Data-driven Workflows in Multi-Cloud Marketplaces

Javier Diaz-Montes[1], Mengsong Zou[1], Rahul Singh [2], Shu Tao [2] and Manish Parashar[1]
[1] *Rutgers Discovery Informatics Institute, Rutgers University, USA*
[2] *IBM T. J. Watson Research Center, USA*
contact author: javier.diazmontes@gmail.com

***Abstract*—Cloud computing is emerging as a viable platform for scientific exploration. The ideas of on-demand access to resources, "unlimited" resources as well as interesting pricing models are making scientist to move their workflows into cloud computing. However, the amount of services and different pricing models offered by the providers often overwhelm users when deciding which option is best for them. Moreover, interoperability across providers remains an open topic that forces users to develop specific solutions for each provider. In this paper, we present a service framework that enables the autonomic execution of dynamic workflows in multi-cloud environments. It also allows users to customize scheduling policies to use those resources that best fit their needs. To demonstrate the benefits of this framework, we study the execution of a real scientific workflow, with data dependencies across stages, in a multi-cloud federation using different policies and objective functions.**

***Keywords*-Data-driven workflow; Autonomics; Cloud computing; Software-defined infrastructure**

I. Introduction

The unprecedented amount of data being generated every day is creating new challenges and opportunities that can lead to extraordinary new knowledge and discoveries in many application domains ranging from science and engineering to business. One of the main challenges in this era of Big Data is how to efficiently manage and analyze such scale of data. This is challenging not only due to the size of the data, but also due to its heterogeneous nature and geographic location. In this sense, clouds are becoming an increasingly popular infrastructure for enabling large-scale data intensive scientific and business applications [26]. Clouds offer on-demand access to computing utilities, an abstraction of unlimited computing resources, customizable environments, and a pay-as-you-go business model. They have a potential for scale-up, scale-down and scale-out as needed, and for IT outsourcing and automation. As a result, it is possible to have hybrid cloud infrastructures that integrate private clouds, local data centers, and public clouds. This creates interesting marketplaces where users can take advantage of different types of resources, quality of service (QoS), geographical locations, and pricing models.

However, realizing multi-cloud markets presents a whole new set of challenges. Cloud technology is in its early stages and lacks of proper standardization. Therefore, inter-operability between cloud providers becomes difficult as each provider uses its own interfaces and protocols. Although there are some initiatives such as Siena [31] or OCCI [29], they are still works in progress that will take some time before cloud providers fully adopt them. Besides, cloud infrastructures lack of mechanisms to dynamically provision resources to meet application and user requirements. This prevents users from optimizing the use of the resources and money. As a consequence, users typically over-provision resources to make sure their workflows run without problems. Finally, there is a wide range of pricing models that vary from vendor to vendor, which makes very difficult for customers to find the best resources for their needs at the lowest cost.

In this paper, we present a framework to enable scientists with mechanisms that ease the autonomic execution of complex workflows in software-defined multi-cloud environments. This framework is built on top of our federation model [10], which enables the dynamic creation of federated "Cloud-of-Clouds". The resulting solution is a platform that takes a workflow description from the user and autonomously orchestrate the execution of such a workflow by elastically composing appropriate cloud services and capabilities to ensure that the user's objectives are met. We evaluate our framework by executing a scientific workflow across three federated clouds under different scheduling policies. The obtained results validate our framework and provide important insights about the application behavior and resource utilization that can be used toward the creation of complex models and multi-objective scheduling policies that can efficiently take advantage of multi-cloud marketplaces.

The rest of the paper is organized as follows. Section II presents the background and related work. Section III describes the design of our framework. Section IV presents the environment for our experiments followed by the problem statement and scheduling policies in Section V. Sections VI and VII present and discuss the results. Finally, in Section VIII we present our conclusions and ongoing work.

II. Background and Related Work

Many scientists rely on workflows to automate the different processes involved in their applications. Scientific workflows combine data and processes into a configurable, structured set of steps that translates a scientific problem into a semi-automated computational solution [24]. Specifically, data-driven workflows are designed mostly to support data-driven applications, where the dependencies represent the flow of data between workflow activities. One of the main challenges here is how to manage and process exponentially growing data volumes, often arriving in time-sensitive streams from sensors and instruments, or as the outputs from simulations. An efficient and comprehensive analysis of such data requires distributed and parallel processing [14].

