Multimedia Systems 10: 302–316 (2005)
Digital Object Identifier (DOI) 10.1007/s00530-004-0164-1

Multimedia Systems

Video server scheduling using random early request migration

Yinqing Zhao[*], **C.-C. Jay Kuo**[**]

Integrated Media Systems Center and Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564, USA

Revised: 24 October 2004
Published online: 8 April 2005 –

Abstract. Video request migration among servers to achieve effective video-on-demand (VoD) services is investigated in this work. Our study is focused on the design and analysis of a random early migration (REM) scheme for user requests. When a new request is dispatched to a video server, the REM-based scheduler decides whether request migration is needed with a certain probability, which is a function of the service load. To analyze the request migration process, we introduce a state matrix representation that stores the service load information of each video server and plays an important role in the determination of migration paths. Based on this representation, we develop two methods to calculate performance metrics: the service failure rate and the system delay in service migration. Simulation results show that the REM scheme outperforms both the DASD dancing algorithm [1] and the traditional migration scheme adopted in [2, 3] with shorter service delay and lower failure rates. It is also confirmed that our theoretical results match well with experimental results.

Keywords: Random early migration (REM) – State matrix – Video server scheduling – Video-on-demand

1 Introduction

This work considers a video-on-demand (VoD) system that consists of a centralized scheduler and a collection of video servers. The scheduler dispatches user requests and maintains the current service status for each video server and video access statistics, i.e., the bookkeeping information. The video server mainly performs I/O operations to deliver multiple streams of different video contents simultaneously to users based on a scheduled result. Due to the large size of video objects and limited storage capacity, each server can only store a limited amount of video streams. Moreover, the service capacity of each server is also limited by its bandwidth constraint. It is therefore a great challenge to solve the video server scheduling problem, which involves balancing the load among video servers to improve aggregate resource utilization and the overall system performance.

In VoD systems, the video server configuration mainly falls into three categories: flat striping, replication, and grouped striping. The flat striping scheme uses the RAID-type disks and stripes a video file evenly over all servers [4–6]. The VoD system under this configuration is called parallel video servers [7]. Striping can transparently distribute data over multiple disks and give users the appearance of a single large and fast disk [8]. However, due to delay variations in processing, networking, and scheduling, it is difficult to synchronize video delivery among striping servers. The replication scheme puts one whole video file in one video server and makes several extra copies of hot movies. Little et al. [9] explored the probability-based data replication policy. Serpanos et al. [10] proposed a data replication scheme called "Mmpacking" in distributed video servers. Other replication algorithms were proposed to reduce the storage overhead and balance the load among replicated servers [11, 12].

Replication is often used in geographically distributed VoD systems where video servers are far from each other so that it is difficult to implement server striping. The grouped striping scheme combines the pure striping scheme and the replication scheme to achieve a certain degree of load balancing and high throughput for hot video files. Each video server has a fixed striped-disk configuration, and hot video titles are replicated based on the forecast access probability. This scheme is used in the SCAMS (SCAlable Multimedia Servers) system of Lancaster University [13]. Some video content allocation algorithms have been proposed under this configuration to achieve static load balance among video servers and increase server connectivity for possible request migration in one service period [1, 2, 14].

Request migration is an important issue in video server scheduling. Due to the highly skewed access probabilities of video content, video servers with hot video copies receive many more client requests than other servers and are more likely to be overloaded. Request migration redistributes system load among servers to achieve a certain degree of load balancing, which in turn reduces the service failure rate, defined as the number of requests rejected by the VoD system divided by the number of requests received during the service period.

[*] e-mail: yinqingz@usc.edu
[**] e-mail: cckuo@sipi.usc.edu

Here, we propose a random early migration (REM) algorithm to reduce the service failure rate, balance the load of video servers, and further reduce the service delay measured by the number of migration steps involved in the migration process. When a new request is dispatched to a video server, REM decides whether request migration is needed with a certain probability using preset migration thresholds and the current server load status. A detailed description of REM algorithm will be given in Sect. 3.1.

A state matrix structure (SMS) is proposed in this study to analyze the performance of VoD systems in terms of the service failure rate and service delay. The state matrix stores the service load information of each video server at certain times and plays an important role in the determination of migration paths. Based on the SMS, we develop methods to calculate performance metrics for the traditional migration scheme and the proposed REM scheme. The derived results match well with those obtained by numerical experiments. The proposed REM algorithm outperforms the traditional migration scheme with lower failure rate and shorter service delay.

The rest of this paper is organized as follows. We briefly review previous work in video server scheduling and highlight our contributions in Sect. 2. We focus on the request migration strategies proposed for video servers in Sect. 2.1 and previous performance analysis work on VoD systems in Sect. 2.2. The REM algorithm for request migration is presented in detail in Sect. 3.1, followed by experimental results and analysis in Sect. 3.2. The mathematical model for a VoD system is given in Sect. 4. The performance analysis of the traditional migration and the proposed REM algorithms is conducted in Sect. 5, where we propose two approaches for analysis, namely, the topology calculation in the state matrix space (SMS) and the state transition method. Finally, concluding remarks and future work are given in Sect. 6.

2 Related work and our contributions

2.1 Request migration among video servers

Video file replication between video servers (which are called peers) enables real-time service migration between them to balance the service load. The traditional migration algorithm considers request migration until a new request gets blocked at the server. This scheme is called the last-minute request migration. There are two main problems in the traditional migration algorithm. First, the real-time service load distribution is highly skewed when migration happens, and such imbalance will degrade the VoD system performance. Second, with a highly skewed video access probability, some video server could already reach its service capacity and not accept client requests from its peer servers without any further migration, which could lead to additional delay in the VoD system. The first case can cause the request being rejected so that the service failure rate of the VoD system will increase. For the second case, the request will be held at the server so that the system response time will increase. None of them are desirable for real-time video service.

Wolf et al. [1] proposed the DASD dancing algorithm to balance the service load through request migration. The VoD system consists of a central processor and a collection of shared disks known as direct access storage devices (DASDs). In real-time scheduling, there is a penalty function associated with each video server to measure the load status. New requests are accepted by the system based on the least load first (LLF) rule most of the time. After a new request is accepted, the dancing algorithm calculates the optimal load distribution that minimizes the sum of all penalty functions. If the difference between the optimal distribution and the current load distribution exceeds a preset threshold (referred to as the *bad threshold*), it greedily migrates requests from overloaded servers to underloaded servers until the difference is below the threshold. The DASD dancing algorithm maintains disk load balance via request migration among video servers. However, when the system is lightly loaded, servers have enough service space to accommodate incoming requests and load balance contributes little to the improvement of the failure rate.

Tsao et al. [2] proposed a connectivity optimization (CO) algorithm for dynamic load balancing among distributed video servers. Their method attempted to maximize the product of access probabilities of servers, thus making all servers equally accessed. Guo et al. [14] proposed the combination load balancing (CLB) algorithm for video replica allocation to reduce the blocking rate of user requests. CLB divides video objects into different types according to the number of their replicas. For all type c ($c = 1, 2, \ldots$) video objects, the replica allocation is processed in such a way as to make the traffic load as evenly distributed as possible on each server combination that contains exactly c servers. This allocation procedure is then repeated for all values of c. The request migration/redirection mechanism used in their systems is the traditional migration scheme that happens only when the server is fully loaded and has no service space for incoming requests.

We propose a random early migration (REM) algorithm in this work to reduce the service failure rate, balance the load of video servers, and reduce service delay. When a new request is dispatched to a video server, REM compares the current service load with preset thresholds and decides whether request migration is needed with a certain probability, which is a function of the service load. The details of the REM algorithm will be given in Sect. 3. Simulation results demonstrate that REM can achieve enhanced system performance.

2.2 Performance analysis of VoD systems

Most work on the performance study of VoD systems is concerned with the network server I/O bandwidth (i.e., the number of video streams assigned) and the storage cost. Doganata et al. [15] presented the characteristics of different storage media and provided an analytical model to obtain the system storage cost. Schaffa et al. [16] studied the tradeoff between the bandwidth requirement of video stream delivery and the storage capability of the overall deployment. Barnett et al. [17] compared the "setup" cost of a centralized video system with a distributed one in terms of cost on storage and streaming. Chan et al. [18] explored the total system cost in terms of local storage and network channels for different caching schemes and service-scheduling mechanisms. Li et al. [19] used the queueing model to integrate both user activities and server batching models to analyze the tradeoff in communication and storage costs in different VoD systems.

Recently, there have been research efforts focusing on the performance of the whole VoD system including the server system, the client population, and the network between them. Mundur et al. [3] studied the server admission control and network transmission mechanism to meet the quality-of-service (QoS) requirement such as data rate and delay constraint of the end-to-end VoD service. The simple redirection (migration) scheme implemented in their system is still a last-minute migration in the sense that redirection happens only for those blocked requests. Shu et al. [20] analyzed the resource requirement for video delivery using batching, patching, and scheduled video delivery (SVD). The resource was defined as the virtual channel used to deliver video streams to one or multiple user requests. None of the above work considers server scheduling and interactivities between video servers.

In this study, we focus on two performance metrics of request migration among servers, i.e., the service delay time and the service failure rate. The difference between our work and all previous work lies in the fact that we model the service activity and try to reveal the scheduling mechanism of video servers. We present the state matrix structure as a description of the VoD system state. It stores the service load information of each video server and plays an important role in the determination of migration paths. Based on this structure, we develop two analytical methods for the request migration process. The first method defines the state matrix space (SMS) consisting of all valid state matrices and studies the relationship between the state matrix and the migration path decision. Both migration delay and service failure rates can be obtained from the topology calculation of SMS. For the second method, we use the state matrix transition to simulate the request migration process. The performance metrics can be obtained through the calculation in each transition step. The derived results are verified by numerical experiments. It is shown in Sect. 5.3 that theoretical and simulation results match well.

For the reader's convenience, key variables used in this paper are summarized in Table 1.

