Chapter 1

Data and Computation Movement in Fog Environments: the DITAS approach

Pierluigi Plebani[1], David Garcia-Perez[2], Maya Anderson[3], David Bermbach[4], Cinzia Cappiello[1], Ronen I. Kat[3], Achilleas Marinakis[5], Vrettos Moulos[5], Frank Pallas[4], Barbara Pernici[1], Stefan Tai[3], and Monica Vitali[1]

[1] Politecnico di Milano – Dipartimento di Elettronica ed Informazione
Piazza Leonardo da Vinci 32, 20133 Milano, Italy

[2] Atos Spain SA
Pere IV 291, 08020 Barcelona, Spain

[3] IBM Research - Haifa
Haifa University Campus, Mount Carmel, Haifa, 3498825 Israel

[4] TU Berlin – Information Systems Engineering Research Group
Einsteinufer 17 – 10587 Berlin, Germany

[5] NTUA – National Techinical University of Athens
9 Iroon Polytechniou Str., 15773 Zografou Campus, Athens, Greece

Abstract: Data-intensive applications are nowadays crucial in several domains: e.g., e-health, smart cities, industry 4.0. In fact, the significant increase of sensor deployment, in conjunction with the huge amount of data that users are producing, especially with their smartphones, requires proper data management. The goal of this paper is to focus on how to improve data management when data are produced and consumed in a Fog environment, where both resources at the edge of the network (e.g., sensors and mobile devices) and resources in the cloud (e.g., virtual machines) are involved and need to operate seamlessly. Based on the approach proposed in the European DITAS project, data and computation movement between the edge and the cloud is studied, to balance between such characteristics as latency, response time - better supported when data are stored on edge-located resources - and scalability, reliability – better supported when data live in the cloud. To enable data and computation movement, an approach based on the principles of Service Oriented Computing applied to a Fog environment has been adopted.

Keywords: Data-intensive application, Service Oriented Computing, Data movement, Computation Movement, Containerized applications, Virtual Data Container.

1.1 Introduction

In various fields, the amount of data produced and processed is ever increasing and applications able to manage this vast amount of data are required. Scientific disciplines, such as bioinformatics or astrophysics, have already encountered this data deluge several years ago and have recognized the data-intensive science as the fourth research paradigm in addition to the experimental, theoretical, and simulation-based paradigms [1,2]. For this reason, parallel computing, even based on specialized hardware - and where scaling-out approach was preferred to a scaling-up approach - has been adopted to produce tools for improving the efficiency of scientific computations.

With the proliferation of smart devices, the same type of scenario is turning up in other domains. Predictions on IoT estimate 32 billion of connected devices in 2020 that will be able to produce 440 Petabytes (accounting for 10% of the world

data) [3]. On top of these data, data processing and analysis is required to extract meaningful information. This becomes crucial when data comes from the operational infrastructure of a large company and analysis is required to compute KPIs, produce reports for the management, and, thus, for driving strategic decisions. Combining this complexity with the need for being faster in management decisions, real-time processing of data streams becomes fundamental [4].

In cloud platforms, these data-intensive applications (DIA) have found the ideal environment for executing tasks and processing significant amounts of data while scalability and reliability are guaranteed by the underlying infrastructure [5]. Furthermore, the adoption of cloud solutions has had a significant impact on costs, as storing and managing data in the cloud is typically inexpensive compared to on-premises solutions. The development of this type of application took advantage of several approaches that had been proposed in the recent years. Especially inside the Apache community[6], several projects have been started to deal with batch processing, e.g., Hadoop, and real-time processing, e.g., Storm, Spark, or Kafka, just to name a few. With these tools, data-intensive application developers can rely on a specific programming model that simplifies access to heterogeneous data sources storing data in different formats and significantly reduces the effort required to make data processing scalable and reliable. To reduce the inherent latency of the cloud, specific architectures, e.g., the Lambda Architecture [7], have been proposed for real-time analytics.

Although these approaches are now widely adopted, there are situations in which relying on a cloud infrastructure for implementing data-intensive applications is not beneficial, especially when data are produced outside of the cloud by devices (e.g., smart objects, laptops, servers, dedicated systems) living on the premises of who wants to analyse the data produced. Firstly, when data are produced at the edge of the network but processed in the cloud using the solutions mentioned above — bandwidth could become a bottleneck thus increasing the latency. Secondly, security and privacy issues are still one of the main reasons why cloud adoption remains limited especially in application domains where data processing mainly involves sensitive information (e.g., e-health) that usually cannot be moved to the cloud as they are. On the other hand, applying the data processing solutions developed for cloud-based infrastructure directly to the edge could be very challenging: each infrastructure has its own characteristics, could be managed differently, and involve a variety of devices with different characteristics that make a one-size-fits-all approach really difficult.

In such a scenario, Fog Computing [8], often also referred to as Edge Computing [9], is an emerging paradigm aiming to extend Cloud Computing capabilities to fully exploit the potential of the edge of the network where traditional devices as well as new generations of smart devices, wearables, and mobile devices -- the so-called Internet of Things (IoT) -- are considered. Especially for data-intensive applications, since IoT is a source for enormous amounts of data that can be exploited to provide support in a multitude of different domains (e.g., predictive machinery maintenance, patient monitoring), this new paradigm has opened new frontiers [10]. In fact, with Fog Computing we can take advantage of resources

living on the edge and the cloud to exploit the respective advantages. Data could be stored closer to where they are produced using edge located resources if the data cannot leave the boundary of the organization that owns them; conversely, cloud solutions could be adopted only after data are transformed to preserve privacy. Orthogonally, data processing could occur on the edge when the available information, computational power, and the response time do not require a scalable solution. On the other hand, data processing will be moved to the cloud when limitation of network bandwidth is not an issue.

