CORE
Metadata, citation and similar papers at core.ac.uk
Provided by ethesis@nitr

Service Level Agreement Aware SaaS Placement in Cloud

Sumit Bhardwaj

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela, Odisha, 769008, India

May 2015

Service Level Agreement Aware SaaS Placement in Cloud

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Technology

in

Computer Science and Engineering

(Specialization: Computer Science and Engineering)

by

Sumit Bhardwaj

(Roll - 213CS1139)

under the supervision of

Prof. Bibhudatta Sahoo

Department of Computer Science and Engineering

National Institute of Technology Rourkela

Rourkela, Odisha, 769008, India

May 2015

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela - 769008, Odisha, India

CERTIFICATE

This is to affirm that the thesis entitled ***Service Level Agreement Aware SaaS Placement in Cloud*** submitted by ***Sumit Bhardwaj***, to the Department of Computer Science and Engineering, in partial fulfillment for the award of the degree of **Master of Technology** (**Computer Science**), is a record of factual exploration work did by him under my supervision during the period 2014-2015. The thesis has satisfied all the necessities according to the regulations of the Institute and in my opinion, has come to the standard required for submission.

Dr. Bibhudatta Sahoo
Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769008, Odisha, INDIA

Place: NIT Rourkela
Date: May 25, 2015

Dedicated to my Parents and Siblings

Acknowledgements

I want to express my utmost appreciation to those who have helped throughout the completion of this thesis.
To my supervisor, Dr Bibhudatta Sahoo, I would like to express my heartfelt thankfulness for his invaluable support, continuous supervision, and greatly appreciated motivation during my research. I am really grateful for his patience and encouragement. His guidance has played a vital role in my academic, professional and personal development and I am very lucky to have had the opportunity to work with him.

I owe my deepest gratitude to the whole faculty members of Computer Science and Engineering department for providing me an excellent academic environment and support.

I must also convey my heartfelt thanks to the ever diligent staff of Computer Science and Engineering department for providing assent in everything we did.

Special thanks go to my parents and family members, who have been my source of motivation. This achievement would never have been possible without their love and support. Last but not least, I would like to thank all my friends, who have stood by me through thick and thin over the years.

Sumit Bhardwaj

213CS1139

May 25, 2015

Abstract

Cloud computing is an encouraging and favourable paradigm for both providers and consumers in diverse scopes of endeavors. Software as a Service (SaaS) is a method of conveying services or applications through the Internet as a service, and it is known to be one of the very crucial computing services in cloud computing. Cloud computing has become a major medium for the SaaS providers to provide their applications because required scalability can be achieved through this. The challenges of SaaS placement process depends on various factors comprising cloud network size, resource requirements, and communication among its components. This thesis analyzes the SaaS Placement Problem (SPP) and proposes a mathematical model for SaaS placement in Cloud. This thesis gives an evolutionary approach, known as Particle Swarm Optimization (PSO) that has been applied to find the optimal placement of SaaS component and aiming to minimize the total cost incurred to the SaaS provider, and then evaluated against the traditional Greedy approach in experiments. The obtained results show our proposed algorithm SPPSO generates better solutions than Greedy approach SPGA.

Keywords: Cloud computing, Software as a Service, SaaS Placement Problem, Particle Swarm Optimization, Greedy approach

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

In the previous couple of years, Information Technology (IT) has board up on a new model - Cloud computing. In spite of fact that Cloud computing is just a distinct method to convey computer assets, instead of another innovation, it has sparkled a rotation in the way of providing information and services by an organization. Cloud computing has turned into an extremely favorable standard for both consumers and providers in different fields of endeavors. An elemental change is created in computer architecture, development of software and tools, and of course, in the way the information is stored, distributed and consumed. A Cloud normally contains numerous resources considered to be distributed and heterogeneous. The suppleness is a function of Cloud computing for the resource allocation on request. Through this, it is possible to use the system's accumulative resources, invalidating the requirement of assigning a particular hardware to a task.

Before coming into the perspective of Cloud computing, websites and applications that were server based used to be executed on a particular system. With the emergence of Cloud computing, resources are utilized as an amassed virtual computer. This consolidated configuration endows an environment in which applications are executed independently without considering any specific configuration.

There are legitimate and notable business and IT explanations for the Cloud computing idea model change.

Reduced cost: With the help of Cloud computing, both capital expense, as well as operating expense costs are reduced because when there is a need then only

the resources are acquired and paid according to usage.
Refined usage of personnel: With the use of Cloud computing, free valued personnel allows them to revolve around delivering value instead of maintaining software and hardware.
Robust Scalability: Immediate scaling is allowed in Cloud computing, either up or down, without long-term responsibility at any time.

Buyya et al. (5) gave the meaning of Cloud as - "*The Cloud is a kind of parallel and distributed system comprising of a gathering of inter-connected and virtualized computers. These are provisioned dynamically and offered as a single or many integrated resource(s) for computing rooted in service-level agreements propounded through a negotiation process between the service consumers and providers.*"

A more recent definition can be found in a special publication on Cloud computing by the National Institute of Standards and Technology (NIST), America. According to NIST(35) "*Cloud computing is a model that enables universal, on-demand and convenient accessibility to a shared collection of the computing resources that can be provisioned dynamically and managed with a very few management efforts made by the Cloud providers.*"

The adoption and deployment of Clouds has many tempting benefits, such as scalability and reliability. Cloud computing has important characteristics which includes on-demand self service, broad network access, resource pooling, rapid elasticity & measured service.(35)
On-demand self service means the clients can ask for their own resources and can also deal with those processing assets.
Broad network access permits to offer services over the Internet or some other private networks.
Resource pooling means the clients can use the resources from a collection of computing resources, mostly in remote data centres. The offered services can be scaled smaller or larger, and the usage of the service is calculated and service users are billed accordingly based on pay-as-you-go model.

The Cloud model comprises of 3 service models(35) i.e. Software as a Service (SaaS), where the Cloud user is capable of using the applications, which are running on a Cloud infrastructure, and provided by Cloud service provider, Platform as a Service (PaaS) and the third is Infrastructure as a Service (IaaS).

Service Type	Service Focus	Existing Cloud Provider
Infrastructure as a Service (IaaS)	Storage, network, computation	Amazon EC2, Network Microsoft LiveMesh, Amazon S2
Platform as a Service (PaaS)	High level integrated development environment, deploying and testing custom applications	Google App Engine
Software as a Service (SaaS)	Software	Salesforce, Forecast

Table 1.1: Main services of Cloud computing

Table 1.1 depicts the characteristics of these main services, in terms of the service focus and some examples of existing providers.

The third Cloud service, SaaS, has received considerable recognition from the providers of software as well as from software users. Software is no more acquired and run by clients themselves on their own infrastructure but is run on the IT infrastructure of a hosting company. The demand for the SaaS is increasing every year (6).
In a report presented by Dubey and Wagle (9), it is reported that companies that are providing SaaS can create up to an 18% increment in revenue within three years.
Some other Analyst firms have also reported the positive growth of SaaS, includ-

ing the IDC, which stated that the worldwide revenue would reach $ 22 billion by 2015. These reports are evidence that SaaS has become a significant service to users, in 2009 revenue for SaaS was $13.1 billion, and that it would reach upto $40.5 billion in 2014 (33). Gartner (34) forecast that the leading to challenges in providing a better SaaS to meet these. In addition, advancement in Cloud computing infrastructure has given an efficient meaning for hosting SaaS, thereby making more accessibility of SaaS to a wide range of software users.

1.2 Software as a Service

Software as a Service (SaaS) endows network-dependent accessibility to the commercially available software that is conveying applications over the Internet as a service. Rather than installing and maintaining the software, consumers simply get to it through the web, liberating consumer from sophisticated hardware and software management and consumers pay for service as per use basis(44). The consumer doesn't control or manage the hidden cloud infrastructure, which includes servers, storage, network, operating systems, or even individual application abilities, with the possibility of exemption of constrained client-specific application setup settings(35).
Examples of SaaS include Acrobat.com, Photoshop.com, Intuit, Netflix, QuickBooks online, Gmail, and Gmail docs.

SaaS describes the prospective for a lower-cost path for organizations to use software; instead of buying the licenses for each of the computer, use the software on demand, especially when it is understood that most of the computers are sitting idle for 70% of the time.
Over the last few years, the use of SaaS has skyrocketed and hinted at no easing off. Gartner forecasted that the worldwide market would grow from US$18.2 billion to US$45.6 billion, from 2012 to 2017. IBM surveyed that among over 800 companies, the top reason given for adopting SaaS was reducing the total cost of ownership of their applications. 41% reached that goal to a higher degree.

1.3 Service Level Agreement

An agreement between a service provider and its inward or outer clients that reports what services the clients are supposed to get, is known as Service Level Agreement (SLA)(12). It is a formal, discussed document that tries to define the services being offered to the clients in quantitative manner. Conveying of applications as services (SaaS) over the Internet and hardware services (IaaS) are the important sections of cloud computing. As customers or organizations are adopting such Service Oriented architectures (44), the reliability and quality of service being offered, becomes main aspects. It is not always possible for service provider to fulfill the demands of customers and organizations using their services and thus a balance must be made. So the service providers and service users commit to an agreement that is referred as SLA (38).
An SLA is composed of three major parts:

1. A collection of promises made to service users.
2. A collection of services not made explicitly to users, i.e. limitations.
3. A set of obligations that user must accept.

The Quality of Service (QoS) attributes that are commonly the SLA parts i.e. **throughput** and **response time**, needs to be closely observed time to time (38).

1.4 SaaS Deployment

The word "*SaaS deployment*" alludes to the SaaS installation and delivery, rather than the traditional on premise model of deployment of software. It is similar to the traditional, accepted state of a utility service that is followed by measuring & charging at continual periods, for the provided or delivered services (48). The delivered services must not violate the constraints defined in the SLA.
SaaS is a combination of different type of components; application component (AC), integration component (IC), business component (BC), and storage component (SC) (50). Each component is to satisfy some business function. SaaS components are deployed on the top of the virtual machines in cloud computing

infrastructure which is provided by cloud vendors to SaaS provider. A cloud data center consist two types of servers; storage servers and computation servers. Each server has some limited processing capacity, I/O capacity, and memory size. On the top of each server some virtual machines are running with different capacities. A server capacity is divided among the virtual machines to satisfy the SaaS components need. SaaS components are then deployed on those virtual machines based on satisfying the requirements of each component.

Figure 1 depicts the three different role for a SaaS i.e.
1) SaaS Consumer, who uses the service provided through internet,
2) SaaS Vendor, who sells and
3)SaaS Provider, who provides an IT infrastructure for that SaaS.
However, SaaS provider and SaaS Vendor may be the same.

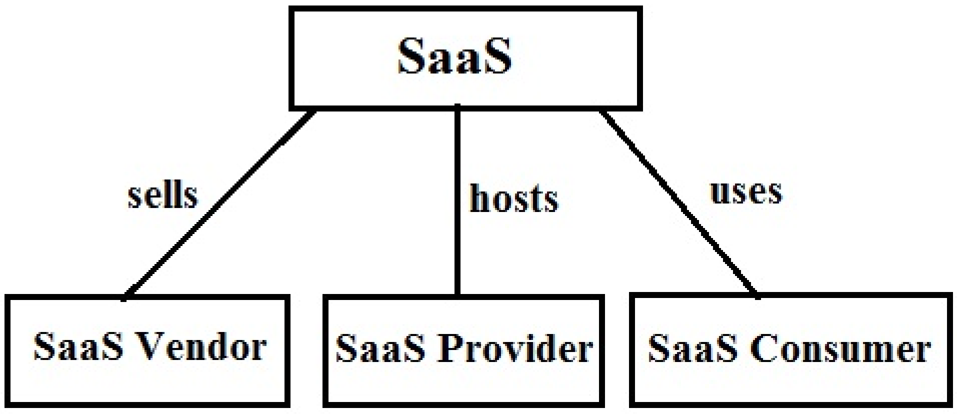

Figure 1.1: Different roles for a SaaS

The chief benefit of SaaS deployment is the reduction in the delivery cost for both SaaS subscribers and the SaaS providers. Since SaaS providers can deploy centrally, fix and update their offerings on the cloud, the maintenance cost is also reduced for the provider of the SaaS. Similarly, the subscribers need not to worry about the software's outdated versions or the cost of licensing for multiple users required for upgrading to the latest version of a software product.

