Fog Computing and Data as a Service: A Goal-Based Modeling Approach to Enable Effective Data Movements

Pierluigi Plebani, Mattia Salnitri(✉), and Monica Vitali

Dipartimento di Elettronica Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci, 32, 20133 Milano, Italy
{pierluigi.plebani,mattia.salnitri,monica.vitali}@polimi.it

Abstract. Data as a Service (DaaS) organizes the data management life-cycle around the Service Oriented Computing principles. Data providers are supposed to take care not only of performing the life-cycle phases, but also of the data movements from where data are generated, to where they are stored, and, finally, consumed. Data movements become more frequent especially in Fog environments, i.e., where data are generated by devices at the edge of the network (e.g., sensors), processed on the cloud, and consumed at the customer premises.

This paper proposes a goal-based modeling approach for enabling effective data movements in Fog environments. The model considers the requirements of several customers to move data at the right time and in the right place, taking into account the heterogeneity of the resources involved in the data management.

Keywords: Data movement · Fog Computing · Decision system
Goal-based model

1 Introduction

The adoption of Service Oriented Architectures [18] has changed the way in which capabilities of an information system are offered and consumed. Although a gap exists between the initial expectations of this research domain and the actual adoption [6] (e.g., about automatic service composition), the Cloud Computing paradigm – where everything is offered as a service – has demonstrated that service orientation has a significant value for both consumers and providers. Nevertheless, limited attention has been paid, so far, to the link between the service oriented paradigm and data management. There are some approaches, under the umbrella of the so-called Data Base as a Service (DBaaS, a.k.a., Cloud Databases), concerning how to provide DBMS functionalities according to the Cloud Computing paradigm [1][1]. Actually, data management's scope is wider

[1] Available commercial solutions are Microsoft Azure SQL Database, Amazon RDS and Oracle Cloud, to name the few.

J. Krogstie and H. A. Reijers (Eds.): CAiSE 2018, LNCS 10816, pp. 203–219, 2018.
https://doi.org/10.1007/978-3-319-91563-0_13

and Data as a Service (DaaS) aims to take care of all the activities needed to collect, process, store, and publish data, which must be accessible on-demand and regardless of the location where they are stored or from where they are requested. Although DaaS providers are mainly focused on taking care of the activities composing the life-cycle, data movement management is also crucial. For instance, in IoT scenarios, data are mainly generated at the edge of the network (e.g., by sensors), but they are usually moved to the cloud, where a theoretically unlimited amount of resources is available to efficiently store and process the data and to make them available to the customers. Indeed, cloud resources ensure high reliability and scalability, but the network capacity might negatively influence the 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 resulting in lower quality of service.

The goal of this paper is to support the data management offered through a DaaS paradigm, by enabling effective data movements able to deliver data at the right time, the right place, and with the right quality and format, to satisfy the customer requirements, as conjectured in a preliminary work [8]. To achieve this goal, the proposed solution is based on two main pillars. Firstly, DaaS provisioning adopts the Fog Computing paradigm, which creates a continuum between the resources living on the edge and on the cloud [19] to exploit the advantages of both: data on the edge are closer to where they are generated or consumed (thus, latency can be reduced), while data on the cloud can have more capacity (thus, processing can be more efficient). Secondly, a goal model is used to design a decision system that includes the customers' requirements, the data movement actions that the environment is able to execute, and the effects of the enactment of a data movement on the satisfaction of the requirements. According to these two pillars, the main contributions of this paper are:

- formalization of data movement actions enriched with data transformations (e.g., aggregation, pseudonymization, encryption) for DaaS provisioning in Fog Computing, considering heterogeneous resources both in the edge and in the cloud belonging to different stakeholders (the data provider and its customers);
- context-based selection of valid movement actions and transformation for each cloud provider based on the definition of the storage resources provided by both the provider and the customers;
- extension of a goal-based modeling language for the definition of customers' non-functional requirements with new concepts such as data movement, data transformations, and a representation of the effect of data movements on goals satisfaction;
- application of a goal-based model at run-time for dynamically selecting a proper movement action to fix the violation of goals.

The rest of the paper is organized as follow. Section 2 motivates the proposed approach by introducing a running example. Section 3 analyzes data movement strategies in Fog environments, while Sect. 4 discusses how to use data movement with the goal-based model. Section 5 demonstrates the scalability of the proposed

approach. Section 6 discusses how the proposed approach is related with the current state of the art, and, finally, Sect. 7 summarizes the proposed approach and identifies future work directions.

2 Motivating Example

Figure 1 shows an example of DaaS providing a data source about traffic information. The data provider manages some sensors and cameras placed on the highways of both Europe and United States to provide real-time information about the traffic. In particular, traffic data from the West Coast go to a *local data storage* where they are temporarily stored. The same occurs to data coming from the highways in the East Coast, as well as from the highways in Europe. Due to the high number of customers, the data provider relies on two cloud sites, i.e., *US site* and *EU site*, where two *Cloud Data Storage*, that must be maintained consistent, are fed by all the local storages, as customers must have visibility of traffic worldwide.

