Internet of Things orchestration using DagOn* workflow engine

Dante D. Sánchez-Gallegos*, Diana Di Luccio†, José Luis Gonzalez-Compean* and Raffaele Montella†
* Adaptivez Heterogeneous Storage Research Group, Cinvestav Tamaulipas, Victoria, Mexico.
† Dpt. of Science and Technologies, University of Naples "Parthenope", Naples, Italy.
Email: {dsanchez, jgonzalez}@tamps.cinvestav.mx, {diana.diluccio, raffaele.montella}@uniparthenope.it

***Abstract*—The increasing of connected tiny, low-power, embedded devices, grouped into the generic definition of "Internet of Things" (IoT), raised remarkably the amount of in-situ collected data. However, at the time of writing, IoT devices have limited storage and computation resources if compared with a cloud computing or on-premises infrastructure. IoT devices often suffer for reduced connectivity due to the place of the deployment or other technical, environmental or economic reasons. In this work, we present the DagOn* workflow engine as a part of an IoT orchestration scenario oriented to operational environmental prediction. Our novel approach is devoted to join the two worlds of workflows, in which each task runs on a dynamically allocated computational infrastructure, with tiny jobs targeted to embedded devices hosting sensors and actuators. We show our preliminary results applied to a demonstration use case. We are confident that further development of the proposed technology will affect positively production applications for massive and geographically distributed data collection.**

***Index Terms*—workflow, IoT, embedded systems, container, environmental data.**

I. Introduction

Embedded systems are becoming powerful and flexible in terms of connectivity, computing power, re-programmable behavior and straightforwardness of usage representing the iconic interface between the ephemeral world represented by the cloud computing based resources and the practical reality [1]. Both experimental use cases and production applications [2], especially in the rising field of smart cities, are demonstrating how the cloud computing and the Internet of Things (IoT) technologies converge [3], melt and evolve [4], depicting scenarios that drive the change to the scientific, industrial and everyday life landscapes [5]. IoT enabled devices are platforms hosting sensors and actuators field-deployed, dispersed, heterogeneous and really often virtually organized as fleets [6] cooperating at once as one single instrumented entity [7]. Since the early attempts to empower data acquisition with the emerging technology of grid computing, the spread of pervasive low-power connected devices increased the data availability reducing the cost per single data sample in a terrific way and, consequently, enlarging the need for computation in order to manage those data and extract from them useful information [8]. The support for large scale data processing is given by the workflow engines orchestrating cloud-provided elastic and virtually infinite computational resources playing a primary role in computational sciences and engineering because the gained overall performance, the affordability, the reliability, the availability and the reproducibility of any computational experiment [9].

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

In this paper, we present an IoT orchestration scenario leveraging on DagOn* [10], an open source Python library enabling the execution of workflows, which are defined as directed acyclic graph (DAG) jobs on anything (https://github.com/DagOnStar). DagOn* based workflow applications (DagOn*Apps) run on a single entry point (DagOn*Machine) spawning tasks on distributed computational resources ranging from the local machine to virtualized or containerized HPC clusters hosted on private, public or hybrid clouds. We improved the DagOn* design and implementation enabling the interaction with IoT connected devices as workflow tasks. As the development of DagOn* has been driven by the need to orchestrate a workflow for operational environmental predictions, the need to collect data from weather stations, radars, webcams and other similar equipment aimed the need for the IoT extension (Fig. 1). This picture represents the real IoT/Workflow production scenario provided by the weather-marine observation and forecasting service run by the Department of Science and Technologies of the University of Naples Parthenope in Italy (*Center for Marine and Atmospheric Monitoring and Modelling*, http://meteo.uniparthenope.it).

In this context, the main contribution of this paper is:

- A schema for the creation of a full featured synergy between distributed IoT based data acquisition components and distributed computation orchestrated using a workflow engine.

The rest of the paper is organized as follows: in Section II, related work is discussed, the approach requirements are motivated in Section III; Section IV is about DagOn* design strategies and IoT interfacing approach; Section V is about a real-world application focusing on environmental data collecting and processing; Section VI gives conclusion remarks and draws the path for future research development.

