Seamlessly Enabling the Use of Cloud Resources in Workflows

Michael Gerhards, Volker Sander
Faculty of Medical Engineering & Technomathematics
FH Aachen, University of Applied Sciences
Jülich, Germany
{M.Gerhards|V.Sander}@fh-aachen.de

Adam Belloum
Institute of Informatics
University of Amsterdam
Amsterdam, The Netherlands
A.S.Z.Belloum@uva.nl

***Abstract*—The hosting of large on-premise computational resources is common practice. Cloud Computing offers a promising, alternative infrastructure for using scalable on-demand off-premise resources. However, outsourcing whole applications is not a cost optimal solution in some scenarios, because the already existing on-premise resources are not considered. A flexible integration of additional resources from the cloud to compensate a shortage of suitable on-premise resources is a tradeoff between costs and efficiency. This paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. The approach is evaluated by running an example workflow.**

Keywords-cloud economics; dynamic resource allocation; cloud computing; cross-cloud workflows; on-demand computing model; service oriented architecture; workflow; workflow orchestration.

I. INTRODUCTION

Refactoring on-premise computational resources to form a computer center is common practice. However, it is not reasonable to provide a solution for all requested resource types in such a center. First of all, the initial purchase costs are very high. For small and medium enterprises (SME) it is nearly impossible to bear these costs alone. Even after a purchase the disadvantages still occur, mainly due to the operational costs. The hosting company is bound to the resources for many years, even if the computational power is no longer required. The old hardware does not benefit from new technologies, which were developed in the meantime. If specific resources are used with unbalanced load, there is the risk of underuse. An overprovisioning is also required for load peaks which also increase the costs.

Cloud computing offers a promising alternative infrastructure for using scalable on-demand resources. Providers such as Amazon allow users to allocate virtualized computational resources. Of course, those providers allow for porting the full application. However, this might not be the most cost-effective solution, because the already existing on-premise resources are not considered. Therefore, for many scenarios it appears to be opportune to integrate cloud resources with easy-scale and dynamic provisioning into the local environment for the execution of computation intensive application parts whereas the other application parts are executed on local available general-purpose computational resources. An example is a highly parallelized application which could use a Graphics Processing Unit (GPU) in the cloud, while the remainder of the program is executed locally.

This paper will briefly present existing complex software stacks which combine on-premise resources with cloud resources. Then it introduces our light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. The paper will focus on workflows because the division of applications into parts is natively supported. The basic ideas apply to a much broader application domain.

The paper is organized as follows: The second section presents the cloud-enabled workflow environment. It introduces the challenges for such an environment and provides solutions. The third section evaluates the presented solutions by describing a run of an example workflow in a specific workflow management system under the use of cloud resources. The last section concludes the lessons learned and provides future work. For simplicity reasons we omitted to refer to related work in an isolated section. Instead we provide references when the according context is discussed.

II. CLOUD-ENABLED WORKFLOW ENVIRONMENT

Many publications deal with cloud computing since it is the greatest IT hype of the last ten years. Surprisingly the combination of cloud computing with workflows is little addressed. "With the emerging of the latest cloud computing paradigm, the trend for distributed workflow systems is shifting to cloud computing based workflow systems [1]." In comparison to the mobile smart domain, approaches like CloneCloud already exists to dynamically partition applications between weak devices and clouds [2]. Nephele is another approach that claims to be "the first data processing framework to explicitly exploit the dynamical resource allocation offered by today's compute clouds for both, task scheduling and execution [3]." Nephele itself is focused on performance in full cloud environments but does not consider available on-premise resources which results in a lower performance but a cost reduction. A tradeoff between costs and performance is missing.

