applied sciences

MDPI

Article

Pareto-Optimised Fog Storage Services with Novel Service-Level Agreement Specification

Petar Kochovski *,†, Uroš Paščinski †, Vlado Stankovski † and Mojca Ciglarič *,†

Faculty of Computer and Information Science, University of Ljubljana, 1000 Ljubljana, Slovenia; uros.pascinski@fri.uni-lj.si (U.P.); vlado.stankovski@fri.uni-lj.si (V.S.)

* Correspondence: petar.kochovski@fri.uni-lj.si (P.K.); mojca.ciglaric@fri.uni-lj.si (M.C.)

† These authors contributed equally to this work.

Abstract: (1) Background: Cloud storage is often required for successful operation of novel smart applications, relying on data produced by the Internet of Things (IoT) devices. Big Data processing tasks and management operations for such applications require high Quality of Service (QoS) guarantees, requiring an Edge/Fog computing approach. Additionally, users often require specific guarantees in the form of Service Level Agreements (SLAs) for storage services. To address these problems, we propose QoS-enabled Fog Storage Services, implemented as containerised storage services, orchestrated across the Things-to-Cloud computing continuum. (2) Method: The placement of containerised data storage services in the Things-to-Cloud continuum is dynamically decided using a novel Pareto-based decision-making process based on high availability, high throughput, and other QoS demands of the user. The proposed concept is first confirmed via simulation and then tested in a real-world environment. (3) Results: The decision-making mechanism and a novel SLA specification have been successfully implemented and integrated in the DECENTER Fog and Brokerage Platform to complement the orchestration services for storage containers, thus presenting their applicable value. Simulation results as well as practical experimentation in a Europe-wide testbed have shown that the proposed decision-making method can deliver a set of optimal storage nodes, thus meeting the SLA requirements. (4) Conclusion: It is possible to provide new smart applications with the expected SLA guarantees and high QoS for our proposed Fog Storage Services.

Keywords: Fog Storage Services; Quality of Service; Service Level Agreement specification; decision making; Pareto

Citation: Kochovski, P.; Paščinski, U.; Stankovski, V; Ciglarič, M. Pareto-Optimised Fog Storage Services with Novel Service-Level Agreement Specification. *Appl. Sci.* **2022**, *12*, 3308. https://doi.org/10.3390/app12073308

Academic Editor: Chen-Kun Tsung

Received: 21 February 2022
Accepted: 21 March 2022
Published: 24 March 2022

1. Introduction

The Internet of Things (IoT) as a paradigm facilitates various devices to communicate between each other and also integrate them with various smart applications in practically all domains [1–3]. IoT-produced data may be processed and stored starting from the sensors and devices on the field, through different levels of computing infrastructures to the users' smartphones. Hence, smart applications and systems are expected to operate under greatly varying computing and networking conditions. IoT devices generate large quantities of unstructured, semi-structured or structured data that are collected, analysed, and processed by smart applications and usually raise multidimensional concerns (e.g., volume, velocity, veracity, variety) that must be addressed in the course of their processing and storage. It is therefore necessary to provide innovative storage mechanisms to deal with the influx of IoT-produced Big Data.

Depending on the application scenario, the IoT data are collected, analysed, processed and stored at different stages of the Big Data pipeline, starting from the highly distributed and less-powerful microprocessors at the Edge, through heterogeneous infrastructures under various administration domains, up to powerful Cloud data centres. For example, some smart applications that are time-critical by nature are required to rapidly respond

(e.g., early-warning systems) to the users, hence, the storage and processing of data must be performed in close proximity to the sources of the information [4]. Fog storage can be considered as an ability to develop, deploy and orchestrate various database management services, such as databases and file systems closer to the edge of the network with sufficiently high performance towards the end users (e.g., low network latency and high bandwidth). Due to the better performance and limited storage capacities, Fog storage can be also used as a temporary storage until smart applications' data are analysed and processed, whilst data centres in the Cloud may still take the role of dependable long-term data storage.

Different smart application designs can significantly affect various Quality of Service (QoS) database-service-related metrics. In order to address these types of problems, currently various DevOps approaches aim at achieving operationally high QoS by introducing specific orchestration and application control technologies, for example, deployment decision-making algorithms, monitoring tools and similar. With regard to database-service operations, previous studies have investigated a variety of non-functional requirements (NFRs) and their trade-offs [5]. The present study focuses on three NFR attributes, namely, data storage availability, throughput and costs. However, the presented approach is flexible and can be extended towards consideration of multiple NFRs that are related to database-service operations.

For any smart application to reach its business objectives, it is necessary to provide assurances that the application will satisfy the necessary QoS. Providing QoS is an essential task because the database-services potentially have to be deployed across different providers, closer to the IoT devices, that is, at the Edge of the network. Hence, stipulating Service Level Agreements (SLAs) between the software engineers and the infrastructure providers is a requirement that should be addressed in the process of providing guarantees that the required QoS will be maintained above the requested thresholds at all times. At this time, to the best of our knowledge, there is no SLA design for database-services that operates under greatly varying conditions between the Things and the Cloud. The present work focuses on the SLA specifications that are necessary to facilitate optimal QoS negotiated between Fog storage providers and software engineers as storage customers.

This work is set in the context of the European Union–Republic of Korea DECENTER project (https://www.decenter-project.eu/, accessed on 5 January 2022), which investigates a variety of IoT-based smart applications. The project's use cases in areas, such as smart city crossing safety, robotic logistics, smart and safe construction site, and ambient intelligence operate under different conditions; thus, they have different requirements for availability, throughput and costs. The project delivered a Fog Computing platform, where an important feature is the need to provide QoS assurances for databases-service operations. The Kubernetes-based [6] orchestration mechanisms allow to dynamically integrate and orchestrate containers across the Things-to-Cloud computing continuum.

In the domain of Fog computing, costumers do not deal with physical resources. Instead, underlying physical resources enable the existence of virtual resources in a form of virtual machines or containers, which can be developed, set up for execution, started, stopped or even rented by customers, each targeting their specific purpose. Virtual resources usually share a common physical infrastructure, in addition to some overhead introduced by the virtualisation, these virtual resources compete for physical resources. This usually leads to reduced performance compared to the model of having one physical instance for each service. However, the resource sharing model and virtualisation enable lower costs of data centre operation while serving more customers at the same time. Hence, there are two opposing sides, the customers who wish to obtain best possible QoS for their virtual resources, and the service providers who wish to achieve lowest possible operational costs.

Considering that customers pay for the services offered by the providers, they must be provided with assurance on the QoS, similarly to the case of buying a physical product, which includes a warranty. On-demand computing and storage providers usually offer

service guarantees under certain terms, commonly referred to as SLAs. SLAs are agreements between service providers and customers, specifying Service-Level Objectives (SLOs) as rates and conditions about the operability and performance of specific services, usually within certain time intervals, together with SLA violations and SLA exclusions. SLA violations define under what terms an SLA does not meet the agreed objectives and what legitimate actions the customer can take when a violation occurs. In case of SLA violations, the providers usually compensate the affected customers by offering discounted prices in the next billing period. SLA exclusions specify the terms and conditions under which the the providers are exempted from the compensations caused by the violations.

Until now, SLA specifications have been investigated in relation to offered computing infrastructures as a service. To the best of our knowledge, this is the first work that introduces an SLA specification and SLA management approach for Fog Storage Services, which includes, but is not limited to, structured databases and file system operations. In the remainder of this section we summarize the research regarding SLAs, especially targeting SLAs and their specification in Fog Computing. For our work, approaches, solutions and techniques for data management operations close to the Edge of the network are of particular interest. The State-of-the-Art review results are presented in Table 1.

Table 1. State-of-the-Art analysis summary for Fog storage. Works are compared with respect to the QoS attributes, the SLA implementation, the SLA specification, the type of nodes and the utility.