The goal of this paper is to introduce how data management in data-intensive applications could be improved through data and computation movement in Fog environments. In particular, this paper focuses on the experience of the European DITAS project [11] which aims to improve, through a combined cloud and fog platform, the development of data-intensive applications by enabling information logistics in Fog environments for delivering information at the right time, the right place, and with the right quality [12] using both resources belonging to the cloud and the edge. The resulting data and computation movement is enabled by Virtual Data Containers (VDCs) which provide an abstraction layer, adopting the Service Oriented Computing [13] principles and hiding the underlying complexity of an infrastructure made of heterogeneous devices. Applications developed using the DITAS toolkit will be able to exploit the advantages of both cloud-based solutions in terms of reliability and scalability, as well as edge-based solutions with respect to latency and privacy.

The rest of this paper is structured as follows. Section 1.2 introduces the characteristics of data-intensive applications when immersed in a Fog Computing-based environment. Section 1.3 discusses the approach adopted in the DITAS project to support the deployment and execution of a data-intensive application in Fog environments, while Section 1.4 specifically focuses on the data and computation movement actions. Finally, Section 1.5 discusses related work, while Section 1.6 concludes the work outlining the future work which DITAS will focus on.

1.2 Data intensive applications in Fog Computing

Data-intensive applications are becoming more and more crucial elements of IT systems due to the ever-increasing amount of data that needs to be managed. Several types of data-intensive applications can be developed to cover one or more phases of the data management, i.e., data capture and storage, data transmission, data curation, data analysis, and data visualization [5].

In recent years, usually under the umbrella of Big Data, researchers and practitioners have focused on providing tools, methods, and techniques for efficiently managing extensive amounts of data in various formats and schemas. As a result, distributed file systems (e.g., HDFS), new generations of DBMS where the relational model is no longer adopted (e.g., MongoDB [14], Cassandra [15]), and new programming models (e.g., MapReduce), as well as new architectures (e.g., Lambda Architecture) have been proposed. Regardless of the specific solution, most of them usually rely on resources available on the cloud, thus offering the possibility to easily scale applications in or out with the amount of data to be processed. Actually, in some cases, relying only on cloud infrastructures cannot be

feasible for two main reasons: (i) data cannot be moved from where they are collected due to privacy/security issues, and (ii) the time required to move data to the cloud could be prohibitive. In such a scenario, Fog Computing [7] aims to support the required synergy between the cloud - where the application usually runs - and the devices at the edge of the network - where the data are generated - especially in IoT environments. In fact, cloud and edge are usually seen as two distinct and independent environments that, based on the specific needs, are connected to each other to move data usually from the edge to the cloud. To create a synergy between these two paradigms, Fog Computing has been coined - initially in the telco sector - to identify a platform able ``to provide compute, storage, and networking services between cloud data centers and devices at the edge of the network'' [16]. Based on this, and also in the light of the definition proposed by the OpenFog Consortium [8], we consider Fog Computing as the sum of Cloud and Edge Computing where these two paradigms seamlessly interoperate to provide a platform where both computation and data can be exchanged in both downstream and upstream direction [17] (see Figure 1).

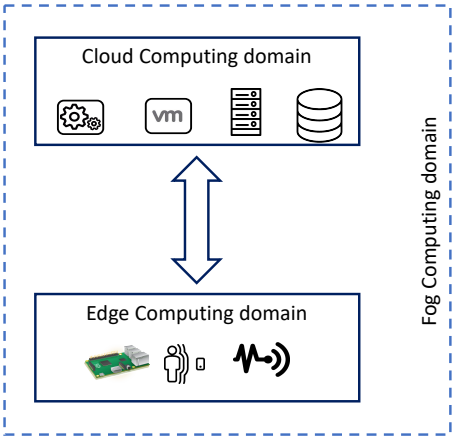

Figure 1 – Fog Computing environment

Based on these definitions, Cloud Computing is mainly related to the core of the network, whereas Edge Computing is focused on supporting the owners of resources through the local 'in-situ' means for collecting and pre-processing data before sending it to cloud resources (for further utilization), thus addressing typical constraints of sensor-to-cloud scenarios like limited bandwidth and strict latency requirements. Furthermore, cloud resources include physical and virtual machines which are capable of processing and storing data. On the other hand, smart devices, wearables, or smartphones belong to the set of edge-located sources. While Cloud Computing is devoted to efficiently managing capabilities and data in the cloud, Edge Computing is focused on providing the means for collecting data

from the environment (which will then be processed by cloud resources) to the owner of the available resources.

Exploiting the Fog Computing paradigm, these two environments seamlessly interoperate to provide a platform where computation and data can be exchanged in both downstream and upstream direction. For instance, when data cannot be moved from the edge to the cloud, e.g., due to privacy issues, then the computation is moved to the edge. Similarly, when there are insufficient resources at the edge, data are moved to the cloud for further processing. The DITAS project wants to provide tools, specifically designed for developers of data-intensive applications, which are able to autonomously decide where to move data and computation resources based on information about the type of the data, the characteristics of applications as well as the available resources at both cloud and edge locations along with application constraints such as the EU GDPR privacy legislation [18].