Federated computing has been explored in various contexts and has been demonstrated as an attractive and viable model for effectively harnessing the power offered by distributed resources [2, 4, 13, 22, 23]. In this area, cloud computing is becoming increasingly popular due to its flexible computational model that offers resources on-demand, and creates the illusion of a unique unlimited pool of resources. Cloud computing has been shown to be effective for certain classes of applications [8, 12, 17, 20]. There are also efforts exploring how to compose providers as a federated cloud [25] and how to combine clouds with integrated computing infrastructures [7, 22]

Since the cloud technology is relatively new, the cost and relative performance of computing in the cloud can be a concern [8, 26]. For this reason, understanding how to efficiently execute workflows on "cloud-of-clouds" infrastructures is a very active research topic. We have observed two major approaches to problem. On the one hand, researchers focus on creating cost models to make sure that the workload is executed according to the defined QoS parameters [1, 11, 15, 16]. On the other hand, researchers put emphasis on the way resources are utilized [6, 18, 19], although the concerns about the costs are always present. A particular case of the latter could be prioritizing the use of local resources and controlling which tasks are outsourced [9, 21].

Although we have the technology to provide access to data and computational resources, there are still obstacles to overcome before scientists can fully benefit from this abundance of data and the powerful computing technologies. Scientists still face having to learn how to use multiple technologies to get their science done, leaving non-expert users in a position of using less efficient techniques and achieving lower execution performance, which in turn consumes more time and resources.

III. Service Framework

In this work we propose a framework to bridge the gap between scientists and technology. This framework enables an efficient execution of workflows across geographically distributed resources. It empowers scientists to easily define complex workflows that can be executed in dynamically federated infrastructures. In addition, it offers a set of mechanisms to define and select scheduling policies, which scientists can use to configure how the workflows are executed.

To execute dynamic workflows across federated resources, we have built a workflow manager and a customizable autonomic scheduler on top of our federation model [10, 21]. Figure 1 shows a general overview of this model, where each federation site sees the rest of sites as a pool of elastic resources.

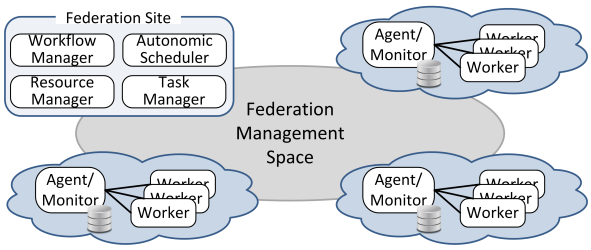


Figure 1: Federation Architecture

In the model, users at every site have access to a set of heterogeneous and dynamic resources, such as public/private clouds, supercomputers, and grids. These resources are uniformly exposed using cloud-like abstractions and mechanisms that facilitate the execution of applications across the resources. The federation is dynamically created in a collaborative way, where sites "talk" to each other to identify themselves, negotiate the terms of adhesion, discover available resources, and advertise their own resources and capabilities. Sites can join and leave at any point. Notably, this requires a minimal configuration at each site that amounts to specifying the available resources, a queuing system or a type of cloud, and credentials. As a part of the adhesion negotiation, sites may have to verify their identities using security mechanisms such as X.509 certificates, or public/private key authentication.

A. Realizing Scientific Workflows

We envision a scenario in which popular applications contributed by the scientific community are deployed within the federation such that they can be accessed on-demand and at the same time benefit from the distributed resources. With applications pre-deployed in the federation, scientists would be only required to define a high-level workflow description. Figure 2 depicts the end-to-end process involved in the execution of a workflow.

The main actor is the actual user, who is required to describe his/her workflow. The description of a workflow

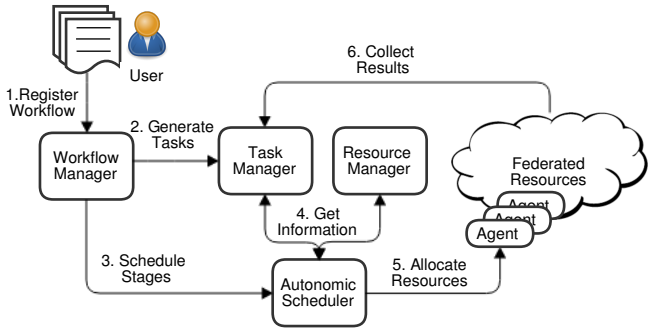


Figure 2: Workflow Execution Process.

is an XML document that collects the properties of the comprising stages. Each stage defines the application to use; location of the the input and output data; dependencies with other stages; and the scheduling policy that will drive the execution. The user initiates the execution of the workflow by registering its description in the *Workflow Manager*. This operation returns an identifier to the user, which can be next used to perform operations such as checking the status of the workflow, or retrieving results. Using the workflow description, the workflow manager can autonomously proceed with the execution. First, it identifies dependencies across stages and selects those stages that are ready to execute, i.e., all their dependencies are satisfied. Then, the description of these stages is sent to the *Task Manager* that generates and keeps track of all tasks composing the stages executed at a given site. The task generation process depends on the type of application and the logic associated with it – we discuss it in detail in Section III-B. Once all tasks are generated, the workflow manager is notified and it passes control to the autonomic scheduler which allocates resources for these tasks.

