Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment

Gong Zhang*, Lawrence Chiu†, Ling Liu*
*College of Computing
Georgia Institute of Technology, Atlanta, Georgia 30332
† IBM Almaden Research Center, San Jose, CA 95120

Abstract—**Multi-tiered storage systems today are integrating Solid State Disks (SSD) on top of traditional rotational hard disks for performance enhancement due to the significant IO improvements in SSD technology. It is widely recognized that automated data migration between SSD and HDD plays a critical role in effective integration of SSD into multi-tiered storage systems. Furthermore, effective data migration has to take into account of application specific workload characteristics, deadlines, and IO profiles. An important and interesting challenge for automated data migration in multi-tiered storage systems is how to fully release the power of data migration while guaranteeing the migration deadline is critical to maximizing the performance of SSD-enabled multi-tiered storage system. In this paper, we present an adatpive lookahead data migration model that can incorporate application specific characteristics and I/O profiles as well as workload deadlines. Our adaptive data migration model has three unique features. First, it incorporates a set of key factors that may impact on the performance of lookahead migration efficiency in our formal model develop. Second, our data migration model can adaptively determine the optimal lookahead window size, based on several parameters, to optimize the effectiveness of lookahead migration. Third, we formally and experimentally show that the adaptive data migration model can improve overall system performance and resource utilization while meeting workload deadlines. Through our trace driven experimental study, we compare the adaptive lookahead migration approach with the basic migration model and show that the adaptive migration model is effective and efficient for continuously improving and tuning of the performance and scalability of multi-tier storage systems.**

I. Introduction

The rise of solid-state drives (SSDs) in enterprise data storage arrays in recent years has put higher demand on automated management of multi-tiered data storage software to better take advantage of expensive SSD capacity. It is widely recognized that SSDs are a natural fit for enterprise storage environments, as their performance benefits are best leveraged for enterprise applications that require high inputs/outputs per second (IOPS), such as transaction processing, batch processing, query or decision support analysis. A number of storage systems supporting Flash devices (SSDs and memory) have appeared in the marketplace such as NetApp FAS3100 system [1], IBM DS8000[2]. Most of these products fall into two categories in terms of the ways of integrating Flash devices into the storage system. The first category involves the products that have taken the approach of utilizing Flash-memory based caches to accelerate storage system performance [1]. The main reason that is in favor of using Flash devices as cache includes the simplicity of integration into existing systems without having to explicitly consider data placement and the performance improvement by increasing cache size at lower cost (compared to DRAM) [3]. The second category includes those vendors that have chosen to integrate SSDs into tiered storage architectures as fast persistent storage [2]. On The arguments for using and integrating flash devices in an SSD form factor as persistent storage in a tiered system include issues such as data integrity, accelerated wear-out and asymmetric write performance. Given the limited capacity of SSD tier in the multi-tiered storage systems, it is critical to place the most IOPS-intensive and latency sensitive data on the fastest SSD tier, in order to maximize the benefits achieved by utilizing SSD in the architecture. However, a key challenge in addressing two-way data migration between SSDs and HDDs arises from the observation that hot-spots in the stored data continue to move over time. In particular, previously cold (i.e., infrequently accessed) data may suddenly or periodically become hot (i.e., frequently accessed, performance critical) and vice versa. Another important challenge for automated data migration is the capability to control and confine migration overheads to be within the acceptable performance tolerance of high priority routine transactions and data operations.

By analyzing the real applications in environments such as banking settings and retail stores with a comercial enterprise multi-tiered storage server, one can discover certain temporal and spatial regularity in terms of access patterns and temperature of extents. For example, in a routine banking environment, the major workload during the daytime centers around transaction processing and thus certain indices becomes hot in the daytime period, while at the night time, the workload switches into report generation type and correspondingly different indices become hot. Generally speaking, such patterns may not only change from day-time to night-time and it is also likely that the pattern changes from day to day, from weekdays to weekends or from working period to vacation period. This motivates us to utilize IO profile as a powerful tool to guide the data migration. Further, in order to improve system resource utilization, the data migration techniques devised for multi-tiered storage architecture needs to be both effective and non-intrusive. By effective, we mean that the migration activity must select an appropriate subset of data for migration, which can maximize overall system performance while guaranteeing completion of the migration before the onset of the new workload (i.e., within a migration deadline). By non-intrusive, we mean that the migration activity must minimize its impact on concurrently executing storage workloads running over the multi-tier storage system.

In this paper, we present an adaptive deadline aware lookahead data migration scheme, called ADLAM, which aims to adaptively migrate data between different storage tiers in order to fully release the power of the fast and capacity-limited SSD tier to serve hot data and thus improves system performance and resource utilization and meanwhile limiting the impact of ADLAM on existing active workload. Concretely, we want to capitalize on the expensive SSD tier primarily for hot data extents based on IO profiling and lookahead migration. IO profiling enables the shifts in hot spots at the extent level to be predictable and exploitable, and based on this lookahead migration practively migrates those extents, whose heat is expected to rise in the next workload, into SSD tier during a lookahead period. To optimally set the lookahead migration window, multiple dimensions that impacts the optimal lookahead length needs to be exploited. By optimal we mean that this lookahead migration should effectively prepare the storage system for the next workload by reducing the time taken to achieve the peak performance of the next workload, and maximizing the utilization of SSDs. We deployed the a prototype ADLAM in an operational enterprise storage system and the results shows that lookahead data migration effectively improves system response with reduced average cost by fully utilizing the scarce resource such as SSDs.