II. Related Work

There is a large number of available workflow engines, with similar management strategy, devoting to enable the user to run complex applications involving up to millions of

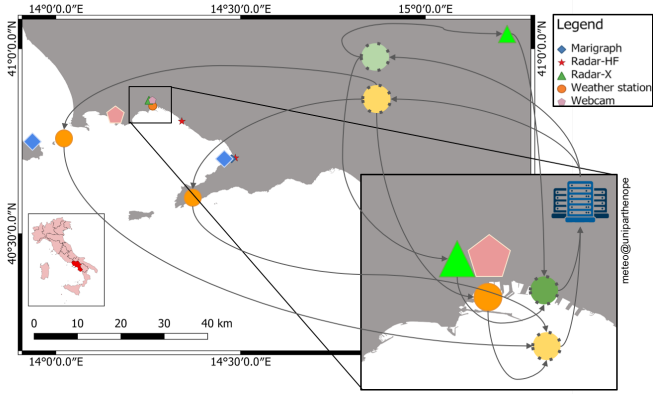

Fig. 1. A production workflow involving IoT components for weather and coastal marine data collection. The gray lines represents the data flow between the IoT/Workflow components.

jobs on different computational resources [11]. At the best of our knowledge, we can consider Swift [12], Parsl [13], Galaxy [14] and Pegasus [15] as the workflow environments that are related to the presented work. Swift parallel programming language has been designed for composing application programs which can be executed on multi-core processors, clusters or cloud, using a C-like syntax with implicit parallelism. Because our application domain (both production and on demand [16] earth system modelling, Internet of Things data processing [17] and machine learning tasks [18]), Swift could represent a valid alternative to DagOn*. Nevertheless, the Swift Parallel programming language is perfect for complex data flow description, executing software across high performance task spawning, but it still lacks in system integration. To overcome this problem we developed a data flow oriented workflow engine, as Swift, but offering the task flow definition as a feature.

Based on Swift model, Parsl is an advanced parallel workflow engine leveraging on Python scripting library, supporting asynchronous and implicitly parallel data-oriented workflows. As Parsl, we offer our workflow as a Python library, but once a workflow has been defined, its representation as JSON file could be saved to be used via a web graphical user interface as Galaxy.

Galaxy and its extension FACE-IT Globus Galaxy [19] represent a web-based genomic workbench which enables users to perform computational data processing [14] in a virtual laboratory environment, characterized by on demand computation. For the routine execution of workflow based applications the use of this tool could appear inappropriate, while the use of DagOn* is well suited.

Pegasus leverages on a set of technologies helping workflow-based applications to run in a diverse and different computational environments. Workflows are represented by graphs coded using the DAX XML schema. The DAX can be generated using APIs available for programming languages like Java and Python. At the best of our knowledge there is no integration between Pegasus and IoT components.

A DagOn*-based application could be developed using multiple workflows with reciprocal interaction. The developer can implement custom task components extending the ones already available. At the time of writing, DagOn* supports Python naive tasks, web tasks, batch tasks, SLURM tasks, AWS tasks, and container tasks.

III. Motivation

DagOn*, the workflow engine proposed in this paper, is the result of our experience in executing massive workflows on a heterogeneous infrastructure powered by different computing resources spread across local web farms and cloud facilities. Unlike the described workflow engines, at the best of our knowledge, the DagOn* features related to IoT device interfacing are remarkably peculiar, representing a novel contribution in both fields of workflows for large scale science and IoT based sensor and actuator networks. DagOn* abstracts the concept of workflow IoT Task enabling the development of applications interacting directly with the IoT hardware or it can be configured using, among the others, the OpenSource platform Stack4Things [20] and the cloud platform Amazon Web Services IoT (AWS IoT) [21], implemented with the aim to get the best from both IoT and Cloud Computing worlds for the sake of offering the best technology to support the rising architectural model based on fog computing and computation at the edge.