Scheduling tasks of a stage involves the autonomic scheduler performing four steps: (i) retrieving the information of the available resources from the *Resource Manager* (i.e., resource availability, relative performance, cost); (ii) retrieving information related to the tasks to execute (i.e., data location and task complexity); (iii) identifying the scheduling policy selected by the user; and (iv) using that scheduling policy to decide which resources to provision, from which site, for how long, and where to execute each task.

Once the scheduling decision is made, the autonomic scheduler dynamically allocates the required resources from each site. Resources at each site are offered to the rest of the federation through a common interface. This interface is implemented by an *Agent* that acts as a gateway. Therefore, the autonomic scheduler has to interact with the appropriate agents to provision/allocate resources. Agents know the status of their resources and how to interact with them. For example, an agent may provision virtual machines (VMs) when working with a cloud or reserve machines using a queuing system when interacting with an HPC cluster. In case of failure, the autonomic scheduler is notified so that it can take appropriate actions. In case of success, a worker is deployed in each machine/VM to immediately start retrieving and executing tasks. Any required input data is automatically transferred upon request.

Task results are collected by the task manager. This component keeps the autonomic scheduler informed about the progress to allow changes in the schedule at runtime and deallocate/terminate resources when they are not needed. When all the tasks of a stage are completed, the workflow manager is notified to identify new stages ready to be executed. This process continues until all stages are completed.

B. Integrating new Applications and Policies

New applications can be integrated into the federation by developing two simple components, namely task generator and worker.

Task generator. This module can be created using a simple API, which is loaded into the *Task Manager*. Its mission is to define the properties of all the tasks that need to be generated by an application in a programmable way. The idea is to provide users with the ability to define dynamic workflows, where the tasks are created at runtime depending on previously obtained results. Results of all stages of a workflow are accessible through the API. This provides tremendous flexibility as the workflow can evolve in different ways depending on the observed data. This proves to be very useful to, for example, investigate large search-spaces in a coordinated manner. Here, classic examples are Monte Carlo methods or stochastic sampling strategies (e.g., sparse grid collocation).

Worker. The worker's sole responsibility is to execute tasks. In many cases a user might be interested in executing third-party, perhaps closed-source, software. In such cases, the resulting worker becomes a mere container that acts as a facade for the target software. This significantly simplifies migration from traditional environments to our federation.

New policies can also be easily integrated to increase the control users have over the resources. Scheduling policies use the workflow and resource information to autonomously provision the appropriate resources and react upon changes to guarantee that objectives and constraints defined in the polices are met. This is achieved by including algorithms in the *Autonomic Scheduler* module, which become available to the community in the form of policies. Section V describes in detail the basic considerations when implementing new algorithms.

IV. Methodology

In this work, we used the Montage workflow in a multi-cloud marketplace, as a use case. We created a synthetic workflow of 500 nodes using the description provided by [5], which is based on actual traces of the real application, and divided the workflow in five stages. The size of the data was increased one order of magnitude to better observe the effect on data movement in the workflow execution.

The federated marketplace used here is composed by three different clouds geographically distributed across the country. These resources are part of the FutureGrid project [27]. Two clouds are based on the OpenStack IaaS [30] namely Sierra, located at San Diego Supercomputer Center (SDSC), and Alamo, located at Texas Advance Computer Center (TACC). The third one, named Hotel, is based on the Nimbus [28] IaaS, and it is located at the University of Chicago. Table I shows the characteristics of the resources available at each site, where the *Speedup* is obtained using the UnixBench benchmark. This speedup is relative to the machine with the closest characteristics to the one originally used in [5]. Table II shows the bandwidth of the interconnection network between sites and internally within each site. To reduce the variables in the system, we considered that the cost of each resource, extracted from Amazon EC2, were uniform across sites, see Table III. During the experiments we assumed that the initial data was located at the Sierra site.

Table I: Resources available at each site and their characteristics.

VM type†	#Cores	Memory	Max. VMs‡	Speedup
Alamo_Large	4	8 GB	2	3.55
Alamo_Medium	2	4 GB	4	2.77
Alamo_Small	1	2 GB	2	1.68
Sierra_Medium	2	4 GB	2	1
Sierra_Small	1	2 GB	3	0.71
Hotel_Small	1	2 GB	6	0.76

Note: † – Name of the site followed by the type of VM.
‡ – Maximum number of available VMs per type

Table II: Network speed in MB/s.

