Optimal Video Replication and Placement on a Cluster of Video-on-Demand Servers *

Xiaobo Zhou
Department of Computer Science
Wayne State University
Detroit, MI 48202, USA
zbo@cs.wayne.edu

Cheng-Zhong Xu
Department of Electrical & Computer Engg.
Wayne State University
Detroit, MI 48202, USA
czxu@ece.eng.wayne.edu

Abstract

A cost-effective approach to building up scalable Video-on-Demand (VoD) servers is to couple a number of VoD servers together in a cluster. In this article, we study a crucial video replication and placement problem in a distributed storage VoD cluster for high quality and high availability services. We formulate it as a combinatorial optimization problem with objectives of maximizing the encoding bit rate and the number of replicas of each video and balancing the workload of the servers. It is subject to the constraints of the storage capacity and the outgoing network bandwidth of the servers. Under the assumption of single fixed encoding bit rate for all videos, we give an optimal replication algorithm and a bounded placement algorithm for videos with different popularities. To reduce the complexity of the replication algorithm, we present an efficient algorithm that utilizes the Zipf-like video popularity distributions to approximate the optimal solution. For videos with scalable encoding bit rates, we propose a heuristic algorithm based on simulated annealing. We conduct a comprehensive performance evaluation of the algorithms and demonstrate their effectiveness via simulations over a synthetic workload set.

1 Introduction

Early large-scale VoD servers were mostly running on massively parallel computers. For cost-effectiveness and scalability, the research interest in cluster-based streaming servers (briefly, VoD cluster) has increased dramatically in the past few years [2, 9, 10, 12, 15]. A VoD cluster consists of a number of back-end servers controlled by a dispatcher that makes admission decisions. To avoid network traffic jams around the dispatcher, the cluster relies on a TCP handoff protocol to enable servers to respond to client requests directly. There are cluster architectures for VoD servers: shared storage and distributed storage. A shared storage cluster is usually built on RAID systems [10, 14]. Video data is striped into blocks and distributed over the shared disk arrays. Such systems are easy to build and administrate and the cost of storage is low. However, they have limited scalability and reliability due to disk access contention. As the number of disks increases, so do the controlling overhead and the probability of a failure [9].

By contrast, in a distributed storage VoD cluster, each server has its own disk storage subsystem [2, 9, 12]. The servers are linked by a backbone network. Due to the server autonomy, this cluster architecture can offer better scalability in terms of storage and streaming capacity and higher reliability. This type of clusters can also be employed in geographically distributed environments [18].

In this article, we investigate a key issue in the design of this type of clusters, *i.e.* initial placement of videos onto the servers. This is because data placement methods are crucial to the scalability of the VoD cluster and the overhead of video placement is huge. Two complementary approaches are data striping and data replication. The main advantages of striping are high disk utilization and good load balancing ability [2, 9, 10]. However, wide data striping can induce high scheduling and extension overhead [4, 12]. Replication tends to isolate the servers from each other. It can simplify the administration and enhance scalability and reliability of the clusters. In this article, we advocate a video replication strategy that duplicates videos according to their popularities and place the replicas wholely on servers for reliability. Data striping and recovery schemes can be employed within the servers to enhance availability.

A defining characteristic with video streams is that a video can be encoded in different bit rates for different qualities at the cost of different storage and streaming bandwidth requirements. It is this unique feature that distinguishes video replication and placement problem from classical file

*This research was supported in part by NSF grants ACI-0203592 and CCR-9988266.

allocation problems (FAP) [6]. Due to huge storage requirements of videos, full replication is generally inefficient if not impossible. For example, a typical 90-minute MPEG-2 video encoded in a constant bit rate of 6 Mbs requires as much as 4 GB storage. Assuming *a priori* knowledge about video popularities, we intend to find an efficient video placement with partial replication.

The problems are how to determine the encoding bit rates and the replication degree of videos and how to place the video replicas on the distributed storage cluster for high quality, high availability and load balancing under resource constraints. Like many related work [1, 5, 12], it considers videos with constant bit rates (CBR). High quality requires a high encoding bit rate. High availability in this context has two meanings: low rejection rate and high replication degree. The objective of load balancing is to improve system throughput in rush-hours and hence reduce the rejection rate [17]. It is known that increasing the replication degree enhances the flexibility of a system to balance the expected load. Multiple replicas also offer the flexibility in reconfiguration and increase the dynamic load-balancing ability. On the other hand, the replication degree is limited by encoding bit rates of videos and storage capacity of the cluster. Encoding bit rates also are constrained by the streaming bandwidth and the peak request arrival rate. This article presents a solid theoretical framework for the video replication and placement problem. Specifically,

1. We formulate the video replication and placement on distributed storage VoD clusters for high service quality and high service availability as a combinatorial optimization problem.

2. Under the assumption of single fixed encoding bit rate for all videos, we give an optimal replication algorithm and a bounded placement algorithm for videos with different popularities. The placement algorithm achieves a tight bound of load imbalance degree. To reduce the complexity of the replication algorithm, we present a time-efficient algorithm that utilizes the information about Zipf-like video popularity distributions to approximate the optimal solution.

3. For videos with scalable encoding bit rates, we propose a simulated annealing algorithm. We conduct a comprehensive performance evaluation of the algorithms and demonstrate their effectiveness via simulations over a synthetic workload set.

The rest of the article is organized as follows. Section 2 is about background and related work. Section 3 presents the formulation of the problem. Section 4 presents a family of algorithms and their analyses. Section 5 gives performance evaluation. Section 6 concludes the article with remarks on future work.

