This is the version, which has been approved for publication. The final version can be accessed at the IEEE Xplore digital library.

IEEE Xplore: http://ieeexplore.ieee.org/document/8397633

DOI: http://dx.doi.org/10.1109/UIC-ATC.2017.8397633

An Application Meta-Model to support the Execution and Benchmarking of Scientific Applications in Multi-Cloud Environments

Markus Ullrich and Jörg Lässig
Department of Computer Science
University of Applied Sciences Zittau/Görlitz
{*mullrich,jlaessig*}*@hszg.de*

Martin Gaedke
Department of Computer Science
Technische Universität Chemnitz
martin.gaedke@informatik.tu-chemnitz.de

Kento Aida and Jingtao Sun
Information Systems Architecture
Science Research Division
National Institute of Informatics
{*aida,sun*}*@nii.ac.jp*

***Abstract*—Cloud computing has proven its importance to scientists around the globe on many occasions already. However, as it is still a relatively new technology for many users, the cloud represents another layer of complexity in any workflow. As a lot of research confirms, especially the efficient provision and management of resources in the cloud is a very complex but also very rewarding task. Upon surveying the research in this area we observed many differences in applied methodologies and application cases which impede not only the comparison of these approaches but also the collective usage of the obtained results, e.g., for more accurate resource estimation algorithms that require less additional benchmarking. We propose a novel application and resource meta-model to model not only applications but also the underlying resource infrastructure for application benchmarks in a generic manner. We show how the meta-model is defined and how it can be used to model an application, using a simple web application as an example. We conclude with highlighting the potential benefits of applying this model in different scenarios but also its limits and how it could be expanded in the future.**

1. Introduction

A variety of complications can be encountered during application deplyoment in cloud environments. Especially in the scientific community when dealing with scientific workflows, HPC or big data applications it becomes more and more vital to efficiently manage and distribute resources for computation and storage requirements of applications with a high resource demand and long execution times. The convenience of the cloud [1] allows for researchers and businesses around the globe to get a head-start when performing new experiments with a high resource demand as there are no upfront cost to pay. In addition, data storage in general is cheap which means data can easily be duplicated to increase its availability, even in different geographic locations. In the same manner, compute resources including deployed applications can easily be duplicated as well as they are also relatively cheap due to the pay-as-you-go concept. However, this convenience can very easily lead to over-provisioning of resources and users paying more on storage and compute resources than necessary. Even if cloud resources seem unlimited, one has to be concerned with their efficient utilization. Furthermore, the initial process of migrating an application to the cloud involves a lot of steps which potential users needs to become familiar with first. The support by a third party or application seems almost necessary to prevent potential delay if the cloud has been selected as execution platform. In the case of scientific experiments which are usually conducted multiple times, e.g., to reduce measurement errors in the process, ideally, the user should not have to be concerned with managing the cloud at all during this process as it adds another level of complexity and thus more room for errors.

This work proposes to support the execution of a variety of applications in an efficient manner in a multi-cloud environment by utilizing a relational application meta-model which can be used to describe an application and its components as well as the underlying resource infrastructure. The resulting model can be used to repeatedly and automatically execute an application using different configurations, e.g., different data sources, virtual or physical machines, load-balancer, etc. This reduces the potential for errors while conducting multiple experiments using the same application over and over. Furthermore, we aim to utilize the resulting model to detect similar applications in terms of resource requirements to predict the requirements of a specific setup based on previous observations.

The rest of this paper is organized as follows. In Section 2 we provide a more detailed formulation of the problem including current obstacles and challenges as well as the major goal of this research. In Section 3 we present our meta-model and highlight its unique properties and advantages but also its limits in the discussed application case. In Section 4 we briefly discuss existing application meta-models and their applications. Finally, Section 5 concludes this work and we discuss the planned future applications of the presented meta-model.

