A Cost-effective Approach for Hybrid Migration to the Cloud

Doaa M. Shawky
Engineering Mathematics Department
Faculty of Engineering, Cairo University
Giza, Egypt
doaashawky@staff.cu.edu.eg

Abstract—**Cloud computing is the latest computing paradigm that delivers hardware and software resources as virtualized services. To take full advantages of cloud services, there is a need to move legacy software systems to the cloud. Migrating legacy applications to the cloud is a non-trivial task as it leads to new technical challenges. The main problem in mapping software applications to cloud services is selecting the best and most compatible software components to ensure a cost-effective model. When selecting components to migrate to the cloud, software engineers must consider many criteria and complex dependencies among other systems' components. Thus, a technique for locating components to be migrated without actually moving them is needed. To overcome these challenges, we propose an approach which can be used in the decision-making process based on a set of measurable factors in the pricing models of cloud providers. In the presented approach, coupling among different components of the system is measured. Then, a proposed cost measuring function is used to choose the optimal migration scenarios. The approach is applied to a real enterprise resource planning (ERP) system. Experimental results show the efficiency, applicability and easy adaptability of the presented approach.**

Keywords- Cloud Computing, Coupling, Hybrid Migration.

I. INTRODUCTION

Cloud computing is the delivery of computing resources on demand with reduced management effort. It delivers infrastructure, platform and software as services. These services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) respectively [1-4]. In IaaS, a service (e.g., Amazon web services) is provided to the user. The application interface accesses the virtual servers and storage hosted by the cloud. PaaS in the cloud is a set of application or software which is hosted in the cloud. The users execute the application in the platforms hosted by the cloud provider through the platform or application program interface. Finally, SaaS model provides both hardware and software infrastructure to the user.

Using clouds, users are able to access and deploy applications from anywhere in the world at competitive costs depending on users QoS (Quality of Service) requirements. In addition, by using clouds, IT companies remove the low level task of setting up basic hardware and software infrastructures and thus, they can focus on innovation and creation of business values [1, 2]. Moreover, clouds are considered a cheap alternative to supercomputers and specialized clusters, and a more reliable platform than grids. Another benefit of using clouds is their ability to scale up immediately and temporarily according to users' demands [3].

Cloud computing systems have some essential characteristics [1, 4]:

- On-demand self-service. A user can provision computing capabilities, such as server time and network storage as needed and, sometimes, automatically without requiring human interaction.
- Resource pooling. The provider's computing resources (storage, processing power, memory and network bandwidth) are pooled to serve multiple users using a multi-tenant model. Different physical and virtual resources are dynamically assigned and reassigned according to users' demands.
- Rapid elasticity. Computing resources can be rapidly and elastically provisioned to quickly scale out and released to quickly scale in.
- Measured Service. Resource usage can be monitored, controlled, and reported providing transparency for both the provider and the user.

In order to leverage past investments as well as the benefits of cloud computing, there is a need for defining methods and techniques for migrating existing legacy systems taking into consideration the investments that have already been done. However, this is a non-trivial task as many challenges arise when migrating to the cloud. These challenges include designing migration plans, evaluating them and moving applications to a targeted cloud computing model. Migration to the cloud also requires experience in IT systems and cloud management, and a structured approach to program management [5]. Other challenges include security, regulations, and fear of vendor lock-in [6]. In addition, defining a migration strategy involves understanding and establishing business priorities, then evolving a strategy that offers a fine balance between costs and meeting business priorities.

There are two main approaches for migration to the cloud. The first approach is to move the whole application to the cloud. On the other hand, we may adopt a partial or hybrid migration. The former approach is likely to provide higher

response times. However, the latter is more suitable for large systems since it is not appropriate to move everything to the cloud [6, 7]. In hybrid migration, some parts of the application are moved to the cloud, while other parts are kept on premise based on their security or performance requirements. Adopting hybrid migration has several advantages. Firstly, the invested money and effort in legacy systems are not abandoned [8]. Secondly, building new systems from scratch would require a larger and relatively long-term investment, which carries more risk than adapting the legacy system step by step.