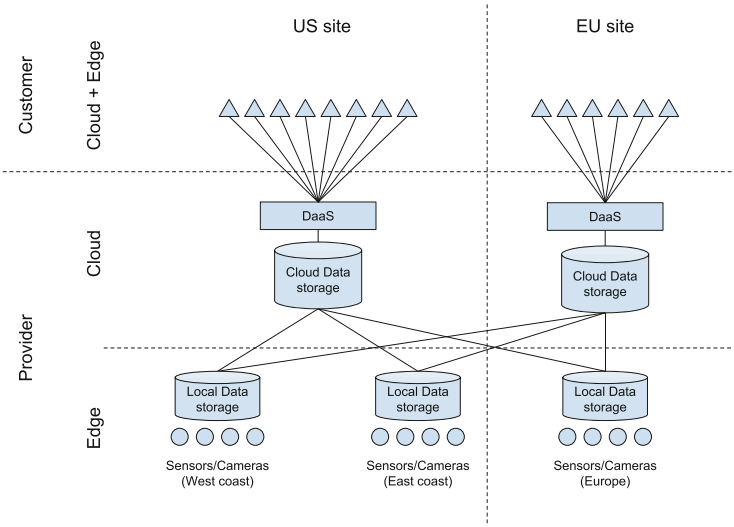

Fig. 1. Traffic information DaaS

Even in this very simple scenario, it is clear how the data continuously move from where data are generated (i.e., cameras and sensors), to the place where data are analyzed, to the data storages where they are saved, to all the final customers' storage devices. To deal with this situation, data providers usually implement solutions aiming to ensure the consistency and the timeliness of the two cloud data storages. In this way, all users see the same data set and the data are provided as soon as possible. Moreover, as the capacity of the local data

storages are limited, and in order to ensure a proper timeliness of data offered to the customers, data are periodically moved from the edge to the cloud. This type of solution is actually reasonable only if all users behave in the same way. Actually, we can assume that the traffic information about EU are mainly used by the European citizens, while the US citizens are more interested on the US traffic data. Furthermore, customers can express different requirements in terms of quality of service (QoS), including latency, timeliness, availability, and so on. As a consequence, we want to support the data providers with a solution that enables effective data movements among the data storages driven by the objective of delivering the right data, at the right time, in the right format.

The proposed solution is based on two pillars: Fog Computing-based infrastructure and a goal-based decision model. The Fog Computing paradigm aims to consider the edge and the cloud resources involved in the service provisioning as a seamless environment. As a consequence, cloud resources do not include only the data provider storages and the scalable applications used to make those data available, but also, if any, the cloud resources used at the client side. Similarly, the edge includes not only the resources that are close to where the data are generated, but also the resources directly managed by the client to store and process the data (e.g., mobile devices).

Referring to our example, we can assume that the following resources are considered:

- data provider edge resources: local storages collecting data about traffic from sensors, positioned in EU and US sites (both in west and east coast);
- data provider cloud resources: data storage located in the cloud containing an aggregation of the information coming from the several edge resources, positioned in EU and US sites;
- customer cloud resources: for customers in EU and US, these are storage resources that belong to the clients but can be used by the data provider to store traffic data useful for the customer application;
- customer edge resources: edge resources with limited storage capabilities that can be used as in the previous case for storing useful information (moving or duplicating data in these additional resources).

We can reasonably assume that storages on the cloud and the edge can have different capabilities: they can store either a complete or a portion of the data set relevant for the specific phase of the data life cycle. Referring to our example, storages at the edge of the provider contain the most recent traffic information about a specific set of highways, while the cloud data storages, due to their capabilities, contain the complete set.

Moving to the second pillar, a goal-based modeling language is used to express the non-functional requirements negotiated between a data customer and a data provider for DaaS provisioning. We chose a goal-based modeling language as this is an intuitive approach for the specification of requirements. The adopted language is the Business Intelligent Model (BIM) [13], used to model trees where each level represents a set of subgoals required to satisfy specific properties of

the data provisioning, and each goal is associated with one or more metrics used to assess the goal satisfaction.

For each customer, a tree is generated to express the QoS agreed with the DaaS provider. An example of such tree is shown in Fig. 2 where there is one top goal, High quality of Service, that represents the main objective to be achieved. This goal is AND-decomposed into three sub-goals, meaning that all sub-goals must be achieved in order to achieve the top-goal. Each sub-goal is OR-decomposed into two sub-goals. The OR-decomposition specifies that at least one of the sub-goals must be achieved, in order to consider the top-goal achieved. For example, Reliable Service is OR-decomposed as Service available and Service scalable: in order to offer a reliable service, the data provider must offer a service with a defined level of availability or a defined level of scalability. A data provider can enrich the model with as many goals as needed to describe the capability of the DaaS service and refine the goals with the needed AND and OR decompositions. The achievement of goals can be defined using metrics that specify properties to be monitored and conditions that determine when goals are satisfied. For example, in Fig. 2, the Fast data process goal is evaluated with the metric Response time and it is considered achieved when its value is lower than 5 s. The overall figure specifies that the data provider and customer agreed in the provisioning of a service that must be reliable, be fast and will maintain data consistency (first AND-decomposition of the top-goal).

