Abstract

One of the key problems in migrating multi-component enterprise applications to Clouds is selecting the best mix of VM images and Cloud infrastructure services. A migration process has to ensure that Quality of Service (QoS) requirements are met, while satisfying conflicting selection criteria, e.g., throughput and cost. When selecting Cloud services, application engineers must consider heterogeneous sets of criteria and complex dependencies across multiple layers impossible to resolve manually. To overcome this challenge, we present the generic recommender framework CloudGenius and an implementation that leverage well known multi-criteria decision making technique Analytic Hierarchy Process to automate the selection process based on a model, factors, and QoS requirements related to enterprise applications. In particular, we introduce a structured migration process for multi-component enterprise applications, clearly identify the most important criteria relevant to the selection problem and present a multi-criteria-based selection algorithm. Experiments with the software prototype CumulusGenius show time complexities.
1 Introduction

The emergence of Cloud computing over the past five years is potentially one of the breakthrough advances in the history of computing. Cloud computing paradigm is shifting computing from physical hardware- and locally managed software-enabled platforms to virtualized Cloud-hosted services. Cloud computing assembles large networks of virtualized services: hardware services (compute services, storage, and network) and infrastructure services (e.g., web server, databases, message queuing systems, monitoring systems, etc.). Cloud providers including Amazon Web Services (AWS), Microsoft Azure, Rackspace, GoGrid, and others give users the option to deploy their application over a pool of virtually infinite resources with practically no capital investment and with modest operating cost proportional to the actual use. Elasticity, cost benefits, and abundance of resources motivate many organizations to migrate their enterprise applications to Clouds.

There are two application engineering layers for compute services: a) software resource, where an engineer builds applications using APIs provided by the Cloud. A software resource (also referred to as an appliance or VM image) is a pre-configured, virtualization-enabled, self-contained, and pre-built Virtual Machine (VM) image that can be integrated with other compatible VM images for architecting complex applications. Major providers at this layer include thecloudmarket.com, 3Tera Applogic, and BitNami; and b) Infrastructure as a Service (IaaS) (hardware resources), where an engineer runs software applications on compute services, using the APIs provided to leverage other infrastructure services. A VM instance is essentially a piece of virtualization software (e.g. Xen, KVM, etc.) running on physical Cloud servers. It is the most common method of exposing the computational power (e.g. CPU cores, physical memory, storage capacity, etc.) to software applications. Amazon EC2, GoGrid, and Rackspace are among the major providers of virtualized hardware resources as services.

Problem Statement. Enterprise applications (e.g. customer relationship management, employee payroll, and supply chain management) can typically be decomposed into three software resource layers: i) front-end web servers to handle end-user requests and application presentation; ii) business logic to perform specialized application logic; and iii) back-end database servers. The flow of requests between these layers is often complex. Each layer may instantiate multiple software resources; each software resource may need to be replicated on multiple compute resources, while load-balancers distribute requests across each instance of VM images. This creates an enterprise application consisting of multiple components: an IT system formation. Optimal application QoS demands bespoke configuration both at software and IaaS layer, yet no detailed, comprehensive cost, performance or feature comparison of Cloud services exists. The key problem in mapping applications in form of multi-component IT system formations to Cloud resources is selecting the best size and mix of software and hardware resources to ensure that application QoS targets are met, while satisfying conflicting selection criteria related to software (e.g. virtualization
format, operating system, etc.) and hardware (e.g., maximizing throughput, minimizing cost) resources. For instance, before mapping a Bitnami’s Web server appliance [2] to a Amazon EC2 virtual machine instance resource, one needs to consider whether they are compatible in terms of virtualization format (e.g., VMWare, AMI, etc.) and other system-level constraints (e.g., Unix or Windows operating system). Figure 1 depicts the selection problem of migrating multi-component enterprise applications, IT system formations, to Cloud infrastructure services.

