance models for implementing high-level policies and MOSt-CB system for the adaptation to multi-cloud environments.

1 Introduction

Over the last years, scientific workflows have been used as high-level abstractions of computational scientific experiments. Such experiments often conduct data analysis operations or complex simulations, and described in terms of a set of tasks and their data and control dependencies. In order to execute them, workflow engines often map such tasks onto third-party computational distributed resources. During the pre-cloud era, the emphasis of the proposed mapping solutions has been in adapting the workflow structure onto the resources. Such mapping process has involved a number of intertwined processes: (i) the provisioning of the required resources, (ii) the transformation of the workflow structure for optimizing the workflow execution prior to the mapping (i.e. by merging tasks and increasing their granularity), and (iii) the scheduling of tasks, which consists of assigning the transformed workflow to the resources and orchestrating their execution. Overall, the mapping considers user-defined Quality of Service (QoS) metrics and the scheduling aims to enforce them.

*This work was supported in part by: the Industry and Innovation department of the Aragonese Government and European Social Funds (COSMOS group, ref. T93) and the Spanish Ministry of Economy (program "Programa de I+D+i Estatal de Investigación, Desarrollo e innovación Orientada a los Retos de la Sociedad" –TIN2013-40809-R)

Nevertheless, current practice of workflow specification often makes use of specific, low-level codes used for the execution, and on the specific workflow catalogues where they were published [7]. Not only does this practice hinder workflow reusability, but also the flexibility required for executing such workflows when the target infrastructures are changing. Moreover, many workflow engines have integrated different target infrastructure middleware (e.g. Condor, PBS, etc.), and delegate to them the execution and the resource management activities, while very rarely workflow structure is modified at runtime. In consequence, the recovery strategies and mechanisms under unexpected situations, such as resource unavailability, failures, or performance degradation is tightly coupled to this specific middleware [16]. In many cases, however, a much more flexible approach would exploit different ways of mapping a workflow, so that the workflow enactor, if needed, can find an alternative to finalize the execution and eventually satisfy users specifications. Moreover, scientific experiments are exploratory in nature, and as recognised in [5], more flexible execution approaches should be adopted, involving, for instance, human intervention in taking some decisions.

The emergence of the cloud computing paradigm enables a complete different approach in the design of a workflow system architecture, since cloud computational resources, namely CPU, communication network, or storage, can be provisioned on demand as needed. While the emphasis in workflow systems has been in transforming the workflow structure onto the resources, the possibility of adapting the resources to the workflow tasks can now be also considered. To the best of our knowledge, current use of cloud technologies by scientific workflow systems is not fully exploiting the cloud paradigm to address the dynamism and flexible requirements of scientific workflows. The existing proposals often replace the physical resources with virtual resources, and the main novelty is on (de-) provisioning the virtual resources exactly when required [11, 13].

In this paper, we examine the concept of Cloud Scientific Workflow, which re-visits traditional workflow architectures and explores new architectural organizations and blueprints. Our architectural approach is based on autonomic principles, therefore, its ultimate goal is to reduce human intervention in mapping workflow tasks. It also aims at reducing the perceived complexity by enabling the autonomic platform to manage workflows in accordance with a combination of high-level and low-level policies. High-level policies enable the workflow enactor to choose among a number of workflow structure transformations that better suit the underlying resources. Hence, such policies allow the platform to (i) interpret the application specifications; (ii) map the specifications onto the target computing infrastructure, so that the workflows are executed and their QoS, as specified in their Service Level Agreements (SLA), enforced; and (iii) adapt automatically such previously established mappings when unexpected behaviors violate the QoS. Such adaptations may involve modifications in the arrangement of the computational infrastructure. On the other hand, low-level policies will enable an autonomic resource manager to: (i) adjust the required computational power to the workload derived from a scientific workflow specification, exploiting the Cloud elasticity property; and (ii) cope with performance fluctuations or unexpected events in Cloud infrastructures, making the mapping workflow-resources an adaptive process. We use a model that support reasoning on the potential workflow-resource mappings and provides with real-time performance information and model refinement capabilities. MOSt-CB system in involved to map the workflow (i.e tasks) specs into (IaaS) Cloud resources (e.g. processors, storage and network).

