Multi-objective optimization framework
01/08/2016
dEcentralized repositories for traNsparent and efficienT vIrtual machine opErations

Multi-objective optimization framework

Responsible author(s): Dragi Kimovski and Radu Prodan

Co-author(s): Nishant Saurabh

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 644179.
Revision history

<table>
<thead>
<tr>
<th>Administration and summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym: ENTICE</td>
</tr>
<tr>
<td>Document identifier: ENTICE D4.1</td>
</tr>
<tr>
<td>Leading partner: UIBK</td>
</tr>
<tr>
<td>Report version: 0.1</td>
</tr>
<tr>
<td>Report preparation date: 01.08.2016</td>
</tr>
<tr>
<td>Classification: PP (Restricted to other programme participants)</td>
</tr>
<tr>
<td>Nature: Other (Internal Document)</td>
</tr>
<tr>
<td>Author(s) and contributors: Dragi Kimovski, Radu Prodan, Nishant Saurabh.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Status</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Plan</td>
</tr>
<tr>
<td>-</td>
<td>Draft</td>
</tr>
<tr>
<td>-</td>
<td>Working</td>
</tr>
<tr>
<td>x</td>
<td>Final</td>
</tr>
<tr>
<td>-</td>
<td>Submitted</td>
</tr>
<tr>
<td>-</td>
<td>Approved</td>
</tr>
</tbody>
</table>

The ENTICE Consortium has addressed all comments received, making changes as necessary. Changes to this document are detailed in the change log table below.

<table>
<thead>
<tr>
<th>Date</th>
<th>Edited by</th>
<th>Status</th>
<th>Changes made</th>
</tr>
</thead>
<tbody>
<tr>
<td>08.01.2016</td>
<td>Nishant Saurabh</td>
<td>Plan</td>
<td>Initial Outline</td>
</tr>
<tr>
<td>10.03.2016</td>
<td>Dragi Kimovski</td>
<td>Plan</td>
<td>Updated Outline</td>
</tr>
<tr>
<td>20.05.2016</td>
<td>Dragi Kimovski</td>
<td>Working</td>
<td>First draft prepared</td>
</tr>
<tr>
<td>15.07.2016</td>
<td>Dragi Kimovski</td>
<td>Final</td>
<td>Implemented reviewers comments</td>
</tr>
</tbody>
</table>

Notice that other documents may supersede this document. A list of latest public ENTICE deliverables can be found at the ENTICE Web page at http://www.entice-project.eu/.

Copyright

This document is © ENTICE Consortium 2015.

Citation

Acknowledgements
The work presented in this document has been conducted in the context of the EU Horizon 2020. ENTICE is a 36-month project that started on February 1st, 2015 and is funded by the European Commission.

The partners in the project are UNIVERSITAET INNSBRUCK (UBIK), MAGYAR TUDOMANYOS AKADEMIA SZAMITASTECHNIKAI ES AUTOMATIZALASI KUTATOINEZET (SZTAKI), UNIVERZA V LJUBLJANI (UL), Flexiant Limited (Flexiant Limited), WELLNESS TELECOM SL (WTELECOM) and Deimos CASTILLA LA MANCHA SL (Elecnor Deimos Satellite Systems). The content of this document is the result of extensive discussions within the ENTICE © Consortium as a whole.

More information
Public ENTICE reports and other information pertaining to the project are available through ENTICE public Web site under http://www.entice-project.eu.
2 MULTI-OBJECTIVE FRAMEWORK FOR FEDERATED REPOSITORY OPTIMISATION

2.1 OPTIMIZATION REQUIREMENTS
2.2 STAKEHOLDERS IDENTIFICATION
2.3 MULTI-OBJECTIVE FRAMEWORK DESCRIPTION
2.4 CONSTRAINTS
2.5 DECISION MAKING AND PARETO SLA