Work	QoS Attributes	SLA Impl.	SLA Spec.	Type of Nodes	Utility
Gill and Buyya, 2019 [7]	Reliability Availability User satisfaction	Yes	No	Cloud	Data storage and processing
Wang et al., 2015 [8]	Cost Reputation Availability Response time	Yes	No	Cloud	Data storage and processing
Serrano et al., 2016 [9]	Cost Reliability Availability Response time	Yes	Yes	Cloud	Data storage and processing
Jrad et al., 2015 [10]	QoS Cost Time complexity Network throughput	Yes	No	Cloud	Data Processing
Yang et al., 2012 [11]	RTFThroughput RTFTickDuration AveragePacketLatency ClientConnectionCount	Yes	No	Cloud	Data storage and processing
García et al., 2014 [12]	Cost Number of failed requests	Yes	Yes	Cloud	Data storage and processing
Yin al., 2020 [13]	Cost I/O performance	Yes	Yes	Cloud	Data storage
Wang et al., 2020 [14]	Storage Throughput	Yes	No	Cloud	Data storage
Conejero et al., 2016 [15]	Cost Energy-efficiency	Yes	No	Cloud	Data processing
Kessaci et al., 2014 [16]	Cost Energy-efficiency	Yes	No	Cloud	N/A

Table 1. *Cont.*

Work	QoS Attributes	SLA Impl.	SLA Spec.	Type of Nodes	Utility
Mayer et al., 2017 [17]	Latency Usage-context	No	No	Fog	Data storage
Gedeon et al., 2018 [18]	Data type Usage-context	No	No	Edge	Data storage
Proposed solution	Cost Availability Throughput	Yes	Yes	Things-to-Cloud computing continuum	Data storage and processing

There are many relevant challenges related to Fog computing [19,20]; however, the focus of our research is narrowed down to assuring high QoS for highly dynamic storage operations, which may be used in the emerging IoT and smart city scenarios (e.g., smart applications executing low-latency operations). Such applications require high QoS for successful real-time processing of Big Data. Virtualised databases in containers can be deployed dynamically on a variety of infrastructures. So far, various studies related to load balancing [21–24], resource allocation and management [25–27], as well as resource provisioning [28–31] have provided means for selecting an optimal IaaS provider.

When data from multiple sources are processed in a shared distributed computing infrastructure, it is necessary to provide QoS guarantees for each data stream. The requirements can be specified in a well-defined SLA. The SLA defines the participants and their responsibilities, terms of the contract, contract's validity period, violation types and respective penalties. In the context of cloud computing, SLA is an officially exchanged document that describes in quantitative or qualitative terms the service that is going to be delivered to the costumer [32]. Due to the lack of studies addressing SLA management for storage in the Fog, the analysis for the state of the art of SLA management is for more generic Cloud computing services.

Gill et al. [7] proposed an SLA management that resides on the side of the service providers. The proposed framework for self-management of Cloud resources is composed of software components that are described in few consecutive studies [33–35]. Their framework provisions and schedules cloud resources, whilst maintaining SLAs and reducing SLA violation rate.

Wang et al. [8] presented an algorithm for selecting optimal web services to build a service composition in geo-distributed Cloud environments. The algorithm chooses the optimal service from a set of functionally equivalent services with respect to SLA constraints and minimises the number of SLA violations.

Serrano et al. [9] proposed a new Cloud model for SLAs and a general control-theoretic approach for QoS-oriented SLA management in which SLA specification are translated into a utility-based objective function. For the selected case studies, the SLA management is illustrated with different QoS aspects of Cloud services, such as performance, dependability and financial energetic costs.

The work of Jrad et al. [10] deals with the implementation of SLAs for computing Clouds and assured the required QoS during complex computations, such as DNA sequencing workflows. They focused on the problem of matching the workflow functional and non-functional SLA requirements to the compute and storage services provisioned by underlying Clouds with different service cost and quality. The study provides a design of an ontological model for a semantic description of the problem and develops a novel utility-based genetic matching algorithm for selecting Cloud services with respect to the user requirements and the properties of the Clouds. The results present the effectiveness of the approach in reducing the total cost and fulfilling the requested QoS even with large-scale service compositions.

Yang et al. [11] proposed a business-oriented federated Cloud computing model where multiple independent infrastructure providers can cooperate seamlessly to provide scalable

infrastructure and QoS-assured hosting services. The business layer in the proposed model provides an enhanced security features and can trigger the on-demand resource provisioning across multiple infrastructure providers, hence helping to maximise the customer satisfaction, business benefits and resources usage.

The work of García et al. [12] proposes the Cloudcompaas platform that allows representation, scheduling and management of Cloud resources. The proposed platform features an extension of the SLA specification, called WS-Agreement, which enables Cloud providers with a generic SLA model to deal with higher-level QoS metrics, closer to end-user perception.

Yin et al. [13] described an SLA-driven deduplication framework for large-scale cloud storage systems. The proposed solution is composed of an SLA notation and deduplication mechanism that facilitates discovery and removal of abundant data in the Cloud, through a performance-cost trade-off to maintain high deduplication quality and meets the necessary SLA.

Wang et al. [14] presented a resource scheduling algorithm for efficient exploitation of cloud storage. The described solution utilizes throughput and storage space as constraints in the decision making for the scheduling process. Its main objective is to reduce the SLA violation rate for cloud storage services.

Conejero et al. [15] studied the trade-off between energy efficiency and performance. They stated that a data centre hosting the Cloud environment is likely to achieve greater energy efficiency (at a reduced cost) compared to a local deployment. With increasing energy prices, it is estimated that a large percentage of operational costs within a Cloud environment can be attributed to energy. This work provides some insight on the relationship between power consumption and QoS related metrics, describing how a combined consideration of these two metrics could be supported for a particular workload.

Kessaci et al. [16] presented an energy-aware multi-start local search algorithm that optimises energy consumption of an OpenNebula-based Cloud. This study proposes the implementation of Pareto Multi-Objective version to deal with both the energy consumption and the SLA. The objective is to find a Pareto trade-off between reducing the energy consumption of the Cloud while preserving the performance of VMs. Mayer et al. [17] proposed a FogStore system, which allows distributing and storing the data of stateful applications in the Fog. Its baseline algorithm utilises Fog-aware requirement (i.e., network latency) and data-context requirements (i.e., geolocation and scenario type) to place the data as close as possible to their clients, and to also place the replicas in close proximity to each other. Similarly, Gadeon et al. [18] presented vStore, which is a framework that facilitates context-aware placement of mobile user data in the Cloud. It uses a rules engine to place and locate data based on its context metadata.

This paper complements the above efforts by delivering a new software engineering technique for dynamic orchestration of containerised databases and file systems across the overall Things-to-Cloud computing continuum. It is supported by DECENTER's novel Kubernetes-based orchestration technology and its Ethereum-based resource brokerage platform. Our solution of Fog Storage Services consists of a novel SLA specification and Pareto front decision-making method that facilitate orchestration of database and file system containers across the Things-to-Cloud computing continuum.

The contributions of this work can be summarised as follows:

- A new Fog Storage Services (an infrastructure layer) for database and file systems storage operations;
- A new SLA specification that is used in the orchestration of Fog storage containers;
- A new Pareto-based decision-making method for the placement of Fog storage containers (e.g., containerised databases and file systems) which is used to provide specific QoS guarantees.

This paper is organised as follows. In Section 1, we presented the background for our research and in the remainder, we review related research in the area of SLAs in Fog computing. Section 2 presents the research methodology. Section 3 explains the infrastructure

for our proposed data storage services. Section 4 introduces SLA from a formal high-level perspective by describing the SLA life-cycle and its relation to Fog storage, followed by detailed SLA design and specification. Section 5 describes the implemented decision-making solution that is based on the selection of non-dominated solutions. Section 6 presents the test environment and the experimental evaluation results. Section 7 concludes the paper and suggests directions for future work.

2. Methodology

Our research methodology follows the main purpose of our research, which is to propose the Fog Storage Service, which is able to provide high QoS guarantees across the Things-to-Cloud computing continuum. To tackle with the QoS requirements, we needed to define a new formalism for SLA specification, which is suitable for use in the orchestration of containerised storage services in the Fog. The last challenge is to define a decision-making process for choosing the optimal location of containers with Fog Storage Services, with regard to the required QoS guarantees. The necessary steps, described in detail in the remainder of this paper, are as follows:

1. Specification of infrastructure that is considered as deployment option in the Things-to-Cloud continuum. This also covered the consideration of deployment requirements for the proposed Fog Storage Services.
2. Definition of SLA specification for the Fog Storage Services, which includes: mapping of the Fog Storage Services' life cycle to the SLA life cycle stages, designing an SLA specification (language) for its use in Fog Storage and selection of relevant SLA parameters for the scenario.
3. Definition of decision-making process for placement of service containers on nodes that guarantee optimal QoS.
4. Implementation and integration of the research and developed decision-making method and SLA specification within the DECENTER Fog and Brokerage Platform.
5. Proof of concept by means of simulation with a set of 100 nodes.
6. Proof of concept in an experimental testbed with a set of 13 nodes and real-time metrics.
7. Interpretation and discussion of results.

3. Dynamic Fog Storage Services

Typical IoT environments are usually composed of many different components, such as: sensors, gateways, computing and storage nodes. Essentially, such environments generate large quantities of unstructured data. However, the type and amount of data differ between use cases. For instance, a smart and safe construction site that facilitates smart, automated, secure and sustainable building process might operate with more than 500,000 data operations per minute [1]; whereas, a scenario for a smart city crossing environment might operate with significantly lower amounts of data [36]. Due to such significant differences between IoT scenarios, the QoS requirements for data storage and processing differ as well, which represents a challenging task for SLA management.

