International Journal of Cloud Computing and Services Science (IJ-CLOSER)
Vol.2, No.2, April 2013, pp. 148~158
ISSN: 2089-3337

A Distributed Service and Business Model for Providing Cloud Computing Service

Shreya Bhadra*, Tirthankar Gayen**
* School of Computer Engineering, KIIT University
** School of Computer Engineering, KIIT University

Article Info

Article history:

Received Jan 5th, 2013
Revised Jan 10th, 2013
Accepted 30th, 2013

Keyword:

PaaS
Cloud Computing
Compilation
Software Piracy

ABSTRACT

Cloud computing model has enabled IT organizations to serve the users globally. It gives the services like Platform as a Service, Software as a Service and Infrastructure as a Service without users being much aware of the details in which the services are provided. As in File Access System Service the users are not aware of the locations of the files in Clouds. File access seems to them as a single coherent file system. Considering this aspect, this paper is concerned with an effective service and cost model for providing Cloud Computing service for writing and compiling source code remotely from any hand held device using the distributed service model. According to this model, for providing a service, the service provider may subsequently use the services of other service providers in the Cloud without the awareness of the client. The cost model effectively evaluates the cost which each client pays to each of its service providers for receiving the services. The model is expected to be mutually beneficial to the client and its service provider. The client is paying only for the service it receives and the service provider only provides the service that is requested by the client .The cost is evaluated based on the kind of service provided. The model is also expected to reduce software piracy to a considerable extent.

Corresponding Author:

Tirthankar Gayen,
School of Computer Engineering,
KIIT University
Email: tgayen77@gmail.com

1. INTRODUCTION

Cloud Computing is a service model where services are providing to the end users with flexible and scalable services available through the internet [1]. The service can be a software, hardware, infrastructure or platform. This is a device independent model because it resources can be accessed not just from any computer on the internet, but also any type of machines, provided that it has an internet connection and web browser. Many private individuals now regularly use an online e-mail application such as Gmail, Yahoo! Mail or Hotmail. Face book is a common social networking site. However, these type of cloud computing are just at the beginning stage. Amazon web service is one of the early implementation of public cloud. Amazon's simple storage service or S3 is a great way to stores data on the cloud that can be access to internet [2]. Others like Google Apps an online e-mail and office suite.

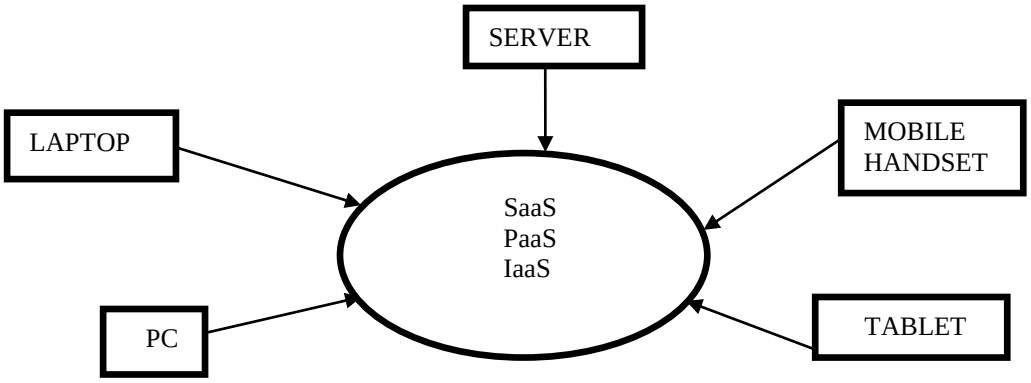

Figure 1. Cloud service users requesting for service

1.1 Characteristics of cloud

As per the Yashpalsing et al.[1] the characteristics of the Cloud model

- On demand and self service are some of the characteristics of Cloud model. The services can be obtained without human intervention and the services are provided by simple logging through user account.
- Pay-per-use character makes it popular in organizations which needs short period services rather than long term or permanent service. Therefore, one pays only when one uses the service..
- Device and location independency makes it accessible from different location and different machines.
- Rapid elasticity in Cloud Infrastructure denotes capabilities to provide new services to consumer with different platform specifications. The Cloud models are adaptable to add more hardware and functionalities.

1.2 Service Models

- Software as a service

SaaS has the capability to deliver the applications over internet which is installed in CLOUD infrastructure. The consumer is not concerned with the software installation details, cloud storage or maintenance of the software. It only sends request for service through internet and gets the application on pay-as-use basis. SaaS helps to improve the software performance through mixing and matching components from many vendors.

