Bridging the Gap Between HPC and Cloud Using HyperFlow and PaaSage

Dennis Hoppe[1(✉)], Yosandra Sandoval[1], Anthony Sulistio[1], Maciej Malawski[2], Bartosz Balis[2], Maciej Pawlik[2], Kamil Figiela[2], Dariusz Krol[2], Michal Orzechowski[2], Jacek Kitowski[2], and Marian Bubak[2]

[1] High Performance Computing Center Stuttgart (HLRS), Stuttgart, Germany
{hoppe,sandoval,sulistio}@hlrs.de
[2] AGH University of Science and Technology, Krakow, Poland
{malawski,balis,kfigiela,dkrol,morzech,kito,bubak}@agh.edu.pl

Abstract. A hybrid HPC/Cloud architecture is a potential solution to the ever-increasing demand for high-availability on-demand resources for eScience applications. eScience applications are primarily compute-intensive, and thus require HPC resources. They usually also include pre- and post-processing steps, which can be moved into the Cloud in order to keep costs low. We believe that currently no methodology exists to bridge the gap between HPC and Cloud in a seamless manner. The goal is to lower the gap for non-professionals in order to exploit external facilities through an automated deployment and scaling both vertically (HPC) and horizontally (Cloud). This paper demonstrates how representative eScience applications can easily be transferred from HPC to Cloud using the model-based cross-cloud deployment platform PaaSage.

Keywords: Cloud computing · HPC · eScience
Workflow management

1 Introduction

Solving computationally-intensive science problems (e.g., climate models) benefits significantly of distributed computing infrastructures such as Clouds and HPC. Cloud infrastructures are first choice when compute resources are needed immediately and temporarily, and when scientific applications require only minimal network communication and I/O. This is often true for pre- and post-processing tasks such as the visualization of a climate model over time. However, the main task of an eScience application is very compute-intensive, and thus requires HPC to be solved within a relatively short period of time. The German meteorological service (Deutscher Wetterdienst), for example, is running more than 16,000 jobs per day in order to predict critical weather conditions within near real-time.

Because HPC has higher costs, as compared to Clouds, the HPC community would benefit from a combination of the strength of the two environments:

R. Wyrzykowski et al. (Eds.): PPAM 2017, LNCS 10777, pp. 432–442, 2018.
https://doi.org/10.1007/978-3-319-78024-5_38

Compute-intensive tasks run on HPC, and pre- and post-processing tasks are delegated to the Cloud. We see a trend to combine HPC with Cloud in order to cope with the ever-increasing demand for high availability, on-demand resources. Still, developing and monitoring large-scale, complex eScience applications on Cloud and HPC is a challenge for researchers, and end users. Therefore, we claim that current HPC centers must aim to lower the barrier to allow users to exploit their infrastructures in order to scale their applications both vertically (HPC) and horizontally (Cloud). Still, porting eScience applications to different infrastructures is time consuming and error prone, leaving users with applications adapted to very specific platforms. Moreover, since the resource requirements depend on application characteristics, no general strategy exists for resource allocation in a hybrid HPC/Cloud scenario.

In this paper, we demonstrate that existing scientific applications—optimized for HPC—can be seamlessly deployed across multiple Cloud platforms without any further changes to the original source code. Section 2 surveys current cross-cloud deployment platforms and elaborates on selecting PaaSage as the tool of choice. Section 3 lists the challenges and barriers that need to be tackled when bridging the gap between HPC and Cloud; the section concludes with a sketch of the final architecture incorporating HyperFlow, setting up an MPI cluster with auto-scaling in the Cloud, and PaaSage. Section 4 introduces two case studies. Firstly, a molecular dynamics simulation, which is a highly-representative eScience application using OpenMP and MPI; and secondly, a data farming experiment using Scalarm. Section 5 presents experimental results obtained by deploying both the MD simulation and Scalarm using PaaSage into the Cloud. Finally, Sect. 6 concludes with a brief summary of the paper.

