Study of Load Balancing Schemes over a Video on Demand System

Priyank Singhal  
Student, Computer Department  
Mumbai University

Ashish Chhabria  
Student, Computer Department  
Mumbai University

Nupur Bansal  
Student, Computer Department  
Mumbai University

Nataasha Raul  
Research Scholar, Computer Department  
Mumbai University

Abstract: Load balancing algorithms on Video on Demand systems very significantly on considering different requests which may differ in structure as I/O intensive or CPU/Memory intensive, due to which the performance of the algorithm varies. However, statistically a large number of tasks are I/O intensive rather than CPU or memory intensive. CPU requirements of the system are minimum. The goal of this proposed algorithm is to balance loads evenly across the entire cluster factoring in the CPU as well as I/O requirements, while at the same time trying to achieve an optimum completion time for each task. An assumption is made to not consider pre-emptive migration of tasks. A transaction in our model can be defined as the time difference in the acceptance and fulfilment of requirements of a task by the system. Task requirements in general for a video-on-demand system are video files stored on the server end on a secondary storage device which is to be streamed continuously to the request initiator. The proposed algorithm called CPU-Memory-I/O Load Balancing (CMIOLB) is compared to a couple of other scheduling and allocation policies namely the First Come First Serve (FCFS) and the Random distribution. We have mathematically shown that the proposed algorithm performs better than the other policies.

Keywords - Video on demand, I/O, intensive task, Load balancing

1. INTRODUCTION

Video-On-Demand commonly called VOD service has been possible due to availability of a number of new and developed technologies such as ADSL (Asymmetric Digital subscribers Line), MPEG (Moving Picture Expert Group), ATM (Asynchronous Transfer Mode). An important challenge involved is in storing a large amount of video objects such that each video stream can be accessed and transmitted to the client in real time. Multimedia servers have become more essential since the demand for multimedia (such as text, audio, image and video) has risen. Content-based retrieval of multimedia data faces 3 major problems. Firstly, the content is subjective; this needs a powerful set of search routines including keywords, sounds, texture, colour, spatial information and motion. Secondly, if a processing technique is designed and developed for one type of feature, it’s usually not applicable for others. For instance, a technique designed for indexing audio data may not be usable for image data or, a method developed for a colour may not be useful for a texture in image and video content. Thirdly, the usual huge size of multimedia data requires an exhaustive search. Video-on-Demand systems can be grouped as a single server or multi server architecture. Multi-server architecture proves to be better since it processes client's requests more efficiently while an isolated server cannot handle the traffic efficiently. A video on demand system is generally comprised of a distributed cluster of computing nodes. These nodes may be homogenous or heterogeneous. In the past, many problems have been addressed in designing a VOD system, such as data placement, disk scheduling, admission control, syncing, resource management, and tolerance. However the problem about dynamic load balancing among the servers is rarely looked at. Although previous studies have explored the load balancing issue, most are applied to general tasks, not to video tasks. Furthermore, the data placement policies above can be used to achieve load balancing in the servers, but, they are static. This is not good enough. One of the most common data placement methods is to strip video objects across all the disk servers on the system, thus, avoid the imbalance. However, the approach suffers from the quite a few drawbacks. It results in additional complexity such as types of synchronization in delivering a single video object from multiple disks/servers. Secondly, it is not entirely practicable to assume that a system must be constructed using homogeneous disks. Third, as client demand and/or data grows, the system requires one or more servers, thus resulting in restriping of all video data objects. Another data architecture strategy is to replicate popular video content among the disks. This too, has some problems such as requiring extra storage space and deciding the time to perform reverse of replication. Certain models are distributed VOD servers, completely different from clusters of VOD servers.
II. SYSTEM ARCHITECTURE

The model of Video on Demand system is shown in Fig 1. All servers in the system are connected with a high-reliability network, such as an ATM switch, a fast Ethernet panel, or a crossbar switch.