3 FRAMEWORK INTEGRATION AND APPLICATION LEVELS

3.1 INTEGRATION WITHIN THE ENTICE ENVIRONMENT
3.2 INITIAL VM IMAGE DISTRIBUTION
3.3 OFFLINE VM IMAGE REDISTRIBUTION

OBJECTIVE FUNCTIONS MODELLING:
SEARCH ALGORITHM AND DECISION MAKING:
3.4 ONLINE VM IMAGE REDISTRIBUTION
3.5 VM IMAGE DISCOVERY AND ASSEMBLY

4 API DESCRIPTION

5 PRELIMINARY EVALUATION
CONCLUSION ........................................................................................................ ERROR! BOOKMARK NOT DEFINED.

BIBLIOGRAPHY AND REFERENCES ........................................................................ 25

ABBREVIATIONS/ GLOSSARY ................................................................................ 26
List of figures
Figure 1. Top level view of the multi-objective optimization framework ........................................13
Figure 2. Top level view of the ENTICE architecture ........................................................................15
Figure 3. An example of a neighbouring sub-graph in a structure with 3 repository sites and 4 different cloud providers ........................................................................................................17
Figure 4. An example individual represented as a solution vector ......................................................19
Figure 5. Execution time Vs. Number of storage sites in case of initial distribution .........................22
Figure 6. Comparison of two Pareto fronts during redistribution with varying storage sites ............23
Figure 7. Execution time Vs. number of evaluations during offline redistribution .........................24
Figure 8. Execution time Vs. size of solution vector during offline redistribution .........................24

List of tables
Table 1. Comparison of the offline VMI redistribution module with "round robin" strategy ........24
1 INTRODUCTION
This deliverable encompasses the concepts pertaining to the development of the multi-objective optimization framework for VMI distribution in federated Cloud environment. 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.

The remainder of the deliverable is structured as follows. First, a brief description on the concepts pertaining to this research is provided, such as multi-objective optimization and heuristic generation based algorithms. Afterwards, the optimisation requirements are identified and the stakeholders are presented. Furthermore, a detailed description of the optimization framework is presented and the decision making policies are explained. Then, the framework’s integration and application levels are presented and discussed. Finally, the service based API is precisely described.

Multi-objective Optimization - 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 [2].

A multi-objective optimization problem usually involves a number of objective functions which have to be minimized or maximized. In the most generic form, the problem can be formulated as:

$$\min(f_1(x), f_2(x), ..., f_v(x))$$

Subject to \(x \in X\)

where \(v \geq 2\) is the number of conflicting objectives functions \(f_i\) that we want to minimize, while \(X\) is a nonempty feasible region enclosing the set of variable (decision) vectors \(x = (x_1, x_2, ... x_n)\).

The generic formulation of the multi-objective optimization is free from any constraints. However, this is hardly the case when real life optimization problems are being solved, which are typically constrained by some bounds. Constraints divide the search space into two
distinctive regions: feasible and infeasible. Logically, the focus of the optimization is in the feasible region and all Pareto optimal solutions must be located in this region.

The multi-objective optimization consists of three distinctive phases: problem modelling, optimization and lastly decision making. Each of these phases is of paramount importance for attaining the optimal set of feasible solutions. Before being able to take any decision on the optimal trade-off solution we should properly model the problem that we want to solve. Actually, describing an appropriate computational model for a specific optimization problem is as critical as the optimization task itself. The modelling usually involves proper formulation of the optimization problem with an appropriate bounding of the decision variables, exact modelling of the objective functions, and like in every real life problem setting the constraints for the set of feasible solutions.

The optimization or search algorithm is the core of the multi-objective optimization. During this stage all feasible solutions are generated by using some procedure and then are evaluated. Multiple different search strategies have been described in the literature, but for the given research work we focus on the population based heuristics, which have been thoroughly explained in the next sub-section. The optimization algorithm result in a set multiple feasible solutions. Every single solution in the set represents an optimal trade-off between the conflicting objectives. This means that there is no single optimal solution, provided by the multi-objective algorithm, but a set of trade-off solution [3].