2. Problem Formulation

Cloud computing is a valuable asset to many companies and research institute already. The XSEDE cloud survey report, conducted in 2013 shows that already at that time the cloud is a valuable asset for many research teams around the

globe [2]. Executing high throughput scientific workflows has been named one of the major reasons to use the cloud by the survey participants. In the same survey, the learning curve for utilizing the cloud, the virtual machine performance and data movement cost have been reported as major challenges in this area. Utilizing the cloud for deploying any application adds another layer of complexity to it. Automating this process could simplify this task for end users significantly. However, fully automating this process is a complex task by itself, since different applications have very diverse properties, e.g. different software and infrastructure requirements. We analyzed a large part of the research in this area and identified several application classes based on workload patterns and resource requirements already [3]. Table 1 shows the identified classes which is by no means a finite selection but based on our observations the most commonly used in the literature. We concluded, that many promising resource management solutions for various application cases exist already. However, we also noted that there is a lack of standardized test environments which impedes the comparability of these solutions which mostly have been tested in quite specific scenarios only. Given the amount of cloud providers available, choosing the right provider for a certain job is an additional problem which needs to be solved in order to use the cloud to its maximum efficiency. Gartner has chosen 205 different evaluation criteria alone which could play an important role when selecting an IaaS provider.[1]

It has been shown on many occasions already that modeling an application or resources is indeed useful to predict its behaviour on a certain infrastructure [4]–[11]. As we already did an in-depth analysis of most of these papers in our previous work, we are going to cover these just briefly in section 4. However, one problem with most approaches is, that they are application specific and thus only useful in a certain application case. Furthermore, this makes the comparison of these approaches difficult if not impossible since no standardized evaluation for those models exist. Additionally, most solutions use one specific resource configuration for their evaluation, e.g. one large instance at cloud provider A, which can also be different between approaches, making it even more difficult to compare them with each other. Research that involves benchmarking these instances, also considering multiple cloud providers to select the most appropriate for one type of application, exist, however, these approaches suffer from the same problem, as different evaluation criteria are used, making it difficult to compare them with each other. Therefore, the goal of our research is to provide a meta-model for applications, which considers the underlying resource configuration as well, as a tool for not only modelling but also benchmarking a variety of applications on different cloud platforms to enable a fair comparison between cloud platforms as well as scientific applications and their usefulness in specific scenarios, e.g. genome analysis or certain big data analysis scenarios.

1. https://www.gartner.com/webinar/3266523?srcId=1-7840328974 - accessed: 2017-04-18

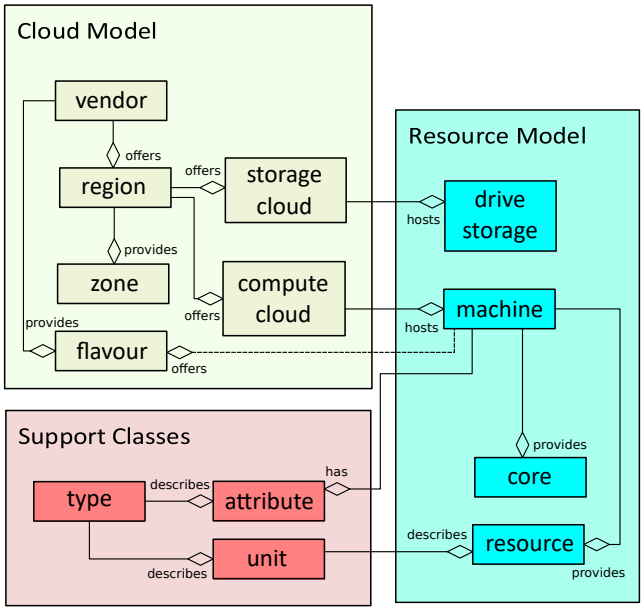

Figure 1. Overview of the proposed meta-model for cloud resources

3. Our Application Meta-Model

As mentioned before, we did an extensive review of existing resource management approaches in the area of cloud computing [3]. We concluded that many promising solutions for various application cases exist already, which are, however, mostly application specific and not comparable to each other due to the lack of standardized test environments. Based on our previous findings, we are proposing a relational meta-model to store information about applications and machines as well as performance benchmarking information about them. 'Machines' is used as a general term to describe not only virtual but also physical machines.