Table 1. Summary of key variables

Symbols	Meaning
M	number of video titles stored on a VoD system
N	number of video servers
L	service capacity of a video server in terms of concurrent video streams
K	storage capacity of a video server
K_j	number of copies of video j
Λ	video copy allocation matrix
G	connectivity matrix of video servers
H	state matrix of a VoD system
τ	state transform through request migration
δ	state transform through request accepted locally
$D(H^{(2)}, H^{(1)})$	distance between state matrices $H^{(2)}$ and $H^{(1)}$
ST	video bookkeeping information
$DIST$	distance matrix used in video content allocation
MP	request migration path
$\Gamma(MP)$	number of state matrices corresponding to migration path MP
μ	request coming rate per minute
$\Delta(\mu)$	service delay at μ requests/min
$F(\mu)$	service failure rate at μ requests/min
l_i	real-time service load on server i
T_s	service time of video file
ψ_i	access probability of server i
ϕ_j	access probability of video j
ϕ_{uj}	unit access probability of video j
$r(i, j)$	request dispatched to server i for video V_j
$e(i, j)$	service end at server i for video V_j
Min_th	minimum threshold in the REM algorithm
Max_th	maximum threshold in the REM algorithm
$p_{\max}$	maximum migration probability
p_i	request migration probability for server i

3 Random early migration algorithm

3.1 Algorithm

In a VoD system, it sometimes occurs that, when a new request for a particular video arrives, all video servers with this video cached on have reached their maximum service capacity. Thus, the request is blocked. However, we may still be able to serve this request by migrating an in-service request on one of these servers to another server in the system. The migrated request may be accommodated by another server immediately, or it may demand a sequence of migration operations before the request can be migrated successfully.

Figure 1 illustrates one example of request migration. There are three video servers in the VoD system. Each server can store two video copies and serve up to six requests simultaneously. Assume that a new request for video A arrives at some time. Since only server 1 has the copy of video A and it has already reached the service capacity, we need to migrate one in-service request of video B to server 2. The migrated request cannot be served immediately at server 2 since it also reached the service capacity. Then, server 2 migrates a request of video C to server 3, which is under its service capacity at that time. Note that, after the migration, the service load on all servers along the migration path except the last one remains unchanged. The last server on the migration path has its service load increased by one.

Here, we propose a random early migration (REM) algorithm that migrates in-service requests between servers to achieve a more balanced service load distribution with a certain probability. The idea of REM is motivated by the random early detection (RED) gateway used in congestion avoidance in packet-switched networks [21]. The RED gateway detects incipient congestion by computing the average queue size and drops or marks the arriving packet with a certain probability when the average queue size exceeds a preset probability. Similarly, REM monitors the current service load of each server to decide whether a request migration is necessary. Let L denote the service capacity of each video server (here we assume that each server has an identical service capacity) and l_i the current load on server i in terms of the number of in-service requests. REM uses two thresholds Min_th and Max_th ($Min\_th < Max\_th$) to measure the load status on each server. When the load is less than Min_th, there is no need to do request migration since only part of the service capacity is used and the server can still accommodate new requests without the risk of reaching its service capacity. When

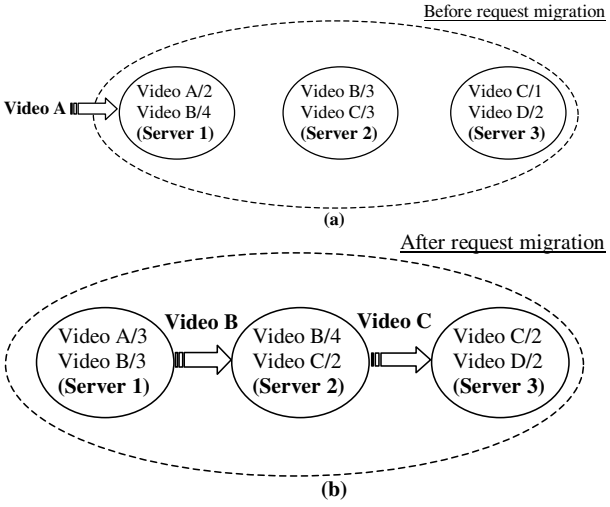

Fig. 1. Example of request migration. **a** Server status when a new request for video A arrives. **b** Server status after request migration

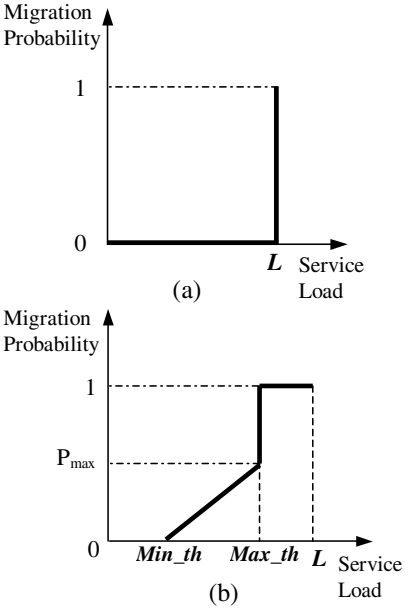

Fig. 2. Relationship between migration probability and current load. **a** Traditional migration algorithm. **b** Random early migration algorithm

the load is between Min_th and Max_th, there is a potential that the load will increase to the service capacity and request migration is performed with a certain probability p_i based on the current load on server i.

Figure 2 illustrates the relationship between migration probability p_i and current service load l_i for the traditional migration algorithm and REM. When the load is above Max_th, the server is heavily loaded and request migration is performed with probability 1. In the REM algorithm, the probability of request migration increases as the service load increases. As shown in Fig. 2, as load l_i varies from Min_th to Max_th, the migration probability p_i varies linearly from 0 to $p_{\max}$, i.e.,

$$p_i = p_{\max}\frac{(l_i - Min\_th)}{(Max\_th - Min\_th)}. \quad (1)$$

We deploy the REM algorithm on the centralized scheduler of the VoD system. Upon receiving a client request for video V_j on server S_i, the following operations are performed.

Step 1 The scheduler calculates the migration probability p_i based on the current load of S_i and preset REM parameters as follows.
1. If $l_i < Min\_th$, then $p_i = 0$. Go to step 5;
2. If $Min\_th \le l_i < Max\_th$, then calculate p_i using (1);
3. If $Max\_th \le l_i$, then $p_i = 1$. Go to step 3.

Step 2 The scheduler generates a random number p, $p \in [0, 1]$. If $p \le p_i$, then go to step 3; otherwise, go to step 5.

Step 3 The scheduler calculates the migration path starting from S_i.
1. If a migration path exists, then go to step 4;
2. If no migration path exists but $l_i < L$, then go to step 5; otherwise, go to step 6.

Step 4 The scheduler sends messages to servers involved in the migration path. The request migration is processed along the path. In the meantime, the incoming request is held at S_i.

Step 5 S_i starts to serve the incoming request. The scheduler updates the state matrix (discussed in detail in Sect. 4.1) of the VoD system. Go to step 7.

Step 6 The incoming request is rejected.

Step 7 The scheduler waits for the next request.

In REM, for each request migration starting at server S_i, we have to find a feasible path. Since the incoming request is held at S_i in the migration process, we want to find the shortest migration path to reduce service delay. When there is more than one shortest path, we choose the path based on the load balance criterion. That is, let us define the load balance degree as $\lambda = \prod_{i=1}^{N}(l_i/L)$. The migration path with the maximum λ value is selected. Note that, given the total number of in-service requests in a VoD system, λ is maximized when $l_1/L \approx l_2/L \approx \cdots \approx l_N/L$.

Let us discuss the complexity of the REM algorithm. According to the above step-by-step description, we can see that the migration decision, migration path calculation, and bookkeeping information update are all made at the scheduler. Thus, the scheduler has the whole view of the system. Upon an incoming request, the scheduler calculates the migration probability and finds the migration path if the migration happens. Then, it updates the bookkeeping information accordingly to keep a consistent view of the system. It can schedule the next request instead of waiting for the migration among media servers. The decision making at the scheduler and the step-by-step migration can be done in parallel. In such a scenario, the complexity of the REM algorithm depends on the migration path calculation, which is the most time-consuming task. Since the destination server is not known a priori, the migration path calculation is a single-source shortest path problem and can be solved using Dijkstra's algorithm [22]. The time complexity is $O(N^2)$, where N is the number of servers in the system.

The REM algorithm can be easily extended to the VoD system composed of heterogeneous servers, where each server may have a different storage capacity and/or service capacity. REM puts no restriction on server storage capacity. The only change is the bookkeeping information at the scheduler. For servers cached more video replicas, there may be more service

status entries. To deal with servers with different service capacities, we define R_{min} and $R_{\max}$ ($0 < R_{min} < R_{\max} < 1$) as the normalized thresholds such that $Min\_th_i = L_i R_{min}$ and $Max\_th_i = L_i R_{\max}$, where L_i is the service capacity of server S_i. We can configure R_{min} and $R_{\max}$ instead of Min_th and Max_th and use a previous implementation at the scheduler.

The difference between Wolf's DASD dancing algorithm [1], which is reviewed in Sect. 2.1, and REM can be summarized as follows. In the DASD dancing algorithm, the optimal value of the cost function is obtained when the service load is balanced among video servers. The DASD dancing algorithm attempts to maintain load balancing without considering the server load status. However, when the server is lightly loaded, it has enough service space to accommodate new requests locally without the need for migration. In this stage, the system can tolerate certain imbalance without the risk of increasing the failure rate. Furthermore, achieving load balance through request migration will increase the system cost. In contrast, REM applies a different migration strategy by taking the server load into consideration. When the server is lightly loaded, new requests are accepted locally, which reduces the migration cost. As the service load increases, the migration probability also increases and REM migrates requests more aggressively. REM achieves a more balanced load distribution than the traditional migration algorithm via earlier migration. Finally, load balancing becomes important and affects the failure rate and the migration cost in a heavily loaded system in REM.

Let us briefly discuss the choice of parameters $p_{\max}$, Max_th, and Min_th based on the intuitive arguments below. More detailed discussion is presented in Sect. 5.3. In RED, the gateway detects the average queue size when a new packet arrives and randomly drops packets in the queue. Packet dropping is a kind of feedback to let TCP decrease the congestion window and retransmit the dropped packet. Parameter $p_{\max}$ is typically set to a small value to reduce oscillations in the average queue size at the gateway and in the packet-dropping probability [21]. In a VoD system, the situation is somewhat different. The in-service request cannot be dropped but must be migrated. Besides directly dispatched requests, each server may need to serve some migrated requests from its peer servers. When the whole VoD system is under heavy load, request migration happens frequently and the number of in-service requests on a video server increases very fast. Thus, we would like to set $p_{\max}$ to a larger value to more aggressively migrate out in-service requests when the service load is near Max_th.

There are also considerations in the setting of thresholds Min_th and Max_th. First, when the server is lightly loaded, it still has a lot of room to accommodate new requests so that there is little chance that the request will be blocked or rejected. Moreover, each request migration involves the cost of control message passing, admission control exercising, and job rescheduling. For simulations in this work, we set Min_th to 60–70% of the full service capacity to avoid unnecessary migration. Second, when service load is between two thresholds, request migration starts. Among all in-service requests on a server, some are migrated from its peer servers. The server must have enough service space to accommodate those migrated requests. Therefore, $Max\_th - Min\_th$ cannot be too small. When the service load is near Max_th, request migration happens frequently and we still need to leave some service space for the request migrated from peer servers. The last point is to set Max_th near the service capacity but not equal to it.