The model is the starting point to monitor the customers satisfaction and to detect possible violations of the agreed QoS. It is worth noticing that the proposed approach can be easily extended to include other requirements (e.g., security, privacy, data quality) by adding an additional top goal connected with AND-decompositions to all these non-functional requirements and their sub-trees. Data movement enactment can be used to avoid violations of the tree, as will be described in the next section.

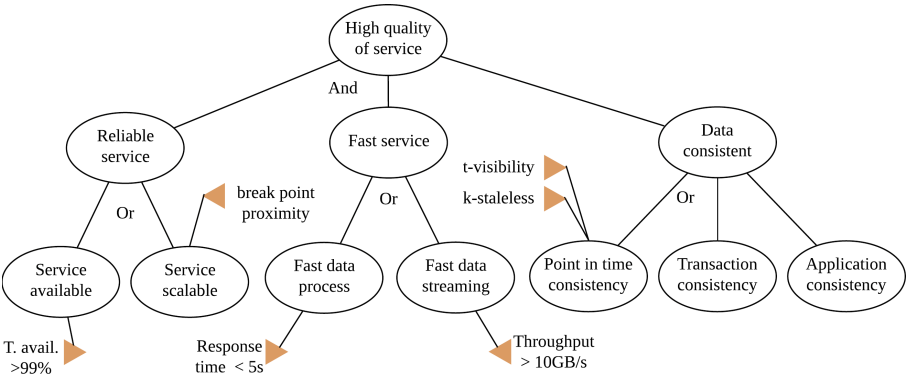

Fig. 2. Example of a goal-based diagram for the specification of QoS requirements

3 Data Movement in Fog Computing

Moving data implies moving portions of the offered data set from a data storage to another in a different location either in the edge or in the cloud. As there might be differences in the way in which the data can be stored, data movement could also require some data transformation. In this section, we classify data movement actions (Sect. 3.1) and discuss their instantiation in a specific context (Sect. 3.2). It is worth noticing that the proposed method is not limited to a specific storage model, even if different models would affect the implementation of the movement actions. The DITAS H2020 project[2] which funded this research is dealing with this issue by providing DaaS independence from the storage model and technology.

3.1 Movement Actions and Transformations

Although the generic term is data movement, the actions to be considered are: the actual movement (M), which consists of deleting the data from the original data storage and move them to a different one, and the duplication (D), where data are copied from a data storage to another while keeping them in the original one. These two classes of actions can be specified at a finer level of detail by considering the location where data are moved. In a heterogeneous environment, where data storages can be placed both in the edge and in the cloud, we have the following scenarios:

- Move/duplicate from cloud to edge (M_{CE}, D_{CE}): data contained in a cloud storage are moved or duplicated in a storage placed in the edge.
- Move/duplicate from edge to cloud (M_{EC}, D_{EC}): data contained in an edge storage are moved or duplicated in a storage placed in cloud.
- Move/duplicate from cloud to cloud (M_{CC}, D_{CC}): data contained in a cloud storage are moved or duplicated in another cloud storage.
- Move/duplicate from edge to edge (M_{EE}, D_{EE}): data contained in an edge storage are moved or duplicated in another edge storage.

Additionally, all the possible classes of movement actions can be subject to an additional data transformations T when data is moved or duplicated from a storage to another. Transformations consist in the manipulation of the content of a data storage and they are requested when the format required by the source and destination data storages are different or when they must be altered for security/privacy reasons. Examples of transformations include:

- aggregation: the content of a data storage is reduced using aggregation operations (e.g., average, maximum, minimum) summarizing several tuples;
- pseudonymization: data are manipulated to substitute identifying fields within a data record with artificial identifiers;
- encryption: the data contained in a data storage are manipulated using encryption algorithms to make them unreadable to unauthorized users.

[2] https://www.ditas-project.eu.

For a given movement action, we can have different sets of transformations, which can be either optional or mandatory. Optional transformations can be executed according to the user requirements, whereas mandatory transformations need to be executed every time data are moved from the data storage. Both movement actions and transformations are associated with metadata defining cost and execution time. These metadata are required to select which action to apply given a specific strategy (cost minimization or time minimization).

