Literature Reviews on Secure scheduling algorithm for web servers and load balancing factors

Syed Mohammed S. Abidi

Student, M.C.A Department, Mumbai University, India

ABSTRACT

This paper is a literature review on secure scheduling algorithm which describes how servers will handle distributed jobs or client request in server cluster farm and cloud computing systems. The proposed algorithm's comparison helps the researcher's for evaluating new secure scheduling algorithm to overcomes the starvation and request failure as, the existing task scheduling algorithms do not suffice this purpose due to their focus on being generic and minimizing the execution time, while failing to use the characteristics of the system architecture and job profiles with the associated user-agent utility which is caused by FCFS (first come first serve), SRPT (Shortest Remaining Processing Time), Process Share, server pool, share resources,. Paper also describe different scheduling technique such as Static scheduling, Dynamic scheduling, Multilevel queue scheduling, Round robin scheduling, Ideal server monitoring algorithm, dispatcher scheduling algorithm, cache replacement algorithm, load balancing algorithm (dynamic routing) here we also describe server security measures such as Web server logs reviewer and intrusion detection technique.

Keyword:- *server farm; load balancing; scheduling algorithm; dispatcher; round around time, Genetic Algorithms; Time Constrained Scheduling; Quality of Service, load-balancing Web Server Logging; Intrusion Detection; Small Medium Enterprises; Information Security.*

1. INTRODUCTION

As technology is evolving day by day according to Moore's law prediction, uses of handheld devices operating system such as android, IOS, Ubuntu and windows devices are multiplying day by day with slight modification of its architecture, starting from 512 MB of RAM to 6 GB RAM of device at present day, also browser portability and its software improvement which generate multiple requests from multiple browser tabs per user worldwide to the server.

So, client request which is generated from multiple devices are continuously striking the server which causes delay and loss of request from the server side, example if we are using single server to deploy our web project then there is a situation when more than one million peoples trying to access that site at particular interval of time which causes congestion and network loss, to overcome this problem instead of implementing web project using single high quality server now days industry using multiple server racks which memory is around 100 to 500 petabytes HDD (Hard disk drive) placed on that rack as per configuration they also maintained sufficient amount of physical and logical memory to serve the cloud request.
For implementing client requests distribution, servers use scheduling algorithm which provides uninterrupted services, congestion control, and load balancing technique.

Long round around time is very critical problems which is handled by limited amount of servers as resources occupied

by these servers goes in to lock state which is not easily unlock until request is served thread processing and time sharing will be better but it also fails when we update something on a server which increases the server load factor, let us take an example of converting PDF to images using imageMagick (Linux free tool) on a particular UNIX server and that server is online, so when admin tries to convert it and at that time if request is generated to viewing that file then that request is failed even website load time is also increases as cache memory of server is full during conversion process of pdf in to image and at the same time to full-fill the client request.. Most evidence also showing that high variability of task size distribution in computer loads. For example, files requested in Web servers and in UNIX fit a fat-tailed distribution [5] . One way to figure out this problem is using server farms. A server farm contains cluster of many computers also called host, server or node, and front-end high-speed dispatcher. Each incoming request is synchronously dispatched via the dispatcher to one of the computers. The main advantages of using server farms are price because they are cost effective and also many slow computers are cheaper than fast computers and it is easy to up or down your server capacity. One of the most important issues in server farms is to effectively managing and maintaining its routing policy. This determining how to assign jobs to hosts. On the Internet, companies whose traffic is usually uses high uses load balancing. To implement load balancing on Web traffic, there are several techniques. For Web serving, one technique is to route each incoming request in turn to multiple servers. In most of the technique, the servers are distributed over different geographic areas.

Servers handle incoming jobs with the help of computer designed programme called "Dispatcher". Dispatcher objective distributes incoming job on the servers in an efficient way. Dispatcher integrates scheduling algorithm which is best for dispatching jobs to other server's. Therefore, scheduling algorithms are very important to achieve better performance in the dispatcher. This paper describes the comparisons of different algorithms which are popularize these days, it also revised the result generated by algorithm under different load tackling situation, which helps the researcher to set benchmarks for each algorithm under different constraints.

2. Web SERVER FARMS

A web server farm consists of multiple servers or cluster of computer stations running simultaneously and synchronously to provide services over networks at all times and without interrupting. It is maintained by specialised organisation which ensures that server is up (online) every time. It consumes huge amount of energy and cooling systems.
Server installation required huge hectare of lands due to which it is known as server farm.

