DOI 10.12694/scpe.v17i4.1202
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 17, Number 4, pp. 299–312. http://www.scpe.org

MULTI-OBJECTIVE MIDDLEWARE FOR DISTRIBUTED VMI REPOSITORIES IN FEDERATED CLOUD ENVIRONMENT

DRAGI KIMOVSKI [*], NISHANT SAURABH *, VLADO STANKOVSKI [†] AND RADU PRODAN *

Abstract. Virtualization represents an essential technology in Cloud computing, which allows virtual machines (VM) to be executed within their own environment on top of physical hardware. The modern methods for software delivery are utilizing the concept of Vitual Machine as a efficient tool for software packaging. Typically, VMs are created using specific templates that are stored in proprietary repositories, thus leading to provider lock-in and reduced portability in the cases of simultaneous usage of multiple federated Clouds. Unfortunately, the current state-of-the-art does not provide any efficient means for streamlined management of VM images across multiple repositories, especially within federated Cloud environments. In this paper we present a novel multi-objective middleware for distributed VMI repositories in federated Cloud environment. The middleware has been designed to provide easy to use interface capable of receiving unmodified and functionally complete VM images from its users, and transparently distribute them to a specific Cloud infrastructure in a federation with respect to their size, configuration, and geographical distribution, such that they are loaded, delivered, and executed faster and with improved QoS compared to their current behaviour.

Key words: Distributed Repositories, Cloud Federation, Multi-Objective optimization

AMS subject classifications. 68M14, 90C26

1. Introduction. Virtualization represents an essential technology in Cloud computing, which allows virtual machines (VM) to be executed within their own environment on top of physical hardware [14]. The modern methods for software delivery are utilizing the concept of Vitual Machine as a efficient tool for software packaging. The exploiting of this concept enables efficient scaling of applications by providing elastic on-demand provisioning of VMs in response to their variable load, thus increasing the utilization efficiency at a lower financial cost [4]. Typically, VMs are created using specific templates that are stored in proprietary repositories, thus leading to provider lock-in and reduced portability in the cases of simultaneous usage of multiple Clouds [8, 15, 10].

Unfortunately, the current state-of-the-art does not provide any efficient means for streamlined management of VM images (VMI) across multiple repositories within federated Cloud environments [cite cloud federation]. In such environments, the VMIs are currently stored by Cloud providers in proprietary centralised repositories without considering application characteristics and their runtime requirements, causing high deployment and instantiation overheads. Moreover, users are expected to manually manage the VMI storage, which is tedious, error-prone and time-consuming process, especially if working with multiple Cloud providers. Furthermore, each Cloud provider utilizes different type of storage technique and implements different interfaces for accessing it, thus reducing the usability even further. In this paper we present a novel Multi-objective middleware for management of VMIs in federated Cloud repositories, which has been developed as an essential part of the H2020 ENTICE project [1]. The middleware has been designed to provide easy to use interface capable of receiving unmodified and functionally complete VM images from its users, and transparently distribute them to a specific Cloud infrastructure in a federation with respect to their size, configuration, and geographical distribution, such that they are loaded, delivered, and executed faster and with improved QoS compared to their current behaviour. The proposed middleware has been focused towards overcoming the barriers that prevent the wide usage of distributed VMI repositories in federated Cloud environment and it aims to: (i) automatically distribute VM images based on multi-objective optimisation to meet application runtime requirements; and (ii) provide interface which enables users to manage their VM images in federated Cloud infrastructures without provider lock-in.

The paper is structured as follows. In Section 2 we present the basic concepts of the ENTICE environment. In Section 3 we introduce the ENTICE functional and non-functional requirements and how those affect the middleware functionality. Furthermore, a detailed overview of the middleware and all provided services is

[*]University of Innsbruck, dragi@dps.uibk.ac.at
[†]University of Ljubljana,
[1]http://www.entice-project.eu/

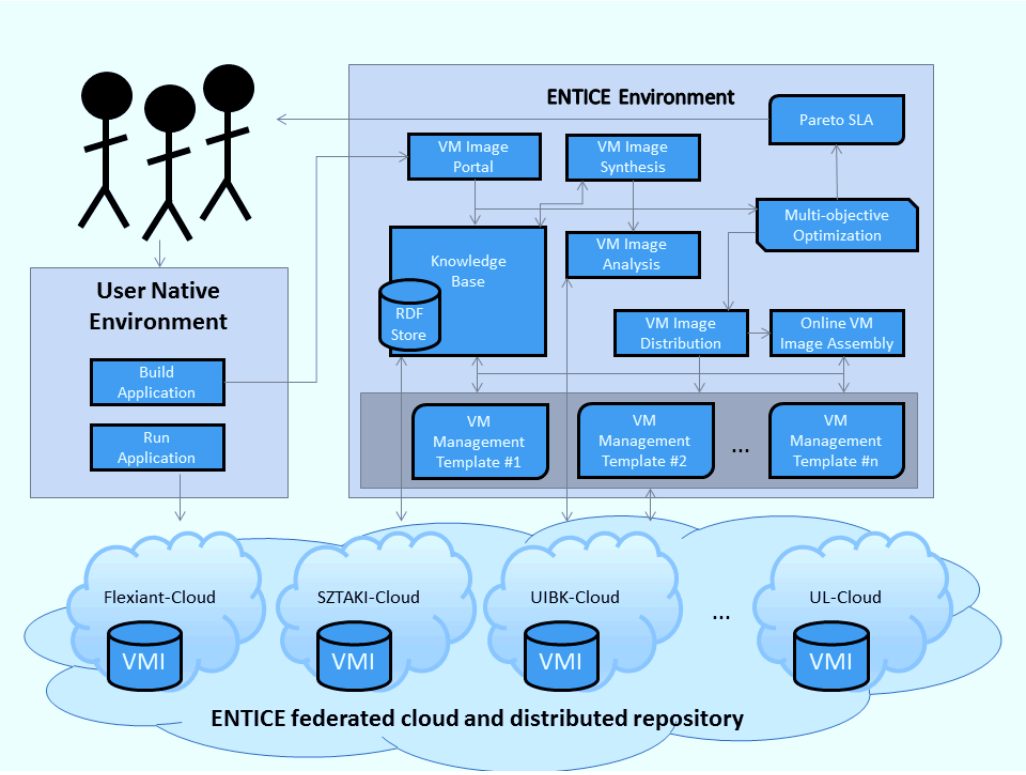

FIG. 2.1. *Top level view of the ENTICE environment*

presented in Section 4. Then we proceed by presenting the developed multi-objective framework for VMI redistribution. Further to this, we present details of the implementation of the framework and it's integration within the middleware. Section 7 summarizes the main results, their impact and future work.