To achieve this goal, DITAS adopts Service Oriented Computing principles [13] where data used by data-intensive application developers are provided through the Data as a Service (DaaS) paradigm. As shown in Figure 2, we assume the existence of several data providers which take care of optimizing the data collection and provisioning. Data sources could be deployed on the edge (e.g., data coming from sensors) or on the cloud (e.g., data about business transactions). The goal of the data provider is to develop and deploy a DaaS which hides the complexity of managing his/her data sources and to provide them through APIs. Such APIs are used by data consumers that, in our scenario, are represented by data-intensive application developers which process the obtained data in order to generate added-value applications.

Focusing on these two main standpoints, the next paragraphs focus on how Service Oriented Computing can be adopted in a Fog environment.

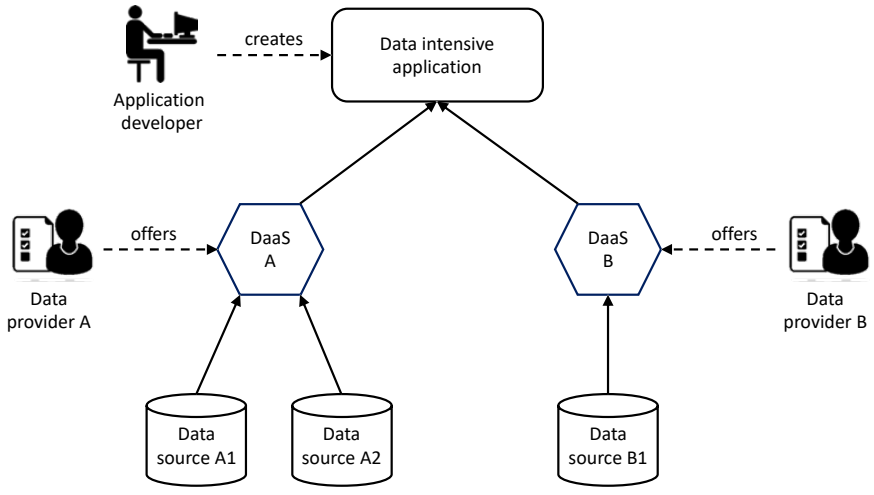

Figure 2 : DITAS Approach

1.2.1 Data provisioning

With respect to the typical data management life-cycle, data providers are usually in charge of collecting, storing, and supporting access to data over which they have complete control. For instance, in IoT scenarios, data are generated at the edge of the network (e.g., sensors, mobile devices) and the data provider has to setup an environment allowing the data consumers to properly access them. This requires moving the data to the cloud, where a seemingly unlimited amount of resources is available for efficient storage and processing of data as well as exposing it through APIs. Although cloud resources ensure high reliability and scalability, network capacity might negatively influence latency when data movements among resources on the cloud and the edge occur, thus, the advantage of the fast processing at the cloud might be wasted and the offered service quality might be negatively affected.

As an example, a data provider could be a highway manager that offers data about the status of the traffic, or time series about the number of vehicles, their type, accidents, etc. Assuming that this information is obtained by reading values of sensors in the field, or based on the information coming from applications, these data are usually moved to the cloud so that they are easily and widely accessible.

While cloud platforms provide solutions where interoperability among different infrastructures is now easy to achieve, we cannot say the same for the edge part. Indeed, agreement about protocols, data formats, and interfaces of smart devices, sensors, and smartphones has not been achieved yet. This results in difficulties when data providers have to deal with heterogeneous devices that need to communicate or, at least, need to send the data to the cloud for further processing.

Focusing on the processing, a significant issue to be taken into account concerns the ever-increasing computational and storage capabilities provided by the resources on the edge. Regarding data storage, once the data are created it is not required to immediately move them to a capable storage in the cloud. Conversely, it is possible to leave the data where they are produced and, in addition, to exploit the computational power to perform some pre-processing directly on the edge.

As a last step in the data management life-cycle, data providers have to make data available to data consumers also considering that they could have different needs and different capabilities. In this light, principles of Service Oriented Computing can be adopted to define the DaaS interfaces that data providers have to propose to allow the final users to properly consume the data. Such interfaces have to consider both functional (i.e., how the data can be accessed) and non-functional (i.e., which is the quality of the data and the service) aspects.

Regarding functional aspects, the offered APIs can adopt the REST [19] architectural style, typical SQL-based interfaces or others. Concerning non-functional aspects, data quality dimensions (e.g., timeliness, accuracy) and service quality dimensions (e.g., response time, latency, data consistency) need to be computed and balanced according to consumer expectations.

1.2.2 Data consumption

From the consumer standpoint, data are accessible by invoking the available DaaS. Assuming that there could be several data providers, each of them in charge of managing different data sources, data consumers have to deal with a plethora of DaaS, each of them providing specific data with different quality of data and service.

Data intensive applications are built on top of these DaaS with the goal of analysing and processing provided data to create added value for the customer. For instance, data coming from the highway manager in the example above can be combined with weather information to analyse the correlation between accidents and severe weather conditions.