In the investigated case (see Figure 1), a user, who is located in Slovenia, may require a smart application that will operate with data from IoT devices that frequently change their geolocation (e.g., smart vehicles, smartphones or wearable devices). To operationally achieve and maintain high QoS, the data are required to be dynamically stored in close proximity to the data sources. The complete workflow is composed of six consecutive steps, which are as follows: (1) the user defines QoS requirements for the preferred storage, (2) the Kubernetes-based orchestrator allocates available resources (i.e., storage infrastructures), (3) the decision-making mechanism derives a set of Pareto non-dominated storage infrastructures, (4) the user manually selects a Pareto-optimal Fog storage node from the set of non-dominated storage nodes by choosing the preferred trade-off between the desired NFRs and signs the SLA agreement, (5) deploy containerised database and commence persisting data, (6) SLA monitoring and reporting.

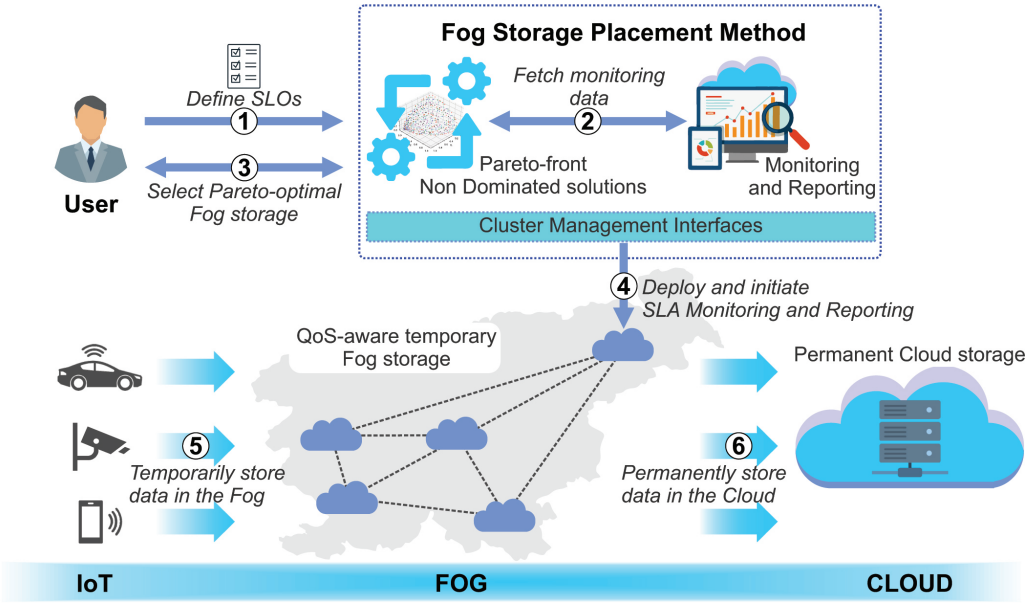

Figure 1. Introducing dynamic Fog Storage Services.

4. SLA Specification for the Fog Storage Services

This section provides detailed insight into the design, specification and implementation of a novel SLA model for our Fog Storage Services. The SLA is used by the users to specify their QoS requirements and by the Pareto-based decision-making process which helps decide an optimal placement of the fog storage containers across the Things-to-Cloud computing continuum. This represents a significant novelty compared to existing SLA specifications.

4.1. SLA Life Cycle

SLA life cycle (see Figure 2) is composed of several phases: Negotiation, Deployment, Monitoring, Violation detection, Reporting and SLA termination [37]. Each phase of the SLA life cycle is explained as follows.

Negotiation: Parties involved in the SLA specify service terms and levels of the provided service on which to agree and may contain also monetary elements. If negotiation adapts to changing QoS demands of the user, the negotiation is dynamic. For example, the user might request for more Fog storage instances. SLA terms might be formalised either by standardised application-agnostic templates, such as WSLA [38] and WS-Agreement [39], or in ad hoc manner that is understood by the enclosed parties. This paper contributes to formalisation of the SLA negotiation by defining custom SLA templates, as described in Section 4.3.

Deployment (Establishment): Service requests from users are provisioned in respective Fog storage nodes. The SLA specification and evaluation define the allocation capacity of resource nodes. It is common to classify SLAs in various classes, such as gold, silver and bronze, to which users are assigned. In our work, SLA is established by applying user-specified QoS requirements and cost constraints in the SLA template, and then invoking the Multi-objective optimisation framework to enforce them.

Monitoring: The deployed services, as well as the resource nodes where the services are run, are being monitored periodically for their health status. In case of substantial service disruptions, the terms of the SLA might get violated. Monitoring can span many dimensions, such as FRs and NFRs of the job, status of resource nodes and network conditions. We use monitoring to estimate performance-related metrics to detect SLA violations.

Violation: The likelihood of a job failing or not meeting its defined service levels represents a violation alert and may be sometimes reported as part of monitored data. Our work focuses on SLA control mechanisms in order to maintain QoS as agreed by the SLA.

Reporting: Provisioning of services and audits is reported in log files that can be securely stored on trusted storage infrastructures, immutable ledgers or similar.

Termination: An SLA may end either when the service is finished or as a response to the violation of one of the parties.

This work is mainly focused on the negotiation and deployment stages, whereas the implementation of the monitoring, violation detection, reporting and termination stages in the context of SLA management are elaborated in our previous work [40].

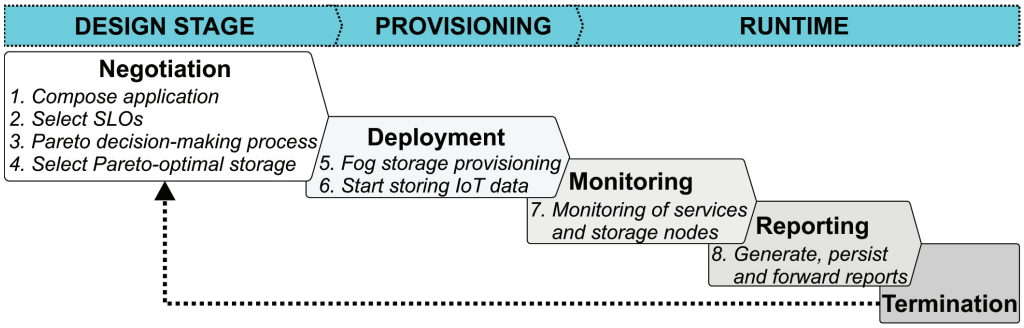

Figure 2. Modelling and implementing the SLA life cycle for Fog storage.

4.2. SLA Specification

In the context of Fog computing, one of the most important components of SLA are the terms of operation, which define SLA parameters of consideration and their respective Service Level Objectives (SLOs). For instance, given that cost is an SLA parameter, then an SLO for it could be

$$\text{cost} \leq \$5/\text{month}.$$

Therefore, SLOs behave as constraints to the agreed SLA parameters. Essentially, SLA parameters can be considered to be the QoS attributes (e.g., availability, throughput and operation cost), whereas software engineers' QoS requirements can be considered as the SLOs. In the scope of this work, the contract is made between two entities: the service customer and the service provider. Essentially, the service customers are the software engineers, whereas the service providers are the Fog node providers. In this paper, it is considered that software engineers that develop applications for various IoT scenarios, have specific QoS requirements and demand their data to be dynamically stored in the Fog. Fog node providers, on the other hand, own the Fog nodes whose computing and storage resources belong to the pool of available sources and are responsible of assuring that their Fog nodes deliver the guaranteed QoS.

SLAs are specified either formally or sometimes via natural language, as a standalone agreement or as a part of general terms of service conditions. For autonomous computing, a representation in natural language is not desired because computers are still not good enough at interpreting natural text in a precise way. Thus, several proposed SLA specification formats and frameworks exist; some of them are standardised.

Bilateral [41] is Java, .NET and Web Service-based protocol, originating from resource reservation in Grids. It supports negotiation, dynamic establishment and management, metrics specification, definition of management actions, reuse, type systems and most of the SLA life cycle steps. Standard Web Services Agreement (WS-Agreement) [39], defined by the Open Grid Forum, offers a framework and protocol, both based on XML syntax. Contrary to the bilateral protocol, WS-Agreement does not support metrics specification, but it does support all the SLA life cycle steps. Web SLA (WSLA) [42], proposed by IBM, is similar and also supports metrics specification. Additionally, a language for Quality of

Service Specification (QML) [43], proposed by HP, also supports type systems. Web Service Offerings Language (WSOL) [44] is not suitable for negotiation. SLAng [45] is feature-wise similar to WS-Agreements, while covering less SLA life cycle steps. Finally, QUO [46] is a CORBA-specific framework with similar features to SLAng.

