Journal of Advances and Scholarly Researches in Allied Education
Vol. 17, Issue No. 2, October-2020, ISSN 2230-7540

Performance Analysis & Evaluation in Cloud Computing

Urvashi Morya[1]* Savita Tiwari[2]

[1] Madhyanchal Professional University, Bhopal

[2] Associate Professor, Madhyanchal Professional University, Bhopal

Abstract – Cloud computing is a type of parallel, virtual, distributed, configurable, and flexible systems, which refers to provision of applications such as hardware's and software's in virtual data centers via internet cloud computing services are configurable and customers pay fees based on the use of resources and services. The most important element of cloud structure is server which is the brain behind the whole processes in cloud. Cloud is the important model for access to distributed computing resources. Pay per use, scalability, use the Internet technology, self-service based on the demand, high performance, quick to implement, easy to maintain and update are key benefits of cloud computing. And the data recovery, lack of control over cloud services, service level agreements, legal problems, different architectures, audit, Reviews and evaluation of the performance cloud computing environment are the major disadvantages of cloud computing.

Keyword: Cloud Computing, Software, Performance, Services

----------X----------

INTRODUCTION

Significant innovations in virtualization and distributed computing, as well as improved access to high-speed Internet, have accelerated interest in cloud computing. Cloud computing is a general term for system architectures that involves delivering hosted services over the Internet. These services are broadly divided into three categories:

Software-as-a-Service (SaaS): In software as a service, cloud service providers provide different software. It leads to better storage utilization of our workstation. Saas vendor provides best software infrastructure like software, network spaces and data center for best solution in developing industry. Examples of SaaS includes: Salesforce.com, Google App

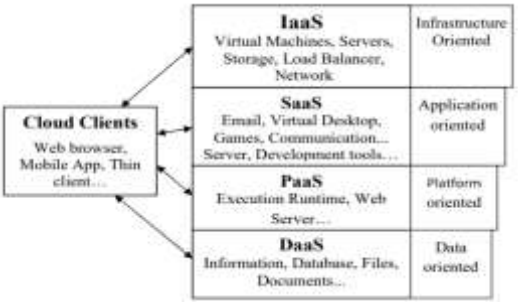

Figure 1.1: Cloud service

Platform as a Service (PaaS): It is the way to use and access service of sboftware without downloading on user's premises or even no need to install it on local machine for any user whether its developer or any end user. It provides great level of platform integration for multitenant systems. When the users are not able to manage the network, servers, operating systems and storage, they opt Platform as a Service. Some examples of PaaS are Force.com, Google App Engine and Microsoft Azure.

Infrastructure as a Service (IaaS): IAAS is a sharing multiple physical resources over network. Main purpose of IAAS is to provide rapid access for server, storage and network by applications and OS. Thus, it offers simple infrastructure on-demand services using Application Programming Interface (API).The user does not need to manage the primary hardware in the cloud infrastructure, but he can control server, application and OS. Some examples of IaaS are Amazon Elastic Cloud Computing (EC2) etc.

Database as a Service (DaaS): DaaS is about the storing of users important document files and other information. This also provides the services related to storing large amount of files which may be mine to fetch relevant information. The database is also an important part of these services which store

users related information like personal information, credential information etc.

CLOUD COMPUTING PERFORMANCE EVALUATION

Cloud computing resources must be compatible, high performance and powerful. High performance is one of the cloud advantages which must be satisfactory for each service. Higher performance of services and anything related to cloud have influence on users and service providers. Hence, performance evaluation for cloud providers and users is important. There are many methods for performance prediction and evaluation; we use the following methods in our Evaluation:

- Evaluation based on criteria and characteristics
- Evaluation based on simulation

Another category which can be considered for evaluating cloud performance is classification of three layers of cloud services evaluation.

