Amato, F.; Moscato, F.; Xhafa, F. Enabling IoT stream management in multi-cloud environment by orchestration. *32nd IEEE International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2018, 16-18 May 2018, Krakow, Poland: proceedings*, p.687-692. Doi: 10.1109/WAINA.2018.00168

Enabling IoT Stream Management in Multi-Cloud Environment by Orchestration

Flora Amato[*], Francesco Moscato[†], Fatos Xhafa[‡]
[*]*Univ. of Naples Federico II, Department of Electrical Engineering and Information Technology, Naples, Italy*
Email: flora.amato@unina.it
[†]*Univ. of Campania ”Luigi Vanvitelli”, Dept. of Scienze Politiche, Caserta, Italy*
Email: francesco.moscato@unicampania.it
[‡]*Universitat Politècnica de Catalunya, Department of Computer Science, Barcelona, Spain*
Email: fatos@cs.upc.edu

***Abstract*—Every-Day lives are becoming increasingly instrumented by electronic devices and any kind of computer-based (distributed) service. As a result, organizations need to analyse an enormous amounts of data in order to increase their incomings or to improve their services. Anyway, setting-up a private infrastructure to execute analytics over Big Data is still expensive. The exploitation of Cloud infrastructure in IoT Stream management is appealing because of costs reductions and potentiality of storage, network and computing resources. The Cloud can consistently reduce the cost of analysis of data from different sources, opening analytics to big storages in a multi-cloud environment. Anyway, creating and executing this kind of service is very complex since different resources have to be provisioned and coordinated depending on users' needs. Orchestration is a solution to this problem, but it requires proper languages and methodologies for automatic composition and execution. In this work we propose a methodology for composition of services used for analyses of different IoT Stream and, in general, Big Data sources: in particular an Orchestration language is reported able to describe composite services and resources in a multi-cloud environment.**

***Keywords*-Cloud Computing; Orchestration; Formal Semantics; Availability**

I. Introduction and Related Works

Originally, framework for Big Data Management, like Hadoop[1] supported distributed file systems inside clusters architectures. On the other hand, Cloud Architecture has emerged as a de facto standard for achieving best performances considerably reducing costs: there are no good reasons (except the ones related to security and data ownership) to do not use Cloud services in IoT Stream and Big Data management. In particular, when Big-Data Source is distributed with different and heterogeneous sources, cloud infrastructure results very appealing both from performance and economic points of view.

One of the main problem here is to overcome problems related to the creation of composite services that uses multi-cloud resources like several virtual storages containing data sources.

Let us imagine a scenario where an organization wants to perform some analytics on data from social networks[2], [3], from weather and other geographical data and from other economical issues[4]. Obviously, because of their heterogeneity, data are distributed and maintained in different data storage. The goal of the analysis my be the identification of co-relation among information drilled-up from the different sources[5]. For example, this can be used by car sellers in order to optimize their production depending on past weather conditions, year period, geographic region etc.

If data are stored in different virtual storage, the execution of proper analyses on data will need the interaction of different Cloud Resources and services, belonging to different cloud providers (both private or public): this is a multi-cloud scenario where a complex service (the analysis of the whole data) needs the execution of component (sub)services that executes on different and heterogeneous resources. The execution of the complex service needs both interoperability among resources and something able to define and execute a workflow process calling proper sub-service when needed.

The National Institute of Standards and Technology (NIST) defined this kind of execution *Orchestration* [6]. Following the NIST definition, orchestration refers to the composition of system components to support arrangement, coordination and management of resources in order to provide (composite) cloud services to cloud consumers.

Some important issues about are: (a) providers are responsible of the management and enactment of activities in orchestrated services; (b) orchestration involves services and resources in all levels of Cloud stack; (c) orchestration must face Quality of Service (QoS) of both component and composite services.

In this context, it is clear that a methodology able to compose cloud Services needs:

- a language like BPEL4WS[7] able to describe not only services composition at Service-as-a-Service (SaaS) level, but Cloud Resources too,
- A framework able to orchestrate at all Cloud Layers[6] services and resources, and also able to analyse and manage the composite service as whole, answering to questions like: are results and Input-Output resources compliant with cloud components services ?

