Dynamic Object Assignment in Object-Based Storage Devices

Lingjun Qin and Dan Feng

Key Laboratory of Data Storage Systems, Ministry of Education of China,
School of Computer, Huazhong University of Science and Technology, Wuhan, China
dfeng@hust.edu.cn, qinlingjun@yahoo.com.cn

Abstract. Object-based Storage Devices (OSDs) are the building block of Object-based Storage Systems. Object assignment in OSDs can largely affect the performance of OSDs. A real-time object assignment algorithm is proposed in the paper. The algorithm aims at minimizing the variance of Mean Response Time (MRT) across disks. To address the online problem with *a priori* unknown workload parameters, the algorithm employs an adaptive mechanism to estimate the parameters of workloads. The simulation results show that the algorithm can effectively balance the MRT in the disk I/O sub-systems.

1 Introduction

Object-Based Storage (OBS) is the new trends in network storage field [1]. Combining the benefits of the NAS and SAN architecture, OBS is allowing storage systems to reach petaBytes-scale levels. As the building block of OBS, Object-based Storage Devices (OSDs) play an important role in OBS and have great effects on the overall performance of the storage systems. Many efforts have been made to improve the performance of OSDs. RAID is one of the performance enhancing techniques that have been studied widely [2]. Through data striping, one object can be distributed among multiple disks to achieve parallel data transfer. Another important technology is object assignment. In order to remove the system bottleneck, objects should also be uniformly allocated among all the disks, thus balancing the load in a disk sub-system.

In on-line storage applications built on OBS systems, real-time object assignment is an essential issue which affects the overall performance. In such applications, data objects usually need to be stored immediately they are produced. For example, in high-energy physics, a particle accelerator can produce 100MB raw data per second. Moreover, these large mount of data should be stored in OSDs as quickly as possible for scientific research. This requires parallel I/O sub-systems to support on-line storage, involving providing a real-time algorithm for assigning objects to multiple disks.

Many data assignment algorithms have been proposed [3,4,5,6,7]. However, these algorithms employ off-line methods. Although off-line algorithms can produce optimized results, they are not suitable for real-time environment. Lin-Wen Lee et al. [6] have introduced an on-line algorithm for file assignment based on Mean Response Time (MRT), and assumed that the workload characteristics are known in advance and the service time of files is fixed. However, in some situation, it is unreasonable to suppose that the characteristics of workload are known *a priori*. To tackle this problem, Scheuermann P. et al. [7] present a method to keep tracking of the change of workload and balance the heat (access rate) of disks using temperature (ratio between block heat and block size) as the criterion for selecting blocks in disks to be reallocated. Balancing heat of disks can reduce system response time, but only consider the access rate as the metric of response time is insufficient. To evaluate the response time of disks, we also need to consider the service time of objects, as well as the access rate of objects.

In this paper, we propose a dynamic algorithm for object assignment adopting MRT as cost function in real-time environment. Different from other algorithms, our algorithm can learn the parameters of workload and adaptively update information according to object access history.

The rest of this paper is organized as follow: In section 2 we introduce the object assignment in OSDs. Section 3 describes the dynamic assignment algorithm. Section 4 gives the simulation results and analyses. At last, we conclude the paper.

2 Object Assignment in OSD

OSD is a real-time system which exports an object-based interface. It contains processor, RAM, multiple Gigabit Ethernet channels and multiple high-speed disk drives, allowing it to process OSD commands (e.g., CREATE_OBJECT, READ_OBJECT and WRITE_OBJECT [8]) and perform much more sophisticated object management.