1.5 Related Work

In this section, existing work done is discussed, related to SaaS components placement problem on the clouds and resource optimization in virtual servers. In cloud computing environment, it is very common to deploy SaaS components onto the Clouds as to satisfy the consumers needs. Placing SaaS components onto cloud servers shares similarities with an existing problem known as Component Placement Problem(CPP) (28). CPP is further categorized into two parts:
1) offline CPP, where component are placed at initial stage,
2) online CPP, where component placement is done during run time.
However, SPP is much similar to offline CPP because the placement is done at initial stage (48). Existing work on CPP is related with data centre's resource allocation to the application components.
Several existing studies formulated CPP as a resource optimization problem and also as a variant of the multiple knapsack problem (45).

Kichkaylo et al. (28) defined the application by as set of interface and component types where each of the component specifies required service for the execution through interface. In this paper, they proposed a general model for CPP and presented an algorithm based on efficient planning algorithms developed by artificial intelligence community. There was a drawback with this algorithm that it may fail if the resources are tight.

Karve et al. (26) proposed a middleware clustering technology through which resources can be allocated to web application through dynamic Application Instance Placement (AIP). AIP is defined as problem of placing the instances of application on a server machine set to satisfy the varying demands of application clusters for resources out of available resources. Their objective was to minimize the placement changes made during deployment. Karve divided the resources into two categories, one is load dependent that depends on the intensity of application load, and another is load independent that do not completely depend on the intensity of application workload. SaaS placement is also said to be NP-hard problem in this paper.

Zimmerova et al. (53) focused on the relational aspect between components of the problem presented in (28), and proposed a solution concerning both interaction and non-interaction properties as well. They used automata language for capturing the inter-component communication. Zimmerova's proposed method was to integrate with an existed method on the non-interaction aspects of components.

Urgaonkar et al. (45) used the first-fit based approximation algorithm in placing component applications in an offline CPP. The algorithm proposed by them places the component at the first server found that can satisfy its demand.

Zhu et al. (52) addressed the Application Component Problem(ACP) for a data center. ACP is to decide which physical server should host the application component in order to minimize the processing, storage and communication requirement and resources are effectively used as well. They formulated the ACP problem by using a mathematical optimization framework and further as a mixed integer program, and proposed a solver using virtualization technology. The ACP solver presented by them worked in many scenarios that includes fail-over and migration, but there was a drawback of not able to deploy several component on the same server.

Yusoh et al.(48) proposed a Genetic algorithm with penalty, for composite SaaS placement problem in cloud, considering both software as well as data components of SaaS. The objective was to optimize the SaaS performance based on its total execution time and optimization of resource consumption in each server, keeping in mind the communication part between data and software components in cloud storage servers. It is also said that the proposed algorithm is scalable.

Yusoh et al.(50) further enhanced the work done in (48), by presenting evolutionary algorithms for placement of Composite SaaS and optimizing resource usage as well. In this paper the authors have taken care of the response time of the services being offered to meet the maximum response time defined in SLA. It is a different approach to deal with the problem of initial placement of composite SaaS onto physical cloud servers and, maintenance phase in which reconfiguration is done for resource usage optimization and subscription cost for users is also minimized. In this paper, they addressed the problems resulted from composite SaaS placement, which was focused on constraints, requirements and inter-dependencies rather than from platform aspects.

Zhenzhang et al. (31) presented a model for deploying the SaaS components in the cloud computing environment and also proposed a method for solving SaaS placement problem based on the Ant Colony Optimization technique, and claimed to perform better than the genetic algorithm for the same.

Zhipiao et al. (32) established a request model for cloud service and proposed a genetic algorithm based cost-aware scheduling technique for servicing request that too cost effective and not violating SLA constraints. Their approach was not only limited to reusing the resources but minimizing rental cost and maximizing providers profit.

1.6 Research Motivation

Software as a service has received a lot of consideration from IT industries. It is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand(51). Some leading companies providing Cloud services are, Media Temple, AT & T, Grid Player, Joynet, Flexiscale and so on. Most of these also provide hosting services, such as Media Temple, ATT, Hosting.com, Hosting365, Grid Player, and so on.
A SaaS delivered as a composite application or as multiple components form, in which the software components are loosely coupled and components communicate to each other in order to provide a high-level functional system(15). For providing software as a service first, there is a need to place the saas on the virtual machines running on the top of cloud servers. The SaaS Placement is said to be NP-Hard Problem(26)(53)(31).

Because the problem nature is NP-Hard, finding the solution of the problem via conventional algorithms is not possible, therefore some heuristic algorithms are required to find a solution that will provide a sub-optimal solution which is very near to an optimal solution. Most of the researchers used Genetic Algorithm to solve this problem, because it is an evolutionary approach. This research will use Particle Swarm Optimization (PSO)because of its easy handling and evolutionary nature to handle the problems' challenges. PSO is a stochastic search terminology which applies biological evolutions in production of solution.

1.7 Research Objective

SaaS components are deployed on top of the Virtual Machines(VMs) in cloud computing infrastructure which are provided by cloud vendors to the SaaS provider. A VM can host multiple components with different requirements at a time. As mentioned in previous section, SaaS Placement is known to be NP-Hard. In this research an optimal placement strategy will be carried out that will place the SaaS components on the VMs that will satisfy the resource constraints such that SLA violations will be minimized considering the QoS parameters as Response time and Cost.

1.8 Research Contribution

This thesis formulates the service level agreement aware SaaS placement problem using resource constraints and service level agreement constraints. In this research particle swarm optimization framework is proposed for placement of SaaS components onto the virtual machines running on the cloud servers. The outcomes of this research can benefit entities that are involved in providing the software as a service.

1.9 Thesis Outline

In this chapter, a brief introduction of Cloud computing, SaaS, Service level agreement, the motivation toward the SaaS placement problem and the objective of this research is discussed.

The rest part of the thesis is organized into the following chapters :

In **Chapter 2**, Cloud computing, service models of Cloud, Cloud deployment models, SaaS, SaaS examples are described in brief. The SaaS Placement Problem is discussed in detail and mathematical formulation with the resource, and SLA constraints is given for the same. The strategies applied for placement of components till now is also discussed.

Chapter 3 deals with the proposed approach i.e. based on Particle Swarm Optimization, for solving the problem presented in the Chapter 2. Algorithm for the proposed strategy to solve the research problem with another algorithm for comparison, results and conclusion based on those are also discussed.

Chapter 4 is the overall conclusion of the this research work and future work also that can be further done to optimize our approach and to get better results.

Chapter 2

SaaS Placement in Cloud

2.1 Introduction

Nowadays Cloud computing is very much popular among IT service providers. The most spoken term *Cloud* in the field of IT was coined by the Google CEO Eric Schmidt (2). In late 2006, he used the term to describe the Google approach for Software as a Service. According to a study by International Data Corporation (IDC), a leading IT analysis firm, identified cloud computing as one of the prevailing technology trends in the new decade (4). Other research by Merrill Lynch, a global financial services firm, predicted that Cloud providers would gain huge revenues from the Cloud's services and advertising (5).

2.2 Cloud Computing

Since the appearance of Cloud computing, several definitions of cloud computing have been published. Although no standard or exact definition is there, most of them share common characteristics which describes the cloud computing concept. Foster et al. (14) defined Cloud computing as a specialized distributed computing infrastructure with four main characteristics:

1) it is enormously scalable,

2) it is an intellectual entity that delivers different levels of services to clients,

3) it is driven by economics by scale, and

4) its services can be configured dynamically.

According to Vaquero et al. (46), major characteristics of Cloud computing infrastructure are 1) a huge collection of virtualized resources, 2) the potential for dynamically configured resources, 3) a pay-as-you-go model and, 4) an infrastructure or service provider offering users' Service Level Agreements (SLAs).
Buyya (5) stated that the resources in Cloud are provisioned based in users' SLAs and users are billed according to their usage of services.

Based on the existing definitions, Cloud computing will be referred to as: *A large-scale computing infrastructure that offers on-demand scalable services i.e., computation power storage, platform for development and applications as services, to the clients over the Internet through thin client devices like web browser. These services are managed by the Cloud provider. For market purposed Cloud, the business model is based on a pay-per-use model or on subscription for a fixed time period. The most important aspect is the services are subject to meet certain SLA constraints with the users.*

2.2.1 Cloud Architecture

Figure 2.1 shows the reference architecture of Cloud computing given by NIST (30). Through this figure different roles, their activities and functions are identified. A generic high-level architecture is given to encourage the comprehension of the prerequisites, uses, qualities and models of cloud computing. This architecture identifies the major roles such as Cloud Consumer, Cloud Provider, Cloud Broker, Cloud Carrier and, Cloud Auditor and their working in cloud computing. Each actor in the figure is an entity which may be a person or an organization that participates in a process and performs tasks in cloud computing.

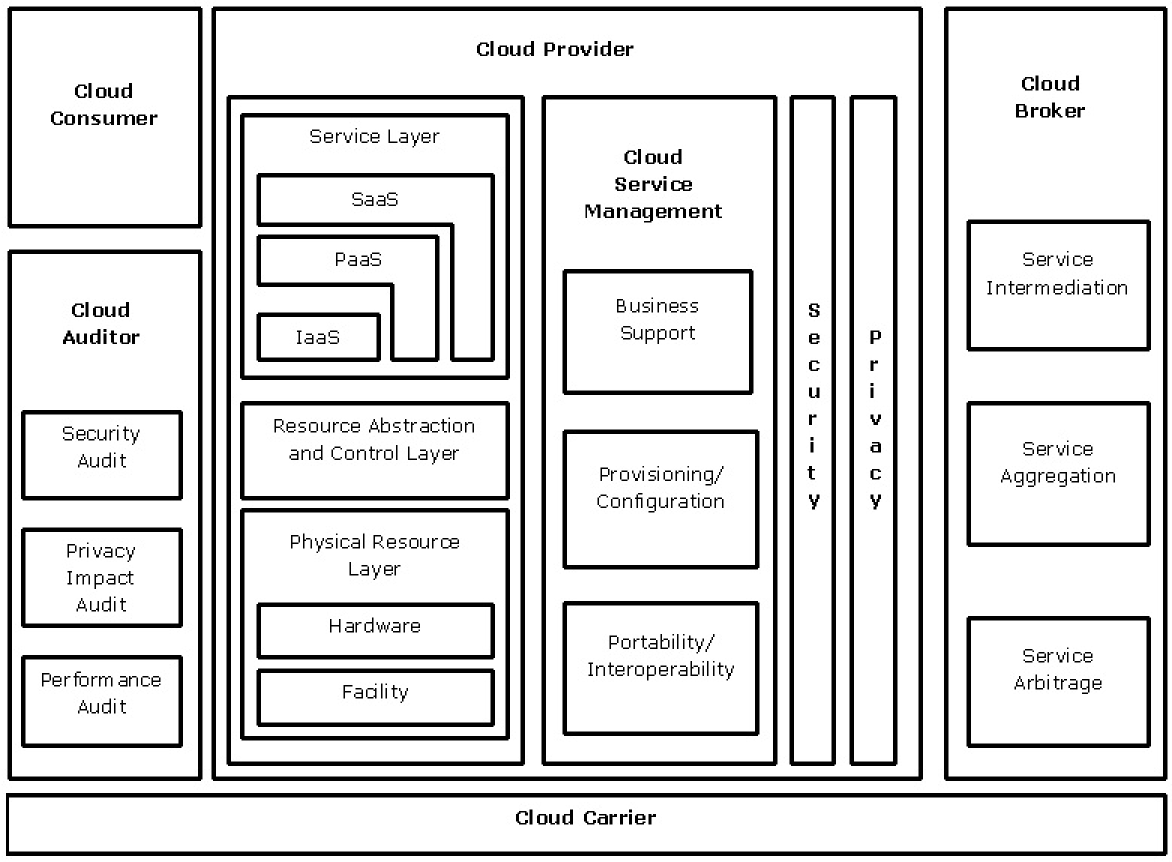

Figure 2.1: Cloud Computing Architecture (30)