So far, not much effort has been devoted towards using standards (e.g., WSLA and WS-Agreements) or designing standards for SLA specifications, while most of the existing papers rely on ad hoc representation of SLAs [37].

4.3. SLA Specification Language for Fog Storage

Business requirements addressing QoS can be translated into SLOs, which are formally expressed in SLA specifications. While most existing public Cloud providers specify SLAs in natural language, many attempts have been made to define machine-readable templates. Some of them have been standardised, but their adoption in either academic or industrial environments is still weak. Machine-readable SLA specifications are important for portability reasons, for simplified SLA negotiations, ease of use and control. Furthermore, most specifications are oriented towards IaaS, PaaS and SaaS Cloud types, while Cloud federations and StaaS Cloud types in particular are less commonly addressed. Moreover, only few existing SLA specifications are able to capture dynamic behaviour of Cloud and network services. Nevertheless, in a federated Cloud environments with a large selection of services with common functionality but distinct QoS, SLA negotiation should not only result in satisfiable SLAs but also in the best possible SLA subject to multiple criteria.

To the best of our knowledge, SLA specifications targeting all of the above concerns are non-existing; therefore, in this paper, we propose a new SLA specification language that is based on CSLA [9]. The main features of our SLA specification are domain-specific orientation (i.e., targeting QoS aspects of Fog computing nodes, cost, delivery and distribution), support for dynamic nature of infrastructures, fine-grained validity time-frame specification, infrastructure-based SLA contracting, differentiation of SLA customers into Fog infrastructure owners and infrastructure users, and multi-criteria optimisation specification, which allows for controllable optimisation of objectives within a collection of SLA-compliant solutions.

Figure 3 shows an SLA specification diagram. The core parameters in our SLA specification are:

- *Parties* are entities specified with unique IDs that sign the SLA agreement. Namely, the *Signatory* represents the service provider and service costumer, whilst *Supporting* represents trusted third party entities (e.g., trusted monitoring service provider).
- *Validity* refers to the agreement validity in the specified time period.
- *Fog Service Definition* is a detailed description of a storage node in the Edge-Fog-Cloud. It represents an IaaS, specified with attributes such as operating system, architecture type, amount of resources and cost.
- *Guarantees* are a set of precondition rules and SLOs, where the precondition rules refer to hard constraints (i.e., regional and/or tier restrictions) and SLO refers to a constraint on which parties are obliged to respect, how long an SLO is valid and which service an SLO applies to.
- *Parameters* are defined as a guarantee in an SLO definition. They are defined by a set of metrics, where a metric is the smallest unit that can be measured, and is defined by an arbitrary name, the type of its value, and the unit of measure. As future work, we will investigate on using simple metrics on monitoring directions in order to specify how often a metric should be monitored.
- *Billing* contains the total price of the reservation, which in our case is a single value. However, as future work, we will investigate on the definition of penalties in case an SLA is violated.
- *Terminations* contain a set of policies describing what events could terminate an SLA. In the future, we will also investigate on potential termination policies and the different associated penalties.

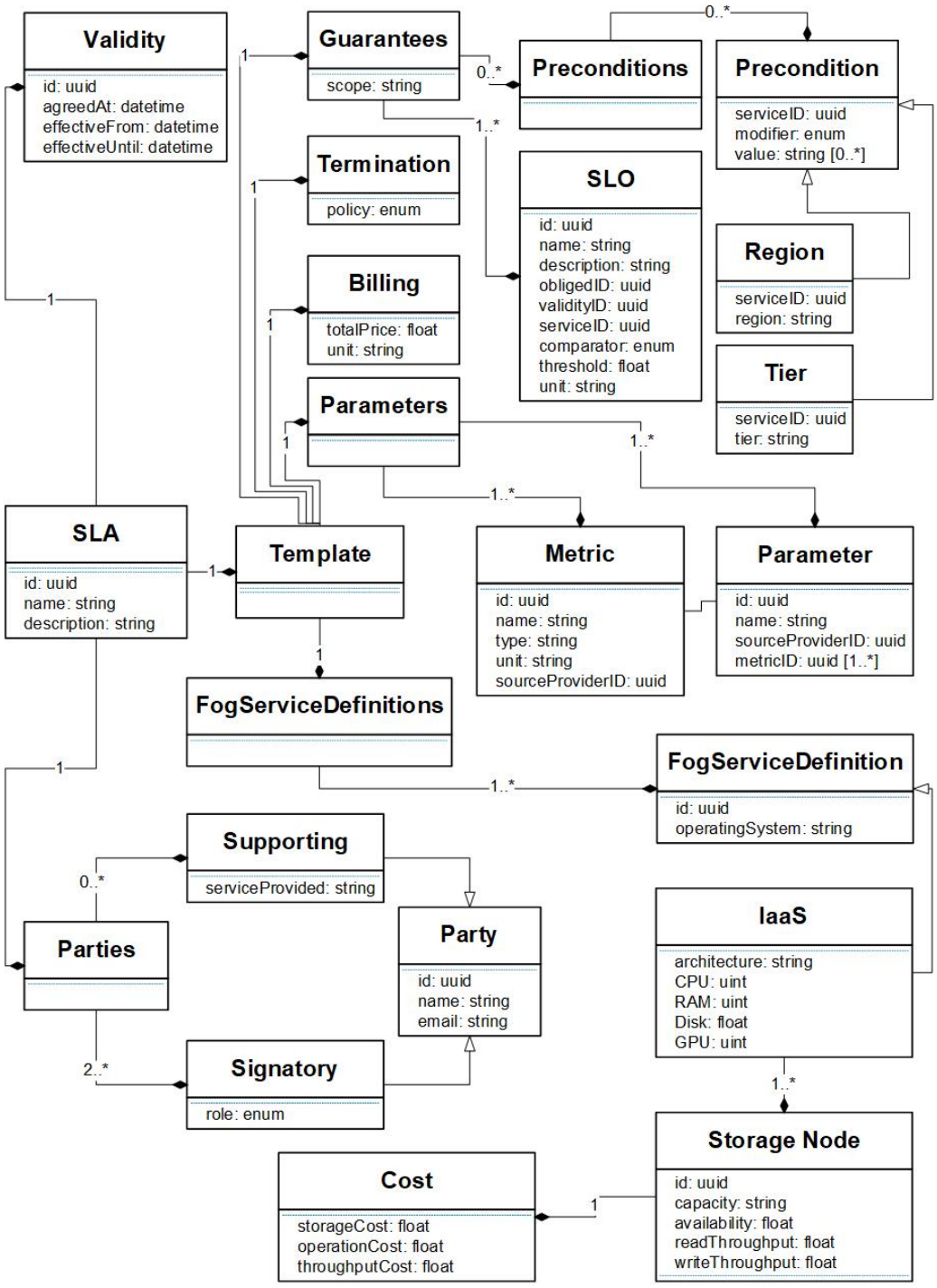

Figure 3. SLA specification diagram for storage in the Fog.

In the SLA specification, section Parties lists exactly two stakeholders, both of which are the holders of the agreement. One of the stakeholder is responsible for providing services under the QoS constraints as specified in the SLA, while the other is a service customer, either storage node owner or storage node user. The owner can also be a user while the reverse is not true. Separation of the customer types allows for fine-grained SLOs and billing specification.

Validity of the SLA specification allows for specification of time frames within which the SLA should be enforced and maintained. The specification offers two different validity time span declarations: (1) an explicit time interval declaration and (2) a lifetime declaration that is valid as long as the storage node has not been deleted or structurally changed.

Validity time spans can also be combined with the following restrictions. Only explicit validity time spans can coexist and are interpreted as a union of time intervals, whereas both the warm up and the lifetime declarations are mutually exclusive from the other two. Explicit interval declarations can also be defined as periodically reoccurring time events. To better understand the applicability of the described SLA specification, we provide open access to an SLA instance (https://bitbucket.org/decenter-ul/sla-specification/src/master/sla-specification, accessed on 5 January 2022) that has been created and used in Section 6.

4.4. SLA Parameters

SLA parameters, as already mentioned, are observable service properties that are used to define the SLOs. The SLA parameters are initially used as input parameters in the Pareto decision-making process to determine optimal storage nodes and then are continuously monitored during the runtime of the application. The selection of SLA parameters in this work is based on in-depth research undertaken by several H2020 research and innovation projects. The baseline information used for the selection of SLA parameters is based on:

- Analysis of the various public Cloud providers and their currently offered SLA contracts (e.g., AWS S3 availability SLA (https://aws.amazon.com/s3/sla/, accessed on 5 January 2022));
- Information gathered from a detailed analysis of the available literature related to SLA contracts for Cloud storage and wider SLAs related to Cloud computing (e.g., WSLA [38], CSLA [9], SYBL [47], Contrail [48] and SLA@SOI [49]);
- Analysis of the monitoring and control possibilities for SLA parameters, which can be implemented as a monitoring system.