2 Related Work

eScience applications are often workflow-based, meaning they are composed of multiple tasks. In order to move existing applications from HPC to Cloud, two prerequisites must be satisfied: a workflow execution engine in order to model and execute a scientific application, and a cross-cloud orchestration tool to deploy the workflow on arbitrary clouds. Both tools should be combined in order to allow for a complete solution to deploy scientific applications on multi-cloud platforms.

Existing workflow execution engines include ASKALON [7], Apache Taverna[1], and HyperFlow [1]. These workflow managements systems allow users to model a workflow either graphical (ASKALON, Taverna) or by using a scripting language such as JavaScript (HyperFlow). As a consequence, additional work is required to execute an existing C/C++ or FORTRAN application. A disadvantage of ASKALON and Apache Taverna is, as compared to HyperFlow, that both ask users to write additional components or services in Java, which is uncommon in the HPC community. Here, HyperFlow excels by imposing minimal overhead, and enables users to invoke existing executables through its generic executor

[1] https://taverna.incubator.apache.org/.

interface. As a result, executing a molecular dynamics simulation in HyperFlow is broken down into modeling three processes: parametrization, actual simulation, and visualization. All three processes are able to invoke simple bash scripts to trigger the execution. Although ASKALON and Apache Taverna allow to deploy workflows into the Cloud, they are limited to Amazon Web Services (AWS). HyperFlow supports by default only Amazon S3 as a remote storage.

Next to a workflow execution engine, a vendor lock-in with respect to Clouds has be avoided. As a consequence, an cross-cloud orchestration tool such as Apache Stratos[2], Cloudify[3], or CLOUDIATOR [3] is required. All three tools use Apache jclouds in order to support a vast range of public and private Cloud providers and platforms. Both Apache Stratos and Cloudify expect to provide a provider-specific description of required resources, so that users cannot seamlessly switch between Cloud platforms. Instead, CLOUDIATOR uses an underlying API named SWORD to unify access for different Cloud platforms.

Finally, a complete Cloud deployment solution is required that provides both scientific workflow execution, as well as deployment to multiple Clouds. Existing solutions are, e.g., CometCloud [4] and PaaSage [2]. CometCloud is an autonomic engine for hybrid grids and Clouds that supports execution of scientific workflows, bag-of-tasks applications and MapReduce tasks. However, it does not provide functionality for performing data farming experiments, and it does not support cross-cloud deployment. PaaSage, on the other hand, provides a seamless multi- and cross-cloud deployment using CLOUDIATOR, and by the integration of HyperFlow, it will also allow to move existing scientific applications into Cloud without any further source code modifications.

3 Bridging the Gap Between HPC and Cloud

Recently, there has been a growing interest from both eScience and HPC communities to exploit Cloud, as they seem to offer just the capabilities required by the researchers because of its well-known advantages: (i) strong computing resources (scalability), (ii) on-demand resources (elasticity), (iii) high availability, (iv) high reliability, (v) large data scope, (vi) reduced capital expenditure (cheap). We introduce PaaSage, a Cloud management platform, which supports both the design and deployment of applications across multiple infrastructures. While using the PaaSage platform, we are—for the first time—able to deploy eScience applications across multiple platforms (e.g., public clouds for researchers, and private Clouds for industry) without code modifications, and also to seamlessly delegate post-processing tasks such as visualization from HPC to the Cloud.

3.1 PaaSage

The PaaSage platform provides the model-based development, configuration, monitoring, and optimisation of applications at run-time. Furthermore, PaaSage

[2] http://stratos.apache.org/.
[3] http://cloudify.co/.