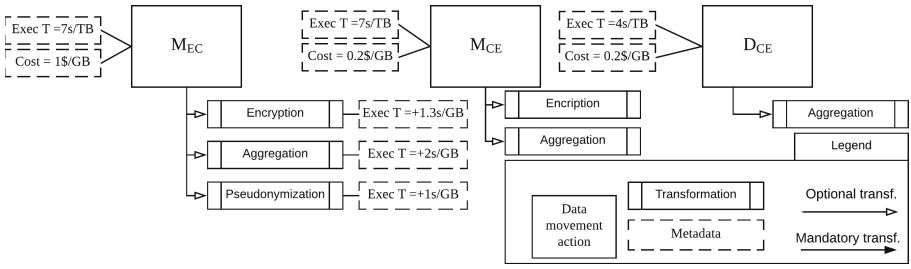

Fig. 3. Data movement actions and transformations example in a fog environment

The BIM modeling language has been extended with the elements described in this section, which create an additional layer. An example is shown in Fig. 3. The extended modeling language allows to specify both data movement actions and transformations. Movement actions are represented as rectangular boxes containing the specification of the movement action (e.g., M_{EC} specifies a movement class from an edge to a cloud storage). Transformations are boxes connected to the movement action to which they are associated. The association can be optional (white arrow) or mandatory (black arrow). Both actions and transformations can be annotated with information about their cost and execution time, represented as dashed rectangular boxes associated to the action or the transformation. For example, in Fig. 3 the data movement action M_{EC} has a cost of 7 s for each Tera Byte moved and of 1$ for each Giga Byte transferred, while its Encryption data transformation add 1.3 s for each Giga Byte transformed.

While the execution time for a transformation can be obtained by testing the algorithms used for this purpose, the execution time of a movement or duplication is affected by the network latency. For this reason, we assume to have some information about the network capacity from which we can derive the needed information. Since we are dealing here with movement classes, without specifying the storage resources and locations involved, the value that we include in the model represents the average behavior of that class of actions. Distinguishing between classes of resources (edge vs cloud) enables us to better predict the metadata associated to a class since, due to the heterogeneity of the resources, edge and cloud storages will behave differently in terms of execution time and cost. This distinction will become implicit when moving from movement classes to movement instances as discussed in Sect. 3.2.

3.2 From Movement Classes to Movement Instances

The data movement classes described in Sect. 3.1 represent all the possible movement actions applicable in a generic context. When instantiating the model in a specific scenario, instances of these classes have to be defined according to the storage resources available. As stated before, storage resources are made available for different customers by the DaaS. Additional data storages can be made available by the customer, near to where the data will be analyzed.

Referring to the running example, at the edge we have three edge data storages: i.e., US West coast S_{wc}, US East coast S_{ec}, and Europe S_{eu}. Two geographical distributed cloud storages are also available in the US and Europe, S_{US} and S_{EU}.

Finally, each customer can provide an edge storage resource S^{E}_{cust} in which the data provider can store a subset of the information contained in an edge or cloud storage.

Distribution of data sets affects data management and in the specific case data movement capabilities of a DaaS provider. Indeed, movement or duplication is possible between two data sets only if their schemas are compatible. A movement action instance should be created for each possible combination of data storages. However, a limitation on movement might derive from security and privacy constraints or from policies defined by the provider. As an example, the provider can decide which classes of movement are allowed and which transformation are mandatory for a specific class. As an example, constraints can be expressed on data localization (e.g., it is not possible to move data from a cloud location in Europe and another in US), or security and privacy constraints (e.g., to be moved from the location in which they are produced, data have to be pseudonymized). The model captures these constraints by removing unauthorized movement instances and properly setting the transformations.

Knowing the available data sets, their location, their relations, and the constraints defined by the provider, it is possible to define which are the data movement actions that can be applied between two data sets. The steps for instantiating the data movement actions from the movement class M_{xy}, with x and y indicating the type of resource (i.e., edge or cloud) are the following:

1. generate a movement instance for each possible combination of resources of type x to resources of type y;
2. remove data movement instances based on the constraints defined by the provider;
3. apply constraints on the transformation and change the required transformation from optional to mandatory (leaving optional the other transformations associated to the action);
4. if additional information is given on resources, mutual location and capabilities, recompute metadata; else inherit metadata from the movement class.

An example on how to get movement instances from the M_{EC} movement class (representing movement from edge to cloud) for our running example is shown in Fig. 4 assuming the provider had specified a localization constraint (e.g., data cannot be moved between Europe and US resources), and a pseudonymization

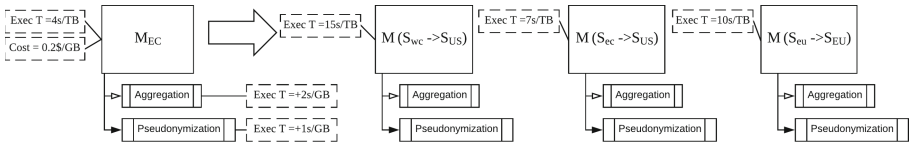

Fig. 4. Data movement actions instances

transformation constraint. In the example all possible combinations of movement instances from edge to cloud are generated, discarding movement actions forbidden due to the localization constraint, and setting mandatory transformations expressed by the provider. As shown, three movement instances are generated from the movement class: from the edge in west and east coast to cloud in the US ($D(S_{wc} \rightarrow S_{US})$ and $D(S_{ec} \rightarrow S_{US})$), and from edge in Europe to cloud in Europe ($D(S_{eu} \rightarrow S_{EU})$). All transformations remain optional with the exception of pseudonymization which changes to mandatory. For the sake of simplicity, inherited metadata are not represented in the figure. Similarly, instances will be generated for movement class D_{EC} and for other allowed movement actions. As can be seen, classes metadata are useful for providing time and cost prediction in unknown contexts. Observations at execution time are used for refining the metadata of both the instance (collecting data about time and cost of the instance execution) and the class (computing the average behavior of all the instances of that class).