- Platform as a Service (PaaS)

PaaS is a service which can be used in software development phase. Software developer can write, check and compile their codes using PaaS. Several concurrent users can also use the same platform to develop software from different locations.

- Infrastructure as a Service (IaaS)

IaaS is approach to provide the equipment (server, hard drive etc) via internet where user can install their own software to use. IaaS is significantly used for starting a business which doesn't have much capital to invest for hard drive. It makes sense for organizations to testing or trial their software for temporary basis. It provides scalability to the organizations which are rapidly growing and adding hardware with itself is problematic.

1.3 Deployment Models

The Cloud deployment models can be categorised into the following categories:

1. Private Cloud: This type of infrastructure is mainly used by the single organization with multiple consumers. It is managed by the organization or a trusted third party. It stores credential data, applications or mechanisms of the organization. It exists within its premises.
2. Public Cloud: The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.

3. Community Cloud: The Cloud model is with specific consumer which has some common work (mission, policy, security requirements). It is managed by the small unit of the community. It may exist on or off premises.
4. Hybrid Cloud: This infrastructure is combination of different Cloud infrastructure (private, community, public) .They are combined for giving portability to maintain their applications or to balance loads within them.

1.4 Comparison between Cloud and Grid

According to Ian Foster et al. [3] both Cloud and Grid have same goal. Reducing the cost of computation, increase the reliability and increase flexibility of the programs are the main goals of the two models. The structure of Grid and Cloud is different. Grid is collection of heterogeneous storage systems. It comprises of dynamic platforms of different organizations. In contrast to Grid, Cloud is a collection of homogenous storage systems. In cloud computing, the users do not compute locally, but on some facilities operated by a third party. As per business model, in cloud the customer will pay on consumption basis. Grids are found in project oriented organizations where each user or community have certain number of groups. As grid computing infrastructure is built from different resources, interoperability and security are the main concerns. Cloud computing is a pool of computing or storage resources. Virtualization has become the core concept of every cloud. Virtualization gives the multi-tenant facility to Cloud. In Grid, the virtualization is not of a concern as in Cloud .Nimbus provide virtual workspace in Grid. Grid structure needs special type of security mechanisms for heterogeneous environment.

2. RELATED WORK

In this section various technical aspects concerned with providing Cloud Computing Services are discussed. In 2011 Smith et al. [4] proposed a file system, Personal Cloud File System which has overcome the problems like network failure, remote machine displacement or disconnections. Personal Cloud File System supports offline operations. It creates a system through which the personal or private Cloud facility can be obtained. User can read and write data from cloud without having to know or care about the locations. It is userspace filesystem which helps to manage files which dispersed across multiple machines. It access files by using FTP OR SFTP protocol. Personal Cloud Filesystem based on single client multiple server models. It does not have dedicated server. It has three major components Userspace Filesystem, List Management and Cache Management**.** Userspace cloud filesystem act as a file manager. It provides location transparency to the users. It mounting and unmounting the files and notify the user of any kind of events on the files. List management is to store the name of the each folder or file. List management collects the information of remote machine through network module. It finds the remote machine where the file resides and connects to the user. Cache management manages the cache. When a user wants to edit a file, the Personal Cloud Filesystem files are transferred from remote machine to local machine before editing. The unused file is discarded according to LRU replacement policy. Whiteout is a type of directory where the deleted files are kept. The model deletes only the local files (cached files). The files marked for deletion are kept in whiteout and are hidden file from user. The model also uses unification filesystem mechanisms to resolve conflict of same naming confliction. Its main limitation is that it does not include any security mechanism which a cloud model needs. This filesystem is generally more suited where read is mainly concerned.

Zeeshan et al.(2010) [5] proposed architecture where the SaaS is delivered to multiple users at the same quantum of time, called Multi-Tenant, Secure, Load Disseminated SaaS Architecture (MSLD). MSLD is made to consider load balancing and security among the clouds. Main concern of MSLD is realized to make the service as light as possible while delivering the service. It supports multitenant architecture. The main components of MSLD on the basis of service functionality are Responder Service, Routing Service, Security Service, Logging Service and Service Realization. The Responder Service layer is worked as a first step entry to MSLD from different types of machine. To handle diverse type of request it introduces a Response Generation Component. It creates an abstraction to developer that they don't need to write different type of code for different service consumer. Routing Service layer validates the request and reduces working load of cloud by forwarding request to the Realization Service where different type of services are provided. The Security service checks whether the request are valid or not. MSLD sets two types of security levels (level-1 and level-2). The first one deals with sessions which have been active for a while for a particular user. The level-2 security deals with newly created session when the user logs into the system for the first time to create a valid session. The limitation of MSLD is that it focuses on overall service interaction between service consumer and provider. It is also very time consuming process for receiving service.

