An I/O Scheduling Algorithm for Soft Real-time Services Oriented iSCSI Storage System

ZHA Qiwen
National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China
Email: zhaqw@dsp.ac.cn

ZHANG Wu, ZENG Xuewen and GUO Xiuyan
National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing, China
Email: {zhangw, zengxw, guoxy}@dsp.ac.cn

***Abstract*—iSCSI storage system is the most widely used network storage system, and soft real-time services are the most common type of business. However, the traditional I/O scheduling algorithms can't work very well for iSCSI storage system oriented soft real-time services. In this paper, the mathematical model of I/O scheduling in iSCSI storage system for soft real-time services is analyzed and discussed, and it proves that this is a multi-objective optimization problem. HB-SCHED is proposed in this paper, which is a heuristic I/O scheduling algorithm for soft real-time services oriented iSCSI storage system. The simulation results show that HB-SCHED not only has better quality in terms of real time than SCAN and can gain higher disk throughput than EDF, but also can adjust the weights between real time and disk throughput by changing the parameter value. HB-SCHED can be well adopted in soft real-time services oriented iSCSI storage system.**

***Index Terms*—I/O scheduling, iSCSI, soft real-time, multi-objective optimization**

I. INTRODUCTION

With the rapid growth of data on the network, a lot of information needs to be processed and transmitted through the network, which has higher requirements of the storage system in terms of capacity, performance, availability, scalability and manageability.

iSCSI (Internet Small Computer System Interface) is a new network protocol for network storage developed by IETF (Internet Engineering Task Force). iSCSI transmits the SCSI (Small Computer System Interface) commands through the IP network, so that it can transfer data on the network more convenient than local storage and enables remote management of the system. Because of the large capacity, flexible deployment, low cost and good extensibility of the network storage systems based on iSCSI, iSCSI is widely supported by the hardware and software industry and used in a variety of service systems.

In recent years, the soft real-time services are rising rapidly, such as electronic commerce, online transactions, real-time database systems, multi-media, and communication systems. In this type of service systems, to ensure the deadline of every task is impossible. The goal of the system is to ensure the deadline of the tasks as much as possible. Responding the task after deadline is also acceptable, but the QoS (Quality of Service) of the task is worse. The missing of deadlines of soft real-time service requests will not result disastrous consequences for the system [1].

Figure 1 shows the block diagram of iSCSI storage system for soft real-time services. The I/O performance of the system directly affects the efficiency of system, so the I/O performance optimization is very important for the overall performance of systems [2]. I/O scheduling algorithm directly determines the I/O performance of the systems, so it is of great importance to study I/O scheduling algorithm. Especially, under the environment of the widely use of iSCSI and the rapid growth of soft real-time services, to study the I/O scheduling algorithm for soft real-time services oriented iSCSI storage system has become a very important and urgent subject.

The traditional I/O scheduling algorithms rarely take into account the network factors，when applied at a iSCSI storage system，it will lead to performance fluctuation. Most of the existing real-time I/O scheduling algorithms only consider the situation of determine deadline [3]. However, in soft real-time service system, such as multi-media, VOD (Video-On-Demand) and iSCSI storage system, the deadline of I/O request is uncertain. This paper proposes a heuristic I/O scheduling algorithm HB-SCHED. The algorithm considers the impact on the iSCSI storage system by the network delay, and analyzes the I/O request of soft real-time service through the fuzzy sets theory. The simulation results show that HB-SCHED can be well adopted in soft real-time services oriented

doi:10.4304/jsw.8.7.1785-1792

iSCSI storage system.

The rest of this paper is organized as follows. The traditional I/O scheduling algorithms are reviewed in Section 2. Section 3 proposes the mathematical model of I/O scheduling in iSCSI storage system for soft real-time services. Section 4 proposes a heuristic I/O scheduling algorithm to solve the model proposed in section 4. Section 5 simulates the algorithm and analyzes the experimental results. Section 6 gives the concluding remarks finally.

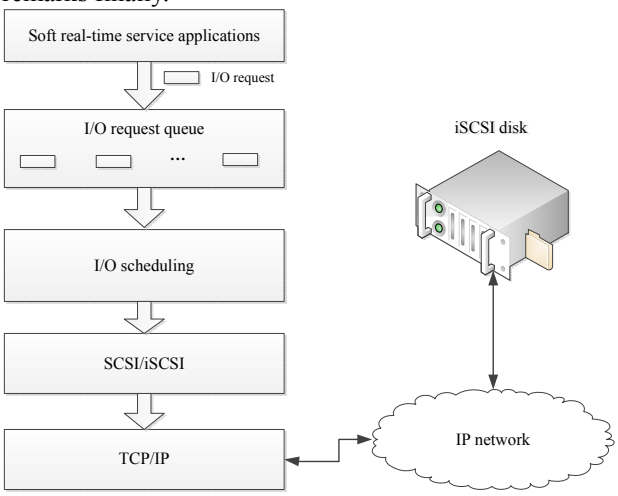

Figure 1. The block diagram of iSCSI storage system for soft real-time services

II. The Traditional I/O Scheduling Algorithms

The scholars have done a lot of research of I/O scheduling algorithms [4].

