A Non-partitioning File Assignment Scheme with Approximating Average Waiting Time in Parallel I/O System

Nianmin Yao, Jinzhong Chen, Shaobin Cai
Department of Computer Science and Technology, Harbin Engineering University, P.R. China
Email: yaonianmin@hrbeu.edu.cn

***Abstract*—As storage system runs an increasing variety of workload, there may be many file requests do not dispatch for a long time. The web clients who wait a long time will leave away. This paper presents a novel non-partitioning file assignment strategy called Static Approximate Fairness algorithm (SAF) in parallel I/O system. The SAF algorithm aims at obtaining approximate mean waiting time for disk file requests, as well as making load balancing in parallel I/O system. The approximate mean waiting time provides the same chance to serve different web clients. The technique we applied is referred to as open queuing network model. The SAF algorithm first selects files according to file load. Next, it assigns files to different disks until reaching their average load. The goal of fairness is obtained by assigning files to disks in terms of file load. Comprehensive experimental results indicated our new algorithm is superior to Sort Partition (SP) in terms of fairness.**

***Index Terms*—load balancing, SAF, mean waiting time, fairness, parallel I/O system**

I. Introduction

In the past decades, the performance of parallel I/O systems had been extensively investigated due to the growth and availability of RAID. File assignment to disks in parallel I/O system had been extensively researched in [1], [2], [12], [14], [15], and [19]. These file assignment algorithms assign files to disks in a way that cost function is minimized. In the general case, the cost function may involve communication costs, update costs, storage costs and queuing costs. However, finding the optimal algorithm is an NP-complete problem [14]. An off-line file assignment algorithm must assign files to disks using all the information of files. By contrast, an on-line file assignment algorithm allocates files to disks using the only information of the current state of all disks and previous assigned files. Generally, off-line mode is corresponding to static file assignment, and on-line mode is corresponding to dynamic file assignment scheme [19]. Most static file assignment algorithms assume that the access statistics are immutable, and hence the file assignment scheme needs to be computed only once and can continuously work for a long period [6], [8], [13]. Dynamic file assignment algorithms [13] update the file allocation scheme potentially upon every request. There are two different file allocation camps which are addressed as partitioning and non-partitioning. While stripping-based file assignment schemes which belong to partitioning scheme are common for file systems with large size files [18], non-partitioning file assignment algorithms are suitable for web and server applications [10], [11], [21].

Least **S**torage Balanced (LSB) [22] placement algorithm takes the least storage capacity as the cost function. Other heuristic algorithms introduce mean response time as an objective function to be minimized in parallel I/O system. Web server applications that publish significant amounts of data stored in a back-end database must answer end-users' requests instantly before they lose patience [10]. More precisely, reducing mean response time of parallel disk storage systems is a must for these applications. There are a wide variety of ways to reduce the mean response time or improve the system throughput for parallel I/O systems [3], [4], [17]. The well-known static file assignment algorithm called Sort Partition (SP) was developed to reduce mean response time in parallel I/O system [7]. SP calculates the aggregate load of all files. It sorts all files in descending order of their service time and assigns contiguous segment of files to each disk until reaching the calculated average load. The remainder files are assigned to the last disk. SP renovates mean response time with minimal variance of service time by separating large files from small files. In order to overcome SP's drawback of assigning all remaining files to the last disk, Perfect Balancing (PB) [9] allocates them to a subset of the disks. Static Round Robin (SOR) algorithm was proposed in [16] to overcome the workload characteristic assumptions by SP and PB. These file allocation algorithms aimed at minimizing response time. However, no attention has been focused on fairness. In the most general case, the web clients with large size file requests wait longer than small size file requests in SP and SOR. The purpose of this paper is to guarantee fairness and load balancing. The basic idea of SAF is to assign all files to disks in terms of file load. The SAF algorithm achieves fairness that each file requestor has approximate mean waiting time. It provides the same chance to serve different file requests. The remainder of the article is organized as follows. Section 2 formulates the fairness problem. In Section 3, we show the proposed algorithm SAF. Experimental

doi:10.4304/jsw.8.2.302-309

results are shown in Section 4. Finally, conclusions and future research are given in Section 5.

II. System Model

File assignment algorithms are used in parallel I/O systems to allocate data properly and efficiently before being accessed by users. Without restriction of generality, we assume that each file is allocated entirely to one disk. We will consider the problem of assigning n files $F=\{f_1, f_2, ...f_n\}$ among m disks in a parallel I/O system $D=\{D_1, D_2, ..., D_m\}$. Each file f_i has request rate λ_i and service time S_i .The aggregate load of parallel I/O system is L , and the load of *kth* disk is $Load(D_k)$. Thus, if load balancing, load of each disk can be calculated as:

$$Load(D_k)=L/m(1\le k\le m) \quad (1)$$

Definition 1(Priority queue): Each disk associates with a request queue. If disk resource is available when a request comes, the request occupies it at once. If no disk resource is available when a request comes, the higher priority requests will preempt lower priority requests, i.e., they can take over and use a disk resource currently being used by a lower priority request whenever no free disk resources are available.