Proposed Approach. To counter the above complexities for migrating multi-component enterprise applications, we propose a novel, flexible, and automated decision-making framework, CloudGenius, that translates Cloud service selection steps into multi-criteria decision-making problems to determine the most applicable VM images (at software resource layer) and compatible compute services (at IaaS layer). CloudGenius provides a framework that guides through a Cloud migration process and offers a model and methods to determine the best combined and compatible choice of VM images and services. The framework leverages the evaluation and decision-making framework \( (MC^2)^2 \) [20] for supporting multi-criteria-based selection. The \( (MC^2)^2 \) framework provides a process depicted in Figure 2 allowing the creation of an evaluation method that comprises a requirements check and evaluates multiple alternatives on an absolute \((0 - 1)\) scale. Within \( (MC^2)^2 \) process our approach proposes the Analytic Hierarchy Process (AHP) as the multi-criteria decision-making method of choice. AHP allows for administrator trade-offs and compensation between multiple criteria and, hence, give alternatives with a low value regarding one criterion the chance to retain within the set of feasible solutions. Compensations are influenced by the weighting which is derived from the trade-offs users make in pair-wise comparisons of criteria.

The paper is structured as follows. First, we reflect related work in Section

Figure 1: Overview of the Selection Problem
and, afterwards, we present the CloudGenius framework in Section 4. The framework introduces a migration process and formal model which forms the basis for decision support within the process. Further, we present CumulusGenius, a prototypical implementation of the framework in Section 4. In Section 5 we present the results of experiments on the time complexity of CumulusGenius before we discuss future work and conclude in Section 6.

2 Related Work

Multiple approaches have been introduced by the Web service community that define multi-component Web services [13][25] but do not address the characteristics of the Cloud. Existing work in the Cloud context provides provider or service evaluation methods but lacks multi-component support [8]. Multiple approaches for multi-component setups in the Cloud have applied optimization [21][22][10][12] and performance measurement techniques [19] for selecting hardware resources (provider side) or Cloud infrastructure services (client side) according to quantitative criteria (throughput, availability, cost, reputation, etc.). While doing so, they have largely ignored the need for VM images, a migration process with transparent decision support and adaptability to custom criteria, and, hence, lack flexibility.

Additionally, there is preliminary work that provides decision support for selecting VM images and infrastructure services. Dastjerdi et al. [9] propose an approach that selects Cloud VM images and Cloud infrastructure services with an ontology-based requirements check but lacks a service evaluation. Khajeh-Hosseini et al. [15][16] developed the Cloud Adoption Toolkit that offers a
high-level decision support for IT system migration of enterprises. The focus of the decision support is on risk management and a cost model which incorporates expected workload on the IT system.

3 CloudGenius Framework

A migration of an IT system formation to Cloud infrastructures is complex and demands the choice of adequate Cloud infrastructure services and Cloud VM images for every component within the formation. We propose CloudGenius, a framework that guides through a Cloud migration process that provides methods that support multi-criteria-based decisions on selecting a Cloud VM images and Cloud infrastructure services component-wise. In the following subsections we present the process and give details on the formal model of the selection problem, the required user input and flexibilities, and the selection and combination steps that choose an image and service from the abundance of offerings and find the best combination. Finally, an alternative evaluation variant is addressed.

3.1 Multi-Component Cloud Migration Process

Figure 3 depicts CloudGenius’ migration process for IT system formations in Business Process Model and Notation (BPMN) 2.0. The process is divided in two lanes: (1) ”user input” lane with domain experts such as application engineers providing input and (2) ”CloudGenius” lane where steps are completed by an implementation of the framework. The process allows for a loop enabling a component-wise migration and cycles for step-wise, incremental improvements of every component’s migration. Within the cycles engineers have to define requirements and preferences and CloudGenius applies the \((MC^2)^2\) decision-making framework to recommend a ranked VM image and Cloud service combinations for a certain component.

3.2 Formal Model of CloudGenius

A formal mathematical model is introduced to formalize the problem addressed by the CloudGenius framework. The model defines all parameters involved in the problem which the evaluation is based on. Table II summarizes parameters of the model.