A particular aspect considered in DITAS concerns the combination between the Fog Computing paradigm and the Service Oriented Computing approach. Consequently, we assume that while resources required for service provisioning are known in advance and under the control of the service provider, there are additional resources that live on the premises of the customers which will be known by the provider right before starting the data consumption along with the quality of data and service requirements. With respect to the typical approach, these additional resources are not part of the client infrastructure, but they can be included in the set of resources belonging to the service provisioning infrastructure. In this way, the data provider can exploit them to improve the user experience of that specific customer. Among the different opportunities that this scenario could open, in this chapter we focus on the possibility of hosting part of the application logic which composes the data provisioning. Similarly, the resource on the edge of the network can be used to host the data that are considered relevant for the consumer and thus to reduce the latency when users request them.

1.3 DITAS approach

From the perspective of a data provider, exposing data following a DaaS paradigm requires to decide where to store data, in which format, how to satisfy the security constraints, and many other aspects. This situation becomes even more complex when dealing with a heterogeneous system where different devices are involved in the data management. For instance, over time, different versions of smart devices might be used to collect data from the sensors installed in manufacturing plants. This implies that the developers have to manage this heterogeneity as well as to properly distribute the data among edge and cloud, to make applications as efficient as possible.

Moving to the data consumer perspective, the development of data-intensive applications requires to select the proper set of DaaS, considering both functional and non-functional aspects, to connect and start interacting with them, ensuring that the agreed quality of data and service is respected, etc. All of these aspects could distract the attention of the data intensive application developer from the business logic, i.e., to organize the actual data processing.

For this reason, in order to improve the productivity of application developers with the DITAS platform, we aim to offer tools for smart data management trying to hide the complexity related to data retrieval, processing and storage. To this end, data-intensive applications in DITAS are not directly connected to the data

sources containing the necessary data, but the access to these data sources is mediated by a specific component called Virtual Data Container (VDC) (see Figure 3), which represents the concrete element able to provide a DaaS. In more detail, a VDC:

- Provides uniform access to data sources regardless of where they run, i.e., on the edge or on the cloud.
- Embeds a set of data processing techniques able to transform data (e.g., encryption, compression).
- Allows composing these processing techniques in pipelines (inspired by the node-RED programming model) and executing the resulting application.
- Can be easily deployed on resources which can live either on the edge or in the cloud.

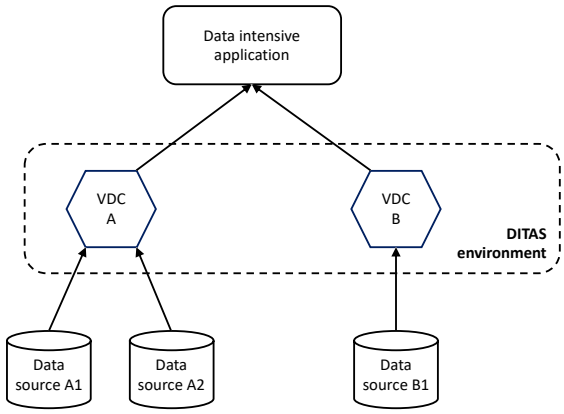

Figure 3 - The role of VDC in the DITAS data management

To manage this indirect access between data sources and the data-intensive application, DITAS distinguishes between the data sources' lifecycle and the data-intensive application's lifecycle. The former defines the relationship between a data source and a VDC, whereas the latter defines the relationship between the data-intensive application and the VDCs which give access to the required data.

For this reason, in the terms adopted in DITAS, a data administrator has a complete knowledge of one or more data sets and is responsible for making them available to applications that might be managed by other developers through a DaaS. On the other hand, a DIA developer defines the functional and non-functional aspects of the data-intensive application and selects the best fitting DaaS needed to compute the analysis. Moreover, the DIA developer is in charge of developing the business logic of the data-intensive application. Exploiting the DITAS-SDK, application developers only focus on data management having the Virtual Data Container (VDC) handle the burden of selecting the best location where to store the data and the most suitable format to satisfy both the functional and non-functional requirements specified by the application designer.

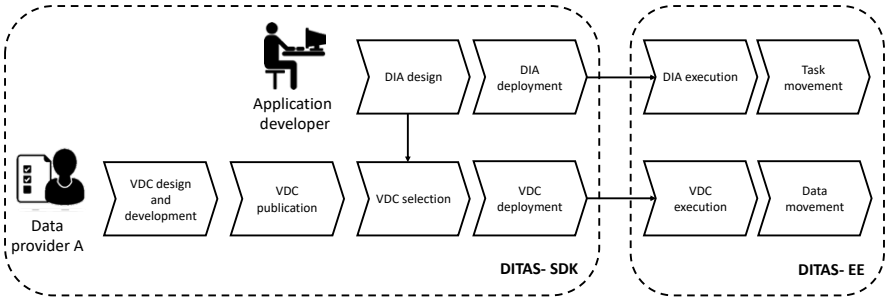

Figure 4: Data-intensive application lifecycle in DITAS

Based on the work done by the data administrator and the DIA developer, DITAS offers two environments able to support the data-intensive application life-cycle. More specifically:

- An SDK assisting both data administrators and DIA application developers.
- An execution environment (EE) where the deployed VDCs and DIA operates.