Definition 2(Fairness): W_k denotes the mean waiting time of the *kth* type of file requests in priority queue. The parallel I/O system is said to be fair if and only if for any requests set F , it is the case that

$$W_1\approx W_2\approx ...\approx W_k\approx ...\approx W_n \quad (2)$$

The objective of this paper is to find a file assignment scheme X to satisfy (1) and (2). Table 1 summarizes the notation used throughout this paper.

III. The SAF Algorithm

This section describes Static Approximate Fairness Algorithm (SAF) and its theoretical basis. The SAF algorithm is based on open queuing model. An open queuing model is more appropriate than a closed queuing model for parallel I/O system with large number of concurrent users [7]. The system model is based on M/G/1 queue.

A. Achieve Fairness

Theorem 1. Assuming that file set is F and load set is ρ . Let $W_k/W_{k-1}=\theta(k=2,...,n)$, the load correlation is obtained as follows:

$$\begin{cases}\rho_{2k}=\dfrac{1}{\theta^{k-1}}-\dfrac{1}{\theta^{k}}-\dfrac{1}{\theta^{k-1}}\rho_1\\ \rho_{2k-1}=\dfrac{1}{\theta^{k-1}}\rho_1\end{cases}(k=1,2,...) \ .$$

where $\theta\to 1$, the fairness is achieved.

To prove the theorem 1, we will use three lemmas. The first lemma specifies the system utilization.

Lemma 1. In M/G/1 queue model, the system utilization is $\rho_s=1-\lim_{n\to\infty}P\{Q_n=0\}$.

TABLE I.
Notations

Symbol	Meaning
F	File set
D	Disk set
L	Aggregate load
Load(D_k)	the load of kth disk
m	The number of disk
λ_i	Request arrival rate of f_i
r_i	The ith request in priority queue
S_i	Service time of f_i
δ^2	Variance of service time
C	The number of file types
W_i	Meaning Waiting time of f_i
Q_n	The number of request in queue when r_n finished
$N(t)$	The number of request in queue within service time t
Y_i	The number of request in queue within S_i
$S(t)$	Distribution function
T_r	Average remaining service time
L_q	Queue length
ρ_i	Load of f_i
θ	W_k/W_{k-1} $(k\ge 2)$
P_i	Priority of f_i

Proof: Let Q_n be the number of requests in queue when request r_n finished and left away, and Y_n denotes the number of requests arrived within S_n. Let $N(t)$ stand for the number of requests arrived within t , and

$$\xi(x)=\begin{cases}1, x>0\\ 0, x\le 0\end{cases}$$

Therefore, we have $Y_n=N(S_n)$

Since the mathematical expectation of Y_n can be computed as

$$\begin{aligned}E(Y_{n+1})&=E[N(S_{n+1})]\\&=\int_0^\infty E[N(t)]dS(t)\\&=\int_0^\infty \lambda_s t dS(t)=\rho_s\end{aligned} \quad (3)$$

In M/G/1 queue model, the following holds for any n

$$Q_{n+1}=Q_n-\xi(Q_n)+Y_{n+1}, n\ge 1$$

Let $E(Q)=\lim_{n\to\infty}E(Q_n)$, therefore,

$$\lim_{n\to\infty}E(Q_{n+1})=E(Q)$$

Since Y_{n+1} and Q_n are mutual independent, it follows that

$$E(Q_{n+1})=E(Q_n)-E(\xi(Q_n))+E(Y_{n+1}) \quad (4)$$

Using formula (3) and (4), we obtain

$$\begin{aligned}&E(\xi(Q_n))=E(Y_{n+1})\\&\Rightarrow 1\times p\{Q_n=1\}+0\times p\{Q_n=0\}=E(Y_{n+1})\\&\Rightarrow p\{Q_n=1\}=\rho_s\end{aligned}$$

$$\Rightarrow \rho_s = 1 - \lim_{n\to\infty} p\{Q_n = 0\}$$

The proof of Lemma1 is completed. ■

The second lemma calculates mean waiting time of f_1 in M/G/1 queue model.

Lemma 2. In M/G/1 priority queue, the mean waiting time of f_i is $W_1 = T_1/(1-\rho_1), T_1 = \lambda_s E(1/\mu_s^2)/2.$ where λ_s is aggregate arrival rates and μ_s is service rate.

Proof: Let T_r stand for remaining service time, and W_q denotes the waiting time in priority queue. Then

(a) The remaining service time of current serving request.

(b)The service time of previous requests in the priority queue W_q is obtained.

