LIVERPOOL JOHN MOORES UNIVERSITY

LJMU Research Online

Kimovski, D, Marosi, AC, Gec, S, Saurabh, N, Kertesz, A, Kecskemeti, G, Stankovski, V and Prodan, R

Distributed Environment for Efficient Virtual Machine Image Management in Federated Cloud Architectures

http://researchonline.ljmu.ac.uk/id/eprint/6642/

Article

Citation (please note it is advisable to refer to the publisher's version if you intend to cite from this work)

Kimovski, D, Marosi, AC, Gec, S, Saurabh, N, Kertesz, A, Kecskemeti, G, Stankovski, V and Prodan, R (2017) Distributed Environment for Efficient Virtual Machine Image Management in Federated Cloud Architectures. Concurrency and Computation: Practice and Experience. ISSN 1532-0626

LJMU has developed **LJMU Research Online** for users to access the research output of the University more effectively. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Users may download and/or print one copy of any article(s) in LJMU Research Online to facilitate their private study or for non-commercial research. You may not engage in further distribution of the material or use it for any profit-making activities or any commercial gain.

The version presented here may differ from the published version or from the version of the record. Please see the repository URL above for details on accessing the published version and note that access may require a subscription.

For more information please contact researchonline@ljmu.ac.uk

http://researchonline.ljmu.ac.uk/

CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2016; **00**:1–20
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/cpe

Distributed Environment for Efficient Virtual Machine Image Management in Federated Cloud Architectures

Dragi Kimovski[1,6*], Attila Csaba Marosi[2], Sandi Gec[3,4], Nishant Saurabh[1], Attila Kertesz[2], Gabor Kecskemeti [5], Vlado Stankovski[3] and Radu Prodan[1]

[1] *University of Innsbruck, Distributed and Parallel Systems Group, Innsbruck, Austria*
[2] *Hungarian Academy of Science, Budapest, Hungary*
[3] *University of Ljubljana, Faculty of Civil and Geodetic Engineering, Ljubljana, Slovenia*
[4] *University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia*
[5] *Liverpool John Moores University, Liverpool, United Kingdom*
[6] *University of Information Science and Technology, Ohrid, Macedonia*

SUMMARY

The use of Virtual Machines (VM) in Cloud computing provides various benefits in the overall software engineering lifecycle. These include efficient elasticity mechanisms resulting in higher resource utilization and lower operational costs. VM as software artifacts are created using provider-specific templates, called VM images (VMI), and are stored in proprietary or public repositories for further use. However, some technology specific choices can limit the interoperability among various Cloud providers and bundle the VMIs with nonessential or redundant software packages, leading to increased storage size, prolonged VMI delivery, stagnant VMI instantiation and ultimately vendor lock-in. To address these challenges, we present a set of novel functionalities and design approaches for efficient operation of distributed VMI repositories, specifically tailored for enabling: (i) simplified creation of lightweight and size optimized VMIs tuned for specific application requirements; (ii) multi-objective VMI repository optimization; and (iii) efficient reasoning mechanism to help optimizing complex VMI operations. The evaluation results confirm that the presented approaches can enable VMI size reduction by up to 55%, while trimming the image creation time by 66%. Furthermore, the repository optimization algorithms, can reduce the VMI delivery time by up to 51% and cut down the storage expenses by 3%. Moreover, by implementing replication strategies, the optimization algorithms can increase the system reliability by 74%.
Copyright © 2016 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Virtual machine images, Distributed VMI repositories, Cloud federation, VMI size optimization, VMI re-distribution

1. INTRODUCTION

Cloud computing technologies are deployed on top of physical hardware to provide highly optimized execution environments for *Virtual Machines (VM)* [1]. The use of VMs (or containers) represents an efficient way to automate the overall software operation lifecycle. For example, elasticity as a main property of Cloud systems usually leads to high resource utilization and thus, lower operational costs [5]. As part of the software engineering process, VMs enable isolated execution of multiple applications within their own environment. VMs are typically created using

*Correspondence to: Dragi Kimovski - dragi@dps.uibk.ac.at

Copyright © 2016 John Wiley & Sons, Ltd.

provider-specific templates, called *VM images (VMI)* and are stored in proprietary or public repositories for further use. However, some technology specific choices, such as virtualization environment and VMI format, can limit the interoperability among various Cloud providers and bundle the VMIs with inessential or redundant software packages, thus leading to increased storage size, prolonged VMI delivery, stagnant VMI instantiation and ultimately vendor lock-in. Furthermore, the utilization of centralized repositories, without considering the geographical and logical distribution of the VMIs, can aggravate these problems even more, especially in federated Cloud environments. Regrettably, current solutions do not provide substantial methods and tools for optimized operation of distributed VMI repositories, thus limiting the adoption of federated Cloud environments [3]. Multiple crucial constrains that largely influence the usability of federated Cloud infrastructures, in the sense of VMI repository operation, are: (i) manual, time consuming and error-prone VMI creation with proprietary templates, (ii) indivisible VMIs, containing unnecessary and redundant software packages, (iii) centralized and unoptimized VMI repositories, and (iv) lack of information to support the process of VMI and repository optimization [4].

To address these constraints, we present in this paper a set of novel functionalities and design approaches for efficient operation of distributed VMI repositories in federated Cloud environments. For this purpose, we developed distinctive software modules, specifically tailored for enabling: (i) simplified creation of lightweight and highly optimized VMIs tuned for specific application requirements; (ii) multi-objective VMI repository optimization; and (iii) efficient reasoning mechanism for streamline support of complex VMI operations. We implemented and integrated the introduced novelties as essentials components of the ENTICE environment [2] [†]. The ENTICE environment, currently under development, is based on an advanced architecture capable of receiving unmodified and fully functional VMIs from its users, and openly adapt and optimize them for specific Cloud infrastructures with respect to various criteria, such as size, configuration, and geographical distribution. ENTICE employs novel technologies to enable faster VMI loading and delivery, and optimized application execution with improved QoS. ENTICE gradually stores information about the VMIs and fragments in a knowledge base used for interoperability, integration, reasoning and optimization purposes. VMI management is supported by ENTICE at an abstract level, independent of the middleware technology supported by the Cloud infrastructures within the federation. To further shield the users from the complexity of underlying VMI repository technologies, ENTICE provides the flexibility for tailoring the VMIs to specific Cloud infrastructures. The ENTICE repository includes, among other features, techniques to optimize the size of VMIs while maintaining their functionality, automatically share similar parts of multiple images, such as identical operating system (OS) kernels and libraries, among various repositories located in different administrative domains, and optimally distributed them in response to application and data center requirements. In a broad sense, the ENTICE environment aims at providing a universal backbone architecture for efficient support of VMI management operations, regardless of the administrative boundaries between multiple distributed repositories.

The remainder of this paper is structured as follows. In Section 2 we describe the related work covering the state-of-the-art concepts in VMI repositories management. Afterwards, we present in Section 3 general use case scenarios for the ENTICE environment and its modules, provide further analysis and identification of the essential requirements. In Section 4, we present the architecture of the ENTICE environment and identify the basic functions required for efficient VMI management. Section 5 describes the VMI synthesis and analysis concepts and their related approaches, followed by the Multi-objective optimization framework with replication support in Section 6.2, and by the novel concepts concerning the knowledge base and reasoning mechanism in Section 7. We present in Section 8 experimental evaluation of the developed functional modules by considering multiple different scenarios. Finally, Section 9 summarizes the main achievements of this work, their overall impact and outlines future work directions.

[†] http://www.entice-project.eu/

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

2. RELATED WORK

The traditional VMI management systems are not able to provide support for complex VMI operations, such as federation wide VMI redistribution or VMI size optimization and similarity-based fragmentation, due to the highly dynamic demand for storing and managing VMI files in distributed Cloud environments. VMware is [24] repository system, which allows uploading and downloading VMIs at users' disposal. In addition, the VMware system offers a weak management system for locally stored VMIs within a single repository, and provides efficient tool for searching specific VMIs in accordance to the application requirements. Science Clouds [25] is another VMI-specific repository management environment, which allows downloading of pre-existing images, but evades the upload or indexing functionality of any user-specific VMIs. FutureGrid [9], an experimental system for high-performance computing and Cloud-based applications, provides another VMI repository management platform targeted at federated storage systems, which empowers the users to upload, download and update VMI's functionality with limited meta-data informations using a REST interface.

