A Broker-based Framework for Multi-Cloud Workflows

Foued Jrad *
foued.jrad@kit.edu

Jie Tao
jie.tao@kit.edu

Achim Streit
achim.streit@kit.edu

Karlsruhe Institute of Technology KIT
Steinbuch Centre for Computing
Hermann-von-Helmholtz-Platz 1
Eggenstein-Leopoldshafen 76344, Germany

ABSTRACT

Computational science workflows have been successfully run on traditional HPC systems like clusters and Grids for many years. Today, users are interested to execute their workflow applications in the Cloud to exploit the economic and technical benefits of this new emerging technology. The deployment and management of workflows over the current existing heterogeneous and not yet interoperable Cloud providers, however, is still a challenging task for the workflow developers. In this paper, we present a broker-based framework for running workflows in a multi-Cloud environment. The framework allows an automatic selection of the target Clouds, a uniform access to the Clouds, and workflow data management with respect to user Service Level Agreement (SLA) requirements. Following a simulation approach, we evaluated the framework with a real scientific workflow application in different deployment scenarios. The results show that our framework offers benefits to users by executing workflows with the expected performance and service quality at lowest cost.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.4 [**Performance of Systems**]: Modeling techniques, Performance attributes; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Cloud Computing; Intercloud Computing; Cloud Broker; Cloud Workflow; Multi-Cloud

1. INTRODUCTION

Workflow is a technology that allows users to split a complex problem into small partitions that can be solved using a single computing unit, like a computing node of a cluster system. According to [12], scientific workflows are "coarse-grained parallel applications that consist of a series of computational tasks logically connected by data- and control-flow dependencies". The workflow technology has been well established on different computing infrastructures, for example, the Grid.

Following the Grid model, Cloud computing provides another computing paradigm that focuses on the on-demand provisioning of computing resources, including hardware, software, storage and network. The concept of Cloud computing has been commonly accepted by different scientific domains. As the number of Cloud users has been widening, the number of Cloud providers is also continuously increased. The currently available Cloud platforms distinguish themselves in the service type, the cost, the Quality of Service (QoS) as well as performance. This fact brings Cloud customers the flexibility of freely selecting their target architecture from a broad range of Cloud platforms. However, it raises at the same time the interoperability problem between the different Clouds.

In the last years many researchers have migrated workflow systems onto the Cloud. However, the current implementations mainly address a single Cloud. This means that the workflow applications can still not be run on a federated environment that involves multiple Clouds. The approach of running a workflow application within a single Cloud may work well for workflows with small requirements on computing resources and small data input. Nevertheless, there are scenarios where a single Cloud cannot run a workflow efficiently as users are expecting, or even the workflow cannot be deployed on a single Cloud. For example, some tasks of a workflow application may have special demand on the underlying hardware, which cannot be fulfilled by the target Cloud. Large scientific workflows contain several thousands of tasks, where the required computing and storage capacity may exceed the limit of an individual Cloud platform. Moreover, like any Cloud customer, workflow users also expect the best performance with the lowest payment. For these use cases a multi-Cloud workflow framework that executes workflow applications using multiple Clouds will show its advantage in terms of service quality, cost-saving and the capability of running workflows with specific hardware requirements.

Motivated by the above considerations, this work developed a multi-Cloud workflow framework. The framework is based on a Cloud service broker we actually developed for automatically selecting a suitable Cloud for the users with consideration of the users' hardware requirements, SLAs of

*Corresponding Author

MultiCloud'13, April 22, 2013, Prague, Czech Republic.

the providers, as well as the cost and the load on a Cloud. With the help of the service broker, different Clouds are selected to run the workflow tasks towards a better performance/cost behavior. For the workflow framework, we extended the service broker with a data management component for dealing with the data flow across the workflow tasks. We also implemented a scheduling mechanism to distribute the workflow tasks to the selected Clouds. In summary, this work made the following contributions:

1. A broker-based framework to support the execution of workflow applications on a multi-Cloud environment.
2. A simulation environment to evaluate the framework.
3. The scheduling and matching policies for selecting the suitable Clouds to run the workflow tasks based on the user requirements.
4. An evaluation of the cost and performance gains of running workflows on top of the broker framework.

The conducted simulation experiments show that our implemented workflow framework offers benefits in term of performance and cost compared to the use of a single Cloud. We found that our proposed broker-based solution unburdens the user in many ways from the complexity of choosing the suitable providers for running workflow applications. We demonstrated also how the use of state of the art Cloud standards can resolve the rising heterogeneity problem of Clouds that hindered the multi-Cloud workflow deployment.

The rest of the paper is organized as follows: Section 2 presents the related work on workflow management systems for Grid and Cloud. Section 3 describes the architecture of the implemented broker-based workflow execution framework. Section 4 presents the simulation-based environment and the workflow application used to validate and evaluate the framework. Section 5 shows the performance and cost evaluation results gathered from the simulation experiments. Finally, Section 6 concludes the paper and provides the future work.

