About the flexible Migration of Workflow Tasks to Clouds

Combining on- and off-premise Executions of Applications

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, Netherlands
A.S.Z.Belloum@uva.nl

***Abstract* - An increasing number of applications target their executions on specific hardware like general purpose Graphics Processing Units. Some Cloud Computing providers offer this specific hardware so that organizations can rent such resources. However, outsourcing the whole application to the Cloud causes avoidable costs if only some parts of the application benefit from the specific expensive hardware. A partial execution of applications in the Cloud is a tradeoff between costs and efficiency. This paper addresses the demand for a consistent framework that allows for a mixture of on- and off-premise calculations by migrating only specific parts to a Cloud. It uses the concept of workflows to present how individual workflow tasks can be migrated to the Cloud whereas the remaining tasks are executed on-premise.**

Keywords - Cloud Computing; Cloud Service Broker; Grid Computing; Workflow; Workflow Orchestration

I. INTRODUCTION

An increasing number of applications target their execution on specific hardware. Field Programmable Gate Arrays (FPGAs) and free programmable general purpose Graphics Processing Units (GPUs) are existing approaches to use cost-effective high performance computational power in specific applications. Image processing and image guided interventions are well-known examples for use cases in which both platforms compete with each other [1].

However, not all parts of those applications are equally suitable for the usage of this hardware. Of course, related applications follow an approach in which only specific parts of a program were optimized for the specialized computation resources that are therefore only used during specific time slots. As a consequence, there is the risk that these resources are otherwise idling so that an own purchase might not be cost-effective. Therefore, for many scenarios it appears to be opportune to outsource computation intensive parts off-premise with easy-scale and dynamic provisioning whereas the other parts are executed on-premise on local available general-purpose computational resources.

Grid and Cloud Computing are potential infrastructures that support this scenario since both provide special resources for suitable application parts, whereas the remaining application parts can be executed on general resources. This concept can be extended to software in deploying software with expensive licenses on only some computers on Grids and Clouds. These computers were used to execute the application parts that require the deployed software, whereas the remaining parts might be executed elsewhere to make the computers available for other applications that rely on the related software.

But, not every organization has access to a Grid or does want to use it because it requires joining a related virtual organization [2]. Cloud Computing offers a promising alternative infrastructure for using scalable on demand resources with specific hardware. Providers such as Amazon allow users to allocate virtualized general purpose GPU-resources. Of course, those providers allow for porting the full application including the parts that rely on specific hardware to their premises. However, as described above, this might not be the most cost-effective solution. This paper addresses the demand for a consistent framework that allows for a mixture of on- and off-premise calculations. The proposed solution is based on workflows. The motivation scenario can therefore be viewed as an example for a concept that applies to a much broader application domain.

Modeling a complex application as workflow supports its division into simpler individual parts that are executed as interacting tasks by a workflow management system. These tasks are reusable for other workflows in the same way that software libraries are reusable in applications. Workflows are frequently used in e-Science for “climate modeling, earthquake modeling, weather forecast, astrophysics and high energy physics” [3] but also in the e-Business domain for Business Process Management (BPM).

The remaining of this paper is organized as follows: Section II introduces workflows with related definitions. It also provides an example in which parts of the workflow rely on specific hardware resources. Further on, it briefly describes the differences between Grids and Clouds according to workflow integration. Since the support of workflows in Cloud infrastructures is surprisingly rather limited, Section III introduces a novel approach to handle workflows in the Cloud computing domain. It provides technical descriptions, discusses possible alternatives, and provides more complex extensions. Section IV describes the related work and delimits the suggested architecture from an existing approach. Finally, the last section concludes the results and describes future work.

 ISBN: 978-1-61208-216-5

II. WORKFLOWS IN GRIDS AND CLOUDS

Complex processes are often modeled as workflows described using a specific workflow modeling language. A workflow is composed of several tasks, which could depend on each other. Therefore, a workflow can be illustrated as directed graph composed of tasks as nodes and task dependencies as directed edges. Directed edges connect the predecessor task with its successor task. A task can only start its execution if its predecessor has finished its own execution.

