Hindawi
Scientific Programming
Volume 2020, Article ID 6843572, 10 pages
https://doi.org/10.1155/2020/6843572

Research Article

MQHD: Dynamic Load Balancing Method for Energy-Aware Storage Systems

Xiaoling Xie

College of Medical Information Engineering, Guangdong Pharmaceutical University, Guangzhou 510006, China

Correspondence should be addressed to Xiaoling Xie; xiexiaoling9@163.com

Received 17 September 2019; Revised 27 September 2020; Accepted 24 November 2020; Published 7 December 2020

Academic Editor: Antonio J. Peña

Aiming at the problem that some disks of energy-aware storage systems are easily overloaded, a dynamic load balancing method based on multiqueue and heat degree (MQHD) is proposed. According to data popularity, MQHD divides data into multiple least recently used (LRU) queues by data access frequency and access temporal locality and uses heat degree to measure the load pressure brought by each data unit to a disk. When a disk is overloaded, MQHD calculates the load pressure ratio (LPR) according to the disk overload degree and then selects some appropriate data to migrate according to the LPR. The experimental results show that, compared with the popular data concentration method (PDC), the workload-adaptive management method (WAM), and the energy model-based file migration strategy (EM-FMS), MQHD is the most effective. Under given experimental conditions, the request change ratio (RCR) value of MQHD is 0.371, EM-FMS is 0.2872, and WAM is 0.0114. Compared with EM-FMS, MQHD has better rapidity and less overhead.

1. Introduction

In a medium-sized data center, the energy cost of storage system can be as high as $700 million to $900 million per year [1]. In order to reduce the energy consumption of storage system, various types of energy-aware storage systems have been proposed. The practice of most energy-aware storage systems is to tilt the access load to some disks, so that other disks can enter a low-power standby state or completely shut down to achieve energy saving [2, 3]. This practice solves the energy-saving problem of the storage system, but it also brings a side effect; that is, the disks with skewed load can easily be overloaded, resulting in a delayed response of requests. It is important to balance the load of these disks dynamically, solve the overload problem of these disks, and eliminate this side effect.

The basic principle of the dynamic load balancing method is that when a disk is detected to be overloaded, migrate the hot data of the disk to other light-loaded disks until the disk is no longer overloaded. The dynamic load balancing method needs to solve two key problems [4, 5]: (i) How to recognize hot data? (ii) How much data should be migrated at a time?

Aiming at the two key problems, a dynamic load balancing method based on multiqueue and heat degree (MQHD) is proposed in this paper. MQHD can recognize hot data accurately based on multiqueue and only maintain the hot data as a migration object, improve the effectiveness of load adjustment, and reduce the overhead. In addition, MQHD uses heat degree to measure the load pressure brought by each data unit to a disk and conducts batch data migration according to the load state of each disk so as to improve the speed and effectiveness of load adjustment. The experimental results show that, compared with the current popular dynamic load balancing methods for energy-aware storage systems, MQHD is the most effective and has better rapidity and less overhead.

The rest of this paper is organized as follows. Section 2 summarizes the relevant research work. Section 3 introduces MQHD in detail. Section 4 verifies MQHD. Section 5 summarizes this paper and outlines future work.

2. Related Work

In order to prevent the disks that store hot data from being overloaded, the popular data concentration method (PDC) estimates the future load of each disk. The expected load

associated with a file is estimated by dividing its size by its average access time interval. PDC migrates data to a disk periodically until the expected load of the disk is close to its maximum bandwidth [6, 7]. PDC is a simple and convenient solution to the problem of disk overload, its overhead is small, but the effect is difficult to guarantee. When the load suddenly increases, the system is difficult to adjust in real time, and the effect is poor.

In order to achieve load balancing among hot disks, a dynamic load balancing method for an energy-aware storage system named workload-adaptive management (WAM) was proposed [1]. Whenever a request arrives at a hot disk, WAM checks whether the disk's request arrival speed is greater than the overload threshold. If so, the disk will be marked as a hot disk. Next, for each request that arrived at the disk, WAM locates the data block of the request after servicing it and then exchanges the data block with an unaccessed data block of an unoverloaded disk. Such exchange will continue until the disk's request arrival speed is less than or equal to the target threshold, and then the hot tag will be removed from the disk. WAM detects the request arrival speed of the disk and compares it with the overload threshold and target threshold. If the disk's request arrival speed exceeds the overload threshold, the data block will be exchanged until the disk's request arrival speed is less than or equal to the target threshold. WAM can always make the request arrival speed of the hot disk less than or equal to the target threshold, but the exchange only occurs when the request arrives at the disk, and only the requested data block is exchanged at a time, which makes it difficult to quickly and effectively eliminate the overload state of the hot disk [8].

In order to achieve load balancing of a heterogeneous storage system, an energy model-based file migration strategy (EM-FMS) was proposed [9]. The file energy model is related to the file's number of accesses, concurrent access situation, size, and access interval. EM-FMS determines whether a storage unit triggers the file migration operation according to the available bandwidth of the storage unit and the probability-based migration trigger function. For all storage units that trigger the file migration operation, EM-FMS selects the file with the highest energy density from the currently accessed files for migration. EM-FMS uses an energy model to measure the load pressure on the storage system caused by the file, which is beneficial to select correct files for migration. However, too much calculation is needed to calculate the energy value of each file, which results in a large system overhead. At the same time, because only the file with the highest energy density is selected for migration at a time, it is difficult to quickly eliminate the overload state of the overloaded storage unit [10].

