Load Balancing in Fault Tolerant Video Server

D. N. Sujatha*, Girish K*, Rashmi B*, Venugopal K. R*, L. M. Patnaik**

*Department of Computer Science and Engineering
University Visvesvaraya College of Engineering Bangalore University
Bangalore-560001, India.
**Microprocessor Applications Laboratory, Indian Institute of Science
Bangalore-560012, India.
suj_sat@yahoo.com

Abstract. The Video-on-Demand (VoD) application is popular as the videos are delivered to the users at anytime and anywhere. The system load is increased due to simultaneous access of VoD system by many users. The proposed architecture discusses load balancing mechanism within a video server and to provide service to the users with small start-up delay. The Video storage is based on the probability of the users requesting for the video. Videos with higher request probability are stored and replicated to ensure guaranteed retrieval. Parity generation scheme is employed to provide reliability to non-popular videos. The system is also capable of handling disk failures transparently and thereby providing a reliable service to the user.

Keywords: Data Striping, Fault Tolerance, Load Balancing, Video Server.

1 INTRODUCTION

Video-on-Demand (VoD) is a real time multimedia application in which videos are stored in the video server, assisting in delivery of video requested at any time under the discretion of the user. Implementation of VoD is not as widespread as Internet with its biggest hindrance being the lack of network infrastructure that can handle large amount of multimedia data. There is need for design mechanisms that can determine how best to exposit upon existing infrastructure in order to support the additional demand inflicted by VoD systems. These mechanisms should strive towards generating scalable architecture. Load Balancing technology addresses these issues making the system scalable and better performing. *Load balancing* is a technique to spread work between many disks in order to get optimal resource utilization and decrease computing time thereby providing faster service to the users. The challenge for researchers is to design mechanisms to support more users and to reduce the average response time. The load balancing among the servers can be achieved if a load balancing mechanism is triggered to perform file migration and request migration when load unbalance is detected. To perform these activities an external hardware device, load balancer is used. A load balancer is capable of increasing the capacity of a

server farm beyond that of a single server. It also allows the service to continue even during server down time due to failure or maintenance. However there are limitations to this approach. Load balancers can become points of failures and performance bottlenecks themselves. Hence, load balancing is performed within the video server rather than maintaining external devices. As the number of servers grows, the risk of a failure at any point in the system increases and must be addressed. The load balancers are capable of balancing the load as well as detecting the failures. Multiple disks in video server significantly improve storage capacity and I/O bandwidth. However, the use of multiple devices increases the probability of failure. In a VoD system, without fault tolerance, a failure would result in a disruption of service to all users. A VoD system should provide a high degree of fault-tolerance. If a disk in the server fails, load balancing automatically redistributes requests to other disks within the disk array or forward the request to other servers in the cluster, hence ensuring continuous service to the users. Server load balancing addresses several issues such as: increased scalability, high performance, high availability and disaster recovery.
Outline: The reminder of this paper is organized as follows. Section 2 presents previous works in the area of load balancing in VoD system. Load balancing Architecture for VoD System is discussed in Section 3. The Algorithm is discussed in Section 4. We elucidate the usefulness of our algorithm through simulation and performance analysis in Section 5 and conclusions in Section 6.

2 PREVIOUS WORKS

Load balancing can be performed using external devices in VoD servers. Niyato et al. [1] describes external load balancing performed in Internet video and audio server. Load balancing in a video server can be carried out with the help of disk arrays in the server. Various techniques are employed to efficiently distribute the data among these disks to ensure that the users are served faster with minimum access times. One of such techniques employed is RAID (Random Array of Inexpensive Disks). There are many levels of RAID existing to store data in reliable fashion. RAID-0 (disk striping) and RAID-1 (disk mirroring) and RAID-5 (disk striping with parity) are few of the popular RAID levels. Lee et al. [2] employs the disk striping with parity in video servers to provide load balancing along with fault tolerance. This scheme suffers with multi-disk failures. A comparison of different RAID levels evolved Random Duplicate Assignment (RDA) [3]. In this strategy the video is striped and instead of being stored sequentially they are randomly allocated in different disks and each strip is mirrored to enable fault tolerance. This scheme enables the video server to balance the load among its disks and also achieve reliability. The time required to access next block in the disk increase as the blocks are randomly allocated. A map between the blocks is to be maintained to handle this problem, which is an additional overhead. Replicating all the videos is considered in this scheme, which may not be feasible all the time. Golubchik et al. [4] discusses fundamental issues associated with providing fault tolerance in multi-disk VoD servers. Storage of videos in