AWS IoT integrates Amazon Web Services, like Amazon Kinesis, Amazon S3, Amazon DynamoDB, Amazon CloudWatch, and AWS CloudTrail, for IoT applications, allowing to connect more devices easily and to securely interacting with cloud applications and other devices using the messaging protocol MQTT (Message Queue Telemetry Transport) [22]. More recently, Stack4Things, an OpenStack-based Internet of Things framework developed at the University of Messina (Italy), has been launched in order to facilitate the management of IoT devices according to a Cloud-oriented approach. In the Stack4Things architecture, the data are sent to MongoDB and the OpenStack Dashboard, extended with an IoT-enabled panel, is used for data visualization [23].

IV. Approach for IoT orchestration by using DagOn*; Design and implementation

We designed DagOn* in order to reduce the runtime footprint within the actual application. DagOn* leverages on the application lifecycle, supported by a Python library providing the main system components, while a not mandatory service component is used for the monitoring and management of workflows. In this work, we extended the DagOn* architecture including the component devoted to coupling the embedded device I/O with the workflow task approach (Fig. 2). A DagOn* application is developed as Python script using any Python extension with a regular sequential approach. Parallel tasks are defined by using the **Task** component. In Fig. 2 the different task types are schematized as: *Native* (regular

Python functions which are executed locally in a concurrent application thread), *Web* (Web tasks have been designed in order to enable the developer to make workflows interacting with remote resources accessible using web services), *Batch* (external software components executed using SSH or a local scheduler), *Cloud* (a task can be embedded in a virtual machine image and executed on the cloud) and *Container*, (a task can be represented by a container script and executed on a *containerized* infrastructure).

The DagOn* Runtime performs the interaction with the executors as on premises local or remote resources instanced by either virtual machines or containers deployed on public/private/hybrid clouds.

The Task component is in charge of the dependencies, the interaction with the workflow, the file staging system and it participates to the monitoring process in conjunction with the DagOn* Service. More specialized Task types are designed as extensions of the Task component. DagOn* design considers the workflow:// schema as the root of current workflow virtual file system. Under these conditions, workflow:**///workflow_unique_name///task_unique_name/** is the root of the scratch directory created by the DagOn* Runtime. DagOn* uses this notation to evaluate the task dependencies using a back-reference approach. Although the workflow:// schema has been designed for batch based tasks, we used the same convention to map files into local variables in the case of native tasks (Fig. 3).

The IoT Task component is built on the functional modules:

- **Interface**. It is made upon two components working in a specular fashion. The Input Adapter maps a file in Python variable accessible by the IoT Task implementation. The Output Adapter works in the opposite way. In this way the worfkflow:// schema works even in the IoT Task context.
- **Proxy**. It is defined as an interface designed after the typical Arduino-like style programming model. The IoT Task is implemented defining the callback functions for sensors and actuators definition and initialization, the main loop kernel and what has to be performed when the exit condition is reached. This component has been designed with the abstraction in mind targeting the independence from specific IoT/Cloud technologies.
- **Abstraction**. The communication with the actual IoT device can be implemented through a direct access to the low-power hardware capabilities, GPIOs and serial ports or using IoT/Cloud management framework as AWS-IoT or Stack4Things.
- **Communicator**. It is a modular component depending on the Abstraction because it is related to the technology used to perform the communication between the machine executing the DagOn* IoT Task and the IoT Device. In a typical IoT application, the usage of the MQTT or CoAP [24] protocols is a common issue. Different types of communicators working alongside the DagOn* IoT Task are the ones based on Web Socket and REST APIs.
- **IoT Device**. Is the actual IoT Device with no restrictions about computational, storage, GPIO and communication capabilities. Devices such as Texas Instruments Simplelink CC1350 SensorTag, Ubuntu Neo, Raspberry Pi 3, Arduino Yun, ESP32 where successfully tested under the development of the IoT Task.

The orchestration of IoT infrastructure by using DagOn* is managed through a life cycle that includes two main stages: the definition of a workflow as well as the deployment of a workflow in different infrastructures and the establishment of data-flow through the workflows (from sensors to the containers to the cloud resources).