The example workflow illustrated in Figure 1 was designed for the Shape Retrieval Contest 2010 (SHREC'10) aiming to classify a set of proteins based on their 3D structure [4]. It consists of five tasks, illustrated as rectangles. The arrows illustrate the dependencies of the tasks. In this workflow data are only fed in at the beginning of the two task pipelines and are then handed over from task to task.

The tasks *APURVA* and *Sort* are computation intensive and well parallelizable. Therefore, they are candidates for a migration to off-premise computation resources like the Cloud, potentially by using specific High Performance Computing (HPC) hardware such as general purpose GPUs or FPGAs. In the following such tasks are called *Cloud Tasks*. The pre-processing of the *PP* tasks and the item duplication of the *X 1000* task should stay for execution on on-premise computation resources to reduce data movements and avoid costs. In the following such tasks are called *Local Tasks.*

A so-modeled workflow is called a workflow template that describes the behavior of a process; thus, it can be referred to as a general workflow definition. It is comparable with a program's source code. Such templates are deployed, instantiated, and executed on a workflow management system [5] that takes care of the individual tasks' progress and dependencies. Workflow instances follow the behavior of their assigned workflow template for a particular incident. It is comparable to a program's execution.

A particular challenge arises when workflows are mapped to resources at different organizations, each providing a heterogeneous system with non-uniform interfaces to access these resources. Thus, the submission of workflow jobs is more difficult due to the fact that different administrative domains have different accounting mechanisms.

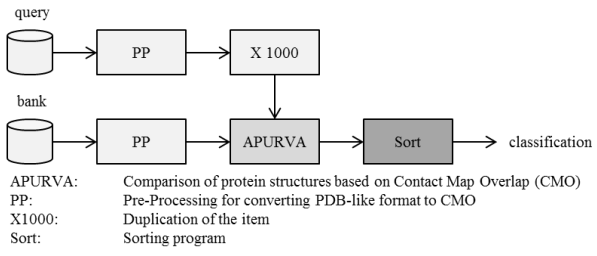

Figure 1. Example workflow with the two computation intensive tasks *APURVA* and *Sort.*

Grid middleware platforms support the execution of workflows in virtual organizations, where the distributed resources are owned by multiple organizations. Abstract Grid workflows are described independently of specific resources because new resources can be established or existing ones can be omitted or blocked. The binding of workflow tasks to Grid resources is done at runtime.

The Grid concept of considering only physical resources is gone in the Cloud vision of infinite resources that just have to be activated. The allocation of resources is different than in Grids. Any number of Cloud resources can be instanced on demand. "With the emerging of the latest Cloud Computing paradigm, the trend for distributed workflow systems is shifting to Cloud Computing based workflow systems [6]."

Cloud resources are not automatically part of a virtual organization and therefore not integrated into a trusted domain. The resource allocation mechanism differs from provider to provider. To execute a workflow task in a Cloud, the software must be deployed on a Cloud instance and be accessible from the workflow management system via a remote procedure call (RPC) mechanism like a web service. Cloud Computing per se does not impose any specific limitations with respect to the usage API while Grid Computing needs a middleware using a particular API that complies to the rules of the virtual organization.

The National Institute of Standards and Technology (NIST) [7] distinguishes the three Cloud service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS providers often focus on standard applications like text processing or customer relation management and will not cover the whole variety of possible tasks. The current existing PaaS offerings only provide standard hardware for general purpose. IaaS is currently the only service model which enables executing programs on specific hardware in the Cloud. Therefore, the rest of the paper will only consider IaaS resources. This should not limit the generality since suitable SaaS or PaaS offerings can be used instead.