a video server may be characterized by popularity of the video. Replication of videos and placement of video blocks based on popularity is discussed in [5], [6] and explores a data placement method based on rate staggering to store scalable video data in a disk-array based video server.

3 SYSTEM ARCHITECTURE

The storage scheme designed, evaluates the popularity of the video before it is stored. The video storage differs with popularity. If the video is popular, then it is striped across array of disks and mirrored else it is striped across the disks with the last disk in the set of disks to store the parity information. This helps to rebuild the video block in case of disk failure. The overview of the architecture is shown in Fig. 1.

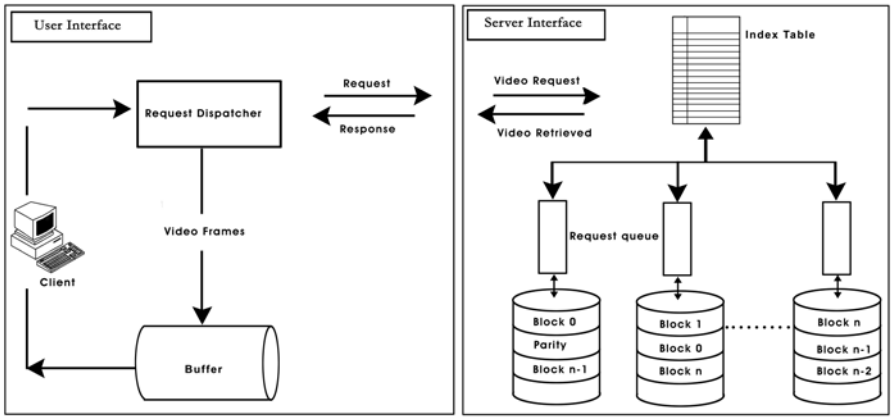

Fig. 1. Overview of the Architecture

- *Request Dispatcher* is a controller that is required at the user side to take the user's command and send the signal to the server through its network interface. The request dispatcher also stores video signals it receives from the server into its buffer, decodes the compressed signals and sends the decoded signals to the display at the appropriate time.
- *Buffer* is the video memory that holds the video image to be displayed on the screen of the user. The amount of memory required to hold the image depends primarily on the resolution of the image and also the color depth used per pixel. Once the server locates the video and moves the video frame by frame to the user, the frames are first stored temporarily in the user's buffer before it is displayed to the user. The advantage of having a buffer in the user's memory is to reduce the jitter and to provide high quality video transmission to the users.

- *Index Table* maintains the list of unique id of all the videos stored in the current video server. When a new video is stored, index table is updated. When a request arrives for the video, index table is searched to find if the requested video is available and if available it retrieves the video from disk and begins streaming.
- *Disk Array* is used to store videos. Disk array refers to a linked group of one or more physical independent hard disk drives generally used to replace larger, single disk drive systems. Disk arrays incorporate controls and a structure that pre-empts disaster. The most common disk array technology is RAID. RAID utilizes disk arrays in a number of optional configurations that benefit the user. Disk arrays organize their data storage into Logical Units (LUs), which appear as linear block places to their users. A small disk array, with a few disks, might support up to 8 LUs; a large one, with hundreds of disk drives, can support thousands.

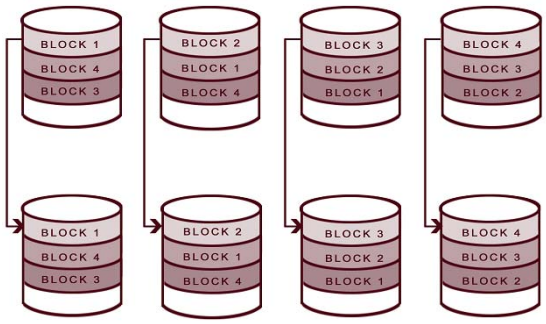

Fig. 2. Storage Mechanism for Popular Videos.

