

Workflow systems for Big Data analysis

Loris Belcastro, Fabrizio Marozzo

Definitions

A *workflow* is a well-defined, and possibly repeatable, pattern or systematic organization of activities designed to achieve a certain transformation of data (Talia et al (2015)).

A *Workflow Management System* (WMS) is a software environment providing tools to define, compose, map, and execute workflows.

Overview

The wide availability of high-performance computing systems has allowed scientists and engineers to implement more and more complex applications for accessing and analyzing huge amounts of data (Big Data) on distributed and high-performance computing platforms. Given the variety of Big Data applications and types

of users (from end users to skilled programmers), there is a need for scalable programming models with different levels of abstractions and design formalisms. The programming models should adapt to user needs by allowing: *i)* ease in defining data analysis applications; *ii)* effectiveness in the analysis of large datasets; *iii)* efficiency of executing applications on large scale architectures composed by a massive number of processors. One of the programming models that meets these requirements is the workflow, which through its convenient design approach has emerged as an effective paradigm to address the complexity of scientific and business Big Data analysis applications.

According to the definition of the Workflow Management Coalition (WFMC (1999)), a workflow is “*the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules*”. The term “process” here indicates a set of tasks, or activities, linked together with the goal of creating a product, calculating a result, providing a service and so on. Hence, each task represents a piece of work that forms one logical step of the overall process (Georgakopoulos et al (1995)). The same definition can be used for scientific workflows composed of several tasks (or activities) that are connected together to express data and/or control dependencies (Liu et al (2004)). Thus, a workflow can be also defined as a “*well defined, and possibly repeatable, pattern or systematic organization of activities designed to achieve a certain transformation of data*” (Talia et al (2015)).

Loris Belcastro (e-mail: lbelcastro@dimes.unical.it) and Fabrizio Marozzo (e-mail: fmarozzo@dimes.unical.it) DIMES, University of Calabria, Rende (Italy)

From a practical point of view, a workflow consists of a series of tasks, activities or events that must be performed to accomplish a goal and/or obtain a result. For example, a data analysis workflow can be designed as a sequence of pre-processing, analysis, post-processing, and evaluation tasks. It can be implemented as a computer program and can be expressed in a programming language for describing the workflow tasks and mechanisms to orchestrate them. Important benefits of the workflow formalism are: *i*) it provides a declarative way for specifying the high-level logic of an application, hiding the low-level details that are not fundamental for application design; *ii*) it is able to integrate existing software modules, datasets, and services in complex compositions that implement scientific discovery processes; *iii*) once defined, workflows can be stored and retrieved for modifications and/or re-execution, by allowing users to define typical patterns and reuse them in different scenarios (Bowers et al (2006)).

A Big Data analysis workflow can be designed through a script or a visual-based formalism. The *script-based formalism* offers a flexible programming approach for skilled users who prefer to program their workflows using a more technical approach. Moreover, this formalism allows users to program complex applications more rapidly, in a more concise way, and with higher flexibility (Marozzo et al (2015)). In particular, script-based applications can be designed in different ways: *i*) with a programming language that allows to define tasks and dependencies among them; *ii*) with annotations that allows the compiler to identify which instructions

will be executed in parallel; and *iii*) using a library in the application code to define tasks and dependencies among them. As an alternative, the *visual-based formalism* is a very effective design approach for high-level users, e.g., domain expert analysts having a limited knowledge of programming languages. A visual-based application can be created by using a visual programming language that lets users develop applications by programming the workflow components graphically. A visual representation of workflows intrinsically captures parallelism at the task level, without the need to make parallelism explicit through control structures (Maheshwari et al (2013)).