NIST [7] also distinguishes four different deployment models: Private Cloud, Community Cloud, Public Cloud, and Hybrid Cloud. Since the example scenario assumes that the specific hardware is not used frequently, a Private Cloud providing such hardware is not feasible. However, the Private Cloud can be used to provide general on-premise resources for the execution of *Local Tasks*. Sharing the specific hardware of a Community Cloud is only possible if such a community exists but this cannot be assumed. Since the paper focuses on outsourcing calculations, renting Public Cloud special resources fulfills all hardware requirements for off-premise calculations. A Hybrid Cloud as combination of a Private Cloud for general on-premise resources with a Public Cloud for special off-premise resources is the required environment for the combination of on- and off-premise calculations.

The rest of paper will only focus on the integration of *Cloud Tasks* that should be executed on IaaS in a Public Cloud.

 ISBN: 978-1-61208-216-5

III. WORKFLOWS WITH CLOUD TASKS

A simple approach to migrate a workflow task to the Cloud is the usage of a service-oriented approach by deploying the task software as web service on the Cloud instance and binding the workflow task to this web service. Web services provide standardized uniform interfaces which supports interoperability of heterogeneous systems. The data to be processed are typically passed as parameter from the workflow task to the assigned web service. An alternative approach for passing larger sets of data is that the web service loads the requested data itself using a onetime access ticket granted by the workflow management system. Independent of the data transfer mechanism, the data should not be stored permanently on the computing Cloud instance because the data are not automatically saved persistently on Cloud images so that a reboot of the resource will result in data loss. On-premise databases or storage Clouds provide permanent, secure, and persistent data storage for the results of the calculation.

Since IaaS resources are frequently provided following a pay-per-time billing structure, any Cloud instance should be terminated after each use to avoid unnecessary costs while the resource is idling. The consequence is that the Cloud instance has to be started again before a re-use is possible. The task execution idles during the bootup of the Cloud instance. Preconfigured machine images contain only the required software to speed up the instantiation. Each abstract Cloud Task could use its own machine image or a basic machine image including all necessary basic systems could be loaded and setup with the task software dynamically on bootup. The required task software is identified using the workflow template. The installation of the software can be done automatically using Secure Shell (SSH).

For a just in time start and termination of the Cloud instance, an automatic mechanism must be available. Otherwise the workflow task idles till the Cloud instance service is available or the Cloud instance service is still available after the workflow task's execution. The Cloud instance start and termination can be included into the workflow template by adding the administrative tasks *Create* and *Destroy* which start and terminate the Cloud instances using a Cloud unification layer or a Cloud agnostic Application Programming Interface (API) like the Open Cloud Computing Interface (OCCI) [8]. The *Cloud Task* is bounded fix to the Cloud instance web service that is only available in the time span between the *Create* and *Destroy* tasks. The usage of automatic template modifications has been already validated in [9].

The concept of the workflow template extension has the benefit of being interoperable with other workflow management systems without individual source code modifications. This makes it even usable for proprietary systems. The same template extension application can be used by different workflow management systems if the same modeling language is supported. Standard workflow modeling languages like XPDL [10] and WS-BPEL [11] benefit most of this approach.

The main disadvantage is that the modeling of workflows becomes more complex because the execution semantic is integrated. Workflows must consider administrative tasks instead of focusing on worker tasks.

Therefore, it is much more comfortable to the user when the administrative tasks are integrated automatically into the template during the workflow instantiation. Because the deployment environment cannot decide where a task should be executed, the usage of task annotations in the template specifies where the task has to be executed. This is similar to MAUI [12] where developers annotate which methods of an application can be offloaded for remote execution.

Figure 2 shows the extended example workflow of Figure 1. The two *Cloud Tasks APURVA* and *Sort* now have administrative predecessor and successor tasks. The so modified workflow is executed instead of the original one. The end user will not notice the difference.

Many users instantiate workflows but not each of them should be able to start arbitrary Cloud resources. Otherwise it would not be possible to map caused costs to individual Cloud usages and an abuse of resources would be possible. Therefore, an authentication service is required on workflow side. This service maps the authentication mechanism of the organization to the authentication mechanism of the Cloud Service Provider. The user privileges can be assigned considering many strategies, e.g., a user could have access only a limited time to a Cloud or she/he could have access only to specific Clouds or for specific workflows. SAML [13] assertions can be used for this. The granularity of user privileges is not in focus of this paper. A standard based security system like WS-Trust [14], Simple Authentication and Security Layer (SASL) [RFC 4422], oAuth [RFC 5849], or OpenID can be integrated into the workflow management system.