2.2.2 Cloud Service Models

Resources in the Cloud refer to the computation power, storage servers, platforms and applications based on the previous definition of Cloud computing. These resources are classified into three type of services: 1) Infrastructure as a Service (IaaS), 2) Platform as a Service (PaaS), and 3) Software as a Service (SaaS) (46) (35). Other services have also been mentioned in the existing literature, such as Hardware as a Service (HaaS) (2) and Shared Application Infrastructure as a Service (37). However because of non-existence of exact and clear definition of these service, mainly three of the services are considered. These 3 services are the most important pillars of the Cloud through which the cloud solutions are provided to the customers. Figure 2.2 shows the Cloud computing service architecture, which includes the three services of the Cloud that can also be considered as

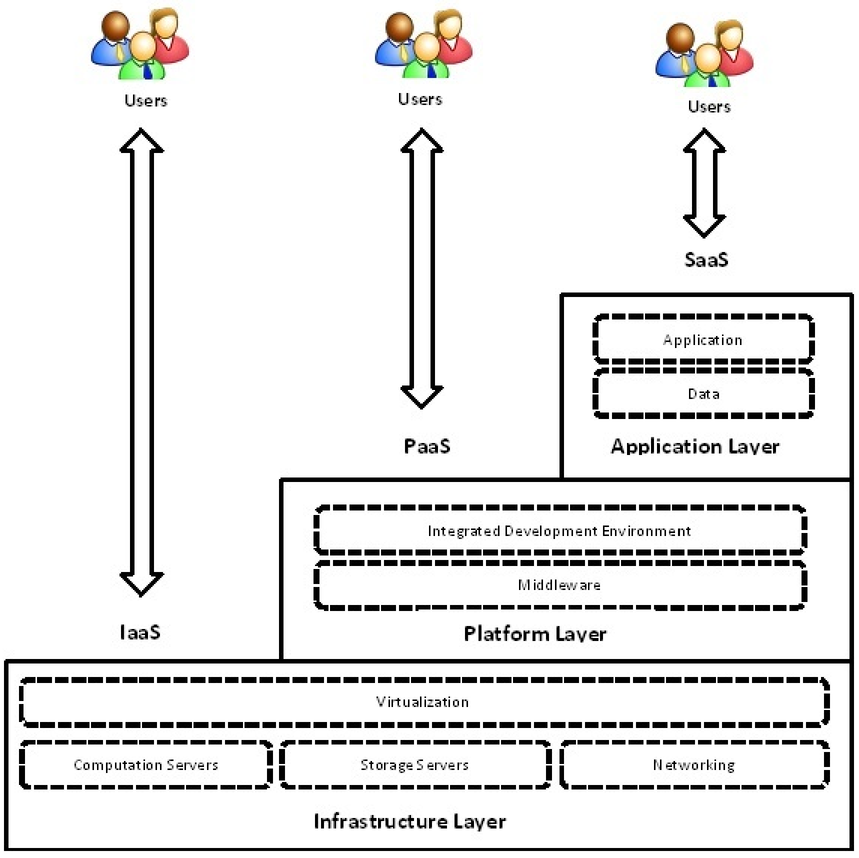

Figure 2.2: Cloud Service Architecture

services-by-layer. IaaS offers fundamental computing resources to the end users such as computation capacity, storage and network(35). The providers of IaaS usually the service in unit if Virtual Machine (VM) instances where a VM is an abstraction of the hardware resources of physical servers that includes CPU, memory and disk drives (43). Example for this category: Amazon web services offers two IaaS services, Amazon EC2 for computation resources and Amazon S3 for storage.

PaaS provides a high level integrated environment for developers to design, deploy, maintain, test and implement their applications (36). Programming languages, libraries, related services and tools are provided to the developers by the providers for developing or implementing their applications. Examples are GoogleApps Engine and Force.com.

SaaS, the most commonly used service, refers to the application hosted by Cloud providers. The main focus of this research is on SaaS. Salesforce and Foresoft are the examples of SaaS.

Apart from being the services stack these layers indicates the roles and responsibilities of the users and providers. As the height of the layer increases, the managing responsibilities are shifted from users to the providers.

2.2.3 Cloud Deployment Models

Cloud deployment models are generally divided into three types: 1)*public*, 2) *private*, and 3) *hybrid* (7). These cloud models share common characteristics but the main difference is because of the different groups of users for whom the Cloud is built.

A public Cloud is a type of Cloud that offers the services for public use; it is owned and operated by an entity that is referred to as the Cloud provider (35). Public clouds are mostly market oriented in which the services are offered on pay-per-use basis. SLAs are usually established between the Cloud providers and users to provide Quality of Service (QoS) guarantees. Amazon EC2 (18), SalesForce, Google App (19)

A private Cloud, also known as enterprise Cloud, is a type of Cloud in which the services are exclusive to a single organization only (21). The data center is owned by the organization and is usually located on the premises. All of the services are used by the organization and also managed within the organization.

The hybrid Cloud, as suggested by the name, is a combination of both public and private Cloud.
All the Cloud providers have massive data centers to provide services to millions of users or more than that. A report by The Economist (1) stated that Google has three dozen servers with more than one million servers across its global network. The report further stated that Microsoft is trying to catch up by adding 20,000 servers a month to their data centers.

2.3 Cloud Software as a Service

Before Cloud computing came into view, SaaS had been successfully implemented in the servers of SaaS vendors and it was delivered via Web (37). But, when there was an increasing demand for SaaS each year (6), SaaS vendors need to find a solution to cope with these growing requests. An obvious solution for this problem is to host the SaaS in a Cloud computing infrastructure as it provides scalability to the SaaS that runs in a Cloud.
Based on the published definitions of Software as a Service (SaaS), the best basic definition of SaaS is provided by Chong and Carraro (8), who referred it as 'software deployed as a hosted service and accessed over the Internet. To characterize it further, various existing definitions of SaaS have been reviewed (3) (17) and it can be concluded that SaaS differs in three criteria from the conventional software or ASP-based applications: its 1) software possession, 2) business model, and 3) software design.

First criterion: Two approaches preceded SaaS: the first, a conventional software approach and the second, the Application Service Provider (ASP) approach. In the conventional software approach, clients need to buy the software licenses from vendors and install the software on their own machines. The software comes along with a package of a CD installation and its manual, and the software price usually includes the maintenance cost by the vendor (22).
In the ASP approach, the software is still bought by customers obtaining a license, but it is installed at the ASP data center. The software is not shared with other clients, and the ASP is responsible for maintaining the infrastructure of data center as well as the software.
In SaaS, however, the software resides at the provider's servers and customers will use the software via the Internet. The software's owner is the vendor and clients use the software whenever they need only. This eliminates the IT overheads cost for the clients, as they need not be worried about any other IT infrastructure and management (except for their personal computers and Internet connections). It can be clearly seen that the main difference between these approaches is the shift of software possession from the users to the software providers.

Second criterion: It is the business model. Conventional software providers offer a one-off price to users, which include the right for using the software as long as customers want to and the support and assistance directed in the terms and conditions agreement. In addition, users have to bear the hardware cost and its maintenance cost. ASP relieves some of the latter cost from users by hosting the software in their own data center. In this approach, users are charged for one price that includes the software license, the hosting and the maintenance. It should be clear that the software is not developed by the ASP or does not belong to ASP; the ASP companies are middle parties that buys the license from the software company and provides the hosting infrastructure to users (22).
In SaaS, users are charged on a usage-derived basis via either subscriptions or a pay-per-use scheme. Users do not need to buy the software license as it is required in the conventional and ASP approach; they pay only when they use the software and for the time period they use it. For the hardware and software infrastructure, as the ownership of the software shifts from the users to the providers, the costs for these are completely covered by the providers.

Third Criterion: The multi-tenancy concept is the fundamental design of SaaS that separates it from the approaches of other applications such as conventional software, ASP or web-based applications. With the help of multi-tenancy, different users can use the service concurrently on the shared hardware and software infrastructure.
A SaaS can be delivered as a composite application, which consists of a group of loosely-coupled individual applications that communicates with each other in order to form a higher-level functional application or system. These components can be either data sources or services that perform a specific function of the SaaS, and can also be interdependent with one another.

2.3.1 SaaS Application Model

To characterize the SaaS for Cloud, the SaaS application model is developed based on two existing SaaS maturity models. Although these models have different views in some aspects, yet they target to the same objective, which is to define the key attributes of a mature SaaS application. The two existing maturity models are proposed, one by Microsoft and other model proposed by Kitagawa et al., outlines a composite SaaS application model.
In the SaaS maturity model developed by Microsoft, three key attributes of SaaS applications are there i.e. configurability, multi-tenant efficiency and scalability that are indicators of the maturity of a SaaS application. The model has four incremental levels, each level is considered to be an upgrade from the previous one in terms of the key attributes.

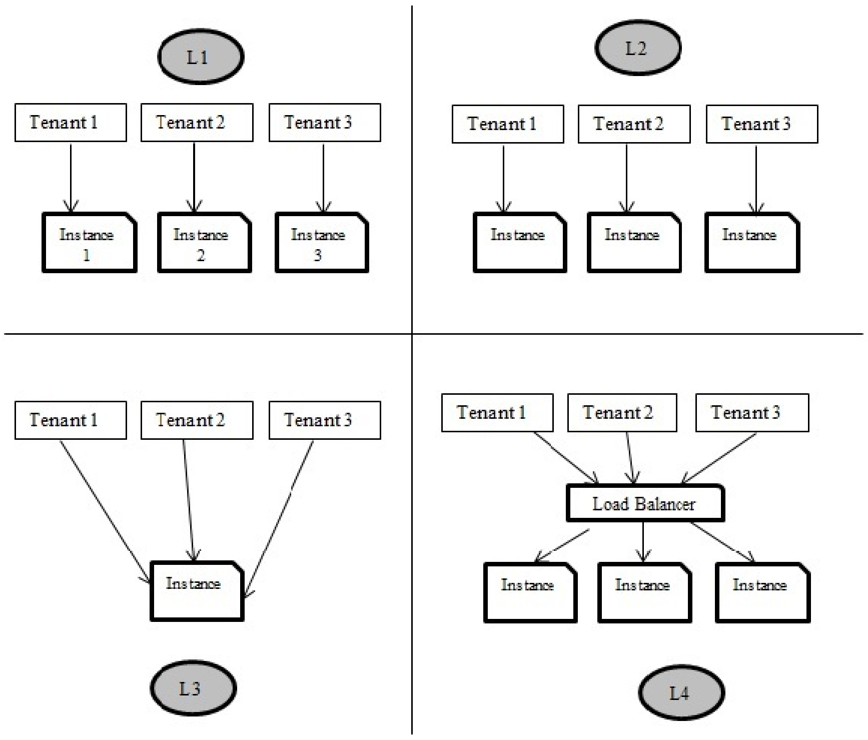

Figure 2.3: Micorsoft's SaaS Maturity Model

Figure 2.3 illustrates all of the four levels of the maturity model. In the figure, an instance is a copy of the SaaS and a tenant is an organization or an individual that is subscribed to the SaaS. Level 1 in the maturity model, denoted as L1, allocates an instance to each tenant exclusively; the instance will be configured and developed specifically to meet the need of the tenant. Level 2 also has a separate instance for each tenant; however, these SaaS instances are not developed exclusively for each tenant. In this level, there will be configuration options to meet the tenant's specific needs. Level 1 and Level 2 apply the multi-instance concept. In a multi-instance concept, one instance of the application is set to serve only one tenant. As such, the SaaS providers create several identical instances of the SaaS in order to serve multiple tenants.

