Fault Tolerant - Optimal Resource Allocator for Cloud

Khiruparaj T. P., Abishek Madhu. V, Ponsy R. K. Sathia Bhama

Abstract*: Cloud Computing is mainly attracted by people for its unlimited storage space and worldwide accessibility from anywhere and anytime. The data that is stored in the cloud has to be retrieved in a faster manner as well as without any faults. Content Delivery Networks dominates Cyberspace traffic heavily due to its increasing demand. Resource allocation plays a vital role in determining the performance of the cloud. Over allocation leads to wastage which can be used for instances that are running short of resources. This work proposes an optimal resource allocation through Genetic algorithm. They help in increasing the download speed which in turn would be helpful in controlling traffic to greater extent. Recently cloud downloading Services have been emerged, in which cloud storage hoards the user interested content and updates the cloud cache. This process takes place in two modes, server mode and the helper mode. The proposed Resource Allocation Policy can well support the server mode of processing by the way it can increment the download speed with dynamic load balancing and fault tolerance.

Keywords: Cloud computing, file downloading, Genetic Algorithm, peer-to-peer, Resource optimization.

I. INTRODUCTION

File downloading and content dissemination is an intriguing research problem because it requires high bandwidth. Nowadays, cyber space traffic is often reported to be dominated by file sharing systems. The mechanisms to disseminate the content include Content Distribution Network (CDN) and Peer-to-Peer [1], [2] (P2P). CDN is a traditional method based on the deployment of servers at the border of the network, nearer to the access points. The limitation of CDN is its lack of scalability. Because, the capacity of the server becomes a constricting factor when large number of concurrent peer requests arrive at the server. Following the CDN systems, P2P [3] - [5] became popular among private file sharing communities and was followed by many other file sharing and content distribution systems [6]. Therefore, the advantage of a P2P system is that, with increase in the peer population the capacity of the downloading service [7] will also increase automatically, making the system highly scalable. But the disadvantage is that there is no stability in peer connections, as they may join or leave the network at any point of time. In this P2P system, no dedicated distributed servers exist for storing the files. Therefore, peers who request files that are not frequently accessed, suffer from a low download rate [8]. To recover from this, a cloud downloading service is positioned on or spread over the P2P downloading systems to improve the downloading experience. In case of mobile devices, due to the limited downlink bandwidth, memory, CPU, unstable availability and limited battery power, it is very hard to rely on an entirely P2P based approach to realize high speed downloads [9]. Hence, there is a need for the development of a cloud based optimal resource utilization system. This helps in assessing the performance of cloud by determining by how well the resources of the cloud server get utilized. The cloud performance may be determined the resource utilization ratio of the cloud server. In this work we propose two modes for cloud server operation. The first mode is a helper mode that tries to service requests that are hosted using cloud hosting service. The second mode is initiated when the interested content is not serviced by cloud. Our work proposes to identify the missed interested content and modifies the cache space so that the awaiting client request are serviced immediately. We call this the client-server mode. An alternative mode in our proposed system is the peer to peer mode, which does not block the unchached / missed content request. Instead, the cloud redirects the request and fetches the content from its peers. A system modelled as both client to server mode and P2P mode produced efficient results. The rest of the paper is organized as follows: In Section 2, we summarize the related work. In Section 3 and Section 4, we discuss about the proposed system. In Section 5, the implementation details are given. In Section 6, we use simulation results to validate our model. Section 7 includes the conclusion part.

II. RELATED WORK

Yipeng Zhou et al [10], proposed an algorithm that helps in fetching video content using two approaches one is the helper mode where the server will function as usual by servicing all the arrived request at the earliest and second is the server mode in which if a preferred video is not available in the cloud storage,

Manuscript published on November 30, 2019.
* Correspondence Author

Khiruparaj T.P*., is with the department of Computer Technology, Madras Institute of Technology – Anna University, Chennai, India. Email: khiruparaj@gmail.com

Abishek Madhu, is with the department of Computer Technology, Madras Institute of Technology – Anna University, Chennai, India. Email: abishekmadhu1999@gmail.com

Ponsy R.K. Sathia Bhama, * is working as an Associate Professor in the department of Computer Technology, Madras Institute of Technology – Anna University, Chennai, India., Email:
ponsy@mitindia.edu

the request will be put in awaiting state until the cloud cache gets updated with the requested content.

Liying Wu et al [11], proposed a system that determines the fault tolerance level on the fly through file access frequency ratio, the time and count for file tolerance conversions that are stored in frequency table.

Ravi Jhawar et al [12], proposed a measurement of fault tolerance in cloud that provides flexibility and availability to users application through virtual machines in cloud. They assess the disruption of server components, their network traffic and their energy through their infrastructure [13] and propose a correlation between failures and their impact on end user's.

Narasinga Rao et al [14], proposed the work on parallel downloading of files at the client level in which mirror sites enable requests from the clients to be processed by any one of available servers, thus reducing the load at individual servers. Normally a client's request will be serviced by a single mirror site. But a new approach has been suggested for the client to access a file from multiple mirror sites in parallel [15] that speedup the downloading mechanism thereby reducing the bottleneck onto single mirror server.

