Semantic approach for multi-objective optimisation of the ENTICE distributed Virtual Machine and container images repository

Sandi Gec[1,2], Dragi Kimovski[3,4], Uroš Paščinski[1,2], Radu Prodan[3] and Vlado Stankovski[1*]

1. University of Ljubljana, Faculty of Civil and Geodetic Engineering, Ljubljana, Slovenia
2. University of Ljubljana, Faculty of Computer and Information Science, Ljubljana, Slovenia
3. University of Innsbruck, Distributed and Parallel Systems Group, Innsbruck, Austria
4. University of Information Science and Technology, Ohrid, Macedonia

Abstract

New software engineering technologies facilitate development of applications from reusable software components, such as Virtual Machine and container images (VMI/CIs). Key requirements for the storage of VMI/CIs in public or private repositories are their fast delivery and cloud deployment times. ENTICE is a federated storage facility for VMI/CIs that provides optimisation mechanisms through the use of fragmentation and replication of images and a Pareto Multi-Objective Optimisation (MO) solver. The operation of the MO solver is, however, time-consuming due to the size and complexity of the metadata,

*Vlado Stankovski, University of Ljubljana, Faculty of Civil and Geodetic Engineering, Jamova cesta 2, SI-1000 Ljubljana, Slovenia.

[2]E-mail: vlado.stankovski@fgg.uni-lj.si

specifying various non-functional requirements for the management of VMI/CIs, such as geolocation, operational cost and delivery time. In this work, we address this problem with a new semantic approach, which uses an ontology of the federated ENTICE repository, knowledge base and constraint-based reasoning mechanism. Open Source technologies such as Protégé, Jena Fuseki and Pellet were used to develop a solution. Two specific use cases: (1) repository optimisation with offline and (2) online redistribution of VMI/CIs, are presented in detail. In both use cases, data from the knowledge base is provided to the MO solver. It is shown that Pellet based reasoning can be used to reduce the input metadata size used in the optimisation process by taking into consideration the geographic location of the VMI/CIs and the provenance of the VMI fragments. It is shown that this process leads to reduction of the input metadata size for the MO solver by up to 60% and reduction of the total optimization time of the MO solver by up to 68%, while fully preserving the quality of the solution, which is significant.

Keywords: semantics, knowledge, reasoning, distributed repository, Virtual Machine or container images

1 Introduction

Today, a common software engineering practice is to reuse Virtual Machine and container images (VMI/CIs), when building component-based cloud applications [1]. VMI/CIs represent suitable packaging for micro-services [2] and fit nicely in the overall software engineering vision [3]. Software engineering dashboards, such as Juju [4] and Fabric8 [5], make it possible to quickly develop and provide new and elastic software services based on VMI/-CIs, which can then run across multiple public or private clouds. These new software engineering approaches and technologies aim at great improvements of the software life-cycle and receive wide attention and adoption by the industry. DevOps [6] is a culture, movement and practice that focuses on the process of software delivery that involves more open collaboration of professionals, including the sharing of such software artefacts. When combined with the Open Source approach, such as the one of OW2 [7], it is becoming possible to significantly reduce the software engineering costs and improve the quality of the delivered software services [3] through reuse of software components. A

particular example is the design and development of component-based cloud applications addressing fluid and solid mechanics problems. The application is made of fully elastic components called Cloudlets, which can be managed across multiple clouds [8] allowing engineers to share solution components and recombine them in applications.

Thus, VMI/CIs are becoming predominant way for the delivery of software. Due to their importance, many new approaches, technologies and services for their storage, delivery, provisioning, deployment, and otherwise management are emerging. When need for scaling the number of software components arises, it is necessary to deliver VMI/CIs from a repository to the selected cloud provider, and then deploy them for further use. This delivery process for VMI/CIs is currently still time consuming and prone to failures, for example, because some storage services have reduced online availability. This problem is currently addressed with the rise of literally thousands decentralized repositories for software components, e.g. CIs [9]. Storage services, such as any Amazon S3-compliant [10] storage or Docker Hub [11]) can be used to store VMI/CIs and to deliver them upon request to cloud storage providers. However, their operation is usually optimised for the VMI/CIs delivery only to specific cloud providers. In order to avoid the vendor lock-in problem much greater flexibility is needed.

An emerging new possibility is to optimise VMI/CI storage facilities through their federation, which is an important new trend aiming to address highly needed non-functional properties of VMI/CIs, such as delivery time, performance, storage location, storage cost, privacy, compliance, security, reliability, availability, maintainability, portability, dependability and similar aspects. Currently, it is very important to realize that the non-functional requirements for each software-component may greatly vary. Sometimes they may be correlated, while sometimes they may be conflicting. For example, a software engineer who wishes to distribute a VMI in a specific geographic region may wish to balance five different non-functional requirements: storage cost for the VMI, geographic availability, delivery, provisioning and cloud deployment time. Once the VMI has been stored in the repository, its geographic availability may also be dynamically changed and optimised based on external factors, such as the geolocation of the cloud application's end-users.

In order to address this range of problems, we are currently engaged in the development of ENTICE, which is a fully distributed storage facility designed for efficient operations with VMI/CIs, particularly their optimization for smaller size, faster distribution, higher availability, lower cost and

timely delivery to cloud computing infrastructures World-wide. Essential to ENTICE is the ability to collect and satisfy non-functional requirements specified by the software engineer on each uploaded VMI/CI. The ENTICE federated repository uses a Multi-Objective Optimisation (MO) algorithm to optimize the distribution of VMI/CIs across all repositories participating in the federation, which aims at satisfying every user's Quality of Service (QoS) needs. In next stages, this work will result in the design of Service-Level Agreements (SLAs) for storage and management of VMI/CIs.

The key aim of this study is to use semantics in order to improve the overall operation of the ENTICE environment by facilitating the management of the QoS requirements for individual users and the time consuming optimisation process of the distributed VMI/CIs. This work includes the development of an ontology, a knowledge base, reasoning mechanisms and their integration with the ENTICE environment. Key expected practical benefits are the reduction of the amount of input metadata used in the computationally-intensive MO process. Additional utility is provided through estimation of the actual free space in each VMI/CI repository, the ability to consider geographic location for each VMI/CIs and VMI fragments, provisioning of metadata for the management of SLAs, consistency checking of VMI fragment provenance data, including checking for corrupted and contradictory facts, and so on. Major focus of this work is the design of an integration framework between the ENTICE knowledge base and the MO service. Generally, the use of semantics in the ENTICE environment is intended to contribute to improved software engineering productivity by raising the level of abstraction in the overall software engineering process.

The paper proceeds as follows. In Section 2 this study is aligned with related works. Section 3 presents the main concepts of this study. Section 4 explains the complex relationships in the ENTICE environment and the semantic modelling approach, including the non-functional requirements for the storage and delivery of VMI/CIs, the repositories themselves, their SLAs in relation to the software developers, and other important aspects. The final result is an ontology of the ENTICE environment expressed in W3C [12] compliant syntax. Section 5 elaborates the design, implementation and integration of the associated ENTICE knowledge base. The MO approach is presented in Section 6. Section 7 presents the implementation of a reasoning mechanism for MO of the repository. Section 8 presents experiments showing benefits of using the ENTICE knowledge base to facilitate improved operation of the environment. Section 9 summarises the main results, their impact

and future work.

2 Related Works

Delivery of VMIs from a VMI repository to the actual computing infrastructure strongly affects the start-up time of VMs. This process is considered to be slow and negatively impacting the dynamic scaling of cloud applications [13]. This problem has been generally addressed with the development of VMI/CI distribution networks aiming to speed up VMI/CI speed up and delivery, which implement various degrees of sophistication. Following is an overview of various approaches.

Peng et al. [14] presented a VMI distribution framework enabling collaborative sharing in cloud data centres. Zeng et al. [15] introduced a solution for VMI backup and recovery comprising similarity retrieval, one-level file-index and adjacent storage. Schmidt et al. [16] discussed the challenge of distributing VMI files to a set of distributed computing nodes in a multi cloud computing environment. They present a hybrid solution that adaptively selects the optimal transfer method depending on network capabilities and cloud site constraints. Razavi and Kielmann [17] proposed an elastic VM deployment mechanism using VMI caches to overcome the VM start-up bottlenecks. Kimovski et al. [18] described mechanisms how to efficient manage virtual machines in federated cloud repositories.

Due to the highly dynamic demand for new VMI files, traditional distribution as de-duplication mechanisms, such as Whole File Detection (WFD) [19], Content Defined Chunking (CDC) [20], Fixed Sized Partitioning (FSP) [21], Variable Sized Partitioning (VSP) [22] and Sliding-block method [23]—along with traditional transmission mechanisms, such as File Transfer Protocol (FTP) and Secure Copy Protocol (SCP), are not efficient enough without an optimised knowledge management at the level of VMIs [24]. The knowledge about cloud resources and virtualization environments such as similarities among VMIs, image clusters and data centre topologies generally supports design of de-duplication and other management mechanisms for VMI files [25]. Jayaram et al. [25] concluded that empirical analysis of VMI files can be leveraged to design smart image distribution schemes, take information about operational environment into account and help VM provisioning, cloning and migration.

Sack et al. [26] proposed a semantic approach for addressing NP-complete

decision problem over a bibliographic domain. Our work differs from the cited literature in a major way that the presented automatic distribution and decomposition of VMIs uses a knowledge intensive approach based on W3C interoperability standards, such as OWL2 [12] and aims to generally improve software engineering practices. Even though there are methodologies describing the knowledge base ontology with various graph-based models [27] and interactive semantic approaches [28], the ENTICE system knowledge base consist of a single graph-based ontology. The use of semantics (ontology, knowledge base and reasoning mechanisms including heuristic reasoning rules [29]) may contribute significantly to the operation of the ENTICE distributed repository for VMI/CIs by providing important logistics, starting from the management of SLA agreements between the software engineers and the ENTICE environment, then by providing input data and information for important functionalities of the environment, such as MO, and other logistics for VMI/CIs migration between the repositories, portability, availability, reliability, costs, and other non-functional properties.