Therefore, although the above research works have achieved some results, the high overhead, low speed, and poor effectiveness problems of dynamic load balancing in energy-aware storage systems have not been comprehensively solved yet.

3. MQHD: Dynamic Load Balancing Method Based on Multiqueue and Heat Degree

3.1. Hot Data Recognition Based on Multiqueue. When a disk is overloaded, some data need to be migrated for load adjustment, so it is important to recognize hot data accurately to select the appropriate data for migration. The multiqueue (MQ) algorithm is a proven effective cache management method [11]. Based on the idea of the MQ algorithm, this paper proposes a hot data recognition method based on multiqueue which is used to identify the data for migration.

For the convenience of describing the method, the following definitions are given.

Definition 1 Hot Data Unit (HDU). It refers to a data unit that has been frequently accessed recently. It is the migration object of load adjustment. According to the different interface levels of an energy-aware storage system, a data unit represents a data block or a file.

Definition 2 Queue. According to the popularity of HDUs, the HDUs are divided into multiple queues by access frequency and access temporal locality. These queues use the LRU rules to arrange HDUs and are represented by Q_i, where i ($i = 0, 1, 2, \ldots,)$ indicates the popularity level of the queue. The larger the number, the higher the popularity level of the queue, such as $Q_0 < Q_1 < Q_2 < \cdots$.

Definition 3 Life Cycle (LC). It refers to the allowed time that an HDU has not been accessed in a queue Q_i. When an HDU stays in a queue Q_i for longer than LC without any request access to it, this HDU will be degraded to the next lower-level queue Q_{i-1}. If this HDU is in the queue Q_0, it will be removed from the queue.

Definition 4 Expire Time (ET). It refers to the time when an HDU needs to be removed from a queue. It is equal to the sum of the time at which this HDU was last accessed and the LC of this HDU.

Definition 5 Logical Time. It refers to the time measured by the number of accesses. The first access of the disk is time 1, and the second access is time 2. The LC and ET defined above are measured by logical time.

The hot data recognition method based on multiqueue is described as follows:

(i) The system maintains multiple LRU queues named as $Q_0, Q_1, \ldots, Q_n$, as shown in Figure 1.

(ii) When an HDU enters a queue, the HDU is placed at the end of the queue and its ET is set to the sum of the current time and LC, as shown in Figure 2.

(iii) Each queue has a maximum access count. If an HDU in the queue is accessed more than 2^i times, then the HDU will be promoted to the end of the queue Q_{i+1} and its ET will be set to the sum of the current time and LC, as shown in Figure 3.

(iv) For each access, compare the ET of the HDU of each queue head with the current time. If the former is smaller than the latter and the queue in which the HDU is located is not the queue Q_0, the HDU will be moved to the end of the next lower-level queue, and its number of accesses will be halved and its ET is reset to the current time plus LC, as shown in

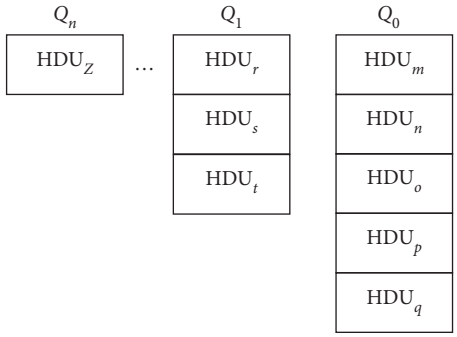

Figure 1: Multiqueue hot data.

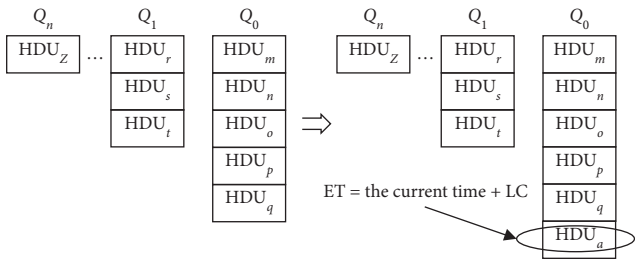

Figure 2: HDUa enters Q_0.

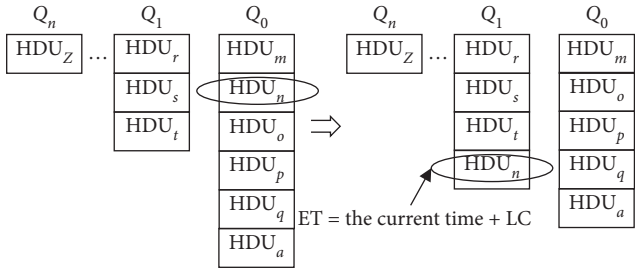

Figure 3: HDUn is promoted from queue Q_0 to queue Q_1.

Figure 4. If the former is smaller than the latter and the queue in which the HDU is located is the queue Q_0, the number of accesses of the HDU will be set to 0 and the HDU will be cleared out of the queue, as shown in Figure 5.

