Composite SaaS Placement and Resource Optimization in Cloud Computing using Evolutionary Algorithms

Zeratul Izzah Mohd Yusoh and Maolin Tang, *Senior Member, IEEE*

***Abstract*—Software as a Service (SaaS) is gaining more and more attention from software users and providers recently. This has raised many new challenges to SaaS providers in providing better SaaSes that suit everyone needs at minimum costs. One of the emerging approaches in tackling this challenge is by delivering the SaaS as a composite SaaS. Delivering it in such an approach has a number of benefits, including flexible offering of the SaaS functions and decreased cost of subscription for users. However, this approach also introduces new problems for SaaS resource management in a Cloud data centre. We present the problem of composite SaaS resource management in Cloud data centre, specifically on its initial placement and resource optimization problems aiming at improving the SaaS performance based on its execution time as well as minimizing the resource usage. Our approach differs from existing literature because it addresses the problems resulting from composite SaaS characteristics, where we focus on the SaaS requirements, constraints and interdependencies. The problems are tackled using evolutionary algorithms. Experimental results demonstrate the efficiency and the scalability of the proposed algorithms.**

***Index Terms*—Software as a Service, Evolutionary Algorithm, Placement, Optimization**

I. Introduction

Nowadays, demands for computation technologies or 'computer utilities' have increased, to the point where it has resulted in the transformation of the computation industry in the twenty-first century. As a result, Cloud computing [1] has emerged offering off-premis, high performance IT facilities including applications, data and computation resources to users. Cloud computing services can be categorized mainly into three categories: 1) Infrastructures as a Service (IaaS) offering computing infrastructure as a solution to user's computing and storage problems [1], 2) Platform as a Service (PaaS) which is targeted at application developers for their applications thereby allowing them to design, develop, deploy and test activities in Cloud platforms [2] and, 3) Software as a Service (SaaS), which offers an alternative for locally installed software [3].

Recently, SaaS is receiving a lot of attention from software providers as well as software users. The users' demand for SaaS is increasing each year [4] and Dubey & Wagle [5] reported that within three years, companies that have provided SaaS could generate up to 18 percent increase of revenue. Not only that, advances in Cloud computing has provided an efficient mean for SaaS hosting; and therefore making SaaS more accessible to a wide range of software users.

A SaaS can be delivered as a composite application, where the software is composed from a group of loosely-coupled individual applications that communicate with each other in order to form a higher-level functional system or application [6]. An example of composite SaaS application can be found in Satake [7] where he discusses about Fujitsu's approach to SaaS. Fujitsu is a provider of ICT-based business solutions for the global marketplace. It has branches in more than 70 countries, and headquartered in Japan. In their SaaS for Japan branches, they offer two types of SaaS, general and business SaaS. The former consist of common services including e-learning and customer relationship management services, while the latter is focusing on specific business services such as administrative task in medical care or procurement submission process. There are data components that are shared between the services. Apart from that, some of the services are interdependent between each other in order to provide a higher-level functionality of the SaaS. At the moment, the interdependency between those services is occurred within a same data centre site. However, in order to make the SaaS more efficient, Fujitsu aims for linking SaaS services that are distributed among multiple sites in the future.

Delivering the SaaS in such approach allows flexibility of the SaaS functionalities where components can be combined and recombined as needed. In addition, SaaS providers can gain a number of benefits including reduced delivery cost, flexible offers of the SaaS functions and decreased cost of subscription for users. However, this emerging approach also introduces new challenges for SaaS resource management in a Cloud data centre.

In the scenario considered in this paper, a Cloud data center provides the hosting infrastructure as well as the SaaS itself. A composite SaaS consists of a number of application components and data components where each component is deployed at Cloud physical servers and later executes insides virtual machines (VM). The SaaS resource management system is responsible for composite SaaS performance by managing Cloud resources through several phases including: 1) the initial placement of the composite SaaS onto Cloud physical servers to optimize its performance, and 2) the maintenance phase where the initial placement is reconfigured in order to optimize its resource usage.

For SaaS initial placement phase, the problem relates to how a composite SaaS should be placed onto a Cloud physical servers by the Cloud's providers such that its performance is optimal based on its estimated execution time. The placement of SaaS components and their related data components in the

978-0-7695-4755-8/12 $26.00 © 2012 IEEE
DOI 10.1109/CLOUD.2012.61

provider's servers that are located in geographically dispersed locations needs to be done strategically, as the placement can directly affect the resource usage as well as the SaaS performance. Existing placement method in Cloud focusing on placing the VM onto physical servers during runtime. This is assuming that each component deployed in a dedicated VM independently where the VM is created during runtime. However, this might not be the case for composite SaaS, where the placement has take into account the dependencies between the components. In addition, more than one component can be executed in a single VM. As such, to optimize the composite SaaS performance, the initial placement of the components will be done at physical servers, so that the creation of the VM for the component will be based on the placement made onto the physical servers.