From Lemma1, (a) is denoted as

$$T_1 = 0 \times (1-\rho_s) + T_r \times \rho_s = \rho_s T_r \quad (5)$$

(b) is given by

$$T_2 = 1/\mu_s \times L_q \quad (6)$$

According to Little formula $L_q = \lambda_s W_q$, it follows from (6) that

$$T_2 = 1/\mu_s \times \lambda_s \times W_q = \rho_s W_q$$

From (5) and (6), the mean waiting time can be computed as

$$\begin{aligned} W_q &= T_1 + T_2 = \rho_s T_r + \rho_s W_q \\ &\Rightarrow W_q \times (1-\rho_s) = \rho_s \times T_r \\ &\Rightarrow T_r = (1-\rho_s) W_q / \rho_s \end{aligned} \quad (7)$$

In M/G/1 queue model, the average waiting time $W_q = \lambda_s E(1/\mu_s^2)/2(1-\rho_s)$, then it follows from (7) that,

$$\begin{aligned} T_r &= (1-\rho_s).\frac{\lambda_s E(1/\mu_s^2)}{2(1-\rho_s)}/\rho_s \\ &= \frac{\lambda_s E(1/\mu_s^2)}{2\rho_s} \\ &= \frac{E(1/\mu_s^2)}{2/\mu_s} \end{aligned}$$

Since the first type of file has the highest priority, the mean waiting time of f_1 can be computed as

$$W_1 = T_1 + \rho_1 W_1 \quad (8)$$

Using (8), we conclude that

$$W_1 = \frac{T_1}{1-\rho_1}$$

Thus, the lemma 2 is proved. ■

The third lemma provides mean waiting time of f_i in file set F.

Lemma 3. The mean waiting time of f_k is given as $W_k = T_1 / \left(1-\sum_{i=1}^{k}\rho_i\right)\left(1-\sum_{i=1}^{k-1}\rho_i\right)$ in M/G/1 queue model.

Proof: Let x_{il} stand for the number of requests for f_i, θ_{1j} is the busy period of *jth* file request for f_1, and S_{ij} is the service time of the *jth* file request for f_i.

For the second type of file requests, the waiting time is

$$W_2^{'} = T_1 + E\left(\sum_{j=1}^{x_{1l}} S_{1j}\right) + E\left(\sum_{j=1}^{x_{2l}} S_{2j}\right) + E\left(\sum_{j=1}^{N(W_G)} \theta_{1j}\right) \quad (9)$$

Where $W_G = T_1 + \sum_{j=1}^{x_{1l}} S_{1j} + \sum_{j=1}^{x_{2l}} S_{2j}$

The mathematical expectation of W_G can be computed as

$$\begin{aligned} E(W_G) &= E\left(T_1 + \sum_{j=1}^{x_{1l}} S_{1j} + \sum_{j=1}^{x_{2l}} S_{2j}\right) \\ &= T_1 + \rho_1 W_1 + \rho_2 W_2 \end{aligned} \quad (10)$$

Using the lemma 2, (10) can be simplified as

$$E(W_G) = W_1 + \rho_2 W_2 \quad (11)$$

Therefore, the mathematical of $\sum_{j=1}^{N(W_G)} \theta_{1j}$ can be computed as

$$\begin{aligned} E\left(\sum_{j=1}^{N(W_G)} \theta_{1j}\right) &= \lambda_1 E(W_G) E(\theta_{1j}) \\ &= \lambda_1 E(W_G)/(\mu_1 - \lambda_1) \\ &= \rho_1 E(W_G)/(1-\rho_1) \end{aligned} \quad (12)$$

Using (11) and (12), it follows from (9) that

$$\begin{aligned} W_2 &= E(W_G) + E\left(\sum_{j=1}^{N(W_G)} \theta_{1j}\right) \\ &= W_1 + \rho_2 W_2 + \rho_1 E(W_G)/(1-\rho_1) \\ &= W_1 + \rho_2 W_2 + \rho_1.(W_1 + \rho_2 W_2)/(1-\rho_1) \\ &= W_1 + (\rho_1 + \rho_2) W_2 \end{aligned}$$

Similarly, we obtain the mean waiting time of $f_k\,(k \geq 2)$

$$\begin{aligned} & W_k = W_k \sum_{i=1}^{k}\rho_i + \left(1-\sum_{i=1}^{k-2}\rho_i\right) W_{k-1} \\ & \Rightarrow \left(1-\sum_{i=1}^{k}\rho_i\right) W_k = \left(1-\sum_{i=1}^{k-2}\rho_i\right) W_{k-1} \\ & \Rightarrow W_k = \left(1-\sum_{i=1}^{k-2}\rho_i\right) W_{k-1} / \left(1-\sum_{i=1}^{k}\rho_i\right) \end{aligned} \quad (13)$$

From (13), we obtain the following formula,

$$W_k = \frac{1-\sum_{i=1}^{k-2}\rho_i}{1-\sum_{i=1}^{k}\rho_i} W_{k-1}$$

$$= \frac{1-\sum_{i=1}^{k-2}\rho_i}{1-\sum_{i=1}^{k}\rho_i} \times \frac{1-\sum_{i=1}^{k-3}\rho_i}{1-\sum_{i=1}^{k-1}\rho_i} \times ,...,\times \frac{1-\rho_1}{1-(\rho_1+\rho_2+\rho_3)} \times \frac{1}{1-(\rho_1+\rho_2)} W_1$$

$$= \frac{1}{\left(1-\sum_{i=1}^{k}\rho_i\right)\left(1-\sum_{i=1}^{k-1}\rho_i\right)} \times (1-\rho_1) W_1$$

From Lemma 2, it follows that

$$W_k = T_1 / \left(1-\sum_{i=1}^{k}\rho_i\right)\left(1-\sum_{i=1}^{k-1}\rho_i\right)$$

Therefore, the Lemma 3 is proved. ■

We are now ready to prove Theorem 1.

