Advances in Modelling and Analysis B
Vol. 61, No. 1, March, 2018, pp. 48-55
Journal homepage: http://iieta.org/Journals/AMA/AMA_B

Performance analysis of local, network and distributed file systems running inside user's virtual machines in cloud environment

Gopi Bhatt[1*], Madhuri Bhavsar[2]

[1] Department of Computer Science & Engineering, Institute of Technology, Nirma University, Ahmedabad 382481, India
[2] Department of Information Technology, Institute of Technology, Nirma University, Ahmedabad 382481, India

Corresponding Author Email: ce.gopi@adit.ac.in

https://doi.org/10.18280/ama_b.610108

Received: 29 March 2018
Accepted: 15 April 2018

Keywords:
cloud computing, file systems, virtual machines, queuing theory, performance analysis

ABSTRACT

Cloud computing, a recently developed paradigm, mainly focuses on resource allocation on demand. Operating Systems running in Virtual Machines can enhance their performances by adjusting resources as and when required. Due to this ever changing resource complexity, it becomes very difficult to model and analyze performance of some of the important components of Operating Systems, especially the File System.
This paper presents a model, based on Queuing Theory, for performance analysis of Local, Network and Distributed File Systems running in Operating Systems of user's VMs. This model takes into consideration parameters like average service time, average waiting time and VM migration time in file system's performance. It also takes into consideration different failures in Cloud environment like Virtual Machine Failures, Hypervisor Failures and Communication Failures. Each File system operation is considered as a service request sent by specific Virtual Machine to the Hypervisor. The performance is evaluated based on the average time taken to service the entire request. A numerical depicting the performance analysis based on this concept has also been illustrated.

1. INTRODUCTION

Virtualization has consistently improved hardware utilization, by providing applications, platforms and infrastructure on demand [1, 7, 13]. Apart from these advantages, virtualized systems are also complex, and thus difficult to model, measure and analyze. One of the important factor that causes this complexity, is sharing of hypervisor's hardware resources among virtual machines [2-3].

In any particular virtualized environment, a Hypervisor maps virtual disk images of virtual machines as regular files residing in the File system of the hypervisor. So, in this case, there are two different file systems – a hypervisor's file system and a VM's file system – both of them are totally unaware about each other's existence. These type of interdependent architecture, complicates the process of optimizing the parameters that enhance performance of VM's file systems. For example, nearly placed blocks in VM's File systems for faster access, may be kept physically at distant locations by the hypervisor's layer file system.

In another alternative is to map hypervisor's logical volume directly to virtual disk images of virtual machines (VMs). The virtual machine, completely ignorant of it, formats this virtual hard disk with the file system that is compatible with the operating system installed in that virtual machine. With a virtualized disk driver, the VM communicates with the emulated Disk I/O Controller provided in the abstraction layer of the hypervisor. As hypervisor's can host more than one virtual machine and the fact that there is a layer of abstraction complicates the situation. Due to multiple VMs running in the hypervisor, the emulated Disk I/O controller can receive more than one request at any given time to perform disk read or write operation on the actual physical storage. This may lead to a queue, where the requests need to wait for the current ongoing request to get completed. Hence, the average waiting time may have a significant impact on the overall performance of the file operations in virtual machines.

This paper takes into consideration the second mechanism discussed above, which maps hypervisor's logical volumes directly to virtual disk images of VMs for our analysis model. The distinguished feature of our analysis model are as follows: Multiple VMs will be share the underlying physical resource to perform their requests. A different read and write queue will be maintained at Disk I/O controller as both are diverse operations. Failure of Virtual Machines, Communication channels, hypervisor and VM migrations are also considered, providing much closeness to real virtualized environment. For performance analysis, the Operating Systems configured in user's Virtual Machines are considered to be of Linux environment.

Initial sections, provide performance analysis of local file systems, residing in VM's Operating Systems. Further sections, focus on performance modeling of Networked (NFS) & Distributed (DFS) File System, in this particular environment. This paper is organized as follows: Section 2 provides insight on the related work. Section 3 discusses our Analytical model for local, network and distributed file systems. In section 4, an illustration is presented. At the end, section 5 provides conclusion and future work.

2. RELATED WORK

Performance model of Cloud Computing resources based on Queue base has been a research focus due to its on demand changing resource environment. In [4, 13], the cloud center is modeled as an M/G/m/m+r queue with arrival rate and service rate generated using probability distribution. A request for resource is be further divided into subtasks [12], to provide a closer model to real time situations. In [6], performance analysis also takes into consideration real time failures like hypervisor failure, virtual machine failure and communication failure. In this paper, we have used many derivations from [6]. Storage is also considered as resource in Cloud environment. Virtual machines will be scheduling the storage resource during file operation. Scheduling of the virtualized I/O for storage resource for increasing performance has been highlighted in [5, 10], while [9] has provided the insight on the role of high performance disk image's role for performance analysis of virtual system's I/O operations.

