MSSF: User-friendly multi-cloud data dispersal

Rafael M. de O. Libardi[1], Stephan Reiff-Marganiec[2]
Luiz Henrique Nunes[1], Lucas J. Adami[1], Carlos H. G. Ferreira[1] and Julio C. Estrella[1]
[1]University of São Paulo, São Carlos - SP - Brazil
[2]University of Leicester, Leicester - UK
mira@icmc.usp.br, srm13@le.ac.uk, {lhnunes, ljadami, chgferreira, jcezar}@icmc.usp.br

***Abstract*—Using a multi-cloud storage solution requires a user to make complex decisions. Making these decisions can be a problem for regular users who are not familiar with multi-cloud storage. We propose MSSF, a Multi-cloud Storage Selection Framework to automatically select a storage dispersal strategy. MSSF formalises the selection process using a knapsack optimisation problem using integer linear programming along with a rule-based system to select a multi-cloud storage strategy that fits the user needs and requires only simple inputs from the user. Our experiments show the performance and usability aspects of our solution, making it useful in real environments.**

***Keywords*—Dispersion algorithms, Cloud services, Multi-cloud selection, Cloud Storage**

I. Introduction

Multi-cloud environments created a new research area with novel challenges to distributed computing. The Multi-cloud paradigm is a natural evolution of the cloud paradigm, in which instead of dealing with one cloud provider, systems have to integrate and manage several cloud providers to offer a better service to the user. In the storage context, this enables users to experience service improvements over the traditional single cloud paradigm. Using multiple clouds, users can avoid vendor lock-in [1] and improve the security, availability and durability of their storage. The current multi-cloud storage platforms depend on user configured parameters rather than providing an user-friendly and intelligent approach [2].

Several traditional techniques can be applied in this new environment. Encryption [3], [4], compression [5], and information dispersal algorithms [6], [7] are some traditional techniques used to improve the storage quality metrics. However, due to the large amount of available modules and their distinct properties, it is impractical to bother a regular user to choose the desired dispersal configuration.

In this paper, we present the Multi-cloud Storage Selection Framework (MSSF). Very early ideas for the framework have been presented in [2], but this paper significantly extends on these ideas with a complete framework, well-defined selection mechanisms and extensive testing not presented before. MSSF is an automated framework, which implements an internal referral strategy, taking into account several distinct storage properties, such as performance and cost. MSSF is composed of a simple user interface, a repository containing modules and providers' QoS (Quality of Service) properties and an automated selection engine. MSSF is able to select the best storage configuration transparently to the user.

The best Multi-cloud dispersal technique is chosen using storage rules, predefined file types, user parameters and integer optimisation. Storage rules are predefined minimum requirements that the dispersal process needs to achieve, such as security requirements. The file types are based on four main categories classified according to file properties, such as access pattern, MIME type (or file extensions) and file size. User parameters define user preferences, such as security, availability, durability and cost. Finally, when the *framework* has to decide on the best technique, it solves a knapsack optimisation problem created according to user parameters and the file type. A selected dispersal technique is composed of three categories of modules: a dispersal algorithm, an encryption module and a compression module.

Our novel contributions can be summarised as:

- clear identification of requirements to store files using multi-cloud storage environments (based on literature analysis);
- a framework for file dispersal called FlexSky;
- a multi-cloud storage selection process modelled as a knapsack optimisation problem able to select the optimum cloud dispersal strategy according to predefined parameters;
- a rule-based system to adjust and calculate weights for parameters based on rules considering file characteristics; and
- an evaluation showing MSSF performance and usability aspects.

The remainder of this paper is organised as follows: Section II analyses and discuss previous work relevant to this paper. Section III introduces the FlexSky test bed and its main aspects. Section IV review some important parameters and features in multi-cloud storage environments. Section V present the MSSF selection process and architecture. Section VI investigates MSSF performance and usability. Section VII summarises the achievements and identifies future work.

II. Related Works

There are several approaches to aggregate cloud providers. Usually they provide a restricted set of options, forcing the user to make manual choices or restricting the properties available to them. A cloud evaluation and selection review is presented in Alabool and Mahmood [8]. It shows most used selection techniques, metrics and QoS parameters to evaluate, compare and select cloud service providers (CSPs). Rehman et al. [9] studies one of these techniques, called MCDM (Multi-criteria

decision-making). It compares some MCDM techniques and shows that they are suited and effective to select cloud services.

The multi-criteria cloud selection problem was further formalised [10] into a rigorous mathematical model and a simple selection methodology that leverages only cloud providers performance was proposed. A more complex MCDM cloud selection approach is presented in Le et al. [11], which uses fuzziness alongside MCDM techniques to deal with uncertainties and interdependence in the selection process. Rehman et al. [12] proposes a framework to automatically select cloud providers in enterprise environments that consider enterprise policies and user requirement inconsistencies.

We deal with multi cloud storage selection problems, which can be formalised as a web services composition selection problem. Zeng et al. [13] present a generic knapsack optimisation problem formalisation for QoS aware web services composition selection along with several techniques that can be used to solve the problem. One of these solving techniques is based on Integer Linear Programming. Ruiz-Alvarez and Humphrey[14] uses such an approach to select the best cloud storage allocation technique using an integer linear programming solver. It shows through experiments that this technique is reliable and scalable in this context. [15] proposes an XML schema to describe the storage entities and a matching system that uses user requirements to select the best storage strategy to use regarding only cost and performance aspects.

