FDTM: Block Level Data Migration Policy in Tiered Storage System

Xiaonan Zhao, Zhanhuai Li, and Leijie Zeng

School of Computer Science and Technology,
Northwestern Polytechnical University, Xi'an 710072, China,
{zhaoxn, lizhh, zenglj}@nwpu.edu.cn

Abstract. ILM and tiered storage system are designed to address the challenge of achieving balance between cost and storage performance. However, both of them are hard to implement fully automatic data migration by traditional solutions for which are mainly relying on administrators experience and need huge manual work for data migration according to storage configuration and IO access patterns. This paper proposes a novel bi-directional migration policy FDTM based on block-level data valuation and fully automatic migration process. FDTM aims to get a trade-off between storage QoS and migration costs by introducing double thresholds to narrow the migration scope of block-level data objects. Experiment and analysis show that FDTM is efficient at block-level data migration comparing with traditional migration policies. In addition, it could help pave the way to implement tiered storage system with fully automatic data migration.

Keywords: Data migration policy; data valuation; feedback; Tiered Storage System

1 Introduction

Industry research has shown that 70-80% of all storage data is inactive, and data is rarely accessed after 30-90 days[11]. So in business perspective, the data value is declining over time dramatically. SNIA proposes ILM to address this issue[18]. In addition tiered storage as "tiering ILM", is the key component in an ILM practice, but not all. Generally, there are three types of storage in tiered storage system: on-line, near-line and off-line [19], in which data is classified into distinct classes and stores in different tier separately with considering performance, availability, recoverability, safety and other requirements. A tiered storage system is usually composed of expensive FC disk arrays for high performance accessing, cheaper SATA storage subsystems for data staging and high-capacity robotic tape libraries for data archiving, which could decrease the system's TCO while keeping the storage performance at the same time. Fig. 1 shows the infrastructure of a tiered storage system. In a tiered storage management system, data classification, data placement and data migration are the core function components. Furthermore data migration is also helpful to optimize storage system

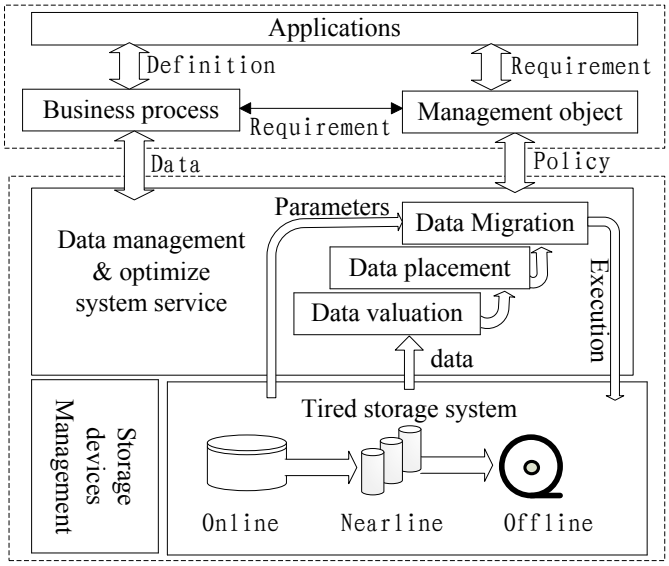

Fig. 1. Infrastructure of Tiered Storage System

performance, which includes single-level migration and multi-level migration. The main goal of single-level migration is to balance workload, however multi-level is more common in tiered storage system in order to manage the system in the form of ILM by reducing the TCO of the system with QoS guaranteed as while.

It's clearly that data migration policy affects performance, flexibility and other properties of the whole system. Today, much researches on migration focus on load balance of distributed system or cluster system [17], and because of the obvious difference on performance, capacity and other aspects among storage devices, data migration is more complex in tiered storage system. However traditional migration policies took little consideration about robustness. Moreover, the trigger of migration today is heuristic and coarse-grained, and almost all of them pay too much attention to one way migration, to migrate from high performance storage to low performance storage, without taking enough consideration of down to up migration. Based on our study of traditional policies for automated tiered storage, this paper proposes a novel bi-directional migration policy FDTM(Feedback based Double Threshold data Migration) at block level. It is fully self-adaptive, deals with the down to up migration carefully as well as up to down migration, and could avoid unnecessary data migration under "false overload" situation by setting of double thresholds and spill over interval.

2 Data Placement by Block-Level Data Valuation

Data classification is a precondition for data migration, and migration policy is usually based on data classification and placement. In addition data valuation is the core of data classification. In this paper, FDTM considers the specification of storage system and the characteristics of data objects, to get data objects value based on block-level evaluation, which is used to evaluate the importance of the data in a tiered storage system.