Proof: The proof proceeds by induction on p, the number of file types. We first show the induction basis for p =1 and then the inductive step.

For $p=1, \rho_1=\rho_1$ and $\rho_2 = 1-1/\theta-\rho_1/\theta$, is trivial.

Assume that the statement of the theorem is true for $p=k$, we have

$$\begin{cases} \rho_{2k} = \frac{1}{\theta^{k-1}} - \frac{1}{\theta^k} - \frac{1}{\theta^{k-1}}\rho_1 \\ \rho_{2k-1} = \frac{1}{\theta^{k-1}}\rho_1 \end{cases} (k=1,2,...) .$$

Now consider the case $p=k+1$, From Lemma 3,

$$W_{2k+1}/W_{2k} = \left(1-\sum_{i=1}^{2k}\rho_i\right)\left(1-\sum_{i=1}^{2k-1}\rho_i\right) / \left(1-\sum_{i=1}^{2k+1}\rho_i\right)\left(1-\sum_{i=1}^{2k}\rho_i\right)$$

$$= \left(1-\sum_{i=1}^{2k-1}\rho_i\right) / \left(1-\sum_{i=1}^{2k+1}\rho_i\right) \quad (14)$$

Since $W_{2k+1}/W_{2k}=\theta$, (14) can be simplified as

$$\left(1-\sum_{i=1}^{2k-1}\rho_i\right) / \left(1-\sum_{i=1}^{2k+1}\rho_i\right) = \theta$$

$$\Rightarrow \left(1-\sum_{i=1}^{2k-1}\rho_i\right) = \left(1-\sum_{i=1}^{2k+1}\rho_i\right)\theta$$

$$\Rightarrow \left(1-\sum_{i=1}^{2k-1}\rho_i\right) = \theta - \theta\sum_{i=1}^{2k+1}\rho_i$$

$$\Rightarrow \theta\sum_{i=1}^{2k+1}\rho_i - \sum_{i=1}^{2k-1}\rho_i = \theta - 1$$

$$\Rightarrow (\theta-1)\sum_{i=1}^{2k-1}\rho_i + \theta(\rho_{2k}+\rho_{2k+1}) = \theta - 1 \quad (15)$$

By our inductive hypothesis above, it follows from (15) that

$$(\theta-1)\sum_{i=1}^{2k-1}\rho_i + \theta(\rho_{2k}+\rho_{2k+1}) = \theta - 1$$

$$\Rightarrow (\theta-1)\sum_{i=1}^{2k-1}\rho_i + \theta\left(\frac{1}{\theta^{k-1}} - \frac{1}{\theta^k} - \frac{1}{\theta^{k-1}}\rho_1 + \rho_{2k+1}\right) = \theta - 1$$

$$\Rightarrow \sum_{i=1}^{2k-1}\rho_i + \theta\left(\frac{1}{\theta^{k-1}} - \frac{1}{\theta^k} - \frac{1}{\theta^{k-1}}\rho_1 + \rho_{2k+1}\right) / (\theta-1) = 1$$

$$\Rightarrow \theta\left(\frac{1}{\theta^{k-1}} - \frac{1}{\theta^k} - \frac{1}{\theta^{k-1}}\rho_1 + \rho_{2k+1}\right) / (\theta-1) = 1 - \sum_{i=1}^{2k-1}\rho_i$$

$$\Rightarrow \theta\left(\frac{1}{\theta^{k-1}} - \frac{1}{\theta^k} - \frac{1}{\theta^{k-1}}\rho_1 + \rho_{2k+1}\right) / (\theta-1) = \frac{1}{\theta^{k-1}} - \frac{1}{\theta^{k-1}}\rho_1$$

$$\Rightarrow \rho_{2k+1} = \frac{1}{\theta^k}\rho_1 \quad (16)$$

From Lemma 3,

$$W_{2k+2}/W_{2k+1} = \left(1-\sum_{i=1}^{2k+1}\rho_i\right)\left(1-\sum_{i=1}^{2k}\rho_i\right) / \left(1-\sum_{i=1}^{2k+2}\rho_i\right)\left(1-\sum_{i=1}^{2k+1}\rho_i\right)$$

$$= \left(1-\sum_{i=1}^{2k}\rho_i\right) / \left(1-\sum_{i=1}^{2k+2}\rho_i\right) \quad (17)$$

