Clustered Multimedia NOD : Popularity-Based Article Prefetching and Placement

Y.J.Kim, T.U.Choi, K.O.Jung, Y.K.Kang, S.H.Park, Ki-Dong Chung
Department of Computer Science, Pusan National University, Korea

Abstract

According to the current profound development of multimedia and networking technologies, the way people communicate with, naturally, has evolved from a text-oriented into a multimedia-oriented. But, the VOD system technologies can't be easily applied to MNOD system because of the difference of the basic data and its properties. This is the reason why we propose the MNOD system.

NOD data composing news articles make a difference to VOD data in terms of media type and size, life cycle of articles and frequency of clients' interaction. Because of NOD data's intrinsic characteristics, NOD article popularity model may be different from that of VOD videos. Hence, we analyze statistically the log data of one electronic newspaper and show the popularity distribution of articles is different form Zipf's, which is a popularity model of VOD data. We propose a new article popularity model for NOD data, which we call Multi-Selection Zipf distribution. Also, we propose the article prefetching policy based on the popularity model and life cycle model of NOD articles for increasing performance in a MNOD system.

And, we consider the data placement on the MNOD system with prefetch cache. The user requests can be serviced from a cache or a disk. Consequently, in order to place a NOD data, we must consider the correlation between disks and cache according to data popularity. In this paper, we propose the data placement policy that considers both disk-existence probability and cache-existence probability of a data using data popularity

1. Introduction

1.1 Motivation

As a VOD system gets more popularity these days, a NOD(News On Demand) system supporting multimedia services, we call a *MNOD (Multimedia NOD)* system in this paper, will be popular as a news service in the near future. Furthermore, the developments of computer network and multimedia technology are making the multimedia-oriented news services possible.

Previous works generally regarded VOD and MNOD systems as an identical category from the viewpoint of using multimedia data. Multimedia data usually need high bandwidth and massive storage space and have real-time criteria. Therefore, we basically utilize the research results of a VOD system. Clearly, a MNOD system can be categorized into the same multimedia application as a VOD system, but it has some intrinsic aspects that a VOD system doesn't possess[4]. First, the articles of NOD service are made on and off in a day while the VOD data is made once a half-month or a month. Second, the more recent article is made, the more users access the article. The good movies are preferred to the others for a long time, but NOD data aren't. The NOD articles, which have the best popularity, don't last more than three days. Third, the number of articles that are requested from each user is varied. Owing to the long length of video data, users serviced in VOD system select just one or two video data. But the length of NOD data is short, as a result, users can select several data at a time. Finally, NOD data have the temporal and spatial access locality. The user requests burst according to the access time and the kind of articles. So the VOD system can't be fully applied to a MNOD system because of these different characteristics. In short, we use the VOD system research results basically, but will revise and decorate it.

The previous work showed that the system with prefetching has the better performance than the system without prefetching[1, 15]. But without considering the other characterizations such as short-term life cycle, the difference of user's access pattern according to time and etc, we can not expect to the increased performance in the MNOD system. The system with prefetching only based on data popularity has the problem, which is data replacement happens too often. Hence, using the analysis of electronic newspaper log-files, we propose the popularity model and the life cycle model of NOD articles. And we will suggest the article prefetching policy based on time window according as the models.

And then we consider the data placement on system with prefetch cache. The user requests can be serviced from a cache or a disk. Consequently, in order to place a MNOD data, we must consider the correlation between disks and cache according to data popularity. In this paper, we will propose the data placement policy that considers both disk-existence probability and cache-existence probability of a data using data popularity.

The remainder of this paper is organized as follow: In section 2 the overall architecture of a proposed MNOD system is presented briefly. Section 3 shows the popularity model and life cycle model of NOD articles via the statistical analysis of log files of electronic newspapers, and in section 4 we propose the article prefetching policy based on the popularity and life cycle model of NOD articles. In section 5

the article placement policy in the MNOD system with prefetch cache is proposed and evaluated. Lastly, in section 6 we mention the conclusion and future works.