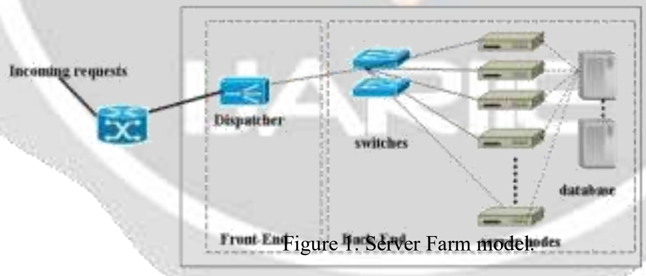


Figure 1: Server Farm model.

3. SCHEDULING ALGORITHMS

Resources which is put in server farms is remains idle in most of the times while some of the server cluster in a farm is handling most of the client request and remains busy which leads to resource imbalance in server farm. Any load imbalance will result in poorer task response times. Scheduling algorithms is used to determine which requests is to be served at any point of time, how much time is spent on each, and what happened when a new request arrives. The goals of scheduling policies are to minimize the mean round around time of the request and to behave fairly to all requests [1].

Static algorithms are the fastest solutions but results are not reliable as they lead to poor decision, such as routing a request to a server node having a long queue of waiting load while there are other almost idle nodes [1]. Dynamic algorithms have the potential to outperform static algorithms by using some state information to help to dispatch decisions. On the other hand, they require mechanisms that collect, transmit and analyze state information thereby incurring in overheads [6]. Round-Robin (RR), random (RAN) and Weighted Round-Robin (WRR) are dynamic algorithms.

3.1 FCFS (First-Come-First-Serve)

FCFS is a policy which is adopted by number of application domains such as scheduling [8] and Operating Systems.
First come first serve algorithm in one of the basic batch system processing algorithm, Client which request server resources first, will allocate first, proper queue is maintained by server processor and scheduling is done on that queue.
Client who requesting resources will be added to the tail of the queue, and request which is to be served is dispatched from the head of the queue.
Problem with algorithm - Non preemptive, low input/output device utilization and cause starvation.
Refer fig.1

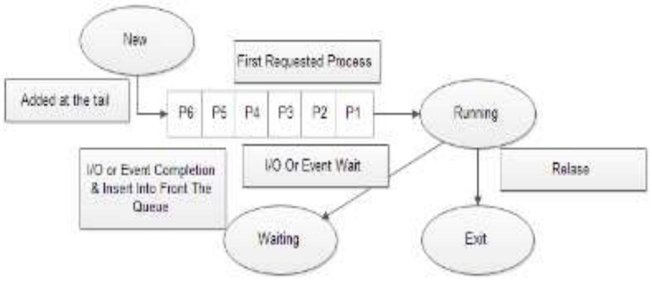
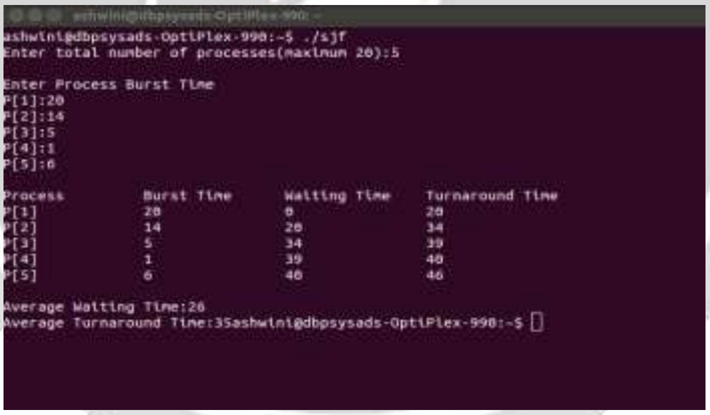


Fig. 1: First Come First Serve Scheduling

PARAMETER REQUIRED
Process sequence and Burst time



```
ashwini@dbpsysads-OptiPlex-990:~$ ./sjf
Enter total number of processes(maximum 20):5

Enter Process Burst Time
P[1]:20
P[2]:14
P[3]:5
P[4]:1
P[5]:6

Process         Burst Time      Waiting Time    Turnaround Time
P[1]            20              0               20
P[2]            14              20              34
P[3]            5               34              39
P[4]            1               39              40
P[5]            6               40              46

Average Waiting Time:26
Average Turnaround Time:35ashwini@dbpsysads-OptiPlex-990:~$ 
```