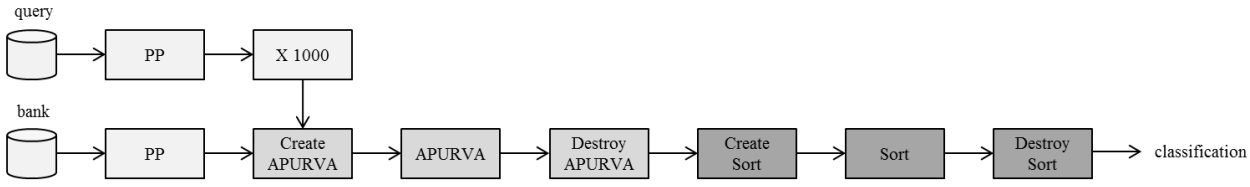

Figure 2. Example workflow extended with administrative Create and Destroy tasks for the two computation intensive tasks *APURVA* and *Sort*.

 ISBN: 978-1-61208-216-5

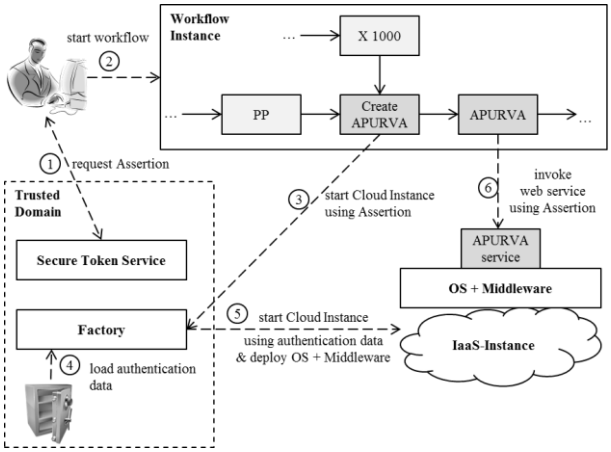

Figure 3. Relationship between workflow instance, Cloud instance, and authentification center.

The process of executing a workflow with *Cloud Tasks* is summarized in the following with reference to Figure 3 where the numbers in circles indicate the order. First the user requests an *assertion token* (1) with only limited use at the *secure token service* by providing her/his own identification together with the identification of all *Cloud Tasks* she/he wants to use. The *secure token service* evaluates the request and decides if the *assertion* can be granted. If the result is positive, the user instantiates the workflow (2). The *Create Task* uses this *assertion* at the factory (3) to proof its eligibility. The *factory* then loads the Cloud account *authentication data* from a secure storage (4) and starts the *Cloud instance* (5) with the deployed *web service*. The *assertion* is now invalidated. The *APURVA Cloud Task* invokes the *web service* (6) that is running on the *Cloud instance*. The *web service* processes the data on the high performance Cloud hardware. After the web service returns its results, the *Destroy Task* shuts down the Cloud instance.

A. Reuse of Web Services

In scenarios like parameter studies, the same workflow task is executed frequently. Other examples of reusing the same task are loops, multiple workflow instances, and different workflows instances using the same *Cloud Task*. The simple approach introduced above starts a new Cloud instance for each Cloud Task instance and terminates the Cloud instance after the web service's execution. The Cloud instance starting overhead slows down the workflow's execution but can be reduced for future invocations by keeping alive the Cloud instance for reusability. A single Cloud web service is then used multiple times by different *Cloud Task* instances of the same abstract *Cloud Task* like APURVA in Figure 4.

The implementation is described in the following: The *Destroy Task* only notifies the *Factory* that the web service is no longer needed by the *Cloud Task*. The integrated scheduler keeps alive the Cloud instance if it expects future web service invocations. Otherwise, the scheduler shuts down the Cloud instance as usual. The prediction is possible by evaluating the assertion requests at the *Secure Token Service*.