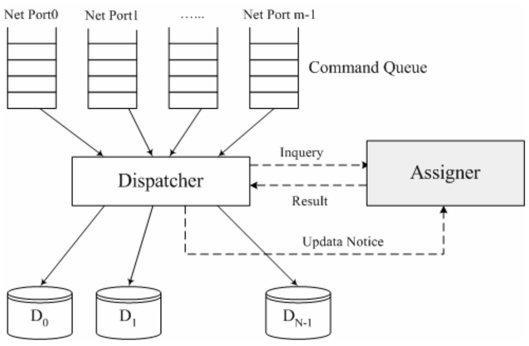


Fig. 1. Object Assignment in OSD. m is the total number of network ports, and N is the total number of disks in OSD

The process of OSD commands is shown in Fig. 1. Each network port has an I/O command queue, and all the commands are scheduled by the Dispatcher. When a CREATE_OBJECT command is processed, the Dispatcher should assign one of the disks for the incoming object. The Assigner is the decision maker for object assignment. It returns the most optimized results in response to the inquiry of the Dispatcher. The Dispatcher also sends update notice to the Assigner after processing an I/O command, so the Assigner can maintain up-to-date information.

The goal of assignment is to minimize the MRT of disks and reduce the variance of MRT between disks. The Assigner uses the dynamic greedy assignment policy to choose a disk with the minimal MRT. In this way, we can do real-time load balancing among all the disks.

Usually, an M/G/1 queue is used to model a single disk. Suppose that D_k ($0 \le k \le N-1$) denotes the k^{th} disk in OSD, and the bandwidth of the D_k is B_k. Let OS_k be the set of indices corresponding to the objects which are stored in D_k, and $O_{k,i}$ be the i^{th} object in D_k. According to the model of M/G/1, the MRT of D_k can be given as following three formulas [6]:

$$MRT_k = E(T_{w,k}) + E(T_{s,k}) = \frac{\lambda_k E(T_{s,k}^2)}{2\left[1 - \lambda_k E(T_{s,k})\right]} + E(T_{s,k}), \tag{1}$$

$$E(T_{s,k}) = \sum_{i \in OS_k} \lambda_{k,i} \frac{S_{k,i}}{B_k} \Big/ \sum_{i \in OS_k} \lambda_{k,i}, \tag{2}$$

$$E(T_{s,k}^2) = \sum_{i \in OS_k} \lambda_{k,i} \left(\frac{S_{k,i}}{B_k}\right)^2 \Big/ \sum_{i \in OS_k} \lambda_{k,i}. \tag{3}$$

$T_{r,k}$, $T_{w,k}$ and $T_{s,k}$ denote the response time, wait time and service time of D_k respectively. λ_k is the total object access rate in D_k, and $\lambda_k = \sum_{i \in OS_k} \lambda_{k,i}$, where $\lambda_{k,i}$ is the access rate for $O_{k,i}$. $S_{k,i}$ denotes the mean request size for $O_{k,i}$. From the formula (1)~(3), we get:

$$MRT_k = \frac{1}{B_k}\left[\frac{P_k}{2(B_k - Q_k)} + \frac{Q_k}{\lambda_k}\right]. \tag{4}$$

Here $P_k = \sum_{i \in SO_k} \lambda_{k,i} S_{k,i}^2$ and $Q_k = \sum_{i \in SO_k} \lambda_{k,i} S_{k,i}$. Formula (4) is the cost function for object assignment. OSD maintains the real-time value MRT_k ($0 \le k \le N-1$) for each disk. When disk allocation is requested, OSD selects the disk with the minimal MRT.

Note that the value of MRT_k is computed by $\lambda_{k,i}$ and $S_{k,i}$. Although we know nothing about the characteristics of workload, we use an adaptive method to learn and dynamically update the value of $\lambda_{k,i}$ and $S_{k,i}$.

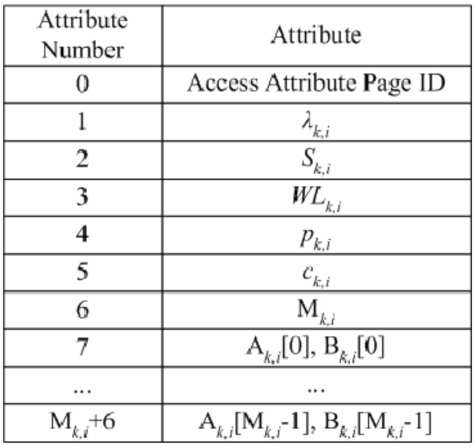
3 Dynamic Object Assignment Algorithm

The load characteristics of objects need to be tracked dynamically since they change with time. To do this, OSD records the history access information of $O_{k,i}$ within a moving window which has the length $WL_{k,i}$.