Factors affective on performance

Nowadays, the term “performance” is more than a classic concept and includes more extensive concepts such as reliability, energy efficiency, scalability and soon. Due to the extent of cloud computing environments and the large number of enterprises and normal users who are using cloud environment, many factors can affect the performance of cloud computing and its resources. Some of the important factors considered in this paper are as follows:

- Security, the impact of security on cloud performance may seem lightly strange, but the impact of security on network infrastructure has been proven. For example, DDoS attacks have wide impact on networks performance and if happen, it will greatly reduce networks performance and also be effective on response time too. Therefore, if this risk and any same risks threaten cloud environment, it will be a big concern for users and providers.
- Recovery, when data in cloud face errors and failures or data are lost for any reason, the time required for data retrieval and volumes of data which are recoverable, will be effective on cloud performance. For example, if the data recovery takes a long time will be effective on cloud Performance and customer satisfaction, because most organizations are cloud users and have quick access to their data and their services are very important for them.
- Service level agreements, when the user wants to use cloud services, an agreement will be signed between users and providers which describes user’s requests, the ability of providers, fees, fines etc. If we look at the performance from personal view, the better , more optimal and more timely the agreed requests , the higher the performance will be .This view also holds true for providers.
- Network bandwidth, this factor can be effective on performance and can be a criterion for evaluations too. For example, if the bandwidth is too low to provide service to customers, performance will be low too.
- Storage capacity, Physical memory can also be effective on the performance criteria. This factor will be more effective in evaluating the performance of cloud infrastructure.
- Buffer capacity: as shown in figure 2, if servers cannot serve a request, it will be buffered in a temporary memory. Therefore, buffer capacity effect on performance. If the buffer capacity is low, many requests will be rejected and therefore performance will be low.
- Disk capacity, can also have a negative or positive impact on performance in cloud.
- Fault tolerance, this factor will have special effect on performance of cloud environment. As an example, if a data center is in deficient and is able to provide the minimum services, this can increase performance.
- Availability, with easy access to cloud services and the services are always available, performance will be increase.
- Number of users, if a data center has a lot of users and this number is greater than that of the rated capacity, this will reduce performance of services.
- Location, data centers and their distance from a user’s location are also an important factor that can be effective on performance from the users’ view.

Performance Evaluation Criteria

There is a series of criteria for evaluation of all factors affecting performance of cloud computing some of which will be used in this paper. These criteria are under development. Some of these criteria have been selected considering the importance and criteria in simulation. It should be mentioned that all of criteria listed in pervious

sections cover the factors mentioned in the previous section but some of the factors will be important in special criteria:

- Average response time per unit time, this criterion will cover all factors completely.
- Network capacity per second (Mbps) or unit time, the most important factor associated with this criterion is network bandwidth ,availability and scalability.
- The number of I / O commands per second (IOPS)or unit time.
- Average waiting time per unit time
- Workload(requests) to be serviced per second(Mbps) or a unit of time
- Throughput (Req / Sec), this criterion will be recovered recovery, buffering capacity and processing power factors.
- The average time of processing(exe / sec)
- Percentage of CPU utilization
- The number of requests executed per unit time
- The number of requests per unit time buffer
- The number of rejected requests per unit time

The Analytical Model

We model a cloud server farm as a M/G/m queuing system which indicates that the inter-arrival time of requests is exponentially distributed, the service times of customers' requests are independent and identically distributed random variables with a general distribution whose service rate is μ ; both μ and CV , the coefficient of variation defined as standard deviation divided by the mean, are finite.

A M/G/m queuing system may be considered as a Markov process which can be analyzed by applying the embedded Markov chain technique. Embedded Markov Chain technique requires selection of Markov points in which the state of the system is observed. Therefore we monitor the number of the tasks in the system (both in service and queued) at the moments immediately before the task request arrival. If we consider the system at Markov points and number these instances 0, 1, 2, . . . , then we get a Markov chain. Here, the system under consideration contains m servers, which render service in order of task request arrivals.