Apart from these systems, there are few VMI repositories such as *VMRC* [10], and Amazon Image Service [26], which come closest to solving significant part of issues pertaining to the efficient VMI management. On one hand, VMRC offers indexing of images via catalog functionality, while Amazon allows publishing and sharing of the most common VMIs with respect to appropriate authorization. However, Amazon utilizes proprietary image repositories, tying down the users to a specific Cloud hypervisor environment and VM templates. VMRC does not contribute to solving the interoperability issue either, and instead offers a unique VM matchmaking service for sharing images among the local users. Moreover, the locally centralized architectural model of these VMI repository systems does not provide scalable image distribution and induces delays in the VM provisioning. To this extent, none of the mentioned production systems contributes towards bridging the gap that limits the deployment of federated IaaS Clouds.

Alongside the popular production systems, there has been limited individual research work focused towards specific independent goals aiming at solving the VMI storage and distribution issues. Razavi et al. [12] optimizes the VMI storage cost based on a reduction of the image size to the bare minimum required for executing a given application, which improved the startup time of VM instances by up to three times. With respect to VMI distribution, Wu et al. [13] provided a theoretical work based on isolated and cross image distribution mechanisms optimizing the average distribution time of VMIs in the repository. Schmidt et al. [16] extended this discussion by including other structural mechanisms such as unicast, binary tree, bit torrent and multicast distribution of VMIs. They concluded that the peer-to-peer-based bit torrent mechanism is the most efficient, especially in the case of remote VMI distribution.

Although, there have been significant advancements in research with respect to VMI-related operations, some important aspects remain to be explored. Furthermore, there is no VMI management environment capable of supporting complex VMI operations in a transparent way. Hence, there is a need of a generic VMI repository environment, which gives transparency to its users with respect to automated contextualization, reduction and fragmentation of VMIs, and which optimizes the storage distribution with support of a semantic reasoning-based management.

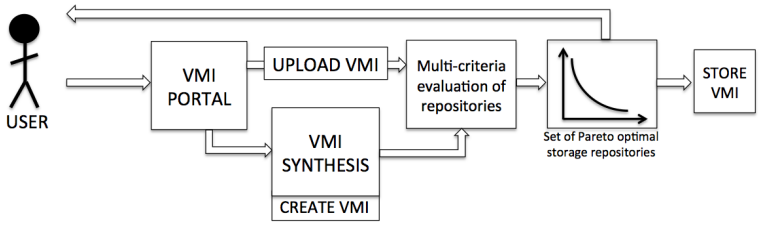
3. USE-CASE SCENARIOS AND REQUIREMENTS ANALYSIS

To tackle the issues that limit the possibilities for an effective Cloud federation and identify the most relevant requirements, we present in this section a broad use-case scenario of the proposed researched environment. Based on these requirements, we identify the imminent functionalities of the ENTICE architecture.

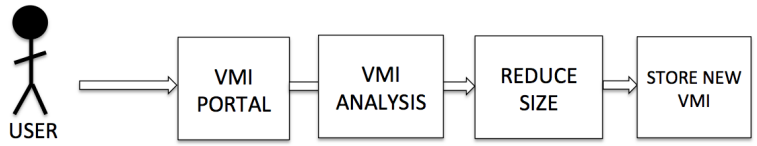
The typical use-case of the ENTICE environment starts with the user issuing a request for storing an unmodified and functionally-complete VMI through an easy to use user interface, enclosing various VM management operations. Subsequently, the environment transparently evaluates

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

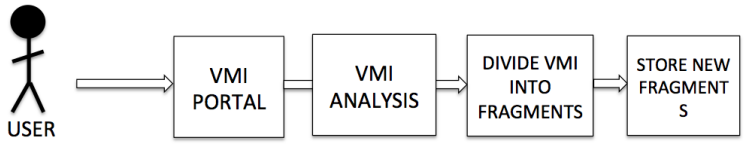
DOI: 10.1002/cpe



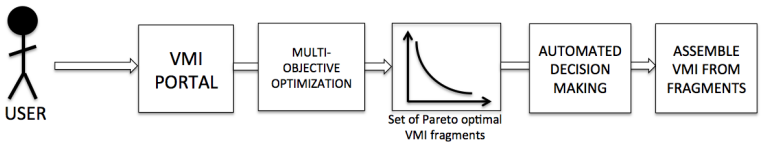
(a) VMI upload.



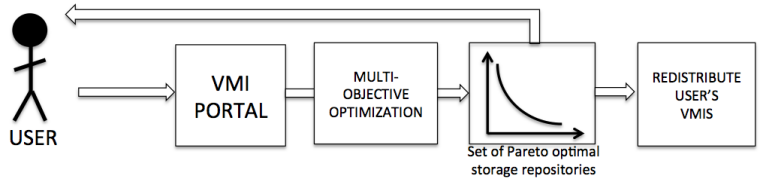
(b) VMI size reduction.



(c) VMI fragmentation.



(d) VMI assembly.



(e) VMIs redistribution.

Figure 1. Use-case scenarios for the ENTICE environment.

multiple different storage repositories where the VMI can be initially stored. The evaluation is conducted based on multiple conflicting criteria, such as the cost for storage or the delivery time. This process results in a set of "optimal" trade-off solutions, i.e. initial storage repositories, from which the user should choose one. Alternatively, the user can choose to directly create a new highly optimized VMI by utilizing generic predefined recipes and novel synthesis techniques (see Figure 1a), instead of uploading complete VMI.

In general, user-specific images are largely sized and comprise various redundant and unessential packages incurring high storage cost and increased instantiation overheads. Henceforth, the user may require for the newly uploaded VMI to be optimized in size, achieved in ENTICE by removing redundant packages and files from the VMIs through efficient analysis techniques (see Figure 1b). Furthermore, the VMI analysis enables discovery of identical portions in apparently unrelated VM images, such as identical OS kernel or library coming from different users. Based on the

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

similarity information, the environment may allow spiting of VMIs into multiple fragments for storing the frequently shared components only once (see Figure 1c). If the user requires for a specific image to be deployed in a computational Cloud, the ENTICE environment employs multi-objective optimization techniques for fast assembly of the previously identified fragments in a stand alone VMI, and converts it in the appropriate Cloud template (see Figure 1c). To reduce the storage costs or to increase the deployment performance, the user can also require improved transparent redistribution of the images in the ENTICE repositories (1e). The process itself can be conducted online during application execution, or offline while the VMIs are not being utilized or optimized. The VMI redistribution is proceeded by a Service Level Agreement (SLA) capable of properly defining multiple optimized trade-off solutions. The composite tradeoff SLA is then provided to the user who acts as a decision maker in the optimization process.

Based on this general use-case scenario, we identified the requirements for the ENTICE environment divided in three broad categories:

- *VMI synthesis, analysis and fragmentation requirements* enclosing the specification of the recipe-based VMI creation, VMI size reduction, and VMI fragmentation and assembly;
- *repository optimization and VMI distribution requirements* covering the needs for more efficient initial distribution of newly uploaded VMIs, redistribution of user specific VMIs, online VMI assembly and VMI interoperability support among various Cloud providers;
- *knowledge base and reasoning requirements* encapsulating the proper storage of relevant data for efficient VMI analysis and fragmentation. We extended these requirements for VMI redistribution and constraint-based multi-objective optimization with SLA support.

4. ENTICE FUNCTIONALITIES AND SYSTEM ARCHITECTURE DESIGN

To satisfy the design requirements, the ENTICE environment transparently supports various complex functionalities, without requiring deep involvement of the users. We identified and classified the functionalities for the ENTICE environment into three distinctive categories by systematically analyzing the use-case requirements:

- *VMI synthesis, analysis and fragmentation functionalities* including (i) recipe-based VMI synthesis, (ii) VMI size optimization by removing OS parts that are not required and have not been defined in the functional description, (iii) decomposition of VMIs in multiple fragments with respect to the required functionalities, removing duplicated software packets across multiple images, and (iv) assembly of fragments into complete VMIs during the deployment stage;
- *Multi-objective VMI optimization with replication functionalities* including (i) optimized distribution of the VMIs and their fragments across distributed storage sites, (ii) optimized extraction of fragments based on functionality for VMI assembly during deployment, (iii) VMI movement from one storage location to another, and (iv) location tracking of each VMI to enhance storage and distribution;
- *Knowledge-based VMI portal and reasoning functionalities* including (i) an easy interface for the users to access the available VMIs and search for fragments, (ii) VMI upload and update functionality for the streamlined provisioning of user images, (iii) VMI management and graphical analysis tool, which allows the users choose SLAs from multiple optimized metrics provided by the multi-objective optimization, (iv) functionality to sort VMIs with respect to user-defined metrics, and (v) easy interface to user profile management.