For the composite SaaS's resource optimization problem, we assumed that the SaaS components are already running on their VMs. However, due to the dynamic environment of a Cloud data centre, where the workload of applications and resource capacities are keep changing over time, the current placement may need to be modified. A typical resource management has a maintenance window for reconfiguration of the initial placement in order to maintain the performance of the composite SaaS as well as to minimize the total resource usage. The maintenance process occurs at different time scales, from seconds to days, depending on the data centre's needs, as long as the SaaS performance is not affected by it. In this phase, the current application component placement will be reconfigured by clustering two or more application components into a VM. Existing techniques emphasize mostly on VM consolidation at virtualization level instead of component clustering at application level. This paper will tackle this problem at the application level.

The remaining paper is organized as follows. Section II discusses related work. The reference infrastructure is described in Section III. Section IV and Section V presents the problem formulation and the proposed algorithm respectively. Then Section VI is about the evaluation that has been carried out. The concluding remarks are presented in Section VII.

II. Related Work

This section discusses the existing work for SaaS components placement onto physical servers and SaaS resource optimization in virtual servers.

The problem of placing composite SaaS components onto Cloud physical servers shares similar characteristics with an existing problem called Component Placement Problem (CPP) [11]. CPP can be further divided into two categories: 1) offline CPP and, 2) online CPP, where in the online CPP the placement of the components is made during the runtime. Most studies on CPP are concerned with allocation of data centre's resources to application's components. This includes computation capacity [8], [9], [10], memory [8], [10], network bandwidth [11] and storage capacity [9]. The main aim for the CPP, usually, is to optimize the resource usage by the components, and at the same time to minimize the total execution time of the application [8], [9]. Zhu et al. [12] proposed a solution to an offline application placement problem, namely Application Component Placement (ACP). In ACP, the location of the data components is considered as one of the decision factors. ACP also considers the application's processing communication and storage requirements in making the decision. However, it is assumed that the location of the data in ACP is already known prior to the placement process. The data location is treated as an input to the placement problem, while in the composite SaaS placement problem data component are being placed together with SaaS application components. A placement that is concerned with SaaS is presented in [9]. The principal rule used by the placement approach is rather straightforward, that is, a new instance of a component should be deployed in a server with the smallest residual resource left after having the instance. This is to reserve servers that have larger residual resources for instances that have a higher resource demand. Although that work is concerned with SaaS placement, it is more focused on the multi-tenant resource model and does not take into consideration the SaaS's data component placement in the data centre.

To sum up, none of researches on CPP has considered the placement of the application's data together with the application's components. In the composite SaaS placement case, the data as well as the application components will be located at the Cloud provider's servers. As such, the Cloud providers must apply a strategic placement method in order to ensure the components and the data are well placed and the SaaS performance is optimized. This paper will propose a placement solution to address this gap.

Recently, virtualized resource management for Cloud data centre has been actively studied and large parts of the work fall into optimizing the resource management in the data centre. The common objectives for optimization including minimizing the resource usage while maintaining the applications' performances [13], [14], [15] and minimizing the data centre's power consumption [16], [17]. These objectives are achieved through various management plans at different level. For instance, at platform level, most existing works focusing on the management of VM mapping to physical servers, while at application level, the plan is to manage VM resources based on the application's workload.

Existing works on resource management at platform level apply migration of the VM as the main method in dealing with dynamic changes of Cloud environment [14], [15]. Most works at platform level consider a VM as an independent entity where it does not need to communicate to other VMs or storage servers in completing its task. This paper proposes a different approach that is concerned with the communication involved between VMs and it will be tackled at application level. The communication among VMs is highlighted in [13] where the authors proposed a solution for reconfiguration placement that supports three types of constraints which are the VMs demands, communications and availability. The data centre is modelled as a hierarchical structure that represents