2. ENTICE environment for efficient and transparent virtual machine operations in distributed Cloud repositories. The ENTICE project aims on development of an dynamic environment capable of receiving unmodified and functionally complete VM images from users, and transparently tailor and optimise them for specific Cloud infrastructures with respect to their size, configuration, and geographical distribution, such that they are loaded, delivered, and executed faster and with improved QoS compared to their current behaviour. ENTICE gradually stores information about the VMI and fragments in a knowledge base that is used for interoperability, integration, reasoning and optimisation purposes.

VM images management is supported by ENTICE at an abstract level, independent of the middleware technology supported by the underlying Cloud computing infrastructure. To further shield the users from the complexity of underlying Cloud technologies and simplify the development and the execution of complex use cases, ENTICE focuses on providing the flexibility for tailoring the VM images to specific Cloud infrastructures.

The ENTICE repository includes, among other features, techniques to optimise the size of VM images while maintaining their functionality, automatically share images (or parts of the images) among repositories (even in multiple administrative domains or cloud infrastructures), and optimally deploy them in response to application and data centre requirements.

The ENTICE environment, depicted in Figure 2.1, aims to prove a universal backbone and middleware for IaaS VM management operations, which accommodate the needs for different use cases with dynamic resource and other QoS requirements. The ENTICE technology is completely decoupled from the applications and their specific runtime environments, but continuously supports them through optimised VM image creation, assembly, migration and storage.

Within the ENTICE environment, the Multi-objective middleware for distributed VMI repositories has

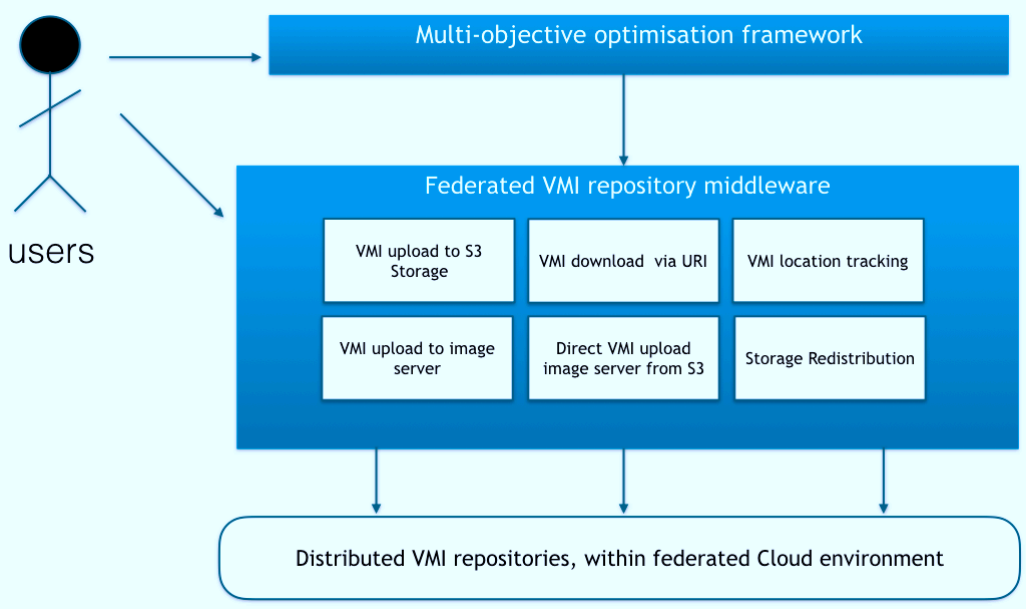

FIG. 2.2. *Top level view of the Multi-objective middleware for distributed VMI repositories*

been layered in multiple modules. To be more concrete, the middleware is composed of VM image distribution module, online VM image assembly module and multi-objective optimization framework. All components interact by using service based API interfaces, thus allowing independent integration of the middleware in various distributied VMI repository architectures. The top level view of the middleware, as an independed fully integrated module, is presented in Figure 2.2.

3. Functional requirements. As discussed in Section 1, in order to overcome the barriers that limit the possibilities for Cloud federation, we present a specific use-case scenario, which is highly relevant for proper definition of the middleware interface that ENTICE environment has to offer. Furthermore, based on these requirements, we list down the imminent functionalities specific to relevant components of the ENTICE environment.

The use-case of ENTICE environment at the outset, typically initiates with the Application Provider uploading VM Image encompassed with appropriate application functionalities. Henceforth, an easy interface to the user which integrates to the funtionalities including image optimization, storage and the knowledge oriented image portal with enhanced reasoning based module for easy access of services (example: upload, download, update, optimize etc.) and VM image management is required. In general, user specific images are large sized comprising of redundant functionalities incurring high storage cost resulting to image distribution and instantiation overheads. Henceforth, the splitting of an image into multiple fragments serving common functionality and storing them only once is necessitated to reduce the redundancy. Furthermore, the repository optimization with regards to the storage location of images evaluating performance metrics of each attached repository medium is imminent for faster VMI distribution across multiple clouds. Based on these factors, we identify the generalized requirements for the ENTICE environment. Hence, we divide the ENTICE design requirements in the following broad categories: (i) Application data-related requirements, (ii) Security-related requirements, (iii) VMI and fragment management-related, (iv) Information and metadata management-related requirements, and (v) QoS metrics-related.

The Application-data related requirements encompass the delivery of various content alongside the VM image, for the efficient execution of the corresponding application. To this extent, the ENTICE environment provides a functional descriptions tool that contains references to the additional VM image content and evaluate its presence in terms of files, folders, sizes, distribution and etc.

Furthermore, the ENTICE environment needs to address various security measures, to prevent unauthorized

access of VMIs. For this purpose, the repository must support content delivery through encrypted channels. Moreover, the analysis methods for VMIs of the ENTICE environment should not compromise the privacy and security of the Cloud application owners and should not expose any user data, which should be securely stored in the knowledge base.

One of the requirements that ENTICE repository holds is to provide fast VM image delivery, hence identify the VMIs and fragment management-related requirements. The management module requires efficient integration of specific modules for VM image and meta-data delivery, with the a Multi-objective optimization framework for online and offline distribution of the data. In addition, the ENTICE environment needs emphasis to support the delivery of data in the form of files or folders containing user specifications, thus requiring relevant informations and metadata management. In such cases, ENTICE must provide appropriate storage and data delivery mechanism together with the VMIs. Further, the data needs to be indexed, so that the appropriate distribution techniques allow efficient delivery to relevant geographical locations. Hence, it is required to miantain index hashing of VMIs and correspondin fragments. The semantic model represented in the Knowledge Base requires to store information about the VMIs and their functionality, the geographic location, the URI, and other details for the search facility.

Lastly, monitoring of QoS metrics is necessary to be able to observe the time needed to move VMIs and/or fragments among repository locations and to deploy a VMI needed by a particular Cloud application. A further requirement is to monitor the infrastructure on which the ENTICE repository is deployed to ensure that the scaling and speed of operation is adequate.