 ISBN: 978-1-61208-271-4

Workflows in cloud computing are addressed by several EU projects. "BREIN takes the e-business concept developed in recent grid research projects, namely the concept of so-called "dynamic virtual organizations" towards a more business-centric model, by enhancing the system with methods from artificial intelligence, intelligent systems, semantic web etc. [4]." BREIN can enhance some cloud features like automatic resource allocation and outsourcing of resources to third party. The approach presented in this paper also focuses on resource allocation and outsourcing but from a more technical sight by combining existing lightweight technologies. It does not consider collaboration between companies. The required components of the overall architecture are similar: A workflow framework with service broker and registry.

A. Service layers and deployment models

The National Institute of Standards and Technology (NIST) distinguishes the three service layers: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) and four different deployment models: Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud [5]. The cloud-enabled workflow environment differs dependent of the used service layer and deployment model. A detailed comparison of the different service models and deployment models is given in [6]. The rest of the paper will therefore focus on Public Cloud IaaS resources to assume the minimum of requirements. This should not limit the generic aspects of the proposed solution since other service layers and deployment models can be used instead with less effort.

B. Security and Governance

This paper assumes that the workflow management system runs on-premise or in a private cloud and is used only by users of a single organization. This assumption simplifies the security handling since the organization is interfacing with the cloud service providers as a whole. Cross-organizational environments can be addressed by applying the concept of virtual organizations [7].

While incurred costs would be billed against the organization, the actual costs still have to be mapped to cost units within the organization. Therefore, an AAAA (Authentication, Authorization, Admission control, and Accounting) is required. Actually, an AAAAA mechanism is demanded, i.e. an additional auditing mechanism like described in Section II.L.

During application runtime off-premise cloud resources will access on-premise data for calculations. To protect the data against unauthorized access credentials are required. These credentials are entered by the user at the start of the application. If a native support is not guaranteed, the credentials can be entered during a WS-HumanTask, which stores credentials in a secured short-lived repository with limited life time [8]. This procedure is used by our approach. The integration of tasks is detailed in Section II.F.

To assure authentication and authorization, we extend the idea of using WS-HumanTask for credentials and propose an architecture we presented in the context of our publication of a security framework for our WS-HumanTask implementation. This publication "presents a generic framework that supports a pull-based work distribution strategy in distributed environments with the help of a task repository that mediates tasks between resources and workflow instances [9]." It provides an implementation for Role Based Access Control (RBAC) based authorization. To provide a certificate repository, we follow the concept of MyProxy which is an authentication technology from the grid domain which lets the workflow impersonating the user [10].

C. Conditions on applications

A condition for executing different parts of the same application on different premises is an application which is divisible into parts. Modeling a complex application as workflow supports its division into simpler individual parts that are executed as interacting tasks by a workflow management system that takes care of the individual tasks' progress and dependencies [11].

The Generic Workflow Execution Service (GWES) is an open source workflow management system which was developed by Frauenhofer-Gesellschaft for the management and the automation of complex workflows in heterogeneous environments [12]. GWES was originally developed basing on grid technologies like Globus Toolkit as Grid Workflow Execution Service (also GWES) and was then adjusted to the cloud domain. To conclude GWES is a specific workflow management system with an own workflow description language.

In contrast, the interoperable approach presented in this paper bases on an extension for existing arbitrary workflow management systems by its loosely coupled connection to a cloud broker to enable the use of additional cloud resources. By choosing a workflow management system independent approach the benefit of using the already known system is given for the end-user.

AMOS is "a system that combines grid and cloud technologies in a novel way to support on-demand execution of e-Science applications [13]." The e-Science applications handled in this paper are also modeled as workflow and executed in the cloud. The main idea is the creation of a "transient grids by automatically installing and configuring grid middleware on the purchased resources". In contrast the approach of this paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack.

"OPTIMIS deliverables will enable clouds to be composed from multiple services and resources. It will support service brokerage via interoperability, and is architecture-independent [14]." It provides "a toolkit for supporting service provisioning using Cloud eco-systems consisting of multiple Cloud infrastructures from different providers with guaranteed Quality of Service (QoS)". A direct integration of workflows is not part of the project but as a future work the usage of OPTIMIS as underlying cloud infrastructure in combination with the workflow tools of this paper could be tested.