We think that compliance cannot be evaluated only by means of syntactical checks or by type checking. In fact, enabling technologies for automatic composition, Cloud architects are now going to use Semantics-based methodologies [8],[9]. Anyway, cloud resources are not like web services messages: they include some complex elements like virtual infrastructures. In addition, in order to achieve automatic Cloud Service composition, resources should carry out a semantic description of their functionality, as well as a semantic description of their parameters.

Actually, several semantics-based approaches for *simple* web services composition exist (a survey is in [10]). Some of them ([11], [12], [7], [13]) exploit BPEL4WS orchestration language and OWL-based ontologies for services description.

In this work we present an architecture and a language able to define, analyse and manage multi-cloud Orchestration and we will show how these can be used for the analysis of a distributed Big-Data sources[14]. For what analysis of the semantics and the soundness of the composition concerns, we do not describe here the Ontology-based description of resources: it is based on the work described in [7] and on the Ontology for the Cloud defined in [15] and [16]. In brief, we use OWL-S with IOPE grounding in order to analyse interoperability among services, but composition is managed as a workflow process and formal operational semantics allow for analysis of composite processes. Similarly, the analysis of the Quality of Service of composite services is out of the scope of this work and it is introduced in [17].

The paper is organized as follows: Sec. II describes the overall architecture of our system and its methodology; Sec.III introduces the workflow-based language. Sec.IV reports an example of the application of the methodology to a Big-Data analysis scenario.Finally, Sec.V contains some concluding remarks.

II. Methodology and Architecture

Fig.1 show the overall architecture of the framework we propose for the composition and orchestration of Cloud services.

We work upon the existence of several, eventually heterogeneous, Cloud Providers (**CloudA,$\cdots$, CloudN**). Each providers is able to instantiate common Cloud Resources like Computing Nodes, Virtual Storages or Virtual Network (inter and intra-clouds). In addition, Virtual Storages can of course maintain different sets of Data.

The architecture of the **Orchestrator** we are describing consists of:

- an **Execution Scheduler**: the scheduler reads the description of a composite service and executes the proper services when needed, eventually scheduling data migration too from a virtual storage to another if needed.
- a **Data Dispatcher**: It executes physical data migration and maintains information about data produced during the execution of the composite service.
- a **Broker**: it enacts common service brokering actions: if the resource is not yet acquired on a provider, it provides for acquisition and management. It is also responsible for the provisioning of the resources and their configuration.
- a **Deployer**: this component deploys needed services at SaaS (from a pool of available services)on proper resource in the Cloud.
- the **Resources Orchestrator Manager** interface with existent resources orchestrators[18]. At the moment this module supports COPE[19] and OpenStack HEAT[20] Orchestrators.

The workflow-based language we use for description of the whole service is called Operational Flow Language (OFL). The language is complex enough to describe several patterns, as well as simple enough to be defined by means of clear operational semantics [21]. Compositional rules enable patterns description.

OFL is able to describe simple workflow graphs that are expressive enough to catch many control-flow and dataflow workflow patterns as explained in the next section. In addition workflow graphs described in OFL allow for the creation of analysis models by using Model Transformation techniques(see [22], [23], [24] for more details).

III. Workflow and Orchestration

In this section we briefly introduce the basic elements and operational semantics of OFL language. Than, we show how patterns can be defined by using OFL.

OFL is a workflow-based language. According to the definition provided by the Workflow Management Coalition[1] we consider a workflow process definition as a network of activities and their relationships. Each activity represents one logical step within a process, i.e. the smallest unit of work to be performed. The completion of an activity and the starting of another activity is a Transition point in the workflow execution. A Transition may be unconditional, but the sequence of the activity execution may also be decided at run time according to the value assumed by one or more logical expressions, in this case the sequence of operations depends on Transition Conditions that are evaluated after an activity has started or ended. The activation of an execution thread may be affected by the evaluation of the associated Transition Conditions.