Hence, based on the discussed requirements, we list down the important use-case functionalities for the Multi-objective VMI repository as a federation middleware:

- *Upload of unoptimised images to S3 storage*
- *Upload of optimised images to S3 storage*
- *VM Image download via URI*
- *VM Image location tracking*
- *Upload of VM Image directly to Image Server of Cloud provider for deployment*
- *Upload of VM Image to the image server of specific Cloud provider from S3*
- *Storage redistribution of VM Images stored within S3 based VMI repository*

Furthermore, for the optimization framework we have identified and implemented the following use-case functions:

- *Applying Multi-objective Optimization for distribution of the VM Images and fragments across distributed storage sites*
- *Optimized extraction of fragments, based on functionality, for assembly of the VM Images during the deployment stage*
- *Movement of VM image from one storage location to another*
- *Location Tracking of each VM Image to enhance storage and distribution*
- *VMI conversion from one provider specific template to another to achieve interoperability over multiple cloud*

4. VMI repository as a federation middleware. The *ENTICE* VMI storage repository evolves from the typical storage systems[13] providing general storage services, and instead focuses to act as a middleware corresponding to specific Virtual Machine operations regading the storage, deployment and interoperability of VMI over multiple clouds. These middleware services enhances flexibility of user with respect to maintaining VMI accomplished with user-specific funtionalities and applications and hence surpasses the manual error-prone VM oeprations performed by the users in a federated cloud model. The *ENTICE* federation middleware is attached to storage systems of varying cloud providers(currently supports S3 object storage) accompanied with the Multi-objective optimization service, providing enhanced availability and hence optimizing the VMI distribution time with enhanced deployment.

In general, individual cloud providers constitute of centralized image repositories with services specific to their environment. *ENTICE* recognizes the limitations including the interoperability of VMI and and hence intializes a middleware API with a number of services corresponding to storage of VMI across federated multiple clouds[11] with enhanced Quality of service[2] with regard to VM provisioning and deployment. This enables

the user to have flexible usage of Cloud Infrastructure as a Service as a global paradigm[9]. Henceforth, in this section, we identify the following middleware services provided by *ENTICE* satisfying the use case functionality defined in the previous section and explain the requirements and service it holds.

4.1. Upload of Un-optimized VMI to S3 Storage. This functionality services the storage of unoptimized images to *ENTICE* S3 repository. The un-optimized images basically refer to the initial version of user-specific images without any optimizations performed with respect to size and application functionality it provides. This image version is often termed as master copy consisting of all the user built applications. The *ENTICE* middleware API intiates the storage of such images onto the corresponding S3 repository with return values namely, VMI name, identifier, S3 storage repository location and Unique resource identifier(URI) of the VMI. The return values are transmitted onto the *ENTICE* knowledge base for future reference by the user.

4.2. Upload of Optimized VMI to S3 Storage. The stored master copy of VMI onto S3 often consists of unused libraries and functionalities. Henceforth, any optimization performed over the VMI by the user by pruning the images, reduces the size of image allowing enhanced provisioning and faster deployment. The upload request of such optimized image is serviced by uploading the latest optimized version along side the master copy of the image. However, the master copy of the image is not deprecated from S3 repository with a view to safeguard the requirement of the user to restore the initial version consisting of complete user specific applications at any point of time. The API return field values namely, name of VMI master copy, name of opimized VMI, storage location, optimized image version and its corresponding URI. These field values allows to maintain the versioning tree for image and its optimizations.

4.3. Upload of VMI to Image Store directly. In general, cloud providers have centralized image store which allows the upload and registry of images suitable to the corresponding cloud infrastructure. The image store is basically attached to varying storage sites differing from one cloud provider to another. For example, Amazon enables the bridging of the S3 storage with the image store, allowing upload of registry of VMI to S3 following a set of rules separating from other content related files and data. The *ENTICE* middleware provides the flexibility to the user to upload their image directly to the image store of their favaorable cloud provider region. As discussed earlier, the upload functionality to S3 is used in the case where decision improbability surfaces within the user about the cloud provider and the region, where VMI has to be deployed. In this functionality, user can choose the image store of the specific cloud provider and corresponding region. The return values of this service provide field values with respect to VMI name, VMI unique identifer, size of the image onto the image store, cloud provider identifier and the corresponding zone identifier.

4.4. Upload of VMI to image store from S3 storage. The *ENTICE* API providing the service enables the transfer of un-optimized or optimized images from S3 repository to the image store of the chosen cloud provider, where it can be registered as an instance and hence deployed. As dicussed earlier, S3 storage is used as a back-up to enhance availability of images specifically in the case of indecision related to the cloud provider and region for the deployment of virtual machine(VM). Once the user is decided over the region in which VM has to be deployed, the API service distributes the image to the corresponding destined image store. Henceforth, the field values namely, VMI name, VMI unique identifer, size of the image onto the image store, cloud provider identifier and the corresponding zone identifier are returned and stored in the knowledge base of future reference.

4.5. VMI Location Tracking. The API service tracks the location of stored and instantiated VMI for every user to return the filed values namely, cloud provider identifier and zones where a specific user usually instantiates or store the image. These image specific details allow the Multi-objective optimization module as discussed in next section, to analyse the popularity of cloud provider and zones for varrying users.

4.6. Redistribution of VMI over storage sites. This service corresponds to the Multi-objective optimization service and is performed internally without the interference of the user. The decision making module provides the input parameters for redistribution of the stored images onto the various repository sites corresponding to the user related Quality of service metrics. This services is likely to enhance the distribution of VMI onto the cloud provider, further improving the deployment time and elatic auto-scaling of VM.

4.7. Download via URI. As defined earlier, each stored VMI onto S3 repository returns a URI allowing users to download images from the *ENTICE* environment to their private virtualized cloud infrastructure. Hence, in such case the image is stored at a repository site closer to user location enabling faster download.

4.8. VMI Interoperability. The varying Cloud providers allow specific image formats to be instantiated as VM over their corresponding infrastruture. For example, Amazon require specific Amazon Machine Images(AMI). Hence, this hinders the federated Infrastruture as a Service model[11] leading to vendor lock-in[12] and promotes the manual conversion of images to suit the respected cloud provider to make the process more error-prone. This unique funtionality of *ENTICE* enables interoperability of any stored images onto S3 over multiple clouds by converting to the image format as per the suitability of the user defined cloud infrastructure irrespective of the user provided image format.

5. Multi-objective Optimization Framework for VM image re-distribution. In this section a detailed description of the multi-objective optimization framework for VMI distribution in Federated Cloud repositories will be presented. The optimization framework is capable of directly interacting with the middleware module and has been applied on two distinctive application levels: (*i*) initial VMI distribution, (*ii*) offline VM image redistribution and (*iii*) online VM image redistribution.