Bin Fan et al [16], states that in Bittorrent a client download a part of a huge file becomes a seeder and starts hosting the same content as origin servers. They even provide attractive offers by tracking their upload to download contents. The origin server decides dynamically based on its performance whether to serve the request or to redirect the request to its nearby peer. Some of the drawbacks for systems based on this mechanism are low transfer ratio of peers, bandwidth restrictions and security issues.

Ricky W. Butler et al [17], proposed about the concept of Architectural Level Fault Tolerance, its main motive is to state the principle and identify the trade-offs between the variety of techniques that are used to achieve good fault tolerance.

Atsushi Kawano et al [18], states that a proxy can handle the client request for downloading a file, by retrieving and ordering the segmented partitions from various mirror servers to the requested end user. But the drawback in this system is the client request is handled by a single node and the request is distributed to multiple nodes via this node, the coordination overhead on the server is high followed by a situation of bottleneck developing at the request receiver.

Deng et.al [19] proposed a new multi-level K-cut graph portioning algorithm for minimizing the volume of data transfers across data centers. In a distributed environment where the data centers are geographically distributed, the movement of input and intermediate datasets causes latency that affects the workload execution. The method also uploads load balancing and fixed data constraints of the workload execution with a consideration that the resources are homogenous and unlimited so that the input data is always available for the tasks to be executed.

In [20] Gideon Juve et al considered different scientific workloads for assessing their performance. The I/O intensive, memory-intensive and computational intensive characteristics are observed for every workload considered. It also includes the analysis of inefficiency caused during execution by different factors. The profiling of such workloads can be useful for selecting algorithms for scheduling and resource provisioning and further analysis of workload performance in different environments.

Sucha Smanchat et al in [21] have elaborated techniques for scheduling in the cloud. The resources considered are VM instances. The structure of the workload is given more importance because of dependencies between them and also considers the difficulties in processing them in parallel instances. To estimate workload makespan, cost and for finding the critical path of the workload, the structure of the workload plays an important role. In [22], Ewa Deelman et al described the design, development, and evolution of the Pegasus Workload Management System where workloads from different scientific domains like astronomy, seismology, and bioinformatics are considered. The work emphasizes scalable workload execution that is I/O intensive, data-intensive and computation-intensive through a variety of computing environments.

Om kumar et.al [23], had proposed core value based workload consolidation technique to assess over loaded and under loaded servers. They then perform fuzzy migration which improves performance of the instances. The fault tolerant module finds an instance that can accommodate and execute the workload without any disruption.

Many works cited above have been explored to conclude that there is a significant delay in using FTP for downloading contents. We propose a Genetic algorithm that reduces the overheads in management and enhances the performance by minimizing the coordination effort.

III. PROPOSED SYSTEM

Cloud Technology is used for storing files of large sizes and must be accessible from anywhere. Resource utilization polices of the cloud determines the download speed and their fault tolerance capacity. Traditional file download mechanisms involve HTTP based mechanisms. We propose to improve resource utilization through Optimal Resource Allocator (ORA) that uses Genetic algorithm to download large and un popular files available over Cloud with dynamic load balancing and fault tolerance. These are the client-side (ORA – P2P) request adapter, server-side (ORA – P2P) load balancer, server mode adapter and the file retriever.