Some points are defined within the workflow that allow the flow of the activities to be controlled: AND-split is a point in where a single thread of control splits into two or more threads which are executed in parallel. AND-join is a point where two or more parallel activities converge into a single thread of execution. XOR-split is a decision point where only one of alternative branches is executed. XOR-join is a point in the workflow where two or more parallel

[1] http://www.wfmc.org/

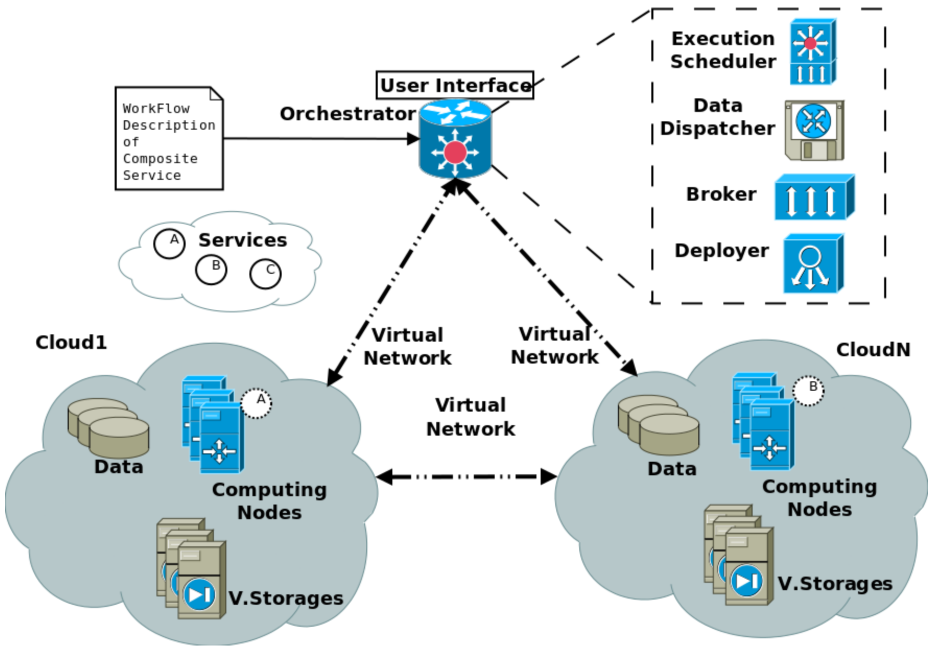

Figure 1. System Architecture

activities converge in a single thread of execution without synchronization. OR-split is a decision point between several alternative workflow branches. OR-join is a point in which several alternative branches re-converge into a single thread. In the following XOR, AND and OR are called split or join Conditions. Fig. 2 illustrates these elements.

Therefore in the OFL language, workflow processes consist of a network of Activity nodes and edges (Transitions) identified by a pair od nodes $(FromActivity, ToActivity)$.

Activities represents atomic Cloud Service or resource invocation, as well as composite activities (i.e. sub workflow processes).

OFL skeletons are graphs defining preconditions and postconditions for cloud services execution and Cloud resources usage.

Activities definitions include effective information about Cloud Services to execute. The main components of this part of the language are:

- *Participant*: it specifies the Cloud Provider where the Orchestrator will execute the activity;
- *API*: this include the description of the service, the REST API used to invoke the service (with Inputs, Outputs, name etc.)
- *Requirements*: the QoS requested by the activity and eventually the SLAs with the Participant

This describes the SaaS level of the Composite Service. In addition, the Activity is linked to further description at PaaS and IaaS levels as depicted in Fig.2. Platform and Resource elements shares the same main components of SaaS level (i.e. Participant, API and Requirement description of the Resource). The only main difference is that at resource level, the API may refer to an existing resource, or to a Resource Orchestrator.

In addition, Activities or Transitions may require the existence of proper virtual-network resources. They connects other Resources and may be required by transitions in order to create channel for data routing or by Activities if services need to exchange data or events with other resources during their execution.

Proper edges will define the aforementioned dependencies: *needsChannel* when declaring the need to exchange data and events; *connectedTo* in order to state which resources (usually at IaaS level) are connected by the channel.

