Optimizing Complex Systems with Advanced Reasoning

Elena Vasquez, Liam Chen
California Institute of Technology, Pasadena, USA; University of Cambridge, Cambridge, UK

***Abstract*—ENTICE is a set of innovative software services currently being developed to facilitate efficient operations of distributed Virtual Machine and container images (VMI/CI) repositories. Its operation necessitates various decision making for which a solver for Multi-Objective Optimisation (MOO) problems is used. However, the solver is a bottleneck due to its computational complexity. In order to be able to reduce the search space for the solver, we have developed an ontology and corresponding Knowledge Base (KB) that underpins the operation of the ENTICE environment. The Knowledge Base is developed based on the Jena Fuseki technology. To address the problem of computational complexity, constraint based queries and different reasoning mechanisms are applied. The Knowledge Base services are then integrated with other ENTICE services including the MOO solver. It is shown that this approach significantly reduces the computational complexity for the MOO, thus it shortens the optimisation time, and makes it possible to use the MOO for both strategic (decisions that can be made up to one day in advance) and dynamic (decisions requiring response within one minute) decision making possible.**

I. Introduction

The software engineering methodologies and processes are radically changing in the last years, which is mainly due to the rise of new, light-weight Cloud computing technologies (e.g. Docker[1]). Nowadays, software engineers are concentrated more on search by needed functionality in well-known Open Source software repositories, satisfying specific elasticity and performance properties, and its combination into suitable services via dashboards (such as Juju[2] or Fabric8[3]), than on actual code writing.

An important new trend which affects greatly the software engineering processes and methodologies is the emergence of literally thousands decentralized repositories of software assets [1], which are made ready to be used in applications. Modern ways of software engineering (e.g. by using micro-services[4]) require the use of Virtual Machine or container images (VMI/CI). The use of VMI/CIs makes it straightforward to search, combine into an application, and then fetch and deploy the needed software solution on a designated Cloud infrastructure.

Thus, the operation of distributed repositories of software assets is an important part of the performance model of Cloud applications. Each and everytime, the software components need to be fetched from a repository and delivered to a specific Cloud provider to be deployed and operated there. The ENTICE project [1] aims to deliver a new set of services and an environment facilitating efficient operations of distributed repositories of VMI/CIs though specific technologies for optimization of the storage minimization and VM and container images delivery mechanisms. While existing Cloud providers mostly focus on Quality of Service (QoS) improvements from the infrastructure and network-level view points, they currently seriously neglect the possibilities for improvements in the software engineering processes and methodologies, which includes providing the software engineer with information and knowledge that can be used to significantly improve the design of its software for better quality including its functional and non-functional properties [2], [3].

Generally speaking, due to the complexity of all the relations in the ENTICE environment, a knowledge management approach may be useful. The use of information and knowledge could contribute to improved software engineering productivity, the delivery of more context-aware software (e.g. software that can be made self-adaptive at runtime given the various user needs, deployment topologies and infrastructures possible), new approaches could be designed for building software out of reusable software components (such as VMI/CI), quality of service guarantees could be provided in various situations, improved requirements engineering process (e.g. component search by functionality), improved privacy and security aspects, software maintainability and so on.

An important goal of the ENTICE project is therefore to address the needs for efficient operations of VMI/CI repositories through the collection and use of information and knowledge. While there are many studies on the use of Semantic Web and Linked Data technologies in various domains, they are currently seriously lacking in the domains of distributed, grid, Cloud and currently, Fog/Edge computing [4]. In this study, we intend to address this gap by designing and developing the ENTICE KB and its reasoning mechanisms. It is designed to tackle some of the knowledge management and information delivery issues involved when achieving efficient operations of distributed VM and container images repositories. In the ENTICE environment information and knowledge could be used to: (1) achieve more simplified creation of lightweight and highly optimized VM images, (2) software search by functionality, (3) automatic decomposition and distribution of VM and container images based on Multiobjective Optimisation (MO) [5] (performance, economic costs, storage size, and QoS needs), (4) elastic auto-scaling of applications on Cloud infrastructures based on their actual load, (5) optimized VM and container images interoperability across Cloud infrastructures, (6) to help avoid the Cloud provider lock-in problem and so on. In this study we focus on its most delicate

[1] http://www.entice-project.eu/

part - the interoperation between the ENTICE KB and the MO service.