Obviously, it is not hard to achieve couple of metrics related to data value at file-level, because metadata contains all the necessary information. However, in storage subsystem at data center, most data are stored and managed at block-level, within one data block there maybe couple of files, so value metric is hard to achieve. But data blocks still have some attributes which are helpful for block-level data valuation:

1. READ FREQUENCY. Read times in given interval to a certain data block.
2. WRITE FREQUENCY. Write times in given interval to a certain data block. Usually, it has different performance in write and read within same storage device, and another fact is that two storage devices maybe have nearly same read performance but have huge difference in write performance. R_i and W_i represent the read/write frequency at the ith time-step.
3. R/W GRANULARITY. The granularity represents the data amount ratio which is related to an IO (whether read or write) to a fixed size data block. P_i is the average R/W granularity in the ith time-step.
4. DATA DISTRIBUTION. It is the location information of accessed data, measured by statistical result which comes from all the operations in data blocks in a given interval. D_i means the data placement value at the ith time-step.
5. RELEVANCE BETWEEN BLOCKS. Similar to relevance between files [5], it means if the IO operations on a data block is similar to the IO operations on another block in a given time interval, in other words these two blocks are associated, then the data value of these blocks is also considered as being correlated. If there are M blocks relate to block d within the same storage device, $\sum_{j=1}^{M} RL_{dj}$ represents block relevance of d and other M blocks. And the association coefficient RL_{dj} which is indicated association degree between data block d and j is defined as equation 1:

$$RL_{dj} = \frac{E_d \cdot E_j}{|E_d||E_j|} \tag{1}$$

in which, $E_d \cdot E_j = \sum_{i=1}^{N} q_{di} q_{ji}, \quad |E_d| = \sqrt{\sum_{i=1}^{N} q_{di}^2}$

here, T means the valuation time interval, and it is divided into N time-steps equally, each time-step's length is $t_i - t_{i-1} = L$. If t indicates the current time, then T is the time period of $[t - N \times L, t]$. $E_d(q_{d1}, q_{d2}, \ldots q_{dN})$ is an IO vector during T of data block d (E_j is IO vector of data block j as well), it is used to

record R/W times information of d during T, and component q_{di} is R/W times in the ith ($i = 1, 2 \ldots N$) time-step.

as following, the data blocks valuation could be calculated by equation 2:

$$V_t(d) = \frac{1 + \sum_{j=1}^{M} RL_{dj}}{N} \sum_{i=1}^{N} (\lambda_w W_i + \lambda_r R_i) P_i D_i \tag{2}$$

$V_t(d)$ is the value of data block d at time t. In which R_i and W_i represent the read/write frequency at the ith time-step $[t - i \times L, t - (i-1) \times L]$ in the valuation interval $[t - N \times L, t]$ respectively; P_i is the average R/W size at the ith time-step; D_i is the data placement value at the ith time-step. If there are M data blocks relate to block d within the same storage device, then sum all M factors as one of valuation factor for block d. λ_r and λ_w are the compensation coefficients for read and write respectively, which are used to measure the time cost difference for the R/W operation between two kinds of storage (mainly include time cost ratio in accessing the same size of data with same times of R/W operations). For more detail about data valuation, please refer to[25].

3 Starting Conditions for Data Migration

In this section, we will discuss when migration should start. In a tiered storage system, the data migration process got to consider many preconditions in the management system for special requirements. Generally, there are four main trigger conditions need be taken into account for the applications.

1. FREE SPACE OF STORAGE DEVICES
   This is the spare capacity threshold of storage devices to keep the specific application running well.
2. DIFFERENCE OF DATA VALUES
   There are two attributes to determine the value of data. One is the characteristic of the application itself, such as requirements of security, reliability of the system, so some data are more important than others inherently sometimes, the other attribute is the data value itself which is declining as time goes by.
3. UTILIZATION RATE OF DATA
   It depends on the data class. Some class is declining when it is created; some class is increasing at first and declining after it reaches the peak utilization rate of itself, and utilization rate of another class changes periodically.
4. RULES DEFINED BY LAW
   There are some rules defined by law in data management. Such as call history of cell phone for billing would only be reserved for several months, people could forecast the trend of some data changing and so on.

Moreover, these conditions are related to each other. So in order to get a more reasonable migration policy, it is necessary to consider these conditions comprehensively. At current, there are two well-used basic polices[24]:

1. FIXED THRESHOLD WITH ACCESS FREQUENCY
   It has both up-migration and down-migration, and setting *min* and *max* thresholds for data access frequency. Migration will be triggered when reaching specific threshold.
2. HIGH-LOW WATER LEVEL OF STORAGE
   Thresholds are setting according to the capacity usage ratio. When the actual capacity usage ratio exceeds the threshold, the system will migrate some data according to their access frequencies.