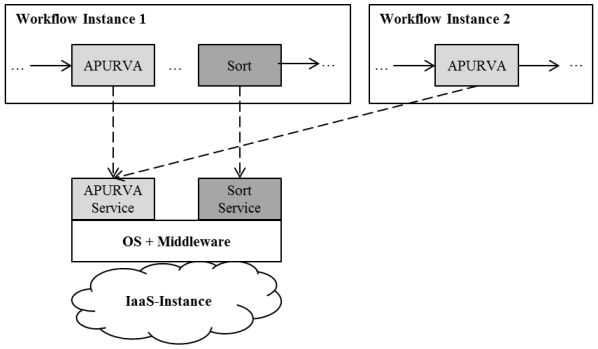

Figure 4. Reuse of Cloud web services and sharing of an IaaS instance.

Listing 1. Shell script to install the web service

```
#/bin/bash
scp -B ~/program.jar user@instance:~/program.jar
ssh user@instance java -jar program.jar parameter
```

B. Multiple Web Services on the same Cloud Instance

To reduce Cloud instance staring overhead and to avoid costs, additional web services can be deployed on the same Cloud instance if they are suitable for the hardware. Figure 4 depicts the *IaaS Instance* that hosts both: *APURVA Service* and *Sort Service*. This optimization is most suitable for workflows with different *Cloud Tasks* that can then be executed in a pipeline on the same Cloud instance. Using this optimization, static machine images cannot be instantiated because additional software must be installed during the uptime of the Cloud instance. The installation can be done using SSH in a shell script like in Listing 1. The first line copies the program via secure copy *scp*. The second line uses *ssh* to start the remote program that will publish its web service as an own endpoint on the Cloud instance by considering the parameter. The password prompt is suppressed using public/private key based authentication.

C. Dynamic Assignment of Tasks to Cloud Resources

The idea of outsourcing only single parts of an application to the Cloud can be extended with a dynamic assignment of the *Cloud Task* to the most suitable Cloud resource at runtime that is illustrated as an example in Figure 5. The selection process is similar to the three-phase cross-cloud federation model described in [15]. In the *discovery* phase, the Cloud Service Broker creates a table in a database which provides information about *Assured Properties* offered by the Cloud Service Providers like in the first three columns of TABLE I. Possible properties are special hardware like general purpose GPUs, best performance, lowest price, performance/price ratio, available volume resources of non-pay-as-you-go contracts, and location of the Cloud for liable reasons or for data nearness as well as data sensitiveness. This table must always be kept up to date. In the workflow template each abstract *Cloud Task* specifies its *Required Properties*. In the example in Figure 5, *APURVA* has the properties *a* and *b* whereas *Sort* has the property *c*. These *Required Properties* are sent to the Cloud Service Broker before the assignment of the *Cloud Task* to its Cloud

resource. Now in the *match-making* phase, the Cloud Service Broker compares the *Cloud Task*'s *Required Properties* with the Cloud Service Providers' *Assured Properties*. The Cloud Service Providers that assure all *Required Properties* of the requesting task are *potential task owners*. The last two columns of TABLE I indicate which resources are the potential owner of which *Cloud Task*. In Figure 5, these potential owners are encircled. In the *authentication* phase, the Cloud Service Broker selects the cheapest *potential owner* as the *current owner* for each *Cloud Task*: Resource 2 for *APURVA* and Resource 3 for *Sort.*

D. Provenance

The importance of validating and reproducing the outcome of computational processes is fundamental to many application domains. It is exposed that there is a need to capture extra information in a process documentation that describes what actually occurred. The automated tracking and storing of provenance information during workflow execution could satisfy this requirement [16]. The amount and the kind of data to be stored are always user and implementation dependent. Provenance traces enable the users to see what has happened during the execution of the workflow. This also 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.