2 Related Work

There are a number of models of computation for scientific workflows [14]. The most common one, however, is the simple parallelism model, where workflows are typically expressed as Directed Acyclic Graphs (DAGs). Nodes in such graphs represent tasks, whereas edges represent (data or control) dependencies among the tasks. Tasks can be executed in parallel providing there are no data dependencies among them. The structure of a given DAG workflow determines the steps required for scheduling and executing the workflow. At each step, there is a maximum number of tasks, which due to their dependencies, can be executed in parallel. There is a significant amount of scheduling algorithms proposed [6, 15] for different scenarios, e.g. computational resources with different characteristics such as type of resources like grid, cluster resources (in other words, heterogeneity vs. homogeneity), etc. optimising different QoS attributes or combinations of such attributes such as cost, energy efficiency or workflow makespan (workflow overall execution time).

Pegasus [6] is one of the most popular scientific workflow systems and it has already explored the use of cloud computing for scientific workflows, focusing on DAGs [10, 9, 20]. More recently, Pegasus made use of Mobius and ShadowQ [13] in order to predict the number of VMs required at each scheduling step. The system provisions VMs in advance and de-allocates them afterward. Similarly, one of the early studies proposing a Scientific Cloud Workflow architecture was that of SwinDeW-C [11]. Its main focused is on QoS enforcement in multi-cloud environments. In contrast, our proposal aims at supporting further dynamism by enabling the workflow enactor to switch from one mapping strategy to an alternative one at runtime.

On the other hand, this idea of greater degrees of the flexibility in executions and the feasibility to perform changes in workflow process instances, while they are being executed is not new in the scientific workflow community [5, 16]. In general, there is a common agreement that dynamism is an intrinsic requirement for scientific workflows, but the understanding about the real needs and functionality to be provided is still confusing. An analysis of requirements for such dynamism is discussed in [5] and it involves the different aspects of scientific workflows, namely changes in the workflow abstract specification [17, 8], changes in the Abstract-to-Concrete transformation (which is part of the mapping process, selecting one or another resource), changes in the datasets (that is, during runtime a given workflow requires an update of some datasets) and changes in the computational resources (as third-party distributed resources, they are always subject to unexpected change, cloud technologies are not an exception).

3 Cloud Workflow Architecture

The mapping of tasks onto distributed computational resources is one the most important challenges for the enactment of scientific workflows. In the past, a plethora of mapping strategies, which can be alternatively used, were developed for workflow DAGs [6], acting on different transformations on the workflow specification. Task clustering is a workflow transformation technique that aggregates fine-grained tasks to increase the granularity, so that the enactment overheads (i.e. typically due to scheduling) are minimized and the overall workflow performance (makespan) is improved.

3.1 Architectural Requirements for Cloud Workflows

As it has already been recognized, the computational resources involved in the enactment process are subject to unexpected change [5] and can exhibit unforeseen behaviors in terms of