Task requests arrival process is Poisson. Task request interarrival time A is exponentially distributed with rate to 1/λ . We will devote its Cumulative Distribution Function (CDF) as A(x) = P rob[A < x] and its probability density function (pdf) as a(x) = $\lambda e^{-\lambda x}$. Laplace Stieltjes Transform (LST) of inter arrival time is

$$A^*(s) = \int_0^\infty e^{-sx} a(x)dx = \frac{\lambda}{\lambda+s}.$$

Task service times are identically and independently distributed according toa general distribution B, with a mean service time equal to $\bar{b} = \frac{1}{u}$. The CDF of the service time is B(x) = Prob [B < x], and its pdf is b(x). The LST of service time is

$$B^*(s) = \int_0^\infty e^{-sx} b(x)dx$$

Residual task service time is time from the random point in task execution till the task completion. We will denote it as B_+. This time is necessary for our model since it represents time distribtion between task arrival z and departure of the task which was in service when task arrival z occurred. It can be shown as well that probability distribution of elapsed service time (between start of the task execution and next arrival of task request B has the same probability distribution.

The LST of residual and elapsed task service times can be calculated in

$$B_+^*(s) = B_-^*(s) = \frac{1 - B^*(s)}{s\bar{b}}$$

The offered load may be defined as

$$\rho \triangleq \frac{\lambda}{m\mu}$$

For practical reasons, we assume that the system never enters saturation, which means that any request submitted to the center will get access to the required facility node after a finite queuing time. Furthermore, we also assume each task is serviced by a single server (i.e., there are no batch arrivals), and we do notdistinguish between installation (setup), actual task execution, and nalization components of the service time; these assumptions will be relaxed in our future work.

The Markov chain

We are looking at the system at the moments of task request arrivals { these points are selected as Markov points. A given Markov chain has a steady-state solution if it is ergodic. Based on conditions

for ergodicity and the above-mentioned assumptions, it is easy to prove that our Markov Chain is ergodic. Then, using the steady-state solution, we can extract the distribution of number of tasks in the system as well as the response time.

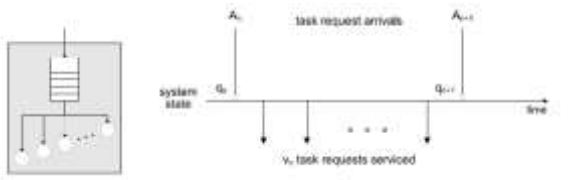

Fig. 2. Embedded Markov points

Let An and An+1 indicate the moment of n^{th} and $(n + 1)^{th}$ arrivals to the system, respectively, while q_n and q_{n+1} indicate the number of tasks found in the system immediately before these arrivals; this is schematically shown in Fig. 2. If v_{n+1} indicates the number of tasks which are serviced and depart from the system between An and An+1, the following holds:

$$q_{n+1} = q_n - v_{n+1} + 1 \quad (3)$$

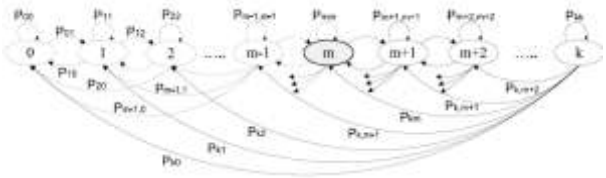

Fig. 3. State-transition-probability diagram for the M=G=m embedded Markov chain.

We need to calculate the transition probabilities associated with this Markov chain, de ned as

$$p_{ij} \triangleq Prob\,[q_{n+1} = j | q_n = i] \quad (4)$$

i.e., the probability that i+1 - j customers are served during the interval between two successive task request arrivals. Obviously for j > i + 1

$$p_{ij} = 0 \quad (5)$$

since there are at most i + 1 tasks present between the arrival of An and An+1. The Markov state-transition-probability diagram as in Fig. 3, where states are numbered according to the number of tasks currently in the system (i.e those in service and those awaiting service). For clarity, some transitions are not fully drown, esp. those originating from states above m. We have also highlighted the state m because the transition probabilities are di erent for states on the left and right hand side of this state (i.e., below and above m).