Key Research Findings

To cope with the need of high-level tools for the design and execution of Big Data analysis workflows, in the past years, many efforts have been made for the development of distributed Workflow Management Systems (WMSs), which are devoted to support the definition, creation, and execution of workflows. A WMS is a software environment providing tools to define, compose, map, and execute workflows. A key function of a WMS during the workflow execution is coordinating the operations of individual activities that constitute the workflow. There are several WMSs on the market, most of them targeted to a specific application domain. Existing WMSs can be grouped roughly into two main classes:

- *Script-based systems*, which permits to define workflow tasks and their dependencies through instructions of

traditional programming languages (e.g., Perl, Ruby or Java) or custom-defined languages. Such languages provide specific instructions to define and execute workflow tasks, such as sequences, loops, while-do, or parallel constructs. These types of instructions declare tasks and their parameters using textual specifications. Typically data and task dependencies can be defined through specific instructions or code annotations. Examples of script-based workflow systems are Swift (Wilde et al (2011)), COMPSs (Lordan et al (2014)) and DMCF (Marozzo et al (2015)).

- *Visual-based systems*, which allows to define workflows as a graph, where the nodes are resources and the edges represent dependencies among resources. Compared with script-based systems, visual-based systems are easier to use and more intuitive for domain-expert analysts having a limited understanding of programming. Visual-based workflow systems often incorporate graphical user interfaces that allow users to model workflows by dragging and dropping graph elements (e.g., nodes and edges). Examples of visual-based systems are Pegasus (Deelman et al (2015)), CloudFlows(Kranjc et al (2012)) and Kepler (Ludäscher et al (2006)).

Another classification can be done according to the way a workflow is represented. Although a standard workflow language like Business Process Execution Language (BPEL) (Juric et al (2006)) has been defined, scientific workflow systems often have developed their own workflow representations. Other than BPEL, other formalisms are used to represent and store work-

flows, such as JSON (Marozzo et al (2015)), Petri nets (Guan et al (2006)), XML-based languages (Atay et al (2007)). This situation makes difficult sharing workflow codes and limits interoperability among workflow-based applications developed by using different workflow management systems. Nevertheless, there are some historical reasons for that, as many scientific workflow systems and their workflow representations were developed before BPEL existed (Andrews et al (2003)).

In the following, we presents some representative example of workflow systems that can be used to implement applications for Big Data analysis. Some of them have been implemented on parallel computing systems, other on Grids, recently some have made available on Clouds.

Swift (Wilde et al (2011)) is a system for creating and running workflows across several distributed systems, like clusters, Clouds, Grids, and supercomputers. Swift is based on a C-like syntax and uses an implicit data-driven task parallelism (Wozniak et al (2014)). In fact, it looks like a sequential language, but all variables are *futures*, thus the execution is based on data availability (i.e., when the input data is ready, functions are executed in parallel).

COMPSs (Lordan et al (2014)) is another workflow system that aims at easing the development and execution of workflows in distributed environments, including Grids and Clouds. With COMPSs, users create a Java sequential application and select which methods will be executed remotely by providing an annotated interfaces. The runtime intercepts any call to a selected method creating a representative task and finding the data dependencies with all the

previous ones that must be considered along the application run.

Data Mining Cloud Framework (DMCF) (Marozzo et al (2016)) is a software system for designing and executing data analysis workflows on Clouds. A workflow in DMCF can be developed using a visual- or a script-based language. The visual language, called VL4Cloud (Marozzo et al (2016)), is based on a design approach for end users having a limited knowledge of programming paradigms. The script-based language, called JS4Cloud (Marozzo et al (2015)), provides a flexible programming paradigm for skilled users who prefer to code their workflows through scripts. Both languages implement a data-driven task parallelism that spawns ready-to-run tasks to Cloud resources.

Pegasus (Deelman et al (2015)) is a workflow management system developed at the University of Southern California for supporting the implementation of scientific applications also in the area of data analysis. Pegasus includes a set of software modules to execute workflow-based applications in a number of different environments, including desktops, Clouds, Grids, and clusters. It has been used in several scientific areas including bioinformatics, astronomy, earthquake science, gravitational wave physics, and ocean science.

CloudFlows (Kranjc et al (2012)) is a Cloud-based platform for the composition, execution, and sharing of interactive data mining workflows. It provides a user interface that allows programming visual workflows in any Web browser. In addition, its service-oriented architecture allows using third party services (e.g., Web services wrapping

open-source or custom data mining algorithms). The server side consists of methods for the client side workflow editor to compose and execute workflows, and a relational database of workflows and data.

