Science Automation in Practice: Performance Data Farming in Workflows

Dariusz Król*, Jacek Kitowski*, Rafael Ferreira da Silva§, Gideon Juve§, Karan Vahi§, Mats Rynge§, Ewa Deelman§
* AGH University of Science and Technology, Department of Computer Science, Krakow, Poland
§ University of Southern California, Information Sciences Institute, Marina Del Rey, CA, USA
Email: {dkrol, kito}@agh.edu.pl, {rafsilva,gideon,vahi,rynge,deelman}@isi.edu

***Abstract*—This paper describes an approach to conduct large-scale parameter studies, where each data point in the study requires the execution of a whole scientific workflow. We show how a parameter studies system can be integrated with a workflow management system to seamlessly execute a large number of workflows, each with different input parameter values using large-scale computing infrastructure. The work is motivated by a need to collect performance-related data to conduct a sensitivity analysis in the context of relation between workflow input parameters and the performance of tasks in the workflow developed for the Spallation Neutron Source facility at the Oak Ridge National Laboratory.**

I. Introduction

Parameter studies in the form of a collection of independent tasks is the most common approach to study various phenomena in computational science [1]. Each task receives a configuration of input parameter values (representing environmental conditions for the phenomena), and returns a response, which constitutes a single data point in the possible output landscape. Despite their simplicity, parameter studies are generic enough to be applied in many simulation-centric fields of science such as fluid dynamics [2] and particle physics [3]. Researchers often need to explore a large number of input parameter values to comprehensively understand the output landscape of some natural phenomena due to their intrinsic complexity.

On the other hand, scientific workflows are an essential way to organize large-scale computations in the form of a graph with vertices representing computational tasks and edges representing data dependencies between the tasks. Examples of large-scale scientific workflows include: the development of a comprehensive understanding of earthquakes in Southern California and elsewhere with Probabilistic Seismic Hazard Analysis [4]; and detecting and measuring gravitational waves predicted by Einstein's theory of general relativity [5]. Other examples cover data-intensive computing [6] or P systems [7].

The practical application of these two approaches in real-life use cases is determined by the availability of software tools. Typically, the parameter studies are incorporated directly into workflows in the form of *split* constructions, resulting rather in monolithic complex workflow applications. It may lead to unnecessary tightly coupling of tasks in such a large study, preventing them from being executed in a more distributed way across multiple computing sites. To mitigate this problem, we present a novel approach by decoupling the workflow management from the parametric studies to improve large-scale scientific computing in distributed environments. Both functionalities are currently supported by independent developed systems, integrated for achieving the required functionality of parametric studies. More specifically, we use two in-house developed systems: the Pegasus Workflow Management System [8], as a state-of-the-art solution for workflows; and the Scalarm platform [9] for parametric studies, as a data farming service for the PL-Grid infrastructure [10]. Both systems are widely used by the research community in several different domains: Pegasus has a rich record of applications [11], and Scalarm has been used in a number of applications including the study of metallurgical processes [12], and as a use case of cross-cloud applications within the EU FP7 PaaSage project [13], [14] that aims at creating a deployment platform for cloud-oriented applications. The proposed approach enables the deployment of an environment for conducting automated data-driven parameter studies across heterogeneous computing sites, where each data point requires the execution of an entire workflow. Although the presented approach can only be applied to a subclass of workflows, which can be decomposed into independent pipelines, this subclass still represent the majority of the workflows used in practice.

II. Problem statement

Let's assume we have a scientific workflow, which takes several input parameters, and a set of computational sites capable of executing the workflow. Our goal is to run the workflow multiple times for different sets of input parameters, and to collect results from all executions. Then our problem is to specify distinct configurations of input parameter values that we want to explore; execute the workflow for each defined configuration seamlessly and independently on the distributed resources; and collect results from all executions into a single (or centralized) location. Additionally, the scientific code should not require any further development. All these steps should be done with minimal programming effort and maximal automation to enable scientists without in-depth knowledge about distributed computing to conduct such studies.