The architecture of the ENTICE environment is modular and highly decentralized in nature, encapsulating variety of different components which interact by exchanging structured information on the available services. Each component in the system provides specific functions that either support a set of complex VMI operations, or enable proper operation of the other components. The

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

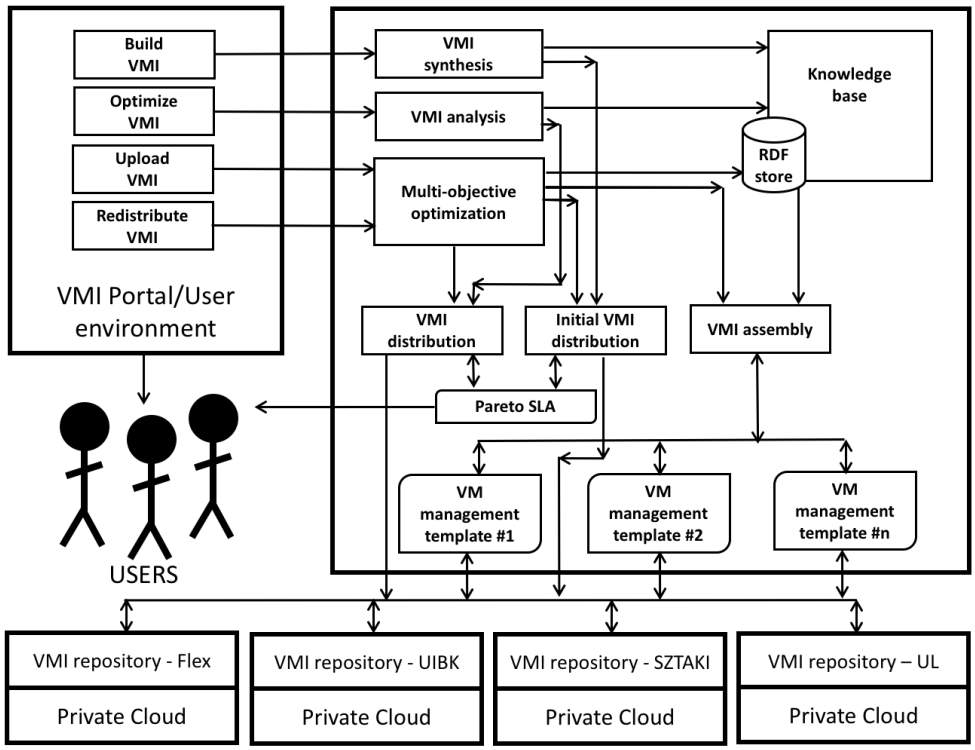


Figure 2. ENTICE functional architecture.

communication between the software components is established by utilizing a RESTful protocol, while the transfer of the VMIs and their related datasets is based on a secure HTTP protocol. We depict in Figure 2 the current state of the ENTICE architecture, together with the corresponding software components:

- *Client-side VMI portal* through which the users interacts with the environment and initiates various complex VMI operations, such as VMI synthesis, VMI upload, VMI size optimization, or VMI redistribution. Furthermore, the VMI portal, provides output information to the users and efficient tools for decision making on the SLA contracts;
- *Distributed services* enabling complex VMI operations, distributed across multiple geographical regions. For example, we deployed the multi-objective optimization framework, together with initial VMI distribution and VMI redistribution modules, in the premises of the University of Innsbruck in Austria, and integrated the VMI synthesis and analysis tools within the production class public Cloud of the Hungarian Academy of Science in Budapest. All these services are supported by a knowledge base and advanced reasoning mechanism located at the University of Ljubljana in Slovenia;
- *VMI repositories* distributed on various locations across Europe.

In the following, we provide an extensive overview of the distributed services, enabling the complex VMI operations, followed by performance and functionality evaluation.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

5. VMI SYNTHESIS AND ANALYSIS

As introduced in Figure 1 of Section 3, VMI synthesis and analysis is an important case in ENTICE. By defining these processes, our goal was to identify a simple methodology to fragment (i.e. decompose) a monolithic application alongside its sub-service boundaries. These sub-services can act as small microservices that can later be composed into other services without the need of the entire monolithic application. To achieve this, we use image synthesis and image analysis methods, which both have pivotal roles within the ENTICE architecture. We previously presented our approach at a high-level of abstraction [18], while in this section we further elaborate and describe these technical mechanisms. Furthermore, we present experimental evaluation in Section 8.1.

The ENTICE environment offers *VM management templates (VMMT)* to be stored in the repositories of the connected Cloud systems that allow the fragmented VMIs to be reconstructed at runtime. For optimal VM instantiation performance, the templates are formulated as stand-alone VMIs or container images (CI) to access and build fragments from the distributed repository. After a VM is instantiated using a VMMT, it ensures that fragments (needed for a particular functionality specified by the user) are placed and enabled for use in the instantiated VM. VMMTs even allow customization of files and directories for specific VMs to serve the needs of various stakeholders.

The VMI synthesis enables users to build new images using several approaches. First, it allows the use of generic user-provided images or software recipes to act as the foundation before specializing them into microservices. Second, VMI synthesis cooperates with the ENTICE image portal to identify the functional requirements to be met by a newly created image (on a microservice basis, thus resulting in a new image for every functional requirement specified). Finally, our synthesis tool modifies the generic (original) images either directly (by altering the image files) or indirectly (through alternative recipes that lead to more compact images). These alterations aim at removing content from the original VMIs that lead the generic images towards their single purpose (i.e. the functional requirements listed in the portal). For the optimized VMIs, the VMI synthesis offers image maintenance operations, allowing software updates to be done on the original image and transforming them to the optimized image.

Alongside synthesis, ENTICE also delivers VMI analysis allowing the discovery of equivalent pieces in apparently unrelated VMIs, possibly originating from different users and communities. Analysis operates independently of the Cloud provider where the image is stored. Equivalence information is then stored in the ENTICE knowledge base for later use. ENTICE also allows decomposing VMIs into smaller fragments for storage of the common image components only once (e.g. same Linux distribution used by two unrelated images). Fragmenting fosters the VMI distribution and enables the optimization of overall storage usage in the repository.

5.1. *Recipe-based VMI synthesis*

This subsection presents the ENTICE functionalities enclosing the creation of new highly optimized VMIs by using user-specific functional recipes, expected to guide the creation of the users' original monolithic service in a generic way.

The recipe-based image synthesis process contains seven steps, depicted in Figure 3, starting with creating an image and ending with an optional cancellation request. The synthesis service also has a backend component that uses other subcomponents to create the requested images. The images that are managed in these processes may be regular VMIs or containers and the content can also vary from microservices to complex ones. As their name suggests, microservices in containers have smaller footprints and are easier to optimize. The REST API enables the following processes:

1. submit build requests;
2. retrieve results of finished builds;
3. query the status of ongoing and completed builds;
4. cancel the in-progress builds.

The image creation process at the backend consists of two parts: the building phase and the testing phase.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

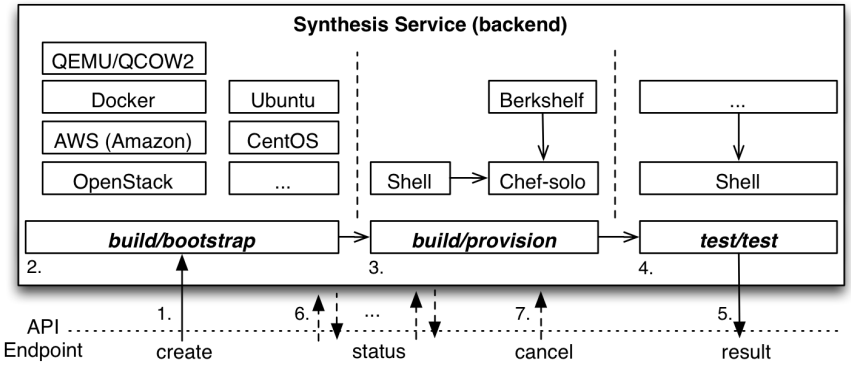


Figure 3. Process of recipe based image synthesis

The *building phase* is initiated by the API call (step 1) by specifying a parametrized build target, the service description for provisioning, and the tests for the testing phase. The first part of the building phase is *bootstrapping* (step 2), responsible for making a base image from one of the following possible ways:

- starting from an empty disk image and building it locally (via QEMU/QCOW2);
- targeting a container (e.g. Docker);
- building an existing image from a repository via a supported cloud (e.g., OpenStack), with no bootstrapping involved.