5.1. Framework description. The framework is encompassed around unified multi-objective optimization module, which can be utilized for multiple different optimization purposes. Internally, the optimization module is branched in two distinctive sub-modules. Each of the sub-modules has been tailored specifically for a given task. The Initial Distribution sub-module covers the multi-criteria evaluation of the possible repository sites where the VMIs or associated data sets can be initially stored. Afterwards, the Offline VMI Redistribution sub-module encapsulates the optimization of the VM images distribution within the federated repository sites. By taking into account the VMIs usage patterns, the algorithm is capable of providing multiple trade-off solutions, where each solution represents a possible mapping between the stored images and available repository sites. The Online VM image Redistribution sub-module aims on dynamical redistribution of particular VM images/fragments during particular application execution. This sub-module is only applied for users specific VM images and in accordance with the technicalities of the application that is being executed. The framework is dependent on the repositorys usage patterns to properly optimize the distribution of the VM images. To this aim a specific module is required to store information on the previous transfers within the federation and to provide the collected data in a proper format. The module has been realized as an ontology-based knowledge base [1]. The framework has been designed to acquire input data from the knowledge base, and also to return the output results there. Moreover, a specific monitoring agent is required for proper documentation of the data transfers. The monitoring tool itself can be realized in multiple different manners, and it is dependent on the specifics of the Cloud infrastructure.

Furthermore, the framework provides a service based API, through which the Decision Maker (DM) can access the list of optimal Pareto solutions in a guided manner, thus reducing the complexity of the VMI storage management process.The high level structure of the optimization framework is presented on Figure 5.1.

5.2. Initial VM image upload. It is of paramount importance to properly store new VMIs and related data sets in federated Cloud repositories. In this section we introduce concepts from the field of Multiple-criteria decision making, to assist image providers and users to efficiently store new VMIs in accordance with their needs and repository characteristics [3]. The described module, provides a tool which mitigates the process of initial VMI upload, when the available storage sites possibilities are so large that can overwhelm the user during the decision process.

The problem of initial VM image upload consist of a finite number of combinatorial alternatives, which are explicitly known in the beginning of the solving process. In this case, each alternative solution represents one storage site in the federated repository, where the image or data-sets can be stored. Every solution is evaluated on the basis of two conflicting objectives. For the specific problem, the following objectives have been defined:

$$f(P) = B_r \tag{5.1}$$

$$f(C) = C_{st} + C_{tr} \tag{5.2}$$

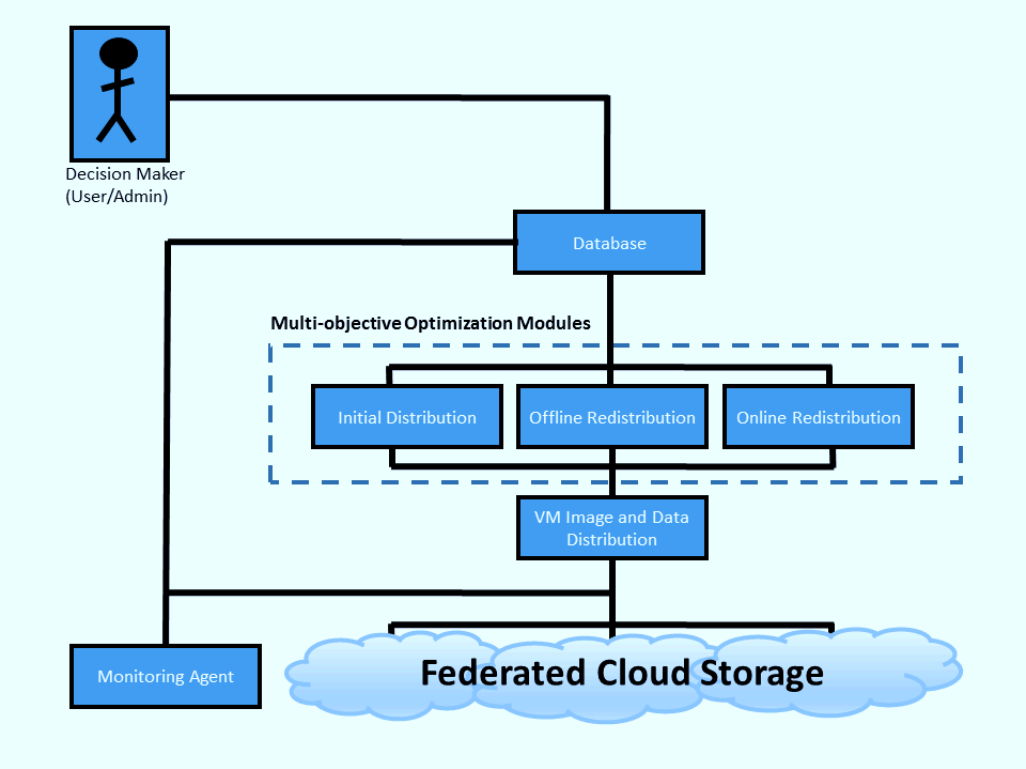

FIG. 5.1. *Top level view of the Multi-objective Optimization Framework for VM Image distribution*

where B_r represents the maximal theoretical performance of the interconnections of the repository, while C_{st} is the cost for storing data on the given repository and C_{tr} is the cost for transfer. Based on the given objectives, all possible storage sites in the repository, are then evaluated. It is important to be noted, that the evaluation is performed only on the feasible solutions, i.e only on the list of available repository sites. This means that prior to evaluation, all constraints for storing the VMI are taken into account. Afterwards, by introducing the concept of domination all evaluated solutions are sorted. The solutions which are non-dominated by any other solution are presented to the user in the form of Pareto front. In a sense, those solutions represent multiple optimal storage sites for storing a single VM image within the federated repository. Next, the user, as a decision maker, can choose where to initially store its own images.

It also worth mentioning, that due to the static nature, this type of evaluation should only be performed when new storage sites have been added or removed from the federated repository. Afterwards, if there are no changes in the structure of the federated repository, the evaluation data can be used for selecting the appropriate storage site for every VM image that might be uploaded in future.

5.3. Offline VM image redistribution. Unlike the initial image upload, the problem of offline VMI redistribution consist of a finite, but very large, number of combinatorial alternatives, which are not known in the beginning of the solving process. The optimization process is conducted by utilizing two conflicting objectives: cost for storing and transferring of the data, which we simply call Cost objective and Performance objective. This process is performed by analyzing the repositories usage patterns, and results in optimized distribution of the VMIs and the associated data-sets across the federated environment. In what follows the exact sequence of steps of the offline VMI redistribution sub-module is presented.