The main contributions of this paper are two folds. First, we describe the needs and impacts of the basic deadline aware data migration on improving system performance from real storage practice. We build a formal model to analzye the benefits of basic data migration across different phase on system response time improvements and integrates the benefits in each phases into the benefits across all the phases.

Second, we present our data migration optimization process which evolves from learning phase reduction, to constant lookahead data migration and to adaptive lookahead data migraiton scheme. The system utility meansure is built to compare the performance gains in each data migration model. Adaptive lookahead migration apparch, which works as an effective solution for performing deadline aware data migration by carefully trading off the peformance gains achieved by lookahead migration on the next workload and the potential impacts on existing workloads. This approach centers around a formal model which computes the optimal lookahead length by considering a number of important factors, such as block level IO bandwidth, the size of SSD tier, the workload characteristics, and IO profiles. Our experiments confirms the effectiveness of the proposed adaptive data migration scheme by testing the IO traces collected from benchmark and commercial applications running on an enterprise multi-tiered storage server. The experiments shows that ADLAM not only improves the overall storage performance, but also outperforms the basic data migration model and constant lookahead migration strategies significantly in terms of system response time improvements.

II. ADLAM Overview

A typical SSD-based multi-tier storage system architecture consists of three layers. The bottom layer contains many physical disks classified into different tiers according to the IO access performance characteristics, (for example as the figure shows SSD provides the high speed IO access as tier 0, and SATA disks and SCSI disks work as tier 1 to provide mass storage), and the storage controller in the middle layer is response for the functionalities such as generating data placement advise, planning data migration and providing device virtualization. As the top layer, virtual LUN presents hosts a logical view of the storage system virtualy comprised of many LUNs and each LUN is further divided into many logical extents. The storage controller manages the mapping between logical extents and the physical extents in the physical layer. The two way data migration between tiers presents different IO features. Data migration from tier 1 to tier 0 achieves higher access speed in the price of higher per GB access cost, while the reverse data migration decreases IO speed and IOPS correspondingly. Two way data migration among different tiers provides us the flexibility to schedule the resource to meet different access speed demand and cost on the data extent level.

A. Motivation Scenario and Assumptions

In this paper, we focus on the working environments like banking scenarios or retail stores where IO workloads typically alternate between periods of high activities and low activities and also the workload characteristics may change from daytime to night time activities. For example, in a typical banking environment, the workload during the daytime (business hours) primarily focuses on the transaction processing tasks which create intense accesses to indices and some small set of data out of the large data collection, such as access permission logs. However, during the night time, the workload type is often switched into report generating style and the high access frequency data is also changed into different indices or different logs. Every 12 hours, one workload finishes its cycle and the next workload starts its cycle. Every 24 hours, a complete cyclic period of workload pattern (with respect to two workloads in this example) finishes and the next period starts. Apparently, data segments like indices and access logs attract significantly dense IO accesses and are hot. Improving the access speed for these “hot data” is highly critical to the performance enhancement of the entire storage system. Through our experiences working with a number of enterprise class storage system applications in banking and retail store industries, we observe that many applications have similar IO workloads as those in banking or retail stores and exhibit similar time-dependent patterns in terms of random IO accesses and batch IO reads.

We run the Industry standard IO benchmark SPC1 and TPCE for a duration of time on a proprietary enterprise storage system respectively. Certain stability for a given workload

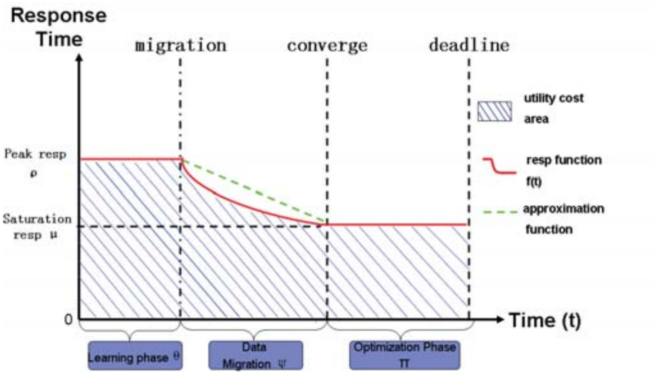

Fig. 1. Migration Computation Model

presented by hot data during its running cycle is confirmed. The experimental results reveals two interesting observations. First the hot extents bands for the two different workloads are totally different. Second, stability holds on both hot data and cold data with given workload active cycle for both SPC1 and TPCE workloads. Readers may refer to our technical report for further details on this analysis.