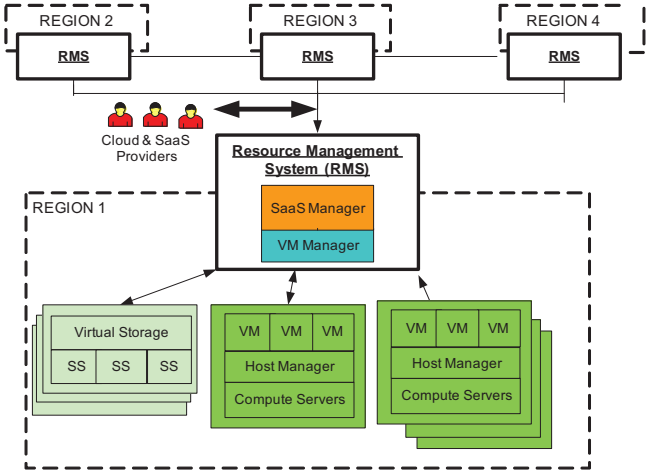

Figure 1. A high level Cloud architecture

communication cost based on its hierarchy. Another work that also concerned with communication among VM is presented in [16]. In the paper, they considered a multi-tier application where the deployment may span over multiple VMs. The proposed solution was designed at two levels. At application level, there is a controller that will dynamically assign resources to applications based on their requirement, and at platform level, they proposed a consolidation algorithm to re-map VMs to physical servers in the case of overload problem. The aim of the platform level is to optimize data centre's power usage. A similar work can be found in [17] where a multi-level solution was also proposed. The authors in this paper highlight the implementation of adaptive technique at application-level, where the application adapts automatically to the availability of the resources, and at resource-allocation level where the resources allocated adapts to the dynamic workload requirements. There is another level considered in this work, where the power consumption is adapted to the demands at resource-power level. Our work differs from all these solutions in the sense that they do not consider a composite application, in which a VM can host multiple components with different requirements. In addition, the components have to work with other component to achieve overall applications' functionalities that subjects to user's SLA.

III. Cloud Architecture & Resource Management System Model

Figure 1 shows a high-level architecture of a Cloud data centre under study. It is loosely based on architectures and models proposed in [18], [19], [20].

The figure depicts multiple data centres in different regions referring to their geographical locations. This is similar to commercial Cloud like Amazon which has data centres in the United States, Asia and Europe, and also Nirvanix which data centres are in the United States, Germany and Singapore [21]. In each region, there are a set of compute servers

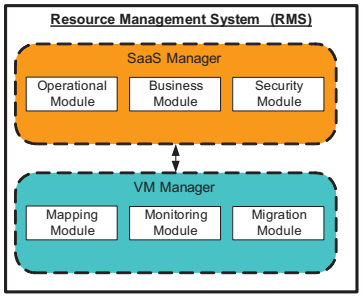

Figure 2. Main modules in RMS

(CS) which have their own resources including computation capacity, memory capacity and secondary storage capacity as well as storage servers (SS) with their capacity. The resources are then partitioned among multiple heterogenous virtual machines (VMs) each running some applications. This is referred as virtualization layer, where the VM is controlled by Host Managers. A host manager operates similar to Hypervisor and Dom0 in Xen virtualization architecture [19]. The host managers responsible for monitoring virtual machines performances and handling basic operational of the VMs including creating a VM, executing resource allocation policies, and executing migration of VMs. The host manager has certain interfaces that enable it to access the physical machines resources as well as receives commands from an upper level VM Manager.

At an upper level from the virtualization layer, there is a resource management system (RMS) that consists of a VM manager (VMM) and a SaaS Manager (SM). For the sake of simplicity, we consider a centralized RMS, however this architecture can easily extended to a decentralized RMS. The RMS may differ from one data centre to another, however the basic main modules are illustrated in Figure 2 [22], [20].

The two managers contained in the RMS is to provide clear separation of responsibilities between layers. The VMM is basically focusing on the management of VMs at virtualization layer, where the SM is responsible for tasks at application/service layer. This is also to hide low level details that occur in virtualization layer from application layer. The tasks for VMM include: 1) Mapping Module- mapping of VMs into CSs subject to inputs and constraint determined by the SM, 2) Monitoring module - monitoring the VM as a group based on inputs from the Host manager, and 3) Migration Module - the VMM is responsible for making decision from migrating the VM from one CS to another based on its need.

There are three modules in SaaS manager: operational, business and security modules. The operational module is responsible for all stages in SaaS cycle, from its initial placement onto CS, its optimization phase while the SaaS is running at VMs, up to the SaaS admission control and service request where this module will interpret request into Quality of Services (QoS) requirements and allocate the request to suitable SaaS. In addition, this module is also responsible for monitoring the SaaS performance based on its QoS. The

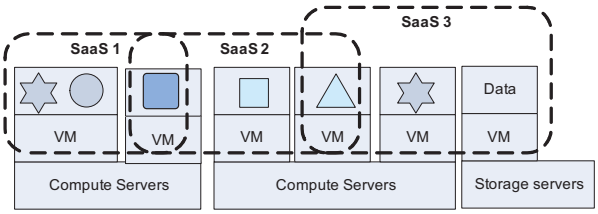

Figure 3. An illustration of a composite SaaS at Cloud data centre

business module is focusing on pricing and billing of the SaaS, and the security module concerns on application security for users. This paper focuses on the operational module of the SM particularly on two tasks: the initial placement and the optimization phase. These phases will receive inputs from other modules in SaaS manager as well as VM manager. The formulation of the problem will be presented in the next section.

IV. Problems Formulation

As mentioned in the in the Introduction, composite SaaS application and data components are placed in Cloud compute servers and storage servers during its initial phase, and later executes in virtual machines. Figure 3 shows a high level illustration of such a scenario, where the different shapes represent different application components of a composite SaaS, and a VM can host multiple components at a time. In order to deliver a higher level of functionality to users, a composite SaaS may span over multiple VM.