The second part of the building phase is *provisioning*, which installs the requested microservice using the specified description in two possible ways: using a custom shell script provisioner (via Packer) containing all the steps to be executed, and using a provisioning tool like Ansible or Chef(-solo) containing cookbooks. These provisioners may be used together when needed, for example performing housekeeping via shell and provisioning the microservice components using Chef.

Different services may require vastly different approaches and tools for testing thus, the *testing* phase allows flexible methods. Here first the image is duplicated, and then a supplied test script executed in the new image. The testing methods can be of any type, a non-zero exit status representing an error. The script may deploy any components via the package manager of the Linux distribution used (e.g., APT or YUM) and a user supplied custom zip file may contain additional components as well, but all other external access is prohibited. Upon completion of the tests the copy of the VMI used for testing is discarded, while the build API provides access to the original image for download. Currently there is no option to link the image to another location or repository, this feature will be considered for future work. Our current implementation uses Packer [21], but our initial experiments were carried out using ImageFactory [20].

5.2. *VMI Size Optimization*

The VMI size optimization functionality, presented in Figure 4, is executed after the recipe based synthesis completed with a monolithic VMI. These monolithic images are referred here as original images of an application, denoted as *O. Img.*. Usually these monolithic images implement a combination of several functionalities. Therefore, the user can use the ENTICE environment to transform the original image into an optimized one, which keeps the required functionality only (resulting in a microservice). Before the microservice can be extracted from the original image, the image creator must prepare a functionality test for the required microservice in the form of an independent, self-contained shell script that must utilize all features of the required microservice. In general, unit tests of the original application represent a good start for the functionality test. Although these tests need manual preparation, they are sufficient for proof-of-concept scenarios. In the future, we will develop techniques to enable semi-automatic functionality test generation too.

The transformations used for VMI size optimization are performed in seven steps:

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

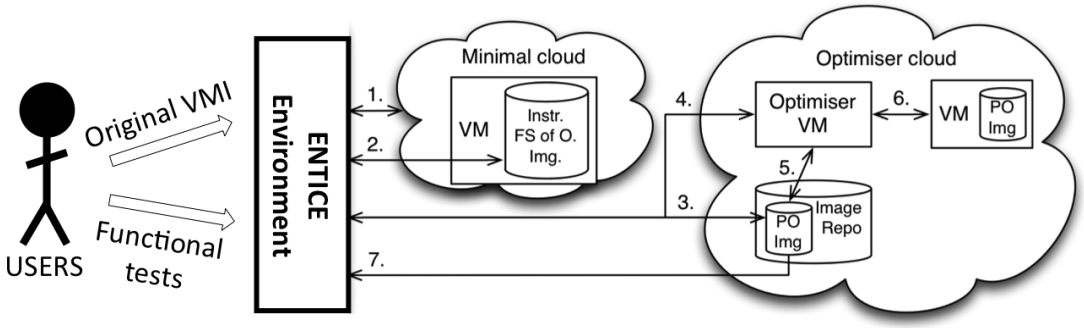


Figure 4. VMI transformation steps for host only the intended microservice.

1. instantiate the original image in a minimized Cloud infrastructure;
2. execute user tests and collect the accessed files;
3. create a partially optimized image file;
4. deploy the optimizer VM;
5. deploy the partially optimized image, and remove further unnecessary parts;
6. run functionality tests on the optimized image;
7. upload the optimized image to the ENTICE environment.

Once the test is uploaded to the ENTICE portal, the pre-evaluation phase starts. In step 1, the original image is instantiated in a minimized Cloud infrastructure maintained by the ENTICE environment that runs the new VMI and instruments its file read operations (i.e. *Instr. FS* in Figure 4). The functionality test is then ran on the recently created VM/container. While this test runs, ENTICE collects the files accessed from the original image in step 2. If this initial test fails, the collected data is discarded and the user is notified about the incorrect test result for the original image (suggesting that the actual test requires more functionality than the original image offers). Upon success, the collected list of files are stored in a so-called *restricted list*. We assume that files not on the restricted list are not relevant for the microservice identified by the functionality test.

In step 3, part of the image optimization phase, the system creates a partially optimized image (*PO image*) that only contains the listed files (i.e. non-referenced files are erased). In step 4, an *Optimizer VM* is deployed and contextualized to use the PO image, and perform the optimization procedure. In step 5, The Optimizer VM starts by altering the PO image (i.e. remove some further contents from it) before running the test script. The removed parts of the image are expected not to be relevant for the microservice's intended functionality, but are assumed to belong to the background activities of the original image instead (e.g. startup functionalities and periodic tasks unrelated to the mircoservice's functionality). Since in this paper we present and describe the methodology of the ENTICE environment, we do not introduce specific selection techniques to applied on the parts of the PO image for further optimizations.

Once the PO image is altered by the removal of further files, the Optimizer VM uploads the new image to the Cloud in step 6 and tests the VMI by instantiating it and evaluating its functionality via the user-provided shell script. Success in the evaluation leads the optimizer to use the altered image as the new PO image. If the evaluation fails, other candidate files are selected for removal from the previously examined PO image. We repeat the optimization steps 3 to 6 until the user-defined cost limits are reached, or no more VMI components can be selected. By the end of this step, the final PO VMI will contain only the intended functionality of the microservice, which is returned to the ENTICE environment in step 7.

The microservice in the optimized image offers a subset of features or functionalities of the original VMI. It is expected that the creator of the original VMI modifies the service interface afterwards to accommodate this. This is a manual operation and the optimization phase must be re-run with the altered interfaces to validate its usability. As the size optimization module reuses the

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

past files selection results (i.e. the ones successfully removed in step 5), this second optimization phase will be executed much faster.

6. MULTI-OBJECTIVE VMI REDISTRIBUTION WITH REPLICATION SUPPORT

Cloud providers currently store VMIs in proprietary centralized repositories without considering application characteristics and their runtime requirements, causing high delivery overheads. The ENTICE environment evolves from the typical storage systems, and focuses on providing means for transparent distribution of VMIs in distributed storage repositories by considering application characteristics and usage patterns. To achieve this ambitious goal, we researched a generic multi-objective optimization framework for VMI redistribution, targeting a number of important objectives such as performance-related, operational cost and storage size, applied to several particular problems. The researched framework utilizes unified multi-objective optimization module, branched in three distinctive sub-modules tailored for specific optimization tasks [8].

To support the optimized VMI distribution, it is essential to consider multiple performance and financial objectives in the upload stage. To this end, the process of initial VMI upload triggers the optimization framework to search for "best" upload repository according to user-specific requirements. The framework utilizes concepts from the field of multi-criteria evaluation to determinate the repository sites where the VMIs or associated data sets can be initially stored.

To reduce the VMI delivery times for complex requests, the framework optimizes the VMI distribution across multiple repositories in the background using a purposely designed optimization module that considers VMI use patterns to redistribute the images according to multiple conflicting objectives. The optimization algorithm provides multiple tradeoff solutions, where each solution represents a possible mapping between the stored images and available repository sites.

Furthermore, we investigated the use of replication of the highly demanded VMIs during the offline optimization, when the size of the data to be accessed on the current VMI repositories is in the order of Terabytes and soon expected to reach Petabytes. Therefore, ensuring efficient access to widely distributed VMI and their associated datasets is a serious challenge that needs to be addressed. Creating replicas to a suitable site by implementing a VMI replication strategy can increase the system performance, shorten the data access time, and increase the system reliability.

To properly distribute the VMIs in a federated environment, the optimization framework depends on a careful analysis of the repositories use patterns based on a knowledge base for storing and processing information on the VMI activities, and providing the collected data in a proper format. The framework provides a SOAP-based API through which the decision maker can acquire the set of optimized trade-off solutions using a graphical representation, thus reducing the complexity of the VMI storage management process.

6.1. Offline redistribution with replication support

The VMI redistribution framework utilizes NSGA-II genetic algorithm to perform repository optimization [7]. The process of offline VMI redistribution, depicted on Figure 5, is conducted on the basis of the VMIs usage patterns and the communication performance between the storage repositories in the ENTICE federation and the computational clouds where those VMIs have been deployed. To begin with, the usage patterns, stored in the knowledge base, are analyzed in order to determinate which VMIs have to be replicated. Consequently, this information is then utilized for the modeling of the individuals, as required by the NSGA-II algorithm.