The high disparity of access speed between traditional HDDs and SSDs and the highly cyclic behaviors of different IO workloads motivate us to migrate the hot data like indices into SSD for each workload type and thus significantly improve the access speed for hot data, which is a known bottleneck that dominates the storage system performance. Another important requirement for devising data migration schedule is the fact that the workload pattern tends to alternate, for example between daytime and nighttime, and thus different workload cycles generate different hot data and hence it is compulsory to finish the desired data migration before the deadline of the corresponding workload to which the hot data is associated. To address the data migration problem that is constrained by the workload deadline, we identify the following assumptions for the adaptive deadline aware data migration problem:

1) The IO workloads exhibit cyclic behaviors and the set of workload types alternate between one another within a given period, such as every 24 hours.
2) The data access patterns, especially the hot data extents, can be learned by continuously monitoring the data heat.
3) Fast storage resource like SSD exhibits scarcity in terms of per unit cost.
4) Data migration for each aworkload is bounded by certain deadline.

Bearing with these constraints, we develop an adaptive deadline-aware lookahead data migration framework, – AD-LAM, aiming at optimizing system resource utilization by adaptively migrating data between faster storage tier and slow storage tier triggered by workload alternation while guaranteeing the deadline requirements.

B. Storage Utility Cost

In multi-tier storage system, response time (for abbreviation,we refer response time as "resp") is the major performance goal whose optimization is concerned the most by the users. Thus, the principal performance optimization objective using data migration is to reduce the response time as the storage system runs. We first define system response time function as follows:

Definition 1. A function recording the response time during a time period from $t1$ to $t2$ for the multi-tier storage system is called *"response time function"* and is denoted as $f(t), t \in [t1, t2]$.

Definition 2. The system utility cost, denoted by U, is the summation of the response time of the multi-tier storage system at each time t over the entire time period of $[t1, t2]$ and $t \in [t1, t2]$. That is, $U_{t_1,t_2}(t) = \int_{t1}^{t2} f(t)dt$.

Our goal is to minimize the system utility cost while guaranteeing the migration deadline. If migration is not employed, then the system response time is maintained at its average peak response time, denoted by ρ.

Definition 3. Let ϕ denote a workload duration (cycle) and U_0 denote the utility cost of the system when data migration is not equipped and $U_0 = \rho \times \phi$. We call the utility improvement from utility cost under a system without data migration to improved utility cost with a system powered by data migration the *utility rise* denoted by ΔU_0, and we have $\Delta U_0 = U_0 - U$.

C. Basic Migration Model

Basically, the migration process can be divided into three phases based on its impact on the overall response time of the system.

(1) Workload learning phase: The primary task in this phase is continuously monitoring the IO profile and sorting the extent list on the order of heat. To ensure data collection process shed ignorible impacts on the ongoing workload, very limited resource is allocated by this process. As shown in Figure 1, if the learning phase takes θ time, then the utility cost of this phase is $U = \int_0^{\theta} f(t)dt$, which can be approximated by $\rho \times \theta$.

(2) Data migration: In this phase, hot data extents are migrated from slow HDD tier to fast SSD tier.

In this phase, the extents are migrated on the heat order of data extents. Because there are limited number of hot expents which plays significant impacts on the response time, thus, the migration impacts on the response time observe the law of "diminishing marginal returns". That is, as the heat of the migrated extent decreases, each additional data extent migrated yields smaller and smaller reduction in response time and finally at some time point, the migration impact is saturated and no reduction in response time can be achieved by further migration. We call this time point the "convergence point". As shown in Figure 1, if the data migration phase takes ψ time, then the utility cost of this phase is $\int_{\theta}^{\theta+\psi} f(t)dt$.

(3) Optimization Phase:

After "convergence point" arrives, system enters into the "optimization phase", and the response time of the ongoing workload maintains at the stable minimum level until the active workload reaches its deadline and finishes its running cyle. As shown in Figure 1, if the optimization phase takes π time, then the utility cost of optimization phase is $\int_{\theta+\psi}^{\theta+\psi+\pi} f(t)dt$. Figure 1 illustrates the three phases that each workload experiences.

D. Utility Cost of Single Data Migration

As illustrated in the basic data migration model, individual workload experiences three phases in one cycle and for simplicity, we call such basic data migration process for an individual workload as *single migration*. The utility cost incurred in this process forms a basic unit in computing the overall system utility cost. In this section, we analyze the utility cost for single migration and further build the model to computing the utility cost.

For an individual workload, the workload cycle time ϕ equals to the sum of the time length of three phases, i.e., $\phi = \theta+\psi+\pi$. Thus, the aggregation of the utility cost in each phase, corresponding to the solid line shaded area in Figure 1, forms the utility cost of the workload over one cycle, as shown in the below lemma.