To help answer the question "Which of the feasible migration scenarios is optimal and how it can be determined?", we examine three critical and measurable characteristics that vary from one migration scenario to another. We propose a method that helps software engineers choose the optimal migration scenarios. The main idea is to choose the option which minimizes coupling between cloud-based and on-premise components. Using coupling, we can estimate a measure of the most important factors which are included in most cloud providers' billing systems. These factors measure the utilization of the provider's computing resources such as network bandwidth and processing power. By, focusing on these two utilities, we can estimate the running cost when adopting a hybrid migration scenario. An optimal migration candidate is then determined by minimizing a proposed cost function that quantifies these factors.

The paper is organized as follows. Section II provides an overview of the proposed method and presents some background about the related concepts. Section III presents the proposed approach for hybrid migration in detail. In Section IV, an experimental study is performed and its results are presented and discussed. Section V discusses the related research in the area. Finally, Section VI presents concluding remarks and outlines ideas for future work.

II. Overview

Cloud computing systems fall into one of five layers; applications, software environments, software infrastructure, software kernel, and hardware [4]. At the bottom of the cloud stack is the hardware layer which is the actual physical components of the system. At the top of the layers is the cloud application layer, which is the interface of the cloud to the users through web browsers and computing terminals. The ability of clouds to add or remove resources within few minutes allows matching resources to workload much more efficiently. Real estimates of server utilization in data centers range from 5% to 20% [9]. Employing elasticity allows reducing this resources waste. Moreover, it makes use of the economic benefits of the cloud by adopting the "pay-as-you-go" concept since hours purchased via cloud computing is usually distributed non-uniformly in time. Thus, a billing system is used to measure the virtualized services. In [10], a comparison of the resource pricing for Amazon AWS [11], Google App Engine [12], Windows Azure [13], Force.com [14], Rack space [15] and Go Grid [16] is presented. Factors which are included in most cloud providers billing systems are the transferred data between the cloud's components and the on-premise components (the outgoing and incoming bandwidths), the processing power utilization (PPU) or the compute utility, and the storage allocated by the client in GB. Other factors include the recipients' emails, additional public IP addresses, and RAM usage, etc. We think that, in different migration scenarios of software components, the amount of the transferred data and the PPU are among the most critical and measurable factors. This is because they contribute to the running cost when cloud solutions are adopted. Thus, if a hybrid method is to be followed, communication between migrated parts and other on-premise parts in addition to processing power utilization must be minimized.

Coupling is a measure of the degree of interaction between software components [17]. Many types of coupling have been identified, including data coupling, stamp coupling, control coupling, and common coupling [18]. A good software system should have low coupling among components, as systems with highly coupled components are usually more difficult to understand, maintain, and reuse [19]. In the proposed approach, a set of dependencies which represents different degrees of coupling among software components is used. A component depends on another if it includes, calls, sets, uses, casts, or refers to that component [20].

We propose a measure of utilization of network bandwidth and processing power using coupling among different components. The rationale behind using coupling is that if two components are tightly coupled and at the same time they are located apart, then it is expected that the amount of data transfer between them is large which results in high bandwidth utilization. On the other hand, if the tightly coupled components are located in the cloud, then the large amounts of data transfer between them result in high utilization of the provided processing power.

III. The Proposed Approach

The proposed approach is depicted in Fig. 1. Corresponding to the basic ideas behind the approach presented above, it consists of the following main steps:

1. Statically scan the implementation files of the program to determine the system's initial set of components (IS) and the references among components.
2. Run Perl scripts to generate the Dependency Table (DT).
3. Generate all possible migration alternatives.
4. For each migration scenario, calculate the corresponding cost function.
5. Choose the optimal migration scenarios

In the rest of this section, each step of the proposed approach is explained in detail. In the first step, as shown in Fig. 1, we statically scan implementation files using the static analysis and metrics generation tool Understand [21]. Thus, we can extract necessary information for our approach. This information includes a list of system's components and a list of dependencies among components.