III. Proposed solution

Typically, since the simulation run for each parameter value is a simple pipeline containing several steps, the simplest approach is to add all the pipelines for all the parameter values

to a single, large workflow. This could be easily accomplished if all of the parameter values are known a priori. If the values are generated dynamically based on the results of previous simulations, then the workflow, which incorporates management of input parameters, may become too complex. In addition, even if the values are known a priori, for very large parameter studies the resulting workflow might contain millions of tasks, which would be difficult to manage the execution of each study independently. Although there are a few approaches to tackle this problem by using workflows solely (e.g. workflow ensembles) [15], the complexity may significantly impact the usability of the solution.

An alternative way to model the problem is to move the parameter study outside the workflow. In this approach, the workflow description is much simpler and is intended to evaluate only a single set of parameters. Different configurations can be run and managed independently since there are no dependencies between them. The specification of the input parameter values and the management of workflow executions are delegated to a parameter study system. This clear decoupling of the experiment definition from the workflow description also fosters workflow reuse. The proposed solution is based on the integration of two science automation tools: Pegasus for handling workflow orchestration, and the Scalarm platform for parameter studies organization.

Pegasus enables workflow applications to execute on distributed computing infrastructures by mapping abstract workflow descriptions onto distributed resources. It enables scientists to construct high-level descriptions of their workflows without worrying about the details of the underlying execution environment. These workflows are represented as directed acyclic graphs, where nodes represent jobs and edges represent data dependencies. Pegasus locates the necessary input data and computing resources required for workflow execution, and generates an executable description of the workflow. Scalarm is a tool supporting input parameter space specification, execution, and collecting results in parameter studies. Scalarm integrates with applications thorugh adapters that are executed in different phases of a parameter study. Adapters are executable scripts created by the application developer, and are executed by Scalarm to exchange data with the application as shown in Fig. 1.

Scalarm executes the user application for each point in the input parameter space. For each execution, Scalarm creates a file (*input.json*) using JSON format that contains input parameter values representing a single point in the parameter space. The `input_writer` adapter is responsible for converting data from the file to the required application-specific format. The `executor` adapter then starts the application and waits for its completion. Application results are handled by the `output_reader` adapter, which collects information in a Scalarm-defined format. Additionally, Scalarm handles intermediate results from the application using another adapter called `progress_monitor`, which is executed periodically to check the current status of the running application.

To integrate Pegasus and Scalarm, we implemented

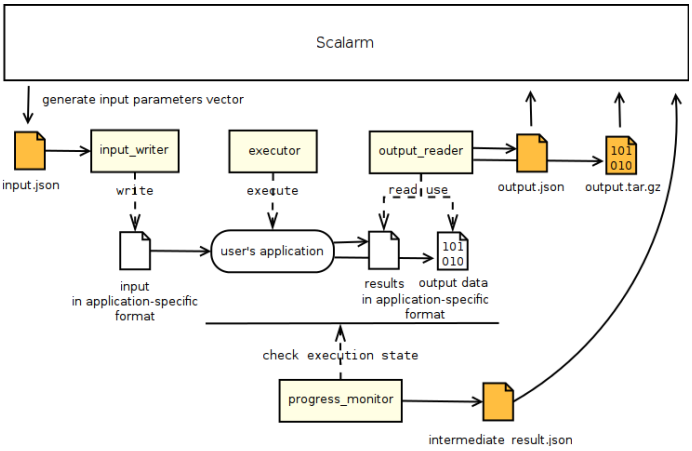

Fig. 1: An integration pattern of a generic application and the Scalarm platform.

Scalarm adapters using the following Pegasus commands: (1) `pegasus-plan`—to plan and schedule a workflow with prepared input files; (2) `pegasus-status`—to check the current status of the workflow (e.g., information about running and completed tasks, error occurrences, etc.); and (3) `pegasus-statistics`—to collect information about tasks statistics (e.g., execution and queue time, etc.).

IV. Experimental evaluation

The integrated system was evaluated by running a workflow developed for the Spallation Neutron Source (SNS) facility at Oak Ridge National Laboratory. The workflow executes an ensemble of molecular dynamics and neutron scattering intensity calculations to optimize a model parameter value, which are then compared with data from experiments such as QENS [16]. This is a typical parameter study case, where a simulation takes a few input parameters and the goal is twofold: to validate simulation model by comparing simulation results with physical experiments results and to provide a sensitivity analysis in the context of relation between input parameters used in simulation and the execution performance of each task in the workflow.