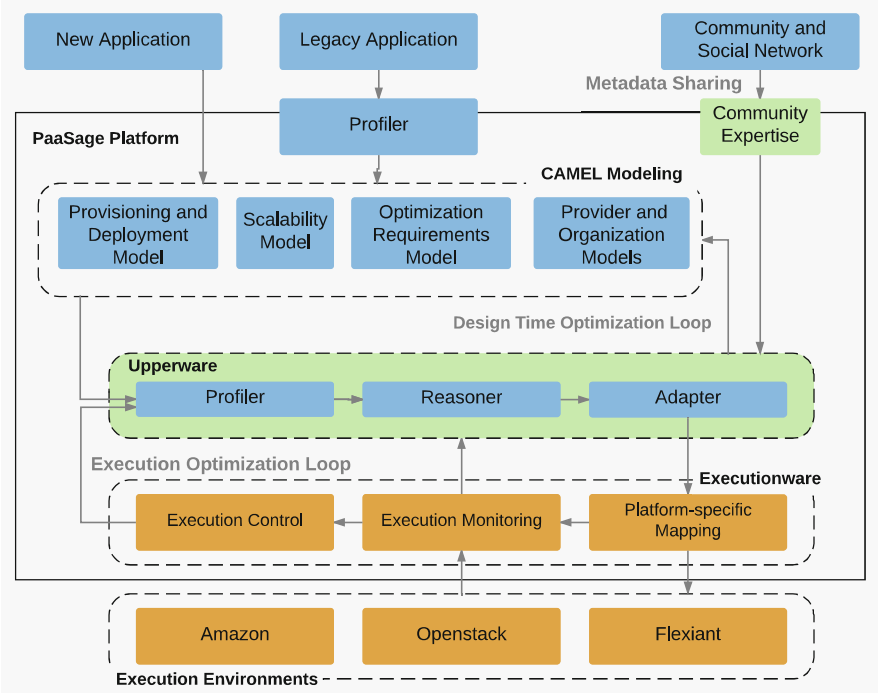

Fig. 1. PaaSage architecture and workflow: metadata workflow (green), Upperware (blue), and Executionware (orange). (Color figure online)

supports single-, multi-, as well as the cross-cloud deployment of existing and new applications. PaaSage's architecture is geared towards its "develop once, deploy everywhere" paradigm (cf. Fig. 1). Users are encouraged to represent (existing) applications using a newly developed Cloud modeling-language named CAMEL—Cloud Application Modeling and Execution Language [8]. PaaSage also provides an Eclipse-based editor for creating application models. CAMEL models include not only required components of an application but also various user requirements such as (i) a set of preferred Cloud providers for deployment, (ii) hardware requirements, (iii) auto-scaling options, and (iv) optimisation criteria (e.g., response time of a Web service below a given threshold).

The CAMEL model is then passed to the so-called Upperware, where application and user requirements are mapped against a metadata database to identify potential Cloud providers that satisfy all requirements. The Upperware returns an initial feasible deployment solution, which is passed to the next component—Executionware. The Executionware provides a unified interface to multiple Cloud providers and thus can handle platform-specific mappings and different Cloud provider architectures and APIs. The purpose of the Executionware is to monitor, re-configure, and optimize running applications. Monitoring data is passed continuously to the Upperware. If the Upperware should find a better deployment solution at run-time, a re-deployment can automatically be triggered.

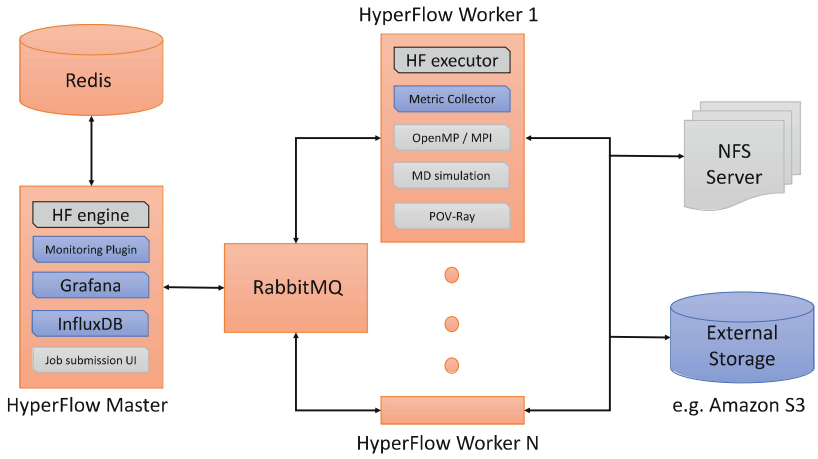

Fig. 2. CAMEL deployment model for the HyperFlow use case.