FCFS (First Come First Serve) is a very simple I/O scheduling algorithm. It schedules the requests according to the arrival sequence of the request. The advantage of FCFS is its fairness for all requests, but on the other hand its performance is poor [5] [6].

SSTF (Shortest Seek Time First) selects the request which is nearest to the current disk head position to schedule. The purpose of SSTF is to minimize the movement of the head [7]. SSTF pays attention to seek optimization, but it will cause some requests waiting to serve for a long time, even result in starvation phenomenon. Therefore, SSTF is not suitable for real-time services, such as multi-media [8].

SCAN [9] is a widely used I/O scheduling algorithm. It is used in Linux operate system. It selects the request which is nearest to the current head position and in the direction of the head's motion to be next served request. This algorithm aims to provide a high I/O bandwidth by minimizing the total seek time and rotational latency. However, no time constraint has been considered in this algorithm. Therefore, SCAN cannot be used in the real time system, where each request has a deadline and must have been served before its deadline.

There are some variants of SCAN, such as C-SCAN, LOOK and C-LOOK [10] [11]. The C-SCAN (Cyclical SCAN) algorithm replaces the bidirectional scan with a single direction of arm travel [12]. LOOK algorithm, another SCAN variation, changes the scanning direction if there are no pending requests in the current direction of travel [13]. C-SCAN and LOOK can be combined, resulting in the C-LOOK algorithm.

The above algorithms are all non-real-time algorithm. Real-time scheduling algorithm is an important branch [14] [15]. EDF (Earliest Deadline First) is used in real-time systems when requests have to be served within deadlines [16]. EDF is a natural choice for the real-time disk I/O scheduling, but it has a large overhead of seek time and rotational latency and thus results in a poor utilization of the disk.

Several hybrid algorithms, which combine EDF with conventional disk scheduling algorithms, are proposed in recent years [17] [18] [19]. FD-EDF (Feasible Deadline EDF) serves requests using the EDF algorithm if there are requests with feasible deadlines in the waiting queue; otherwise, it serves the queue using FCFS. SCAN-EDF [20] serves the requests with EDF and requests with the same deadline using SCAN algorithm. This would efficiently make use of the disk bandwidth in addition to maintaining the time constraints. However, the performance of SCAN-EDF depends on the requests that have the same dead-lines [21] [22].

The I/O scheduling algorithms discussed above do not take into account the effects of network in the iSCSI disk system and the characteristics of soft real-time services. This paper analyzes the mathematical model of I/O scheduling in iSCSI storage system for soft real-time services, and then HB-SCHED is proposed, which is a heuristic I/O scheduling algorithm.

III. The Modle Of The Problem

A. Soft Real-time Service Requests and Fuzzy Sets

Soft real-time services usually have a deadline, but the deadline is not determined. This kind of problem can be represented by fuzzy set [23] [24] [25]. This paper uses fuzzy interval to represent the request deadline, and fuzzy membership function is used as the QoS of the completion time of the request [26] [27].

Figure 2 shows the membership function of the fuzzy deadline. For fuzzy deadlines, if the request is completed before the time d, the QoS is the largest. But if the request is not completed before time d, the scheduling system will not immediately cancel the request, and the request is allowed to be completed before the time D. Service quality may change over time as the membership function. Completed time of the request after time D is not allowed in the scheduling system.

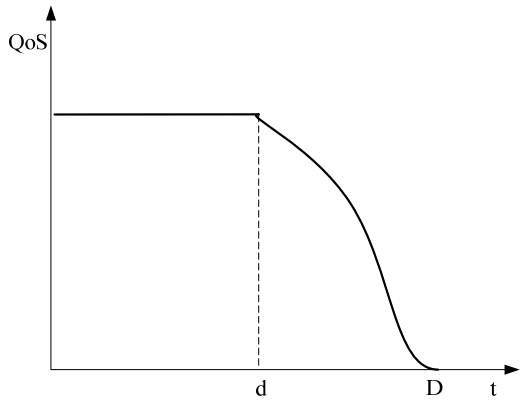

Figure 2. The membership function of the fuzzy deadline

B. Problem Description

In the soft real-time services oriented iSCSI storage system, let the total request queue be $Q[q_1, q_2,\ldots, q_n]$. We define that q_i (r_i,d_i,D_i,st_i,len_i) is a I/O request. Where r_i is the time of arrival, $[d_i, D_i]$ is the fuzzy deadline interval, st_i is the seek address, len_i is the request length. Let the initial address of the head be st_0.