The paper is structured as follows. In Section II we present generic use cases for the ENTICE KB and an analysis of the requirements. In Section III we present a high-level overview of the KB architecture, the developed ontology and the implementation. Further to this, in Section IV we present the conducted experiments. Section V summarizes the main results, their impact and future work.

II. Use cases and requirements analysis

The software development and engineering life-cycle is constantly evolving due to the emergence of new Cloud computing technologies. While building applications from reusable software components packed onto VMIs or CIs, software engineers have to deal with complex requirements and, among other, consider interoperability and performance issues of their applications. Achieving efficient operation of the repositories of software components, including data, is therefore extremely important. In this context, a KB could be a very useful support for both strategic decision making by the software engineer and more dynamic decision making used by the various services of the distributed repositories of software assets, that is, the ENTICE environment.

A. Use cases

As illustrated in Figure 1, our intention is to develop a KB which will be used by both the software engineer in the development process and also by the running services of the ENTICE environment allowing for more efficient operations of distributed software assets.

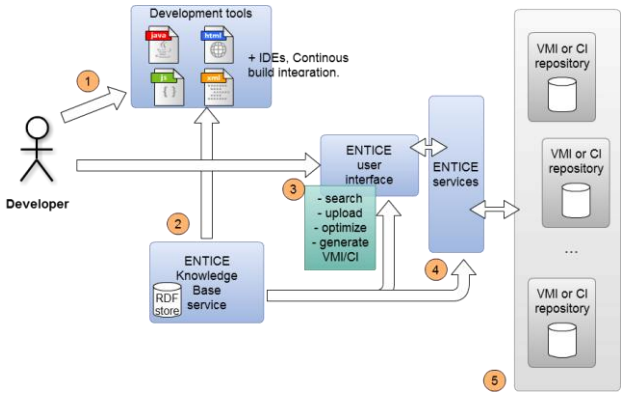

Fig. 1. Information services and the ENTICE KB in the context of the ENTICE environment

The developer has the possibility to access VMIs or CIs from public repositories and combines them into an application, which is then deployed on a Cloud provider. The overall software engineering life-cycle is used to gain, exchange and manage information through the use of the KB. The KB itself is made available as a software service. Here, we proceed by analysing some basic use cases:

1) Development tools (such as Juju) are basic instruments used by a developer of a Cloud application. Such tools do not yet provide enough information to the developer concerning the deployment stage. Programs, APIs and other services has to be deployed using a server that runs on a specific operating system. Creating VMI and CI images, 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 KB RESTful service or through a specifically designed User Interface (UI) facilitating repositories-wide software search and discovery.
2) ENTICE KB service can be described as a service that uses a RDF store and mechanisms such as validations, reasoners, rules and so on, serving information to all other components of the architecture. The service interchanges the data through the ENTICE external interfaces, the developer API as described above 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 including pre-installed user specific services and applications, or even system based settings (e.g. network, SSH, system credentials etc.). Beside the upload of images, the user has the possibility to generate images by using a script (e.g. Chef[2]or Puppet recipe[3]) that includes functional requirements. In the UI search screen the developer can search among all available public VMIs/CIs, which are stored in public VMI/CI repositories, such as those of Amazon S3[4] or Docker.
4) The services of the ENTICE environment process 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, 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, documentation etc.) in order to significantly reduce the VMI size and not affecting the mandatory running applications

[2] https://www.chef.io/chef/
[3] https://puppet.com/
[4] https://aws.amazon.com/s3/

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.

5) Currently a large amount of public VMI/CI repositories are available on the internet (e.g. more than 161.000 repositories can be accessed using JFrog Bintray[5]). Thus, the idea to integrate those repositories in the ENTICE system seems an interesting advantage. All the information of VMIs/CIs available in those repositories can be stored in the KB with addition to new constraints, dependencies and other derived data from reasoning or user experience. For example, the majority of CIs are specific mainly running only one service (e.g. Apache Tomcat) and user specific applications (e.g. WEB application deployed on Tomcat). By using the ENTICE system user is able to obtain a size reduced optimised CI and constraints about the upgrade possibilities of the service (e.g. change the database from MySQL to NoSQL) with preliminary requirements (e.g. platform, library etc.).