In the second step, we run Perl scripts to create the DT. This table consists of the following attributes:

- The Component C_i, i=1, 2, .., N, where N is the total number of components in the analyzed system.

Depending on the type of the analyzed system, we can choose a component to be a class or a group of classes performing a certain function.

- The component C_j, j=1, 2, .., N, i≠ j which C_i depends on. It is denoted by Depends_On.
- The number of dependency relationships from C_i to C_j which is denoted by Depends_On_Strength.
- The component C_k, k=1, 2,.., N, k≠ i which depends on C_i. It is denoted by Depended_On_By.
- The number of dependency relationships from C_k to C_i which is denoted by Depended_On_By_Strength.

Thus, each tuple is of the form: C_i, C_j, a, C_k, b, where i, j, k=1, 2, .., N, and a, b are two positive integers indicating the number of dependency relationships from C_i to C_j and C_k to C_i respectively.

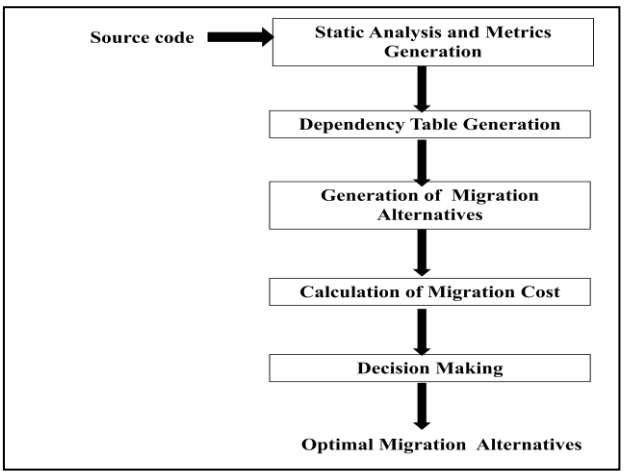

Figure 1. The basic steps of the proposed approach

In the third step, we generate all possible combinations of the systems' components. This step is implemented by using the function Combination $(k, C_1, C_2, .., C_N)$ which takes as inputs the IS and k=1, 2, .., N. The function produces as an output $\frac{N!}{k!(N-k)!}$ sets. Each set consists of k components corresponding to one of the k combinations of the C_i's. For example, if k=1, then Combination(k, C_1, C_2, …, C_N) returns the set {{C_1},{C_2},..,{C_N}}. Similarly, if k=2, then Combination(k, C_1, C_2, …, C_N) returns the set {{C_1,C_2},{C_1,C_3},..,{C_{N-1},C_N}}, etc. It should be mentioned that the total number of possible migration alternatives equals to $\sum_{k=1}^{N} \frac{N!}{k!(N-k)!}$. The pseudo code of the main algorithm is presented in Fig. 2.

In the fourth step, a cost function is defined to quantify the cost of each generated migration alternative. The proposed cost function can be easily adapted to correspond to any cloud provider's billing system such as Google's App Engine billing system [22], Amazon S3 [23] or Azure [24]. Moreover, the proposed cost function does not correspond to the actual cost imposed by the cloud provider, since the actual cost cannot be calculated unless the migration scenario is deployed and run. However, it can be used as a measure of the cost of different migration alternatives. Hence, a comparison among recommended scenarios can be done.

```
Main Algorithm:
Input: IS, where IS={Ci | i=1,2,..,N}, DT.
Output: Optimal Migration Alternatives (OMA), where OMA={{Ci, Cj,.., Ck} | i, j, k =1,2,..,N}
Inititialization:        Set of components to be migrated M={}.
On-premise components P=IS.
Cost_Function=Cost_of_Migration= Migration_Candidates ={}
Algorithm:
    For k=1 to N
      K_Candidate_Set= Combination(k, C1, C2, . . , CN).
          For l=1 to N!/(k!(N−k)!)
            M= K_Candidate_Set{l}
          Cost_of_Migration{l} =Calculate_Cost_Indicator(M).
          Append Cost_of_Migration{l} to Cost_Function{l}.
          Append M to Migration_Candidates{l}
        End for
    End for
```