All this is resumed in Fig.2

OFL is an XML based language, but here we report only its graphical representation for simplicity's sake.

IV. A CASE STUDY

The example we want describe is inspired by a recent commercial study[2] reporting the importance of performing analytics over Weather, Geo-referenced data.

The main problem here is that analytics must cope with proper data about commercial activities in a given field and all drilled-up results both from weather and commercial fields, have to be collected and further analysed in order to achieve good results.

This obviously leads to the need of automatize a composite analytics service by exploiting Cloud platforms and resources.

[2]http://advertising.weather.com/big-data-weather-data-enhanced-business-strategy/

Fig.3 resumes this scenario and the OFL representation of the composite process.

The top and the bottom of the figure depicts the two Cloud providers (here called CloudWeather and CloudPrivate) where some resources (four computational nodes and a Virtual Storage in CloudWeather; two computational nodes and a VirtualStorage in CloudPrivate) are deployed. On the CloudWeather provider, a PaaS Apache Spark Service is available, while an Hadoop service is available on the other provider.

The Workflow process describing composition is in the middle of the figure. It consists in two Activities executing in parallel after the beginning of the process: the first (W) requests the execution of analytics on the Spark PaaS, while the second (BS) enacts the execution of analytics on Hadoop. When both terminates, the Orchestrator controls the existence of the Virtual Network VN1. If it is not allocated, the Broker provides (if necessary resources exist) the necessary resource. Then, the Collect service is ready to start. The Deployer deploys the service on CN6: the activity can execute on proper data that are collected by this very service. The process can finally terminate leaving results on CloudPrivate Virtual Storage.

Data routing and some minor details are not reported for brevity.

V. Conclusions and Future Works

We have described a language and a framework for Cloud service composition. The language, OFL, is able to describe complex composition patterns, navigating all Cloud architectural layers: SaaS, PaaS, IaaS.

We an example showing how the language and the framework can be exploited in order to describe Composite Big-Data processes based on the use of Multi-Cloud platforms[25],[26].

Future work will integrate ontology-based reasoning into the methodology in order to automatically build composite Cloud Services with given semantics and QoS.

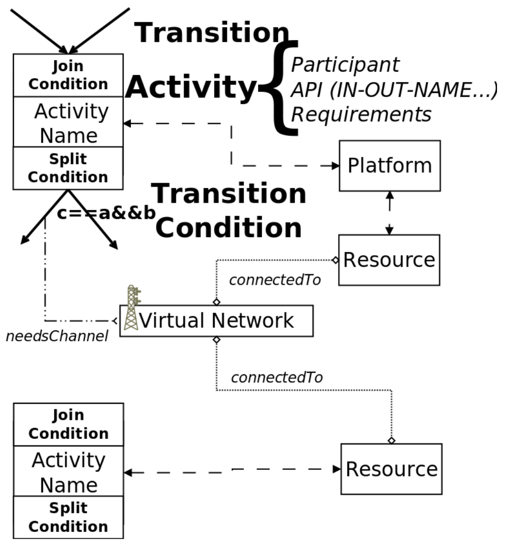

Figure 2. Workflow elements

References

[1] T. White, *Hadoop: The definitive guide.* " O'Reilly Media, Inc.", 2012.

[2] W. Balzano and F. Vitale, "Dig-park: a smart parking availability searching method using v2v/v2i and dgp-class problem," in *Advanced Information Networking and Applications Workshops (WAINA), 2017 31st International Conference on.* IEEE, 2017, pp. 698–703.

[3] W. Balzano, A. Murano, and F. Vitale, "Snot-wifi: Sensor network-optimized training for wireless fingerprinting," *Journal of High Speed Networks*, vol. 24, no. 1, pp. 79–87, 2018.

[4] S. Kwoczek, S. D. Martino, and W. Nejdl, "Predicting and visualizing traffic congestion in the presence of planned special events," *J. Vis. Lang. Comput.*, vol. 25, no. 6, pp. 973–980, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.jvlc.2014.10.028

[5] A. Minutolo, M. Esposito, and G. De Pietro, "Design and validation of a light-weight reasoning system to support remote health monitoring applications," *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 232–248, 2015.

