Smart deployment and execution of engineering simulation workflows on cloud architectures

Marco Panzeri, Roberto d'Ippolito (Noesis Solutions NV)

Emad Heydari Beni, Bert Lagaisse (imec-DistriNet, KU Leuven)

Martin Motzer, Franz Stöckl (DRÄXLMAIER Group)

1 Summary

The current state-of-practice approaches to run engineering simulation workflows rely either on desktop-based or HPC cluster-based solutions. However, these strategies are both characterized by well-known limitations hampering the performance and suitability of the target execution infrastructure.
Desktop-based solutions are normally characterized by limited capacity and performance to execute routine simulations and require substantial manual work to deploy the required software and set up the environment. HPC cluster-based solutions can provide a viable and more powerful alternative to desktop-based approaches but require the users to reserve a predefined time-slot to execute the target analyses. To properly exploit the HPC potentials, engineers sometimes also have to tailor the definition of the simulation workflow and of the analysis tool (e.g., file transfer mechanisms, grid partitioning, ...) on the specific needs and characteristics of the HPC cluster.
Desktop-based and HPC-based solutions are also severely bounded by their limited capability to scale out and manage in a flexible way workload peaks, whereby it is desirable to have a more direct and transparent control on the committed resources and it is possible to trade off the amount of resources and the expected execution time.
This work proposes a viable alternative that bypasses the limitations of desktop- or HPC-based approaches and addresses the challenge to research a more flexible and scalable way to configure, deploy and execute engineering simulation workflows on private or public cloud infrastructures.
The methodology has been successfully deployed and tested on an industrial use case defined in the context of the IDEaliSM [1] research project. This use case entails the optimization of an automotive wire harness design, where the cloud-based solution was proved to be effective in reducing the set-up time and increasing the scalability, flexibility and performance of the execution infrastructure employed to perform an extensive design space exploration analysis.

2 Methodology

The proposed approach is based on a software stack whose main components are:

- An adaptive middleware, lying between the simulation workflow engine and the cloud infrastructure and embedding a web-based simulation workflow deployment service and a cloud orchestrator. The latter manages the instantiation, monitoring and destruction of the necessary cloud resources
- A simulation workflow platform capable to flexibly connect and adapt to the cloud infrastructure to distribute the engineering analyses on the available virtual resources

2.1 Adaptive middleware

The adaptive middleware for the cloudification of engineering simulation workflows, called InfraComposer [2], is responsible for the deployment of all cloud and software resources based on the annotations defined within the simulation workflows.
InfraComposer consists of three main components (see Figure 1): (i) a workflow manager to expose a workflow deployment application programming interface (API) and to identify the annotated workflow activities, (ii) a deployment configurator to generate configurations based on the given annotations and (iii) a deployment plan composer to produce deployment plans based on reusable elementary deployment modules for the cloud orchestrator component.

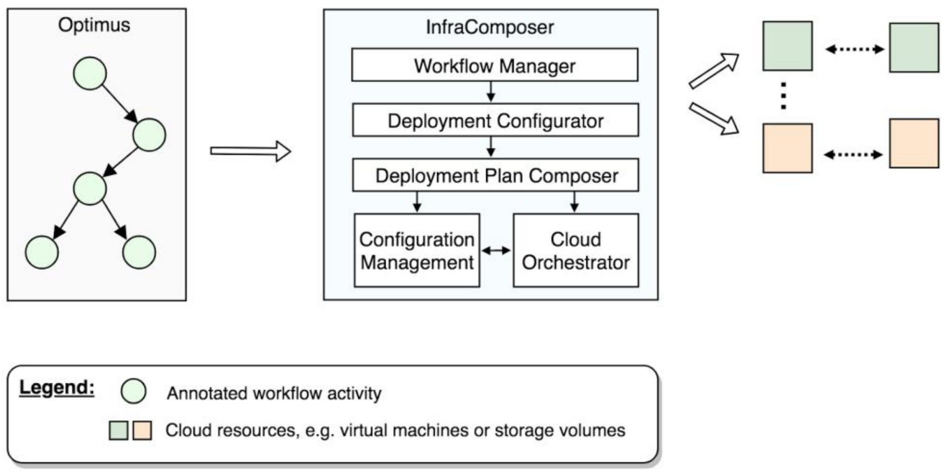

Fig. 1
Overview of the workflow cloudification architecture

2.2 Simulation Workflow Environment

The Optimus platform was used as simulation workflow environment thanks to its native capabilities to execute simulation workflows on distributed architectures through the Optimus Parallel Services (OPS), a lightweight resource management system designed for launching and tracking parallel jobs in a network of (possibly) heterogeneous machines.
The OPS system consists of three type of servers that can run on different machines in a network:

- The MASTER server is the central dispatching unit, which keeps a list of execution servers, receives jobs from client servers, and assigns jobs to execution servers
- The EXECUTION servers run on every machine in the network on which calculations can be executed. They receive jobs from the master server, and communicate directly with a client server to transfer files and get information about the jobs that have to be run.
- A CLIENT server is started whenever an Optimus method (e.g., a DOE or an optimization) is launched. The client server submits jobs to the master server and is responsible to keep track of the status of the jobs.

The OPS system is well suited to support the execution of Optimus workflows on cloud architectures. The master, execution and client OPS services can be started programmatically, without manual intervention, therefore allowing their automatic deployment and start up on dedicated machines belonging to the target virtual infrastructure.
The master node has the key capability to recognize the presence of new execution nodes at runtime, accommodating the possibility to dynamically instantiate or destroy virtual machines and to reflect these modifications in the number and type of execution nodes. This feature is of utmost importance for supporting the implementation of self-adaptive mechanisms aimed at tuning the configuration of the execution infrastructure based on information that can be inferred by monitoring the performances of the previous analyses as well as the current run.

3 Industrial Application

This section describes the industrial application where the proposed methodology was adopted to optimize the 3D-routing of an automotive cockpit wire harness designed by DRÄXLMAIER.

3.1 Use Case Configuration

The design process has been fully automated by defining all the tasks required to execute the 3D-routing tool in a Optimus workflow, including:

- The mapping of the five design inputs (i..e, the geometrical coordinates and orientations of two connectors that are positioned within the cockpit) to the input file that is read by the routing analysis tool
- The execution of the Design Compiler 43 (DC43), a Java program implementing the 3D-routing analysis tool developed in partnership between the University of Stuttgart and the company IILS mbh

- The extractionof the design output of interest representing the total wire length from one of the output files produced by DC43
    - The capability to interact with a remote Product Data Management (PDM) server (i.e., the EDMOpenSimDM of Jotne) to (a) download the required routing design rules and input geometry files and (b) upload the output results to support their long-term archival and exchange among different users

The CPU time required to run one analysis is approximately 15 minutes. This poses severe burdens on the possibility to properly explore the design space through a Design of Experiments (DOE) plan and makes this problem a suitable test bench for the workflow cloudification methodology.
The main challenge addressed in the context of this use case was represented by the need to share the input files retrieved from the PDM server among the different worker nodes without incurring in problems related to the saturation of the network bandwidth. This problem has been bypassed by using a virtual drive shared among the client and the execution nodes according to the following access permission scheme:

- The workflow running on the master node access the shared drive in write mode to deploy all the required files before any analyses is launched
- The analysis tools running on the worker nodes access the shared drive in read mode to retrieve all these files and use them to compute their calculations

The execution architecture has been set up following the scheme depicted in Figure 2. The master node hosts the Optimus simulation workflow together with the OPS CLIENT and MASTER servers. It manages (a) the exchange of data with the PDM server, (b) the deployment of the input files to the shared drive and (c) the submission and synchronization of the computation jobs through the OPS master service. The execution nodes run on Linux and provide the environment where the DC43 analyses are executed. In our set up, these machines are characterized by a CPU with 4 cores and 2.0 GHz of frequency and 8GB of RAM.