As depicted in figure 1, machines can also be represented as nested machines to cover as many deployment options as possible, e.g., a virtual machine or container on a physical machine or another virtual machine. Potentially, although not intended, a machine could also represent a server rack or even a data center. However, that would not only be confusing in terms of terminology, additionally, data centers can already be represented in our model as a so called compute cloud. It is further possible to model the physical machines that provide the basis for a compute cloud which makes it possible to even model data centers that are deployed completely inside another compute cloud. The model allows basic information about a machine to be stored as a so called flavour which includes number of CPU cores, amount of memory and storage, storage type, f.e., SSD or not, and the network speed as a string, e.g., 'slow' or 'fast', as this is usually the way this information is provided by the cloud service vendor. If more detailed information about machine resources and attributes like location, IP address or name is available it can be stored in two separate tables linked to machine. Beside compute clouds, we decided to allow

TABLE 1. APPLICATION CLASSES IN RESOURCE MANAGEMENT FOR IAAS CLOUDS BY DIFFERENT CATEGORIES [3]

Type of Workload							
Web Server and SaaS			**Scientific Applications**		**Benchmark Applications**		
Communication	File Storage	Processing	Big Data	Learning Algorithms	Micro-Benchmarks	System-Benchmarks	Application Benchmarks
Online Shops	Interactive	DBMS	Workflows				
Scalability							
Multi-Tenancy			Vertical Scalability		Horizontal Scalability		
Favored Resource							
CPU Intensive		Memory Intensive		IO Intensive		Network Intensive	
Deployment Setup							
Single Layer				Multiple Layers			