Network (Down/Up)	Alamo	Sierra	Hotel
Alamo	-	10/0.9	15/15
Sierra	11/11	-	11/11
Hotel	18/18	12/1	-
Internal Network (Down/Up)	11/2.3	30/30	45/45

V. Scheduling policies

In this section we describe the basic considerations when implementing new scheduling policies. We also define two scheduling policies aimed to execute workflows in a multi-cloud marketplace.

Table III: Cost of the resources

Type	Cost
Large	0.24 $/hour
Medium	0.12 $/hour
Small	0.06 $/hour
Data In	0.01 $/GB
Data Out	0.12 $/GB

A. Defining the Problem

We consider that a workflow has three main properties. First, a workflow is composed by a set of stages that have some dependencies among them. Thus, one stage cannot be scheduled until its dependencies are satisfied. Second, the number of tasks of a stage may not be known until its dependencies are resolved. We can have dynamic workflows where the results of a stage are needed to create the tasks of the next stage and therefore we cannot schedule the whole workflow at once. Third, we can select different scheduling strategies for each stage as they could have different type of workloads.

The problem to tackle is how to schedule a set of tasks $\mathrm{T} = \{t_1, ..., t_n\}$ composing a single stage in a federation composed by a set of cloud sites $\mathrm{S} = \{s_1, ..., s_m\}$, where each site $s_i \in \mathrm{S}$ has a set of heterogeneous VMs available $V_i = \{v_{i1}, ..., v_{iq}\}$, $i = 1, ..., m$. Considering a given $t_j \in \mathrm{T}$, $v_{iz} \in V_i$, and $s_i \in \mathrm{S}$, we define the following variables:

- e_{ijz} - estimated units of time needed to execute task t_j in VM v_{iz} of cloud s_i
- d_{ijz} - units of time needed to transfer data of task t_j to VM v_{iz} of cloud s_i
- r_{iz} is the estimated units of time that VM v_{iz} of cloud s_i requires before it can execute the next task
- c_{iz} - cost of VM v_{iz} of cloud s_i per unit of time
- p_{ijz} - cost of transferring data of task t_j to VM v_{iz} of cloud s_i

B. Minimum time of completion policy

We define the well-known minimum time of completion (MTC) [3], which allocates each task to the resource that offers the minimum completion time. The heuristic is defined in Algorithm 1.

Algorithm 1: MTC. Find the resource that provides the minimum estimated completion time (MTC).

```
while T not empty
    select a task t_j
    find VM v_iz  such that  e_ijz + d_ijz + r_iz is minimum
    allocate task t_j in VM v_iz
    delete task t_j from T
    update r_iz
endwhile
```

C. Deadline-based policy

This policy finds the minimal set of resources needed to complete all tasks within a given deadline while satisfying the objective function $F(ijz)$, see Algorithm 2. This objective function allows us to focus on specific properties (performance, cost, or data movement) when scheduling our workflow.

Algorithm 2: Deadline. Find minimal set of resources that complete all tasks within the deadline D while satisfying the function F.

```
while T not empty
    select a task t_j
    find VM v_iz  such that  F(ijz)  and
         e_ijz + d_iz + r_iz < D
    if found
      allocate task t_j in VM v_iz
    else
      allocate task t_j in VM v_iz  such that  F(ijz)
    endif
    delete task t_j from T
    update r_iz
endwhile
```

We have considered four objective functions:

Performance optimization (Proc). This policy selects the resource with the highest performance available at each scheduling event. This policy defines $F(ijz) = minimum\ e_{ijz}$

Data locality optimization (Data). In this policy we exploit data locality by selecting the resource that minimizes the time spent moving data. This policy defines $F(ijz) = minimum\ d_{ijz}$

Performance and data optimization (Proc-Data). In this policy we try to find an equilibrium between the performance of the VM and the time required to transfer the data, $F(ijz) = minimum\ e_{ijz} + d_{ijz}$

Cost optimization (Cost). This policy aims to minimize the cost of the provisioned resources by selecting the cheapest resource at every scheduling event. This policy defines $F(ijz) = minimum\ c_{ijz} + p_{ijz}$

VI. Results

In this section, we present the results obtained when considering the scenario presented in Section IV under the two scheduling policies presented in Section V.