In above policies there are couple of demerits: although it's an up-down migration, it will become time-consuming badly when the system has huge amount of data objects. Moreover, if a system with average low workload is changing dramatically at capacity ratio (called "false overload"), it will trigger migration operations many times and maybe introduces oscillation migrations.

4 Feedback-based Double Threshold Data Migration

Data migration policy got to address four questions that are known as when, where, what and how much, by which to determine the migration candidates, migration quantity, and migration target. Migration target is obvious in tiered storage system, either is higher performance storage or lower performance storage. So the migration candidates and the quantity are key elements. Based on the summary introduction in early paper [26], an extended discussion of this policy will be given out completely in following sections. Other 3 migration situations will be proposed, however we just discuss two-tier storage structure for better description and understanding.

4.1 Parameters Definition and Initialization

Suppose the defined capacity usage ratios are same across different storage tiers. And using letters h and l at subscript of parameters to represent high performance storage and low performance storage. If $C_T = C_h + C_l$ represents total capacity and C_h is the capacity of high storage, C_l is the capacity of low storage, then high storage's usage ratio is $H_C = C_h/C_T$, and low storage's usage ratio is $L_C = C_l/C_T$.

Actual data size at each tier has 4 thresholds: base-high, base-low thresholds, limit-high and limit-low thresholds, they will be described in detail later. Here, β represents high threshold and α represents low threshold, superscript 0 means "base" thresholds, superscripts *max* and *min* represent "limit" thresholds.

We use two queues, the up-migration queue and the down-migration queue which are represented by two vectors Q^u and Q^d respectively for each tier. The elements in queue vectors are metadata of each migration candidates. The information about data value will be collected periodically after the system startup, and the interval is same with data valuation but without any calculation of data value until the enqueue condition was triggered.

4.2 When to Migration

An ideal storage management system should keep online for applications during data migration, and the storage performance should not be impacted badly at any time. So, the migration timing depends on bandwidth, migration speed and migration time window. The D-value of high and low thresholds δ should be the function of $E(t)$ and $g(B_i)$. $E(t)$ is the function of time cost of data migration, and $g(B_i)$ represents the migration speed, B_i is the bandwidth for data migration. In other words, D-value δ is a function of expected time cost and system migration bandwidth: $\delta \rightarrow f(g(B_i), E(t))$.

The system could be regarded as stable, if high storage data size $\beta_h(t)$ at time t is not bigger than β_h^0 , in which $\beta_h^0 = \beta_T \times H_C$, $\beta_T(C_T > \beta_T)$ is the total data size of storage devices, and the limit-high threshold is

$$\beta_h^{max} = K\beta_h^0 \tag{3}$$

$$K = min(\gamma, 0.5 + 0.5C_h/\beta_h^0), \tag{4}$$

in equation 4, γ is the ratio of actual data capacity usage ratio H'_C and the original H_C when the high storage provides same QoS with low storage. And when the ratio is much larger than 2, a ratio which can reach the right mid-point of β_h^0 and C_h should be used.

As to the base-low threshold $\alpha_h^0 = \beta_h^0 - \delta$ of high performance storage, the limit-low threshold is defined as $\alpha_h^{min} = \alpha_h^0 \times L_C$. The 4 thresholds for low performance storage could be derived with similar approach with considering two-tiered storage, δ' and K' are defined as following:

$$\delta' = \delta \times C_l/C_h \tag{5}$$

$$K' = min(\gamma', 0.5 + 0.5C_l/\beta_l^0) \tag{6}$$

4.3 Migration

FDTM has up-migration and down-migration conditions for both high storage and low storage. The up-migration means data migrating from low storage to high storage, which could be passively migration triggered by the actual data size on low storage exceeds its base-high threshold β_l^0, or be proactive migration triggered by the actual data size on high storage lower than its base-low threshold α_h^0. In reverse, the down-migration is data migrating from high to low performance storage, which could be triggered by the actual data size on high storage exceeds β_h^0, or be triggered by the actual size on lower than α_l^0 in low storage.

Storage device specification determines the timing of enqueue and migration, to answer the "when" question, and the characteristics of data objects determine data valuation. If the base thresholds were triggered, then each storage tier starts to enqueue migration candidates into Q^u and Q^d based on data blocks' value, which answers the "what" and "how much" questions. Adjusting the entry queue conditions by each migration operation, can improve the queue length (or the migration quantity), enhance the migration policy efficiency, and keep the stability of the tiered storage.