Microsoft Azure Machine Learning (Azure ML)¹ is a SaaS that provides a Web-based machine learning IDE (i.e., Integrated Development Environment) for creation and automation of machine learning workflows. Through its user-friendly interface, data scientists and developers can perform several common data analysis/mining tasks on their data and automate their workflows. Using its drag-and-drop interface, users can import their data in the environment or use special readers to retrieve data from several sources, such as Web URL (HTTP), OData Web service, Azure Blob Storage, Azure SQL Database, Azure Table.

Taverna (Wolstencroft et al (2013)) is a workflow management system developed at the University of Manchester. Its primary goal is supporting the life sciences community (biology, chemistry, and medicine) to design and execute scientific workflows and support in silico experimentation, where research is performed through computer simulations with models closely reflecting the real world. Even though most Taverna applications lie in the bio-informatics domain, it can be applied to a wide range of fields since it can invoke any REST or SOAP-based Web services.

Kepler (Ludäscher et al (2006)) is a graphical workflow management system that has been used in several projects to manage, process, and analyze scientific data. Kepler provides a graphical user

¹ <https://azure.microsoft.com/it-it/services/machine-learning-studio/>

interface (GUI) for designing scientific workflows, which are a structured set of tasks linked together that implement a computational solution to a scientific problem.

Examples of Application

Workflows are widely used by scientists to acquire and analyze huge amount of data for complex analysis, such as in physics (Brown et al (2007)), medicine (Lu et al (2006) and sociology (Marin and Wellman (2011)).

The Pan-STARRS astronomical survey (Deelman et al (2009)) used Microsoft Trident Scientific Workflow Workbench for loading and validating telescope detections running at about 30 TB per year. Similarly, the USC Epigenome Center is currently using Pegasus for generating high throughput DNA sequence data (up to 8 billion nucleotides per week) to map the epigenetic state of human cells on a genome-wide scale (Juve et al (2009)). The Laser Interferometer Gravitational Wave Observatory (LIGO) uses workflows to design and implement gravitational wave data analysis, such as the collision of two black holes or the explosion of supernovae. The experiment records approximately 1 TB of data per day, which is analyzed by scientists in all parts of the world (Brown et al (2007)). In this scenario, workflow formalism demonstrates its effectiveness in programming Big data scientific applications.

The workflow formalism has been also used for implementing and executing complex data mining applications on large datasets. Some examples are: *Par-*

allele clustering (Marozzo et al (2011)), where multiple instances of a clustering algorithm are executed concurrently on a large census dataset to find the most accurate data grouping; *Association rule analysis* (Agapito et al (2013)), which is a workflow for association rule analysis between genome variations and clinical conditions of a group of patients; *Trajectory mining* (Altomare et al (2017)) for discovering patterns and rules from trajectory data of vehicles in a wide urban scenario. In some cases, the workflow formalism has been integrated with other programming models, such as MapReduce, to exploit the inherent parallelism of the application in presence of Big Data. As an example, in (Belcastro et al (2015)) a workflow management system has been integrated with MapReduce for implementing a scalable predictor of flight delays due to weather conditions (Belcastro et al (2016)).

Future Directions for Research

Workflow systems for Big Data analysis require high-level and easy-to-use design tools for programming complex applications dealing with huge amount of data. There are several open issues that will require research and development in the near future, such as:

- *Programming models.* Several scalable programming models have been proposed, such as MapReduce (Dean and Ghemawat (2008)), Message Passing (Gropp et al (1999)), Bulk Synchronous Parallel (Valiant (1990)), Dryad (Isard et al (2007)) or Pregel (Malewicz et al (2010)). Some development works must be

done for extending workflow systems to support different programming models, which could improve their capabilities in terms of efficiency, scalability and interoperability with other systems.

- *Data storage.* The increasing amount of data generated every day needs even more scalable data storage systems. Workflow systems should improve their capabilities to access data stored on high-performance storage systems (e.g., NoSQL systems, Object based storage on Clouds) by using different protocols.
- *Data availability.* Workflows systems have to deal with the problem of granting service and data availability, which is an opened challenge that can negatively affect performances. Several solutions have been proposed for improving exploitation, such as using a cooperative multi-Cloud model to support Big Data accessibility in emergency cases (Lee et al (2012)), but more studies are still needed to handle the continue increasing demand for more real time and broad network access.
- *Local mining and distributed model combination.* As workflow-based applications often involve several local and distributed data sources, collecting data to a centralized server for analysis is not practical or, in some cases, possible. Scalable workflow systems for data analysis have to enable local mining of data sources and model exchange and fusion mechanisms to compose the results produced in the distributed nodes. According to this approach, the global analysis can be performed by distributing the local mining and supporting the global combination

of every local knowledge to generate the complete model.

