Scientific Journal Impact Factor (SJIF) : 3.518
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745 ISSN (PRINT) : 2393 - 8161

An Integrated Cost Management Framework for Commercial Clouds

S. Deepa[1] and S. Muruganandham[2]
[1]*M.Phil Full Time Research Scholar, Department of Computer Science,*
[2]*Asst. Professor, Department Of Computer Science and Applications,*
Vivekanandha College of Arts and Sciences for Women. (Autonomous), Tiruchengode, India.

Abstract--Cloud service providers facilitate services to the users based on the request with pay by use model. Hardware resources are provided by the infrastructure vendors for the cloud service providers. Income and cost factors are considered in the pricing process for the commercial cloud environment. Service charges are categorized as income and expenses are indicated as cost. Service provider profit is estimated with the service charge and cost factors. Different cost functions are applied in the service cost estimation process. Energy, margin, Demand and supply factors are also considered in the service cost estimation process. Services are provided with different speed levels based on the request strategies.

The service cost management scheme is enhanced to handle immediate, reservation, peak supply and peak demand request levels. Service pricing policies are upgraded with data usage and data transfer properties. The service provider is also capable to select services dynamically. The cost management system is also improved to estimate and increase the profit for the service providers.

I. Introduction

Cloud Computing is a computing paradigm in which different computing resources such as infrastructure, platforms and software applications are made accessible over the Internet to remote user as services. Infrastructure-as-a- Service (IaaS) cloud providers, such as Amazon EC2 and IBM Cloud, deliver, on-demand, operating system (OS) instances provisioning computational resources in the form of virtual machines deployed in the cloud providers data center. A cloud service differs from traditional hosting in three principal aspects. First, it is provided on demand; second, it is elastic since users that use the service have as much or as little as they want at any given time; and third, the service is fully managed by the provider. Due to dynamic nature of cloud environments, diversity of user requests and time dependency of load, providing agreed quality of service (QoS) while avoiding over-provisioning is a difficult task.

Service availability and response time are two important quality measures in cloud's users perspective. Quantifying and characterizing such performance measures requires appropriate modeling; the model ought to cover vast parameter space while being tractable. A monolithic model may suffer from intractability and poor scalability due to large number of parameters. Instead, we develop and evaluate tractable functional sub-models and their interaction model and solve them iteratively. We construct separate sub-models for different servicing steps in a complex cloud center and then the overall solution is obtained by iteration over individual sub-model solutions. We assume that the cloud center consists of a number of Physical Machines (PM) that are allocated to users in the order of task arrivals. More specifically, user may share a PM using virtualization technique. A cloud user may ask for more

than one virtual machine (VM) by submitting a single compound request, hereafter referred to as a supertask

.

II. Related Work

Most related work on privacy preserving recommendations is secure in the semi-honest model, so parties are assumed to follow the rules of the protocol. As mentioned by Lagendijk et al. [2], "Achieving security against malicious adversaries is a hard problem that has not yet been studied widely in the context of privacy-protected signal processing." Erkin et al. [3] securely computed recommendations based on collaborative filtering. They used homomorphic encryption within a semi-honest security model just like Bunn and Ostrovsky. Goethals et al. stated that although such techniques can be made secure in the malicious model that will make them unsuitable for real life applications because of the increased computational and communication costs.

Polat and Du used a more lightweight approach by statistically hiding personal data, which unfortunately has been proven insecure by Zhang et al. Atallah et al. used a threshold secret-sharing approach for secure collaborative forecasting with multiple parties. Nikolaenko et al. [9] securely computed collaborative filtering by means of matrix factorization. They used both homomorphic encryption and garbled circuits in a semi-honest security model. In another paper [10], these authors use similar techniques to securely implement Ridge regression, a different approach of collaborative filtering. Catrina and de Hoogh [6] developed an efficient framework for secure computations in the semi-honest model, based on secret sharing and statistical security, which could also be used for a recommender system. Peter et al. [7] considered a model where users can outsource computations to two non-colluding servers. They use homomorphic encryption, each user having its own key, but require the servers to follow the rules of the protocol.