2 Related Work

VoD applications have long been a research topic. Early server-side studies focused on RAID-based storage subsystem designs as well as disk I/O and network bandwidth scheduling in single servers. The research on storage subsystems has been directed toward: 1) Data striping schemes in storage devices for disk utilization and load balancing [3, 13]; 2) Data retrieval from storage subsystems in order to amortize seek time [14]; 3) Buffering of data segment for disk I/O bandwidth utilization [8]; and 4) Disk scheduling to avoid jitter [14, 16]. There were other complementary studies on hierarchical storage subsystems, admission control and VCR operations. Video streams require huge amount of bandwidth in both disk I/O and network interfaces. Many work focused on reducing I/O requirements by caching, multicasting and broadcasting [1, 7]. Eager *et. al.* give a complete review of these techniques [7]. The research of this topic is out the scope of this paper.

Recent work on VoD applications focused on scalability and reliability in distributed VoD servers [4, 9, 10, 12]. The video replication and placement problem has been studied extensively in the literature [3, 4, 5, 15, 16]. Chervenak *et. al.* argued replication based on Zipf-like distribution access patterns could improve throughput [3]. The authors did not show how to take the advantage of Zipf-like distributions for replication and placement. Chou *et. al.* gave a comparison of scalability and reliability characteristics of servers by the use of striping and replication [4]. The work focused on the tradeoff between the degree of striping and the degree of replication. Dan *et. al.* proposed an online video placement policy based on network bandwidth to storage space ratio of storage devices [5]. The work focused on load balancing between storage devices of a VoD server. The placement policy is heuristic but not optimization-based. Venkatasubramanian *et. al.* proposed a family of heuristic algorithms for dynamic load balancing in a distributed VoD server. Wolf *et. al.* proposed DASD Dancing, for load balancing in a multi-disks server [16]. It employed a replication optimization technique borrowed from the theory of resource allocation problems [11]. In this article, we give an optimal video replication scheme between servers. We then present an efficient algorithm that utilizes the information about Zipf-like video popularity distributions to approximate the optimal solution. In addition, we give a placement strategy that achieves a tight bound for the load imbalance degree. We improve and complement previous work by constructing a theoretical framework for the problem and present a family of efficient replication and placement algorithms. Overall, the defining characteristic with our work is that we consider the tradeoff between the encoding bit rate for service quality and the replication degree for service availability and load balancing.

3 The Optimization Problem

3.1 The Model

We consider a cluster of N homogeneous servers, $S = (s_1, s_2, \ldots, s_n)$, and a set of M different videos, $V = \{v_1, v_2, \ldots, v_m\}$. Each server has a storage capacity C and an outgoing network bandwidth B. Like many other work [1, 15, 16], we consider all videos in set V have the same duration d, say d =90 minutes for typical movies. If a video v_i is encoded in constant bit rate b_i, the storage space for video v_i is calculated as $c_i = d \cdot b_i$.

It is known that the popularity of videos varies with a number of videos that receive most of the requests. We consider the replication and placement for the peak period of length d. This is because one of the objectives of the replication and placement is high service availability during the peak period. Load balancing is critical to improving throughput and service availability during the peak period. Like many other work [1], we consider that outgoing network bandwidth is the major performance bottleneck. We make the following two assumptions, regarding the video relative popularity distributions and the request arrival rates.

1. The popularity of the videos, p_i, is assumed to be known before the replication and placement. The relative popularity of videos follows Zipf-like distributions with a skew parameter of θ. Typically, $0.271 \leq \theta \leq 1$ [1, 15]. The probability of choosing the i^{th} video is $p_i = i^{-\theta} / \sum_{j=1}^{M} j^{-\theta}$.

2. The peak period is same for all videos with various arrival rates. Let $\bar{\lambda}$ denote the average arrival rate during the peak period. Because of the same peak period assumption, the video replication and placement is conservative as it places videos for the peak period. This conservative model provides an insight into the key aspects of the problem and facilitates its formulation.

3.2 The Formulation of the Problem

The objective of the replication and placement is to have high service quality and high service availability. Replication enhances availability and load balancing ability by placement. Load balancing improves system throughput and hence the availability. Increasing the encoding bit rate receives high quality but decreases the replication degree due to the storage constraint. The encoding bit rate of videos is also limited by the outgoing bandwidth constraint of the servers. The objective of the replication and placement hence is to maximize the average encoding bit rate and the average number of replicas (replication degree), and minimize the load imbalance degree of the cluster. Let L denote the communication load imbalance degree of the cluster. Let r_i denote the number of replicas of video v_i. Specifically, we define the optimization objective as:

$$Obj = \sum_{i=1}^{M} b_i/M + \alpha \cdot \sum_{i=1}^{M} r_i/M - \beta \cdot L, \qquad (1)$$

where α and β are relative weighting factors. There are many ways for the definition of load imbalance degree [17]. Two typical examples are

$$L = \max_{\forall s_i \in S} \mid l_i - \bar{l} \mid, \qquad (2)$$

and

$$L = \sqrt{\sum_{i=1}^{N} (l_i - \bar{l})^2 / N}, \qquad (3)$$

where $\bar{l}$ is the mean outgoing communication load of the servers, *i.e.* $\bar{l} = \sum_{i=1}^{N} l_i/N$. Unless otherwise specified, we use the definition of Eq. (2) in the following discussions.