Although these works offer very interesting solutions to the cloud storage selection problem, they are limited to individual cloud provider selection and hence not suited to multi-cloud environments, in which several providers must be selected along with a dispersal strategy that satisfy the overall user requirements. A review leveraging multi-cloud environments is presented in [16]. It also proposes a multi-cloud taxonomy along with issues and requirements to tackle when creating multi-cloud middleware. Their paper offers a good background on multi-cloud service selection, but does not comment on multi-cloud storage.

Several multi-cloud storage architectures were proposed, implemented [17] and reviewed [18]. They usually ask the user to select the desired dispersal strategy, which can be a hard task for a regular user. Scalia [19] proposed to automatically select a multi-cloud storage strategy without user intervention. It uses historical data, file access patterns, file lifetime to continuously optimise data placement at multiple cloud providers. However, it leverages only provider parameters, such as cost, availability and durability lacking other parameters, like security for example. In addition, Scalia lacks support for different dispersal strategies and data transformations. [19] also lacks scalability experiments to check whether this selection approach is suited for real environments.

Our work differs from these related works by providing a selection mechanism that considers distinct dispersal strategies, providers, and transformation modules, such as encryption and compression algorithms. We also combine integer linear programming with a specialist rule-based system to enable a simple and easy to use user interface, increasing the usability of our solution and selection method.

III. FLEXSKY

MSSF was embedded in the FlexSky test bed; however, it could be used in other frameworks. The FlexSky test bed eases the study and comparison of several multi-cloud dispersal features in a real environment using real providers infrastructure. Due to its flexibility to design and execute experiments considering several features and parameters, FlexSky aims to be easily extensible.

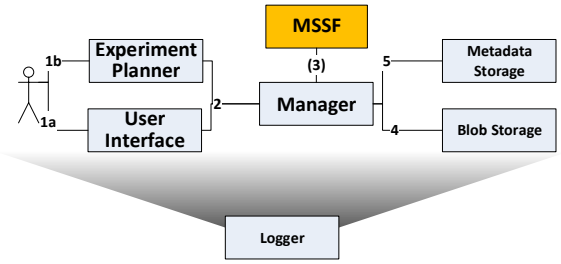

Fig. 1. FlexSky Overall Architecture

Figure 1 describes the overall FlexSky architecture and its integration with MSSF. Currently, MSSF can be used from the experiment planner and from the user interface, which will receive user parameters and choose the best strategy to disperse a file (step 3). Then, it provides to the manager module the strategy selected, including which servers will be used in the dispersal process and the required modules. Finally, the manager executes the blob storage module to store the file (step 4) and provides the metadata to the metadata storage module (step 5).

The MSSF architecture enhances the design concepts of FlexSky and offers the possibility to test several selection methods through a common interface. This increases the flexibility and the possibility to test and compare several multi-cloud storage selection techniques performance aspects, such as the selection time.

IV. FEATURES FOR SELECTING THE DISPERSAL METHOD AND MODULES

A. Availability and Durability

Two important storage quality parameters to study are the availability and durability. The former is defined as "the ratio of (a) the total time a functional unit is capable of being used during a given interval to (b) the length of the interval." [20] and is related to the probability that the system is working when required. Durability, on the other hand, can be defined as "a weaker property that ensures that data can be eventually retrieved, but with some delay if it is currently unavailable" [21]. In summary, availability measure the likelihood to retrieve the data when needed, while durability measures the probability that no data loss occurs over time.

Cloud providers need to provide higher levels of data availability and durability for their users, but it is not rare to find events when those properties are not respected[1]. One way to increase those properties' levels is by using multiple cloud providers' storage using erasure encoding. When using

[1] http://goo.gl/S8H4AV

this method, those properties can be aggregated according to the Equations 1 and 2, which measures availability (A) and durability (D), respectively.

$$A = \sum_{i=n}^{m} \sum_{P \in F_i}^{m} \prod_{i \in P} Ai \prod_{j \in P^c} (1 - Aj) \quad (1)$$

$$D = \sum_{i=n}^{m} \sum_{P \in F_i}^{m} \prod_{i \in P} Di \prod_{j \in P^c} (1 - Dj) \quad (2)$$

These equations are based on the cumulative distributed function of the Poisson Binomial distribution [22]. Equation 1 represent the probability that from m servers with distinct availability levels $A_1, A_2, ..., Ai$, at least n servers will be working when needed to retrieve the file. Equation 2 represent the probability that from m servers with distinct durability levels $D_1, D_2, ..., Di$, at least n servers will not lose the data stored within a time period. F_i is the set of all subsets of i providers that can be selected from the providers set and P^c is the complement of P. It is possible to notice through the equation that the bigger the difference between m and n, the more computationally complex it is to compute in practice (combination problem). However, there are more efficient ways to compute or approximate the result [22].

B. Performance

The performance is an important metric to consider when dealing with multi-cloud dispersal methods because some of these methods are computationally quite expensive. This metric aggregates two sub metrics: provider performance and module performance.