Lemma 1. The utility cost of an individual workload over one cycle ϕ is:

$$\begin{aligned} U &= \int_0^{\phi} f(t)dt \\ &= \int_0^{\theta} f(t)dt + \int_{\theta}^{\theta+\psi} f(t)dt + \int_{\theta+\psi}^{\theta+\psi+\pi} f(t)dt. \quad (1) \end{aligned}$$

Data migration time refers to the time duration length, χ, taken to migrate the data to SSD until it is full. For SSD tier with capacity C and bandwidth B allocated for data migration, then the total time duration length that is needed to fill in the SSD tier is $\chi = \frac{C}{B}$. We define "saturation time" λ as the time duration from the beginning of data migration to the time point when the impacts of migration on response time reduction is saturated or convergence point (see section II-C) is reached. Depending on inherent SSD properties and the allocated migration bandwidth, saturation time may be longer or shorter than data migration time and it depends on specific applications and SSDs. For simplicity, data migration time is used as an reasonable approximation for saturation time, i.e., $\lambda = \chi$.

With the peak response time ρ which is achived when data migration is not employed and the saturation response time μ, the response time function $f(t)$ in data migration phase is approximated with a linear function $f(t) = \frac{\mu-\rho}{\theta} \times t + \rho$, $t \in [\theta, \theta+\psi]$,which is shown as green dotted line above response time funtion curve in Figure 1. Thus, the response time function for single migration is:

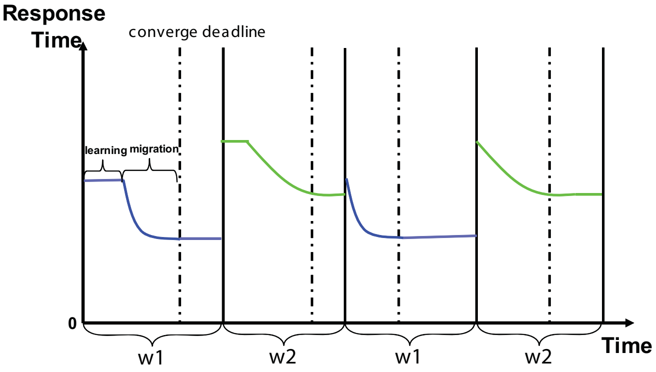

Fig. 2. Workload cycles with learning phase reduced

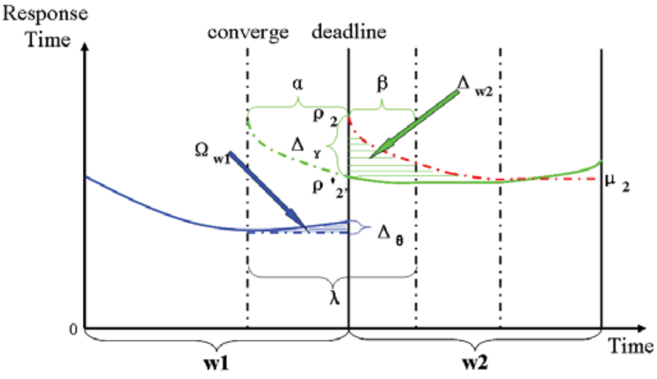

Fig. 3. Lookahead migration

$$f(t) = \begin{cases} \rho & t \in [0,\theta] \\ \frac{\mu-\rho}{\psi} \times t + \rho & t \in [\theta, \theta+\psi] \\ \mu & t \in [\theta+\psi, \phi] \end{cases} \quad (2)$$

From equation 1 and equation 2, the utility cost for a single migration process is derived as follows:

Theorem 1. The utility cost of a single migration is approximated as:

$$U = \int_0^{\phi} f(t)dt = (\rho-\mu) \times (\frac{1}{2} \times \psi + \theta) + \mu \times \phi \quad (3)$$

And the utility rise over the baseline case in which no migration is employed is:

$$\begin{aligned} \Delta U_0 &= \rho \times \phi - ((\rho-\mu) \times (\frac{1}{2} \times \psi + \theta) + \mu \times \phi) \\ &= (\rho-\mu) \times (\phi - \theta - \frac{1}{2} \times \psi) \quad (4) \end{aligned}$$

E. Migration Model with Reduced Learning Phase

By utilizing the data heat stability discussed earlier, one can reuse the stable IO profile in subsequent workflow cycles and thus reduce the learning phase significantly. Concretely, one can turn on the learning phase in the beginning workload cycles. Once the data heat is stabilized, the learning phase can be reduced for the subsequent workload cycles by reusing the data heat information learned at the beginning rounds of workload cycles. The reduction of learning cycles cuts off the redundant learning cost and enables the system to enter the optimization phase earlier than the basic migration model. In Figure 2, the first run of w1 and w2 does not eliminate the learning phase and in the second run the learning phase is eliminated.