TABLE I. ASSURED PROPERTIES OF CLOUD RESOURCE

Cloud Resource	**Assured Properties**	**Price**	**Potential Owner of**	
			APURVA	*Sort*
Resource 1	a	3	×	×
Resource 2	a, b	4	✓	×
Resource 3	a, b, c	6	✓	✓
Resource 4	b, c	7	×	✓

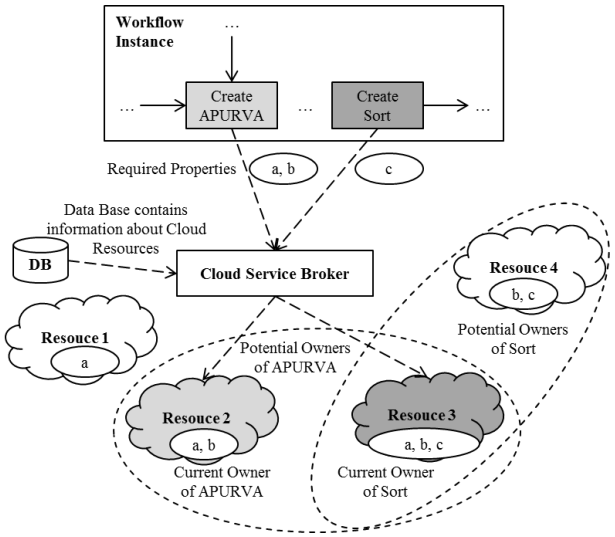

Figure 5. Dynamic assignment of Cloud Tasks to Cloud resources.

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.. Also the runtime of *Cloud Tasks* can be examined in the provenance trace to optimize future *Cloud Task* to Cloud resource assignments.

A provenance model describes how the gathered provenance data are interpreted and stored in the provenance trace. Several provenance models exist and two of them are described briefly in the following. A detailed comparison is done in [17]. The Open Provenance Model (OPM) [18] is very prominent in the e-Science domain. It provides a comprehensive set of concepts to capture how things came out to be in a given state and is designed to achieve inter-operability between various provenance systems. Another provenance model is the so-called History-tracing XML (HisT) [9]. It was developed within the HiX4AGWS project [19] and provides provenance following an approach that directly maps the workflow graph to a layered structure within an XML document. The *Create* and *Destroy* workflow tasks can be used to identify and transmit the provenance data according to the Cloud instances. HisT directly supports the integration of digital signatures and is therefore optimized for the e-Business and cross-organizational domain where responsibility and liability play an important role.

IV. RELATED WORK

Cloud Computing is the greatest IT hype of the last ten years. Therefore, many publications deal with Cloud Computing. Surprisingly the combination of Cloud Computing with workflows is little addressed. The integration of single off-premise *Cloud Tasks* into on-premise workflows is not supported yet. In comparison to the mobile smartphone domain, approaches like CloneCloud [20] already exists to dynamically partition applications between weak devices and Clouds. Some workflow management systems claim to be ready for the Cloud but they are mostly ported from the Grid domain and only support running in the Cloud as extension to running in the Grid. The flexible selection and interaction with Cloud resources is not implemented in the workflow management systems considering the requirements identified in section III. One approach is presented in the following and then delimited to the approach presented in this paper.

The Generic Workflow Execution Service (GWES) [21] is an open source workflow management system and was developed by Frauenhofer-Gesellschaft for the management and the automation of complex workflows in heterogeneous environments. The service orchestration goes through five abstraction levels: *User Request*, *Abstract Workflow*, *Service Candidates*, *Service Instances*, and *Resources*. The formal described *User Request* represents an abstract operation and is automatically composed into an infrastructure independent non-executable *Abstract Workflow*. This *Abstract Workflow* is mapped at runtime down to available *Resources*. During

 ISBN: 978-1-61208-216-5

this process *Service Candidates* web services are searched and optimally selected to become *Service Instances*. 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.