Level 3 introduces multi-tenant support, through which an instance of the application can be shared among a number of tenants. The tenants' functionality will be configured according to their needs. In Level 4, scalability features are added through a load balancer mechanism that balances the allocation of the instance to its tenants. Based on the current technology of Cloud computing and SaaS demand, Level 3 and Level 4 are the mature levels being practiced by SaaS providers.

Kittagawa et al. (24) proposed a more comprehensive SaaS maturity model. They defined the core criteria of SaaS using two axes: service component as the x-axis and the maturity level as the y-axis. The service component axis represents four features of structuring software business: 1) data, 2) system, 3) service and 4) business. The other axis represents the maturity levels of the SaaS in respect of four types to SaaS offering: 1) ad hoc/base, 2) standardization, 3) integration, and 4) virtualization.

For the x-axis, only the system and service components are discussed because these two are relevant to the scope of this thesis. The maturity model is depicted in Figure 2.4.

In Figure 2.4, each level in the maturity model is described based on the service components. Level 1 is similar to the Microsoft maturity models that were discussed before, where the application is developed to cater for a single tenant's requirements only. The applications in this level are more akin to the ASP ap-

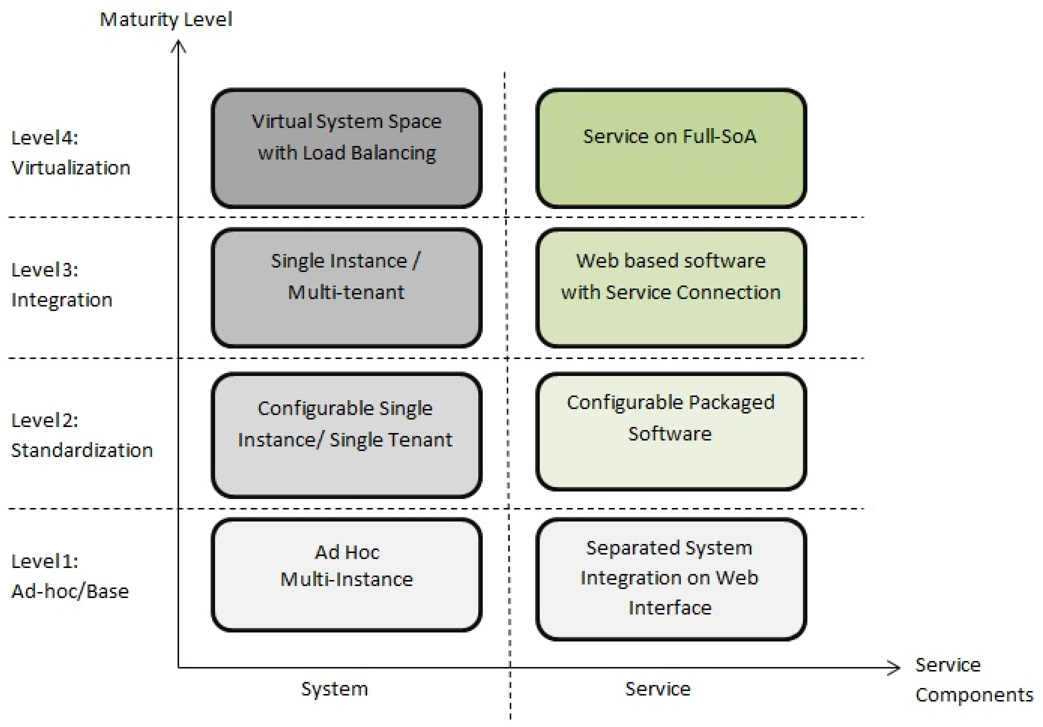
Maturity Level
Level4: Virtualization
Level3: Integration
Level2: Standardization
Level1: Ad-hoc/Base
Virtual System Space with Load Balancing
Service on Full-SoA
Single Instance / Multi-tenant
Web based software with Service Connection
Configurable Single Instance/ Single Tenant
Configurable Packaged Software
Ad Hoc Multi-Instance
Separated System Integration on Web Interface
System
Service
Service Components

Figure 2.4: A General SaaS Maturity Model

proach than to the SaaS approach. In Level 2, the configurable application is introduced with no multi-tenant support. Level 3 applies multi-tenant support with service connection. The service connection refers to the combination of services to serve various users' functions. Level 4 presents the most mature SaaS, which uses multi-tenant with load balancing, and the application architecture is fully on SoA. SaaS at this level largely uses virtualization technology in a Cloud. Levels 3 and 4 in this model include software with service connection and service on SoA. The authors further stated that the service connection can be achieved by web services or mash ups.
The two maturity models discussed above indicate that SaaS deployed in a Cloud infrastructure is regarded as the most mature SaaS, with several fundamental features including 1) configurability, 2) multi-tenancy, and 3) scalability.

2.3.2 SaaS Examples

Several companies are there which offer Software as a Service. Salesforce is one of the earliest commercial providers, began its SaaS operation in 1998 (20). The main product of Salasforce, Customer Relationship Management (CRM) solu-

tions, is divided into two parts: 1) Sales Cloud which caters for sales personnel tasks, including sales managers, sales representative, sales marketers, providing functionalities such as sales forecast and analysis, customer information management and others. 2) Service Cloud, through which Salesforce aims to provide help to sales personnel clients by providing a customer service center with various means of social media channels including chat, online calls, portals and forums.
Salesforce (20) also offers a development platform, Force.com, where customers can develop their own applications to be used along with SaaS.
Google also has its own SaaS offerings (19), named as Google Apps, which covers a large collection of SaaS i.e. 1) communication applications which includes Gmail, Hangout, Google calendar, 2) office applications which includes Google docs, spread sheets, and presentations, and 3) a mash-up service (iGoogle) and Google sites. Google Apps for work with vault is available for $10 per user per month and Google Apps without vault is for Rs. 150 per month.
IBM also offers "Blue Cloud" named SaaS solution. Via Blue Cloud corporate data centres will be operated across a globally distributed accessible resources by enabling computing. It is based on open source software provided by IBM.
Microsoft also offers SaaS named as Microsoft Office Live Small Business. Microsoft Office Live Small Business offers services i.e. storage manager, an e-commerce tool to sell products for helping a small business, and E-mail Marketing Beta. Microsoft also offers Office 365 for home (Rs. 420.0 per month), personal (Rs. 330.0 per month), and business.

2.4 SaaS Placement Problem

A SaaS is composed of several components. SaaS components are deployed on top of the Virtual Machines(VMs) in cloud computing infrastructure which are provided by cloud vendors to the SaaS provider. A VM can host multiple components with different requirements at a time.
SaaS Placement problem is known to be NP-Hard. In this research, an optimal placement strategy will be carried out that will place the SaaS components on the VMs that will satisfy the resource constraints such that SLA violations will

be minimized considering the QoS parameters as Response time and Cost. The cloud infrastructure for placing the SaaS components is described below. Let we are having some virtual machines denoted by a set

$VM = \{vm_1, vm_2, vm_3, vm_4\}$

And a SaaS composed of 8 components denoted by a set

$SC = \{sc_1, sc_2, sc_3..., sc_8\}$

These SaaS components are to be placed on top of these available virtual machines. There may be many possible placements for these components as a VM can host multiple components at a time based on its capacity.

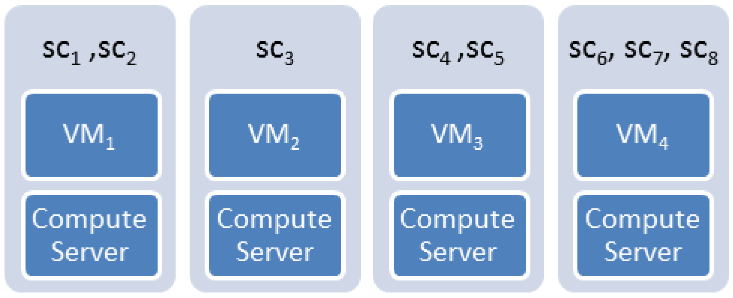

Figure 2.5: First possible scenario of SaaS Component placement

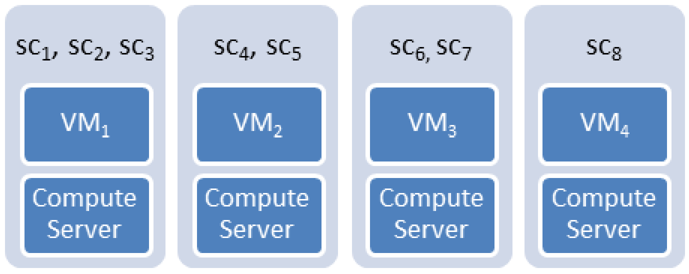

Figure 2.6: Second possible scenario of SaaS Component placement

Figure 2.5, 2.6 and 2.7 shows three different possible scenarios for placing the SC on the available VMs. There are various solutions ($= 4^8$) possible other than these but not all are optimal.

Let we are having m virtual machines denoted by a set

$VM = \{vm_1, vm_2, vm_3..., vm_m\}$

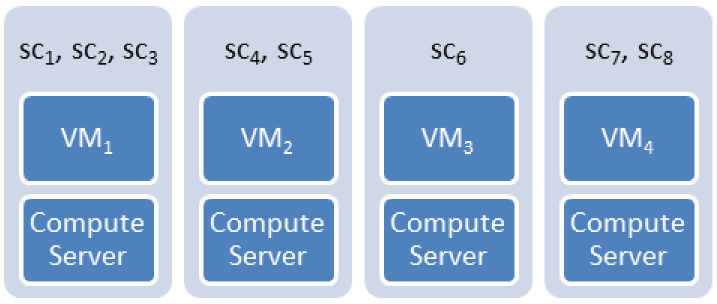

Figure 2.7: Third possible scenario of SaaS Component placement

And n SaaS Components denoted by a set
$SC = \{sc_1, sc_2, sc_3..., sc_n\}$

there will be a large number of placements possible of placing the SaaS components on to the Virtual Machines. So, total number of possible solutions for placing these components will be $= m^n i.e. exponential$.
As we know $m >> n$, finding an optimal solution, that meet our requirement is NP-Hard.

2.5 Problem Formulation

In the Cloud computing infrastructure, a set of servers with their resource capacities and virtual machines, is connected with the communication network with its links. To made the service or computing available to the end users Virtual machines are deployed onto the set of servers available (29). Virtual machines that are placed on the servers have their own capacities: memory capacity, storage capacity, processing capability and input output capacity. The set of the SaaS components with its requirements: memory requirement storage requirement, processing requirement and input output requirement, is placed on the virtual machines. The objective is to determine the placement of each SaaS component onto the virtual machines running on the servers, such that the performance of the SaaS is optimal based on the cost, while satisfying all the resource constraints. And SLA's constraints must not also be violated in placing the SaaS components. Hence the resulted set of VMs (by the placement of SaaS components) should be satisfying resource as well as SLA's constraints.

Cloud infrastructure, consists of cloud data center that is a set of cloud servers $CS = \{cs_1, cs_2, cs_3, ..., cs_p\}$ and a set of Virtual machines (VMs) denoted as $VM = \{vm_1, vm_2, vm_3..., vm_m\}$ running on those cloud servers. The resource availability or capacity a VM is represented with a tuple $(PC_{vm_i}, M_{vm_i}, SS_{vm_i}, BW_{vm_i})$, $1 \leq i \leq m$. Where PC_{vm_i} is the Processing Capability of vm_i, M_{vm_i} is Main memory capacity of vm_i, SS_{vm_i} is Storage capacity of vm_i, and BW_{vm_i} is IO Capacity or Bandwidth of vm_i. This problem modelling for the cloud gives a general idea of VMs and their resource capacities.