The provider performance is measured using the historical provider throughput and calculating an average and is related to the upload/download speed of the cloud providers. The module performance is measured comparing the modules' throughput using a benchmark on the same testing machine creating a performance linear ranking for distinct file sizes. Both the provider and module measures are then normalised and inserted in the database to be used by the optimisation module.

C. Confidentiality

This metric defines how likely your data can be accessed by unauthorised entities. Several factors can influence the storage confidentiality, such as the dispersal algorithm used, number of parts required to retrieve the file, the encryption level and the providers confidentiality features.

The dispersal algorithm can offer weaker or stronger levels of confidentiality depending on its construction [23], which influences the overall security level. The number of parts required to retrieve the file also influences confidentiality, because the larger the number of parts to retrieve, the less information each part can reveal on its own. The encryption level, which relates to the encryption algorithm and key size, also influences the confidentiality level. On the provider side, the number of confidentiality features, such as credential protection, physical security and communication protection also influence in each providers confidentiality.

There is a trade off between confidentiality and performance. The confidentiality level is indirectly proportional to the performance, because a higher confidentiality level requires stronger security mechanisms which create a performance overhead[24].

D. Storage overhead

The Storage Overhead is a metric that represents the amount of extra space the file takes after being dispersed. It is given by the ratio $\frac{Size_{final}}{Size_{initial}}$. Some dispersal algorithms present distinct characteristics regarding this property due to the algorithms nature. In addition, the compression algorithm and compression level influence this metric, by making the files final size smaller. Although this metric is related to the costs, we decide to separate it, thus allowing to analyse the influence of this property separately from the provider costs.

E. Costs

The costs to store, maintain and retrieve the data is relevant, because these costs can influence and even be the decisive factor to choose the dispersal configuration. The main costs involved in the multi cloud storage are the storage cost and the bandwidth cost.

The storage cost is the price needed to pay to keep the data stored in the providers for a time. The bandwidth cost is the price to transfer data between the provider and the client computer. This cost is categorised into input and output bandwidth cost. The former is the cost to write data to the provider and the latter is the cost to read data from the provider.

The storage and in/out bandwidth costs are present in SLAs (Service Level Agreements) for the major cloud storage providers and are already widely used to price their services offer. As an example, Amazon S3 offers several storage offers with distinct prices and properties[2].

It is important to notice that the dispersal processing costs are not leveraged because all the dispersal process is executed locally and hence assumed free. Although this cost can be easily added to our selection proposal if required. A summary of all QoS parameters and the categories that influence them are shown in Table I.

QoS Parameter	Providers	Dispersal Algorithm	Encryption	Compression
Availability	X	X		
Durability	X	X		
Read Performance	X	X	X	X
Write Performance	X	X	X	X
Confidentiality	X	X	X	X
Storage Overhead		X	X	X
Storage Cost	X			
Bandw. Cost (IN)/ GB	X			
Bandw. Cost (OUT)/ GB	X			

TABLE I. SELECTION CATEGORIES INFLUENCES ON QOS PARAMETERS

F. File classification

A preliminary survey for this research revealed that the users expect different dispersal methods for different file characteristics. We propose a classification based on data access patterns to identify user's required storage needs.

[2]http://aws.amazon.com/pt/s3/pricing/

File type	Access pattern	Parameters to maximise	Parameters to minimise
Working	High reading and High writing	Read and Write Performance, Availability, Durability	Storage Overhead, Bandwidth In/Out Cost, Storage Cost
Active	High reading and Low writing	Read Performance, Availability, Durability, Write Performance	Storage Overhead, Bandwidth In/Out Cost, Storage Cost
Backup-Active	Low reading and High writing	Write Performance, Availability, Durability, Read Performance	Storage Overhead, Bandwidth In Cost, Storage Cost, Bandwidth Out Cost
Backup-Inactive	Low reading and Low writing	Durability, Availability, Read Performance, Write Performance	Storage Overhead, Storage Cost, Bandwidth In/Out Cost

TABLE II. SUMMARY OF FILE TYPES AND CHARACTERISTICS

Our classification is shown in Table II and shows four types of files classified according to their access pattern and the desired optimisation parameters to maximise and minimise according to their weight levels (from most important to less important) in the dispersal process. The file classification can be provided by the user or automatically inferred according to its properties and attributes [25].

V. MULTI CLOUD STORAGE SELECTION FRAMEWORK (MSSF)

Having understood the multitude of parameters needed to enable appropriate storage, we now focus on the developed system, consider how the user interface captures key facts, and then show how these are used to enable dispersal technique selection.