The next paragraphs detail the steps composing the data-intensive application life-cycle as shown in Figure 4. In particular, we have identified two main phases: the design and development phase, which takes advantage of the DITAS-SDK and the execution phase which relies on the DITAS-EE.

1.3.1 Design and development phase

The first step of the application life-cycle concerns the work performed by the data administrator (a.k.a. data provider, who – based on the managed data sources – creates a VDC Blueprint which specifies the characteristics of a VDC in terms of:

- The exposes data sources
- The exposed APIs
- How the data from the data sources needs to be processed in order to make them available through the API.
- The non-functional properties defining the quality of data and service.
- The components cookbook: a script defining the modules composing the container as well as their deployment.

As DITAS follows the Service Oriented Computing principles, the visibility principle requires to publish a description of a service to make it visible to all the potential users. As a consequence, the data administrator publishes the VDC Blueprint. At this stage, no specific approach for the VDC discovery has been adopted (i.e., centralized registry, distributed publication), as it is an issue to be taken into account for future work.

Once published, the DIA developers come into play. In fact, their role is to search for the data that are relevant for the applications they are developing. As the information included in a VDC Blueprint also concerns functional and non-functional aspects, a DIA developer relies on this information to select the most suitable VDC according to its purposes. It is worth noticing that, based on the nature of the DIA, the developer could select different VDCs referring to different data. A peculiar aspect of the DITAS approach concerns the data utility, that is defined as the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics [20]. In this way, data utility considers both functional (i.e., which data is needed) and non-functional (e.g., the data accuracy, performance) aspects.

Finally, the developer designs and develops the DIA and deploys it on the available resources which can be located on the edge or in the cloud. The initial deployment is the key element in the approach, as in this phase it is required to know which are all the possible resources on which the VDC can be executed. As introduced in Section 1.2, the considered Fog environment implies that DaaS can be provided using resources belonging to both the provider and the consumer. Without loss of generality, we can assume that the provider resources are always in the cloud, while the consumer resources are always on the edge. In this way, a VDC living in the cloud has more capacity and it probably lives close to the data source to which it is connected. Conversely, a VDC living on the edge has the advantage of living closer to the user, thus reducing latency when providing the requested data. Deciding where to deploy the VDC depends on the resources required by the VDC (e.g., it might happen that the amount of resources to process the data before making them available to the user cannot be provided at the edge), the network characteristics (e.g., the connection at the consumer side can support a high-rate transmission), and security (e.g., not all the data can be moved to the consumer side, thus even the processing cannot be placed at the edge).

Once the DIA has been deployed, DITAS supports a flexible execution that initiates data and computation movement when necessary to ensure the fulfillment of the non-functional requirements.

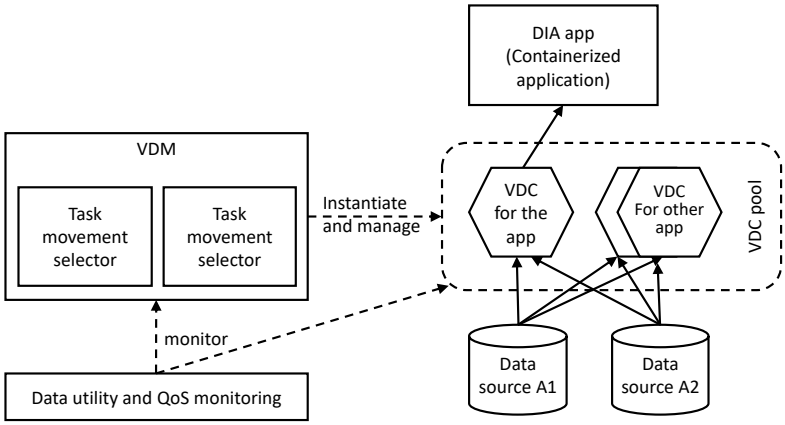

Figure 5: DITAS execution environment

1.3.1 Execution phase

Before introducing the steps of the DIA life-cycle related to the execution, it is worth introducing some of the elements composing the DITAS Execution Engine (DITAS-EE). As reported in Figure 5, the DITAS-EE solution is built on top of a kubernetes [21] cluster. In fact, given a VDC Blueprint, based on the cookbook section, a docker [22] container is generated and deployed. Furthermore, given a VDC Blueprint, many application developers can select it for their own application. As a consequence, the DITAS-EE has to manage several DIAs which operate with different VDCs. Moreover, as the same VDC Blueprint can be adopted in different applications, each of these applications includes instances generated from the VDC Blueprint, thus, they are connected to the same data sources.

To properly manage this concurrent access, given a VDC Blueprint, the DITAS-EE, includes a VDM (Virtual Data Manager) that has to control that the behavior of the different instances of the same VDC Blueprint correctly operate on the data sources and no conflict arises when enacting data and/or computation movements.

Thanks to the abstraction layer provided by the VDC, applications deployed through the platform can access the required data regardless of their nature and location (cloud or edge). Due to the distributed nature of the applications to be managed, to the execution environment being distributed by definition and because of the different computational power offered by the devices, it might happen that only a subset of the modules can be installed on a specific edge device. For this reason, at deployment time, not only the data-intensive application is distributed over the cloud and edge federations, but also the execution environment is properly deployed and configured to support the data and computation movement. The decision on where to locate both the application and the data required by the application itself is taken at design time, but can be updated during the application execution, according to the detected state of the execution environment. Some details about the approach followed to support the data and computation movement are introduced in the next section.