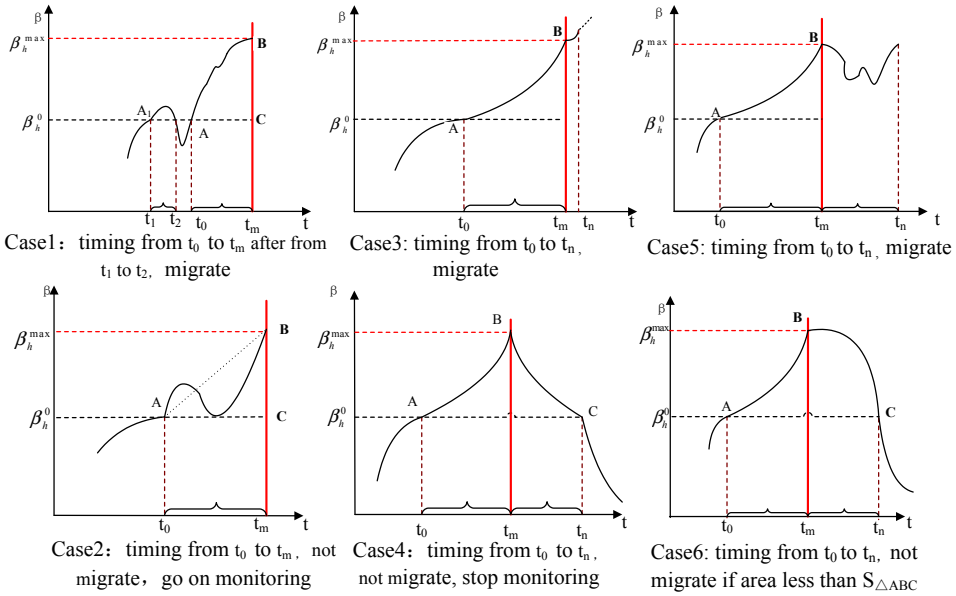

Fig. 2. Condition cases for trigger migration. The area formed by three points A,B,C and whether the special case need to be migrated according to the conditions discussed in algorithm 2

Situation 1: Down-migration for Exceeds β_h^0
There are several input parameters used in the algorithm 2
L, checkpoint interval during monitoring; t, checkpoint; $\beta_h(t)$, actual data size of high storage at time t; A_h^d, data amount in down queue of high storage; V_{last}^d, data value of the last data object in Q_h^d; V_{min}^u, data value of the min data object in up queue of low storage; C_1, represents expression $\int_{t_0}^{t_n}(\beta_h(t)-\beta_h^0)dt \geq \frac{1}{2}(t_n-t_0)(\beta_h^{max}-\beta_h^0)$; D_{mig}, data amount has been migrated till current time.

In fact, the determined conditions of FDTM are based on the fitting of a linear function to the trend of data amount in the tiered storage. When the growth trend of the curve is less than the slope of the linear function, also known as the curve integral is less than the area of the triangle ($S_{\triangle ABC}$) in the determined condition (the area composed by the slash and dash between two time points and $y=\beta_h^0$, as shown in case 4 of the Fig.2.), we regard the system as stable and doesn't need migration. Or else, the data amount growth could be considered as linear growth and the data migration should be implemented at once to reduce the data amount in high storage. It is the passive down-migration in high storage.

Situation 2: Up-migration Exceeds β_l^0

Situation 2 and situation 1 are similar, but the length of up queue is limited by δ instead of δ' which is related to low storage, for the usage stability of

Algorithm 1 ImpMigration(Q_h^d)

1: **while** $(D_{mig} < \delta) and (\beta_l(t) < \beta_l^0)$ **do**
2: migration data objects in Q_h^d
3: **end while**
4: record last data value migrated by V_{max}^d;
5: **return** V_{max}^d;

high storage is the precondition of up-migration, data amount of migration is determined by high storage's capacity.

At time t_0 and $\beta_l(t_0) = \beta_l^0$, start timing and enqueue data objects into Q_l^u according to their value descending, the length of Q_h^d is determined by the amount of data objects in it, it should be less than δ or the value of last data object in Q_h^d less than V_{max}^d. Continue to monitor the system at interval L, and do steps similar to situation 1, the differences are as following:

All the conditions which related to the high storage in situation 1 (such as β_h^0, β_h^{max}, $\beta_h(t)$) will be replaced by the corresponding ones of the low storage (β_l^0, β_l^{max}, $\beta_l(t)$) here, and vice versa;

The inequation used to judge whether the tiered system needs to migrate is changed at the right side: give a value of L_C as an coefficient when calculate the area of triangle, because with the same data changing rate, the performance of low storage is lower;

Furthermore, the min data value in up-migration should be recorded as V_{min}^u, which is used for a new migration cycle after migration.

Situation 3: Up-migration Below α_h^0

Moreover, to make a complete migration policy to achieve higher storage efficiency it should implement migration when the actual data size of high storage or low storage is less than base-low threshold, in situation 3 and situation 4, we'll discuss it in detail.