In 2012 Muhammad et al. proposed [6] an autonomous cloud computing model which dynamically configured and reorganized the services interaction. Unpredictable events such as crashes, network

disconnection and service unavailability will typically cause service unavailability. The cloud model should able to recover such type of failure with itself with little human intervention. The model has been implemented in real world system. The technical issues in this model are that the self organizing elements will manage its behavior, internal services and relationships with other elements. The elements are Monitor, Analysis, Plan and Execute. Monitor is managed by Manager. Manager monitors the behavior of the overall system. In addition to this, the availability of services, addition of services, request of services and request/query from user are also monitored. If there is a change on the service model it will keep as a knowledge base. The manager will retrieve previous cases from the knowledge base and analyse the service name, type, providers name etc. In planning module, the Manager plans how services are provided to the users. It retrieves the records from analysis module and using the details it provides its services to the user. Executing phase execute the service request coming from the service broker. Service broker negotiate between the service provider and consumer. Broker translates the requirement of the service the consumer messages from the formal messaging. The limitation of their work is that it gives only the overview regarding how to design the model. No specific application design has been mentioned. Dynamic model should able to configure itself to provide service to user. It must be robust, flexible and autonomous in nature. The details of implementation of this feature are not mentioned.

Kefei et al. [7] they proposed in 2012, a network data collection program based on Android platform. Android network data collection tool is equipment which can monitor the data packets in and out of the network equipment of Android network. The packets are filtered and after filtering the packets are saved on SD card by file style. This mechanism can be applied on any network management. The network device contains four layers which are Network Device Layer, Capture Layer, Filtering Layer and Submission Layer. Network device layer set itself such that it captures all the packets which are coming to the network. Capture module collects the packets and copy the packets from the network device driver buffer to the user mode process space. Packet filtering module captures the packets according to the rules defined. Submission layer submit the packet to the application module. The drawback of this model is that it needs a good packet filter rule to detect the malicious packet. According to technical advancement the intrusion may be from other ways. It focuses only on capturing file and analysis of the packets. In 2003, Hsiu-Hui-Lee et al. [8] proposed a model where the message can be sent with dynamic IP address scheme. In the Internet environment the computer's IP address are not fixed. Java message service needs static IP. To apply the java message service in such environment the proposed model is suitable. The model is able to support for heavy message service. According to this model the servers are divided into MainServer, GateServer and MiniGateServer. MainServer act as a gateway. GateServer is storage for destinations shared by multiple MainServer. It also acts as a router for requests and messages. And the MiniGateServer is responsible for load balancing of GateServer. MiniGateServer can dynamically attach to a GateServer. Thus clients get messages from different network machines. But, there is no mention as to whether this model can be applied to Cloud.

In 2012, Yanfei Li et al. [9] built online power metering procedure which can calculate the cost for using virtual machines over cloud environment. It calculates the cost in online mode. The CPU, memory and disk has been chosen for performance metrics. This model is capable of calculating cost for homogeneous virtual machines. The model also meets the high accuracy and simplicity to achieve online power consumption metering for virtual machine. It needs little amount of time to calculate the cost. But, its limitation is that it is confined to enterprise servers which means small cloud environment. The model is applicable only where CPU, memory and disk is considered for power consumption. Wonil Kim et al. [10] in 2012, proposed a system called ISMS (Integrated software Management System). A main master table and other different tables for different task are maintained in the system to keep record of user authority information and software. ISMS authenticates the software user after consulting with the master tables. In their proposed model the files and packages are sent from one service provider to another service provider on a need basis. The main drawback of this system is that it uses only one centralized service provider to provide SaaS and nothing is mentioned for providing PaaS. As a result, it sometimes becomes very expensive and difficult to use it in various operating environments. Since the service is restricted to one centralized service provider, there is no scope to use the service of other service providers when the requested service provider do not possess adequate resources for providing the service. Although, Young Choon Lee et al. proposed two scheduling algorithm [11] *maxutil* algorithm and *maxprofit* algorithm to schedule the service requests and prioritizing the data access in the cloud with the main objective of maximizing the profit, there is little focus on the consumers benefits. Since, there is no interaction between the service providers (corresponding to various vendors), if the service requested requires some resource which is unavailable to the requested service provider (corresponding to a specific vendor) then there is no scope for this service provider to make a request to another service provider (possessing the required resource) for providing the resource. Rather, it seems to be a headache of the consumer to take care of such situations. In 2013, Slaven Brumec et al. [12] made a cost comparison of using commercial cloud model and local resources based on

the performance parameters like response times, number of computer which is used to do jobs etc. But, they did not focus on providing a cost effective Cloud computing service.