Modeling file systems is a complex task as variety of file systems with different functionalities have been developed. For our work, based on the functionality, we have categorized file systems as local, network and distributed file system. Performance analysis of these file systems in non-virtual environment provides an insight on the parameter metrics used to optimize the performance. Comparing the workload [15] of different file systems help in understanding the impact on their metrics and their relation with overall performance. In [11, 14], impact of workload on metrics regarding NFS has been highlighted. The impact of scaling on performance of distributed file systems has been elaborated in [16]. Some more benchmarking features of File system have been provided in [17].

3. PERFORMANCE ANALYSIS MODEL

The performance model is the instance-based representation of how a system uses all kinds of its resources and manages different impacts on its performance. In case of File Systems, reading and writing files are the two basic operations. Therefore, the Performance Model majorly focuses on two parameters, T_{READ}: time required to read a file and T_{WRITE}: time required to write a file.

3.1. Local file systems

In case of Local File Systems, whenever a user process requests to fetch a file, VFS forwards this request to the underlying local file system, which signals device drive to fetch a particular block or set of blocks to the main memory.

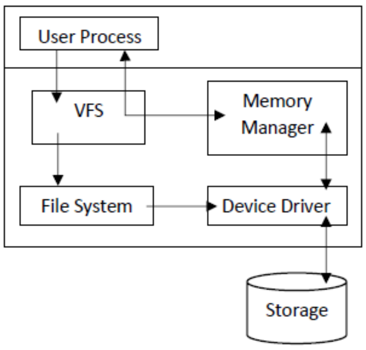

Figure 1. Working architecture of local file system, in linux environment

Thus, the File reading and writing time, in case of local file system can be summarized as:

$$T_{READ} = T_{READ\_REQUEST} + N_B \times T_{READ\_BLOCK} \tag{1}$$

$$T_{WRITE} = T_{WRITE\_REQUEST} + N_B \times T_{WRITE\_BLOCK} \tag{2}$$

where N_B is number of blocks to be read or written. Here and is the time taken by a request from user process to reach File System, which can be considered as negligible.

3.2 Network file system

In Network File Systems, a file is accessible from any node in the network, maintaining location transparency. This type of file access along the network generates significant amount of network delay involved during file block's transfer. Acknowledgements can be ignored as they are either piggybacked or considered as minor criterion, considering its size.

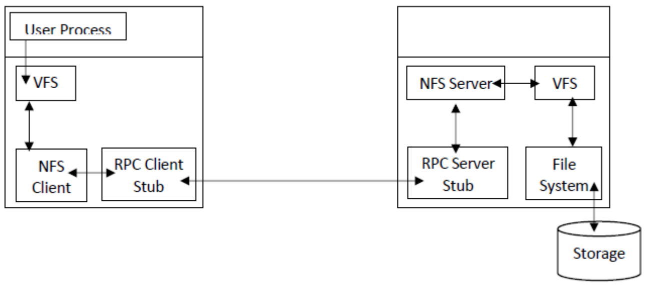

Figure 2. Working architecture of network file system, in linux environment

The read and write file operations are characterized by two independent variables of storage and network, i.e. time required to read/write a block with time required to transfer that block to the system requesting it. This can be expressed as:

$$T_{READ} = T_{READ\_REQUEST} + N_B \times T_{BLOCK\_READ} + N_N \times T_{NETWORK\_TRANSFER} \tag{3}$$

$$T_{WRITE} = T_{WRITE\_REQUEST} + N_B \times T_{BLOCK\_WRITE} + N_N \times T_{NETWORK\_TRANSFER} \tag{4}$$

where N_B is number of blocks to be read or written and N_N is number of packets to be transferred.

3.3 Distributed file system

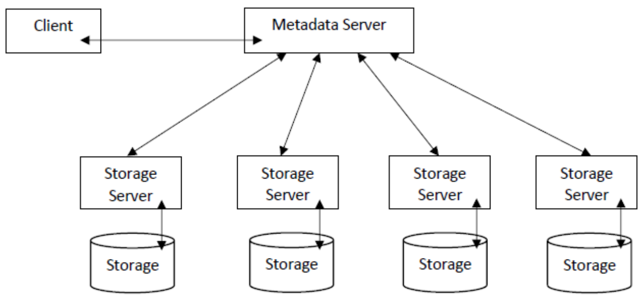

Figure 3. Architecture of distributed file system

The architectural model presented in this section is general for typical DFSs, which consists of Metadata Server, which handles location and access transparency and Storage Server which provides or stores the data.