A. Completion time

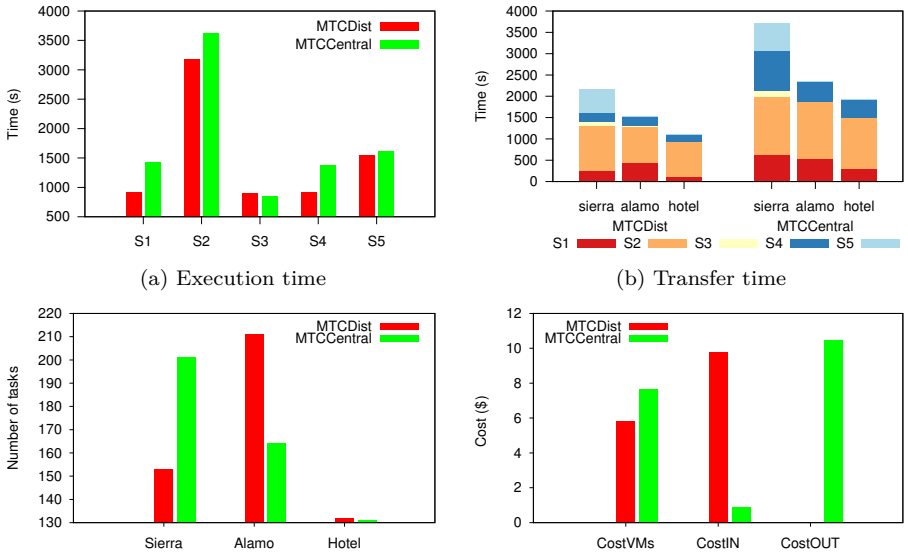
In the first set of experiments, we focus on completing the workflow in the minimum time possible by using the MTC scheduling (Algorithm 1). We consider two scenarios: (i) distributed storage space where each site of the federation has its own local storage (MTCDist), (ii) centralize storage where there is a unique storage system in the federation (MTCCentral). In these experiments we evaluate the impact of these two scenarios in the execution time, data movement, and the overall cost of the experiment. Experimental results are summarized in Figure 3.

Figure 3a shows that for those stages that involve the execution of multiple tasks (S1, S2, S4), having a storage located at each site to store intermediate data can reduce the execution time by up to 35 %. As we can see in Figure 3b, this is mainly motivated by the shorter time spent transferring data. We also observed that this scheduling strategy tends to concentrate the workload in the site with the most powerful machines (Alamo) when using the distributed storage scenario, see Figure 3c. On the other hand, in the case where a stage has a single task that requires many input files, such as in S3 and S5, we could expect that having all data in a central location would have an advantage as data transfer across sites is not needed. However, in both cases these stages were executed in the same site (Sierra) and, as we can see in Figure 3b, the time spent transferring data is very similar due to the speed of the network across sites.

A very important factor to consider in cloud computing is the cost of the resources. Figure 3d shows the cost of executing our workflow under the distributed and centralized storage scenarios. We can observe that the cost of the used VMs is higher for the centralized case due to the longer execution time. However, what is more interesting here is the cost of the data movement. We can observe that in the MTCDist case, we only spend money in transferring input data to the site where it is needed as we leave the results in the local storage of the site. In the MTCCentral, we have the opposite situation, where most of the cost goes to transferring results to the central location and very little money is spent on transferring input files. This is due to the fact that the scheduling algorithm considered data movement when allocating tasks, which concentrates most of the workload in the site with the central storage. Overall, the cost of moving data in our experiments is 12 % lower when leaving the result data in the site where it was generated (MTCDist).

B. Deadline

We perform a set of experiments using the deadline policy (Algorithm 2) with the different objective functions described in Section V-C. In these experiments, we consider a distributed storage scenario where each site has local storage. We set the deadline of each stage to a value that is 50 % higher than the one obtained in the first set of experiments by MTCDist. Since this policy provisions the minimum number of resources needed to compute that workload within the deadline, this deadline allows us to observe how different objectives make the scheduler to provision a different number and type of resources at different sites. Figure 4 illustrates the results of the experiments.