 ISBN: 978-1-61208-271-4

D. Resource independent modeling of workflows

The various tasks from a workflow are of different task types. Most task types like control flow or script tasks are executed on the workflow management system's computer. But service tasks are computation demanding and therefore executed as service - or other remote procedure call (RPC) - on suitable hardware resources. The workflow will run in a Service Oriented Architecture (SOA) in a combination of services, which are deployed on-premise and in the cloud. The invocation of the cloud services must be protected against unauthorized usage using a system like described in Section II.B. The data flow for large data sets is not integrated into the workflow but in the service software directly. Since pushing the data in a web service invocation message results in bad performance through marshaling the data, only the data location and an authorization ticket is send to the service. The service then loads the data using a third party high performance file transfer mechanism like GridFTP. The security aspect is handled in Section II.B. This paper presents how these script tasks can be executed on enabled cloud resources without workflow modification. If additional cloud resources are enabled is decided during runtime.

The concept of considering only physical resources is gone in the cloud vision of elastic resources, which can be instantiated on-demand. Therefore, workflows are modeled independently of specific resources by abstracting service endpoints as service names. This enables the easy exchange of an on-premise endpoint with an off-premise endpoint, e.g., in the cloud. The binding of workflow tasks to endpoints is done at runtime by dissolving the service names. The service registry contains assignments between all service names to available service endpoints independent if the endpoint is located on-premise or off-premise in the cloud. In Figure 1 both tasks "T1" and "T2" fetches their endpoints from the service registry. A so modeled workflow can be executed in the usual way without disadvantages.

Enterprise service buses (ESB) like Mule or Fiorano are also able to manage dynamic endpoints independently of the endpoint location [15]. However, compared to our solution, ESBs are rather heavyweight software products which increase the complexity of the architecture. Connectors between workflows and ESB are application dependent.

This paper provides a light-weight approach that focuses on seamlessly enabling cloud resources for workflow-based applications without requiring installing a rather complex software stack. Such an approach lowers the entry barrier. This empowers workflow users to benefit from the cloud in an easy way.

E. Enabling cloud resources using a broker

In cloud economics, resources are frequently provided following a pay-per-time billing structure. The time is billed when they are available even when the resources are not used. Therefore these resources are shut down when idling. If a shutdown resource is required at the service registry the resource must first be instantiated. According to the National Institute of Standardization (NIST) Cloud Computing Reference Architecture [5], the dynamic allocation of cloud resources is done by a cloud broker. The cloud broker is "an entity that manages the use, performance and delivery of cloud services, and negotiates relationships between cloud providers and cloud consumers [5]." The cloud broker publishes endpoints of instantiated cloud resources at the service registry.

F. Connection between workflow and cloud broker

The connection between workflow management system and cloud broker can be established at different locations in the overall workflow environment. Possible locations are tasks, called functions of tasks, the workflow, and the workflow management system itself is the source code is available. The advantages and disadvantages of the different connection locations are discussed in [1].

To not change the workflow management systems source code, the cloud broker connection is integrated into the workflow template itself. The workflow template can be seen as the source code of the workflow but not of the invoked services. A preprocessor creates a new extended workflow template out of the original workflow template. It consists of all original tasks in the given order but with interposed administrative tasks to handle the cloud broker connection for service tasks which should be executed in the cloud. The preprocessing process is also used to customize the workflow execution like described in Section II.G and to feed the provenance service of Section II.L.

The additional administrative tasks are similar to ESB adapters or cloud connectors. This new extended workflow is executed instead [16]. In Figure 1 the administrative task "AT" connects to the cloud broker to enable the cloud resource before its service is invoked by the service task "T2".

G. Identification of cloud tasks

Before the start of the workflow, the scheduler has to check if enough suitable on-premise resources are available. To realize this task, a resource description language like the Job Submission Description Language (JSDL) can be used to describe the different requirements for each individual task [17]. If not enough suitable local resources are available some tasks have to be redirected to cloud resources. Here, the scheduler must have all information about all constraints that apply to tasks that might be handled by cloud resources.