3.2 HyperFlow Distributed Workflow Engine

HyperFlow is a lightweight workflow execution engine implemented in Node.js. It enables execution of workflow tasks in the form of commands (invoking external executables), or arbitrary JavaScript functions. Figure 2 presents the overall architecture of the molecular dynamics (MD) demonstrator using HyperFlow and PaaSage, including: (1) Database service implemented using Redis; (2) Master node on which the HyperFlow workflow engine is installed. The node offers a Web front-end (UI) for users to configure and submit new simulations. For monitoring the progress of the workflow execution, we include the InfluxDB and the Grafana dashboard for visualisation of the metrics; (3) Worker nodes that are pre-configured with the MD simulation tool and the HyperFlow job executor; (4) Message broker service implemented using RabbitMQ to establish communication between the master and worker nodes; (5) NFS Server for intermediate data exchange between workers; (6) Permanent storage service that provides, for example, an interface to Amazon S3 to store simulation results.

3.3 Scalarm Data Farming Platform

Scalarm [5] stands for Massively Self-Scalable Platform for Data Farming. It is a complete multi-tenancy platform for data farming, which implements all phases of the data farming process, starting from experiment definition through simulation execution to result analysis. The architecture of Scalarm, as depicted in Fig. 3, follows the master-worker design pattern, where the master part is responsible for coordinating data farming experiments, while the worker part handles actual application execution. The four core services include: (1) Experiment Manager: handles all interaction between the platform and end users via a graphical user interface, acting as a gateway for analysts, provides a complete

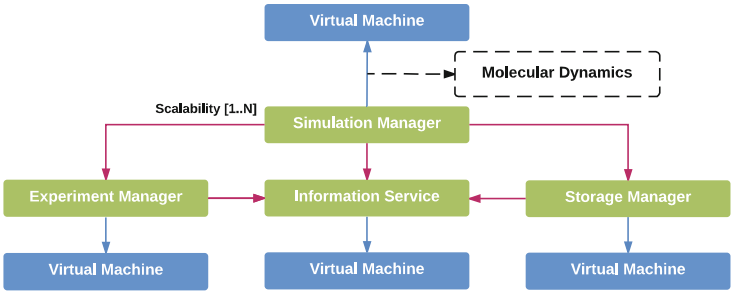

Fig. 3. CAMEL deployment model for the Scalarm use case.

view of running and completed data farming experiments and acts as a scheduler for simulations assigned to Simulation Managers. (2) Storage Manager: implements the concept of the persistence layer. Other components use this service to store information on executed experiments and results of simulations. (3) Information Service: is a place, where each component can find information about other components locations. (4) Simulation Manager: is a worker node that acts as a wrapper for actual simulations. It implements the concept of the pilot job and supports multiple types of computational infrastructures. The first three services constitute the master part and the Simulation Manager is the worker part, remaining Data Explorer provides extra data analytics capabilities.

4 Case Studies

Molecular dynamics simulation (MD) are representative for eScience; it is both compute- and communication-intensive exploiting parallel programming methodologies including MPI and OpenMP. We introduce two ways of executing an HPC-based MD simulation in the Cloud without any source code modifications using PaaSage in combination with HyperFlow and Scalarm. Firstly, this section demonstrates the general capability to deploy an eScience application in the Cloud using HyperFlow, and secondly, Scalarm will showcase the parameter sweeping aspect of MD simulations. Before highlighting the achievements, more details on the MD simulation are given. As a demonstration for common MD, we perform a water droplet simulation, where a water droplet drips into a basin of water. The simulation then predicts the movement of water molecules under a given set of parameters; parameters include, for instance, the length of the simulation, the ambient temperature, and the density of the molecules. In the case of the parameter sweeping experiment, the list of possible parameters is significantly extended. Result data is then visualized and stored as a video showing the movement of molecules using PovRay.