The last step of the optimization involves the process of decision making. As all elements in the Pareto front are equally optimal, some strategy for selecting the most suitable solution for the given problem should be investigated. The entity which decides on the proper selection of the appropriate trade off solution is called Decision Maker (DM). The DM can be implemented as a fully automated entity, or specific tools can be integrated to allow for the decision making to be realized directly by a person.

2 MULTI-OBJECTIVE FRAMEWORK FOR FEDERATED REPOSITORY OPTIMISATION

2.1 Optimization requirements

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 introductory considerations, the following optimization requirements have been identified:

1. Initial VMI distribution;
2. Offline VMI redistribution;
3. Online VMI redistribution;
4. VMI discovery and assembly.

ENTICE has been envisioned to support VM image redistribution both offline, as well as online during application execution by proactively moving the most demanded VM image fragments close to the resources the application is currently running on. In case of online image delivery, ENTICE will automatically discover user demand patterns by analysing the metadata (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-offs.

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.

### 2.2 Stakeholders identification

For the use cases regarding the distributed VMI storage, which has been enclosed around the multi-objective optimization framework, we have classified the stakeholders in two major categories with different roles:

- **VMI Owner.** The Owner of a VMI is responsible for selecting its optimisation parameters, resulting storage location, storage cost, and for maintaining or updating the image to latest versions. After uploading a VMI to the ENTICE repository, the owner can chose to optimise the image by providing non-functional descriptions and setting the parameters for the optimised distribution of the VMI or its fragments. A Cloud Provider or Cloud Operator can move their own images to the ENTICE repository and thus become VMI owners. An Application developer and a Cloud Provider will need to be able to upload a VMI to the ENTICE repository if they are not already provided by the Cloud Provider (which is the typical case for our applications from DEIMOS and WT).

- **VMI User.** The stakeholders allowed to use a VMI are the second important category. Every VMI Owner has the VMI User rights included. The user is missing the functionalities, which can result in changes to the VMI, its optimisation or cost. For example, moving a VMI to a different location might change the resulting cost for the VMI Owner, and is therefore prohibited for the VMI User. Again, a Cloud Provider and a Cloud Operator might need to test and/or use the VMI for which they received user rights for (i.e. debugging tasks or support for users). A Cloud Application Provider might only get VMI User rights on applications developed by an Application Developer. Finally, an Application Developer might use VMI from other entities within their application structure and therefore, can have the VMI User role.

Additionally, for proper enactment of the offline VMI redistribution throughout a distributed repository environment we have also identified a distinctive stakeholder, so-called administrative entity, with a single role to apply the decision making policies for proper selection of the best trade-off solutions based on the decision making model.
In regards with the scope of the Multi-objective optimization framework the Stakeholders, VMI Owners and VMI Users, will be able to specify one or multiple objectives to optimise the VMI and fragments and VMI distribution and the resulting delivery. The objectives are from three types of conflicting metrics: QoS-related (performance), operational cost, and storage. The ENTICE environment will enable the opportunity to specify what objectives to optimise for, or select a trade-off point from a Pareto front. Each image will be delivered to the VMI User based on the optimised distribution. The VMI Owner and user are not direct stakeholders of this use case. The VMI Owner, VM user or both will use the different child sub-use cases affiliated to this functionality.

2.3 Multi-objective framework description
In this section a detailed description of the multi-objective optimization framework for VMI distribution in Federated Cloud repositories will be presented. The optimization framework can be applied on multiple distinctive application levels:

1. Initial VMI distribution;
2. Offline VM image redistribution;
3. Online VM image redistribution;
4. VM image discovery and assembly.