In this research, the following SLA parameters are chosen for Fog storage services: availability, throughput and cost. In the following paragraphs, each of the selected parameters is defined and described.

4.4.1. Availability

Service availability a is commonly defined as $a = \frac{u}{u+d}$, where uptime u and downtime d respectively refer to the percentage of time a service is being up and functional or being down, i.e., not functional [50]. For example, many public Cloud service providers expose storage service over an HTTP API and the storage service is considered non-functional if an HTTP request towards the storage returns "HTTP 50x" errors. To the Cloud provider this means that a particular HTTP request cannot be fulfilled with respect to the expected result, where the causes of the error could be many, such as hardware equipment failures (e.g., disks, network switches, cables), software bugs, software updates, misconfiguration, and high requests rate. However, the availability SLA does not account for API service or network failures that might happen before a request reaches the API service.

In our setting, we attempt to take into account any Cloud provider, being private or public, as well as any Fog and Edge provider. Another consideration is the service availability from client's perspective rather than service provider's. This means that any failures in delivering a storage service to a client that could be attributed to the respective service provider, count towards the unavailability. Therefore, SLA for the storage availability also includes uptimes of the corresponding API service, DNS service and the network paths between the client and the service provider. Under a reasonable assumption that the public Internet offers sufficient redundancy in network connectivity, we claim that long-term observations of storage service through periodic monitoring should lead to good estimations of the actual service availability figures.

4.4.2. Throughput

Throughput refers to the volume of application data that are transmitted over a period of time, excluding the bits from protocol overhead and retransmission, such as TCP/IP and

HTTP headers, and acknowledgements. In the context of storage, throughput differs for read and write operations (i.e., download and upload operations).

Within this work, read throughput refers to the expected number of bytes received per unit of time at the client from a storage server, whereas write throughput refers to the expected number of bytes sent per unit of time from the client to the file server. In both cases, only the bytes that comprise the file are counted. In some published works, such as [51], this metric is also known as the goodput (or application throughput).

4.4.3. Cost

Cost is another important SLA parameter. Public Cloud storage providers define several separate cost units: storage cost, operation cost, data transfer cost, storage API access cost, Content Delivery Network cost, early object deletion penalty, transfer acceleration cost and other costs. Deriving cost as a single amount billed per month to a user for storing and using a file requires compounding of the individual cost units with the respective information about the file size, the number of read and write operations, the file life-span and so on. In addition to that, commercial Cloud storage providers usually implement a multi-tariff block pricing model [52], in which the storage cost per stored byte decreases for larger accrued amount of stored data. As a result, modelling cost precisely is overly complex and for our purposes unnecessary, yet we do agree that a different cost model could affect the results presented in this work. The complexity is likely one of the reasons why in the literature cost models vary significantly. For a fine-grained cost model, we refer the reader to the work of Weibel et al. [53]. However, in their work, the pricing model specifically targets the pricing schemata of commercial Clouds or, in more general terms, the schemata that follow such a model.

In the literature, there is no common definition of cost models for Fog environments to be found. In the context of this work, we define cost as a composition of three cost parameters: (1) storage cost c_s, which is the price that the user has to pay to the storage provider for storage of data on the Fog node; (2) data transfer cost c_t, which is the accrued transferred amount of bytes per some time unit; and (3) operation cost c_o, which is the cost of read and write operations (i.e., API call costs) per time unit.

Given a file f, which has to be stored over time t in storage s. Then, the cost of storing c_s can be expressed as

$$c_s(s, f, t) = t \cdot \text{size}(f) \cdot c_s(s), \tag{1}$$

where $\text{size}(f)$ designates the size of file f in GB, time t is measured in months and $c_s(s)$ is the unit storage price of storage s, expressed in a chosen currency per GB per month.

Furthermore, the data transfer cost c_t for transferring file f from storage s to client p is defined as follows:

$$c_t(s, p, f) = \text{size}(f) \cdot c_t(s, p) \tag{2}$$

with $c_t(s, p)$ designating the unit cost per GB for transferring data from storage s to client p. It is a common practice among commercial Cloud providers to only charge for downstream transfers while keeping the upstream transfers free of charge. Thus, without sacrificing the practicality, we follow the same assumption.

The operation cost $c_o(s)$ refers to the cost of a single request for read or write operation. Most often, a batch of bytes is read or written in a single API call.

Finally, by combining the storage cost, data transfer cost and operation cost, the accrued cost of storing and accessing file f in storage s from client p over time t is expressed in the following way:

$$
\begin{aligned}
c(s,p,f,t) = & \\
& c_s(s,f,t) && \text{(cost for storing)} \\
& + n_r \cdot (c_t(s,p,f) + c_o(s)) && \text{(cost for reading)} \\
& + n_w \cdot c_o(s) && \text{(cost for writing)}
\end{aligned}
$$

with n_r read and n_w write API calls. In our case, we assume that the SLA is defined separately for each file (thus we know its size) and that the client p is known in advance. In addition, we assume that the number of reads n_r and writes n_w are also known, or at least, we know their tight upper bounds. In an SLO, this cost parameter is expressed as an accrued cost per month. One may see such a definition of cost also as a monthly budget for storing and accessing file f, although it may happen that the budget is exceeded if the respective SLO is violated.

5. Pareto-Based Decision Making for Placement of Storage Containers

The users of Cloud storage services usually need to consider multiple quality requirements, which often appear to be conflicting. Finding an optimal placement option for the necessary storage containers, that is, the deployment of the storage infrastructure is a complex task. Particularly, altering one parameter may strongly influence the other parameters. For instance, higher storage availability requires increased system redundancy, which could lead to higher operational costs. In order to facilitate the process, the proposed solution implements the concept of domination and Pareto optimality. This concept of domination in the field of multi-criteria decision making has been used and described in prior studies [54,55].

Pareto front is an efficient tool in the decision-making process, because the shape and spread of the Pareto front can provide efficient exploration of the non-dominated solutions' space and investigate the trade-offs.

In this section, we briefly introduce the reader to the multi-objective optimisation problem and to some of the related terms, and then finally express the investigated problem as a multi-objective optimisation problem.

For a set of solutions $X = \{x_1, x_2, \ldots, x_n\}$ and a set of objective functions $f_i : X \to \mathbb{R}$, $i \in \{1, 2, \ldots, m\}$, the efficient set is a set of non-dominated solutions. A solution x_k dominates solution x_ℓ if and only if x_k is not worse than x_ℓ in any of the objectives and is strictly better in at least one objective. The Pareto front is the objective space of non-dominated solutions. The efficient set is the solution to the corresponding multi-objective optimisation problem.

In our problem, we are concerned with storage nodes selection in such a way that the SLOs are respected. This can be effectively translated to the constrained multi-objective optimisation problem, as follows. The set of solutions is represented by storage nodes. The objective functions characterise the storage nodes in terms of the SLA parameters as defined in Section 4.4; thus, the number of objectives is three. What means better depends on each of the SLOs. For the availability and throughput, higher values are better than smaller while for the cost, smaller values are better than higher. By negating the availability and throughput values, the dominance relation translates to minimisation in all the objectives.

The method used to calculate the efficient set (and the Pareto front) is non-dominated sorting, such as the algorithm in NSGA-II [56] used for deriving the Pareto front of one generation of solutions. However, in our setting, no genetic operations of crossover and mutations are required, because all the solutions are known in advance. In other words, as we do not implement data partitioning (i.e., split a single file into data chunks and distribute them into several storages), no service composition is required.

In terms of SLA, all the solutions presented in a Pareto set are feasible, i.e., they comply with the SLOs. Essentially, the Pareto set is important from the user perspective as it provides control over the optimisation step. It enables a user to express the QoS requirements and later on select the preferred storage option among the offered Pareto

optimal storage infrastructures (deployed storage containers). After a point is selected, the point is translated to a decision vector, expressing the relative preference over multiple objectives. The decision vector is written into an SLA specification (see Section 4.3 for the format). Let $x^* \in X^*$ denote a selected point in the Pareto set X^*. Then, the decision vector w is obtained as

$$w = \frac{x^*}{|x^*|},$$

i.e., w equals to the selected point normalised by its second norm.

Let (c, r, p) respectively denote storage cost, reliability and performance objectives. Then, the above formulation allows for simple expression of various preferences, which can be predefined, such as "best storage cost" $(1, 0, 0)$, "best reliability" $(0, 1, 0)$ and "best performance" $(0, 0, 1)$. Whenever the Pareto front is recomputed for a set of Fog storage nodes, both the number of points in the Pareto set as well as their positioning might change. Therefore, it is unlikely that in the newly recomputed Pareto set X^*, there would exist a point $x \in X^+$ that will exactly match the decision vector w, i.e., $\Pr(\exists \lambda \in R : x = \lambda w) = \epsilon$ for some small ϵ. The reason for this is that at least one of the QoS aspects is very likely to change very often with time as, for example, performance. For the same reason, the existing mapping between storage nodes no longer complies with at least one of the SLOs, which causes the re-computation of the Pareto set in the first place.