The present work focuses on the area of semantic modelling and knowledge engineering for multi cloud environments. Semantic approaches have already been used to address problems in distributed, grid and cloud computing environments in a variety of studies. Some past projects covering various semantic aspects have been: myExperiment [30], InteliGrid [31], OntoGrid [32], mOSAIC [33] and others. Currently ongoing related projects are Smart Cloud Engine [34] with which the projects SWITCH [35] and ENTICE [36] can compare. Among those, ENTICE is the only project focusing on the development of a distributed repository, the projects Smart Cloud Engine and SWITCH are concerned with the runtime of cloud applictions. These projects have concentrated on the delivery of high quality, fully functional and elastic cloud applications. An example is the design and development of a component-based cloud application for solving fluid and solid mechanics problems. It is composed of fully elastic components called Cloudlets, which can be managed across multiple clouds [8]. This application has compute, memory and communication intensive software components, which can greatly influence the user experience. It is therefore important to provide for elasticity to applications like this one, for example, an ability to move, deploy and scale the number of running VMs or containers.

This research domain poses some very specific challenges, such as the needs to support both strategic decisions, when the software engineer negotiates a SLA for her/his VMI with the ENTICE repository, and dynamic

decisions, when automated elasticity has to be obtained by the cloud application at runtime. Our goals are therefore, not only to semantically model the ENTICE distributed repository environment, but also to provide mechanisms in the knowledge base that will support both strategic and dynamic reasoning and information support. In a wider context, this work is posed to contribute to more optimised and modern software engineering approaches that include more machine processable knowledge and information in the software engineering life-cycle.

3 Using semantics in the ENTICE environment

Today, it is a common software engineering practice to pack software components (services, scripts) into VMIs or CIs and uploaded them to storage facilities for further use by elastic cloud applications. During this process, the developer has to deal with complex requirements, considering among other, interoperability and performance issues. It is therefore very important that non-functional requirements, including QoS requirements for the management of VMI/CIs are taken into account when optimising the operation of the federated storage facility. In this context, semantics could be used in the decision making process at both strategic (by the software engineer) and dynamic (by the various software services) levels, when operating a distributed repository of VMI/CIs, such as the ENTICE environment. Following is a short overview of the ENTICE environment, its use cases and the semantic approach proposed in this work.

3.1 The ENTICE federated repository

The ENTICE federated repository is developed to allow for efficient operations, including moving, replication and delivery of VMI/CIs, as illustrated in Figure 1. The developer has the possibility to access VMI/CIs via a graphical User Interface (UI) and from there deploy them on a cloud provider. The overall software engineering process is informed by using metadata which is systematically gathered, exchanged and otherwise managed in a knowledge base. The key components of the ENTICE environment that encapsulate all the use cases are presented as follows:

1. **Development tools** (such as Fabric8 [5]). These are basic instruments used by a developer of a cloud application. Such tools do not yet pro-

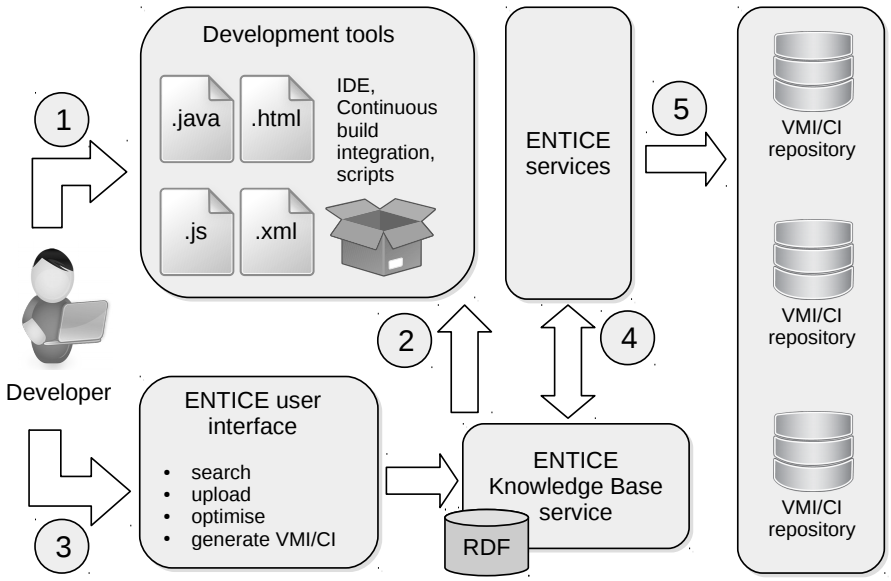

Figure 1: The ENTICE environment uses knowledge for optimal operations with Virtual Machine and container images.

vide enough information to the developer concerning the deployment stage (e.g. the best deployment location). Programs, APIs and other services have to be deployed using a server that runs on a specific operating system. Creating VMI/CI, even for only testing purposes, that contain specific services takes time and the developer must have some knowledge about library dependent software of the service, the scalability of the software and similar. By using the ENTICE environment a developer can access the images through a RESTful service or through a specifically designed UI facilitating repositories-wide software search and discovery.

2. **ENTICE knowledge base** service comprises of a RDF store and mechanisms, such as validations, reasoners, rules and similar, serving information to all other components of the architecture. The service interchanges the data through the ENTICE external interfaces, the developer API and the ENTICE dedicated services. All the data is

organised according to a domain ontology. Based on the ontology, sets of reasoning mechanisms are implemented.

3. The **ENTICE User Interface (UI)** provides to the developer several features such as optimising and searching VMIs/CIs, uploading new images to the repositories that include pre-installed user specific services and applications, or even system based settings (e.g. network configuration, SSH and system credentials). Besides the image upload, the user has the possibility to generate images by using a script (e.g. Chef [37] or Puppet recipe [38]) that includes functional requirements. In the UI search screen the developer is able to search among all available public VMIs/CIs, which are stored in public VMI/CI repositories, such as those of Amazon S3 [10] or Docker Hub [11].

4. The **services of the ENTICE environment** provide functionalities that are exposed through the UI. These services are elastic and geographically distributed in order to optimise the operation based on information and knowledge on network performance (e.g. latency and current bandwidth) through monitoring services. For each VMI/CI containing user developed software components (e.g. services, databases, APIs and other applications) a special ENTICE service can be used to reduce the image size (e.g. remove unused libraries and documentation) in order to significantly reduce the VMI/CI size while not affecting the mandatory running applications functionalities. By performing an operation like this, the user may save significantly on the storage cost as soon as the software asset is deployed in the VMI/CI repositories and experience faster auto-scaling operations of a running VM.

5. **VMI/CI repositories**. Currently, ENTICE includes private repositories and can be extended with public repositories that are available on the Internet. For example, more than 200,000 software packages on different repositories can be accessed via JFrog Bintray [39].

   Thus, the idea to integrate those repositories in the ENTICE environment may provide important benefits. All the information of VMIs/CIs available in those repositories can be stored in the knowledge base with addition of new constraints, dependencies and other derived data from reasoning or user experience. For example, the majority of CIs are specifically running only one service (e.g. Apache Tomcat) and user

specific applications (e.g. a Web application deployed on Tomcat). The ENTICE environment can reduce the size of a CI and facilitate the selection of technologies (e.g. migrate the database from MySQL to NoSQL) by integrating the appropriate libraries into the newly created CI. Same can also be applied to VMIs.

3.2 Using the ENTICE environment

A VMI/CI distributed upload is more specific use case which the best describes the intent of this paper. Figure 2 shows the steps involved from a viewpoint of the ENTICE user (e.g. VMI/CI developer).

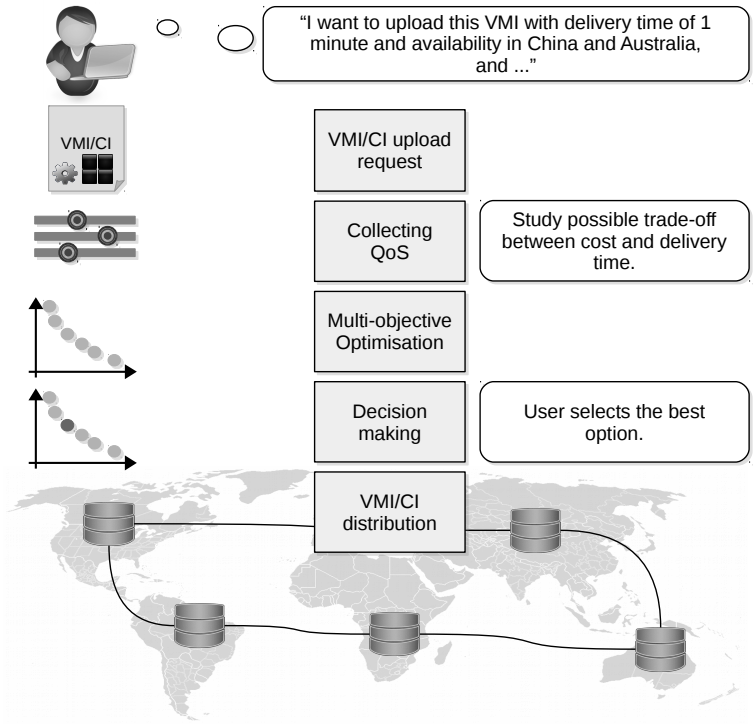

Figure 2: VMI/CI distributed upload by specification of non-functional properties.

First, the image upload request is made to the ENTICE environment.

Then the user is able to specify QoS requirements about the image such as geographic location of the service, storage and/or network latency, image availability, image delivery time to the cloud, and cost of storing and downloading image. Some of these requirements might be specified as constraints to the system (e.g. due to legislations upon countries of storing data, or due to storage quota/capacity limitations) while others as optimisation objectives (e.g. find the storage with the best price). A multi-objective optimisation algorithm then considers the constraints and optimisation goals and tries to find the best storage for the request. An outcome of the algorithm is a set of feasible storages (with respect to the constraints) that are at the same time the best fit (with respect to the optimisation objectives); thus, the outcome facilitates user in decision making. Because the user is able to specify several optimisation objectives at the same time, the best solution does not result in a single storage but instead in a set of best solutions, known as Pareto front. The final decision upon the image placement is then left to the user to allow for fine-tuning her optimisation objectives before the image is stored into the best storage.