The corresponding response time function for learning phase reduced data migration process is:

$$f(t) = \begin{cases} \frac{\mu-\rho}{\psi} \times t + \rho & t \in [0, \psi] \\ \mu & t \in [\psi, \phi] \end{cases} \tag{5}$$

From equation 1 and equation 2, the utility cost for a learning phased reduced migration process is derived as follows:

Theorem 2. The utility cost of a reduced learning data migration is as:

$$U = \int_0^\phi f(t)dt = (\rho - \mu) \times \frac{1}{2} \times \psi + \mu \times \phi \tag{6}$$

And the utility rise over the baseline case in which no migration is employed is:

$$\begin{aligned} \Delta U_0 &= (\rho - \mu) \times \frac{1}{2} \times \psi + \mu \times \phi - \rho \times \phi \\ &= (\rho - \mu) \times (\phi - \frac{1}{2} \times \psi) \end{aligned} \tag{7}$$

III. Adaptive Lookahead Migration

A. Lookahead Migration

Adaptive lookahead migration is a further improvement over learning phase reduced data migration model. As shown in Figure 3. For a configurable lookahead window length α, the solid green line and the dashed green line in $w2$ workload time together show the effects of migration on response time of workload $w2$. The dashed red curve shows the "virtual" situation on how $w2$ response time evolves if no lookahead data migration is deployed. Similary, the migration effects under α lookahead on $w1$ is shown with the blue solid curve and blue dashed curve in $w1$ workload time. α indicates that data migration for $w2$ starts α units of time ahead of the time point when the $w1$ workload finishes its running cycle. The power force of lookahead migration is rooted from the the fact that after $w1$ enters into its convergence point, further migration of $w1$ extents creates ignorible benefits on system response time. Thus it is more cost-effective to start migrating the hottest extents of $w2$ into SSD tier ahead of activating time of $w2$.

In practice, because the capacity of SSD is limited, it is very likely that the SSD has already been filled with $w1$ extents when lookahead migration starts. Thus typically, lookahead migration incurs the swapping process in which relatively cold $w1$ extents are replaced by the hottest $w2$ extents. Such swapping increases the response time of ongoing workload $w1$ to some level, which is called "bending effect", as shown in Fig. 3, the solid blue response time curve of $w1$ is slightly bending upward when lookahead migration of $w2$ starts. The dashed green line in $w2$ cycle essentially depicts the implicit data migration employed for $w2$ and we can see that because of such implicit data migration is executed in advance, the peak response time of $w2$ workload experienced by the client applications by the amount of Δ_γ. The implicit undergoing lookahead migration also reduces the time length taken to reach the convergence point perceived by the clients from $(\alpha + \beta)$ in the case of no lookahead migration to β under the lookahead migration for workload $w2$.

The shaded green area, denoted by Δ_{w2} represents the resource utilization gain achieved by lookahead migration in $w2$, and the shaded blue area, denoted by Ω_{w1}, indicates the resource utilization loss experienced by $w1$. As long as the gain by Δ_{w2} is greater than the loss of Ω_{w1}, i.e., $\Delta_{w2} > \Omega_{w1}$ holds, lookahead migration with window size α can benefit the system utilizations in a cost-effective manner. Obviously, when difference of Δ_{w2} and Ω_{w1} is maximized, the corresponding looahead length releases its full power and lookahead data migration achieves its maximum effects in term of system utility optimization. The quesiton becomes how to decide the optimal lookahead length. In [4], we proposed a greedy algorithm, which can compute a near optimal lookahead length depending on the granularity used. Next, we build a formal model on the system utility optimization using lookahead migration and derive the approach to compute the optimal lookahead length precisely.

B. Computing Optimal Lookahead Migration Window Size

In this section, we design an algorithm which computes the near optimal lookahead window size by taking into account several important factors, such as the IO bandwidth, the size of SSD tier (number of SSDs), and IO workload characteristics. Let $w1$ denote the current dominating workload running in the system and $w2$ denote the next workload that will dominate the IO access after $w1$ approaches the end. We determine the optimal lookahead migration window size in the following three steps.

1) Finding the utility improvement of workload $w2$: As shown in Figure 3, similarly we can apply a linear function to approximate the response time function for workload $w2$ in the migration function and thus derive the following function to depict the response time curve.

$$f(t) = \begin{cases} \frac{\mu_2-\rho_2}{\psi_2} \times (t + \alpha) + \rho_2 & t \in [0, \beta] \\ \mu_2 & t \in [\beta, \phi_2] \end{cases} \tag{8}$$

In Figure 3, ρ_2 is the peak response time for original response time curve without employing lookahead migration for $w2$.

Theorem 3. The utility gain of $w2$ through using α lookahead migration compared with purely learning phase reduced migration is

$$\begin{aligned}\Delta_{w2} &= U_2(0) - U_2(\alpha) \\ &= (\rho_2 - \mu_2) \times \alpha - \frac{\rho_2 - \mu_2}{2 \times \psi_2} \times \alpha^2 \qquad (9)\end{aligned}$$

2) Finding the utility loss of workload $w1$: As shown in Figure 3, starting migration of $w2$ workload causes the increment of response time of workload $w1$ and the peak response time in this increment process happens at the deadline of workload $w1$, which is called *reverse peak response time* η. The reverse response time increment process can be approximated by function $f\prime(t) = \frac{\rho_1 - \mu_1}{\psi_1} \times t + \mu_1$. Let $t = \alpha$, we derive $\eta = f\prime(\alpha) = \frac{\rho_1 - \mu_1}{\psi_1} + \mu_1$. Thus, the performance loss of workload $w1$ is as follows:

Theorem 4.

$$\Omega_{w1} = \frac{\rho_1 - \mu_1}{2 \times \psi_1} \times \alpha^2 \qquad (10)$$

3) Compute the near optimal lookahead window size: The lookahead migration utility can be derived from the performance gain of $w2$ and performance loss of $w1$ using the following theorem.