HyperFlow. HyperFlow has been integrated with PaaSage, enabling a whole class of workflow applications to be automatically deployed and scaled in the multi-cloud infrastructure. Adaptation of the HyperFlow engine to use with PaaSage required defining of CAMEL model, preparation of deployment scripts, and implementation of monitoring plugins. The CAMEL model in addition to the definition of components and their requirements includes also a definition of actions for life-cycle events such as installation, connection or auto-scaling. These actions are implemented in scripts using Chef of Bash. The monitoring plugins collect the data from the workflow engine and report such metrics as task execution time or queue length to the Executionware sensors of PaaSage.

The basic scenario is as follows: A user accesses the Web front-end, sets desired input parameters, and then submits a new job. The job gets added to the job queue of the HyperFlow workflow engine operated by RabbitMQ. Since worker nodes are registered in the job queue, a worker fetches a new job request from the queue and starts processing immediately. Next, an MD simulation with the given parameters is started on that particular worker node. During execution, PaaSage features such as automatic scaling ensure that the task is executed accordingly, for instance, additional worker nodes are added to execute the simulation if the CPU load is above a predefined threshold. Results of the simulations are presented to the user through the Web front-end.

Scalarm. Since Data Farming experiments rely on the ability to conduct massive computations, today's cloud systems seem to be a perfect solution. By integrating Scalarm with PaaSage, the whole Scalarm installation can be deployed and scaled accordingly in the multi-cloud infrastructure, what gives it a great advantage in terms of flexibility, infrastructure features selection and ability to minimize the cost of running data farming experiments.

We created a CAMEL model to use PaaSage that describes how Scalarm services depend on each other and how the worker nodes (Simulation Managers) can be scaled depending on the specific metrics. In particular, our CAMEL model defines four goals, which PaaSage optimization loops try to fulfill: (1) minimization of a response time of *InformationService*, (2) maximization of an availability of *StorageManager*, (3) minimization of performance degradation of *ExperimentManager*, and (4) minimization of the overall cost of the experiment.

In order to use Scalarm for running a sample scientific application like MD, a user has to prepare a data farming experiment configuration that consists of (i) an application wrapper that passes arguments to an application, (ii) a specification of a vector of input parameters. An extra, optional parameter: optimization goal was added as a result of integration with PaaSage platform. That parameter is directly related to the flexibility given by the multi-cloud solution in terms of performance, availability and managing the cost of computation. Possible goals include: maximizing performance, minimizing cost or finishing experiment by a given deadline while minimizing costs.

We consider the process of configuration of a scientific application as data farming experiment with Scalarm to be relatively simple. Thus using the Scalarm

platform as a gateway for scientific application to the world of multi-cloud computations seem to be an attractive solution.

5 Performance Evaluation of PaaSage Platform

In order to test our use cases on different clouds, we performed the following experiment. Performance variables considered: (1) total deployment time (DT) of Passage and application on each of the architecture components; (2) total time it takes for the application to run the simulation (AET). As shown in Fig. 4, the factors influencing the performance variables are: (i) on what Cloud infrastructure is the application executed? (Amazon (`A`), Omistack (`O`)[4], or Cross-Cloud (`C`)); (ii) on what Cloud is the PaaSage platform executed? (Amazon or Omistack); leading to the six possible combinations of these variables. The experiments were running at 8 VMs on the different Cloud providers for the application's components. The configuration of Omistack Cloud included VMs of `m1.medium` size, and on Amazon EC2 we used `m3.medium` and `m3.xlarge` instance types[5]. All VMs were running Ubuntu 14.04 LTS Linux distribution. The PaaSage platform was running on an external VM located at Omistack, EC2 or PL-Grid (in the Scalarm case) Cloud. Regarding the MD simulation, a single conducted experiment contains 100 simulation runs, each with a different input parameter values. This number was dictated by the experiment parameter space, which in every case includes 85 different values of the temperature parameter and 5 values of the simulation time step parameter.