This objective is subject to the following constraints: (1) server storage capacity, (2) server outgoing network bandwidth, and (3) distribution of all replicas of an individual video to different servers. All r_i replicas of video v_i have the same encoding bit rate since they are replicated by the same video. Let $\pi(v_i^j)$ be the index of the server on which the j^{th} of replicas of video v_i, v_i^j, places. And, $\pi(v_i) = k$ means that a replica of video v_i is placed on server s_k. The *communication weight* of each replica of video v_i is defined as $w_i = p_i/r_i$. By the use of a static round robin scheduling policy, the number of requests for video v_i to be serviced by each replica of v_i during the peak period is $w_i \cdot \bar{\lambda} \cdot d$. Let l_k be the outgoing communication load on server s_k. Specifically, we give resource constraints from the perspective of server $s_k (1 \leq k \leq N)$ as:

$$\sum_{\pi(v_i)=k, \forall v_i \in V} b_i \cdot d \leq C, \qquad (4)$$

and

$$l_k = \sum_{\pi(v_i)=k, \forall v_i \in V} w_i \cdot \bar{\lambda} \cdot d \cdot b_i \leq B. \qquad (5)$$

The third constraint is the requirement of distribution of all replicas of an individual video to different servers. That is, all r_i replicas of video v_i must be placed on r_i servers. Specifically,

$$\pi(v_i^{j_1}) \neq \pi(v_i^{j_2}) \qquad 1 \leq j_1, j_2 \leq r_i, j_1 \neq j_2. \qquad (6)$$

Note that if multiple replicas of a video are placed to the same server, it implies that these replicas be merged to one replica. For the same reason, we have one more replication constraint

$$1 \leq r_i \leq N \qquad \forall v_i \in V. \qquad (7)$$

In summary, we formulate the video replication and placement problem as a maximization of Eq. (1) and Eq. (2) or Eq. (3) subject to constraints of Eq. (4) to Eq. (7).

4 Algorithms and Analyses

4.1 Replication of Videos in a Fixed Bit Rate

We first present replication and placement algorithms under the assumption of single fixed encoding bit rate b [1, 15]. Since the storage requirement for each video is a constant, *i.e.* $c_i = d \cdot b$, we re-define the storage capacity of each server C in terms of the number of replicas. Unless otherwise specified, we use the re-definition of C in the following discussions. Note that because the encoding bit rate is fixed, the constraint Eq. (5) may be violated when communication load exceeds the outgoing bandwidth of the cluster.

Replication can achieve fine granularity in terms of communication weight of replicas that offers more flexibility to place replicas for load balancing. We will prove in Section 4.2 that the upper bound of the load imbalance degree L generated by our placement strategy is non-increasing as the replication degree increases. It is desirable to increase the replication degree to saturate the storage capacity of the cluster. The optimization of Eq. (1) is reduced to assigning the number of replicas to each video and minimizing the load imbalance degree L by placement of these replicas.

The objective of the replication is to get fine granularity of replicas in terms of communication weight for later placement. Specifically, we have

$$\text{Minimize} \qquad max_{\forall v_i \in V}\{w_i\}, \tag{8}$$

subject to constraints of Eq. (7) and $\sum_{i=1}^{M} r_i = N \cdot C$.

If the video popularity distribution is uniform, a simple round-robin replication achieves an optimal replication scheme with respect to Eq. (8). However, we know the popularity distribution of videos is usually not uniform. The problem of assigning the number of replicas to each video in proportion to its popularity distribution is close to a classical apportionment problem [11]. One known approach is Adams' monotone based divisor method [11, 16]. The difference is the number of replicas of each video is bounded by the number of servers due to constraint Eq. (7).

4.1.1 Bounded Adams' Monotone Divisor Replication

We give the bounded Adams' monotone divisor replication algorithm. It firstly assigns one replica to each video. For the rest replication capacity of the cluster, *i.e.* $N \cdot C - M$ replicas, at each iteration it gives one more replica to the video, whose number of replicas is less than the number of servers and its replica(s) has the currently greatest communication weight.

Figure 1 illustrates the replication of five videos on three servers. Without loss of generality, suppose $p_1 \geq p_2 \geq \cdots p_5$. The storage capacity of each server is three replicas. Initially, each video is given one replica and $w_i = p_i$. For the rest replication capacity of the cluster (4 replicas), at each iteration a video is chosen to do duplication. Its number of replicas is smaller than the number of servers and the replica(s) of this video has the greatest communication weight currently. For example, in the first iteration, video v_1 is duplicated into two replicas. The communication weight of its each replica is hence $w_1 = p_1/2$. In the second iteration, if $w_1 = p_1/2 = max\{p_1/2, p_2, p_3, p_4, p_5\}$, the two replicas of v_1 will be duplicated into three replicas. The communication weight of the three replicas is $p_1/3, p_1/3$ and $p_1/3$, respectively. In the third iteration, if video v_1 has already three replicas, it won't be duplicated any more. Video v_2 has the greatest communication weight and it is duplicated into two replicas with $w_2 = p_2/2$, and so on.

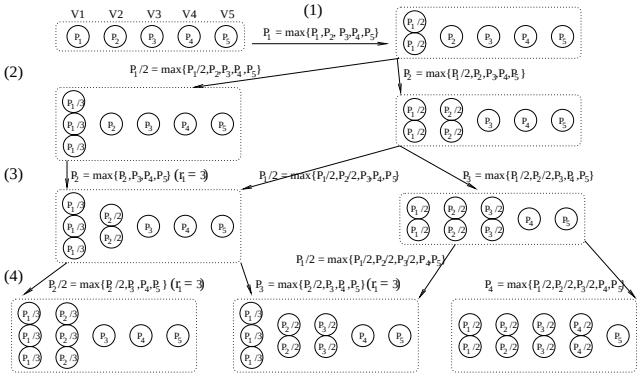


Figure 1: An illustration of the Adams' replication.