3.3 Semantic approach

The role of the knowledge base in the above use case ranges from collecting and storing the QoS for every image, to providing information about the storages, and supporting the decision making process. The relation between the knowledge base and the multi-objective optimisation service is of particular interest in this study. We experimented with two different procedures, which mainly differ in the information exchange between the knowledge base and the multi-objective optimisation service. They are described in the following.

In the first procedure the knowledge base supplies the multi-objective optimisation service with the list of storages and their properties, as well as the list of constraints for the image upload request. The optimisation algorithm then performs the Pareto front computation by considering the whole set of storages and takes care of the constraints after the Pareto front is computed. This serves as a reference model that we want to improve with the second procedure.

The second procedure is therefore designed to efficiently reduce the initial set of storages provided to the Pareto front computation. This way less data has to be exchanged between the knowledge base and the multi-objective optimisation service, and the optimisation is performed on a feasible set of

storages only, with respect to the user's constraints. Therefore, the multi-objective optimisation algorithm in this case does not need the list of constraints.

The second procedure has performance and cost advantages over the first one. Applying constraints to the set of storages directly in the knowledge base consumes less network bandwidth which can in some cloud settings result in lower cost. It can also improve transfer times, if application of rules is not too computationally intensive for the knowledge base. Next, if the whole set of storages is very large and the applied constraints eliminate most of the storages, the computation of the Pareto front might become difficult, because the multi-objective optimisation algorithm works on random subsets of its input set, as is described in Section 6. Even if that is unlikely to occur with carefully designed genetic algorithm, if is more likely to fit a small subset of storages into a memory than the larger set, which should also improve the time required for the Pareto computation. Nevertheless, if the multi-objective optimisation procedure needs to combine storages in order to meet the constraints (e.g. replicate image over a subset of storages to meet high request for image availability), the number of combinations grows exponentially with the number of storages. Some of these issues are summarised in Figure 3.

The detailed analysis of the various use cases leads to a collection of functional and non-functional requirements for the development of the ENTICE ontology and the associated knowledge base.

4 Requirements analysis and development of the ENTICE ontology

In order to develop an ENTICE ontology and associated knowledge base it was first necessary to collect and analyse key requirements. Here, we first identify the functional requirements, which are needed to integrate the semantic technology with other services of the ENTICE environment. Following this, non-functional requirements are elaborated. These are necessary to preserve elevated overall system Quality of Service (QoS) and Quality of Experience (QoE). The QoE measures the actual user's experience with the ENTICE environment. The second part of the section explains the development of the ENTICE ontology.

4.1 Functional requirements

The new system can be accessed only by registered users. Therefore, the knowledge base must support different authorization and authentication levels (e.g. only administrators can access to special panels and further interact with operations in progress, check current status of services and compare them to the knowledge base content).

Various search mechanisms should be provided, from the simple basic search queries to obtain data stored in a single entity (e.g. search VMI/CI by different criteria, check a repository resource status, etc.) to more complex search mechanisms in order to satisfy the capabilities for:

- search between individuals of the same type and their chronological differences, usually derived from the same ancestor, due to updates (e.g. updating VMI operating system, adding new applications or functionalities, etc.). In those cases the mechanism should be able to find redundant and outdated individuals in order to remove them;
- rule based constraint verification (e.g. constraint check of compatibility for the new added software in CI/VMI).

From the ENTICE system heterogeneity the knowledge base service must also support connectivity through external interfaces. Thus, the knowledge base has to support common used connectivity protocols like REST. In some cases the knowledge base service has to access sensitive data and certification mechanism must be supported (e.g. Secure Shell (SSH)) to fulfil the security requirement aspect.

4.2 Non-functional requirements

The knowledge base needs to address a variety of non-functional requirements that can be identified in the majority of ENTICE environment components. The most important of them are:

- achieving high performance of the services inter-connected via the knowledge base (e.g. fast query responses);
- ability to distribute the knowledge base service or even the knowledge base environment itself to avoid single point of failure;

- scalable knowledge base architecture, its ontology (e.g. by adding new cloud providers the system must be able to store new pricing metrics) and possibility to increase resources for the growing knowledge base storage;
- availability where monitoring of RDF store service has to be running and minimize the down time of the knowledge base service or notify the administrators for major system faults;
- security that overlaps with functional requirements;
- data integrity by using different validation supporting reasoners (e.g. HermiT, Pellet etc.) and adequate action in case of data integrity violation (e.g. data types or relationships don't match the ontology scheme definition).

In the following, we elaborate the design and development of the ENTICE ontology.

4.3 Ontology development

The development of the ENTICE ontology was done iteratively, by identifying the functionalities and their matching entity classes among all subsystems, their dependencies in a form of relationships and the constraints of entity attributes to satisfy the data exchange between ENTICE services. The aim of the ontology design process is to create a robust ontology schema with possibility to be scaled in order to support new functionalities by minimally affecting the existing ones (e.g. by only adding new relationships and entity attributes). Particular care was put on developing reasoning capabilities with Pellet and possibilities to use RDFS [40] and SWRL [41] rule engines. By using reasoners, it is possible to facilitate environment information flow, minimize human errors while inserting new RDF data, check existing ontology data inconsistency and even simplify data transfer between the ENTICE services.

The entire ENTICE ontology with interconnected entities is presented in Figure 4. Since the knowledge base is the main database of the ENTICE environment all essential use case data must be covered. The most important ontology entities are described in Table 1 and are used as main knowledge assets in the experiments. Thus, besides the ontology scalability aspect other important criteria are also satisfied, including:

- data must be efficiently accessed through queries and reasoning mechanisms by following Ontology Design Patterns [42];
- new expressivenesses must be achieved by inferencing the ontology content. Therefore, new relationships should be generated to facilitate the querying through matching the ontology;
- minimum redundancy level should be reached during the ontology development; and
- classes must be designed in a way to support the future scalability in a way of generalize concepts (e.g. ProvenanceData class should not be used only for Fragments).

Table 1: Important concepts modelled by the ENTICE ontology.

ENTITY NAME	SHORT DESCRIPTION	USE CASES
DiskImage, VMI, CI	VMIs and CIs.	Explore, upload, deploy, optimize, recipe build
Repository	characteristics of available cloud repositories	Explore, upload, deploy, optimize, recipe build
Fragment	preliminary description of fragments	MO
User	information about users including user type	Login, VMI ownership
Geolocation	geographical information about the repositories	Explore
Pareto	calculated pareto possibilities provided by MO.	MO
DiskImageSLA	information about resulting (and new) SLA	Upload, MO
Quality	all information about recipe builds use case	Build VMI/CI from recipes
ProvenanceData	all phases of fragment distribution	MO
Delivery	information about deployment of VMIs/CIs	Deploy VMI/CI

The developed ontology stores important concepts about the entire ENTICE environment, such as:

- concepts of software resources (e.g. VMI/CIs, cloud-based environmental settings);
- programming concepts (e.g. storage complexity, taxonomy of functional properties etc.);
- virtual organization concepts (e.g. privileges, credentials, ownership);
- resource negotiation-related concepts (Pareto SLAs);
- QoS concepts and
- runtime environment concepts (e.g. monitoring).

Based on the ontology design it may be understood that the ENTICE knowledge base focuses on the cloud domain with specific use cases covering broad aspects, such as VMI/CI distribution, fragmentation, SLAs management and so on.

4.4 Role of the ENTICE knowledge base

General complexity of the nowadays cloud based systems is increasing due to integration of new functionalities, arising from new users which leads to the needs for continous maintenance of datasets. There are common approaches to improve or at least maintain reasonable data flow performances such as migration to new technologies (e.g. from MySQL to NoSQL storage systems), hardware upgrades and software optimisation approaches. The later in terms of semantic design and optimized access of ENTICE metadata is the main role of the ENTICE knowledge base. The semantic approach is used not only to describe the data in a interoperable graph-based representative ontology, but also to enrich the relationships among the entities by using reasoning mechanisms.

Through the process of system implementation and integration, new entity data can be seamlessly created including attributes with constraints, relationships and rules which have to be fulfilled in order to inference new knowledge. Moreover, the complexity of specific SPARQL queries can be significantly reduced that leads to reduced query results and faster metadata management for the functioning of the overall ENTICE environment. The concrete result, which is evaluated further in this study is the simplification of SPARQL queries by using pre-inference knowledge created with rule based reasoning mechanisms.

5 Designing and implementing the knowledge base

In order to be able to use the ENTICE ontology it is necessary to develop specific mechanisms for managing complex-structured data. In the course of this study, experiments were performed with various degrees of expressiveness to model the relationships among the entities, the use of advanced constraint mechanisms, RDF validation [43], complex data querying, inferencing new knowledge using reasoning mechanisms and other approaches. The primary

use of the knowledge base is to provide RDF metadata to all subsystems and services of the ENTICE environment through a main API. The API is developed in a way that supports various queries and reasoning mechanisms and other aspects, such as security. The knowledge base is also designed in a way that its software components can further scale, e.g. via the use of new container instances.

In order to fulfil some basic interoperability requirements the knowledge base service must support the exchange of RDF data using simple queries, different reasoners, and rule based constraint verification to minimize human errors that may occur through the graphical UI when executing write based requests into the knowledge base or scripts (e.g. functional descriptions of VMIs/CIs). Those mechanisms are indirectly described in the following section on how the knowledge base provides metadata to other ENTICE services, particularly, the Pareto SLA and Multi-objective optimisation part with substantiate calculation time improvements through experimental results.