To read a File, the client first contacts the Metadata Server. Metadata server translates the file name into the list of blocks IDs along with their location information. The client node communicates with the nearest Storage server to access the specific block. So, the File read operation consists of three parts: Communicate with Storage server for block information, receive block information and transfer of blocks along the network. T_{READ} can be expressed as:

$$T_{READ} = T_{READ\_REQUEST} + N_B \times (T_{BLOCK\_LOCATION} + T_{BLOCK\_READ}) + N_N \times T_{NETWORK\_TRANSFER} \quad (5)$$

$$T_{WRITE} = T_{WRITE\_REQUEST} + N_B \times (T_{BLOCK\_LOCATION} + T_{BLOCK\_WRITE}) + N_N \times T_{NETWORK\_TRANSFER} \quad (6)$$

$T_{BLOCK\_LOCATION}$ is the time, when client communicates with the name node to obtain block's metadata information. Its value can be neglected compared to other parameters.

4. THE PERFORMANCE & ANALYSIS MODEL IN CLOUD ENVIRONMENT

In Cloud Computing Environment, the hypervisor provides each hardware component as resource which is shared among virtual machines. The hypervisor maintains a separate Resource allocator (RA) for each hardware component. A Resource allocator (RA) can be considered as a queue, which handles and manages requests received from virtual machine.

In a typical Cloud setup each virtual machine is configured with their local virtual storage, but actually, they are sharing the physical storage of the hypervisor. Whenever, a process in any virtual machine requests to access a file from the local, network or distributed file system configured in that specific virtual machine, the Virtual Device driver forwards this request to the Resource Allocator (RA). In case of Local File System, only Storage Resource Allocator (SRA) will be accessed, where as in NFS and DFS both Network Resource Allocator (NRA) and Storage Resource Allocator (SRA) will be requested.

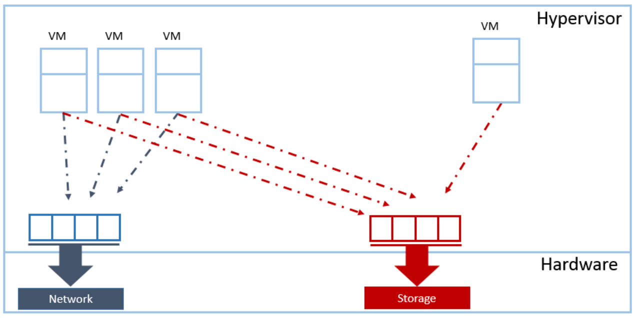

Figure 4. Queue based performance model of file systems running in vms, in a hypervisor

With a closer look at this Storage Resource Allocator (SRA) and Network Resource Allocator (NRA), they can be equated with single server queue, where virtual machine's file and network operation requests are received and stored for further service.

For our performance model, we have assumed a queue with the notation $M / M_{[k]} / n / m / FCFS$ where M : mean arrival rate, $M_{[k]}$: mean service rate for k tasks, *n*: number of hypervisors, *m*: buffer size, and *FCFS*: first come first serve policy.

Applying Queuing theory, and considering equations (1) and (2), the equations for file read and write, for local file system can be expressed as:

$$T_{READ} = T_{READ\_WAITING} + N_B \times T_{BLOCK\_READ} \quad (7)$$

$$T_{WRITE} = T_{WRITE\_WAITING} + N_B \times T_{BLOCK\_WRITE} \quad (8)$$

The equations for file read and write, after referring to equation (3), (4), (5) and (6), for Network and Distributed File System can be obtained as:

$$T_{READ} = \left(T_{READ\_WAITING} + N_B \times T_{BLOCK\_READ}\right) + \left(T_{N/W\_WAITING} + N_N \times T_{N/W}\right) \quad (9)$$

$$T_{WRITE} = \left(T_{WRITE\_WAITING} + N_B \times T_{BLOCK\_WRITE}\right) + \left(T_{N/W\_WAITING} + N_N \times T_{N/W}\right) \quad (10)$$

Here, we are assuming that there are separate queues maintained for Read, Write and Network requests.

3.2 Assumptions

Based on the model presented above, we make following assumptions:

(a). A Virtual Machine's read or write request will be considered as a single task, and cannot be divided into further subtask.

(b). Each file operation request will be treated as an independent event and their arrival rate λ will be considered as independent and identically distributed (i.i.d) random variable arriving according to a Poisson Process.

(c). The average service rate of the physical server j follows an exponential distribution with the parameter μ_j . The service rates of one physical server is independent of the other physical server.