Xiaoyu et al. proposed different brokering [13] policies for different layer resource sharing. Another work on brokering for resource failures while executing a job in cloud it may incurs huge amount of loss both the service provider and consumer. Bahman Javadi et al. proposed [14] various type of brokering strategies for hybrid cloud system for failure handling. The involvement of brokers instead of the service providers involves the incurrence of a considerable amount of expenses for paying the brokers. In Lianyong Qi et al. [15] a QoS-aware composition method was investigated for supporting cross-platform service invocation in cloud environment. A decision-making method was discussed to determine whether a QoS-aware WSC problem has a QoS qualified composite solution; and subsequently, a QoS-aware service composition method named LOEM was put forward, aiming at improving the composition efficiency when a large number of composite solutions are available in the Cloud environment. But, there exist several shortcomings in determining whether a QoS-aware WSC problem has a QoS qualified composite solution is essentially a 0–1 CSP, whose time complexity is exponential, so only partial QoS-aware WSC problems could be determined in polynomial time by their proposed decision-making method. The obtained composite solution may not be the QoS-optimal one. Besides, the time complexity of their proposed LOEM is still exponential, which usually cannot deliver a satisfactory result if an end user requires a real time response.

3. OBJECTIVE

Considering various approaches present in the literature till date it is found that there exists no suitable service and cost model for providing distributed cloud computing service. Hence, it is the intention to develop an approach to build a versatile and effective service and cost model for providing cloud computing service for compiling source code remotely from any hand held device (having limited resources) using the distributed service model. According to this model, for providing a service, the service provider may subsequently use the services of other service providers in the cloud without the awareness of the client. The cost model effectively evaluates the cost which each client pays to each of its service providers for receiving the services. The approach will provide services to the end user having limited configuration in their hand held devices. The Cloud service provider will provide its services through web and the consumer will pay for using the services. In it, one service provider may subsequently use the services of other service providers without the awareness of consumers. The model is expected to be mutually beneficial to the client and its service provider.

4. PROPOSED APPROACH

The approach is to develop a suitable Cloud Computing service and cost model for providing cloud computing service (like Compilation service, file service etc.)

SERVICE MODEL

The steps of this model are summarized as follows:-

Step 1: In accordance with the service model the client (CL) requests for a service from a suitable service provider (SP).

Step 2: The service provider on receiving the request ensures whether it is possessing sufficient resource to provide the service requested by its client.

Step 3: If the service provider finds that it possesses sufficient resource to provide the service then it provides the service to its client directly

else

{

if the service provider finds that it does not possess all the necessary resource then it finds suitable service providers and requests those service providers to provide it with the required resource.

Again the steps from Step 2 onwards are repeated for each service provider requested (for providing service).