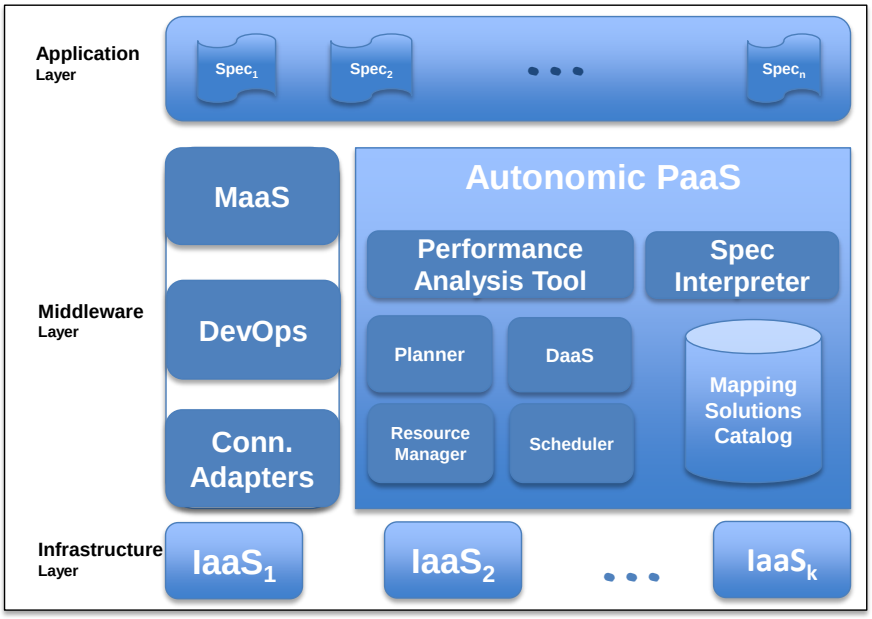


Figure 1: Generic Cloud Scientific Workflow Architecture

performance and faults. Hence, more flexible enactment approaches are desired. For instance, when a chosen mapping strategy reveals unexpected performance degradation, one would desire the workflow enactor to detect such circumstance, and autonomously correct it; for instance, by migrating the mapping to an alternative one where the workflow QoS can be enforced. Cloud technologies enable such flexibility, as the infrastructure can be provisioned and configured on demand.

Nevertheless, in order to enable such autonomic behavior, two main architectural requirements need to be addressed in the traditional Scientific Workflow Architecture: (R1) Once a mapping strategy in chosen, the computational resources, namely CPU, storage, and network, need to be (de-) provisioned and configured on demand, helping the infrastructure to enforce the SLA. (R2) However, in case the adaptation mechanisms at resource-level cannot achieve the desired QoS targets, the system needs to find alternative ways of mapping the tasks onto resources. Therefore, a workflow enactment can be temporarily suspended, migrated to a new mapping, and then the enactment resumed. Then, resources also need to be (de-) provisioned and configured on demand.

3.2 System Architecture

Our generic architectural blueprint for Scientific Cloud Workflow Architecture is depicted in Fig. 1. It follows a classical 3-tier architecture, consisting of workflow level, Platform as a Service (PaaS) level, and resource level. The architecture is designed to meet Requirements R1 & R2 by following the principles described in [18]. Based on autonomic computing principles, the ultimate goal of the approach is to reduce human intervention when mapping scientific workflows onto a Cloud infrastructure.

The mapping process is subject to the inherent complexities of distributed infrastructures. Therefore, the system architecture incorporates performance analysis tools for modeling such mapping solutions. Given the fact that a performance engineer can find a number of alternative workflow mappings onto distributed resources, the usage of performance models enables an autonomic model-driven workflow enactment in accordance with high-level policies. Such policies allow the platform to (i) interpret the workflow specifications; (ii) map the specifications onto the target computing infrastructure, so that the workflows are executed and their QoS, as specified in their SLA, are enforced; and, most importantly, (iii) adapt automatically such previously established mappings when unexpected behaviors occur. Such adaptations may involve modifications in the arrangements of the computational infrastructure, i.e. by re-designing a different communication network topology that dictates how computational resources interact, or even the live-migration to a different computational infrastructure.

The underlying actions to achieve the aforementioned self-adaption is to (de-) provision computational machines, storage and networking links. This is accomplished by the resource manager component. It is based on MOSt-CB, specified in Section 3.5. Besides, the architecture, as depicted in Fig. 1, integrates emerging paradigms such as DevOps for automate deployments, Monitoring as a Service (MaaS) for accurate and large-scale monitoring, or well-known formalisms such as Petri Nets for building performance models (Performance Analysis Tool component), which enable the workflow enactor to evaluate and choose a mapping strategy.

3.3 Performance Models of Workflows

Our performance workflow models consists of 3 different abstraction levels: (i) Abstract Workflow: The workflow is specified as a platform-independent model, i.e. as a Directed Acyclic Graph, where nodes represent workflow tasks, and edges represent data and control dependencies. This is a popular model of representation of workflows in the community. (ii) Operational model: The Operational Model is obtained from the Abstract Workflow Model, by refining it and adding operational resources and data transmission tasks to it when required (i.e. to move data from a resource location to another). An operational resource can be seen as a generic computational resource, which is able to perform the required computation. Its behavior is ideal, in the sense that it is not subject to failures or unexpected performance variations. From this model, worst- & best-case performance and cost boundaries can be obtained (though, perhaps, other QoS attributes could also be incorporated). (iii) Infrastructure-specific (Cloud) model: The Infrastructure-specific model is obtained by refining the Operational model with constraints and characteristics from the target infrastructure. It typically incorporates two aspects: (i) The operational resource become real resources by incorporating in them real-time performance information, so that an actual and more accurate QoS parameter estimation is obtained. It lies somewhere between the worst- & best-case boundaries of the Operational model. (ii) The real resources are typically configured forming a topology in a communication network. Many topologies can be possible, having each different impacts in the QoS. An operation model analyses one mapping workflow-resources. It can assess resource provisioning and scheduling strategies, therefore, its objective is to enable comparison among multiple mappings. In general terms, the construction of the models can involve complex mapping of tasks, requiring a number of transformation of the original structure and different resource configurations –i.e. machines and their corresponding network links, conforming network topologies. As a result, this process should be human-directed: In many cases, only a performance engineer can deal with such complexity of better establishing a number of resource configurations and elaborating a catalogue of mappings that can be subsequently used by the system at runtime.

