PREDICTIVE FAULT TOLERANT PLACEMENT IN DISTRIBUTED VIDEO SERVERS

Xiaoping Wei and Nalini Venkatasubramanian

Department of Information and Computer Science
University of California, Irvine, CA 92697
{wei, nalini}@ics.uci.edu

Abstract: A high degree of data availability is required to ensure QoS in distributed video server environments. In this paper, we propose an algorithm for fault tolerant data placement in a distributed video server that determines the placement of video objects so as to ensure continuity of video requests in the event of node failures. Our performance studies indicate that object distribution strategies have a significant impact on the video placement process.

1. INTRODUCTION

With emerging interest in video-on-demand services for entertainment, distance learning etc., distributed video server architectures are required to provide scalable services. Requests for videos in a distributed video server usually have stringent resource requirements (bandwidth, memory etc.); thus effective resource management and load balancing is needed to optimize resource utilization and admit as many user requests as possible [12, 8, 2, 7, 4, 13, 15]. To ensure QoS requirements, it is necessary to provide fault tolerance so that when a data source fails, requests assigned to that data source can migrate to another data source and proceed to complete without the user being aware of the failure [1, 9, 10, 3, 6, 11].

In this paper, we present a predictive fault-tolerant placement (**PFTP**) algorithm that aims to provide high fault tolerance and efficient resource management in a distributed video server. We define a *Replication Degree(RD)* for each video object as the number of replicas of a video that must exist to enforce the desired level of availability. The **PFTP** algorithm enforces the replication degree by predictively making replicas of each video and placing them on different data sources. By ensuring that multiple replicas exist at any time, the system is able to deliver the desired degree of availability and fault tolerance. Meanwhile, since we create replicas of videos based on their access patterns (predictive placement) and schedule a user request based on existing system load (dynamic request scheduling), effective resource utilization is achieved.

We assume the following architecture for a typical distributed video server. A distributed video server has a number of independent and distributed data sources, each consisting of processors, memory, high speed network interface and high storage capacity (large hard disk etc.). The videos are stored on those data sources and accessed by users. A distributed video server has a central distribution controller, which controls the admission and scheduling of new user requests. The rest of the paper is organized as follows. Section 2 describes the **PFTP** algorithm. Section 3 explains different object distribution strategies that restrict where a replica can be placed when enforcing the replication degree for a video. Section 4 presents the performance evaluation of the **PFTP** algorithm. We conclude in section 5 with future research work.

2. PREDICTIVE FAULT TOLERANT PLACEMENT

In this section we present the predictive fault tolerant placement (**PFTP**) algorithm for a distributed video server that predictively makes replicas of videos on different data sources and enforces the replication degree for each video. The goals of the **PFTP** algorithm are to (a) ensure performance in terms of request acceptance; (b) provide fault tolerance and data availability. The **PFTP** algorithm consists of two phases. In the first phase, we make the initial video placement based on user access patterns using a greedy matrix based algorithm similar to the one in [12]. In the second phase, we enforce the RD for those videos whose RDs are not yet satisfied. In this process, assignments of videos to data sources that are obtained after the first phase might be revised in the second phase.

2.1. Phase 1: Initial video placement

The objective of this phase is to determine a tentative assignment of videos to data sources based on the history of past video accesses. We use a placement matrix (Table 1); here $v_i (i = 1, 2, 3)$ are different video objects; $S_i (i = 1, 2)$ are data sources which have the necessary computing resources such as high disk storage capacity, processor, memory and network connection; r_i is the revenue associated with v_i (i.e. the cost for a user to watch video v_i); R_i is a request for v_i; N_i is the expected number of requests for video v_i in the next period. $LF(R_i, S_j)$ represents the number of requests for v_i that can be scheduled to S_j given the current available resources on S_j. $PM(v_a, S_b)$ is the entry (or cell) in the placement matrix that represents the maximum revenue generated by scheduling requests for video v_a to data server S_b. In each iteration, we pick the maximum revenue generating assignment in the placement matrix. In other words, we make a tentative assignment of v_i to S_j if and only if $PM(v_i, S_j) = \forall a \forall b max(PM(v_a, S_b))$. The placement matrix is updated as follows: if all the expected requests for v_i in the next period could be accommodated on the video server S_j, the row corresponding to v_i will be eliminated, otherwise the column corresponding to S_j will be eliminated. The first phase will terminate when the placement matrix has either no rows or columns left.