At time t_0, when high storage capacity usage size declines to less than α_h^0, start timing and enqueue block-level data objects into the queue Q_l^u in low storage, data amount in Q_l^u should not exceed δ or include all the active data objects in low storage, and do as following steps:

1. If it raises above α_h^0 again before reaching the limit-low threshold α_h^{min}, goto step 3, else continue timing until the actual data size drops to α_h^{min}, then record the current timing as t_m, and goto step 2;
2. Implement up-migration until all the data objects in Q_l^u are migrated, and goto step 3;
3. Stop timing and enqueuing monitoring. Waiting for next condition trigger $\alpha_h(t) = \alpha_h^0$, then goto step 1 for next cycle.

Situation 4: Down-migration Below α_l^0

Algorithm 2 Down Migration for exceeds β_h^0

while 1 **do**
  keep monitoring the tiered storage system;
  if $\beta_h(t) = \beta_h^0$ **then**
    $t_0 \Leftarrow t$; start to enqueue Q_h^d by block-level data value, length limited by $A_h^d < \delta$ or $V_{last}^d < V_{min}^u$
  end if
  while $\beta_h^0 \leq \beta_h(t)$ **do**
    if $\beta_h(t) = \beta_h^{max}$ **then**
      $t_m \Leftarrow t$;
      if $\int_{t_0}^{t_m}(\beta_h(t) - \beta_h^0)dt \geq \frac{1}{2}(t_m - t_0)(\beta_h^{max} - \beta_h^0)$ **then**
        ImpMigration(Q_h^d);
        break;
      else
        $t = t + L$;
        if $\beta_h(t) > \beta_h^{max}$ **then**
          ImpMigration(Q_h^d);
          break;
        else
          for t; $t - t_m \leq t_m - t_0$; $t + L$ **do**
            $t_n \Leftarrow t$
            if $(\beta_h(t) \geq \beta_h^{max}) or ((t_n - t_m = t_m - t_0 or \beta_h(t) \leq \beta_h^0) and (\int_{t_0}^{t_n}(\beta_h(t) - \beta_h^0)dt \geq \frac{1}{2}(t_n - t_0)(\beta_h^{max} - \beta_h^0)))$ **then**
              ImpMigration(Q_h^d);
              break;
            end if
          end for
        end if
      end if
    end if
  end while
  stop timing and enqueuing;
end while

At time t_0, the data amount in low storage declines to the base-low threshold α_l^0 and at the same time the data in high storage exceeds base-high threshold β_h^0, then start timing and enqueuing block-level data objects into the queue Q_h^d in high storage, data amount in Q_h^d should not exceed $(\beta_h(t) - \beta_h^0) \cdot L_C$ or δ' (take the smaller one from them), and do as following steps:

1. If it raises above α_l^0 again before reaches the limit-low threshold α_l^{min}, goto step 3, else continue timing until the data amount drops to α_l^{min}, then record the current timing as t_m, and goto step 2;
2. Implement down-migration until all the data objects in queue Q_h^d are finished migrating or the actual data size of low storage reaches $\beta_T \cdot L_C$, and goto step 3;

3. Stop timing and enqueuing monitoring. Waiting for next condition triggers $\alpha_l(t) = \alpha_l^0$, then goto step 1 for next cycle.

It's obviously that situation 1 and 2 would happen in higher probability than situation 3 and 4, if there are a reasonable data placement method in the tiered storage system which is out of discussion in this paper.

4.4 Discussion on Better Migration Effect

FDTM introduces feedback mechanism to improve its thresholds parameters for the parameters used at beginning maybe not very appropriate. So the parameters of migration policy could be more reasonable and the application could get more performance benefit with time going by.

For example, the value of V_{max}^d which indicates the length of down-migration queue Q_h^d of high storage, could be adjusted by its history values. Here, a simple method is introduced: F_i, $(i = 1 \ldots n$, n is the migration times) represents the last block-level data object's value in the ith migration queue Q_h^d, after the first migration $V_{max}^d = F_1$, after the second migration, V_{max}^d should be $\frac{F_1+F_2}{2} \ldots$ and after the nth migration, the adjusting is $V_{max}^d = \frac{\sum_{i=1}^{n} F_i}{n}$, which is the average value of F_i in n times of migrations. Certainly, with considering the different importance of its history value, we can get some different adjusting method, such as equation 7

$$V_{max}^d(n+1) = \sum_{i=1}^{n} \theta_i V_{max}^d(i), \qquad (\sum_{i=1}^{n} \theta_i = 1) \tag{7}$$

here, θ represents the importance of history values of V_{max}^d.

At the same time, if V_{min}^u is up-migration enqueue condition, after the nth time of migration, it should be adjusted by all values of the last data objects in the queues at each migration. In order to make FDTM more robust and efficient, the other parameters or thresholds could be adjusted according to real migration monitoring and feedback, such as $V_{min}^u, \alpha, \beta, \delta$ and etc.