2. RELATED WORK

Cloud computing is a novel computing paradigm. However, it is based on existing technologies such as Grid and utility computing. This section describes related work in the field of workflow management on Grid and Cloud, where the common problem as this work, i.e., how to automate the deployment of service workflows over heterogeneous, distributed computing resources, have to be solved.

2.1 Workflow Management Systems for Grid

Grid computing allows the community based sharing of computing resources across different administrative domains. Due to the heavy heterogeneity and geographical distribution of the Grid resources, it is mandatory to have a kind of orchestrator component responsible for workflow's tasks management across the Grid sites. This component is called Workflow Management System (WfMS). The main tasks performed by a WfMS are: workflow description, resource discovery, task scheduling, data movement and monitoring. A comprehensive taxonomy and detailed comparison of existing WfMSs for Grid is provided by Yu and Buyya [17] as well as in the Grid Workflow Forum[1].

[1] http://gridworkflow.org

The execution of scientific workflows over Grid resources has been heavily investigated for more than a dozen of years by researchers in the field of distributed computing. Many well-known Grid middleware stacks already integrated a workflow engine as a part of their distribution. Examples are the workflow service [7] contained in UNICORE[2], the DAGMan component provided within the Condor middleware[3] and the Karajan workflow engine developed as an extension to the Globus toolkit[4]. Alternatively, the workflow engine can be implemented as a single high level component to be independent from the Grid middleware, e.g. the Gridbus Workflow Engine (GWFE) developed in the context of the Gridbus Project[5] at the CLOUDS Lab, University of Melbourne, and the P-Grid Portal[6], which supports both the gLite[7] and Globus-enabled resources. Some workbench-based WfMSs, originally designed to run a user's workflow on the client machine, such as Kepler[8] and Taverna[9], have been extended to make remote calls to Grid services for processing specific tasks inside the workflow. A major issue of this approach is that the client computer needs to stay online during the workflow execution.

A well established WfMS used to plan, execute and monitor scientific workflows on the Grid is Pegasus[10]. It can interact with Condor as well as Globus Grid services and is actually used to run large scale workflows in many academic infrastructures like FutureGrid[11]. Another widely used Grid WfMS by the German D-Grid communities[12] is GWES (Generic Workflow Execution Service)[13]. Its deployment as Web service allows the easy integration of this framework into Grid portals and with other workflow platforms like Taverna. Although the GWES modular design allows the support for several Grid middleware stacks, the current implementation supports only Globus.

2.2 Workflow Management Systems for Cloud

Cloud computing adopts many of the gained experiences in workflow management from Grid computing. Although Cloud brings many technological benefits [12] to the workflow developers, such as on-demand provisioning, elasticity, provenance and reproducibility, it also brings them new challenges [6]. In addition to the resource heterogeneity and the lack of common Cloud standards, other factors, such as the economic and the QoS considerations, known as non-functional service requirements, become crucial with the execution of workflows on the Cloud. Furthermore, the data management and security issues need to be concerned before migrating workflows to the Cloud. While WfMSs for Grids are established in the market, the automation of workflows in Cloud environments is currently in the research phase. The following are several existing works dealing with the execution of workflows on the Cloud.

[2] http://www.unicore.eu
[3] http://research.cs.wisc.edu/condor/
[4] http://www.globus.org
[5] http://www.cloudbus.org/broker/
[6] http://portal.p-grade.hu
[7] http://web.infn.it/gLiteWMS/
[8] https://kepler-project.org
[9] http://www.taverna.org.uk/
[10] http://pegasus.isi.edu
[11] https://portal.futuregrid.org
[12] http://www.d-grid.de/
[13] http://gridworkflow.org/kwfgrid/gwes-web/

A common approach for deploying workflows on the Cloud is to adapt existing Grid WfMSs to the Cloud environments. An example is the work in [9], which extended the Kepler WfMS to use Amazon EC2[14] Cloud services. Juve et al. [13] evaluated the cost-performance trade-off of executing real workflow applications on EC2 with data pre-staged from the Amazon S3[15] Cloud storage. For this, they used the Wrangler tool [13] together with the Pegasus WfMS to provision and configure virtual clusters on top of the EC2 instances on the same way as on the HPC/Grid clusters. They found that the choice of a storage system has a significant impact on the workflow runtime and execution costs. Buyya et al. [14] propose, in context of the Cloudbus project, a visionary architecture for market-oriented Cloud computing. A key component of the proposed architecture is a Market-Maker broker, acting as a mediator between user requests and the available Cloud resources, and a workflow engine used to schedule workflow tasks to the resources based on the QoS requirements. The Cloudbus framework is still under development, but a first prototypical implementation based on the Aneka Cloud platform [3] using EC2 proved the performance benefits of the envisioned architecture in executing workflows on the Cloud. The EU funded mOSAIC[16] project aims to simplify the development of Cloud applications including workflows. It proposes a framework composed of a Cloud Agency, which maintains the best resources configuration that satisfies the application SLA requirements, and a platform-independent programming model called mOSAIC API to support Cloud federations. A first prototypical implementation of the framework based on the agent technology is available and its evaluation with real applications and Clouds is ongoing. One of the shortcomings of this approach for workflow developers is that they need to rebuild their applications to be compatible with the mOSAIC API. In [5] Oliveira et al. introduce the SciCumulus middleware, which follows the Many Task Computing (MTC) paradigm [15] to automate the execution of workflows on the Cloud. Their simulation-based evaluation of the SciCumulus architecture shows the performance gains from using parameter sweep and data fragmentation as parallelism techniques but not yet the monetary impact in executing workflows on commercial Clouds. Garcia et al. [8] propose a multi-agent architecture for the concurrent and parallel execution of workflows on multi-Clouds, where consumers, brokers, Cloud providers, and Cloud resources are represented by agents. Based on a simulation testbed, they evaluate the performance benefits from the agent-based workflow execution. However, their used service selection mechanism based on the Contract-Net Protocol (CNP) does not support yet the negotiation of non-functional SLA constraints. Further, Tao et al. [16] present a framework for Intercloud service combination consisting of a workflow system, capable of managing workflow tasks running on different Clouds, and a cost-performance prediction model. The prototype implementation of the framework is restricted to the EC2-based Cloud services and does not support yet the automatic service selection.