Canny's approach private collaborative filtering is able to cope with malicious behaviour. In the last several years, a couple of computation protocols have been developed, which are both practical and secure in the malicious model. The idea is to use public-key techniques in a data-independent pre-processing phase, such that cheap information-theoretic primitives can be exploited in the online phase, which makes the online phase efficient [8]. In 2011, Bendlin et al. [1] presented such a framework with a somewhat homomorphic encryption scheme for implementing the pre-processing phase. This has been improved lately by Damgård et al. [4], which has become known as SPDZ. Last year, Damgård et al. [5] showed how to further reduce the precomputation effort.

III. Commercial Cloud Services

Cloud computing is quickly becoming an effective and efficient way of computing resources and computing services consolidation. By centralized management of resources and services, cloud computing delivers hosted services over the Internet, such that accesses to shared hardware, software, databases, information and all resources are provided to consumers on-demand. Cloud computing is able to provide the most cost-effective and energy-efficient way of computing resources management and computing services provision. Cloud computing turns information technology into ordinary commodities and utilities by using the pay-per-use pricing model. Cloud computing will never be free and understanding the economics of cloud computing becomes critically important.

One attractive cloud computing environment is a three tier structure, which consists of infrastructure vendors, service providers and consumers. The three parties are also called cluster nodes, cluster managers and consumers in cluster computing systems and resource providers, service providers and clients in grid computing systems. An infrastructure vendor maintains basic hardware and software facilities. A service provider rents resources from the infrastructure vendors, builds appropriate multiserver systems and provides various services to users. A consumer submits a service request to a service provider, receives the desired result from the service provider with certain service-level agreement and pays for the service based on the amount of the service and the quality of the service. A service provider can build different multiserver systems for different application domains, such that service requests of different nature are sent to different multiserver systems. Each multiserver system contains multiple servers and such a multiserver system can be devoted to serve one type of service requests and applications. An application domain is characterized by two basic features, i.e., the workload of an application environment and the expected amount of a service. The configuration of a multiserver system is characterized by two basic features, i.e., the size of the multiserver system and the speed of the multiserver system.

Like all business, the pricing model of a service provider in cloud computing is based on two components, namely, the income and the cost. For a service provider, the income is the service charge to users and the cost is the renting cost plus the utility cost paid to infrastructure vendors. A pricing model in cloud computing includes many considerations, such as the amount of a service, the workload of an application environment, the configuration of a multiserver system, the service-level agreement, the satisfaction of a consumer, the quality of a service, the penalty of a low-quality service, the cost of renting, the cost of energy consumption and a service provider's margin and profit. The profit is the income minus the cost. To maximize the profit, a service provider should understand both service charges and business costs and in particular, how they are determined by the characteristics of the applications and the configuration of a multiserver system.

The service charge to a service request is determined by two factors, i.e., the expected length of the service and the actual length of the service. The expected length of a service is the execution time of an application on a standard server with a baseline or reference speed. Once the baseline speed is set, the expected length of a service is determined by a service request itself, i.e., the service requirement measured by the number of instructions to be executed. The longer the expected length of a service is, the more the service charge is. The actual length of a service is the actual execution time of an application. The actual length of a service depends on the size of a multiserver system, the speed of the servers and the workload of the multiserver system. Notice that the actual service time is a random variable, which is determined by the task waiting time once a multiserver system is established.

There are many different service performance metrics in service-level agreements. The performance metric is the task response time, i.e., the time taken to complete a task, which includes task waiting time and task execution time. The service-level agreement is the promised time to complete a service, which is a constant times the expected length of a service. If the actual length of a service is within the service-level agreement, the service will be fully charged. If the actual length of a service exceeds the service-level agreement, the service charge will be reduced. The longer the actual length of a service is, the more the reduction of the service charge is. In other words, there is penalty for a service provider to break a service-level agreement. If the actual service time exceeds certain limit, a service will be entirely free with no charge. Notice

that the service charge of a service request is a random variable and we are interested in its expectation.