3.2 Experimental results and analysis

We evaluate the performance of three request migration algorithms: the traditional request migration, DASD, and REM. In the traditional algorithm, request migration occurs when the server is at its full service capacity and a new request is blocked at this server, which is referred to as the last-minute request migration. The following parameters are adopted for the VoD system. There are a total of 140 video programs and 24 video servers. Each server has 4 physical disks, and 4-way striping is implemented. Thus, the total number of physical disks is 96. Each physical disk has a storage capacity of 3 video files. It is assumed that all physical disks are identical in performance to a service capacity of 30 concurrent video requests per disk. Consequently, the storage capacity of a video server is 12 video files while the service capacity is 120 concurrent requests.

The access patterns adopted by the simulation are described below. The user request is modeled as a Poisson process with an arrival rate of μ requests/minute, where μ varies from 26 to 34. The access probability of each of 140 video programs is modeled by the Zipf-like distribution [23] with $\theta = 0.27$. All video files are assumed to have the same length in terms of playback time, which is set to 90 min. The VoD system is initially idle. Then, it can serve up to 2880 concurrent user requests in a 90-min period. This implies an average arrival rate of 32 requests/minute. In the simulation, we use arrival rate (μ) to represent the system load. Our simulation work was conducted over a consecutive period of 7 days. Relevant data were recorded every 5 min. As for the parameters of the REM algorithm, the lower and upper thresholds were set to 80 and 105, respectively, and $p_{\max}$ was set to 0.8. We set the badness threshold of the DASD dancing algorithm to 10, which is the same as in Wolf's simulation [1].

First, let us evaluate service failure rate. Figure 3 shows the service failure rate versus the system service load in terms

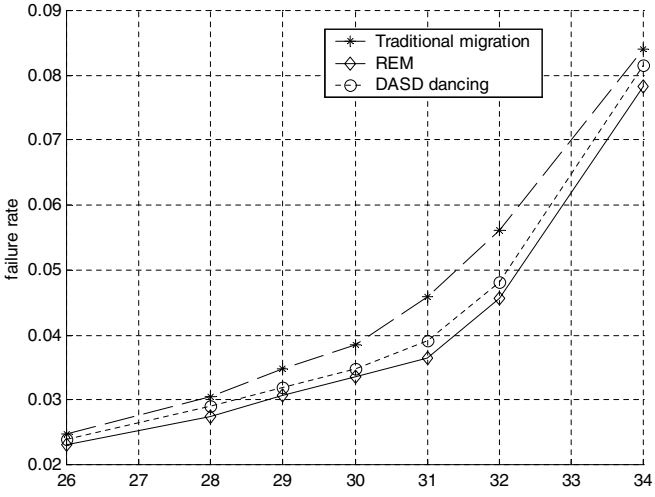

Fig. 3. Service failure rate as a function of system load

of the user request rate. We see from the figure that the service failure rate increases as the average request arrival rate μ increases, which is an indicator of the increase of the service load. When the system is lightly loaded ($\mu < 30$), all three schemes can achieve good performance. When the system load is around its service capacity ($30 \leq \mu \leq 34$), REM alone can reduce the service failure rate by 1015% from the traditional request migration algorithm. DASD performs slightly worse than REM. Both algorithms attempt to balance the load and redistribute load among servers, which can reduce the chance that a server will reach its service capacity too early to directly accept future requests without further migration.

We use load variance $LV \stackrel{\text{def}}{=} \frac{1}{N}\sum_{i=1}^{N}(l_i - \frac{1}{N}\sum_{i=1}^{N} l_i)^2$ as a measurement of the degree of load balance in the simulation. Figure 4 illustrates the average load variance value in the simulation period under a different user request rate (or, equivalently, the service load). When the system is lightly loaded, most requests are served locally. Some servers with a popular video get accessed more frequently than others. As the service load increases, frequent migration is observed and the service load is gradually balanced among servers. Thus, the load variance decreases as the service load increases. Again, REM has a much smaller load variance value than the traditional request migration scheme all the time in our simulation. The DASD dancing algorithm achieves a more balanced load than REM. However, there is little difference between REM and DASD, and their values almost overlap when the system load increases to near system capacity.

In the request migration process, if a server along the migration path has already reached its service capacity, it has first to migrate one of the in-service requests out to make room for the incoming request. In the mean time, the new request is held at the starting server waiting to be served, which results in service delay. For example, in Fig. 1, server 2 needs to migrate one request for video C to server 3 before it can accommodate the incoming request from server 1. At the same time, a new request for video A is held on server 1. Let us assume that each migration step takes the same amount of cost and the service delay can be measured by the number of intermediate servers along the migration path that have reached their service capacity. In this study, we use the service delay as the system cost to measure server request migration.

Figure 5 shows the migration cost in the simulation period for traditional migration, REM, and DASD dancing. As the system load increases, more and more servers reach their service capacity. Therefore, service delay increases. The DASD dancing algorithm migrates requests from overloaded servers to underloaded servers. It requires all servers, except the starting server, along a migration path not to reach their service capacity. Thus, the whole migration process can finish in one step. REM and the traditional scheme attempt to find the shortest migration path. Request migration is processed step by step from the end server to the starting server. We count the number of migration steps as the cost for that migration process. REM achieves the lowest cost among the three since it considers the server load status in the migration decision and avoids excess request migration in a lightly loaded system environment. The DASD dancing algorithm, on the other hand, only considers the load balance effect. It has a much higher cost than REM, especially when the system is lightly loaded.

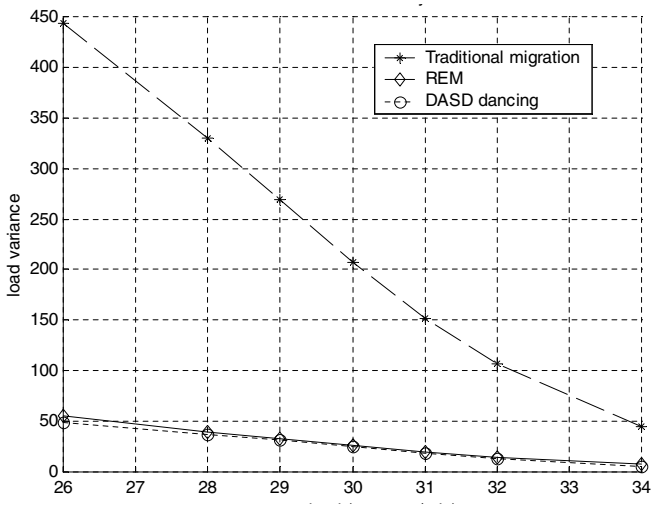

Fig. 4. Average load balance as a function of system load

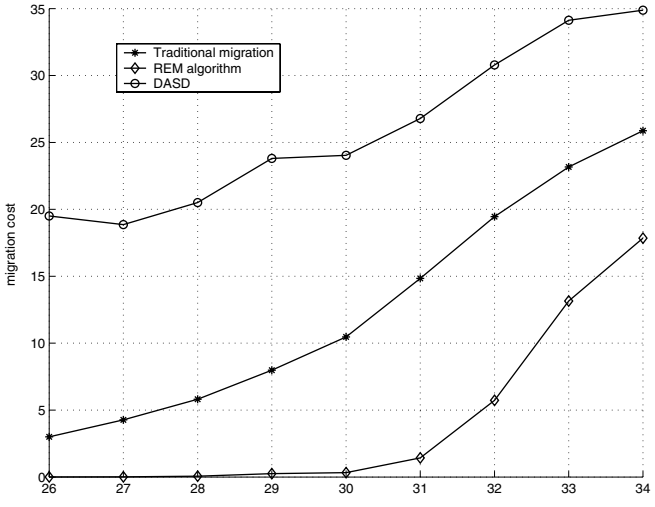

Fig. 5. Average service request delay as a function of system load

Finally, let us examine the performance of REM in VoD systems of different service capacity. Figures 6 and 7 show the failure rate and service delay of the system, in which each server can handle 40 concurrent user requests. Figures 8 and 9 show the metrics when each server can handle 240 requests at the same time. We see that REM can achieve the best performance in both cases. We can draw the conclusion that the performance of REM scales well under different system service capacities.

4 Mathematical modeling of VoD systems

We use a state matrix structure to describe a VoD system in this section. A state matrix stores the load status of each video server at a certain time instance. When a request migration process happens, the state matrix together with the connectivity graph, which describes the physical connection topology of video servers, is used to determine the shortest migration path. The length of the migration path can be obtained by calculating the "distance" (whose formal definition will be given

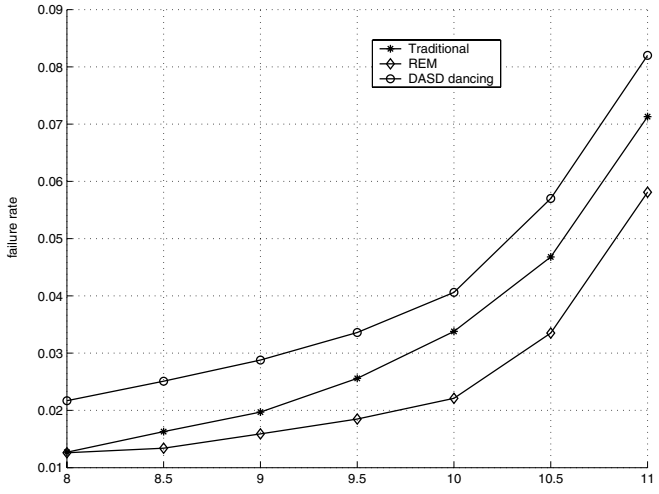

Fig. 6. Service failure rate as a function of system load (server capacity: 40 requests)

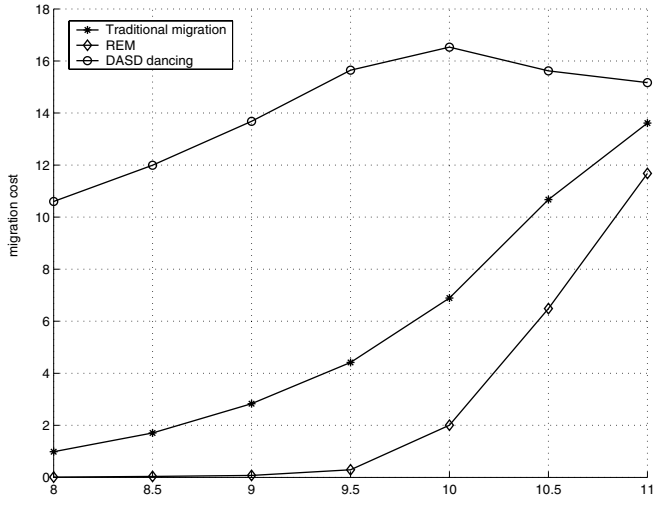

Fig. 7. Average service request delay as a function of system load (server capacity: 40 requests)

