Heuristics Load Balancing Algorithms for Video on Demand Servers

Alok Kumar Prusty, Bibhudatta Sahoo

Abstract**—** The Web Services have gained considerable attention over the last few years. Video-on-Demand (VoD) systems have resulted in speedy growth of the web traffic. Therefore the concept of load balancer aimed to distribute the tasks to different Web Servers to reduce response times was introduced. This paper attempts to analyze the performance of FCFS, Randomized, Genetic algorithms and Heuristics algorithms for selecting server to meet the VoD requirement. Performances of these algorithms have been simulated with parameters like makespan and average resource utilization for different server models. This paper presents an efficient heuristic called Ga-max-min for distributing the load among servers. Heuristics like min-min and max-min are also applied to heterogeneous server farms and the result is compared with the proposed heuristic for VOD Servers. Ga-max-min was found to provide lower makespan and higher resource utilization than the genetic algorithm.

Keywords**—** Makespan, Resource Utilization, FCFS, Random, Genetic, Max-min, Min-min.

I. Introduction

Webserver is a program that provides content like web pages over the World Wide Web. The simultaneous open connections to the web server are generally limited. Thus the waiting time becomes high when the number of requests to the web server is large resulting in DOS (Denial of Service) attack. An effective solution to this problem is the use of multiple servers known as clustered Web Servers or a server farm. Multimedia communications require continuous service, i.e. read, process and transfer the information should be done with minimum delay which is vastly improved if we use a server farm.

The performance of a server farm depends on the type of routing, server capacity and scheduling policies used. The server capacity can be homogeneous or heterogeneous. In case of homogeneous systems, each of the servers in the server farm are of equal capacity and the request is processed by the server having the least number of tasks in the queue, i.e. Join the shortest queue policy[3].Heterogeneous systems scores over homogeneous systems if tasks are of different sizes. Heterogeneous systems can also include task-specific systems, i.e. for more computation oriented tasks we can use an array processor.

Load Balancing Policy consists of load index policy, information collection policy, task location and task transfer policy. In our approach we assume that the nature of task coming to the web server is known beforehand. Load index policy keeps track of the number of tasks in the queue and information collection policy has the knowledge about the type of tasks coming to the server farm and the nature of web traffic distribution. This can be done by checking the server log file and obtain information like average page views, busy times, visit duration and the most requested page by the customer. Task transfer policy decides whether the task has to be serviced in the local servers or sent to other servers located remotely. Our main focus is on the task location policy which describes scheduling algorithm for the various tasks. We also assume an infinite capacity front end dispatcher which assigns the tasks to various servers.

In this paper we examine the different scheduling algorithms, first come first serve, random and genetic algorithm. The metric for comparing different algorithms is makespan. Makespan is defined as the maximum time taken to complete all the tasks given to the dispatcher or load balancer. An advantage for using genetic approach is that there is no need to set any threshold values on the number of tasks or utilization of the server. The server load can be represented by the following equation [21]

$$\begin{aligned} Bandwidth &= AverageDailyVisitors \\ &\times AveragePageViews \times AveragePageSize \times 31 \\ &\times FudgeFactor \end{aligned} \tag{1}$$

If people are allowed to download files from the site, the bandwidth calculation becomes:

$$Bandwidth = \left[\begin{pmatrix} AverageDailyVisitors \times AveragePageViews \times \\ AveragePageSize \end{pmatrix} + \\ \left(AverageDailyFileDownloads \times AverageFileSize \right) \right] \times 31 \times FudgeFactor \tag{2}$$

Average Daily Visitors - The number of people expected to visit a site, on average, each day. It may vary significantly on the basis of how a site is marketed.

Average Page Views It represents the average number of web pages visited by a person.

Bibhudatta Sahoo is with Department of Computer Science & Engineering, National Institute of Technology, Rourkela, ODISHA, INDIA, PIN-769008. (phone: 91-661-2462358; fax: 91-661-2462351; e-mail: bdsahu@nitrkl.ac.in)

Alok Kumar Prusty is with Department of Computer Science & Engineering, National Institute of Technology, Rourkela, ODISHA, INDIA, PIN-769008. (phone: 91-8050249700; e-mail:aloksworld1988@gmail.com)

Average Page Size It shows the average size of the web pages, expressed in kilo-bytes (KB).

Average Daily File Downloads - The number of downloads expected to occur from a site. It depends on number of visitors and average downloads per visitor.

Average File Size - Average size of files that are downloadable from the site.