3.4 The Enactment of a Workflow

Once a workflow specification is generated, it is sent to the autonomic Platform as a Service (PaaS). Then, the autonomic PaaS in conjunction with the workflow designer builds operational (logic) QoS workflow models and starts to perform analysis to meet users SLAs. At that stage, operational (logic) computational resources are represented and performance figures (i.e. execution time) are added to the model. As a result of this human-assisted analysis, a catalog of workflow operational (logic) resource assignments is obtained. The objective is to map the workflow onto cloud resources managed by a cloud IaaS infrastructure. So, the next step is to refine the model with the cloud resources constraints: measured in terms of performance time (i.e. execution time), limitations in the number of resources (if any), limitations in the network (if any), etc. Such refined models need to be updated constantly with real time information from the infrastructure. In case anything goes wrong (e.g. unexpected performance variations) and the cloud IaaS cannot do anything to deal with it, an event is generated and triggered to the autonomic PaaS. Then, the autonomic PaaS, in accordance with the catalog of logic assignments, it can decide whether to choose among alternative workflow mappings, and proceed with the actions to replace the executions in the IaaS infrastructure.

3.5 Resource Management: MOSt-CB

The resource manager component of our architecture is based on the Multi-Site Orchestration System (MOSt), also based on Cloud Brokerage (CB) [2]. It was designed to calculate an optimal provisioning plan in a multi-site Cloud environment distributed over multiple geographical location and managed by a Site Manager (e.g., Openstack). MOSt relies on a placement algorithm IGM (Iterative Graph Mapping) described in [19], which was updated by adding, at each call, a parameter containing a specified list of target sites (Target Cloud Infrastructure).

The challenges addressed by MOSt-CB are (i) dynamic provisioning, (ii) configuration, (iii) re- configuration and (iv) optimization of: computing resources nodes, i.e. Virtual Machines; and networking resources links (i.e. the traffic exchange between Cloud Provider domains). MOSt-CB operates in four different phases: (i) *Partitioning and coordinating phase*: MOSt-CB extracts the virtual network graph and calculates its optimal partitioning given the participating Cloud Providers and their respective SLAs. More precisely, it divides the requested network graph into as many sub-graphs as necessary based on the terms of the SLA, i.e. cost, compute, storage, network, type of service, and requested Geographical zone, and the available characteristics of provided resources by underlying CPs. (ii) *Provisioning phase*: MOSt-CB calculates the optimal provisioning plan and engages the resources (e.g., Virtual Machines) in the multiple underlying CP domains. (iii) *Post-configuration phase*: MOSt-CB launches the post- configuration of the deployed virtual machines based on the request graph (e.g. OpenStack manifest). (iv) *Networking provisioning phase*: MOSt-CB launches the configuration of the network connections between the various sites to fulfill the service nodes communication requirements (as specified in the service graph). This requires to instantiate specific network gateways in each site to build the necessary VPL (Virtual Private Links) to allow Virtual Machines (VMs) of the same project to communicate. For the SLA assurance, MOSt-CB monitors the provisioned services in the different domains and verify whether the terms of the CCs SLA are fulfilled. In case of SLA violation, it can take re-configuration actions, that involves virtual network topology adaptation and node reconfiguration (i.e. VM migration could be considered, but perhaps, in this context, instead of VM migration, another alternative VM could work perfectly). In [2], the MOSt-CB shows the re-configurability mechanisms it offers in order to find virtual network and VMs alternative configurations that enforce the QoS.