1.4 Data and computation movement in Fog environments

Movement strategies provide solutions for moving data or computation in a Fog environment, taking into consideration all the factors which affect the application execution, data usability and trying to keep the QoS and the data quality at the levels required by the application designer.

Data and computation movement strategies are used to decide where, when, and how to save data - on the cloud or on the edge of the network - and where, when, and how to move tasks composing the application to create a synergy between traditional and cloud approaches able to find a good balance between reliability, security, sustainability, and cost.

The driver for data and computation movement is the evaluation of the Data Utility [20]. When an application is deployed through the DITAS platform, the application designer expresses application requirements about the QoS and quality of data used both to lead the data source selection and to select a proper computation and data location. In the application requirements both hard and soft constraints are expressed. When the evaluation of the data utility does not respect the application designer requests, the VDM will enact the most suitable data and computation movement strategies to balance the posed requirements such as reducing the latency or the data size, ensuring a given accuracy, while maintaining — if requested — privacy and security. Data and computation movements are executed to satisfy all the hard-constraints and, as much as possible, soft-constraints and requirements expressed by the user with the final objective of executing the requested functionality having in mind also the maximization of the user experience. As computation movement requires a dynamic deployment of the data processing tasks, data movement could require only a transformation of the data format (e.g., compression or encryption) or could also affect the quality of the data (e.g., data aggregation).

Data and computation movement are managed over the entire life-cycle of the application, from its deployment to its dismissal. During this time, the application and its data sources are monitored and evaluated in order to satisfy the hard and soft requirements expressed by the application developer. As the decision of when, how, and where data and computation movement must occur depends on the current situation in which the data-intensive application operates, the execution environment includes a distributed monitoring system.

The management of data and computation movement is a life-cycle composed of the steps of the MAPE (Monitor-Analyze-Plan-Execute) control loop:

- Monitor: a DIA is monitored (using the Data Utility and QoS Monitor module) through a set of indicators providing information about both the application behavior and the data source state. This set of indicators can be enriched by a dependency map where such indicators are related to each other, giving a more refined knowledge about the execution environment.
- Analyze: The result of the previous phase is used to compare the current situation with the required data utility values. If the data utility provided to the application does not satisfy the application requirements, an exception is raised. Such an analysis is one of the main tasks to be executed by the VDM.

- Plan: according to the detected violation, some movement actions should be enacted. These actions are in fact data movement and computation movement strategies. To support the planning phase, we will study dependencies among the different data and computation movement techniques in order to identify positive or negative effects to the indicators related to aspects such as reliability, response time, security, and quality of data that need to be measured during the execution of applications. Knowing these relations, it is possible to select the proper action for a violation. The impact on data utility derived from the enactment of the data and computation movement strategies will be analyzed and predicted applying data mining techniques to logs obtained from previous executions. Referring to the DITAS-EE architecture, the task movement selector and the data movement selector are the two modules in charge of the planning.
- Execute: once the strategies have been selected, they can be enacted in order to fix violations. For data movement, specific modules are executed to move the data from the source to the destination, whereas for computation movement, the set of possible actions corresponds to the kubernetes capabilities.

We define a movement strategy as a modification in the placement of a computing task or a set of data. The abstract movement strategy is characterized by one movement action and an object category (e.g., data, computation). The abstract movement strategy is also characterized by the effects on the environment that the enactment of such a strategy will cause. This information can be retrieved by a knowledge base built from the observation of previous enactments using machine learning techniques like reinforcement learning techniques.

In order to enact a movement strategy, it is necessary to specify also the actual object of the movement and its initial and final location. We define this as movement strategy instance. A movement strategy instance may be subject to some constraints given by the object of movement. If we consider data movement, a constraint can be related to privacy and security policies on the moved data. These policies are independent from the context and need to be applied anytime a movement action is applied to an object affected by them.

In order to enact a movement action, several alternative techniques may be used. A movement technique is a building block of a movement strategy. Strategies will combine these building blocks according to the specific needs of an application. Similarly, computation movement techniques will be proposed to define how to distribute the tasks to be executed among the available nodes taking into account the requirements of the task, the resource made available by a node and general requirements at application level. For instance, in case of a movement strategy requiring data aggregation, one of a set of different movement techniques may be selected, each technique implementing a different aggregation algorithm.

The selection of the most suitable movement action for a given context should be driven by the expected utility improvement, which is computed on the basis of the detected violation and the known effects of the strategy on the environment.

Based on this definition of the movement strategy, the primary goal of DITAS is to find a coherent mechanism for deciding which is the best data/computation movement action to be enacted based on the application characteristics, the nature of the data, privacy and security issues, and the application's non-functional requirements. Creating rules and selecting parameters for an automatic selection of data movement actions in different contexts represents now a major challenge. In Figure 6 we show a preliminary model specifying the influences that need to be mapped between several elements of the environment. More specifically, through the analysis of the data collected by the monitoring system some events are raised by violations in the data utility, which compose the context in the top layer of the figure. These events are linked to the goals in the middle layer, which are a representation of the user requirements composing the data utility evaluation. At the bottom level, we represent the available movement strategies together with their effects on the goals.