5.1 Design

The ENTICE environment can be seen as repository-based system that encapsulates a variety of subsystems. Basically, it provides a universal backbone for Infrastructure as a service (IaaS) VMI/CI, which supports different use cases with dynamic resource (e.g. running resources for few seconds or continuously for years) and other QoS requirements. The ENTICE technology is strongly decoupled from the application and their specifics such as runtime environments, but continuously supports them through optimised VMI/CI image creation, assembly, migration and storage. The ENTICE environment inputs are unmodified and functionally complete VMIs or CIs from users. Unlike the other cloud provider environments in the market, our environment transparently tailor and optimise them for specific Cloud infrastructures with respect to their size, configuration, geographical distribution, such that they are loaded, delivered (across Cloud boundaries), and executed faster, with improved QoS and decreased final cost for the end users. The proposed high-level architecture is shown in Figure 5 and comprised of the following key subsystems:

- knowledge base service, which is a central part of our work, presented in our study and supports all the other services of the ENTICE environment with information needed for strategic and dynamic decisions

making,

- VMI/CI portal, which is the ultimate graphical UI used by the developer to search for software artefacts across the distributed repositories,
- VMI/CI Synthesis, which facilitates the synthesis of VMI/CIs based on recipes,
- VMI/CI Analysis, which facilitates the optimisation of the size of VMI/-CIs,
- VMI/CI Distribution, which facilitates VMI/CI movements and other operations among the potentially unlimited set of geographically distributed repositories, thus, optimising VMI/CI delivery time at particular geographic locations and storage costs,
- Multi-objective Optimisation (MO) Framework, which addresses the needs for optimised operation of the overall environment through its MO methods implemented via JMetal and can also support VMI/CI Distribution and Online VMI/CI Assembly,
- Pareto Service Level Agreements (Pareto SLA), which is a method used by the users to negotiate terms of contract with the distributed repositories of VMI/CI,
- Online VMI/CI Assembly, which is in charge of assembling the VMIs/-CIs from fragments and
- VMI/CI Management Template that represents available Cloud management systems (e.g. OpenNebula, OpenStack etc.) which are deployed on different geographical locations (Slovenia, Hungary, Austria and UK) and the ENTICE environment can access them through their APIs. In the future new Cloud management systems can be added.

As it can be seen in Figure 5, knowledge management and information supply play a crucial role in the operation of the overall ENTICE environment. Due to space limitations, in the following we focus on the use of the newly developed knowledge base and its reasoning methods in relation to some NP-hard optimisation problems, which are addressed by a combination of MO and the Pareto SLA technique. NP-hard optimisation problem, for

example, is the optimal distribution of VMIs across the distributed repositories which allows to agree (e.g. via SLA) specific maximum allowed delivery time at a specific maximum allowed cost.

5.2 Implementation

The ontology of ENTICE project was developed by using the ontology editor Protégé. The main language for the ENTICE ontology is implemented in the OWL2 [44] using Turtle format due its human readability. The knowledge base service was developed using Java based technologies and frameworks such as Java Jersey for RESTful web service, Apache Maven for software management and Apache Jena Fuseki for serving RDFs. The last was chosen because it supports various powerful reasoners (e.g. Pellet, TrOWL, ELK etc.) [45], its default integrated reasoner and have satisfactory performance [46]. During the implementation phase of the project new mechanisms will be integrated to facilitate the knowledge base service deployment, testing and security (e.g. Apache Shiro).

The integration process of the ENTICE services followed a well defined path: (i) identification of services/APIs as presented in previous chapter, (ii) definition of communication protocols between systems, (iii) detailed description of requests and communication flow and (iV) actual integration and detailed implementation.

The main supported communication protocol between ENTICE services, in particular between the knowledge base service and other services, is HTTP REST protocol with JSON based exchange data format. For services involving VMI/CI management cases (e.g. VMI/CI deployment and redistribution), the knowledge base service is adapted to support other communication protocols such as Web Services Description Language – WSDL.

Detailed work-flow for the supported use cases was identified through a detailed UML diagram definition which followed the requests definition supporting the data scheme of the ENTICE ontology. To reduce the risks concerning errors during the development, a testing system based on Unit tests was introduced in a continuous build integration using Jenkins [47], for example at each code update (e.g. git push), updated ENTICE services were build and Unit tests were executed.

The final stage of the implementation and integration mainly involves implementation of the graphical UI and identification of possible boundary conditions that can affect the system.

6 Multi-objective Optimisation Framework

This section elaborates the development and integration of the ENTICE multi-objective optimization framework for optimized VMI distribution [48]. The optimization framework can be applied on multiple distinctive application levels within the ENTICE environment. For the implementation of each application level, within the optimization framework, diverse heuristic tracks have been pursued. Above all, a consolidated service based application program interface has been provided for easy integration of the framework within heterogeneous cloud environments.

6.1 Background

Optimization is a process of denoting one or multiple solutions that relate to the extreme values of multiple specific objective functions within given constraints. When the optimization task encompasses a single objective function it typically results in a single solution, called an optimal solution. Furthermore, the optimization could also consider several conflicting objectives simultaneously. In such circumstances, the process will result in a set of alternative trade-off solutions, so-called Pareto solutions, or simply non-dominated solutions. The task of finding the optimal set of non-dominated solutions is known as multi-objective optimization. In what follows, an outline of the basic concepts considering multi-objective optimisation theory is provided.

For a point $o \in O$ *is said to dominate* $o' \in O$ if o' is better than o in respect to all objectives and o' is worse for at least one of them. A point $o' \in O$ is considered to be *non-dominated* if there is no other point $o \in O$, which dominates o'. A particular solution $x \in X$ is called *Pareto optimal* if its "position" in the objectives space is non-dominated by any other point. The set of all Pareto optimal solutions is called *Pareto optimal set*.

The set of all optimal solutions in the objective space is called *Pareto frontier*. The Pareto front can be considered as a efficient tool for aiding the decision making process. The shape of the front can provide insights, which allow to efficiently explore the space of non-dominated solutions with certain properties, and reveal regions of particular interest which cannot be seen in advance, before the optimizations process has started. Therefore, the users do not need to set their preferences before finding a set of optimal solution.

Furthermore, when considering multiple different Pareto solutions, a spe-

cific method is required to compare the quality of the solutions set. A Pareto set is considered to be of good quality if it provides accuracy and diversity. One way of determining the quality of a set of Pareto solutions is the *hypervolume*. Given a set of trade-off solutions X, the hypervolume $HV(X)$ calculates the area encircled between the points in X and a given reference point W. This way, the better the points contained in X and the most diverse they are, the $HV(X)$ will be higher.

6.2 Designing Multi-Objective Optimisation (MO) framework for ENTICE

VM images 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 VM image storage which is tedious, error-prone and time-consuming process especially if working with multiple cloud providers. Current state-of-the-art does not provide any substantial means for streamlined adaptation of distributed repositories and efficient utilization of the storage resources. The vast majority of existing work in this field has been focused towards optimization of the utilization of the computational resources. Regrettably, limited research has been conducted on the management of the VM images, as essential storage resources in federated environments. Inadequate management of those crucial resources can easily lead to inefficient utilization and overall degradation of the computational performance of the whole system. In this context, the optimisation of the VMI's distribution across federated repositories is required both by the applications and by the underlying cloud providers for improved resource usage, operational costs, elasticity, storage use, and other desired QoS-related features. Multiple optimization requirements have been identified and suitable strategies have been proposed for overcoming of those barriers.

Based on these considerations, the following optimization modules have been developed: (i) initial VMI distribution, (ii) Offline VMI redistribution and (iii) online VMI redistribution. In this paper the focus will be on (ii) and (iii) due to the possibility of knowledge base collaboration in the problem solving.

ENTICE is developed to support VM image redistribution both offline, as well as online during application execution by pro-actively moving the

most demanded VM image fragments close to the resources the application is currently running on [49]. In case of online image delivery, ENTICE will automatically discover user demand patterns by analysing the meta-data (e.g. sequence and number of downloads of particular images or fragments) published by the provider-operated repositories (e.g. similar to Glance from OpenStack) and replicate the highly demanded images or fragments according to user demands. For example, if some VM images are always instantiated at a high frequency, they will be placed at other providers where the users might need them. Furthermore, based on the performance requirements, use patterns, and structure of images or location of input data, ENTICE (assisted by its knowledge base) will automatically optimise in the background the distribution and placement of VM images to significantly lower their provisioning time for complex resource requests (which can be in the orders of hours using today's provider lock-in technologies) and for executing the user applications (thus focusing on their functional use scenarios). The optimisation will consider the requirements of applications built as a composition of VMs and arrange for simultaneous delivery of multiple VM images to selected clouds, optionally enhanced with application input data. Moreover, the online discovery and assembly of VM image fragments employs the same multi-objective optimisation framework by assembling a VM image in a running VM at the provider with the best performance, lowest instantiation overheads (fragment transfers and VM deployment), and execution cost trade-off. Additionally, the ENTICE multi-objective optimization framework has been envisioned to provide a specific module for efficient initial distribution of the VM images across the distributed storage repositories. The advantage of implementing such a module is twofold, as it can provide means for balanced distribution of the VMIs and it can reduce the complexity for selecting an initial storage repository by the VMI's owner.

6.2.1 Offline VMI redistribution

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 analysing the repositories usage patterns, and results in optimized distribution of the VMIs and the associated data-sets across the federated

environment.

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 data storage and transfers should be provisioned by all cloud providers within the federation.

The performance model includes 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 framework's 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.

The core of the offline VMI redistribution sub-module is constructed over the NSGA-II multi-objective optimization algorithm. 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 neighbouring sub-graph 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.

6.2.2 Online VMI redistribution

One very important aspect that should be considered in federated cloud environment and repositories is the optimization of specific user's 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 user's request, and only on its own images. When the user asks 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 user's 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 interval of time.

7 Reasoning for the Multi-Objective Optimisation

Semantic representation of the data by using ontologies and associated knowledge bases can be exploited to identify new knowledge by using different reasoning mechanisms. In comparison with traditional relational databases

that allow only simple logical connections between tables, knowledge bases have the ability to describe connections between entities in a more descriptive way by using Description Logics (DL). Thus, different facts that are not explicitly expressed in an ontology, can be derived such as satisfiability of a concept, subsumption of concepts, consistency of ABox with respect to TBox, checking if individual is an instance of a concept, retrieval of individuals and realisation of an individual. On the level of the entire ontology, reasoning is used to reduce redundancy of information and to find conflicts in knowledge content. The ENTICE environment supports Jena default reasoners and others that are supported in Jena Fuseki (e.g. BUNDLE, Pellet, TrOWL and DLEJena).