Fudge Factor - A number greater than 1. A fudge factor of 1.5 implies that the estimate is off by 50.Usually, bandwidth is offered in terms of Gigabytes (GB) per month. Hence the entire formula is multiplied by 31.

We then focus on a particular application of web servers: Video on Demand. VOD servers are different from normal web servers because they demand a consistent and higher data rate. They find applications in Video Conference (VC), IP telephony, Multimedia Mail and Digital Libraries [9]. The demand for on demand video services have increased significantly in the recent years and is expected to rise further due to advancement in technology to meet the high Qos required by VOD applications. In fact, commercial VoD services with complete video cassette recorder (VCR) functions have appeared. However, owing to ever increasing user demands, when the user access rates increase, several issues need to be tackled, e.g., long startup delay, jitters etc. The Qos as desired by the users are generally subjective in nature. So they must be mapped to an appropriate objective (quantitative) parameter so that we get a tech-nically correct application.

VOD networks followed centralized architecture in the early days. But with increase in number of requests the trend has shifted to distributed architecture for VOD networks. As the number of requests increases the number of servers required to cater to those request increases which adds to additional cost. If by some heuristics or means, we can efficiently allocate the tasks to the different available servers such that it optimizes the value of a metric like makespan and throughput, then the customer requirements can be met in a better manner.

VOD is a relatively new concept. Many of the existing load balancing algorithms has not been applied to VOD systems. Further, the need of proper server selection is necessary for maintaining high data rate (e.g. 1.5Mbps for MPEG video) and minimizing the cost of service. The objective of this paper is twofold. Firstly to analyze the existing algorithms and heuristics in the context of VOD based systems, and secondly to analyze the performance of the proposed heuristic for two metrics namely Makespan and Average resource utilization.

II. Related Work

Server Selection, Load balancing and scheduling issues have been studied quite extensively in the past. Most notable of the server selection algorithms [16] are the closest server algorithm that selects server based on the proximity to the client, optimized closest server algorithm that chooses the closest server among the free channels, Register all algorithm where the clients request is added to the queue of all the servers and Maximum-MFQ-rank-first algorithm which computes the rank at the various server queues and assigns the request to the server having the best rank.

In light of the load balancing problems, Haight(1958), Halfin(1985), and King-man(1961) are among the many people that studied join the shortest queue policy using two parallel servers with infinite buffer size. Gupta et al.[3] analyzed the join the shortest queue policy on processor sharing server farms. They used a single queue approximation and investigated the sensitivity of the queuing model to variations. Niyato et al. [4] studied load balancing for Internet video and audio server. They studied and compared various algorithms like Adaptive bidding, Diffusion and State change broadcast along with traditional round-robin and random algorithms. Wang et al. studied load balancing in heterogeneous systems, first considering two servers with different service rates and then extending their observations to multiple servers. This involved multiple thresholds setting which was done by heuristic methods. Ciardo et al. [6] devised a strategy for task allocation in web servers based on size distributions of the requested documents. Zhang et al. [2] analyzed the central load balancing model, derived average response time and the rejection rate and compared three different routing policies. The retrieval schemes for VOD can be classified into two categories, a) Disk level retrieval schemes[9] which focuses on synchronizing and efficiently using the data between different storage devices and b)server level retrieval schemes[9] which delivers data to the client whenever the need arise. Our approach is based on the server level. Similar requests can be batched or the server can be replicated [16] to achieve low latency and thus serve a higher number of requests.

File Access Model is useful during the video caching where the cache content is determined by the popularity or the hit ratio of the multimedia file. As time progresses, the cache content needs to be updated so that the cache contains the most popular video files. Previous studies have followed the Zipf's Law to calculate the popularity of the video files [12-14]. In Zipf-like distributions, the access frequency for a file of popularity rank i is equal to C/i_a, where C is a normalization constant and a(a>0) is the distribution parameter[8].The file usage patterns like which category of videos are accessed at which point of time during a day can also be analyzed and the cache be maintained accordingly.

III. VoD System Architecture

The adapted Figure 1[17] depicts the prevalent 3-tier architecture for web servers. The main components are a set of web servers, a set of database server nodes and a switch which executes the logic for server selection. It can divide the tasks into classes on basis of quality metric like burst time, etc. The Front end servers are designed to de-liver the static pages mostly and in case of any query from the client the appropriate database server is connected. The routing and firewall switch ensures authorization and authentication and forbids any unintended user from accessing the files on the servers.