A. User parameters

The user interface of MSSF requests the following set of input parameters from the user, which is deliberately kept as simple as possible: (a) File path, (b) Number of provider failures to support (redundancy level), (c) Confidentiality (None, Confidential, Very Confidential) and (d) File type (according to the file characteristics)

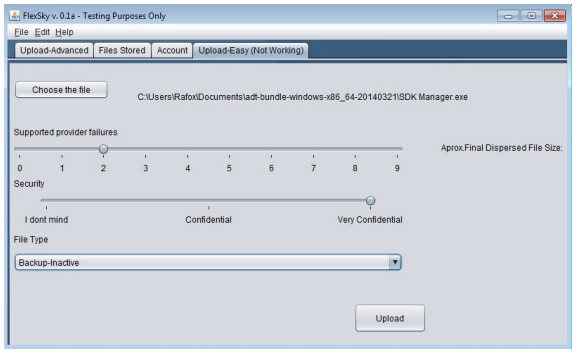

Fig. 2. MSSF user interface

Figure 2 shows the MSSF user interface. MSSF needs three user inputs for each file (in addition to the actual file). The first is the number of provider failures to support, which is the redundancy level desired (if up to this number of provider fails, the user can still access their files). The second is the confidentiality level desired and the third is the type of the file, according to the predefined file types. These sets of inputs will define the corresponding optimisation model parameters, such as the features weight and the constraints to be satisfied.

The predefined confidentiality levels available to the user are *none*, *confidential*, and *maximum* confidentiality. The level *none* means that there is no special requirement regarding the confidentiality of the file. The level *confidential* means that the file is confidential and thus requires a high confidentiality level, but not at too high a cost on other factors. The *maximum* confidentiality requires the best algorithms and providers regarding the confidentiality to store the file, even if this incurs high cost elsewhere.

The user will choose the file type according to the file characteristics represent in Table II. By doing this, the framework can tweak the optimisation parameters to suit the user needs.

B. Optimisation model and engine

The core module of MSSF is its optimisation engine, which is loaded with a knapsack optimisation mixed integer model to select the best combination of modules respecting user requirements and needs.

The mixed integer optimisation approach was chosen because the dispersal process is based on three module categories (dispersal, encryption and compression) and the number of providers, which is limited. For example, it is unlikely that the system will need to choose between thousands of modules and providers in reality. Our model is based on the multi-criteria global optimisation linear programming method presented in [26] and comprises three elements: the objective function, the decision variables and the set of constraints. This model allows to maximise a set of parameters while minimising others, which is particular useful for our mechanism. The chosen method to solve the optimisation problem uses integer linear programming [26], [27]. It requires integer decision variables representing whether a specified module is selected for a category. It also requires some constraint rules to limit the search space and direct the optimisation model. Although it is an NP-hard process, we show with our results that considering current cloud storage environments, it is a feasible solution. The equations that compose this optimisation model are outlined below:

Objective function:

Maximise:

$$\left(\sum_{j=0}^{3}\left(\frac{Q_j^{max}-Q_{k,j}}{Q_j^{max}-Q_j^{min}}*w_j\right)+\sum_{j=4}^{7}\left(\frac{Q_{k,j}-Q_j^{min}}{Q_j^{max}-Q_j^{min}}*w_j\right)\right) \tag{3}$$

Subject to:

$$\sum_{i=0}^{j} w_i = 1, \qquad w_i \in [0,1] \tag{4}$$

$$\sum_{i=1}^{N_{ida}} D_i = 1, \qquad D_i \in \{0,1\} \tag{5}$$

$$\sum_{i=1}^{N_{enc}} E_i \leq 1, \qquad E_i \in \{0,1\} \tag{6}$$

$$\sum_{i=1}^{N_{comp}} C_i \leq 1, \qquad C_i \in \{0,1\} \tag{7}$$

$$Q_{k,j} \geq Q_j^{user} \tag{8}$$

$$\sum_{i=1}^{N_{prov}} P_i = P_{req}, \qquad P_i \in \{0,1\} \tag{9}$$

Equation 3 is the objective function of our model, and aims to minimise the storage overhead (j_0), provider storage cost (j_1), provider bandwidth input cost (j_2), provider bandwidth out cost (j_3) and maximise the confidentiality (j_4), performance (j_5), availability (j_6) and durability (j_7).

$Q_{k,j}$ is the aggregated value for the j_{th} QoS attribute, offered by the k_{th} combination of modules and w_j is the normalised weight for the j_{th} QoS attribute. Q_j^{max} is the maximum aggregated value and Q_j^{min} is the minimum aggregated value for QoS attribute j offered by all the possible combination of modules.

Equation 4 defines that the sum of all normalised weight (w_i) parameters equals 1 to ensure that all weights are using the same scale.

Equations 5, 6 and 7 are responsible to select the modules. Variables $N_{ida}, N_{enc}, N_{comp}$ represent the amount of modules available in each category. The decision variables D_i (dispersal), E_i (encryption), C_i (compression) indicate whether the module i will be chosen (=1) or not (=0). Note that the dispersal algorithm (Equation 5) is always required to be selected while the selection of the encryption and compression module are optional.

The QoS user requirements constraints are satisfied by using Equation 8. Again, variable $Q_{k,j}$ is the aggregated value for the j_{th} QoS attribute, offered by the k_{th} combination of modules. Q_j^{user} is the user required value for QoS attribute j. This ensures that the solution satisfy user's hard requirements for the desired QoS attributes.

The provider selection is represented by Equation 9. N_{prov} is the number of available providers to choose. P_i is a decision variable used to indicate whether the provider i will be chosen (=1) or not (= 0) and P_{req} is the number of providers that the model needs to choose for the dispersal process. Table III summarises the model variables.