Departure Probabilities

Due to ergodicity of the Markov chain, an equilibrium probability distribution will exist for the number of tasks present at the arrival instants; so we de ne

$$\pi_k = \lim_{n \to +\infty} Prob\,[q_n = k] \quad (6)$$

From, the direct method of solution for this equilibrium distribution requires that we solve the following system of linear equations:

$$\pi = \pi \mathbf{P} \quad (7)$$

where = [π_0; π_1; π_2; : : :], and P is the matrix whose elements are one-step transition probabilities p_{ij}

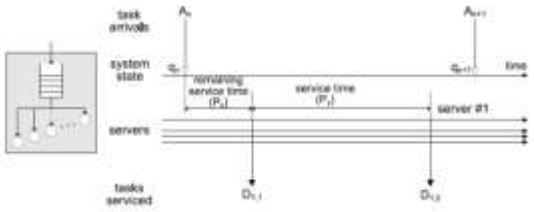

Fig. 4: System behaviour in between two arrivals

To find the elements of the transition probability matrix, we need to count the number of tasks departing from the system in between two successive arrivals. Consider the behaviour of the system, as shown in Fig. 4. Each server has zero or more departures during the time between two successive task request arrivals (the inter-arrival time). Let us focus on an arbitrary server, which (without loss of generality) could be the server number 1. For a task to finish and depart from the system during the inter-arrival time, its remaining duration (residual service time de ned in (1)) must be shorter than the task inter-arrival time. This probability will be denoted as P_x, and it can be calculated as

$$P_x = Prob\,[A > B_+] = \int_{x=0}^{\infty} P\{A > B_+ | B_+ = x\} P\{B_+ = x\} = \int_0^{\infty} e^{-\lambda x} dB_+(x) = B_+^*(\lambda) \quad (8)$$

Physically this result presents probability of no task arrivals during residual task service time.

In the case when arriving task can be accommodated immediately by an idle server (and therefore queue length is zero) we have to evaluate the probability that such task will depart before next task arrival. We will denote this probability as P_y and calculate it as:

$$P_y = Prob[A > B] = \int_{x=0}^{\infty} P\{A > B | B = x\} P\{B_+ = x\}$$
$$= \int_0^{\infty} e^{-\lambda x} dB(x) = B^*(\lambda) \quad (9)$$

However, if queue is non-empty upon task arrival following situation may happen. If between two successive new task arrivals a completed task departs from a server, that server will take a new task from the non-empty queue. That task may be completed as well before the next task arrival and if the queue is still non-empty new task may be executed, and so on until either queue gets empty or new task arrives. Therefore probability of k > 0 job departures from a single server, given that there are enough jobs in the queue can be derived from expressions (8) and (9) as:

$$P_{z,k} = B_+^*(\lambda)(B^*(\lambda))^{k-1} \quad (10)$$

note that Pz,1 = P_x.

Using these values we are able to compute the transition probabilities matrix.

Transition Matrix

Based on our Markov chain, we may identify four di erent regions of operation for which di erent conditions hold; these regions are schematically shown in Fig. 5, where the numbers on horizontal and vertical axes correspond to the number of tasks in the system immediately before a task request arrival (i) and immediately upon the next task request arrival (j), respectively.

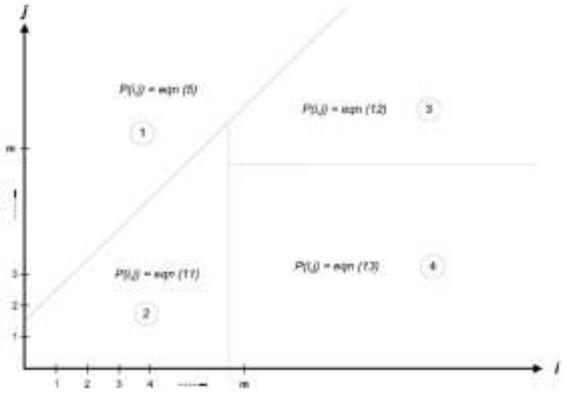

Fig. 5. Range of validity for p_{ij} equations

Regarding the region labelled 1, we already know from Eq. 5 that p_{ij} = 0 for i + 1 < j.