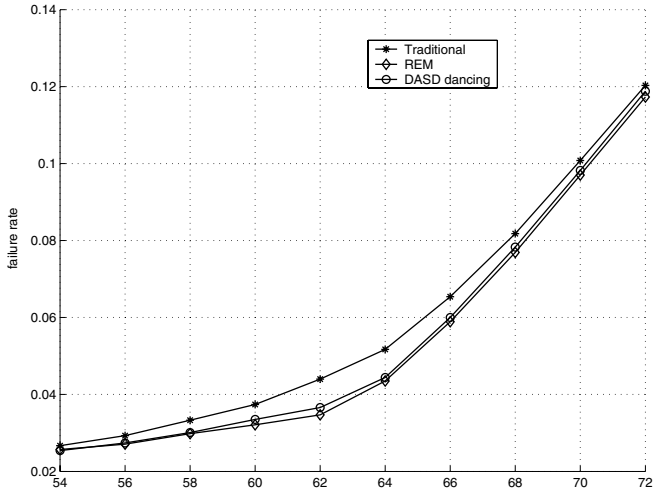

Fig. 8. Service failure rates as a function of system load (server capacity: 240 requests)

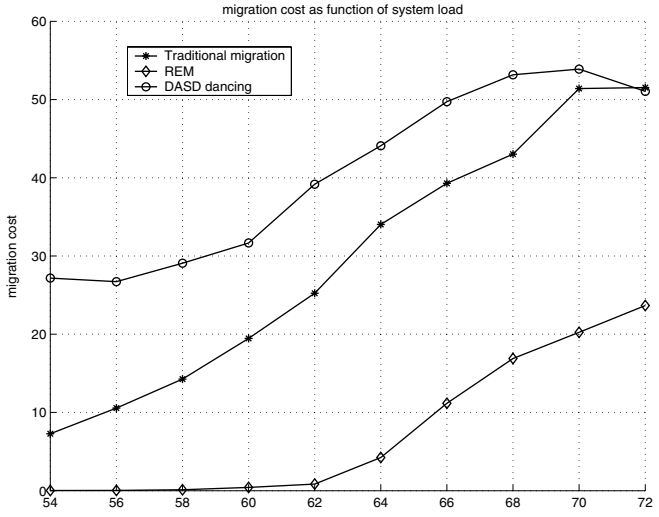

Fig. 9. Average service request delay as a function of system load (server capacity: 240 requests)

below) of the two state matrices before and after the migration process.

4.1 State matrix and migration path

When a new request is dispatched to a server (henceforth referred to as the starting server), it decides whether to accept the request locally or migrate it out based on the server's current load status. For the traditional migration algorithm, when the current load of a server reaches its capacity, request migration happens. For REM, request migration happens with a certain probability when the load is between Min_th and Max_th and surely happens when the load is above Max_th. If request migration should happen, the migration path is determined based on the connectivity graph of the VoD system and the service state of each server. The connectivity graph decides whether there is a path between two servers, while the service state decides if the server can really adopt that path. Therefore, the service state plays an important role in the migration path selection.

We use the following structure $H = (H_{i,j})_{N\times M}$ to represent the current load on servers, where $H_{i,j} \geq 0$ denotes the number of current in-service requests for video j on server i. We can interpret H as the state information of the VoD system. Before the VoD service starts, there is no in-service request on servers so that the initial state matrix is a zero matrix. Here, we use $O_{N\times M}$ to represent the initial state. A change of the state matrix occurs when a new request is accepted by a server. This request is either accepted locally or through a sequence of request migration steps. We use δ and τ to represent these two transforms, respectively. Next, we provide the definition of a *valid* state matrix.

Definition 4.1. *The valid state matrix must satisfy the following constraints: (1) If* $\Lambda_{i,j} = 0$*, then* $H_{i,j} = 0$ *for all time; (2)* $\sum_{j=1}^{M} H_{i,j} \leq L, \forall i = 1, ..., N$.

The first constraint implies that, in order for a server to serve the requested video title, the video title must have one copy cached on the server. The second one restricts the total number of in-service requests on a server not to exceed

its service capacity L. Every valid state matrix is *reachable* from the initial zero state through a sequence of δ and/or τ transforms. Two state matrices $H^{(1)}$ and $H^{(2)}$ are said to be *consecutive states* if one state can be directly reached from the other through a one-step δ or τ transform. From the system state viewpoint, when a new request is accepted by a server, there is a state transition between two consecutive state matrices, and the "difference" between them reflects the effect of the applied transform. To study the relation between the state matrix transform and the migration cost, we define the *distance* between two consecutive state matrices $H^{(1)}$ and $H^{(2)}$ as follows.

Definition 4.2. *The distance between two consecutive state matrices $H^{(1)}$ and $H^{(2)}$ is* $D(H^{(2)}, H^{(1)}) \stackrel{\text{def}}{=} \sum_{i,j} |H^{(2)}_{i,j} - H^{(1)}_{i,j}|$.

Now we can calculate the distance of consecutive state matrices under the δ or τ transform. Assume that the new request is dispatched to starting server S_{i_1}. As defined earlier, the δ transform means the new request is accepted locally on the starting server and no migration happens. Under this situation, the load of the starting server is increased by one and the load on all other servers remains unchanged. Thus, the distance is 1. When there is a request migration process, the relation between the migration path length $|MP|$ and the distance defined above can be expressed by the following lemma.

Lemma 4.3. *The migration path length can be calculated as*

$$|MP| = \frac{D(H^{(2)}, H^{(1)}) - 1}{2}, \tag{2}$$

where $H^{(1)}$ and $H^{(2)}$ are two consecutive state matrices.

Proof. We prove Lemma 4.3 by considering a general migration example as shown in Fig. 10. The migration path can be represented as $S_{i_1} \xrightarrow{V_{j_1}} S_{i_4} \xrightarrow{V_{j_2}} S_{i_2} \xrightarrow{V_{j_3}} S_{i_3}$. After the migration, the load on server S_{i_3} is increased by 1, which comes from one request for migrate-in video V_{j_3}. For all other servers in the migration path, their requests for the migrate-in video title are increased by 1 each, while their requests for the migrate-out video title are decreased by 1 each. The service load of these intermediate servers does not change after migration. Thus, the distance in this case is $(2 \times |MP| + 1)$. This completes the proof. □

The above property establishes the relation between the state matrix distance and the migration path length. If we consider that each step in a migration process takes the same delay, the service delay is proportional to the path length. Therefore, we can obtain the service delay through the calculation of the "difference" between consecutive state matrices.

4.2 State matrix space (SMS)

According to Lemma 4.3, we can calculate the length of a migration path through the distance between two consecutive state matrices before and after the migration. When a migration is needed, the scheduler constructs a dynamic migration tree with its root at the starting server. The topology of the tree is

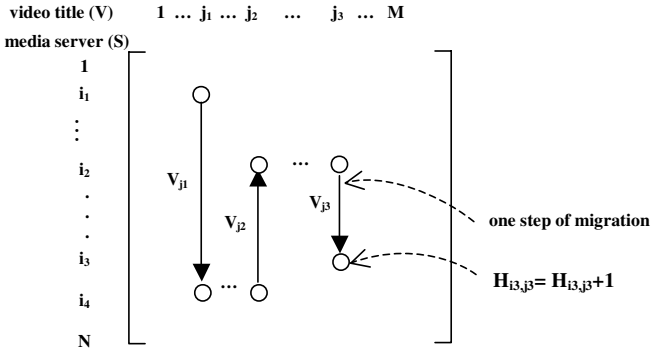

Fig. 10. Request migration with state matrix representation

based on the static connectivity graph among servers as well as the current state matrix of the VoD system. Servers at the same level of the tree are sorted in increasing order of their service load. The shortest migration path is obtained by a breadth-first search on the migration tree. The first available server under its service capacity is chosen as the end server in the migration path.

In the following performance analysis, it is assumed that the system is stable. That means that the performance metrics (i.e., failure rate and service delay) will converge to some distributions as long as the experiments are carried out enough times. Under this assumption, the goal of our analysis is to determine the statistical mean of these metrics. We propose a state matrix space (SMS) model to serve as the basic framework of our analysis. It is defined as follows.

Definition 4.4. *The state matrix space (SMS) is a mapping from all valid state matrices H to a directed graph $G_{SMS} = (V_H, E_H)$ such that*
(1) $V_H = \{H | H \text{ is valid}\}$;
(2) $E_H = \{(H^{(1)}, H^{(2)}) | H^{(1)} \xrightarrow{\delta} H^{(2)} \text{ or } H^{(1)} \xrightarrow{\tau} H^{(2)}\}$.

Figure 11 illustrates the SMS model. Note that SMS is a function of basic system parameters such as the number of servers, the number of video titles, and the video allocation matrix. Once these parameters are determined, SMS is uniquely specified. Each valid state matrix is mapped to a node in SMS. There is a root node in the space, which is the initial zero state

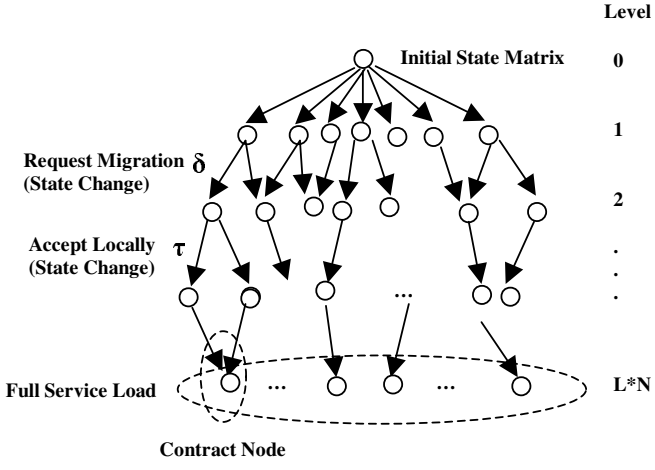

Fig. 11. Illustration of state matrix space (SMS)

matrix. Every other state can be reached from the initial state by a sequence of δ and/or τ transforms. The difference between SMS and a typical tree structure is that one state matrix can be directly reached from multiple state matrices. Thus the in-degree can be greater than 1, while the in-degree of a typical tree is equal to 1.