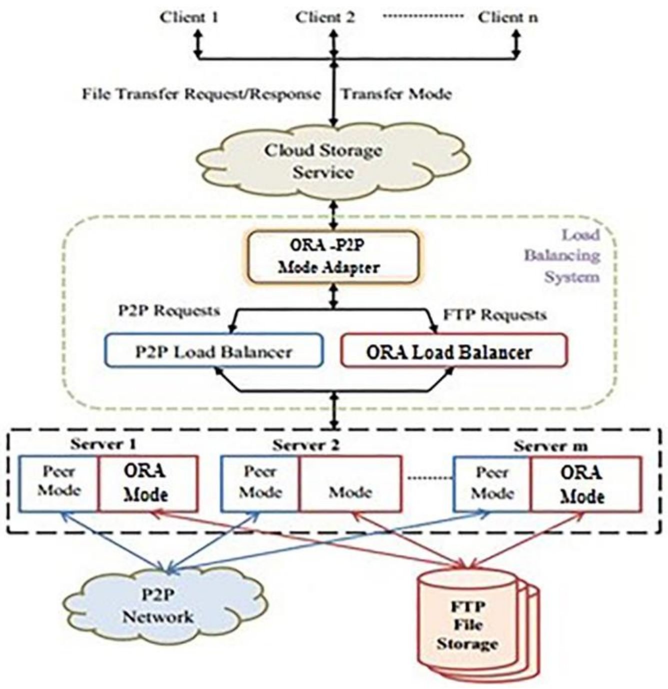


Fig. 1.Architecture of the proposed system

The client-side request adapter module decides where to send the client's file request within the server. It is sent to the P2P load balancer if the client is on a P2P network or to an ORA load balancer if client uses the FTP protocol. Also it adapts the request from P2P to FTP and vice-versa in case there is a server overload in any one of these network types. It reverses this change once the required data is cached and is to be sent back to the client. The server-side load balancer module resides in-front of the data servers and acts a gateway between client and server systems. Its only purpose is to re-route the client requests. The P2P Load Balancer connects the client to the optimal peers with good bandwidth and transfer ratio, while the ORA Load Balancer connects the clients to different Servers/Server Databases, so as to not overload the server network. The server mode adapter resides within the server systems itself. It consists of two modes of operation: the peer mode and the ORA mode. The modes are chosen dynamically based on the client request, the type of permissible traffic, server load and so on. Unlike the client-server model of the ORA mode, in the P2P mode, the server simply acts as one of the seeding peers in a P2P network, through which the clients can connect to other peers. In the file retriever module whose algorithm is given in Figure 2 and 3 serves as the portal through which the files are transferred between the server and the client.

```
BEGIN
  If (server load (L) < network threshold (T))
    If (user requests file in ORA mode)
     If (mode is P2P)
          Switch to ORA mode.
     End If
     Connect to the Server.
     Search the server for the requested file.
     Retrieve the located file and provide it to user.
     Disconnect after files are downloaded.
    Else
     CALL P2P();
    End If
    Else
     //transfer file using P2P mode
     CALL P2P();
    End If
END
```

Fig. 2. File Retriever Module

```
Function P2P():

BEGIN
    If (mode is ORA)
     Switch to P2P mode.
    End If
    Refresh peer information;
    Share peer information with client;
    If (file available with other peers)
     Disconnect from client.
    End If
    Exit Program
END
```

Fig. 3.P2P() Function Call

Also we propose to implement a switching mechanism that allows the server to act as both as server machine or a peer machine that can simply act as a connection between other nodes in the network, thus enabling support for peer to peer based downloading in the cloud environment, thereby reducing the load on the server as well as ensuring the availability of data. The ORA mechanism coupled with P2P switching thus allows a very fast download speed throughout the network, while imposing a minimal computational overhead in the form of the switching mechanism, which would be negligible when compared to the improvement in the network download speed and the availability of data from the server.

IV. PEER TO PEER MODE

In the peer to peer mode, the P2P protocol is used for downloading the files instead of the standard TCP. This allows the clients to make use of the resources available among themselves to share a file with each other and so reducing the load on the server.

In this case, the server that receives the download request simply connects the client machine to other clients via the tracker list, or it becomes one of the peers that connect with the requesting client to service its download request. The major difference between the server and the other systems is that, the server peer would have the entire file and so becomes a seeder, while the other peers may or may not be having the entire file and so can act as both seeders as well as mutual leechers. Our performance is measured by calculating the throughput as given below:

P denotes the peers on the network. File size, total Files and the cached files on the network are represented as F, T and S. The peer upload capacity through U. The upload capacity of the Server (that acts as a source of a file) is L. The cloud upload capacity is denoted by H. The maximum capacity or the throughput in the peer to peer mode is calculated as

$$C = \min\left\{L, U + \frac{L + H\left(\frac{P-1}{P}\right)}{P}\right\} \quad (1)$$

If the content is already available in the cache, the maximum throughput is

$$C = \min\left\{L + H, U + \frac{L+H}{P}\right\} \quad (2)$$

The above results are achieved using distributed scheduling using the parameters used in Eq. (1).
We determine a fitness function to further asses the performance of the cloud by
Fitness = (tvm.get(i). zvalue-((treq.get(j). zvalue + distances.get(j).get(i))); (3)
The algorithm shown in Figure 3 is for retrieving a file stored in the cloud network which supports both ORA and P2P based file transfer from a network of virtual storage containers. It first checks for the server load as per the requirements of the download request and the file availability. If the Server Load (L) is below the tolerable threshold limit that the servers can handle, then the file is distributed by means of ORA mode.

The P2P mode may be used at this stage only if the user has explicitly requested the P2P mode be used.
If on the other hand the server has achieved its maximum limits in serving download requests and no other alternate servers can be allocated, then the mode of download automatically switches to the P2P mode. But the switching of modes in the server side is not enough as the client may not have support for or may not desire P2P mode.
Therefore, an intermediary system resource is allocated for storing the acquired files for a temporary time in a memory cache, so that the user may be able to download the file via normal mode of file transfer. An alternative to this method of allocating a temporary storage is to provide P2P support to the client over the internet by making used of the Software as a Service (SaaS) model of cloud computing.

In this method, the technology used for P2P is loaded on to a virtual machine and it is served as a user interface for the client over the internet in an abstract manner. Thereby, allowing any web browser to act as a P2P client. The Figure 4 gives the algorithm for the P2P function call that was mentioned in the algorithm for File Retriever.

V. IMPLEMENTATION

The implementation part consists of configuring a FTP based client-server File Exchange mechanism. It makes use of the FileZilla FTP server for the server machine which would be hosting the data for various users/clients to download from and the client-server communication mechanism will be done using Java programming language with the help of the collective basic network protocols API known as Apache Commons-Net APIs from the Apache Software foundation. The main inputs, when it comes to client-server communication via FTP are, the server name, which during the testing phase will be the same as the machine name as the client, and hence will be termed as local-host with the default server IP of 127.0.0.1 and the user name will be a one assigned by the administrator through the FileZilla server software along with their password and the folders and files that would be accessible to those users. Once these credentials are established, the user then chooses the file that they want to download from the server location and initiate the download by giving a local file location for storage purpose.