The storage mechanism for popular videos is shown in Fig. 2. The video is divided into blocks based on number of disks. Each block is stored in different disks sequentially so that only once a block of video stored on the disk. The first disk to store the video is rotated to ensure that the load is balanced among the disks. If the block requested is stored in a disk, which is serving another user, the request is queued. The requests in the request queue are serviced in round robin fashion. If the video is a popular then, the video is stripped across the array of the disks and mirrored. In case of increase of load for the popular video the request may be served by the mirrored storage. The mirrored data is also accessed in case of disk failure containing the popular videos. The storage mechanism for non-popular videos is shown in Fig. 3. If the video is non-popular video, it is striped across the disks with the last disk in the set of disks to store

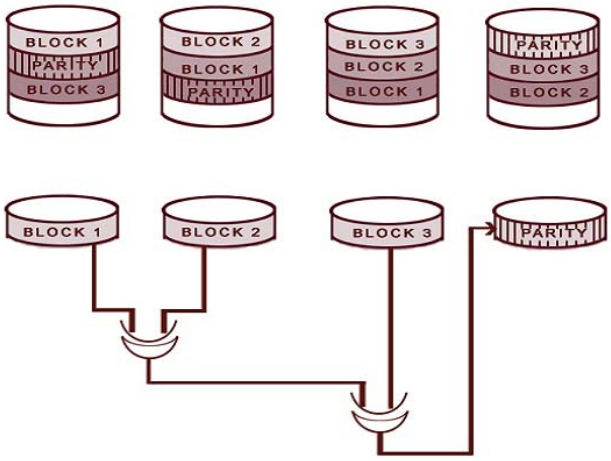

Fig. 3. Storage Mechanism for Non-Popular Videos.

the parity information. Performing XOR operation between all the blocks of data generates parity block. This helps to rebuild the video block in case of disk failure.

4 ALGORITHM

The storage routine (Refer Table 1) shows how the videos are stored in the server to facilitate load balancing and provide fault tolerance feature. The video is checked for popularity using Zipf distribution. If the video is popular the block size is determined by total video size by *N* (number of disks). If the video is non-popular, its block size is determined by video size by number *N-1* (disks). Here, one disk is used to store the parity information. Rotational storage scheme is used to store the video blocks. The first block of the first video is stored sequentially from first disk. The first block of next video is stored from second disk. This enables load distribution when simultaneous requests for various videos are made. The blocks thus stored are either replicated in mirrored storage area for popular videos or parity block is generated for non-popular videos. The index table is then updated to store video information. It holds information regarding video file-id, the size of the video and the disk in which the first block of the video is stored.

The HandleRequest routine (Refer Table 2) is designed to illustrate the behavior of the server on arrival of the request. When the server receives a request for a video with id, the server checks the video-id against the video-id of the

Table 1. Video Storage

```
Video_Storage (VideoId)
begin
determine the popularity of the video.
   if (video is popular)
     BlockSize = VideoSize / n
   else
     BlockSize = VideoSize / n-1
   i=0
  j = 0
  if ( space available to store video at BlockSize on disk j)
    begin
      until ( VideoSize≥ 0)
      begin
        j = (( VideoId % n )+ i)% n
        reduce VideoSize by BlockSize.
        store block on diskj
        increment i by 1
      end
    end
  Update_Index_Table (VideoId)
  if (video is not popular)
    P_j = B_0 ⊕ B_1
     while(i < n)
       p_j = p_j ⊕ B_i
     Store P_j in disk j
   else
     mirror the blocks in mirror storage.
end.
```