4 Case Study: Image Stitching Workflow

A number of disciplines such as medicine or astronomy make use of the image stitching process in order to combine various overlapping images to generate a larger panoramic image. Although some digital cameras can already generate panoramic images, the *Image Stitching* process can also be accomplished by means of software libraries. The authors in [4] present the standard pipeline workflow (see Fig. 2), which shares some analogies with the more complex and famous Montage workflow [6] for astronomic images, to achieve *Image Stitching*. The workflow consists on four main steps: a. *Feature Extraction*: that consists on a set of tasks to perform on all images to identify distinctive points (or keypoints) in each image. Therefore, a feature descriptor [3], [12] is computed for each keypoint. b. *Image/Feature Matching*: that consists on a set of tasks to perform on all images of the set to match features between pairs of images to estimate relative camera transformations. c. *Global Refinement*: that consists on a set of tasks to achieve camera transformation parameters across all images (e.g. differences in lighting, colors, etc.). d. *Seam Blending*: that consists on a set of tasks to estimate seams pairs of images and then blending task is performed.

As presented in [1], "the set of images are represented as a data graph $G = (V, E)$, where each vertex corresponds to an image and two vertices are connected by an edge, if the two images overlap in content, i.e. capture a part of the scene from two viewpoints". The stitching workflow consists in this case on the four main steps, however the individual tasks execution could be executed in parallel in different possible configuration (onto different groups of images, different parts of the same image, etc.).

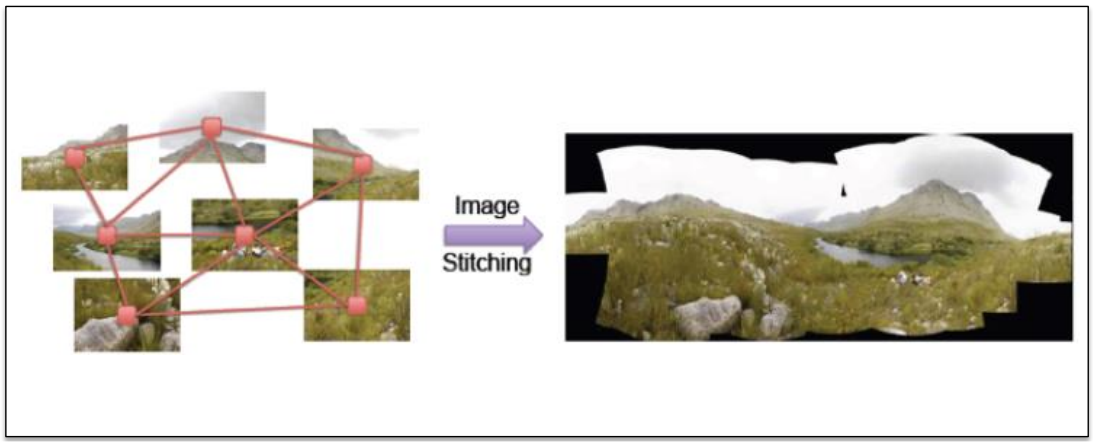
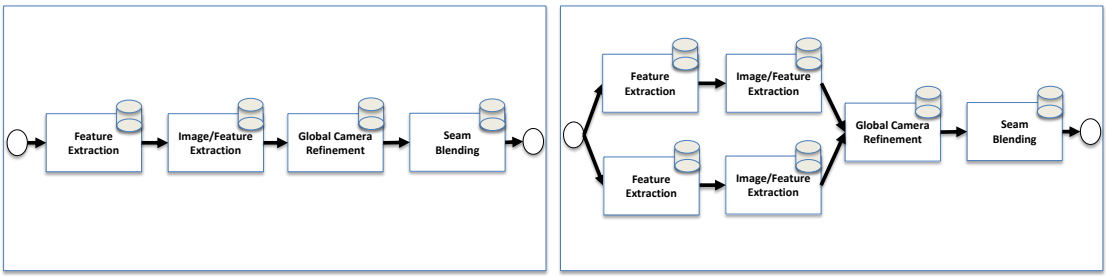


Figure 2: Datagraph and Panorama (source : arXiv:1506.04130v2 [cs.CV] 15 Jun 2016)

As explained in [1], in case the number of images is large, it is possible to process the images independently as far as a minimum information about the camera parameters are communicated to the neighboring process. Therefore, the performance engineer could design workflow transformations, as depicted in Fig. 3b, enabling different forms of parallel tasks as far as the data (images) are available to them.