Theorem 5. The utility of α lookahead migration,

$$\begin{aligned}\Gamma(\alpha) &= \Delta_{w2} - \Omega_{w1} \\ &= (\rho_2 - \mu_2)\alpha - \frac{\rho_2 - \mu_2}{2 \times \psi_2}\alpha^2 - \frac{\rho_1 - \mu_1}{2 \times \psi_1}\alpha^2 \\ &= (\rho_2 - \mu_2)\alpha - (\frac{\rho_2 - \mu_2}{2 \times \psi_2} + \frac{\rho_1 - \mu_1}{2 \times \psi_1})\alpha^2 \qquad (11)\end{aligned}$$

As our goal is to maximize $\Gamma\alpha$ and its maximum value is achieved at $\alpha = \frac{\psi_1\psi_2(\rho_2 - \mu_2)}{\phi_1(\rho_2 - \mu_2) + \phi_2(\rho_1 - \mu_1)}$ and the maximum lookahead utility is $\Gamma_{max} = \frac{3\phi_1\phi_2(\rho_2 - \mu_2)^2}{2\psi_1(\rho_2 - \mu_2) + 2\psi_2(\rho_1 - \mu_1)}$. With this approach to compute the optimal lookahead length, even the system parameter changes, such as migration bandwidth or SSD sizes changes, a new lookahead length can be adaptively recomputed and thus guarantee the adaptivity of the proposed approach.

IV. Experimental Evaluation

This section evaluates the performance of our adaptive, deadline-aware lookahead migration scheme through four sets of experiments.

To examine the benefits of lookahead migration, we first collected IO trace from real enterprise storage controller and implemented the lookahead data migration scheme in a simulator that simulates the real enterprise storage system. We developed a trace driven simulator, which consists of a storage unit with dual SMP server processor complexes as controllers, sufficiently large memory with large size cache, multiple types of disk drives (including SATA, SCSI and SSD), multiple FICON or ESCON adapters connected by a high bandwidth interconnect, and management consoles. Its design is to optimize both exceptional performance and throughput in supporting critical workloads and around the clock service.

Our simulator simulates the IO processing functionalities of the enterprise storage systems driven by the IO pressure trace collected from running benchmark workloads on the real enterprise storage system. For the experiments in this paper, the simulator simulates a multi-tiered storage hierarchy consisting of SSD drives as tier 0 and SATA drives as tier 1 and the total physical capacity is up to hundreds of terabytes in terms of hundreds of disk drives. For the experiments in this paper, we run the simulations on a server machine with 4 processors and 12 GB memory running Linux Fedora 10.

IO traces for the simulator were generated by running the Industry standard IO benchmark SPC1 and TPCE on the referred proprietary enterprise storage system for a duration about 24 hours. In all the following experiment, “Average response time” refers to the average of the accumulative response time of all the requests for a extent in the extents pool at a particular time point and it is in the unit of miliseconds.

A. Effect of Basic Data Migration

Using the IO trace collected from running the benchmarks on real enterprise storage system to the simulator, we measured the average response time across all data extents every 15 minutes (1 cycle). We compare three different approaches: no data migration functionality, the basic data migration scheme described in Section II-C and reducing the learning phase by reusing the IO profile. Figure 4 shows the comparison of system response time with the three basic migration approaches under the IO trace collected from the TPCE benchmark. The results show that the average response time without data migration is more than twice the response time achieved by the other two approaches which perform basic data migration. The response time remains flat because TPCE workload presents even distribution over time interval. When we turn on the basic data migration scheme, in the first 1 hour, the IO profiles of extents are collected and the extents are sorted based on their heat levels. Then the basic data migration scheme is activated and the extents are migrated in the descending heat order. During the first hour learning phase, the system response time maintains at the same level as the case without data migration. After the first hour learning phase, as the data migration process begins, the system response time starts to drop. Within a few cycles, the data migration process reaches the convergence point and enters into the optimization phase as further data migration no longer creates notable improvement on response time. The graph also shows that reusing IO profiles for a workload with stable characteristics can improve performance at a faster rate than an approach which independently learns the IO profiles each time.