4 A Goal-Based Approach for Data Movement Management

The goal model expresses data customer requirements that the provider has to keep satisfied. When a requirement is violated, the model supports the selection of the best data movement action in order to restore goal achievement. To enable this, we need to enrich the goal model, expressing the agreement on QoS between the provider and the customer, taking into account the role of data movements among the data sources available.

4.1 A Goal-Based Modeling Language for Data Movement Management

As the initial goal model only specifies what has been agreed between the provider and the customer in terms of QoS, we propose to extend this model also taking into account the effect of actions, i.e., data movement and transformations over goals. To model the relations between data movement actions or transformations and goals, we use *contribution links*. A *contribution link* specifies that the execution of the action (and transformation) has an impact on the achievement of the goal. Contribution links can have a positive effect (the execution of an action or transformation helps the achievement of a goal) or a negative effect (the execution

of an action or transformation hurts the achievement of the linked goal). Contribution links can be defined by the data provider according to its specific platform and data resources (i.e., the data provider knows that duplicating data between two data sets has a negative effect on the consistency of the data).

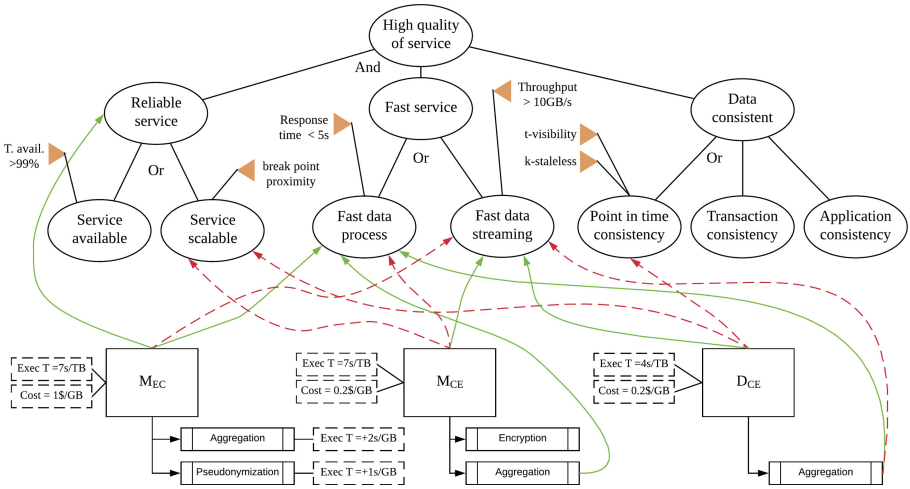

Fig. 5. Example of a diagram for the selection of data movement actions (Color figure online)

Figure 5 shows an example of the goal tree and the data movement actions connected with contribution links, which constitute the proposed extension of the BIM. For example, M_{EC} action is connected with a positive contribution link to Fast data process, since its adoption will improve the metric Response time and, therefore, it will help the achievement of the goal. Similarly, the same action impacts negatively Fast data streaming, since the movement of a data set in the cloud, in this example, will move the data set farther from the sensors that are creating data. For the sake of simplicity, only movement classes are represented in the figure. Movement instances will inherit contribution links from their classes, since the kind of effect (positive or negative) is the same for all the instances. In this work, we represent positive (green) or negative (red) effect with these links. In future work we are planning to assign a quantitative value to these links. In this case, each instance will have a different contribution value. To model this, the data provider and the data customer may customize the contribution links according to expected behavior. Other automatic approaches can be used for setting or refining the contribution links. As an example, in [22], the authors have proposed a reinforcement learning approach to update the knowledge of the effect of a set of actions over a set of goals using a Multi-Armed Bandit inspired algorithm, thus refining the confidence of such link every time the action is enacted.

4.2 Using the Goal-Based Model

Given the requirements of a customer, our extension of BIM is used at *design time* by the data provider, to produce a customized goal-based model, containing the constraints on the requirements relevant for the data customer and the movement actions filtered according to the available resources.

After that, at *runtime*, i.e. when the data customer uses the DaaS, the data provider monitors the goals satisfaction through the associated metrics. When a metric is out of the defined thresholds, an automated *controller*, in charge of managing the DaaS resources, selects to execute the movement action that might improve the current situation. To this aim, the controller analyzes the goal model negotiated with the data customer and selects a set of data movement actions that affect the violated goal.