(d). A Virtual Machine can at a time send one file operation request. The maximum number of requests that can arrive at any given instance is equal to maximum number of virtual machines n_{MAX}, on any hypervisor.

(e). The failure rate of Virtual Machines is in accordance with a Poisson process with parameter λ_v (The parameters mentioned in assumption e, f, g, h, i and j will be taken into consideration, for Network and Distributed file system only and not for Local file systems)

(f). The migration and recovery rate of VMs is an i.i.d following an exponential distribution with the parameter μ_v

(g). The failure rate of Hypervisors is generated using a Poisson process with parameter λ_p.

(h). The recovery rate of a Hypervisor is an i.i.d with exponential distribution with parameter μ_p

(i). The failure rate of Communication link is also generated using Poisson process with parameter λ_c

(j). The recovery rate of Communication channel is i.i.d with exponential distribution with parameter μ_c

(k). The maximum number of Virtual Machines in a Hypervisor is n_{MAX}.

(l). The Request queue or the buffer size is *m* which is very much greater than *n*

(m). If the number of requests (queued + arrived) is greater than the buffer size, then the latest arrived requests will be dropped in LIFO manner.

3.3 Markov Model

A file read or write request comes directly from the Virtual Machine to the hypervisor's Storage resource request queue. The moment of the request queue buffer can be marked as Markov points. Such process can be modeled as Markov process with the state $n(1,2,3\ldots m)$, where *n* represent the number of file operation request in the buffer. Figure.5, Represents the initial stage, where the buffer is empty. For *i* number of requests arriving simultaneously, there will be a transition to state *i* with a transition probability of $P(i)\lambda$. According to assumption (d) and (k) the maximum transition probability that can be obtained is $P(n)\lambda$. The transition probability $P(i,j)$ can be categorized into three following sections;

For $i-1>j$ or $i=j$, $P(i,j)=0$

For $i<j$, $P(i,j)=P(j-i)\lambda$

For $i-1=j$, $P(i,j)=\mu_n$ where μ_n is the service rate at state n.

A Markov process chain consisting of all possible transitions at any intermediate state *n* has been depicted in Figure 6.

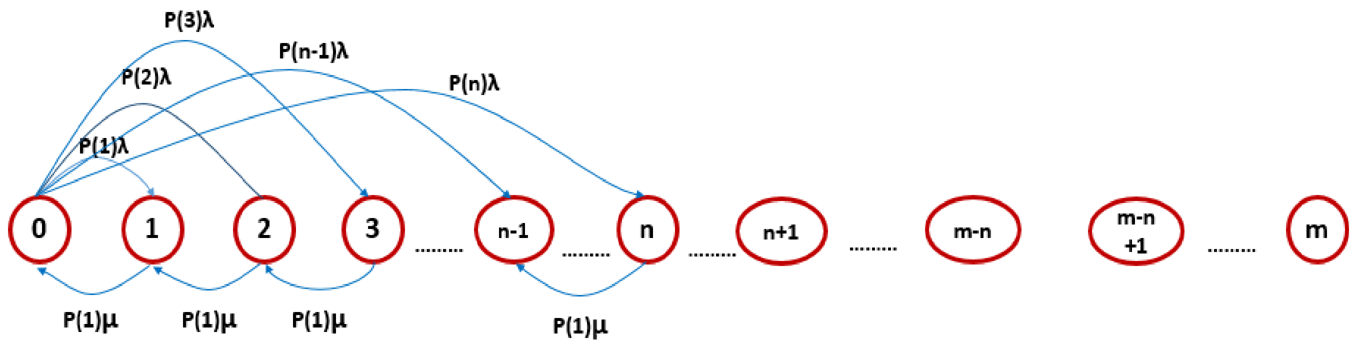

Figure 5. Markov chain process model depicting the initial phase of the resource allocator queue