In region 2, no tasks are waiting in the queue, hence i < m and j ≤m. In between the two successive request arrivals, i + 1 - j tasks will complete their service. For all transitions located on the left side of state m in Fig. 3, theprobability of having i + 1- j departures is

$$p_{ij} = \binom{i}{i-j} P_x^{i-j}(1-P_x)^j P_y + \binom{i}{i+1-j} P_x^{i+1-j}(1-P_x)^{j-1}(1-P_y)$$
$$\text{for } i < m, j \le m \quad (11)$$

Region 3 corresponds to the case where all servers are busy throughout the inter-arrival time, i.e., i; j ≥ m. In this case all transitions remain to the right of state m in Fig. 3, and state transition probabilities can be calculated as

$$p_{ij} = \sum_{s=\phi}^{\sigma} \binom{m}{s} P_x^s (1-P_x)^{m-s} P_{z,2}^{i+1-j-s}(1-P_{z,2})^s$$
$$\text{for } i, j \ge m \quad (12)$$

In the last expression, the summation bounds are σ = min [i + 1 – j, m] and Φ= min [i + 1 j, 1].

Finally, region 4, in which I ≥ m and j ≤ m, describes the situation where the first arrival (A_n) nds all servers busy and a total of i- m tasks waiting in the queue, which it joins; while at the time of the next arrival (An+1) thereare exactly j tasks in the system, all of which are in service. The transition probabilities for this region are

$$p_{ij} = \sum_{s=1}^{\sigma} \binom{m}{s} P_x^s (1-P_x)^{m-s} \binom{\eta}{\alpha} P_{z,2}^{\psi}(1-P_{z,2})^{\varsigma\beta}$$
$$\text{for } i \ge m, j < m \quad (13)$$

where we used the following notation:

$$\sigma = \min [m; i + 1- j]$$
$$\eta = \min [s; i + 1- m]$$
$$\alpha = \min [s; i + 1- j, s] \quad (14)$$
$$\psi = \max [0; i + 1-j, s]$$
$$\varsigma = \max [0; j ,m + s]$$
$$\beta = \begin{cases} 1 & \text{if } \psi \le i+1-m \\ 0 & \text{otherwise} \end{cases}$$

Numerical Validation

The steady-state balance equations outlined above can't be solved in closed form, hence we must resort to a numerical solution. To obtain the steady-state probabilities = [π 0; π1; π2; :::], as well as the mean number of tasks in the system (in service and in the queue) and the mean response time, we have used the probability generating functions (PGFs) for the number of tasks in the system:

$$P(z) = \sum_{k=0}^{\infty} \pi_z z^k \quad (15)$$

and solved the resulting system of equations using Maple 13 from Maplesoft, Inc. Since the PGF is an in finite series, it must be truncated for numerical solution; we have set the number of equations to twice the number of servers, which allows us to achieve satisfactory accuracy (as will be explained below), plus the necessary balance equation

$$\sum_{i=0}^{i=2m} \pi_i = 1. \quad (16)$$

the mean number of tasks in the system is, then, obtained as

$$E[QS] = P'(1) \quad (17)$$

while the mean response time is obtained using Little's law as

$$E[RT] = E[QS]/\lambda \quad (18)$$

We have assumed that the task request arrivals follow the gamma distribution with different values for shape and scale parameters; however, our model May accommodate other distributions without any changes. Then, we have per-formed two experiments with variable task request arrival rate and coefficient of variation CV (which can be adjusted in the gamma distribution independently of the arrival rate).To validate the analytical solutions we have also built a discrete even simulator of the cloud server farm using object-oriented Petri net-based simulation engine Artifex by RSoftDesign, Inc. The diagrams in Fig. 6 show analytical and simulation results (shown as lines and symbols, respectively) for mean number of tasks in the system as functions of the offered load , under different number of servers. Two different values of the coefficient of variation, CV = 0:7 and 0.9, were used; the corresponding results are shown in Figs. 6(a) and 6(b). As can be seen, the results obtained by solving the analytical model agree very well with those obtained by simulation.