The goal-based model supports the detection of goal violations. When a metric goes beyond the thresholds defined in a goal model, the linked goal is considered unsatisfied, and the model is analyzed to check, using backward analysis [7,11,21], whether the top goals are satisfied. For example, in Fig. 5, imagine that the goals Service available, Fast data streaming and Transaction consistency are achieved; in this context the top-goal is achieved too. After a while, the Throughput metric goes below 10 GB/s and the goal Fast data streaming becomes unachieved. In this setting, the top goal is not achieved anymore and a data movement action has to be enacted. For further details about goal analyses in BIM please refer to [12].

The action selection is led by the knowledge of the contribution links of both the actions and (if needed) the transformations with the goals. Before this analysis is executed, contribution links on non-leaf nodes are moved to leaf nodes. For example, in Fig. 5, the positive impact propagation of the action Move from edge to Cloud to Reliable service, is propagated to all leaf nodes Service available and Service scalable.

In order to select relevant movement actions, the controller considers all actions and transformations that have a positive impact on the unachieved goal. If a data movement action and one or more of its data transformations have conflicting contribution links on the same goal, we assume the positive contribution link from the transformation is always stronger (it overcomes) than the negative contribution from the data movement action. In the complementary case, the negative transformation link from the transformation nullifies the positive contribution link from the action. For instance, in Fig. 5 the data movement action Move from cloud to edge has a negative impact on Fast data process while its transformation Aggregation has a positive impact, therefore, the combination of the data movement action and transformation have a positive contribution to the goal. The rationale behind this decision lays on the idea that transformations are used to fix the weaknesses of the data movement actions, therefore, even if transformations have negative effects, such effects should never overcome the positive effects of the whole data movement action.

Three possible outcomes are expected: (i) no movement action is selected, meaning that the situation is so critical that none of the possible actions, that can be executed by the data provider, can solve the violation; (ii) one movement

action is selected; (iii) multiple movement actions are selected. We do not investigate further the first option since other research work [4,10] already faced similar problems and can be adopted as solutions for this case. For the third option, the controller considers the metadata associated to movement actions and transformations which express costs and time for enacting the action. Indeed, when several alternative actions are available for fixing a violation, the action selection might be led by the movement strategy selected by the customer. Two main strategies can be expressed: (i) **cost minimization strategy**: the controller selects the action that maximizes the goals satisfaction while minimizing the cost of enactment; (ii) **time minimization strategy**: the controller selects the action that maximizes the goals satisfaction while minimizing the time of enactment. The application of such strategies creates a ranked list of data movement actions.

The decision on which data movement action to apply cannot been taken for a single customer without considering other customers who concurrently access the same data sources. Indeed, the applications sharing the same data sources interfere with each other and a movement action might improve the QoS of one of them while negatively affecting another one. As an example, using the traffic information DaaS, let's consider the situation in which to bring data about traffic in the EU zone nearer to a customer, a movement action moves a subset of them from the cloud storage to a customer's edge storage. This action will improve the Fast service goal of the customer without violating the Data consistent goal. However, another concurrent application using the same data will be affected and its Reliable service goal will be violated.

To avoid interferences, after the selection of a set of candidate actions, the controller might check their effect on the goal trees of other customers. Each action selected is analyzed against all goal trees related to the data source that is being moved. If in at least one goal tree, the action negatively impacts a goal that has no positive contribution links from other movement actions, and therefore no action can be later adopted to restore the goal satisfaction, then the action is moved down in the ranking.

According to where the decision for each customer tree is performed, it is possible to implement the framework in a centralized fashion (global decision and global monitoring) or in a decentralized fashion (distributed monitoring and distributed decision). The first solution is easier in terms of management but the controller is a bottleneck since it has to manage all the customers. The second solution is more scalable but introduce a higher complexity in the coordination of the movement actions.

When the best movement action is selected, the framework will ignore violations that will be signaled in the period immediately after the enactment, in order to avoid oscillations of data sources between two or more locations. More complex mechanisms can be adopted to avoid subtler oscillations of data sources, however this is out of the scope of the paper and will be considered for a future work.

5 Scalability Evaluation

The efficiency of our solution mainly relies on the BIM engine. Thus, an efficient decision making depends on the ability of the BIM engine to produce a result in an acceptable amount of time. Being the complexity of the algorithms for the forward and backward propagation of goal satisfaction implemented in the BIM engine depends on the number of goals, we evaluated the response time of the BIM engine considering a variation of goals from 1 goal to 31 goals[3]. The tests have been executed on a virtual machine with 4 GB of RAM and 2 dedicated 3,3 GHz cores with Linux Ubuntu 16.04 installed. Future work will concentrate on evaluating the effectiveness of the proposed decision making system, as the infrastructure able to move data among cloud and edge storages is under development.

Figure 6 shows results for the backward analysis: on the x-axis there is the number of goals while the y-axis shows the execution time in milliseconds. The dotted line represents the linear regression, which indicates that the execution time increases linearly with the number of nodes. The results, especially in the right side of the chart, may appear distant from the linear regression, however, considering the scale of the chart, the distance can be considered minimal. The maximum execution time with 31 goals is 10 ms, which indicates the software returns the results almost immediately.