We treat the history information as attributes of object. Here we define a new attribute page following the T10 OSD protocol to save this information [8]. The page, named Access Attribute Page, is shown in Fig. 2.

In Fig. 2, the attributes with subscript "k,i" denotes the attributes belonging to the i^{th} object in D_k. Array $A_{k,i}[j]$ and $B_{k,i}[j]$ ($0 \le j \le M_{k,i}-1$) store the access time and request size for the same request. Here $M_{k,i}$ is the maximal length of array. $M_{k,i}$ can be determined by $M_{k,i} \le WL_{k,i} \bullet MAX(\lambda_{k,i})$, where $MAX(\lambda_{k,i})$ is the maximal possible access rate of $O_{k,i}$. The pointer $p_{k,i}$ is the index of array $A_{k,i}$ and $B_{k,i}$, and points to the latest request information within $WL_{k,i}$. The counter $c_{k,i}$ is the total access number of the object in $WL_{k,i}$.

We also need to store the up-to-date value of Q_k, P_k and MRT_k. Because they are only related to D_k, we save them in attribute page of Root object in D_k.



Attribute Number	Attribute
0	Access Attribute Page ID
1	$\lambda_{k,i}$
2	$S_{k,i}$
3	$WL_{k,i}$
4	$p_{k,i}$
5	$c_{k,i}$
6	$M_{k,i}$
7	$A_{k,i}[0]$, $B_{k,i}[0]$
...	...
$M_{k,i}+6$	$A_{k,i}[M_{k,i}-1]$, $B_{k,i}[M_{k,i}-1]$

Fig. 2. Access Attribute Page for $O_{k,i}$

Using the attributes of objects, the Assigner automatically learns load characteristics and adaptively assigns objects. When inquired by the Dispatcher, the Assigner should return a disk ID with the minimal MRT. While I/O command except for CREATE_OBJECT is processed, the Assigner will be informed to update values of $\lambda_{k,i}$ and $S_{k,i}$ and recompute MRT_k. The estimated value of $\lambda_{k,i}$ can be obtained from the access frequency within $WL_{k,i}$. Similarly, $S_{k,i}$ can be get from the average request size within $WL_{k,i}$.

The message passing between the Dispatcher and the Assigner contains following items:

- *Type*: The type of the message;
- O_i: The object to be accessed, and i is the index of the object;
- k: The index of disk which O_i is located in;
- O: The object to be created;
- m: The index of disk which O will be assigned to;
- *CT*: Current time;
- *RS*: Request size for O_i in D_k