The diagrams in Fig. 8 show the mean response time, again for the same range of input variables and for the same values of the coefficient of variation. As above, solid lines correspond to analytical solutions, while different symbols correspond to different number of servers. As could be expected, the response time is fairly steady up to the offered load of around = 0:8, when it begins to increase rapidly. However, the agreement between the analytical solutions and simulation results is still very good, which confirms the validity of our modeling approach.

CONCLUSION

According to prediction and evaluation of cloud computing performance, we can reach different Conclusions. Performance evaluation of server farms is an important aspect of cloud computing which is of crucial interest for both cloud providers and cloud customers. As an example, increasing power and speed of the data center is not always efficient, and sometimes it only has additional costs. So, one should not expect to increase efficiency more than what was required and should find standard based on requests and user types. we have proposed an analytical model for performance evaluation of a cloud computing center. Due to the nature of the cloud environment, we assumed general service time for requests as well as large number of servers; in the other words, our model is exible in terms of scalability and diversity of service time. We have further conducted numerical experiments and simulation to validate our model. Numerical and simulation results showed that the proposed method provided a quite accurate computation of the mean number of tasks in the system and mean response time. Distribution of data centers and use of the closest data center is better and more optimal. Due to increase in development of cloud computing, it is predicted that storage and computing on personal computers will be forgotten and all of these things will be transferred into the Clouds. Therefore, architecture and evaluation of the optimal and efficient data centers should be performed for the future of computing world through suitable prediction.

REFERENCE

[1] O. J. Boxma, J. W. Cohen, and N. Hu_el (1979).. Approximations of the mean waiting time in an M=G=s queueing system. Operations Research, 27: pp. 1115-1127.

[2] P. Hokstad (1978). Approximations for the M=G=m queues. Operations Research, 26: pp. 510-523.

[3] T. Kimura (1983). Discussion approximation for an M=G=m queue. Operations Research, 31: pp. 304-321.

[4] L. Kleinrock (1975). Queueing Systems, volume 1, Theory. Wiley-Inter science.

[5] B. N. W. Ma and J. W. Mark (1998). Approximation of the mean queue length of an M=G=c queueing system. Operations Research, 43: pp. 158-165.

[6] Maplesoft, Inc. Maple 13. Waterloo, ON, Canada, 2009.

[7] M. Miyazawa (1986). Approximation of the queue-length distribution of an M=GI=s queue by the basic equations. J. Applied Probability, 23: pp. 443-458.

[8] S. A. Nozaki and S. M. Ross (1978). Approximations in _nite-capacity multi-server queues with poisson arrivals. J. Applied Probability, 15: pp. 826-834.

[9] R. Soft Design. Artifex v.4.4.2. R Soft Design Group, Inc., San Jose, CA, 2003.

[10] H. Takagi (1991). Queuing Analysis, volume 1, Vacation and Priority Systems, Part 1.Elsevier Science Publisher B.V..

[11] Y. Takahashi (1977). An approximation formula for the mean waiting time of an M=G=cqueue. J. Operational Research Society, 20: pp. 150-163.

[12] H. C. Tijms, M. H. V. Hoorn, and A. Federgru. Approximations for the steady-state probabilities in the M=G=c queue. Advances in Applied Probability, 13:186{206, 1981.

[13] L. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner (2009). A break in the clouds: towards a cloud definition. ACM SIGCOMM Computer Communication Review, 39(1).

[14] B. Yang, F. Tan, Y. Dai, and S. Guo (2009). Performance evaluation of cloud service considering fault recovery. In Cloud Computing, volume 5931 of Lecture Notes in Computer Science, pages 571-576. Springer Berlin Heidelberg.

[15] D. D. Yao (1985). Remaining the discussion approximation for the M=G=m queue. Operations Research, 33: pp. 1266-1277.

[16] N. Yigitbasi, A. Iosup, D. Epema, and S. Ostermann (2009). C-meter: A framework for performance analysis of computing clouds. In CCGRID '09: Proceedings of the2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, pages 472-477, Washington, DC, USA, 2009.

Corresponding Author

Urvashi Morya*

Madhyanchal Professional University, Bhopal

Indiaresearch05@gmail.com