By our inductive hypothesis above, we have

$$\rho_1 + \rho_2 + ,...,+\rho_{2k} = 1-1/\theta^k \quad (18)$$

Using (16) and (18), it follows from (17) that,

$$\frac{1}{\theta^k} / \left(1-\sum_{i=1}^{2k+2}\rho_i\right) = \theta$$

$$\Rightarrow \frac{1}{\theta^k} / \left(\frac{1}{\theta^k} - \rho_{2k+1} - \rho_{2k+2}\right) = \theta$$

$$\Rightarrow \frac{1}{\theta^k} / \left(\frac{1}{\theta^k} - \frac{1}{\theta^k}\rho_1 - \rho_{2k+2}\right) = \theta$$

$$\Rightarrow 1/\left(1-\rho_1-\theta^k\rho_{2k+2}\right) = \theta$$

$$\Rightarrow 1-\rho-1/\theta = \theta^k\rho_{2k+2}$$

Therefore, the following formula has been gained

$$\rho_{2k+2} = \frac{1}{\theta^k} - \frac{1}{\theta^{k+1}} - \frac{1}{\theta^k}\rho_1 \quad (19)$$

It follows from (16) and (19) that

$$\begin{cases} \rho_{2k+2} = \frac{1}{\theta^k} - \frac{1}{\theta^{k+1}} - \frac{1}{\theta^k}\rho_1 \\ \rho_{2k+1} = \frac{1}{\theta^k}\rho_1 \end{cases} (k=1,2,...)$$

Therefore, the proof of Theorem 1 is completed. ■

According to the theorem 1, fairness is achievable if the condition $\theta \to 1$ in priority queue model.

In the general case, the file requests wait in the usual FCFS (first-come-first-served order). Although the theorem 1 is based on priority queue model, we obtain the following theorem in non-priority queue model.

Theorem 2. Given a file set $F=\{f_1, f_2, ...f_n\}$ and load set $\rho=\{\rho_1,\rho_2,.....\rho_n\}$, the fair goal is achievable in non-priority queue if the theorem 1 is established.

Proof: In priority queue model, r_i is served before r_j. The mean waiting time set of file requests is defined as

$W_{old}=\{W_1,W_2,...,W_i,...,W_j,...,W_n\}$. In non-priority queue model, r_j is served before r_i. The new mean waiting time set is $W_{new}=\{W_1,W_2,...,W_j,...,W_i,...,W_n\}$.

From Theorem 1, in the W_{old}, we have

$$\lim_{\theta\to 1^+}\frac{W_i}{W_j}=1\ (\forall i,j=1,2,...,n) \tag{20}$$

We prove the following formula,

$$\lim_{\theta\to 1^+}\frac{W_j}{W_i}=1\ (\forall i,j=1,2,...,n) \tag{21}$$

Using lemma 3, we obtain

$$\frac{W_i}{W_j}=\frac{\left(1-\sum_{p=1}^{2k}\rho_p\right)\left(1-\sum_{p=1}^{2k-1}\rho_p\right)}{\left(1-\sum_{p=1}^{2m}\rho_p\right)\left(1-\sum_{p=1}^{2m-1}\rho_p\right)}=\frac{\theta^{2k}}{\theta^{2m}} \tag{22}$$

From (20) and (22), we obtain

$$\lim_{\theta\to 1^+}\frac{W_j}{W_i}=1\ (\forall i,j=1,2,...,n)$$

In other words, the fairness is achievable in non-priority queue model of parallel I/O system. ■

Algorithm 1: SAF algorithm with detail description

Input: file set $F=\{f_1,f_2,\ldots,f_n\}$ and disk set $D=\{D_1,D_2,\ldots,D_m\}$
Output: A file Assignment Scheme X
Step 1:Calculate average disk load $\rho\equiv\sum_{i=1}^{m}\lambda_i S_i.$;
Step 2:Initialize ρ_1 as the largest load,then according to formulas;

$$\rho_k=\begin{cases}\frac{1}{\theta^{k-1}}-\frac{1}{\theta^k}-\frac{1}{\theta^{k-1}}\rho_1 & k=2n\\ \frac{1}{\theta^{k-1}}\rho_1 & k=2n+1\end{cases}$$

Selects the nearest values $\rho'_1,\rho'_2,\ldots,\rho'_n$ from load set ρ;
Step3:

```
f_i = 1; /*the id of the first file*/
for k = 1, k ≤ m; do
    load_k = 0; /*the id of the first file*/
    while load_k ≤ ρ do
        if f_i ≤ n then
            load_k = load_k + ρ'_k;
            X(d_k,;) = X(d_k,;) ∪ f_i; /*assigning file to disk*/
            f_i = f_i + 1 ; /*the next file id*/
        end
    end
end
if (f_i ≤ n) then
    X(d_k,;) = X(d_k,;) ∪ f_i; /*assigning remaining file to the last disk*/
end
```

Figure 1. The Static Approximate Fairness Algorithm with detail description

B. SAF Algorithm Description

The SAF algorithm assigns files to disks according to load. Fig. 1 outlines SAF algorithm with some detailed descriptions. SAF calculates the average disk load in step 1. In step 2, SAF selects files with the closest optimal load from file set F. Step 3 assigns files to different disks until reaching their average load. Remainder files will be assigned to the last disk. The load of remainder files is almost equal to the load of files assigned to the last disk. So it is very reasonable to assign remainder files to the last disk. SAF achieves fairness that each request has approximate mean waiting time according to the theorem 1. In addition, SAF guarantees load balancing where each disk load does not exceed average load.