1.2 Related work

• Data caching and Prefetching

In order to establish caching policy for NOD data, the popularity and life cycle model are important conditions. [10] showed that the popularity distribution of real video was a double exponential distribution rather than a Zipf's and proposed cache policy based on the life cycle of videos. In [11], the long-term model of movie popularity was proposed for a hierarchical VOD system, which was the modified exponential distribution with three parameters. The strategies of these papers are not acceptable for MNOD system because these analyzed only a VOD data, whose life cycle is longer than that of NOD article. [9] made a psychological analysis of access pattern of users, based on a model on human memory and insisted that the recency of data was more important criterion in data caching algorithm than the frequency. But because of various characteristics of NOD articles, considering only access recency is insufficient for caching or placement of articles. [4] proposed a news article prefetching policy but had not the correct criterion for the data replacement.

In this paper, we will show the measure of short-term and long-term of article popularity. And we will propose article prefetching policy considering that measures.

• Data placement

In Multimedia system, data placement policies have been studied so as to increase the success ratio of user requests. Therefore, most studies were continued to get the high disk bandwidth. Data Striping method is the representative placement policy. The famous striping methods are round-robin, random[3], staggered[7], permutation striping[8] and etc. And then, the policies to prevent disk bottleneck, such as dynamic replica policy[5] and data placement policy based on popularity were proposed.

But these policies considered only disk conditions based on data popularity. But in these days, since almost all systems have data cache, the user requests are serviced from a cache or a disk. Consequently, in order to place data, we must consider the correlation between disks and cache according as data popularity.

2. MNOD system overview

Though a NOD(News On Demand) system has many differences from the well-known VOD(Video On Demand) system, a NOD system is still, in many aspects, similar to a VOD system in terms of using multimedia data. Hence, we can draw out the structure of a MNOD system using that of a VOD system. And the scalability of an MNOD system has to be investigated to support hundreds of thousands of simultaneous users. As shown in Figure1, we suggest the two-layered clustered MNOD server system by considering the scalability. The suggested system structure contains two main modules: a *control node* and a *storage node*. Many studies on the VOD system architectures, theoretically, assume that there are two logical parts in the server[1, 2, 15]. Roughly speaking, the control node deals with the admission control for the storage node and the storage node schedules the admitted requests from the control node and treats storing and retrieving data.

The control node is entrusted with the control part of the whole MNOD server. It controls the permission into entering the storage node. If a NOD stream arrives at the control node, the admission manager checks if the requested stream may be admitted or not and whenever a request is permitted, it is passed from the control node to the storage node. The storage node is responsible for both storing and retrieving NOD data. Now that the storage node only schedules the permitted streams from the control node, no special admission control is required.

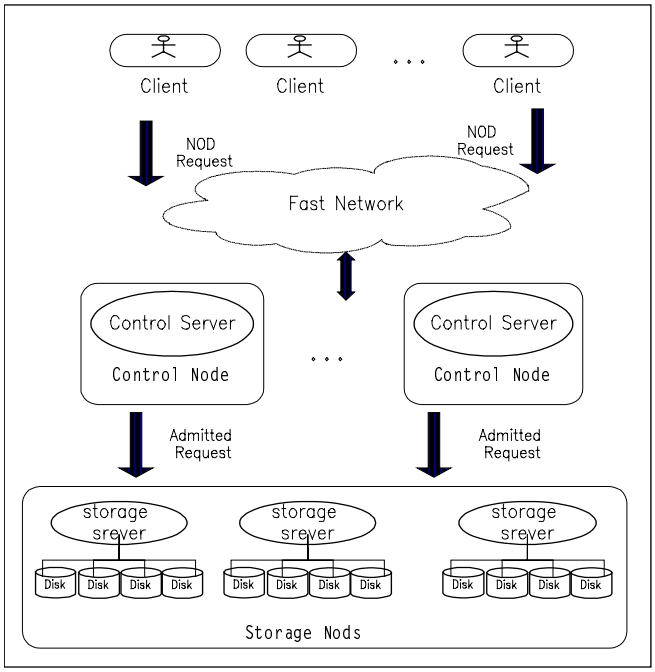


Figure 1. Proposed clustered MNOD system

3. Article popularity model

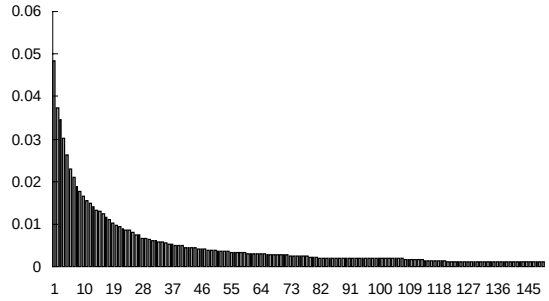
NOD data composing news articles make a difference to video data of VOD in terms of media type and size, life cycle of article and frequent clients' interaction etc. And, NOD data may be created on and off at any times, and it is popular and recent articles that most of the clients access mainly. Because of these characteristics, NOD article

popularity model may be different from that of VOD movie.