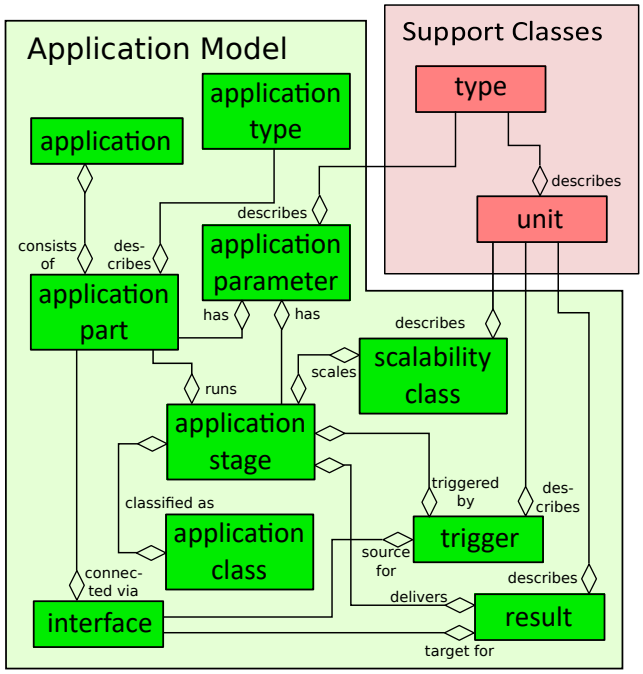

Figure 2. The core of our proposed application meta-model

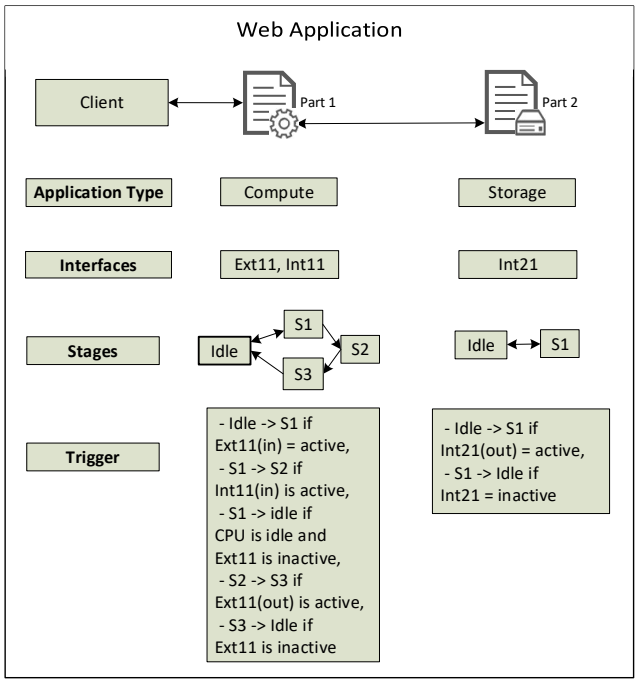

Figure 3. An example for modeling stage transitions for a simple web application that receives a request, fetches data from a data storage and delivers it back to the requester

for storage clouds to be stored separately in our model. A storage cloud refers to any service that can be used to create any means of storage in a cloud environment. Therefore, also block storage devices, which are called 'drivestorage' in our model, can be represented as part of a machine but managed with a storage cloud service. Lastly, CPU cores are also stored separately in this model. Although this information might be redundant with data from the resource table, it enables to link benchmark results to a specific core to determine how many cores an application part is able to utilize which can be important information for scaling decisions.

Figure 2 shows the part of the meta-model for modeling applications. An application in our model consists of at least one application part. Separating an application into parts can be used to represent multi-layered applications that are deployed in a distributed manner, f.e., every application part is deployed on a different machine. Additionally, every application part has a type. This can either be a storage part like a database or a bucket in a storage cloud or a compute part which is true for all other application parts that are not majorly used for storing or providing data. Exchanging data between application parts is modeled through interfaces which can be linked to a trigger or a result of an application stage which is a section during the execution of an application part that has a unique resource consumption pattern. As an example, the simplest application stage imaginable would be an idle stage in which the application part does nothing at all. Using a trigger can further be utilized to model the transition between application stages. Ideally, we would like to make the assumption, that every application part behaves similar to a deterministic finite automaton [12] whereas the application stages represent the states of the automaton and the triggers represent the transition function using system values like network, disk or memory IO as the alphabet.

Figure 3 shows an example for modeling stage transi-

tions in this manner for a simple blocking web application which consists of two parts for demonstration purposes. Part 2 in our example is a storage application part that only has two stages. The first one is the idle stage in which the part does nothing but waiting for requests. Since incoming requests are too short, they are not considered as an additional application stage. The process of sending data from the storage to another part that requested it will be considered however. In the example this stage is called S1. The job of the first application part is solely to forward a request for certain data from a client to the storage part and to send it back to the client. Therefore, it has four stages in total, an idle stage, S1 in which the client request will be received and processed, S2 in which the data from part 2 is received and S3 for sending the data back to the client. Again, the actual request from the client is not actually considered as separate stage but rather processing the request which could be a simple look-up in a local database to check if the client has the appropriate rights to request the data. Stage transitions can be detected automatically based on which of the connecting interfaces become active or inactive. One special case occurs if the client does not have the appropriate access rights. In this case the part goes back to the idle stage and waits for another request which can be detected by the CPU going into an idle state itself. If the application behaves different from the last time it has been executed, e.g., assuming we did not detect beforehand that it is possible for the application part to refuse the request from the client and to go back to the idle stage immediately, all that needs to be changed in the database is the addition of another trigger for the idle stage. In fact, any behavioural changes that are detected at a later point in time will lead to either adding a trigger or adding an application stage and a trigger. Obviously, for any modern web-application, such a behaviour would be far too simple. Modeling such an application with this approach is therefore one potential weakness as it is likely that hundreds of stages will be detected if the same number of clients tries to access the application at once. Consequently, it is important so have the ability to adapt certain thresholds, e.g., the time in which an interface is active or the CPU utilization of the benchmarked application part, for the automatic detection of application stages. For some applications this approach might still be too generic after all which makes using this model in the proposed manner not possible or too inefficient. Another problem that is likely to occur with a generic approach such as this is related to the prediction of stage changes. In case multiple triggers or stage changes are possible at a certain point in time, we currently only know which one will occur if the user provides the expected workflow for an application. In future work it might be possible to estimate stage changes based on previous observations as well, but this is not in the scope of this paper.