[6] VV.AA., "Us government cloud computing technology roadmap release 1.0 (draft)," in *Special Publication 500-293.* NIST, 2011, vol. 2, pp. 1–85.

[7] G. D. Lorenzo, N. Mazzocca, F. Moscato, and V. Vittorini, "Towards semantics driven generation of executable web services compositions," *International Journal of Software, JSW*, vol. 2, no. 5, pp. 1–15, 2007.

[8] F. Amato, F. Colace, L. Greco, V. Moscato, and A. Picariello, "Semantic processing of multimedia data for e-government applications," *Journal of Visual Languages and Computing*, vol. 32, pp. 35–41, 2016.

[9] F. Amato, A. Mazzeo, A. Penta, and A. Picariello, "Using nlp and ontologies for notary document management systems," in *Database and Expert Systems Application, 2008. DEXA'08. 19th International Workshop on.* IEEE, 2008, pp. 67–71.

[10] S. Dustdar and W. Schreiner, "A survey on web services composition," *International journal of web and grid services*, vol. 1, no. 1, pp. 1–30, 2005.

[11] P. Traverso and M. Pistore, "Automated composition of semantic web services into executable processes," in *The Semantic Web–ISWC 2004.* Springer, 2004, pp. 380–394.

[12] E. Sirin, J. Hendler, and B. Parsia, "Semi-automatic composition of web services using semantic descriptions," in *1st Workshop on Web Services: Modeling, Architecture and Infrastructure*, 2003, pp. 17–24.