Definition of the input parameter space. The input parameters of the SNS workflow include: type of atomistic structure of the material, the number of CPU cores used for computation, the number of timesteps in simulations, and the frequency at which the output data is written. For demonstration purposes, we used a factorial design, called 2^k, to investigate two levels of each factor, e.g. for an application with two input parameters (A, B), A_{low} represents the lower level of A, and A_{high} represents the higher level; and similarly B_{low} and B_{high} represent two levels of factor B. The main effect of factor A is calculated as a difference between the average response for high and low values of A. The result should be interpreted as a change in the response due to a change in the level of the factor. In our case, this method led to 16 configurations of input parameter values.

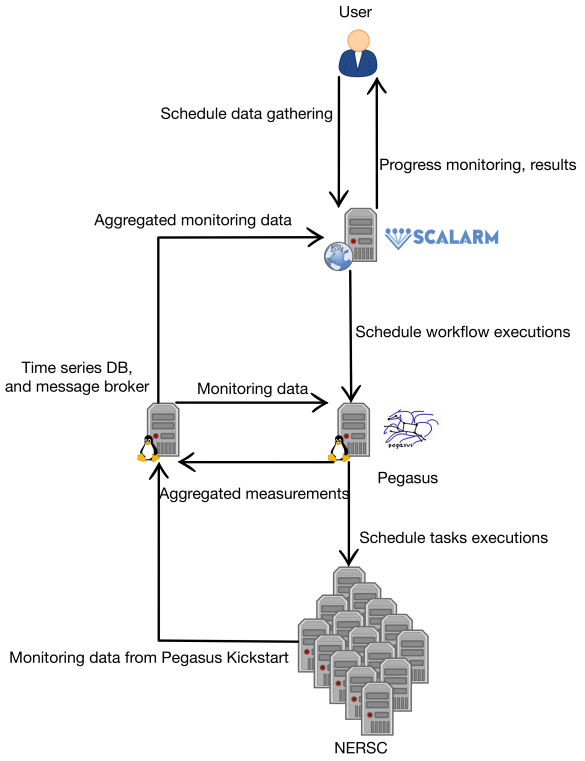

Fig. 2: Testing environment for experimental evaluation.

Orchestrating workflow execution. In the presented use case, Scalarm determines which instance of the SNS workflow and where it should be executed (i.e., the remote computing site), while Pegasus determines where and in which order workflow tasks should be executed to satisfy task dependencies. While this organization is very flexible, we used the testbed shown in Fig. 2. In the integrated system, the management of workflow executions is transparent to the user—automatically managed by Scalarm's adapters. In the user's perspective, each workflow execution is seen as a study of the input parameter space. Scalarm and Pegasus run on two separate desktop PCs, while the workflow tasks are scheduled onto the NERSC Hopper supercomputer, a Cray XE6 system with peak performance of 1.28 petaflops.

Gathering the execution results. The last part of the evaluation is results gathering. This step is handled automatically by Scalarm and the `output_reader` adapter created to transmit workflow outputs to Scalarm. Monitoring data is collected and stored for each task during the workflow execution. In the presented use case, the collected results are used to compute sensitivity analysis of workflow task performance and input parameters as shown in Figs 3 and 4.

Each presented bar chart includes normalized values of main effects for input parameters grouped by performance metrics related to CPU (*stime*, *utime*) and IO usage (write_bytes, read_bytes). Each bar is calculated as a ratio of the main effect of a given input parameter to total main effects of all input parameters on the given performance metric (hence no units). In this way, the most influential input parameters can be easily determined in the context of different performance metrics. For example, the timestamp value of 0.65 for *utime* in Fig. 3 means that the timestamps parameter is responsible for 65% of the total change in the *utime* performance metric.

CPU usage and the write_bytes metric of a NAMD task (which is a software package for parallel molecular dynamics simulation) is influenced primarily by the number of timesteps in the simulation, which is expected as the parameter determines the duration of the simulation and thereby influences the number of intermediate data dumps. The read_bytes metric is influenced by the type of atomistic structure of the material as it determines the amount of data that has to be read from files as input of these tasks. Rather strong influence of the number of cores on the *stime* metric is not obvious, it can be explained by a very small value of *stime* in these tasks, therefore its relevance is limited.