2.2. Phase 2: Global replication degree enforcement (GRDE)

In phase 2, we try to enforce the replication degree (RD) for those videos whose RDs have not been satisfied using a global RD enforcement procedure. After phase 1, four different types of video objects might exist:

- **Unplaced video** V^{up} - videos that have no replica created,
- **Partially placed video** V^{pp} - videos that have at least one replica created, but have less replicas than that required by the RD.
- **Fully placed video** V^{fp} videos that have the exactly same number of replicas created as that required by the RD.

2001 IEEE International Conference on Multimedia and Expo
ISBN 0-7695-1198-8/01 $17.00 © 2001 IEEE
0-7695-1198-8/01/$10.00 (C) 2001 IEEE

	S_i	S_2
v_1	$min(N_1, LF(R_1, S_1)) \times r_1$	$min(N_1, LF(R_1, S_2)) \times r_1$
v_2	$min(N_2, LF(R_2, S_1)) \times r_2$	$min(N_2, LF(R_2, S_2)) \times r_2$
v_3	$min(N_3, LF(R_3, S_1)) \times r_3$	$min(N_3, LF(R_3, S_2)) \times r_3$
	$max(PM(v_i, S_1))$	$max(PM(v_i, S_2))$

Table 1: Placement matrix for request

1. *Sort the overplaced videos in ascending order of the expected number of requests into a set V^{op}.*
2. *Replication degree enforcement for partially placed videos.*
   (a) *Sort the partially placed video in descending order of the expected number of requests into a set V^{pp},*
   (b) *$\forall v_i \in V^{pp}$ from the most popular to the least popular {*
   find appropriate data sources and create replicas of v_i on those data sources if possible; else
   Replace replicas of some overplaced videos by replicas of v_i if possible; else
   Replace replicas of some partially placed videos by replicas of v_i
   }
3. *Replication degree enforcement for unplaced videos.*
   (a) *Sort the unplaced video objects in descending order of the expected request numbers into a set V^{up},*
   (b) *$\forall v_i \in V^{up}$ from the most popular to the least popular {*
   find appropriate data sources and create replicas of v_i on those data sources if possible; else
   Replace replicas of some partially placed videos by replicas of v_i if possible; else
   Replace replicas of some overplaced videos by replicas of v_i.
   }

Figure 1: Global Replication Degree Enforcement

• **Overplaced video** V^{op} videos that have more replicas created than that required by the RD.

The **GRDE** procedure has three steps: (1) sort the over placed videos in ascending order of their expected number of requests in the next prediction period (i.e. their expected popularity); (2) try to enforce RDs for partially placed videos from the most popular to the least popular; (3) try to enforce RDs for unplaced videos from the most popular to the least popular. These steps are illustrated in Figure 1.

In the above algorithm, an *appropriate* data source for v_i is a data source that does not already have a replica of v_i and has enough free disk space to create a replica of v_i. If an *appropriate* data source cannot be found, a replica of video v_j can be replaced by a replica of video v_i if all of the following 3 conditions hold:

1. The data source of v_j does not already have a replica of the partially placed video object v_i

2. the size of v_j combined with the free disk space on its data source is larger than that of the partially placed video v_i ;

3. More benefit is incurred by replacing the replica v_j by a replica of v_i . A replacement is more beneficial if and only if the following formula holds: $(\frac{N_i}{N_j} \times rdp_i) \geq BF$. Here rdp_i is the *replication degree percentage* which determines the degree to which video v_i satisfied its RD; it is computed as $\frac{number\ of\ replicas\ of\ v_i}{replication\ degree}$. BF is the *bias factor* that signifies how much an overplaced video is favored over a partially placed or unplaced video. BF may also be chosen based on experimental results.

3. OBJECT DISTRIBUTION STRATEGIES

Various object distribution strategies offer different combinations of performance and fault tolerance abilities [6, 5] when used with the PFTP algorithm. We study the following strategies:

1. **Random:** Randomly select a data source with enough disk space to create a replica of a video.
2. **Chained Declustering:** Logically order all data sources and divide them into disjoint clusters of size C. All replicas of a video are restricted to be within one cluster.
3. **Overlapping Clustering:** Logically order all data sources and divide them into clusters of size C, one cluster overlaps with the next cluster by $\frac{C}{2}$ data sources. For each video v_i, there must exist a cluster containing all replicas of v_i.