The framework is encompassed around unified multi-objective optimization module, which can be utilized for multiple different optimization purposes. Internally, the optimization module is branched in four distinctive sub-modules. Each of the sub-modules has been tailored specifically for a given task. The “Initial Distribution” sub-module covers the multi-criteria evaluation of the possible repository sites where the VMIs or associated data sets can be initially stored. Afterwards, the “Offline VMI Redistribution” sub-module encapsulates the optimization of the VM images and fragments distribution within the federated repository sites. By taking into account the VMIs usage patterns the algorithm is capable of providing multiple trade-off solutions, where each solution represents a possible mapping between the stored fragments/images and available repository sites. The “Online VM image Redistribution” sub-module aims on dynamical redistribution of particular VM images/fragments during particular application execution. This sub-module is only applied for user’s specific VM images and in accordance with the technicalities of the application that is being executed. The framework is dependent on the repository’s usage patterns to properly optimize the distribution of the VM images. To this aim a specific module is required to store information on the previous transfers within the federation and to provide the collected data in a proper format. The ontology based Knowledge Base provided by WP5 has been utilized for the purpose of the multi-objective optimization framework. The framework has been designed to acquire input data from the knowledge base, and also to return the output results there. Moreover, a specific monitoring
agent has been provided by WP5 for proper documentation of the data transfers. The monitoring tool itself can be realized in multiple different manners, and its exact structure will be elaborated in the related Deliverable.

Furthermore, the framework provides a service based API, through which the Decision Maker (DM) can access the list of optimal Pareto solutions in a guided manner, thus reducing the complexity of the VMI storage management process. The high level structure of the optimization framework has been presented on Figure 1. The proposed framework has been developed by leveraging the jMetal Multi-objective optimization library [1].
2.4 **Constrains**
The generic formulation of the multi-objective optimization is free from any constraints. However, this is hardly the case when real life optimization problems are being solved, which are typically constrained by some bounds. The ENTICE environment, like every real life problem, induces constrains on the optimization process. 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. To this aim, the KB is an essential tool that provides means for setting the constraints on the input data in advance, before the process of evaluation of the solutions, thus prompting efficient operation of the optimisation framework. 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 KB. The KB has been constructed in such a way that it provides only the relevant input data by parsing the appropriate constraints.

2.5 **Decision making and Pareto SLA**
Decision making on the alternatives discovered by the optimization algorithm requires an explicit model of the decision maker preferences. For the case of offline VMI redistribution the DM model will be developed by WP5 as a part of the reasoning module. On the other hand, due to the requirement for real time execution, the online VMI redistribution module should utilize automated DM entity. This entity will also be developed as an essential part of the WP5 reasoning module. The exact decision making strategies for every application level individually, have been presented in Section 3.

Furthermore, it is important to note, that the result of the optimization process will be presented as part of a novel Pareto SLA (P-SLA). The aforementioned decision making tool will support the optimisation by allowing to selection of the “best” Pareto point from the P-SLA that fulfils the user-specified deployment characteristics and the Cloud Provider's operational requirements.

3 **FRAMEWORK INTEGRATION AND APPLICATION LEVELS**

3.1 **Integration within the ENTICE environment**
The Multi-objective optimization framework has been integrated within the ENTICE environment as an independent module (Figure 2). The module provides service based API, which can be utilized for initiating the optimization within the described application levels. The framework is directly connected to the Knowledge base and reasoning module, from where it can acquire input data or store the output of the optimization. Indirectly, by using the
Knowledge base, all other ENTICE modules, such as the VMI distribution, can use the output data to properly distribute the images, either online or offline. In what follows the exact implementation of the separate sub-modules of the framework is presented.

![Diagram of ENTICE architecture](image)

**Figure 2. Top level view of the ENTICE architecture**

### 3.2 Initial VM image distribution

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

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

\[
   f(P) = B_r \\
   f(C) = C_{st} + C_{tr}
\]
where $B_r$ represents the maximal theoretical performance of the interconnections of the repository, while $C_{st}$ is the cost for storing data on the given repository and $C_{tr}$ is the cost for transfer.

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

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

### 3.3 Offline VM image redistribution

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

#### 3.3.1 Objective functions modelling

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 $C_{tr-new}$. The exact values of the financial expenses for data storage and transfers should be provisioned by all Cloud providers within the federation.