The composite SaaS placement problem and resource optimization problem use three main inputs: 1) A Cloud's data centre with its compute servers and storage servers. The compute servers may consist of at least one VM, 2) Cloud's data centre network topology with its links between compute servers and storage servers, and 3) Multiple composite SaaS with their resource requirements and constraints according to their SLA, and the current placement of the components in Cloud data centre. These inputs are obtained from other modules in the RMS.

A. Cloud Data Centre Formulation

A Cloud data centre consists of compute servers and storage servers. Each server has its own resource capacities including processing capacity, memory size, secondary storage capacity and storage capacity. Each compute server may consist of at least one virtual machine (VM), where the VM is given slices of the resources capacity of a compute server. A value is assigned to every VM which will represent as the 'cost' of the VM. Each resource type is given a value, and the VM cost is determined by the capacity of the resources that the VM has. Table 1 summarizes the data centre attributes.

B. Cloud Network Topology

The Cloud network is represented by an undirected graph $G = \langle V, E \rangle$, where $V = \{CS \cup SS\}$ is the sets of vertices including physical servers and storage servers, $e \in E$ is the set of undirected edges connecting the vertices. An edge $e = \langle v_i, v_j \rangle$ if and only if there exists a physical link transmitting information from v_i to v_j, where $v_i, v_j \in V$. $B_{v_i,v_j} : E \rightarrow \mathbb{R}^+$and $L_{v_i,v_j} : E \rightarrow \mathbb{R}^+$is a bandwidth and latency functions of the link from v_i to v_j respectively.

Table I
Sets and attributes of Cloud resources

Notation	Description
$cs_x \in CS$	The x^{th} compute server, cs_x, in CS, where CS is a set of k compute servers and $1 \leq x \leq k$
$ss_i \in SS$	The i^{th} storage server, ss_i, in SS, where SS is a set of r storage servers and $1 \leq i \leq r$
$vm_{x,y} \in VM$	The y^{th} virtual machine, vm, for cs_x and VM is a set of all virtual machine, $y \leq \mathbb{N}$
PC_z	Processing capacity for z where z can be either a CS or a VM
MC_z	Memory capacity for z where z can be either a CS or a VM
ST_z	Secondary storage for z where z can be either a CS or a VM
$C_{vm_{x,y}}$	Cost of $vm_{x,y}$
SM_{ss_i}	Storage capacity for ss_i

C. Composite SaaS Formulation

Each of the composite SaaS has its own application and data components with its requirements for resources. For the second problem, since the SaaS is considered already running, the SaaS will have its SLA which refers to the maximum response time of the SaaS. The SaaS modelling presented here is made general enough to represent a composite SaaS. Table 2 summarizes the SaaS components' requirements, and its workflows.

D. Problem's Constraints

In both problems, there are three types of constraints that need to be satisfied by the proposed algorithms. The constraints are:

1) Resource constraints: The total resource requirements for SaaS components that are placed in either compute servers/storage servers (in the intial placement problem) or virtual machines (in the resource optimization problem) must not exceed the machines' resources capacity.

2) Execution time constraints: In the initial placement problem, there is no constraints for the execution time as the SaaS has yet subscribed by users. However, to ensure the optimal performance of the SaaS, the placement of the components is based on its estimated total execution time. For the SaaS resource optimization problem, the execution time is considered as user's SLA and given as an input. In both problems, the total execution time is calculated based on four numerical attributes which are: a) the time taken for transferring data between the storage servers and the virtual machine, b) the processing time of a component in a selected

Table II
SETS, PARAMETERS AND REQUIREMENTS OF COMPOSITE SAAS

Notation	Description
$SC_i \subseteq S$	The i^{th} composite SaaS, SC_i in S. S is a set of n composite SaaS, SC , and $1 \leq i \leq n$
$ac_{i,j} \in AC$	The j^{th} application component, $ac_{i,j}$ for SC_i and AC is a set of all application component, $1 \leq j \leq z$
$dc_{i,q} \in DC$	The q^{th} data component, $dc_{i,q}$ for SC_i and DC is a set of all data component, $1 \leq q \leq x$
$wf_{i,p} \in WF$	A p^{th} business workflow for SC_i where $WF \subseteq AC$, $1 \leq p \leq y$
rt_{SC_i}	The maximum response time for SC_i
$TS_{ac_{i,j}}$	Task size of $ac_{i,j}$
$M_{ac_{i,j}}$	Memory requirement of $ac_{i,j}$
$SZ_{ac_{i,j}}$	Size of $ac_{i,j}$
$AD_{ac_{i,j}}$	Amount of read/write task of $ac_{i,j}$
$SZD_{dc_{i,q}}$	Size of $dc_{i,q}$
$W_{wf_{i,p}}$	Weighing for $wf_{i,p}$

virtual machine, c) the execution time of a path in the SaaS workflow, and d) the sum of the execution time of the critical path of each workflow multiplied by its weighting. These attributes have been defined in our previous work [24]. Based on these four values, the total execution time of the SaaS, $TET(SC_i)$, is determined.