The pseudocode of the hot data recognition Algorithm 1 is described as follows.

3.2. Heat Degree. After recognizing the hot data, a certain number of HDUs should be selected for load adjustment, which is related to the rapidity of load adjustment. Since the load pressure on the disk caused by each HDU is different, the load pressure needs to be measured by an appropriate parameter. In this paper, heat degree is used as the parameter, as defined in Definition 6.

Definition 6 Heat Degree (HD). It reflects the degree of the load pressure that an HDU brings to a disk in a certain period of time. It is represented by a number between (0, 1]. The larger the number, the greater the load pressure that HDU brings to a disk. The sum of HDs of all HDUs on the same disk is equal to 1.

In general, the HD of an HDU is related to the following two factors:

(i) Access frequency: The HDU that is with more number of accesses in the recent period is more likely to be accessed multiple times in the future period, and its HD is greater.

(ii) Time frequency: The HDU that was frequently accessed in the past but was not accessed in a relatively long period of time is less likely to be accessed in the future, and its HD should be smaller. In other words, the HD of an HDU that is frequently accessed recently should be greater.

In the hot data recognition method based on multiqueue, the access frequency of an HDU is reflected by the queue number, and which queue the HDU is placed in is determined by

$$i = [\log_2 \text{times}]. \quad (1)$$

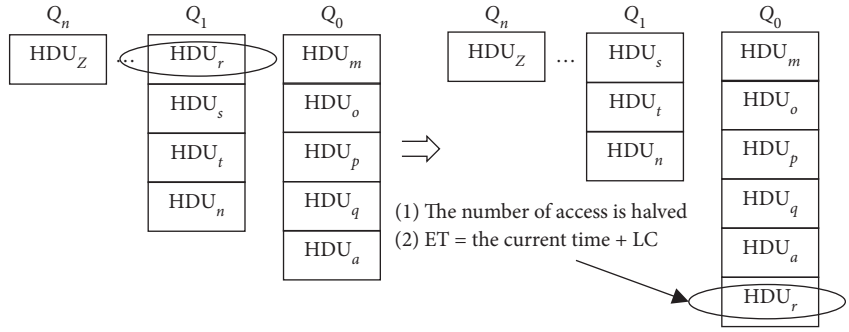

Figure 4: HDUr is degraded to the end of Q_0 from Q_1.

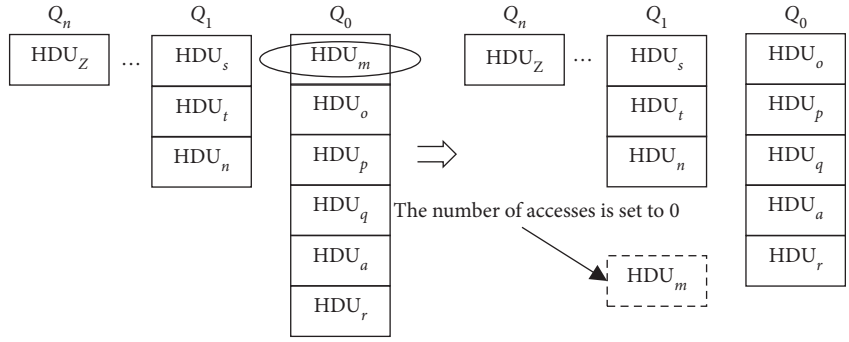

Figure 5: HDUm is cleared out of the queue.

In the above equation, i is the queue number where the HDU is located, times is the number of accesses of the HDU, and [] is a rounding symbol.

The time frequency is reflected by the LC of the HDU. For an HDU that has not been accessed during the LC period, its number of accesses will be halved and it will be moved to the end of the next lower-level queue. If this HDU is in the queue Q_0, it will be removed from the queue. Therefore, in the hot data recognition method based on multiqueue, the time frequency of HDU is also directly related to the queue number.

Based on the above considerations, equation (2) is used to calculate the HD of an HDU. In the same queue, the HDs are considered to be the same. $\mathrm{HD}_S(i)$ is used to represent the HD of a single HDU belonging to the disk S in queue i.

$$\mathrm{HD}_S(i) = \frac{2^i}{\sum_{j=0}^{n-1} L_s(j) 2^j}. \tag{2}$$

In the above equation, i is the queue number where the HDU is located, j is the queue number, n is the number of queues, and $L_S(j)$ is the number of the HDUs belonging to the disk S in queue j.

In order to measure the load pressure brought by a certain number of HDUs to a disk, the load pressure ratio is used.

Definition 7 Load Pressure Ratio (LPR). It refers to the load pressure brought by a certain number of HDUs to a disk. Its value is the sum of HD values of the selected HDUs belonging to a disk. The calculation equation is as shown in equation (3).

$$\mathrm{LPR}(S) = \sum_{i=0}^{n-1} \left(\mathrm{num}_S(i) \times \mathrm{HD}_S(i)\right). \tag{3}$$

In the above equation, i is the queue number where the HDU is located, n is the number of queues, and $\mathrm{num}_S(i)$ is the number of the HDUs belonging to the disk S selected from queue i.