Constraint-based reasoning, as a concept, has connections to a wide variety of fields, including formal logic, graph theory, relational databases, combinatorial algorithms, operations research, neural networks, truth maintenance, and logic programming. RDF based stores can be deducted as a combination between relational databases that includes formal and graph theory logic. For ontologies such as ENTICE, satisfying the syntactic constraints, the most suitable candidates are rule-based reasoner, tableaux reasoner and other query engines such as ABox. One of the most used for Web Ontology Language OWL1 is the Jena's default reasoner, however, for processing OWL2 the Pellet reasoner can offer more powerful reasoning capabilities.

The ENTICE knowledge base and its reasoning mechanisms are used in two steps of the MO process: (i) for the online repository redistribution and (ii) for the offline redistribution.

7.1 Using the ENTICE knowledge base

In the first step of online distribution the knowledge base is used as an asset to speed up the execution time of MO algorithm. Due to the need of fast execution time to assure reasonable QoE, the ENTICE environment tends to use different subsystems and their approaches, in this case knowledge base RDF based data retrieval. To be more concrete, the knowledge base provides only those data that are reasonable to be used as the input for the MO algorithm and it constructs this data by taking into account knowledge base defined constraints that are sorted by their relevance as shown in the SPARQL query 1. There are also other constraints considered in this stage, that are presented in the following section. The online redistribution flow is depicted in Figure 6.