The system architecture of a Video on Demand system basically consists of three major parts [11]: a client, a network, and a server. Each part can be subdivided further into components and interfaces.

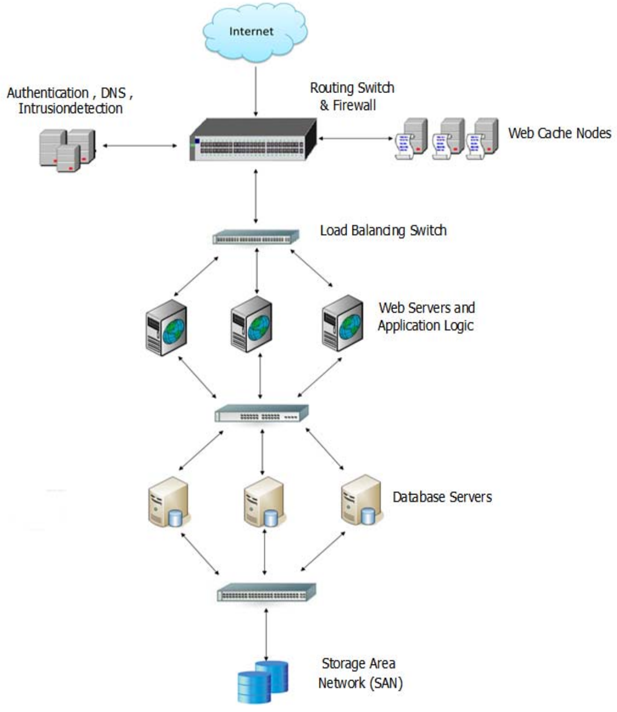
Internet
Authentication , DNS , Intrusiondetection
Routing Switch & Firewall
Web Cache Nodes
Load Balancing Switch
Web Servers and Application Logic
Database Servers
Storage Area Network (SAN)

Figure 3.1 Three tire architecture for web server

VOD system from the client's point of view is a simple operation. The user makes a selection from a list of available videos and the video is delivered to the user within the accepted QoS limits. Most networks use proxy servers or replicas to minimize delay .This is done by a process called request routing which directs the request to a particular web server on the basis of certain metrics. According to [7, 8], there are 4 kinds of architecture for VOD networks a) Centralized, all the requests from the clients are handled at the original server, b) proxy based servers that are located close to the user end to reduce the load on the original server by caching, c) Content delivery networks, the servers are deployed close to the edge of the network to serve a fraction of clients request and d) hybrid, is basically a peer to peer approach.

IV. Server Model And Performance Metrics

In our proposed solution we use a distributed architecture where the request first comes to a front end server from the client and after successfully passing through the authentication phase the video list is displayed in the web page and if the requested video is found in the page then the video is delivered to the client through local caching else it goes to a VOD server decided by the server selection strategy which then sends the desired video to the client. The arrival rate follows a Poisson distribution because that is the common mode of distribution for most of the internet traffic. The tasks are also assumed to have negligible inter dependency among them. We neglect the different cost parameters and our sole focus is based on server selection keeping other parameters fixed. The detail sequence diagram for the centralized system is shown in the figure 2.

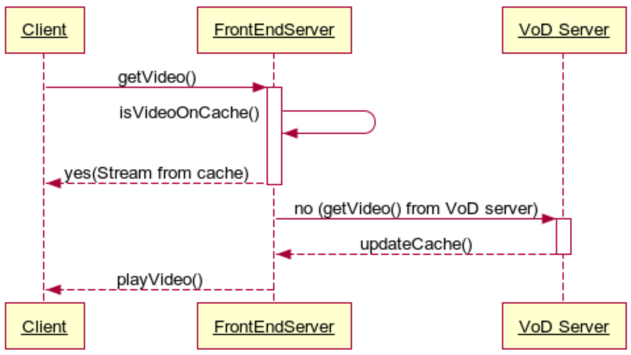

Figure 4.1 Sequence diagram for VoD retrieval

Many different metrics are used to evaluate the performance of VOD server. They can be classified as Technology based and user based [9]. We have used Makespan and Resource utilization as two metrics for comparing algorithms. Makespan indirectly refers to the Response time of the system as a certain response time of say 0.5 sec implies that the requests should complete execution within 0.5 sec which suggests the makespan should not exceed 0.5 sec. Resource utilization suggests what fraction of the total time a server is working.