Figure 2. Example based on FCFS

3.2 SRPT (Shortest Remaining Processing Time)

New process is added in to control unit while executing current process, the judgement of allocating CPU resources is depending up on the Shortest Remaining Process Time of that process.
Another policy that provably optimal mean round around time for all requests is SRPT "Shortest Remaining Processing Time". Shortest-Remaining-Processing-Time(SRPT)
scheduling policy is an optimal algorithm for managing client request by minimizing process mean response time [9] and [10]. While handling scheduling policies on web servers SRTP is well aware of the job size, which is well known to the server, refer to processing time (response time) of that job for implementing SRPT the processor response to least processing time jobs first then as per time serving other processing request.
There are two problems, this policy not fair. Jobs with a large size may be waiting for a while and Dispatcher must know the size of jobs beforehand.

3.3 P.S. (Process Sharing)

In process sharing the capacity “C” or actual available resource of servers is equally shared between the incoming requests. This policy assures max-min fair allocation and easy to implement.

3.4 Server Pool

Server pool, is a cluster of real servers that provide services. All the results of the request are computed and given feedback by the computers within the pool server.

3.5 Share Resources

Cloud Computing is state which gives proper and on-demand network access to shared pool of computing resources like network, storage, servers and services that are to be rapidly released with the efficient way in minimum management.

3.6 Static Scheduling

In static scheduling task which is going for assigning to the particular cluster of servers is already known to the server processor. Processor logical unit known well in advance about the task they are going to handled and when they have to redirect the process if resource is out i.e. memory is insufficient to carry out the request and to avoid dead lock within server resources.

With this goal in mind, static scheduling methods [Lo 1988, Sarkar 1986, Shirazi 1990, Stone 1977] attempt to:

• predict the program execution behaviour at compile time (that is, estimate the process or task, execution times, and communication delays).
• Differentiate the single task in to smaller one to overcome the communication cost.
• allocate processes to processors.

Advantage:
Scheduling process is incurred at compile time, resulting in a more efficient execution time environment compared to dynamic scheduling method.

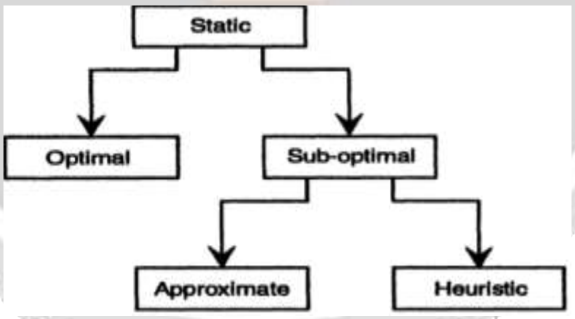


Figure . Static Scheduling

3.7 Dynamic Scheduling

At the time of execution, the process of redistribution a task within multiple processor dynamically is known as dynamic scheduling. This redistribution is performed by transferring tasks from the heavily loaded processors to the lightly loaded processors (called load balancing) with the aim of improving the performance of the application. A typical load balancing algorithm is defined by three inherent policies:
• information policy, which specifies the amount of load information made available to job placement decision-makers; • transfer policy, which determines the conditions under which a job should be transferred, that is, the current load of the host and the size of the job under consideration (the transfer policy may or may not include task migration, that is, suspending an executing task and transferring it to another processor to resume its execution); and
• placement policy, which identifies the processing element to which a job should be transferred.

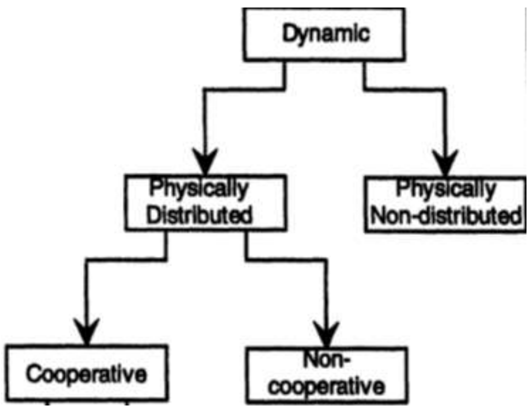


Figure. Dynamic Scheduling.