The detailed analysis of the various use cases related to information and knowledge management for software engineering Cloud computing applications led us to a list of functional and non-functional requirements that we present in the following section.

B. Requirements

This section introduces the identified development requirements related to ENTICE KB. First we identify the functional requirements in order to fulfil the overall ENTICE system capabilities supported by the KB. Then we present the most critical non-functional requirements, which are necessary to preserve elevated overall system Quality of Service (QoS) and Quality of Experience (QoE). The last measures the actual user's experience based on its satisfaction with throughout service in use.

1) Functional requirements: The system can be accessed only by registered users in the ENTICE system. So, the ENTICE system KB 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 KB 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 KB service must also support connectivity through external interfaces. Thus, we have to support common used connectivity protocols like REST and others (e.g. Web Service Definition Language (WSDL)). In some cases the KB service has to access sensitive data and certification mechanism must be supported (e.g. Secure Shell (SSH)).

2) Non-functional requirements: KB 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:

- seemly high performance of the services interconnecting through KB (e.g. fast query responses);
- distributivity of KB service that will be addressed in detail during the project;
- scalable KB 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 KB storage;
- availability where monitoring of RDF store service has to be running and minimize the down time of KB service or notify the administrators for major system faults;
- security that overlaps with functional requirements;
- data integrity by using different reasoners supporting validation (e.g. HermiT[6], Pellet, etc.).

III. KB architecture, ontology and implementation

We experimented with various degrees of expressiveness to model the relationships among the entities, the use of advanced constraint mechanisms, RDF validation [20], complex data querying by using of reasoning mechanisms and other approaches. A primary use of the KB in ENTICE project is to store RDF based data for all project subsystems and services through main central service that directly communicates with KB and supports all the available integrated KB mechanisms. To fulfil the system interoperability, KB service must support the ability of exchanging RDF data using simple queries, different reasoners, and rule based constraint verification to minimize human errors that may occur through UI or scripts (e.g. functional descriptions of VMIs/CIs). Those mechanisms will be indirectly described in the following section on how KB intersects other ENTICE services, especially the Pareto SLA and Multi-objective optimisation part.

[5] https://bintray.com/

[6] http://www.hermit-reasoner.com/

A. High-level architecture

The ENTICE environment can be seen as repositorybased 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. We proposed the high-level architecture that is depicted in Figure 2 and has the following key subsystems:

- KB 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 User Interface used by the developer to search for software artefacts across the distributed repositories,
- VMI/CI Synthesis, which facilitates the synthesis of VMI/CI based on recipes,
- VMI/CI Analysis, which facilitates the optimisation of the size of a VMI/CI,
- 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[7] 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 avail-able Cloud management systems (e.g. OpenNebula[8]

  OpenStack [9] 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.

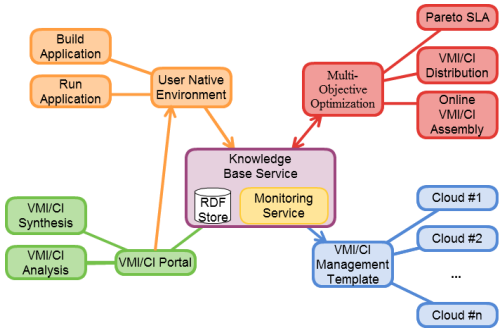

Fig. 2. A high-level overview of the architecture of the ENTICE KB

As it can be seen in Figure 2, knowledge management and information supply plays a crucial role in the operation of the overall ENTICE environment. Due to space limitations, in the following we focus on the use of our KB and its reasoning methods in relation to some NP hard optimisation problems, which are addressed by a combination of MO and Pareto SLA technique. NP hard optimisation problem, for example is the optimal distribution of a specific VMI across the distributed repositories which allows for specific maximum delivery time at a

specific maximum cost.