For the second problem, the TET must not exceed the maximum response time of a SaaS as agreed in users' SLA, r_{sc_i}. This constraint is defined as below:

$$TET(SC_i) \leq r_{SC_i} \tag{1}$$

3) Sequence of migration constraints: : For the resource optimization problem, a current placement of SaaS application and data components are given as:

- A current placement configuration, P, of application components AC, onto virtual machines, VM:

$$P: AC \rightarrow VM, where\, ac_{i,j} \mapsto P(ac_{i,j}) = vm_{x,y} \tag{2}$$

- A current location, L, of the data components, DC, at storage servers, SS :

$$L: DC, \rightarrow SS\, where\, dc_{i,q} \mapsto L(dc_{i,q}) = ss_k \tag{3}$$

To optimize the resource usage, the solution will modify the current placement. Hence, the solution has to consider the sequence of components that need to be moved based on the current placement at that time. This sequence may affect the cost of changing placement directly. Two scenarios will be considered in this problem based on constraint presented in [15]:

- Sequential move: A particular component can only be moved when another one has been completed. This is in the case of where two components' migrations cannot be done in parallel because the destination VM contains another component that due to be migrated. As such, the latter component needs to be moved first to free some resources for the other component.
- Cyclic move: A set of components' migration may need an intermediate destination machine. This is in the case of when two or more components need to be exchanged places. This can create a cyclic constraint if the machines involved have insufficient resources.

E. Problems Objectives

Given the inputs and constraints as above, the objective of the problem are:

a) Composite SaaS initial placement problem: : To place the SaaS application components, AC and data component, DC onto Cloud servers such that the requirements are satisfied and the SaaS performance is optimal based on its estimated execution time, TET.

b) Composite SaaS resource optimization problem: To find a new placement of S onto VM by clustering the application components AC, such that the placement will minimize the resources' costs while satisfying the SaaS constraints. As component placement reconfiguration is an expensive process, the proposed solution will try to achieve the objective with a minimum number of changes to the current placement configuration.

V. THE PROPOSED SOLUTIONS

Both problems presented in this paper are categorized as combinatorial optimization problems in resource management automation. From computational point of view, the problems are NP complete. Hence, evolutionary algorithms (EAs), specifically genetic algorithms (GA) are proposed to tackle the problems. EA is an approach that imitates the natural evolution process. Over the years, EAs have become one of the well-established optimization techniques in various fields [23]. Several models have been developed for EA including genetic algorithm (GA), simulated annealing and genetic programming.

GA starts with a population of solutions that is generally randomly generated. The population then evolves according to the rules of selection and mutation referred to as genetic operators. The essence of the GA is the survival of the fittest solutions, which this is evaluated based on a fitness function of the problem.

We propose a Cooperative Coevolutionary GA (CCGA) for the SaaS initial placement problem and a Repair-based Grouping GA (RGGA) for the SaaS resource optimization problem.

A. CCGA for the SaaS Placement Problem

The population in a CCGA is divided into several subpopulations. The decomposition of the subpopulations is based on a divide-and-conquer strategy where all parts of the problem evolve separately. The fitness of a subpopulation is calculated on how well it 'cooperates' with the other subpopulations

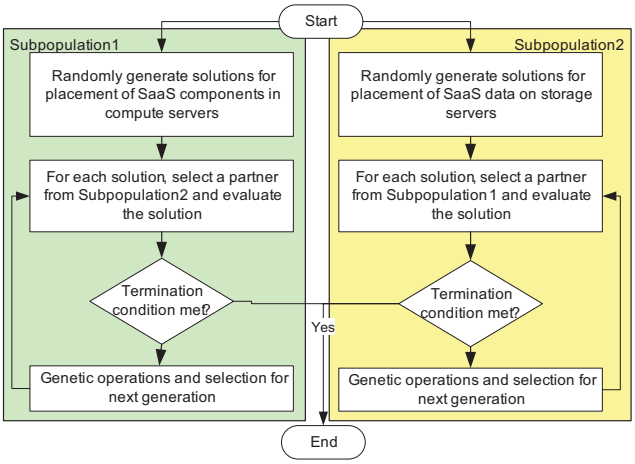

Figure 4. CCGA Flowchart

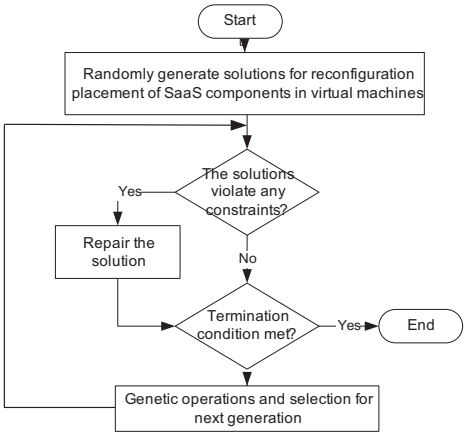

Figure 5. RGGA Flowchart

in order to produce a good solution. In the problem, we decompose the problem into two interacting subproblems: 1) the placement of SaaS components in compute servers, and 2) the placement of SaaS's data component in storage servers.

