Real Time Network Server Monitoring using Smartphone with Dynamic Load Balancing

Dhuha Basheer Abdullah [1], Zeena Abdulgafar Thanoon [2],

[1] Computer Science Department, Mosul University, Computer Sciences and Mathematics College Mosul, Iraq

[2] Computer Science Department, Mosul University, Computer Sciences and Mathematics College Mosul, Iraq

Abstract

The services provided by network servers are very important, therefore, the monitoring of these servers in real time is required to discover the obstacles in order to improve servers performance. In this paper, a file server system was designed to be monitored by a smartphone in real time through the use of dynamic scheduling algorithm for three main metrics: CPU usage, Memory usage and Free hard disk space. Dynamic load balancing (DLB) was implemented between servers depending on CPU usage of servers. Also, a dynamic checkpoint (DCP) technique was introduced to minimize files download time. The experiments show that the use of DLB reduced the CPU Usage for both servers and the download time for all clients. Also, the use of the proposed DCP technique is more efficient than the traditional static checkpoint technique.

Keywords: *Real time, dynamic load balancing, checkpoint, smartphone.*

1. Introduction

Applications of Real Time Network Monitoring Systems RTNMS provide a clear view for the metrics that are being monitored in the network, also give a flexible dealing with network problems in a proactive way.

Real time systems can be defined as those systems that consider the time as an important factor. Those systems are categorized depending on time constraints into Hard real time and Soft real time. RTNMS can be considered as a soft real time.

Network administrators are always the first defense line in diagnosing network faults, therefore it is very important to facilitate and support them by monitoring tools and software like smartphones for remote monitoring. One of the important objects to be monitored is the Network Servers [1]. With the rapid growth of both information and users, the need of effectively improve the quality of services provided by servers becomes an urgent problem to be addressed. This situation caused by heavy demand on the server node which led to slowness in responding to requests. For these reasons load balancing algorithms should be implemented for better performance and to increase the ability to handle more users. So server farm is of wide interest to enterprises, which enables a group of independent servers to be managed as a single system for higher availability, easier manageability and greater scalability [2][3][4]. The main objective of load balancing is to minimize the response time and maximize throughput of the network

2. Contributions

In this work, a real time monitoring system was designed to monitor the performance of a network file server in a server farm environment supported by DLB between the servers. This system offers the following contributions:

- Implementing real time constraints to monitor the most important metric associated with the servers like CPU usage, Memory usage and Free hard disk space.
- Developing a new method associated with the checkpoint technique that gives the clients the ability to resume file download from the stopped point without the need to restart the download operation when changing from the crowded server to less loaded server.
- Developing a DLB algorithm to obtain better performance.

3. Related works

Previous works in this field:

- L. Yucheng, L. Yubin [5] designed a system using B/S mode which enables administrators to view the server-side situation in the performance testing and network maintenance.
- G. Kanagaraj, N. Shanmugasundaram, S. Prakash [6] suggested a method to dynamically load balance using

service queue depending on the summation of CPU utilization, memory utilization and network utilization of web servers.
- N. Bessho, T. Dohi [7] developed a stochastic model to evaluate the expected total recovery overhead for cluster computing system with three well-known checkpoint and rollback recovery schemes, one of which is for central file server checkpointing.

4. Real time system

A real-time system is a computing system which has timing constraints and the accuracy of such a system depends not only on its logical results but also on the time at which the results are available. Real-time systems can be classified as hard real time systems in which the consequences of missing a deadline can be catastrophic and soft real time systems in which the consequences are relatively tolerable.
There are two types of Real time scheduling algorithms: fixed priority and dynamic priority. In fixed priority algorithm the priorities are assigned to each task before the activation of all tasks, an example of fixed priority algorithm is the Rate-Monotonic (RM) which assigns priorities to tasks on the basis of their period times. In dynamic priority algorithm the priorities are computed during the execution of the system, an example of dynamic priority algorithm is the Earliest-Deadline-First (EDF) which assign priorities to individual jobs on the basis of their absolute deadline; the shorter the deadline, the higher the priority. [8].