1) Multi-objective Optimisation Framework: The ENTICE multi-objective optimisation framework for VMI distribution in Federated Cloud repositories is encompassed around unified multi-objective optimisation module, which can be utilized for multiple different optimisation purposes. Internally, the MO module is branched into distinctive sub-modules. Each of the sub-modules has been tailored specifically for a given task. The “Initial Distribution” sub-module covers the multi-criteria evaluation of the possible repository sites where the VMIs or associated data sets can be initially stored. Afterwards, the “Offline VMI Redistribution” sub-module encapsulates the optimisation of the VM images distribution within the federated repository sites. Additionally, the framework provides a possibility for “Online VMI Redistribution”, which encloses the optimisation of specific VMIs while an application is being executed. By taking into account the VMIs usage patterns the algorithm is capable of providing multiple trade-off solutions, where each solution

[7] http://jmetal.sourceforge.net/

[8] http://opennebula.org/

[9] https://www.openstack.org/

represents a possible mapping between the stored images and available repository sites. The framework is also envisioned to provide a means for Online VMI assembly during image provisioning.

2) Pareto SLA: The process of Multi-objective optimisation usually results in a set of trade-off solutions called Pareto front. A Pareto front is an essential tool for decision support and preference discovery, whose shape can provide new insights and allow Cloud providers and users explore the space of non-dominated solutions with certain properties, possibly revealing regions of interest that are impossible to see otherwise. The ENTICE environment introduces strategies from the field of multiobjective optimisation, which have resulted in a novel term called Pareto SLA (P-SLA). Together with a proper decision making tool the optimisation will allow to the users to select the "best" Pareto point from the P-SLA that fulfils their specified deployment characteristics.

B. Ontology

This section presents the developed ontology, which is used in the operation of the ENTICE KB. Special care was put to model the overall ENTICE environment. The design of ontology was done iteratively, at the beginning by identifying the entity classes of all subsystems, their dependencies in a form of relationships and the constraints of entity data to satisfy data exchange between ENTICE services. Particular care was put on developing reasoning capabilities (e.g. with Pellet) and on the possibilities to use RDFS and SWRL rule engines. By using reasoners, it is possible to facilitate environment information flow and minimize human errors, data inconsistency and even simplify data transfer between the ENTICE services. The developed ontology stores the concepts of entire ENTICE environment, such as:

- concepts of resources (e.g. software components, VMI/CI, 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).

The entire ENTICE ontology with interconnected entities is presented in Figure 3.

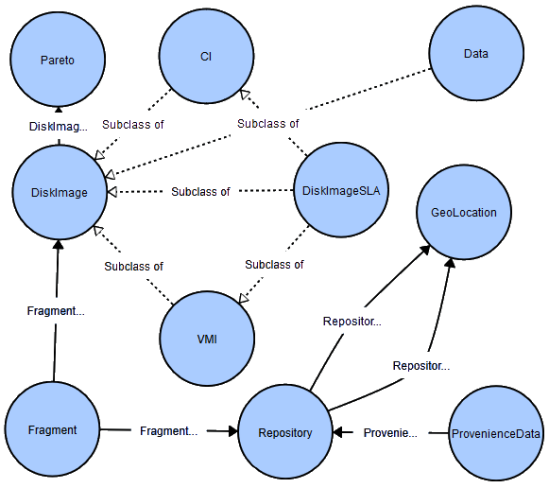

Fig. 3. Graph representation of ENTICE ontology

C. Constraint based reasoning

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 are rule-based reasoners.

The introduction of constrain based reasoning within the ENTICE environment could reduce the execution time and improve the efficiency of the MOO framework. The execution complexity of the framework can be estimated by using the number of objectives m and the size of the population of the genetic algorithm N. In order to uncover the full set of non-dominated solutions and all subsequent fronts, the NSGA-II based search algorithm implemented in this work requires algorithm complexity of $O(mN^2)+ O(N^2)+O(mN^2)$. As with any other genetic optimization algorithm the estimation of the optimal set of solutions will largely depend on the population size and the search space. Reducing the search space will require smaller population size, thus reducing the execution time. To decrease the search space, the reasoner applies constrains before delivering the full set of input variables to the optimization framework. Therefore, the search algorithm only considers narrower search space and requires smaller population size to sort the feasible solutions by dominance.