As with any other population based genetic algorithm, individuals are used to represent and evaluate multiple possible solutions to the problem. For the purpose of VMI redistribution, we model each individual as a vector with a size equal to the number of VMIs that have to be redistributed and replicated. The values stored in each vector field corresponds to a storage repository. Therefore, every vector contains a unique mapping between the redistributed VMIs and replicas to the available storage sites in the federated repositories. Figure 6 depicts the vector structure for a

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

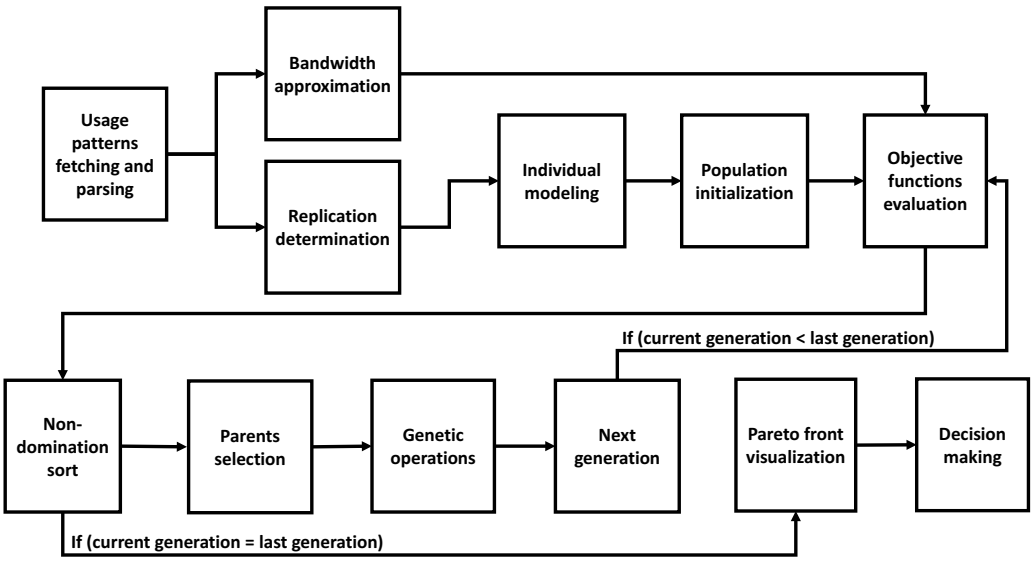


Figure 5. VMI redistribution framework flow-chart

sample individual containing seven original VMIs and three replicas. Afterwards, we initialize the population of the NSGA-II algorithm by randomly assigning a storage site to each VMI in every individual. Next, every individual is evaluated based on three conflicting objective functions (see Section 6.2). For each VMI i mapped in particular individual, we calculate the performance P_i (represented as delivery time) and the cost C_i. The average delivery time and cost objectives across all VMIs give the overall fitness of an individual, while the VMI reliability objective is calculated per whole individual.

After evaluating and sorting the initial population based on dominance [11], we apply mutation and crossover operators over the parents to create a children population of the same size. We then evaluate the children population, merge it together with the parents, and sort the grouped population according to the non-dominance criterion. The best individuals for the next generation are then selected from the grouped population. We repeat this process until the maximal number of generations is reached. The final non-dominated trade-off solutions are then visualized in the form of Pareto front to the administrative entity of the federation, which acts as a decision maker and selects the most appropriate distribution based on a predefined decision making policy.

6.2. *Objective function modeling*

The optimization problem of offline VMI redistribution with replication consist of a finite, although very large number of possible alternatives. The identification of the "optimal" set of distribution possibilities requires proper modeling of the conflicting objectives: cost for storing and transferring VMIs, system reliability, and VMI delivery time. We performed the modeling of the objectives by analyzing the VMI use patterns, resulting in an optimized image distribution across multiple distributed repositories.

6.2.1. VMI cost objective To model the cost objective, we use the notion of the financial resources required to store particular VMI in a given repository site $C_{storage}$ and the cost for transferring the data from the initial to the new repository in the federation $C_{transfer}$. For each VMI, we calculate the cost objective by adding the two components: $C = C_{storage} + C_{transfer}$.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

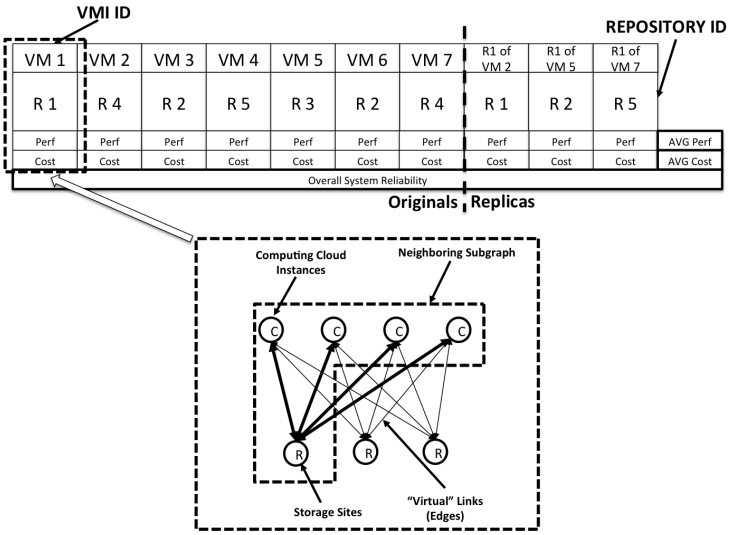


Figure 6. Individual vector for the VMI redistribution algorithm.

6.2.2. VMI delivery objective We modeled the VMI delivery objective by approximating the average bandwidth between two known points in the Cloud federation. Utilizing the raw theoretical bandwidth of the interconnecting structure to describe the factual communication performance is not a suitable approach, as it is difficult in practice to predict the actual route the packets may take through the interconnection network to reach the destination, which increases the performance uncertainty. Furthermore, the load on the intermediate communication channels can deviate, rendering the theoretical communication performance unsuitable for the researched purpose. To solve this problem, the optimization framework leverages the VMIs use patterns stored in the knowledge base to perform close approximation of the actual bandwidth between any pair of nodes in the federation. We estimated the virtual link bandwidth B_{rc_i} based on existing transfer events previously recorded by the monitoring module. We use the monitoring data to determinate if there have been previous VMI or data transfers between a given cloud and a repository, and take the average value of the bandwidth across all transfers a performance metric for the virtual link.

The introduction of virtual links allows for an undirected fully connected weighted graph to be modeled between a particular storage repository and all registered computational Clouds. In this graph, the vertices correspond to either a single repository site or a computational Cloud instance and the edges are represented by the "virtual" links. The weights of the edges correspond to the estimated average bandwidth B_{rc_i} on the corresponding virtual links.

To properly model the delivery objective for storing a VMI in a given storage repository, we introduce so-called *adjusted delivery function* P, which considers the total number of downloads of single VMI from particular repository to all Clouds D_{tv} and the number of downloads to exact neighboring Cloud D_i. The ratio of those two values is then multiplied with the estimated bandwidth of the corresponding virtual link to adjust for the delivery frequency relevance. We repeat this process n times until the adjusted delivery has been calculated for every virtual link emerging from a given repository. Lastly, the average value from this process is weighted as the delivery objective for storing a single VMI in an exact repository:

$$P = \sum_{i=1}^{n} B_{rc_i} \cdot \left(\frac{D_i}{D_{tv}} \right). \tag{1}$$

6.2.3. VMI reliability objective Introducing the concept of replication for VMI redistribution encloses multiple different aspects, as it can affect both the system performance and the VMI

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

availability. Consecutively, we propose a specific replication strategy focused on increased availability and reduced access time to the most frequently used VMIs.

To determinate the VMI to be replicated, we primarily consider the download frequency. First, the optimization framework calculates the median download frequency $D_{f_{med}}$ for all VMIs stored in the ENTICE environment. Afterwards, the difference between the median and maximal value for the download frequency is determined and multiplied by the replication coefficient k, where $0 < k < 1$. If $k = 0$ none of the VMI images will be replicated. If $k = 1$ on the other hand, half of the stored VMIs will be replicated by factor of one. The value of k has direct influence on the number of VMIs that will undergo the replication procedure, directly influencing not only the availability of the VMIs, but also the cost and performance objectives. Due to these reasons, we decided to isolate the system reliability as a separate objective and to consider the influence of the replication in the evaluation of the performance and cost objectives. In cases when a specific VMI needs to be replicated, the replica is treated in the same way as the original image in context of the cost and performance objectives and the objective values are calculated in accordance with methods presented in the previous Section.