C. Provider Parameters Retrieval

It is important to retrieve the providers' QoS parameters to provide the optimisation model with the last values. To obtain these, we designed a local daemon service. It retrieves some of the providers data automatically. It populates the database with the respective values for availability, read performance, write performance, storage costs and bandwidth costs. Availability is given by periodically checking if the service is available. Read and write performance are calculated by the average throughput (KBps) of each operation and the costs are retrieved directly from the providers website through JSON. Storage cost is the cost to store one gigabyte of data for one month. The bandwidth cost is the cost to transfer one gigabyte of data. Durability is retrieved from the provider SLA and is measured in number of nines. Table IV shows the providers (anonymously) and the selection parameters used.

TABLE III. MODEL VARIABLES

Name	Type	Description	Source
$Q_{k,j}$	Integer	Aggregated value for j_{th} QoS attribute, offered by the k_{th} combination of modules	Solver
Q_j^{max}	Integer	Maximum aggregated value for QoS attribute j	Solver
Q_j^{min}	Integer	Minimum aggregated value for QoS attribute j	Solver
D_i, E_i, C_i	Binary	Decision variables to indicate whether module i will be selected for each category	Solver
P_i	Binary	Decision variable used to indicate whether the provider i will be chosen	Solver
N_{ida}, N_{enc}, N_{comp}	Integer	Number of modules available in each category	Database
N_{prov}	Integer	Number of providers	Database
P_{req}	Integer	Number of providers to choose	Specialist system
w_j	Float	Normalised weight for j_{th} QoS attribute	Specialist system
Q_j^{user}	Integer	User required value for QoS attribute j	Specialist system

TABLE IV. PROVIDERS

Prov.	Avail.	Dur.	Read perf.	Write perf.	Confid.	Cost sto.	BwIn	BwOut
P1	99.9	9	350	80	75	10	0.02	0.02
P2	99.99	11	653	115	75	15	0.06	0.06
P3	99	8	832	80	75	10	0.05	0.05
P4	99.999	12	221	35	75	7	0.01	0.01

D. Rule-based QoS parameters weight specialist system

The QoS parameter weight for confidentiality is directly based on the user set of inputs. The other QoS parameter weights are defined based on a specialist system composed of predefined rules that leverage the user set of inputs, file size and file extension. The rules can increase or decrease each weight to reflect the desired dispersal characteristics.

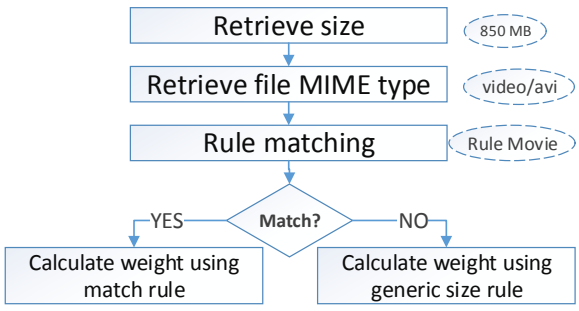

Fig. 3. Rule-based specialist system

Figure 3 shows the specialist system flow. The figure also shows as an example (rounded dashed shapes) a movie file (.avi extension with 850 MB) chosen to be dispersed. The MSSF system is able to recognise that this file is a movie (because of the file MIME type and file size) and will calculate (increase or decrease) the QoS parameter weights according to the "movie" rule. The rules were created based on a survey[3], the literature and feedback from storage specialists. If there is no match, then the generic rule based on the file size is used. The rules are encoded in a database, but to give an idea of the content we will show the video and the generic rule as examples (using a condition-action notation) now:

```
Rule Movie:
if File.MIME=video && File.SIZE >= 600MB
then w_prf_rd=⋆1.25 and w_prf_writ=⋆0.6 and w_sto=⋆1.15
     and w_stcost=⋆1.15 and w_bw_in=⋆0.7 and w_dur=⋆1.15

Rule BigFile:
if File.SIZE >= 100MB
then w_perf_rd=⋆0.6 and w_avail=⋆1.2 and w_dur=⋆1.2
```

As can be seen, the rules describe the change one would like to see to weights for the individual factors based on the file type. The rules multiply the initial QoS user parameters according to the desired specialist storage strategy; for example for a movie, it will be more likely to be read then written often so read performance is important. Using the rules above, the specialist system increases read performance weight by 25%, decreases write performance by 40%, increases storage weight by 15%, increases storage cost weight by 15%, the dispersal bandwidth weight decreases by 30% and the durability weight increases by 15%.

E. Workflow

MSSF is composed of three main modules, the graphical user interface (GUI), a specialist system and the integer linear programming solver. Figure 4 shows the workflow for the dispersal selection process. First, the user needs to provide the input parameters for the desired file. Then, the specialist system uses some predefined rules to determine weights for each QoS parameter. Finally, MSSF runs the solver with the calculated QoS parameters and selects the optimum dispersal option (which providers to use, dispersal algorithm and modules) to disperse the file.

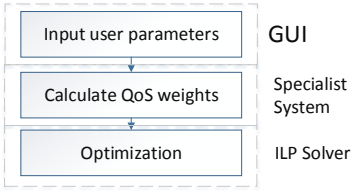

Fig. 4. MSSF workflow

VI. Experiments

MSSF was evaluated using two distinct experiments. The first experiment evaluates MSSF performance to check whether it is a feasible solution for public cloud storage environments using a performance evaluation. The second experiment is composed of an usability test with real users to check if our solution is easy to use and is able to select an adequate dispersal configuration according to user needs regarding their profile and file properties.