4.1. Real time network monitoring

Real time network monitoring applications are able to aggregate, quantify, and analyze network traffic for a clearer view. Thus, without network monitoring systems, it would be difficult to identify and resolve network problems. Real time network monitoring includes the monitoring of network performance in real time. It is the network administrator's job to ensure that their network and network applications are performing properly. However, advanced tools are needed to help network administrators to intensively and continuously monitor their network performance. Real time network monitoring should be accurate in detecting network disruptions and the cause of these disruptions [1].

5. Load balancing

The main objective behind load balancing is to distribute a set of requests among the different server nodes, to prevent some server nodes from being heavily loaded while others are lightly loaded [2][3]. Load balancing technology can balance conflicting factors such as cost, performance, and scalability, through a relatively low total cost of the server farm to achieve a strong performance that cannot be achieved by stand-alone system. Load balancing algorithms generally can be classified as either static or dynamic [4]:

A- Static load balancing algorithm:
In this algorithm, load balancing decisions, are made at compile time and takes less time, which doesn't refer to the state of the servers, so it does not need to constantly monitor the nodes for performance statistics [6]. Such a hypothesis may not apply to a distributed environment. Because the static approach cannot respond to a dynamic runtime environment, it may lead to load imbalance on some nodes. However, static algorithms only work well when there is not much variation in the load on the workstations; In addition it is not satisfactory for parallel programs that are of the dynamic and/or unpredictable kind.

B- Dynamic load balancing algorithm (DLB):
DLB algorithm, make changes to the distribution of work among nodes at run-time; they use current or recent load information when making distribution decisions. Despite the higher runtime complexity, dynamic algorithms can potentially provide better performance than static algorithms [9]. DLB system mainly includes two processes: monitoring the load state of servers and assigning requests to the servers. In the dynamic approach, the load balancing decisions are based on the current state of the system; tasks are allowed to move dynamically from an overloaded node to an under-loaded node to receive faster service. This ability to react to change in the system is done through the use of checkpoint technique.

5.1. Checkpointing

One commonly used technique to recover from application failure is the use of checkpoints. During static checkpointing, an application writes its entire state to non-volatile secondary storage so that when the application is interrupted, it can resume its work from the last checkpoint rather than from the beginning. While writing and reading the checkpoint data is a type of overhead that consumes valuable system resources, the savings in rework times due to failure can often outweigh the cost of performing checkpointing in the first place.
One aspect of employing static checkpointing is properly assigning a checkpoint interval, i.e., the time from the beginning of one checkpoint to the beginning of the next. Given a set of jobs and failure parameters, it is possible to assign this interval in such a way that it maximizes application efficiency, i.e., the ratio of time the job spends making forward progress compared to the entire wall-clock time that includes checkpoint, restart, and rework

overload[10]. Here the proposed technique depends on DCP without the need to store fixed points for the downloaded files. In this technique, when an overload occurs on the main server, DCP will start by saving the current point (byte number of the file that is being downloaded) to complete downloading operation on the other server instead of restarting from the starting point.

6. System model

The proposed system was based on a model for a network file server used by the clients for downloading files. The infrastructure of the system has the hardware components shown in figure (1).