The solutions for these two optimisation problems evolve interchangeably and cooperatively in two separate subpopulations. The communication between subpopulations occurs during the evaluation of individual solutions. Solutions from one subpopulation are evaluated based on their performance when combined with solutions from the other subpopulation and vice versa. In order to calculate the fitness of an individual, a partner from the other subpopulation is selected and combined with the individual tp form a complete SaaS placement solution. This solution then is evaluated using the fitness function in Equation 8:

$$F(X) = \frac{TET(X)}{TET^*(P)} \tag{4}$$

where $TET(X)$ is the *Total Execution Time* for a solution X, and $TET^*(P)$ is the maximum value of *Total Execution Time* in the population, P .

The partner selection is based on the individual's fitness from the previous iteration of the algorithm. The fittest 50% individuals from each subpopulation are selected, and paired up randomly. A fitter individual represents a better placement solution. Figure 4 presents the algorithm. Details explanation on the algorithm is presented in [24].

B. RGGA for the SaaS Resource Optimization Problem

As the approach for this problem is to cluster components into VMs, Grouping Genetic Algorithms (GGA) suits naturally. GGA is a modified version of GA where it is designed for solving grouping problems. GGA divides its solutions based on relevant groups and optimization of fitness functions is done based on the grouping. As defined in Section 4.4, there are several constraints that the solution has to comply. All the solutions that do not comply with constraints will be repaired where a new value will be generated randomly to replace the invalid one. As for sequence of migration constraint, it will be incorporated in the fitness function. The fitness function is defined as follow:

$$F(X) \quad = \quad (F(TC) \times w_1) \quad + \quad (F(MC) \times w_2) \tag{5}$$

where TC refers to the total cost of VMs used by the SaaS in the new placement, and MC is the the migration cost from its initial to the new placement. The cost for each VM is given as an input (refer Table I). The migration cost is determined based on the size of the components and its memory requirement. The VMs cost for the initial placement is used as a benchmark to control the new cost, such that $F(TC)$ and $F(MC)$ will always be between 0 to 1. Figure 4 presents the flow of the algorithm. Further explanation of the algorithm can be found in [25]

VI. Evaluation

In this section, we discuss the experiments that have been done to the proposed algorithms. Both algorithms have been implemented using C++ programming language. The experiment evaluates the scalability of the proposed GAs as well as the quality of the solutions produced. For all experiments, we tested the algorithms on a large number of randomly generated Cloud network. The attributes of the nodes, including compute servers and storage servers, were randomly generated using the models presented in [26]. The composite SaaS is randomly created as well. We fixed the total number of SaaS and its component. All experiments were carried out in desktop computers with 3 GHz Intel Core 2 Duo CPU and 4GB RAM.

A. CCGA for the SaaS Placement Problem

In order to evaluate the proposed algorithm, we developed a classical GA (CGA). In the CGA, both of the subproblems are treated as one large problem where an individual contains two solution compartments that represent the placement solution for SaaS application components and data components. In order to get unbiased experimental results, parameters for size

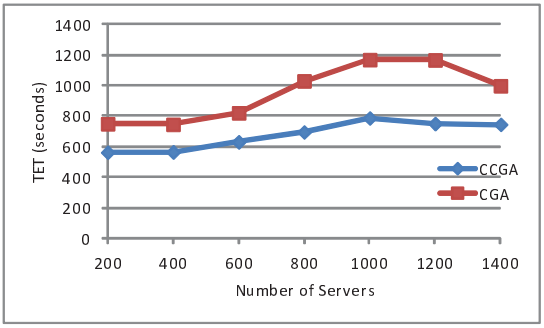

Figure 6. Comparison of the estimated TET obtained by the CCGA and the CGA

Figure 7. Comparison of computation time for the CCGA and the CGA

of population, selections and termination condition are same for both versions. The experiments were run for a Cloud that contained from 200 to 1400 compute servers and storage servers, with an increment of 200. The numbers of SaaS components and data chunks were both fixed at 10.

The experiment was to compare the quality of the solutions produced by the CCGA and the CGA. The comparison was based on the estimated total execution time of the SaaS, TET, that was calculated using Equation 4. Due to the stochastic nature of the algorithms, each of the test cases was repeated 20 times.

Figure 6 illustrates the average of the TET values of both the CCGA and the CGA. It can be seen that for all test cases, the CCGA has always a shorter TET, which implies it has a better placement solution for the application and data components of the composite SaaS. The average TET of the CCGA is only between 65% to 75% of that of the CGA.