A. Performance Evaluation

The first set of experiments was designed to evaluate the performance of MSSF, its performance and suitability for real cloud environments with several servers and modules. The goal was to evaluate only the selection mechanism performance and not the other aspects, such as communication The environment used for the tests was a laptop with Intel Core i3-2330M 2,2 GHz, DDR3 4 GB, Windows 7 SP1 with GLPK solver v4.55[28]. The goal was to evaluate the selector performance

We design a full factorial experiment using our framework. We analysed the response variable selection time with ten replications for each experiment. The selection time represents the time needed for the integer linear optimiser engine to discover the combination of modules and servers that leads to the optimal solution. Table V shows the factors and levels considered. The number of modules is the total number of modules (IDA, encoder and compression) loaded in the system. The providers are the possible places to store the slices. The set of input selection parameters used was:

```
MIN_SEC=10;MIN_PERF=10;MIN_STO=10;
WEIGHT_SEC=0.4;WEIGHT_PERF_READ=0;
WEIGHT_PERF_WRITE=0.1;WEIGHT_STO=0;
WEIGHT_STOCOST=0.1;WEIGHT_BWCOST_IN=0.1;
WEIGHT_BWCOST_OUT=0.1;WEIGHT_AVAIL=0.2;
WEIGHT_DUR=0;PROV_REQ=75%.
```

Additionally, a Scenario XML Configuration file for FlexSky is available for this experiment (included as Appendix for completeness).

Factor	Levels
Number of Modules	300, 3000, 30000
Number of Providers	100, 1000, 10000

TABLE V. Experiment I factors.

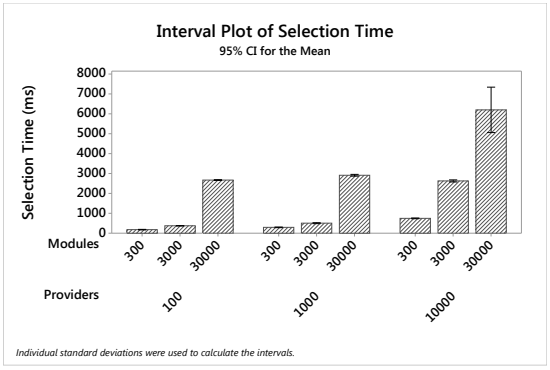

Fig. 5. Results for selection time.

Figure 5 shows the selection time. The x axis represents the experiment factor combinations and the y axis represents the selection time in milliseconds. Even with values as high

[3]https://www.surveymonkey.com/results/SM-FML36X87/

as ten thousand providers and thirty thousand providers, the selection time average was under seven seconds, for the large-scale scenario. For the small scenario (100 providers and 300 modules), more likely for a regular user, the selection time average was 180,6 milliseconds. This indicates that our solution is fast and can be scalable.

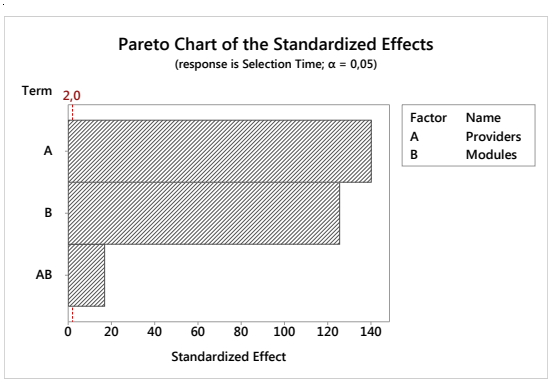

Fig. 6. Factors statistical influence

We also want to determine the influence of factors on the selection time. Figure 6 shows the Pareto chart of these influences. The bar size indicates the influence of the factor on the response variable. According to this statistical analysis, the factor number of providers has slightly more influence on the selection time than the factor number of modules.

B. Qualitative Test

A usability test was designed and executed to check MSSF selection accuracy according to user needs. The usability test was conducted with five users from distinct backgrounds (academic, IT, business and personal) aiming to check if the users would be able to insert the right input parameters using the graphical user interface according to each type of file. It was composed of four tasks involving four distinct file types presented below:

1) A document file (.docx) of a business report with the financial data of their company that they are working on.
2) A public instructions manual (.pdf) they need to read every day at their work.
3) A picture file (.jpg) from their childhood.
4) An access log from a company that is automatically written.

We used the think aloud method [29] for the usability test. This method encourages the user to talk about what they are thinking when performing the actions. This is useful to verify if the users thinking flow correspond with the desired thinking flow for our solution and if the user parameters selection is adequate for the user. For this test we used five users, which is shown to be enough to detect most of usability issues [30].

The tasks along with a brief description of the file types and the think aloud usability-testing method were presented to each user. Then we show the folder where the files are located and we start the test. After finishing, we compare the dispersal selection result from the test with the most suited selection according to a storage specialist. We can then evaluate whether if the user input parameters is suited to the file characteristics and discover usability errors on the UI.