Define $P[p_1, p_2,\ldots, p_n]$ as one of the schedule of Q. p_i is one of the request in Q. It is equivalent to the 0-1 programming problem by (1). The meaning of "$x_{i,j}= 1$" is that the first i scheduled request is q_j. And "$x_{i,j}= 0$" means that the first i scheduled request is not q_j.

$$X=\begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1n} \\ x_{21} & x_{22} & \ldots & x_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nn} \end{bmatrix},\ x_{i,j}=\begin{cases} 0, & p_i = q_j \\ 1, & p_i \neq q_j \end{cases} \quad (1)$$

Disk service time is mainly composed of three parts as follow.

- Seek time, the time disk head moving from the current position to the request track.
- Rotational latency, the waiting time for the required sector rotating to the head after the head moving to the right track.
- Data transmission time, it is mainly determined by RTT (Round-Trip Time) of the network in iSCSI storage system.

The rotational latency always keeps approximately constant and much less than seek time, so this paper considers only seek time and data transmission time.

Let the seek time function be $S(l)$, and l be the seek length. We assume that the network is a high speed LAN (local area network). RTT is a constant and $RTT=rtt$.

Define the service time of p_i is c_i, so we can easily get (2) as follow.

$$c_i = rtt + S(|\sum_{j=1}^{n}(x_{i,j} * st_j) - \sum_{j=1}^{n}(x_{i-1,j} * st_j)|) \quad (2)$$

Figure 3 shows the time and the seek address of requests.

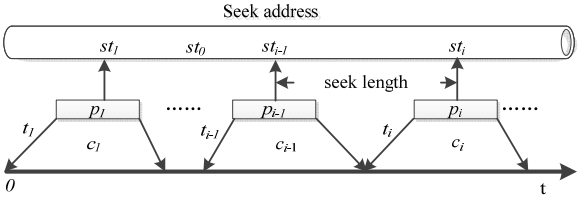

Figure 3. The time and seek address of requests

The I/O request is usually non-preemptive. We assume that the scheduling time of the first I/O request is 0. Define the scheduling time of p_i is t_i. It is easy to see (3) as follow.

$$t_1 = 0\ ,\ t_i = \sum_{j=1}^{i-1} c_j \quad (3)$$

Let the membership function of p_i be $f_i(t)$, and define the QoS of p_i to be QoS_i. It is easy to see (4) as follow.

$$QoS_i = f_i(t_i + c_i) = f_i(\sum_{j=1}^{i} c_j) \quad (4)$$

Substituting (2) and (3) into (4), we obtain (5) as follows.

$$QoS_i = f_i(i * rtt + \sum_{j=1}^{i} S(|\sum_{k=1}^{n}(x_{j,k} * st_k) - \sum_{j=1}^{n}(x_{j-1,k} * st_k))|) \quad (5)$$

Define $F(X)$ as the QoS of X, so we can get (6) as follow.

$$F(X) = \sum_{i=1}^{n} QoS_i \quad (6)$$

Obviously, one of the goals of the scheduling algorithms is $max\ F(X)$.

On the other hand, the disk throughput is also of great importance to the disk performance. Define $T(X)$ is the finish time of all requests, so we can see that:

$$T(X) = \sum_{i=1}^{n} c_i = n * rtt + \sum_{i=1}^{n} S(|\sum_{j=1}^{n}(x_{i,j} * st_j) - \sum_{j=1}^{n}(x_{i-1,j} * st_j)|) \quad (7)$$

$$throughput = \frac{1}{T(X)} \sum_{i=1}^{n} len_i \quad (8)$$

We can see from (8) that the $T(X)$ must be a minimum in order to make the throughput maximum. So another goal of the scheduling is $min\ T(X)$.

C. Constraint Conditions Analysis

In order to describe the mathematical model of the problem more clearly, the constraint conditions are analyzed as follow.