C. SAF Algorithm Complexity

The algorithm complexity is divided space complexity and time complexity. The input of SAF is file set and disk set. Therefore, the space complexity of SAF is $O(m+n)$. The time complexity of Step 1 is $O(1)$. The time of Step

2 is $O(1)$. In Step 3, SAF algorithm includes " *for* " and " *while* " operations. The time complexity of Step 3 is $O(m \times n)$. Therefore, the time complexity of SAF is $O(m \times n)$.

IV. Performance Evaluation

Here we compared the performance of the proposed SAF against the well-known file assignment schemes Greedy and SP. We considered fairness, load balancing, and mean response time as the primary performance metric in parallel I/O system.

A. Description of Test Data and Experiments

The experimental testbed is based on SimPy. SimPy is a process-based discrete-event simulation language based on standard Python and released under the GNU LGPL. In the simulation, SimPy randomly generated 1000 requests of each f_i in file set $F = \{f_1, f_2, ... f_n\}$, and access rates ranged from 1 to 1000.

B. The Number of File Types

We found SAF guaranteed fairness in contrast to SP. SAF implements fairness for different parameter θ as shown in Fig.2-4. For example, for $\theta = 1.0002$, $C = 10$ and $m = 4$, SAF provides approximate mean waiting time between file requests compared with SP.

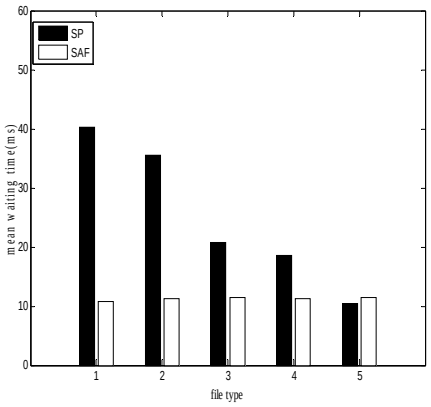


Figure 2. Mean waiting time for $C = 5$, $\theta = 1.0001$, and $m = 1$

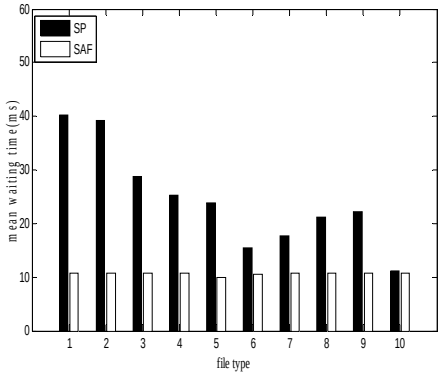


Figure 3. Mean waiting time for $C = 10$, $\theta = 1.0002$, and $m = 4$

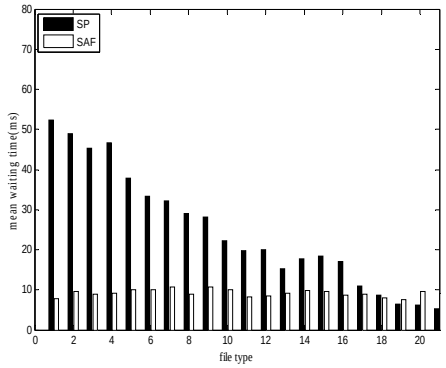


Figure 4. Mean waiting time for $C = 20$, $\theta = 1.0005$, and $m = 5$

C. Two Types of Access Rates Distributions

Uniform access rates distribution and non-uniform access rates distribution with the same aggregate access rate are shown in Fig. 5. As Figs. 6 and 7 show, the variance of mean waiting time in SAF is smaller than SP, which means SAF achieves better fairness than SP under two types of access rates distributions.

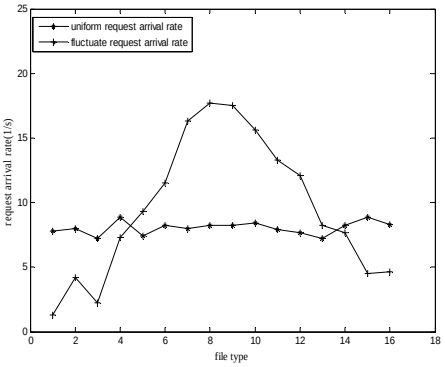


Figure 5. Two types of access rates distributions with the same aggregate access rate

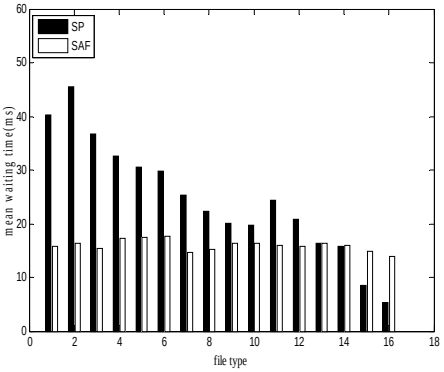


Figure 6. Mean waiting time of SP and SAF for uniform access rates distribution under $C = 16$, $\theta = 1.008$, and $m = 6$