Figure 2. The pseudo code of the main algorithm

Components to be migrated are denoted by C_i. On the other hand, on-premise components are denoted by C_j. The outgoing and incoming transferred data are denoted by OG and IC respectively. Thus, coupling between C_i (cloud-located components) and C_j (on-premise components) contributes to the outgoing data which are modeled by OG_{ij}. Whereas, coupling between C_j and C_i is modeled by IC_{ji} which contributes to the incoming data. If there is no dependency (coupling) between C_i and C_j, OG_{ij} and IC_{ji} are set to zero. In migration scenarios where the two components are both located in the cloud, the number of dependency relationships among them is added. Since in such cases, the dependencies contribute to the estimated PPU.

It should be noticed that not only the coupling strength between components is considered but also its direction. This is because we have to differentiate between the outgoing and the incoming transferred data, since for some cloud providers, e.g. Google App Engine, the cost of the amount of incoming data is different from that of the outgoing data. In a migration scenario, the set of components to be migrated to the cloud is denoted by M={ C_i | i=1,2,..,N} where |M| = m. Meanwhile, the set P=IS-M={C_j | j=1,2,..,N} where |P| = n and i ≠ j denotes the set of on-premise components.

Moreover, if we consider the case where database components are to be migrated, then another factor that accounts for the storage utility is to be added. This is reflected by the coefficient *d* as shown in (1) which must be multiplied by the size of the migrated database component. Hence, the proposed cost function is given by (1).

$$Cost = a*\sum_{\substack{i=1 \\ i\neq j}}^{m}\sum_{\substack{j=1 \\ j\neq i}}^{n} OG_{ij} + b*\sum_{\substack{j=1 \\ j\neq i}}^{n}\sum_{\substack{i=1 \\ i\neq j}}^{m} IC_{ji} + c*\sum_{i=1}^{m} PPU(C_i, C_j) + d*size(C_i) \quad (1)$$

Where a, b, c and d are real constants that represent the weight of each factor. PPU is estimated using (2).

$$PPU(C_i,C_j)=\begin{cases}\sum \text{Dependency relationships among } C_i \text{ and } C_j, & \text{if } C_i \text{ and } C_j \text{ are cloud} - \text{located} \\ 0 & \text{otherwise}\end{cases} \quad (2)$$

Fig. 3 presents the pseudo code of the proposed cost function for a certain migration scenario.

```
Global variables: IS, DT.
Function Calculate_Cost_Indicator(M)
Input: M={ Ci | i=1,2,..,N} where |M| = m
Output: A real number representing the cost of the migration alternative
Initialization: OG=IC=PPU=0, P=IS-M={Cj | j=1,2,..,N} where |P| = n and i≠ j
   For each Ci in M
         For each Ck in Ci.DT.Depends_On
         If Ck∉ M      OG=OG+ Ci.DT.Depends_On_Strength
                Else
                PPU=PPU+ Ci.DT.Depends_On_Strength
         End for
         For each Ck in Ci.DT.Depended_On_By
         If Ck∉ M     IC=IC+Ci.DT.Depended_On_By_Strength
         Else          PPU=PPU+ Depended_On_By_Strength
         End for
   End for

Cost= a*OG+b*IC+c* PPU+ d*Size(Ci)   // d=zero if Ci is not a database component.
```

Figure 3. The pseudo code of the proposed cost function

IV. An Experimental Study

An ERP system is used to evaluate the proposed approach. The system is used in a medium-sized higher education institute to manage and integrate work throughout the whole organization. It is a web application that is implemented using Visual C# with SQL Server. In addition, it adopts the three tier architecture. The main components of the system are student affairs, human resources and financial management. These components are implemented and managed using two front-end components denoted by C_{fe}, three business logic components denoted by C_{bl} and three database components denoted by C_{db}. The user can access and manage the collaboration among these components usually through C_{fe} components. In addition, she can access the C_{bl} directly. Moreover, some business-logic components can communicate directly. In order to model the ability of a user to access some components of the business-logic tier directly, we added a component denoted by C_{ui}. Fig. 4 shows a high-level overview of the used system where the arrows indicate that connected components can communicate. In addition, Table I provides some size metrics about the used system.

