A Fast Start-up and Scalable Algorithm for Continuous Media Servers

P Jayarekha[1] , A S Manjunatha[2]
[1]Research Scholar Dr. MGR University, Department of ISE BMSCE, Bangalore
[2]CEO & MD Manvish eTech Pvt Ltd

.

Summary

In this paper, we propose a novel VOD server architecture and a dynamic prefetch buffer allocation algorithm based on popularity. This can drastically improve the maximum number of streams a drive can support, with a small startup latencies. The primary design goal is to guarantee glitch free operation for maximum number of users. Detailed simulation is carried out and compared with the performance parameters of the static allocation algorithm.

Keywords:
Prefetch cache, Start-up latency, concurrent users, Dynamic buffer allocation

Introduction:

Storage servers for digital movie retrieval are the central components of many multimedia systems. The emerging technologies in data storage have great impact on system design and issues related to that. Research efforts are directed towards the development of "smart disks", which exploit the additional resources like prefetch buffer so as to improve the overall system performance. Dynamic prefetching strategies for continuous media into disk and host cache which is the theme in this paper, holds the promise of drastically improving disk performance for continuous media applications.

VOD system provides video data to users upon user requests. There are two important characteristics of video data. First, the amount of video data is voluminous second, video data must be continuously provided to the user. The former requires that VOD systems use buffers for managing data by block units because system cannot store the entire video data in memory. The latter mandates buffer management of VOD system to retrieve new data blocks into buffers before a user request uses up the data in the buffer.

Several buffer scheduling methods for VOD system have been proposed that minimize memory requirements and initial latency [1,2].The static buffer allocation scheme determines the minimum buffer size based on the assumption that the system consistently allocates the buffer size to all user requests regardless of the systems load, VOD system must allocate larger buffers to user requests, as the number of user requests in service increases. Thus, the static buffer allocation scheme has a disadvantage as it uses memory inefficiently by allocating a larger buffer than necessary. Hence the static scheme increases memory requirements and initial latency of system [1,3].

Both buffering and prefetch caching concern to avoid delay and overhead in accessing secondary storage. Anticipating the video needed by a client may already present in the disk buffer or prefetched from the secondary storage. The videos are prefetched from the secondary storage on behalf of the currently consuming data stream and for future request even after they are delivered to the current stream.

This paper proposes a dynamic buffer allocation scheme that dynamically allocates the minimum buffer size. The number and size of the buffer is dependent on the number of requests that is the popularity of that request. The advantage of this scheme is as follows First this scheme removes the static buffer allocation schemes problem of allocating unnecessarily large buffers. Second, by allocating the minimum buffer size, our scheme significantly improves the average initial latency and the average number of concurrent user requests that can be supported. Thus, this scheme uses a predictive prefetching policy that confidently improves the performance. Predictive prefetching refers to the mechanism of deducing the forthcoming page access of the client based on the popularity. In this paper we have proposed a new context for prefetching the pages based on dynamic buffer allocation.

In [2], stavrous harizopoulos has proposed an algorithm that work in parallel to retrieve or prefetch data from disk surface to the disk or the SCSI controller caches, and transfer the data from the lower cache to the host cache, concurrently. At the same time, the host streams the data from its RAM through the network to the clients and this parallelism has significantly affected the performance of a media server.

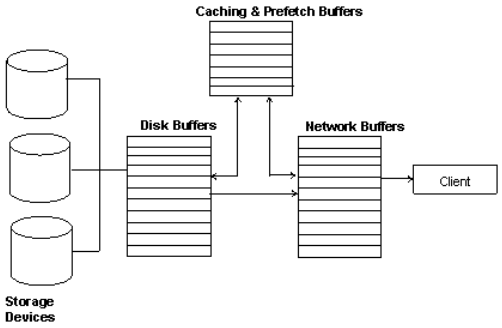
In [4], huanxu pan has proposed a dynamic scheduling policy based on buffer inventory, that guarantees non-

Manuscript received November 5, 2008
Manuscript revised November 20, 2008

zero inventory and non-overflow of data in the buffers to meet the continually requirement and no-loss of data for each media stream. It has proved that a static scheduling policy cannot easily deal with transient situations such as arrival of new request and server completions of existing requests, since such situations will break the strict balance between the retrieval and playback rates that a static scheduling policy is trying to maintain.

2.Proposed architecture and algorithm

We consider a storage multimedia system as shown in the fig 1. In this system when a request for a particular media (video or audio) stream is received a server retrieves from a disk the requested stream and places it into a disk buffer.



Storage system of a Multimedia server

Fig 1

Retrieving data from the disk involves data search, and the time needed for the search is not certain but depends directly on the physical locations of the data on the disk.