![Fig 1. System Architecture](image)

Every server has its CPU, memory, and I/O, but they do not have the same capacity. This means that the system could achieve good scalability. When a client issues a request, a specific server known as ‘dispatcher’ will filter the above request. It then transfers it to the destination server with the requested object. Then the destination server then delivers the video stream to the client at a given rate through the external network until the display of the stream is finished. VOD system is disk-bound. The network bandwidth from servers to clients is fast enough to deliver video streams and it is also fault free. The VOD clusters can be modelled by the queuing theory showed in Fig 2.

![Fig. 2. Queuing Model](image)

Here, the cluster is connected to the web by the network which provides a two-way path, for the requests to the cluster, and the response. We assume a constant arrival rate that is Poisson's distribution. The nodes are heterogeneous, thus they have different service rates denoted by \([S1, S2, S3 ..., Sn]\). The requests arrive from the user via the web to the dispatcher. The dispatcher allocates requests to the different nodes. This then generates the response back to the user.

III. ALGORITHM OF DYNAMIC LOAD DISTRIBUTION

The proposed CMIOLB algorithm is meant to be suited for a wide variety of versatile workloads that may be either I/O intensive or CPU intensive. However, is in the general case, in our paper we have considered CPU-intensive tasks to be at the minimum as most requests coming to the server end are video retrieval tasks which require low CPU but high I/O consumption. Our proposed algorithm has the goal to balance requests across a
A cluster of servers taking into consideration various parameters of each task and the resources (CPU, IO and Memory) available on each server such that the response time is minimum at the same time considering even task distribution. Here, pre-emptive task migrations are assumed to be negligible in magnitude and overhead and not considered. A typical turn-around time of a request on our system can be defined as the time difference of the user submitting the request and the request being fulfilled. The requirements are usually to live-stream video files stored on secondary storage on the server end.

Proposed Algorithm:

For each request made to the cluster:
1. Estimate or compute the expected amount of resources required to service the task (CPU/Memory or IO)
2. As in the expected case, if I/O requirement is the maximum of the three, do the following:
3. Choose the set of nodes with maximum unused I/O resources and capability, while fulfilling CPU and Memory requirements for the task.
4. Assign the task to the node which is expected to produce lowest turn-around time.
5. Else if the memory requirement of the task is the maximum of all three, do the following:
6. Choose the set of nodes with maximum unused memory resources and capability, while fulfilling CPU and I/O requirements for the task.
7. Assign the task to the node which is expected to produce lowest turn-around time.
8. Else if the CPU requirement of the task is the maximum of all three, do the following:
9. Choose the set of nodes with maximum unused CPU resources and capability, while fulfilling Memory and I/O requirements for the task.
10. Assign the task to the node which is expected to produce lowest turn-around time.
11. Update status of the particular node indicating it is busy on the particular task.

This proposed algorithm can be used to schedule tasks across multiple heterogeneous servers. The following three load status indices are introduced with respect to CPU, Memory or I/O intensive: To decide whether a CPU of a node is loaded, we refer to the waiting queue for tasks on the node. Memory loading of a node is defined as the total amount of memory allocated to all tasks currently running on the node. I/O load can be either implicit or explicit. Implicit I/O can arise from page faults and secondary memory accesses for the node. Explicit I/O is used to service tasks with streaming video. The pseudo code of our algorithm is shown above. We now describe our algorithm. When a set of independent tasks are submitted to the load manager of the cluster, our algorithm tries to balance the cluster load among its nodes by distributing each task to a node where its response time is minimised. The first two steps are responsible to decide to admit the task into the system and predict the CPU, Memory and I/O requirements it would need. All this computation is done centrally at the dispatcher node. Step 3 is also performed at the dispatcher. Step 3 decides whether the task is CPU, Memory or I/O intensive. This step produces a result which jumps to either:
(a) Step 3 -if the I/O requirement of the task is maximum.
(b) Step 5 - if the memory requirement of the task is maximum.
(c) Step 9 - if the CPU requirement of the task is maximum.
Furthermore, at each of the 3 conditions, two more sub-conditions need to be satisfied that is, the non-maximum parameters for the resource allocation still to satisfy the node availability. Also, when selecting a node, we choose the node whose response time is estimated to be minimum for the node. When a node is finally selected, the dispatcher dispatches the job to the node.