Theorem 1. *The bounded Adams' divisor algorithm receives an optimal replication scheme with respect to Eq. (8).*

It can be proved that the algorithm complexity increases with the storage capacity of the cluster. In the worst case, the algorithm complexity is $O(M \cdot N \cdot \log M)$ [11].

4.1.2 Zipf-like Distribution based Replication

The replication algorithms can be applied for dynamic replication during run-time. The disadvantage of the algorithm above is its complexity. To reduce it, we give an efficient algorithm that utilizes the information about the Zipf-like video popularity distributions to approximate the optimal solution. The key idea is to classify the popularities of the videos into N intervals. The videos, whose popularities are within the same interval, are assigned the same number of replicas according to their interval indices. This popularity classification is done in a heuristic way with a Zipf-like distribution. There are two key functions in the algorithm. The function $generate(\psi, u_j)$ partitions the range of [0, $p_1 + p_m$] into N intervals according to a Zipf-like distribution with a skew parameter of ψ, *i.e.* $\frac{u_j}{p_1+p_m} = j^{-\psi} / \sum_{k=1}^{N} k^{-\psi}, 1 \leq j \leq N$. The boundaries of intervals are given by z_j. Function $assignment(u_j, r_i)$ assigns the number of replicas r_i to video v_i according to the interval index of its popularity. Figure 2 illustrates a replication

senario with the setting of 20 videos, 6 servers and popularity parameter $\theta = 0.75$. The storage capacity of the cluster is 24 replicas. The replication algorithm has $\psi = -2.33$.

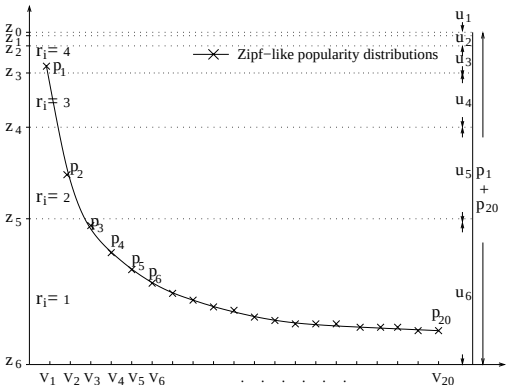


Figure 2: A replication senario.

The parameter ψ determines the total number of replicas generated by the algorithm. we employ a binary search approach to search the parameter ψ. Two key factors of the algorithmic complexity are the search space of parameter ψ and the termination condition of the iterative process.

Lemma 4.1. *In the Zipf-like distribution based replication algorithm, the total number of replicas is non-decreasing as the parameter ψ increases and non-increasing as the parameter ψ decreases.*

Proof: As the parameter ψ increases, the boundary $z_j (1 \leq j \leq N-1)$ decreases due to features of Zipf-like distributions. The total number of replicas is non-decreasing according to function $assignment(u_j, r_i)$. Similarly, we can prove the claim as the parameter ψ decreases.

We give the bounded search space of $[\psi_{min}, \psi_{max}]$ for the parameter ψ. If the first interval $\frac{u_1}{p_1+p_m} \geq \frac{p_1}{p_1+p_m}$, we have $r_i = N, \forall v_i \in V$ according to the function $assignment(u_j, r_i)$ and Lemma 4.1. It can be derived that $\psi_{max} = \theta \cdot \log M + \log N$. If the last interval $\frac{u_n}{p_1+p_m} \geq \frac{p_1}{p_1+p_m}$, we have $r_i = 1, \forall v_i \in V$ according to the function $assignment(u_j, r_i)$ and Lemma 4.1. It can be derived that $\psi_{min} = -\frac{\psi_{max}}{\log N - \log(N-1)}$.

It is known that $p_{m-1} - p_m = min_{\forall v_i, v_j \in V}\{\mid p_i - p_j \mid\}$. The binary search approach terminates when the change of ψ, denoted by ψ_δ, becomes smaller than $\frac{p_{m-1}-p_m}{p_1+p_m}$. Hence, the bound of ψ_δ is calculated as $\frac{(M-1)^{-\theta} - M^{-\theta}}{1+M^\theta}$. Due to $0.271 \leq \theta \leq 1$, the bound of ψ_δ is minimized when $\theta = 1$ and the minimum approximately equals to M^{-2}. Given the given search space and termination condition for the binary search approach, the complexity of the Zipf-like distribution based replication algorithm is $O(M \cdot \log M)$.

4.2 Video Placement

Video placement is to map all replicas of M videos to N servers to minimize the load imbalance degree L. If the previous replication leads to a uniform communication weight for all replicas, *i.e.* $w_i = w_j, \forall v_i, v_j \in V$, a round-robin placement achieves an optimal solution. It supposes that the replicas are arranged in groups in an arbitrary order such as $v_1^1 \ldots v_1^{r_1}, v_2^1 \ldots v_2^{r_2}, \ldots, v_m^1 \ldots v_m^{r_m}$.

However, in most cases, the communication weight of replicas of different videos may be different. For example, the ratio of the highest popularity to the lowest popularity is M^θ. If $M^\theta > N$, $\frac{w_1}{w_m} = \frac{p_1/r_1}{p_m/r_m} \geq \frac{p_1/N}{p_m/1} > 1$ due to constraint Eq. (7). This placement problem is more related to load balancing problems than to bin packing problems [17]. The difference with bin packing problems is that the number of servers for placement is given, rather than to be minimized. The differences with load balancing problems are there are storage limitations for placement and replicas of one video have to be placed to different servers due to constraint Eq. (6). We propose a placement algorithm called *smallest load first placement* in Algorithm 1.