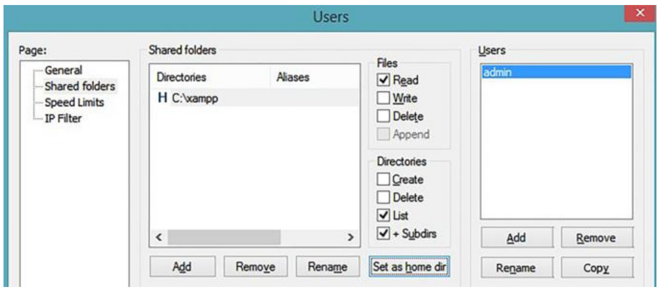


Fig. 4.Server File-System Configuration

Figure 5 shows the configuration of a FileZilla Server while Figure 6 shows the hypervisor running three virtual machines with different operating systems and FileZilla.

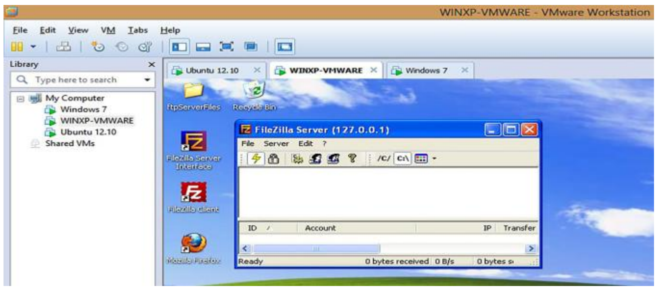
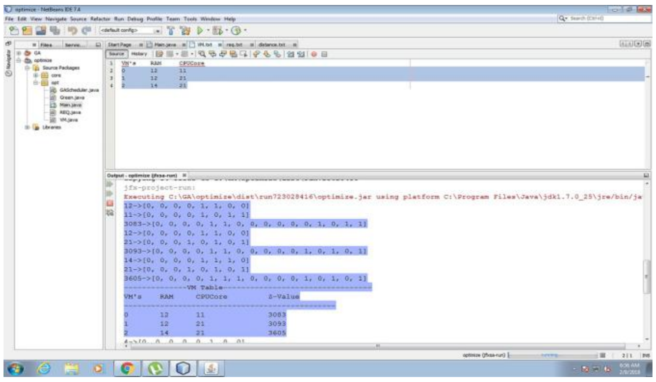


Fig. 5.Hypervisor with guest OS in cloud environment

This system was simulated in a cloud environment using the VMWare Workstation version 10.0.1. Virtual Machines were created to simulate the multiple client-server model over the internet, where the network connectivity between these virtual machines and the host operating system was achieved using the VMWare's virtual network adapters. These virtual network adapters allow the hypervisor to assign unique IP addresses to each of the virtual/guest systems, such that, these machines appear to be connected to each other only via the network, just as it is in the real world cloud network where the machines are connected via the
network called internet.

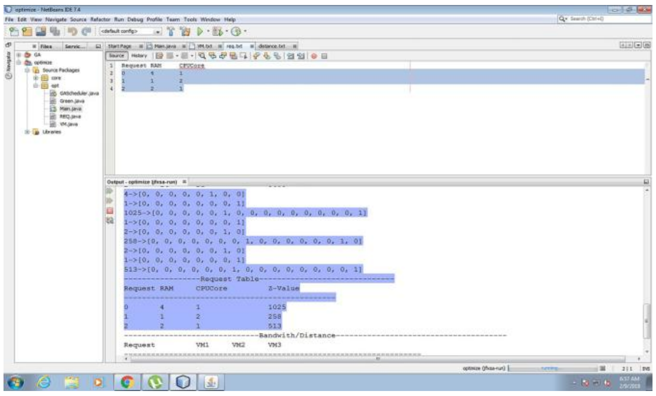
Even though the VMWare workstation is a type-II hypervisor, which shares the hardware resources with its host operating system, the same setup can be replicated by using any number of physical machines with huge resources running a bare metal hypervisor. By using such Type-I hypervisors, it is also possible to create virtual server systems or virtual peers dynamically on demand, making the entire network highly scalable, by taking full advantage of the cloud deployment technique.



```
VM's   RAM   CPUCore
0      12    11
1      12    21
2      14    21
```

```
jfx-project-run:
Executing C:\GA\optimize\dist\run723028416\optimize.jar using platform C:\Program Files\Java\jdk1.7.0_25\jre/bin/ja
12->[0, 0, 0, 0, 1, 1, 0, 0]
11->[0, 0, 0, 0, 1, 0, 1, 1]
3083->[0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1]
12->[0, 0, 0, 0, 1, 1, 0, 0]
21->[0, 0, 0, 1, 0, 1, 0, 1]
3093->[0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1]
14->[0, 0, 0, 0, 1, 1, 1, 0]
21->[0, 0, 0, 1, 0, 1, 0, 1]
3605->[0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1]
--------------VM Table--------------------------------
VM's    RAM     CPUCore         Z-Value
-------------------------------------------------
0       12      11              3083
1       12      21              3093
2       14      21              3605
```