The VMIs reliability R_s is then evaluated as a separate objective by taking into account both the original and the replicated VMIs. To calculate the reliability of all VMIs stored in the federation, we rely on the reliability of each repository in the ENTICE environment in relation to the replication coefficient and the storage mapping. In the general case, the reliability objective is calculated by utilizing the concept of series and parallel systems reliability [19]:

$$R_s = \prod_{i=1}^{o_{nr}} R_i \cdot \prod_{j=1}^{r} \left(1 - \prod_{k=1}^{N_r} (1 - R_i) \right), \tag{2}$$

where R_i is the reliability of the repository where the given VMI will be stored, o_{nr} is the number of original VMIs that will not undergo replication, r is the number of VMIs that will be replicated, and N_r is the number of replicas to be created for a single VMI.

7. KNOWLEDGE BASE REASONING AND DECISION MAKING

An important hypothesis of this work is that knowledge management methods can help address a variety of problems stemming from the wide range of needed functional and non-functional properties of software artifacts (such as VMIs) and the complexity of the federated repository in which they are stored and managed. Thus, the choice of using a knowledge base for storage and complex querying and even inference of new knowledge from the data is the most suitable in comparison to relational and other types of storage technologies.

The implementation role of the knowledge base in the overall system consists of three aspects:

1. design of an ontology describing the entities used in the ENTICE environment, which includes models of functional and non-functional properties of VMIs, models of the individual storage repositories and so on;
2. development of a knowledge base web service that supports CRUD (create, read, update and delete) operations to manipulate RDF-based instance data, and
3. implementation of reasoning mechanisms to automate the process of using the knowledge.

Moreover, the knowledge management methods are reflected in the VMI portal and support the multi-objective optimization framework.

This section is logically split into two parts. The first part focuses on high-level technological concepts relevant to ENTICE, which include ontology with reasoning in a form of objects, properties and relations between them, as well as an architectural design. The second part describes the implementation details, including the selection of the technologies and important issues and challenges related to them.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

7.1. Conceptual modeling

Initially, we built an ontology that is a conceptual model of all the entities, their attributes, and relationships in the complex ENTICE environment. The ontology design is based on the identified use cases, their data flow, and data definition of the service behavior. Instance data for the Knowledge Base can be generated and retrieved from different sources, for example, from the VMI portal, from the individual Cloud repositories and their SLAs, and similar. All these entities need be modeled with the ENTICE ontology, designed to also support temporary lifecycle states and statistics. Some of these data has to be stored persistently for future decision making, reasoning, and algorithmic calculations.

ENTICE gradually stores information related to all the concepts in the Knowledge Base used for interoperability, integration, reasoning, and optimization purposes. These concepts could be Cloud resources (e.g. ENTICE components and their environmental functionalities, storage resources, VMIs and their functionalities), programming concepts (e.g. storage complexity, taxonomy of functional properties), virtual organization concepts (e.g. privileges, credentials, ownership), resource negotiation-related concepts (e.g. SLAs), and desired level of QoS-related features (operational costs, elasticity, storage use). In ENTICE, the Knowledge Base is in charge of storing the taxonomies, thesauri and ontologies used for describing the semantic models of all entities like VMIs and repositories, as well as for linking them with already published ontologies. The ENTICE Knowledge Base as a triple-store is able to not only retrieve pure data, but also to represent meaningful information and knowledge. Key entities and concepts of the developed ontology are shown in Table I.

Table I. Important entity concepts modelled in the ENTICE ontology.

ENTITY NAME	DESCRIPTION
DiskImage, VMI, CI	Representative attributes of VMIs (e.g. name, version, encryption etc.) and relationship interconnections with entities such as other DiskImages/Fragments that are predecessors, DiskImageSLA, user, functionality, quality, OS and Pareto frontier. This entity is one of the most relational-descriptive entities in ENTICE
Pareto	Pareto stage objects (distribution possibilities) for online and offline redistribution achieved as a result of applying the multi-objective optimization framework
Repository	Available Cloud repositories characteristics.
Fragment	Preliminary description of fragments
OperatingSystem	Carious types of VMI OSs supported by ENTICE.
Data	content delivered alongside the VMI (e.g. delivery of service calculations in different format types such as images, text files and binaries) required by some use cases
User	User information including stakeholder types
Geolocation	Detailed geographical information about the repositories, obtained using the geographical database GeoNames [23]
DiskImageSLA	Information on the potentially new and resulting SLAs to support the overall lifecycle of applications deployed as VMIs and to facilitate automatic optimization, setup and management in federated Cloud environments
Quality	Parameter and result information about VMI synthesis and analysis
Delivery	Overall VMI deployment event consisting of the actual QoS, QoE and SLA

The knowledge management and information supply play a crucial role in the operation of the overall ENTICE environment. An example where the use of the Knowledge Base is

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

particularly effective is the support for solving some NP-hard optimization problems, addressed by a combination of multi-objective and SLA techniques. NP-hard optimization problem, for example, is to find the optimal distribution of a specific VMI across the distributed repositories, which allows for a specific minimum delivery time at a specific cost as presented in Section 6.2.

In relation to the implemented Knowledge Base, the design includes the integration of the following subsystems:

- VM Image synthesis notifies the Knowledge base with all the lifecycle phases of the service including the execution parameters and actual computation times. The Knowledge Base stores the metadata in the ontology entities to assess the approximate time needed to execute operations of the service in the future executions;
- VMI analysis uses the Knowledge Base for storing all the information about the fragments, and for offering enriched metadata information through the reasoning mechanisms by identifying redundancies, conflicts, and new relationships among the fragments;
- Multi-objective optimization stores the Pareto metadata in the Knowledge Base and exploits its reasoning mechanisms to optimize the multi-objective optimization process. More concretely, the data set needed for the optimization is filtered in the Knowledge Base through reasoning mechanisms (e.g. fragment characteristics, geolocation information) and SPARQL purposive queries. By doing this, the filtered results reflect a subset of optimized solutions that significantly reduces the optimization execution time.
- VMI assembly retrieves all information of an VMI needed to be constructed from the fragments. A crucial research aspect of the Knowledge Base is how to use the available ontology information (e.g. SLA, bandwidth, Cloud characteristics, fragment positioning) to propose an optimized assembly plan for the VMI assembly service.

The Knowledge Base is presented as a REST service based on the Jersey framework [22], and uses the Jena Fuseki technology, an RDF database described with the Web Ontology Language (OWL). Among the available framework functionalities, additional reasoning tools needed to be integrated (e.g. HermiT, Pellet etc.) to fully exploit the data manipulation, data validation, introduction of new individual relational characteristics, and other mechanisms.

8. EXPERIMENTAL EVALUATION

We present in this section a broad experimental evaluation of the researched methods and techniques for efficient VMI management in federated Cloud environments though a set of evaluation indicators, specifically selected for each individual VMI operation covering: (i) the VMI synthesis and analysis module and (ii) the multi-objective VMI redistribution and replication module, in tight interaction with the knowledge base and reasoning mechanism.

We conducted the evaluation experiments on a functional testbed distributed across various geographical locations. We deployed the VMI synthesis and analysis module at the premises of the Hungarian Academy of Science, which provided four physical nodes totaling 176 physical cores and 128 GB of memory per node. The multi-objective framework is located at the University of Innsbruck, which includes seven physical nodes, totaling 168 physical cores and 32 GB of memory per node. Lastly, the Knowledge base is located at the University of Ljubljana including 4 physical nodes, each node offering eight physical cores and 8 GB of RAM per node. The physical interconnection between the modules takes place over dedicated optical links, while the logical communication between the modules utilizes RESTful and SOAP protocols.

8.1. VMI synthesis and analysis

As introduced in Section 5, VMI synthesis offers image maintenance operations. In order to evaluate this feature of the ENTICE environment, we used two VMIs provided by the project industrial partners: WordPress 4.3.1 on CentOS 7.2 from Wellness Telecom and Redmine on CentOS 6.8 from

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

FlexiOPS. Table II summarizes the size of the original images, recipe-based created images and the optimized images. According to these results, we observe that the contents of the WordPress image have been reduced by 55%, while the contents of the Redmine image by 30% after five iterations of optimization steps. The build time of each recipe-based images is 17 minutes.