Random object distribution offers the highest flexibility for video object distribution, but has limited predictability for fault tolerance because the maximum number of data source failures is bounded by the replication degree. The chained declustering strategy has less flexibility for video object distribution since it explicitly restricts where a video object should be placed. However, chained declustering offers better predictability for fault tolerance because it can tolerate multiple data source failures, as long as the number of failed data sources within one cluster is smaller than the replication degree. Overlapping clustering provides more flexibility for video object distribution than chained declustering since an object can belong to multiple clusters. However, with overlapping clustering, the predictability of fault tolerance is lower than with chained declustering since failure of a data source affects multiple clusters. In our implementation, we have enhanced the 2 phases of the PFTP algorithm to suitably accommodate the three object distribution strategies.

4. SIMULATION AND PERFORMANCE ANALYSIS

One goal of our simulation is to evaluate and compare performance of request load balancing and fault tolerance capability of the three object distribution strategies. The other goal of the simulation is to explore the performance impact that various parameters, e.g. replication degree, cluster size etc. have over request load balancing and fault tolerance capability of the three object distribution strategies. For a more detailed performance study, see [14].

Simulation Model: The video server configuration used in the simulations has a varying number of data sources, each with 50 Gbits storage space and 310 MBits/second transfer bandwidth. A tertiary storage that serves as permanent repository of video objects and replication source has a maximum replication bandwidth of 155 Mbits/second. 50 MPEG-2 movies are stored on the video server and each movie has an average bandwidth requirement of 3 Mbps. Duration of each movie varies uniformly from 30 to 120 minutes. The popularity of movies is assumed to follow the Zipf's law such that the request arrivals per day for each movie m_i is given by: Pr.(movie m_i is requested) = $\frac{K_M}{i}$ where $K_M = (\Sigma_i^M \frac{1}{i})^{-1}$. The simulation also assumes the hourly request arrival rate of movies follows a Zipf-like distribution [12]. An adaptive scheduling mechanism is used to schedule user requests; a request for video v_i is scheduled to a feasible data source S_j for which $LF(v_i, S_j)$ is the largest.

2001 IEEE International Conference on Multimedia and Expo
ISBN 0-7695-1198-8/01 $17.00 © 2001 IEEE
0-7695-1198-8/01/$10.00 (C) 2001 IEEE

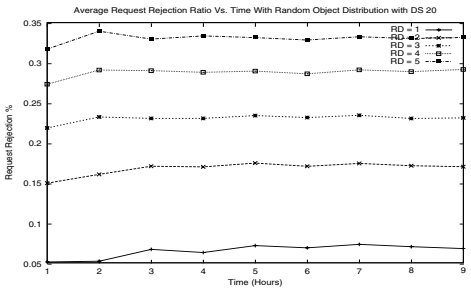

Figure 2: Average Request Rejection Ratio Under Normal Operation using Random Object Distribution

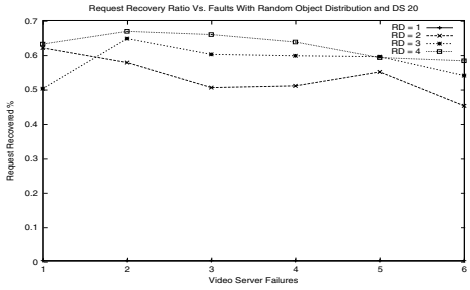

Figure 3: Request Recovery Ratio VS. Faults with Random Object Distribution

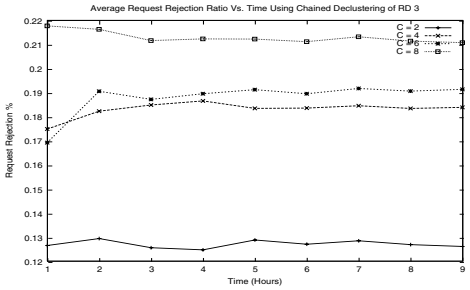

Figure 4: Request Rejection Ratio Vs. Time Under Normal Operation with Chained Declustering Object Distribution

In this simulation we only consider data source crash failures. Furthermore several data source failures could happen simultaneously. We divide the simulation time into fixed length intervals and assume that in each interval the same number of data sources will fail. We also assume that any failed data sources have recovered before the next failure interval. A memoryless recovery is assumed, i.e. no video exists after a data source recovers. When a data source fails, replicas of all videos on that data source will be inaccessible. The algorithm attempts to recover the requests scheduled to those replicas by migrating those requests to other normal data sources, again using adaptive scheduling. We set the factor BF to be 0.5. In other words, we twicely favor a over placed video object over a partially placed one. We use 2 performance measurement metrics: ***request rejection ratio*** and ***request recovery ratio***. The request rejection ratio signifies request load balancing; it is computed as the number of rejected requests divided by total number of requests. The request recovery ratio shows the fault tolerance capability; it is computed as the number of successfully recovered requests in case of data source failure, divided by the total number of failed requests.