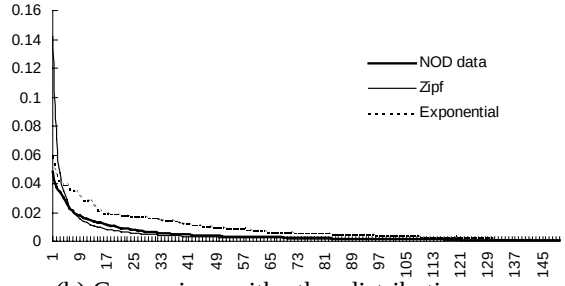
In this section we give a statistical analysis of log data of electronic newspaper running currently, and show that the popularity distribution of article is different from Zipf's, which is a popularity model of VOD. In particular, we propose a new article popularity model, which we call **Multi-Selection Zipf distribution,** and the algorithm generating its probability distribution.

This article popularity model is a short-term popularity model that denotes popularity distribution among articles in an instant time. While the short-term popularity model notifies what articles are prefetched, long-term popularity models, denoting the life cycle of articles, notify when the articles are prefetched. In addition, these models can be used for caching and placement of articles. Thus, in this section we also present a long-term popularity model of NOD articles.

3.1 Multi-Selection Zipf distribution



(a) Average access popularity



(b) Comparison with other distributions

Figure 2. The real access distribution of NOD article

The typical popularity model of VOD data is Zipf distribution; it denotes access probability of the *i*-th popular movie when the movies are sorted in the order of decreasing popularity. Then, do the popularity of NOD data follow a Zipf distribution? We analyzed statistically the log files of electronic newspapers running currently and found that news article's popularity model is different from Zipf's.

In Figure2, (a) shows the mean access popularity of news articles for a month, resulted from making a statistical analysis of log data of June 1998. (b) shows the comparison of NOD article popularity with Zipf's and Exponential's; The upper portion of NOD article distribution decrease less drastically than Zipf's but more fast than Exponential's. When a user connects with a NOD server, the user requests several articles rather than only one article. This imply that popularity does not concentrate on one article but spread over several articles. Consequently the curve of popularity distribution of NOD articles is less steep than Zipf's. For comparison, we use a pure Zipf distribution without skew and an exponential distribution with λ =0.01.

A user does not watch only one article but several articles while he is connected with a NOD server. If a user selects *k* articles on average in NOD server with *N* articles, the possible number of *k* article selections is as follow:

$$\mathrm{M} = \binom{N}{k} = \frac{N!}{k!(N-k)!}$$

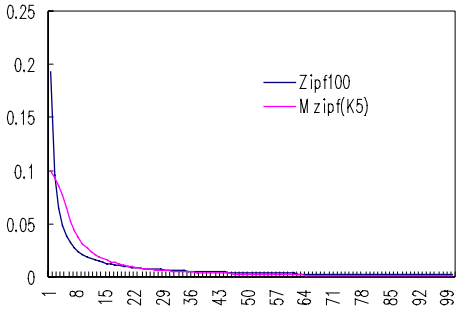
In other words it may be thought that he selects a group composed of *k* articles in M article groups. Suppose that the popularity of these M article groups follow a Zipf distribution, the probability on which each article is selected is the normalized sum of probabilities of groups containing the article. The definition of Multi-Selection Zipf distribution and the algorithm generating the probability distribution are as follows:

Definition: Suppose that a user selects *k* articles on average in NOD server with *N* articles and the popularity of the M article groups follows a Zipf distribution, we define article access probability as the normalized sum of probabilities of article groups containing each article. We call the distribution of article access probability *Multi-Selection Zipf distribution*