A. *Makespan*

Makespan is defined as the largest completion time of all the tasks in the system. In the VOD scenario, it is an indicator of the response time .For example; if the makespan of a group of tasks exceeds a certain threshold then the tasks are not allowed as the response time QoS is not met.

B. *Resource Utilization*

Average Resource utilization for a system is defined as the average of the resource utilization of various servers. For a single server, utilization is given by

$$\mathrm{Re}\,sourceUtilization(R_u) = (AmountOfTimeServerisIdle)/TotalTme \qquad (3)$$

V. Typical Server Selection Algorithms

Let there be a task set T consisting of n(T1, T2... Tn) tasks and let there be M servers. The basic problem is where to map a task Ti among the M possible servers. This is done by the server selection strategy. The tasks should be allocated in such a way that after allocation of all the n tasks among the M servers, the performance metrics should be optimized.

The need for server selection arises in case of distributed system architecture. Choosing a good strategy is important because of the following reasons

1. It can reduce the overall cost of maintenance of the system.
2. It can reduce the response time of the system, thereby increasing customer satisfaction.

3. It can efficiently distribute the load among various servers and thus reduces the chance of breakdown of a particular system due to server overloading.
4. It can provide robustness and easy scalability to the system.

So selecting a good strategy is of paramount importance. But each of the existing algorithms does not apply well to all the scenarios. The existing algorithms can be further divided into Traditional and Heuristic based algorithms. Traditional ones include FCFS, Random and Genetic algorithms. There is another class of algorithms called the Heuristic algorithms which comprises of Min-min, Max-min and Weighted mean time scheduling [18]. These heuristics are applicable for heterogeneous task systems where we have servers of different capacity.

A. First Come First Serve

This is a simple scheduling policy used in various load balancing servers. Whichever request comes first is served first irrespective of any other criteria. So these processes undergo starvation.

B. Random

This is another scheduling policy where the tasks are distributed randomly to the available processors. If the distribution is truly random, then the random outweighs other algorithms in the long run.

C. Genetic

The genetic algorithm is an optimization technique that has it base on the basis of natural selection. A GA consists of candidates or populations which evolve based on some predefined rules such that each evolution produces a better population (i.e. population which minimizes the cost function). Some of the advantages of GA are

- It optimizes both continuous and discrete variables.
- It simultaneously searches from a wide sampling space.
- It is well suited for parallel computing.
- It optimizes complex cost functions quite well (there are several local minima) and produces the global minima.
- It provides a list of optimal solutions not the single best optimum solution.
- Encoding the variables is easy when they are represented in terms of genes. GA essentially operates in five steps initialization, evaluation of fitness function, selection, crossover, mutation. [1]

D. Min-min

This consists of two phases [18, 19]. First we choose a fixed arbitrary order and then for each task we choose the server with the minimum burst time. In the second phase, the task with the minimum burst time among the group chosen is phase 1 is selected and assigned the corresponding server and the ETC matrix is updated with new completion times for the remaining tasks while the chosen task is deleted from the matrix. Completion time is given by the equation.

$$CT(i,\ j) = ET(i,\ j) + r(j) \tag{4}$$

Where r(j) is the ready time of machine j, i.e. the time taken by the machine to complete all its pending tasks from the moment the task i is assigned to machine j. The maximum time to complete all the tasks is represented by the makespan.

E. Max-min

This algorithm is similar to min-min except in the second phase the task with the maximum completion time is mapped first. This algorithm is known to provide better resource utilization than the Min-min algorithm.

F. WMTS

The algorithm is adopted as described in [18] where the weighted sum of expected time is used. The weights are proportional to the server capacity.

G. Composite GA-Max-min Server Selection Algorithm

This algorithm merges the genetic algorithm and the Max-min algorithm. This results in the enhancement of the performance of the genetic algorithm. So this kind of algorithm can be used where the number of tasks is very large.

Algorithm **GA-Max-min**

```
1. For all tasks i in the task set
2. Divide the tasks into classes on basis of burst time or previous history
3. Send the tasks to the appropriate queue
4. Apply different selection algorithms as applicable to the different queues
5. Makespan=Calculate makespan (Task set)
6. Resource utilization =Calculate resource utilization (Task set)
7. End
```

The task set is divided into classes based on burst time or previous history. Then for each queue Resource utilization and makespan is calculated by the calculate resource utilization() and calculate makespan() functions. Both these functions take task set as the input.