Random Object Distribution: Figure 2 depicts the request rejection ratio over time with respect to changing replication degree under normal operation. It is seen that the request rejection ratio increases as the replication degree gets larger. This is due to the fact that the larger the replication degree, there are more replicas of fewer video objects stored on the video server in a prediction period. Therefore more requests are rejected. Similarly, the request rejection ratio increases with replication degree in case of faulty operation.

Figure 3 shows the request recovery ratio in case of data source failures. When RD=1, no other replicas are available for requests on the failed data source to migrate to, thus the recovery ratio is consistently 0. The recovery ratio increases along with replication degree because with a larger replication degree, a request will have more data sources to migrate to, thus more potential to recover when a data source failure occurs.

Chained Declustering Object Distribution: When chained declustering object distribution is used, the request rejection and recovery patterns observed are similar to that of the random strategy; request rejection ratios increase with replication degree under both normal and faulty data source operations; request recovery ratios of RD $>$ 1 are significantly better than that of RD = 1, and request recovery ratios of RD $>$ 1 are close to each other.

Figure 4 shows the request rejection ratio versus cluster size. It is observed that request rejection ratio becomes higher as the cluster size gets larger. When cluster size is smaller, the distributed video server has more clusters and fewer videos will be assigned to the same cluster, thus more videos exist in a prediction period. On the other hand, when cluster size is larger, there are fewer videos and more replicas of each of those videos in a prediction period. Therefore, more requests will be rejected and rejection ratios are higher when cluster size is larger. This relationship also holds when faults occur.

Figure 5 shows the impact of cluster size on request recovery ratio. When cluster size is larger, it is more likely that in the cluster there exists another data source to which a request could migrate to, thus potentially more requests could successfully recover when a data source fault occurs. It is observed from the above graph that the request recovery ratios of cluster size larger than 2 are all better than that of cluster size 2(where the choice to select a data source to which a request could be migrated to is extremely limited). Meanwhile, when cluster sizes larger than 2 are used, the request recovery ratios with a larger cluster size are also generally better than that with a smaller cluster size, particulary when more data sources fail simultaneously.

Overlapping clustering Object Distribution: The simulation results for overlapping clustering project strong simularities to those of the chained declustering object distribution strategy. With the same replication degree, overlapping clustering exhibits marginally better load balancing performance than chained declustering in request load balancing, as shown in figure 6. On the other hand, with the same replication degree, request recovery ratio of overlapping clustering is slightly below that of chained declustering. Similar to chained declustering, the request recovery ratio goes down when cluster size increases, however, compared to chained declustering, the request recovery ratio degrades a little faster than chained declustering.

Performance Summary: Figures 6 compares the request load

2001 IEEE International Conference on Multimedia and Expo
ISBN 0-7695-1198-8/01 $17.00 © 2001 IEEE
0-7695-1198-8/01/$10.00 (C) 2001 IEEE

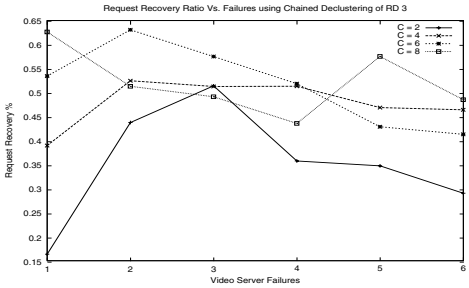

Figure 5: Request Recovery Ratio Vs. Faults with Chained Declustering Object Distribution (various Cluster Size)

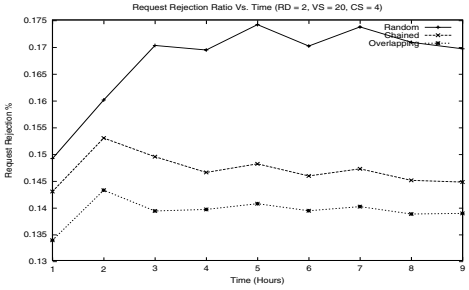

Figure 6: Request Rejection Ratio Vs. Time Under Normal Operation

balancing of different object distribution strategies Our results indicate that overlapping declustering exhibits the best performance on request load balancing measured in terms of request rejection ratio while chained declustering performs better than the random strategy. This is because with chained declustering or overlapping clustering strategy, we observe that with smaller cluster size fewer replicas of distinct video objects are created, however, the number of distinct video object are larger. As a result, more requests are admitted. Note that random object distribution is actually a specical case of either chained declustering or overlapping clustering where the cluster size is simply the total number of data sources in the distributed video server. Similar results were observed under faulty operation.