The model of CloudGenius consists of \(l\) components which are part of a formation, \(m\) images \(a_i\), \(n\) services \(s_j\) and \(o\) providers \(p_k\). \(C\) is the corresponding set of software components reflecting the formation, \(A\) the set of VM images, \(S\) the set of Cloud infrastructure services and \(P\) the set of Cloud providers. \(a_i\) and \(s_j\) own numerical, measurable and non-numerical attributes of the sets \(\hat{A}_{a_i}, \hat{A}_{s_j}\), \(\hat{B}_{a_i}\) and \(\hat{B}_{s_j}\). \(\chi\) represents a value connected with a numerical attribute \(\alpha\) or non-numerical attribute \(\beta\). Furthermore, the model introduces \(r\) requirements and a goal/criteria hierarchy including \(g\) goals and \(c\) leaf criteria.
Figure 3: Multi-Component Migration Process of the CloudGenius Framework
| Parameter | Description |
|-----------|-------------|
| l         | number of software components |
| m         | number of Cloud VM images |
| n         | number of Cloud infrastructure services |
| o         | number of Cloud providers |
| r         | number of formulated Requirements |
| C         | set of software components |
| I         | set of relations between components |
| A         | set of Cloud VM images |
| S         | set of Cloud infrastructure services |
| P         | set of Cloud providers |
| R_A       | set of Cloud image requirements |
| R_S       | set of Cloud service requirements |
| D         | set of image-service compatibilities |
| E         | set of inter-image compatibilities |
| F         | set of inter-service compatibilities |
| \(\hat{A}\)_{a_i} | set of numerical attributes of i-th image |
| \(\hat{A}\)_{s_j} | set of numerical attributes of j-th service |
| \(\hat{B}\)_{a_i} | set of non-numerical attributes of i-th image |
| \(\hat{B}\)_{s_j} | set of non-numerical attributes of j-th service |
| c_h       | h-th software component to be migrated to the Cloud |
| a_i       | i-th image in the set of m images |
| s_j       | j-th service in the set of n services |
| p_k       | k-th provider in the set of o providers |
| d_l       | l-th dependency \((a_i, s_j)\) in the set \(D\) |
| \(\tau\)  | Cloud image or service \(\tau \in A \cup S\) |
| \(\chi(\alpha)\) | Value of numerical attribute \(\alpha\) in CloudGenius database |
| \(\chi(\beta)\) | Value of non-numerical attribute \(\beta\) in CloudGenius database |
| \(\alpha_{h,\tau}\) | h-th numerical attribute of i-th \(\tau\) |
| \(\beta_{g,\tau}\) | g-th non-numerical attribute of i-th \(\tau\) |
| \(v_{\tau_{i}}\) | value of i-th \(\tau\) calculated with \((MC^2)^2\) |

Based on the model CloudGenius determines the best combination \((a_i, s_j)\) where \(a_i\) and \(s_j\) are the image and infrastructure service of provider \(p_k\) that have the highest value of all combinations according to the user’s preferences and with \(a_i\) deployable on \(s_j\).

An evaluation method built with \((MC^2)^2\) can be employed to determine the best image \(a_i \in A\) and service \(s_j \in S\). Therefore, \((MC^2)^2\) is interpreted as \(f(a_i, \hat{A}_{a_i}, \hat{B}_{a_i}) \mapsto v_{a_i}\) which allows to find
\[
max\{v_{a_1}, ..., v_{a_m}\}
\]
and a function \(f(s_j, \hat{A}_{s_j}, \hat{B}_{s_j}) \mapsto v_{s_j}\) upon which \(max\{v_{s_1}, ..., v_{s_n}\}\) can be determined. The results can be merged to the function \(f(a_i, s_j) \mapsto v_{(a_i, s_j)}\) in order to determine a combined value.

Only feasible combinations of an image and service combination must be
considered, meaning an image has to be deployable on a service. The feasibility of a combination is indicated by the set $D$ which holds all compatibility dependencies between images and services. The feasibility of a multi-component formation is defined by set $E$ defining VM image compatibilities and set $F$ defining service compatibilities. $E$ and $F$ hold pairs of VM images resp. Cloud services that are compatible.

### 3.3 IT System Formation Model

In an initial step, stakeholders of the migration process, typically application engineers, have to define an IT system formation in the model. All components of the formation setup must be added as component items $c$ to set $C$ and interconnections must be defined in form of component pairs in set $I$. Further, a software feature must be assigned that categorizes the component. The set of available features comprises Web Server, Application Server, ERP system and CRM system and limits the set of plausible VM images. Next, the CloudGenius migration process supports engineers in the component-wise realization of the formation.

### 3.4 Software Component Requirements & Preferences

Within CloudGenius’ process, after a software component has been chosen, domain experts resp. engineers have to formulate requirements and preferences. Requirements formulation comprises choosing an attribute to set a constraint on and specifying a minimum or maximum value for numerical values, and a set of allowed items for non-numerical values. Table 2 gives an overview of four requirement types aligned with the $(MC^2)^2$ framework which uses conjuctive/disjunctive satisficing methods. The table assumes $\chi$ to be the attribute value under consideration, $v_r$ to be the given numerical requirement value, $s$ a given non-numerical value and $S$ a given set of non-numerical values. The applied requirements check are formulated in boolean expressions. Attributes available for requirements definitions can be drawn from attributes of Cloud VM images and Cloud infrastructure services listed in Table 3, 4, 5 and 6. Requirements set upon attributes of VM images belong to set $R_A$, requirements for services to $R_S$.