Resources	**Description**
$cs_x \in CS$	x^{th} cloud server cs_x in CS, where CS is a cloud server set. $x \leq p$
$vm_i \in VM$	i^{th} virtual machine vm_i in VM, where VM is a virtual machines set. $i \leq m$
PC_{vm_i}	Processing Capability of i^{th} vm
M_{vm_i}	Main memory capacity of i^{th} vm
SS_{vm_i}	Storage capacity of i^{th} vm
BW_{vm_i}	IO Capacity or Bandwidth of i^{th} vm

Table 2.1: Cloud resources' attributes

SaaS Components, denoted by a set $SC = \{sc_1, sc_2, sc_3..., sc_n\}$ that need to be placed on the top of the VMs running on the cloud servers. The resource requirement of a component can be represented with a tuple $(TS_{sc_i}, MM_{sc_i}, S_{sc_i}, IO_{sc_i})$, $1 \leq i \leq n$ and $n \ll m$. Where TS_{sc_i} is task size of sc_i, MM_{sc_i} is main memory requirement of sc_i, S_{sc_i} is size of sc_i, and IO_{sc_i} is IO requirement of sc_i. The modelling of SaaS components gives a general idea about the resource needs for a component.

Resources	**Description**
$sc_x \in SC$	x^{th} saas component sc_x in SC, where SC is a SaaS components set. $x \leq n$
TS_{sc_x}	Task size of x^{th} component
MM_{sc_x}	Memory need of x^{th} component
S_{sc_x}	Storage requirement of x^{th} component
IO_{sc_x}	IO requirement of x^{th} component

Table 2.2: SaaS components resources' requirements

2.5.1 Objective

Our objective is to find the optimal placement plan of SaaS components onto the set of available VMs,

$$P : SC \rightarrow VM \tag{2.5.1}$$

where $sc_j \longrightarrow P(sc_j) = vm_j$, $1 \leq i \leq n$, $1 \leq j \leq m$, means the j^{th} SaaS component is placed on i^{th} virtual machine, such that the placement minimizes the total cost incurred to the SaaS provider while deploying a SaaS, that can be mathematically shown as below:

$$min(\Sigma_{i=1}^{m}\ \Sigma_{j=1}^{n} x_{i,j} * C_{i,j}) \tag{2.5.2}$$

Where,

$$x_{i,j} = \begin{cases} 1 & \text{if } P(sc_j) = vm_i, \\ 0 & \text{otherwise .} \end{cases} \tag{2.5.3}$$

$$C_{i,j} = tet_{i,j} * cost_{vm_i} \tag{2.5.4}$$

where $C_{i,j}$ is the cost incurred due to j^{th} SaaS component placement on i^{th} virtual machine and is a multiplication of $tet_{i,j}$, which is the total execution time of j^{th} SaaS component when it is placed on i^{th} virtual machine, $cost_{vm_i}$.
tet_{sc_j}, the total execution time of sc_j is calculated based on the processing and transferring of the component data, and the $cost_{vm_i}$ is the cost of i^{th} virtual machine, which is combination of processing cost, memory cost, storage cost, and bandwidth cost.

2.5.2 Resource Constraints

While placing the SaaS components, the total resource requirements for those SaaS components that are to be placed in virtual machines must not exceed the resource capacities of virtual machines.

$$\forall vm_i \in VM\ \Sigma_{sc_j \in SC}\ TS_{sc_j} \leq PC_{vm_i} \mid P(sc_j) = vm_i$$

$$\forall vm_i \in VM\ \Sigma_{sc_j \in SC}\ MM_{sc_j} \leq M_{vm_i} \mid P(sc_j) = vm_i$$

$$\forall vm_i \in VM\ \Sigma_{sc_j \in SC}\ S_{sc_j} \leq SS_{vm_i} \mid P(sc_j) = vm_i$$

$$\forall vm_i \in VM\ \Sigma_{sc_j \in SC}\ IO_{sc_j} \leq BW_{vm_i} \mid P(sc_j) = vm_i$$

2.5.3 SLA and Execution time Constraints

To ensure the SLA, we have considered QoS parameter *response time.* For optimal SaaS performance, the placement or deployment of SaaS components onto the virtual machines is based on the total execution time. The total execution time of the SaaS is calculated based on the time for transferring the data between the virtual machines and storage servers, the processing time of a SaaS component

on a virtual machine on which it is placed, and the total of SaaS workflow critical paths execution time.
We have considered the following SLA constraint to make this placement plan SLA aware.

$$\forall sc_i TET_{sc_i} \leq mrt_{sla}$$

A SaaS must not violate the constraint which says that the total execution time of a SaaS must not exceed the maximum response time agreed in user's SLA.

2.6 Current State of Work

Placing SaaS components onto cloud servers shares similarities with an existing problem known as Component Placement Problem(CPP) (28). CPP is further categorized into two parts:
1) offline CPP, where component are placed at initial stage,
2) online CPP, where component placement is done during run time.
The following table shows the techniques used for SaaS placement by various researchers. The approach used in this research is justified by the prior techniques used for the placement purpose.

Researcher	Techniques	SLA Considered
Kichkaylo et al.(28)	Planning algorithm developed by Artificial Intelligence community	No
Karve et al.(26)	Middleware Clustering technology	No
Zimmerova et al.(53)	Automata language for inter-component communication capturing	No
Zhu et al.(52)	Virtualization technology for ACP	No
Yusoh et al.(48)	Genetic Algorithm with Penalty	No
Zhenzhang Liu et al.(31)	Ant Colony Optimization Algorithm	No
Yusoh et al.(50)	Cooperative Co-evolutionary Genetic Algorithm	Yes (Response time)
Zhipiao et al.(32)	Cost-aware placement of SaaS using genetic algorithm	Yes
Urgaonkar et al. (45)	first-fit approximation algorithm for offline application component placement	No

Table 2.3: Current State of Work for SPP

2.7 Conclusion

In this chapter we have discussed the Cloud computing service models, deployment models,and Cloud SaaS with examples. We have formulated the problem for SaaS placement with all the required assumptions of resources and service level agreement constraints as well.
Sometimes a simple idea works for the optimization problems. Greedy algorithms are the first choice to solve an optimization problem because these approach optimization problems in a short sighted way and tries to get as close as possible to a solution quickly, but does not guarantee to provide an optimal solution. By seeing the current state of SaaS Placement , it is proved that SaaS Placement Problem is the candidate of Particle Swarm Optimization.

Chapter 3

PSO Framework for SaaS Placement

3.1 Introduction

Through Cloud, SaaS models allow software applications to be offered as a service instead of installing on the individual machines. A SaaS can be offered as a set of services, which composes of a group of loosely coupled separate components that communicates each other to form a high-level functional service (24). The placement of SaaS components onto the cloud infrastructure has to be done tactically. One example of SaaS is Google Apps (19), which is offered by Google. There are various service categories of the Google Apps, targeted for different groups. Two different kinds of services are provided through Google Apps, one is the Google Apps for business without the vault, and another is the Google Apps with unlimited storage and vault. Multiple services are offered like Gmail, Hangouts, Calendar, Drive, Docs, and Slides, etc. For both categories, Google put some usage charges, which are different for the different type. Google also offers flexible and annual plan so as to provide user convenience.
Most Cloud and SaaS companies do not reveal their model or architecture details; as such information is regarded as a companys competitive advantage. Several basic assumptions have to be made regarding the placement of SaaS components placement in the Cloud discussed in the previous chapter (5)(50).

In this research, SaaS is considered to be in a composite form or a set of components called as SaaS components (49) (31). Delivering SaaS in such manner instead of atomic SaaS allows resilient offerings of the services. SaaS providers indulge in a numerous benefits by utilizing the SaaS in a composite form. Re-

duced resource costs, because components are reused, flexible offerings of SaaS functions, and reduced subscription cost for clients, are the important benefits. However, it also elevates some challenges for SaaS providers in SaaS management. The most important challenge is to place each of the SaaS components onto Cloud servers. The placement problem of the SaaS components on the VMs, residing on the top of the servers, is known as SaaS Placement Problem (SPP).
SaaS placement problem deals with discovering the ideal solution that is the set of VMs on which the components are placed such that all the components resource requirements are satisfied, response time is meeting the SLA (13) (12) and the solution should minimize the cost incurred to the SaaS provider. A comprehensive research is being carried out for SaaS placement problem; it closely resembles the Component Placement Problem (CPP) (28). Although the existing research do not consider the resource constraints and SLA constraints at the same time. Various existing studies formulated CPP as a resource optimization problem and also as a variant of the multiple knapsack problem (26) (45). Another closely related problem is Task Assignment Problem (TAP) (41) (42), which refers to the problem of assigning a number of tasks to a number of processors available, in such a way that the given objective is minimized or maximized. Hence SPP can also be treated as a multiple knapsack problem because of its nature of many to many mapping. As the CPP, TAP and multiple knapsack problems have been proven as an NP-complete problem (26)(28)(48), and SPP closely resembles these, SPP can be said to an NP-complete problem. NP-complete nature of SPP is proved in previous chapter also. The proposed algorithm, which is an evolutionary approach, finds the sub-optimal solution for the SPP.

3.2 SaaS Placement using Particle Swarm Optimization

In our research Particle Swarm optimization (PSO) is used for solving the SaaS placement problem i.e. formulated in Chapter 2. SPP can be considered as a large-scale and complex combinatorial optimization problem that deals with the allocation of VMs running on Cloud servers to the SaaS components, subjected to a constraints set (45). It is not advisable option to find an optimal solution to the large-scale and complex problems because of its extensive amount of time requirement (10). For finding an ideal or sub-optimal solution, Meta-heuristics are often used for solving such combinatorial problems (11).

Particle swarm optimization (27) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995. PSO is induced by social behaviour of bird flocking or fish schooling. There are various similarities among PSO and other evolutionary computation techniques such as Genetic Algorithms (GA). PSO has been successfully applied to a large number of problems (25), including standard function optimization problems (41), permutation problems, and to the similar kind of problems like Task Allocation Problem and Data Placement in Cloud computing (47) (16). The use of Particle Swarm Optimization is rapidly increasing.

In this approach, the system is initialized with a population of random solutions, known as particles in this case and searches for optima by updating generations (27). However, opposed to GA, PSO has no evolution operators such as crossover and mutation found in GA. In PSO, the probable solutions, known as particles, fly through the problem space by following the current most favorable particles. PSO might sound complex but it is a very simple optimization technique. Comparing with GA, PSO is advantageous because of easier implementation of PSO and very few parameters to adjust unlike GA. In PSO, the information sharing mechanism is considerably different. In Genetic Algorithms, chromosomes share the information with each other. So the complete population moves like one batch towards an optimal area. In PSO, only the global Best (or local Best) gives out the information to others. It is a one-way information sharing mechanism. The evolution only figures for the best solution. In PSO unlike GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

The problem presented in this research is shown to be a combinatorial optimization problem (41). The problem is proved to be an NP-complete problem from computational point of view. Hence, Evolutionary Algorithm (EAs) specifically Particle Swarm Optimization (PSO) (27) is used to solve this problem. PSO is successfully applied to several optimization problems which were similar to this and obtained better solutions (42)(23)(40).

3.2.1 PSO Parameters

PSO imitates the behaviour of bird flocking. Consider the following framework: a group of birds are searching for food in an area randomly. Only one piece of food is there in the area that is being searched by birds. The birds are not informed about where the food is. But in each iteration they know how far the food is. So what's the master plan to find the piece of food?
The powerful strategy to know where the food is to follow the bird which is nearest to the food.

PSO acquired a knowledge from the scenario and used it to solve the optimization problems. In PSO, each single solution is a "bird" in the search space. It is called a "particle". All of the particles have fitness values which are evaluated by the fitness function that needs to be optimized, and have velocities which direct the flying of the particles. The particles fly from one side to the other in the problem space by following the current most favorable particles.
There are not many parameters to be taken care of in PSO, unlike pther nature inspired optimization techniques. (10). The main parameters of PSO are 1) the particle representation 2) the population size, 3) the dimension of particles, 4) the range of particles, 6) velocity, 5) the evaluation function 6) learning factors, and 7) stopping criterion.