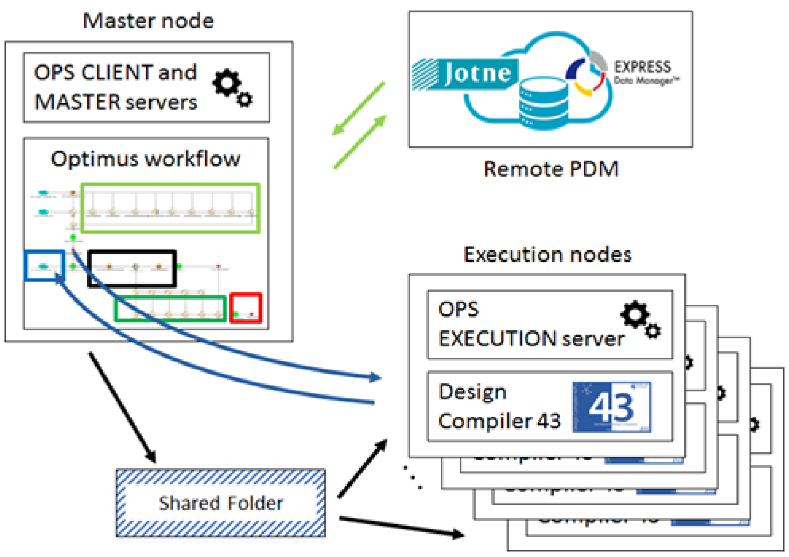

Fig. 2
Components of the simulation workflow execution environment

The feasibility and effectiveness of the proposed methodology has been assessed by comparing the set-up and execution times required to run a DOE campaign with 10 experiments considering different number of virtual machines (i.e., 1, 2, 5 and 10). The Optimus simulation workflow was prepared upfront and never modified throughout the investigated scenarios. Each DOE campaign has been performed by:

- Uploading the simulation workflow to the server hosting the InfraComposer middleware through the workflow deployment API
- Running the deployment configurator to automatically generate the deployment plan for the workflow at hand
- Runnig the composer component to automatically instantiate the virtual infrastructrure
- Starting the DOE method to execute the entailed experiments

The observed set-up time is relatively small compared to the execution time and is scarcely influenced by the number of virtual machines. Its value ranges between 1.5 minutes (1 virtual machine) to 2.11 minutes (10 virtual machines). The execution times observed for different numbers of worker nodes, shown in Figure 3, demonstrates that the infrastructure is characterized by an overall good scalability and efficiency. The latter has been measured as the ratio between T_1, the execution time using 1 virtual machine, and $p \cdot Tp$, where Tp is the execution time achieved by employing p virtual machines.

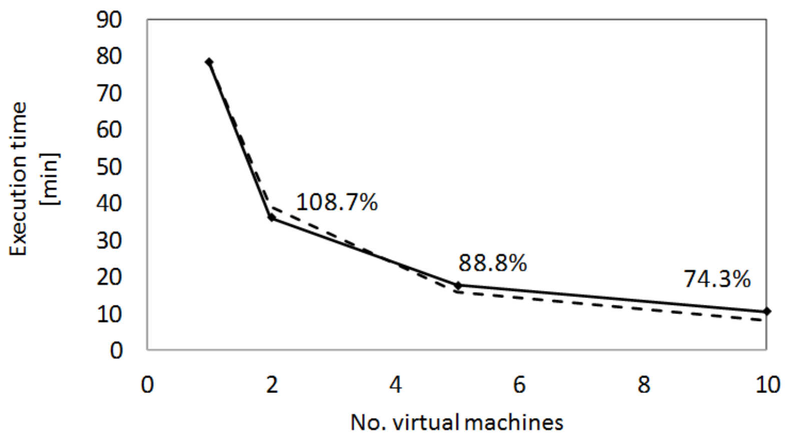

Fig. 3
Cloud-based execution times observed when running a DOE with 10 experiments for different number of virtual machines (solid line). The data labels report the performance efficiency, *E*, while the dashed line shows the ideal execution times associated with *E* = 1.

4 Conclusions

A novel approach for executing simulation workflows on cloud environments has been proposed and tested on an industrial use case entailing the 3D-routing of an automotive cockpit wire harness.
The key benefits of the proposed approach are

- No need for repetitive, manual tasks to configure the execution environment. The environment can now be configured only one time and upfront
- Flexible management of computational resources, accomodating the possibility to trade off resources capacity and execution costs
- Easy usage of engineering knowledge thanks to the possibility to steer the deployment of the virtual resources by means of a system of workflow annotations that provide information about the specific needs of the analysis tools to be executed

The proposed solution and the tests presented in this study has opened the road for more advanced investigations aimed at increasing the maturity level of this solution and enhancing it with the self-adaptive capability to steer the deployment plan based on the current run and on a history of previous executions.

5 References

[1] IDEaliSM, Integrated & Distributed Engineering Services Framework for MDO, ITEA Project 13040, https://itea3.org/project/idealism.html

[2] E.H. Beni, B. Lagaisse, W. Joosen, Adaptive and reflective middleware for the cloudification of simulation & optimization workflows, Middleware, Proceedings of the 16th Workshop on Adaptive and Reflective Middleware, ARM 2017, Las Vegas, Nevada (USA), December 11-15, 2017