Stating preferences is carried out by selecting and weighting given goals and criteria. CloudGenius proposes hierarchies of goals and criteria and asks
engineers to select and weight the items in pair-wise comparisons (in analogy to AHP). Figure 4 and Figure 5 depict the proposed hierarchies of goals and criteria offered by CloudGenius. Goals are high level and group criteria while criteria are associated with attributes to be evaluated.

![Figure 4: CloudGenius VM Image Goal Hierarchy](image)

![Figure 5: CloudGenius Infrastructure Service Hierarchy](image)

Since the \((MC^2)^2\) evaluation framework would support more complex goal hierarchies CloudGenius is not limited in this aspect. However, a higher complexity of the goal hierarchy increases the amount of input a user must provide and opposes CloudGenius’ goal of being lightweight. Possible additional goals are QoS proposed by Kalepu et al. [14] such as reliability, accessibility or sup-
port.

Finally, domain experts have to state the importance of a Cloud VM image’s and a Cloud infrastructure service’s value within a combined value in a pair-wise comparison.

### 3.5 Cloud VM Image Selection

VM images can be characterized by functional and non-functional attributes with numerical or non-numerical values assigned stating technical details or qualities. Table 3 gives an overview of the measurable, numerical attributes the CloudGenius framework proposes. The table lists their influence direction, metrics and the range of possible values. A list of non-numerical attributes is presented in Table 4. The variability of an attribute represents the ability to change the value over the lifetime of a VM image, e.g., the popularity of an image might dynamically change over time and its operating system is static. In Table 4 all non-numerical attributes are static. Numerical attributes, additionally, have either a negative or positive influence on the value the higher their value. For instance, the higher the license costs the less interesting becomes an image.

| Name                  | Influence | Variability | Metric | Value Range    |
|-----------------------|-----------|-------------|--------|----------------|
| Hourly License Price  | Negative  | Dynamic     | $/h    | 0-$/h          |
| Popularity            | Positive  | Dynamic     | %      | 0-100%         |
| Age                   | Positive  | Dynamic     | Days   | 0-∞ Days       |
| OS Version            | None      | Static      | Version| 0-∞            |
| Software Version      | None      | Static      | Version| 0-∞            |

| Name                 | Example Values                                    |
|----------------------|---------------------------------------------------|
| Virtualization Format| Xen, VMWare, ...                                  |
| Operating System (OS)| Linux, Windows, ...                               |
| Implementation Language| Java, Perl, Ruby, ...                            |
| Software Feature     | Web Server, Application Server, ...               |
| Software             | Apache HTTP, JBoss Appl. Server, ...              |

The selection of attributes is drawn from own observations and literature on VM images and services [9][13], providing a basic set of attributes essential to Cloud VM image selection. Service attributes have a limited applicability to VM images being only the software resource. Therefore, we hope to be able to build a list of important attributes from usage data of a publicly available prototype[5][4] and VM image databases such as thecloudmarket.com[6] or BitNami[2].

The search for a proper VM image is hugely influenced by the categorization of a VM image, the software feature, that must match a components feature.
CloudGenius supports generally all sorts of software features but we limit our focus on enterprise application systems and provide general attributes for those types of systems. Support for more or more detailed features for specific enterprise application systems requires feature-specific attributes.

Evaluation of VM images requires an evaluation method to be built with \((MC^2)^2\) based on domain experts’ input and CloudGenius’ formal model. In \((MC^2)^2\) all images from a database which holds the set of images including all image attributes become the evaluation alternatives. Further, all influential (influence \(\neq\) none) numerical attributes of VM images (see Table 3) become leaf criteria in the customized goal and criteria hierarchy weighted earlier by engineers (see Section 3.4). CloudGenius defines all requirements set on image attributes and defines AHP to be the multi-criteria decision-making method.