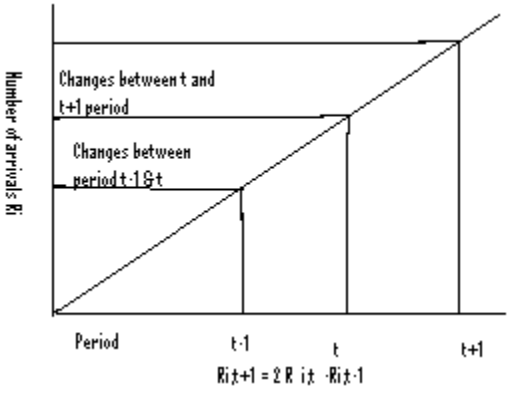
Disk generally have a rated bandwidth, which represents the maximum rate which data can be transferred from the disk. The effective bandwidth of the disk will generally be less than the disk's rated bandwidth due to various overheads. First the seek time, or the time required for the disk head to move to the desired track, second the rotational delay ,the time required for the desired sector to come below the read head. Start up latency is the sum of seek time and rotational delay.

A critical development has been the incorporation of drive-level caches into the disk drives, which accompanied with caching and prefetching techniques, can significantly increase the drive's performance. A read that hits in the cache can be satisfied much quicker than seeking to the data and reading it off the disk.

The primary performance metric is the maximum through put, achievable by a drive, which is defined to be the maximum number of continuous streams that can be supported by the drive and the considerable reduction in the start up latency.

This paper proposes Popularity-based placement policy to place the video on the prefetch cache.This attempts to minimize the average number of request rejected due to storage device overload under high load conditions. The prediction on the arrival request is made depending on the access probabilities. Depending on the prediction, the videos are prefetched and placed on the prefetch cache The prediction probabilities $p_{i,t}$ are fixed over fixed period and and the system is reconfigured fixed period to reflect new access probabilities. For predicting $p_{i,t+1}$ from the historical data on period t-1 and t is used. If $R_{i,t}$ is the number of requests for multimedia object i on that period t ,a reasonable estimate can be done for $R_{i,t+1}$ by linear interpolation: $R_{i,t+1} = 2 R_{i,t} - R_{i,t-1}$.

The value of $p_{i,t+1}$ can be computed as $p_{i,t+1} = R_{i,t+1}/\sum_i R_{i,t+1}$.The values of $p_{i,t+1}$ can be used at the end of a period to reconfigure the videos of the prefetch cache.



Prediction of future demand using popularity based assignment

Fig 2

For a given duration of time, there is a specific upper bound in the number of streams N that can be supported by a disk in a feasible media server. The maximum number N of concurrent user requests that can be supported is determined by the video data consumption rate CR and disk transfer rate TR.

If TR=NXCR however, a disk cannot guarantee the time-wise continuity because disk latency occurs whenever the disk service a user request and hence N < TR/CR.

Initially when a new request R arrives for any video stream, a check is made whether that request is being already being streamed from the diskbuffer.If yes continue streaming by updating the respective finish of the request R.Otherwise, find whether that video is present in the prefetchcache.If present, start streaming from the prefetchcache by dynamically allocating the buffer space in the diskbuffer depending on its popularity.If no free space is currently available in the diskbuffer search for the offline videos with the least popularity and dynamically replace it with the new request R.Otherwise the request waits in a queue.

If request R is neither present in the disk buffer nor in the prefetchcache, start streaming from hard disk to

diskbuffer.This is associated with the overhead of initial latency.
At the end of each unit of time refreshment is done for the prefetch cache considering the Popularity-based assignment.

Algorithm:

Algorithm Dynamic-Prefetch-Allocation
[Nomenclature
N=TR/CR
N : the maximum number of concurrent users at any instant of time.
TR : disk data transfer rate
CR : video data's consumption rate
T : total simulation time.
$P_{i,t}$: Access probabilities to each request R_i on for a period t.
v : number of videos streaming from prefetch-cache to diskbuffer.
d : number of videos streaming from hard disk to diskbuffer.
n : total number of videos streaming from both prefetch cache and harddisk to diskbuffer.]

```
While ( t < T)
  {
   For each new request Ri, or request Ri waiting
     in Request-queue
     If (Check still-streaming-from-diskbuffer)
        Update the finish_time of the request Ri.
     Else
      n=n+1
          If (n> N) reject the request Ri
         Else
           If (Found in Prefetch-cache)
                If(diskbuffer is not full and
                 dynamic-buffer-allocation is
possible)
                    v= v+ 1
                   Req-Q = Req-Q + {Ri}
                Else
                 If(offline-videos are found )
                 replace R dynamically with lpv
                 v= v+1
                 Req-Q = Req-Q + {Ri}
               Else
                  load requested R from hard disk
                    replacing R with the lpv
                Compute startup latency
                 d = d + 1

          n = v + d
           For each request R in the Request-
Queue
           If Ri is offline
                n = n -1
                Req-Q= Req-Q– {Ri}
                 t = t + t1
    For each unit of time
    Reconfigure the prefetch-cache using the probabilities
    p_{i,t+1} = R_{i,t+1}/∑_i R_{i,t+1}
    }end while
 For each request waiting in Request-Queue
          Reject the service
```

In our simulation model ,we use a video library that holds the videos that are 10 to 60 minutes long. We order the videos according to popularity and they follow a Zipfian distribution. Our trace generator considers newly arrived videos with a Poisson distribution. We have assumed a single disk for our simulation with a disk capacity of 10GBytes,disk block size of 32 Kbytes and maximum seek time is 25 msec ,maximum rotational latency 15msec.Average transfer rate is 15 Mbytes/sec.Hard disk buffer of size 4 Bytes ,with a average transfer rate of 5 Mbytes/sec. Prefetchcache size as 1GBytes. The average transfer rate of prefetchcache is assumed to be larger than diskbuffer.