3.8 Multilevel Queue Scheduling

A multilevel queue scheduling algorithm partitions the ready queue into separate queues. For example, a common division is made between foreground (interactive) processes and background (batch) processes. Processes are permanently assigned to one queue. Each queue has its own scheduling algorithm. For example, foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm. In addition, there must be scheduling between the queues. This is commonly a fixed-priority preemptive scheduling. For example, the foreground queue may have absolute priority over the background queue. Another possibility is to time slice between the queues. Each queue gets a certain portion of the CPU time, which it can then schedule among the various processes in its queue. For instance, in the foreground-background queue example, the foreground queue can be given 75 percent of the CPU time for

RR among its processes, while the background queue receives 25 percent of the CPU to give its processes in an FCFS manner.

3.9 Round Robin

In Round robin scheduling algorithm, each request is handled consecutively in a manner that each server will load equal amount of request for example, there are 3 servers A, B and C which handled request in away such that server A will handled first request, server B will have handled second request and server C will have handled third request and again server A will handle request and so on this cyclic chain continue working until last request will served.

3.10 Weighted Round Robin

In web server farm there is an architectural difference between different server cluster so to handle each server effectively, weighted round-robin scheduling is designed to better handle servers with different processing capacities. Each server in a particular cluster is assigned a weight, an integer value that indicates the processing capacity. Servers with higher weights will serve first compair those with fewer weights, and servers with higher weights get more connections than those with fewer weights and servers with equal weights get equal connections. For example, the real servers, P, Q and R, have the weights, 5, 4, 3 respectively, a good scheduling sequence will be PPQPQRPQR in a scheduling period (mod sum Wi).

Scheduling sequence is depending up on the weight assign as per server architecture. Round robin scheduling works well when architecture of server is different. If the load of the requests varies highly then it leads to dynamic load misbalancing. In short, there is the possibility that a majority of requests requiring large responses may be directed to the same real server.

3.11 Idle Server Monitoring Algorithm

The idle-server monitoring algorithm runs within each cluster to monitor the set of servers in its cluster. The algorithm checks for any idle server in its cluster. If found, it assigns the task to the identified server. If the task cannot be assigned to server, the task is put into the queue. Within each cluster, Idle-Server Monitoring

Algorithm maintains all the server status in the table.

Thus, the Idle-Server Monitoring algorithm within the medium processing power server's cluster checks for any idle high processing power cluster. If a free server is found, it assigns its task to the identified high processing server. Similarly, Idle-Server Monitoring Algorithm within low processing power server's cluster checks for any free medium processing power server within medium processing power server's cluster. If free server is found, it assigns its task to the identified medium processing power server. By doing this, the resources are utilized effectively and the tasks are completed within the scheduled time. The system experiences no overloads and reduces request rejection.

3.12 . Dispatcher scheduling algorithm

Dispatcher scheduling algorithm maintains the concurrency within web server farm it keeps the record or watch over all the idle server and dispatch the client request to that server.
Dispatcher is clocked i.e. time assign to each process is recorded if process time exceed then it will have dispatched the incoming client request or shift the executing i.e. on running process to another idle process.

3.13 Cache replacement algorithm

Cache replacement algorithm is used to take decision on client as well as server side which helps to decide which part or section of cache memory is to be cleared when it is full or running out of memory.
The average memory reference time is

$T= m \times T_m + T_h + E$
where

T = average memory reference time
m = miss ratio = 1-hitRato.
T_m = time to make a main memory access when there is a miss (or, with multi-level cache, average memory reference time for the next-lower cache)
T_h= the latency: the time to referenc to the cache hit
E = multiple secondary effects, such as queuing waiting time effects in multiprocessor systems
Note: Latency and hit rate are main cause that effect cache, Cache must be automatic replaced (corn update) at the time of client server communication.

3.14 Dynamic Routing

Dynamic routing is used to route client request in an effective manner, it uses two algorithm
a.) R.I.P. Routing information protocol.
b.) O.S.P.F. Open shortest path first.
both algorithm is used to route packets using real time logical networking path, routing protocol operating on the router is responsible for the creation, maintenance and updating of the dynamic routing table.