With \((MC^2)^2\) an evaluation method is created that matches a function \(f(a_i, \hat{A}_{a_i}, \hat{B}_{a_i}) \mapsto v_{a_i}\), which is applied on all images \(a_i\). By setting all \(\chi(a)\) as parameters in the function a value \(v_{a_i}\) and image \(a_i\) can be determined with \(v_{a_i} = 0\) whenever requirements are violated. In detail, \((MC^2)^2\) creates the function described in Equation 1 where \(w_j\) is the global criteria weight of j-th positive and normalized numerical attribute with value \(\chi(\alpha_j,a_i,+)\). \(f(a_i, \hat{A}_{a_i}, \hat{B}_{a_i})\) reflects the multiplicative index of AHP that divides positive goals by negative ones.

\[
\begin{align*}
\forall r \in R_A & : r = \text{true} \mapsto v_{a_i} \\
0 & \text{ else}
\end{align*}
\]

In case, none of the images meets all requirements in a subsequent step all images that meet all but one requirement in \(R_A\) are considered. This procedure repeats until a non-empty set of images is found that fulfills a smaller number of requirements.

To find the best image \(\max\{v_{a_1}, ..., v_{a_m}\}\) is to rank all images by the resulting values comparable on an absolute \([0, 1]\) scale.

### 3.6 Cloud Infrastructure Service Selection

In parallel to a VM image selection, for the selection of a Cloud compute service, such as Amazon EC2 or Joyent Public Cloud, CloudGenius proposes a range of numerical and non-numerical attributes and leverages the \((MC^2)^2\) framework to gain an evaluation method \(g(s_j, \hat{A}_{s_j}, \hat{B}_{s_j}) \mapsto v_{s_j}\) formulated in Equation 2.

\[
\begin{align*}
\forall r \in R_S & : r = \text{true} \mapsto v_{s_j} \\
0 & \text{ else}
\end{align*}
\]
A list of numerical attributes for the Cloud service selection evaluation method is given in Table 5 (all dynamic), non-numerical attributes are given in Table 6 (all static).

| Name                  | Influence | Metric   | Value Range     |
|-----------------------|-----------|----------|-----------------|
| Hourly CPU Price      | Negative  | $/h      | 0-$\infty $/h   |
| Network Send Price    | Negative  | $/B      | 0-$\infty $/Byte |
| Network Receive Price | Negative  | $/B      | 0-$\infty $/Byte |
| Internet Send Price   | Negative  | $/h      | 0-$\infty $/h   |
| Internet Receive Price| Negative  | $/B      | 0-$\infty $/Byte |
| CPU Performance       | Positive  | Flops    | 0-$\infty$ Flops |
| CPU Cores             | Positive  | Cores    | 0-$\infty$ Cores |
| RAM Performance       | Positive  | Flops    | 0-$\infty$ Flops |
| RAM Size              | Positive  | Bit      | 0-$\infty$ Bit  |
| Disk Performance      | Positive  | Flops    | 0-$\infty$ Flops |
| Disk Size             | Positive  | Bit      | 0-$\infty$ Bit  |
| Max. Latency          | Negative  | ms       | 0-$\infty$ ms   |
| Avg. Latency          | Negative  | ms       | 0-$\infty$ ms   |
| Uptime                | Positive  | %        | 0-100 %         |
| Service Popularity    | Positive  | %        | 0-100 %         |

| Name                  | Example Values |
|-----------------------|-----------------|
| Provider              | Amazon, Rackspace, ... |
| Location Country      | Germany, Australia, ... |

Cloud infrastructure services own multiple numerical attributes that imply a measurement or benchmarking. Hourly Usage, Network and Internet Traffic Prices are typically provided from the provider or must be calculated with a provider’s price model or cost calculation schemes \cite{17,16}. For performance and latency attributes benchmarking tools are required as those are often not provided by Cloud providers \cite{18} in contrast to attributes such as CPU core or RAM size. The uptime of a service is a long-term experience that can be provided as guaranteed uptime by a provider or his SLA, or from user experiences. In parallel, service popularity can be gained from user experiences.

In parallel to VM image selection an evaluation method is created with $(MC^2)^2$. With all parameters set (VM images as alternatives, a criteria hierarchy built from attributes, requirements $R_S$ and criteria weights) the $(MC^2)^2$ process can be completed and a new Cloud service evaluation method based on AHP is created. The Cloud service evaluations can be retrieved from applying the new evaluation method $g(s_j, \hat{A}_{s_j}, \hat{B}_{s_j})$. The highest ranked Cloud infrastructure alternative is $\max\{v_{s_1}, ..., v_{s_n}\}$. 