- It is easy to get the constraint condition as follow.

$$\sum_{i=1}^{n} x_{i,j} \leq 1\ ,\ j = 1,2,...,n \quad (9)$$

$$\sum_{j=1}^{n} x_{i,j} \leq 1\ ,\ i = 1,2,...,n \quad (10)$$

- For all the requests, the completion time must be earlier than the deadline. So there is another constraint condition as follow.

$$i * rtt + \sum_{j=1}^{i} S(|\sum_{k=1}^{n}(x_{j,k} * st_k) - \sum_{j=1}^{n}(x_{j-1,k} * st_k)|) \leq D_i \quad (11)$$

$$i = 1,2,...,n$$

- For all the requests, the arrival time r_i must be

earlier than the scheduling time, so we can get the constraint as follow.

$$(i-1)*rtt+\sum_{j=1}^{i-1}S(|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k)|)\geq r_i \quad (12)$$
$$i=2,3,...,n$$

D. The Mathematical Model

Based on the analysis above, we can clearly know that I/O scheduling algorithm for soft real-time services oriented iSCSI storage system discussed in this paper is a 0-1 programming multi-objective optimization problem as follow.

$$\max\ F(X),$$
$$\min\ T(X),$$
$$s.t.\begin{cases}\sum_{i=1}^{n}x_{i,j}\leq 1\ ,\ j=1,2,...,n;\\ \sum_{j=1}^{n}x_{i,j}\leq 1\ ,\ i=1,2,...,n;\\ i*rtt+\sum_{j=1}^{i}S(|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k)|)\leq D_i\\ \quad i=1,2,...,n;\\ (i-1)*rtt+\sum_{j=1}^{i-1}S(|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k)|)\geq r_i\\ \quad i=2,3,...,n;\end{cases} \quad (13)$$

IV. The Solution Of Algorithms

A. Algorithm Description

In order to simplify the problem, we make the assumptions as follow in this paper.

- It is a simple network environment, and the value of *rtt* is fixed.
- The seek time function *S(l)* is a simple linear function as follow.

$$S(l)=rl,\ r>0 \quad (14)$$

- Simplify the membership function of service quality as follow.

$$f_i(t)=\begin{cases}1\ ,\ t<d_i\\ \frac{1}{d_i-D_i}t+\frac{D_i}{D_i-d_i}\ ,\ d_i\leq t\leq D_i\\ 0\ ,\ t>D_i\end{cases} \quad (15)$$

According to the above assumptions, the mathematical model can be simplified as the following multi-objective optimization problem.

$$H(X)=\sum_{i=1}^{n}f_i(i*rtt+\sum_{j=1}^{i}|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k))|)\ ,$$
$$G(X)=\sum_{i=1}^{n}|\sum_{j=1}^{n}(x_{i,j}*st_j)-\sum_{j=1}^{n}(x_{i-1,j}*st_j)|\ ,$$
$$\max\ H(X)\ ,$$
$$\min\ G(X)\ ,$$

$$s.t.\begin{cases}\sum_{i=1}^{n}x_{i,j}\leq 1\ ,\ j=1,2,...,n\\ \sum_{j=1}^{n}x_{i,j}\leq 1\ ,\ i=1,2,...,n\\ r\sum_{j=1}^{i}|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k)|\leq D_i-i*rtt\ ,\ i=1,2,...,n\\ r\sum_{j=1}^{i-1}|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k)|\geq r_i-(i-1)*rtt\ ,\ i=2,3,...,n\end{cases} \quad (16)$$

This paper translates the multi-objective optimization problem into single objective optimization problem by building an evaluation function as follow.

$$J(X)=aH(X)-(1-a)G(X)\qquad 0<a<1 \quad (17)$$

Let *J(X)* be the benefit of schedule *X*. The goal of the I/O scheduling algorithm is to find out the *X* that *maxJ(X)*.

In (17), *a* is the evaluation function parameters. It controls the weight of the two optimization goals in (16). It can be set according to the actual needs of applications in practical application. For example, high real-time required systems can increase the value of *a*, and high throughput required systems can reduce the value of *a*.

We change the form of the solution *X* as follow.

$$X=\begin{bmatrix}X_1\\ X_2\\ ...\\ X_n\end{bmatrix}\ ,\ X_i=[x_{i,1},x_{i,2},...,x_{i,n}] \quad (18)$$

From the definition of *X* and $x_{i,j}$, it is obvious that X_i is a sub-solution of the problem. X_i is the scheduling result of p_i.

Define the benefit of X_i to be $J_i(X_i)$ as follow.

$$J_i(X_i)=aH_i(X_i)-(1-a)G_i(X_i) \quad (19)$$

The definition of $H_i(X_i)$ and $G_i(X_i)$ are (20) and (21) as follow.

$$H_i(X_i)=\sum_{i=1}^{n}f_i(i*rtt+\sum_{j=1}^{i}|\sum_{k=1}^{n}(x_{j,k}*st_k)-\sum_{j=1}^{n}(x_{j-1,k}*st_k))|) \quad (20)$$

$$G_i(X_i)=|\sum_{j=1}^{n}(x_{i,j}*st_j)-\sum_{j=1}^{n}(x_{i-1,j}*st_j)| \quad (21)$$

From (20) and (21), it is easy to get that $H_i(X_i)$ is the QoS of p_i and $G_i(X_i)$ is the head seek length of p_i.