A. Defining IoT infrastructure as a DagOn workflow*

In the workflow definition stage, a directed graph is created (See Fig. 4) to represent the data-flows in IoT infrastructure. In this graph, the nodes represent applications to be connected to a set of sensors, whereas edges represent the data-flows produced by applications.

1) Parallel pattern design: In the definition of the workflow stage, patterns can be created and added to the fog nodes to improve the efficiency of the data processing in the workflows. The patterns executing applications are encapsulated into the fog nodes in parallel to reduce the response time of the processing of data. It is quite useful to process data in critical scenarios such as medical and environmental disasters.

2) Workflow implementation: In the first stage, we use DagOn* to define a script to launch a workflow through different IT infrastructures such as a single server, an HPC cluster, Docker containers, and a public, private or hybrid cloud. In this context, we use the Cloud and Docker tasks as nodes in an IoT infrastructure, where the Docker tasks are collocated in the Fog, closer than the cloud to IoT nodes than the Cloud. A workflow with DagOn* is defined as a Python script which is processed to resolve data dependencies between the tasks. To define a task, an infrastructure where this is going to be launched must be defined. All the kinds of tasks managed by DagOn* are interoperable between them, which simplifies the exchange of data between nodes.

*B. Deploying an IoT infrastructure using DagOn**

In the deployment stage, the graph is materialized in the form of a system running on containers (see nodes and cloud resources in Fig. 4). The nodes include applications such as acquisition, preprocessing and delivery, which are encapsulated into virtual containers created by using Docker images. In the Fog, the $acquisition$ application is in charge to connect with the sensors and IoT devices to retrieve sensed data, the $preprocessing$ identifies and deletes outliers, whereas the $delivery$ is in charge of processing the sensed data and sending results forward to the cloud resources by using I/O DagOn* interfaces. This first processing of data is performed to discover information which is critical. The IoT devices receive in the short term after submitting the data to the Fog. The deployment of the solution is orchestrated by DagOn* using the information provided by the user in the definition of the tasks (such as the type of task, data dependencies, the machine to deploy it, and access credentials). Also, environmental data