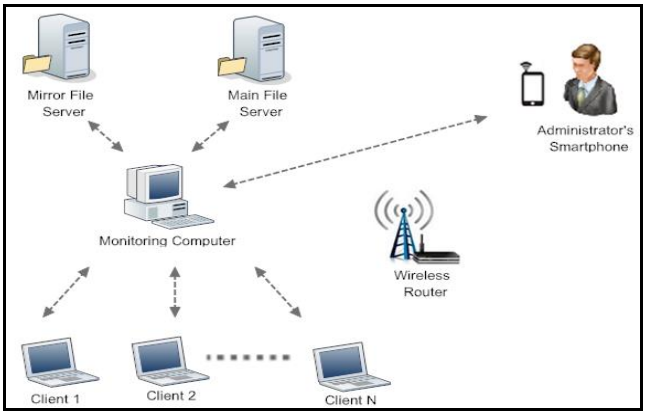

Fig. 1 System model

The monitoring computer was responsible for monitoring the performance of the two servers like CPU usage, Memory Usage and Free hard disk space and it saves the values of these metrics in Google drive. Clients communicate with the monitoring computer to download files; the monitoring computer will send the requests to the server that has less load on its CPU usage using DLB algorithm. The monitoring process will be done depending on the metric of real time constraints. The EDF algorithm was used to schedule the sending of performance metrics. When the administrator wants to use the smartphone for monitoring the metrics of two file servers, the request will be send from smartphone to the monitoring computer, and then the metrics will be displayed periodically in smartphone. If an urgent case occurs in the main server - like CPU usage exceeds the threshold value – the administrator will send an order to the monitoring computer to start re-load balancing between the two servers by implementing the new DCP technique.
The connection between the network parts was done using WiFi IEEE 802.11protocol. Figure (2) illustrate the general behavior of the system.

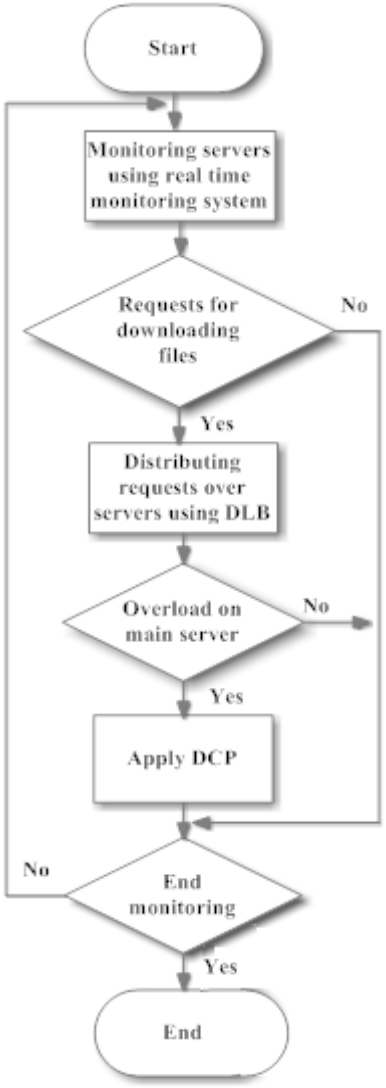

Fig. 2 System general behavior

6.1 Software description

Three algorithms were used in this system; all of them were implemented in the monitoring computer. These algorithms are:

A- Servers real time monitoring algorithm:
Connections will be open between the monitoring computer and the two servers. Both servers will apply the EDF algorithm to periodically send the performance metrics to the monitoring computer depending on the priority configured by the administrator. The monitoring computer saves the values of those metrics in a text file and uploads them to the cloud drive to archive them for future works. When the administrator wants to use the smartphone for monitoring the metrics of two file servers, a request is sent to the monitoring computer, which will

send these metrics periodically in a real time base. In case of one or all metrics exceed the threshold values; a notification alarm will notify the administrator via the smartphone. Figure (3) shows the flowchart of this algorithm