(a) Execution time (b) Transfer time

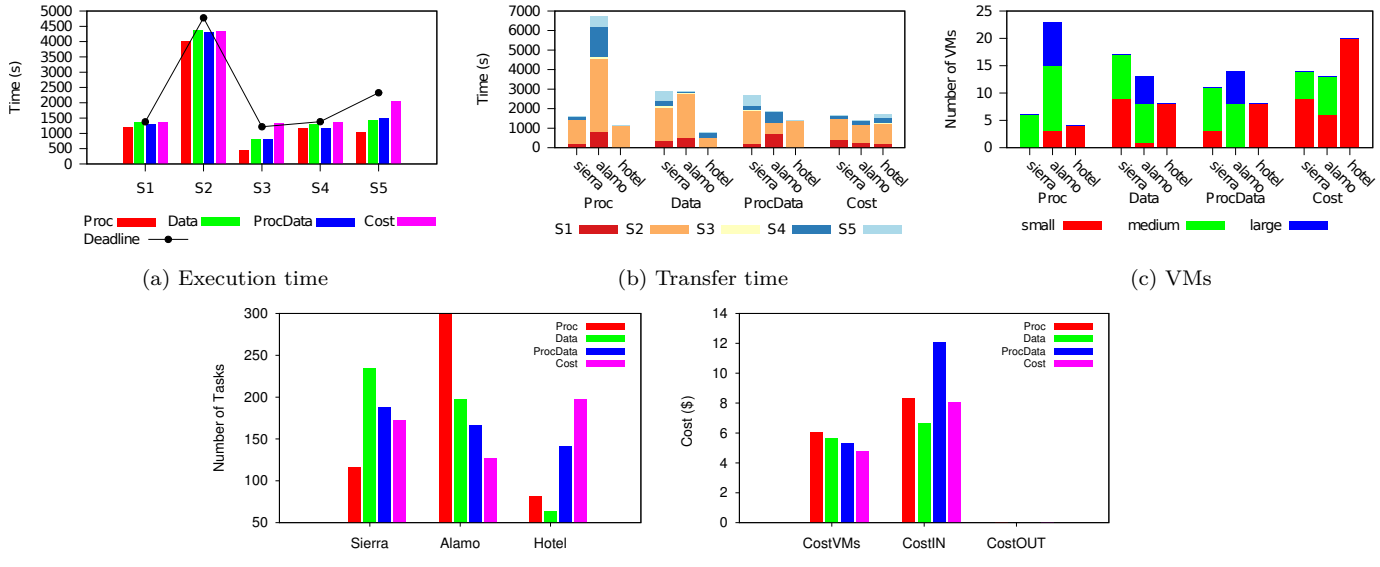
(c) Workload (d) Cost

Figure 3: Summary of experimental results using the MTC policy with distributed storage (MTCDist) and central storage (MTCCentral). 3a collects the execution time of each stage of the workflow; 3b collects the time spent transferring data to each site and from the site to the VMs; 3c shows the workload of each site; and 3d collects the cost of the used VMs, the cost of transferring input files, and the cost of transferring output files.

Figure 4a collects the execution time of each stage when using different objective functions. The line with dots shows the deadline set for each stage. We can observe that, as expected, all of them get close to the deadline, although they optimize the use of resources aiming for different objectives. The largest variations are found in those stages that only have a single task (S3 and S4). Since we only need to use a single VM for these stages, the objective function has important impact on the overall execution time. For example, in S3 we have a very short execution time when using the *Proc* objective because we use the VM that has the highest performance, and a very long execution time, even missing the deadline, when using the cheapest VM under the *Cost* objective. In the first case, we are spending too much money to execute that stage because we are far from the deadline, while in the second case we may incur in extra costs due to missing the deadline. Thus, there are certain situations that may require an adaptive function which can change its behavior upon changes in the environment to complete the workload while meeting the defined objectives.

Figure 4c shows how our autonomic scheduler is able to shape and adapt the federation to meet the user's requirements. This can also be clearly observed in Figure 4d, which shows how the workload shifts from one site to another depending on the objective function. The workload will be concentrated in the site with the most powerful resources when looking for performance (*Proc*). If we try to exploit data locality (*Data*), the workload will be allocated across the two sites with more powerful resources as this restricts the data movement to only two sites most of the time. When trying to find an equilibrium between the two previous objectives (*ProcData*), we observe a better balance in the workload across sites, but this involves transferring more data across sites. Finally, when minimizing the cost (*Cost*), the workload moves to the sites with a larger number of cheapest VMs (Sierra and Hotel).

Figure 4b and Table IV highlight the data movement derived from the scheduling decisions made to satisfy the different objective functions. We clearly observe that the *Proc* objective involves the largest time in transferring data among all of them. Interestingly enough, this is not caused by transferring large amounts of data across sites. In fact, Table IV shows that the *Proc* objective involves the least number of transferred files because it concentrates the majority of the workload in a single site (i.e., Alamo). Therefore, this is caused by the slow internal network in the Alamo site, which executes a large portion of the tasks, and consequently transfers large amounts of data between the staging storage and the VMs. Another surprising observation is that the *Data* objective function, which specifically aims to reduce the time spent transferring data, shows an overall higher transferring time than the *ProcData* and *Cost* functions. Although the number of transferred files is lower, the distribution of the tasks across only two sites created a



(a) Execution time (b) Transfer time (c) VMs

(d) Workload (e) Cost

Figure 4: Summary of experimental results using the deadline policy with different objective functions. 4a collects the execution time for each stage; 4b collects the time spent transferring data to its destination; 4c shows the number and type of provisioned VMs at each site for each objective function. 4d shows the workload of each site; and 4e collects the cost of the used VMs, the cost of transferring input files, and the cost of transferring output files.

bottleneck in the network that is not observed in the other two cases.