We considered the case where the database components are kept on-premise due to security and privacy concerns attached with moving them to the cloud. Hence, the components that will be included in the analysis are only the front-end and business-logic components.

Table II presents the total incoming and outgoing dependency relationships among the different components of the system. For instance the sum of the calls from C_{fe1} to C_{bl1} equals to 84. Meanwhile C_{bl1} calls C_{fe1} 52 times. It is worth pointing out that these values are calculated by simply adding the calls to and from the whole set of classes in each component. Also, notice that the empty cells above the diagonal mean that there is no direct relation between involved components. In addition, since there are no direct relationships among the last four components, we remove their corresponding rows from Table II.

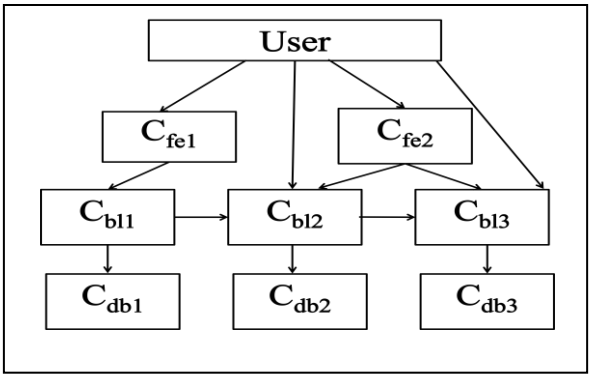

Figure 4. An overview of the used system

TABLE I. Some Size Metrics for the Used System

# of classes	3,125
# of Files	1,970
# of Methods	19,581
# of Lines	900,245
Size of C_{db1} (GB)	5.2
Size of C_{db2}(GB)	8.3
Size of C_{db3}(GB)	2.4

TABLE II. Sum of Outgoing/Incoming Dependency Relationships Among Different Components

C_i	C_{fe1}	C_{fe2}	C_{bl1}	C_{bl2}	C_{bl3}	C_{db1}	C_{db2}	C_{db3}	C_{ui}
C_{fe1}			84 /52						
C_{fe2}				68 /13	37 /20				
C_{bl1}				25 /16		158 / 49			
C_{bl2}					11 /42		214 / 38		13 /29
C_{bl3}								118 /45	16 /35

The suggested migration scenarios are implemented and deployed in the cloud using Microsoft Windows Azure as a platform. We used the Cloud Services option with two medium instances (one instance has 1.6 GHz CPU, 3.5 GB memory and 490 GB storage). The setup involves two data centers; one of them is the local data center and the other one is the cloud data center which is located in West Europe. Evaluations were based on the assumption that all requests are internal i.e., they

are initiated from within the organization. Each recommended migration scenario is deployed in the cloud and tested for a complete normal working day. In addition, real costs associated with each scenario are calculated.

The use cases correspond to the normal system usage which usually involves registration of a new student in the Students Database, enrolling students in a list of subjects, generating time tables, issuing a registration-fees receipt and generating salaries' sheets for the employees including professors and teaching assistants. Since the evaluation is based on Azure billing system where the compute, storage and outgoing bandwidth utilities are the most important measured factors, we set the values of a, b, c and d in (1) to 2, 0, 4 and 0 respectively. These values were chosen using Azure price calculator [25]. In addition, since we only consider the case where database components are kept on-premise, the value of d is set to zero. These values reflect the fact that PPU (compute utility) usually costs more than the bandwidth utility in Azure. It should be mentioned that the proposed approach can be adapted to reflect different pricing models. For example, if we use Google App Engine instead of Azure, the suggested suitable values for a: b: c can be chosen to form the ratios 0.12: 0.1: 0.1, where the value of d is set to zero for non-database components.