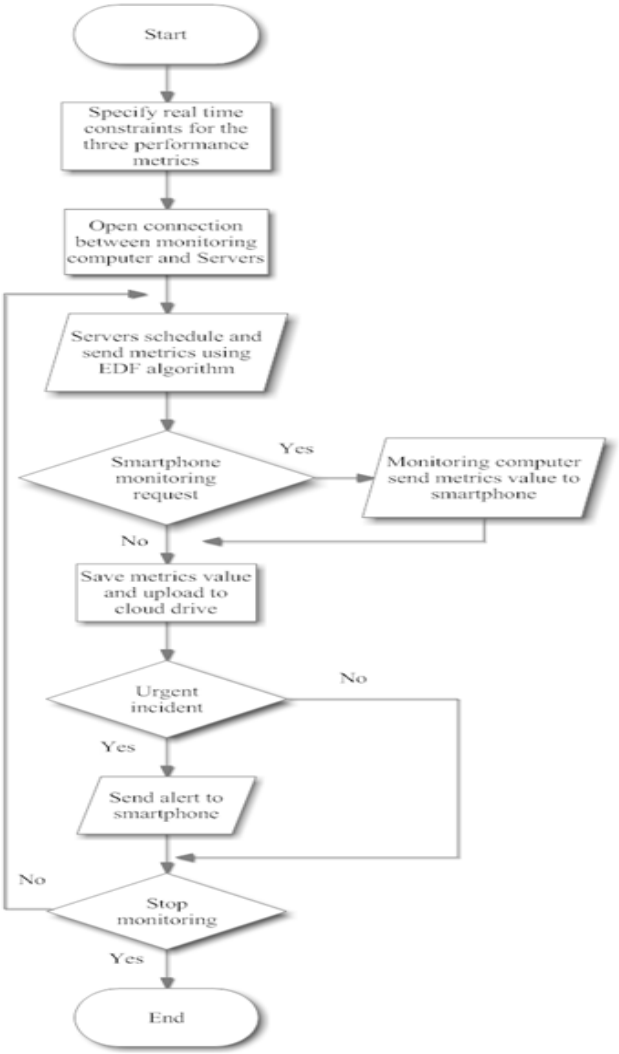

Fig. 3 Monitoring algorithm

B- Servers load balancing algorithm:

When the client wants to download a file, a request is sent to the monitoring computer. The monitoring computer will send a list of all existed files in the main file sever to the client; the client will choose the file from the list. Then the monitoring computer will specify the server with the lowest value of the CPU usage. After that the IP address of the client and the name of the selected file will be send to the specified server to connect with the client and start sending the file. Figure (4) shows the flowchart of this algorithm.

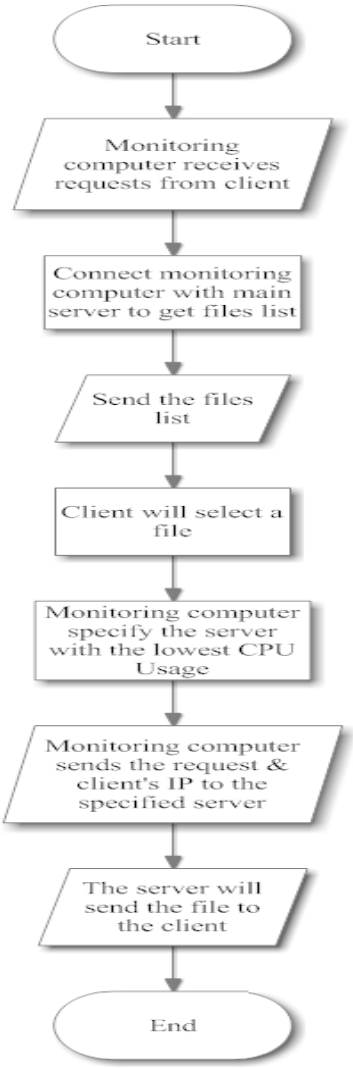

Fig. 4 Load balancing algorithm

C- Servers re-load balancing with DCP algorithm:

An alert will be send to the administrators' smartphone when the CPU usage of the main server exceeds the threshold value. Therefore, the administrator will send a command to the monitoring computer to start the implementation of re-load balancing algorithm. The monitoring computer will send a request to the main server to obtain information concerning the files currently under downloading. The monitoring computer will send an order to all clients to stop download and disconnect the connection with the main file server. The monitoring computer requests these clients to send the name of the file plus the number of the last received byte (that will be a checkpoint for each file) to the mirror server. The mirror server will connect the clients and complete the downloading process by starting from the checkpoints. Figure (5) shows the flowchart of this algorithm.