4.Results and Discussion.

The results presented below are an average of several simulations. The values considered for simulation are as follows.
Fig 3 compares the number of concurrent users that can serviced in static and dynamic allocation. It is proved from the experimental results that dynamic prefetch services considerably more number of concurrent users. Fig 4 demonstrates the reduction in the initial start up latency time for the dynamic prefetch allocation. Fig 5 Illustrates the increase in the number of cache hits when prefetch cache is used.

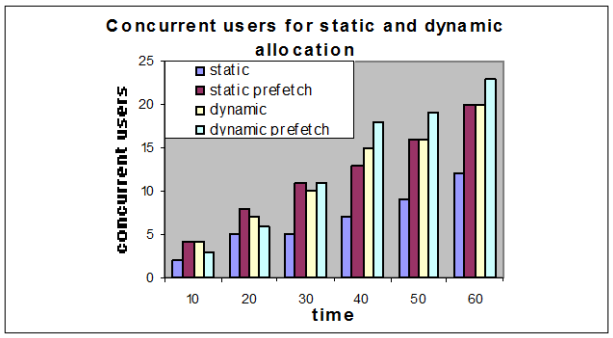


Fig 3

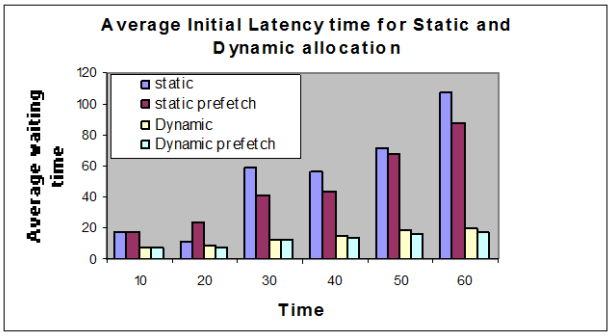


Fig 4

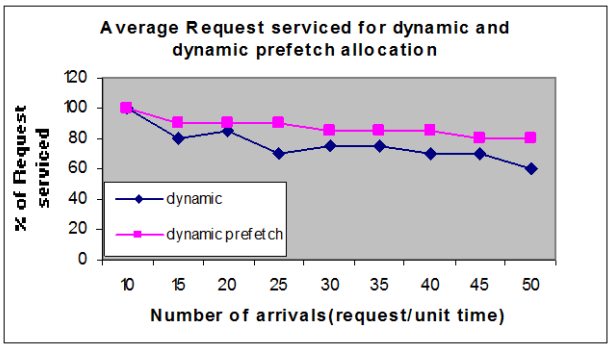


Fig 5

5.Conclusion

In this paper we have concentrated on the prefetch and streaming the video in the cache buffer depending on the popularity .Further to support the VCR control operations such as fast-forward , fast-backward, skip and so on can be implemented predicting the access pattern .Concurrent access to the same request at different interval of time for the popular video can be implemented using Interval-caching and prefetching.

References

[1] Kyung Oh Lee ,Jeong Bae Lee,Kee wook Rim "A Dynamic scheduling Algorithm for Video-On demand Servers" *IEEETransactions on consumer electronics* vol 50 no 4 2004

[2]StavrosHarizopoulos,costas,Peter Triantafillou"Hierarchical caching and prefetching for High Performance Continuous Media servers with Smart Disks"*IEEE concurrency* 2000.

[3]Sang-Ho Lee Kyu-Young Whang,Yang-Sae Moon "Dynamic Buffer Allocation in Video-On Demand systems" *ACM May 2001*.

[4]H S Guruprasad M Dakshayini P Jayarekha H D Maheshappa P Geethavani"Dynamic Buffer Allocation for VOD system Based On Popularity"PSG Tech Coimbatore 2006.

[5] Amit Cohen and Reuven Cohen" A Dynamic Approach for Efficient TCP Buffer Allocation" The 7'th International Conference on ComputerCommunications and Networks, Lafayette (Louisiana), October 1998

[6] Alexandros Nanopoulos, Dimitrios Katsaros, andYannis Manolopoulos, Member, IEEE Computer Society" A Data Mining Algorithm for Generalized Web Prefetching*"IEEE Transaction on data and Knowledge 2003*

[7] Huanxu Pan, Lek Heng Ngoh & Aurel A. Lazara Buffer-Inventory- Based Dynamic Scheduling Algorithm for Multimedia-On–Demand Servers" IEEE Multimedia Systems 96[8].

[8] Reza Rejaie Hanley Yu Proxy caching Mechanism Playback Streams in the Internet Information science Institute Jan 22 1999