}

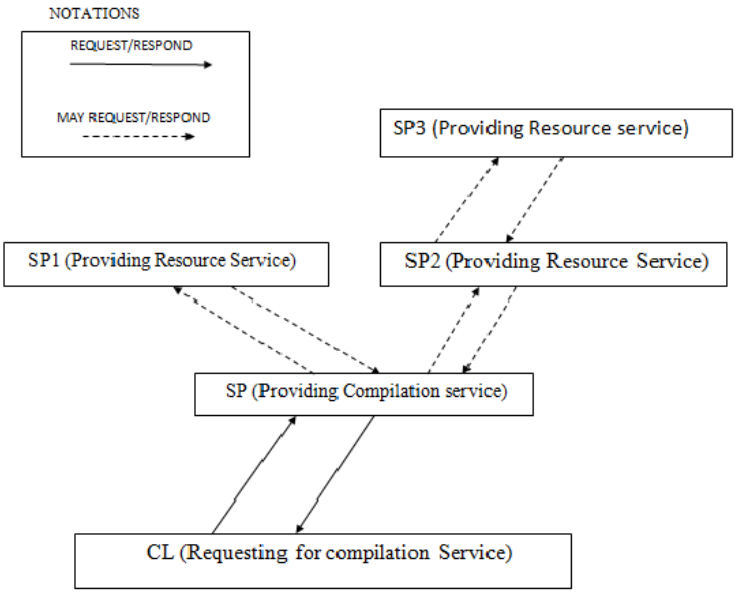

Figure 2. Diagrammatic representation of an implementation of the service model

Considering an implementation of the service model (as shown in Fig.2) where client will request to provide a Platform for writing and compiling their programs/source code. The service provider will provide the user friendly platform for writing and compiling programs on the fly using any hand held device having internet access facility. In accordance with this implementation the following activities takes place:

1. Client (CL) will write his/her program or source codes using the user friendly interface provided in the web page and request for compilation to the service provider (SP).
2. The service provider (SP) on receiving the request will check whether all the necessary packages required for compilation are available or not.
3. If the service provider finds that it possesses all the necessary packages which are required for compilation then it provides the service to its client directly.
   Else
   { it finds suitable service providers and requests the service providers (SP1,SP2,..) to provide it with the required resource.
   3.1 The new service providers (SP1,SP2,…) on receiving the request may again check whether it has all the packages which is requested by its client.
   3.2 If the new service provider finds that it possess all the necessary packages which are requested it provides the service to its client directly by providing the packages requested.
   Else
   {
   if it finds that it does not possess all the packages requested then it finds appropriate service providers and makes requests to provide it with the required (missing) package.
   Step 3.1 is repeated again.
   }
   }
4. Once the service provider (SP) receives all the packages necessary for compiling the program it compiles the program and provides the results/output of the compilation to the client (CL).

COST MODEL
The cost model is based on the following steps which are enumerated as follows.

Let the cost which X pays for receiving services be c(X)
Let the service charge for X providing a service be S_x.

When the client (M) receives the service which requested from the service provider (N) then the cost that client (M) pays to its service provider (N) is $c(M)=c(N)+S_x$
Where c(N) is the cost which N pays for receiving services from other service providers.
S_x is the service charge (based on companies policies) of N for providing the service to its client (M).

c(N)=0 , when N does not make any request for providing any service to it.
if N requests services from k number of service providers (labeled as Q_1, Q_2,…, Q_k)

then $c(N)=S_{Q1}+ c(Q_1) + S_{Q2}+c(Q_2)+\ldots+ S_{Qk}+c(Q_k)$

$$= \sum_{i=1}^{k} S_{Qi}+c(Q_i)$$

In accordance with the implementation in Fig. 2 the cost is evaluated in the following manner.

Step1. Cient (CL) writes his/her program or source code using the user friendly interface provided in the web page and sends a request for compilation to the service provider (SP).

Step2. On receiving the request service provider (SP) checks whether all the necessary packages are available to it for compilation of source code.

Step3. If the service provider (SP) finds that it possess all the packages necessary to compile the source code then it provides its service (i.e, the result of the compilation) to its client (CL) by deducting the service charge s(SP) from client (CL)'s account.

Step 4. If the service provider (SP) finds that it does not possess all the packages necessary to compile the source code then it requests for providing the necessary packages from other service providers.

Consider a scenario where the service provider (SP) requests for packages from service providers SP1 and SP2. SP2 may not have all the packages requested by SP. Hence, SP2 sends a request to SP3 to provide the required (missing) package. Once, SP3 provides the packages requested to SP2. SP2 provides the packages requested to SP. SP1 also provides the packages requested to it by SP. When the service provider (SP) receives all the packages necessary for compilation, it does compilation of the source code and provides its service (i.e, the result of the compilation) to its client (CL).

In this case the service provider
SP2 pays a charge $c(SP2)=S_{SP3}$ ….(i) (since c(SP3)=0)
SP pays a charge $c(SP)= S_{SP1}+S_{SP2}+c(SP2)$ ….(ii) (since c(SP1)=0)
$= S_{SP1}+S_{SP2}+ S_{SP3}$ ….(iii) (Substituting (i) in (ii))
for receiving the packages
Hence the client (CL) pays a charge c(CL)
$=S_{SP}+c(SP)$ ….(iv)
$= S_{SP}+ S_{SP1}+S_{SP2}+ S_{SP3}$ (Substituting (iii) in (iv))