TABLE VI. USABILITY TEST RESULTS

User/Task	Task 1	Task 2	Task 3	Task 4	Usability Obs.
User 1	✓	✓	✓	✓	Comfortable
User 2	✓	✓	✓	✓	Folder instead of file selection; Comfortable
User 3	✓	✓	✓	✓	Comfortable
User 4	✓	✓	✓	✓	Asked for the advanced interface; Wanted to limit by budget; Comfortable
User 5	✓	✓	✓	✓	Comfortable

Table VI summarises the usability test. All users were able to complete all the tasks according to the specialist. The input parameters were slightly different, but the system loaded with the predefined rules was able to adjust the parameter weights that lead to an adequate multi cloud storage strategy. The users also felt comfortable with the input parameters.

User 2 suggested a folder-based strategy instead of a file-based strategy, because for him it was more practical to select these properties for each folder and then store his files according to the desired folder. User 4 (IT background) asked for an advanced interface in which he were able to control all the parameters directly. He also asked for a budget option to limit the maximum storage price for the selected storage strategy. Although our model leverages the storage cost, we decided not to ask for this input from the regular user because it might be impractical for normal users. Most of the public storage providers business plans are based on the storage folder size available on the provider and not on the amount of stored data, so in real scenarios it is not feasible for the user to specify storage cost prior to uploading files. Although we decided not to ask for this information from our user in the simple interface, our model could support such a parameter and an advanced user interface would be able to provide such options for the user.

VII. CONCLUSIONS

We presented MSSF (Multi-cloud Selection Storage Framework), a QoS aware selection framework to select the optimum multi-cloud storage strategy. MSSF is part of FlexSky, a highly flexible multi-cloud storage architecture to ease the study and comparison of several dispersal modules, algorithms and infrastructure. MSSF's goal is to improve and automate the dispersal storage selection process. We first model the multi-cloud dispersal storage as a knapsack optimisation problem leveraging several QoS parameters, we then propose and implement an integer linear programming solver to select the optimum combination of providers, dispersal algorithm and modules to store the file according to the desired requirements. We also present a rule-based specialist system that uses file characteristics to adjust the parameter weights provided by the user interface.

Finally, we show the performance and the usability of our proposed solution through performance evaluation and usability experiments respectively. The further development of our specialist system, using distinct approaches and the development of a better user interface are left for future works.

ACKNOWLEDGEMENTS

The authors thank FAPESP, CNPq and CAPES for the financial support. Part of the work was conducted while Rafael M. de O. Libardi was at the University of Leicester supported by FAPESP through process 2013/05567-9 and 2014/05888-2.

REFERENCES

[1] J. McKendrick, "Cloud computings vendor lock-in problem: Why the industry is taking a step backward," *Forbes, November*, 2011.

[2] R. De Oliveira Libardi, M. Naves Bedo, S. Reiff Marganiec, and J. Estrella, "Mssf: A step towards user-friendly multi-cloud data dispersal," in *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*, June 2014, pp. 952–953.

[3] M. D. Ryan, "Cloud computing security: The scientific challenge, and a survey of solutions," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2263 – 2268, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121212003378

[4] F. Kerschbaum and L. Chaves, "Secure sharing of item-level data in the cloud," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011, pp. 756–757.

[5] S. Agarwala, D. Jadav, and L. Bathen, "icostale: Adaptive cost optimization for storage clouds," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, July 2011, pp. 436–443.

[6] M. Li, "On the confidentiality of information dispersal algorithms and their erasure codes," *CoRR*, vol. abs/1206.4123, 2012.

[7] A. Juels and A. Oprea, "New approaches to security and availability for cloud data," *Commun. ACM*, vol. 56, no. 2, pp. 64–73, Feb. 2013.

[8] H. Alabool and A. Mahmood, "Review on cloud service evaluation and selection methods," in *Research and Innovation in Information Systems (ICRIIS), 2013 International Conference on*, Nov 2013, pp. 61–66.

[9] Z. ur Rehman, O. Hussain, and F. Hussain, "Iaas cloud selection using mcdm methods," in *e-Business Engineering (ICEBE), 2012 IEEE Ninth International Conference on*, Sept 2012, pp. 246–251.

[10] Z. ur Rehman, F. Hussain, and O. Hussain, "Towards multi-criteria cloud service selection," in *Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2011 Fifth International Conference on*, June 2011, pp. 44–48.

[11] S. Le, H. Dong, F. Hussain, O. Hussain, J. Ma, and Y. Zhang, "Multi-criteria decision making with fuzziness and criteria interdependence in cloud service selection," in *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, July 2014, pp. 1929–1936.

[12] Z. ur Rehman, O. Hussain, and F. Hussain, "Multi-criteria IaaS service selection based on QoS history," in *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, March 2013, pp. 1129–1135.

[13] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, May 2004.

[14] A. Ruiz-Alvarez and M. Humphrey, "A model and decision procedure for data storage in cloud computing," in *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*, May 2012, pp. 572–579.