D. Implementation

The entire ontology development of ENTICE project was done by using the ontology editor Protégé[10]. An ontology of the

[10] http://protege.stanford.edu/

ENTICE environment is implemented in the OWL2 [21] using Turtle format due its human readability. The 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.) [22], its default integrated reasoner and have satisfactory performance [23]. During the implementation phase of the project new mechanisms will be integrated to facilitate the KB service deployment, testing and security (e.g. Apache Shiro[11]).

IV. Experimental Study

This section describes the experimental study of KB mechanisms on MO framework. Even the KB mechanisms are involved in the entire ENTICE environment, we present one use case developed so far. The focus of the section is divided into two parts: (i) presentation of the Multi-objective optimisation service, more concrete the work flow between MO framework and KB and (ii) evaluation that emphasizes the main advantages of the KB with the focus on the main contributions.

A. 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 KB 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 KB 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 KB 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

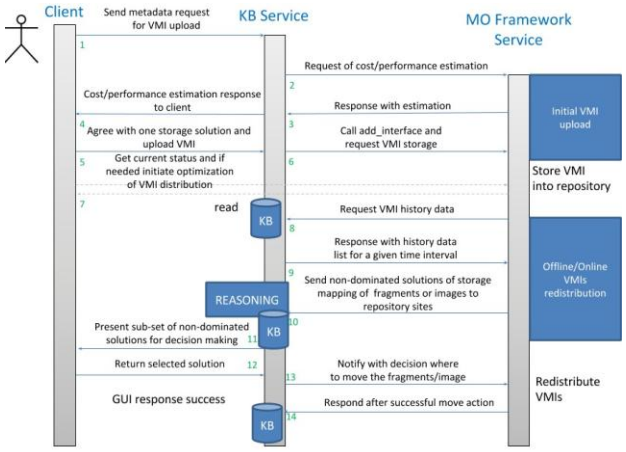

Fig. 4. Interactions between the KB and MO framework

performed in a similar manner. When the optimisation process is initiated, the framework sends a query for the required data to the KB. The KB has been constructed in such a way, to provide only the relevant input data as was described in Section III.

B. Evaluation

The KB is in the first place been utilized for storing of the output data of the optimisation process, i.e. the set of non-dominated or Pareto solutions. Through the KB the optimal set of solutions is provided to the administrative entity of the federation or the corresponding user, which acts as a decision maker, and should select the most appropriate solution based on the pre-defined decision making policy.

The flow of interactions between the KB and the optimisation algorithm, in the cases of initial VMI upload and offline VMI redistribution is presented in Figure 4. In several steps the data request from the KB service is required. 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 KB provides means for setting the constraints on the input data in advance, 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 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 be taken into account by users requirements.

[11] http://shiro.apache.org/

Most of the requests are simple RDF SELECT based queries (or nested ones), but in the phase of preparing non-dominated solutions additional inference engines (e.g. rule reasoners using forward chaining, Jena's default RDFS rule reasoner, Pellet's SWRL reasoner etc.) are needed to be called to reduce the result list. During the development those mechanisms were particularly analysed to meet the needed query criteria. To summarize the main contribution of the constraint based reduction of search space, the Table I depicts that by reducing the input data for the MOO algorithm, average cost can be decreased by approximate 2% on 10000 evaluations (iterations) which is the effect of the reduced search space and usually meets the user requirements in the online distribution.

In order to present the effect of the number of evaluation on the algorithm's execution time we have experimentally evaluated the optimization framework in regards with the total number of evaluations and have present it on Figure 5. In the given scenario the population size of the MOO algorithm and the size of the solution vector were kept constant, while the number of evaluations was gradually increased. It can be easily concluded that the KB base is essential for controlling the execution time of the optimization algorithm in specification with the user's requirements. Furthermore, the number of evaluations can have large effect on the quality of the solutions provided by the MOO algorithm. As shown in Table I as the number of evaluation increases, the improvements in quality of both objectives is substantial, at a cost of increased execution time and utilization of more hardware resources.