For each VM image the cost objective can be calculated by using the formula below:

$$ f(C) = C_{st} + C_{tr-new} $$
The performance model includes much more complex reasoning behind it. It is based on the VM image usage patterns and it requires proper monitoring tool for efficient execution. The raw theoretical throughput of the interconnecting structure within a Cloud federation does not properly describe the factual communication performance, as it is difficult to predict the actual route the packets may take to reach the destination and the load on the intermediate communication channels. Opportunely, it is possible to leverage the data from the 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 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.

![Diagram of a neighbouring sub-graph in a structure with 3 repository sites and 4 different cloud providers.](image)

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

Furthermore, it is possible to model an undirected weighted graph, where the vertices correspond to either a repository site or a computational Cloud instance and the edges of the
graph are represented by the “virtual” links. The weighted graph actually enclosed a union of multiple neighboring subgraphs, where each storage site vertex, as direct neighbour, is linked to all known computational cloud vertices. The weights of the edges in the graph are determined by leveraging the estimated average bandwidth $B_{rc_i}$ on the corresponding “virtual” links. The weights are calculated dynamically, based the VMI distribution that is being considered. To properly model the weight of the edges, we introduce weight function, which considers the total number of downloads of the VMI to all neighbours $G_{tv}$ and the number of downloads to particular Cloud neighbour $G_i$. The ratio of those two values is then multiplied with the estimated bandwidth of the particular “virtual” link to provide the final value of the edge’s weight. The structure of the neighbouring sub-graph has been represented on Figure 3.

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

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

### 3.3.2 Search Algorithm and Decision Making

The core of the offline VMI redistribution sub-module is constructed over the Non-dominated Sorting Genetic Algorithm II (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 corresponds to the index of the vector element. Respectively, all storage sites in the federation are also assigned with unique IDs that 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 subgraph is constructed and the appropriate objective functions are applied. Those values are
then grouped together and the median value is selected as the overall fitness of the given individual. An example of a single individual that correspond to a solution vector for mapping 9 VMI s to 3 storage repository sites in a given federation is presented on Figure 4.

![Figure 4. An example individual represented as a solution vector](image)

When all individuals in the initial population have been successfully evaluated, the proper mutation and crossover operators are applied to create the children population. Then, the parents and children populations are grouped together and sorted according to dominance. Afterwards, only the best solutions of the newly formed group is selected for the next iteration. This process is then repeated for a predefined number of iterations. The solutions which have been acquired after the last iteration are sorted based on the dominance. The non-dominated solutions are then presented to the administrative entity of the federation, which acts as a DM, and should select the most appropriate solution based on the pre-defined decision making policy.

Decision making on the alternatives discovered by the optimization algorithm requires an explicit model of the decision maker preferences. For the case of offline VMI redistribution the DM model will depend on the implementation of the federated infrastructure. As the offline image redistribution envelops federation wide distribution of the VMIs we envision that the DM will be an administration entity, which will implement the federation storage policy based on the decision making model.

### 3.4 Online VM image 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. In a sense it can be commenced, that the offline VM image storage optimization is coarse grained and infrequently applied on all VM images and datasets stored in the federated repository. Contrary, the online VM image optimization is fine grained, and it involves only a user’s specific VM images that are frequently deployed in a given time interval. From an algorithmic point of view, the cost and performance objectives are modeled in the same way. The only difference are the input parameters, which in the case of online VM image optimization only involves few VM images, and the output solution vector is also limited only to the images that need to be transferred to more optimal place. As this process only involves fraction of the images stored in the federated repository it can be performed more frequently with reduced computational penalty. As initially planed this module will be fully developed and described by month 24 of the project duration.