[15] ——, "An automated approach to cloud storage service selection," in *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing*, ser. ScienceCloud '11. New York, NY, USA: ACM, 2011, pp. 39–48. [Online]. Available: http://doi.acm.org/10.1145/1996109.1996117

[16] D. Petcu, "Consuming resources and services from multiple clouds," *Journal of Grid Computing*, vol. 12, no. 2, pp. 321–345, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10723-013-9290-3

[17] J. Spillner, G. Bombach, S. Matthischke, J. Muller, R. Tzschichholz, and A. Schill, "Information dispersion over redundant arrays of optimal cloud storage for desktop users," in *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on*, Dec 2011, pp. 1–8.

[18] D. Slamanig and C. Hanser, "On cloud storage and the cloud of clouds approach," in *Internet Technology And Secured Transactions, 2012 International Conference for*, Dec 2012, pp. 649–655.

[19] T. G. Papaioannou, N. Bonvin, and K. Aberer, "Scalia: An adaptive scheme for efficient multi-cloud storage," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. Los Alamitos, CA, USA: IEEE Computer Society Press, 2012, pp. 20:1–20:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389024

[20] F. Miller, A. Vandome, and M. John, *Federal Standard 1037C*. VDM Publishing, 2010. [Online]. Available: http://books.google.co.uk/books?id=SyEaYAAACAAJ

[21] G. Lefebvre and M. J. Feeley, "Separating durability and availability in self-managed storage," in *Proceedings of the 11th Workshop on ACM SIGOPS European Workshop*, ser. EW 11. New York, NY, USA: ACM, 2004. [Online]. Available: http://doi.acm.org/10.1145/1133572.1133576

[22] S. Chen and J. Liu, "Statistical applications of the poisson-binomial and conditional bernoulli distributions," *Statistica Sinica*, vol. 7, no. 4, pp. 875–892, 1997, cited By (since 1996)46. [Online]. Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-0031313967&partnerID=40&md5=3316a763aea912beb33d5bf32d2bd96b

[23] M. Li, "On the confidentiality of information dispersal algorithms and their erasure codes," *CoRR*, vol. abs/1206.4123, 2012. [Online]. Available: http://arxiv.org/abs/1206.4123

[24] V. Casola, A. Cuomo, M. Rak, and U. Villano, "Security and performance trade-off in perfcloud," in *Euro-Par 2010 Parallel Processing Workshops*, ser. Lecture Notes in Computer Science, M. Guarracino, F. Vivien, J. Trff, M. Cannatoro, M. Danelutto, A. Hast, F. Perla, A. Knpfer, B. Di Martino, and M. Alexander, Eds. Springer Berlin Heidelberg, 2011, vol. 6586, pp. 633–640. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-21878-1_78

[25] M. Mesnier, E. Thereska, G. Ganger, D. Ellard, and M. Seltzer, "File classification in self-* storage systems," in *Autonomic Computing, 2004. Proceedings. International Conference on*, May 2004, pp. 44–51.

[26] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "Qos-aware middleware for web services composition," *Software Engineering, IEEE Transactions on*, vol. 30, no. 5, pp. 311–327, May 2004.

[27] D. Ardagna and B. Pernici, "Adaptive service composition in flexible processes," *IEEE Trans. Softw. Eng.*, vol. 33, no. 6, pp. 369–384, Jun. 2007. [Online]. Available: http://dx.doi.org/10.1109/TSE.2007.1011

[28] A. Makhorin, "Glpk (gnu linear programming kit)," 2008.

[29] C. Lewis and J. Rieman, *Task-centered User Interface Design: A Practical Introduction*. University of Colorado, Boulder, Department of Computer Science, 1993. [Online]. Available: http://books.google.com.br/books?id=j8mOYgEACAAJ

[30] J. Nielsen and T. K. Landauer, "A mathematical model of the finding of usability problems," in *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, ser. CHI '93. New York, NY, USA: ACM, 1993, pp. 206–213. [Online]. Available: http://doi.acm.org/10.1145/169059.169166

APPENDIX

```
<scenario name="CLOUD-Experiments" type="simulation">
    <repeat>10</repeat>
    <db resetOnStart="true">/data/.../CLOUD-experiments-files.db</db>
    <log_db>/data/.../CLOUD-experiments-log.db</log_db>
    <providerList amount=100></providerList>
<modulesList>
<module type="ida" amount=100></module>
<module type="enc" amount=100></module>
<module type="comp" amount=100></module>
</modulesList>
    <operationList> <operation action="selectOnly">
<select_param name="MIN_SEC" value=10></select_param>
<select_param name="MIN_PERF" value=10></select_param>
<select_param name="MIN_STO" value=10></select_param>
<select_param name="WEIGHT_SEC" value=0.4></select_param>
<select_param name="WEIGHT_PERF_READ" value=0></select_param>
<select_param name="WEIGHT_PERF_WRITE" value=0.1></select_param>
<select_param name="WEIGHT_STO" value=0></select_param>
<select_param name="WEIGHT_STOCOST" value=0.1></select_param>
<select_param name="WEIGHT_BWCOST_IN" value=0.1></select_param>
<select_param name="WEIGHT_BWCOST_OUT" value=0.1></select_param>
<select_param name="WEIGHT_AVAIL" value=0.2></select_param>
<select_param name="WEIGHT_DUR" value=0></select_param>
<select_param name="PROV_REQ" value=0.75></select_param>
        </operation> </operationList>
```

</scenario>