12
3.7 Best Combination

In the final evaluation step, VM images and Cloud services are combined to solutions. Every VM image $a_i$ is combined with a service $s_j$ to create a possible solution pair $(a_i, s_j)$. The CloudGenius model holds the set $D$ of all compatibility dependencies between images and services where an existing dependency between image $a_i$ with service $s_j$ implies feasibility and, hence, a value $v_{a_i,s_j} > 0$.

Equation 3 formulates the function that maps solution pairs to a value using an operator $\bullet$. For more complex combination value computations a function $h(f(\cdot), g(\cdot))$ may return the overall value based on the image and service evaluation functions $f$ and $g$ instead of an operator $\bullet$ only.

$$f(a_i, s_j) = \begin{cases} f(a_i, \hat{A}_{a_i}, \hat{B}_{a_i}) \cdot g(s_j, \hat{A}_{s_j}, \hat{B}_{s_j}) & (a_i, s_j) \in D \\ 0 & \text{else} \end{cases} \quad \mapsto v_{a_i,s_j} \quad (3)$$

CloudGenius promotes the $+$ operator to sum up VM image and service value to a total value of a combination. The sum of both values allows for compensation between image and service selection where a low quality service and high quality image might have a lower total value than a combination of medium quality image and service. In contrast, $*$ or $\times$ operator helps finding VM image infrastructure service combinations with most balanced values.

Equation 4 gives the evaluation function with user defined weights $w_a$ and $w_s$ that sum up to 1 that describe the importance of the VM image and service within a combined solution. Services of a Cloud provider that are located at different locations might cause internet traffic costs billed by the provider.

Equation 5 shows the $\Delta$ of total network traffic costs (internet and local network) for a component $i \in C$ connected with other components according to $I$ where $I_i$ holds $i$’s relations. The $\Delta$ represents the extra costs for network traffic introduced by the combined solution added as component to the multi-component formation.

Let $c_{i,o,R_l}, c_{i,o,S_l}, c_{i,o,R_g}, c_{i,o,S_g}$ be expected cost of incoming and outgoing local network and internet traffic between components $i$ and $o$, given by domain experts. All $\Delta_{\text{network},i}$ should be normalized to $(0, 1)$ scale, e.g. using AHP with one criterion, to follow AHP multiplicative index propositions.

$$f(a_i, s_j) = \begin{cases} \frac{w_a f(a_i, \hat{A}_{a_i}, \hat{B}_{a_i}) + w_s g(s_j, \hat{A}_{s_j}, \hat{B}_{s_j})}{\Delta_{\text{network},c_h,\text{normalized}}} & (a_i, s_j) \in D, c_h \in C \\ 0 & \text{else} \end{cases} \quad \mapsto v_{a_i,s_j} \quad (4)$$

$$\Delta_{\text{network},i} = \sum_{o}^{I_i} \begin{cases} c_{i,o,R_l} + c_{i,o,S_l} & \text{provider and location equal} \\ c_{i,o,R_g} + c_{i,o,S_g} & \text{else} \end{cases} \quad (5)$$

The best overall solution pair to be added to the IT system formation has a value $\max\{v_{a_1,s_1}, ..., v_{a_l,s_l}\}$. In a multi-component setup, however, relations between component restrain the number of actually feasible combined solutions.
Thus, introduced constraints are defined by sets $E$ and $F$ allowing only multi-component setups where compatible VM images and services are composited. An added combined solution must be tested for its co-operation compatibility with a formation’s components it has relations with according to set $I$.

After automated decision support, the migration process continues with a selected combined solution, deployment of the VM image on the service and further customization and re-evaluation cycles by engineers. With all components deployed and customized an applied migration strategy results in the multi-component IT system available on Cloud infrastructure services.

### 3.8 Integrated Evaluation Approach

Alternatively, a best combination can be determined in a single $(MC^2)^2$ process that process uses a single criteria hierarchy depicted in Figure 6.

![Figure 6: CloudGenius Integrated Goal Hierarchy](image)

In this variant the compensatory relation between VM image and service are influenced by weighting of the criteria hierarchy. Due to the lack of transparency regarding partial results we do not promote this variant and leave it to an implementation to provide any or both evaluation options to a user.