To the best of our knowledge, only few of the mentioned approaches, i.e., [8, 14], have investigated the deployment of workflows in multi-Cloud environments. Although many of the approaches promise the support of heterogeneous Clouds, their concrete implementations mostly support only homogeneous Cloud environments. Moreover, most of the approaches, except [13], ignore to investigate the data transfer and inter-network communication between the workflow tasks, which are crucial for Intercloud computing. Furthermore, the selection of the needed Cloud resources for the workflow deployment is mostly based on functional SLA requirements and costs, while the non-functional SLA requirements are usually not considered.

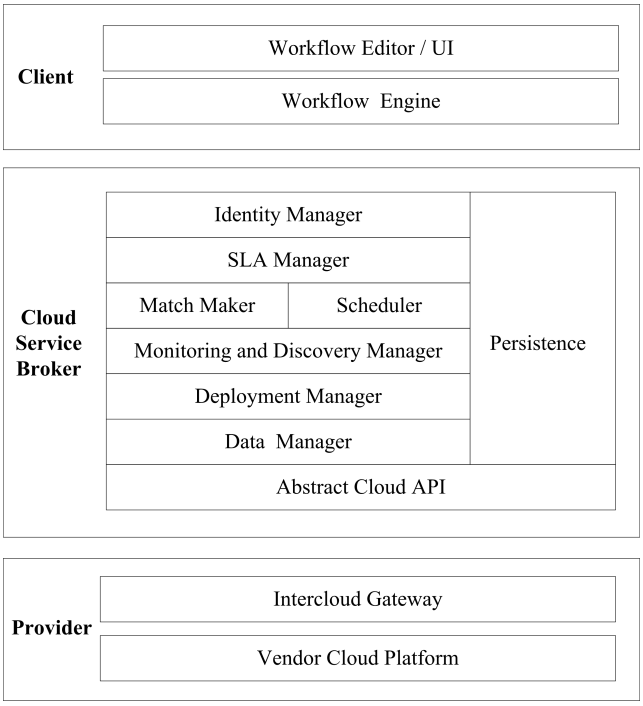

Figure 1: Software architecture of the multi-Cloud workflow framework

The broker-based workflow framework proposed in this paper aims to solve the above issues by providing interesting features, such as SLA negotiation, automatic SLA-based service selection, Cloud interoperability as well as data management. In addition, its high-level generic architecture design allows more extendability and integration into current Cloud infrastructures. The framework components has been implemented in a simulation environment using latest Cloud standards and workflow technologies.

3. THE WORKFLOW FRAMEWORK

Based on a previous work, the Cloud Service Broker [11], we designed a broker-based workflow framework supporting Intercloud computing.

Figure 1 depicts the software architecture of the proposed workflow framework. The Cloud Service Broker, as shown in the middle of the architecture, serves as a mediator between the users and the Cloud providers. As mentioned, this broker is actually developed to help Cloud customers find a suitable target platform for running their applications. Its main component is a Match Maker that performs a matching process, where the requirements of users are compared to the SLA properties as well as the runtime status of the Clouds.

[14] http://aws.amazon.com/ec2/
[15] http://aws.amazon.com/s3/
[16] http://www.mosaic-fp7.eu/