5. IMPLEMENTATION

The proposed approach has been implemented by developing a Web application using Java servelets and Oracle. In accordance with the implementation a new client first creates an account by filling up a registration form (by submitting the required information) online and paying a charge (as fixed by the service provider). Once the client is registered, he/she can log into the service provider's website using his/her user-id and password. If the client successfully logs in then he will be provided with the options to edit his profile,

compile a source code, view balance, make payment etc. Using the user name and password the service provider authenticates the clients. After authentication the service provider will provide a user friendly interface where the authorized user writes the program (Java or C source code), specifies a filename for saving it, submits it for compilation (as shown in Fig.3). The service provider compiles the source code and displays the results of compilation. If the compilation is successful then the files generated after compilation (Java class files/bytecodes or binary executable or binary object files) are provided to the client. If the compilation is not successful then the error messages generated are displayed to the client.

Figure 3. Snapshot of the user interface

Every time a request is made by the client for compilation a charge (based on the proposed cost model) is deducted from his/her account. If a service provider finds that it does not possess all the necessary packages required for compilation then it finds appropriate service providers and requests it to provide the missing (necessary) package in accordance with the proposed service model. The implementation has been made in such a way that the packages provided by the service providers can be used only once. This aspect has been incorporated to prevent reusability (thereby preventing software piracy to a considerable extent) of the packages once provided by any service provider. It implies that every time you require a package, you need to make a request to the vendor specific service provider thereby paying the requisite amount to the provider of the package. This would provide the service providers a great impetus and would help in boosting up their business.

Figure 4. Snapshot of Web page showing files available for download after compilation

After downloading the required files (class files or binary object or binary executable) the client can logout by clicking on the logout option.

6. RESULTS AND DISCUSSION

This system works perfectly fine for compiling various programs written in high-level languages(C, java, etc.) remotely from any handheld device (like mobile phones, PDA, etc.) To write and execute programs using this system, one does not require a desktop PC or laptop with a compiler along with the libraries necessary for compilation. But, any person, from any location can write and execute programs using

their handheld devices (like mobile phones, etc. having internet access facility) provided the runtime environment is available. After a thorough survey of the various scenarios for providing and receiving Cloud Computing services, some trends have been observed and are represented in plots of Fig. 5 and Fig. 6. The proposed model after implementation has been found to provide better results compared to the traditional approach where one pays the entire amount by purchasing the product (containing several features or components) irrespective of the features or components one uses (as discussed in section 6.1).

6.1 Comparison of tradional and proposed method

In Fig.5 C_0 is the cost which one pays in the traditional approach to purchase the entire product (containing n_0 number of components where each component belongs to a specific vendor) irrespective of the number of components used. In the proposed approach the best case shows that initially (when few components are used) the consumer pays a very low cost and the cost increases gradually as more and more components are being used (shown in Fig.5). Hence, the consumer pays very less as compared to the traditional approach for getting a service (component) from various service providers (corresponding to various vendors). In the worst case initially (when few components are used) consumer pays very high cost and the cost increases gradually as more components are being used. In accordance with Fig. 5, it is found that when the number of components used is n_0 then the consumer in all the cases (i.e tradinational, proposed approach: best case and worst case) pays the cost C_0. Otherwise, in the proposed approach both under best and worst cases it is found that the cost which a consumer pays is always less than the actual price of the product (C_0) when the number of components used is less than n_0.