Metric	*WordPress 4.3.1 (CentOS 7.2)*	*Redmine (CentOS 6.8)*
Original image contents (Bytes)	1 548 718 080	2 273 243 136
Recipe-based image contents (Bytes)	1 995 644 433	1 550 196 435
Optimized image contents (Bytes, 5 iterations)	860 170 897	684 330 624
Optimized image size (Bytes, QCOW2 format)	1 269 628 928	1 146 617 856
Original build time (minutes, estimated)	60	60
Recipe-based build time (minutes)	17	17

Table II. Image synthesis evaluation results.

Figure 7 presents the results of the iterative VMI size optimization. We observe how the content sizes of the images were iteratively reduced during the optimization steps for the two cases. The results show that the first three iterations achieved higher amount of content reduction, while the last two could only improve with a small amount of additional bytes.

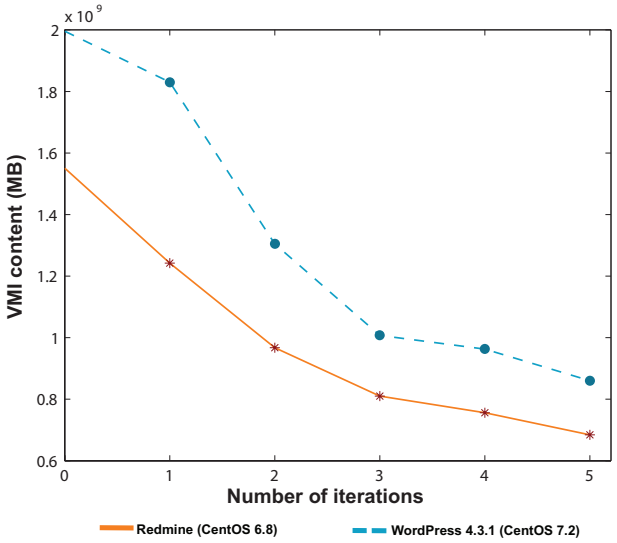


Figure 7. Wordpress and Redmine optimization progress: image contents in million Bytes (MB) after the different optimization iterations (0-5), where iteration 0 represents the recipe-based built image

8.2. *Multi-objective VMI repository optimization*

The essential feature of the proposed multi-objective optimization beyond the state-of-the-art is the possibility for replication of the most frequently used VMIs during redistribution. We therefore focused the evaluation on identifying the optimal value of the replication coefficient k with respect to required execution time of the optimization algorithm. Figure 8 provides an insides on the execution time of the optimization algorithm in regards to multiple different values of k. For the current evaluation scenario we created synthetic benchmark data stored in the knowledge base which includes 100 repository sites and computational Clouds, and an identical number of VMIs. We configured the optimization algorithm with an initial population of 100 individuals and 10000 separate evaluations/generations.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

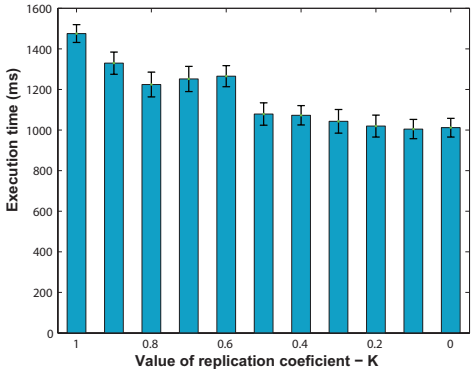


Figure 8. Optimization algorithm execution time for different values of k

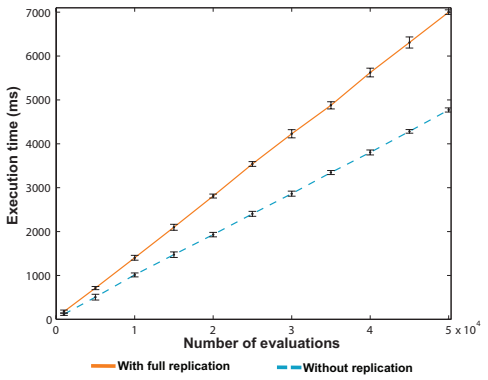


Figure 9. Optimization algorithm execution time for different numbers of individual evaluations.

The evaluation results show that, when the replication coefficient is below 0.5, the execution time of the optimization algorithm increases by only 10%. This implies that, if 25% of the most frequently used images are required to be replicated, the effect on the optimization execution time is minimal.

As we are dealing with an evolutionary multi-objective algorithm, it is crucial to asses the algorithm's scalability with respect to the number of separate evaluations of the individuals within the population. Figure 9 provides a comparison between the execution times of the optimization framework without support for replication [2] and with replication support. The benchmark data and population size used in this test are identical to the previous one. Both optimization algorithms scale linearly, with a relatively constant performance difference of around 40% in the case of no replication.

Furthermore, the optimization framework with replication support has been evaluated on the bases of the quality of the Pareto optimal solutions for different values of the genetic parameters, such as number of individuals and number of evaluations/generations. The assessment was conducted for a specific scenario enclosing the re-distribution of 100 VMIs with replication coefficient $k = 1$. The hypervolume indicator [14] has been used to compare the quality of the separate solutions provided during the experiments. Figure 10 presents a comparison of the mean value and standard deviation of the quality of the solutions in contrast to the number of utilized individuals and performed evaluations per experiment. By analysing the results, it can be concluded that in the cases when low computational time is required, it is essential to reduce both the number of individuals and performed evolutions, thus guaranteeing best computational performance to quality ratio. Contrary, when the computational time is not and issue, like in the case of off-line VMI redistribution, higher populations sizes with increased number of evolutions should be used. For example, in the case when 100 individuals are being utilized, we can expect best quality of the solutions when less then 5000 separate individual evaluations are performed. Above this threshold, the low number of individuals can easily lead to low diversity and reduced search capabilities, thus producing solutions with limited hypervolume values. In the case of ENTICE, the re-distribution and replication are performed off-line, thus requiring high population size and increased number of evolutions. To better illustrate the difference in quality among the solutions, on Figure 11 we present a comparison between the best solutions gathered from the experiments with 100 and 500 individuals.

Lastly, in Table III we provide a comprehensive review of the exact objective values for the optimal trade-off distribution solutions calculated by the optimization framework. Moreover, a comparison has been presented with a set of mapping solutions determined by using non-optimized "round robin" mapping model for storing VMIs, and a multi-objective re-distribution algorithm without replications support [8]. The cost objectives have been calculated based on the

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

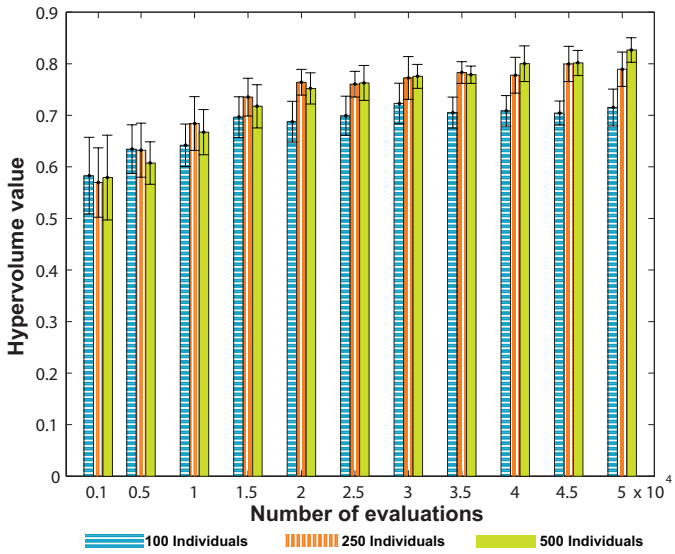


Figure 10. Quality of the optimal solutions in contrast with the number of individuals

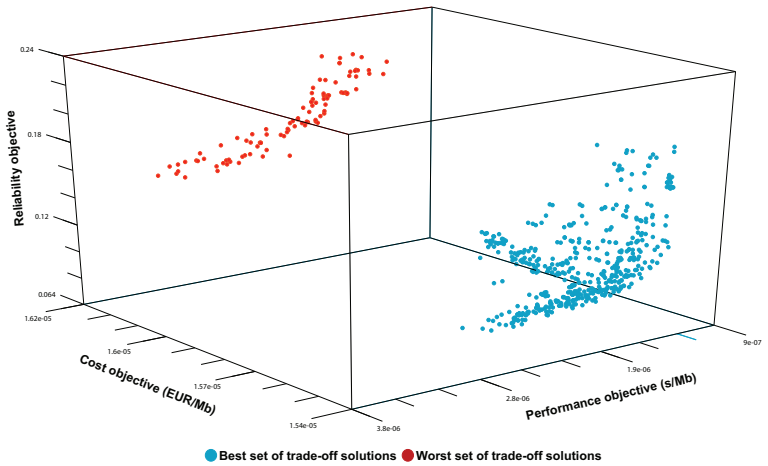


Figure 11. Visualized comparison of the best and worst solutions provided by the optimization framework

publicly provided price list for storing data in the Cloud by public companies, such as Amazon S3 and Microsoft Azure. The delivery time (performance) objective has been modelled based on the reported communication performance measures for 10Gbit and 1Gbit Ethernet [17]. For optimization reasons, the communication bandwidth values, were represented as delivery time needed for 1Mbit of data to be transferred from the source repository to the computational Cloud within the ENTICE environment. The system reliability values, were synthetically generated based on real values provided by multiple public Cloud providers, and do not include any other replication strategies.