By comparing (19) and (17), we can see (22) as follow.

$$J(X)=\sum_{i=1}^{n}J_i(X_i) \quad (22)$$

Define $K_i(X)$ as the maximum benefit of the first *i* schedules. So that Ki(X) satisfies (23).

$$K_i(X)=\max\sum_{j=1}^{i}J_j(X_i) \quad (23)$$

From (23), we can see that *maxJ(X)*, which is the optimization goal of the scheduling, is $K_n(X)$.

From (22) and (23), we can get (24) as follow.

$$K_i(X) = K_{i-1}(X) + \max J_i(X_i) \quad (24)$$

According to the analysis of (24), we can know that the solution X can be found out by working out X_i step by step. And the solution of X_i depends on the previous results and the calculation result of $J_i(X_i)$. According to the above characteristics, this paper proposes a heuristic algorithm HB-SCHED. The main idea of HB-SCHED is as follow.

- Step1: Calculate $J_1(X_1)$ for all requests in the request queue as the first scheduling request. Select the request of $maxJ_1(X_1)$ and satisfied the constraint conditions to be the first scheduling request. Dequeue this request from Q and enqueue it to P. Get the sub-solution X_1. And then the maximum benefit of the first schedule $K_1(X) = maxJ_1(X_1)$. If no requests meet the constraints, there is no solution to this problem.
- Step2: As the same method of step 1, find out the request of $maxJ_2(X_2)$, and the maximum benefit of the first 2 schedules $K_2(X) = K_1(X) + maxJ_1(X_1)$. Dequeue this request from Q and enqueue it to P. Get the sub-solution X_2. If no requests meet the constraints, backtrack to the step 1, select the request of the second $maxJ_1(X_1)$ and satisfied the constraint conditions to be the first scheduling request. And then update Q, P, $K_1(X)$ and X_1, and then look for X_2 again.
- Step3: According to the method by step2, get X_3, …, X_n, and $K_3(X)$, ..., $K_n(X)$ step by step. If there is no request meets the constraint conditions in the process of solving X_i, backtrack to the solution X_{i-1}, and recalculate $K_{i-1}(X)$.
- Step4: the solution X of the scheduling is found out step by step of X_i.
  $X=[X_1, X_2, \dots, X_n]^T$.

Code 1 is the pseudo code of HB-SCHED algorithm as follow.

Code 1 The pseudo code of HB-SCHED algorithm

```
Input: request queue Q[q1,q2,…,qn],
       request output queue P.
Output: the solution X of the scheduling.

/* initialize X with a matrix of all 0s */
/* initialize M with a matrix of all 0s, mij is used for recording
   the value of Ji(Xi) */
/* initialize P with an empty queue */

static   i=0;                  /* the steps of the schedule */

HB_SCHED(Q, P)
{
    request   *q=NULL;
    i++;
    if(Q != NULL){
        if(no request in Q satisfies the constraint condition){
            /* backtrack to the previous schedule */
            Back_Trace(Q, P);
            if(i == 0){
                /* have been back to the first step */
                return NULL;
            }
        }
        else{
            for(all qj in Q){
                if(qj satisfies the constraint condition)
                    mij = Ji(Xi) for qj;
            }
            q = qk that max mij;
            Dequeue(Q, q);
            Enqueue(P, q);
            xi,k = 1;       /* get the sub-solution Xi */
            HB_SCHED(Q, P);
        }
    }
    else{
        X=[X1, X2, … , Xn]T ;
        return X;
    }
}/*end of HB_SCHED()*/
```

And the pseudo code of sub-algorithm Back_Trace() is code 2 as follow.

Code 2 The pseudo code of Back_Trace()

```
Input: request queue Q[q1,q2,…,qn],
       request output queue P.
Output: void

Back_Trace(Q, P)
{
    request   *q = NULL;
    i--;              /* backtrack to the previous step */
    if(i == 0){
        /* have been back to the first step */
        return;
    }
    q = pi in P;      /* the request of the previous step */
    Dequeue(P, q);
    Enqueue(Q, q);
    mij for pi = 0;   /* exclude the solution of the previous
                         step */
    if(all mij == 0){
        /*no eligible request, continue to backtrack */
        Back_Trace(Q, P);
    }
    else{
        q = qk that max mij;
        Dequeue(Q, q);
        Enqueue(P, q);
        xi,k = 1;         /* get the new Xi */
          HB_SCHED(Q, P);
    }
}/* end of Back_Trace() */
```

If the result of HB-SCHED is NULL, we handle the problem using FCFS.