VI. Experimental Results

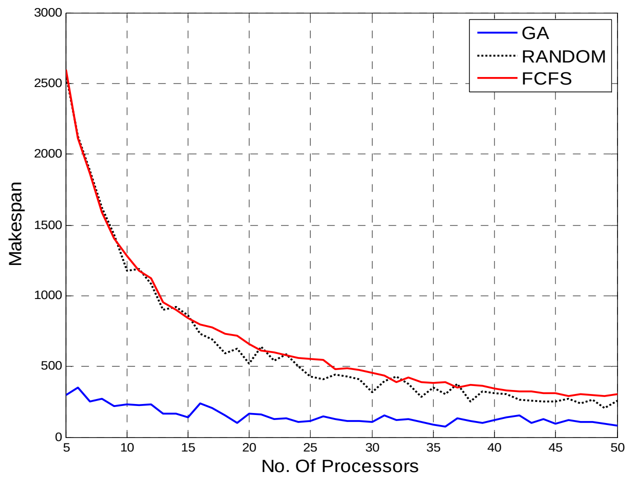

Figure 6.1 Comparison between GA Random and FCFS for maximum 50 nodes

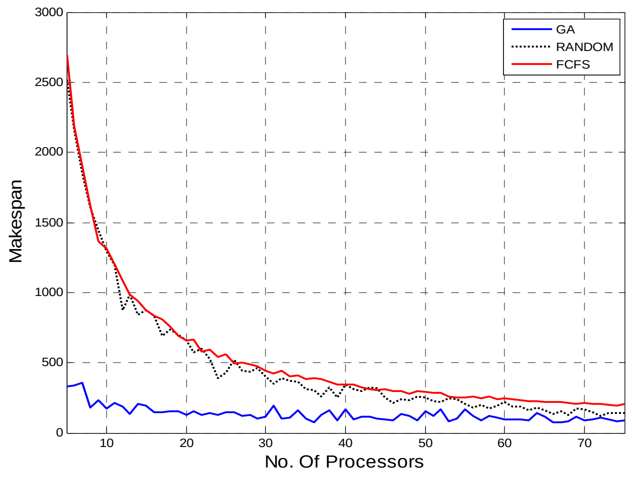

Figure 6.2 Comparison between GA Random and FCFS for maximum 75 nodes

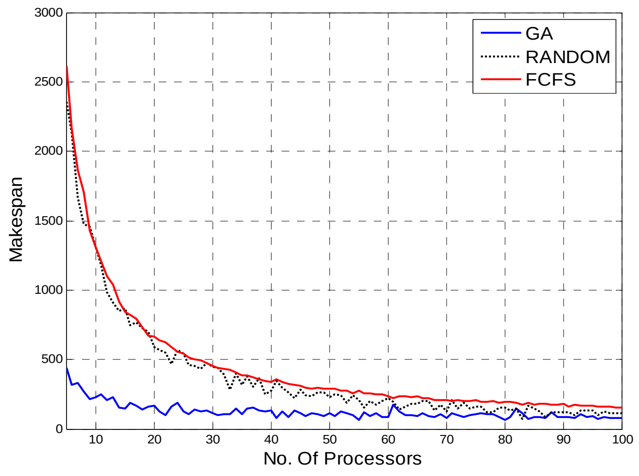

Figure 6.3 Comparison between GA Random and FCFS for maximum 100 nodes

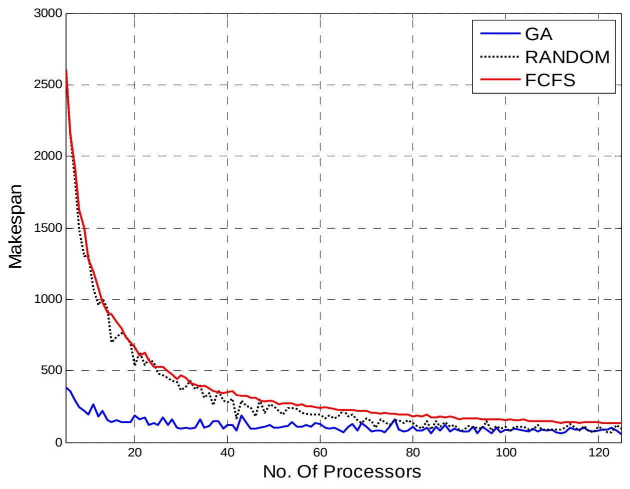

Figure 6.3 Comparison between GA Random and FCFS for maximum 125 nodes