Table 2. Video Retrieval

```
Handle_Request (Video_id)
begin
  found = search index table for VideoId
  if (found)
    retrieve video_info from index table.
  until (EOF)
  begin
    if (block not corrupted)
    begin
      if (disk not loaded)
        begin
          stream block i from disk j
          j = j+1
        end
      else if (video is popular)
        handle request from replica disk
      else
        Forward_Request (Video_Id)
      end
  else if (video is not popular)
     rebuild block from parity block
  else
      stream from replica.
  end
end.
```

available list of videos in the index table. If the requested video-id is found, it retrieves the information indicating to the disk in which the first block of video is present. Before the video frames are retrieved, it is necessary to check whether the block is not corrupted and is retrievable. If the block of popular videos is corrupted, the request is handled by the mirrored disks. If the block of non-popular video is corrupted, the block is retrieved with the help of special parity block available. The block of video is streamed from mirrored storage, if request queue of the disk is full. If no mirrored data is available, then the request is forwarded to other server in server farm.

5 SIMULATION AND PERFORMANCE ANALYSIS

Fig. 4 illustrates the start-up delay of the users to begin downloading the video after the request is made. The average delay increases with the increase in load, as the users are queued if the disk is busy serving other users. The delay increases by 17% with increase in load. The start up delay can be further decreased with increase in number of disks in the system which is evident from the graph. Fig. 4 also depicts the decrease in start up delay with increase in number of disks.

Bandwidth utilization for varying load is ascertained in Fig. 5. The increase in bandwidth decreases download time of the entire video. Variations of load increase the time to download by 2% to 4%. Increase in bandwidth reduces the

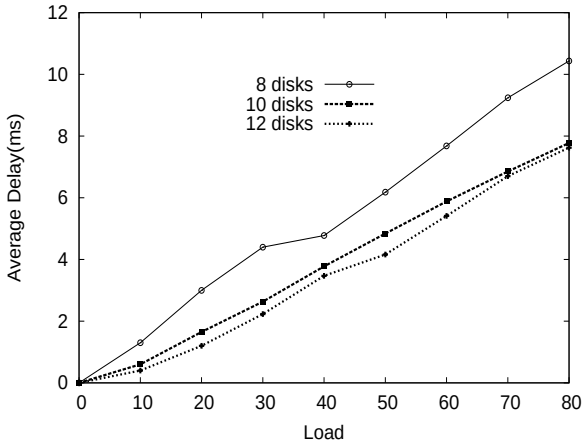

Fig. 4. Start-up delay.

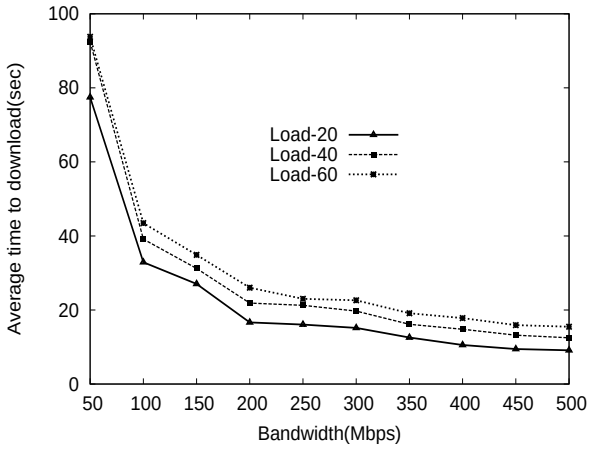

Fig. 5. Bandwidth Utilization for Varying Load..

time to download by 75% reaching saturation at higher bandwidth rates.

The performance of system with disk failure is shown in Fig. 6. Delay is ini-

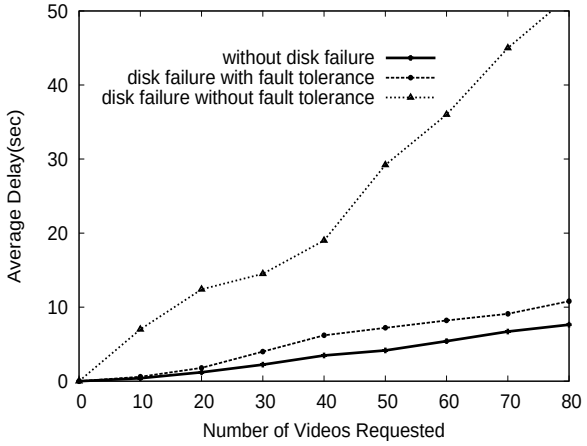

Fig. 6. Performance of System with Disk Failure.

tially the same with disk failure and a normal system. It increases gradually with the increases in load by 2 to 2.5% and the difference remains constant. A drastic increase can be noted if the fault tolerance mechanism was not implemented. Popular videos have a higher tolerance to load than non-popular videos. As the load increases the percentage of rejection increases in non-popular videos by 25%.