- *The particle representation*, is the way to represent a possible solution. For SPP the particle consists of the vm# on which the components are to be placed.
- *The population size*, tells how many particles in population, typically it ranges from 20 to 40. For most of the problems, results are obtained using 10 particles only.

- *The dimension of particles*, is determined by the optimization problem, like number of SaaS components will be the particle size for the proposed method.
- *The range of particles*, is also defined by the optimization problem, different ranges can be specified for different dimension of particles. In this research, the range is the number of virtual machines.
- *velocity*, random value assigned to each particle to move about the cost surface.
- *The evaluation function*, describes the fitness of the function. Deployment cost is the fitness function used here.
- *Learning factors*, usually these are equal to 2. However, other settings are also used for different problems.
- *Stopping criterion*, the maximum number of iterations to obtain the solution. The stop condition also depends on the optimization problem.

3.2.2 SPPSO

The SaaS Placement based on PSO (SPPSO), is a nature inspired heuristic inspired from Biological evolution theory. It starts with a particles' population, where each particle is a possible solution for SaaS components placement on virtual machines. The population is a random matrix, where each row is a particle or position vector or the possible solution. Each particle proceeds about the surface with a velocity (16), which is also initialized randomly. The position vectors (or particles) and velocities are updated based on the local and global best solutions obtained using the following equations:

$$\begin{aligned} Vel_k^{new}(m,n) = Vel_k^{old}(m,n) + c_1 * r_1 * (localbest_k^{old}(m,n) - X_k^{old}(m,n)) \\ +c_2 * r_2 * (globalbest_k^{old}(m,n) - X_k^{old}(m,n)) \end{aligned} \quad (3.2.1)$$

$$X_k^{new}(m,n) = \begin{cases} 1 & \text{if } Vel_k^{new}(m,n) = maxVel_k^{new}(m,n), \\ 0 & \text{otherwise .} \end{cases} \tag{3.2.2}$$

Where,

$Vel_k(m,n) = k^{th}$ particle velocity,

$X_k(m,n) = k^{th}$ particle variable or position vector,

$r_1, r_2 =$ independent uniform random numbers,

$c_1, c_2 =$ learning factors,

$lbest_k(i,j) =$ best local solution, and

$gbest_k(i,j) =$ best global solution.

In this approach the velocity vector is updated for each particle, in every iteration. Velocity updates are affected by both the best global solution associated with the lowest cost ever found by a particle and the best local solution associated with the lowest cost in the present population(39). If cost of the best local solution is less than the cost of the current global solution, then the best local solution replaces the best global solution.

As our problem of placement of SaaS components on the Virtual Machines is proved to be an optimization problem previously, there are some constraints also which needs to be taken care of while optimization. Hence it can be said to a constrained optimization problem (28). For this purpose we have adopted Penalty function method (16), and introduced some penalties. If any constraint is violated by the solution then only a penalty is added for every constraint violation.

$$Penalty_1 = H\left[\left(|M_{vm_j} - MM_{sc_i}|\right)\right] \tag{3.2.3}$$

$$Penalty_2 = H\left[\left(|SS_{vm_j} - S_{sc_i}|\right)\right] \tag{3.2.4}$$

$$Penalty_3 = H\left[\left(|BW_{vm_j} - IO_{sc_i}|\right)\right] \tag{3.2.5}$$

$$TotalPenalty = Penalty_1 + Penalty_2 + Penalty_3 \tag{3.2.6}$$

$Penalty_1$, $Penalty_2$, and $Penalty_3$ are the penalties resulted from the violation of constraints. H is the penalty factor, which is a proper positive number.
Total Penalty, is the sum of all the penalties introduced in our optimization problem. This Penalty is added to the cost calculated in SaaS placement i.e. defined in the previous Chapter.

The SPPSO is proposed for placement of SaaS components. The algorithm, SaaS components Placement using Particle Swarm Optimization (SPPSO), allocates a fixed number of Saas components to available VMs. After a number of iterations the optimal solution is found. Algorithm 3.2.2 describes the SPPSO.

The input to the SPPSO are the SaaS components' requirements and the virtual machines' capacities to allocate each of the component to a virtual machine. In the first step of the SPPSO, a random population of particles and their velocity matrix is initialized, where each particle represent the possible solution.
In the next step $cost_{localbest}$ and $cost_{globalbest}$ is initialized to ∞, where $cost_{localbest}$ is the local best of a particle obtained so far, and the $cost_{globalbest}$ is the global cost i.e. best among all of the particle so far. As our objective is to minimize the total cost, so lesser the cost is, the corresponding placement solution is considered to be the best.

The **for** loop of lines 4-16 is used to calculate the cost of each particle using the function defined in Chapter 2, the cost is considered as the fitness of a solution. If a particle or the solution satisfies the resource constraints and SLA constraint as well the cost is calculated simply but if any of the constraint is violated then calculation of the cost is done with penalty. After finding the cost of each particle, local best and the global best is calculated.

After obtaining the local best and the global best, the **for** loop of lines 17-19 is used to update the velocity and the particle or the solution based on the local and global best obtained from previous for loop.

Algorithm 1 SPPSO

Input: SCs and VMs with their requirements and capacities respectively

Output: Sub-optimal solution for placement of SaaS

```
1:  Initialize Population (particles)
2:  Initialize $cost_{localbest} \leftarrow \infty$ and $cost_{globalbest} \leftarrow \infty$
3:  repeat
4:    for each particle $i = 1, ..., P$ do
5:      if sc resource requirements $\leq$ vm capacity and sc can be placed on vm
        then
6:        Calculate cost for the particle
7:      else
8:        Calculate cost with penalty for the particle
9:      end if
10:     if $cost_{X_i} < cost_{localbest_i}$ then
11:       $localbest_i \leftarrow X_i$
12:     end if
13:     if $cost_{localbest_i} < cost_{globalbest_i}$ then
14:       $globalbest_i \leftarrow localbest_i$
15:     end if
16:   end for
17:   for each particle $i = 1, ..., P$ do
18:     Update velocity and particles or position vectors using Equations 3.2.1
        & 3.2.2
19:   end for
20: until maximum iterations reached
21: return $cost_{globalbest_i}$ and $X_i$
```

Lines 4-19 are repeated until the defined maximum number of iterations are performed in order to achieve the optimal solution.

After all the iterations are performed, most of the particles converge to a single solution and that is minimum among all. The corresponding particle to the minimum cost after completion of the iterations is the best placement solution obtained. SPPSO returns the minimum cot and the corresponding particle.

3.2.3 Working of SPPSO

SPPSO is an approach that emulates the natural evolution process for placing the SaaS components onto VMs. The working of SPPSO can be easily understood with the figure 3.1.

After initializing the particles, that are the possible solutions for SaaS compo-

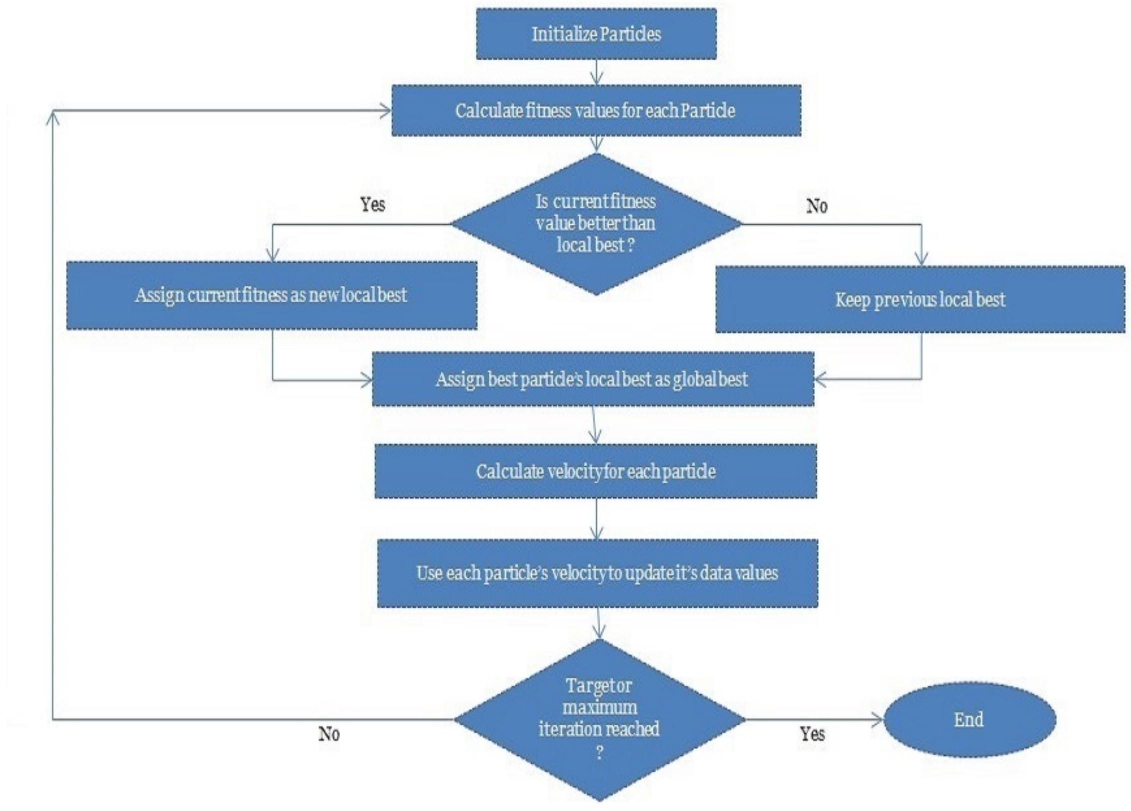
Initialize Particles
Calculate fitness values for each Particle
Yes
Is current fitness value better than local best ?
No
Assign current fitness as new local best
Keep previous local best
Assign best particle's local best as global best
Calculate velocity for each particle
Use each particle's velocity to update it's data values
Target or maximum iteration reached ?
No
Yes
End

Figure 3.1: Flow chart of SPPSO

nents placement, fitness value is calculated for each based on the optimization function defined in Chapter 2 and penalty using equation 3.2.6, then this value is compared against the local best obtained so far for the particle and updated accordingly. Then each particle's velocity is calculated based on the equations 3.2.1 and then the particle or the solution is updated based on the equation 3.2.2. This process is followed for the number of iterations defined so as to find the optimal placement solution for the SaaS components onto the Virtual Machines.

3.2.4 Example

Assume we have 5 VMs on which 6 SaaS components are to be placed. Initially we have initialized a population of particles or position vector randomly. One position vector (or a particle) can be shown as in table 3.2.4.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
$vm\#$	3	2	1	5	4	2

Table 3.1: Direct representation of a position vector

Each column of the position matrix represents a component placement and each of the row represents the placed component on a VM. The position matrix is converted to a matrix of size *p* x *q*, where *p* is the number of VMs and *q* is the number of components to be placed. The equivalent indirect representation of matrix table 3.2.4 is given as in table 3.2.4.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
vm_1	0	0	1	0	0	0
vm_2	0	1	0	0	0	1
vm_3	1	0	0	0	0	0
vm_4	0	0	0	0	1	0
vm_5	0	0	0	1	0	0

Table 3.2: Indirect representation of a position vector

Every particle is initialized with a random velocity initially. This velocity is also a matrix of size p x q. With this velocity the particle tries to find the optimal solution or improvised itself in each iteration. The velocity of the particle is represented by the table 3.2.4.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
vm_1	5	-2	5	1	-3	7
vm_2	6	3	0	4	-2	5
vm_3	-2	6	1	3	1	-3
vm_4	1	7	4	2	4	2
vm_5	4	8	2	-5	2	4

Table 3.3: Velocity matrix

Local and global solutions are obtained from the initial population of particles using the Equation 3.2.1.