Listing 1: SPARQL query for online redistribution. In the inner SELECT statement the repository subjects are sorted according to their relevance, and in the outer SELECT statement all the attributes of the repository class are extracted.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX kb: <http://project-entice/2015/knowledgebase#>
SELECT ?s ?p ?o
WHERE {
   ?s ?p ?o {
      SELECT ?s ?p ?o ?storage ?operational
      WHERE {
        ?s a kb:Repository .
        ?s kb:Repository_OperationalCost ?operational .
        ?s kb:Repository_StorageLevelCost ?storage
      }
      ORDER BY ASC (xsd:double (?operational * 0.9 + ?storage))
      LIMIT 50
   }
}
```

For the offline redistribution the data needed for MO is more because derives from different entities that stores information about VMI or CI and the entire life-cycle of redistributions containing delivery and deployment time. This two times are used to represent practical SLA that can differ from a theoretical one. The approach that provides relevant input data for MO is the following:

1. With a SELECT query 20% of the cheapest repositories with a minor delivery and deployment times are removed in the further steps.

2. With a SWRL based rule by using Pellet reasoner a property inference is applied on the same provenance data on the same repository to be stored on each of those individuals. The rules are mainly applied on entity property level by using Assertional Box (Abox) to represent OWL facts. For example, by using *owl:propertyDisjointWith* data is additional marked as not relevant for additional querying.

3. The relevance of the VMIs/CIs that should be redistributed, is sorted through a SELECT query where the selection criteria is based on higher deployment time, delivery time and SLA of the cloud. For these experiments additional QoS metrics (e.g. services can be deployed only in one continent) were not applied.

The use of the knowledge base in connection to the MO algorithm is analysed in the following section.

7.2 Integration with the MO framework

The multi-objective optimisation framework is reliant upon the user's usage patterns to properly optimise the distribution of the VMIs and associated dataset across the federation. To this aim, the knowledge base is an essential tool that provides crucial information for proper modelling of the usage patterns, thus prompting efficient operation of the optimisation framework. Furthermore, to be able to accurately evaluate the objective functions, the framework requires information on the previous data transfers within the distributed repository. In addition, various other parameters, such as cost for storing, interconnections bandwidth and latency, are necessary.

There are two pivotal points of interest in the integration of the knowledge base within the domain of the optimisation framework: (i) provisioning of decision variables and (ii) performing the decision making policy on the obtained Pareto trade-off solutions.

For the purpose of the presented research work, the Multi-objective optimisation framework is viewed in the terms of its inputs and outputs, without providing any deeper knowledge of its internal workings. The interaction between the knowledge base and the MO framework is performed through RESTful service based API. As previously described, the Multi-objective optimisation framework can be applied on multiple distinctive levels. Nevertheless, on each level, the provisioning of the input variables is performed in a similar manner. When the optimisation process is initiated, the framework sends a query for the required data to the knowledge base. The knowledge base has been constructed in such a way, to provide only the relevant input data that will be described in the following subsection and substantiated through the presented evaluation.

The flow of interactions between the knowledge base and the optimisation algorithm, in the cases of online redistribution is depicted in Figure 7.

The entities involved in the process are: (i) the client who initiates the upload process, (ii) the knowledge base service which manages the data, (iii) the MO service which computes the Pareto front, and (iv) Image redistribution system that actually performs the requested action. The process starts when the client requests an upload of the VMI/CI to the ENTICE federated repository, by specifying the non-functional properties of the image, such as geographic location (e.g. preferred location, political legislations, etc.). Then the knowledge base serves to the MO service a subset of repository metadata information, depending upon the constraints selected by the client in the request. The MO then computes the Pareto front and returns the result to the client through the knowledge base service. The client selects a desired cost/performance trade-off from the Pareto front of optimal solutions (i.e. selects a single point from the set that best matches client requirements) and passes the information to the Image redistribution service through the knowledge base service which then uploads the VMI/CI. Finally, the Image redistribution service notifies the client upon the success of the upload action.

The offline redistribution flow, as shown in Figure 8, involves the same entities as presented in the online redistribution, albeit with a different aim — to redistribute VMIs/CIs residing in the ENTICE federated repositories in order to minimize the overall cost while preserving the performance (e.g. images deployment times). The major difference compared to the online redistribution entails in the steps from 2 to 5 that includes several metadata exchange between the knowledge base service and the MO service. Basically, the knowledge base service triggers the MO service with a VMI/CI redistribution request that returns the list of provenance metadata. The provenance metadata contains the tracks of previous redistribution executions of the fragments comprised of deployment times of images to the clouds, delivery times of images from one repository to another, and timestamps information. The MO service uses this information to calculate the new candidate redistribution placement and passes them to the client through the knowledge base service. After the client confirms the new redistribution placement, the Image redistribution service is notified through the knowledge base service and starts the redistribution of images. Finally, the Image redistribution service notifies the client upon the success of the redistribution action.

8 Experimental Evaluation

The performance and behaviour of the knowledge base, in case of providing the input data for the MO framework, have been evaluated implicitly by assessing and analysing the outcomes of the Multi-objective optimisation framework in different simulation scenarios.

Moreover, a comprehensive examination was conducted to determine the dependencies between the aforementioned modules. Essentially, the experimental evaluation provided crucial insides on the influence of the knowledge base reasoner on the efficiency of the optimisation process. Lastly, broad analysis was performed in isolation, both on the knowledge base and the optimisation framework, to determine the most suitable execution parameters for both modules.

The evaluation activities were conducted by utilizing the ENTICE test bed environment, which has been distributed across multiple locations in Europe. To be more concrete, for the purposes of this research work, the Multi-objective framework was deployed at the University of Innsbruck premises, that includes 7 physical nodes, four of which have AMD processor with 16 cores and 32 GB RAM. Two of them have AMD processors with 32 cores and 64 GB RAM. The last one has an Intel Xeon processor with 40 cores and 128 GB RAM. On the other hand, the Knowledge base has been arrayed at the University of Ljubljana, where 4 physical nodes of the ENTICE test bed are located. Each of these nodes has dual socket Intel Xeon processor with 8 logical cores per socket, 8 GB of RAM, and 2 x 2 TB of storage. The physical interconnection between the two sites has been established over the Internet network, while the logical communication between the processes was based upon RESTful and SOAP services.

To begin with, both online and offline redistribution modules share the same heuristics, thus entailing unified assessment of the most suitable execution parameters. Therefore, it is essential to properly evaluate the optimisation framework, and thus enable the specification of the proper inputs for the reasoning mechanism behind the knowledge base.

Table 2 provides a comprehensive examination of the quality values for the Pareto optimal set of solutions and the required execution time by the optimisation framework in contrast to the number of evaluations within the genetic algorithm. To properly asses the quality of the Pareto solutions, a comparison has been presented with a set of mapping solutions determined by using "round robin" mapping model for storing VMIs in the ENTICE

federation. The statistical significance of the results has been analysed by applying ANOVA test, which has shown significant difference between the proposed algorithm and the "round robin" mapping strategy, both in respect with the cost and performance objective. The cost objective has 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 10 Gbit and 1 Gbit Ethernet [50]. For readability reasons, the bandwidth values were converted to delivery time needed for 1 Mbit of data to be transferred from the source to the destination. The optimisation framework was specified to search for a set of optimal trade-off distribution solutions for a problem size of 1000 VM images.

Table 2: Assessment of the VMI redistribution algorithm in respect with the number of evaluations. The execution time, cost objective and performance objective are given as a median value from fifteen distinctive executions per experiment. From the optimisation perspective for both, cost and performance objectives lower values mean better. Additionally, statistical significance (i.e. p-value) for both objectives has been computed and is in all the evaluations greater than 0.5%.