The proposed approach of this paper differs from the basic GWES concept. GWES is a specific workflow management system with an own workflow description language. In contrast the interoperable approach of this paper bases on an extension for existing modeling languages of arbitrary workflow management systems by the integration of the Cloud administrative tasks *Create* and *Destroy* which connect the workflow instance with the Cloud Service Broker to select, start, and stop the Cloud instance. By choosing a workflow management system independent approach the usage of the already known system is given for the end-user. The approach is the migration of only individual workflow tasks to the Cloud whereas the remaining tasks stay in the local environment for execution.

V. CONCLUSION AND FUTURE WORK

This paper presented a general concept for the hybrid execution of workflows by allowing the off-premise execution of specific tasks in the Cloud whereat the remaining tasks stay on-premise to avoid unnecessary costs. The proposed architecture has the advantage that it is neither depending to a particular workflow engine nor to a particular workflow description language. It follows the approach of automatically modifying workflow templates to incorporate the steps for assigning the appropriate off-premise resource in a flexible manner. This approach has been already validated in the domain of provenance [9]. The Cloud Service Broker automatically selects the most suitable Cloud resource to guaranty 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 the implementation of the introduced Cloud Service Broker including an analysis of an according selection metric. The occurred 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. The time overhead for migrating tasks across Cloud and organizational boundaries has to be measured and set it into relation with the avoided costs to figure out if the costs reduction is worth the time overhead. Even data movement strategies have to be implemented.

The security of the whole architecture plays an important role which is minor addressed in this paper. The Secure Token Service and the Factory are together the single point of access. Unauthorized Cloud resource instantiations and unauthorized Cloud web service invocations must be protected against requests without permission to avoid a misuse.

ACKNOWLEDGMENT

This work was carried out in the context of HiX4AGWS [19]. HiX4AGWS is supported of the Federal Ministry of Education and Research in Germany. Grant No.: 17N3409.

References

[1] S. Asano, T. Maruyama, and Y. Yamaguchi; "Performance comparison of FPGA, GPU and CPU in image processing", International Conference on Field Programmable Logic and Applications, 2009. FPL 2009, pp. 126-131.

[2] W. H. Davidow and M. S. Malone; "The virtual Corporation". New York: HarperBusiness, 1992.

[3] E. Deelman, D. Gannon, M. Shields, and I. Taylor: „Workflows and e-science: an overview of workflow system features and capabilities", Future Gener. Comput. Syst., 25 (2009), pp. 528–540.

[4] BIOWIC, Bioinformatics Workflow for Intensive Computation, http://biowic.inria.fr/workflows/shrec.html 05.05.2012.

[5] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", Journal of Grid Computing, Vol. 3, No. 3-4, pp. 171-200, 2005.2. Oxford: Clarendon, 1892, pp.68–73.

[6] 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.

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

[8] Open Grif Forum (OFG), Open Cloud Computing Interface (OCCI), June 2011.

[9] 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), November 2010.

[10] R. M. Shapiro, Workfl ow Management Coalition Working Group One, "XPDL 2.1 Integrating Process Interchange & BPMN", January 2008.

[11] D. Jordan and J. Evdemon, "Web Services Business Process Execution Language Version 2.0 (BPEL)", OASIS Standard, April 2007.

[12] 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.

[13] S. Cantor, J. Kemp, R. Philpott, and E. Maler, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", OASIS Standard, 15 March 2005.

[14] K. Lawrence and C. Kaler, "WS-Trust 1.3 OASIS standard", March 2007.

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

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

[17] 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), November 2011, pp. 57-66.

[18] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. Van den Bussche, "The Open Provenance Model Core Specification (v1.1)," Future Generation Computer Systems, vol.27(6) pp.743-756, June 2011.

[19] History-tracing XML for an Actor-driven Grid-enabled Workflow System, http://www.fh-aachen.de/en/research/projekt-hixforagws/ 05.05.2012

[20] B. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti: "CloneCloud: Elastic Execution between Mobile Device and Cloud", Proceedings of the sixth conference on Computer systems (EuroSys '11), 2011, 301-314.

[21] Generic Workflow Execution Service (GWES) http://www.gridworkflow.org/kwfgrid/gwes/docs/ 05.05.2012.

 ISBN: 978-1-61208-216-5