Table IV: Total number of files transferred across sites and time spent transferring files. Note that in both cases the given value is the result of adding up the values obtained at each site.

Objective	ProcData	Proc	Data	Cost
Total Number of Files	1625	1330	1455	1601
Total Transfer Time (s)	5892	9431	6504	4691

Finally, Figure 4e presents the cost of executing the workflow. The largest variations are found in the cost of moving data. Since the objective *ProcData* causes the largest amount of data movement across sites, see Table IV, this objective spends up to 29 % more money on data movement alone. While we hoped to get a good trade-off between the performance of *Proc* and the low data movement of *Data*, we observed that the direct combination of both factors does not result in a very efficient policy from the cost perspective. A possible improvement could be to consider different weights for each objective depending on the final optimization goal. For example, if we increase the importance that we give to the time spent transferring data across sites, we would be able to reduce the cost of the input files (*CostIN*).

VII. Discussion

The results presented in the previous section show the feasibility and capability of our framework in executing data-driven workflows in a multi-cloud environment. We observed that for different scheduling policies, our framework allocated resources and workload as expected.

When executing workflows in distributed environments, it is important to consider the trade-offs between minimizing data transfer, which may involve using less suitable resources, and finding the best resources, even if it involves higher network overheads. We observed that when looking to minimize the completion time, without any other restrictions, having staging storage at each site is more beneficial. A distributed approach allows us to maximize the use of the most powerful resources while minimizing intermediate data movement, and hence reduces the overall network overheads. On the other hand, a centralized approach may be more beneficial in scenarios where it is important to maximize the use of local resources. This is typical when users own resources and they are interested in using the federated resources to outsource computation under certain circumstances, such as spikes in the demand or when specific capabilities are required.

If we have a deadline constraint, ideally we would like to complete the workload as close as possible to the deadline because this would minimize the overall cost. Here we would face questions such as: shall we choose

fewer resources that are very powerful hence individually expensive, or shall we choose a larger number of less powerful resources that are individually cheaper. Answering this kind of questions in an optimal way is challenging, as many factors need to be considered. In the second set of experiments we studied how different factors affected the workload and the provisioned resources. We observed that by allocating tasks to the resource that involved the minimum cost, we were able to minimize the overall money spent on VMs. However, since data locality was not properly exploited, this approach incurred excessive data transfer costs. The problem here is that the cost of the VM has too much weight in the decision making because if the VM is already deployed we do not have to pay for it again. Therefore, it would be interesting to consider a multi-objective function that tries to find the cheapest resource while exploiting data locality. This strategy would allow to minimize the overall cost.

VIII. Conclusion

In this paper, we presented an integral solution aimed to serve as a bridge between scientists and distributed computing technology. The proposed framework has proved to be able to execute a complex data-driven workflows in a marketplace of federated clouds, adapting the provisioned resources to the needs of the users which are expressed in the form of policies. More importantly, this framework eases the integration of scientific applications into the federation as well as the customization of the scheduling policies. In particular, in this work we used a representative application of constructing astronomical image mosaics, to perform several experiments to validate our framework. These experiments also helped us to better understand the behavior of both the provisioned resources and workload execution when using different policies and objective functions. The obtained results provide important insights of the environment that we are currently using towards the creation of multi-objective policies. These policies will be aimed to improve the resource utilization by analyzing existing knowledge that allows us to anticipate changes in the environment.

Acknowledgement

This work is supported in part by the NSF under OCI 1339036, OCI 1310283, DMS 1228203, IIP 0758566, and by IBM via OCR and Faculty awards. This project used resources from FutureGrid supported in part by NSF OCI-0910812.

References

[1] S. Abrishami, M. Naghibzadeh, and D. Epema. Cost-driven scheduling of grid workflows using partial critical paths. In *Intl. Conf. Grid Computing (GRID)*, pages 1400 – 1414, 2010.

[2] G. Allen and D. Katz. Computational science, infrastructure and interdisciplinary research on university campuses: Experiences and lessons from the center for computation & technology. Technical Report CCT-TR-2010-1, Louisiana State University, 2010.

[3] R. Armstrong, D. Hensgen, and T. Kidd. The relative performance of various mapping algorithms is independent of sizable varie,nces in run-time predictions. In *7th IEEE Heterogeneous Computing Workshop (HCW '98)*, pages 79 – 87, 1998.

[4] F. Berman, G. Fox, and A. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley & Sons, 2003.

[5] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, et al. Characterization of scientific workflows. In *3rd Wksp. on Workflows in Support of Large-Scale Science (WORKS)*, pages 1 – 10, 2008.