For all three object distribution strategies, request rejection rate increases along with RD. Fault tolerance capability generally improves when replication degree or cluster size increases. In particular, the fault tolerance capability of RD ≥ 2 is much better than that of RD $= 1$, and the fault tolerance capability of cluster size ≥ 4 is better than that of cluster size $= 2$. The interested reader may refer to [14] for further performance details.

5. CONCLUSION AND FUTURE WORK

In this paper we presented a predictive fault tolerant placement algorithm that aims to provide high data availability, fault tolerance and enhanced performance in a distributed video server environment. The algorithm defines the notion of *replication degree* that ensures that multiple replicas of a video object exist, thus offering high availability and fault tolerance. It also explores different object distribution strategies so as to provide flexibility in load balancing and therefore enhanced performance. Our simulation results show that the choice of the object distribution strategy plays an important role in performance.

We plan to further investigate system performance under other forms of failures such as video stream failures. Accuracy of prediction and its impact on performance is another interesting research dimension for further exploration. This paper explores the placement of video objects in a faulty environment. We intend to explore the performance of fault tolerant placement of multimedia objects in web servers where access patterns differ significantly. We also intend to study mechanisms for data placement in mobile environments where failures occur frequently.

6. REFERENCES

[1] D. Bitton and J. Gray. Disk shadowing. In *Proc. of VLDB '88*, Los Angeles, California, 1988.

[2] P. M. Chen and E. K. Lee. Striping in a raid level 5 disk array. In *ACM SIGMETRICS'95*, Otta, Ontario, Canada.

[3] C. Chou, L. Golubchik, and J. C. Lui. A performance study of dynamic replication techniques in continuous media servers. In *Proc. of the Intl. Conf. on Measurement and Modelling of Computer Systems*, Atlanta, GA USA, May 1 - 4 1999.

[4] L. Golubchik, J. C. Lui, and R. R. Muntz. Chained declustering: Load balancing and robustness to skew and failures. In *Second Intl. Workshop on Research Issues on Data Engineering*, pages 88–95, Tempe, AZ USA, February 1992.

[5] H. Hsiao and D. Dewitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Intl. Conf. on Data Engineering*, pages 456–65, 1990.

[6] R. Muntz, J. R. Sanos, and F. Fabbrocino. Design of a fault tolerant real-time storage system for multimedia applications. In *Intl. Computer Performance and Dependability Symposium*, September 1998.

[7] R. R. Muntz and X. Ju. Staggered striping in multimedia information systems. *ACM SIGMOD*, 1994.

[8] B. Ozden, R. Rastogi, and A. Silberschatz. Disk striping in video server environments. In *Data Engineering*, 1995.

[9] D. A. Patterson, G. Gibson, and R. H. Katz. A case for redundant arrays of inexpensive disks (raid). *ACM SIGMOD*, June 1988.

[10] P. J. Shenoy and H. M. vin. Failure recovery algorithm for multimedia servers. *ACM Multimedia Systems Journal*, 8(1):1–19, January 2000.

[11] R. Tewan, D. M. Dias, Mukherjee, and H. M. Vin. High availability in clustered multimedia servers. In *Proc. of ICDE'96*, 1996.

[12] N. Venkatasubramanian and S.Ramanathan. Load management for distributed video servers. In *Proc. of the ICDCS'97*, May 1997.

[13] H. M. Vin, P. Goyal, A. Goyal, and A. Goyal. A statistical admission control algorithm for multimedia servers. In *ACM Multimedia'94*, 1994.

[14] X. Wei and N. Venkatasubramanian. Fault tolerant placement in distributed multimedia servers. Technical report, Info. and Comp. Sci. Dept, UC, Irvine, 2001.

[15] J. Wolf, P. Yu, and H. Shachnai. Dasd dancing: A disk load balancing optimization scheme for video-on-demand computer systems. *ACM SIGMETRICS'95, Performance Evaluation Review*, pages 157–166, May 1995.

2001 IEEE International Conference on Multimedia and Expo
ISBN 0-7695-1198-8/01 $17.00 © 2001 IEEE
0-7695-1198-8/01/$10.00 (C) 2001 IEEE