Evaluations	Cost ($\times 10^{-8}$)	STD (+/-) ($\times 10^{-8}$)	Difference (%)	Performance ($\times 10^{-8}$)	STD (+/-) ($\times 10^{-8}$)	Difference (%)	Execution time (ms)
10 000	3273	5	0.49	5356	272	18.48	5164
20 000	3262	5	0.84	4732	287	34.11	9742
30 000	3251	5	1.17	4109	316	54.43	15285
40 000	3247	6	1.32	3793	263	67.29	16522
50 000	3240	6	1.53	3526	314	79.98	18492
60 000	3237	5	1.63	3281	259	93.39	25038
70 000	3230	6	1.85	3119	269	103.45	27075
80 000	3225	6	2.00	2905	226	118.47	30747
90 000	3224	6	2.03	2686	178	136.23	34502
100 000	3220	6	2.17	2605	168	143.62	39904

The experimental results clearly show that the number of evaluations within the genetic algorithm has substantial impact on the execution time and the quality of the solutions. For example, increasing the number of evaluation from 10 000 to 100 000 can lead to 140% better quality and 700% higher execution time. Therefore, it can be deduced that for online VMI redistribution, which requires real time optimisation, it is essential to select the minimal number of evaluations which guarantees satisfactory quality of

the solutions. On the contrary, the offline VMI redistribution is not time depended, which implies that higher limit on the number of evaluations can be specified.

Once proper execution parameters for the multi-objective optimisation framework have been determined, it is possible to proceed with the evaluation of the knowledge base and its role in the reduction of the optimisation search space. Optimisation problems are typically constrained by some bounds. Constraints divide the search space into two distinctive regions: feasible and infeasible. The stage in which the constraints are applied can have a great effect on the computational performance of the algorithm. If the constraints are applied after the evaluation of the solutions, it would induce unnecessary computational overhead. The knowledge base provides means for setting the constraints in advance and reducing the input data set, before the process of evaluation of the solutions, thus allowing higher computational efficiency. Some of the constraints that are currently considered are:

- actual free space of the (private) repository,
- geographical distance which is determined by user IP or even manually by user from selecting the preferred continent (e.g. targeted audience) and
- SLA which is taken into account by users requirements.

Table 3 presents the correlation between the execution time, the quality of online redistribution and the input data set provided by the ENTICE knowledge base reasoner. The experiments have been conducted on a set of 500 VMIs, with 1000 repetitions of the optimisation algorithm and the population size of 50 individuals. For each scenario, the experiment was repeated 10 times.

In the case of online VM image redistribution the number of redistributed VM images is usually fixed, thus limiting the opportunities for reducing the search space. This implies, that the knowledge base can only constrain the optimisation by reducing the number of possible repository sites. This process results in lower execution time of the optimisation process by up to 8%, which could be essential for real-time applications. Additionally, reducing the search space induces lower query times for the data access and reduced network bandwidth. Furthermore, to guarantee the quality of the solutions, the hypervolume of the optimal solution has been measured and compared in

relation with different input data sets. From the analysis of the hypervolume values it can be concluded that there is no statistically significant difference between the distributions, thus implying that the quality of the solutions is not affected by reducing the input data set, except for the case in which the dataset was reduced to 20% of the original size.

Table 3: Assessment of the online VMI redistribution algorithm in respect with the fraction of provided data on the available repositories by the knowledge base. The execution times and hypervolume are given as a median value from fifteen distinctive execution per experiment.

	Full Data Set	80%	60%	40%	20%
Query Time (ms)	58	56	50	47	40
STD (+/-)	3.52	5.04	2.61	3.41	3.85
Execution Time MOO (ms)	93	93	92	91	86
STD (+/-)	15.68	15.41	17.51	17.51	9.89
Difference (%)	\	0.12	1.07	2.61	8.29
Hypervolume	0.65	0.63	0.64	0.67	0.53
STD (+/-)	0.13	0.09	0.08	0.091	0.14
Difference (%)	\	-2.07	-1.17	3.49	-18.89
p-value	\	0.42	0.44	0.25	0.01

Lastly, Table 4 shows the correlation between the execution time, the quality of the offline redistribution and the input dataset provided by the knowledge base reasoner. The experiments have been conduced on a varying set of VM images determined by the reasoner. The genetic search algorithm was executed with a population size of 100 individuals until it reached 10 000 distinctive evaluations. For each scenario, the experiment was repeated 10 times.

The offline VM image redistribution is usually conduced across all repositories that fulfil the relevant SLA criteria. This implies, that the knowledge base can apply reasoning in advance to constrain the number of VM images that are required to be redistributed, thus leading to significant reduction of the search space. This process results in lower execution time of the optimisation framework by up to 68%. It is essential to be noted that in the case of offline redistribution it is difficult to measure the quality of the solutions directly. The main reason behind this limitation is the fact every reduction of the number of redistributed images results in smaller size of each individual

Table 4: Assessment of the offline VMI redistribution algorithm in respect with the fraction of provided data on the available VM images by the knowledge base. The execution times and spread are given as a median value from fifteen distinctive execution per experiment.

	Full Data Set	80%	60%	40%	20%
Query Time (ms)	374	315.5	285.5	250.5	206
STD (+/-)	6.29	8.90	10.09	10.90	3.88
Execution Time MOO (ms)	918	745	555	398	297
STD (+/-)	57.12	75.13	90.41	70.54	195.92
Difference (%)	\	18.87	39.62	56.66	67.70
Pareto Spread	0.65	0.66	0.71	0.67	0.71
STD (+/-)	0.15	0.07	0.088	0.13	0.10
% Difference	\	1.69	9.01	2.71	8.96
p-value	\	0.31	0.66	0.38	0.85

in the population, thus making the hyper-volume unsuitable for comparison. To overcome this limitation, the quality of the Pareto solutions has been compared based on the spread of each individual solutions compared to a given centroid. From the analysis of the spread values it can be concluded that there is no statistically significant difference between the distributions, thus implying that the quality of the solutions is not affected by the reducing the input data set as is clearly shown in Figure 9.

9 Conclusions

This study represents viable usage scenarios for knowledge management in the cloud computing domain in general, and an application to the area of distributed VMI/CI storage repositories. It is shown that semantics can be used to facilitate faster optimisation process and management of complex non-functional requirements, including QoS and QoE requirements of the software engineers and applications' end users. This work complements recent research and innovation projects that use semantic technologies in the cloud computing domain including the Smart Cloud Engine, SWITCH and mOSAIC projects. However, all present developments concentrate on using semantics for the runtime of cloud applications and services and not on the storage of software components.

The multi-objective optimisation problem, which is addressed by this study is well-known to be NP-hard [51]. This poses significant computational complexity in case of increasing input data size to the MO solver. In such circumstances, it is shown that the knowledge management and reasoning approach developed in this study can be effectively used to reduce the input for the optimisation algorithm, which in turn reduces the total computational time. The use of the approach leads to more efficient operation of the ENTICE environment and better resulting performance.

Other areas where the ENTICE knowledge and information management approaches is useful are in the actual cloud application design stage. The software engineer may directly pose queries in the ENTICE knowledge base and receive guidance leading to more informed selection of software components (VMI/CIs) and better overall quality and self-adaptive properties of the resulting cloud application.

The lesson learned from this work is that a knowledge management approach can be very instrumental when dealing with heterogeneous federated cloud environments. This area poses some new challenges for the use of semantics, such as the needs for greater expressiveness and for using more complex reasoning mechanisms (other than constraints-based reasoning).

An important area which was not been addressed in the present study is the possibility to geographically distribute the ENTICE knowledge base similarly to the federated storage. The overall ENTICE environment is designed to be distributed and non-centrally managed. Hence, our future goal is to improve the design of the knowledge base so that centralized RDF-storage and management will not be needed.

Acknowledgment

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreements No 644179 (ENTICE project: dEcentralised repositories for traNsparent and efficienT vIrtual maChine opErations) and No. 643963 (SWITCH project: Software Workbench for Interactive, Time Critical and Highly self-adaptive cloud applications).

References

[1] Felter W, Ferreira A, Rajamony R, Rubio J. An updated performance comparison of virtual machines and linux containers. *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015; 171–172, doi:10.1109/ISPASS.2015.7095802.

[2] Microservices Web site. URL http://microservices.io.

[3] Casale G, Chesta C, Deussen P, Di Nitto E, Gouvas P, Koussouris S, Stankovski V, Symeonidis A, Vlassiou V, Zafeiropoulos A, *et al.*. Current and Future Challenges of Software Engineering for Services and Applications. *Cloud Futur. From Distrib. to Complet. Comput.*, Zenodo: Madrid, Spain, 2016, doi:10.5281/zenodo.59258. URL https://doi.org/10.5281/zenodo.59258.