The pseudocode for the dynamic assignment algorithm used in the Assigner is depicted as follows:

```
for(;;){
    Receive message from the Dispatcher;
    if($Type \neq$ CREATE_OBJECT_NOTICE){
        Get $Q_k$ and $P_k$ from Root object attribute page;
        Get Access Attribute Page associated with $O_i$ in $D_k$, as is shown in Fig. 2;
    }
    switch($Type$){
        case CREATE_OBJECT_NOTICE:
                For $0 \le j \le N-1$, get $MRT_j$ from Root object attribute page in $D_j$;
                $MRT_m$=MIN{$MRT_j$ | $0 \le j \le N-1$ and $Q_j/B_j<1$};
                if(Can not find $MRT_m$)
                    exit();            /* OSD is overloaded*/
                Assign $O$ to $D_m$ (Suppose the index of $O$ is $l$ after $O$ is created);
                Create Access Attribute Page associated with $O_l$ in $D_m$;
                $\lambda_{m,l} = S_{m,l} = p_{m,l} = c_{m,l} = 0$;
                Initialize $M_{m,l}$ and $WL_{m,l}$ according to the type of $O_l$;
                Initialize $A_{m,l}[j]$ and $B_{m,l}[j]$ ($0 \le j \le M_{m,l}-1$);
                break;
        case REMOVE_OBJECT_NOTICE:
                $Q_k = Q_k - \lambda_{k,i} S_{k,i}$ ; $P_k = P_k - \lambda_{k,i} S_{k,i}^2$ ;
                Update $MRT_k$ using formula (4);
                break;
        case READ_OBJECT_NOTICE:
        case WRITE_OBJECT_NOTICE:
                X=Φ; c=$c_{k,i}$;
                if($p_{k,i}+1<c_{k,i}$)
                        init_index = $p_{k,i}+M_{k,i}-c_{k,i}+1$;
                else
                        init_index = $p_{k,i}-c_{k,i}+1$;
                $p_{k,i}=p_{k,i}+1$; $A_{k,i}[p_{k,i}]=CT$; $B_{k,i}[p_{k,i}]=RS$; $c_{k,i}=c_{k,i}+1$;
                index = init_index;
                while(index $\neq$ init_index){
                        if($A_{k,i}$[index]<$CT-WL_{k,i}$){
                                $c_{k,i}=c_{k,i}-1$;
                                index=(index+1)%$M_{k,i}$;
                                X=X$\cup${index};
                        }
                }
                $\lambda'_{k,i} = c_{k,i} / WL_{k,i}$ ; $S'_{k,i} = (S_{k,i}\Box c - \sum_{j \in X} B_{k,i}[j] + RS) / c_{k,i}$ ;
                $Q_k = Q_k - \lambda_{k,i} S_{k,i} + \lambda'_{k,i} S'_{k,i}$ ; $P_k = P_k - \lambda_{k,i} S_{k,i}^2 + \lambda'_{k,i} S'^2_{k,i}$ ;
                $\lambda_{k,i} = \lambda'_{k,i}$ ; $S_{k,i} = S'_{k,i}$ ;
                Update $MRT_k$ using formula (4);
    }
}
```

It should be pointed out that different types of objects need different $WL_{k,i}$. For example, a multimedia object should have a short $WL_{k,i}$ since users access pattern changes frequently, whereas a history archive object needs a long $WL_{k,i}$ due to its low access frequency.

Note that the utilization of D_k is Q_k/B_k. We avoid assigning objects to an overloaded disk when its utilization is close to 1 and assign object to the disk which satisfy the condition $Q_k/B_k<1$.

4 Simulation Results and Analyses

The simulation is based on the synthetic workload. In our tests, NO=5000 objects are partitioned to N=4 disks. The objects are numbered from 0 to 4999, and created in OSD with the ascending order. For the sake of simplification, objects will not be removed from OSD after they are assigned to disks. In order to generate unbalanced MRT across disks, the access rate and request size of objects are distributed according to exponent distribution. To make sure that the utilization of each D_k ($0 \le k \le N-1$) is less then 1, the mean values of the access rate and request size of objects in D_k , that is $\bar{\lambda}_k$ and $\bar{S}_k$, will satisfy the following condition: $\bar{\lambda}_k \square \bar{S}_k < \frac{N}{NO} B_k$. In our experiments, we set $\bar{\lambda}_k = 2$ (accesses/minute), $\bar{S}_k = 20$ (ms). We also assume that all the disks have the same bandwidth, that is B_k=1.