Table III presents the components involved in each of the best five migration scenarios together with their estimated costs and averaged real costs per month as imposed by Azure. Meanwhile, Fig. 5 presents a comparison between estimated and real costs for the best five recommended scenarios after normalizing the values given in Table III. As shown in the figure, real costs increase as estimated ones increase for the five scenarios except for the third one. This is may be due to the variations in the set of use case scenarios. Also, the best five suggested migration scenarios include the front-end components as they are loosely coupled with other components in comparison with business-logic components.

TABLE III. THE ESTIMATED AND THE ACTUAL COSTS OF THE BEST FIVE MIGRATION SCENARIOS

Migration Scenario	Estimated Cost	Real Cost ($/month)
C_{ui}	28.23	332.11
C_{fe1}	42.12	347.26
C_{fe2}	73.85	288.13
C_{ui} , C_{fe1}	88.22	417.52
C_{ui} , C_{fe2}	180.84	534.13

It should be mentioned that the suggested migration scenarios vary in the difficulty of the actual deployment to the cloud. For example, moving C_{ui} resulted in the refactoring and reconfiguring most of other components' classes. Thus, the exerted effort in the deployment of suggested scenarios needs to be estimated and added to the actual costs of a migration scenario.

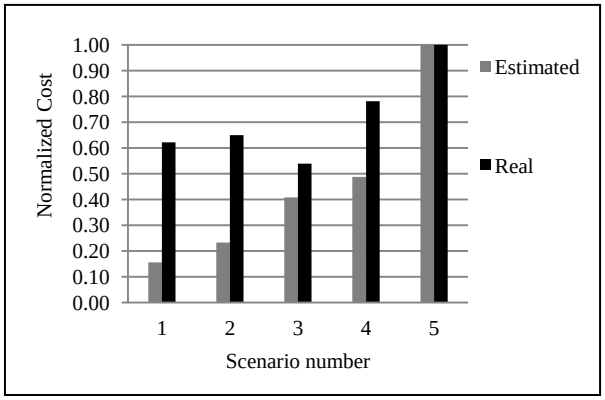

Figure 5. A comparison between estimated and real costs for the best five suggested migration scenarios.

V. RELATED RESEARCH

Cloud computing is an emerging research topic. Every month a large number of works that address related issues to this field appears. Many works addressed the problem of migration to the cloud. In [8], for example, the authors investigated the hybrid migration of architectures where part of the enterprise operation is hosted on-premise and the other part is in the cloud. The approach is based on optimization to identify application components to migrate to the cloud. The chosen components maximize the benefit taking into account enterprise-specific constraints, cost savings, and increased transaction delays and wide-area communication costs that may result from the migration. Evaluations based on real enterprise applications and Azure-based cloud deployments show the benefits of a hybrid migration approach, and the importance of planning which components to migrate. The authors conclude that hybrid migration can provide significant savings and lower delay time than moving the complete application to the cloud. In addition, they prove that the interaction among components has a strong effect on the migration decision.

Moreover, in [26], the authors presented a methodology and tools for model-driven migration of legacy applications to a service-oriented architecture with deployment in the cloud. They decomposed and decoupled the clients' architecture to take advantage of cloud computing services.

In addition, in [27], a study of the basic parameters for estimating the potential costs deriving from building and deploying applications on cloud and on-premise assets is presented. The authors defined a cloud migration framework to drive the new and existing applications to cloud platform. Venugopal et al. [28] presented a connection oriented framework for migration to multi-core cloud. This is accomplished using a set of tuning parameters for the web services to smoothly migrate the enterprise application to the cloud. In addition, in [29], a framework for cloud migration called the Cloud Motion Framework is presented. The framework evaluates alternative ways to host each component

based on the application model. The approach assumes that the components are independent and self-contained.