[4] Juju Charms Web site. URL https://jujucharms.com.

[5] Fabric8 Web site. URL http://fabric8.io.

[6] DevOps Wiki page. URL https://en.wikipedia.org/wiki/DevOps.

[7] The OW2 Open Source Community. URL https://www.ow2.org/bin/view/Main/.

[8] Juzna J, Cesarek P, Petcu D, Stankovski V. Solving Solid and Fluid Mechanics Problems in the Cloud with mOSAIC. *Comput. Sci. Eng.* may 2014; **16**(3):68–77, doi:10.1109/MCSE.2013.135. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6695748.

[9] Pahl C, Lee B. Containers and Clusters for Edge Cloud Architectures – A Technology Review. *2015 3rd Int. Conf. Futur. Internet Things Cloud*, IEEE, 2015; 379–386, doi:10.1109/FiCloud.2015.35. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7300842.

[10] Amazon Simple Storage Service Web site. URL https://aws.amazon.com/s3.

[11] Docker Hub Web site. URL https://hub.docker.com/.

[12] OWL 2 Web Ontology Language. URL https://www.w3.org/TR/owl2-overview/.

[13] Razavi K, Razorea LM, Kielmann T. Reducing VM Startup Time and Storage Costs by VM Image Content Consolidation. *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8374 LNCS. Springer Berlin Heidelberg, 2014; 75–84, doi:10.1007/978-3-642-54420-0_8. URL http://link.springer.com/10.1007/978-3-642-54420-0{_}8.

[14] Peng C, Kim M, Zhang Z, Lei H. VDN: Virtual machine image distribution network for cloud data centers. *2012 Proc. IEEE INFOCOM*, IEEE, 2012; 181–189, doi:10.1109/INFCOM.2012.6195556. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6195556.

[15] Zeng L, Xu S, Wang Y. VMBackup: an efficient framework for online virtual machine image backup and recovery. *Concurr. Comput. Pract. Exp.* 2016; **28**(9):2630–2643, doi:10.1002/cpe.3724. URL http://doi.wiley.com/10.1002/cpe.3724.

[16] Schmidt M, Fallenbeck N, Smith M, Freisleben B. Efficient Distribution of Virtual Machines for Cloud Computing. *2010 18th Euromicro Conf. Parallel, Distrib. Network-based Process.*, IEEE Computer Society: Pisa, Italy, 2010; 567–574, doi:10.1109/PDP.2010.39. URL http://ieeexplore.ieee.org/document/5452476/.

[17] Razavi K, Kielmann T. Scalable virtual machine deployment using VM image caches. *SC '13 Proc. Int. Conf. High Perform. Comput. Networking, Storage Anal.*, ACM Press: Denver, CO, USA, 2013; 65, doi:10.1145/2503210.2503274. URL http://dl.acm.org/citation.cfm?doid=2503210.2503274.

[18] Kimovski D, Marosi A, Gec S, Saurabh N, Kertesz A, Kecskemeti G, Stankovski V, Radu P. Distributed environment for efficient virtual machine image management in federated cloud architectures. *Concurrency and Computation: Practice and Experience*, 2017.

[19] Jin W, Li M, Khasnabish B. Content De-duplication for CDNi Optimization 2013. URL http://www.ietf.org/internet-drafts/draft-jin-cdni-content-deduplication-optimization-02.txt.

[20] Sheng Y, Xu D, Wang D. A Two-Phase Differential Synchronization Algorithm for Remote Files. *Algorithms Archit. Parallel Process.*, Hsu CH,

Yang LT, Park JH, Yeo SS (eds.). Springer Berlin Heidelberg, 2010; 65–78, doi:10.1007/978-3-642-13119-6_6. URL http://link.springer.com/10.1007/978-3-642-13119-6{_}6.

[21] Policroniades C, Pratt I. Alternatives for Detecting Redundancy in Storage Systems Data. *Proc. Annu. Conf. USENIX Annu. Tech. Conf.*, ATEC '04, USENIX Association: Boston, MA, USA, 2004; 73–86. URL http://dl.acm.org/citation.cfm?id=1247415.1247421.

[22] Jain N, Dahlin M, Tewari R. TAPER: Tiered Approach for Eliminating Redundancy in Replica Synchronization. *Proc. 4th USENIX Conf. File Storage Technol.*, USENIX Association: San Francisco, CA, USA, 2005; 281–294. URL http://dl.acm.org/citation.cfm?id=1251028.1251049.

[23] Kulkarni P, Douglis F, LaVoie J, Tracey JM. Redundancy elimination within large collections of files. *Proc. 2004 USENIX Annu. Tech. Conf.*, USENIX Association: Boston, MA, 2004; 59–72, doi:10.1.1.85.7036. URL http://static.usenix.org/events/usenix04/tech/general/full{_}papers/kulkarni/kulkarni{_}html/.

[24] Fonville M, van Deventer O, de Boer PT, van Sinderen M. The Virtual Machine Delivery Network. Master thesis, University of Twente 2014.

[25] Jayaram KR, Peng C, Zhang Z, Kim M, Chen H, Lei H. An empirical analysis of similarity in virtual machine images. *Proc. Middlew. 2011 Ind. Track Work. - Middlew. '11*, ACM Press: Lisbon, Portugal, 2011; 6, doi:10.1145/2090181.2090187. URL http://dl.acm.org/citation.cfm?doid=2090181.2090187.

[26] Sack H, Krüger U, Dom M. A knowledge base on np-complete decision problems and its application in bibliographic search 2006.

[27] Zhuge H. Semantic linking through spaces for cyber-physical-socio intelligence: A methodology. *Artificial Intelligence* 2011; **175**(5):988 – 1019, doi:http://dx.doi.org/10.1016/j.artint.2010.09.009. URL http://www.sciencedirect.com/science/article/pii/S0004370211000208.

[28] Zhuge H. Interactive semantics. *Artificial Intelligence* 2010; **174**(2):190 – 204, doi:http://dx.doi.org/10.1016/j.artint.2009.11.014. URL http://www.sciencedirect.com/science/article/pii/S0004370209001441.

[29] Zhuge H. Communities and emerging semantics in semantic link network: Discovery and learning. *IEEE Transactions on Knowledge and Data Engineering* June 2009; **21**(6):785–799, doi:10.1109/TKDE.2008.141.

[30] MyExperiment Web site. URL http://www.myexperiment.org/home.

[31] InteliGrid Project Web site. URL http://inteligrid.eu-project.info.

[32] OntoGrid Project Web site. URL http://mayor2.dia.fi.upm.es/oeg/index.php/en/completedprojects/80-ontogrid.

[33] mOSAIC Project Web site. http://developers.mosaic-cloud.eu/confluence/display/MOSAIC/mOSAIC. URL http://developers.mosaic-cloud.eu/confluence/display/MOSAIC/mOSAIC, 2016-09-20.

[34] Bellini P, Cenni D, Nesi P. Smart Cloud Engine and Solution Based on Knowledge Base. *Procedia Comput. Sci.* 2015; **68**(Vm):3–16, doi:10.1016/j.procs.2015.09.219. URL http://dx.doi.org/10.1016/j.procs.2015.09.219http://linkinghub.elsevier.com/retrieve/pii/S1877050915030641.

[35] SWITCH Project Web site. URL http://www.switchproject.eu.

[36] ENTICE Project Web site. URL http://www.entice-project.eu.

[37] Chef Web site. URL https://www.chef.io/chef.

[38] Puppet Web site. URL https://puppet.com.

[39] JFrog Bintray Web site. URL https://www.jfrog.com/bintray.

[40] RDF Vocabulary Description Language 1.0: RDF Schema (RDFS). URL https://www.w3.org/2001/sw/wiki/RDFS.

[41] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. URL https://www.w3.org/Submission/SWRL/.

[42] Ontology Design Pattern proposals from ODP users. URL http://ontologydesignpatterns.org/wiki/Community:ListPatterns.

[43] Prud'hommeaux E, Labra Gayo JE, Solbrig H. Shape expressions: An RDF validation and transformation language. *Proc. 10th Int. Conf. Semant. Syst. - SEM '14*, ACM Press: Leipzig, AA, Germany, 2014; 32–40, doi:10.1145/2660517.2660523. URL http://dl.acm.org/citation.cfm?doid=2660517.2660523.

[44] Heflin J. An Introduction to the OWL Web Ontology Language. chap. 2, 2007. URL http://www.cse.lehigh.edu/{~}heflin/IntroToOWL.pdf.

[45] Abburu S. A Survey on Ontology Reasoners and Comparison. *Int. J. Comput. Appl.* 2012; **57**(17):33–39, doi:10.5120/9208-3748.

[46] Voigt M, Mitschick A, Schulz J. Yet Another Triple Store Benchmark? Practical Experiences with Real-World Data. *Proc. 2nd Int. Work. Semant. Digit. Arch. (SDA 2012)*, vol. 912, Mitschick A, Loizides F, Predoiu L, Nürnberger A, Ross S (eds.), CEUR-WS.org: Paphos, Cyprus, 2012; 85–94. URL http://dblp.uni-trier.de/db/conf/ercimdl/sda2012.html{#}VoigtMS12.

[47] Jenkins automation server – building, deploying and automating any project. URL https://jenkins.io/.

[48] Kimovski D, Saurabh N, Gec S, Stankovski V, Prodan R. *Multi-objective Optimization Framework for VMI Distribution in Federated Cloud Repositories*. Springer International Publishing: Cham, 2017; 236–247, doi:10.1007/978-3-319-58943-5_19. URL http://dx.doi.org/10.1007/978-3-319-58943-5_19.

[49] Kimovski D, Saurabh N, Stankovski V, Prodan R. Multi-objective Middleware for Distributed VMI Repositories in Federated Cloud Environment. *Scalable Comput. Pract. Exp.* oct 2016; **17**(4):299–312, doi:10.12694/scpe.v17i4.1202. URL http://www.scpe.org/index.php/scpe/article/view/1202.

[50] Feng WC, Balaji P, Baron C, Bhuyan LN, Panda DK. Performance characterization of a 10-gigabit ethernet TOE. *Proc. - Symp. High Perform. Interconnects, Hot Interconnects*, vol. 2005, IEEE, 2005; 58–63, doi:10.1109/CONECT.2005.30. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1544578.

[51] Diakonikolas I. Approximation of Multiobjective Optimization Problems. Phd thesis, Columbia University 2011.

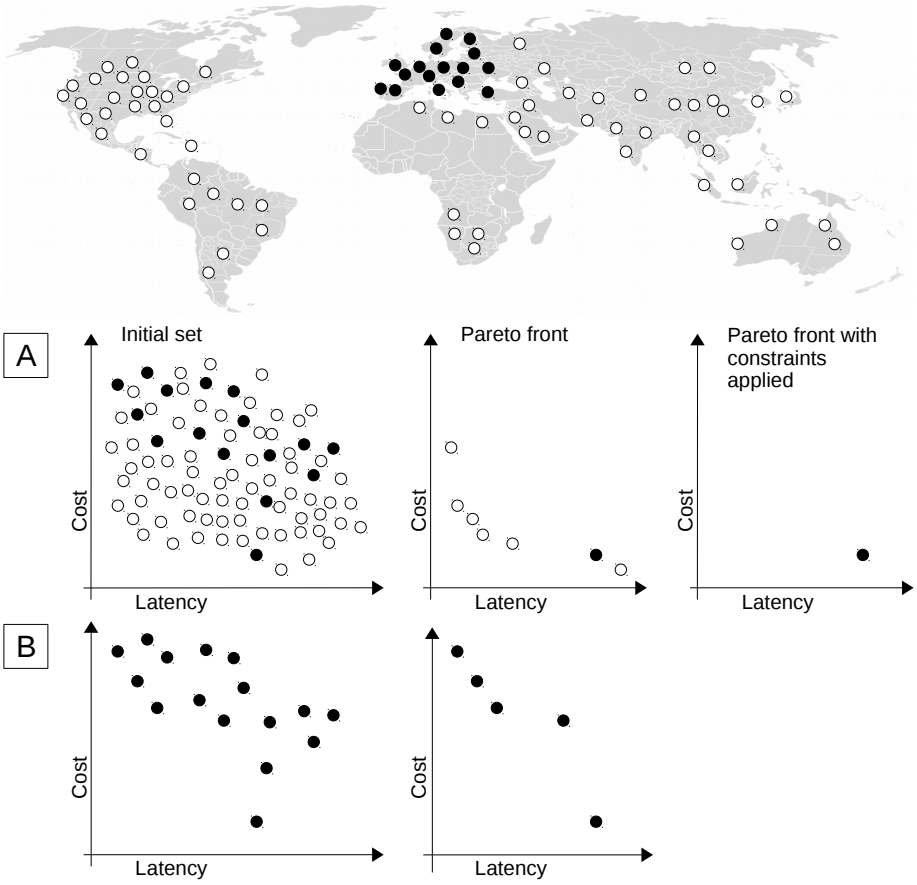

Figure 3: Comparison of two different procedures (A and B) of data preparation for the Pareto front computation. Black dots denote storages subject to the user's constraint, e.g. particular VMI has to be stored in Europe; white dots denote storages that do not meet the constraints. Plots below the map signify the difference between the two procedures. The first (A) procedure starts with all storages and computes the Pareto front on the samples taken from the whole set of storages. In case when the white dots outnumber the black dots by a large margin, the method may require significant computations to find the Pareto front satisfying the constraints. This is not the case for the second (B) procedure.

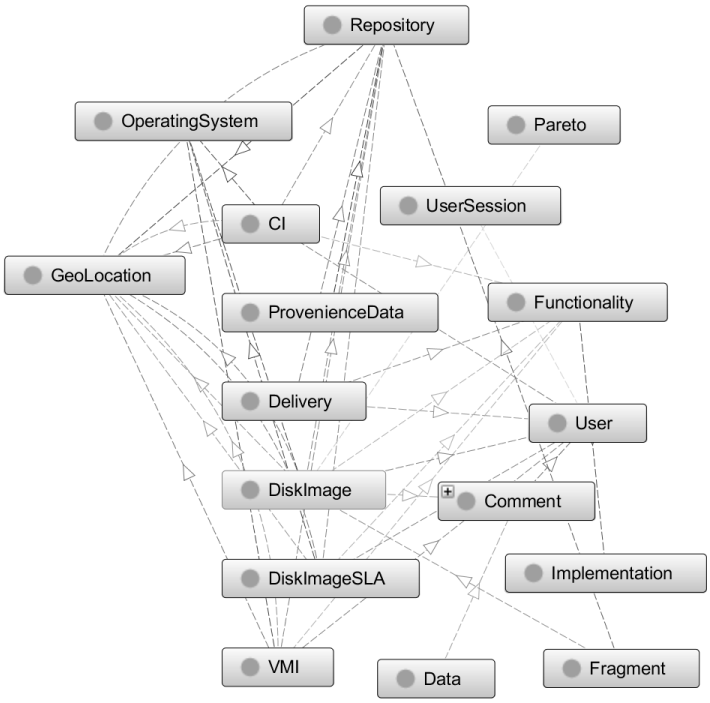

Figure 4: Graph representation of the ENTICE ontology.

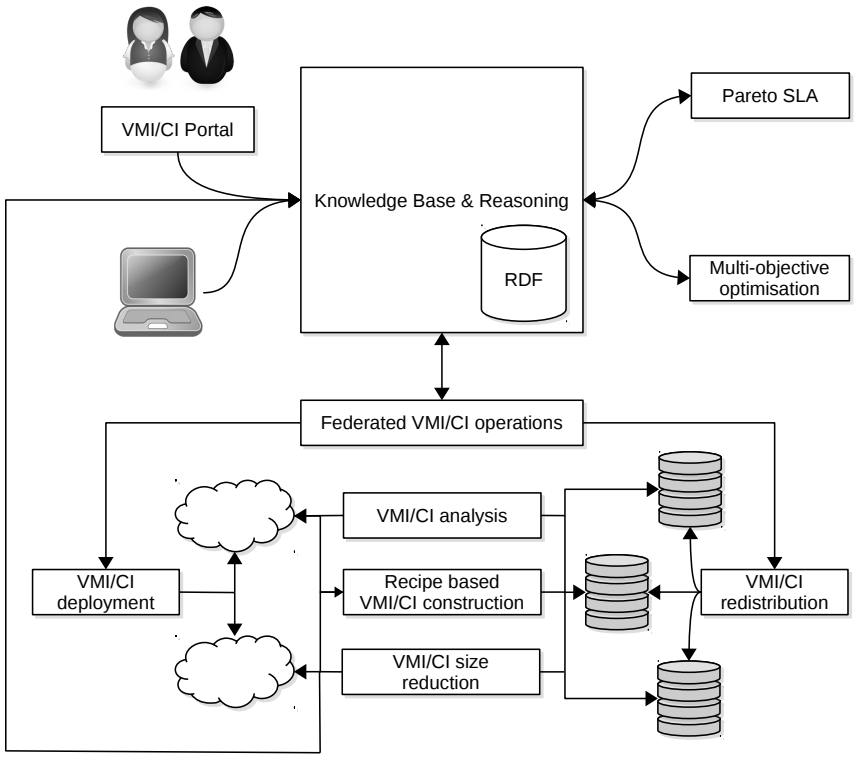

Figure 5: Design and integration of the knowledge base with services of the ENTICE environment.

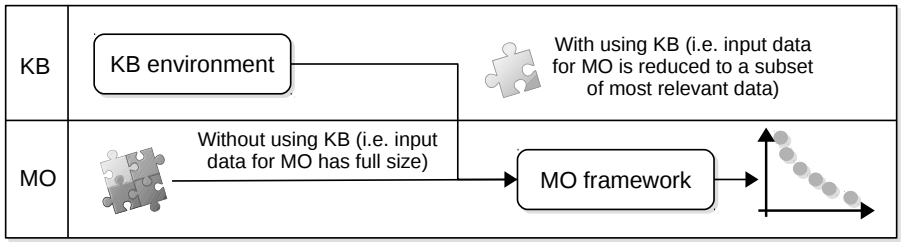

Figure 6: Online distribution of MO — by using MO itself and by using the knowledge base.

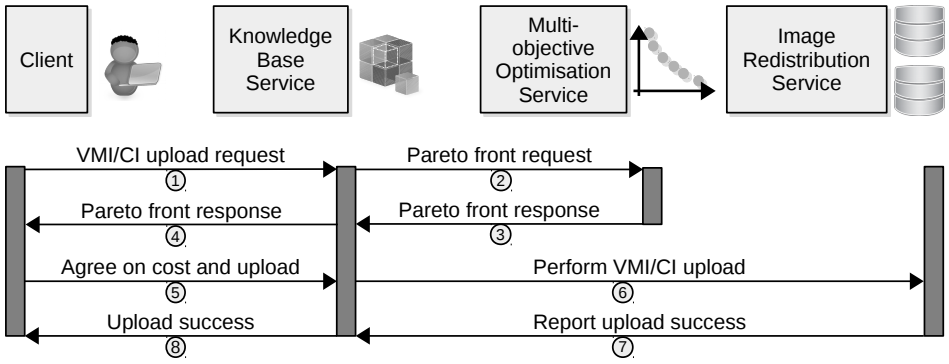

Figure 7: Interactions between the knowledge base and MO framework for online redistribution.

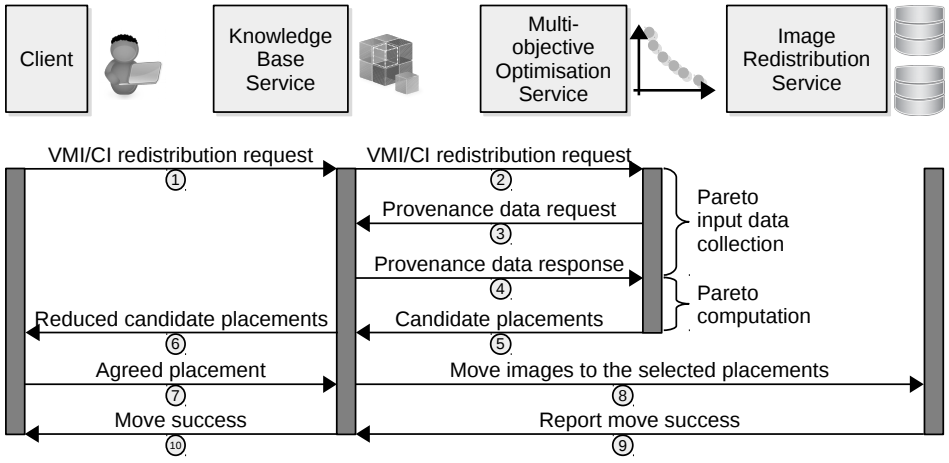

Figure 8: Interactions between the knowledge base and MO framework for offline redistribution.

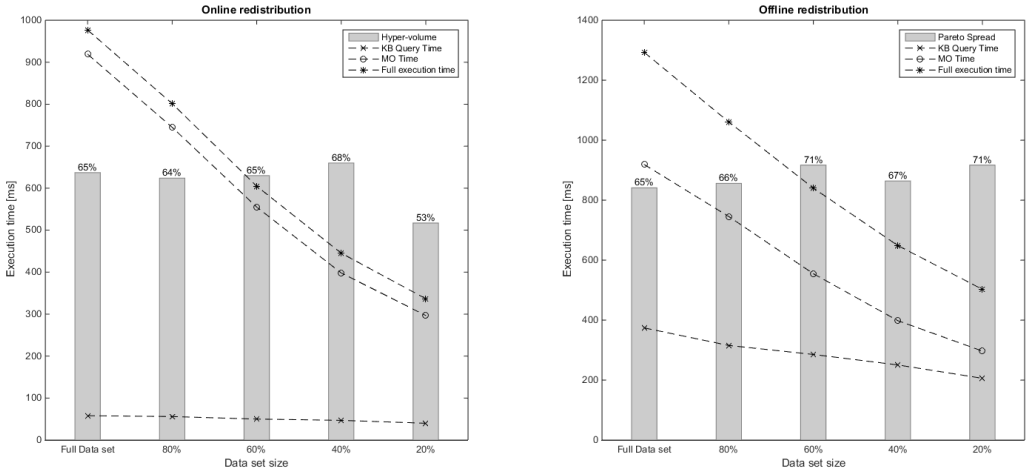

Figure 9: Online and offline redistribution - execution times and quality of the solution on different data set subsets.