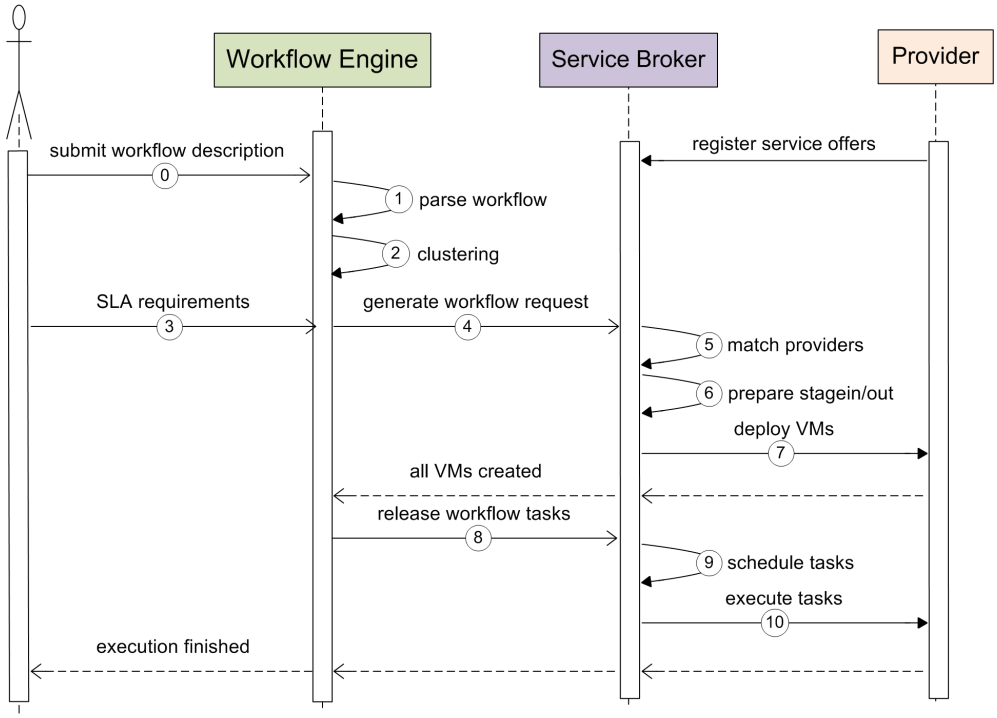

Figure 2: Workflow deployment flow

For this, it considers both the functional and non-functional SLA parameters. The former are hardware/software configurations such as number of cores, memory size or OS type. The latter are QoS related metrics like price, throughput and availability. We implemented different matching algorithms to make a trade-off between cost and performance of the selected Clouds. A scheduler is used to assign workflow tasks to the selected resources. The Data Manager is additionally included in the broker architecture to support the workflow data management. The details about its implementation will be given in Section 4. In addition, the service broker contains several other components, which are out of the focus of this paper. The Identity Manager is adopted to handle user authentication issues. The SLA Manager takes care of the SLA negotiation process. The Monitoring and Discovery Manager acquires the SLA and runtime information about the underlying Clouds. The Deployment Manager finally deploys the user-requested services on the selected Clouds. The service broker includes also an abstract Cloud API to interact with the underlying heterogeneous Clouds. This interaction is realized through the provider-hosted Intercloud Gateways, which provide a common interface to their corresponding vendor Cloud platforms.

The workflow system works on top of the Cloud Service Broker. An editor is offered for users to describe and submit their workflow applications. The Workflow Engine delivers the workflow tasks to the underlying Cloud Service Broker and takes care of their dependencies. In this way, it releases tasks to the Cloud Service Broker only when their parent tasks are terminated and their input data are available. The Match Maker component of the broker adopts its matching algorithm to assign the tasks to the Clouds that best fit the user requirements. The broker scheduler brings then the tasks to the selected Cloud platforms according to different scheduling policies, while the Data Manager takes care of the data transfer across the tasks.

Figure 2 shows the needed interactions between the Workflow Engine, the Cloud Service Broker and the Cloud platforms for a workflow deployment. In a first step, the user submits a workflow description to the Workflow Engine. After parsing the description (step 1), the Workflow Engine starts a clustering process (step 2) to reduce the number of workflow tasks. In a next step, the user submits his functional and non-functional SLA requirements (step 3). As response, a new workflow request is generated (step 4) and then forwarded to the Cloud Service Broker. In the following, the Match Maker is requested to match the Clouds that can fit these requirements by applying different matching policies (step 5). Additional tasks are added to the workflow by the Data Manager to prepare the data stage-in/out (step 6). After that all the requested resources are deployed on the selected Clouds (step 7), the Workflow Engine asks (step 8) the broker for assigning the workflow tasks to the underlying Clouds. The broker makes a scheduling plan (step 9), where each task is ordered to a target Cloud according to different scheduling policies. In the following, the broker brings the tasks to the Clouds and manages the data flow between them. Finally, it delivers the results of the execution to the user via the Workflow Engine (step 10).

In summary, we designed and implemented a software architecture of a workflow framework with the focus on implementing a working solution rather than a theoretical model. Therefore, the implemented technical features are the fundamental ones, leaving space for more advanced ones to be added at a later stage. The overall architecture is therefore stable to work with and extensible for future work.