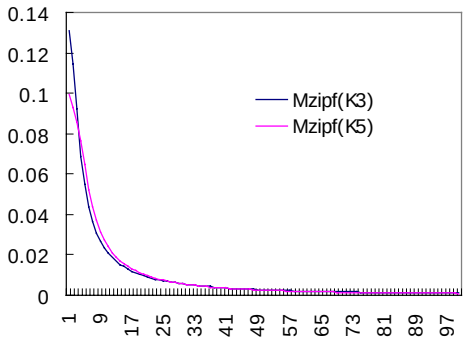
```
Input:  k (the number of selections),
     N (the number of article)
Algorithm:
   Generate M article groups;
   Decide popularity of the groups and sort in
     Popularity order;
   Calculate the popularities of groups using
     Zipf distribution;
   while (all articles) {
           adding the probability of groups containing
     the article;
     normalize such that the sum of calculated
     probabilities is 1;
   }
```

Figure 3. Multi-Selection Zipf probability generation algorithm

Figure4 compares Multi-Selection Zipf distribution with Zipf's; (a) shows that the curve of Multi-Selection Zipf's is less steep than Zipf's. In (b), as the number of

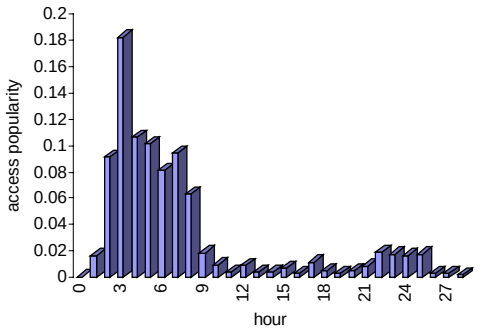


(a)Comparison of k=5 with Zipf

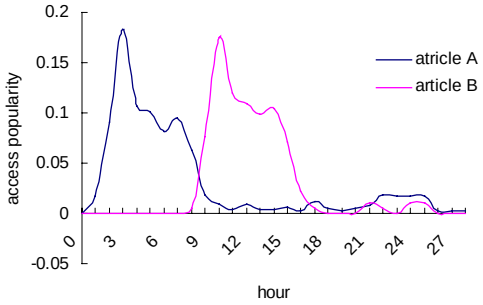


(b)Comparison of k=5 with k=3

Figure 4. Multi-selection Zipf distribution and Zipf's



(a) Access Popularity as article life cycle



(b) Life cycle of two articles

Figure 6. Article life cycle

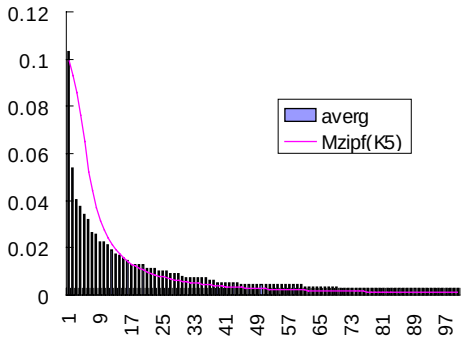


Figure 5. Multi-selection Zipf distribution & the real distribution of NOD article

selections is 3 and 5, It shows that the distribution with 5 selections decrease slower than 3 selections. That implies that the more selections are, the less steep the curve is. Figure5 shows the comparison of the real popularity resulted from log files with Multi-Selection Zipf's with 5 selections.

3.2 Long-term popularity model

The popularity of articles changes as time passes because new articles are generated at any time and users prefer new articles. Long-term popularity model, called as article life cycle, reflects the changes of the popularity of an article. In Figure6, (a) shows a long-term popularity distribution as a result of statistical analysis of log files; it has left skew, which shows that users prefer new articles to the old ones.

Long-term popularity model can be used for deciding the time to prefetch an article into the cache to decrease the retrieval overhead. In Figure6 (b), the popularity of article A decrease and article B increase, and the best point that the

article A should be replaced with article B in the cache is around the hour 10, when the curves of two articles meet.

4. Article prefetching policy

In previously suggested clustered MNOD server, clients request articles from CSVR(Control Server), and CSVR fetches articles from SSVR(Storage Server) and sends them to clients. If popular articles are prefetched in CSVR, the service delay can be reduced and network traffics can be decreased by eliminating the time to retrieve articles from SSVR[15].

In prefetching popular articles, old articles must be replaced with new popular ones when new articles are generated and the popularities of old articles decrease. So, we need to consider two popularity models mentioned in the previous section for better performance in article prefetching. In this section we present the new prefetching measure combined two popularity models and suggest the article prefetching algorithm employing the measure.