### 3.9 Complexity of CloudGenius Approach

The whole problem addressed by CloudGenius seems rather complex and involves a number of calculations. Hence, analyzing the actual computational complexity of the approach becomes of interest to ensure its applicability. To our best knowledge we define $O$ of CloudGenius as following:

$$O(l \ast (m \ast |B_a| + n \ast |B_s| + \text{requirements check} \bigg) + m \ast |A_a| + n \ast |A_s| + \text{evaluation} \bigg) + 2 \ast (l - 1) \ast (l - 2)}$$
The computational complexity is mainly influenced by the number of VM images \( m \), the number of services \( n \) and the number of attributes resp. criteria. Part of the computations are requirements checks for all \( m \) images and \( n \) service regarding all requirements in \( \hat{B}_a \) and \( \hat{B}_s \). The evaluation adds computation steps for the VM image and service evaluation which depend on the number of images and services and the number of attributes. The defined \( O \) only counts the number of evaluation steps to sum up a total value for an image or service. In more detail, AHP introduces more steps to normalize matrices and derive global weights. For simplicity these steps are omitted. Computations required for combined solutions contain the filtering of infeasible solutions by making all possible \( m \times n \) combinations and comparing with \( D \). Finally, a combined value for all combinations is calculated. The complexity must be multiplied by the number of components \( l \) and network costs are added in a multi-component setup. The summed effort for all network cost calculations is doubled to reflect send and receive operations. The calculation effort is \( \frac{(l-1)(l-2)}{2} \) as there is no network costs calculation effort for the first step where a first migrated component is added to the formation.

4 CumulusGenius: An Implementation

With CumulusGenius [5] we present an implementation of the decision-making support of the framework. The CumulusGenius java library offers a data model that enables the evaluation of VM images, Cloud infrastructure services and best combinations programmatically. A Web frontend that supports the framework’s process and provides a database of VM images and Cloud services of the current Cloud provider landscape is currently under development [4].

5 Experiments

We tested our implementation CumulusGenius in experiments on a test machine with Intel Core i7 2.7 Ghz and 8 GB of RAM. The experiments allow analyzing the resulting time complexity of CumulusGenius. The parameters of the experiments are the number of VM images, service and components. VM images and services are synthetically generated with all attributes having random values. There is a fix number of three providers and no requirements are defined to keep a full search space of combined solutions. Components are randomly assigned to a provider and all inter-connected to one another. When components are assigned to the same provider low network costs occur, in case of different providers high internet costs are assumed. Figure 7 depicts the exponentially growing total time to find a solution for three components and Figure 8 the linearly growing computation time when adding components. Evaluation of combined solutions produces the major part of the computation time.
6 Conclusion & Future Work

In this paper, we presented the CloudGenius recommender framework that transforms Cloud service selection from manual time-consuming scripting to a process that is flexible, and to a large extent automated. It provides a migration process and helps enterprise application engineers to select best resource mix at both software and IaaS layers over provider boundaries. We believe that CloudGenius framework leaves space for a range of enhancements and, yet, provides an amicable approach. To our knowledge no existing approach has addressed the problem of inter-dependencies between software and IaaS layers while selecting software and hardware resources for Cloud-based engineering of enterprise applications.

A major issue in Cloud service selection is the domain of the search space (i.e. the completeness of VM images and Cloud services database), the criteria catalogs, and the quality and correctness of measured values. To address these issues, we will focus on integrating existing benchmarking services such as CloudHarmony [3] in the CloudGenius framework. Work related to automated benchmarking is already in progress [11]. A critical mass of data on VM images and IaaS level services might be gained by integrating existing databases such as thecloudmarket.com [6] or CloudHarmony [5]. Further, we aim at making data decision and user-specific, like e.g. latency measurements.

The proposed framework expects VM images to provide one feature instead of whole software stacks representing a whole formation or basic VM images only containing an OS. Future work should predict expected customization ef-
forts and consider it in decisions. Additionally, explicit support for hybrid setups, quality concerns, in particular, reliability (choice of multiple, different VM images per component and geographical locations), and middleware and persistence layer services is future work. Also, CloudGenius’ step-by-step approach depends on the component order within a migration what shall be overcome by global optimization in the future. Moreover, requirement checks might be extended with a feature model-based approach [23]. Regarding automated application deployment we want to provide a process based language that considers control and data flow dependencies and elastic behavior of individual appliances.

It is planned to drive an evaluation with industry partners of the Cumulus-Genius prototype providing a Web-frontend [4].