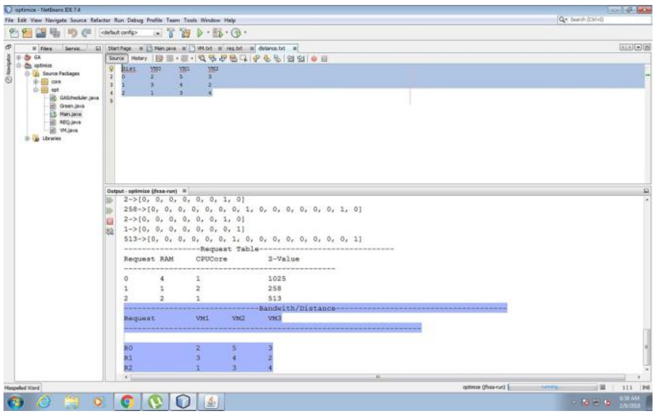
Fig. 6.Configuration of VM, RAM and Cores



```
Request RAM   CPUCore
0       4     1
1       1     2
2       2     1
```

```
4->[0, 0, 0, 0, 0, 1, 0, 0]
1->[0, 0, 0, 0, 0, 0, 0, 1]
1025->[0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
1->[0, 0, 0, 0, 0, 0, 0, 1]
2->[0, 0, 0, 0, 0, 0, 1, 0]
258->[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]
2->[0, 0, 0, 0, 0, 0, 1, 0]
1->[0, 0, 0, 0, 0, 0, 0, 1]
513->[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]
----------------Request Table----------------------------
Request RAM     CPUCore         Z-Value
-------------------------------------------------
0       4       1               1025
1       1       2               258
2       2       1               513
------------------------------Bandwith/Distance-------------------------------------
Request         VM1     VM2     VM3
```

Fig. 7.Request generation



```
Dist   VM0   VM1   VM2
0      2     5     3
1      3     4     2
2      1     3     4
```

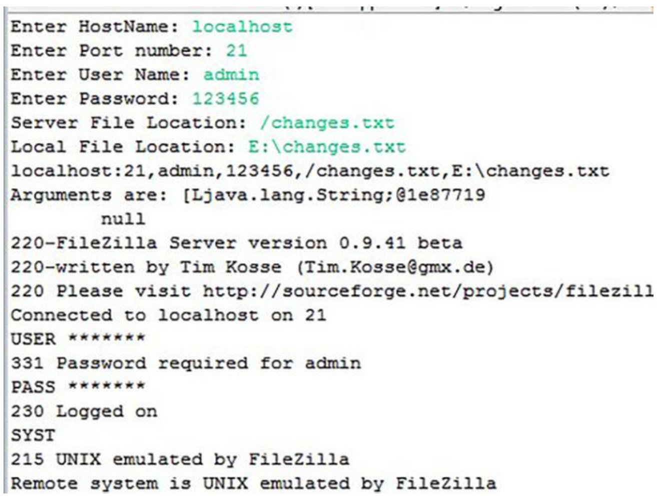
```
2->[0, 0, 0, 0, 0, 0, 1, 0]
258->[0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]
2->[0, 0, 0, 0, 0, 0, 1, 0]
1->[0, 0, 0, 0, 0, 0, 0, 1]
513->[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1]
----------------Request Table----------------------------
Request RAM     CPUCore         Z-Value
-------------------------------------------------
0       4       1               1025
1       1       2               258
2       2       1               513
------------------------------Bandwith/Distance-------------------------------------
Request         VM1     VM2     VM3
------------------------------------------------------------------

R0              2       5       3
R1              3       4       2
R2              1       3       4
```