B. Effect of Look-ahead Window Length

In this set of experiments, we study the effects of lookahead window length on data migration. The storage layer is configured with 3 SSDs and 33 HDDs. The total capacity of SSD is set to be much less than the capacity of HDD. The input trace is a hybrid trace with TPCE trace as the first half and SPC1 trace as the second half and works as workload $w1$ and workload $w2$ respectively. In order to understand the impacts of lookahead window on system performance,

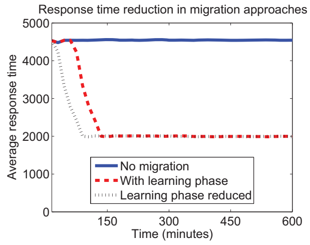

Fig. 4. Response time evolvement in scenarios without migration, with normal migration and with learning phase reduced migration approach in TPCE trace with 2 SSDs and 32 HDDs and with 4GB/5mins migration bandwidth allocated

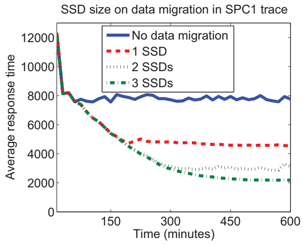

Fig. 5. Data migration with different SSD sizes with SPC1 trace in cluster 1 with 32 HDDs and 4GB/5mins migration bandwidth allocated

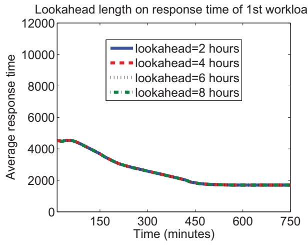

Fig. 6. The impacts of lookahead length on the first workload (TPCE trace on cluster 2) when 2 ≤lookahead length≤ 8 and migration bandwidth=2GB/5mins in the TPCE-SPC1 sequential workload with 3 SSDs and 32 HDDs

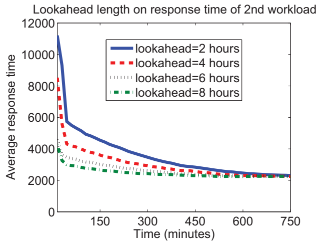

Fig. 7. The impacts of lookahead length on the second workload (SPC1 trace on cluster 1) when 2 ≤lookahead length≤ 8 and migration bandwidth=2GB/5mins in the TPCE-SPC1 sequential workload with 3 SSDs and 32 HDDs

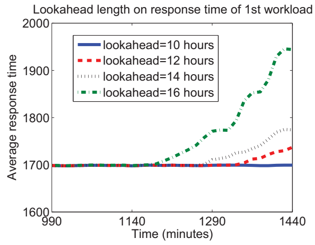

Fig. 8. The impacts of lookahead length on the first workload (TPCE trace on cluster 2) when 10 ≤lookahead length≤ 16 and migration bandwidth=2GB/5mins in the TPCE-SPC1 sequential workload with 3 SSDs and 32 HDDs

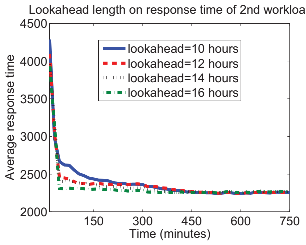

Fig. 9. The impacts of lookahead length on the second workload (SPC1 trace on cluster 1) when 10 ≤lookahead length≤ 12 and migration bandwidth=2GB/5mins in the TPCE-SPC1 sequential workload with 3 SSDs and 32 HDDs

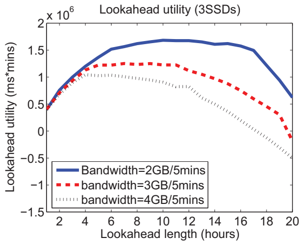

Fig. 10. Lookahead utility cost on incremental lookahead length over multiple migration bandwidth for TPCE-SPC1 workload with 3 SSDs

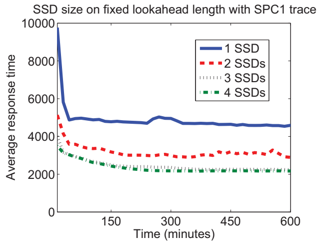

Fig. 11. SSD size on response time in fixed 4 hours lookahead length in SPC1 cluster 1 trace with 32 HDDs and 4GB/5mins migration bandwidth allocated

we vary lookahead window length from 1 hour to 16 hours. For each window size increment, we output the system wide average response time every 15 minutes to observe the impacts of lookahead migration on the performance of workload $w1$ and $w2$. To simplify the problem, we isolate the lookahead migration only to workload $w2$ which help to extract segment that is repeated among the continuous workloads and center on the impacts of lookahead migration on the ongoing workload and the next active workload.

Fig.6 and Fig.7 examines the impacts of lookahead window length on the performance of both first workload and second workload respectively, where lookahead length sets to $2, 4, 6, 8$ hours and the migration bandwidth is set to 2GB/5mins in the TPCE-SPCE workload. Fig.7 shows the impacts of lookahead length on the second workload $w2$. The peak response time of workload $w2$ experiences a continuous dropping as we increase lookahead window length from 2 hour to 8 hours. The peak response time when employing 8 hours lookahead length is $1/3$ of the peak response time with 2 hours lookahead length. This is mainly because larger lookahead length enables more hot data extents to be migrated into SSD, given the fixed bandwidth allocated for migration purpose. The more data extents migrated into SSD, the higher reduction in response time is achieved through access disparity between HDD and SSD. Also larger lookahead length enables the migration process to arrive the minimum stable response time level much earlier. This confirms that lookahead window length is a critical knob to control the effectiveness of lookahead migration. Fig.6 shows the impacts of lookahead length on the first workload. It is notable that lookahead migration with window length within the range of $(2 \leq LookaheadLength \leq 8)$ introduces almost negligible influence on the ongoing workload $w1$.