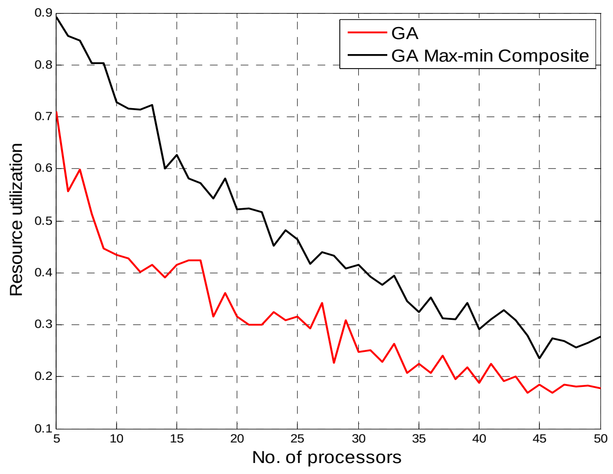

Figure 6.4 Resource Utilization Vs No. of processor (Max 50)

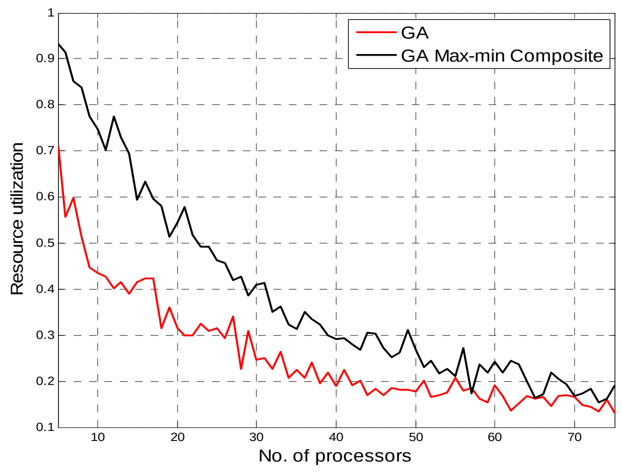

Figure 6.5 Resource Utilization Vs No. of processor (Max 75)

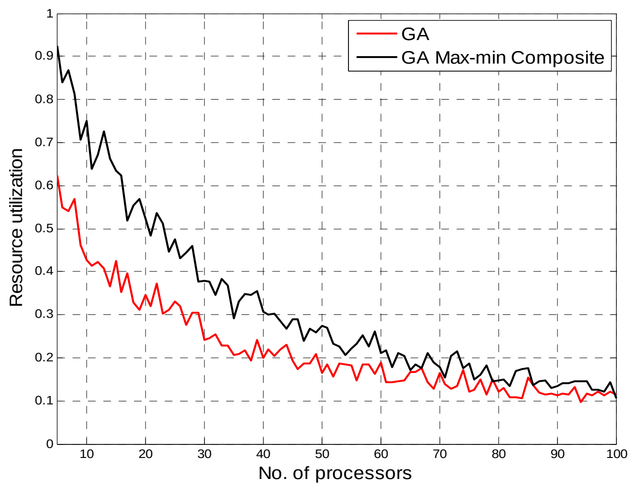

Figure 6.6 Resource Utilization Vs No. of processor (Max 100)

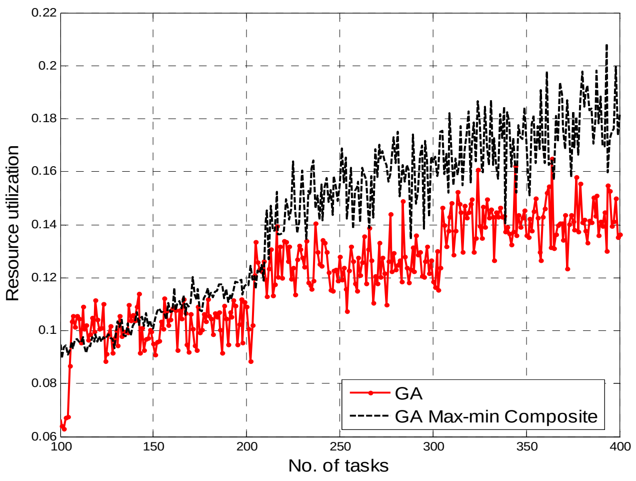

Figure 6.8 Resource Utilization Vs No. tasks (Max processor 100)