Also, in [30], a genetic-algorithm-based approach is proposed to compose services in cloud computing. A comparison is presented between the proposed approach and the random selection algorithms to prove its efficiency. In [31], another framework called CloudGenius is presented. CloudGenius is based on a compatible mix of software images to ensure that Quality of Service (QoS) targets of an application are achieved. Also, an automated decision-making process is presented. The framework employs the well known multi-criteria decision making technique, called Analytic Hierarchy Process, to automate the selection process based on the model and QoS parameters related to the application. Moreover, Tak et al. [32] studied the economics of moving to the cloud. They show that application characteristics such as workload intensity, growth rate, storage capacity and software licensing costs produce complex combined effect on overall costs and investigate costs of different deployment choices.

To the best of our knowledge, none of the existing techniques provide a framework for locating parts to be moved to the cloud statically i.e., by investigating the code and analyzing relationships among different components. Moreover, all of these techniques depend on a large number of assumptions in addition to stochastic analysis which make them more subject to errors.

VI. Conclusions And Future Work

In hybrid migration, software engineers face the problem of locating the optimal set of components to be migrated statically before actually moving them and calculating the benefit associated with each available migration scenario. This is even more difficult for large systems such as enterprise applications. In addition, if software engineers need to re-factor their applications into SaaS systems, the locations of refactoring to be applied must be determined. In this paper, an approach to tackle this problem is presented. The approach is based on measuring coupling among systems components and choosing the migration scenario which minimizes a proposed cost function. The cost function focuses on some measurable factors which are included in the billing systems of cloud providers. Also, the proposed approach is integrated into a decision support system which helps the software engineer make the right decision about candidate parts for migration.

We conclude that migration scenarios can be guided statically by measuring the degree of coupling among migration candidates. Experimental results emphasize the opinion that less coupled and more generic components are more suitable for migration. Adaptation of the presented approach can be easily done to represent different granularity levels, starting from components in component-based software (CBS) down to the finest possible level of the system to be migrated. For a CBS migration for example, metrics proposed for measuring coupling in CBS can be used (e.g., [33]).

The presented approach can be applied to any application whether it is a web application or not. However, web applications are more suitable to cloud migration than desktop applications. In comparison with other approaches, the presented approach is simple, easily adaptable, and less complex since we do not have to solve an optimization problem.

Currently, we are studying other factors that may affect the migration decision. For instance, performance issues such as reliability and availability not only cost-effectiveness issues should be considered. Also, security issues must be taken into account and integrated within the proposed approach for migration. In addition, the scenarios when database components are to be migrated are currently considered.

Future work includes the study of the effect of test case scenarios on the results of the proposed approach as this point constitutes a threat to validity. Moreover, to generate migration candidates, we apply exhaustive search which is a problem of exponential order. However, usually the number of software components is within the order of few tens for small-sized organization which makes the search process feasible in this case. This step must be modified if the number of considered components is high.

References

[1] M.A. Vouk, Cloud computing — Issues, research and implementations, in: Proceedings of the 30th International Conference on Technology Interfaces, pp.31-40, June, 2008.

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience (SPE). 41(2011), pp. 23-50.

[3] V. Chang, G. Wills, and D. De Roure, A Review of Cloud Business Models and Sustainability, in: Proceedings of the IEEE 3rd International Conference on Cloud Computing, pp 43-50, July, 2010.

[4] A. Iosup, S. Ostermann, M.N. Yigitbasi, R. Prodan, T.Fahringer and D.H.J. Epema, Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing, IEEE Transactions on Parallel and Distributed Systems, 22(6)(2011), pp.931-945.

[5] L. Badger, T. Grance, R. P.-Comer and J. Voas, Draft cloud computing synopsis and recommendations, National Institute of Standards and Technology, Special Publication 800-146, 2011.