The initial experimental evaluations, which are presented on Figure 6, have shown that the size of the solution vector (number of fragments to be rearranged) have a large effect on the execution time of the MOO algorithm. For that reason the KB reasoning will first select which fragments are relevant for redistribution and only then will execute the optimization, thus reducing the execution time.

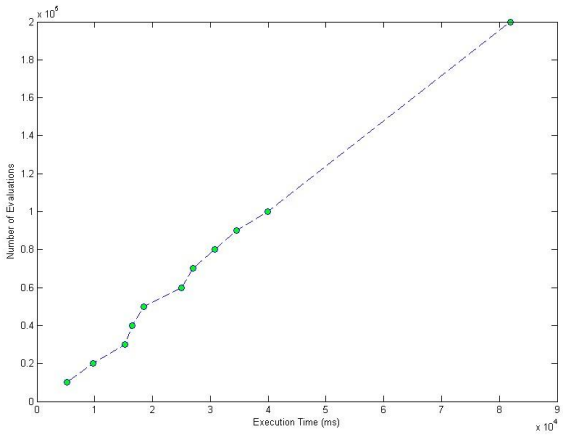

Fig. 5. Execution time versus number of evaluations during offline redistribution with the maximum

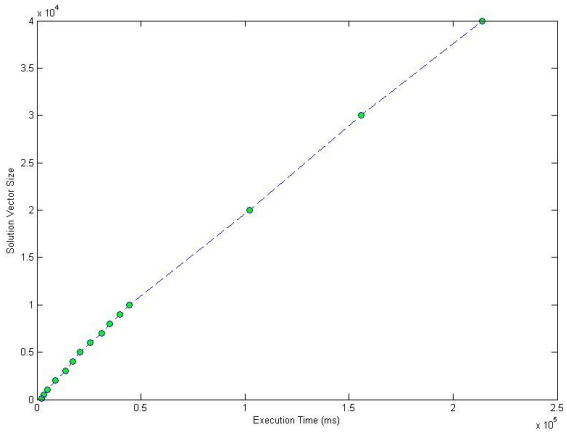

Fig. 6. Execution time versus size of solution vector during offline redistribution

V. Discussion and Conclusions

It has become apparent in recent years that complex structured knowledge and information are needed in the overall life-cycle of Cloud computing applications. This is an area where knowledge management approaches may be used. Some new studies using semantic technologies for Cloud computing have emerged, such as those of the Smart Cloud Engine, the SWITCH and mOSAIC projects. However, the ontologies and knowledge management approaches of these projects require many improvements. The present study aims to complement these efforts by focusing on very specific aspects of the software engineering life-cycle, particularly on optimisation methods for software packaging, search and discovery, storage, and delivery.

Some of the most difficult problems tackled by the study, such as the multi-objective optimisation problems, are well-known to be NP hard [24]. In such circumstances, it is shown that a reasoning approach can be effectively used to reduce the search space of all possibilities to those that are most plausible ones. This leads to more efficient operation of the ENTICE environment and better resulting performance.

Other areas where the ENTICE knowledge and information management approaches could be useful are in the actual Cloud application design stage. The software engineer can directly pose queries in the KB and receive guidance leading to more informed selection of software components and better overall quality and self-adaptive properties of the resulting Cloud application.

The lesson learned from this work is that reasoning can be very instrumental when addressing NP-hard problems in modern software engineering. This area poses some new challenges for the use of semantics, such as the needs for greater expressivity than it is usual with many productiongrade applications.

An important area which was not tackled in the present study, but it will be in the focus of our research in the next period is the possibility to geographically distribute the KB. The overall ENTICE environment is designed to be distributed and non-centrally managed. Hence, our goal is to also design the KB in a way that does not need centralized storage and management.

References

[1] C. Pahl, and B. Lee: Containers and Clusters for Edge Cloud Architectures – A Technology Review. Future Internet of Things and Cloud (FiCloud), 3rd International Conference on, Rome, pp. 379-386, 2015.

[2] M. Kassab, and G. El- Boussaidi: Towards a Knowledge-Based Representation of Non-Functional Requirements. In *The Seventh International Conference on Software Engineering Advances*, Lisbon, Portugal, pp. 18-23, 2012.