6. Experimental Evaluation

The recently developed DECENTER Fog and Brokerage Platform is used to orchestrate containers across the Things-to-Cloud computing continuum. The DECENTER Platform is used to evaluate the applicability of the proposed decision-making mechanism alongside the SLA specification. This section presents results from the experimental evaluation.

6.1. Implementation

The DECENTER platform offers means to compose AI-empowered applications that perform AI processing and storage operations over large quantities of data gathered from IoT devices; discover and orchestrate resources across the Edge-Fog-Cloud continuum. It is organised in three functional layers:

- Application Services offer services for the composition of smart applications;
- Brokerage Platform facilitates resource sharing among infrastructure owners. This layer allows brokerage and negotiation of resources that can belong to different administrative domains by implementing blockchain-based mechanisms. These mechanisms are described in our prior works [40,57];
- Fog Platform facilitates resource allocation, monitoring and orchestration in the Edge-Fog-Cloud continuum. Hence, the proposed SLA specification (i.e., SLA management component and Pareto-based decision-making mechanism) is part of this layer. The SLA specification is represented by a JSON structure that is exchanged among the components in this layer. In particular, the SLA management component dynamically certifies SLAs stipulated with other cloud/fog providers when some resources are rented through the Brokerage Platform, by taking as input the needed monitoring data and notifying the Brokerage Platform if any SLA violations occur. Essentially, the Pareto-based decision-making mechanisms complement the set of (re)deployment algorithms that are available in the platform [58,59]. The Pareto-based decision-making mechanisms are implemented as a RESTful Java microservice that are run in a Docker container.

6.2. Experimental Evaluation and Discussion

The process of deriving a Pareto front starts by selecting all available Fog storage nodes that meet the minimal objectives' criteria, requested by the user. Once the nodes that meet the minimal requirements are selected, they are pair-wise compared and a non-dominated sorting is performed. The result provides a subset of optimal Fog storage nodes that have

optimal balance in relation to the three conflicting objectives. Hence, the user is offered a reduced number of storage nodes, which are closest to the chosen requirements.

To prove the feasibility of the described decision-making method, the Pareto front is evaluated in two separate cases: (1) with randomly generated set of 100 nodes with simulated metrics and (2) with a set of 13 storage nodes with real-time metrics (see Table 2). All available storage nodes are situated in the countries within the European Union.

Table 2. Experimental testbed infrastructures with metrics measured from Slovenia.

Infrastructure	Location	Availability [%]	Throughput [Mb/s]	Cost [$]
aws-eu-west-3	France	99.999	12.1	0.306
aws-eu-south-1	Italy	99.999	5.41	0.297
aws-eu-central-1	Germany	99.999	11.81	0.237
gck-eu-west-3	Germany	99.999	21.35	0.200
gck-eu-west-6	Switzerland	99.999	20.73	0.221
azr-de-central	Germany	99.95	18.22	0.021
azr-eu-west	The Netherlands	99.95	19.62	0.020
si-node-0	Slovenia	89.0	12.01	0.001
si-node-1		89.0	17.99	0.015
si-node-2		90.0	10.88	0.050
si-node-3		90.0	20.49	0.022
si-node-4		90.0	15.1	0.001
si-node-5		90.0	12.3	0.001

The metrics for the 13 real storage nodes were gathered from Slovenia in a period of two months prior to the evaluation process.

The Pareto front is derived on the basis of the SLA parameters that are described in Section 4.4. Essentially, the Pareto front is built based on the trade-offs between storage availability, network throughput and cost as conflicting objectives. The result of the decision-making process for the simulated environment is illustrated in Figure 4. From the plots, it is noticeable that in the moment when the user has to choose a storage from 100 available storage nodes, the Pareto front significantly narrows down the choice to the non-dominated nodes by removing up to 85% of the nodes that do not belong to the Pareto front.

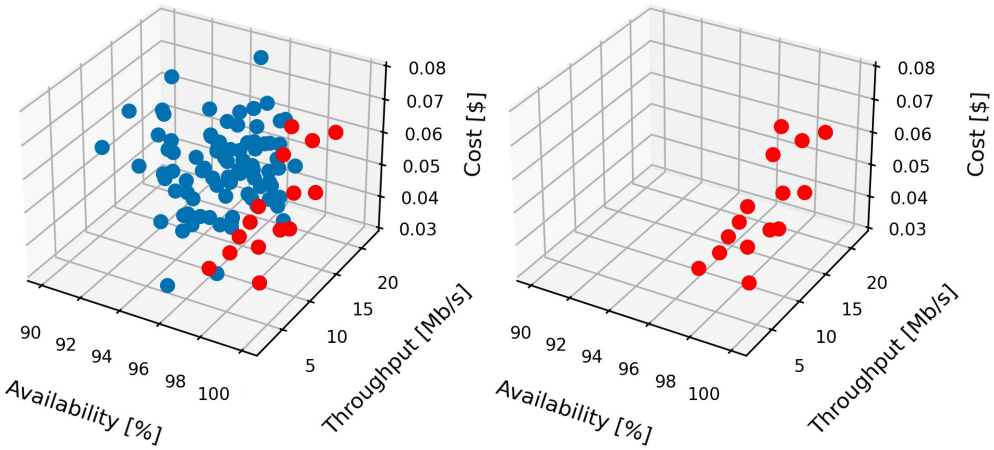

Figure 4. Pareto front derived from the experimental evaluation with simulated input data. The plot on the left illustrates all available Fog storage nodes, whereas the plot on the right illustrates only the non-dominated nodes that form the Pareto front.

Similar to the simulated environment, the Pareto-based decision-making mechanism successfully derived the optimal nodes among the pool of available Fog nodes in the DE-

CENTER Fog and Brokerage Platform, where it narrowed the choice of optimal Fog storage nodes to five possible non-dominated solutions, as illustrated in Figure 5. Upon deriving the Pareto-optimal storage nodes, the user has a significantly easier task selecting the most desired storage infrastructure, which is done by performing a trade-off between the SLOs as illustrated in Figure 6.

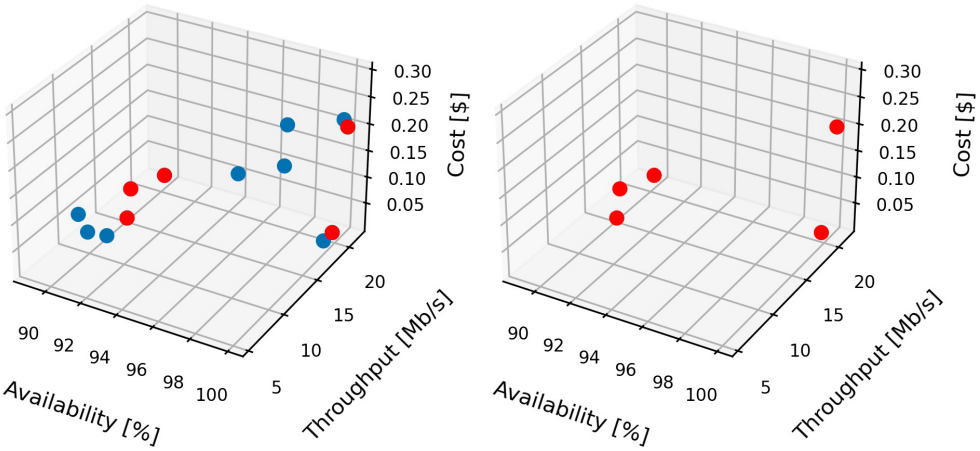

Figure 5. Pareto front derived from the experimental evaluation with existing storage nodes located in Europe. The plot on the left illustrates all available Fog storage nodes, whereas the plot on the right illustrates only the non-dominated nodes that form the Pareto front.