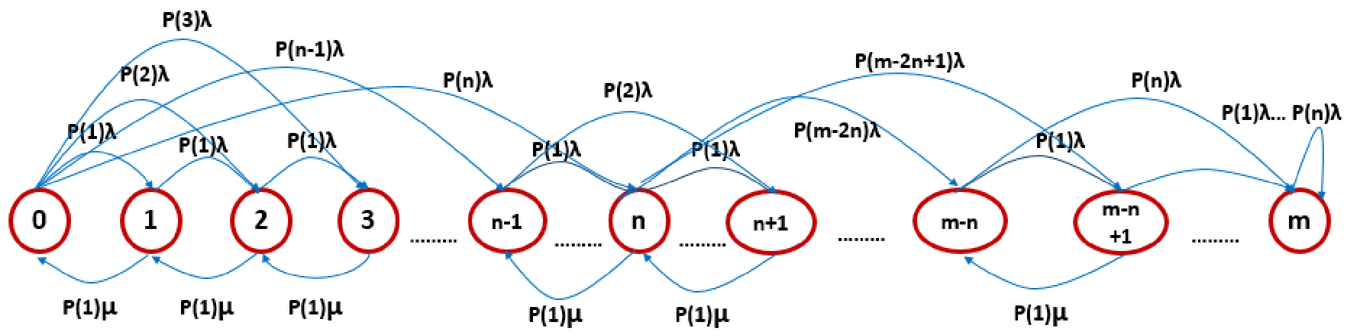

Figure 6. Markov chain process model depicting the general phase of the resource allocator queue

The transition probability for a steady state probability $q(n)$, can be defined as:

$$P(i,j)=P[q(n+1=j)|q(n)=i] \qquad (11)$$

Deriving the equation of $q(n)$ as:

$$q(1)\times P(1,0)=\sum_{j=1}^{m}\left(\ q(0)\times P(0,j)\ \right) \qquad (12)$$

$$q(i+1)\times P(i+1,i)+\sum_{j=0}^{i-1}\left(\ q(j)\times P(j,i)\ \right)=\sum_{j=i+1}^{m}\left(\ q(i)\times P(i,j)\ \right)$$
$$(i=1,2,3\ldots m) \qquad (13)$$

$$q(m)\times P(m,m-1)=\sum_{i=0}^{m-1}\left(\ q(i)\times P(i,j)\ \right) \qquad (14)$$

3.4 Parameter metrics

A Calculation of Arrival Rate:

For local, Network and Distributed File System, according to assumption (b), the file operation request arrival rate is considered as i.i.d generated with Poisson Process, stated as:

$$\text{Arrival rate} = \lambda \qquad (15)$$

In real time systems, majority of file operation requests are fulfilled by the cache. Hence taking cache into consideration, the file operation request arrival rate becomes:

$$\text{Arrival rate} = \lambda_{CACHE} = \left[P_r(1-h_r)+P_wF_w\right]\times\lambda \qquad (16)$$

where h_r is the hit rate, so, $1-h_r$ is the miss rate. P_r is the read probability, and $P_w = 1-P_r$ is the write probability when the cache is either dirty or is full with a probability F_w

B. Calculation of Service Rate:

For local file systems, according to assumption (c), the average service rate of the storage resource allocator of the hypervisor is μ_s. The service rate can be considered as independent and identically distributed random variable.

$$\text{Service rate} = \mu_s \qquad (17)$$

In case of Network or Distributed File System, the Metadata Server or the Storage Server may reside in different hypervisors, and the hypervisors may be sharing the physical resources as resource pool. Denote R_j as the service rate of the Storage resource allocator, for hypervisor j. The average service rate to process each $n < n_{MAX}$ requests in *h* hypervisors is given by:

$$\mu_s(n) = \sum_{i=1}^{h} R_i(n) \qquad \text{Where } 0 < n < m \qquad (18)$$

Similarly, we can calculate the average service rate of the Network Resource Allocator to process *n* requests given by:

$$\mu_n(n) = \sum_{i=1}^{h} N_i(n) \qquad \text{Where } 0 < n < m \qquad (19)$$

where N_j is the service rate of the Network resource allocator, for hypervisor j and μ_n is the average service rate of the Network.

C. Average Waiting Time:

Suppose there are $n(0 < n \le n_{MAX} < m)$ file operation requests pending in the Storage resource allocator queue at any given time interval *t*. Assume that r number of requests arrive at a time. If $r < (m-n)$ then all the requests will be added into the storage resource allocator queue, else $r-(m-n)$ requests will be dropped according to assumption (m). So, the probability that a File operation request will be dropped is given by:

$$P_{drop} = \sum_{i=0}^{r-(m-n)} q_i \qquad (20)$$

Moreover, if $\lambda_s \le \mu_s$, that is request arrival rate is less than or equal to the service rate, then the services will be served immediately. Denote T_w as the waiting time of the *r* new request to be served, where *n* requests are already waiting in the queue. Its cumulative distribution function can be expressed which follows gamma distribution with parameter μ_s can be expressed as:

$$W_T(T_w \le t) = P(t) = \begin{cases} 0 & \lambda_s \le \mu_s \\ \dfrac{\Gamma(r+n-m,\mu_s t)}{(r+n-m)!} & \lambda_s > \mu_s \end{cases} \qquad (21)$$