The cost of a service provider includes two components. The renting cost is proportional to the size of a multiserver system, i.e., the number of servers. The utility cost is essentially the cost of energy consumption and is determined by both the size and the speed of a multiserver system. The faster the speed is, the more the utility cost is. To calculate the cost of energy consumption, we need to establish certain server speed and power consumption models. To increase the revenue of business, a service provider can construct and configure a multiserver system with many servers of high speed. Since the actual service time contains task waiting time and task execution time, more servers reduce the waiting time and faster servers reduce both waiting time and execution time. Hence, a powerful multiserver system reduces the penalty of breaking a service-level agreement and increases the revenue. More servers increase the cost of facility renting from the infrastructure vendors and the cost of base power consumption. Furthermore, faster servers increase the cost of energy consumption.

The problem of optimal multiserver configuration for profit maximization in a cloud computing environment. The approach is to treat a multiserver system as an M/M/m queuing model, our optimization problem can be formulated and solved analytically. We consider two server speed and power consumption models, namely, the idle-speed model and the constant-speed model. Our main contributions are as follows. We derive the probability density function (pdf) of the waiting time of a newly arrived service request. This result is significant in its own right and is the base of our discussion. We calculate the expected service charge to a service request. Based on these results, we get the expected net business gain in one unit of time and obtain the optimal server size and the optimal server speed numerically. To the best of our knowledge, there has been no similar investigation in the literature, although the method of optimal multicore server processor configuration has been employed for other purposes, such as managing the power and performance tradeoff.

One related research is user-centric and market-based and utility-driven resource management and task scheduling, which have been considered for cluster computing systems and grid computing systems. To compete and bid for shared computing resources through the use of economic mechanisms such as auctions, a user can specify the value of a task. A utility function, measures the value and importance of a task as well as a user's tolerance to delay and sensitivity to quality of service, supports market-based bidding, negotiation and admission control. By taking an economic approach to providing service-oriented and utility computing, a service provider allocates resources and schedules tasks in such a way that the total profit earned is maximized. Instead of traditional system-centric performance optimization such as minimizing the average task response time, the main concern in such computational economy is user-centric performance optimization.

IV. Problem Statement

Pricing model of a service provider in cloud computing is based on two components, income and cost. In a service provider the income is the service charge to users and the cost is the renting cost plus the utility cost paid to infrastructure vendors. Service charge and business cost factors are used to maximize the profit for a service provider. Multiserver configuration, Service Level Agreement (SLA), service and application load properties are used to assign service costs. Consumer satisfaction, Quality of Service (QoS) and penalty parameters are used to decide the service costs. Renting cost, energy cost and service provider margin are also used

for the service cost estimation process. Multiserver system is treated as M/M/m queuing model. Server speed and power consumption strategy is divided into two models such as idle-speed model and the constant-speed model. The weighting time of a service request is derived using the probability density function. The following problems are identified in the current commercial cloud services. Data access information are not used in the cost functions. Static pricing model. Service charging function selection is not provided. Profit level prediction is not provided.

V. Cost Management Model for Cloud Services

A cloud computing service provider serves users' service requests by using a multiserver system, which is constructed and maintained by an infrastructure vendor and rented by the service provider. The architecture detail of the multiserver system can be quite flexible. Examples are blade servers and blade centers where each server is a server blade, clusters of traditional servers where each server is an ordinary processor and multicore server processors where each server is a single core. We will simply call these blades/processors/cores as servers. Users submit service requests to a service provider and the service provider serves the requests on a multiserver system. We use P[e] to denote the probability of an event e. For a random variable x, we use f_x (t) to represent the probability density function of x and f_x (t) to represent the cumulative distribution function (cdf) of x and x to represent the expectation of x.

Assume that a multiserver system S has m identical servers. In this paper, a multiserver system is treated as an M/M/m queuing system which is elaborated as follows. There is a Poisson stream of service requests with arrival rate λ, i.e., the interarrival times are independent and identically distributed (i.i.d.) exponential random variables with mean 1= λ. A multiserver system S maintains a queue with infinite capacity for waiting tasks when all the m servers are busy. The first-come-first-served (FCFS) queuing discipline is adopted. The task execution requirements are i.i.d. exponential random variables r with mean r. The m servers of S have identical execution speed. Hence, the task execution times on the servers of S are i.i.d. exponential random variables x = r=/s with mean x = r/s.