5.3.1. Objective functions modeling. The cost model is described around the notion of the financial expenses which are needed to store a unit of data in a given repository site C_{st} and the economical burden for transferring the data from the initial to the optimal site C_{trnew}. The exact values of the financial expenses for

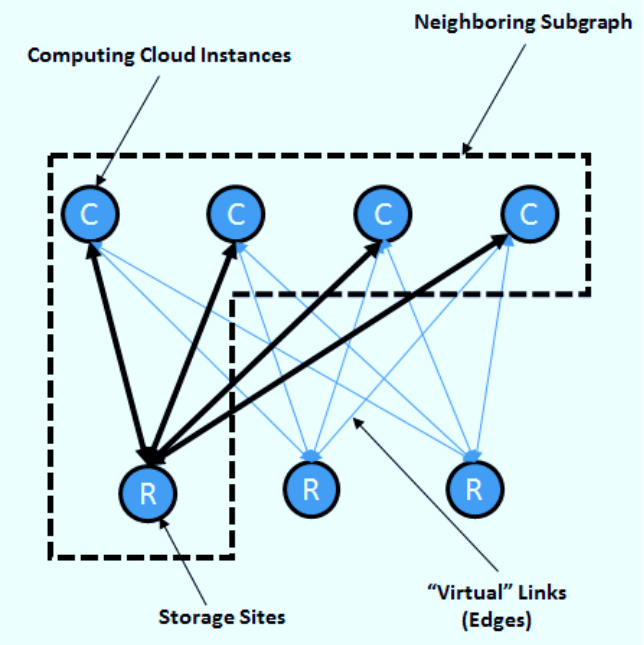

FIG. 5.2. *An example of a neighbouring sub-graph in a structure with 3 repository sites and 4 different cloud providers*

data storage and transfers should be provisioned by all Cloud providers within the federation. For each VM image the cost objective can be calculated by using the formula below:

$$f(C) = C_{st} + C_{trnew} \tag{5.3}$$

The performance model includes much more complex reasoning behind it. It is based on the VM image usage patterns and it requires proper monitoring tool for efficient execution. The raw theoretical throughput of the interconnecting structure within a Cloud federation does not properly describe the factual communication performance, as it is difficult to predict the actual route the packets may take to reach the destination and the load on the intermediate communication channels. Opportunely, it is possible to leverage the data from the frameworks monitoring module to perform a coarse but sufficient estimation on the actual throughput between any pair of end points in the federation. In this way, if there is a sufficient information on the previous transfers among the repository sites and the Cloud computing instances, a direct virtual links between the above mentioned entities can be abstracted over the physical network and their bandwidth can be estimated.

Furthermore, it is possible to model an undirected weighted graph, where the vertices correspond to either a repository site or a computational Cloud instance and the edges of the graph are represented by the virtual links. The weighted graph actually enclosed a union of multiple neighboring subgraphs, where each storage site vertex, as direct neighbor, is linked to all known computational cloud vertices. The weights of the edges in the graph are determined by leveraging the estimated average bandwidth B_{rc_i} on the corresponding virtual links. The weights are calculated dynamically, based the VMI distribution that is being considered. To properly model the weight of the edges, we introduce weight function, which considers the total number of downloads of the VMI to all neighbours G_{tv} and the number of downloads to particular Cloud neighbor G_i. The ratio of those two values is then multiplied with the estimated bandwidth of the particular virtual link to provide the final value of the edges weight. The structure of the neighbouring sub-graph has been represented on Figure 5.2.

Subsequently, for modeling of the performance objective, the sum of the weights of the edges in the neighbouring subgraph is exploited, thus the performance can be described as:

$$f(P) = \sum_{i=1}^{n} B_{rc_i}(\frac{G_i}{G_{tv}}) \tag{5.4}$$

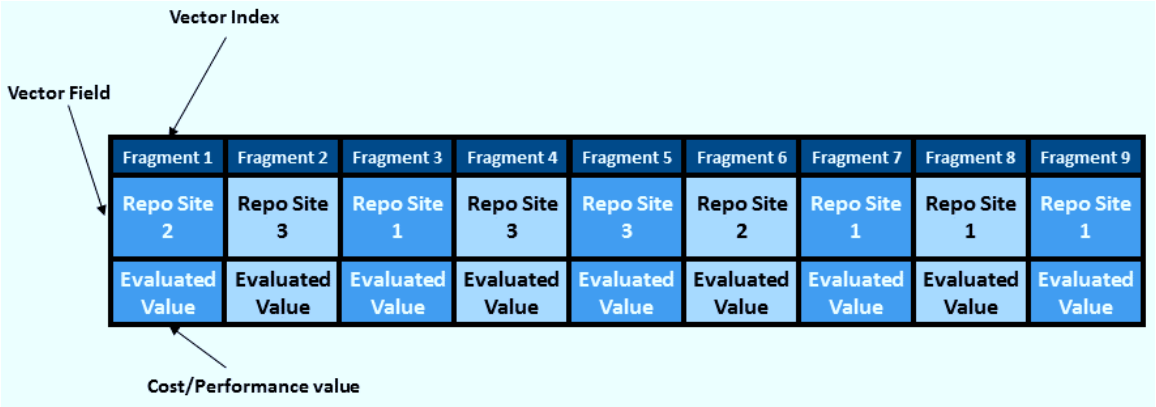

FIG. 5.3. *An example individual represented as a solution vector*

5.3.2. Search Algorithm and Decision Making. The core of the offline VMI redistribution sub-module is constructed over the NSGA-II multi-objective optimization algorithm [5]. As with any population based genetic heuristic the basic entity is the individual. Within the given problem description the individual has been represented as vector with a size equal to the number of stored VMIs. The value kept in every element of the vector corresponds to a single storage repository where a particular VMI can be stored. For accomplishing the above statement, within the proposed framework, each VMI is assigned with a unique ID value, which correspond to the index of the vector element. Respectively, all storage sites in the federation are also assigned with unique IDs that are parallel to the appropriate values saved in the vector elements. In such way, each individual corresponds to a solution vector that represents unique global mapping of all VMIs to storage sites in the federated repository.

Afterwards, multiple solutions vectors are created and then randomly populated with values in the range from one to the number of available storage sites, thus creating the initial population. Every single individual represents one possible distribution solution that has to be evaluated. Then, the evaluation of each individual is performed by reading the values stored in the vector fields. Based on those values, starting from every element in the vector, a neighboring subgraph is constructed and the appropriate objective functions are applied. Those values are then grouped together and the median value is selected as the overall fitness of the given individual. An example of a single individual that correspond to a solution vector for mapping 9 VMIs to 3 storage repository sites in a given federation is presented on Figure 5.3. When all individuals in the initial population have been successfully evaluated, the proper mutation and crossover operators are applied to create the children population. Then, the parents and children populations are grouped together and sorted according to dominance. Afterwards, only the best solution of the newly formed group are selected for the next iteration. This process is then repeated for a predefined number of iterations. The solutions which have been acquired after the last iteration are sorted based on the dominance. The non-dominated solutions are then presented to the administrative entity of the federation, which acts as a DM, and should select the most appropriate solution based on the pre-defined decision making policy.