HyperFlow. The results of experiment runs for HyperFlow are presented in Fig. 4. As we can see, the deployment times are in the order of several minutes, and it includes the instantiation of the empty VM, installation of generic services of PaaSage Executionware, installation and configuration of application-specific software component, and start-up of all these services. The VM creation is mostly sequential due to service interdependencies (e.g., Master node requires Redis DB), but the deployment is done in parallel.

As results we can observe that the experiments performed with the Omistack Cloud take much more time than the experiments performed with the Amazon Cloud. On the other hand, since the worker nodes are always executed on Amazon, we got better execution time in the cases in which Amazon was involved. Finally, the execution time on Cross-Clouds was the smallest because the assigned VMs were of size m3.xlarge for the worker nodes executed on Amazon.

[4] OmiStack is a private Cloud provided by the University of Ulm based on OpenStack. OpenStack is a leading software to manage Clouds.

[5] It should be noted that VM sizes were selected due to cost constraints and availability. `m1.medium` has 2 vCPUs, 40 GB disk, and 4 GB RAM; `m3.medium` has 1 vCPU, 4 GB SSD, and 3.75 GB RAM; `m3.xlarge` has 4 vCPUs, 2×40 SSD, and 15 GB RAM. `m3.xlarge` was unavailable on Omistack.

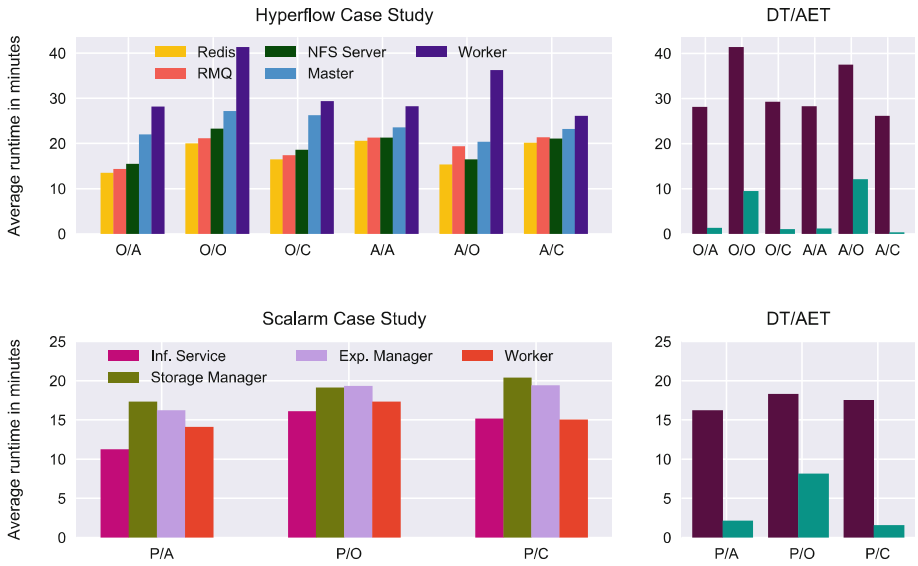

Fig. 4. Evaluation results for the HyperFlow and Scalarm case studies.

We consider the measured deployment times as satisfactory, taking into account that the job waiting time in HPC centers is usually much longer [6]. Moreover, these results demonstrate that there is no significant overhead of cross-cloud deployment versus a single cloud one.

Scalarm. Figure 4 presents results of deploying Scalarm using PaaSage with regard to multiple cloud providers. Each time measurement includes provisioning of an empty VM, installation of generic services of PaaSage Executionware, installation, configuration and startup of a service. As in Hyperflow, the times per component vary slightly depending on the cloud provider. The service is considered deployed and started when it becomes responsive when connecting to specific ports of a particular VM. Respective times are smaller than in the case of Hyperflow as Scalarm services are lighter, what results in a shorter startup time.