So, accordingly, the mean of this gamma distribution function is:

$$E[T_w] = \frac{r+n-m}{\mu_s} \qquad (22)$$

Thus, the average waiting time for r requests in storage resource allocator can be obtained as follows:

$$T_{BLOCK\_WAITING} = \begin{cases} 0 & \lambda_s \le \mu_s \\ \displaystyle\sum_{i=0}^{n+r} q_i \times \left(\frac{r+n-m}{\mu_s}\right) & \lambda_s > \mu_s \end{cases} \qquad (23)$$

Similarly, we can also calculate the average waiting time for Network in Network resource allocator queue as:

$$T_{N/W\_WAITING} = \begin{cases} 0 & \lambda_n \le \mu_n \\ \displaystyle\sum_{i=0}^{n+r} q_i \times \left(\frac{r+n-m}{\mu_n}\right) & \lambda_n > \mu_n \end{cases} \qquad (24)$$

where $TN_{WAITING}$ is the average waiting time in Network resource allocator to grab the Network resource.

D. Average Completion Time:

The completion time is considered to be the time required for processing the request. This processing time is affected by catastrophic situations like hypervisor failure, communication failure and Virtual machine failure (in DFS or NFS where data nodes are residing in different hypervisors sharing the resource pool). The recovery time and migration time, of all these failures need to be taken into consideration while calculating the request completion time.

D.1 Ideal average Service Time (Assuming No failures)

Servicing time is calculated as the time required to perform the specific task. In this case, servicing time is the time required to read or write a block. So, the Service time to read a block i.e. $T_{BLOCK\_READ\_IDEAL}$, can be expressed as:

$$T_{BLOCK\_READ\_IDEAL} = \frac{Block\_Size}{Disk\_IO\_Rate} \qquad (25)$$

Similarly, the service time to write a block can be calculated as:

$$T_{BLOCK\_WRITE\_IDEAL} = \frac{Block\_Size}{Disk\_IO\_Rate} \qquad (26)$$

Data communication time depends on transmission time, bandwidth of the hypervisors and the virtual machines sharing the communication link. Assume the bandwidth of *j*th hypervisor is B_j, and according to assumption (d), n_{MAX} virtual machines are evenly sharing this bandwidth. The bandwidth of each virtual machine is expressed as $\frac{B_j}{n_{MAX}}$.

Let D_{size} is the data packet size that needs to be transmitted. The data transmission time T_{N/W_IDEAL} of the virtual machine residing in *j*th hypervisor is expressed by:

$$T_{N/W\_IDEAL} = \frac{D_{size}}{n_{MAX} \times B} \qquad (27)$$

D.2 Virtual Machine Failure

Virtual Machine Failure will affect the File System performance in case of Network or Distributed File Systems, where data needs to be accessed through different Storage Servers. In a typical case, when client requests to access a File over DFS or NFS, the request is sent to the Metadata Server and it forwards it to corresponding Storage Server. Storage Server whether residing in same or different hypervisor, when fails, will be migrated to another hypervisor and then will be started again. Denote $V_F(l)$ the probability that *l* failures occur between time interval $[0,t]$, then

$$V_F(l) = \frac{(\lambda_V t)^l}{l!} \times e^{-\lambda_v t} \qquad l = 0,1,2\ldots \qquad (28)$$

Denote μ_V as the migration and recovery time of the virtual machine. According to assumption (f), μ_V is i.i.d with exponential distribution. So the average failure and recovery of the virtual machine can be expressed as:

$$E[V_R] = \sum_{l=0}^{\infty}\left(V_F(l) \times \sum_{i=0}^{l} \mu_V^i\right) \qquad (29)$$

Solving the above equation, we have:

$$E[V_R] = \frac{\lambda_v t}{\mu_v} \qquad (30)$$

D.3 Hypervisor Failure & Recovery

Let $H_F(l)$ represent the probability that l failures occur during the time interval $[0,t]$, which is an independent event and is derived using Poisson process as:

$$H_F(l) = \frac{(\lambda_p t)^l}{l!} \times e^{-\lambda_p t} \qquad l = 0,1,2\ldots \qquad (31)$$

Denote $H_R(l)$ is the recovery time of *l* failures of the hypervisor. The recovery time which also includes VM migration time, as per assumption (h), is i.i.d generated over exponential distribution with parameter μ_p. The mean recovery time can be expressed as:

$$E[H_R] = \sum_{l=0}^{\infty}\left(H_F(l) \times \sum_{i=0}^{l} \mu_p^i\right) = \frac{\lambda_p t}{\mu_p} \qquad (32)$$