4.1 The measure of prefetching

We express the measure of short-term article popularity in terms of the access frequency during some time intervals and the measure of long-term article popularity in terms of the increased or decreased amount of frequency during some time interval. For example, the article whose access frequency increase rapidly may be a new article while the article whose access frequency decrease may be less old one and the article that access frequency hardly decrease may be old one.

Note that we use time window whose interval is several ten minutes and calculate f_i, the access frequency, by cumulating references to the article during any window *i*. By taking a weighted average of fluctuations in all the windows, we can calculate a_i, the expected variation amount of access frequency in next window as follows:

$$a_i = sf\,(f_{i-1} - f_i) + (1 - sf)\,a_{i-1} \qquad \text{[Eq.1]}$$

In Eq1, sf, denoting a smoothing factor, is the weight that indicate the importance of frequency variations of current window; The larger sf is, the heavier the weight of the current window's frequency is. Consequently we propose a new measure LBF_i (Life-cycle Based Frequency) by combining f_i with a_i as follows:

$$LBF_i = f_i + a_i \qquad \text{[Eq.2]}$$

4.2 Prefetching scheme

CSVR maintains logs of reference to each article during each window. From the logs, access frequency f_i of each article in the window can be calculated. Compute a_i using [Eq.1] and LBF_i of each article using [Eq.2]. Because the article that has the largest LBF_i is expected to be accessed most frequently in next window, the articles that have large LBF_i have to be prefetched. Once articles are prefetched in CSVR, the articles can't be replaced with others during current window. Figure7 shows the prefetching algorithm.

```
while (1) {
  if (a new window) {
      while (for all articles in window) {
          a_i = sf(f_i − f_{i−1}) + (1 − sf)a_{i−1}
          LBF_i = f_i + r_i
      }
      sort the articles in LBF_i order;
      prefetch the articles that have large LBF_i
        until cache memory is full;
  }
}
```

Figure 7. Article prefetching algorithm

4.3 The effects of prefetching

In the proposed clustered MNOD server, the prefetching in CSVR promotes the efficiency of the server system because of decreasing the overhead of retrievals from disks in SSVR. We employ the miss rate and the concurrent users to show the effects of prefetching. For simulation, we used the log files of Pusanilbo's electronic newspaper in Korea, and the parameters expressed in Table1.

Table 1. Simulation parameters

Parameter	Values
Block size	256k
The data rate that a request require	1 M/s
The data size that a request require	30M
The bandwidth of Storage Server	60M/s

Figure8 shows the relations between the amount of prefetching and the miss rate; as the amount of prefetching increases, the miss rate decreases. Figure9 also shows that as the prefetching amount increases, the number of concurrent users by LBF scheme and ones only including prefetching overheads. Because prefetching an article is equal to processing a request, the overheads of article prefetching are calculated by multiplying the number of prefetched articles and the overheads that a request requires**.** Thus the real effect of article prefetching is expressed as the difference between the number of concurrent users by LBF scheme and

the number of concurrent users only including prefetching

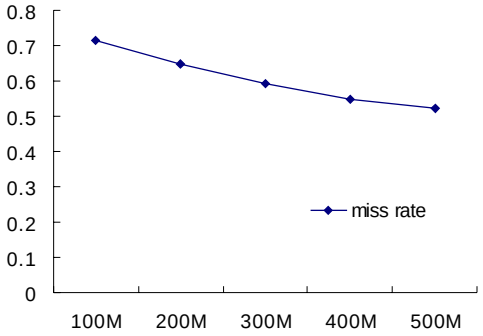


Figure 8. Miss rate according to prefetching size

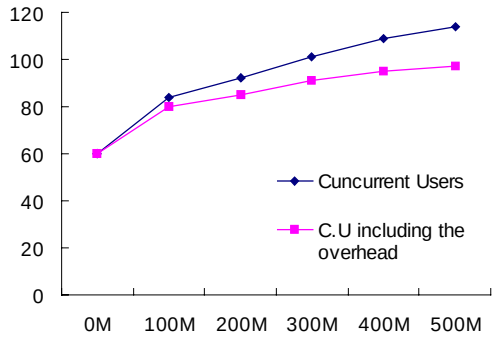


Figure 9. Concurrent users according to prefetching size

overhead. As a result, prefetching articles in CSVR increases the performance of MNOD server system by 1.5 times.

5. Article Placement Policy

5.1 Previous Data Placement

We intended to increase user service throughput by arranging replica of popular data on disks. This method, however, had a few problems.