4. THE PROPOSED ALGORITHM

In this paper, consider a new scheduling algorithm in server farms called Highest Response Ratio Next (HRRN). This algorithm used in process scheduling in operating system as well as server farm approach. The base of this algorithm is dispatcher scheduling. This algorithm is discussed in this paper and shown how to evaluate it. HRRN is perform well compare to SRPT as it solves a problem whereby long tasks may never get CPU time. If you imagine a system running with the SRPT algorithm that has a steady stream of processes coming in, it may be the case that a really long process never runs because there is always a shorter task waiting for the CPU. The HRRN algorithm fixes this by adjusting the priority of processes which are waiting to be run. If a process which will take a long time waits around for a while as a bunch of shorter processes come and go the system shortens the time that the scheduler thinks the long process will take. This of course doesn't make the long process complete any faster, but it does make it more likely to be scheduled. This repeats as long as the big job is waiting. Eventually, as the scheduler thinks the long job will be shorter and shorter, it is guaranteed to get the CPU. Just how long this will take depends on how the system is designed. Figure 3 is illustrating how can calculate priority of jobs in this algorithm.

$$Priority = \frac{wating\_time + estimated\_run\_time}{estimated\_run\_time} = 1 + \frac{waiting\_time}{estimated\_run\_time}$$

Figure 3. Priority in HRRN scheduling algorithm

5. THE SIMULATOR

The simulator was implemented using the C programming language on a Lenovo PC with CORE I3 processors and 4GB of RAM, running Microsoft Windows 10. The simulator uses: The job generator and the job dispatcher. The job generator is responsible for generating jobs randomly. Jobs are sequentially saved in a file in order to be served in servers. The role of job dispatcher is to select the jobs in correct order to reduce the average round around time. Finally, round around time of each job is saved in an output file and charted in Microsoft Excel.

6. SCENARIO

At first, jobs are generated randomly with random-served time, and then saved in a file. The dispatcher reads jobs from the file. Dispatcher is programmed to select the jobs whose average turn around time is very low. The FCFS, SRPT and HRRN algorithms are implemented in the simulator. To perform an experiment, 100 jobs were generated and the dispatcher dispatched the jobs on random servers. For each algorithm average turn around time is calculated

The SRPT has the best performance load balancing and average round around time, but starvation may be occurring when it is used. Therefore, this algorithm is not practical and is mainly used in comparison to other algorithms, i.e. the closest the average round around time of an algorithm to that one of SRPT, the better. To make simulate accurately different server architecture were virtually installed to logged down the impact on average turn around time. Algorithms act differently when the number of servers changed. When server count is less than 2, HRRN algorithm generate output which is same as SRPT algorithm. When numbers of servers are less in quantity or server cluster size is small then, HRRN behaves approximately similar to SRPT algorithm. When server cluster is huge in web farm then HRRN and SRPT algorithms performance is improved over FCFS algorithms. Figure 4 illustrates the scenario of simulation for evaluation and comparison of the algorithms.

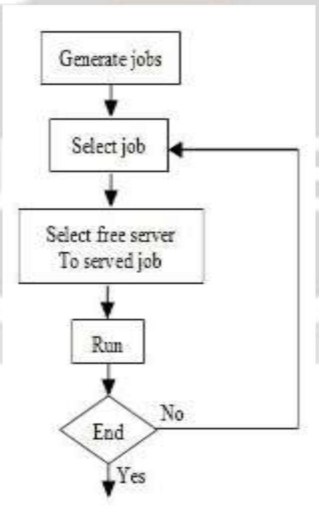


Figure 4. Scenario of simulation[1]

7. COMPARISON

The load balancer simulation was used for analyzing as well as comparing the scheduling algorithms presented in the last section. For experimenting, 100 jobs are generated randomly with random estimated times. Then the simulator dispatches incoming jobs on servers and calculates the served time for each job. The mean round around time is also computed all jobs. For studying the dependency of algorithms, different servers in server farms were configured for handling the same defined jobs. The results are different based on the number of servers used. This means that the number of servers impacts on the performance of the algorithms. The results are present in Table1. According to numbers of servers used in simulation, the mean around time for each algorithm is calculated[1].

TABLE 1. The simulation results

Number of servers	25	15	6	1
FCFS	2342 ms	3657 ms	8588 ms	49740 ms
SRPT	1880 ms	2763 ms	6116 ms	34174 ms
HRRN	2034 ms	2976 ms	6383 ms	34179 ms

As illustrated in fig a, results for HRRN and SRPT algorithm is same when one server is used The vertical axis indicates the number of jobs and horizontal axis indicates the mean round around time in milliseconds.