The user has the ownership of the data and decides which individual tasks are allowed for execution on integrated cloud resources. One possibility to model that is the usage of JSDL task annotations in the workflow template. This is similar to MAUI where developers annotate which methods of an application can be offloaded for remote execution [18]. If annotations are not supported in the workflow modeling language, another possibility is outsourcing the annotations to a workflow or task dependent configuration in a separate file with references to the original workflow template. Figure 1 illustrates the input of the scheduler and the annotation files together with the original workflow template to the preprocessor, which forms the extended workflow template. The administrative tasks of Section II.F are customized evaluating the annotations described above.

 ISBN: 978-1-61208-271-4

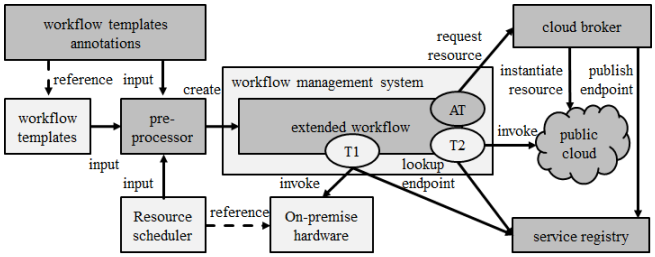

Figure 1. Cloud-enabled workflow environment. Components with bright background are the legacy system and components with dark background are extensions.

All tasks that will stay on-premise for execution are called local tasks whereas the tasks executed off-premise in the cloud are called cloud tasks. The cloud tasks get administrative predecessors and successors to connect to the cloud broker to enable the cloud resources. So all tasks are now arranged in one of these two categories. Since cloud tasks cause administrative overhead, they should only be used for computation intensive tasks like service tasks.

H. Endpoint selection strategy

At this point the workflow itself is prepared for an execution across organizational boundaries. The binding of service tasks to service endpoints is done at runtime by dissolving the service names at the service registry. Since the number of idling active cloud resources is kept to a minimum to avoid costs it is not guaranteed that the service registry holds an entry for the required service. The decision making plan to select an endpoint is illustrated in Figure 2 and explained in the following paragraphs.

The simplest case is illustrated in the first two branches: The service is already available and registered at the service registry. This is common if it is deployed on on-premise resources or in the cloud, e.g., from a previous run or as SaaS solution.

If the required web service is not available at the service registry, the service broker checks if a suitable underutilized or idling resource is running which represents the 3rd branch of Figure 2. The cloud broker re-installs the required software from a repository on that resource and publishes the new endpoint at the service registry. The installation process is described in Section II.I. This procedure is most suitable for workflows with different cloud tasks that can then be executed in a pipeline on the same cloud instance. It also reduces the data movement.

If neither suitable service nor resource is available a new resource representing the last branch of Figure 2 must be instantiated.. This process is presented in Section II.J. The instantiation takes time during provisioning and software installation which pause the task execution. It also causes new costs for renting an additional cloud resource.

Independent of the endpoint provisioning variant, the endpoint is now available and registered at the service registry. Like illustrated in Figure 1 the service tasks fetches their endpoints from the service registry and invokes the service directly. This proceeding is implemented in the workflow management system in its natural way.

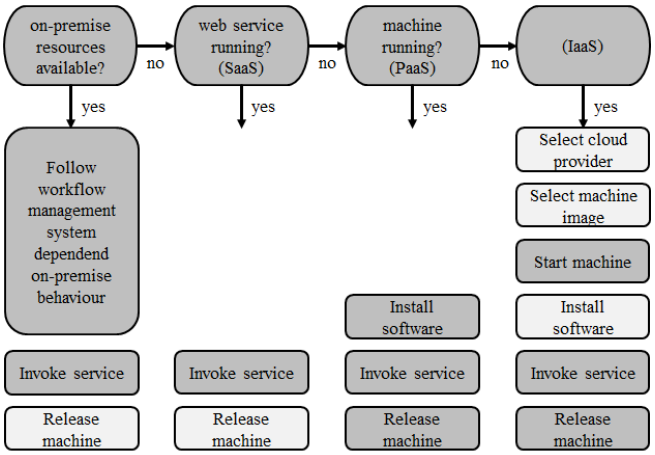

Figure 2. Endpoint selection decision process. Steps with bright background are optional and depend on the implementation.