NOD article is not as big as VOD data in size. In MNOD system, relatively large amount of data exists in the cache, and the probability that hot data is in the cache is higher than VOD system. Naturally, disk access probability will be lower relatively. And, the number of NOD articles is very much than VOD data, and by reason of storage overhead the number of articles being replicated needs to be restricted to articles with high popularity.

Thus, the data that has lower popularity but more growing probability of disk access will be suffered from long service delay because of small number of replica due to low popularity.

5.2 New Scenario of Data Placement

The new scenario we suggest for data placement in the MNOD system is to manage various numbers of replicas according to data popularity.

The article data with higher popularity will be cached by some caching method based on data popularity and have lower probability of disk access. The article data that has lower popularity but more growing probability of disk access needs higher disk I/O bandwidth. So, more many of replicas need to be assigned for article data with lower popularity.

By reason of this, we emphasize the article data that has relatively lower popularity than some hot data.

5.3 Article Placement Algorithm

5.3.1 Determining the number of article replica

Article data is arranged in decreasing order according to its popularity and has an identity number of its own that is able to be discerned from other ones. We will bestow the replication level, T_i , on article data a_i in order to determine the number of replica.

The replication level T_i is the inverse of difference between mean article popularity and popularity value of article data. This algorithm also takes advantage of standard derivation for assigning largest value of T_i to the data that has to be repeated many times. And, by that result, the average is shifted.

We can express T_i as [Eq.3] by integrating the expressed mentions above

$$\frac{1}{\left| E(X) + c\sigma(X) - p(x_i) \right|}, \quad -1.5 \le c \le 1.5 \quad \text{[Eq. 3]}$$

where x_i : identity of article data

$p(x_i)$: popularity of x_i

X : random variable of $p(x_i)$

$E(X)$: expected value (mean) of X

$\sigma(X)$: standard deviation of X

5.3.2 Article placement on disk

Article data consists of several data blocks that are units of placement. We place article data on the disk using round-robin method and the disk where the first block of news data is placed is selected randomly.

If replicas of article data exist, we will place them using the same scheme, but we must utilize a random seed for seeking a start disk that is not the same disk storing original article data, so as to secure higher disk I/O bandwidth from disk access collision. Figure10 shows the article placement algorithm proposed in this paper and the

parameters used in the algorithm are expressed in Table 2.

Table 2. Parameters of article placement policy

C_{i_i}	The number replicas of each article data, x_i
B_{news}	The number of blocks of each article data
D_i	Disk number
T_{disk}	Total amount of disk
S_{disk}	Storage capacity for disk (the amount of blocks)
s_i	The amount of block occupied on the disk, D_i

```
s_i = 1;
while ( less than total number of news data) {
 while ( less than C_i ) {
  D_i is initialized by random function;
  while (less than B_news ){
   while (1) {
    if ( s_i <= S_disk ) {
     store a data block on the disk D_i ;
     /* for using round-robin method */
     increase D_i by one ;
     D_i = D_i % T_disk ;
     increase s_i by one;
     break;
        }
    else { /* disk D_i is full */
     increase D_i by one ;
     D_i = D_i % T_disk ;
    }
   }
  }
 }
}
```

Figure 10. Article placement algorithm

5.3.3 Processing flow of user request

If the user's request arrives at a CSVR(Control Server) in Figure 1, the CSVR will divide the request into several service blocks and look into whether the request exists in cache memory or not with server control information. If the request exist in cache memory, the user request is served from cache memory. Otherwise, retrieval requests of blocks will be sent to the SSVR(Storage Server) having requested blocks and be serviced form disks. In order to reduce the service delay of user request, the SSVR will process each block as follows:

First, the SSVR will look for disks that have the block and next, compare the length of the queue of each disk to seek the disk that has the minimum length of its own queue. Finally, the SSVR assigns the block request to the minimum loaded disk.

Therefore, by using this method load balancing can be achieved among disks, and the service delay is decreased. Figure11 is a flowchart of user request processing.