The global architecture that encompasses the workflow manager and MOSt is presented as follows: The workflow manager receives the images stitching request with the URL where the images are. The workflow manager identifies the initial workflow and specifies the manifest (set of tasks to execute, source of data, SLA, etc.) to transmit to MOSt. This later identifies



(a) Initial Image Stitching Workflow (b) Transformed Workflow with parallel processing

Figure 3: Autonomic Workflow Transformation

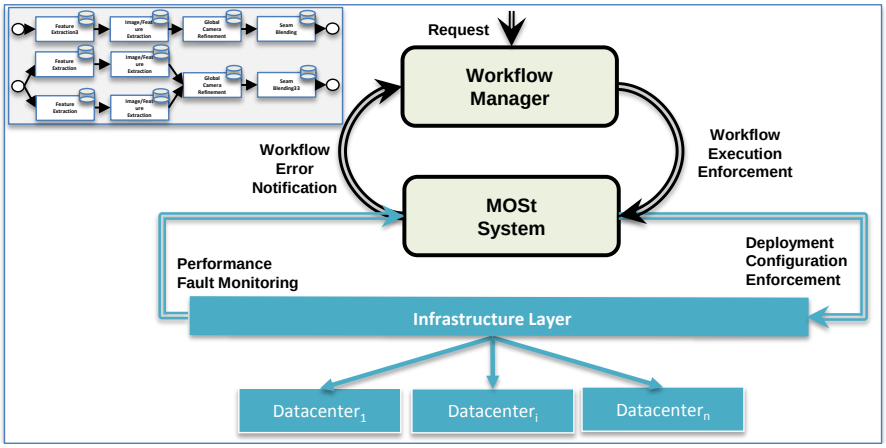


Figure 4: Global Proposed Architecture: Two loops self-optimization

the VM to engage and the data-centers where these VM should be executed. Once all the VM are engaged, MOSt triggers the execution of the initial workflow and monitors the execution. When the workflow is fully executed, MOSt collects the results and sends the response to the workflow manager which in turns sends it to the client (URL from where to download the panorama image). In case of issues in the execution of one the VMs, MOSt receives alarms. If the problem is a lack of resources, MOSt could enforce a vertical elasticity allocating more resources to the VM. In case it is a failure, MOSt notifies the workflow manager to ask for the appropriate action to perform. The workflow manager could either ask to MOSt to re-execute the VM without any consequence on the global execution of the workflow or find another transformed workflow to continue the execution and sends therefore the required information to MOSt.

5 Conclusions

In this paper, we proposed a novel Cloud Scientific Workflow based on autonomic principles. While, traditional workflow systems often adapt the workflow structure to the underlying physical resources, our architectural approach also supports the adaptation of computational re-

sources to the workflow structure. Such combination enables a greater degree of flexibility and dynamism, which can reduce human-intervention at runtime and can help reduce the growing complexity of distributed infrastructures. The autonomic behavior of the system is provided by the combination of high-level and low-level policies. High-level policies enable the workflow enactor to choose among a number of workflow structure transformations that better suit the underlying resources dynamically depending on the context, whereas low-level policies enable the autonomic resource manager to adjust the required computational power to the workload derived from a scientific workflow specification, exploiting the cloud elasticity property, and to cope with performance fluctuations or unexpected events in cloud infrastructures. The novelty of the approach is the combination of both policies that can lead to higher degrees of dynamism. The key enabling architectural components for such dynamism are the petri-net based performance analysis tools for implementing high-level policies and MOSt-CB system for the adaptation to multi-cloud environments. In the future, we aim at (i) studying how we can exploit autonomic principles to explore different mapping of tasks in an automated way, (ii) to further develop the mechanisms required for accomplishing low-level policies (i.e. involving MOSt), and (iii) to perform experimentation on the existing platform and highlight the performance of our approach against other approaches.

References

[1] Harsh Agrawal, Clint Solomon Mathialagan, Yash Goyal, Neelima Chavali, Prakriti Banik, Akrit Mohapatra, Ahmed Osman, and Dhruv Batra. Cloudcv: Large scale distributed computer vision as a cloud service. *arXiv preprint arXiv:1506.04130*, 2015.