3.3. MQHD Method. The MQHD method determines overloaded disks according to the load state of each disk, calculates LPR according to the overload degree of each overloaded disk, and selects appropriate HDUs for data migration according to the LPR. For every migration, MQHD considers the new load state of the overloaded disk and rediscovers the disk with the lightest load as the load migration object, thus ensuring the dynamics of the load adjustment.

MQHD uses $L(S)$ to measure the load state of disk S. According to the actual situation, $L(S)$ can be the average response time of requests, the request queue length of the disk, the system throughput, and so on [12].

In order to determine whether a disk needs to migrate data out or whether data can be migrated to the disk, two thresholds are used, namely, overload (OL) and safe load (SL). The regulations are as follows:

(i) If $L(S) > \mathrm{OL}$, it means that disk S is overloaded and needs to migrate data out.

(ii) If $L(S) < \mathrm{SL}$, it means that disk S has a light load and data can be migrated to it.

(iii) If $\mathrm{SL} \le L(S) \le \mathrm{OL}$, it means that the load of disk S lies in a hysteresis region; it is not necessary to migrate data out and cannot migrate data to it. By means of setting the hysteresis region, it can effectively avoid the situation that the same piece of data is migrated back and forth across the same disks.

In order to determine the HDUs that need to be migrated, the concepts of overload amount and safety margin are introduced, as defined as follows.

Definition 8 Overload Amount. The ratio of $L(S_n) - \mathrm{OL}$ and $L(S_n)$, where $L(S_n)$ is the load of the overloaded disk S_n, is denoted by $\alpha\%$.

$\alpha\%$ can be expressed by equation (4).

$$\alpha\% = \frac{L(S_n) - \mathrm{OL}}{L(S_n)} \times 100\%. \tag{4}$$

Definition 9 Safety Margin. The ratio of $\mathrm{SL} - L(S_m)$ and $L(S_n)$, where $L(S_m)$ is the load of the light-loaded disk S_m and $L(S_n)$ is the load of the overloaded disk S_n, is denoted by $\beta\%$.

$\beta\%$ can be expressed by

$$\beta\% = \frac{\mathrm{SL} - L(S_m)}{L(S_n)} \times 100\%. \tag{5}$$

For an overloaded disk, the load pressure caused by HDUs is measured by LPR. Therefore, the LPR value of the HDUs that need to be migrated is calculated by

$$\mathrm{LPR}(S_n) = c\alpha\%. \tag{6}$$

In the above equation, $\mathrm{LPR}(S_n)$ is the LPR value of the HDUs that need to be migrated out of the overloaded disk S_n; c is the migration-out adjustment coefficient. c is used to adjust the corresponding relationship between the overload amount and LPR, it is adjusted according to the adjustment effect during the dynamic adjustment process, and the initial value is 0.8.

For a light-loaded disk, the LPR value of the HDUs that can be migrated into it is calculated by

$$\mathrm{LPR}(S_m) = k\beta\%. \tag{7}$$

In the above equation, $\mathrm{LPR}(S_m)$ is the LPR value of the HDUs that can be migrated into the light-loaded disk S_m; k is the migration-in adjustment coefficient. k is used to adjust the corresponding relationship between the safety margin and LPR, it is adjusted according to the adjustment effect during the dynamic adjustment process, and the initial value is 0.8.

The MQHD method is described as follows:

(i) Find an overloaded disk S_n. If it was previously used as a migration-in disk (the migration-in disk flag is 1), then reduce its migration-out adjustment coefficient $c(S_n)$ by 10%. Set its migration-in adjustment coefficient $k(S_n)$ to the initial value, migration-in disk flag to 0, and migration-out disk flag to 1 and then calculate its overload amount $\alpha\%$ and the load pressure ratio $\mathrm{LPR}(S_n)$ that need to be migrated.

(ii) Find a disk S_m that has the lightest load and can be migrated data to it. If it was previously used as a migration-out disk (the migration-out disk flag is 1), then reduce its migration-in adjustment coefficient $k(S_m)$ by 10%. Set its migration-out adjustment coefficient $c(S_m)$ to the initial value, migration-out disk flag to 0, and migration-in disk flag to 1 and then calculate its safety margin $\beta\%$ and the load pressure ratio $\mathrm{LPR}(S_m)$ it can accept. If the acceptable $\mathrm{LPR}(S_m)$ of the disk is greater than the $\mathrm{LPR}(S_n)$ that needs to be migrated out of the overloaded disk S_n, then the HDUs shall be selected for migration according to $\mathrm{LPR}(S_n)$, equation (3), and the principles described in the next paragraph. Otherwise, the HDUs shall be selected for migration according to $\mathrm{LPR}(S_m)$, equation (3), and the principles described in the next paragraph.

(iii) The principles of selecting HDUs according to LPR are as follows: (i) Global perspective: try to get HDUs from each queue to avoid the problem of the excessive local error caused by selecting HDUs from a certain queue. (ii) Introducing randomness: when getting one HDU from one queue, randomly select one HDU from the queue to avoid the problem of the excessive local error caused by selecting the HDU in a fixed order.

(iv) Rejudge the load state of the overloaded disk S_n after a migration. If it is still overloaded, repeat the load adjustment process.

The pseudocode of the MQHD Algorithm 2 is described as follows.

4. Experimental Verification and Analysis

This section verifies MQHD and compares it against PDC, WAM, and EM-FMS.