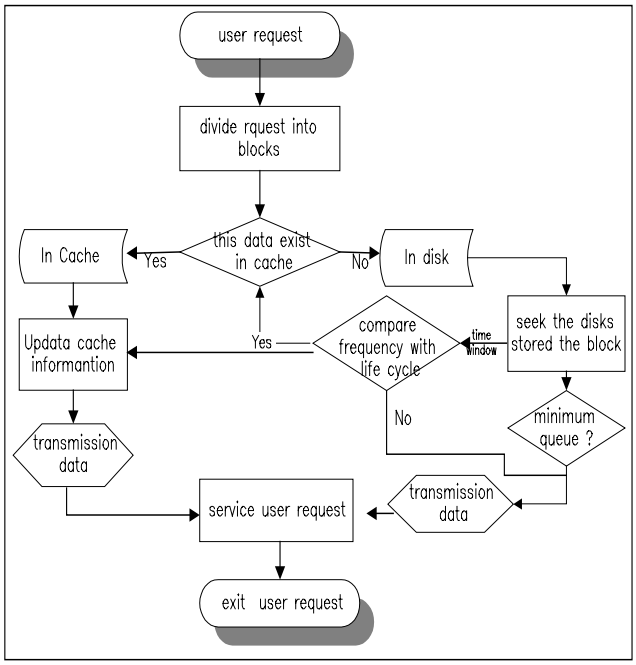


Figure 11. Flowchart of service procedure for user requests

5.4 Evaluation of article placement policy

5.4.1 Simulation environment

We enforced simulations in Sun Sparc station-20. We supposed that user requests require 1Mbyte/s capacities of disk I/O bandwidth, that the size of data which is compressed in MPEG II is 90Mbytes (90 seconds), and that the number of data is restricted to forty. One block size of disks is 256K and 240 blocks are processed within one second. The volume of a disk is 1Gbyte and the number of disks is 4 – this size accommodates total data as twice.

5.4.2 Evaluation of Simulation

We examined the service failure rate (deadline miss request/total service request) when the popularity level being replicated fully is changed into the higher level 0%~10%, 10%~20%, and 20%~30%.

Figure12 shows the service failure rate of a MNOD system with a cache. The total service performance is better than that of no caching system. In this case, articles with the second best popularity have the highest disk access probabilities because the cache hit ratios of them are lower.

Thus, maintaining the fully replicated popularity level to the higher level 10%~20% makes the best performance.

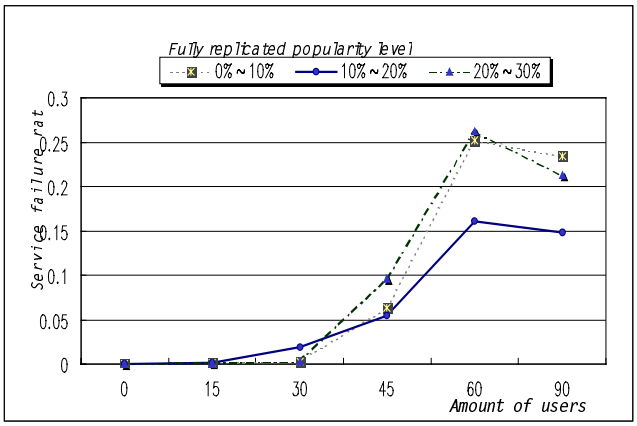


Figure 12. Service failure to amount of users

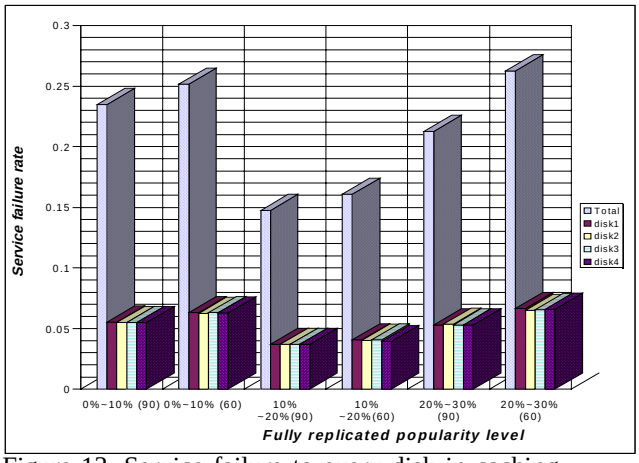


Figure 13. Service failure to every disk in caching system

Figure13 present service failure rates of each disk when the number of concurrent users is 60 and 90. The results show that the proposed article placement policy controls the load balance of every disk as well.

6. Conclusion

In this paper we propose popularity-based article prefetching and placement policy for NOD article data.