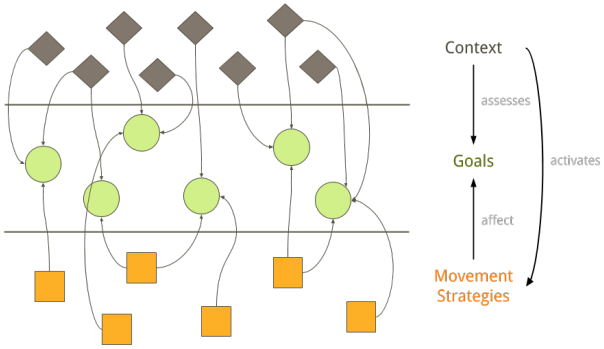

Figure 6 - Influences among context goals and movement strategies

1.5 Related work

Since the 1990s, when interconnection of heterogeneous information systems managed by different owners became easier and when the Web started managing significant amounts of information, the problem of delivering such information has become more and more relevant: the more the data are distributed, the more difficult it is to find the information needed. Thus, tools are required to guide the users in this task. In this scenario, Information Logistics has emerged as a research field for optimizing the data movement, especially in networked organizations [12]. As discussed in [23], Information Logistics can be studied from different perspectives: e.g., how to exploit the data collected and managed inside an organization for changing the strategy of such organizations, how to deliver the right information to decision makers in a process, or how to support supply chain management. In our case, according to the classification proposed in [23], we are interested in user-oriented Information Logistics: i.e., the delivery of information at the right time, the right place, and with the right quality and format to the user [24], thus data movement becomes crucial. As a general framework, there are three sets of data movement techniques.

The first includes techniques that affect neither the content nor the structure of the moved data. In this case, most of the approaches are application-driven, i.e., applications accessing this data heavily influences the adoption of the techniques [25-27].

The second set concerns techniques that do not modify the content of the data but only their format. Lossless, both spatial and temporal compressions as well as data encryption mechanisms fall into this category with the objective of either reducing the amount of data or making the communication secure [29, 30].

Finally, the third set includes techniques that operate on the data transmitted aiming to improve the performance of data movement while maintaining a sufficient level of data quality [24, 31, 32].

To support data movement in a Fog environment which has to deal with heterogeneous devices, data virtualization becomes fundamental. Data virtualization [33] is a data integration technique that provides access to information through a virtualized service layer regardless of data location [34]. Data virtualization [35] allows applications to access data, from a variety of heterogeneous information sources, via a request to a single access point so that it provides a unified, abstracted, real-time, and encapsulated view of information for query purposes and can also transform the information to prepare it for consumption by various applications. Data virtualization solutions add levels of agility (business decision agility, time-to-solution agility or resource agility) that are difficult to achieve with traditional ETL solutions.

Container-based Virtualization is one of the two approaches of lightweight virtualization [36], which minimize the use of processor and memory resources by sharing system calls with the host operating system. Managing data in containers can be done either by keeping the data with the container or by implementing a dedicated data layer [37]. Keeping the data with the container requires the use of techniques that move the data within the container. An example is from ClusterHQ's Flocker [38], which ensures that when an application container moves, the data container moves with it. By implementing a dedicated data layer for the storage container, data services (databases, file systems) can be implemented on more persistent entities such as virtual machines and physical servers.

1.6 Concluding remarks

Fog Computing is an emerging paradigm able to support the development, deployment and execution of distributed applications. This chapter has focused on a specific type of applications: data-intensive applications which have to deal with the gathering, processing, provisioning, and consumption of data. Following the Service Oriented Computing principles, this paper introduces the approach provided by the DITAS project that allows a flexible execution of data-intensive applications. Such flexibility is provided through data and computation movement actions allowing the data-intensive application to change the way in which the processing is distributed at run-time, as well as to optimize how the data are distributed among the different nodes involved in the execution which may belong to edge and cloud environments. Computation movement is ensured by the adoption of a containerized solution which creates self-contained modules, i.e., VDC, that

can be easily moved around the Fog environments. Data movement is supported by the execution of specific actions that are driven by the data utility that defines the relevance of data for the usage context, where the context is defined in terms of the designer's goals and system characteristics.

Since the DITAS platform is still under development, future work will focus on the validation in real applications. In particular, the approach will be tested in a real case study concerning an industry 4.0 scenario. Here data coming from sensors installed on some machinery need to be quickly processed exploiting both the computational power provided at the edge, which ensures reduced latency, and the computational power available in the cloud, which ensures significant scalability. In particular, the impact in terms of overhead introduced the DITAS platform will be analyzed.