Figure 7 shows the average computation times taken by the two algorithms. The CCGA has longer computation times than the CGA in all test cases that have been carried out. However, the longest time it took was less then eight minutes which is acceptable considering the placement plan is conducted in an offline mode. Apart from that, the result shows that the CCGA can scale well with the size of Cloud. Its computation time increased close to linearly when the number of Cloud servers increased. A sudden drop for the test case of 1000 servers is most probably because of the randomly generated Cloud server and network attributes.

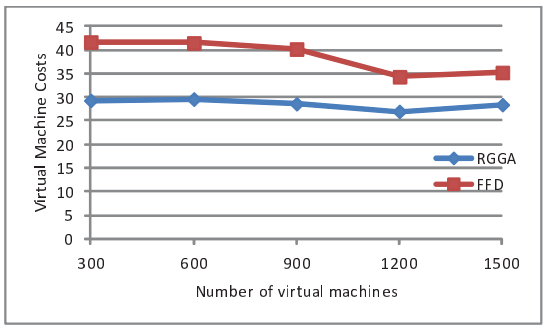

Figure 8. Comparison of the VM cost obtained by the RGGA and the FFD

B. RGGA for the SaaS Resource Optimization Problem

We tested the RGGA on a Cloud data centre that contained from 300 to 1500 virtual machines, with an incremental of 300. We fixed the total SaaS in the Cloud at three, with the total of 15 application components and 6 data components. Our approach is to deal with the dynamic environment of the Cloud at a static point of time, where a whole data centre will be considered.We also developed a First Fit Decreasing (FFD) heuristic for comparison. All the test cases were repeated for 20 times for both approaches.

Figure 8 shows the comparison between the RGGA and FFD in terms of VM costs. It can be seen that the proposed algorithm can find solutions with VM cost that is 20% to 30% less than the one proposed by FFD in every test cases, with a lower migration cost too. In addition to that, the solutions produced by the RGGA have also lower VM cost than the initial placement costs in all test cases. This shows that the proposed algorithm can succesfully reduced the cost of resources used without compromising the performance of the SaaS.

Figure 9 visualizes the average computation time for the RGGA and the FFD. Based on the result, RGGA spent about 1 to 2 minutes to find a solution while FFD took less than one seconds for every test cases. Although it is a big gap, it should be noted that in most cases, the solution produced by FFD has higher costs compared to the cost of the initial placement. The result also indicates that the proposed algorithm can come up with a feasible solution in certain duration of time regardless the size of the network, which can be used for the RMS in scheduling their maintenance's time.

VII. Conclusion & Future Work

We have presented the problem of composite SaaS in Cloud data centre, specifically on its initial placement and resource optimization problems. Both problems are formulated as combinatorial optimization problems aiming at improving the SaaS performance based on its execution time as well as minimizing the resource usage. This work differs from all existing research work as it addresses the problems resulting from composite SaaS characteristics, focusing on its requirements, constraints and interdependencies rather than from platform (virtualization/hardware) aspects.

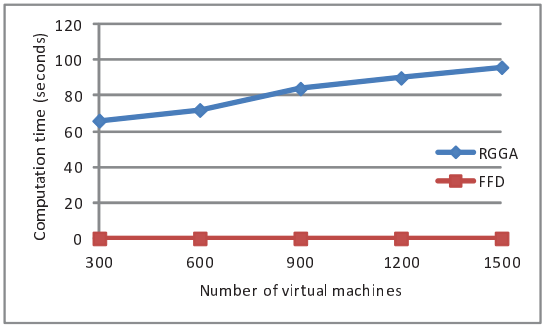

Figure 9. Comparison of computation time for the RGGA and the FFD

We tackled the two combinatorial optimization problems using two different paradigms of evolutionary algorithms. Based on the experimental results, the proposed algorithms always produce a feasible and satisfactory solution to all the test problems. Although the computation time taken is quite long, it is still acceptable considering that there are various types of maintenance in a data centre that are conducted at different time scales - in term of seconds to hours.

As for our future research work, first, we note that we can improve the implementation of the algorithms to reduce their computation time. The algorithms can be implemented in a parallel manner where the Cloud network can be decomposed into several segments. Second, we plan to investigate on possible integration of solutions for these problems.

ACKNOWLEDGMENTS

This research was carried out as part of the activities of, and funded by the Smart Services Cooperative Research Centre (CRC) through the Australian Government's CRC Programme (Department of Innovation, Industry, Science and Research).

The study of Zeratul Izzah Mohd Yusoh was sponsored by the Ministry of Higher Education Malaysia through Universiti Teknikal Malaysia Melaka.

REFERENCES

[1] Foster, I., Yong, Z., Raicu, I., & Lu, S. (2008). Cloud Computing and Grid Computing 360-Degree Compared. In *Grid Computing Environments Workshop*. Austin, Texas: IEEE.

[2] Motahari-Nezhad, H. R., Stephenson, B., & Singhal, S. (2009). Outsourcing Business to Cloud Computing Services: Opportunities and Challenges. In *HP Lab Publication.* Hewlett-Packard Development Company.