Because NOD data composing news articles make a difference to video data of VOD in terms of media type and size, life cycle of articles and frequency of clients' interaction, a new NOD article popularity model is needed to explain user access patterns. As a result, we show that the popularity distribution of articles is different from Zipf's, which is a popularity model of VOD data. In particular, we propose a new article popularity model, which we call Multi-Selection Zipf distribution. The proposed model is the short-term popularity model that denotes popularity distribution among articles in an instant time. If the article prefetching algorithm may be considered, while the short-term popularity model notifies what articles are prefetched, long-term popularity model, denoting the life cycle of articles, announces when the articles are prefetched. Thus, we propose the article prefetching policy based on the popularity and life cycle model of NOD articles for increasing performance in a MNOD system.

And, we propose the data placement policy for NOD data that considers both article popularity and disk access probability. In this policy, we replicate data that have not best popularity, but high disk access probability because the probability that the best popular data is in cache memory is high and relatively it's disk access probability is low. And, original data and replica are placed to minimum loaded disks to achieve load balancing among disks. The proposed article placement policy decreases the service failure rate and increases the service throughput by way of load balancing among disks.

Our future work is to develop a mathematical model of Multi-Selection Zipf distribution and to investigate article placement and prefetching algorithms in clustered architecture environment.

Reference

[1] Louis P. Slothouber, "A Model of Web Server Performance", http://vorlon.biap.com/webperformance/modelpaper.html, 1996.

[2] Odysseas I. Pentakalos, Daniel A. Menascé, "An Approximate Performance Model of a Unitree Mass Storage System", 14th IEEE Symposium on Mass Storage Systems, pp. 210-224, Sep.1995.

[3] J. Alemany, J. S. Thathacher, "Random Striping for News on Demand Servers", Technical Report UW-CSE-97-02-02, Univ. Washington, 1997.

[4] Y. U. Park, G. H. Back, W. I. Seo, Y.J. Kim, K. D. Chung, "New Buffer policy for NOD data", Conference on Korea Broadcast Engineering, Korea, 1997.

[5] A. Dan, M. Kienzle, D. Sitaram, "A dynamic policy of segment replication for load balancing in video-on-demand servers", Multimedia System 3:93-103, 1995.

[6] T. D. C. Little, D. Venkatesh, " Populary - based assignment of movies to storage devices in video-on-demand system", Multimedia System 2:280-287, 1995.

[7] S. Berson, R. Muntz, S. Ghandeharizadeh, X. Ju, "Staggered Striping in Multimedia Information Systems", SIGMOD Conference, pp. 79-90, 1994.

[8] R. Flynn, W. Tetzlaff, "Disk Striping and Block Replication Algorithm For Video File Servers", IEEE 3rd International Conference on Multimedia Computing Systems, 1996.

[9] J. E. Pitkow, M. M. Recker, "A Simple Yet Robust Caching Algorithm Based on Dynamic Access Patterns",

In the Proceeding of the Second International WWW Conference, 1997.

[10] S. A. Barnett, G. J. Anido, H. W. Beadle, “Caching Policies in a Distributed Video-on-Demand System”, In the Proceeding of Australian Telecommunication Networks and Applications Conference, 1995.

[11] C. Griwodz, M. Bar, L. C. Wolf, “Long-term Movie Popularity Models in Video-on-Demand Systems or The Life of an on-Demand Movie”, The fifth ACM International Multimedia Conference, 1997.

[12] I. Tatarinov, V. Soloviev, A. Rousskov, “Static Caching in Web Servers”, submitted to the 6th international Conference on Computer Communications and Networks.

[13] Lee D., Choi J., J. Kim, S. H. Noh, S. L. Min, Y. Cho, C. S. Kim, “LRFU Replacement Policy: A spectrum of block replacement policies”, In Seoul National University Technical Report SNU-CE-AN-96-004, Korea, 1996.

[14] Renu Tewari, Harrick M. Vin, Asit Dan & Dinkar Sitaram, “Resource Based Caching for Web Servers”, Proceedings of SPIE/ACM Conference on Multimedia Computing and Network, 1998.

[15] Renu Tewari, Rrajat Mukherjee, Daniel M. Dias, Harrick M. Vin, "Real-Time Issues for Clustered Multimedia Servers", IBM Research Center, Technical Report, 1995.