In the case of Sassena (which are calculating neutron and xray scattering intensities), its performance *utime*, write_bytes, and read_bytes metrics are mainly influenced by the number of timesteps in the simulation. Explaining sensitivity of the IO-related metrics as highly influenced by the number of timesteps will be subject of our future work as it was not expected. The actual value of *stime* is very small in these tasks as well, therefore its relevance is limited as in NAMD tasks.

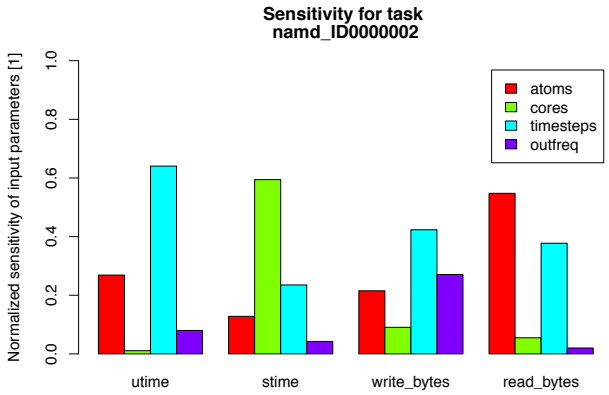

Fig. 3: Sensitivity for a NAMD task in the SNS workflow.

V. Related Work

Currently, parameter studies are mainly supported by tools aimed to perform computation on distributed computing resources, e.g. workflow management systems, grid middleware, and queuing systems. HTCondor [17] is a well-established system for building high-throughput computing environments. The system has been used by the research community as a batch system for distributed computing resources for more than three decades. In spite of its ability to efficiently handle complex workloads in distributed environments, its support to parameter studies, in terms of experiment definition and handling, is rather manual—the user needs to manually specify the input parameter values for each subsequent task. Also, there is no support for any other parametrization types.

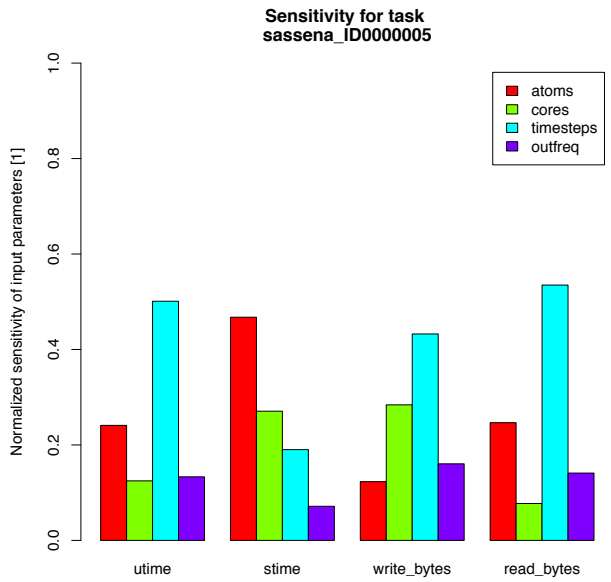

Fig. 4: Sensitivity for a Sassena task in the SNS workflow.

MOTEUR [18], [19] is a workflow management system that provides an asynchronous grid-aware enactor. The MOTEUR enactor is architected as a two-level engine. At the upper level, the core engine interprets the workflow representation, evaluates the resulting data flows and produces computational tasks ready for remote execution through a generic interface. At the lower level, the tasks descriptions are converted into jobs targeting a specific computing infrastructure. MOTEUR provides data composition patterns to explore the input parameter space. Such patterns include *one-to-one*, *all-to-all* (Cartesian product), or ternary operator (combination of the two previous patterns). However, there is no support to design of experiment techniques, nor structured outputs.

In general, from the above review it follows that no one solution offers a dedicated support for workflows used for parametric studies.