I. Deployment of software on a running machine

The deployment of the web service including its required container can be done simply by using scripting (SCP / SSH | PowerShell). Password prompts can be suppressed using public/private key based authentication. The required keys are stored by the user in a secure key repository as provided for file transfer. The workflow is empowered to read these key using the mechanism described in Section II.B. A more sophisticated solution in comparison to scripting is to use cloud agnostic interfaces such as the Open Cloud Computing Interface (OCCI) or the compute API tool of jclouds [19][20]. The OCCI Working Group has highlighted the need for machine-readable Service Level Agreements (SLAs) associated with the dynamic provisioning of cloud computing resources.

J. Instantiation process of a new cloud resource

Preconfigured machine images contain only the required software for immediate use to speed up the instantiation. Each abstract cloud task uses its own machine image which is identified evaluating the abstract task's description in the workflow template. The cloud instance loads its machine image from its storage system. After startup, the web service endpoint is published to the service registry. An alternative is the use of a generic machine image which only contains the rudimentary software and is customized at runtime by additional software installation like described in Section II.I.

The billing period of a public cloud provider would start now together with the instantiation of the cloud resource instance.

K. Cloud Provider selection strategy

The flexible enabling of resources of the most suitable cloud provider for each individual task is an optimization to form a cross-cloud workflow with intra- and inter-cloud communications. The selection process can be modeled similar to the three-phase cross-cloud federation model described in [21]. In the discovery phase, the cloud broker collects information about assured properties offered by the

cloud providers. Each abstract cloud task specifies its requirements. “Each object is characterized by a set of properties/attributes; each property is a tuple (name, value), with name a string of characters and value [22].” In the match-making phase, the cloud broker compares the cloud task’s requirements with the cloud providers’ assured properties. The cloud providers that assure all requirements of the requesting task are potential task owners. In the authentication phase, the cloud broker selects the cheapest potential owner as the current owner for each cloud task. Matchmaking between requirements and properties was already handled in the grid domain. A “formal definition of matchmaking, overview algorithms to evaluate different matchmaking expressions, and develop a matchmaking service for an intelligent grid environment” is presented in [22].

One challenge arises if the workflow execution depends on large data because the data movement costs and time have to be considered. In [23], “a Network and Data Location Aware job scheduling has been proposed for data intensive jobs. The proposed scheduling algorithm takes into account network characteristics, disk read speed of data sources, and data locations of input files, as well as other computational factors (CPU power, memory, CPU load, etc.) when making scheduling decisions.”

L. Provenance

The importance of auditing the outcome of computation processes is a fundamental quality characteristic to many application domains. The automated tracking and storing of provenance information during workflow execution could satisfy this requirement [24]. The required data can be pushed out of the workflow by the administrative tasks introduced in Section II.F. Provenance traces enable the users to see what has happened during the execution of the workflow. This enables failure analysis and future optimization. Provenance becomes even more important in distributed environments because workflow tasks are loosely bound to computational resources. Using provenance in the cloud-workflow domain enables the identification of task to cloud assignments so that it is visible where the cloud task has been executed and where its data have been stored.

Provenance also shows at which time the cloud instance was running and therefore causing costs. Based on provenance traces, statistics can be created showing which workflows cause which costs, which users cause which costs, which clouds cause which costs, which users instantiate which workflows, which clouds execute which cloud task, etc.. A detailed comparison of two possible provenance models is done in [25].