Decision making on the alternatives discovered by the optimization algorithm requires an explicit model of the decision maker preferences. For the case of offline VMI redistribution the DM model will depend on the implementation of the federated infrastructure. As the offline image redistribution envelops federation wide distribution of the VMIs we envision that the DM will be an administration entity, which will implement the federation storage policy based on the decision making model. Based on the decision done by the DM entity the storage redistribution sub-module would be capable to transparently moving all affected VMIs in the more optimal position.

5.4. Online VMI redistribution. One very important aspect that should be considered in federated cloud environment and repositories is the optimization of specific users VM images and corresponding data sets while correlated applications are being executed. Even though the offline VM image redistribution should place

the VM images in the optimal storage site, there might be cases where the optimization is required only locally, for some particular images or data sets. For example, if a user continuously deploys particular VM image within a short period of time, the position where that image is stored can be additionally optimized based on the newly available data. Consequently, the image can temporarily be transferred to the more optimal solution for the given scenario. The same principle can be applied to the associated datasets, which can be redistributed closer to the physical machines where the VM images are deployed.

By using the same methods implemented in the offline VM image redistribution, the online VM image provisioning can be managed. As both processes are analogous, the only difference comes from the scope and the time interval in which the optimization is performed. With the online VM image redistribution, the optimization is only executed by users request, and only on its own images. When the user ask for optimization of the VM images storage position while deployment, the algorithm is initiated with a limited scope. The input data of the optimization module is only narrowed to the user images and the optimization only takes into account the users usage patterns in a previously set time interval. In this way it becomes possible to further optimize the position of VM images in the cases when they are frequently deployed in a short intervals of time. In a sense it can be commenced, that the offline VM image storage optimization is coarse grained and infrequently applied on all VM images and datasets stored in the federated repository. Contrary, the online VM image optimization is fine grained, and it involves only a users specific VM images that are frequently deployed in a given time interval.

From an algorithmic point of view, the cost and performance objectives are modeled in the same way. The only difference are the input parameters, which in the case of online VM image optimization only involves few VM images, and the output solution vector is also limited only to the images that need to be transferred to more optimal place. As this process only involves fraction of the images stored in the federated repository it can be performed more frequently with reduced computational penalty.

6. Evaluation. In this section, the efficiency of the Multi-objective middleware has been experimentally evaluated based on a synthetic set of benchmark data. As our research deals with the implementation of a combinatorial multi-objective problem in federated Cloud environment, we present an experimental results that demonstrate the ability of our approach to provide an adequate VMI distribution across federated repositories.

With respect to the different application levels of the middleware and the multi-objective optimization framework overall, distinctive set of experiments were conducted. The initial VMI upload module has been evaluated on the basis of the degree of scalability, while the behaviour of the redistribution module has been examined from multiple aspects, such as accuracy, scalability and computational performance.

The simulation experiments on the middleware and Multi-objective optimization framework were conducted on a standalone Linux based machine with an Intel core i7-3770K processor (4 cores and 2 threads per core) working at 3.5 GHz with a 16 GB of RAM. Furthermore, the functionality and behavior of the middleware ware tested in a distributed VMI repository, composed of three storage sites located in Austria, Slovenia and Hungary. In order to guarantee efficient integration and platform independency, all software modules were developed in Java. For the purpose of multi-objective optimization we have leveraged the jMetal optimization library [6].

To begin with, the scalability and computational performance of the initial VMI upload module have been evaluated by varying the number of repository sites in the federation from 10 up to 10000 sites. Figure 6.1 shows the correlation between the average execution time and the number of storage sites in the federation. It is evident that the module can be lightly scaled up to large sizes. For relatively small federations the module can be invoked at each VMI upload, as it requires only few milliseconds to be executed.

On the other hand, the VMI redistribution module encloses diverse operations that can affect its behavior to a various degree. Due to the nature of the algorithm it is not adequate to evaluate it's computational performance based on the number of repositories in the federation. Increasing the number of storage sites, influences on the number of possibilities where to store a single VMI image, which translates into reduced quality of the proposed solutions, but relatively constant execution time. For example, on Figure 6.2 a scenario in which the vector size (number of fragments) and number of evaluations have been kept constant, while the number of available repositories has been increased from 10(blue) to 100 (red), is presented. The Pareto fronts from both executions have been plotted together to show the difference in quality of the final solutions. The

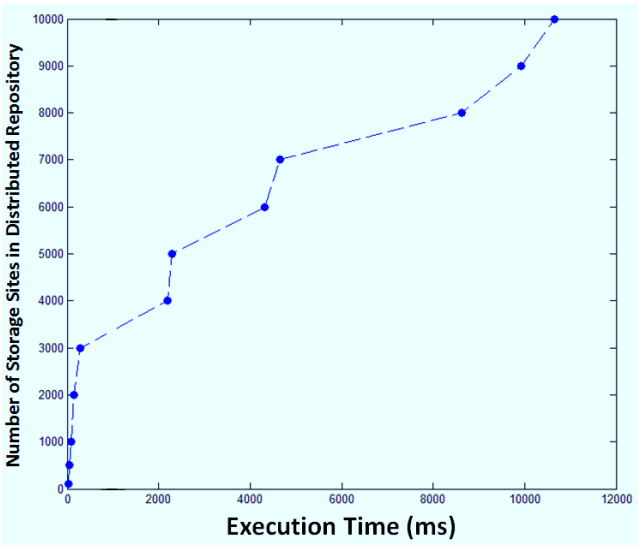

FIG. 6.1. *Correlation between the execution time and the number of storage sites in the case of initial distribution*

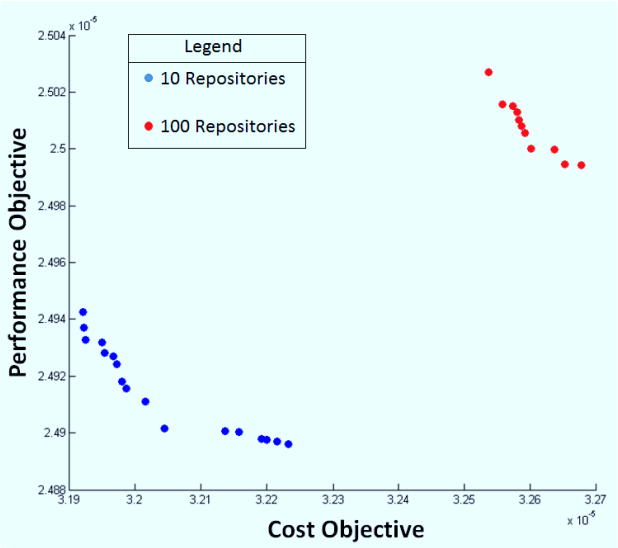

FIG. 6.2. *Comparison of two Pareto fronts during redistribution with varying storage sites*