Algorithm 1 Smallest Load First Placement

1: arrange all replicas of each video in a corresponding group;
2: sort these groups in a non-increasing order by the communication weight of the replicas in the groups;
3: **for** each of C iterations (C is the number of replicas that each server can contain) **do**
4: select N replicas with the greatest communication weights;
5: distribute these N replicas to the N servers – the distribution should satisfy that the replica with the greatest communication weight should be placed to the server with the smallest load and this server has not been placed with a replica of the same video;
6: **end for**

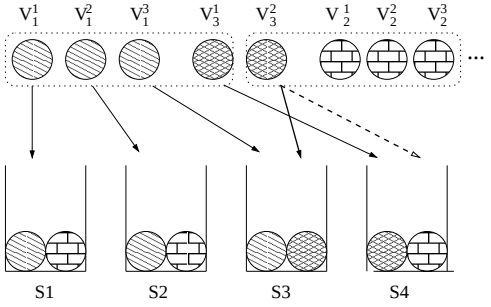


Figure 3: The smallest load first placement.

Figure 3 illustrates the placement strategy in a cluster of 4 servers. The replicas are arranged as $v_1^1, v_1^2, v_1^3, v_3^1, v_3^2, v_2^1, v_2^2, v_3^1 \ldots$ in a non-increasing order according to their communication weights. The first iteration places the first 4 replicas to the 4 servers. In second iteration, the server 4 has the current smallest load and the replica v_3^2 has the greatest communication weight. However, since the server 4 has already been placed with a replica of video v_3, *i.e.* v_3^1, the replica v_3^2 is placed to the server 3 with the second smallest load, and so on.

Theorem 3. *By the use of the smallest load first placement algorithm, the load imbalance degree defined by Eq. (2) is bounded by the difference between the greatest communication weight and smallest communication weight of the replica(s),* i.e. $L \leq max_{\forall v_i \in V}\{w_i\} - min_{\forall v_i \in V}\{w_i\}$.

Proof: Let $w_1, w_2, \ldots\ w_n, w_{n+1}, w_{n+2}, \ldots, w_{n \cdot C}$ be the communication weight of the $N \cdot C$ replicas in a non-increasing order. Clearly, $w_1 = max_{\forall v_i \in V}\{w_i\}$ and $w_{n \cdot C} = min_{\forall v_i \in V}\{w_i\}$. In the worse case, the change of L at iteration $j (0 \leq j \leq C-1)$ of the placement is bounded by $w_{j \cdot n+1} - w_{j \cdot n+n}$. The summation of these changes would be the worst case for L after the placement. We have $L \leq w_1 - w_n + w_{n+1} - w_{2n} \cdots + w_{n \cdot (C-1)+1} - w_{n \cdot C}$. It follows that $L \leq max_{\forall v_i \in V}\{w_i\} - min_{\forall v_i \in V}$.

Theorem 4. *By the use of the proposed replication and placement algorithms, the upper bound of the load imbalance degree, defined by Eq. (2), is non-increasing as the replication degree increases.*

4.3 Replication and Placement for Videos in Scalable Bit Rate

We consider the general case that the encoding bit rate is scalable and different videos can have different bit rates. The encoding bit rate is a discrete variable and its set is given. This framework provides a flexible way to maintain multiple copies of a video with different encoding bit rates. We propose a heuristic algorithm based on simulated annealing to solve the optimization formulated in Section 3. We construct the algorithm based on the parSA library [18]. The parallelization and generic decisions are connected with parameters of the parSA library itself and are transparent to users. Our implementation focuses on a class of problem-specific decisions. They include:

1. *Cost Function*: The objective function defined in Eq. (1) is the cost function of the simulated annealing.
2. *Initial Solution*: Initial solution place the videos encoded with the lowest possible bit rate to servers in a round-robin way. This makes sense for practical applications as each video can have one replica at least in a low bit rate quality.
3. *Neighborhood Structure:* To compute a neighborhood placement, a server in the cluster is identified by random. The bit rate of one video that has been placed on this server is increased or one new video is placed on the server. If above operation induces that the storage or the outgoing communication capacity of the server exceeds its limitation, the algorithm will decrease the bit rate of one or more videos that have been placed on the server, or delete one or more videos that are placed with the lowest bit rate so that the storage and communication constraints can be satisfied.

5 Performance Evaluation

In this section, we present the simulation results by the use of different replication and placement algorithms with various replication degrees under the assumption of a fixed encoding bit rate. Due to the space limitation, the results of the simulated annealing algorithm are omitted.

We simulated the Zipf-like distribution based replication (Zipf replication) and the bounded Adams' monotone divisor replication (Adams replication). We found that the Zipf replication and the Adams replication achieved nearly the same results in most test cases, except their time complexities. For brevity of the representation, we give the results of the Zipf replication only. To better understand the impact of different replication algorithms on performance, we simulated a feasible and straightforward algorithm called classification based replication [19]. For placement, we compare the round-robin placement algorithm and the smallest load first placement algorithm.

In the simulations, the VoD cluster consisted of 8 homogeneous servers. Each server had 1.8 Gbs outgoing network bandwidth. The storage capacity of each server ranged from 67.5 GB to 202.5 GB. The cluster contained 200 videos with duration 90 minutes each. The encoding bit rate for videos was fixed with the typical one for MPEG II movies, *i.e.* 4 Mbs. The storage requirement of a video hence was 2.7 GB. The storage capacity of the cluster ranged from 200 to 600 replicas and the replication degree ranged from 1.0 to 3.0.