B. The Algorithm Complexity Analysis of HB-SCHED

Let the queue length be n. In the best case, the solution X can be found without back trace. It only need to calculate $J_i(X_i)$ for times of the length of request queue every time. So the best complexity of the algorithm is

$O(n + n - 1 + ... + 1)$, which is $O(n^2)$. In the worst case, we need to backtrack every time, the complexity of the algorithm is $O(n^2 + (n-1)^2 + ... + 1)$, which is $O(n^3)$.

From the analysis above we can see that the algorithm complexity of HB-SCHED is growing with the increase of queue length. Because the I/O request queue is always not too long, HB-SCHED can be applied for I/O scheduling problem.

V. Simulation and Analysis

In order to confirm the effectiveness of HB-SCHED, the simulation experiment based on MATLAB is done in this paper. Meanwhile, we compare HB-SCHED with the classic algorithms SCAN and EDF.

In the following, we provide the detail parameter settings of the simulation. The experiment selects five different values of request queue length, [10, 30, 50, 70, 90]. The value of *rtt* is fixed as 3ms. The seek address is a random value in interval [0~10000]. By the empirical value of iSCSI disk, we set the seek time parameter r in (14) to be 0.0025ms/track. The parameter settings are listed in TABLE I.

If there is no solution of HB-SCHED algorithm, the schedule result of the simulation is designed to be as the same as FCFS algorithm. To illustrate the influence of the parameter a on the result of algorithm, we simulate three values of a, [0.4, 0.6, 0.8] to see the change of the experiment result.

TABLE I

SIMULATER PARAMETER VALUES

rtt	3ms
st_i	random [0~10000]
r	0.0025ms/track
st_0	random [0~10000]
$f_i(t)$	defined as (15)
d_i	random [0~1000ms]
D_i	d_i + random [0~1000ms]
r_i	d_i - random[0~1000ms] and $r_i>0$
n	10, 30, 50, 70, 90
a	0.4, 0.6, 0.8

The sum of QoS of all requests is one of key performance indicators of I/O scheduling algorithms. And it is also one of the optimization goals in (13). The sum of QoS reflects the real-time performance of the algorithm. The result of "sum of QoS" in the simulation is shown in Figure 4.

The sum of QoS of EDF is the highest, as EDF is a real-time scheduling algorithm and it schedules the request considering only the deadline of the request. The SCAN algorithm's sum of QoS is the lowest, because SCAN algorithm only considers the seek address order. HB-SCHED algorithm proposed in this paper has a much better real time performance than SCAN algorithm. And with the increase of the value of a, the sum of QoS of HB-SCHED algorithm also increases, and more and more close to EDF algorithm.

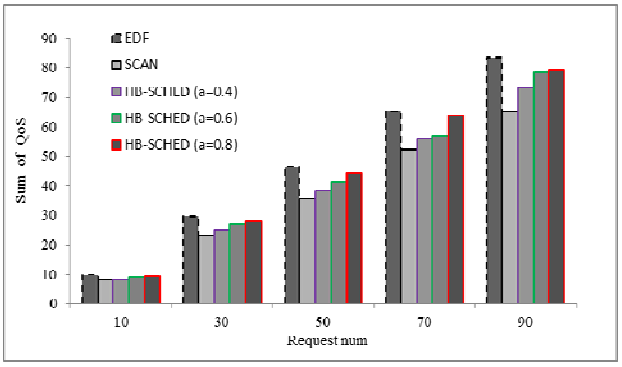

Figure 4. The simulation result of "Sum of QoS"

The throughput of disk is another key performance indicator of I/O scheduling algorithms. From (8) we can see that it is equivalent to research the sum of completion time of all requests, and it is also another optimization goal in (13). The result of "sum of time" in the simulation is shown in Figure 5. The sum of time reflects the throughput of disk, from which we can see the disk utilization.

The sum of time of EDF is the longest, as EDF does not consider the seek address, spending a lot of time on seeking. The SCAN algorithm's sum of time is the lowest, because it schedules the requests only considering the seek address and the movement direction of the head. SCAN aims to provide a high I/O throughput by minimizing the total seek time. HB-SCHED algorithm proposed in this paper has a much smaller sum of time than EDF algorithm. And with the smaller value of a, the sum of time of HB-SCHED algorithm is also decreased, and more and more close to the sum of time of SCAN algorithm.