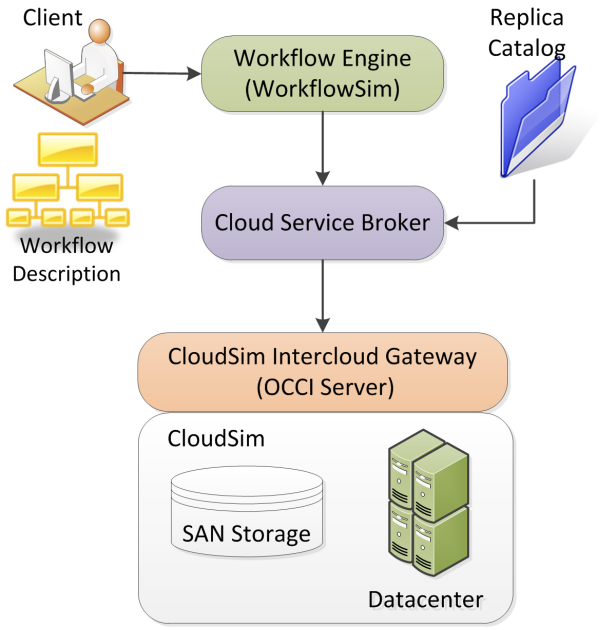

Figure 3: Simulation environment

4. VALIDATION PLATFORM

4.1 Simulation Environment

In oder to validate the proposed framework presented in Section 3 and to evaluate its provided workflow execution features, we implemented a simulation environment based on the CloudSim toolkit [2], which is able to model and monitor Infrastructure As a Service (IaaS) Clouds provisioned by large scale datacenters with internal brokers, physical hosts, as well as virtual machines (VMs). The simulation environment, as depicted in Figure 3, contains a CloudSim Intercloud Gateway that establishes a unified frontend interface for accessing the underlying datacenters (i.e., Clouds) and their local SAN storage. The Intercloud Gateway is implemented as a server using the Open Cloud Computing Interface (OCCI)[17] API to assure a minimum compatibility with the current Cloud standards. The Cloud Service Broker components were implemented as value-added services on top of the modeled CloudSim datacenters. For managing large scale workflows, we simply applied WorkflowSim [4], a modeled WfMS developed on top of CloudSim. Similar to the Pegasus WfMS, WorkflowSim contains a Workflow Mapper to map abstract workflows to concrete workflows, which are dependent on execution sites, a Workflow Engine to handle the tasks and data flow dependencies, and a Clustering Engine to reduce the number of tasks by applying different merging techniques. For the purpose of this work, we extended WorkflowSim to use the Cloud Service Broker as the scheduler instead of using an external one. In addition, a Replica Catalog keeps a list of data replicas by mapping input/output filenames to their current site locations. Currently, the data transfer is initiated by workflow tasks during their execution on the respective datacenters, whereas the Replica Catalog is managed by the Data Manager. We are now working to extend the Data Manager to handle all the workflow file transfer tasks. We assume that at the begin of the workflow execution all input files are located in the client local storage. The used data transfer policy by each task T executed on a datacenter D with respect to its type (stage-in/out or regular) and input files locations is described using the following pseudo-code:

```
Input: task T, datacenter D, replica catalog R

FileList=T.getFileList()
For all files F in FileList Do
 If tasktype(T)=stage-in Then
        transfer F from Client to D
 Elseif tasktype(T)=stage-out Then
        transfer F from D to Client
 Else
  If filetype(F)=input Then
     If R.get(filename(F))=null Then
        exit
     Else
      siteList=R.get(fileame(F))
      If D in siteList Then
         transfer F from local SAN of D
      Else
         For all sites S in siteList Do
          If S=Client Then
             bwth=Client-To-Cloud(D)
          Else if region(S)=region(D) Then
             bwth=Cloud-To-Cloud Intra-continental
          Else
             bwth=Cloud-To-Cloud Inter-continental
          Endif
          If bwth>maxBwth Then
             maxBwth=bwth and SourceSite=S
          Endif
         Endfor
         transfer F from SourceSite to D
      Endif
     Endif
  Else if  filetype(F)=output Then
        transfer F to local SAN of D
  Endif
 Endif
Endfor
```

4.2 Simulation Setup

In order to model a heterogeneous IaaS multi-Cloud environment, we configured up to 20 datacenters located in four world regions. Each datacenter is made up of 50 hosts, which are equally divided between two different host types. The detailed configuration for each datacenter is depicted in Table 1. All the datacenters are characterized by their own pricing policies and SLA parameters, which are availability, Client-to-Cloud throughput and response time. The pricing information for each modeled datacenter is acquired from a respective real IaaS provider, while the information on availability, response time and throughput was acquired from CloudSleuth[18] and CloudHarmony[19] from the same client host. The cost for network traffic and storage is not considered yet. For the purpose of evaluation, we modeled two simulation scenarios. In the first scenario, named "single Cloud", we deploy all the workflow tasks on the EC2 UK Cloud. In the second scenario, named "multi-Cloud", we use all the configured 20 datacenters and let the broker automatically select the suitable datacenters to deploy the workflow according to the used matching policy. For all the experiments, we configured the Match Maker component to use a simple matching policy called "Sieving" [10]. This policy