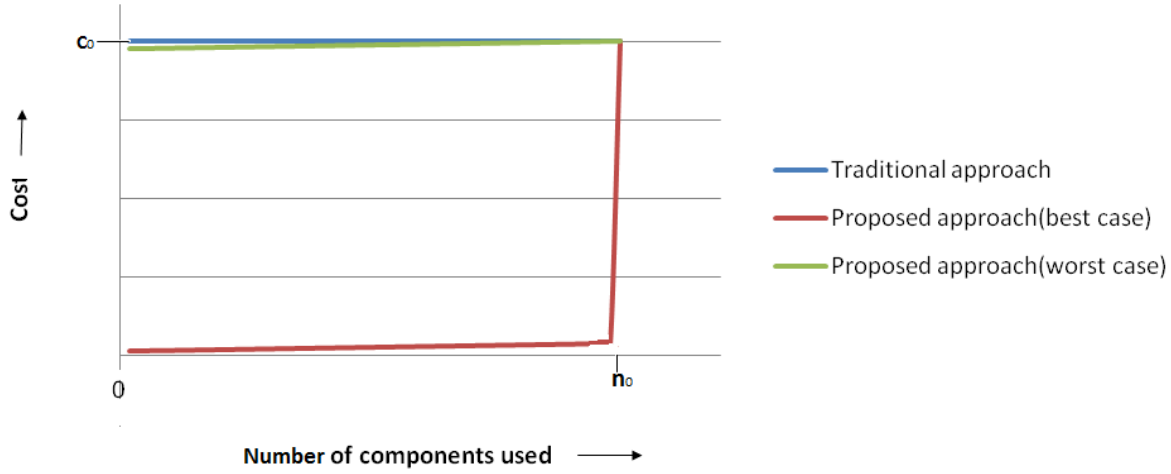

Figure 5. Plots showing the cost incurred by the consumer based on the number of components used

Looking from the perspective of the service provider it is found that in traditional approach the service provider ear ns the revenue (profit) from a customer only once (after selling the product to the customer).But in the proposed approach the revenue (profit) earned goes on increasing depending upon the number of times a service is requested and provided. This aspect is best explained in the following example. Considering that a service provider earns a profit of p for providing a particular service once. If n number of times a service provider is requested for the same service then the service provider earns a profit np, by providing the service n number of times.Therefore, it is found that the profit or the revenue goes on increasing linearly with the number of times a service is provided (as shown in Fig.6).

According to Fig.6, it is found that in the proposed approach initially (when less than n_c number of times a service is requested) the revenue (profit) earned is low as compared to traditional approach. But, when more than n_c number of times a service is requested the revenue (profit) becomes more than the revenue (profit) earned in the traditional approach.

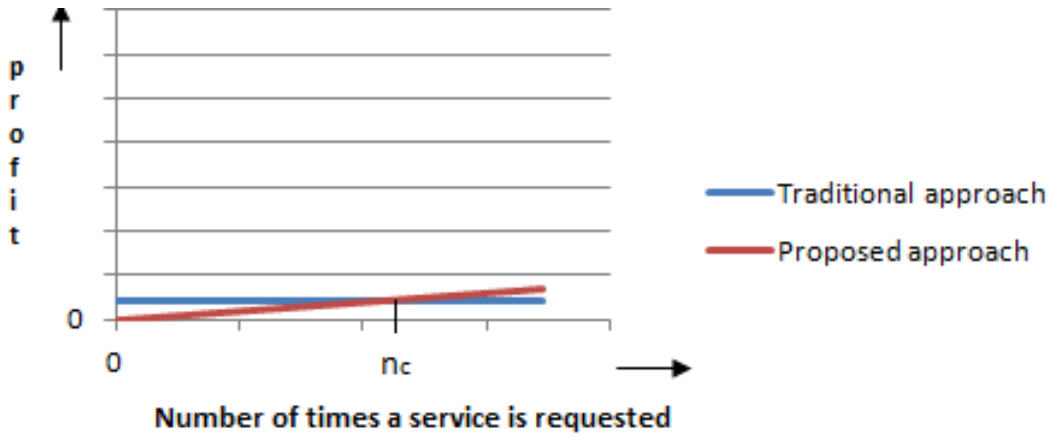

Figure 6. Plots showing the profit gained by the provider based on the number times a service is provided

7. CONCLUSION

The proposed approach comprising of service and cost model for providing Cloud computing service for writing and compiling source code remotely from any hand held device using the distributed service model is expected to be mutually beneficial to the client and its service provider. According to this model, for providing a service, the service provider may subsequently use the services of other service providers (which may in turn use the services of other service providers to provide their service) in the cloud without the awareness of the client. The cost model effectively evaluates the cost which each client pays to each of its service providers for receiving the services. The model is expected to be mutually beneficial to the client and its service provider. The client is paying only for the service it receives and the service provider only provides the service that is requested by the client. The cost is evaluated based on the kind of service provided. Features are being incorporated for providing better data security and PaaS support for various other tasks. Since, the packages provided by the service providers can be used only once, reusability of the packages once provided by any service provider can be prevented (thereby preventing software piracy to a considerable extent). It implies that every time you require a package, you need to make a request to the vendor specific service provider thereby paying the requisite amount to the provider of the package. Hence. this model is also expected to reduce software piracy to a considerable extent thereby helping the vendors in boosting up their business.

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to Prof. A. K Bisoi, Dean, School of Computer Engineering, KIIT University for his inspiration and support behind this work.