3.5 VM image discovery and assembly
The ENTICE environment was envisioned to support VM management templates (VMMT) that allow VMs to be assembled at runtime from particular fragments. The templates will be stand-alone VM images containing the necessary components to access repositories in the ENTICE environment. After a VM is instantiated from a VMMT, it will customise the contents of the instantiated VM with the VM fragments required to match the user’s functional requirements. The Multi-objective optimization framework will also be utilized for the purposes of online discovery and assembly of VM image fragments. For this application level the framework has been envisioned to employ strategies from the area of multi-objective decision making to propose the best combination for assembling multiple fragments into a single VMI. As initially planed this module will be fully developed and described by month 32 of the project duration.
4 API DESCRIPTION

In this subsection a description of the services provided by the Multi-objective optimization framework for VMI distribution in Federated Cloud repositories will be presented. The optimization services can be accessed through the UIBK infrastructure, by providing the appropriate API command.

Generally, the optimization framework can be applied on multiple distinctive application levels:

1. Initial VMI distribution;
2. Offline VM image redistribution;
3. Online VM image redistribution;
4. VM image discovery and assembly.

The optimization framework has been envisioned to acquire the input data and then return the output data in the Knowledge base, thus allowing streamlined integration. The input data has been provided in JSON format by utilizing the WP5 RESTful services. Furthermore, the output data follows the same format to store the resulting data in the Knowledge base.

Depending on which optimization layer needs to be accessed, the framework provides the following API:

- serverIPadress/StageIIMOE_main() - initial VMI distribution;
- serverIPadress/StageIIIMOE_main() - Offline VM image redistribution;
- serverIPadress/StageIIIMOE_main(Running VMI owner) - Online VM image redistribution;
- serverIPadress/StageIIIIMOE_main(VMI_ID) - VM image discovery and assembly.

5 PRELIMINARY EVALUATION

The proposed framework and two of its modules have been preliminary evaluated based on a synthetic set of benchmark data. As ENTICE optimization framework deals with the implementation of a combinatorial multi-objective problem in federated Cloud environment, we present experimental results that demonstrate the ability of the framework to provide an adequate VMI/fragment distribution across federated repositories.

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

To begin with, the scalability and computational performance of the initial VMI upload module have been evaluated by varying the number of repository sites in the federation from 10 up to 10000 sites. Figure 5 shows the correlation between the average execution time and the number of storage sites in the federation. It is evident that the module can be lightly scaled up to large sizes. For relatively small federations the module can be invoked at each VMI upload, as it requires only few milliseconds to be executed.

![Figure 5. Execution time Vs. Number of storage sites in case of initial distribution](image)

On the other hand, the VMI redistribution module encloses diverse operations that can affect its behaviour to a various degree. Due to the nature of the algorithm it is not adequate to evaluate its computational performance based on the number of repositories in the federation. Increasing the number of storage sites, influences on the number of possibilities where to store a single VMI image, which translates into reduced quality of the proposed solutions, but relatively constant execution time. For example, on Figure 6 a scenario in which the vector size
(number of fragments) and number of evaluations have been kept constant, while the number of available repositories has been increased from 10 (blue) to 100 (red), is presented. The Pareto fronts from both executions have been plotted together to show the difference in quality of the final solutions. The experimental scenario clearly shows that if we increase the number of storage sites, while maintaining constant number of evaluations, the quality of the solutions will decrease.

![Diagram showing Pareto fronts comparison with varying storage sites.](image)

Figure 6. Comparison of two Pareto fronts during redistribution with varying storage sites

Furthermore, on Figure 7 and Figure 8, respectively, the influence that the number of evaluations and the size of the solution vector have on the computational performance is presented. In both cases, the number of associated cloud computing instances and storage sites were maintained constant; only the corresponding parameters were increased gradually. The presented results support the assumption of satisfactory scalability, both in a sense of increased number of stored VMI s and number of iterations needed to provide mapping solutions with good quality.
Lastly, Table 1 provides a comprehensive review of the quality values for the trade-off mapping solutions calculated by the redistribution module. Moreover, a comparison has been presented with a set of mapping solutions determined by using "round robin" mapping model for storing VMIs in the federation. The statistical significance of the results has been analyzed 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 10Gbit and 1Gbit Ethernet [4]. For readability reasons, the bandwidth values, were converted to delivery time needed for 1Mbit of data to be transferred from the source to the destination.