5.1. Power Consumption Models

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static and short-circuit power dissipation; the dominant component in a well-designed circuit is dynamic power consumption P, which is approximately $P = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage and f is the clock frequency. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V \alpha f^{\phi}$ for some constant $\phi > 0$. The processor execution speed s is usually linearly proportional to the clock frequency, namely, s α f. For ease of discussion, we will assume that $V = bf^{\phi}$ and s = cf, where b and c are some constants. Hence, we know that power consumption is $P = aCV^2f = ab^2Cf^{2\phi+1} = (ab^2C/c^{2\phi+1})s^{2\phi+1} = \xi s^{\alpha}$, where $\xi = ab^2C/c^{2\phi+1}$ and $\alpha = 2\phi+1$. For instance, by setting b = 1:16, aC = 7:0, c = 1:0, ϕ = 0:5, $\alpha = 2\phi+1 = 2:0$ and $\xi = ab^2C/c^{\alpha} = 9:4192$, the value of P calculated by the equation $P = aCV^2f = \xi s^{\alpha}$ is reasonably close to the Intel Pentium M processor.

We will consider two types of server speed and power consumption models. In the idle-speed model, a server runs at zero speed when there is no task to perform. Since the power for speed s is ξs^{α}, the average amount of energy consumed by a server in one unit of time is $p\xi s^{\alpha} = \lambda/m\ r\xi s^{\alpha-1}$, where we notice that the speed of a server is zero when it is idle. The average amount of energy consumed by an m-server system S in one unit of time, i.e., the power supply to the

multiserver system S, is $P = mp\xi s^{\alpha} = \lambda r\xi s^{\alpha-1}$, where $mp = \lambda x$ is the average number of busy servers in S. Since a server still consumes some amount of power P* even when it is idle, we will include P* in P, i.e., $P = p\xi s^{\alpha} + P^*) = \lambda\, r\xi s^{\alpha-1} + mP^*$. Notice that when P* = 0, the above P is independent of m. In the constant-speed model, all servers run at the speed s even if there is no task to perform. Again, we use P to represent the power allocated to multiserver system S. Since the power for speed s is ξs^{α}, the power allocated to multiserver system S is $P = m(\xi s^{\alpha} + P^*)$.

5.2. Waiting Time Distribution

Let W denote the waiting time of a new service request that arrives to a multiserver system. We find the pdf fW(t) of W. To this end, we consider W in different situations, depending on the number of tasks in the queuing system when a new service request arrives. Let W_k denote the waiting time of a task that arrives to an M/M/m queuing system under the condition that there are k tasks in the queuing system when the task arrives.

5.3. Service Charge

If all the servers have a fixed speed s, the execution time of a service request with execution requirement r is known as x = r/s. The response time to the service request is T = W + x = W + r/s. The response time T is related to the service charge to a customer of a service provider in cloud computing. To study the expected service charge to a customer, we need a complete specification of a service charge based on the amount of a service, the service-level agreement, the satisfaction of a consumer, the quality of a service, the penalty of a low-quality service and a service provider's margin and profit. The above function is defined with the following rationals:

- If the response time T to process a service request is no longer than $(c/s_0)r = c(r/s_0)$, where the constant c is a parameter indicating the service level Agreement and the constant s_0 is a parameter indicating the expectation and satisfaction of a consumer, then a service provider considers that the service request is processed successfully with high quality of service and charges a customer ar, which is linearly proportional to the task execution requirement r, where a is the service charge per unit amount of service.
- If the response time T to process a service request is longer than $(c/s_0)r$ but no longer than $(a/d + c/s_0)r$, then a service provider considers that the service request is processed with low quality of service and the charge to a customer should decrease linearly as T increases. The parameter d indicates the degree of penalty of breaking the service-level agreement.
- If the response time T to process a service request is longer than $(a/d + c/s_0)r$, then a service provider considers that the service request has been waiting too long, so there is no charge and the service is free.