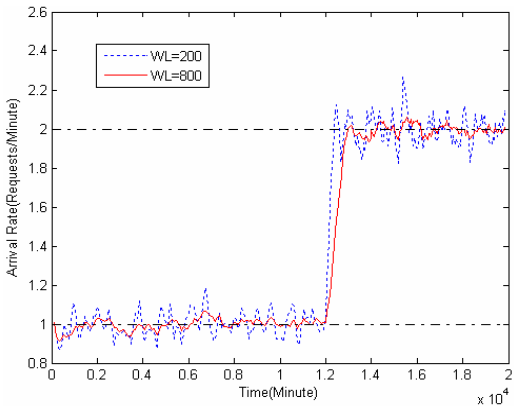


Fig. 3. Workload characteristics tracking

In order to evaluate the effect of workload tracking, we produce an access sequence. For each object, we generate a Poisson arrival sequence according to its arrival rate. The item in the sequence indicates the time the object will be accessed. We also associate each item with a service time according to the mean service time of the object. Then we synthesize all the sequences into one big sequence in ascending time order and input it into our simulation program. The program reads the sequence and gets the object access time and service time, and dynamically estimates the mean arrival rate and mean service time for each object. Fig. 3 plots the arrival rate estimates for one object. The object has the changing workload with Poisson request arrival rate. At first the mean arrival rate $\bar{\lambda}$ is 1, and then changes to $\bar{\lambda}$=2. We can see that the estimated value fluctuates around its actual value, which means that our algorithm can successfully learn the workload. In order to show the influence of the length of the moving window (*WL*), we set two window length, that is *WL*=200(minutes) and *WL*=800. Fig. 3 tells us that the longer the length of window is, the more precise the estimated value can get. But the long window also leads to low sensitivity to the change of workload. In Fig. 3, the curve of *WL=200* keeps up with the changing workload more quickly (but with larger fluctuation) than the curve of *WL*=800. So it is important to choose an appropriate length of moving window.

We compare the dynamic assignment algorithm with a random assignment algorithm. The MRT of each disk and the covariance of MRT of all the disks are plotted in object creating order as shown in Fig. 4 and 5. The results show that, under dynamic algorithm, the covariance of MRT among disks tends to be smaller, while the covariance under random algorithm rapidly increases with more and more object being assigned to disks. It is worth to notice that there is some fluctuation at the beginning of curve. This is because the accumulated MRT of each disk is small at first, and any coming object makes the MRT change drastically.

We also study the efficiency of the dynamic object assignment when the workload characteristics changes. To do this, we randomly change the access rate and request size of all the assigned objects after the 5000th object is assigned. Due to the change of workload, the MRT of each disk is markedly different from others. Then we generate a new workload with another 4000 new objects and continue to assign these objects to disks. In this way we evaluate the adaptability of our algorithm, and the results are drawn in Fig. 6.