On the basis of obtained local and global solutions velocity matrix is updated. Let we have the updated velocity matrix 3.2.4 for the particle which has the position matrix 3.2.4.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
vm_1	4	6	9	1	6	5
vm_2	-2	3	1	8	5	4
vm_3	5	-6	0	3	2	7
vm_4	1	5	4	-4	7	-2
vm_5	0	4	7	6	8	3

Table 3.4: Updated velocity matrix

Based on the updated velocity matrix shown 3.2.4, corresponding position vector is also updated using Equation 3.2.2, and can be shown as matrix 3.2.4.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
vm_1	0	1	1	0	0	0
vm_2	0	0	0	1	0	0
vm_3	1	0	0	0	0	1
vm_4	0	0	0	0	0	0
vm_5	0	0	0	0	1	0

Table 3.5: Updated position vector

The updated position vector is the indirect representation, it can be converted back to the direct representation.

	sc_1	sc_2	sc_3	sc_4	sc_5	sc_6
$vm\#$	3	1	1	2	5	3

Table 3.6: Direct representation of updated position vector

This example shows how a particle or the position vector for the SaaS placement is updated based on the velocity. In every iteration a particle, which is a possible solution improvise itself to find an optimal result.

3.3 SaaS Placement using Greedy Approach

Greedy algorithms are used as a first choice to solve optimization problem because of the nature of algorithm to find the solution very quickly. The Algorithm 3.3 presents greedy algorithm that uses first come first serve as the heuristic for placement of SaaS components on VMs. This algorithm allocates the SaaS components to the VMs. Algorithm terminates with a fixed set of iterations.

Algorithm 2 SPGA

```
Input: SCs and VMs with their requirements and capacities respectively
Output: Sub-optimal solution for placement of SaaS
 1: Sort SC in descending order based on its processing requirement, $SC_{sort}$
 2: for $sc \in SC_{sort}$ do
 3:    for $vm \in VM$ do
 4:       if sc resource requirements $\leq$ vm capacity and sc can be placed on vm
          then
 5:          Calculate $ET_{sc}$
 6:          if $ET_{sc} < mrt_{sla}$ then
 7:             $P(sc) \rightarrow vm$
 8:             break
 9:          end if
10:       end if
11:    end for
12:    if $P(sc) \rightarrow vm$ then
13:       Update vm
14:    end if
15: end for
16: Calculate cost for the placement of components.
17: return Cost and corresponding placement solution.
```

SPGA starts with the SaaS components' requirements and Virtual Machines' capacities as inputs to the algorithm. In the first step, SaaS components are sorted based on the processing requirements of the components. For each of the SaaS components we need to find a virtual machine, on which it can be placed with satisfying resource constraints and it must not violate SLA constraint i.e. the execution time of the component should not exceed the maximum response time defined in SLA. The **for** loop of lines 2-15 finds the placement of each SaaS component or the virtual machine on which a component is placed. For each of the SaaS component the **for** loop of lines 3-11 is run. If the a vm satisfies the component requirements and it can be placed on that particular virtual machine then ET_{sc} is calculated. After calculation of ET_{sc}, in next step it is checked

whether ET_{sc} is less than mrt_{sla}, if the condition is found to be true the component is placed on the virtual machine. Same process is repeated for all of the components.
After all the components are placed on some virtual machines, the cost of deployment is computed. SPGA returns the cost and the placement solution obtained.

3.4 Simulation Results

Experiments have been conducted using inhouse simulator to study the performance of algorithms 3.2.2, 3.3 by varying number of VMs and SaaS components. The proposed approach **SPPSO** was compared against the Greedy approach **SPGA** for different number of SaaS components.

Assumptions(31): Some basic assumptions made for performance evaluation of the proposed algorithm are as follows:
For Virtual Machines capacities:
PC_{vm_x}= 1 to 10 Gbps,
MM_{vm_x}=1000 to 20000 B,
SS_{VM_x}= 20 to 2000 MB,and
IOC_{vm_x}= 10 Mbps

For SaaS Components requirements:
TS_{sc_i} = 20 to 200 MB,
M_{sc_i} = 1000 to 10000 B,
S_{sc_i} = 10 to 100 MB,
and IO_{sc_i} = 100 to 200 MB.

3.4.1 Performance Evaluation with Different Number of Virtual Machines

The experiment run with varying number of SaaS components on different number of virtual machines. The population size for the SPPSO algorithm is 30 and the algorithm is run for 200 iterations. Figure 3.2, Figure 3.3, and Figure 3.4 illustrates the cost incurred to the SaaS provider by both SPPSO and SPGA algorithms. The comparison is made based on the calculated cost using objective function.

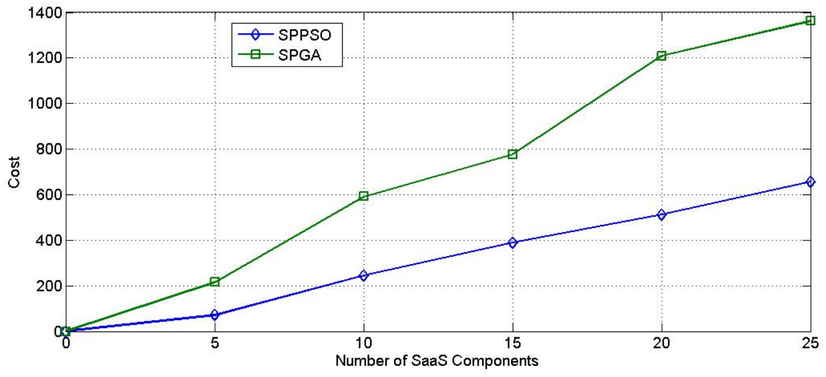

Figure 3.2: Experiment on number of SaaS Components [for 100 VMs)]

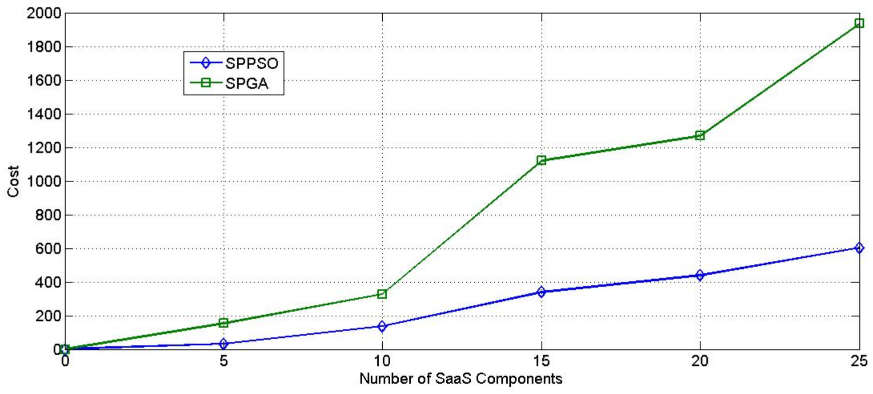

Figure 3.3: Experiment on number of SaaS Components [for 200 VMs]

Due to being stochastic nature of SPPSO experiments repeated several times. It can be observed from the graphs in Figure 3.2, 3.3, and 3.4, the SPPSO has always a lower cost value than SPGA, which implies SPPSO gives a better placement option for SaaS components placement.

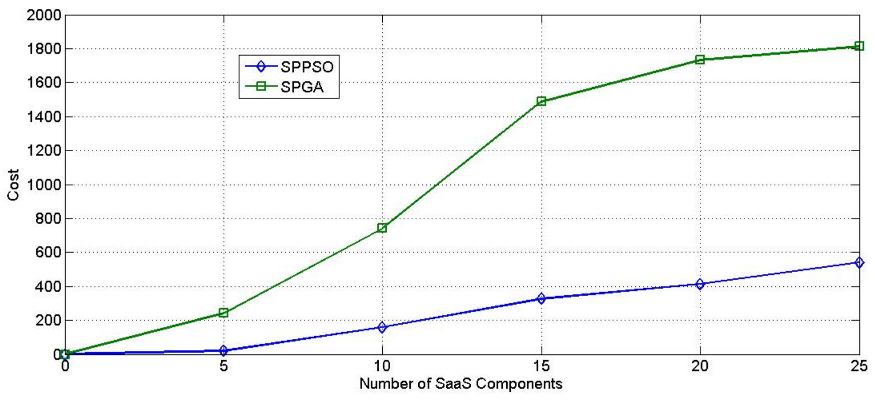

Figure 3.4: Experiment on number of SaaS Components [for 300 VMs]

3.4.2 Performance Evaluation with Different Number of SaaS Components

The experiments run for different number of SaaS components on variable number of virtual machines. The population size for the SPPSO algorithm is 30 and the algorithm is run for 200 iterations. Figure 3.5, Figure 3.6, and Figure 3.7 illustrates the cost incurred to the SaaS provider by both SPPSO and SPGA algorithms. The comparison is made based on the calculated cost using objective function.

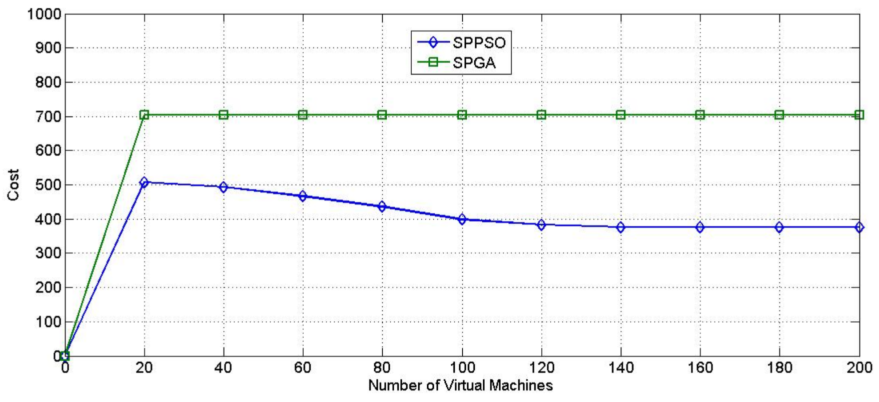

Figure 3.5: Experiment on number of VMs [for 20 SaaS Components]

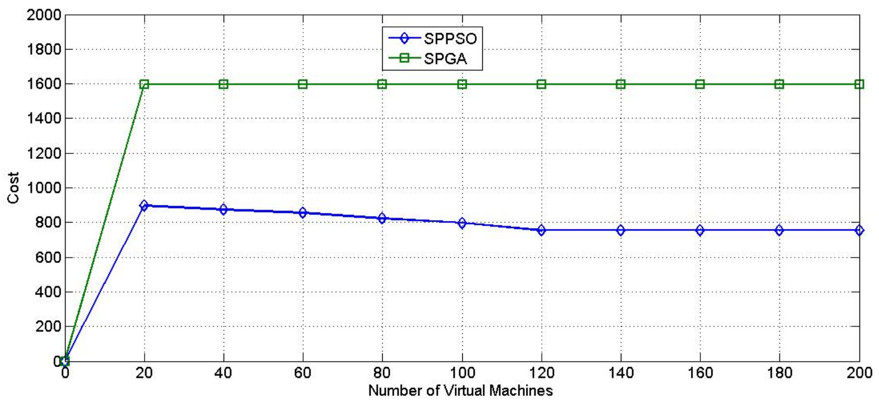

Figure 3.6: Experiment on number of VMs [for 30 SaaS Components]

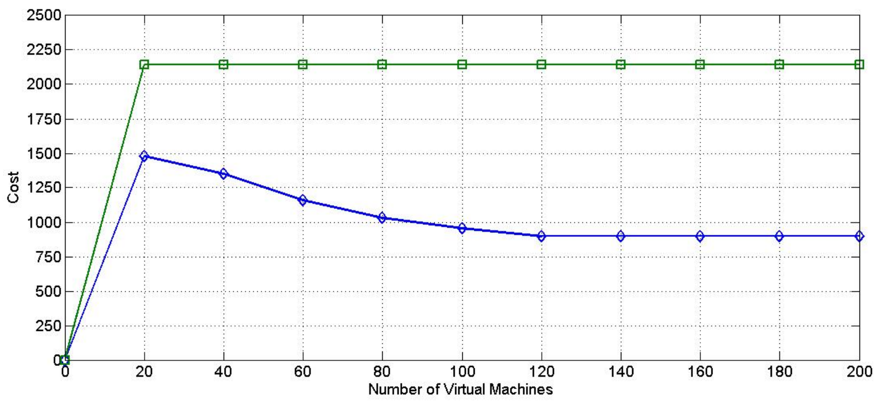

Figure 3.7: Experiment on number of VMs [for 40 SaaS Components]

Due to being stochastic nature of SPPSO experiments repeated several times. It can be observed from the graphs in Figure 3.5, 3.6, and 3.7, the SPPSO has always a lower cost value than SPGA, which implies SPPSO gives a better placement option for SaaS components placement. Deployment cost of SaaS components can be minimized by using SPPSO.