[17] http://www.occi-wg.org

[18] http://www.cloudsleuth.net

[19] http://www.cloudharmony.com

Table 1: Datacenters Setup

Parameter	Value or Range
Number of hosts per datacenter	50
CPU cores per host	8..16
Host CPU speed	1860..2660 MHZ
Host RAM size	8..16 GB
Host local storage	1 TB
VM-to-local-SAN bandwidth	100 Mbit/s
Intra-continental Cloud-to-Cloud bandwidth	30 Mbit/s
Inter-continental Cloud-to-Cloud bandwidth	10 Mbit/s
Number of datacenters	20
Datacenters' regions	Europe, USA, Asia and Australia

Table 2: Number of Jobs with different Cluster Numbers *k*

k	20	40	60	80	100
Jobs number	92	152	212	272	332

selects only the datacenters that satisfy all the user predefined functional and non-functional requirements. In case that more than one datacenter is matched, a random one is selected. In addition, the broker scheduler is configured to use the round robin scheduling policy. This policy allocates tasks to the first free VMs regardless of their type or datacenter location.

4.3 Workflow Application

For evaluating our workflow framework, we used a sample trace of the Montage workflow application generated by a real execution of the Pegasus WfMS on the FutureGrid Cloud testbed. Montage [1] is an astronomy application used to construct large image mosaics of the sky. A Montage workflow can be modeled as directed acyclic graph (DAG), where the vertices represent tasks, and the edges represent either data-flow or control-flow dependencies. All the tasks at a horizontal level are invocations of the same binary code operating on different input data. The used sample trace contains 7463 tasks within 11 horizontal levels, requires 3.07 GB of input data and produces about 31,06 GB of output data. As Montage spends more than 80% of the execution time in data transfer operations, it is considered as a data-intensive workflow. We imported the workflow trace formatted as DAG XML into the simulation environment with the help of the WorkflowSim Workflow Mapper. The runtime as well as the file size information for each task are imported from separate text files. Each task requires 1 CPU core to run. In order to reduce the scheduling overhead resulted when executing all the 7463 tasks, we use horizontal clustering with different cluster numbers to merge tasks at the same horizontal levels into clustered jobs. The cluster number k is defined as the maximum number of clustered jobs per horizontal level. Table 2 shows the resulted jobs number for each used k. In order to run the above described Montage workflow, the user requests 10 VMs of the type *small* and 10 VMs of the type *medium*. The resource requirements for each VM type are reported in Table 3. Both of the VM types

Table 3: VMs Setup; 1 CPU Core: 1GHZ Xeon 2007 Processor of 1000 MIPS; OS: Linux 64 bits.

VM Type	Cores	RAM (GB)	Disk (GB)
small	1	1.7	75
medium	2	3.75	150

Table 4: User non-functional SLA Requirements; Availability (Av); Response time (Rt); Throughput (Th).

Max willingness US$/hour	Min Av (%)	Max Rt (sec)	Min Th (Mb/s)
2.7	96	10	10

are offered by the modeled IaaS Clouds. We assume that all the VMs deployed on the same datacenter share a local SAN storage. The non-functional SLA requirements, given in Table 4, express the maximum payment willingness of the user as well as his minimum values of the SLA attributes to deploy the workflow with an acceptable QoS. These values are consumed by the brokering policy for a decision making to select the datacenters on which the requested 20 VMs will be deployed.

5. EXPERIMENTAL RESULTS

To evaluate the proposed framework with a real large scale workflow, we conducted a series of experiments using the above described simulation environment. All the simulations experiments are done on a host located in Germany. In all our conducted experiments we suppose that the client is located in Germany. The simulations results are presented in details in the following subsections.

5.1 Scalability Results

In order to evaluate the scalability of the implemented brokering framework with respect to increasing cluster number k, we measured in a first experiment the total time needed to execute a single run of the sample Montage workflow in minutes for both the single and the muti-Cloud scenario. In addition, for the single Cloud scenario, we simulated the case of adding a stage-in job to transfer all required input files before executing the workflow tasks. We repeated the experiment five times and then computed the average value. Figure 4 illustrates the results achieved: Note that the measured execution time includes a layered overhead composed of the clustering delay, workflow engine delay, postscript delay, and the queue delay. The used delay values were extracted from the workflow trace to make the simulation realistic as possible. The Workflow Engine can release maximal 5 jobs to the broker in a scheduling interval. As depicted in the figure, a 20 stepwise increase of k results for all the simulated scenarios in an increase of about 10 minutes in the workflow execution time. This demonstrates that our framework scales well with increasing number of the workflow jobs. Clearly, the single Cloud case benefits from a shorter execution time because for the multi-Cloud case the Intercloud data transfer is more time-consuming than a transfer from the local SAN storage. However, when adding a stage-in job to the single Cloud case, the execution of the workflow spends more time, as all the other tasks should wait for the stage-in job to finish before their execution start.