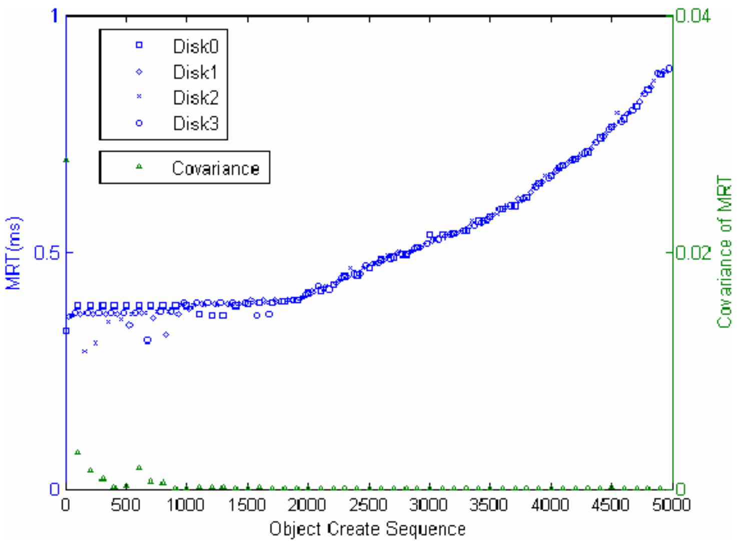


Fig. 4. Effect of dynamic algorithm

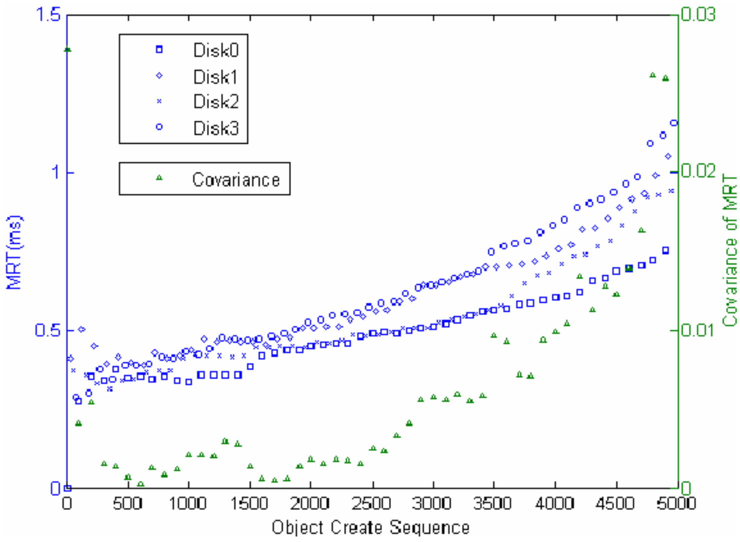


Fig. 5. Effect of random algorithm

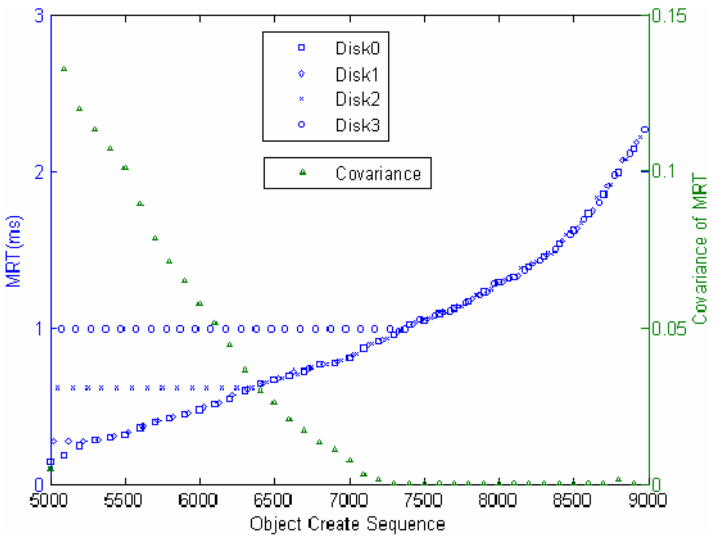


Fig. 6. Effect of dynamic algorithm after workload changes

From Fig. 6, we can see that the algorithm quickly learns the new workload parameters and has the ability to balance the MRT among disks. With more objects assigned, the covariance of MRT drops to a low value, despite the covariance was large before.

5 Conclusions

In this paper, we have presented a dynamic object assignment algorithm that aims at minimizing the covariance of Mean Response Time (MRT). The algorithm can learn the parameter of the workload under the real-time environment with the changing workload. The simulation results show that the algorithm can effectively balance the MRT across disks.

Acknowledgements

This work was supported by "973" project of China No.2004CB318201, National Science Foundation of China No.60273074 and Huo Yingdong Education Foundation No.91068.

References

1. Mesnier M., Ganger G.R., Riedel, E.: Object-based Storage. IEEE Communications Magazine, Vol 41, No. 8. (2003)84-91
2. Sahai, A.K.: Performance aspects of RAID architectures. Performance, Computing, and Communications Conference, Phoenix, USA. (1997) 321-327
3. Pattipati K.R., and Wolf, J.L.: A file assignment problem model for extended local area network environments. 10th International Conference on Distributed Computing Systems, Paris, France. (1990) 554-561
4. Xiangwu Meng, and Hu Cheng: Solving file allocation problem based on genetic algorithms. Journal of Software, Vol 8, No. 2. (1997) 122-127
5. Deng-Jyi Chen, Ruey-Shun Chen, Hol W.C., and Ku K.L.: A heuristic algorithm for the reliability-oriented file assignment in a distributed computing system. International Conference on Parallel and Distributed Systems, Hsinchu, Taiwan. (1994) 454-459
6. Lin-Wen Lee, Scheuermann P., Vingralek R.: File assignment in parallel I/O systems with minimal variance of service time, IEEE Transactions on Computers, Vol 49, No.2. (2000) 127-140
7. Scheuermann P., Weikum G., Zabback P.: Data partitioning and load balancing in parallel disk systems, The VLDB Journal - The International Journal on Very Large Data Bases, Vol. 7, No. 1. (1998) 48-66
8. Draft OSD Standard, T10 Committee, Storage Networking Industry Association (SNIA), ftp://ftp.t10.org/t10/drafts/osd/osd2r00.pdf