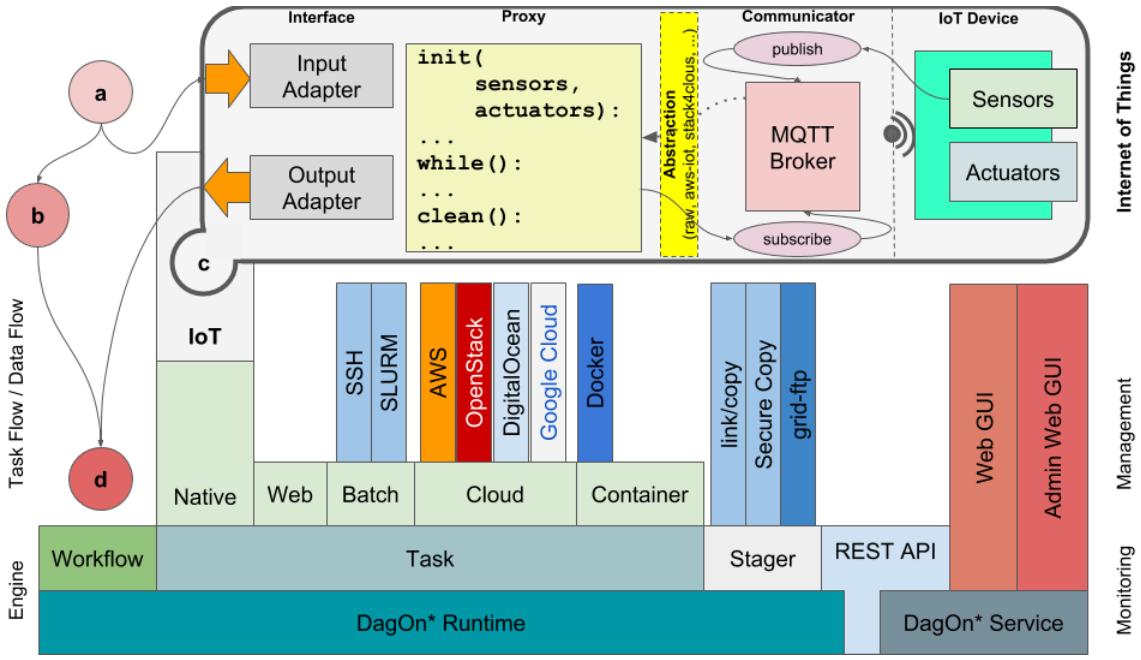

Fig. 2. The DagOn* architectural schema with a focus on the different task types.

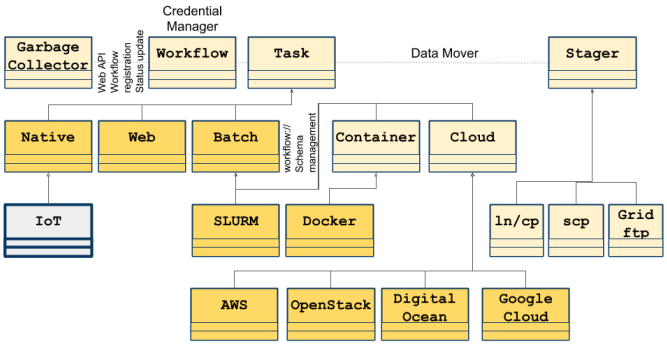

Fig. 3. The DagOn* class diagram.

The IoT Task is an extension of the native regular task. It implements the adapter interface and the proxy framework enabling the communication between the DagOn* application and the IoT device(s).

are collected from the machine where the task is going to be executed, such as the data transference protocols available (HTTP, SCP, GridFTP or link) and the access credentials to transfer data. With this information collected, DagOn* stager chooses the best transference protocol between tasks. DagOn* has support to multiple cloud providers such as OpenStack, Amazon EC2, Google Cloud Services, and DigitalOcean. The tasks deployed in instances of these cloud providers can exchange data between them. This allows deploying a heterogeneous IoT solution decentralizing the services in different cloud provider instances. All DagOn* containers expose an API Rest, which is called by the IoT devices to transmit data and attend their requests. Our model allows to increment the number of DagOn* containers executed in the Fog in a flexible way. This allows a large number of IoT devices on the edge, without impacting in the performance of other containers.

When a workflow has been deployed on an IoT infrastructure, a continuous dataflow is created to transport the data extracted from the IoT devices to fog nodes and to cloud resources.

V. Use Case

In this section we present a use case of the orchestration of an IoT infrastructure to attend diverse and different IoT devices. We developed a real-world application focused on the environmental data collecting and processing.

We have deployed a Manager/Worker pattern, as the one showed in Fig. 4, to attend a different number of IoT devices, which collect data such as device temperature, external environment temperature, humidity, barometer, motion, light, and Received Signal Strength Indicator.

A. Use case metrics

The metrics chosen to evaluate the performance of the prototype were the deployment time and the service time. The deployment time is the time required by DagOn* to launch an architecture pattern of virtual containers, whereas the service time is the time required by an application encapsulated into a virtual container to complete a given task.

B. Evaluation results

We performed experiments and the metrics (deployment and service times) were captured and evaluated. We first measured the deployment time produced by DagOn* in an