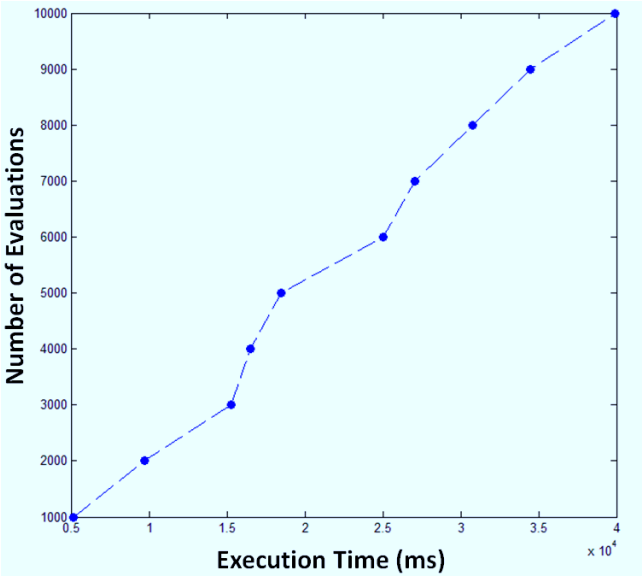

FIG. 6.3. *Influence of the number of evaluations over the execution time during offline redistribution*

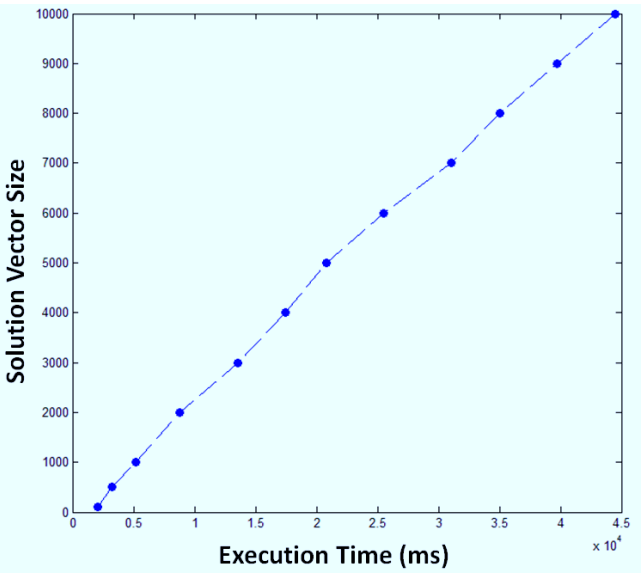

FIG. 6.4. *Influence of the solution vector size over the execution time during offline redistribution*

experimental scenario clearly shows that if we increase the number of storage sites, while maintaining constant number of evaluations, the quality of the solutions will decrease.

Furthermore, on Figure 6.3 and Figure 6.4, respectively, the influence that the number of evaluations and the size of the solution vector have on the computational performance is presented. In both cases, the number of associated cloud computing instances and storage sites were maintained constant; only the corresponding parameters were increased gradually. The presented results support the assumption of satisfactory scalability, both in a sense of increased number of stored VMIs and number of iterations needed to provide mapping solutions with good quality.

Moreover, on Figure 6.5, the influence of the number of known computational Cloud instances over the

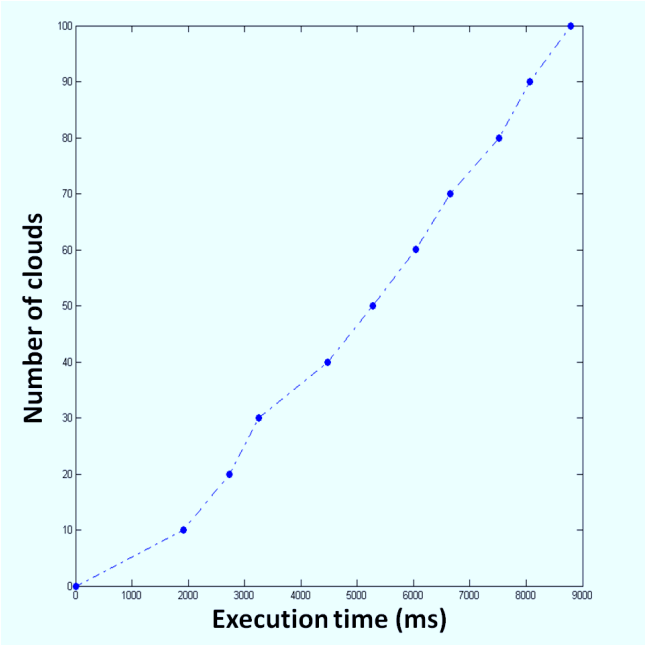

FIG. 6.5. *Influence of the number of known computational Cloud instances over the computational performance of the optimization framework during offline redistribution*

computational performance of the optimization framework is presented. During execution, the number of known computational Clouds was linearly increased, thus inducing higher execution times. The evaluation results are clearly supporting the assumption that the proposed framework scales-up very well with the increment of the problem complexity.

Lastly, Figures 6.6 and 6.7 provide a comparison of the quality values for the conflicting objectives considered in this work. The values for the cost objective have been calculated based on the publicly provided price list for storing data in the Cloud by Amazon. The performance objective has been modelled based on the reported communication performance measures for 10Gbit and 1Gbit Ethernet [7]. For readability reasons, the bandwidth values, were converted to delivery time needed for 1Mbit of data to be transferred from the source to the destination.

With respect to the parameters of the evolutionary algorithms, we have used a population of 1000 individuals, that iterates from 1 to 100 generations across populations. Every single individual(solution vector) is comprised of 1000 chromosomes, thus inducing mapping solutions for 1000 VMIs. Taking into account the results obtained in preliminary experiments, we have used simulated single point crossover with a crossover probability of 0.9, a mutation probability equal to 1/n (n is the number of decision variables). The results indicate very high efficiency of the redistribution module, as it can provide better quality mapping solutions, especially in regards with the performance objective. It can be concluded that in the cases where public Cloud providers, such as Amazon, are being used, it is possible to decrease the delivery time by 143%, while maintaining lower price for storing the VM images.