- *Data and tool interoperability and openness.* Interoperability is a main open issue in large-scale distributed applications that use resources such as data and computing nodes. Workflow systems should be extended to support interoperability and ease cooperation among teams using different data formats and tools.
- *Integration of Big Data analysis frameworks.* The service-oriented paradigm allows running large-scale distributed workflows on heterogeneous platforms along with software components developed using different programming languages or tools. This feature should improve the integration between workflows and other scalable Big Data analysis software systems, such as frameworks for fine grain in-memory data access and analysis. In such way, it will be possible to extend workflows towards exascale computing, since exascale processors and storage devices must be exploited with fine-grain runtime models.

References

- Agapito G, Cannataro M, Guzzi PH, Marozzo F, Talia D, Trunfio P (2013) Cloud4snp: Distributed analysis of snp microarray data on the cloud. In: Proc. of the ACM Conference on Bioinformatics, Computational Biology and Biomedical Informatics 2013 (ACM BCB 2013), ACM Press, Washington, DC, USA, p 468, ISBN 978-1-4503-2434-2
- Altomare A, Cesario E, Comito C, Marozzo F, Talia D (2017) Trajectory pattern mining for urban computing in the cloud. Transactions on Parallel and Distributed Systems

- 28(2):586–599, ISSN:1045-9219
- Andrews T, Curbera F, Dholakia H, Golland Y, Klein J, Leymann F, Liu K, Roller D, Smith D, Thatte S, et al (2003) Business process execution language for web services
- Atay M, Chebotko A, Liu D, Lu S, Fotouhi F (2007) Efficient schema-based xml-to-relational data mapping. *Information Systems* 32(3):458–476
- Belcastro L, Marozzo F, Talia D, Trunfio P (2015) Programming visual and script-based big data analytics workflows on clouds. In: Grandinetti L, Joubert G, Kunze M, Pascucci V (eds) *Post-Proc. of the High Performance Computing Workshop 2014*, IOS Press, Cetraro, Italy, *Advances in Parallel Computing*, vol 26, pp 18–31, ISBN: 978-1-61499-582-1
- Belcastro L, Marozzo F, Talia D, Trunfio P (2016) Using scalable data mining for predicting flight delays. *ACM Transactions on Intelligent Systems and Technology* 8(1)
- Bowers S, Ludascher B, Ngu AHH, Critchlow T (2006) Enabling scientific workflow reuse through structured composition of dataflow and control-flow. In: *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pp 70–70, DOI 10.1109/ICDEW.2006.55
- Brown DA, Brady PR, Dietz A, Cao J, Johnson B, McNabb J (2007) A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. *Workflows for e-Science* pp 39–59
- Dean J, Ghemawat S (2008) Mapreduce: simplified data processing on large clusters. *Communications of the ACM* 51(1):107–113
- Deelman E, Gannon D, Shields M, Taylor I (2009) Workflows and e-science: An overview of workflow system features and capabilities. *Future generation computer systems* 25(5):528–540
- Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, Mayani R, Chen W, da Silva RF, Livny M, et al (2015) Pegasus, a workflow management system for science automation. *Future Generation Computer Systems* 46:17–35
- Georgakopoulos D, Hornick M, Sheth A (1995) An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases* 3(2):119–153
- Gropp W, Lusk E, Skjellum A (1999) *Using MPI: portable parallel programming with the message-passing interface*, vol 1. MIT press
- Guan Z, Hernandez F, Bangalore P, Gray J, Skjellum A, Velusamy V, Liu Y (2006) Grid-flow: a grid-enabled scientific workflow system with a petri-net-based interface. *Concurrency and Computation: Practice and Experience* 18(10):1115–1140
- Isard M, Budiu M, Yu Y, Birrell A, Fetterly D (2007) Dryad: distributed data-parallel programs from sequential building blocks. In: *ACM SIGOPS operating systems review*, ACM, vol 41, pp 59–72
- Juric MB, Mathew B, Sarang PG (2006) *Business process execution language for web services: an architect and developer's guide to orchestrating web services using BPEL4WS*. Packt Publishing Ltd
- Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman BP, Maechling P (2009) Scientific workflow applications on amazon ec2. In: *E-Science Workshops, 2009 5th IEEE International Conference on*, IEEE, pp 59–66
- Kranjc J, Podpečan V, Lavrač N (2012) Clowdflows: a cloud based scientific workflow platform. In: *Machine Learning and Knowledge Discovery in Databases*, Springer, pp 816–819
- Lee S, Park H, Shin Y (2012) Cloud computing availability: multi-clouds for big data service. In: *Convergence and Hybrid Information Technology*, Springer, pp 799–806
- Liu L, Pu C, Ruiz DD (2004) A systematic approach to flexible specification, composition, and restructuring of workflow activities. *Journal of Database Management* 15(1):1
- Lordan F, Tejedor E, Ejarque J, Rafanell R, Ivarez J, Marozzo F, Lezzi D, Sirvent R, Talia D, Badia R (2014) Servicess: An interoperable programming framework for the cloud. *Journal of Grid Computing* 12(1):67–91
- Lu Q, Hao P, Curcin V, He W, Li YY, Luo QM, Guo YK, Li YX (2006) Kde bioscience: platform for bioinformatics analysis workflows. *Journal of biomedical informatics* 39(4):440–450
- Ludäscher B, Altintas I, Berkley C, Higgins D, Jaeger E, Jones M, Lee EA, Tao J, Zhao Y (2006) *Scientific workflow manage-*