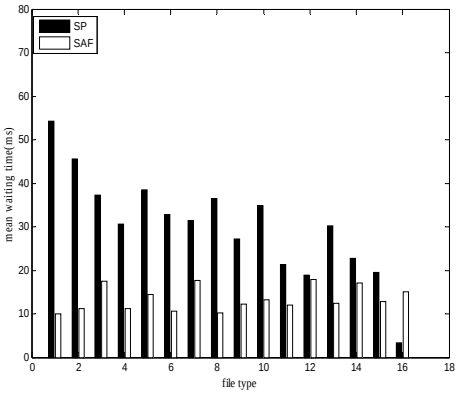


Figure 7. Mean waiting time of SP and SAF for non-uniform access rates distribution under $C=16$, $\theta=1.008$, and $m=6$

D. Workload Characteristic Assumption

We found that SAF satisfied the fair goal on workload characteristic assumption. As observed in real system traces [5], [18], there exists a strong inverse correlation between file access frequency and file size in the real-world web applications requests. The most popular files are typically small in size, while the large size files are relatively unpopular. Fig. 8 shows the distribution of access rates across 30 different types of files. There are three types of small size files with small service time, and access rates are 160,250 and 200. The mean waiting time of SP and SAF are shown in Fig. 9. The mean waiting time of SP starts with a steady decline because it assigns files to the disks according to descending order of service time. SAF guarantees approximate mean waiting time between disk requests due to it assigns files to disks in terms of load.

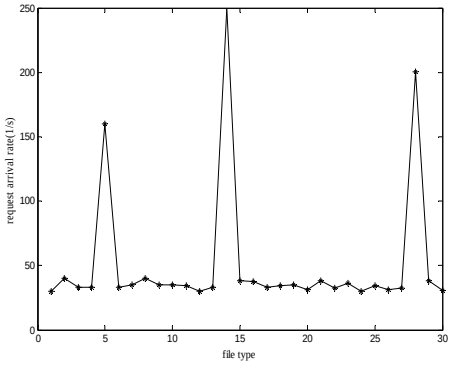


Figure 8. The distribution of access rates across 30 types of files.

E. Load Balancing

The SAF algorithm guaranteed load balancing among different disks. Fig. 10 shows a sample of coefficient of variation of disk load under different number of disks. The Greedy algorithm leads to the best load balance because load balancing is its only goal. SAF and SP also guarantee load balancing. These results confirm our expectation that SAF leads to load balancing.

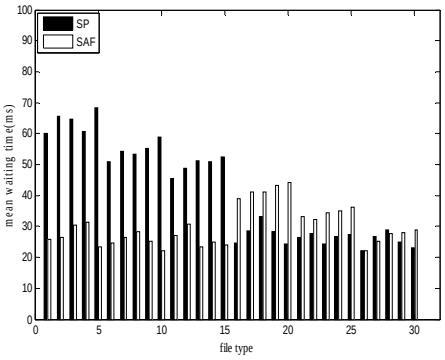


Figure 9. Mean waiting time of SAF and SP for $C=30$, $\theta=1.0001$, and $m=6$.

disks	SAF	SP	Greedy
4	0.135	0.138	0.030
8	0.132	0.135	0.028
12	0.130	0.131	0.025

Figure 10. Disk load variance

F. Mean Response Time of SP,SAF and Greedy

Fig. 11 shows the mean response time of three algorithms. SP provides 20% improvement in mean response time compared with Greedy, and 10% improvement in mean response time compared with SAF. These results confirm our intuitive expectation that SP leads to the best response time because reducing response time is its main goal. On the other hand, Greedy leads to the worst response time because it does not explicitly attempt to reduce response time.

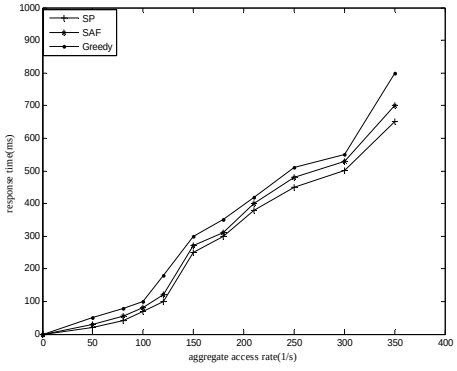


Figure 11. Mean response time of three algorithms

V. CONCLUSION

In this article, we have presented a static non-partitioning file allocation algorithm SAF where disk accesses each file are modeled as a Poisson process. The

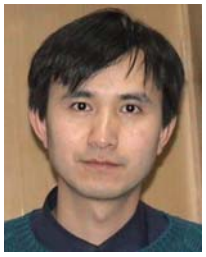
SAF algorithm allocates files to disks based on load. Therefore, the mean waiting time between file requests is approximate. In addition, The SAF algorithm guarantees load balancing. The SAF algorithm provides the same chance to serve different file requests. Experimental results show SAF achieves fairness compared with SP and reduces the mean response time compared with Greedy in parallel I/O system. In future work, we will investigate non-partitioned dynamic file assignment scheme.

ACKNOWLEDGMENT

The authors thank the editor and the anonymous reviewers for their helpful comments and suggestions. We are grateful for financial support from the State Key Laboratory of High-end Server and Storage Technology and Fundamental Research Funds for the Central Universities.