VI. Conclusions

In this paper, we described a novel approach for the organization of parameter studies using workflow-like applications, with efficient separation of concerns, resulting in flexibility in setting of computational experiments, hence worth for computational problems. The approach is based on integration of two widely used tools, one for parameter studies (Scalarm) and the other a workflow management system (Pegasus). The integration is based on a decoupled architecture, where interactions between the systems are performed through adapters. As a result, this approach maximizes the individual capability of each system: Scalarm focuses on the specification and management of the input parameter space, data points handling, and output data structuring; while Pegasus addresses workflow orchestration and execution. A key advantage of this solution is that workflows can be used as standalone applications, or as part of parameter studies (workflow reuse). The feasibility of the integration was evaluated using a performance sensitivity analysis problem involving workflow input parameters. For each data point of the input parameter space, we collected performance monitoring data that was aggregated and structured by the system. Future work include the execution of scientific workflows in cross-cloud environments with the PaaSage platform.

Acknowledgment

This work has been supported by the EU FP7-ICT project PaaSage (317715), Polish grant 3033/7PR/2014/2, and DOE contract #DE-SC0012636, "Panorama—Predictive Modeling and Diagnostic Monitoring of Extreme Science Workflows".

References

[1] A. Iosup and D. Epema, "Grid computing workloads," *Internet Computing, IEEE*, vol. 15, no. 2, pp. 19–26, March 2011.
[2] A. Almuttahar and F. Taghipour, "Computational fluid dynamics of high density circulating fluidized bed riser: Study of modeling parameters," *Powder Technology*, vol. 185, no. 1, pp. 11–23, 2008.
[3] T. Åkesson, "The lhc computing grid project," Tech. Rep., 2007.
[4] R. Graves *et al.*, "Cybershake: A physics-based seismic hazard model for southern california," *Pure and Applied Geophysics*, vol. 168, no. 3-4, pp. 367–381, 2011.
[5] E. Deelman, et al., "Griphyn and ligo, building a virtual data grid for gravitational wave scientists," in *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, 2002, pp. 225–234.
[6] D. Monge and C. G. Garino, "Logos: Enabling local resource managers for the efficient support of data-intensive workflows within grid sites," *COMPUTING AND INFORMATICS*, vol. 33, no. 1, pp. 109–130, 2014.
[7] A. Balasko, "On a workflow model based on generalized communicating p systems," *Computer Science*, vol. 17, no. 1, pp. 45–68, 2016.
[8] E. Deelman, et al., "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.
[9] D. Król and J. Kitowski, "Self-scalable services in service oriented software for cost-effective data farming," *Future Generation Computer Systems* , vol. 54, pp. 1 – 15, 2016.
[10] J. Kitowski, et al., "Distributed computing instrastructure as a tool for e-science," in *Parallel Processing and Applied Mathematics*. Springer, 2015, pp. 271–280.
[11] The Pegasus group, "Showcase of Pegasus WMS applications," https://pegasus.isi.edu/application-showcase, last viewed June 2016.
[12] D. Król, et al., "Model-based approach to study hot rolling mills with data farming," in *Proc. of 30th European Conf. on Modelling and Simulations, Regensburg, OTH Regensburg*, 2016, pp. 495–501.
[13] "FP7 PaaSage project website," http://www.paasage.eu/.
[14] D. Król, et al., "A cloud-based data farming platform for molecular dynamics simulations," in *Utility and Cloud Computing (UCC), 2014 IEEE/ACM 7th International Conference on*, Dec 2014, pp. 579–584.
[15] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.
[16] A. al-Wahish, et al., "A new apparatus design for high temperature (up to 950c) quasi-elastic neutron scattering in a controlled gaseous environment," *Review of Scientific Instruments*, vol. 86, no. 9, p. 095102, 2015.
[17] D. Thain *et al.*, "Distributed computing in practice: The condor experience: Research articles," *Concurr. Comput.: Pract. Exper.*, vol. 17, no. 2-4, pp. 323–356, Feb. 2005.
[18] T. Glatard *et al.*, "Flexible and efficient workflow deployment of data-intensive applications on grids with moteur," *International Journal of High Performance Computing Applications*, vol. 22, no. 3, pp. 347–360, 2008.
[19] R. Ferreira da Silva *et al.*, "Multi-infrastructure workflow execution for medical simulation in the virtual imaging platform," in *HealthGrid 2011*, 2011, pp. 1–10.