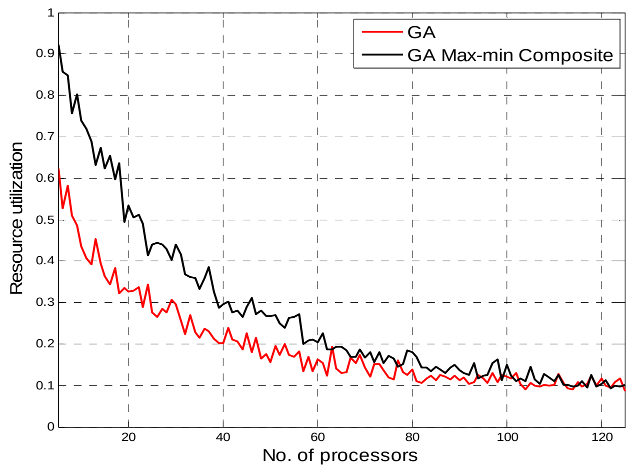

Figure 6.7 Resource Utilization Vs No. of processor (Max 125)

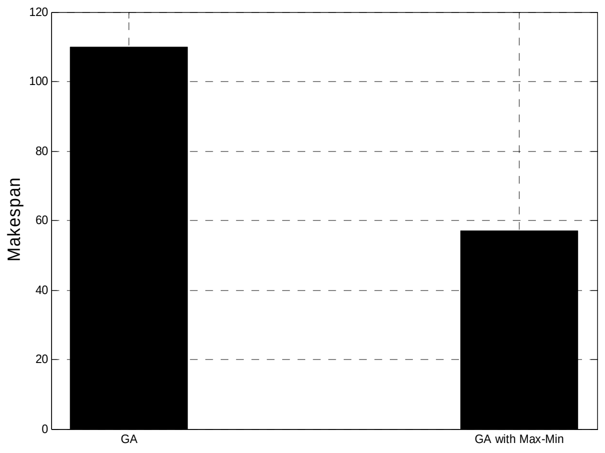

Figure 6.9 Makespan Comparison between GA and Composite GA (max processor=50)

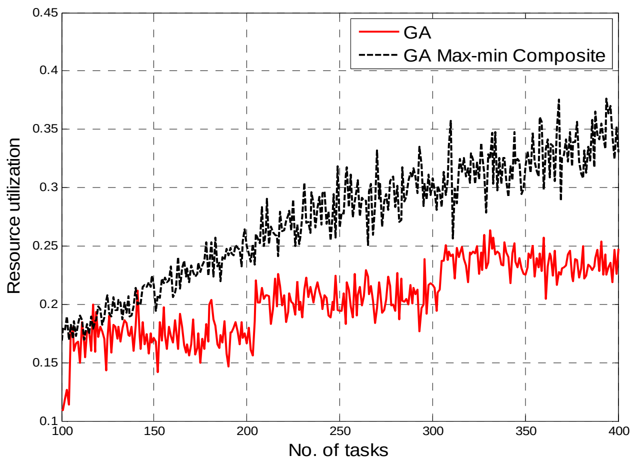

Figure 6.7 Resource Utilization Vs No. tasks (Max processor 50)

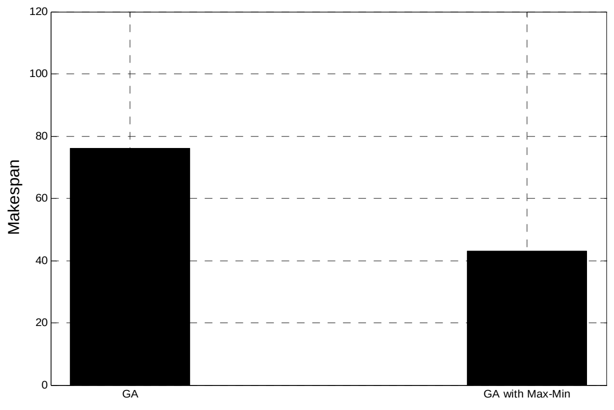

Figure 6.11 Makespan Comparison between GA and Composite GA (max processor=100)

The GA-Max-Min algorithm gives less makespan and high resource utilization if we vary number of processors.

VII. Conclusion

In this paper we compared the various server selection algorithms like FCFS, Random, Min-min on the basis of makespan and Average resource utilization and showed the composite GA algorithm performs better. We combined two heuristics Genetic algorithm and max min to get a new heuristic GA-max-min. We chose Genetic algorithm as one component of the combined heuristic as it was feasible to apply genetic algorithms for large data sets and the max min algorithm as another component as it provides the best resource utilization. In future other parameters other than makespan and resource utilization like scalability and throughput can be used to analyze the performance of these algorithms. Further a complete algorithm framework can be formed by combining various algorithms as an extension of Ga-max-min which caters to varying nature of the tasks and the switch can dynamically change algorithms as the request rate varies.