Notice that the task response time T is compared with the task execution time on a server with speed s_0. The actual speed s of a server can be decided by a service provider, which can be either lower or higher than s_0, depending on the workload and system parameters and the service charge function, such that the net business gain to be defined below is maximized.

VI. An Integrated Cost Management Framework for Service Providers

The service pricing model is improved to manage on demand, reservation, peak demand and peek supply situations. Data usage cost and communication cost metrics are added to the service charge functions. Dynamic service function selection model is integrated with the system. Service request and access levels are analyzed to estimate the profit level of the service

provider. The commercial cloud service provisioning scheme is improved with dynamic pricing models. Charging function selection mechanism is used in the system. Profit prediction and analysis mechanism is integrated with the system. The system is divided into five major modules. They are infrastructure vendor, service provider, cloud consumer, pricing process and service usage analysis.

Cloud resources are provided under infrastructure vendor module. Service provider provides services for the consumers. Cloud service requests are submitted by the cloud consumers. Pricing process module is used to calculate resource prices. Service usage analysis module is designed to estimate the profit levels. The infrastructure vendor provides the hardware and software resources for the service providers. Infrastructure vendor collects prices for the resource usage levels from the service providers. Computational resource usage, storage space usage and network bandwidth usage details are used in the pricing process. Software facilities are also charged by the vendors. Shared services are provided by the service provider. Services are loaded under different infrastructure vendors. Services are provided with pricing scheme. Dynamic pricing scheme is used to estimate the service prices.

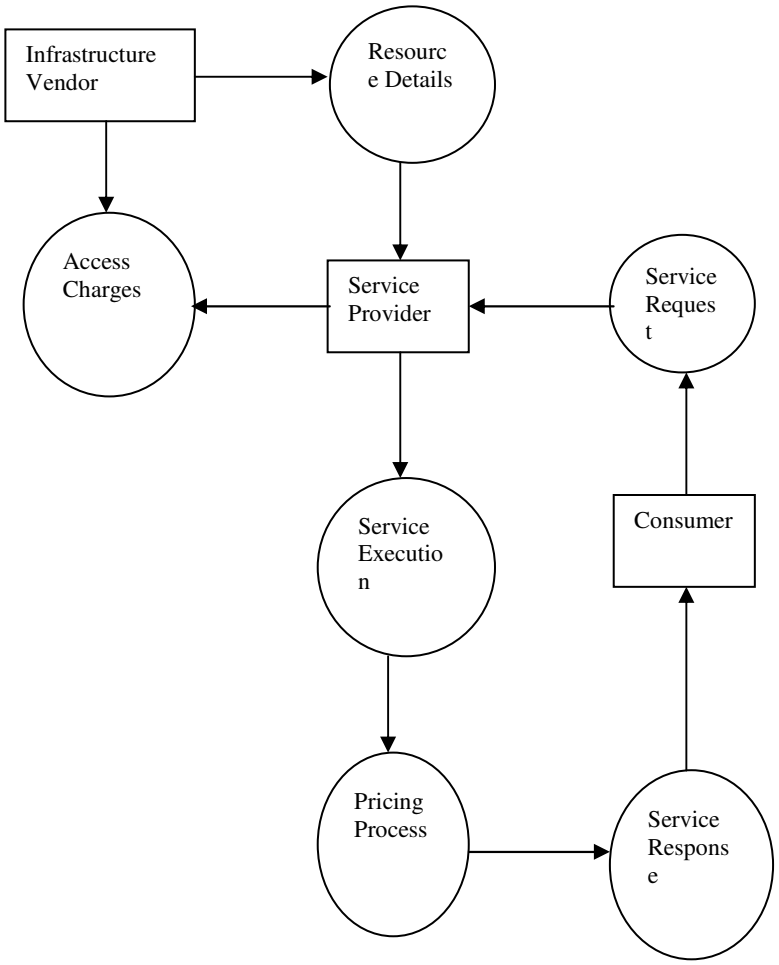

Fig No: 6.1. An Integrated Cost Management Framework for Service Providers

The consumer application is used to access services under the cloud. Service requests are issued by the consumers to service providers. Services are executed under the service provider and the results are