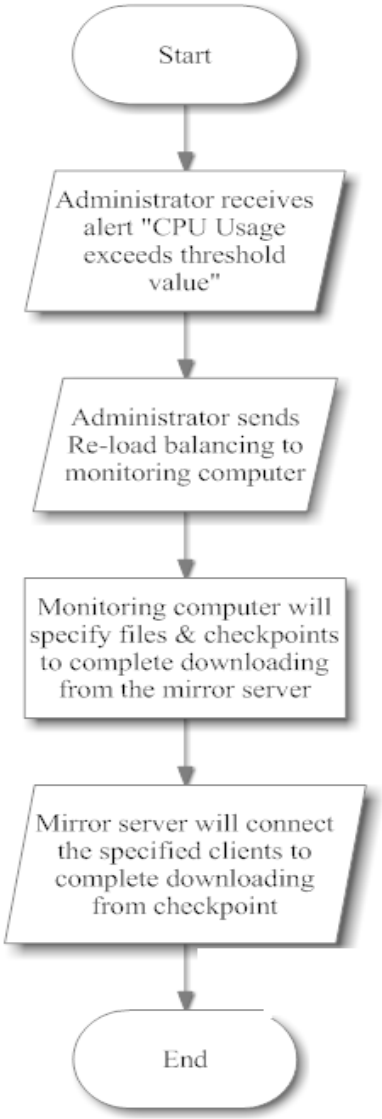

Fig. 5 Re-load balancing with DCP algorithm

7. Results and discussion

In this section, the experimental results for both of DLB and the proposed DCP technique will be explained. Assuming that; there are two servers, one computer acting as load balancer plus monitor, and three clients. Table (1) shows the hardware specification of both severs.

Table 1: Servers' hardware specifications

H/W	*Main server*	*Second sever*
CPU	Core i5 2.5 GHz	Core i3 2.4 GHZ
RAM	4 GB	2 GB

A- DLB (Less CPU usage)

The experimental result presented in table (2) depends on two metrics, CPU Usage and Download Time.

Table 2: Experimental results

File size (MB)	*Single server (Without DLB)*		*Two servers (With DLB)*		
	CPU Usage (%)	*Average download time (Second)*	*Serv.1 CPU Usage (%)*	*Serv. 2 CPU Usage (%)*	*Average download time (Second)*
6	1.89	5.2	1.6	1.59	2
28	4.5	21	3.4	3.51	8.2
65	7.6	31	5.9	5.8	13

An average percentage was calculated for servers' CPU Usage, and also an average download time for clients' requests. The measurements were performed on two cases. The first does not use DLB; all the clients' requests are forwarded to the main server only. The second one uses DLB by distributing clients' requests on both servers. The experiment was performed by assuming that all the clients requested the same file at the same time. The experiment was repeated many times with different file size.

The results show that the use of DLB reduced the CPU Usage for both servers and the download time for all clients. Figure (6) shows the results of the average download time-in the case of the use of DLB and without using it.

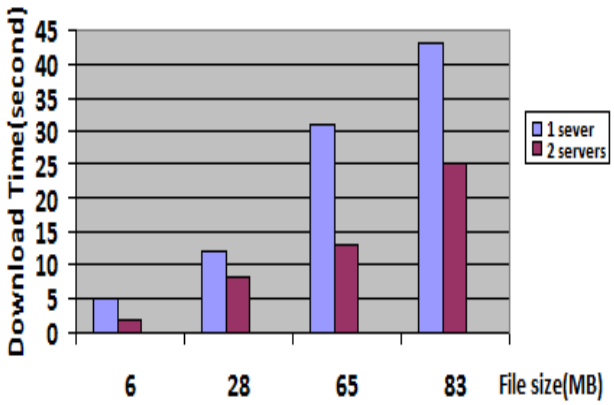

Fig. 6 Download time with/without DLB

B- DCP

The results proved that the use of the proposed DCP technique is more efficient than the traditional static checkpoint technique. In DCP technique, the time needed for switching between servers ranges between (0.5~1) second depending on the speed of the network.

Total download time using DCP technique was calculated depending on the following equation:

$$Total\ Time = T1 + ST + T2$$

Where:
T1: download time before switching
ST: switching time
T2: download time of the remaining size.

While in static checkpointing, assuming that there are three checkpoints, therefore the total download time was calculated depending on the following equation.

$$Total\ Time = T1 + ST + TC + T2$$

Where:
T1: download time before switching
ST: switching time
TC: needed time to go back to the last checkpoint before switching
T2: download time of the remaining size.

Figure (7) shows the results for downloading a specified file with 28MB using static checkpointing technique and the proposed DCP technique. Assuming that the switching time is one second and the interrupt or switching operation occurred in five different locations.