[2] Mustapha Ait-Idir, El Hadi Cherkaoui, Elie Rachkidi, Nada Chendeb, and Nazim Agoulmine. Most-cb: Sla enforcement and smart vne (virtual network embedding) in a multi cloud providers environment. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 86–92. IEEE, 2014.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). In *Compt Vis. Image Underst*, pages 346–359, 2008.

[4] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. In *IJCV*, pages 59–73, 2007.

[5] Manuel Caeiro-Rodrıguez, Thierry Priol, and Zsolt Németh. Dynamicity in scientific workflows. Technical report, Technical report, CoreGRID, 2008.

[6] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good, et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[7] Daniel Garijo and Yolanda Gil. A new approach for publishing workflows: abstractions, standards, and linked data. In *Proceedings of the 6th workshop on Workflows in support of large-scale science*, pages 47–56. ACM, 2011.

[8] Yolanda Gil, Varun Ratnakar, Jihie Kim, Pedro A González-Calero, Paul Groth, Joshua Moody, and Ewa Deelman. Wings: Intelligent workflow-based design of computational experiments. *IEEE Intelligent Systems*, 26(1):62–72, 2011.

[9] Christina Hoffa, Gaurang Mehta, Tim Freeman, Ewa Deelman, Kate Keahey, Bruce Berriman, and John Good. On the use of cloud computing for scientific workflows. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 640–645. IEEE, 2008.

[10] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P Berman, and Phil Maechling. Scientific workflow applications on amazon ec2. In *2009 5th IEEE International Conference on E-Science Workshops*, pages 59–66. IEEE, 2009.

[11] Xiao Liu, Dong Yuan, Gaofeng Zhang, Wenhao Li, Dahai Cao, Qiang He, Jinjun Chen, and Yun Yang. *The design of cloud workflow systems*. Springer Science & Business Media, 2011.

[12] D.G Lowe. Distinctive image features from scale-invariant keypoints. In *Int. J. Comput. Vision*, pages 91–110, 2004.

[13] Anirban Mandal, Paul Ruth, Ilya Baldin, Yufeng Xin, Claris Castillo, Gideon Juve, Mats Rynge, Ewa Deelman, and Jeff Chase. Adapting scientific workflows on networked clouds using proactive introspection. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 162–173. IEEE, 2015.

[14] Cesare Pautasso and Gustavo Alonso. Parallel computing patterns for grid workflows. In *2006 Workshop on Workflows in Support of Large-Scale Science*, pages 1–10. IEEE, 2006.

[15] Arun Ramakrishnan, Gurmeet Singh, Henan Zhao, Ewa Deelman, Rizos Sakellariou, Karan Vahi, Kent Blackburn, David Meyers, and Michael Samidi. Scheduling data-intensiveworkflows onto storage-constrained distributed resources. In *Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid'07)*, pages 401–409. IEEE, 2007.

[16] Ian J Taylor, Ewa Deelman, Dennis B Gannon, and Matthew Shields. *Workflows for e-Science: scientific workflows for grids*. Springer Publishing Company, Incorporated, 2007.

[17] Rafael Tolosana-Calasanz, José A Bañares, Omer F Rana, Pedro Álvarez, Joaquín Ezpeleta, and Andreas Hoheisel. Adaptive exception handling for scientific workflows. *Concurrency and computation: Practice and experience*, 22(5):617–642, 2010.

[18] Rafael Tolosana-Calasanz, José Ángel Bañares, and José-Manuel Colom. On autonomic platform-as-a-service: Characterisation and conceptual model. In *Agent and Multi-Agent Systems: Technologies and Applications*, pages 217–226. Springer, 2015.

[19] Khanh-Toan Tran, Nazim Agoulmine, and Youssef Iraqi. Cost-effective complex service mapping in cloud infrastructures. In *2012 IEEE Network Operations and Management Symposium*, pages 1–8. IEEE, 2012.

[20] Jens-Sönke Vöckler, Gideon Juve, Ewa Deelman, Mats Rynge, and Bruce Berriman. Experiences using cloud computing for a scientific workflow application. In *Proceedings of the 2nd international workshop on Scientific cloud computing*, pages 15–24. ACM, 2011.