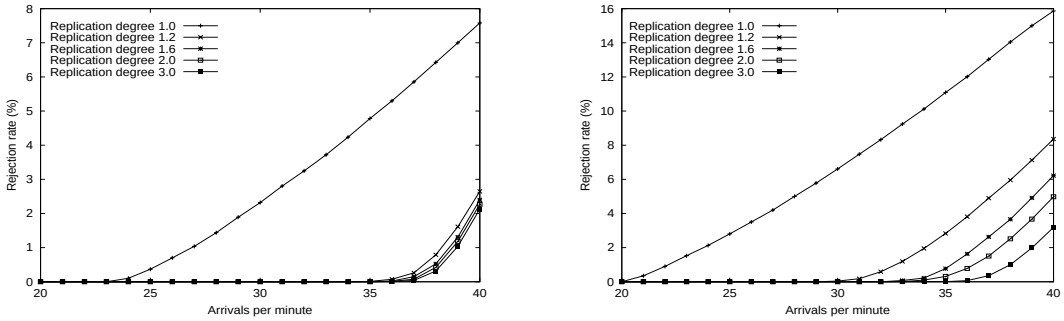
Within the peak period of 90 minutes, the request arrivals were generated by a Poisson process with arrival rate λ [1]. Since the outgoing network bandwidth of the cluster was 3600 streams of 4 Mbs, the peak rate of λ was 40 requests per minute. The video popularity distribution was governed by a Zipf skew parameter θ ($0.271 \leq \theta \leq 1$) [1, 15].

The simulation employed a simple admission control that a request was rejected if required communication bandwidth was unavailable. We use the rejection rate as the performance metric. In the following, we give some representative results. Each result was an average of 200 runs.

5.1 Impact of Replication on Rejection Rate

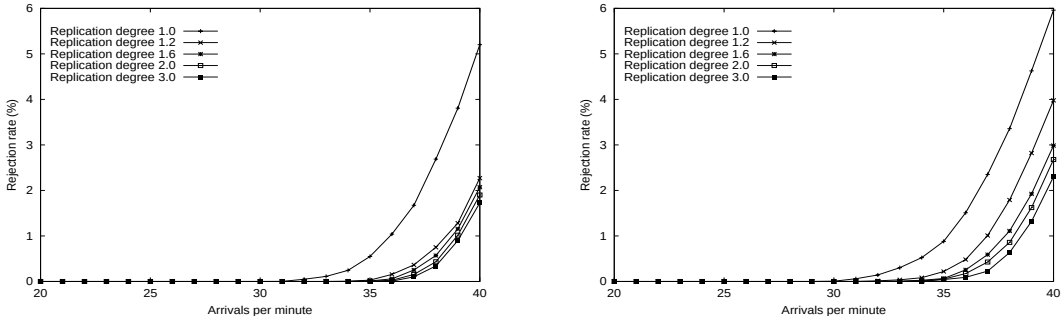
First we investigate the impact of replication degree (average number of replicas) on rejection rate. Figure 4 plots the representative results with a set of replication degree {1.0, 1.2, 1.6, 2.0, 3.0}. Different replication and placement algorithms were employed with popularity parameter $\theta = 1.0$ and 0.5, respectively.

Figure 4 shows that the rejection rate in all subplots decreases with the increase of the replication degree. In the subplot 4(a), the rejection rate decreases dramatically from non-replication to low replication degree 1.2. This is because the most popular videos are given the most replicas, which significantly reduces the granularity of replicas in



(a) Algorithm: the Zipf replication with the smallest load first placement when θ = 1.0.

(b) Algorithm: the classification replication with the round−robin placement when θ= 1.0.



(c) Algorithm: the Zipf replication with the smallest load first placement when θ = 0.5.

(d) Algorithm: the classification replication with the round−robin placement when θ= 0.5.

Figure 4: Impact of different replication degrees on rejection rate.

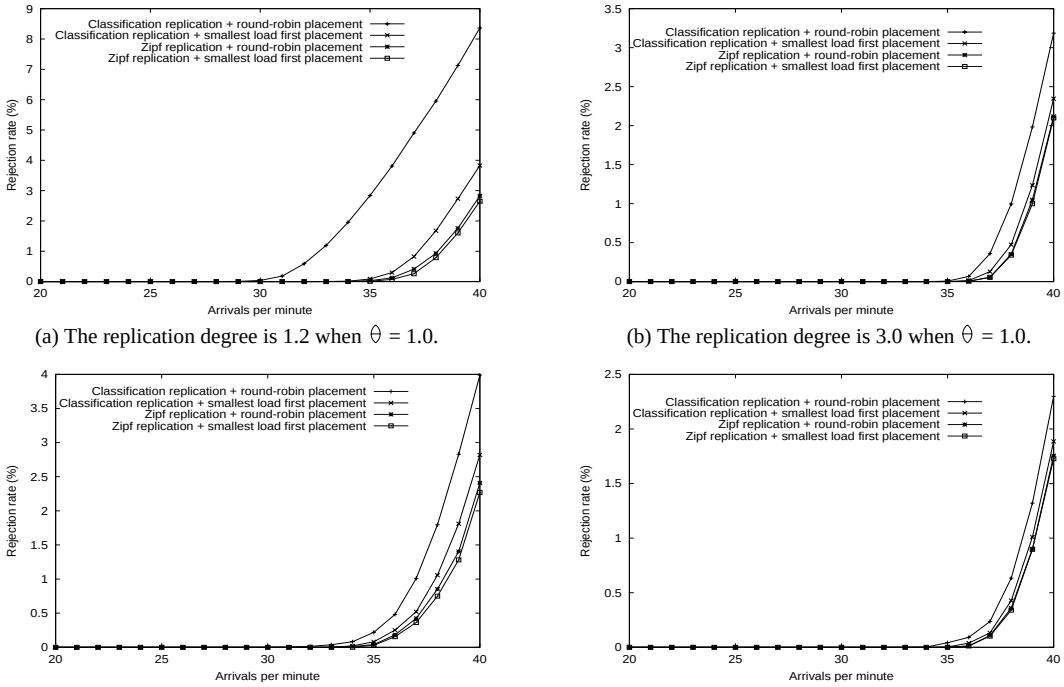
terms of communication weight for load balancing placement. There are no dramatic differences between the results of other replication degrees. In the subplot 4(b), the rejection rate gradually decreases with the increase of replication degree, although the change of rejection rate from non-replication to low replication degree 1.2 is still most significant. This means that the Zipf replication with the smallest load first placement can utilize the available storage space more efficiently than the classification based replication with the round-robin placement.