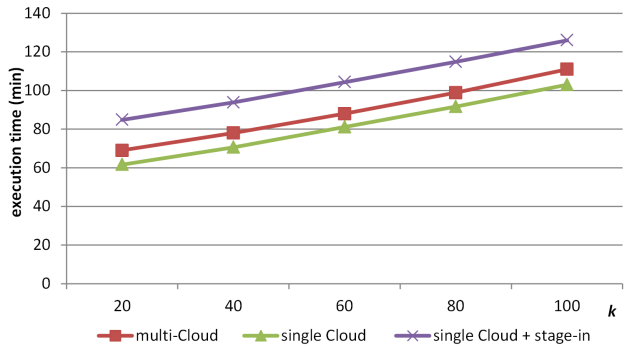

Figure 4: Workflow execution time with different cluster numbers k

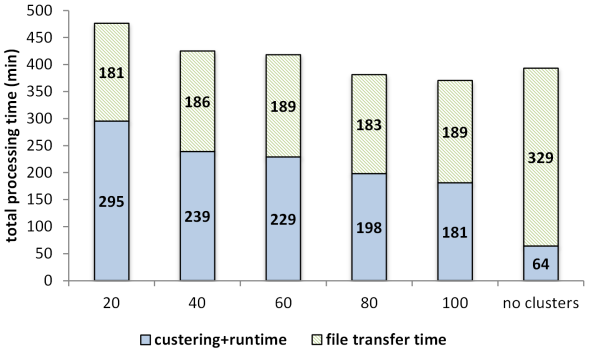

Figure 5: File transfer overhead for the multi-Cloud case with different cluster numbers *k*

5.2 Data Transfer Overhead

We conducted another experiment to evaluate the resulted data transfer overhead during a workflow deployment with the multi-Cloud scenario. We repeated the previous experiment to measure the proportion of time consumed to transfer all the required files compared to the total pure runtime (includes clustering delay) for all workflow tasks. The results with different cluster numbers are shown in Figure 5. Note that for the Intercloud transfer time calculation the real network latency between Clouds is not considered yet. It can be seen from the figure that the total pure runtime decreases with larger k. This decrease is caused by the logically decrease of the clustering delay, since with a bigger cluster number the number of merged tasks in a clustered job is reduced. On the other hand, the data transfer time remains between 180 and 190 minutes for a k between 20 and 100. To show the benefit from using clustering, we repeated the same experiment by disabling the clustering. We found that about the double time is spent for the data transfer compared to a deployment with enabled clustering. In order to classify the types and origins of the transferred files, we counted during the simulation the VM-to-SAN-Storage (local), Cloud-to-Cloud (Intercloud) and Client-to-Cloud transfers for each of the workflow tasks. The total number and size of the transferred files during a deployment with k=20 are depicted in Figure 6. The results approve that the transfer time overhead is heavily affected by the Intercloud transfers. Therefore, a reduction of the number of transfers between the Clouds will probably approve the workflow execution performance. One possible solution for this is to use an optimized brokering and scheduling policy in the Cloud Service Broker.

5.3 Service Quality

We compared the average non-functional SLA values (cost, availability, response time and throughput) of the matched datacenters for the multi-Cloud case with the corresponding values in the case of deploying the workflow on the single EC2 UK Cloud. The results are depicted in Table 5. It can be seen that besides EC2 UK, FlexiScale UK[20] and CityCloud Sweden[21] also fulfill all the SLA user requirements

[20] http://www.flexiscale.com
[21] http://www.citycloud.eu

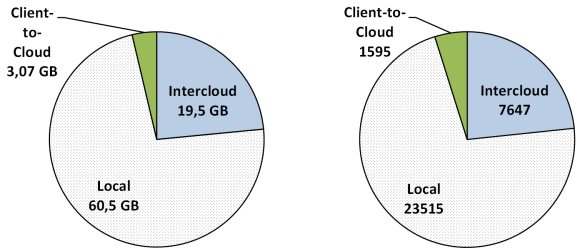

Figure 6: Total size (left) and number (right) of transferred files for the multi-Cloud case with k=20

listed in Table 4. The other Clouds fail either because their expected execution costs are above the maximum user's payment willingness or they cannot deliver the minimal required Client-to-Cloud throughput due to their non-closeness to the client (i.e., they are not located in the same geographical region as the client). The table shows also that the multi-Cloud case works better than the single Cloud case in terms of cost-saving. This lies in the fact that FlexiScale and CityCloud deliver the requested VM types with lower costs and with almost the same or a better service quality than EC2.

5.4 Results Discussion

As described above, we performed different simulation experiments and measured the workflow execution time, the data transfer time as well as the quality parameters. Overall, our experimental results show that one Cloud may execute a workflow application faster than using multiple Clouds because the inter-task data transfer is performed within a sin-

Table 5: Average non-functional SLA Values For the matched Datacenters.

Datacenters	cost US$/h	Av (%)	Rt (sec)	Th (Mb/s)
single-Cloud (EC2 UK)	2.55	99.97	3.63	19.11
multi-Cloud (EC2 UK +FlexiScale UK+ CityCloud SW)	2.27	99.86	3.91	21.66

gle platform. However, when the input data of the workflow task must be transferred from other locations, e.g. other Clouds, to the target platform, the multi-Cloud workflow framework may work better in terms of execution time. We found also that clustering is a necessary technique for running workflows on multiple Clouds because it reduces the number of tasks and hence reduces the amount of data to be transferred between the tasks. Finally, the broker-based workflow framework generally introduces better QoS while reducing the user payment by selecting Clouds with lower cost and better service quality.