D.4 Communication Link Failure & Recovery

According to assumption (i), failures of communication link occur are in accordance to a Poisson process with parameter λ_c. Let $N_F(l)$ denote the probability that the communication link fails for *l* times between time interval [0, t], which can be expressed as:

$$N_F(l) = \frac{(\lambda_c t)^l}{l!} \times e^{-\lambda_c t} \qquad l = 0,1,2\ldots \qquad (33)$$

Denote, μ_c, as per assumption (j), as the recovery time of the communication channel and assuming that at any given time, $c \quad (o < c < n_{MAX})$ virtual machines are accessing the communication channel at a given time. The mean recovery time can be calculated as:

$$E[N_R] = \sum_{c=1}^{n_{MAX}}\left(\binom{n_{MAX}}{c} \times [p_t \times (1-p_t)] \times T_{N/W\_IDEAL}\right) + \sum_{l=0}^{\infty}\left(N_F(l) \times \sum_{j=0}^{l} \mu_c^j\right) \qquad (34)$$

where p_t: the probability that virtual machine will use the network to transmit the packet.

Which can be further solved as:

$$E[N_R] = \sum_{c=1}^{n_{MAX}}\left(\binom{n_{MAX}}{c} \times [p_t \times (1-p_t)] \times T_{N/W\_IDEAL}\right) + \frac{\lambda_c t}{\mu_c} \qquad (35)$$

Using all the equations from (22) to (33), the parameters of $T_{BLOCK\_READ} \mid T_{BLOCK\_WRITE}$ and $T_{N/W}$ can be obtained as:

$$T_{BLOCK\_READ} = T_{BLOCK\_READ\_IDEAL} + E[V_R] + E[H_R] \qquad (36)$$

$$T_{BLOCK\_WRITE} = T_{BLOCK\_WRITE\_IDEAL} + E[V_R] + E[H_R] \qquad (37)$$

$$T_{N/W} = T_{N/W} + E[N_R] \qquad (38)$$

In case of local file systems, Hypervisor Failure, Virtual Machine Failure and Communication Link Failure will not affect the servicing time.

5. NUMERICAL RESULTS

5.1 Analytical example

Suppose the capacity of the Storage Resource Allocator queue is 20, and there are 5 hypervisors running. The files are divided into blocks each of size 64KB and the Disk IO transfer rate is 128 Mb/s. The bandwidth of the communication link is 100 Mbps, with a network packet size of 64K. The file read or write request arrive in accordance with the Poisson process with the rate λ_1 = 0.75 sec, λ_2 =1.00 sec, and λ_3 =1.25 sec.

5.2 Average completion time in local file system

In case of Local file system, VM, Communication link and other hypervisor failures does not affect the average reading and writing time of the files. Hence, the time required to perform File read or write operation depends on queue length and the rate at which the requests arrive.

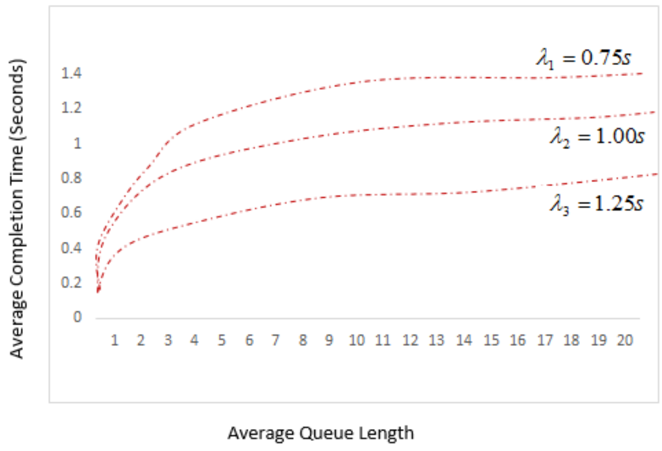

Figure 7. Average completion time for local file systems, with varying arrival rate

5.3 Average completion time in network or distributed file system

In case of Network & Distributed File Systems, the failure of Hypervisor, Virtual Machines and Communication links matter, when Storage Servers are located in different hypervisors. The failure rates of Virtual Machines (λ_V) and Network Link (λ_C) are both 0.005 s^{-1}. Hypervisor failures are rare and hence the failure rate of Hypervisor (λ_P) is 0.00001 s^{-1} .The migration and recovery rate of the Virtual Machines (μ_V) and communication link (μ_C) are μ_V =0.03 s^{-1} and μ_C =0.04 s^{-1} and the recovery rate of hypervisor μ_p = 0.003 s^{-1}.