6 Conclusions and Future Work

Today's HPC centers are in need to lower the barrier for existing and new customers to fully leverage the potential of HPC and Cloud resources by providing methods and tools to seamlessly execute applications on different target systems without the need for adapting them. In this paper, we have presented PaaSage in combination with HyperFlow and Scalarm as tools for such a seamless transition between multiple target systems. It has been shown that this solution has three key advantages over current commercial offerings. Firstly, PaaSage

and HyperFlow allow to deploy arbitrary eScience applications across different platforms without the need of modification. Secondly, the solution significantly lowers the hurdle for end users to specify hardware requirements for deployment. Before PaaSage, users had to define the required hardware resources explicitly in advance, while PaaSage automatically selects the best resource combinations and also initiates automatic scaling if needed. Thirdly, users were previously restricted to wait for HPC resources becoming available. With PaaSage, tasks can be easily delegated to the Cloud, enabling users to run their jobs immediately without having to wait in a long queue for HPC resources. Non-critical tasks such as post-processing can be easily migrated to the Cloud in order to save expensive HPC resource usage.

Although it has been shown that a deployment of scientific applications requires less effort than competitive solutions, the actual deployment times of an application are currently quite long, as compared to the actual runtime of the simulation at hand. It should be noted that the focus of the current work is on the deployment itself, and the simulations are configured to have short execution times. The runtime of simulation will de facto exceed the deployment times by several orders of magnitude. Still, future work will address to reduce deployment times by reusing pre-built containers by leveraging technologies such as Docker.

Acknowledgements. We thankfully acknowledge the support of the EU 7th Framework Programme (FP7/2013-2016) under grant agreement number 317715. Access to Omistack Cloud resources was kindly provided by University of Ulm, Germany. HyperFlow and Scalarm are partially supported by the AGH Statutory Fund.

References

1. Balis, B.: HyperFlow: a model of computation, programming approach and enactment engine for complex distributed workflows. Future Gener. Comput. Syst. **55**, 147–162 (2016). https://doi.org/10.1016/j.future.2015.08.015
2. Balis, B., Figiela, K., Malawski, M., Pawlik, M., Bubak, M.: A lightweight approach for deployment of scientific workflows in cloud infrastructures. In: Wyrzykowski, R., Deelman, E., Dongarra, J., Karczewski, K., Kitowski, J., Wiatr, K. (eds.) PPAM 2015. LNCS, vol. 9573, pp. 281–290. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-32149-3_27
3. Baur, D., Domaschka, J.: Experiences from building a cross-cloud orchestration tool. In: Proceedings of the 3rd Workshop on CrossCloud Infrastructures and Platforms, pp. 4:1–4:6. ACM (2016). https://doi.org/10.1145/2904111.2904116
4. Kim, H., el Khamra, Y., Jha, S., Parashar, M.: Exploring application and infrastructure adaptation on hybrid grid-cloud infrastructure. In: Proceedings 19th ACM International Symposium on High Performance Distributed Computing, HPDC 2010, pp. 402–412. ACM (2010). https://doi.org/10.1145/1851476.1851536
5. Krol, D., Kitowski, J.: Self-scalable services in service oriented software for cost-effective data farming. Future Gener. Comput. Syst. **54**(C), 1–15 (2016). https://doi.org/10.1016/j.future.2015.07.003

6. Marathe, A., Harris, R., Lowenthal, D.K., de Supinski, B.R., Rountree, B., Schulz, M., Yuan, X.: A comparative study of high-performance computing on the cloud. In: Proceedings of the 22nd International Symposium on High-Performance Parallel and distributed Computing, pp. 239–250. ACM (2013). https://doi.org/10.1145/2462902.2462919
7. Qin, J., Fahringer, T.: Scientific Workflows - Programming, Optimization, and Synthesis with ASKALON and AWDL. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30715-7
8. Rossini, A.: Cloud application modelling and execution language (CAMEL) and the PaaSage workflow. In: Advances in Service-Oriented and Cloud Computing—Workshops of ESOCC, vol. 567, pp. 437–439, September 2015. https://doi.org/10.1007/978-3-319-33313-7