[3] Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2009). A Break in the Clouds: Towards a Cloud Definition. *SIGCOMM Computer Communication Review*, 39(1), p. 50-55.

[4] Candan, K. S., Li, W.-S., Phan, T., & Zhou, M. (2009). Frontiers in information and Software as Services. In *Proceeding of the IEEE 25th International Conference on Data Engineering*. Shanghai, China: IEEE.

[5] Dubey, A., & Wagle, D. (2007). Delivering Software as a Service. *The McKinsey Quarterly*, p. 1-12.

[6] Cisco System Inc. (2008). Cisco Service-Oriented Network Architecture: Support and Optimize SOA and Web 2.0 Applications [Electronic Version]. Retrieved Mac 2011, from http://www.cisco.com/

[7] Satake, K. (2009). Fujitsu's approach to saas in japan Fujitsu saas platform. *Fujitsu Scientific and Technical Journal*, 45(3), p. 265-274.

[8] Karve, A., Kimbrel, T., Pacifici, G., Spreitzer, M., Steinder, M., Sviridenko, M., et al., (2006) Dynamic placement for clustered web applications. In *Proceeding of 15th International Conference on World Wide Web*. Edinburgh, Scotland: ACM

[9] Kwok, T. & Mohindra, A., (2008). Resource calculations with constraints, and placement of tenants and instances for multi-tenant SaaS applications. In *Proceeding of Sixth International Conference on Service-Oriented Computing*. Sydney, Australia: Springer.

[10] Tang, C., Steinder, M., Spreitzer, M., & Pacifici, G., (2007) A scalable application placement controller for enterprise data centers. In *Proceedings of the 16th International World Wide Web Conference*. Canada: ACM.

[11] Kichkaylo, T., Ivan, A., & Karamcheti, V., (2003) Constrained component deployment in wide-area networks using AI planning techniques. In *Proceedings of International Parallel and Distributed Processing Symposium*. Washington, USA: IEEE.

[12] Zhu, X., Santos, C., Beyer, D., Ward, J., & Singhal, S., Automated application component placement in data centers using mathematical programming. *International Journal of Network Management*, 2008. 18(6): p. 467-483.

[13] Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whally, I., & Snible, E. (2011). Improving Performance and Availability of Services Hosted on IaaS Clouds with Structural Constraint-Aware Virtual Machine Placement. In *Proceeding of the IEEE International Conference on Services Computing*. Washington, USA: IEEE.

[14] Verma, A., Ahuja, P., & Neogi, A. (2008). pMapper: power and migration cost aware application placement in virtualized systems. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*. New York, USA: Springer-Verlag New York, Inc.

[15] Hermenier, F., Lorca, X., Menaud, J. M., Muller, G., & Lawall, J. (2009). Entropy: a consolidation manager for clusters. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*. New York, USA: ACM.

[16] Wang, Y., & Wang, X. (2010) Power optimization with performance assurance for multi-tier applications in virtualized data centers. In *Proceeding of the 2010 39th International Conference on Parallel Processing Workshops (ICPPW)*. San Diego, CA: IEEE.

[17] Cucinotta, T., Palopoli, L., Abeni, L., Faggioli, D., & Lipari, G. (2010). On the Integration of Application Level and Resource Level QoS Control for Real-Time Applications. *IEEE Transactions on Industrial Informatics*, 6(4), p. 479-491.

[18] Rochwerger, B., Breitgand, D., Levy, E., Galis, A., Nagin, K., Llorente, I., et al. (2009). The reservoir model and architecture for open federated cloud computing. *IBM Systems Journal*, 53(4).

[19] Matthews, J. N., Dow, E. M., Deshane, T., Hu, W., Bongio, J., Wilbur, P. F., et al. (2008). Running Xen: a hands-on guide to the art of virtualization. Prentice Hall.

[20] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), p. 599-616.

[21] Broberg, J., Buyya, R., & Tari, Z. (2008) MetaCDN: Harnessing storage clouds for high performance content delivery, in *Proceeding of the Sixth International Conference on Service-Oriented Computing*. Sydney, Australia, ACM.

[22] Chong, F., & Carraro, G. (2006). Architecture Strategies for Catching the Long Tail [Electronic Version]. Retrieved Nov 2009, from http://msdn.microsoft.com/en-us/library/aa479069.aspx

[23] Chis, M. (2010). Evolutionary computation and optimization algorithms in software engineering: Application and techniques. USA: IGI Global.

[24] Mohd Yusoh, Z., & Tang, M. (2010). A cooperative coevolutionary algorithm for the composite SaaS placement problem in the Cloud. In *Proceeding of the Neural Information Processing. Theory and Algorithms*. Sydney: SpringerLink.

[25] Mohd Yusoh, Z., & Tang, M. (2010). Clustering composite SaaS components in Cloud Computing using a Grouping Genetic Algorithm. In *Proceedings of IEEE World Congress on Computational Intelligence*. Brisbane: IEEE.

[26] http://www-07.ibm.com/storage/au/