4.1. Experimental Environment. Request generator and energy-aware storage system simulator are designed in the experiment.

The request generator can generate storage system request records based on the number of requests, request rate, disk coverage, data coverage, disk popularity, and data popularity specified by the user. The request rate is the number of requests that arrive at the storage system in unit time, and it is assumed that the request arrival time is subject to an exponential distribution. Disk coverage represents the number of disks that are accessed as a percentage of the number of all disks. Data coverage represents the number of data units that are accessed as a percentage of the number of all data units. Disk popularity represents the degree of centralization of disks to be accessed; it is expressed by a coefficient in the range of (0–1) in Zipf's law. The closer the

Input: a request
Output: $Q(0), Q(1), \ldots, Q(n)$
(1) current Time ⟵ current Time + 1;
(2) ET (current File) ⟵ current Time + LC;
(3) **if** access Times (current File) = 0 **then**;
(4) place the current file node at the end of $Q(0)$
(5) **else**
(6) old QueueID ⟵ $\log_2$access Times (current File);
(7) access Times (current File) ⟵ access Times (current File) + 1;
(8) new QueueID ⟵ $\log_2$access Times (current File);
(9) **if** old QueueID = new QueueID **then**
(10) place the current file node at the end of the original queue;
(11) **else**
(12) place the current file node at the end of the new queue;
(13) **end if**
(14) **end if**
(15) **if** $Q(0) \wedge$ ET $(Q(0) \cdot$ file$) \leq$ current Time **then**
(16) access Times $(Q(0) \cdot$ file$) \longleftarrow 0$;
(17) $Q(0) \longleftarrow Q(0) \cdot$ next;
(18) **end if**
(19) **for** $i = 1$ to n **do**
(20) **if** $Q(i) \wedge$ ET $(Q(i) \cdot$ file$) \leq$ current Time **then**
(21) access Times $(Q(i) \cdot$ file$) \longleftarrow$ (access Times $(Q(i) \cdot$ file$)/2)$;
(22) ET $(Q(i) \cdot$ file$) \longleftarrow$ current Time + LC;
(23) new QueueID ⟵ $\log_2$access Times $(Q(i) \cdot$ file$)$;
(24) move the head node of $Q(i)$ to the end of the new queue;
(25) **end if**
(26) **end for**

Algorithm 1: The hot data recognition algorithm.

coefficient is to 1, the more the accesses are centralized on partial disks. For a specified disk, the Zipf distribution formula is used to calculate its assessed probability, as shown in equation (8). Data popularity is similar to disk popularity in meaning, indicating the degree of concentration of data is accessed.

$$P(i) = \frac{(1/i)^{\theta}}{\sum_{j=1}^{n} (1/j)^{\theta}}. \tag{8}$$

In the above equation, $P(i)$ is the probability of disk be accessed, n is the number of disks be accessed, i and j are the ranking of the disk's accessed probability, and θ is the disk popularity.

In the experiment, the parameters of request generator are set as follows: the number of requests is 1000; the request rate of the first 200 requests is 50/s; the request rate of the last 800 requests is 300/s; the disk coverage is 25%; the disk popularity is 0.5; the data coverage is 5%; the data popularity is 0.5. Table 1 shows the parameters used by the request generator when generating records.

In the energy-aware storage system simulated by the energy-aware storage system simulator, disks are divided into two groups: data disk group and cache disk group. The disks of the data disk group are not always rotating. Some disks of the data disk group that have been not accessed for a while will be transited into a standby state according to the power management method to save energy. The disks of the cache disk group keep rotating; they store the data that is frequently accessed. The controller tries to ensure that frequently accessed data is copied to the disks of the cache disk group [11].

In the energy-aware storage system simulator, besides, MQHD, PDC, WAM, and EM-FMS are also implemented.

In order to observe the overload situation and focus on the overload situation of the local storage units, a small size energy-aware storage system is configured. The parameters of the energy-aware storage system used in the experiment are configured as follows: the number of cache disks is 2, the number of data disks is 4, the capacity of a single cache disk is 500, and the capacity of a single data disk is 1000. In the energy-aware storage system simulator, the average service time of requests is set to 8 ms. The disk load is measured by the disk's request response time, OL is set to 48 ms, and SL is set to 24 ms.

4.2. Evaluation Methods and Indicators. We evaluate the load balancing methods from three aspects: effectiveness, rapidity, and overhead.

4.2.1. Effectiveness. The effectiveness of the load balancing method means that when disk overload occurs, the method can eliminate the overload state of the disk by timely load adjustment. For an effective load balancing method, its effectiveness is measured by the request change ratio (RCR).

```
(1) procedure LOADBALANCING
(2)   while L(i) > OL do
(3)     if iflag(i) = 1 then c(i) ⟵ 0.9 × c(i);
(4)     k(i) ⟵ 0.8;
(5)     iflag(i) ⟵ 0;
(6)     oflag(i) ⟵ 1;
(7)     α(i) ⟵ (L(i) − OL/L(i)) × 100%;
(8)     LPR(i) ⟵ c(i) × α(i);
(9)     min ⟵ 1
(10)    for j = 2 to n do
(11)      if L(j) < L(min) then min ⟵ j;
(12)    end for
(13)    if L(min) < SL then
(14)      if oflag(min) = 1 then k(min)⟵0.9 × k(min);
(15)      c(min) ⟵ 0.8;
(16)      oflag(i) ⟵ 0;
(17)      iflag(i) ⟵ 1;
(18)      β(i) ⟵ (SL − L(min)/L(i)) × 100%;
(19)      LPR(min) ⟵ k(min) × β(i);
(20)      if LPR(i) ⟵ LPR(min) then
(21)        get the serial numbers of the HDUs for migration based on LPR(i);
(22)        migrage the HDUs selected for migration;
(23)      else
(24)        get the serial numbers of the HDUs for migration based on LPR(min);
(25)        migrage the HDUs selected for migration;
(26)      end if
(27)    end if
(28)  end while
(29) end procedure
```