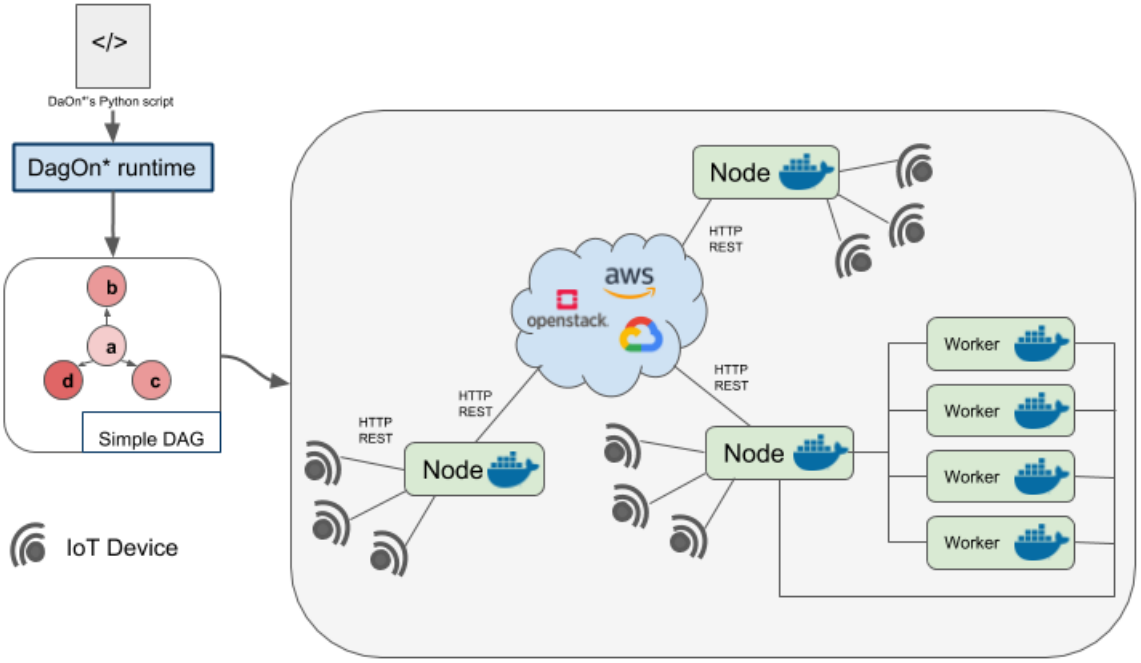

Fig. 4. Conceptual design of a IoT infrastructure by using DagOn*.

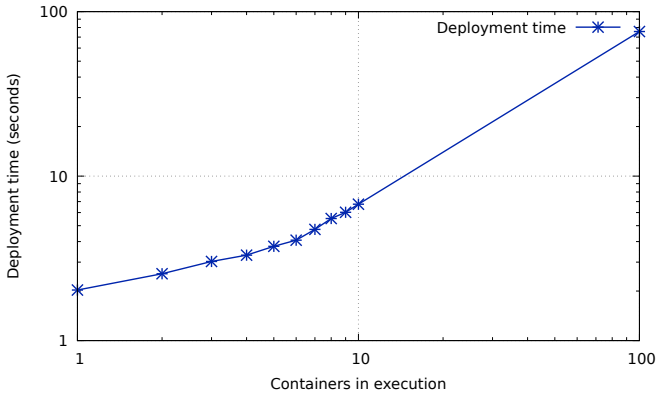

Fig. 5. Deployment time of DagOn* patterns including different number of Docker containers.

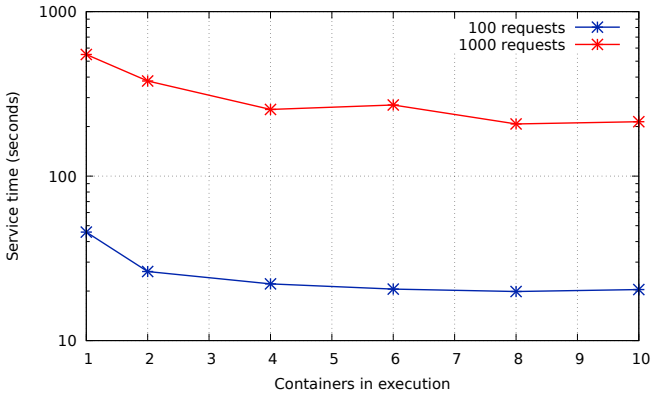

Fig. 6. Service time of a pattern deployed with DagOn* to attend different number of concurrent requests.