III. EVALUATION

The prototype of [26] following the ideas of Section II is evaluated in this section. First an example workflow was modeled. Then required software products were chosen and deployed together with the self-developed cloud broker to form the cloud-enabled environment illustrated in Figure 1. Finally the example workflow was executed in the established testing environment. This evaluation shows how the lightweight system works basing on an example workflow. Not all components of the prototype were ready when this paper was written. Therefore, some are simulated using a mock like indicated at the corresponding place.

One advantage of combining on- and off-premise resources is a cost reduction attributable to the performance. Since cost structures vary they are not considered in this evaluation.

A. Example Workflow

The example BPMN 2.0 workflow illustrated in Figure 3 is taken from [26] where additional information like the source code is given. It solves a linear equation system. To not repeat previous work, only the minimum required information to understand this paper is given here.

The workflow consists of two script tasks, two service tasks, two parallel gateways, and the start as well as the end. The arrows indicate the task dependencies and the data flow which define the execution order of the tasks. A task can only start its execution after its predecessor has finished its own execution. The two script tasks are executed on the local computer. The two service tasks are executed on high-performance computation resources which can be on-premise or off-premise, e.g., in the cloud. The two parallel gateways split and merge the service tasks “Gauss” and “LuDecomposition”. That means that they can be executed independent of each other in an arbitrary order with no dependencies between them or even in parallel on different computers.

B. Used Software

The open-source flexible Business Process management (BPM) Suite jBPM of the JBoss community was used to evaluate the approach by running the example workflow of Figure 3. It provides an application server, a workflow engine to run workflows, an Eclipse Integrated Development Environment (IDE) with a Business Process Model and Notation 2.0 (BPMN 2.0) conform editor as plugin to model workflows, a data base to persist workflow runs, and a WS-HumanTask implementation to integrate human interactions into workflows in a standard conform way [8].

OpenNebula is an open-source software toolkit that enables the creation of Private, Public, and Hybrid Clouds [27]. This evaluation uses OpenNebula for local tests to simulate a Public Cloud provider on local resources to avoid expenses.

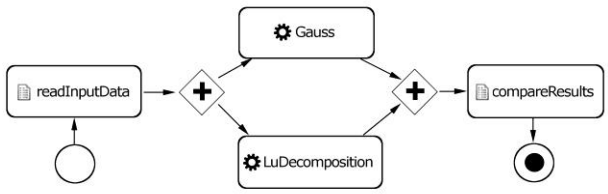

Figure 3. The example workflow consists of two script tasks, two service tasks, and two parallel gateways.

 ISBN: 978-1-61208-271-4

RESERVOIR is a FP7 project which bases on OpenNebula. "RESERVOIR's open-source approach supports the definition of open standards for Cloud computing in order to break the lock-in imposed by vendors today and allowing any organization to build its own local or public cloud infrastructure [28]." It allows building "on-demand infrastructure services, reducing investment and operational costs, increasing energy efficiency and elasticity while ensuring security and Quality of Service" (QoS). Future versions of our prototype could replace OpenNebula with RESERVOIR to get access to a more advanced toolkit and to integrate public cloud infrastructure resources in a standard conform way.

The clients for the equation solver web services are created by Java API for XML Web Services (JAX-WS) using the Java interface, the web service endpoint, and the web service description language (WSDL) file.

The software implementation to extend workflows is presented in [16]. The cloud service broker was self-developed following the prototype described in [26].