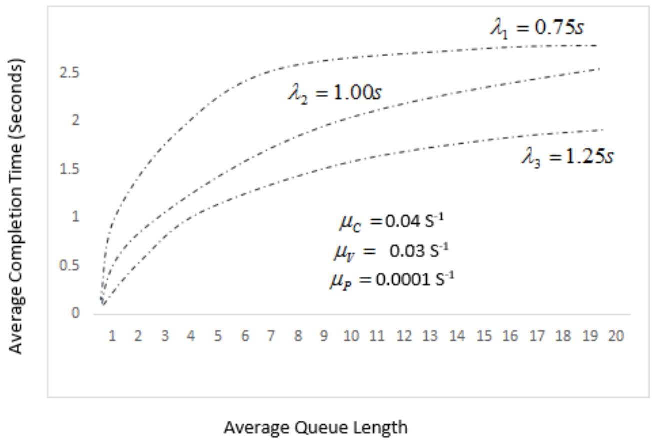

Figure 7. Average completion time for network & distributed file systems, with varying arrival rate

6. CONCLUSION AND FUTURE WORK

In this paper, we analyzed the performance of local, network and distributed file system running in user's virtual instance using queue model. We tried to provide more realistic analysis by including different type of failures that occur in real time environment. We also presented a numerical calculation to show the overall effectiveness of our queue model in analyzing the performance of File Systems running in Virtual instances, in any Cloud environment.

In this analysis, we focused on requests coming from different virtual instances to the resource allocator queues of the hypervisors. In some situations, hypervisors also use the physical resources for their own execution. In the next, we will try to analyze bottlenecks created by these type of situations on file system's performance.

REFERENCES

[1] Madni SHH, Latiff MSA, Coulibaly Y, Abdulhamid SM. (2016). Resource scheduling for infrastructure as a service (IaaS) in cloud computing: challenges and opportunities. Journal of Network and Computer Applications, Elsevier 68: 173-200.

[2] Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. (2009). Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Gener Comput Syst 25(6): 599–616.

[3] Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I. (2009). Above the clouds: a Berkeley view of cloud computing, electrical engineering and computer sciences. University of California, Berkeley.

[4] Khazaei H. (2012). Performance analysis of cloud computing centers using M/G/m/m+ r queuing system. IEEE Trans Parallel Distrib Syst 23: 936–943.

[5] Seelam SR, Teller PJ. (2007). Virtual I/O scheduler: A scheduler of schedulers for performance virtualization. In ACM VEE'07.

[6] Liu XD, Tong WQ, Zhi XL, Fu ZR, Liao WZ. (2014). Performance analysis of cloud computing services considering resources sharing among virtual machines. The Journal of Supercomputing 69(1): 357-374.

[7] Mustufa S, Nazir B, Hayat A, Khan AR, Madani S. (2015). Resource management in cloud computing: Taxonomy, prospects, and challenges. Computer & Electrical Engineering, Elsevier 47: 186-203.

[8] Le D, Huang H, Wang H. (2012). Understanding performance implications of nested file systems in a virtualized environment. In Proceedings of the Tenth USENIX Conference on File and Storage Technologies (FAST ’12).

[9] Tang C. (2011). Fvd: A high-performance virtual machine image format for cloud. In Proceedings of the 2011 USENIX conference on USENIX annual technical conference, Portland, OR.

[10] Shenoy PJ, and Vin HM. (1997). Cello: A disk scheduling framework for next generation operating systems. In Proceedings of ACM SIGMETRICS Conference.

[11] Ellard D, Ledlie J, Malkani P, Seltzer M. (2002). Everything you always wanted to know about NFS trace analysis, but were afraid to ask. Technical Report TR-06-02, Harvard University, Cambridge, MA.

[12] Losup A, Ostermann S, Yigitbasi MN, Prodan R, Fahringer T, Epema DHJ. (2011). Performance analysis of cloud computing services for many-tasks scientific computing. IEEE Trans Parallel Distrib Syst 22: 931–945.

[13] Suresh Varma P, Satyanarayana A, Sundari R. (2012). Performance analysis of cloud computing using queuing models. In: 2012 international conference on cloud computing, technologies, applications and management, pp. 12–15

[14] Leung AW, Pasupathy S, Goodson G, Miller EL. (2008). Measurement and analysis of large-scale network file system workloads. In Proceedings of the USENIX Annual Technical Conference (ATC ’08).

[15] Roselli D, Lorch JR, Anderson TE. (2000). A comparison of file system workloads. In Proceedings of the Annual USENIX Technical Conference.

[16] Howard JH, Kazar ML, Menees SG, Nichols DA, Satyanarayanan M, Sidebotham RN, West MJ. (1988). Scale and Performance in a Distributed File System ACM Trans. Computer Systems 6(1): 51-81.

[17] Smith KA, Seltzer MI. (1997). File system aging - increasing the relevance of file system benchmarks. In Proceedings of the 1997 International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 1997).