The scalability tests for the forward analysis return similar results in terms of execution time. In particular, the linear regression indicates that the execution time remains almost constant. The results are not included in the paper because of lack of space.

Both tests indicate that the backward and forward algorithms are executed in few milliseconds and, therefore, they can be integrated in the software for the decision of the best data movement action at runtime. In order to select the best action, multiple goal trees are evaluated, however, each tree is considered separately, therefore, the analysis can be executed in parallel for each goal tree. Other operations will be executed for the selection of a data movement action, such as the creation of the ranking using the data movement strategies, or the update of the ranking based on the impact of the actions in other goal trees. However, such operations are very fast and they do not impact on the performance of the overall approach.

Since the measured values are very low, they may be influenced by external factors, such as other CPU consuming operations. We solved this possible threat by executing the test 10 times and by excluding the minimum and maximum values.

The scalability test measures the execution time of the forward and backward reasoning software engine, however, other factors should be considered, such as the expressiveness of the modeling language and its usability. Although BIM has been used and validated with many case studies [9,12] we, nevertheless, will perform empirical experiments to evaluate our extension of BIM and the overall framework.

[3] The maximum value corresponds to a binary goal tree with depth equals to 4, a size that, from our experience, we believe is much bigger the goal model that will be used for the purposes of this paper.

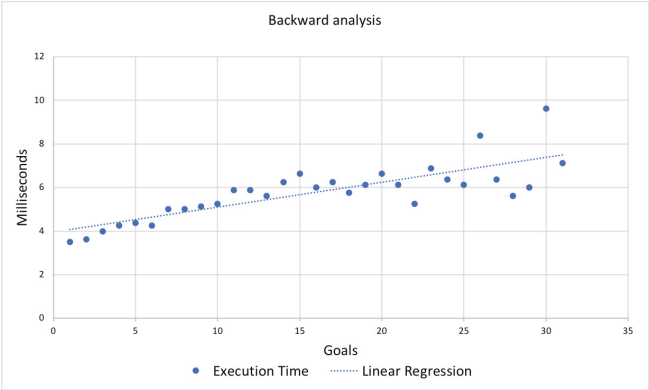

Fig. 6. Scalability tests results

6 Related Work

Initially introduced in the telecommunication domain by Cisco [5], Fog Computing has recently emerged as a hot topic also in the software domain, and especially for data-intensive applications, with the goal of creating a continuum between the resources living on the cloud and the ones living on the edge [19]. The adoption of Fog computing enables an effective data provisioning [20] as data can be moved among different environments in a seamless way. Data movement has been widely studied from different perspectives in the literature to try to reduce the problems arising from the management and use of large quantities of data from different sources and represented in different formats [2,17]. With respect to these approaches, this paper proposes a method for selecting which is the best one to be enacted.

Data movement is also the focus of Content Delivery Networks (CDN) with the aim of geographically distributing a service to ensure high availability and performance. CDNs have been evolving since their first implementation [15] and new solutions also considers deployment on edge facilities[4]. The main limitation of CDNs is that resources used for caching data are predefined and owned and managed by the provider. Moreover, the caching algorithm is only addressing performance and availability optimization of all the users. In the proposed approach, resources are dynamic and can be also controlled by the customers. Also, the data movement policies are driven by the requirements of each specific user and not by a general purpose.

Goal models are used in requirement engineering to specify the objectives of users and applications to be designed. In this paper, we have decided to use BIM [13] as a reference model. However, other approaches are available. In particular, the Goal-oriented Requirement Language (GRL) [3] is a rich modeling language that covers most of the concepts of BIM. However, GRL is a very rich language and may

[4] https://aws.amazon.com/cloudfront/.

prevent a correct usage of the method since many concepts of GRL are not used by our method and may confuse users. We, therefore, decided to extend BIM since it contains the minimal set of concepts needed. Yet, GRL may be considered for a future work.

The tree-like structures of goal models can be used to take decisions on which subset of goals to achieve. A great variety of analyses techniques have been proposed for analyzing goal models for this purpose [14,16]. The satisfaction analyses propagate the satisfaction or denial of goals forward and backward in the goal tree structure. The forward propagation [16] can be used to check alternatives while the backward propagation [7,11,21] can be used to understand what are the consequences of a satisfied or denied goal. Such approaches however, were defined for other domains and, therefore, they do not include concepts needed for this paper.

7 Concluding Remarks

This paper proposes a solution to support data provisioning based on a DaaS paradigm in a Fog Computing environment by enabling an effective data movement among the data storages belonging not only to data providers but also to data consumers. Data movement is driven by a goal-based model capturing the agreement between a provider and its customers and can be used to figure out the most suitable data movement strategy. The model is enriched with data movement actions, defined and classified in the paper. We mapped the effect of actions over goals using contribution links that enable the method to be used at runtime for selecting the best action given a goal violation. At this stage, the validation of the approach is limited to the analysis of scalability, which demonstrated a linear increase of the response time with respect to the increase of the number of goals. Additional experimentations are planned in the near future to also demonstrate that the enactment of the data movement is able to improve the satisfaction of the customer requirements. In future work, we are going to refine the existing model by exploring the outcome of dealing with partially satisfied goals, instead of boolean conditions, also enabling the customer to set weights indicating the most relevant requirements. We are also refining the contribution links associating to them a quantitative value expressing the expected impact of the movement on the indicators associated to the goal, similarly to [22]. We are also going to investigate the implementation of controllers to manage multiple goal-based models for supporting multi-client requirements satisfaction.