IV. PROOF OF CONCEPT

The video-on-demand system is a group of nodes. Here, we select a main node whose job is as follows. The main node also called as ‘head’ node in the cluster. This is responsible for load balancing and monitoring the resources available at the concerned node. The head node processes all tasks in First Come First Serve (FCFS) manner. The computing nodes in the cluster depend completely on the information available with the head node doing the job of deciding the allocation. Tasks that are to be executed in the cluster arrive at the head node. We assume that the rate of arrival is constant in nature. After allocation by the head node, the tasks are dispatched to the best suited nodes for execution. Each of these nodes have a queue themselves, which executes these tasks in parallel.

A. Performance of the System

The model that we have used for simulating results, has been verified by the graph in Figure 3, which has been generated by plotting time taken to execute a group of tasks by using the number of servers involved. As shown below, the time of execution time for the tasks decreases as we increase the number of servers. But it is noted
that a certain threshold value is reached after a while and the execution time becomes constant. This clearly shows a saturation value for the completion of tasks.

From the Figure 3, we can prove that for a given group of tasks, changing number of servers from up to twenty, we realize that our system reaches a saturation point i.e. the execution time doesn't improve any more. Thus, our model is successful.

B. Performance Comparison of IOCMLB

For simulation and comparison purposes, two more allocation policies, namely the FCFS (First Come First Serve) and the random allocation policy are considered. By this experiment, we aim to compare how various allocation schemes perform when we consider a constant arrival of tasks. To assist the observations, the standard deviations of the load balancing of the servers are shown in the graphs. The lower the standard deviation is, the better the policy is considered to be.

1. FCFS: In the First Come First Serve allocation policy, the tasks are assigned to nodes in a linear fashion in the order in which they arrive. The graph obtained is shown as follows:

2. Random Allocation: Random allocation policy simply randomly distributes tasks to any node irrespective of the order or arrival of tasks. This policy helps to balance load evenly and equally among all nodes. The graph obtained for such an allocation scheme is shown as follows:

3. CMIO LB Algorithm: Using the proposed CPU-Memory-I/O Load Balancing algorithm, the graph obtained is as such:
The comparative graph obtained in Fig.7 shows that the proposed CMOILB algorithm performs better under heavy as well as light server loads of video-on-demand clusters. Furthermore, as observed from the other graphs, the performance of other allocation schemes deteriorates with increase in load. It should be thus noted that the algorithm will have a certain threshold as to the number of tasks it can balance well before the computation overhead for the algorithm exceeds the benefits.

V. CONCLUSION

In any video-on-demand system, it is essential to reduce the average response time. At the same time, supporting a growing number of clients is equally critical. The study has focused primarily on the dynamic allocation schemes for load balancing in video-on-demand servers to achieve these goals. Cluster computing has become an essential component due to development of technology and availability of cheaper, high performance processors and high end, reliable networks. This along with the development of tools for high performance distributed computing has supported the growing need of computational power. There are many different schemes available for load balancing on video-on-demand system, however, there overall efficiency depends upon the topology of the communication network between these nodes. The research has proposed an algorithm for I/O intensive tasks. It uses a method to calculate the load at the individual nodes, thus helping in load balancing. The algorithm strives to achieve the goal of effective global disk resources usage, especially in the VOD cluster. This can help to minimize the slowing down of parallel tasks running on the cluster and hence, reduces the average response time. We have compared our algorithm to some other common strategies, namely FCFS and Random. It is thus seen that we get better overall average results using our algorithm.

REFERENCES

[7] Paul Werstein ,Hailing Situ and Zhiyi Huang, Load balancing in cluster computer, Proceeding of the seventh international conference on Parallel and Distributed Computing, Applications and Technology (PDCAT’06).