Figure 4 also shows that the Zipf replication with the smallest load first placement yield a lower rejection rate than the the classification based replication with the round-robin placement, especially in the cases with low replication degrees. When the replication degree increases, the difference between the algorithms decreases. It can be expected that as the replication degree reaches full replication, there would be no difference between the algorithms. In comparison with the subplots 4(a) and 4(c), the impact of replication degree decreases as parameter θ decreases. This is because as parameter θ decreases, the video popularity skew decreases which induces the fine granularity of communication weight of replicas for placement. The same conclusion can be made from the comparison between subplots 4(b) and 4(d). It is also agreed by the simulation results with many other θ values in between 0.271 to 1.0.

5.2 Impact of Algorithms on Rejection Rate

Figures 5 depicts the impact of four algorithm combinations on rejection rate when parameter θ is 1.0 and 0.5, respectively. Due to the space limitation, we give the representative results when the replication degree is 1.2 and 3.0.

Figure 5 shows that the algorithm combinations with either the Zipf replication or the smallest load first placement improves over the classification based replication with the round-robin placement significantly. The gap between the subplots of the classification replication with the round-robin placement and the classification based replication with the smallest load first placement shows the improvement of the placement strategy. The gap between the subplots of the classification based replication with the round-robin placement and the Zipf replication with the round-robin placement shows the improvement of the replication strategy. The results also show that the Zipf replication with the round-robin placement and the Zipf replication with the smallest load first placement have nominal differences. This demonstrates the effectiveness of the Zipf replication from another perspective. It can generate finely grained replicas in terms of communication weight for load balancing placement. The difference between algorithm combinations decreases with the increase of replication degrees. It is shown that as the replication degree increases, the curve of the Zipf



(a) The replication degree is 1.2 when θ = 1.0.

(b) The replication degree is 3.0 when θ = 1.0.

(c) The replication degree is 1.2 when θ = 0.5.

(d) The replication degree is 3.0 when θ = 0.5.

Figure 5: Impact of different replication and placement algorithms on rejection rate.

replication with the round-robin placement and the curve of the Zipf replication with the smallest load first placement are going to merge.

Figure 5 also shows that the curves have the same basic shapes as parameter θ decreases, although the differences in term of rejection rate are progressively less. From the comparisons of the four algorithm combinations, we make the conclusion that the Zipf replication with the smallest load first placement receive the desired result even under a very low replication degree. Other sensitivity analyses varied the number of videos, the video duration, the number of servers, the server outgoing bandwidth, as well as the encoding bit rate. We did not reach any significantly different conclusions regarding to the relative merits of the algorithms.

5.3 Impact of Algorithms on Load Imbalance

Figure 6 shows the change of the load imbalance degree L defined in Eq. (2) with various arrival rates. It helps us understand the performance curves of the four algorithm combinations depicted in Figure 5. Due to the space limitation, we give the results with the setting of $\theta = 1.0$ only.

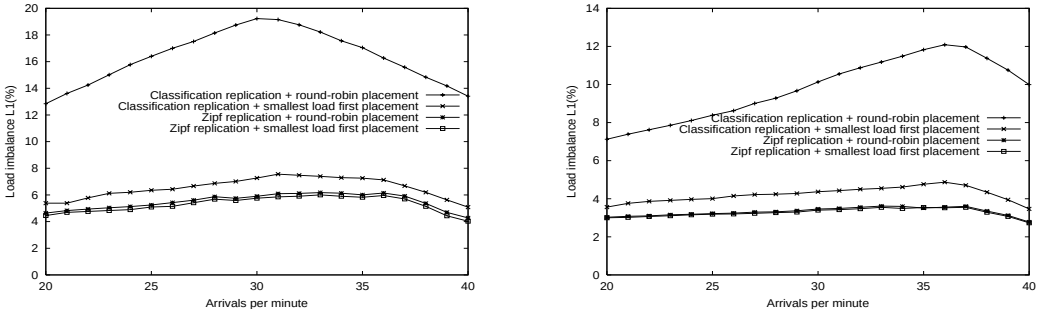
For the classification based replication with the round-robin placement, the load imbalance degree is affected by the arrival rate significantly. The algorithm combinations with either the Zipf replication or the smallest load first placement receive more stable results with arrival rates.

When system load is light, the load imbalance degree increases as the arrival rate increases. It reaches the peak values when the arrival rate is between 30 to 35 requests per minute. As the arrival rate is close to the outgoing bandwidth capacity of the cluster, the load imbalance degree in all figures decreases. Actually, when the arrival rate exceeds the throughput capacity about 20%, we found that the performance curves of all replication degrees almost merged because all servers were overloaded.