[3] F.O. Bjørnson: Knowledge Management in Software Process Improvement. PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, 2007.

[4] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli: Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing* (MCC '12). ACM, New York, NY, USA, 13-16, 2012.

[5] J. Branke, K. Deb, K. Miettinen, and R. Slowinski: *Multiobjective Optimization, Interactive and Evolutionary Approaches*. Springer, Vol. 5252, 2008.

[6] P. Bellini, D. Cenni, and P. Nesi: Smart Cloud Engine and Solution based on Knowledge Base. In *Procedia Computer Science*, volume 68, pp. 3 – 16, 2015.

[7] G. Cretella, B. Di Martino, and V. Stankovski: Using the mOSAIC's semantic engine to design and develop civil engineering cloud applications. In Proceedings of the *14th International Conference on Information Integration and Web-based Applications & Services (IIWAS '12)*. ACM, New York, NY, USA, pp. 378-386, 2012.

[8] K. Razavi, L. M. Razorea, and T. Kielmann: Reducing vm startup time and storage costs by vm image content consolidation. In *Euro-Par 2013: Parallel Processing Workshops*, pp. 7584. Springer Berlin Heidelberg, 2013.

[9] B. Khasnabish, W. Jin, and M. Li: Content De-duplication for CDNi Optimization. IETF, 2013.

[10] Y. Sheng, D. Xu, and D. Wang: A Two-Phase Differential Synchronization Algorithm for Remote Files. In *Algorithms and Architectures for Parallel Processing*, pp. 65–78, 2010.

[11] C. Policroniades, and I. Pratt: Alternatives for detecting redundancy in storage systems data. In *Proceedings of the 2004 USENIX Annual Technical Conference*, pp. 73–86, 2004.

[12] N. Jain, M. Dahlin, and R. Tewari: Taper: Tiered approach for eliminating redundancy in replica synchronization. In *Proceedings of the 4th Usenix Conference on File and Storage Technologies (FAST)*, 2005.

[13] P. Kulkarni,F. Douglis, J. LaVoie, and J. Tracey: Redundancy elimination within large collections of files. In *The USENIX Annual Technical Conference*, General Track, pp. 59–72, 2004. [14] M. Fonville: The Virtual Machine Delivery Network. Master Thesis, University of Twente, Netherlands, 2014.

[15] K. R. Jayaram, C. Peng, Z. Zhang, M. Kim, H. Chen, and H. Lei: An empirical analysis of similarity in virtual machine images. In *Proceedings of the Middleware 2011 Industry Track Workshop*, p. 6., ACM, 2011.

[16] C. Peng, , M. Kim, Z. Zhang, and H. Lei: VDN: Virtual machine image distribution network for cloud data centers. In INFOCOM, 2012 Proceedings IEEE, pp. 181-189. IEEE, 2012.

[17] L. Zeng, S. Xu, and Y. Wang: VMBackup: an efficient framework for online virtual machine image backup and recovery. Concurrency and Computation: Practice and Experience, 2015.

[18] M. Schmidt, N. Fallenbeck, M. Smith, and B. Freisleben: Efficient distribution of virtual machines for cloud computing. In *Parallel, Distributed and Network-Based Processing (PDP)*, 2010 18th Euromicro International Conference on, pp. 567-574. IEEE, 2010.

[19] K. Razavi, and T. Kielmann: Scalable virtual machine deployment using VM image caches. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 65. ACM, 2013.

[20] E. Prud'hommeaux, J. E. Labra Gayo, and H. Solbrig: Shape expressions: an RDF validation and transformation language. In Proceedings of the *10th International Conference on Semantic Systems (SEM '14)*, ACM, New York, NY, USA, pp. 32-40, 2014. [21] J. Heflin: *An Introduction to the OWL Web Ontology Language*.

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

[23] M. Voigt, A. Mitschick, and J. Schulz: Yet Another Triple Store Benchmark? Practical Experiences with Real-World Data. In *Proceedings of the 2nd International Workshop on Semantic Digital Archives (SDA)*, Cyprus, 2012.

[24] I. Diakonikolas: Approximation of Multiobjective Optimization Problems. PHD thesis, Columbia University, US, 2011.