Table 1. Comparison of the offline VMI redistribution module with "round robin" strategy

<table>
<thead>
<tr>
<th>Evaluations</th>
<th>Average Cost STD (±) Difference (%) p-value</th>
<th>Average Performance STD (±) Difference (%) p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>0.00003237 0.00000005 0.49133799 &gt;0.005</td>
<td>0.00005356 0.00000272 18.48472759 &gt;0.005</td>
</tr>
<tr>
<td>20000</td>
<td>0.00003262 0.00000005 0.83933772 &gt;0.005</td>
<td>0.00004732 0.00000287 34.10821694 &gt;0.005</td>
</tr>
<tr>
<td>30000</td>
<td>0.00003251 0.00000005 1.16811082 &gt;0.005</td>
<td>0.00004109 0.00000316 54.42734792 &gt;0.005</td>
</tr>
<tr>
<td>40000</td>
<td>0.00003247 0.00000005 1.32053992 &gt;0.005</td>
<td>0.00003793 0.00000263 67.29142667 &gt;0.005</td>
</tr>
<tr>
<td>50000</td>
<td>0.00003240 0.00000005 1.52613233 &gt;0.005</td>
<td>0.00003526 0.00000314 79.98110629 &gt;0.005</td>
</tr>
<tr>
<td>60000</td>
<td>0.00003237 0.00000005 1.62638639 &gt;0.005</td>
<td>0.00003281 0.00000259 93.39303541 &gt;0.005</td>
</tr>
<tr>
<td>70000</td>
<td>0.00003230 0.00000005 1.84912539 &gt;0.005</td>
<td>0.00003119 0.00000269 103.44799654 &gt;0.005</td>
</tr>
<tr>
<td>80000</td>
<td>0.00003225 0.00000005 2.00020742 &gt;0.005</td>
<td>0.00002905 0.00000226 118.47484440 &gt;0.005</td>
</tr>
<tr>
<td>90000</td>
<td>0.00003224 0.00000005 2.03453036 &gt;0.005</td>
<td>0.00002686 0.00000178 136.23449371 &gt;0.005</td>
</tr>
<tr>
<td>100000</td>
<td>0.00003220 0.00000005 2.17047745 &gt;0.005</td>
<td>0.00002605 0.00000168 143.62117311 &gt;0.005</td>
</tr>
</tbody>
</table>
6 CONCLUSION AND FUTURE DIRECTIONS

In this deliverable the ENTICE multi-objective optimization framework for distribution of VMIs/fragments, as an essential storage resources, across distributed repositories has been described. The research and development work conducted during the first 18 months of the project timeline has resulted in development of a optimization framework that exploits multiple different factors, such as communication performance requirements, VMI use patterns, and structure of images, in order to optimize the distribution and placement of VMI’s fragments across distributed repositories and to significantly lower their provisioning time for complex resource requests and for executing the user applications. Two different modules of the framework have been developed and evaluated based on synthetic simulation benchmark. As the project work deals with the implementation of a combinatorial multi-objective problem, where the main incentive is to find the proper mapping of VMIs/fragments across storage sites, we present experimental results that demonstrate the ability of the proposed approach to provide very satisfactory VMI distribution across federated repositories.

As described in the project objectives for months 24 and 32, novel heuristic algorithms will be implemented to further improve the performance and quality of the redistribution process. Furthermore, lightweight optimization algorithms will be utilized for performing time sensitive fine-grained optimization of the distribution of the fragments and the associated data-sets during application execution.

7 BIBLIOGRAPHY AND REFERENCES


8 ABBREVIATIONS/ GLOSSARY

VMI – Virtual Machine Image

VMMT – Virtual Machine Management Template

MO – Multi-objective optimization

DM - Decision Making

API – Application Programming Interface