3.5 Conclusion

This chapter presented the techniques for SaaS components placement in Cloud. New challenges and constraints are introduced for SPP. To deal with these challenges, a nature inspired optimization approach SPPSO is proposed for SPP. The performance of SPPSO is evaluated by number of experiments performed, with different scenarios, variable number of virtual machines and SaaS components. The SPPSO performance is than compared against SPGA, which is first come first serve greedy approach. The evaluation is done so as not to violate the resource and SLA constraints. The obtained results shows that the proposed heuristic SPPSO outperforms SPGA in all set of experiments.

Chapter 4

Conclusions & Future Work

In this thesis work, we have discussed about different placement strategies for Software as a Service, proposed by various researchers which mainly considers the resource constraints i.e. processing capability, main memory and storage. We have also considered Service Level Agreement constraint i.e. response time. This research is focused on computing the cost of placement of SaaS components in the Cloud, while not violating Service level agreement constraint as well as resource constraints, which makes this placement SLA aware. In our research, a detailed framework for PSO implementation has been presented. The performance of the proposed algorithm SaaS components Placement using Particle Swarm Optimization (SPPSO) is compared with Greedy based SaaS components placement,SPGA. Simulation experiments conducted shows in favor of SPPSO.

SaaS placement problem being an intractable problem some more investigation is required using the other Nature inspired meta-heuristic techniques like *Firefly technique* and *Bat technique.* Future work can include the issue of scalability, some more constraints for placing SaaS components like: number of cores, number of processors etc.

Bibliography

[1] Abreu, E., "Down on the server farm," *The Industry Standard*, vol. 4, no. 7, p. 4, 2008.

[2] Aymerich, F. M., Fenu, G., and Surcis, S., "An approach to a cloud computing network," in *Applications of Digital Information and Web Technologies, 2008. ICADIWT 2008. First International Conference on the*, pp. 113–118, IEEE, 2008.

[3] Bedin, W. and Moinuddin, M., "An overview of software as a service in retail," 2007.

[4] Borovick, L. and Mehra, R., "Architectine the network for the cloud [white paper]." `http://www.idc.com`, 2011.

[5] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I., "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[6] Candan, K. S., Li, W.-S., Phan, T., and Zhou, M., "At the frontiers of information and software as services," in *New Frontiers in Information and Software as Services*, pp. 283–300, Springer, 2011.

[7] Carolan, J., Gaede, S., Baty, J., Brunette, G., Licht, A., Remmell, J., Tucker, L., and Weise, J., "Introduction to cloud computing architecture," *White Paper, 1st edn. Sun Micro Systems Inc*, 2009.

[8] Chong, F. and Carraro, G., "Architecture strategies for catching the long tail," *MSDN Library, Microsoft Corporation*, pp. 9–10, 2006.

[9] Dubey, A. and Wagle, D., "Delivering software as a service," *The McKinsey Quarterly*, vol. 6, no. 2007, p. 2007, 2007.

[10] Eberhart, R. C. and Kennedy, J., "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1, pp. 39–43, New York, NY, 1995.

[11] Eberhart, R. C. and Shi, Y., "Particle swarm optimization: developments, applications and resources," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1, pp. 81–86, IEEE, 2001.

[12] Emeakaroha, V. C., Brandic, I., Maurer, M., and Breskovic, I., "Sla-aware application deployment and resource allocation in clouds," in *Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual*, pp. 298–303, IEEE, 2011.

[13] Farokhi, S., "Towards an sla-based service allocation in multi-cloud environments," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, pp. 591–594, IEEE, 2014.

[14] Foster, I., Zhao, Y., Raicu, I., and Lu, S., "Cloud computing and grid computing 360-degree compared," in *Grid Computing Environments Workshop, 2008. GCE'08*, pp. 1–10, Ieee, 2008.

[15] Furht, B., "Cloud computing fundamentals," in *Handbook of cloud computing*, pp. 3–19, Springer, 2010.

[16] He, F., "An improved particle swarm optimization for knapsack problem," in *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pp. 1–4, IEEE, 2009.

[17] Hudli, A. V., Shivaradhya, B., and Hudli, R. V., "Level-4 saas applications for healthcare industry," in *Proceedings of the 2nd bangalore annual compute conference*, p. 19, ACM, 2009.

[18] Inc., A., "Amazon elastic compute cloud." `http://aws.amazon.com.ec2`, 2015.

[19] INC, G., "Googleapps for business." `http://www.google.com/apps/intl/en/business/index.html`, 2015.

[20] INC., S., "Salseforce crm.." `http://www.salesforce.com/in/service-cloud/overview/`.

[21] INC., S. M., "Open source cloud computing: On-demand, innovative it on a massive scale," 2009.

[22] INFOTECH, L., "Difference betweeb the asp model and the saas model," *URL http://www. luitinfotech. com/kc/saas-asp-difference. pdf*, 2013.

[23] IZAKIAN, H., LADANI, B. T., ABRAHAM, A., and SNASEL, V., "A discrete particle swarm optimization approach for grid job scheduling," *International Journal of Innovative Computing, Information and Control*, vol. 6, no. 9, pp. 1–15, 2010.

[24] KANG, S., MYUNG, J., YEON, J., HA, S.-W., CHO, T., CHUNG, J.-M., and LEE, S.-G., "A general maturity model and reference architecture for saas service," in *Database Systems for Advanced Applications*, pp. 337–346, Springer, 2010.

[25] KAO, C.-C., "Applications of particle swarm optimization in mechanical design," *J Gaoyuan Univ*, vol. 15, pp. 93–116, 2009.

[26] KARVE, A., KIMBREL, T., PACIFICI, G., SPREITZER, M., STEINDER, M., SVIRIDENKO, M., and TANTAWI, A., "Dynamic placement for clustered web applications," in *Proceedings of the 15th international conference on World Wide Web*, pp. 595–604, ACM, 2006.

[27] KENNEDY, J., "Particle swarm optimization," in *Encyclopedia of Machine Learning*, pp. 760–766, Springer, 2010.

[28] KICHKAYLO, T., IVAN, A., and KARAMCHETI, V., "Constrained component deployment in wide-area networks using ai planning techniques," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, pp. 10–pp, IEEE, 2003.

[29] Kumar, A., "Placement of software-as-a-service components in cloud computing environment," 2014.

[30] Liu, F., Tong, J., Mao, J., Bohn, R., Messina, J., Badger, L., and Leaf, D., "Nist cloud computing reference architecture," *NIST special publication*, vol. 500, p. 292, 2011.

[31] Liu, Z., Hu, Z., and Jonepun, L. K., "Research on composite saas placement problem based on ant colony optimization algorithm with performance matching degree strategy.," *Journal of Digital Information Management*, vol. 12, no. 4, p. 225, 2014.

[32] Liu, Z., Wang, S., Sun, Q., Zou, H., and Yang, F., "Cost-aware cloud service request scheduling for saas providers," *The Computer Journal*, p. bxt009, 2013.

[33] Mahowald, R. P., "Worldwide software as a service 2010-2014 forecast: software will never be the same," *IDC Report*, no. 223628, 2010.

[34] McHall, T., "Gartner says worldwide software as a service revenue is forecast to grow 21 percent in 2011," *Gartner. com. Gartner. Retrieved*, vol. 28, 2011.

[35] Mell, P. and Grance, T., "The nist definition of cloud computing," 2011.

[36] Motahari-Nezhad, H. R., Stephenson, B., and Singhal, S., "Outsourcing business to cloud computing services: Opportunities and challenges," *IEEE Internet Computing*, vol. 10, 2009.

[37] Natis, Y. V., "Introducing saas-enabled application platforms: Features, roles and futures," *Gartner Inc*, 2007.

[38] Patel, P., Ranabahu, A. H., and Sheth, A. P., "Service level agreement in cloud computing," 2009.

[39] Poli, R., Kennedy, J., and Blackwell, T., "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[40] Rapaić, M. R., Kanović, Ž., and Jeličić, Z. D., "Discrete particle swarm optimization algorithm for solving optimal sensor deployment problem," *Journal of Automatic Control*, vol. 18, no. 1, pp. 9–14, 2008.

[41] Rosendo, M. and Pozo, A., "Applying a discrete particle swarm optimization algorithm to combinatorial problems," in *Neural Networks (SBRN), 2010 Eleventh Brazilian Symposium on*, pp. 235–240, IEEE, 2010.

[42] Salman, A., Ahmad, I., and Al-Madani, S., "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002.

[43] Silberschatz, A., Galvin, P. B., Gagne, G., and Silberschatz, A., *Operating system concepts*, vol. 4. Addison-Wesley Reading, 1998.

[44] Tsai, W.-T., Sun, X., and Balasooriya, J., "Service-oriented cloud computing architecture," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pp. 684–689, IEEE, 2010.

[45] Urgaonkar, B., Rosenberg, A. L., and Shenoy, P., "Application placement on a cluster of servers," *International Journal of Foundations of Computer Science*, vol. 18, no. 05, pp. 1023–1041, 2007.

[46] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M., "A break in the clouds: towards a cloud definition," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2008.

[47] Yin, P.-Y., Yu, S.-S., Wang, P.-P., and Wang, Y.-T., "Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization," *Journal of Systems and Software*, vol. 80, no. 5, pp. 724–735, 2007.

[48] Yusoh, Z. I. M. and Tang, M., "A penalty-based genetic algorithm for the composite saas placement problem in the cloud," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pp. 1–8, IEEE, 2010.

[49] Yusoh, Z. I. M. and Tang, M., "Clustering composite saas components in cloud computing using a grouping genetic algorithm," in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pp. 1–8, IEEE, 2012.

[50] Yusoh, Z. I. M. and Tang, M., "Composite saas placement and resource optimization in cloud computing using evolutionary algorithms," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pp. 590–597, IEEE, 2012.

[51] Zhou, M., Zhang, R., Zeng, D., and Qian, W., "Services in the cloud computing era: A survey," in *Universal Communication Symposium (IUCS), 2010 4th International*, pp. 40–46, IEEE, 2010.

[52] Zhu, X., Santos, C., Beyer, D., Ward, J., and Singhal, S., "Automated application component placement in data centers using mathematical programming," *International Journal of Network Management*, vol. 18, no. 6, pp. 467–483, 2008.

[53] Zimmerova, B. and others, "Component placement in distributed environment wrt component interaction," in *Proceedings of the 2nd Doctoral Workshop on Mathematical and Engineering Methods in Computer Science*, pp. 260–267, 2006.

Dissemination of Work

Published

Sumit Bhardwaj and Bibhudatta Sahoo, “A Particle Swarm Optimization Approach for Cost Effective SaaS Placement on Cloud”, in *Proceedings of the IEEE International Conference on Computing, Communication and Automation 2015*, pp. 571-575, IEEE, 2015.