Fig. 8.Zscore calculation

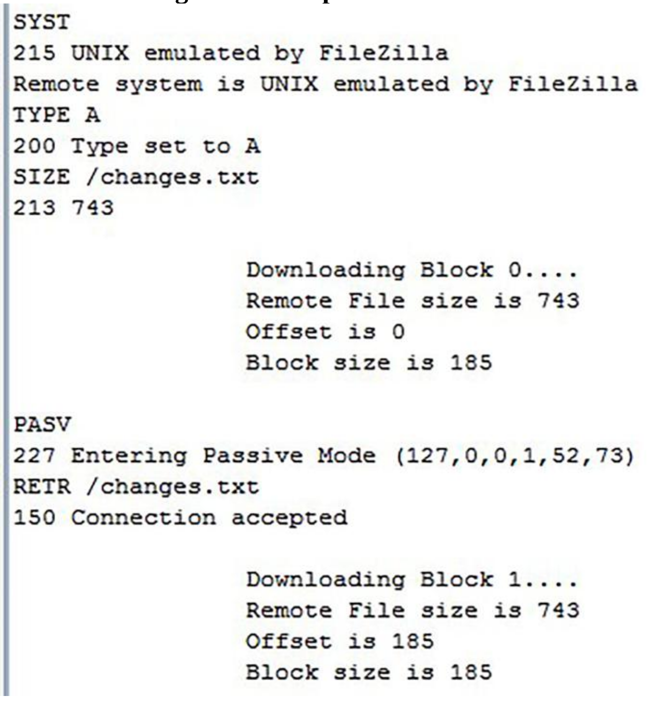
VI. RESULTS

After performing the above steps, the program is run with the inputs that are given by the client in order to download a file from the server which is shown in the screenshots given below.



```
Enter HostName: localhost
Enter Port number: 21
Enter User Name: admin
Enter Password: 123456
Server File Location: /changes.txt
Local File Location: E:\changes.txt
localhost:21,admin,123456,/changes.txt,E:\changes.txt
Arguments are: [Ljava.lang.String;@1e87719
        null
220-FileZilla Server version 0.9.41 beta
220-written by Tim Kosse (Tim.Kosse@gmx.de)
220 Please visit http://sourceforge.net/projects/filezill
Connected to localhost on 21
USER *******
331 Password required for admin
PASS *******
230 Logged on
SYST
215 UNIX emulated by FileZilla
Remote system is UNIX emulated by FileZilla
```

Fig. 9.:ORA Input Parameters



```
SYST
215 UNIX emulated by FileZilla
Remote system is UNIX emulated by FileZilla
TYPE A
200 Type set to A
SIZE /changes.txt
213 743

                    Downloading Block 0....
                    Remote File size is 743
                    Offset is 0
                    Block size is 185


PASV
227 Entering Passive Mode (127,0,0,1,52,73)
RETR /changes.txt
150 Connection accepted

                    Downloading Block 1....
                    Remote File size is 743
                    Offset is 185
                    Block size is 185
```

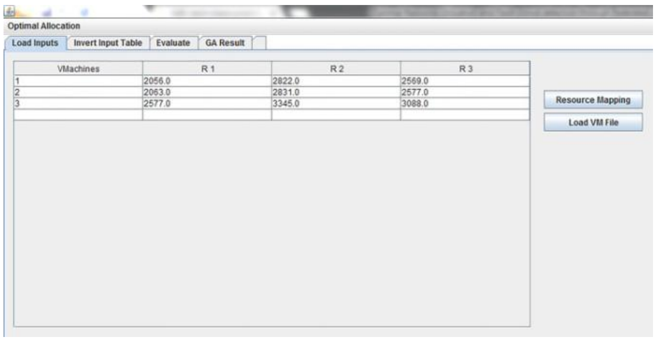
Fig. 10. ORA block download of files

The performance of the system is measured using the following parameters: For a file of size ‘x * u’ with a network transfer speed of ‘y * v’, where ‘u’ and ‘v’ are the unit multipliers to convert x and y to Bytes. Then the Normalized Transfer Time T_N is given by

$$T_N = \frac{x*u}{y*v}\, seconds \qquad (4)$$

Therefore, the overall time (T) taken by the proposed system is, the sum of the Input Request Process Time (T_I) and the Normalized File Transfer Time (T_N). This is for all ‘n’ blocks in the case of ORA and for all ‘n’ pieces in case of P2P mode.

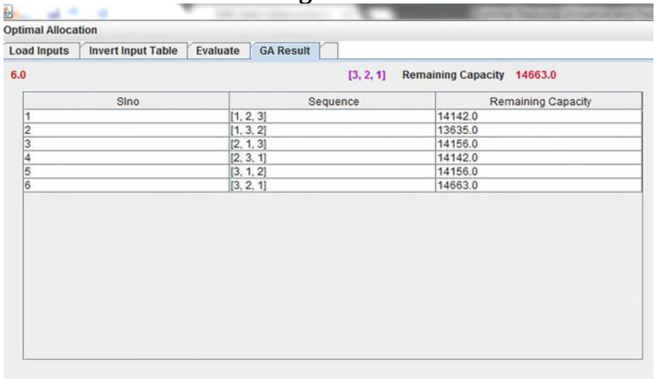
$$T = \sum_1^n (T_I + T_N), \quad n > 0 \qquad (5)$$



VMachines	R 1	R 2	R 3
1	2056.0	2822.0	2569.0
2	2063.0	2831.0	2577.0
3	2577.0	3345.0	3088.0

Fig. 11. ORA for a permutation of 3 request

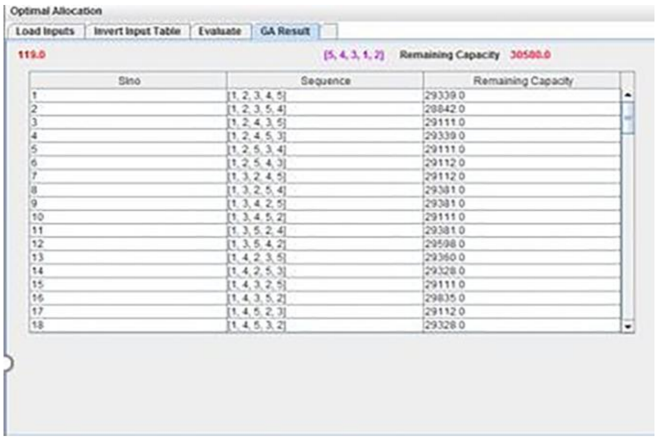
Fig. 12.