Acknowledgments. DITAS project is funded by the European Union's Horizon 2020 research and innovation programme under grant agreement RIA 731945.

References

1. Agrawal, D., Abbadi, A.E., Emekci, F., Metwally, A.: Database management as a service: challenges and opportunities. In: Proceedings of IEEE International Conference on Data Engineering, pp. 1709–1716 (2009)
2. Amarasinghe, S.P., Lam, M.S.: Communication optimization and code generation for distributed memory machines. SIGPLAN Not. **28**(6), 126–138 (1993)
3. Amyot, D., Mussbacher, G.: User requirements notation: the first ten years, the next ten years. JSW **6**(5), 747–768 (2011)
4. Aydemir, F.B., Giorgini, P., Mylopoulos, J.: Multi-objective risk analysis with goal models. In: Proceedings of the Research Challenges in Information Science, pp. 1–10. IEEE (2016)
5. Bonomi, F., Milito, R., Zhu, J., Addepalli, S.: Fog computing and its role in the Internet of Things. In: Proceedings of the MCC Workshop on Mobile Cloud Computing, pp. 13–16 (2012)
6. Bouguettaya, A., et al.: A service computing manifesto: the next 10 years. Commun. ACM **60**(4), 64–72 (2017). http://doi.acm.org/10.1145/2983528
7. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: Non-functional Requirements in Software Engineering. International Series in Software Engineering, vol. 5. Springer, New York (2012). https://doi.org/10.1007/978-1-4615-5269-7
8. D'Andria, F., Field, D., Kopaneli, A., Kousiouris, G., Garcia-Perez, D., Pernici, B., Plebani, P.: Data movement in the Internet of Things domain. In: Dustdar, S., Leymann, F., Villari, M. (eds.) ESOCC 2015. LNCS, vol. 9306, pp. 243–252. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24072-5_17
9. Francesconi, F., Dalpiaz, F., Mylopoulos, J.: Models for strategic planning: applying TBIM to the Montreux Jazz Festival case study. In: 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS), pp. 229–238. IEEE (2015)
10. Gembicki, F., Haimes, Y.: Approach to performance and sensitivity multiobjective optimization: the goal attainment method. IEEE Trans. Autom. control **20**, 769–771 (1975)
11. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Formal reasoning techniques for goal models. J. Data Seman. **1**(1), 1–20 (2003)
12. Horkoff, J., Barone, D., Jiang, L., Yu, E., Amyot, D., Borgida, A., Mylopoulos, J.: Strategic business modeling: representation and reasoning. Softw. Syst. Model. **13**(3), 1015–1041 (2014)
13. Horkoff, J., Borgida, A., Mylopoulos, J., Barone, D., Jiang, L., Yu, E., Amyot, D.: Making data meaningful: the business intelligence model and its formal semantics in description logics. In: Meersman, R., et al. (eds.) OTM 2012. LNCS, vol. 7566, pp. 700–717. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33615-7_17
14. Horkoff, J., Yu, E.: Interactive goal model analysis for early requirements engineering. Requir. Eng. **21**(1), 29–61 (2016)
15. Leighton, F.T., Lewin, D.M.: Content delivery network using edge-of-network servers for providing content delivery to a set of participating content providers, 22 April 2003
16. Letier, E., Van Lamsweerde, A.: Reasoning about partial goal satisfaction for requirements and design engineering. ACM SIGSOFT Soft. Eng. Notes. **29**, 53–62 (2004)
17. Lu, P., Zhang, L., Liu, X., Yao, J., Zhu, Z.: Highly efficient data migration and backup for big data applications in elastic optical inter-data-center networks. IEEE Netw. **29**(5), 36–42 (2015)

18. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference model for service oriented architecture 1.0. Technical report, OASIS (2006)
19. OpenFog Consortium Architecture Working Group: OpenFog Architecture Overview, February 2016. http://www.openfogconsortium.org/ra
20. Plebani, P., Garcia-Perez, D., Anderson, M., Bermbach, D., Cappiello, C., Kat, R.I., Pallas, F., Pernici, B., Tai, S., Vitali, M.: Information logistics and Fog computing: the DITAS approach. In: Proceedings of the Forum and Doctoral Consortium at CAISE 2017, pp. 129–136 (2017)
21. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and minimum-cost satisfiability for goal models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-25975-6_4
22. Vitali, M., Pernici, B., O'Reilly, U.M.: Learning a goal-oriented model for energy efficient adaptive applications in data centers. Inf. Sci. **319**, 152–170 (2015)