Algorithm 2: The MQHD algorithm.

Table 1: Request parameters.

Parameter	Number of requests	Request rate	Disk coverage (%)	Disk popularity	Data coverage (%)	Data popularity
Value	1000	50 or 300	25	0.5	5	0.5

RCR is equal to the ratio of the number of requests n_{after} that arrive at the disk when using the load balancing method to the number of requests n_{before} that arrive at the disk when the load balancing method is not used:

$$\text{RCR} = \frac{n_{\text{after}}}{n_{\text{before}}}. \tag{9}$$

The larger the RCR value is, the more accurately the load balancing method can control the change of load size, without causing too much data to be migrated, and the more effective it is.

4.2.2. Rapidity. The rapidity of the load balancing method is measured by the adjusting time t_a and the average response time of requests t_r. t_a is the time taken to adjust the disk out from an overload state, which is represented in logical time in this paper. t_r is the average response time of all requests in the stage of adjusting the disk from overload to not overload.

4.2.3. Overhead. In order to select appropriate HDUs for migration during load adjustment, the load balancing method should maintain the information of a certain amount of HDUs. The load adjustment overhead is represented by the ratio of the number of HDUs maintained to the number of data units of the overloaded disk.

4.3. Experimental Results and Analysis. Figure 6 shows a cache disk response without using the load balancing method. As it can be seen from Figure 6, when the request rate is increased from 50/s to 300/s, the disk begins to overload, and the response of the disk deteriorates with time, resulting in a large number of delayed requests.

Figures 7(a)–7(d) show the disk response when using PDC, WAM, EM-FMS, and MQHD, respectively.

4.3.1. Effectiveness Analysis. PDC periodically performs hot data migration and estimates the future load of each disk as it migrates data, ensuring that the expected load of the disk is close to its maximum bandwidth. As shown in Figure 7(a), when disk overload occurred at the beginning of the 212 requests, PDC did not perform load adjustment in time. PDC did not perform load adjustment until about 300 requests, when the timing migration time arrived, which had caused a large number of requests to be

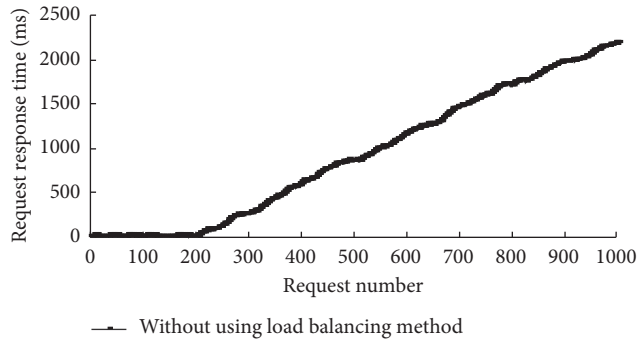

Figure 6: A cache disk response without using the load balancing method.