6.0 [3, 2, 1] Remaining Capacity 14663.0

Slno	Sequence	Remaining Capacity
1	[1, 2, 3]	14142.0
2	[1, 3, 2]	13635.0
3	[2, 1, 3]	14156.0
4	[2, 3, 1]	14142.0
5	[3, 1, 2]	14156.0
6	[3, 2, 1]	14663.0

Fig. 13. ORA Permutation Result for 3



119.0 [5, 4, 3, 1, 2] Remaining Capacity 30500.0

Slno	Sequence	Remaining Capacity
1	[1, 2, 3, 4, 5]	29339.0
2	[1, 2, 3, 5, 4]	28842.0
3	[1, 2, 4, 3, 5]	29111.0
4	[1, 2, 4, 5, 3]	29339.0
5	[1, 2, 5, 3, 4]	29111.0
6	[1, 2, 5, 4, 3]	29112.0
7	[1, 3, 2, 4, 5]	29112.0
8	[1, 3, 2, 5, 4]	29381.0
9	[1, 3, 4, 2, 5]	29381.0
10	[1, 3, 4, 5, 2]	29111.0
11	[1, 3, 5, 2, 4]	29381.0
12	[1, 3, 5, 4, 2]	29598.0
13	[1, 4, 2, 3, 5]	29360.0
14	[1, 4, 2, 5, 3]	29328.0
15	[1, 4, 3, 2, 5]	29111.0
16	[1, 4, 3, 5, 2]	29835.0
17	[1, 4, 5, 2, 3]	29112.0
18	[1, 4, 5, 3, 2]	29328.0

Fig. 14. ORA for a permutation of 5 request

The chart shown in Figure 9 represents the comparative analysis done between the existing FTP protocol and our proposed ORA system. For the existing system, the FileZilla client was used and random files of different sizes were chosen from different sites. The livecdlist.com site was useful by providing single ISO image files of larger sizes.

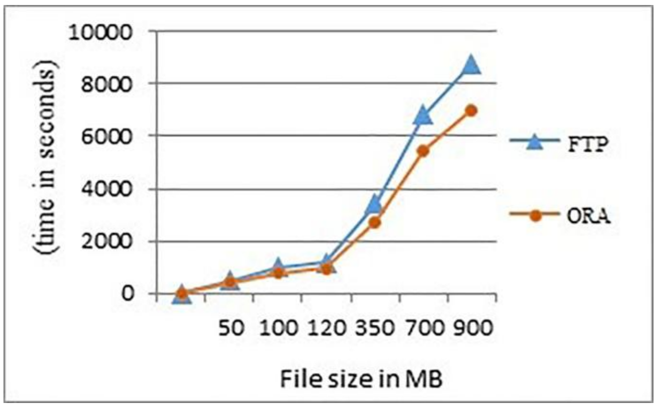


Fig.14. Evaluation of the ORA Mode

Results prove that the ORA download scheme has improved download time when compared to the normal FTP download scheme and the increase in the download times is more profound with increase in the size of the files being downloaded. Thereby the proposed system provides a certain level of scalability as well.

VII. CONCLUSION

We have simulated a model to address the downloading schemes of a cloud through a bifurcated approach – client server mode and peer to peer mode. The analysis of the model shows that each scheme can be advantageous, for certain operating conditions. We have also discussed a novel method to allow simultaneous downloads from multiple sources without harnessing the performance. Today's internet traffic makes us to replicate content to many mirror servers. In that case, Resource optimization plays a major role for both the cloud provider (CP) and the end user. CP's tries to optimize its resource utilization to service their other customers. End users do not want latency and so they expect immediate service from any nearby mirror server. Our proposed ORA ensures that all the servers contribute equally in achieving a task concurrently and see to that no single server is overloaded nor left to run in underload condition

This work can be further extended by testing the performance of ORA through two, four and eight servers and benchmark them with regular multiple client and multiple servers' conditions. Similarly, when the server load is too much for the server to serve any further client requests, it switches to the P2P mode, where the server becomes a seeding peer or provides tracker information to the client so that the client can connect with other client peers who are sharing the requested files among themselves. The approach is promising and can be refined even more by experimenting with different configurations and runtime scenarios.

REFERENCES

1. Jiajun Wang, Chuohao Yeo, Vinod Prabhakaran, Kannan Ramchandran, "On the Role of Helpers in Peer-To-Peer File Download Systems: Design, Analysis and Simulation", In 4th Int'l Workshop on Peer to-Peer Systems (IPTPS), Feb 2005.
2. Bhardwaj, D. and R. Kumar, "A Parallel File Transfer Protocol for Clusters and Grid Systems," in Proc. of 1st International Conference on e-Science and Grid Computing, 2005.