Fig. 7 illustrates the disk failure of the popular videos against the non-popular videos. The popular videos demonstrates lower rejections thereby increasing the throughput of the system.

6 CONCLUSIONS

An efficient architecture has been proposed to balance the load in the video server. Storage mechanism to distribute the load in an array of disks are considered. The video in divided to blocks and these blocks are stored on different disks in disk array along with disk rotation. The video server is ensured to be fault tolerant by using both mirroring and parity schemes. The selection of fault tolerance scheme is based on popularity of the video using mirroring and thus providing back-up to handle load imbalance and disk failure situations. In case of a non-popular video the probability of failure is less. Nevertheless to provide reliable service, in case of non-popular video single disk failure is handled using parity blocks. The simulation and resultant graphs depict a high system throughput of 100% up to 30 simultaneous requests. There is a gradual increase in the

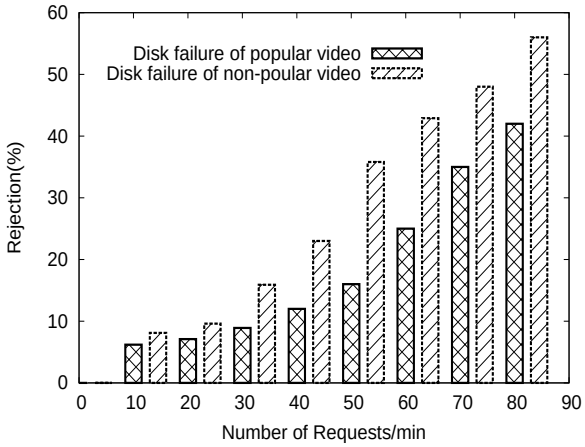

Fig. 7. Illustration of Disk Failure with Popular and Non-popular Videos.

rejection rate beyond 30 requests. The system also has a very low latency of 2ms-10ms for a varying load up to 80 requests. The increase in number of disks further decreases the latency. The system provides fault tolerance against disk failures with a very low performance degradation. Hence the proposed architecture distributes the load in the video server efficiently to handle simultaneous requests and provides high degree of fault tolerance there by enabling efficient and effective service to the users requesting the video.

References

1. Dusit Niyato., Chutim Srinilta.: Load Balancing Algorithms for Internet Video and Audio Server, 9th IEEE Intl. Conference on Networks, 2001, pp. 76-80.
2. Y.B. Lee., P.C. Wong.: VIOLA-A Scalable and Fault-Tolerant Video-on-Demand System, Hong Kong Intl. Computer Conference, 1997, pp. 1-5.
3. Yung Ryn Choe., Vijay S. Pai.: Achieving Reliable Parallel Performance in a VoD Storage Server Using Randomization and Replication, Intl. Parallel and Distributed Processing Symposium, 2007, pp. 1-10.
4. Leana Golubchik., Richard R. Muntz., Cheng-Fu Chou., Steven Berso.: Design of Fault-Tolerant Large-Scale VOD Servers: With Emphasis on High-Performance and Low-Cost, IEEE Transactions on Parallel and Distributed Systems, vol. 12, no. 4, 2001, pp. 97-109.
5. Xiaobo Zhou., Cheng-Zhong Xu.: Optimal Video Replication and Placement on a Cluster of Video-on-Demand Servers, Intl. Conference on Parallel Processing, 2002, pp. 547- 555.
6. Xin-Mao Huang., Cheng-Ru Lin., Ming-Syan Chen.: Design and Performance Study of Rate Staggering Storage for Scalable Video in a Disk-Array-Based Video Server, Intl. workshop on Network and Operating Systems support for Digital Audio and Video, 2005, pp. 177- 182.