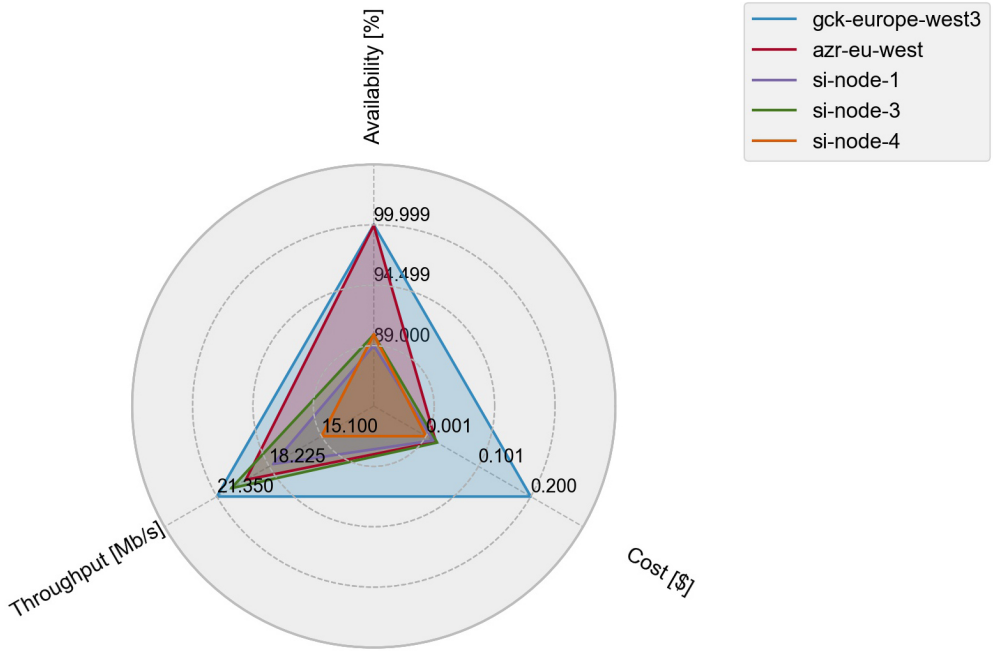

Figure 6. Trade-off between the nodes forming the Pareto front.

In summary, the proposed decision-making method successfully derived optimal storage nodes by utilising the newly proposed SLA specification that consisted of the chosen input parameters for the decision-making process. The proposed method successfully highlights the optimal storage nodes and reduces the number of possible choices to only a few, hence simplifying the selection process for the service user.

7. Conclusions

With the rise of the IoT and the number of smart applications in various domains, there is an increased demand to provide assurances that the applications will satisfy the

requested QoS. Hence, providing means to establish the connection between software engineers and service providers by stipulating SLAs is of high importance.

By introducing a novel SLA specification and Pareto-based decision-making method, this paper provides means to deploy databases and file systems with guarantees for high QoS in the Things-to-Cloud continuum. The newly proposed SLA specification, extended the existing CSLA specification with attributes allowing to define a node in the Things-to-Cloud continuum. Moreover, this paper investigated specific QoS parameters that are particularly important for data management operations. Significant effort was made to precisely define SLA parameters, and chose explicit formulae for their calculation based on lower-level QoS parameter measurements. In the context of this work, three distinctive SLA parameters (i.e., throughput, availability and cost), are used, however, the choice can be extended towards more parameters. In addition, the applicability of the Pareto-based decision-making method for the provisioning stage was evaluated, which showed that the proposed solution was able to select optimal storage nodes and thus remove up to 85% of the initially allocated nodes that were not optimal storage nodes in respect to the defined QoS requirements.

The usability of the proposed solution was validated by using the DECENTER Fog Computing and Brokerage platform and can now be used with the platform. As a result to the successful results, our specification can be used for interoperability purposes with other Cloud storage platforms. In further research, the developed SLA approach will be used in various smart applications involving the IoT and will be further improved with a goal to become a standard.

Author Contributions: Conceptualization, P.K., U.P., V.S. and M.C.; Methodology, P.K., U.P., V.S. and M.C.; Software, P.K. and U.P; Validation, P.K., U.P., V.S. and M.C.; Resources, V.S. and M.C.; Writing—review and editing, P.K., U.P., V.S. and M.C.; Supervision, V.S. and M.C.; Project administration, V.S.; Funding acquisition, V.S. All authors have read and agreed to the published version of the manuscript.

Funding: The research and development reported in this paper have received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement no. 815141 (DECENTER: Decentralised technologies for orchestrated Cloud-to-Edge intelligence) and grant agreement no. 957338 (ONTOCHAIN: Trusted, traceable and transparent ontological knowledge on blockchain).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest

References

1. Kochovski, P.; Stankovski, V. Building applications for smart and safe construction with the DECENTER Fog Computing and Brokerage Platform. *Autom. Constr.* **2021**, *124*, 103562.
2. Li, X.; Lu, R.; Liang, X.; Shen, X.; Chen, J.; Lin, X. Smart community: An internet of things application. *IEEE Commun. Mag.* **2011**, *49*, 68–75.
3. Asghari, P.; Rahmani, A.M.; Javadi, H.H.S. Internet of Things applications: A systematic review. *Comput. Netw.* **2019**, *148*, 241–261.
4. Kochovski, P.; Stankovski, V. Supporting smart construction with dependable edge computing infrastructures and applications. *Autom. Constr.* **2018**, *85*, 182–192.
5. Kochovski, P.; Sakellariou, R.; Bajec, M.; Drobintsev, P.; Stankovski, V. An architecture and stochastic method for database container placement in the edge-fog-cloud continuum. In Proceedings of the 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), Rio de Janeiro, Brazil, 20–24 May 2019; IEEE: Rio de Janeiro, Brazil, 2019; pp. 396–405.
6. The Linux Foundation. Kubernetes Documentation. 2021. Available online: https://kubernetes.io/docs/ (accessed on 5 January 2022).
7. Gill, S.S.; Buyya, R. Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: From fundamental to autonomic offering. *J. Grid Comput.* **2019**, *17*, 385–417.

8. Wang, D.; Yang, Y.; Mi, Z. A genetic-based approach to web service composition in geo-distributed cloud environment. *Comput. Electr. Eng.* **2015**, *43*, 129–141.
9. Serrano, D.; Bouchenak, S.; Kouki, Y.; de Oliveira, F.A., Jr.; Ledoux, T.; Lejeune, J.; Sopena, J.; Arantes, L.; Sens, P. SLA guarantees for cloud services. *Future Gener. Comput. Syst.* **2016**, *54*, 233–246. [CrossRef]
10. Jrad, F.; Tao, J.; Brandic, I.; Streit, A. SLA enactment for large-scale healthcare workflows on multi-Cloud. *Future Gener. Comput. Syst.* **2015**, *43–44*, 135–148. [CrossRef]
11. Yang, X.; Nasser, B.; Surridge, M.; Middleton, S. A business-oriented Cloud federation model for real-time applications. *Future Gener. Comput. Syst.* **2012**, *28*, 1158–1167. doi: 10.1016/j.future.2012.02.005. [CrossRef]
12. García, A.G.; Espert, I.B.; García, V.H. SLA-driven dynamic cloud resource management. *Future Gener. Comput. Syst.* **2014**, *31*, 1–11. [CrossRef]
13. Yin, J.; Tang, Y.; Deng, S.; Zheng, B.; Zomaya, A.Y. MUSE: A multi-tierd and SLA-driven deduplication framework for cloud storage systems. *IEEE Trans. Comput.* **2020**, *70*, 759–774.
14. Wang, Y.; Tao, X.; Zhao, F.; Tian, B.; Vera Venkata Sai, A.M. SLA-aware resource scheduling algorithm for cloud storage. *EURASIP J. Wirel. Commun. Netw.* **2020**, *2020*, 1–10.
15. Conejero, J.; Rana, O.; Burnap, P.; Morgan, J.; Caminero, B.; Carrión, C. Analyzing Hadoop power consumption and impact on application QoS. *Future Gener. Comput. Syst.* **2016**, *55*, 213–223. [CrossRef]
16. Kessaci, Y.; Melab, N.; Talbi, E.G. A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager. *Future Gener. Comput. Syst.* **2014**, *36*, 237–256. [CrossRef]
17. Mayer, R.; Gupta, H.; Saurez, E.; Ramachandran, U. Fogstore: Toward a distributed data store for fog computing. In Proceedings of the 2017 IEEE Fog World Congress (FWC), Santa Clara, CA, USA, 30 October–1 November 2017; IEEE: Santa Clara, CA, USA, 2017; pp. 1–6.
18. Gedeon, J.; Himmelmann, N.; Felka, P.; Herrlich, F.; Stein, M.; Mühlhäuser, M. vStore: A context-aware framework for mobile micro-storage at the edge. In Proceedings of the International Conference on Mobile Computing, Applications, and Services, Shanghai, China, 12 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 165–182.
19. Ficco, M.; Esposito, C.; Xiang, Y.; Palmieri, F. Pseudo-dynamic testing of realistic edge-fog cloud ecosystems. *IEEE Commun. Mag.* **2017**, *55*, 98–104.
20. Singh, J.; Singh, P.; Gill, S.S. Fog computing: A taxonomy, systematic review, current trends and research challenges. *J. Parallel Distrib. Comput.* **2021**, *157*, 56–85.
21. Li, L.E.; Woo, T. Dynamic Load Balancing and Scaling of Allocated Cloud Resources in an Enterprise Network. U.S. Patent App. 12/571,271, 31 March 2011.
22. Chaczko, Z.; Mahadevan, V.; Aslanzadeh, S.; Mcdermid, C. Availability and load balancing in cloud computing. In Proceedings of the International Conference on Computer and Software Modeling, Singapore, 19–23 March 2011; Volume 14.
23. Puthal, D.; Obaidat, M.S.; Nanda, P.; Prasad, M.; Mohanty, S.P.; Zomaya, A.Y. Secure and sustainable load balancing of edge data centers in fog computing. *IEEE Commun. Mag.* **2018**, *56*, 60–65.
24. Talaat, F.M.; Saraya, M.S.; Saleh, A.I.; Ali, H.A.; Ali, S.H. A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 4951–4966.
25. Manvi, S.S.; Shyam, G.K. Resource management for Infrastructure as a Service (IaaS) in cloud computing: A survey. *J. Netw. Comput. Appl.* **2014**, *41*, 424–440.
26. Ni, L.; Zhang, J.; Jiang, C.; Yan, C.; Yu, K. Resource allocation strategy in fog computing based on priced timed petri nets. *IEEE Internet Things J.* **2017**, *4*, 1216–1228.
27. Hong, C.H.; Varghese, B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–37.
28. Chaisiri, S.; Lee, B.S.; Niyato, D. Optimization of resource provisioning cost in cloud computing. *IEEE Trans. Serv. Comput.* **2012**, *5*, 164–177.
29. Singh, S.; Chana, I. Q-aware: Quality of service based cloud resource provisioning. *Comput. Electr. Eng.* **2015**, *47*, 138–160.
30. Skarlat, O.; Schulte, S.; Borkowski, M.; Leitner, P. Resource provisioning for IoT services in the fog. In Proceedings of the 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, 4–6 November 2016; IEEE: Macau, China 2016; pp. 32–39.
31. Naha, R.K.; Garg, S.; Chan, A.; Battula, S.K. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. *Future Gener. Comput. Syst.* **2020**, *104*, 131–141.
32. Buyya, R.; Garg, S.K.; Calheiros, R.N. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In Proceedings of the 2011 International Conference on Cloud and Service Computing, Hong Kong, China, 12–14 December 2014; IEEE: Hong Kong, China, 2014; pp. 1–10.
33. Singh, S.; Chana, I. Resource provisioning and scheduling in clouds: QoS perspective. *J. Supercomput.* **2016**, *72*, 926–960.
34. Gill, S.S.; Chana, I.; Singh, M.; Buyya, R. CHOPPER: An intelligent QoS-aware autonomic resource management approach for cloud computing. *Clust. Comput.* **2018**, *21*, 1203–1241.
35. Singh, S.; Chana, I.; Buyya, R. STAR: SLA-aware autonomic management of cloud resources. *IEEE Trans. Cloud Comput.* **2017**, *8*, 1040–1053.