[6] S. Mallya, Migrate Your Application to Cloud Practical Top 10 Checklist. (2010), available at:
http://www.prudentcloud.com/cloud-computing-technology/migration-to-cloud-top-10-checklist-24042010/

[7] C. Kothari and A. K. Arumugam, Cloud Application Migration. (2010), available at: http://cloudcomputing.sys-con.com/node/1458739

[8] M. Hajjat, X. Sun, Y. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, Cloudward bound: planning for beneficial migration of enterprise applications to the cloud. SIGCOMM Comput. Commun. Rev. 40(4) (2010), pp. 243-254.

[9] L. Youseff, M. Butrico, and D. Da Silva, Toward a Unified Ontology of Cloud Computing, in: Proceedings of Grid Computing Environments Workshop, pp.1-10, 2008.

[10] T. Harris, Comparison of cloud computing services. (2012)
Available at:
http://www.whitepapersdb.com/white-paper/9202/cloud-computing-services--a-comparison

[11] Amazon AWS, available at: http://aws.amazon.com/.

[12] Google App Engine, available at: https://appengine.google.com/

[13] Windows Azure, available at: www.windowsazure.com/

[14] Force.com, available at: http://www.force.com/.

[15] Rack space, available at: www.rackspace.com/.

[16] Go Grid, available at: www.gogrid.com/.

[17] E. E. Mills, Software Metrics, SEI Curriculum Module SEI-CM-12-1.1, Seattle University, 1988.

[18] C. Kaner and W.P. Bond, Software Engineering Metrics: What do they measure and how do we know? in: 10th Intl. Software Metrics Symposium, Sept, 2004.

[19] L. Yu and S. Ramaswamy, Categorization of Common Coupling in Kernel-Based Software, in: Proceedings of the 43rd ACM Southeast, pp. 207–210, 2005.

[20] Understand 3.0 User Guide and Reference Manual, available at: www.scitools.com/documents/manuals/pdf/understand.pdf.

[21] Understand, a static analysis tool, available at: www.scitools.com/

[22] Pricing of Google App Engine, available at: https://developers.google.com/appengine/docs/billing

[23] Pricing of Amazon EC2, available at: http://aws.amazon.com/s3/#pricing.

[24] Pricing of Windows Azure, available at: http://www.windowsazure.com/en- us/pricing/.

[25] Windows Azure Calculator, available at: http://www.windowsazure.com/en-us/pricing/calculator/

[26] P. Mohagheghi and T. Sæther, Software Engineering Challenges for Migration to the Service Cloud Paradigm: Ongoing Work in the REMICS Project, in: Proceedings of IEEE World Congress on Services, pp.507-514, July, 2011.

[27] S. Bibi, D. Katsaros, and P. Bozanis, Application Development: Fly to the Clouds or Stay In-house?, in: 19th IEEE International Workshop on Infrastructures for Collaborative Enterprises, pp.60-65, 2010.

[28] S.Venugopal, S. Desikan, K. Ganesan, Effective Migration of Enterprise Applications in Multicore Cloud, in: Proceedings of the 4th IEEE International Conference on Utility and Cloud Computing, pp.463-468, 2011.

[29] T. Binz, F. Leymann, D. Schumm, CMotion: A framework for migration of applications into and between clouds, in: Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications, pp.1-4, Dec. 2011.

[30] Z. Ye, X. Zhou, and A. Bouguettaya. Genetic algorithm based QoS-aware service compositions in cloud computing, in: proceedings of the 16th international conference on Database systems for advanced applications: Part II. Springer-Verlag, Berlin, Heidelberg, pp. 321-334, 2011.

[31] M. Menzel and R. Ranjan, CloudGenius: decision support for web server cloud migration, in: Proceedings of the 21st international conference on World Wide Web (WWW '12). ACM, USA,pp. 979-988, 2012.

[32] B. C. Tak, B. Urgaonkar, and A. Sivasubramaniam, To move or not to move: the economics of cloud computing, in: Proceedings of the 3rd USENIX conference on Hot topics in cloud computing USA, 2011.

[33] J. C. Yeap, W.K. Bruda, A Review of Component Coupling Metrics for Component-Based Development, World Congress on Software Engineering, pp.65-69, May, 2009.