C. Workflow run

Before the instantiation of the workflow, the preprocessor requests the workflow template, the workflow annotations, and the information about available resources of the scheduler. The workflow annotations allow both service tasks to be executed off-premise. The scheduler was configured to indicate only enough available resources for one of the service tasks, the "LuDecomposition". That means that the "Gauss" task must be executed in the cloud which resources will be enabled during workflow runtime. The preprocessor then inserts the two administrative tasks "create" and "destroy" as predecessor and successor of the "Gauss" script task into the workflow template as only communication points between workflow and cloud broker. This new modified workflow template is then forwarded to the workflow management system for execution. The first script task reads the input data and forwards it to both service tasks. The "LuDecomposition" service task requests its service endpoint from the service registry. Since the endpoint is available on on-premise resources, the execution behavior of this service task is not influenced by the new architecture's components. The merge control flow task stops the execution branch until the "Gauss" service task finishes execution. The administrative "create" task connects to the cloud broker and forwards the execution requirements of its assigned "Gauss" service task. The cloud broker performs the decision making algorithm described in Section II.H. Suppose neither a service nor a computer is available. So the cloud broker selects the best cloud provider, instantiates a resource, and deploys the software. In this example only the private OpenNebula cloud was available and therefore chosen. The cloud broker requests the endpoint of the cloud resource and publishes it at the service registry. Now the "create" administrative task finishes execution. The "Gauss" service task first requests the endpoint from the service registry to invoke the service. The service task does not know that it is executed off-premise because of the design decision to abstract endpoints with service names, which are replaced during runtime. After the service returns the result to the workflow, the administrative task "destroy" notifies the cloud broker, that the service is no longer needed. The cloud broker terminates the cloud resource because no future cloud requests are predicted. Now all execution braches finished and the merge task starts the final script task which compares both results on the local computer.

IV. Conclusion and Future Work

This paper presented a general concept for the hybrid execution of workflows by enabling Cloud resources to compensate a shortage of on-premise resources. The proposed prototype has the advantage that it neither depends on a particular workflow management system nor on a particular workflow description language. It follows the approach of automatically modifying workflow templates to incorporate the steps for dynamically enable the appropriate off-premise resources in a flexible manner. The cloud broker automatically selects the most suitable cloud resource to guarantee the fulfillment of all task requirements. The end users' interfaces are not changed so that workflows can be used the same way as before.

Next steps of work will be an analysis of an according selection metric for the cloud broker to select the most suitable cloud service provider. The incurred costs of a partial off-premise execution will be compared with the costs of a full off-premise execution to calculate a costs reduction ratio and a cost-performance tradeoff. The time overhead for migrating tasks across cloud and organizational boundaries has to be measured for different providers and set it into relation with the avoided costs. Additionally, in the meantime developed technologies will be analyzed for a possible integration to benefit from related work.

Acknowledgment

This work was carried out in the context of HiX4AGWS [29]. HiX4AGWS is supported of the Federal Ministry of Education and Research in Germany. Grant No.: 17N3409.

References

[1] X. Liu, D. Yuan, G. Zhang, W. Li, D. Cao, Q. He, J. Chen, and Y. Yang, "The Design of Cloud Workflow Systems," SpringerBriefs in Computer Science, November 2011

[2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic Execution between Mobile Device and Cloud," Proceedings of the sixth conference on Computer systems (EuroSys11), pp. 301-314, April 2011

[3] D. Warnecke and O. Kao, "Nephele: Efficient Parallel Data Processing in the Cloud," Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS09), pp. 8:1-8:10, November 2009

[4] Business objective driven REilable and Intelligent grids for real busiNess (BREIN) FP7 project http://www.eu-brein.com/ [retrieved: March, 2013]

[5] P. Mell and T. Grance, National Institute of Standards and Technology (NIST), "The NIST Definition of Cloud Computing", Special Publication 800-145, September 2011

[6] M. Gerhards, V. Sander, and A. Belloum, "About the flexible Migration of Workflow Tasks to Clouds: Combining on- and off-premise Executions of Applications," Proceedings of the

Third International Conference on Cloud Computing, GRIDs, and Virtualization (CLOUD COMPUTING 2012), pp. 82-87, July 2012

[7] I. Foster and C. Kesselman, The Grid 2, ISBN 1-55860-933-4

[8] Web Service HumanTask V1.1 Committee Specification, August 2010, http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.pdf form [retrieved: March, 2013]