36. DECENTER Consortium. DECENTER Use Cases: Smart City Crossing Safety. 2021. Available online: https://www.decenter-project.eu/use-cases-2/ (accessed on 5 January 2022).
37. Faniyi, F.; Bahsoon, R. A Systematic Review of Service Level Management in the Cloud. *ACM Comput. Surv.* **2015**, *48*, 1–27. [CrossRef]
38. Ludwig, H.; Keller, A.; King, R.P.; Franck, R. *Web Service Level Agreement (WSLA) Language Specification*; IBM: Armonk, NY, USA, 2003; pp. 815–824.
39. Andrieux, A.; Czajkowski, K.; Keahey, K.; Dan, A.; Keahey, K.; Ludwig, H.; Pruyne, J.; Rofrano, J.; Tuecke, S.; Xu, M. *Web Services Agreement Specification (WS-Agreement)*; 2004.
40. Kochovski, P.; Stankovski, V.; Gec, S.; Faticanti, F.; Savi, M.; Siracusa, D.; Kum, S. Smart Contracts for Service-Level Agreements in Edge-to-Cloud Computing. *J. Grid Comput.* **2020**, *18*, 673–690.
41. Venugopal, S.; Chu, X.; Buyya, R. A Negotiation Mechanism for Advance Resource Reservations Using the Alternate Offers Protocol. In Proceedings of the 2008 16th International Workshop on Quality of Service, Enschede, The Netherlands, 2–4 June 2008; pp. 40–49. [CrossRef]
42. Ludwig, H.; Keller, A.; Dan, A.; King, R. A service level agreement language for dynamic electronic services. In Proceedings of the Fourth IEEE International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002), Newport Beach, CA, USA, 26–28 June 2002; IEEE: Newport Beach, CA, USA, 2002; pp. 25–32. [CrossRef]
43. Frolund, S.; Koistinen, J. *QML: A Language for Quality of Service Specification*; Hewlett-Packard Laboratories: Palo Alto, CA, USA, 1998.
44. Sakellariou, R.; Yarmolenko, V. On the flexibility of WS-agreement for job submission. In Proceedings of the 3rd International Workshop on Middleware for Grid Computing, Grenoble, France, 28 November–2 December 2005; Volume 7, pp. 1–6. http://doi.acm.org/10.1145/1101499.1101511.
45. Skene, J.; Lamanna, D.; Emmerich, W. Precise service level agreements. In Proceedings of the 26th International Conference on Software Engineering, Edinburgh, UK, 23–28 May 2004; IEEE: Edinburgh, UK, 2004; Volume 11, pp. 179–188. [CrossRef]
46. Loyall, J.; Schantz, R.; Zinky, J.; Bakken, D. Specifying and measuring quality of service in distributed object systems. In Proceedings of the First International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC '98), Kyoto, Japan, 20–22 April 1998; IEEE: Kyoto, Japan, 1998; pp. 43–52. [CrossRef]
47. Copil, G.; Moldovan, D.; Truong, H.L.; Dustdar, S. SYBL: An Extensible Language for Controlling Elasticity in Cloud Applications. In Proceedings of the 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Delft, The Netherlands, 13–16 May 2013; pp. 112–119. [CrossRef]
48. Cascella, R.G.; Blasi, L.; Jegou, Y.; Coppola, M.; Morin, C. Contrail: Distributed Application Deployment under SLA in Federated Heterogeneous Clouds. In *The Future Internet*; Galis, A., Gavras, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 91–103.
49. Kearney, K.T.; Torelli, F. The SLA Model. In *Service Level Agreements for Cloud Computing*; Wieder, P., Butler, J.M., Theilmann, W., Yahyapour, R., Eds.; Springer: New York, NY, USA, 2011; pp. 43–67.
50. Nabi, M.; Toeroe, M.; Khendek, F. Availability in the cloud: State of the art. *J. Netw. Comput. Appl.* **2016**, *60*, 54–67.
51. Amini, M.R.; Baidas, M.W. GoodPut, Collision Probability and Network Stability of Energy-Harvesting Cognitive-Radio IoT Networks. *IEEE Trans. Cogn. Commun. Netw.* **2020**, *6*, 1283–1296.
52. Naldi, M.; Mastroeni, L. Cloud storage pricing: A comparison of current practices. In Proceedings of the 2013 International Workshop on Hot Topics in Cloud Services, Prague, Czech Republic, 20–21 April 2013; pp. 27–34.
53. Waibel, P.; Matt, J.; Hochreiner, C.; Skarlat, O.; Hans, R.; Schulte, S. Cost-optimized redundant data storage in the cloud. *Serv. Oriented Comput. Appl.* **2017**, *11*, 411–426. [CrossRef]
54. Kimovski, D.; Saurabh, N.; Stankovski, V.; Prodan, R. Multi-objective middleware for distributed VMI repositories in federated cloud environment. *Scalable Comput. Pract. Exp.* **2016**, *17*, 299–312.
55. Štefanič, P.; Kimovski, D.; Suciu, G.; Stankovski, V. Non-functional requirements optimisation for multi-tier cloud applications: An early warning system case study. In Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; pp. 1–8.
56. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
57. Kochovski, P.; Gec, S.; Stankovski, V.; Bajec, M.; Drobintsev, P.D. Trust management in a blockchain based fog computing platform with trustless smart oracles. *Future Gener. Comput. Syst.* **2019**, *101*, 747–759.
58. Kochovski, P.; Drobintsev, P.D.; Stankovski, V. Formal quality of service assurances, ranking and verification of cloud deployment options with a probabilistic model checking method. *Inf. Softw. Technol.* **2019**, *109*, 14–25.
59. Faticanti, F.; Savi, M.; De Pellegrini, F.; Kochovski, P.; Stankovski, V.; Siracusa, D. Deployment of Application Microservices in Multi-Domain Federated Fog Environments. In Proceedings of the 2020 International Conference on Omni-layer Intelligent Systems (COINS), Barcelona, Spain, 31 August–2 September 2020; IEEE: Barcelona, Spain, 2020; pp. 1–6.