Consider two directly connected state matrices $H^{(1)}$ and $H^{(2)}$ in SMS, where $H^{(2)}$ is obtained from $H^{(1)}$ by either the δ or the τ transform. The system load is defined as the total number of in-service requests in the system. Then, the transform from $H^{(1)}$ to $H^{(2)}$ causes the system load to increase by 1. If two state matrices are reachable from the root through the same number of transform steps (i.e., they are at the same level), these two states have the same system load.

The relation between state matrices in SMS and the migration path selection is examined below. Upon receiving a new request, both the traditional migration and REM make the admission decision based on the current load status. If there is a need to do request migration, the migration path is determined based on the current state matrix and the connectivity matrix of the VoD system. It is proper to let the starting server choose the migration path. For the migration path illustrated in Fig. 1, consider the one-step migration server 1 $\stackrel{VideoB}{\longrightarrow}$ server 2. This migration is valid if and only if there is a connection between server 1 and server 2 and server 1 has an in-service request for video B as well. In general, the whole migration path is decided by the tuple (S_{start}, H, G).

Since the VoD system is stable, we can calculate the average delay and use it as the performance measurement. As mentioned earlier, we can use the migration path length to measure delay. We need to find out the average migration path length. There are two kinds of transforms from one state to another. Accordingly, there are two kinds of edges in SMS: the δ-edge and the τ-edge. For the δ-edge, which means the request is accepted locally without migration, the length of migration path is 0. For the τ-edge, which means certain request migration, we assign the cost to be the migration path length. Then, the average migration length is the average cost. We have the following property for the average path length.

Lemma 4.5. *Let $|\tau|$ denote the cost of a τ transform and $|E_\tau|$ the total number of τ-edges. The average path length is $\frac{\sum_{\tau \in E_\tau} |\tau|}{|E_\tau|}$.*

4.3 Request processing and state transition

The relationship between the request processing (including acceptance, rejection, and migration) and system parameters is examined in this section. Here we do not consider dynamic content update but assume the video content on each server will not change during the service period. Consider a video server system under a certain state determined by the video copy distribution (Λ), the current state matrix (H), and the video bookkeeping information (ST). The video bookkeeping information ST is an array of M elements. The jth element ST_j is a list of two tuples $(Server, Starting\ time)$ containing the information of current in-service requests on video V_j. Let $r(i,j)$ be the request dispatched to server i for video V_j. We can treat the video service as the state matrix transition process. Upon receiving $r(i,j)$, the request migration decision is made based on different scheduling algorithms. If the request is rejected by the system, there is an identical transition between the current state matrix and the next state matrix. If the request is accepted, either at the starting server or at some other server through migration, the current state matrix will change and the transition will depend on the migration path. When there is an end at server i for video V_j (denoted by $e(i,j)$), the state matrix will also change. We can summarize our discussion above into the following two lemmas about the video system modeling.

Lemma 4.6. *The video server system is defined by the tuple (Λ, H, ST, $Scheduling$), where Λ is the video copy allocation matrix, H is the current state matrix, ST is video bookkeeping array, and $Scheduling$ is the scheduling algorithm used, which can be either "traditional" or "REM."*

Lemma 4.7. *The video service is defined as a function $f : \mathcal{H} \mapsto \mathcal{H}$, where $\mathcal{H} \stackrel{\text{def}}{=} \{H | H \text{ is valid}\}$ is the set of all valid state matrices. Let $H^{(1)}$ denote the original state matrix and $H^{(2)}$ the mapping of $H^{(1)}$ under f. The mapping satisfies the following properties:*

1. If request $r(i^*, j^*)$ is rejected, then f is an identical mapping and $H^{(2)}_{i,j} = H^{(1)}_{i,j}$ for $\forall\, i, j$.

2. If request $r(i^*, j^*)$ is accepted directly on server i^*, then

$$H^{(2)}_{i,j} = \begin{cases} H^{(1)}_{i,j} + r(i^*, j^*) & \text{if } i = i^*,\ j = j^* \\ H^{(1)}_{i,j} & \text{otherwise} \end{cases}.$$

3. If request $r(i^*, j^*)$ causes migration along the path $\stackrel{V_{j^*}}{\longrightarrow} i^* \stackrel{V_{j_1}}{\longrightarrow} i_1 \longrightarrow \cdots \longrightarrow i_{k-1} \stackrel{V_{j_k}}{\longrightarrow} i_k$, then

$$H^{(2)}_{i,j} = \begin{cases} H^{(1)}_{i,j} + r(i^*, j^*) & \text{if } i = i^*,\ j = j^*;\ i = i_1, \\ & \quad j = j_1; \ldots; i = i_k,\ j = j_k \\ H^{(1)}_{i,j} - r(i^*, j^*) & \text{if } i = i^*,\ j = j_1;\ i = i_1, \\ & \quad j = j_2; \ldots; i = i_{k-1},\ j = j_k \\ H^{(1)}_{i,j} & \text{otherwise} \end{cases}.$$

4. If service end $e(i^*, j^*)$ is on server i^* for video V_{j^*}, then

$$H^{(2)}_{i,j} = \begin{cases} H^{(1)}_{i,j} - e(i^*, j^*) & \text{if } i = i^*,\ j = j^* \\ H^{(1)}_{i,j} & \text{otherwise} \end{cases}.$$

5 Performance analysis for request migration

To facilitate the analysis of the request migration process and the computation of system metrics under different scheduling algorithms, in the previous section we presented the state matrix structure to characterize the server system state at a certain time. Based on the proposed model, we developed two approaches to calculate the failure rate and service delay. The first approach was derived based on the topology calculation in the state matrix space (SMS). In the second approach, the video service was modeled as the state transition between two state matrices. We studied the transition rules under the events of request rejection, acceptance, and ending. An event-driven transition model was built to simulate the real video service. Both approaches were verified by numerical experiments.

5.1 Approach I: Analysis using topology calculation in SMS

According to the video service model, the state matrix transition depends on the migration path when a request migration happens. Under the assumption that a request migration happens with an incoming request, we consider the migration path. Recall that the state matrix stores the load status of each video server at a certain time instance. When a request migration process happens, the state matrix together with the connectivity graph, which describes the physical connection topology of video servers, is used to determine the shortest migration path.

We first identify the property of a valid migration path. Based on the property, we present a way to calculate the failure rate and service delay for the traditional migration algorithm and REM. We denote by $MP\{S_{i_1}, V_{j_1}, \cdots, S_{i_k}, V_{j_k}, S_{i_{k+1}}; k\}$ the following migration path obtained from the connectivity matrix: $S_{i_1} \xrightarrow{V_{j_1}} S_{i_2} \longrightarrow \cdots \longrightarrow S_{i_k} \xrightarrow{V_{j_k}} S_{i_{k+1}}$.

Definition 5.1. *The migration path MP is valid if and only if there exists a tuple (S_{start}, H, G) such that there exists a request migration and MP is chosen as the shortest path.*

The following lemma illustrates the characteristics of a valid migration path.

Lemma 5.2. *If a migration path MP satisfies the following constraints, it is a valid path.*
1. S_{i_m} and $S_{i_{m+1}}$ must have a copy of V_{j_m}, $m = 1, 2, \ldots, k$.
2. S_{i_1}, S_{i_2},..., $S_{i_{k+1}}$ are different from each other.
3. V_{j_1}, V_{j_2},..., V_{j_k} are different from each other.
4. S_{i_m} cannot have a copy of V_{j_1}, V_{j_2},$\cdots$, $V_{j_{m-2}}$, $m = 3, 4, \ldots, k+1$.

Proof. The first constraint is obvious. The two adjacent servers along a path must have the same copy of a video for request migration. Otherwise, the migration process cannot happen. The second and third rules ensure the migration path is cycle free. Suppose that we have a path in which server S_u appears twice, like $\cdots \to S_{i_l} \to S_u \to \cdots \to S_u \to S_{i_m} \to \cdots$. Then, we can replace this path with $\cdots \to S_{i_l} \to S_u \to S_{i_m} \to \cdots$, which is shorter than the original one. It is similar when applied to video duplication on the migration path. The fourth rule eliminates the probability that there exists a "shortcut" on the path. Suppose we have the following migration path: $S_{i_1} \xrightarrow{V_{j_1}} S_{i_2} \xrightarrow{V_{j_2}} S_{i_3}$. If server S_{i_3} also has a copy of V_{j_1}, then we can skip S_{i_2} and make a direct migration from S_{i_1} to S_{i_3}: $S_{i_1} \xrightarrow{V_{j_1}} S_{i_3}$. This new path is shorter than the original one.

Now, we prove if a migration path satisfies all the above constraints, there exists a tuple (S_{start}, H, G) such that this path is chosen as the shortest path. We construct the state matrix H and the starting server to satisfy the following conditions: (1) $h_{i_m,j_m} = L$, $m = 1, 2, ..., k$; (2) $\sum_{j=1}^{M} h_{i_{k+1},j} < L$; (3) $h_{i,j} = 0$, for all other items; (4) $S_{start} = S_{i_1}$. Under these conditions, when a new request is dispatched to S_{i_1}, which has reached its service capacity, request migration happens in the traditional migration scheme as well as in REM. We construct H, and the migration path $MP\{S_{i_1}, V_{j_1}, \cdots, S_{i_k}, V_{j_k}, S_{i_{k+1}}; k\}$ is the only path starting from S_{i_1}, which is also the shortest path to be chosen. Thus, it is a valid migration path. Finally, we prove that H is a valid state matrix. We can construct the request sequence that contains L requests for each video title $V_{j_1}, V_{j_2}, \ldots, V_{j_k}$. When a new request for video V_{j_m}, $m = 1, 2, \ldots, k$ arrives, it is dispatched to server S_{i_k} and accepted locally. After $L \times k$ steps, we get state matrix H. Thus, H can be reached from the initial state matrix by a sequence of δ transforms. This completes the proof. □

Lemma 5.2 provides a way to enumerate all valid migration paths from connectivity matrix G. Each τ-edge in SMS corresponds to a migration path. Since the tuple (S_{start}, H, G) determines the migration path, the total number of τ-edges is equal to the total number of such tuples. However, given a tuple (S_{start}, H, G), it is difficult to find out what the migration path is, if there exists a migration. Here, we start from each valid migration path. Given a path, we calculate the number of tuples corresponding to that path. This method is valid since SMS contains all possible combinations of state matrices and the starting server for each valid migration path. The following lemma illustrates this feature.