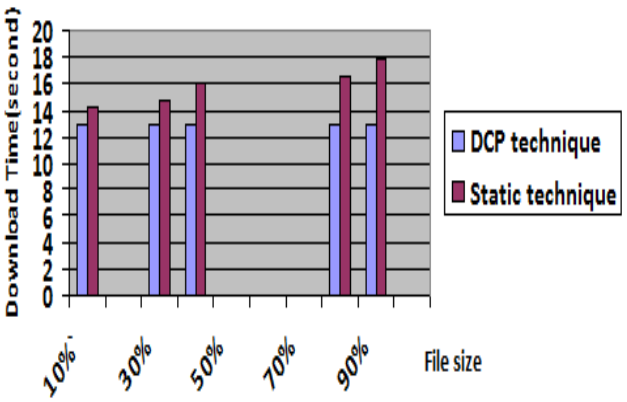

Fig 7. Comparing between DCP & static technique

8. Conclusions

The use of a real time scheduling algorithm in the proposed system maximizes the quality of servers' monitoring operation by giving the administrator the ability to specify the priorities for the monitored metrics. This priority was specified depending on real time constraints. The use of smartphone by the administrator gives flexibility in the job. The DLB proved high efficiency in load balancing between servers because it takes into consideration the current state of the servers. Also, the use of the new DCP technique achieved a noticeable reduction in download time of files compared with traditional static checkpoint technique.

References

[1] A. Manasrah, "Real Time Distributed Network Monitoring Platform (RTDNM)", Ph. D thesis, School of computer science, University Sains Malysia, Malysia 2009.
[2] A.Meiappane, and V. Prasanna Venkatesan, et al, "The Pattern as a Reusable Component to Adaptive Framework for Load Balancing Mobile Agent in Internet Banking", IJCSI, Vol. 9, Issue 2, No. 2, 2012, pp. 332-338.
[3] Y. Jiao, and W. Wang, "Design and implementation of load balancing of distributed systems based web server", IEEE International Conference on High Performance Computing and Communications, 2010, pp. 337 – 342.
[4] K. Kungumaraj, and T. Ravichandran, "An Efficient Load Balancing Algorithm for a Distributed Computer System", IJCTA, Vol. 2, No. 6, 2011, pp: 4012-4020.
[5] L. Yucheng L., and L. Yubin, "A Monitoring System Design Program Based on B/S Mode", IEEE, Vol. 1, 2010, pp. 184-187.
[6] G. Kanagaraj, N. Shanmugasundaram, and S. Prakash, "Adaptive Load Balancing Algorithm Using Service Queue", 2nd International Conference on Computer Science and Information Technology (ICCSIT) Singapore, 2012, pp. 143-146.
[7] N. Bessho, and T. Dohi, "Comparing Checkpoint and Rollback Recovery Schemes in a Cluster System", Springer-Verlag Berlin Heidelberg, 2012, pp. 531-545.
[8] W. S. Jane Liu, "Real-Time Systems", New Jercy, Prentice Hall, 2000.
[9] S. Kumar, and N. Singhal, "A Study on the Assessment of Load Balancing Algorithms in Grid Based Network", International Journal of Soft Computing and Engineering (IJSCE), Vol. 2, Issue: 1, 2012, pp. 402-405.
[10] W. M. Jones, J. T. Daly., and N. DeBardeleben, "Impact of Sub-optimal Checkpoint Intervals on Application. Efficiency in Computational Clusters", ACM, USA, 2010, pp. 276-279.

[1] **Dhuha Basheer Abdullah** is the head of Computer Science Department, Computer Sciences and Mathematics College, Mosul University. She received her PhD degree in computer sciences in 2004 in the specialty of computer architecture and operating system. She supervised many Master degree students in operating system, computer architecture, dataflow machines, mobile computing, real time, and distributed databases. She has three PhD students in FPGA field, distributed real time systems, and Linux clustering. She also leads and teaches modules at both BSc, MSc, and PhD levels in computer science. Also, she teaches many subjects for PhD and MSc students.

[2] **Zeena Abdulgafar Thanoon** is an MSc. student in Computer Science Department, Computer Sciences and Mathematics College, Mosul University. She is interested in Computer Networks, Distributed Systems, and Operating System subjects.