experiment where manager/worker patterns are deployed on a single computer. In this experiment, we launched (31 times) eleven patterns including different number of virtual containers (workers). In the first ten patterns, the number of workers was increased one-by-one from 1 to 10, whereas in the eleventh pattern launched 100 containers. Fig. 5 shows, on the vertical axis, the deployment time in seconds spent by DagOn* to deploy the virtual containers of the patterns evaluated in this experiment (horizontal axis). As expected, the more containers in the pattern, the more deployment time. Nevertheless, this time is acceptable as a pattern of 10 virtual containers can be operable in only 7,5 seconds, whereas a pattern of 100 containers in 75 sec.

Fig. 6 shows the service time produced by DagOn* patterns including different number of virtual containers (horizontal axis) when processing 100 and 1000 registers of data recollected from five sensors (vertical axis), which were processed by the applications encapsulated into worker fog nodes (virtual containers). As it can be seen, the patterns processing data in concurrent manner improve the performance of that process. The performance gain of patterns processing 100 registers was from 42.40% for two workers to 56.47% for four workers, whereas for 1000 registers the improvement was from 31.03% for two workers to 62.18% for four workers. The improvement is steady when launching more than four workers as the computer where the pattern was deployed on only included 4 physical cores; as a result, workers end up using virtualized cores when launching more than four containers.

VI. Conclusion

In this work we introduced a novel DagOn* application focused on the orchestration of several diverse and different IoT connected devices through a Python based tool for data intensive scientific workflows. In the presented scenario we leveraged heavily on the deployment of Docker containers at the edge of the cloud, physically close to the IoT devices. The DagOn* programming model enables scientists and engineers to manage embedded devices, low-power single board computers and, in general, any pervasive connected object hosting sensors and actuators as part of a large scale data science workflow [25].

Acknowledgment

This work has been supported by the University of Napoli Parthenope, Italy (project DSTE333 "Modelling Mytilus Farming System with Enhanced Web Technologies" funded by Campania Region/Veterinary sector). The authors are grateful to the CMMMA (*Center for Marine and Atmospheric Monitoring and Modelling*, http://meteo.uniparthenope.it), the forecast service of the University of Napoli "*Parthenope*" for providing the necessary hardware and software resources. The authors are grateful to the Mexican National Council of Science and Technology (CONACyT) for providing the scholarship with number 847116/634574.

References

[1] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things: Resource-oriented architecture and best practices," in *Architecting the Internet of things*. Springer, 2011, pp. 97–129.

[2] R. Montella, D. Di Luccio, L. Marcellino, A. Galletti, S. Kosta, G. Giunta, and I. Foster, "Workflow-based automatic processing for internet of floating things crowdsourced data," *Future Generation Computer Systems*, vol. 94, pp. 103–119, 2019.

[3] R. Di Lauro, F. Lucarelli, and R. Montella, "Siaas-sensing instrument as a service using cloud computing to turn physical instrument into ubiquitous service," in *Parallel and Distributed Processing with Applications (ISPA), 2012 IEEE 10th International Symposium on*. IEEE, 2012, pp. 861–862.

[4] G. Laccetti, R. Montella, C. Palmieri, and V. Pelliccia, "The high performance internet of things: using gvirtus to share high-end gpus with arm based cluster computing nodes," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2013, pp. 734–744.

[5] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog computing: A platform for internet of things and analytics," in *Big data and internet of things: A roadmap for smart environments*. Springer, 2014, pp. 169–186.

[6] G. Fortino, C. Savaglio, C. E. Palau, J. S. de Puga, M. Ganzha, M. Paprzycki, M. Montesinos, A. Liotta, and M. Llop, "Towards multi-layer interoperability of heterogeneous iot platforms: the inter-iot approach," in *Integration, Interconnection, and Interoperability of IoT Systems*. Springer, 2018, pp. 199–232.

[7] C. Catlett, T. Malik, B. Goldstein, J. Giuffrida, Y. Shao, A. Panella, D. Eder, E. van Zanten, R. Mitchum, S. Thaler *et al.*, "Plenario: An open data discovery and exploration platform for urban science." *IEEE Data Eng. Bull.*, vol. 37, no. 4, pp. 27–42, 2014.