There would be no rejection before the arrival rate reaches the outgoing bandwidth capacity of the cluster, if communication traffic is perfectly balanced within the cluster. The primary reason is that the instances of arrival rate could not always be the mean value. It is the variance of arrival distributions that induces considerable dynamic load imbalance and hence rejections. When the arrival rate is 40 requests per minute, the rejection rate in both figures can be about 2%. We conclude that the Zipf replication with the smallest load first placement receive desirable performance with the accurate prediction of video popularities.

6 Conclusions and Future Work

We have investigated the video replication and placement problem in distributed storage VoD clusters. Under the assumption of single fixed encoding bit rate, we have given an optimal replication algorithm - the bounded Adams' monotone divisor replication and a bounded placement algorithm - the smallest load first placement. We have also presented



(a) The replication degree is 1.2 when θ = 1.0.

(b) The replication degree is 3.0 when θ = 1.0.

Figure 6: Impact of different replication and placement algorithms on load imbalance degree.

an efficient algorithm utilizing Zipf-like popularity distribution to approximate the optimal replication. For videos with scalable encoding bit rates, a simulated annealing algorithm has been proposed. Comprehensive experiments have demonstrated the effectiveness of the algorithms.

The work in this article was based on the assumptions of a priori knowledge of video popularities and the same peak period of request rates for all videos. To complement the conservative video replication and placement strategies, we have given a request redirection strategy that utilizes the internal backbone bandwidth to balance the outgoing network traffic between the servers during the runtime [19].

The replication and placement framework in this article provides a flexible way to maintains multiple replicas of a video with different encoding bit rates. The flexibility can facilitate providing different qualities to requests for various videos or to requests from various clients/devices. We will report our experience in future work.

References

[1] C. C. Aggarwal, J. L. Wolf, and P. S. Yu. The maximum factor queue length batching scheme for video-on-demand systems. *IEEE Trans. on Computers*, 50(2):97–110, 2001.

[2] W. Bolosky, J. Barrera, R. Draves, G. Fitzgerald, Gibson, M. Jones, S. Levi, N. Myhrvold, and Rashid R. The Tiger video fileserver. In *Proc. NOSSDAV'96*, 1996.

[3] A. L. Chervenak, D. A. Patterson, and R. H. Katz. Choosing the best storage system for video service. In *Proc. ACM Multimedia'95*, pages 109–119, 1995.

[4] C. F. Chou, L. Golubchik, and J. C. S. Lui. Striping doesn't scale: how to achieve scalability for continuous media servers with replication. In *Proc. IEEE ICDCS'00*, pages 64–71, 2000.

[5] A. Dan and D. Sitaram. An online video placement policy based on bandwidth to space ratio (BSR). In *Proc. ACM SIGMOD'95*, pages 376–385, 1995.

[6] L.W. Dowdy and D.V. Foster. Comparative models of the file assignment problem. *ACM Computing Surveys*, 14(2):287–313, 1982.

[7] D. Eager, M. Vernon, and J. Zahorjan. Minimizing bandwidth requirements for on-demand data delivery. *IEEE Trans. on Knowledge and Data Engineering*, 13(5), 2001.

[8] W-C. Feng, B. Krishnaswami, and A. Prabhudev. Proactive buffer management for the streamed delivery of stored video. In *Proc. ACM Multimedia'98*, pages 285–290, 1998.

[9] J. Gafsi and E. W. Biersack. Modeling and performance comparison of reliability strategies for distributed video servers. *IEEE Trans. on Parallel and Distributed Systems*, 11(4):412–430, 2000.

[10] L. Golubchik, R. R. Muntz, C. Chou, and S. Berson. Design of fault-tolerant large-scale VoD servers: with emphasis on high-performance and low-cost. *IEEE Trans. on Parallel and Distributed Systems*, 12(4):363–386, 2001.

[11] T. Ibarkai and N. Katoh. *Resource allocation problem - Algorithmic approaches*. The MIT Press, 1988.

[12] Y. B. Lee and P. C. Wong. Performance analysis of a pull-based parallel video server. *IEEE Trans. on Parallel and Distributed Systems*, 11(12):1217–1231, 2000.

[13] P. J. Shenoy, P. Goyal, S. Rao, and H. M. Vin. Symphony: An integrated multimedia file system. In *Proc. ACM/SPIE Multimedia Computing and Networking*, pages 124–138, 1998.

[14] F. Tobagi, J. Pang, R. Baird, and M. Gang. Streaming RAID: A disk array management system for video files. In *Proc. ACM Multimedia'93*, pages 393–400, 1993.

[15] N. Venkatasubramanian and S. Ramanathan. Load management in distributed video servers. In *Proc. IEEE ICDCS'97*, pages 31–39, 1997.

[16] J. L. Wolf, P. S. Yu, and H. Shachinai. Disk load balancing for video-on-demand systems. *ACM/Springer Multimedia Systems Journal*, 5(6):358–370, 1997.

[17] C. Xu and F. Lau. *Load Balancing in Parallel Computers: Theory and Practice*. Kluwer Academic Publishers, 1997.

[18] X. Zhou, R. Lüling, and L. Xie. Solving a media mapping problem in a hierarchical server network with parallel simulated annealing. In *Proc. 29th ICPP*, pages 115–124, 2000.

[19] X. Zhou and C. Xu. Request redirection and data layout for network traffic balancing in cluster-based video-on-demand servers. In *Proc. IEEE PDIVM Workshop (in conjunction with IPDPS)*, 2002.