4.5 Experiment

In this section, we validate and analyze the FDTM policy with DiskSim simulator [3] for we dont have a real tiered storage environment. In addition we choose Seagate Cheetah 15.5K FC disk and SSD disk to compose a two tiered storage system within DiskSim. The Seagate Cheetah 15.5K FC disk specification is extracted by DIXtrac disk characterization tool, and in other side the SSD DiskSim extension by Microsoft research is an idealized SSD that is parameterized by the properties of NAND flash chips instead of a simulator for any specific SSD.

We show the device capabilities difference by dollar cost in Fig. 3 according to the latest research on the tradeoffs between FC disk and SSD disk [13] for better understanding of the storage devices in the experiment environment. It's

Table 1. Storage Device Characters

Tier 1 storage	SSD Disk	SSD extension for DiskSim by Microsoft research
	Size	32 GB
	Block Size	256 KB
	Block Number	131072
	The SSD is composed by 8 elements, each el ement has 8 planes, and each plane has 2048 blocks	
Tier 2 storage	FC Disk	Seagate Cheetach 15.5K FC Disk
	Size	146.8 GB
	Block Size	512 B
	Block Number	28679487

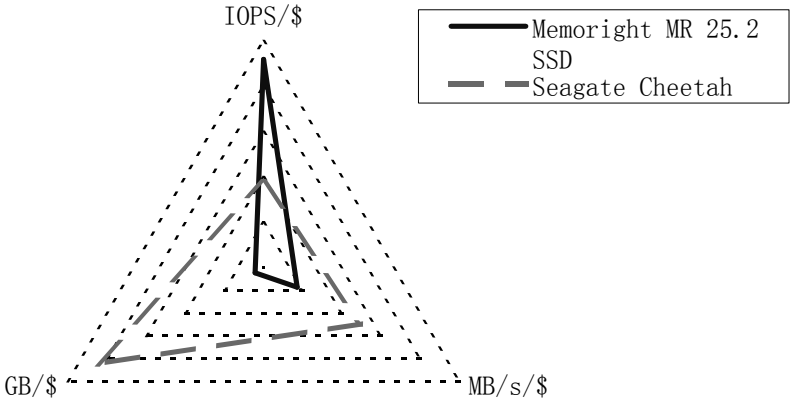

Fig. 3. device capabilities normalized by dollar cost

obviously that SSD disk could provide better IO performance, which could be around 100 times [13], than FC disks under same workload but cost more dollars. Obviously in enterprise data center it's very worthy to adopt tiered storage infrastructure by FC disks and SSD disks, to provide improved IO performance with reasonable storage cost. Tab. 1 is the basic configuration for this simulated environment.

The workload IO trace for experiment is randomly generated by using DiskSim at start, moreover, in order to play the same workload trace on four kind of storage systems (the SSD disk, the FC disk and the tiered storage system) respectively, so the block access range in this trace is limited by the size of the SSD disk, which is the smallest in capacity. Besides, we have two assumptions at here:

Assumption 1: We assume that the initial data placement has little performance impact on experiment results, so we randomly place the data block objects across the tiered storage system to avoid the initial data placement issue.

Assumption 2: We assume that the overhead of data block objects migration between SSD and FC disks could be ignored for the experiment, for it's hard