[8] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[9] G. Fortino, A. Guerrieri, W. Russo, and C. Savaglio, "Integration of agent-based and cloud computing for the smart objects-oriented iot," in *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*. IEEE, 2014, pp. 493–498.

[10] R. Montella, D. Di Luccio, and S. Kosta, "Dagon*: Executing directed acyclic graphs as parallel jobs on anything," in *(in press)*, 2018.

[11] A. Barker and J. Van Hemert, "Scientific workflow: a survey and research directions," in *International Conference on Parallel Processing and Applied Mathematics*. Springer, 2007, pp. 746–753.

[12] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster, "Swift/t: Large-scale application composition via distributed-memory dataflow processing," in *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*. IEEE, 2013, pp. 95–102.

[13] Y. Babuji, K. Chard, I. Foster, D. S. Katz, M. Wilde, A. Woodard, and J. Wozniak, "Parsl: Scalable parallel scripting in python," in *10th International Workshop on Science Gateways*, 2018.

[14] J. Goecks, A. Nekrutenko, and J. Taylor, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences," *Genome biology*, vol. 11, no. 8, p. R86, 2010.

[15] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger, "Pegasus: a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[16] E. Chianese, A. Galletti, G. Giunta, T. Landi, L. Marcellino, R. Montella, and A. Riccio, "Spatiotemporally resolved ambient particulate matter concentration by fusing observational data and ensemble chemical transport model simulations," *Ecological Modelling*, vol. 385, pp. 173–181, 2018.

[17] R. Montella, S. Kosta, and I. Foster, "Dynamo: Distributed leisure yacht-carried sensor-network for atmosphere and marine data crowdsourcing applications," in *Cloud Engineering (IC2E), 2018 IEEE International Conference on*. IEEE, 2018, pp. 333–339.

[18] R. Montella, A. Petrosino, and V. Santopietro, "Whoareyou (way): A mobile cuda powered picture id card recognition system," in *International Conference on Image Analysis and Processing*. Springer, 2017, pp. 375–382.

[19] R. Montella, A. Brizius, D. Di Luccio, C. Porter, J. Elliot, R. Madduri, D. Kelly, A. Riccio, and I. Foster, "Using the face-it portal and workflow engine for operational food quality prediction and assessment: An application to mussel farms monitoring in the bay of napoli, italy," *Future Generation Computer Systems*, 2018.

[20] S. Distefano, A. Puliafito, G. Merlino, F. Longo, and D. Bruneo, "A stack4things-based platform for mobile crowdsensing services," in *ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT), 2016*. IEEE, 2016, pp. 1–8.

[21] W. Tärneberg, V. Chandrasekaran, and M. Humphrey, "Experiences creating a framework for smart traffic control using aws iot," in *Proceedings of the 9th International Conference on Utility and Cloud Computing*. ACM, 2016, pp. 63–69.

[22] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "Mqtt-sa publish/subscribe protocol for wireless sensor networks," in *Communication systems software and middleware and workshops, 2008. comsware 2008. 3rd international conference on*. IEEE, 2008, pp. 791–798.

[23] V. Diaconita, A.-R. Bologa, and R. Bologa, "Hadoop oriented smart cities architecture," *Sensors*, vol. 18, no. 4, p. 1181, 2018.

[24] C. Bormann, A. P. Castellani, and Z. Shelby, "Coap: An application protocol for billions of tiny internet nodes," *IEEE Internet Computing*, vol. 16, no. 2, pp. 62–67, 2012.

[25] R. Montella, D. Di Luccio, P. Troiano, A. Riccio, A. Brizius, and I. Foster, "Wacomm: A parallel water quality community model for pollutant transport and dispersion operational predictions," in *Signal-Image Technology & Internet-Based Systems (SITIS), 2016 12th International Conference on*. IEEE, 2016, pp. 717–724.