However, if the above assumption holds true and it is possible to automatically detect an appropriate number of application stages, we further assume that every stage will have indeed a unique resource consumption pattern which can be identified automatically based on the benchmarking data and assigned to a certain class which further allows the detection of similar application stages based on their resource consumption patterns. As the purpose of this model is to enable a simple and fair benchmarking process for applications on different resource configurations to support the automatic selection of the optimal resources for an application even before executing it, we also have to assure that the runtime and resource consumption of every application stage can be predicted based on previous observations using this model. We propose a novel methodology for this process as follows. The model supports the collection of a variety of not only benchmarking values but also configuration values for every part of the application and every machine the parts are running at. Based on which of these values are available, f.e., have been collected beforehand, and based on the class of the application stage and previous observations with similar stages the most optimal applications/algorithms for predicting the initial resource demand and, if necessary, load balancing are selected beforehand.

A potential problem with this approach is that it heavily relies on benchmarking as most other approaches do. However, the utilization of a meta-model which is designed to consider experimental results and setups from many other research papers in a systematic manner is aiming to solve this problem. This creates a unique way of comparing not only different benchmarking but also resource demand estimation and load balancing approaches for a variety of applications.

Admittedly, this is still a work in progress as some details like the thresholds for automatically detecting application stages have not been clarified yet. However, the meta-model in its current state can and has already been utilized to model a variety of applications. Currently, we are in the progress of conducting a variety of application benchmarks for genomic applications on different setups, e.g., several public and private clouds. The collected data can be utilized for the automatic execution and benchmarking of these applications and further for automatically selecting the most appropriate infrastructure for future experiments. Lastly, we are also developing a web-application to easily access and manage the data about applications, resources and benchmarks to reduce manual input of data as much as possible which in turn reduces potential errors in the data as well.

4. Related Work

We present the most important and recent work in the area of resource and application modeling. Specifically, we take a closer look at research in application meta-modelling.

Resource Modeling. We discussed all the presented papers in this section in our previous work in more detail already. Therefore, we are only going to briefly cover the following approaches.

EMUSIM [4] is a work by Calheiros et al. in which an application model is created through emulating the application behaviour in the cloud which can be used to simulate

the behaviour with a larger number of requests afterwards. This approach shows that modeling can indeed reduce the benchmarking effort for estimating the performance of applications in the cloud. Hajjat et al. [5] developed a model that can be used calculate how different application components affect each other which is a simple way to determine if these can be geographically distributed or not. A rather different approach from Wu et al. [13] is used to model performance fluctuations on a resource level. In future work, this approach can be utilized to predict how the performance of the underlying resource infrastructure for an application varies over time. The approaches by Li et al. [7] and Tak et al. [8] are similar as they both include the creation of a dummy application that simulates the application behaviour and resource consumption in the cloud. These are not particularly modeling approaches but they show that it is possible to benchmark the resource consumption of an application in such detail, that it is even possible to imitate its supposed behaviour on a different infrastructure.

Application Modeling. Kounev et al. [9] presented an approach to model the interactions between the application workload and resource contention at multiple levels in the execution environment. It can be utilized to predict the performance of applications in different scenarios, with different constraints and ultimately to create a self-aware computing system that constantly monitors the state of the applications running on it and creates strategies if the expected future workload might lead to over- or under-provisioning of resources. The approach has been tested in a variety of scenarios with a largely positive result. The model also has some similarities with our proposed meta-model, especially the resource landscape model. However, as this work is mostly concerned with preventing SLA violations and online analysis of benchmarking results, the focus of the model for applications is different from ours, as we are concerned with mostly offline analysis of results and a more generic approach to modeling the actual application behaviour. Furthermore, we specifically want to support the ability to compare applications with each other and potentially utilize benchmarking results from previously observed applications to support a resource management task for a completely new application.