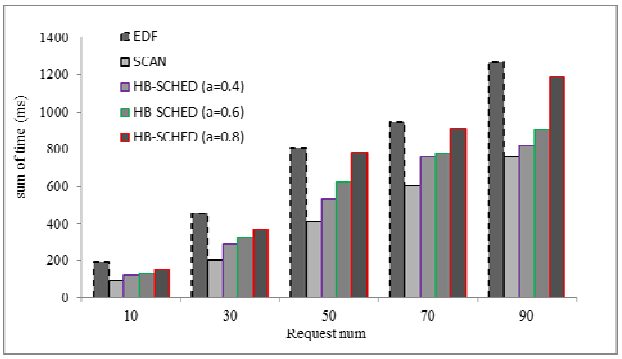

Figure 5. The simulation result of "Sum of Time"

Through the above analysis of the simulation experiment results we can conclude that HB-SCHED algorithm has a better real-time performance than SCAN and a better throughput performance than EDF. And take "a = 0.6" for example, figure 6 shows the experiment results of $J(X)$ defined in (17) , which is the single objective optimization target of HB-SCHED. We can see that HB-SCHED is higher than EDF and SCAN.

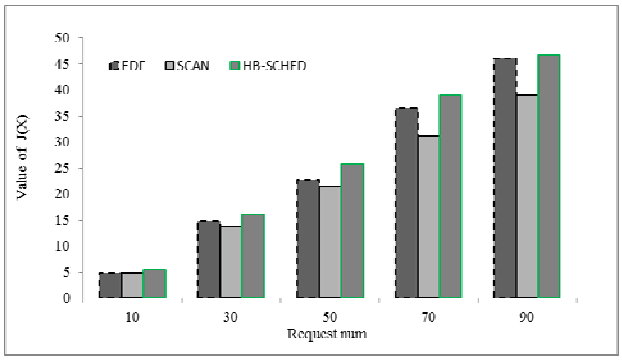

Figure 6. The simulation result of "value of *J(X)*"

Because HB-SCHED schedules the I/O requests considered the real-time performance and the iSCSI disk throughput at the same time. The target of the algorithm is to meet the real-time requirements of the soft real-time service requests, at the same time to maximize the iSCSI disk throughput as much as possible. HB-SCHED can adjust the weights between real-time performance and iSCSI disk throughput by changing the value of parameter *a*. The simulation experiment results proves that HB-SCHED can be well adopted in soft real-time services oriented iSCSI storage system.

VI. CONCLUSION

In the iSCSI storage systems, I/O scheduling is one of the most important factors that affect the overall system performance. Soft real-time services such as video are the most common services in nowadays. So it is of great importance to study the I/O scheduling algorithm for soft real-time services oriented iSCSI storage system under the environment of the widely use of iSCSI and the rapid growth of soft real-time services.

This paper proposes a 0-1 programming multi-objective optimization problem by analyzing the mathematical model of I/O scheduling in iSCSI storage system for soft real-time services. And then solves the problem by HB-SCHED, a heuristic algorithm. The simulation experiment result proves that HB-SCHED algorithm considers the real-time performance of the soft real-time service system and the iSCSI disk throughput at the same time, and can adjust the weights between real-time performance and iSCSI disk throughput by changing the value of parameter.

In conclusion, HB-SCHED can be well used in soft real-time services oriented iSCSI storage system for I/O scheduling.

ACKNOWLEDGMENT

The research of this paper was supported by National High-tech Research and Development Projects of China (2011AA01A102) and Strategic Leading Project of Chinese Academy of Sciences (XDA06010302). The authors wish to thank the reviewers for their valuable comments on this paper.

