Fundamenta Informaticae 128 (2013) i-iii
DOI 10.3233/FI-2013-945
IOS Press

Special Issue introduction: Scalable Workflow Enactment Engines and Technology

Preface

This Special Issue originates from the First International Workshop on Scalable Workflow Enactment Engines and Technologies (SWEET), held in conjunction with the 2012 SIGMOD conference in Scottsdale, Arizona, USA on May 20th, 2012. The goal of the workshop was to bring together researchers and practitioners to explore the state of the art in workflow-based programming for scientific data-intensive applications, and the potential of cloud-based computing in this area.

Amongst the main motivations for the adoption of workflow technology is the potential ability for computational scientists, who are the immediate beneficiaries of e-science infrastructure, to assemble data-centric science pipelines without the need for deep technical knowledge of the underlying data management infrastructure. In order to fulfill this potential, workflow middleware for science must offer ease of programming whilst providing portability across different computational environments, as well as transparent access to large pools of data storage and distributed computational resources. These user requirements translate into the need for a robust underlying data management infrastructure. The SWEET workshop was an initial exploration into the state of the art in such data-centric workflow technology, and its ability to exploit in particular the potential of elastic cloud infrastructure to achieve scalability over large size data problems.

This special issue extends this initial exploration into this space. While it includes the extended version of just one of the SWEET papers, it also collects three additional high quality contributions. Collectively, these paint an exciting landscape of emerging cloud-aware workflow technology for science.

The SWEET paper in this collection, entitled *Turbine: A distributed-memory dataflow engine for high performance many-task applications*, comes from the Argonne National Laboratory. Justin Wozniak, Timothy Armstrong and colleagues describe the *Turbine* workflow system, which is aimed at specifying programs on large-scale, high performance computing (HPC) systems using the *Swift* language. *Turbine* allows for distributed-memory evaluation of dataflow programs such that the overhead of program evaluation and task generation is spread throughout an extreme-scale computing system. This involves for example the introduction of *futures*, i.e., objects that act as proxies for results that are not yet available. Notable features are the detection of parallel loops and concurrent function invocations, which

are translated into parallel executable fragments that optimally use distributed memory and message passing to synchronize. The scalability of *Turbine* is demonstrated on several use cases and in particular analyzed on separate aspects: (i) raw task distribution, (ii) data operations, (iii) distributed data structure creation, (iv) distributed data structure creation and (v) distributed iteration.

The next paper, *Hybrid Analytic Flows— the Case for Optimization* by Simitsis and colleagues at HP Labs, addresses the problem of optimizing *Hybrid Flows*. These are dataflows that span multiple data processing engines and where the tasks are diverse, ranging from data and text analytics, to ETL, to model builders using machine learning. One flow may for instance include text analysis using Hadoop, followed by a search over relational databases using SQL queries over a data warehouse. The diversity of the engines involved in executing the flow, and of the underlying software tools used by each, make it difficult both for users to design such hybrid flows, and for the underlying runtime environment to optimize flow execution.

While the notion of a unifying design layer or heterogeneous systems is not new, the main contribution found in the paper is the optimization of hybrid flow execution. This is addressed by separating the logical view of the flow, independent of implementation constraints, from its realization. The platform described in the paper, called HFMS (Hybrid Flow Management System), automatically performs the translation from the former to the latter, optimizes the resulting execution plan, and coordinates its execution across multiple orchestration engines. The resulting optimization module, called QoX Optimizer, can optimize for different, and possibly mutually exclusive objectives, in addition to performance — including *maintainability* and *recoverability*. The cost models used by the optimizer account for both estimates of costs for individual operations, obtained using microbenchmarks, and estimates of of the costs of transitions that cross engine boundaries. The experimental evaluation, performed on a representative hybrid, demonstrates the effectiveness of the optimizer for different objective functions and cost models.

Next, a combination of two well-known workflow systems, namely Galaxy and Taverna, is described in *Towards scalable and cost-aware bioinformatics workflow execution in the cloud - recent advances to the Tavaxy workflow system*, by Mohamed Abouelhoda, Shady Issa, and Moustafa Ghanem. Taverna's strengths are mainly in its ability to orchestrate remote web service invocations, with no restriction on the data domain, and whilst exposing data parallelism in the form of a list processing model that is essentially based on the *map* higher-order function. In contrast, in Galaxy the collection of available tasks is focused on genomics and metagenomics, and execution occurs on a local host.

The focus of the paper is twofold. Firstly, it describes the programming and execution model of the Tavaxy system, which combines the benefits of Taverna and Galaxy. Secondly, it describes in detail cloud support for Tavaxy, including multiple deployment models on a computer cluster in the cloud. Advanced features include support for Elastic MapReduce (EMR) Amazon clusters, and support for dynamic resource pricing, with the goal to make execution on the Amazon cloud cost-efficient.

Finally, a small army of fifteen authors, led by Marcin Płóciennik and Tomasz Zok, assembled to produce *Approaches to Distributed Execution of Scientific Workflows in Kepler*, an account of multiple approaches for the distributed execution of workflows using the well-known Kepler environment. Kepler is an open source project for the design, execution, and sharing of scientific workflows. Kepler builds upon Ptolemy II, a dataflow system for the modeling and simulation of real-time embedded systems. Distributed execution in Kepler may occur at three different levels, namely: the workflow level, where

the entire workflow is executed on a remote cluster node; the sub-Workflow level, where each actor within a workflow may be executed on a different nodes within a cluster; and the atomic Actor level, where distributed computing and data resources are controlled within each single atomic actor.

The paper describes in detail how the Kepler Distributed Data-Parallel (DDP) framework can be utilized to realize different distributed computing patterns, making it possible for developers to exploit MapReduce implementations such as Hadoop, as well as some of its extensions, such as Stratoshpere.

Special issue editors

Jan Hidders
TUDelft, The Netherlands
a.j.h.hidders@tudelft.nl

Paolo Missier
Newcastle University, UK
paolo.missier@ncl.ac.uk

Jacek Sroka
University of Warsaw, Poland
j.sroka@mimuw.edu.pl

Jan Van den Bussche
Universiteit Hasselt, Belgium
jan.vandenbussche@uhasselt.be