References

[1] A. Y. Zomaya and Y. H. Teh. on using Genetic algorithms for dynamic load balancing. IEEE transactions on Parallel and Distributed Systems, vol. 12,no. 9,September 2001.

[2] Z. Zhang and W.Fan. Web server load balancing: A queuing analysis. European Journal of Operational Research, vol. 186, no. 2, pp. 681-693, April 2008.

[3] V. Gupta, M.H. Balter, K. Sigman and W.Whitt. Analysis of join-the-shortest-queue routing for web server farms. Performance Evaluation, vol. 64, no. 9-12, pp. 1062-1081, October 2007.

[4] D. Niyato and C.Srinilta. Load balancing algorithms for Internet video and audio server. Proceedings of 9th IEEE International Conference on Networks, pp. 76, 2001.

[5] J.L. Wang, L.T.Lee and Y.J.Hunag. Load balancing policies in heterogeneous distributed systems. Proceedings of 26th Southeastern Symposium on System Theory,pp. 473-477, 1994.

[6] G.Ciardo, A.Riska and E.Smirni. EQUILOAD: A load balancing policy for clustered web servers. Performance Evaluation,vol. 46, no.2-3, pp. 101-124, October 2001.

[7] F.T houin. VOD equipment allocation.Tech. report, Mcgill University Montreal, Canada.

[8] F.Thouin and M.Coates. VOD networks: Design approaches and future challenges, Proceedings of Network IEEE.pp. 42-48, montreal,March-april 2007.

[9] N. Panigrahi and B. Sahoo. Qos based retrieval strategy for video on demand.Available

[10] Online at http://dspace.nitrkl.ac.in:8080/dspace/bitstream/2080/789/1/bdsahoo2009.pdf. Last visited may 08 2011.

[11] N. Carlsson and D.L.Eager. Server Selection in large scale Video on Demand. Proceedings of ACM transactions on multimedia, computing and communications, February,2010.

[12] M. Ko and I. Koo. An Overview of Interactive Video On Demand System. Technical Report, The University of British Columbia, Dec 13,1996.

[13] N. Jian et al. Hierarchical Content Routing in Large-Scale Multimedia Content Delivery Network .Proc. IEEE Intl. Conf. Commun. (ICC), Anchorage, AK, May 2003

[14] B. Wang et al. Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution .Proc. IEEE Infocom, New York, NY, June 2002.

[15] T.Wauters et al. Optical Network Design for Video on Demand Services. Proc. Conf. Optical Network Design and Modeling, Milan, Italy, Feb. 2005.

[16] D. Ligang, V. Bharadwaj, and C. C. Ko. Efficient movie retrieval strategies for movie-on- demand multimedia services on distributed networks. Multimedia Tools Appl., vol. 20, no. 2, pp. 99133, June 2003.

[17] M. Guo et al. selecting among replicated batching video on demand servers. Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video, May 2002.

[18] S. Shari_an, S.A. Motamedi and M.K. Akbari. A predictive and probabilistic load balancing algorithm for cluster based web servers.Proceedings of applied soft computing, pp. 970, Jan 2011.

[19] S. S Chauhan and R.C.Joshi. A weighted time min min max-min selective scheduling strategy for independent tasks on grid. Proceedings of Advance Computing Conference (IACC), Patiala, India, Feb 2010.

[20] M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. Freund. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. 8th IEEE Heterogeneous Computing Workshop (HCW '99), pp. 30-44, San Juan, Puerto Rico, April 1999.

[21] A. Narasimhan, Distributed multimedia applications-opportunities, issues, risk and challenges: a closer look .IASTED International Conference on Intelligent Information Systems , pp.455-460, 1997

[22] http://www.findmyhosting.com/bandwidth-explained/,Last visited 16th march 2011.

Alok Kumar Prusty is a graduate student from the department of Computer Science and Engineering NIT Rourkela. He is currently working as a software engineer in Sapient, Bangalore, India. His areas of research interest include distributed computing, algorithms, performance evaluation methods, machine learning and data mining.

Bibhudatta Sahoo is presently Assistant Professor in the Department of Computer Science & Engineering, NIT Rourkela, INDIA. His technical interests include performance evaluation methods and modeling techniques in distributed computing system, networking algorithms, scheduling theory, cluster computing and web engineering. He is a member of IEEE and ACM.