- ment and the kepler system. *Concurrency and Computation: Practice and Experience* 18(10):1039–1065
- Maheshwari K, Rodriguez A, Kelly D, Madduri R, Wozniak J, Wilde M, Foster I (2013) Enabling multi-task computation on galaxy-based gateways using swift. In: 2013 IEEE International Conference on Cluster Computing (CLUSTER), IEEE, pp 1–3
- Malewicz G, Austern MH, Bik AJ, Dehnert JC, Horn I, Leiser N, Czajkowski G (2010) Pregel: a system for large-scale graph processing. In: *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ACM, pp 135–146
- Marin A, Wellman B (2011) *Social network analysis: An introduction*. The SAGE handbook of social network analysis 11
- Marozzo F, Talia D, Trunfio P (2011) A cloud framework for parameter sweeping data mining applications. In: *Proc. of the 3rd IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, IEEE Computer Society Press, Athens, Greece, pp 367–374, iSBN 978-0-7695-4622-3
- Marozzo F, Talia D, Trunfio P (2015) Js4cloud: script-based workflow programming for scalable data analysis on cloud platforms. *Concurrency and Computation: Practice and Experience* 27(17):5214–5237
- Marozzo F, Talia D, Trunfio P (2016) A workflow management system for scalable data mining on clouds. *IEEE Transactions On Services Computing*
- Talia D, Trunfio P, Marozzo F (2015) *Data Analysis in the Cloud*. Elsevier, iSBN 978-0-12-802881-0
- Valiant LG (1990) A bridging model for parallel computation. *Commun ACM* 33(8):103–111
- WFMC T (1999) *Glossary*, document number wfmc, issue 3.0. TC 1011
- Wilde M, Hategan M, Wozniak JM, Clifford B, Katz DS, Foster I (2011) Swift: A language for distributed parallel scripting. *Parallel Computing* 37(9):633–652
- Wolstencroft K, Haines R, Fellows D, Williams A, Withers D, Owen S, Soiland-Reyes S, Dunlop I, Nenadic A, Fisher P, et al (2013) The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research* 41(W1):W557–W561
- Wozniak JM, Wilde M, Foster IT (2014) Language features for scalable distributed-memory dataflow computing. In: *Data-Flow Execution Models for Extreme Scale Computing (DFM), 2014 Fourth Workshop on*, IEEE, pp 50–53