3. Chengchen Hu, Danfeng Shan, Yu Cheng and Tao Qin, “Inter-Swarm Content Distribution Among Private BitTorrent Networks”, IEEE Journal On Selected Areas In Communications, Vol. 31, No. 9, September 2013.
4. Raymond Lei Xia and Jogesh K. Muppala, “A Survey of BitTorrent Performance”, IEEE Communications Surveys & Tutorials, Vol. 12, No. 2, 2010.
5. Cohen, B., “Incentives build robustness in BitTorrent,” in Proc. of Workshop on Economics of Peer-to-Peer Systems, 2003.
6. J. Cheng, Y. Li, L. Jiao, and J. Ma, “A new mobile peer-to-peer architecture,” in Proc. 5th WSEAS Int. Conf. Appl. Comput. Sci., 2006, pp.1083–1088.
7. Yan Huang, Zhenhua Li, Gang Liu and Yafei Dai, “Cloud Download: Using Cloud Utilities to Achieve High-quality Content Distribution for Unpopular Videos”, IEEE/ACM Communications on cloud, Vol. 12, No. 2, 2011.
8. H.-C. Hsiao and C.-T. King, “Bristle: A mobile structured peer-to-peer architecture,” in Proc. Int. Parallel Distrib. Process. Symp., 2003.
9. J. Cheng, Y. Li, L. Jiao, and J. Ma, “A new mobile peer-to-peer architecture,” in Proc. 5th WSEAS Int. Conf. Appl. Comput. Sci., 2006, pp.1083–1088.
10. Yipeng Zhou , Tom Z. J. Fu, Dah Ming Chiu and Yan Huang , “An Adaptive Cloud Downloading Service”, IEEE Transactions on Multimedia, Vol. 15, No. 4, June 2013.
11. Liying Wu ,Bo Liu and Weiwei Lin , “A Dynamic Data Fault-tolerance Mechanism for Cloud Storage” , Fourth International Conference on Emerging Intelligent Data and Web Technologies,2013.
12. Ravi Jhawar and Vincenzo Piuri, “Fault Tolerance Management in IaaS Clouds”, IEEE Journal On Selected Areas In Cloud Computing, Vol. 13, No. 19, June 2012.
13. Mousumi Paul, Goutam Sanyal, “Survey and Analysis of Optimal Scheduling Strategies in Cloud Environment”, IEEE, 2011.
14. G. N. Rao and S. Nagaraj, “Client Level Framework for Parallel Downloading of Large File Systems,” in International Journal of Computer Applications, Vol. 3, No. 2, pp. 0975–8887, June 2010.
15. Hsu, C-H., C-W. Chu and C-H. Chou, “Bandwidth Sensitive Co-allocation Scheme for Parallel Downloading in Data Grid,” in Proc. of IEEE ISPA, pp.34-39, 2009.
16. Bin Fan, John C. S. Lui, and Dah-Ming Chiu, “The Design Trade-Offs of BitTorrent-Like File Sharing Protocols”, IEEE/ACM Transactions on Networking, Vol. 17, No. 2, April 2009.
17. R.W. Butler, “A Primer on Architectural Level Fault Tolerance”, Technical Memorandum, NASA/TM-2008-215108, February 2008.
18. Kawano, A., J. Funasaka and K. Ishida, “Parallel Downloading Using Variable Length Blocks for Proxy Servers,” in Proc. of 27th ICDCS Workshops, pp.59-64, 2007.
19. Deng K, Ren K, Zhu M, Song J, “A data and task co-scheduling algorithm for scientific cloud workflows,” IEEE Transactions on Cloud Computing 2015.
20. Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K,“Characterizing and profiling scientific workflows,” Future Generation Computer Systems ,Vol 29, no3, pp.682–692, 2013.
21. Smanchat S, Viriyapant K, “Taxonomies of workflow scheduling problem and techniques in the cloud,” Future Generation Computer Systems, Vol 52, pp.1–12, 2015.
22. Deelman E, Vahi K, Juve G, Rynge M, Callaghan S, Maechling PJ, et al., “Pegasus, a workflow management system for science automation,” Future Generation Computer Systems Vol 46: pp.17–35, 2015.
23. Om Kumar C.U., Ponsy R.K. Sathia Bhama, “Fuzzy based energy efficient workload management system for flash crowd”, Computer Communication, Elsevier Vol 147, pp.225-234, 2019.

AUTHORS PROFILE

T.P. Khiruparaj was born in Madurai, Tamil Nadu, India in the year 1999. He is an engineering student in the department of Computer Technology at Madras Institute of Technology – Anna University, Chennai, Tamil Nadu, India. His research interests include computer networks and IoT security, forensics, IoT networks.

V. Abishek Madhu was born in Thiruvananthapuram, Kerala, India in the year 1999. He is an engineering student in the department of Computer Technology at Madras Institute of Technology – Anna University, Chennai, Tamil Nadu, India. His research interests include operating systems, computer networks and machine learning.

Ponsy RK Sathia Bhama received her B.Tech degree from Bangalore University, M.Tech degree in Computer Science from Madras University and Ph.D. from Anna University. She is currently working as an Associate Professor in the Department of Computer Technology at Madras Institute of Technology- Anna University. Her research interests include Grid computing, Cloud Computing, Soft Computing and Operating System.