Lemma 5.3. *For each valid migration path $MP\{S_{i_1}, V_{j_1}, \cdots, S_{i_k}, V_{j_k}, S_{i_{k+1}}; k\}$, SMS contains all state matrices H such that MP is chosen as the shortest path under tuple (S_{i_1}, H, G).*

Proof. According to the definition of SMS, it contains all valid state matrices. For a valid migration path, if it is chosen as the shortest path under some tuple (S_{i_1}, H, G), then state matrix H must be valid. Otherwise, the system cannot reach this state from its initial zero matrix. Thus, SMS must contain H. This completes the proof. □

In the request migration process, if a server along the migration path has already reached its service capacity, then it first has to migrate one of the in-service requests out to make room for the incoming request. In the mean time, the new request is held at the starting server waiting to be served, which results in service delay. For example, in Fig. 1, server 2 needs to migrate one request for video C to server 3 before it can accommodate the incoming request from server 1. At the same time, a new request for video A is held on server 1. Let us assume that each migration step takes the same amount of time. The service delay can then be measured by the number of intermediate servers along the migration path that have reached their service capacity. Let $\Delta(\mu)$ and $\overline{|MP|}(\mu)$ denote the average delay and the migration path length, respectively, at a request arrival rate μ. Under the above definition, the relationship between delay and path length is given by

$$\Delta(\mu) = \overline{|MP|}(\mu) - 1. \tag{3}$$

Now let us consider the service delay and the failure rate in the traditional migration algorithm. Assume that the user request rate is μ. For a valid k-step migration path $MP\{S_{i_1}, V_{j_1}, \cdots, S_{i_k}, V_{j_k}, S_{i_{k+1}}; k\}$, we want to find the number of state matrices H that corresponds to MP (i.e., MP is chosen as the shortest path under tuple (S_{i_1}, H, G)). According to Lemma 5.3, SMS contains all those state matrices. Thus, we can get all the τ-edges that correspond to MP. The

load distribution in H must satisfy the following constraints.
1. The first k servers in the path have reached service capacity, $\sum_{j=1}^{M} h_{i_m,j} = L,\ m = 1, 2, ..., k$.
2. There must be in-service requests for migrated-out video files, $h_{i_m,j_m} > 0,\ m = 1, 2, \ldots, k$.
3. There is service space on the end server, $\sum_{j=1}^{M} h_{i_{k+1},j} < L$.
4. There is no migration path existing on H whose length is less than k.

In the following derivation, we use $C_i^j \stackrel{\text{def}}{=} \frac{i!}{(i-j)!j!}\ (0 \le j \le i)$ to denote the combinatorial calculation. Given the above constraints, we can calculate the number of corresponding state matrices at request rate μ as

$$\Gamma\{MP\} = \begin{cases} 0, \ \mu T_s < kL \\ A - B, \ \mu T_s \ge kL \end{cases}, \tag{4}$$

where

$$A = \left[\sum_{l=0}^{L-1} C_{l+K-2}^{l}\right]^k \left[\sum_{l=0}^{L-1} C_{l+K-1}^{l}\right] \left[\prod_{i \in C1} C_{l_i+K-1}^{l_i}\right], \tag{5}$$

$$C1 \stackrel{\text{def}}{=} \left\{ i \Big| \sum_{i \notin i_1,\ldots,i_{k+1}} l_i = \mu T_s - (kL + l_{i_{k+1}}) \right\},$$

and

$$B = \frac{\left[\sum_{l=0}^{L-1} C_{l+K-2}^{l}\right]^{k-1}}{\left[\sum_{l=0}^{L} C_{l+K-1}^{l}\right]^{k-1}} \Gamma\{S_{i_1}, V_{j_1}, *; 1\} + \cdots + \frac{\sum_{l=0}^{L-1} C_{l+K-2}^{l}}{\sum_{l=0}^{L} C_{l+K-1}^{l}} \Gamma\{S_{i_1}, V_{j_1}, \cdots, S_{i_k-1}, V_{j_k-1}, *; k-1\} \tag{6}$$

In (6), $*$ represents any server except those on migration path MP such that it constructs a valid migration path. In (4), A represents constraints 1 to 3 and B represents constraint 4. For one-step migration paths, we can rewrite Eqs. 5 and 6 as

$$A = \left[\sum_{l=0}^{L-1} C_{l+K-2}^{l}\right] \left[\sum_{l=0}^{L-1} C_{l+K-1}^{l}\right] \left[\prod_{i \in C2} C_{l_i+K-1}^{l_i}\right], \tag{7}$$

$$C2 \stackrel{\text{def}}{=} \left\{ i \Big| \sum_{i \notin i_1,i_2} l_i = \mu T_s - (L + l_{i_2}) \right\},$$

$$B = 0. \tag{8}$$

We add all $\Gamma(MP)$ together to get the total number of τ-edges at request rate μ, i.e.,

$$|E_\tau|(\mu) = \sum_{MP \text{is valid}} \Gamma(MP). \tag{9}$$

With Eqs. 4–9 and Lemma 4.5, we can obtain the average migration path length under request rate μ. The average migration delay can be obtained by Eq. 3.

Next, we calculate the failure rate $F(\mu)$ under request rate μ. Let us first consider the situation where the system load is below service capacity, i.e., $\mu T_s \le LN$. Since each server has the same access probability, we only check whether a server can accept a new request. Here, "accept" means the new request is accepted either locally by that server or through request migration. In the traditional migration scheme, if a server reaches its capacity, it cannot serve a new request locally. Given state matrix H, the number of servers that have not reached their service capacity is equal to the number of δ-edges from that server. For each migration path MP, there is a τ-edge from the corresponding servers. We can calculate the failure rate as

$$F(\mu) = \frac{\sum_{m=0}^{\lfloor \frac{\mu T_s}{L} \rfloor} \left[(N-m) C_N^m (C_{L+K-1}^{L})^m \prod_{n \in C3} C_{l_{i_n}+K-1}^{l_{i_n}} \right]}{N \prod_{\sum_{i=1}^{N} l_i = \mu T_s} C_{l_i+K-1}^{l_i}} + \frac{|E_\tau|(\mu)}{N \prod_{\sum_{i=1}^{N} l_i = \mu T_s} C_{l_i+K-1}^{l_i}} \tag{10}$$

$$C3 \stackrel{\text{def}}{=} \left\{ n \Big| \sum_{n=1}^{N-m} l_{i_n} = (\mu T_s - mL),\ l_{i_n} < L \right\},$$

where $|E_\tau|(\mu)$ denotes the number of τ-edges at request rate μ. When the system load is above the total system capacity, the failure rate can be represented by

$$F(\mu) = F\left(\left\lfloor \frac{LN}{T_s} \right\rfloor\right) + \left(1 - \frac{LN}{\mu T_s}\right). \tag{11}$$

REM differs from the traditional migration scheme when request migration begins. When a new request is dispatched to a video server, REM compares the current service load with preset thresholds Min_th and Max_th ($Min\_th < Max\_th$) and decides whether request migration is needed. When the service load is below Min_th, migration is not needed. When the load is between Min_th and Max_th, migration occurs with a certain probability. When the load is above Max_th, migration happens with probability 1. This decision only applies to new requests from a client but not those requests shifted among servers. Accordingly, we need to change the migration threshold on the starting server. Here, we use a traditional migration scheme to replace REM. In this migration scheme, the migration threshold L_{alt} obtained from Eq. 12 is no longer equal to the service capacity (cf. Fig. 2 and Eq. 1). We have

$$L_{\text{alt}} = Max\_th - p_{\max}(Max\_th - Min\_th)/2\,. \tag{12}$$

With this transform, all equations used in calculating the delay and the failure rate in the traditional migration scheme can be applied to REM, with the migration threshold for the starting server set to L_{alt}.

Let us discuss the complexity of the SMS-based approach. According to Eqs. 4–8, the SMS-based approach is a recursive method in that the counting of state matrices for valid migration paths of length k ($1 \le k \le N-1$) is based on the results for valid migration paths of length from 1 to $k-1$. The complexity of the SMS-based approach depends on the calculation

of valid migration paths of length k. According to constraints 2 and 3 in Lemma 5.2, for given k, we have $P_N^{k+1}P_M^k$ candidates (where P_i^j denotes the permutation of j elements from a set of i elements and M is the number of video objects). We can see that the time complexity is exponential. However, the SMS-based approach is not a real-time scheduling algorithm for the VoD system. Instead, we use it as an offline algorithm to demonstrate the performance improvement of REM compared to the traditional migration. The comparison is shown in Figs. 12 and 13. In the following section, we propose a more efficient approach using the state transition model of the request processing discussed in Sect. 4.3.

5.2 Approach II: Analysis using state transition

From a statistical point of view, the probability for an incoming request for video V_j equals the popularity of that video. Moreover, the request is dispatched randomly to the server with copy of V_j. We can model the request as $r(i,j) = \phi_{uj}\Lambda_{i,j}$. According to the analysis in Sect. 4.3, the calculation of the failure rate and service delay depends on the state matrices before and after a one-step state transition. Since a state transition happens when there is some incoming request or service completion at a video server, we need to update the current state matrix in these two events accordingly. For the four cases shown in Lemma 4.7, it is easy to calculate the resulting state matrix when the incoming request is directly accepted or there is a service end. When request migration happens, we need to find the shortest migration path and then update the server status along that path. Another important issue in state matrix transition is the scheduling algorithm adopted by the system, which decides when the migration should happen. In what follows, we discuss how to find the shortest path and then study state matrix transition under the traditional migration algorithm and REM. The following lemma states the property of request migration.

Lemma 5.4. *If a request migration happens under request $r(i^*, j^*)$, let H denote the current state matrix; then we have the following: (1) Server i can migrate out video V_j if and only if $H_{i,j} \geq r(i^*, j^*)$; (2) the last server i on the migration path satisfies $L - \sum_{j=1}^{M} H_{i,j} \geq r(i^*, j^*)$.*

The above property tells us that for server i to migrate video V_j, it must store a copy of V_j and, at the same time, have an in-service request for V_j. In other words, service migration depends on the video allocation matrix (Λ) and current state matrix (H). We further define a companion matrix H^c from H, with $H^c_{i,j}$ indicating the "ability" of server i to migrate out video V_j.

Definition 5.5. *If the incoming request is $r(i^*, j^*)$, the companion matrix H^c is derived from state matrix H as* $H^c_{i,j} = \begin{cases} 1, \textit{ if } H_{i,j} \geq r(i^*, j^*) \\ 0, \textit{ otherwise} \end{cases}$.