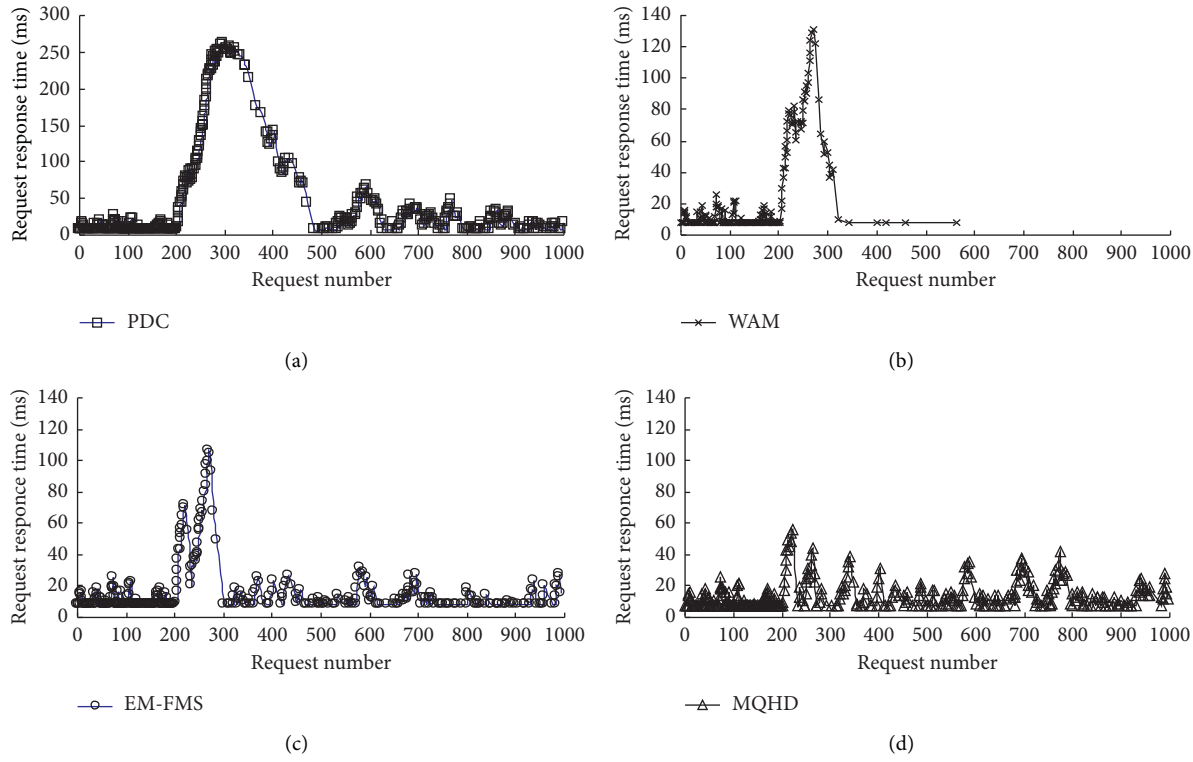

Figure 7: Disk response after using PDC, WAM, EM-FMS, and MQHD, respectively. (a) PDC. (b) WAM. (c) EM-FMS. (d) MQHD.

delayed. It can be seen that the PDC's adjustment is affected by the timing time. When disk overload occurs, PDC takes more time to respond to load adjustment.

When disk overload occurs, for each request that arrived at the disk, WAM locates the data block of the request after servicing it and then exchanges the data block with an unaccessed data block of an unoverloaded disk. Such an exchange will continue until the disk's request arrival speed is less than or equal to the target threshold. Because there is no distinction between the popularity of data, WAM is blind in choosing blocks, which may cause too much data to be migrated and cause the RCR to be too small, so that the effectiveness of load adjustment is bad. As shown in Figure 7(b), after load adjustment by WAM, disk requests are greatly reduced, and the RCR value is only 0.0114. Therefore, the WAM method is less effective.

When disk overload occurs, EM-FMS selects the file with the highest energy density from the currently accessed files for migration. Because the popularity of data has been distinguished, EM-FMS is specific in selecting files, so its load adjustment is more effective. As shown in Figure 7(c), after load adjustment, the disk request is not greatly reduced, and the RCR value is 0.2872.

When the disk is overloaded, MQHD determines the appropriate LPR according to the overload degree of the overloaded disk and selects appropriate HDUs to perform data migration according to the LPR. By considering the overload degree and the heat degree of HDU, MQHD is more targeted than other methods. As shown in Figure 7(d), after load adjustment, the disk utilization is still high, the RCR value is 0.3721.

It can be seen that, among the four methods, PDC takes more time to respond to load adjustment and WAM is less

Table 2: RCR of PDC, WAM, EM-FMS, and MQHD.

Method	PDC	WAM	EM-FMS	MQHD
RCR	0.3013	0.0141	0.2872	0.3721

Table 3: t_a and t_r of PDC, WAM, EM-FMS, and MQHD.

Method	PDC	WAM	EM-FMS	MQHD
t_a (logic time)	422	108	85	20
t_r (ms)	114.1	70.4	55.8	37.8

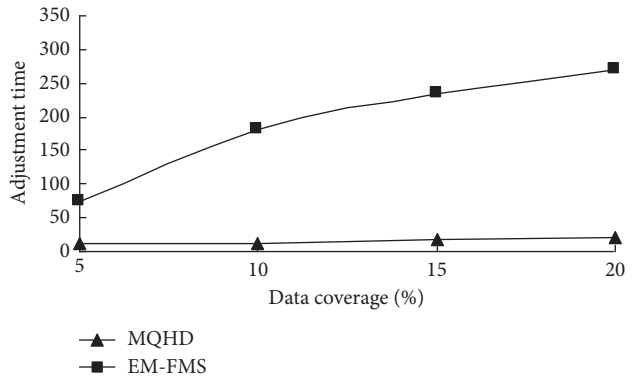

Figure 8: Adjustment time comparison between MQHD and EM-FMS.

effective, while EM-FMS and MQHD are more effective. Table 2 shows RCR of PDC, WAM, EM-FMS, and MQHD.

4.3.2. Rapidity Analysis. Table 3 shows t_a and t_r of PDC, WAM, EM-FMS, and MQHD. As it can be seen from Table 3, among the four methods, MQH and EM-FMS have better rapidity, because they distinguish the popularity of data and adjust the load more targeted.

Figure 8 shows how the adjustment time of MQHD and EM-FMS varies with data coverage. As it can be seen from the figure, MQHD has better rapidity than EM-FMS. Under different data coverage, the adjustment time of MQHD method is small and has little change, while the adjustment time of EM-FMS is large and increases with the increase of data coverage. This is because MQHD can select an appropriate number of HDUs to migrate according to the overload degree of the disk, so the overload state of the disk can be quickly eliminated. However, EM-FMS only migrates one file at a time, and because of the error in energy estimation of a single file, the adjustment time of EM-FMS is longer, and the adjustment time will increase as the data coverage increases.