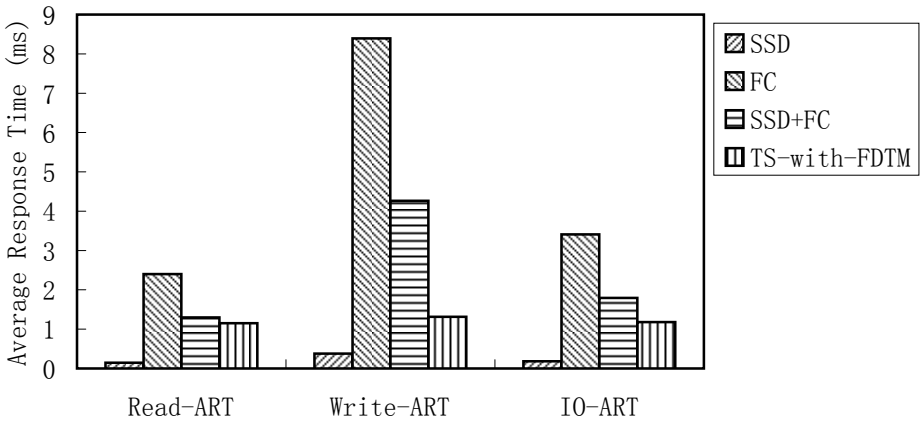

Fig. 4. Average response time of 4 kinds of storage system

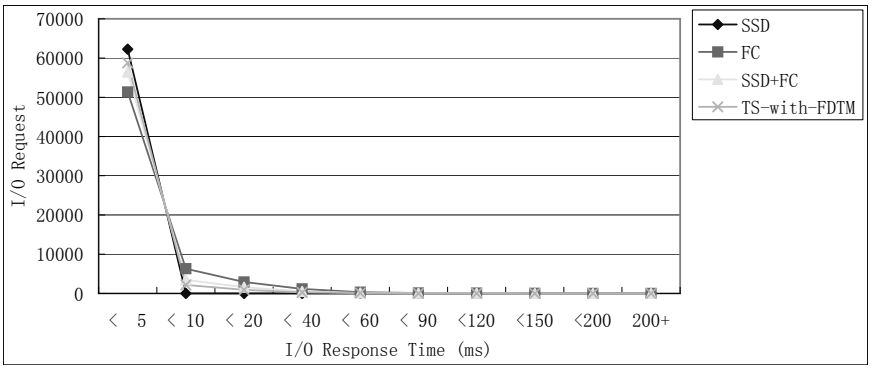

Fig. 5. Overall IO system response time distribution of the experiment

to simulate the overhead of data migration in this simulated environment and we'll pay more attention on policy itself. So we only change the IO request target devices between SSD and FC disks in the workload trace to simulate the environment with data migration, in fact the data migration doesn't happen in the experiment.

In experiment the workload trace is played 4 rounds on the 4 kinds of simulated storage devices, which are SSD disk, FC disk, tiered storage composed by SSD and FC disks, and tiered storage composed by SSD and FC disks with the FDTM policy control. The experiment results are collected from the DiskSim outputs. Fig. 4 shows the average response time results of the experiment. We also give out overall IO system response time distribution in Fig. 5

It could be concluded from the experiment results that the tiered storage system with FDTM policy could provide better IO performance and it's easy to be implemented in current storage system. However, we just valuated FDTM

with a simulated environment and synthetic workload trace, in which we take little consideration on the overhead of FDTM, so this experiment is just used to validate FDTM policy and we aim to valuate it in real environment with ongoing work.

5 Related Work

As one of the most important concepts, DMF(Data Management Forum) in SNIA promotes ILM as a special topic in the field of data management system research and application. In [14, 20], an deep analysis and discussion was done to the relationship between ILM and tiered storage system.

There are several systems providing tiered storage system management functions transparently, such as SGI InfiniteStorage Data Migration Facility [19], VERITAS' NetBackup Storage Migrator [1] and IBM's HPSS[23]. Additionally, early research mainly focuses on the management and employment of tertiary storage, an offline storage tier in the tiered storage system, such as the applications in VoD(Video on Demand) system and DBMS[12, 15], researches [6, 16] mainly focus on the problem of performance in tertiary storage. With the development of tiered storage applications, studies on data classification and data placement are more popular, in [21] the necessity and importance in data classification and its basis are discussed, and [4, 18] give out some specific methods for classification. References [8, 9] focused on data placement under different storage infrastructure.

There are also a lot of researches on data migration, such as to implement online migration by leveraging control theory [10] to reduce the overhead mostly, to describe the migration algorithm with edge-coloring theory [2, 7]. All of above researches aim to achieve workload balance under a homogeneous storage environment, and suppose that data objects are independent with each other. However, it is almost impossible that happens in real environment. [22] researches on block-level data migration, but it just takes data access frequency into consideration, which affects its accuracy of candidate migration objects.

6 Conclusion and Future Work

This paper proposes the novel bi-directional block-level data migration policy FDTM for tiered storage system, the migration with double thresholds based on feedback, which could help improve the IO performance of tiered storage system.

In addition, we also give out the detail process of block-level data migration, and how to select the migrated candidates, in which the data valuation mechanism could narrow the candidates' scope and avoid the oscillation migrations by leveraging the relevance between different data blocks. The result of experiment shows that the tiered storage system with FDTM policy could provide better IO performance, and it's easy to be implemented in current storage systems. As ongoing and future work, we'll focus on the study of data placement, the interaction between data placement and data migration in tiered storage system.

Acknowledgments This work was supported by a grant from the National High Technology Research and Development Program of China (863 Program) (No. 2009AA01A404).