7 Acknowledgments

Initial research work on Cloud virtual machine image and infrastructure service data models was done when Dr. Rajiv Ranjan was employed at University of New South Wales on strategic eResearch grant scheme.

References

[1] Amazon Web Services Elastic Compute Cloud. http://aws.amazon.com/ec2 accessed 2011-11-21.

[2] BitNami Cloud Images, http://bitnami.org/learn_more/cloud_images accessed 2011-12-02.
[3] CloudHarmony. http://cloudharmony.com accessed 2011-10-19.

[4] CumulusGenius Online Prototype. http://cumulusgenius.appspot.com/ accessed 2011-11-06.

[5] CumulusGenius Prototype. http://code.google.com/p/cumulusgenius/ accessed 2011-11-06.

[6] The Cloud Market. http://cloudmarket.com accessed 2011-10-19.

[7] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al. Above the clouds: A berkeley view of cloud computing. EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[8] H. Chan and T. Chieu. Ranking and mapping of applications to cloud computing services by SVD. In Network Operations and Management Symposium Workshops (NOMS Wksp), 2010 IEEE/IFIP, pages 362–369. IEEE.

[9] A. Dastjerdi, S. Tabatabaei, and R. Buyya. An effective architecture for automated appliance management system applying ontology-based cloud discovery. In Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pages 104–112. IEEE Computer Society, 2010.

[10] H. Goudarzi and M. Pedram. Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems. Contract.

[11] S. Haak and M. Menzel. Autonomic benchmarking for cloud infrastructures: an economic optimization model. In Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics, pages 27–32. ACM, 2011.

[12] M. Hajjat, X. Sun, Y. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani. Cloudward bound: Planning for beneficial Migration of Enterprise Applications to the Cloud. ACM SIGCOMM Computer Communication Review, 40(4):243–254, 2010.

[13] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In Proceedings of the 14th Australasian database conference—Volume 17, pages 191–200. Australian Computer Society, Inc., 2003.

[14] S. Kalepu, S. Krishnaswamy, and S. Loke. Verity: a qos metric for selecting web services and providers. In Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on, pages 131–139. IEEE, 2003.

[15] A. Khajeh-Hosseini, D. Greenwood, J. Smith, and I. Sommerville. The Cloud Adoption Toolkit: Supporting Cloud Adoption Decisions in the Enterprise. Software: Practice and Experience, 2010.
[16] A. Khajeh-Hosseini, I. Sommerville, J. Bogaerts, and P. Teregowda. Decision support tools for cloud migration in the enterprise. Arxiv preprint arXiv:1105.0149, 2011.

[17] M. Klems, J. Nimis, and S. Tai. Do clouds compute? a framework for estimating the value of cloud computing. Designing E-Business Systems. Markets, Services, and Networks, pages 110–123, 2009.

[18] A. Lenk, M. Menzel, J. Lipsky, S. Tai, and P. Offermann. What are you paying for? performance benchmarking for infrastructure-as-a-service offerings. In Cloud Computing (CLOUD), 2011 IEEE International Conference on, pages 484–491. IEEE, 2011.

[19] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In Proceedings of the 10th annual conference on Internet measurement, pages 1–14. ACM, 2010.

[20] M. Menzel, M. Schönherr, J. Nimis, and S. Tai. (MC²)²: A Generic Decision-Making Framework and its Application to Cloud Computing. In Proceedings of the International Conference on Cloud Computing and Virtualization (CCV 2010), Singapore, Mai 2010. GSTF.

[21] M. Menzel, M. Schönherr, and S. Tai. (MC²)²: Criteria, Requirements and a Software Prototype for Cloud Infrastructure Decisions. Software: Practice and Experience, 2011.

[22] H. Wada, J. Suzuki, Y. Yamano, and K. Oba. Evolutionary deployment optimization for service-oriented clouds. Software: Practice and Experience, 41(5):469–493, 2011.

[23] E. Wittern and C. Zirpins. On the use of feature models for service design: the case of value representation. In Towards a Service-Based Internet. ServiceWave 2010 Workshops, pages 110–118. Springer, 2011.

[24] Z. Ye, X. Zhou, and A. Bouguettaya. Genetic algorithm based qos-aware service compositions in cloud computing. In Database Systems for Advanced Applications, pages 321–334. Springer, 2011.

[25] Z. Zhang, F. Hong, and H. Xiao. A colored petri net-based model for web service composition. Journal of Shanghai University (English Edition), 12(4):323–329, 2008.