REFERENCES

[1] H. Kopetz, Real-time systems: design principles for distributed embedded applications, 2nd ed.. Springer Science Business Media, 2011, pp.79-109.
[2] W.W. Hsu and A.J. Smith, The performance impact of I/O optimizations and disk improvements. IBM Journal of Research and Development, 2004. 48(2): p. 255-289.
[3] X. Geng, et al., A Task Scheduling Algorithm for Multi-Core-Cluster Systems. Journal of Computers, 2012. 7(11): p. 2797-2804.
[4] Y. Hu, X. Long and J. Zhang, I/O Behavior Characterizing and Predicting of Virtualization Workloads. Journal of Computers, 2012. 7(7): p. 1712-1725.
[5] W. Zhao and J.A. Stankovic. Performance analysis of FCFS and improved FCFS scheduling algorithms for dynamic real-time computer systems. in Real Time Systems Symposium, 1989., Proceedings. 1989: IEEE.
[6] Y. Deng, et al., Evaluating disk idle behavior by leveraging disk schedulers. Computing, 2012. 94(1): p. 69-93.
[7] M. Hofri, Disk scheduling: FCFS vs. SSTF revisited. Communications of the ACM, 1980. 23(11): p. 645-653.
[8] S. Yashvir and O. Prakash, Disk Scheduling: Selection of Algorithm. arXiv preprint arXiv:1210.6447, 2012.
[9] P.J. Denning. Effects of scheduling on file memory operations. in Proceedings of the April 18-20, 1967, spring joint computer conference. 1967: ACM.
[10] M. Lee, K. Kim and C. Park. Real-time disk scheduling algorithms based on the two-way SCAN technique. in Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009. SCALCOM-EMBEDDEDCOM'09. International Conference on. 2009: IEEE.
[11] P. Valente and F. Checconi, High throughput disk scheduling with fair bandwidth distribution. Computers, IEEE Transactions on, 2010. 59(9): p. 1172-1186.
[12] P.H. Seaman, R.A. Lind and T.L. Wilson, On teleprocessing system design, part IV: an analysis of auxiliary-storage activity. IBM Systems Journal, 1966. 5(3): p. 158-170.
[13] A.G. Merten, Some quantitative techniques for file organization. 1970.
[14] S. Chen, et al., Performance evaluation of two new disk scheduling algorithms for real-time systems. Real-Time Systems, 1991. 3(3): p. 307-336.
[15] P. Bosch and S.J. Mullender. Real-time disk scheduling in a mixed-media file system. in Real-Time Technology and Applications Symposium, 2000. RTAS 2000. Proceedings. Sixth IEEE. 2000: IEEE.
[16] C.L. Liu and J.W. Layland, Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM (JACM), 1973. 20(1): p. 46-61.
[17] J. Yin, B. Zheng and Z. Sun, A Hybrid Real-time Fault-tolerant Scheduling Algorithm for Partial Reconfigurable System. Journal of Computers, 2012. 7(11): p. 2773-2780.
[18] V. Tarasov, et al., Efficient I/O Scheduling with Accurately Estimated Disk Drive Latencies. OSPERT 2012, 2012: p. 36.
[19] C. Staelin, et al., Real-time disk scheduling algorithm allowing concurrent I/O requests. HP Laboratories. HPL-344, 2009.
[20] A.L. Reddy and J. Wyllie. Disk scheduling in a multimedia I/O system. in Proceedings of the first ACM international conference on Multimedia. 1993: ACM.
[21] N. YAO, J. CHEN and A. LI, An Inter-group Seek-optimizing Disk Scheduling Algorithm for Real-time System⋆. Journal of Computational Information Systems, 2012. 8(18): p. 7579-7586.

[22] W. Qiu and L. Zhang, Research on Real-Time Software Development Approach. Journal of Software, 2012. 7(7): p. 1593-1600.
[23] H.J. Zimmermann, Fuzzy set theory-and its applications, 4th ed.. Springer, 2001, pp.329-433.
[24] P.K. Yadav, K. Bhatia and S. Gulati. Reliability Driven Soft Real-Time Fuzzy Task Scheduling in Distributed Computing Environment. in Proceedings of the International Conference on Soft Computing for Problem Solving (SocProS 2011) December 20-22, 2011. 2012: Springer.
[25] H. Zimmermann, Applications of fuzzy set theory to mathematical programming. Information sciences, 1985. 36(1): p. 29-58.
[26] P. Angelov and X. Zhou. Evolving fuzzy systems from data streams in real-time. in Evolving Fuzzy Systems, 2006 International Symposium on. 2006: IEEE.
[27] P. Li, et al., Directional Fuzzy Data Association Filter. Journal of Software, 2012. 7(10): p. 2286-2293.

ZHA Qiwen is a PhD student in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing. He received the B.S. degree majored in communication engineering from the Information & Electronic Engineering department of Huazhong University of Science & Technology, Wuhan. His research is supported by National High-tech Research and Development Projects of China and Strategic Leading Project of Chinese Academy of Sciences. His research interest is network new media technology, high performance embedded server and high speed network technology.

ZHANG Wu is a Research Associate in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the PhD degree from Institute of Acoustics, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing. His research interest is computer network, network new media technology and high performance embedded server.

ZENG Xuewen is a Research Fellow in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the PhD degree from Institute of Acoustics, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing. His research interest is multimedia network technology, audio and video codec, digital content copyright protection and network communication technology.

GUO Xiuyan is an Assistant Researcher in the National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing. He received the PhD degree from the University of Science and Technology of China. His research interest is multimedia network technology.