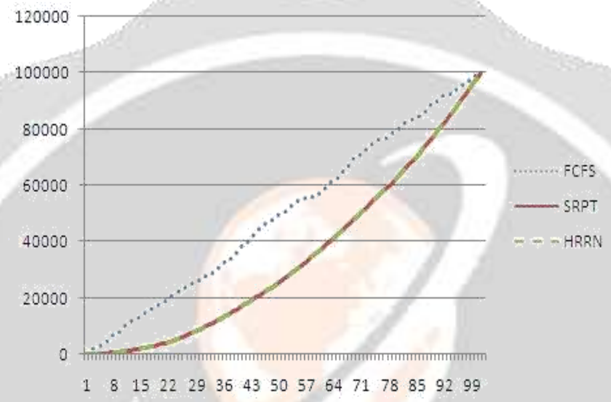


Figure a. Results for one server configuration

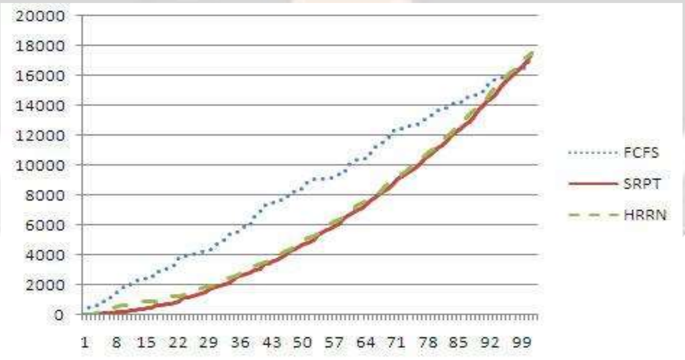


Figure b. Results for six server configuration

This graph shows that HRRN and SRPT algorithms act similarly. According to Table 1, the number of servers impacts on the server farm performance. The functionality of HRRN algorithm becomes more similar to that of SRPT As the number of servers started decreasing. As the number of servers increases, the difference between HRRN round around time and SRPT round around time grows.

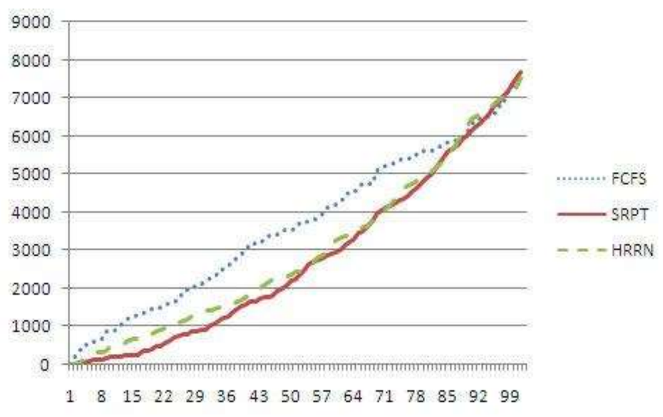


Figure c. Results for 15 server configuration

Increasing the number of servers has a small impact on this difference. This fact is illustrated in Fig c and d.

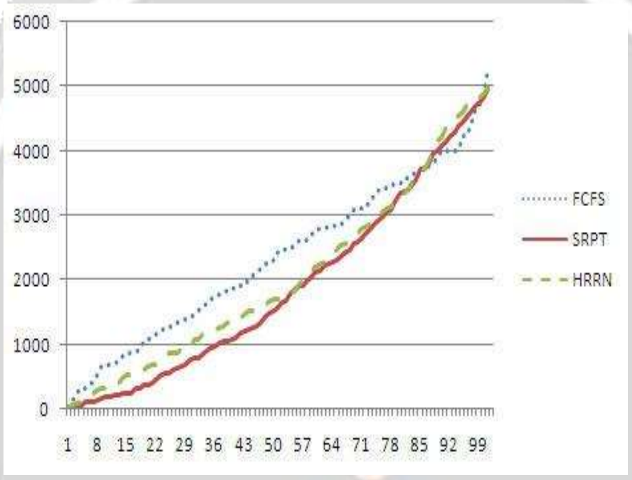


Figure d. Results for 25 server configuration

8. SERVER FARM SECURITY

8.1 Web server logs reviewer

Web server logs reviewer is suitable to understand who is currently login or who is anonymously watch over the server resources, it will help in backup data or any intrusion is detected through anonymous IP address is logged over system files, one of the major drawback is that log reviewing only help after intruder enters the system and alter the server resources, logs helps to detect path, pattern and location of infected file which intruder placed on server. Logs is not an early warning system instead it is used for recovery purpose.