4.3.3. Overhead Analysis. Figure 9 shows the overhead comparison between MQHD and EM-FMS. As can be seen from the figure, the overhead of EM-FMS is much larger than that of MQHD. This is because EM-FMS needs to maintain access information for all data at all times while MQHD only needs to maintain access information for the data that has been frequently accessed recently.

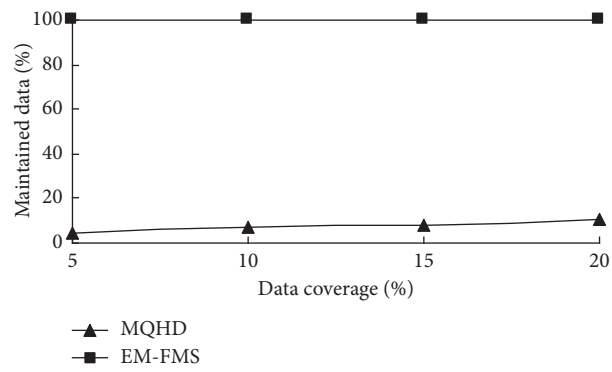

Figure 9: Overhead comparison between MQHD and EM-FMS.

5. Conclusion

Energy-aware storage system brings the problem that some disks are easily overloaded while achieving energy saving. To solve this problem, a dynamic load balancing method named MQHD for energy-aware storage systems is proposed in this paper, which relies on multiple queues to determine what data and in what amount to migrate to other disks. The method is verified by means of custom request generator and energy-aware storage system simulator. The results show that the method can effectively solve the problem that some disks of the energy-aware storage system are easily overloaded. The application of the method can improve the performance of the energy-aware storage system, which is beneficial to promote the development of energy-aware technology of the storage system.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The author declares no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61501128, the Young Innovative Talents Project of Guangdong Universities under Grant no. 2017KQNCX107, the Medical Science and Technology Research Fund Project of Guangdong under Grant no. A2018101, the Key Research Platforms and Projects of Colleges and Universities in Guangdong Province under Grant no. 2020ZDZX3060, and the College Level Scientific Research Project of Guangdong Pharmaceutical University under Grant no. 201704.

References

[1] E. Otoo, D. Rotem, and S. C. Tsao, "Workload-adaptive management of energy-smart disk storage systems," in *Proceedings of the 2009 IEEE International Conference on Cluster*

Computing and Workshops, pp. 1–11, IEEE, New Orleans, LA, USA, 2009.

[2] A. Kumar, R. Tandon, and T. C. Clancy, "On the latency and energy efficiency of distributed storage systems," *IEEE Transactions on Cloud Computing*, vol. 5, no. 2, pp. 221–233, 2017.

[3] Z. Sun, Q. Zhang, Y. Li et al., "DPPDL: a dynamic partial-parallel data layout for green video surveillance storage," *IEEE Transactions on Circuits & Systems for Video Technology*, vol. 28, no. 1, pp. 193–205, 2018.

[4] Y. Xiong, Z. Cheng, C. Lu et al., "An energy-aware workload balancing method for cloud video data storage management," in *Proceedings of the 2016 International Conference on Advanced Cloud and Big Data*, pp. 7–12, IEEE, Chengdu, China, 2016.

[5] Z. Wang, H. Chen, Y. Fu, D. Liu, and Y. Ban, "Workload balancing and adaptive resource management for the swift storage system on cloud," *Future Generation Computer Systems*, vol. 51, pp. 120–131, 2015.

[6] E. Pinheiro and R. Bianchini, "Energy conservation techniques for disk array-based servers," in *Proceedings of the 18th Annual International Conference on Supercomputing*, pp. 68–78, ACM, Malo, USA, 2004.

[7] J. C.-Y. Chou, T.-H. Lai, J. Kim, and D. Rotem, "Exploiting replication for energy-aware scheduling in disk storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2734–2749, 2015.

[8] P. Behzadnia, Y. C. Tu, B. Zeng et al., "Energy-aware disk storage management : online approach with application in DBMS," *International Journal of Database Management Systems*, vol. 9, no. 1, pp. 1–22, 2017.

[9] H. Q. Huang, X. H. Song, and Y. D. Cao, "File migration strategy based on energy model in heterogeneous storage system," *Journal of Beijing University of Aeronautics and Astronautics*, vol. 33, no. 9, pp. 1107–1111, 2007.

[10] G. Liu, H. Shen, and H. Wang, "Towards long-view computing load balancing in cluster storage systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1770–1784, 2017.

[11] Y. Zhou, J. F. Philbin, and K. Li, "The multi-queue replacement algorithm for second level buffer caches," in *Proceedings of the General Track: 2001 USENIX Annual Technical Conference*, pp. 91–104, ACM, Berkeley, CA, USA, 2001.

[12] S. Eswaran and M. Rajakannu, "Multiservice load balancing with hybrid particle swarm optimization in cloud-based multimedia storage system with QoS provision," *Mobile Networks & Applications*, vol. 22, no. 4, pp. 1–11, 2017.