redirected to the consumer. Consumer pays the service charges with reference to the usage levels. Service prices are dynamically assigned by the service provider. Supply demand factors are considered in the pricing process. Pricing function selection mechanism is used to decide service charges. Data usage and transmission levels are used in the pricing process. Profit level for the service provider is estimated under service usage analysis. Infrastructure expenses are paid by the service provider. Profit level is decided with reference to the service income and infrastructure expenses. Data and communication usages change the profit levels.

VII. Conclusion and Future Enhancement

Cloud service providers provide services to the consumers based on demand model. Different charging parameters are used to estimate the service cost for a consumer. Supply / demand based pricing model is used to increase the profit level of the service providers. The system also supports dynamic service charge function insertion mechanism. The system uses optimal server size and optimal server speed. Cost and energy efficient system. The system achieves high profit level under the service provider. Supply demand based pricing mechanism increases the service provider income. The system handles the transactions between the infrastructure vendor, service provider and consumer. Supply demand based pricing scheme is used in the system. Data and communication usages are analyzed in the system. The system estimates the profit levels for the service providers. The System can be improved with the following features. The cloud service management scheme can be integrated with privacy preserved discovery model. The cloud service management mechanism can be enhanced to handle anonymous and malicious requests. The system can be improved to support data and service security with integrity verification methods. The system can be integrated with computational resource sharing and task execution scheme.

REFERENCES

[1] R. Bendlin, I. Damgård, C. Orlandi and S. Zakarias, "Semihomomorphic encryption and multiparty computation," in Advances in Cryptology (Lecture Notes in Computer Science), vol. 6632. Berlin, Germany: Springer-Verlag, 2011, pp. 169–188.

[2] R. Lagendijk, Z. Erkin and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," IEEE Signal Process. Mag., vol. 30, no. 1, pp. 82–105, Jan. 2013.

[3] Z. Erkin, T. Veugen, T. Toft and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," IEEE Trans. Inf. Forensics Security, vol. 7, no. 3, pp. 1053–1066, Jun. 2012.

[4] I. Damgård, V. Pastro, N. Smart and S. Zakarias, "Multiparty computation from somewhat homomorphic encryption," in Advances in Cryptology (Lecture Notes in Computer Science), vol. 7417. Berlin, Germany: Springer-Verlag, 2012, pp. 643–662.

[5] I. Damgård, M. Keller, E. Larraia, V. Pastro, P. Scholl and N. P. Smart, "Practical covertly secure MPC for dishonest majority—Or: Breaking the SPDZ limits," in Computer Security, vol. 8134. Berlin, Germany: Springer-Verlag, 2013.

[6] O. Catrina and S. de Hoogh, "Improved primitives for secure multiparty integer computation," in Security and Cryptography for Networks (Lecture Notes in Computer Science), vol. 6280. Berlin, Germany: Springer-Verlag, 2010, pp. 182–199.

[7] A. Peter, E. Tews and S. Katzenbeisser, "Efficiently outsourcing multiparty computation under multiple keys," IEEE Trans. Inf. Forensics Security, vol. 8, no. 12, pp. 2046–2058, Dec. 2013.

[8] Thijs Veugen, Robbert de Haan, Ronald Cramer and Frank Muller, "A Framework for Secure Computations With Two Non-Colluding Servers and Multiple Clients, Applied to Recommendations", IEEE Transactions On Information Forensics And Security, Vol. 10, No. 3, March 2015

[9] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft and D. Boneh, "Privacy-preserving matrix factorization," in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2013, pp. 801–812.

[10] V. Nikolaenk, U.Weinsberg, S. Ionnidis, M. Joye, D. Boneh and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in Proc. IEEE Symp. Secur. Privacy, May 2013, pp. 334–348.