8.2 *Intruder Detection*

Intruder detection is a process to detect the intruder while accessing unauthorised information from the server and intentionally updating and patching files under server, to detect these type of intruder software system is used which is known as Intrusion Detection (ID). ID used to prevent intrusions as well as provide notified information in the case of a successful intrusion it will look over all authorised IP address and create secure channel towards host networks to overcome the intrusion attack.

8.3 *Honeypot*

Honey Pot Systems are servers systems which is setup to provide integrate information regarding an attacker or intruder into your system. Honeypot is addition to security system, it is not replacing traditional security system.

Honey Pots is installed both inside or outside or in the DMZ of a firewall design .In a sense, they are variants of standard Intruder Detection Systems (IDS) but with more of a focus on information gathering and deception.

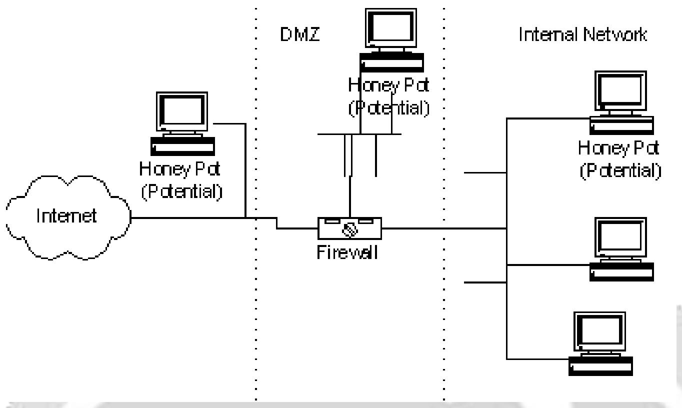


Figure. Honeypot Installation on DMZ firewall

.9. CONCLUSION

Amongst all scheduling algorithm, the shortest remaining process time (SRPT) algorithm has best performance among but suffers from starvation. The mean around time of First-Come-First-Served (FCFS) algorithm is too high. Any scheduling algorithm that is more similar to SRPT algorithm has better a performance. The proposed algorithm described in this paper uses HRRN algorithm in server farms dispatcher to distribute incoming jobs on the servers. The advantage of using this algorithm is better performance. The mean around time of SRTP and HRRN is same. The most advantage of HRRN algorithm is overcoming starvation [1]. This algorithm does not suffer from any starvation and has the same performance as SRPT does; therefore, using the HRRN algorithm in the server farms dispatcher can increase the performance.

Now to protect server farm reviewing logs is not good idea so, most organisation implement either Honeypot or Intruder detection both are best in case of security apart from this most organisation prefer paid firewall setup such as Norton firewall.

10. REFERENCES

[1] A new scheduling algorithm for server farms load balancing

[2] A Predictive Modified Round Robin Scheduling algorithm for web server cluster- PProceding of the 34th Chinese Control Confrence July 28-30-2015

[3] A new load balancing scheduling algorithm based on Linux Virtual Server 2013 International Conference on Computer Science Application.

[4] How South African SMEs address cyber security: the case of web server logs and intrusion detection- 2016 IEEE International Conference on Emerging Technologies and Innovative Business Practices for the Transformation of Societies.

[5] Assessing and Comparing Security of Web Servers 2008 14th IEEE Pacific Rim International Symposium on Dependable Coputing

[6] E. Casalicchio, S. Tucci, "Static and Dynamic Scheduling Algorithms for Scalable Web Server Farm", Proc. of 9th IEEE Euromicro Workshop on Parallel and Distributed Processing, PDP2001, Mantova, Italy Feb. 2001.

[7] E. A. Brewer. (2001, Jul/Aug) Lessons from giant- scale services. IEEE Internet Computing. 5(4). pp. 46-

55

[8] Pinedo, M.: Scheduling: theory, algorithms, and systems. Prentice-Hall, Englewood Cliffs (1995)

[9] Schrage, L., Miller, L.: The queue M/G/1 with the shortest remaining processing time discipline. Operations Research 14(4), 670–684 (1966)

[10] Schrage, L.: A proof of the optimality of the shortest remaining processing time discipline. Operations Research 16(3), 678–690 (1968)

BIOGRAPHIES.

	Syed Mohammed S. Abidi MCA Student, Mumbai University Working On Computer Application, Web Development and Mobile Networks Technology. Mac_universe@live.com