Next, we calculate the connectivity matrix G of video servers. Let $(G^{(k)}_{i,j})_{N*N}$ denote the k-step connectivity matrix, where $G^{(k)}_{i,j} \in \{0,1\}$ and $G^{(k)}_{i,j} = 1$ if the shortest path from server i to j is of k steps. To calculate $G^{(k)}$, we define a new matrix operator $\otimes$ that is similar to matrix production except that it replaces the scalar multiplication and addition with logical operations "AND" and "OR." Then $G^{(k)}, k = 1, 2, \ldots, N-1$ can be calculated as $G^{(k)} = (H^c \otimes \Lambda^T)^k - \sum_{i=0}^{k-1} G^{(i)}$, where $G^{(0)} = I$ is the unit matrix and $G^{(k)}_{i,j} = 0$ for $i = j$. The server connectivity matrix G can be derived from $G^{(k)}, k = 1, 2, \ldots, N-1$ as $G = \sum_{k=1}^{N-1} kG^{(k)}$. Assume that the shortest migration path under request $r(i^*, j^*)$ is $\xrightarrow{V_{j^*}} i^* \xrightarrow{V_{j_1^*}} i_1^* \longrightarrow \cdots \longrightarrow i_{k-1}^* \xrightarrow{V_{j_k^*}} i_k^*$. Given that the path is the shortest among all the viable paths, and according to Lemma 5.4, we can find the last server on the path

$$i_k^* = \min_{i_k} \Big\{ G_{i^*,i_k} | L - \sum_{j=1}^{M} H_{i_k,j} \geq r(i^*, j^*), G_{i^*,i_k} > 0, \; i_k \neq i^* \Big\}. \tag{13}$$

Since the path is the shortest, all the intermediate servers already reach their service capacity. Using this property, we can find all the servers along the path

$$i_{k-1}^* = \min \left\{ i | G_{i,i_k^*} = 1, G_{i^*,i} = G_{i^*,i_k^*} - 1 \right\}, \ldots,$$
$$i_1^* = \min \left\{ i | G_{i,i_2^*} = 1, G_{i^*,i} = 1 \right\}. \tag{14}$$

All the migrate-in and migrate-out video files can be found using Lemma 5.4 as

$$j_1^* = \min \left\{ j | H^c_{i^*,j} = 1, \Lambda_{i_1^*,j} = 1 \right\}, \ldots,$$
$$j_k^* = \min \left\{ j | G_{i_{k-1}^*,j} = 1, \Lambda_{i_k^*,j} = 1 \right\}. \tag{15}$$

Here we choose the server and video with the smallest index if there are multiple options.

The request scheduling algorithm decides when request migration should happen. We discuss state matrix transition rules under the traditional migration algorithm and REM below. The traditional migration algorithm uses "last-minute" migration. A new request is accepted directly at the starting server until it reaches service capacity. Then the server starts to migrate out the request. If there is no viable path, the request is rejected. The four cases stated in Lemma 4.7 can be directly applied to the state transition in traditional migration schemes.

REM uses two thresholds to decide when request migration should happen. If the current load is less then the lower threshold, the incoming request is accepted directly at the starting server. If the load is between the lower and upper thresholds, the request is migrated in a certain probability, which is a linear function of server load. If the load is greater than the upper threshold, the request is migrated out in probability 1. Let $H^{(21)}$ and $H^{(22)}$ denote the result state matrix when the request is accepted directly and when there is migration. We need to change Lemma 4.7 as follows.

Lemma 5.6. *The mapping f defined in Lemma 4.7 satisfies the following properties under REM:*
1. If $L - r(i^*, j^*) < \sum_{j=1}^{M} H^{(1)}_{i^*,j}$ and there is no viable migration path, then $H^{(2)} = H^{(1)}$.

2. If $\sum_{j=1}^{M} H^{(1)}_{i^*,j} < Min\_th$ or $Min\_th \leq \sum_{j=1}^{M} H^{(1)}_{i^*,j} \leq$

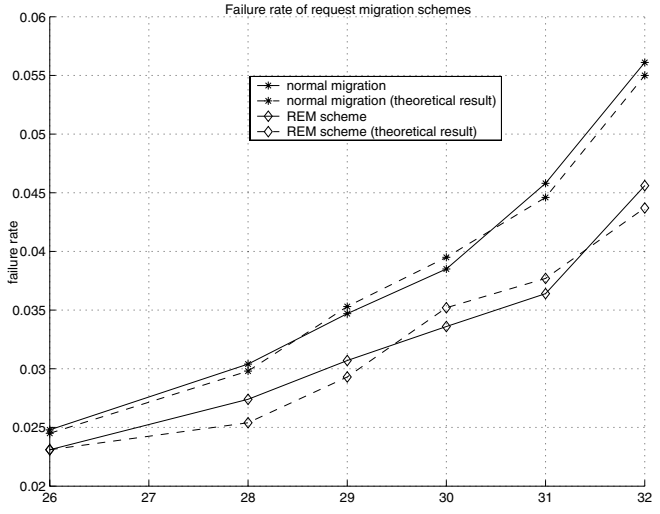

Fig. 12. Service failure rate as a function of system load (approach I); *solid line* – numerical results, *dotted line* – theoretical results

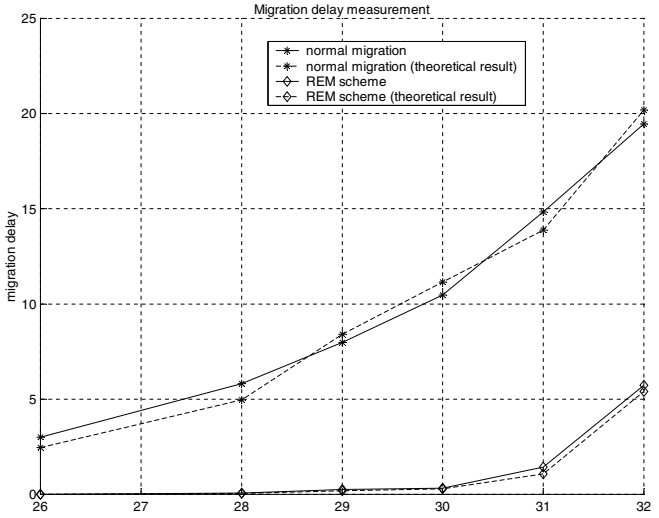

Fig. 13. Average service delay as a function of system load (approach I); *solid line* – numerical results, *dotted line* – theoretical results

$L - r(i^*, j^*)$ but there is no viable migration path, then $H^{(2)} = H^{(21)}$.
3. If $Min\_th \le \sum_{j=1}^{M} H_{i^*,j}^{(1)} < Max\_th$ and there is a viable migration path, then $H^{(2)} = (1-p_i)H^{(21)} + p_i H^{(22)}$.
4. If $Max\_th \le \sum_{j=1}^{M} H_{i^*,j}^{(1)}$ and there is a viable migration path, then $H^{(2)} = H^{(22)}$.
5. If service end $e(i^*, j^*)$ on server i^* for video V_{j^*}, then

$$H_{i,j}^{(2)} = \begin{cases} H_{i,j}^{(1)} - e(i^*, j^*) & \text{if } i = i^* \text{ and } j = j^* \\ H_{i,j}^{(1)} & \text{otherwise} \end{cases}.$$

Let us discuss the time complexity of the state-transition-based approach. First, we can calculate $G^{(k)}$ recursively as

$$G^{(k)} = (H^c \otimes \Lambda^T)^k - \sum_{i=0}^{k-1} G^{(i)} = (H^c \otimes \Lambda^T) \otimes G^{(k-1)}. \quad (16)$$

According to Eq. 16, we only need to calculate $H^c \otimes \Lambda^T$ once and save the result for later use. The calculation of $G^{(k)}$ involves matrix operation $\otimes$. Since the $\otimes$ operation is composed of logical "AND" and "OR" and the elements of both $H^c \otimes \Lambda^T$ and $G^{(k-1)}$ are either 0 or 1, it can be implemented very efficiently using bit operations. Second, $G^{(k)}$, $1 \le k \le N-1$, contains all server pairs with the shortest distance between each other being k-steps. If $G_{i,j}^{(k)} = 1$, then $G_{i,j}^{(t)} = 0$ for any t such that $t \ne k$. After we obtain $G_{i,j}^{(k)} = 1$, we can immediately set $G_{i,j}^{(t)} = 0$ for all $t > k$, which will save a lot of computation. Last, the time complexity of Eq. 13 is $O(NM)$. Equations 14 and 15 both have time complexity of $O(N^2)$. The analysis shows that the state-transition-based approach is more efficient than the SMS-based approach.

5.3 Verification of theoretical derived results

In this section, we apply the two analytical approaches derived in Sects. 5.1 and 5.2 to traditional migration and REM

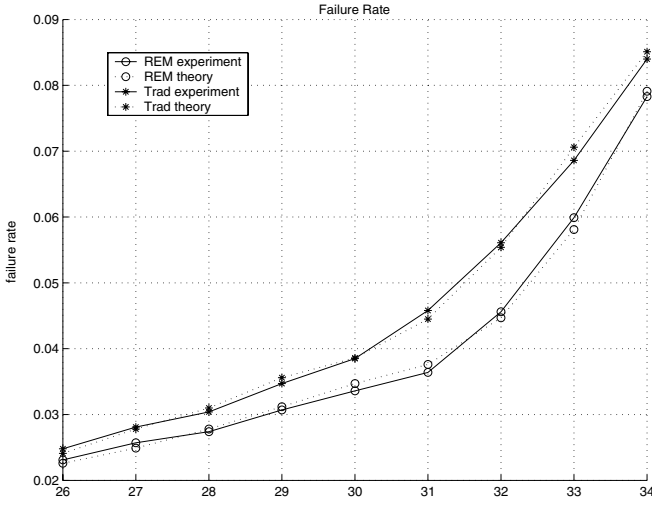

Fig. 14. Service failure rate as a function of system load (approach II); *solid line* – numerical results, *dotted line* – theoretical results

and evaluate their performance in terms of the failure rate and service delay. To compare with the numerical results given in Sect. 3.2, we use the same system configurations here. The server capacity is set to 120 concurrent requests. As for the parameters of the REM scheme, the lower and the upper thresholds are set to 80 and 105, respectively, and $p_{\max}$ is set to 0.8.