[6] T. Bicer, D. Chiu, and G. Agrawal. Time and cost sensitive data-intensive computing on hybrid clouds. In *Intl. Symp. on Cluster, Cloud and Grid Computing (CCGrid)*, pages 636–643, 2012.

[7] A. Celesti, F. Tusa, M. Villari, et al. How to enhance cloud architectures to enable cross-federation. In *Proc. IEEE Int. Conf. on Cloud Computing (Cloud)*, pages 337–345, 2010.

[8] E. Deelman, G. Singh, M. Livny, et al. The cost of doing science on the cloud: the Montage example. In *Proc. SC*, 2008.

[9] R. V. den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Intl. Conf. on Cloud Computing (CLOUD)*, 2010.

[10] J. Diaz-Montes, Y. Xie, I. Rodero, J. Zola, B. Ganapathysubramanian, and M. Parashar. Exploring the use of elastic resource federations for enabling large-scale scientific workflows. In *Proc. of Workshop on Many-Task Computing on Clouds, Grids, and Supercomputers (MTAGS)*, pages 1 – 10, 2013.

[11] H. Fard, R. Prodan, J. Barrionuevo, and T. Fahringer. A multi-objective approach for workflow scheduling in heterogeneous computing environments. In *Intl. Symp. Cluster, Cloud and Grid Computing (CCGrid)*, pages 300 – 309, 2012.

[12] G. Fox and D. Gannon. Cloud programming paradigms for technical computing applications. In *Cloud Futures Wksp.*, 2012.

[13] G. Garzoglio, T. Levshina, M. Rynge, et al. Supporting shared resource usage for a diverse user community: the OSG experience and lessons learned. *J. of Physics: Conf. Series*, 396, 2012.

[14] I. Gorton, P. Greenfield, A. Szalay, and et. al. Data-intensive computing in the 21st century. *Computer*, 41(4):30–32, 2008.

[15] C. Hao, S. Li, Y. Yang, et al. An autonomic performance-aware workflow job management for service-oriented computing. In *Intl. Conf. on Grid and Cooperative Computing*, pages 270 – 275, 2010.

[16] T. Huu and J. Montagnat. Virtual resources allocation for workflow-based applications distribution on a cloud infrastructure. In *Intl. Conf. on Cluster, Cloud and Grid Computing (CCGRID)*, pages 612–617, 2010.

[17] K. Keahey and T. Freeman. Science clouds: Early experiences in cloud computing for scientific applications. In *Proc. Cloud Computing and Its Applications (CCA)*, 2008.

[18] M. Mao, J. Li, and M. Humphrey. Cloud auto-scaling with deadline and budget constraints. In *Intl. Conf. on Grid Computing (GRID)*, pages 41 – 48, 2010.

[19] S. Ostermann, R. Prodan, and T. Fahringer. Dynamic cloud provisioning for scientific grid workflows. In *Intl. Conf. on Grid Computing (GRID)*, pages 97 – 104, 2010.

[20] M. Parashar, M. AbdelBaky, I. Rodero, and A. Devarakonda. Cloud paradigms and practices for computational and data-enabled science and engineering. *CiSE*, 15:10–18, 2013.

[21] I. Petri, T. Beach, M. Zou, and et. al. Exploring models and mechanisms for exchanging resources in a federated cloud. In *Intl. Conf. on cloud engineering (IC2E 2014)*, 2014 - Accepted.

[22] P. Riteau, M. Tsugawa, A. Matsunaga, et al. Large-scale cloud computing research: Sky computing on FutureGrid and Grid'5000. In *ERCIM News*, 2010.

[23] I. Rodero, F. Guim, J. Corbalan, and A. Goyeneche. The grid backfilling: a multi-site scheduling architecture with data mining prediction techniques. In *Grid Middleware and Services*, pages 137–152. Springer US, 2008.

[24] I. Taylor, E. Deelman, D. Gannon, and M. Shields. *Workflows in s-Science*. Springer, 2006.

[25] D. Villegas, N. Bobroff, I. Rodero, et al. Cloud federation in a layered service model. *J. of Computer and System Sciences*, 78(5):1330–1344, 2012.

[26] K. Yelick, S. Coghlan, B. Draney, et al. The Magellan report on cloud computing for science. Technical report, U.S. DoE Office of Advanced Scientific Computing Research (ASCR), 2011.

[27] FutureGrid. https://portal.futuregrid.org/.

[28] Nimbus Project. http://www.nimbusproject.org/.

[29] Open Cloud Computing Interface (OCCI). http://occi-wg.org/.

[30] OpenStack Project. http://www.openstack.org/.

[31] Siena Initiative. http://www.sienainitiative.eu.