References

1. What is storage virtualization. Veritus white paper, http://file.doit.com.cn/upfiles/2006/1027/0_230404_f1.pdf
2. Anderson, E., Hall, J., Hartline, J.D., Hobbs, M., Karlin, A.R., et al., J.S.: An experimental study of data migration algorithms. In: Proceedings of the 5th International Workshop on Algorithm Engineering. pp. 145–158. Springer-Verlag (2001)
3. Bucy, J.S., Schindler, J., Schlosser, S.W., Ganger, G.R.: The disksim simulation environment version 4.0 reference manual. Technical report cmu-pdl-08-101, carnegie mellon university (2008)
4. Golubchik, L., Khanna, S., Khuller, S., Thurimella, R., Zhu, A.: Approximation algorithms for data placement on parallel disks (2000)
5. Jin, H., Xiong, M., Wu, S.: Information value evaluation model for ilm. In: Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and parallel/Distributed Computing. pp. 543–548. IEEE Computer Society (2008)
6. Johnson, T., Miller, E.L.: Performance measurements of tertiary storage devices. In: Proceedings of the 24rd International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc. (1998)
7. Khuller, S., Kim, Y.A., Wan, Y.C.: Algorithms for data migration with cloning. In: Proceedings of the Twenty-second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems. pp. 27–36. ACM, San Diego, California (2003)
8. LI, C., ZHOU, L.Z., XING, C.X.: A cost model for data placement and access path selection problem in fc-san. Journal of Software (05) (2004)
9. Li, J.T., Prabhakar, S.: Data placement for tertiary storage. In: Proceeding of the 10th NASA Goddard Conference on Mass Storage Systems and Technologies/19th IEEE Symposium on Mass Storage Systems (MSS 2002). pp. 193–207. Adelphi, Maryland, USA (2002)
10. Lu, C., Alvarez, G.A., Wilkes, J.: Aqueduct: Online data migration with performance guarantees. In: Proceeding of the USENIX Conference on File and Storage Technologies (FAST). pp. 219–230. Monterey (2002)
11. Massiglia, P.: Exploiting multi-tier file storage effectively. Snia tutorial, http://www.snia.org/education/tutorials/2009/spring/file/paulmassiglia_exploiting_multi-tier_file_storagev05.pdf, SNIA (2009)
12. Myllymaki, J., Livny, M.: Disk-tape joins: Synchronizing disk and tape access. In: Proceedings of the 1995 ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems. pp. 279–290. ACM, Ottawa, Ontario, Canada (1995)
13. Narayanan, D., Thereska, E., Donnelly, A., Elnikety, S., Rowstron, A.: Migrating server storage to ssds: analysis of tradeoffs. In: EuroSys '09: Proceedings of the 4th ACM European conference on Computer systems. pp. 145–158. ACM, New York, USA (2009)
14. Peterson, M.: Ilm and tiered storage. Snia tutorial, http://www.snia.org/forums/dmf/knowledge/dmf-ilm_and_tiered_storage_20060221.pdf (January, 2006)

15. Prabhakar, S., Agrawal, D., Abbadi, A.E., Singh, A.: A brief survey of tertiary storage systems and research. In: Proceedings of the 1997 ACM symposium on Applied computing. pp. 155–157. ACM, San Jose, California, United States (1997)
16. Reiner, B., Hahn, K.: Optimized management of large-scale data sets stored onttertiary storage systems. Distributed Systems Online, IEEE 5(5) (2004)
17. Seo, B., Zimmermann, R.: Efficient disk replacement and data migration algorithms for large disk subsystems. ACM Transactions on Storage 1(3), 316–345 (2005)
18. Shah, G., Voruganti, K., Shivam, P., del Mar Alvarez Rohena, M.: Ace: Classification for information lifecycle management. NASA Mass Storage Systems and Technologies (2006)
19. Shepard, L.: Sgi infinitestorage data migration facility(dmf) a new frontier in date lifecycle management. White paper, sgi. http://www.sgi.com/pdfs/3631.pdf
20. SNIA: Ilm definition and scope an ilm framework. http://www.snia.org/forums/dmf/programs/ilmi/dmf-ilm-vision2.4.pdf (July 2004)
21. SUN: Best practices in data classification for information lifecycle management. Sun white paper, http://www.sun.com/storage/white-papers/best_practices_data_classification_ilm.pdf
22. Wang, D., Shu, J.W., Xue, W., Shen, M.M.: Self-adaptive hierachical storage management in san based on block-level. Chinese High Technology Letters (02) (2007)
23. Watson, R.W.: High performance storage system scalability: Architecture, implementation and experience. In: Proceedings of the 22nd IEEE / 13th NASA Goddard Conference on Mass Storage Systems and Technologies. pp. 145 – 159. IEEE Computer Society (2005)
24. Xu, N.: A frequency-based self-adaptive data hierarchy policy. SCIENCE and TECHNOLOGY ASSOCIATION FORUM (03) (2009)
25. Zhao, X.N., Li, Z.H., Zeng, L.J.: A hierarchical storage strategy based on block-level data valuation. In: the proceeding of the Fourth International Conference on Networked Computing and Advanced Information Management 2008 NCM '08. pp. 36 – 41. IEEE, Korea (2008)
26. Zhao, X., Li, Z., Zhang, X., Zeng, L.: Block level data migration in tiered storage. In: Proceeding of International Conference on Computer and Network Technology (ICCNT 2010). IEEE, Bangkok, Thailand (2010)