First, let us compare theoretical results obtained by approach I with numerical results. Figure 12 shows the service failure rate versus the user request rate for the traditional migration scheme and the REM scheme. We see from the figure that the service failure rate increases as the average request arrival rate μ increases, which is an indicator of the increase of the service load. When the system is lightly loaded (say, $\mu < 30$), both schemes achieve good performance with a failure rate less than 4%. When the system load is around its service capacity ($30 \le \mu \le 32$), REM reduces the service failure rate significantly in comparison with the traditional request migration scheme. Theoretical results are plotted in

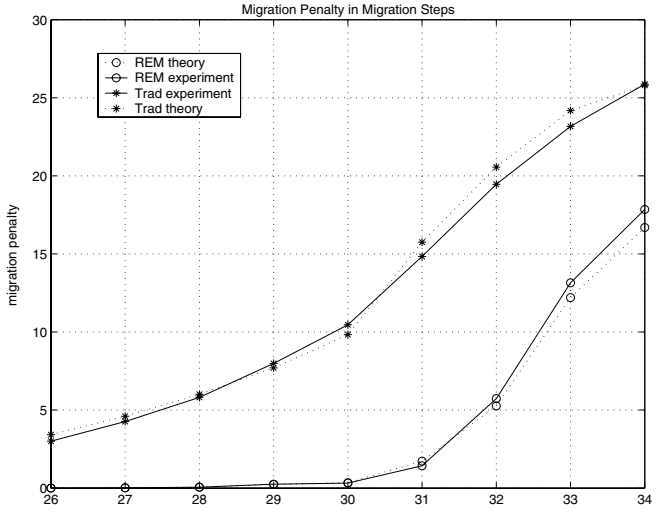

Fig. 15. Average service delay as a function of system load (approach II); *solid line* – numerical results, *dotted line* – theoretical results

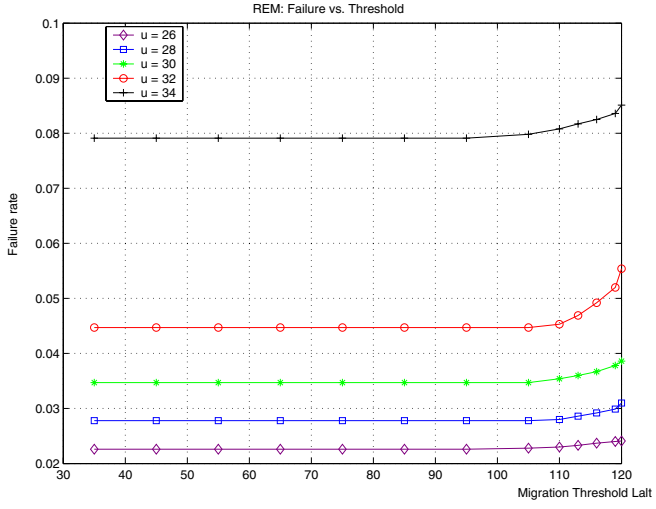

Fig. 16. REM parameter effect on failure rate

dashed lines, which match well with experimental curves in both cases. The error is less than 10% of experimental results. Figure 13 shows the average service delay for the traditional migration scheme and REM during the simulation period. Here we vary the request rates from 26 to 32 per minute. As the system load increases, more and more servers reach their service capacity and, as a result, the service delay increases. For all system loads, delay in REM is much smaller than that in the traditional migration scheme. Theoretical results match well with experimental ones in service delay, too.

Next, we examine results obtained by approach II. Figures 14 and 15 show the service failure rate and service delay under different user request rates, respectively. Theoretical results are plotted in dotted lines, which also match well with experimental curves in both cases. The error is less than 5% with respect to experimental results.

Last, we consider the effect of REM parameters on the service failure rate and service delay. There are three important parameters in REM configuration: the upper threshold (Max_th), the lower threshold (Min_th), and the maximum

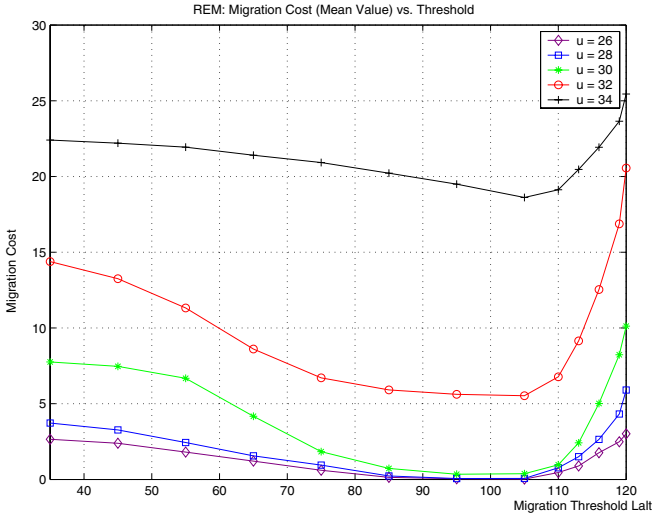

Fig. 17. REM parameter effect on service delay

migration probability ($p_{\max}$). From a statistical viewpoint, the impact of parameters is primarily on the setup of migration thresholds. We use Eq. 12 to represent the "virtual" threshold used in REM. The failure rate and service delay as a function of L_{alt} are shown in Figs. 16 and 17, respectively. The request rate ranges from 26 to 34 requests per minute, as shown by the different styled curves.

We see from these two figures that, for small L_{alt} value, there is no change in the failure rate while service delay increases as L_{alt} decreases. This is because migration happens more frequently when L_{alt} is small, which will increase the service delay but contribute little to the failure rate. Thus, a certain degree of migration can reduce the failure rate while maintaining relatively low service delay. The failure rate will become "saturated" when more migration requests are processed. On the other hand, overmigration will result in longer service delays. When L_{alt} increases to a larger value, both the failure rate and service delay increase. When L_{alt} equals 120, which is the service capacity of a single server, REM degenerates to the traditional "last-minute" migration algorithm. As the threshold increases, the system becomes unbalanced and one-step migration decreases since more and more servers along the migration paths have already reached their full service capacity. Consider the performance of the failure rate and service delay. We should choose Min_th, Max_th, and $p_{\max}$ to make L_{alt} around 100.

6 Conclusion and future work

The video scheduler plays an important role in video-on-demand (VoD) systems. We proposed the random early migration (REM) scheme to reduce the service failure rate and service delay. REM considers the relation between load balance and system metrics under different service load situations. When the system is lightly loaded, servers have enough service space to accommodate incoming requests and load balance contributes little to the improvement of the failure rate. Moreover, excess request migration increases the system cost. Compared to the DASD algorithm, which only considers the load balance issue, REM reduces the migration cost by

accepting requests locally when the system is lightly loaded. REM migrates requests progressively as the system load increases. Compared with the traditional "last-minute" migration scheme, REM redistributes the service load among servers through request migration at a much earlier stage. As a result, REM achieves a more balanced load distribution and a lower failure rate than the traditional migration scheme. To facilitate the analysis of the request migration process and the computation of system metrics under different scheduling algorithms, we presented the state matrix structure to characterize the server system state at certain times and developed two approaches to calculate the failure rate and service delay. Both approaches were verified by numerical experiments, and it was shown that REM outperforms traditional migration in both the failure rate and service delay.

In this paper, we consider server scheduling under a normal playback environment, which is the most frequently used function in VoD service. In a full-fledged VoD system, clients may issue interactive operations such as pause, fast forward and rewind with the video server. Those interactive operations usually have different bandwidth requirements compared to normal playback. Moreover, the duration of those operations is not known a priori. In the future, we would like to study the server scheduling problem in such a complicated context and try to integrate REM with interactive VoD service. The deployment of REM in a distributed video server environment is another area of our interest.

Acknowledgements. We thank the referees for their constructive comments and suggestions. This study has been funded by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the National Science Foundation.

References

1. Wolf JL, Yu PS, Shachnai H (1997) Disk load balancing for video-on-demand systems. Multimedia Syst 5:358–370
2. Tsao S, Chen MC, Ko M, Ho J, Huang YM (1999) Data allocation and dynamic load balancing for distributed video storage server. J Vis Commun Image Represent 10:197–218
3. Mundur P, Simon R, Sood AK (2004) End-to-end analysis of distributed video-on-demand systems. IEEE Trans Multimedia 6:129–141
4. Tewari R, Dias D, Mukherjee R, Vin H (1995) High availability in clustered video server. Technical Report RC20108, IBM TJ Watson Research Center
5. Tewari R, Mukherjee R, Dias D, Vin H (1996) Design and performance tradeoffs in clustered video servers. In: 3rd IEEE international conference on multimedia computing and systems, pp 144–150
6. Bolosky WJ, Barrera JS, Draves RP, Fitzgerald RP, Gibson GA, Jones MB, Levi SP, Myhrvold RF, Rashid NP (1996) The tiger video file server. In: Proceedings of the 6th international workshop on network and operating system support for digital audio and video
7. Lee JYB (1998) Parallel video servers: a tutorial. IEEE Multmedia 5:20–28
8. Chen P, Lee E, Gibson G, Katz R, Patterson D (1994) Raid: high-performance, reliable secondary storage. ACM Comput Surv 26:145–185
9. Little TDC, Venkatesh D (1994) Probability-based assignment of videos to storage devices in a video-on-demand system. Multimedia Syst 2:280–287
10. Serpanos DN, Georagiadis L, Bouloutas T (1996) Mmpacking: a load and storage balancing algorithm for distributed multimedia servers. Technical Report RC20410, IBM TJ Watson Research Center
11. Bisdikian CC, Patel BV (1995) Issues on movie allocation in distributed video-on-demand systems. In: Proceedings of the international conference on communications (ICC '95), pp 250–255
12. Venkatasubramanian N, Ramanthan S (1997) Load management in distributed video servers. In: Proceedings of the 17th international conference on distributed computing systems, pp 528–535
13. Lougher P, Lougher R, Shepherd D, Pegler D (1996) A scalable hierarchical video storage architecture. In: SPIE conference on multimedia computing and networking, pp 18–29
14. Guo J, Taylor PG, Zukerman M, Chan S, KS Tang, Wong EWM (2003) On the efficient use of video-on-demand storage facility. In: IEEE international conference on multimedia and expo (ICME'03), 2:329–332
15. Doganata YN, Tantawi AN (1994) Making a cost-effective video server. IEEE Multimedia 1:22–30
16. Schaffa F, Nussbaumer J-P (1995) On bandwidth and storage tradeoffs in multimedia distribution networks. In: INFOCOM '95, 14th annual joint conference of the IEEE computer and communications societies, 3:1020–1026
17. Barnett SA, Anido, GJ (1996) A cost comparison of distributed and centralized approaches to video-on-demand. IEEE J Select Areas Commun 14:1173–1183
18. Chan S-HG, Tobagi F (2001) Distributed servers architecture for networked video services. IEEE/ACM Trans Network 9:125–136
19. L VOK, Liao W, Qiu X, Wong EWM (1996) Performance model of interactive video-on-demand systems. IEEE J Select Areas Commun 14:1099–1109
20. Shu W, Wu M (2004) Resource requirements of closed-loop video delivery services. IEEE Multimedia 11:24–37
21. Floyd S, Jacobson V (1993) Random early detection gateways for congestion avoidance. IEEE/ACM Trans Network 1:397–413
22. Dijkstra EW (1959) A note on two problems in connexion with graphs. Numerische Mathematik 1:269–271
23. Zipf G (1949) Human behavior and the principle of least effort. Addison-Wesley, Reading, MA