REFERENCES

[1] Yashpalsing Jadeja, Kirit Modi "Cloud computing concepts, architecture and challenges", *International Conference on Computing, Electronics and Electrical Technologies [ICCEET] 2012, Kumaracoil,* pp.877-880, 2012.

[2] Janakiram MSV "Demystifying the Cloud Title An Introduction to Cloud Computing", *version 1.0-http://www.janakiramm.net*, March 2010.

[3] Ian Foster, Yong Zhao, Ioan Raicu, Shiyoung Lu "Cloud Computing and Grid Computing 360-degree compared" , *USA IEEE Grid Computing Environments (GCE08),* 2008.

[4] Smith Dhumbumroong, Krerk Piromsopa, "Personal Cloud Filesystem: A distributed Unification File system for Personal computer and portable Device", *Computer Science and Software Engineering (JCSEE), Eight International Joint Conference 2011, Thailand,* pp.58-61, 2011.

[5] Zeeshan Parvez, Syngyoung Lee, Young KooLee "Multi-Tenant, Secure, Load Disseminated SaaS Architecture", *Advance Communication Technology (ICACT) 2010 South Korea vol-1,* pp. 214-219, 2010.

[6] Muhammad Agni Catur Bhakti, Hermawan Nugroho, Petrus Tri Bhaskaro, Firmanysha, Irving Vitra Paputungan ,Unan Yusmanir Oktiawati "Taking up Autonomous SOA Framework into Cloud Computing", *International Conference on Cloud Computing and Social Networking (ICCCSN) 2012 Bandung ,West Java*, pp.1-4, 2012.

[7] Cheng Kefei, Cui Yanglei "Design and Implementation of Network Packets Collections Tools Based on the Android Plat form", *9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012 China*, pp.2166-2169, 2012.

[8] Hsiu-Hui Lee and Chun-Hsiun Tseng "A software Framework for Java message service Based Internet Messaging System", *4th International Conference on Parallel and Distributed Computing, Applications and Technologies2003*, pp.161-165, 2003.
[9] Yanfei Li, Ying Wang, Bo Yin, Lu Guan, "An online Power metering model for cloud environment", *IEEE 11th International Symposium on Network Computing and Application (NCA), Cambridge, MA*, pp. 175-180, Aug 2012.
[10] Wonil Kim, Joo Hwan Lee, Chuleui Hong, Changhee Han, Hanku Lee , Bongshik Jang , "An innovative method for data and software integration in SaaS", *Computers and Mathematics with Applications* Vol. 64, pp.1252–1258, 2012.
[11] Young Choon Lee, Chen Wangb, Albert Y. Zomayaa, Bing Bing Zhoua, "Profit-driven scheduling for cloud services with data access awareness*", Journal of Parallel Distriuted. Computing*, Vol. 72, pp.591–602, 2012.
[12] Slaven Brumec, NevenVrcek, "Cost effectiveness of commercial computing clouds", *Information Systems*, Vol. 38 pp.495–508, 2013.
[13] Xiaoyu Yang, Bassem Nasser, Mike Surridge, Stuart Middleton, "A business-oriented Cloud federation model for real-time applications", *Future Generation Computer Systems*, Vol.28, pp.1158–1167, 2012.
[14] Bahman Javadi, Jemal Abawajy, Rajkumar Buyya, "Failure-aware resource provisioning for hybrid Cloud infrastructure", *Journal of Parallel Distributed Computing*, Vol.72, pp.1318–1331, 2012.
[15] Lianyong Qi, Wanchun Dou , Xuyun Zhang , Jinjun Chen, "A QoS-aware composition method supporting cross-platform service invocation in cloud environment", *Journal of Computer and System Sciences*, Vol.78, pp.1316–1329, 2012.

BIOGRAPHY OF AUTHORS

Shreya Bhadra is currently pursuing M.Tech from KIIT University, Bhubaneswar, India. She received her Bachelor of Technology degree in Information Technology from West Bengal University of Technology, West Bengal. Her research area includes Cloud Application and Cloud Privacy. Email: shr88bha@gmail.com

Tirthankar Gayen, B.E, M.Tech, Ph.D is currently working as Associate Professor, in the School of Computer Engineering, at KIIT University. Earlier he worked as Associate Scientist at ABB and remained faculty at IIIT and other institutes. He has published papers in many International Conferences and Journals. He is a reviwer of several peer reviewed International Journals. His area of interest includes Software Engineering, Natural Language Processing and Cloud Computing.