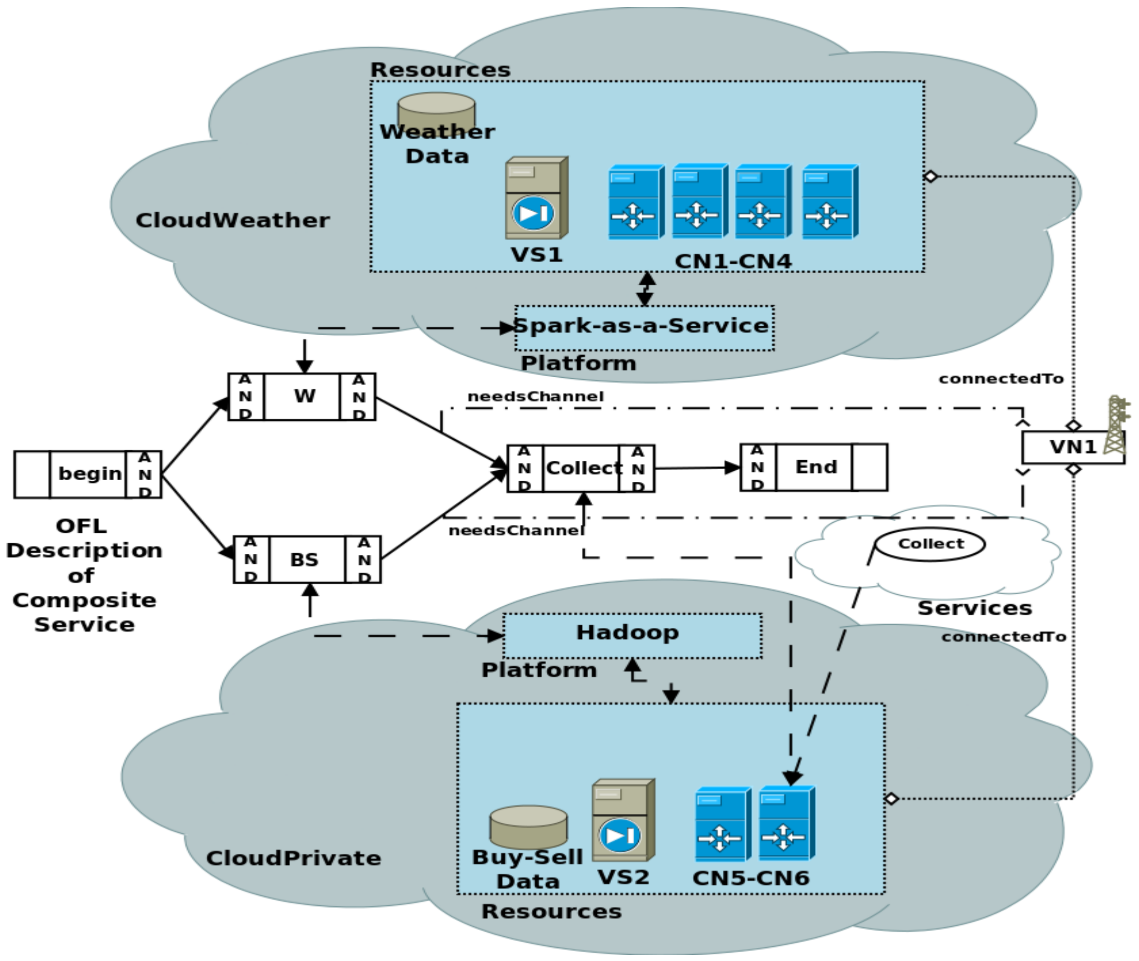

Figure 3. Case Study

[13] G. D. Lorenzo, F. Moscato, N. Mazzocca, and V. Vittorini, "Automatic analysis of control flow in web services composition processes," in *PDP*, 2007, pp. 299–306.

[14] F. Amato, M. Barbareschi, V. Casola, A. Mazzeo, and S. Romano, "Towards automatic generation of hardware classifiers," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8286 LNCS, no. PART 2, pp. 125–132, 2013.

[15] F. Moscato, B. D. Martino, and R. Aversa, "Enabling model driven engineering of cloud services by using mosaic ontology," *Scalable Computing: Practice and Experience*, vol. 13, no. 1, 2012.

[16] F. Moscato, R. Aversa, B. D. Martino, T.-F. Fortis, and V. I. Munteanu, "An analysis of mosaic ontology for cloud resources annotation," in *IEEE Proc. of FedCSIS 2011 Conference*, 2011, pp. 973–980.

[17] F. Moscato, "Model driven engineering and verification of composite cloud services in metamorp(h)osy," in *Proc. of 6th, International Conference on Intelligent Networking and Collaborative Systems INCoS-2014*. IEEE, 2014.

[18] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud resource orchestration programming: Overview, issues, and directions," *Internet Computing, IEEE*, vol. 19, no. 5, pp. 46–56, 2015.

[19] C. Liu, B. T. Loo, and Y. Mao, "Declarative automated cloud resource orchestration," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 26.

[20] R. Kumar, N. Gupta, S. Charu, K. Jain, and S. K. Jangir, "Open source solution for cloud computing platform using openstack," *International Journal of Computer Science and Mobile Computing*, vol. 3, no. 5, pp. 89–98, 2014.

[21] G. D. Plotkin, "A structural approach to operational semantics," 1981.

[22] "A taxonomy of model transformation," *Electronic Notes in Theoretical Computer Science*, vol. 152, no. 0, pp. 125 – 142, 2006, proceedings of the International Workshop on Graph and Model Transformation (GraMoT 2005), Graph and Model Transformation 2005.

[23] F. Moscato, R. Aversa, and A. Amato, "Describing cloud use case in metamorp(h)osy," in *IEEE Proc. of CISIS 2012 conference*, 2012, pp. 793–798.

[24] F. Moscato, F. Amato, A. Amato, and R. Aversa, “Model–driven engineering of cloud components in metamorp (h) osy,” *International Journal of Grid and Utility Computing*, vol. 5, no. 2, pp. 107–122, 2014.

[25] L. Barolli, X. Chen, and F. Xhafa, “Advances on cloud services and cloud computing,” *Concurrency Computation*, vol. 27, no. 8, pp. 1985–1987, 2015.

[26] F. Pop, C. Dobre, V. Cristea, N. Bessis, F. Xhafa, and L. Barolli, “Reputation-guided evolutionary scheduling algorithm for independent tasks in inter-clouds environments,” *International Journal of Web and Grid Services*, vol. 11, no. 1, pp. 4–20, 2015.