REFERENCES

[1] B.Wah, Jan. 1984. File Placement on Distributed Computer Systems. Computer, vol. 17, no. 1, pp. 23-32.
[2] G. Copeland, W. Alexander, E. Bougher, and T. Keller, 1988. Data Placement in Bubba, Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 99-108.
[3] HUANG, H., HUNG, W., AND SHIN, K.G. 2005. FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Perormance and Energy Consumption. In Proceedings of the 12th ACM Symposium on Operating Systems Principles. 263-276.
[4] Hsu, W. W., SMITH, A.J., and YOUNG, H.C.2005. The AutoMatic Improvement of Locality in Storage Systems. ACM Trans. Comput.Syst. 23,4,424-473.
[5] J. Ousterhout, H. Da Costa, D. Harrison, J. Kunze, M. Kupfer, and J. Thompson, "A Trace-Driven Analysis of the UNIX 4.2BSD File System" Technical Report CSD-85-230, Univ. of California at Berkeley, 1985.
[6] Kangasharju, J., Roberts, J., AND Ross, K. 2002. Object Replication Strategies in Content Distribution Networks. Comput Comm. 25, 4,367-383.
[7] LEE, L.W., SCHEUERMANN, P., AND VINGRALEK, R .2000.File Assignment in Parallel I/O Systems with Minimal Variance of Service Time. IEEE Trans.Comput.49, 2, 127–140.
[8] Loukopoulos, T. AND AHMAD, I.2000. Static and Adaptive Data Replication Algorithms for Fast Information Access in Large Distributed Systems. In Proceedings of the 20th International Conference on Distributed Computing Systems. 385-392.
[9] Madathil, D.K. Thota, R.B. Paul, P. Tao Xie. 2008. A Static Data Placement Startegy towards Perfect Load-balancing for Distributed Storage Clusters. PROC. IEEE International Symposium on Parallel and Distributed Processing, 1-8.
[10] MERIDO, P., ATZENI, P., AND MECCA, G. 2003. Desig n and Development of Data Intensive Web Sites: The Aran eus approach. ACM Trans.Inter.Tech.3, 1, 49–92.
[11] NISHIKWA, N., HOSOKAWA, T., MORI,Y., YOSHIDA, K., AND TSUJI, H.1998. Memory Based Architecture for Distrbuted WWW Caching Proxy. In Proceedings of the 7th International Conference on World Wide Web. 205–214.
[12] P. Scheuermann, G. Weikum, and P. Zabback, 1998. Data Partitioning and Load Balancing in Parallel Disk Systems. VLDB J., vol. 7, no. 1, pp. 48-66.
[13] QIU, L., PADMANABHAN, V.N., AND VOELKER, G. M.2001.On the Placement of Web Server Repli-cas. In Proceedings of the 21th Annual Joint Conference on Comp uter and Communications. 1587–1596.
[14] R. Dewan and B. Gavish, July 1989. Models for the Combined Logical and Physical Design of Databases. IEEE Trans. Computers, vol. 38, no. 7, pp. 955-967.
[15] S. March and S. Rho, Mar. /Apr. 1995. Allocation Data and Operations to Nodes in Distributed Database Design. IEEE Trans. Knowledge and Data Eng., vol. 7, no. 2, pp. 305-317.
[16] Tao Xie and Yao Sun, 2009. A File Assignment Strategy Independent of Workload Characteristic Assumptions. ACM Transactions on Storage, vol.5, Issue 3.
[17] TEWARI, R, 1992. Distributed File Allocation with Consistency Constraints. In Proceedings of the 12th International Conference on Distributed Computing Systems. 408-415.
[18] T. Kwan, R. Mcgrath, and D. Reed, "Ncsas World Wide Web Server Design and Performance," Computer, vol. 28, no. 11, pp. 67-74, Nov. 1995.
[19] W. Dowdy and D. Foster, 1982. Comparative Models of the File Assignment Problem. ACM Computing Surveys, vol. 14, no. 2, pp. 287-313.
[20] XIE, T.2008. SEA: A Striping-Based Energy-Aware Strategy for Data Placement in RAID Structured Storage Systems. IEEE Trans.Comput.57, 6, 748–761.
[21] Y. Azar, A. Broder, and A. Karlin, 1994.On-Line Load Balancing.Theoretical Computer Science, vol. 130.
[22] Yung-Cheng Ma, Jih-Ching Chiu, Tien-Fu Chen, Chung-Ping Chung, 15 May 2003. Variable-size Data Item Placement for Load and Storage Balancing, Journal of Systems and Software, v.66 n.2, p.157-166.



Nianmin Yao, born in 1974, a Professor of Harbin Engineering University of China. He Got the Bachelor, Master and doctor degree from Jilin University. His main research interests include network storage, system performance analysis, theory of computation etc.

Jinzhong Chen, born in 1986, a doctor of Harbin Engineering University of China. The main research direction is disk scheduling, parallel system, quality of service (QoS), real-time task scheduling and Solid State Drivers (SSD).