6. CONCLUSIONS AND FUTURE WORK

This work implemented a workflow framework for running workflow applications on a multi-Cloud environment. The framework is based on a Cloud Service Broker we developed to help users choose the target platform with respect to their requirement on hardware, cost, performance, etc. The developed framework was validated with a large scale workflow application in different scenarios. The experimental results show the advantages of running workflows with multiple Clouds, in contrast to the case of using a single Cloud platform.

In the next step of this research work, we will take care of the data locality issues in the multi-Cloud environment. A locality mechanism will be implemented to bring the computation to its data for use cases where the input data of the workflow tasks are distributed across different Clouds. The mechanism aims to reduce the time for data transfer between the Clouds. Furthermore, we will extend the framework to support Storage as a Service (STaaS) Clouds like S3 and to consider the cost for network traffic and storage usage. Finally, we will try different brokering and scheduling policies (e.g. MCT scheduling) in order to improve the workflow execution performance.

7. ACKNOWLEDGMENTS

We would like to thank Weiwei Chen from the University of Southern California for his contribution to this work by providing us with the WorkflowSim source code.

8. REFERENCES

[1] G. B. Berriman, E. Deelman, J. C. Good, J. C. Jacob, D. S. Katz, C. Kesselman, A. C. Laity, T. A. Prince, G. Singh, and M.-H. Su. Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In *Proceedings of the Society of Photo-Optical Instrumentation Engineers Conference*, volume 5493 of *SPIE 04*, pages 221–232, September 2004.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, 41(1):23–50, January 2011.

[3] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya. The Aneka platform and QoS-driven resource provisioning for elastic applications on hybrid Clouds. *Future Generation Computer Systems*, 28(6):861–870, June 2012.

[4] W. Chen and E. Deelman. WorkflowSim: A Toolkit for Simulating Scientific Workflows in Distributed Environments. In *Proceedings of the 8th IEEE International Conference on eScience*, Chicago, USA, October 2012.

[5] D. de Oliveira, E. Ogasawara, F. Baião, and M. Mattoso. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. In *Proceedings of the 3rd IEEE International Conference on Cloud Computing*, CLOUD '10, pages 378–385, Washington, DC, USA, 2010.

[6] E. Deelman, G. Juve, and G. B. Berriman. Using Clouds For Science, is it Just Kicking The Can Down the Road? In *Proceedings of the International Conference on Cloud Computing and Services Science*, CLOSER 2012, pages 127–133, Porto, Portugal, April 2012.

[7] B. Demuth, S. Bernd, H. Sonja, M. D. Jason, G. André, H. Valentina, and S. Sulev. The UNICORE Rich Client: Facilitating the Automated Execution of Scientific Workflows. In *Proceedings of 6th IEEE International Conference on eScience*, pages 238–245, 2010.

[8] J. O. Gutierrez-Garcia and K. M. Sim. Agent-based Cloud Workflow Execution. *Integrated Computer-Aided Engineering*, 19:39–56, 2012.

[9] M. Hardt, T. Jejkal, I. Campos, E. Fernandez, A. Jackson, D. Nielsson, B. Palak, and M. Plociennik. Transparent Access to Scientific and Commercial Clouds from the Kepler Workflow Engine. *Computing and Informatics*, 31:1001–1015, 2012.

[10] F. Jrad, J. Tao, R. Knapper, C. M. Flath, and A. Streit. A utility-based approach for customised cloud service selection. *Int. J. Computational Science and Engineering*, forthcoming.

[11] F. Jrad, J. Tao, and A. Streit. SLA Based Service Brokering in Intercloud Environments. In *Proceedings of the International Conference on Cloud Computing and Services Science*, CLOSER 2012, pages 76–81, Porto, Portugal, April 2012.

[12] G. Juve and E. Deelman. *Scientific Workflows in the Cloud*, pages 71–91. Computer Communications and Networks. Springer London, 2011.

[13] G. Juve, E. Deelman, G. Berriman, B. P. Berman, and P. Maechling. An Evaluation of the Cost and Performance of Scientific Workflows on Amazon EC2. *Journal of Grid Computing*, 10:5–21, 2012.

[14] S. Pandey, D. Karunamoorthy, and R. Buyya. *Workflow Engine for Clouds*, pages 321–344. John Wiley, Inc, 2011.

[15] I. Raicu, I. T. Foster, and Y. Zhao. Many-Task Computing for Grids and Supercomputers. In *Proceedings of the IEEE Workshop on Many-Task Computing on Grids and Supercomputers*, MTAGS 08, pages 1–11, Austin, TX, USA, November 2008.

[16] J. Tao, D. Franz, H. Marten, and A. Streit. An Implementation Approach for Inter-Cloud Service Combination. *International Journal on Advances in Software*, 5:65–75, 2012.

[17] J. Yu and R. Buyya. A Taxonomy of Scientific Workflow Systems for Grid Computing. *SIGMOD Rec.*, 34(3):44–49, September 2005.