The results indicate very high efficiency of the VMI re-distribution framework with replication support, as it can provide higher quality mapping solutions, especially in regards with the performance and reliability objective, yielding improvements of 51% and 74% respectively. On the other hand, the cost requirements can be increased by up to 46% if full replication is required. Using lower replication coefficient, such as $k = 0.5$, can diminish this issue and provide almost 50% lower delivery time, while incurring only 11% increase in financial costs.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

	Round-robin	**No-replication**	**Replication** $k = 0.5$	**Replication** $k = 1$
Cost objective (EUR/MB)	0.00001612	0.00001569	0.00001805	0.00002360
Difference-Cost (%)	/	-2.66638854	11.96255695	46.36499402
Delivery time objective (s/Mbit)	0.00000285	0.00000162	0.00000142	0.00000137
Difference-Delivery time (%)	/	43.13983662	50.00573665	51.94813878
Reliability objective (1-System R.)	0.00000285	0.00000162	0.00000159	0.00000137
Difference-Reliability (%)	/	28.61160910	42.08287635	74.05835564

Table III. Comparison between different VMI re-distribution strategies

9. CONCLUSIONS

In order to address the challenges that arise with the adoption of federated Cloud environments, in this paper a set of novel technologies and design approaches for efficient operation of distributed VMI repositories has been present. To accomplish this rather ambitious goal, extensive research was conducted on the current state-of-the-art and distinctive software modules have been developed to support the following operations: (i) simplified creation of light weight and highly optimised VMIs tuned for specific application requirements; (ii) VMI repository optimization based on multi-objective approach; and (iii) efficient reasoning mechanism for streamline support of complex VMI operations.

The introduced VMI management techniques have been implemented, integrated and evaluated as essentials elements of the ENTICE environment. Based on the evaluation results, it can be concluded that the proposed VMI synthesis and analysis technique can reduce the size of the VMIs by up to 55%, while trimming the image creation time by 66%. Furthermore, the repository optimization framework, can reduce the VMI delivery time by up to 51% and cut down the storage expenses by 3%. Moreover, by implementing optimized replication strategies, the framework can increase the system reliability by 74%, even in the cases when no other reliability strategies are implemented. All these processes, have been supported by efficient knowledge base and reasoning mechanism, which can reduce the communication time between the separate modules within the ENTICE environment to bare minimum.

Regarding the future work, we plan to extended the current environment to support fragmentation and re-assembly of VMIs based on similar functional blocks, thus allowing more optimal VMI storage and distribution.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 644179 (ENTICE project: dEcentralised repositories for traNsparent and efficienT vIrtual maChine opErations).

REFERENCES

1. Iosup Alexandru, Simon Ostermann, M. Nezih Yigitbasi, Radu Prodan, Thomas Fahringer, and Dick Epema, *"Performance analysis of cloud computing services for many-tasks scientific computing"*, IEEE Transactions on Parallel and Distributed systems 22, no. 6 (2011): 931-945.
2. Dragi Kimovski, Nishant Saurabh, Sandi Gec, Polona Štefanič, Gabor Kecskemeti, Vlado Stankovski, Radu Prodan and Thomas Fahringer, *"Towards an Environment for Efficient and Transparent Virtual Machine Operations:The ENTICE Approach"*, IEEE 4th International Conference on Cloud Networking (CloudNet), Pisa, Italy, 2015.
3. Goiri, I., Guitart, J., Torres, J., *"Characterizing cloud federation for enhancing providers' profit"*, 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD), pp. 123-130.

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe

4. A. Celesti, F. Tusa, M. Villari, and A. Puliafito, *"How to enhance cloud architectures to enable cross-federation"*, IEEE CLOUD 2010, pp. 337–345.
5. P. C. Brebner, *"Is your cloud elastic enough?: Performance modelling the elasticity of infrastructure as a service (iaas) cloud applications"*, Proceedings of the 3rd ACM/SPEC International Conference on Performance Engineering, ICPE 2012, ACM, pp. 263–266.
6. Wes Felter, Alexandre Ferreira, Ram Rajamony, Juan Rubio, *"An Updated Performance Comparison of Virtual Machines and Linux Containers"*, RC25482 (AUS1407-001) July 21, 2014, Computer Science
7. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T, *"A fast and elitist multiobjective genetic algorithm: NSGA-II" Evolutionary Computation"*, IEEE Transactions on, 6(2), 182-197.
8. Dragi Kimovski, Nishant Saurabh, Sandi Gec, Vlado Stankovski and Radu Prodan, *"Multi-objective Optimization Framework for VMI Distribution in Federated Cloud Repositories"*, Large Scale Distributed Virtual Environments, LSDVE 2016 in conjunction of Euro-Par 2016, Grenoble, France.
9. Diaz, G. von Laszewski, F. Wang, A. Younge, and G. Fox, *"Futuregrid Image repository: A Generic catalog and Storage System for Heterogenous Virtual Machine Images"* Third IEEE International Conference on Coud Computing Technology and Science (CloudCom2011), 2011.
10. J.V. Carrión, G. Moltó, C. De Alfonso, M. Caballer, and V. Hernández, *"A Generic Catalog and Repository Service for Virtual Machine Images"*, 2nd International ICST Conference on Cloud Computing (CloudComp 2010).
11. Li, Xiaodong, *"A non-dominated sorting particle swarm optimizer for multiobjective optimization"*, Genetic and Evolutionary Computation—GECCO, Springer Berlin/Heidelberg, 2003.
12. Kaveh Razavi, Liviu Mihai Razorea, and Thilo Kielmann, *"Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation"*, 1st Workshop on Dependability and Interoperability In Heterogeneous Clouds, Euro-Par 2013: Parallel Processing Workshops,2013.
13. Di Wu, Yupeng Zeng, Jian He, Yi Liang, and Yonggang Wen, *"On p2p mechanisms for vm image distribution in cloud data centers: Modeling, analysis and improvement"*,CloudCom 2012, pp. 50–57. IEEE Computer Society, 2012.
14. Zitzler, Eckart, Dimo Brockhoff, and Lothar Thiele, *"The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration"*, international Conference on Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg, 2007.
15. Sandi Gec, Dragi Kimovski, Radu Prodan, and Vlado Stankovski, *"Using constraint-based reasoning for Multi-Objective Optimisation of the ENTICE environment"*, IEEE The 12th Semantics, Knowledge and Grids on Big Data (SKG2016). Beijing, China, 2016.
16. Matthias Schmidt, Niels Fallenbeck, Matthew Smith, and Bernd Freisleben, *"Efficient distribution of virtual machines for cloud computing"*, in Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, PDP '10, pages 567–574, Washington, DC, USA, 2010. IEEE Computer Society.
17. Feng W. C., Balaji P., Baron C., Bhuyan L. N., Panda D. K., *"Performance characterization of a 10-Gigabit Ethernet TOE"*, High Performance Interconnects, 2005. Proceedings. 13th Symposium on (pp. 58-63). IEEE.
18. Kecskemeti, G., Marosi, A. Cs., Kertesz, A. *"The ENTICE approach to decompose monolithic services into microservices"*, IEEE High Performance Computing & Simulation (HPCS).
19. Jorge R., *Understanding Series and Parallel Systems Reliability*, Selected Topics in Assurance Related Technologies 11, Issue 5, 2014.
20. Image Factory - http://imgfac.org/
21. Packer - https://www.packer.io/
22. Jersey - RESTful Web Services in Java: https://jersey.java.net/
23. GeoNames - http://www.geonames.org/
24. VMWare - http://www.vmware.com/appliances
25. Science Cloud - http://scienceclouds.org/marketplace
26. Amazon Image Service - http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

Copyright © 2016 John Wiley & Sons, Ltd.
Prepared using cpeauth.cls

DOI: 10.1002/cpe