References

1. Szalay A and Gray J, (2020), Computing: Science in an exponential world, Nature 440, pp. 413-414, 2006.
2. Bell G, Hey T and Szalay A, (2009), Beyond the Data Deluge, Science 323 (5919), pp. 1297-1298.
3. Turner V, Reinsel D, Gatz J F and Minton S, (2014), The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things, IDC White Paper.
4. Plattner H and Zeier A, (2012), In-Memory Management, Springer-Verlag.
5. Chen C L P and Zhang C, (2014), Data-intensive applications, challenges, techniques and technologies: A survey on Big Data, Information Sciences, Volume 275, pp. 314-347.
6. http://www.apache.org
7. Marz N and Warren W, (2013), Big Data: Principles and best practices of scalable realtime data systems, Manning Publications.
8. OpenFog Consortium Architecture Working Group, (2016), OpenFog Architecture Overview, http://www.openfogconsortium.org/ra, Accessed 31 January 2018
9. Shi W and Dustdar S, (2016), The Promise of Edge Computing, Computer 49(5), pp. 78–81.
10. Bermbach D, Pallas F, Garcıa Pérez D, Plebani P, Anderson M, Kat R and Tai S, (2017), A Research Perspective on Fog Computing, ICSOC-ISYCC Workshop, Malaga.
11. http://www.ditas-project.eu
12. Sandkuhl K, (2008), Information Logistics in Networked Organizations: Selected Concepts and Applications, Springer Berlin Heidelberg.
13. MacKenzie C M, Laskey K, McCabe F, Brown P F and Metz R, (2006), Reference model for service oriented architecture 1.0. Tech. rep., OASIS.
14. http://www.mongodb.com
15. https://cassandra.apache.org
16. Bonomi F, Milito R, Zhu J and Addepalli S, (2012) Fog computing and its role in the internet of things, Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. pp. 13–16.
17. Plebani P, Garcia-Perez D, Anderson M, Bermbach D, Cappiello C, Kat R I, Pallas F, Pernici B, Tai S and Vitali M, (2017), Information Logistics and Fog Computing: The DITAS Approach, In Proceedings of the Forum and Doctoral Consortium Papers Presented at CAISE 2017, Essen, Germany, pp. 129–136. CEUR Vol-1848.

18. http://www.eugdpr.org/
19. Fielding R and Taylor R, (2002), Principled design of the modern web architecture. ACM Trans. Internet Technology, 2(2), pp. 115–150.
20. Cappiello C, Pernici B, Plebani P and Vitali M, (2017), Utility-Driven Data Management for Data-Intensive Applications in Fog Environments. In Advances in Conceptual Modeling, ER Workshops, pp. 216-226.
21. http://www.kubernetes.io
22. http://www.docker.com
23. Michelberger B, Andris R J, Girit H and Mutschler B, (2013), A Literature Survey on Information Logistics. In Proc. of 16th Int.l Conf. of Business Information Systems, BIS 2013, Poznan, Poland, June 19-21, Springer Int.l Publishing.
24. D'Andria F, Field D, Kopaneli A, Kousiouris G, Garcia-Perez D, Pernici B and Plebani P, (2015), Data Movement in the Internet of Things Domain. In Service Oriented and Cloud Computing: 4th European Conf., ESOCC 2015, Taormina, Italy, September 15-17, 2015, Springer Int.l Publishing.
25. Cappiello C, Hinostroza A, Pernici B, Sami M, Henis E, Kat R I, Meth K Z and Mura M, (2011), ADSC: Application-Driven Storage Control for Energy Efficiency, ICT-GLOW 2011, pp. 165-179.
26. Logothetis D, Olston C, Reed B, Webb K C and Yocum K, (2010), Stateful bulk processing for incremental analytics, In Proceedings of the 1st ACM symposium on Cloud computing (SoCC'10), ACM, New York, NY, USA, pp. 51-62.
27. Leyshock P, Maier D and Tufte K, (2014), Minimizing Data Movement through Query Transformation, in Proc of IEEE IEEE International Conference on Big Data (Big Data), 27-30 October 2014, pp. 311-316.
28. Lim H, Herodotou H and Babu S, (2012), Stubby: a transformation-based optimizer for MapReduce workflows, in Proc. VLDB Endowment 5(11) pp. 1196-1207.
29. Srisooksai T, Keamarungsi K, Lamsrichan P and Araki K, (2012), Practical data compression in wireless sensor networks: A survey, Journal of Network and Computer Applications, 35(1), pp. 37-59.
30. Lu P, Zhang L, Liu X, Yao J and Zhu Z, (2015), Highly Efficient Data Migration and Backup for Big Data Applications in Elastic Optical Inter-Data-Center Networks, IEEE Networks, 29(5), pp. 36-42.
31. Ho T T N and Pernici B, (2015), A data-value-driven adaptation framework for energy efficiency for data intensive applications in clouds, in Proc. of IEEE Conference on Technologies for Sustainability (SusTech), pp. 47-52.
32. Peng X and Pernici B, (2016), Correlation-Model-Based Reduction of Monitoring Data in Data Centers, in Proc. of Smart Cities and Green ICT Systems (SMARTGREENS), pp. 135-153.
33. Davis J R and Eve R, (2011), Data Virtualization: Going Beyond Traditional Data Integration to Achieve Business Agility, None Five One Press.
34. Mousa A H, Shiratuddin N, Abu Bakar, M S, (2015), RGMDV: An approach to requirements gathering and the management of data virtualization projects. In Proc. of the 2nd Innovation and Analytics Conference & Exhibition (IACE 2015), vol. 1691, AIP Publishing.
35. Hopkins B, Cullen A, Gilpin M, Evelson B, Leganza G and Cahill M, (2011), Data virtualization reaches the critical mass, Forrester Report.
36. Vaughan-Nichols S J, (2006), New approach to virtualization is a lightweight, Computer, 39(11), pp. 12-14.

37. Sahoo S K and Agrawal G, (2004), Supporting XML based high-level abstractions on HDF5 datasets: a case study in automatic data virtualization, In Languages and Compilers for High Performance Computing, pp. 299-318, Springer Berlin Heidelberg.

38. https://github.com/ClusterHQ/flocker

Index