5. Conclusion and Future Work

We proposed a novel application and resource meta-model with the purpose of modeling applications the underlying resource infrastructure as well as associated benchmark information about the whole setup in a generic manner. We showed with a simple example that modeling an application with this approach is very straightforward but it can lead to complications as the concept of automatically identifying stages is too generic at this stage and needs to be refined for such application cases. We also mentioned the theoretical possibility that this meta-model can be used to store results from related work in the area of resource management to potentially enable a fair comparison of approaches in the future. Another benefit of having access to data from all these different sources is that benchmark results for a variety of application classes can also be utilized collectively, potentially resulting in a reduced benchmark effort for future research.

In the future we plan to provide a more detailed description about how exactly the meta-model can be applied to a variety of application cases. We also working towards automating the benchmarking process and conducting further benchmarks with a wider range of applications. Furthermore, we want to integrate detailed information about pricing in public clouds in our model to simplify the calculation of the expected cost for a benchmark or an application execution. Finally, we hope to utilize the collected data to create a proof of concept for a resource and cost demand estimation algorithm that is able to predict the demand for new applications with a low benchmarking effort based on the concepts presented in this paper by utilizing existing benchmarking data from similar applications.

References

[1] P. Mell and T. Grance, *The NIST Definition of Cloud Computing*, NIST Std., Jan 2011.

[2] D. Lifka, I. Foster, S. Mehringer, M. Parashar, P. Redfern, C. Stewart, and S. Tuecke, "Xsede cloud survey report," XSEDE (Cornell Center for Advanced Computing and ANL and The University of Chicago and Rutgers University and Indiana University), Tech. Rep., 9 2013.

[3] M. Ullrich, J. Lssig, and M. Gaedke, "Towards efficient resource management in cloud computing: A survey," in *The IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud 2016)*, 2016.

[4] R. N. Calheiros, M. A. Netto, C. A. De Rose, and R. Buyya, "Emusim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications," *Software: Practice and Experience*, vol. 43, no. 5, pp. 595–612, 2013.

[5] M. Hajjat, S. Pn, A. Sivakumar, and S. Rao, "Measuring and characterizing the performance of interactive multi-tier cloud applications," in *Local and Metropolitan Area Networks (LANMAN), 2015 IEEE International Workshop on*, April 2015, pp. 1–6.

[6] H. Wu, S. Ren, T. Steven, and G. Garzoglio, "A step toward deploying real-time applications on cloud - modeling cloud performance fluctuation," in *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, 2015.

[7] A. Li, X. Zong, S. Kandula, X. Yang, and M. Zhang, "Cloudprophet: towards application performance prediction in cloud," in *ACM SIGCOMM*, vol. 41, Aug 2011, pp. 426–427.

[8] B. C. Tak, C. Tang, H. Huang, and L. Wang, "Pseudoapp: Performance prediction for application migration to cloud," in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*, May 2013, pp. 303–310.

[9] S. Kounev, N. Huber, F. Brosig, and X. Zhu, "A model-based approach to designing self-aware it systems and infrastructures," *Computer*, vol. 49, no. 7, pp. 53–61, July 2016.

[10] R. Coutinho, Y. Frota, K. Ocaña, D. de Oliveira, and L. M. A. Drummond, "A dynamic cloud dimensioning approach for parallel scientific workflows: a case study in the comparative genomics domain," *Journal of Grid Computing*, vol. 14, no. 3, pp. 443–461, 2016. [Online]. Available: http://dx.doi.org/10.1007/s10723-016-9367-x

[11] P. Mata, A. H. Baarah, C. Kuziemsky, and L. Peyton, "An application meta-model for community care," *Procedia Computer Science*, vol. 37, pp. 465 – 472, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050914010357

[12] S.-Y. Kuroda, "Classes of languages and linear-bounded automata," *Information and Control*, vol. 7, no. 2, pp. 207 – 223, 1964. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0019995864901202

[13] J. Wu, Q. Liu, and X. Liao, "A secure and efficient outsourceable group key transfer protocol in cloud computing," in *Proceedings of the 2Nd International Workshop on Security in Cloud Computing*, ser. SCC '14. New York, NY, USA: ACM, 2014, pp. 43–50.