[9] M. Gerhards, S. Skorupa, V. Sander, P. Pfeiffer, and A. Belloum, "Towards a Security Framework for a WS-HumanTask Processor," 7th International Conference on Network and Service Management (CNSM 2011), pp. 1-5, October 2011

[10] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey, "Identity Federation and Attribute-based Authorization through the Globus Toolkit, Shibboleth, Gridshib, and MyProxy," 5th Annual PKI R&D Workshop, April 2006

[11] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, Vol. 3, Issue 3-4, pp. 171-200, September 2005

[12] Generic Workflow Execution Service (GWES) http://www.gridworkflow.org/kwfgrid/gwes/docs/ [retrieved: March, 2013]

[13] R. Strijkers, W. Toorop, A. van Hoof, P. Grosso, A. Belloum, D. Vasuining, C. de Laat, and R. Meijer, "AMOS: Using the Cloud for On-Demand Execution of e-Science Applications," Sixth International Conference on e-Science (e-Science), pp. 331–338, December 2010

[14] OPTIMIS FP7 project http://www.optimis-project.eu/ [retrieved: March, 2013]

[15] R. Woolley, "Enterprise Servcie Bus (ESB) Product Evaluation Comparisons", October 2006

[16] M. Gerhards, A. Belloum, F. Berretz, V. Sander, and S. Skorupa, "A History-tracing XML-based Provenance Framework for Workflows". The 5th Workshop on Workflows in Support of Large-Scale Science (WORKS), New Orleans, pp. 1-10, November 2010

[17] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva, Job Submission Description Language (JSDL) Specification, Version 1.0, 7 November 2005

[18] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A Survey of Mobile Cloud Computing: Architecture", Applications, and Approaches, Wireless Communications and Mobile Computing, Oktober 2011, DOI: 10.1002/wcm.1203

[19] Open Grid Forum (OFG), Open Cloud Computing Interface (OCCI), June 2011

[20] jclouds http://www.jclouds.org/documentation/gettingstarted/what-is-jclouds/ [retrieved: March, 2013]

[21] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation", 3rd International Conference on Cloud Computing (CLOUD), pp. 337-345, 2010

[22] X. Bai, H. Yu, Y. Ji, and D. Marinescu, "Resource Matching and a Matchmaiking Service for an Intelligent Grid", World Academy of Science, Engineering and Technology 1, pp. 666-669, 2005

[23] S. Kumar and N. Kumar, "Network and Data Location Aware Job Scheduling in Grid: Improvement to GridWay Metascheduler", International Journal of Grid and Distributed Computing, Vol. 5, No. 1, pp. 87-100, March 2012

[24] Y. L. Simmhan, B. Plale, and D. Gannon, "A Survey of Data Provenance in e-Science", SIGMOD RECORD, vol. 34, pp. 31-36, 2005

[25] M. Gerhards, V. Sander, T. Matzerath, A. Belloum, D. Vasunin, A. Benabdelkader, "Provenance Opportunities for WS-VLAM: An Exploration of an e-Science and an e-Business Approach", The 6th Workshop on Workflows in Support of Large-Scale Science (WORKS), pp. 57-66, November 2011

[26] M. Gerhards, M. Jagodzinska, V. Sander, and A. Belloum, "Realizing the flexible Integration of Cloud Resources into Workflows", Systemics and Informatics World Network (ISSN 2044-7272), Special Issue on Cloud Computing and Services, Dezember 2012 (in-press)

[27] OpenNebula Enterprise Cloud and Datacenter Virtualization http://www.opennebula.org [retrieved: March, 2013]

[28] RESERVOIR (Resources and Services Virtualization without Barriers FP7 project http://www.reservoir-fp7.eu/ [retrieved: March, 2013]

[29] History-tracing XML for an Actor-driven Grid-enabled Workflow System (HiX4AGWS), http://www.fh-aachen.de/en/research/projekt-hixforagws/ [retrieved: March, 2013]

 ISBN: 978-1-61208-271-4