7. Conclusion. In this paper a novel Multi-objective middleware for management of VMIs across distributed repositories in federated Cloud environment has been proposed. The research work has resulted in development of a optimization framework that exploits multiple different factors, such as communication performance requirements, VMI use patterns, and structure of images, in order to optimize the distribution and placement of VMI across distributed repositories and to significantly lower their provisioning time for complex resource requests and for executing the user applications. Furthermore, streamlined interface that implements a wide array of behavioural functions beyond the typical storage has been provided. The middleware interface offers complex functionalists, such as:

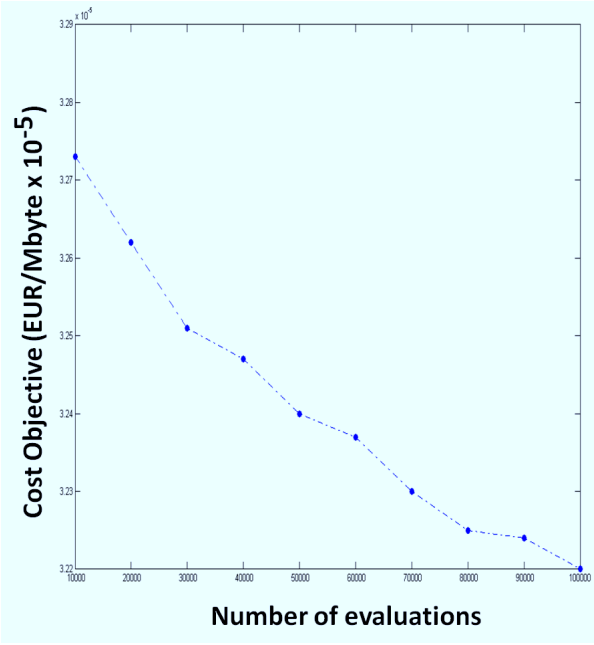

FIG. 6.6. *Average quality values of the Cost objective in comparison with the total number of evaluations*

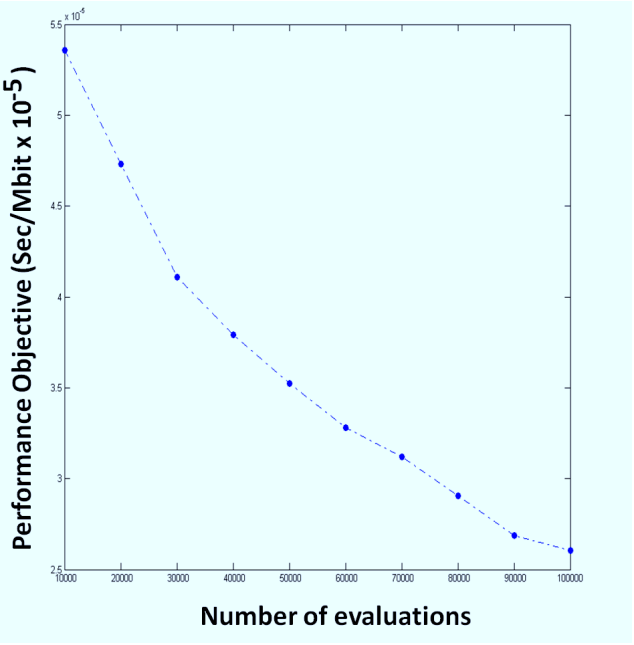

FIG. 6.7. *Average quality values of the Performance objective in comparison with the total number of evaluations*

- *Upload of unoptimised images to S3 storage*
- *Upload of optimised images to S3 storage*
- *VM Image download via URI*
- *VM Image location tracking*
- *Upload of VM Image directly to Image Server of Cloud provider for deployment*
- *Upload of VM Image to the image server of specific Cloud provider from S3*
- *Storage redistribution of VM Images stored within S3 based VMI repository*

The optimization framework, and the middleware overall, have been evaluated based on synthetic simulation benchmark. As our research deals with the implementation of a combinatorial multi-objective problem, where the main incentive is to find the proper mapping of VMIs across storage sites, we present an experimental results that demonstrate the ability of our approach to provide an adequate VMI distribution across federated repositories.

This research area still poses multitude new challenges that urge additional efforts for the accomplishment of the ultimate goal, effortless management of VM images and associated meta-data in federated Cloud environments. In near future, we plan to introduce more complex reasoning and decision making mechanisms, to further improve the multi-objective optimization framework with support for automated creation of VM images in multiple different formats by utilizing virtual machine management templates, and to provide more efficient user interface.

Acknowledgments. This work is being accomplished as a part of project *ENTICE: "dEcentralised repositories for traNsparent and efficienT vIrtual maChine opErations"*, funded by the European Unions Horizon 2020 research and innovation programme under grant agreement No 644179.

REFERENCES

[1] S. ABBURU. *A Survey on Ontology Reasoners and Comparison.* International Journal of Computer Applications (0975 8887), Volume 57 No.17, November 2012.

[2] A. K. Bardsiri and S. M. Hashemi. *Qos metrics for cloud computing services evaluation.* International Journal of Intelligent Systems and Applications (IJISA), 2014.
[3] J. Branke et al., eds. *Multiobjective optimization: interactive and evolutionary approaches.* Vol. 5252. Springer, 2008.
[4] P. C. Brebner, *"Is your cloud elastic enough?: Performance modelling the elasticity of infrastructure as a service (iaas) cloud applications".* In Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering (ICPE), 2012, ACM, 263266.
[5] K. Deb, A. Pratap, S. Agarwal and T. A. M. T. Meyarivan. *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE Transactions on Evolutionary Computation 6(2), 2002, 182-197.
[6] J.J. Durillo and A. J. Nebro. *jMetal: A Java framework for multi-objective optimization.* Advances in Engineering Software 42(10), 2011, 760-771.
[7] W. C. Feng, P. Balaji, C. Baron, L.N. Bhuyan, and D.K. Panda. *Performance characterization of a 10-Gigabit Ethernet TOE.* In 13th Symposium on High Performance Interconnects, IEEE, 2005, 58-63.
[8] I. Goiri, J. Guitart and J. Torres. *Characterizing cloud federation for enhancing providers' profit.* In 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), 2010, 123-130.
[9] A. Iosup, R. Prodan and D. Epema. *IaaS Cloud Benchmarking: Approaches, Challenges, and Experience.* In Proc. of 5th Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS), 2012.
[10] K. Kaya, Kamer. *Heuristics for scheduling file-sharing tasks on heterogeneous systems with distributed repositories.* Journal of Parallel and Distributed Computing 67 (3), 2007, 271-285.
[11] G. Kecskemeti, A. Kertesz and Z. Nemeth. *Developing Interoperable and Federated Cloud Architecture.* IGI Global, 2016, 1-398.
[12] J. Opara-Martins, R. Sahandi and F. Tian. *Critical analysis of vendor lock-in and its impact on cloud computing migration: a business perspective.* J. Cloud Comput. 5 (1), 2016, Article 54.
[13] M. Placek and R. Buyya, *A Taxonomy of Distributed Storage Systems*, www.cloudbus.org/reports/DistributedStorage Taxonomy.pdf.
[14] K. Razavi and T. Kielmann, *"Scalable virtual machine deployment using vm image caches".* In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, SC 2013, ACM, 65:165:12.
[15] D. Villegas, N. Bobroff, I. Rodero, J. Delgado, Y. Liu, A. Devarakonda and M. Parashar. *Cloud federation in a layered service model.* Journal of Computer and System Sciences, 78(5), 2012, 1330-1344.

Edited by: Dana Petcu
Received: June 27, 2016
Accepted: August 16, 2016