Fig.8 and Fig.9 present the results if we continue to increase lookahead length to be over 8 hours. Fig.9 shows that the peak response time of $w2$ is reduced at a much smaller rate. From 10 hours to 16 hours lookahead length, the peak response time of $w2$ is only reduced by about 8%. The reduction on the response time is significantly fading as we continue to increase lookahead length. In other words, the migration effect is saturated when lookhead length is larger than 10 hours. This is because the number of hot extents that are visited in high frequency and thus contribute noteworthy influences on the response time is limited in quantity, if these extents are all migrated into SSD during particular lookahead length, then further increment on lookahead length produces no apparent impacts on reducing response time.

On the other hand, Fig.8 shows that as the lookahead length increases from 12 hours upward the response time of the ongoing workload $w1$ also increases. This is because, the longer the lookahead length is, the more extents of $w1$ are

TABLE I
OPTIMAL LOOKAHEAD LENGTH

Bandwidth	Greedy	Formal
2GB/5mins	10 h	10.11 h
3GB/5mins	7 h	7.24 h
4GB/5mins	4 h	4.08 h

swapped out to accommodate the $w2$ extents. Although the $w1$ extents is swapped in the order from cold extents to hot extents, excessive growth of lookahead length may pose the danger that certain useful hot extents of $w1$ are swapped out, leading to increased response time of workload $w1$.

C. Optimal Lookahead Length Computation

Fig.10 shows the evolvement of lookahead utility as the lookahead length increments in different bandwidth settings. From the result, we can see that as the lookahead length increments, lookahead utility increases first and then starts to drop at particular lookahead length. For example, when bandwidth is 4GB per 5mins, lookahead utility reaches maximum value at 4 hours lookahead length. This verifies the effectiveness of the lookahead utility model. Also because of larger bandwidth enables the faster migration of hot extents, thus larger bandwidth reaches the maximum lookahead utility values at a faster speed.

Table I compares the optimal lookahead migration window length experimented through the use of a greedy algorithm with the optimal lookahead window size computed using our formal model for adaptive lookahead migration, under different bandwidth scenarios with 3 SSDs using the same workload as previous experiments. The results show that our formal model for computing optimal lookahead window size is able to derive the optimal lookahead length quickly and precisely without resorting to a greedy approach. .

D. SSD size on Fixed Lookahead Length

Fig.11 shows the response time reduction process when we use SSD size ranging from 1 to 4 SSDs. As the SSD size increases, the converged response time is further reduced. The larger the SSD size is, the lower the peak response time will be. As the size of SSD tier increases, more and more hot extents are migrated into the SSDs. Given that the majority of the hot extents are migrated into the SSDs, further increasing the SSD size does not help to further reduce the response time. This can be exemplified by the overlap of the dotted line representing the 3 SSD scenario and the dash-dotted line representing the 4 SSDs scenario.

V. RELATED WORK

In storage systems, data migration has been employed to achieve load balancing, system scalability, reliability or a myriad of other objectives. Placing data in different tiers and hence moving data plays an important role in tuning the system resource utilization and guaranteeing quality of service. For example, AutoRAID [5] divides its block storage into a high frequency access layer and a low frequency access layer. As the block access frequency changes, the system automatically migrates frequently accessed blocks to higher layer and moves less frequently accessed blocks to the lower layer.

Recently, sereral migration techniques have been proposed as a part of the feedback control loop [6]. [7] propose a storage system which evaluates the component's ability in supporting ongoing workload, and selects the candidates to migrate, aiming at satisfying the workload requirements. However, the migration scheme they proposed lacks of consideration on IO bandwidth, SSD size, and migration deadline and other constraints. [8] studies how to use a control-theoretical approach to adjust the migration speed such that the latency of the foreground workload is guaranteed.

VI. CONCLUSION

In this paper we have developed an adaptive lookahead migration scheme for efficiently integrating SSDs into the multi-tier storage systems through deadline aware data migration. Our lookahead migration scheme takes into account several factors in determining the optimal lookahead window size, including the heat information of data extents from the IO profiles, the migration deadline, the migration bandwidth, and the tradeoff between the gain in response time reduction of the workload being migrated to the fast storage tier and the performance degradation of the ongoing workload. The experimental study demonstrates that the adaptive lookahead migration scheme not only enhances the overall storage system performance but also provides significantly better IO performance as compared to the basic migration model and constant lookahead migration scheme.

VII. ACKNOWLEDGMENTS

The first and last authors are partially supported by grants from NSF NetSE and NSF CyberTrust program, an IBM faculty award, and an IBM SUR grant.

REFERENCES

[1] "NetApp FAS3100 System," http://www.netapp.com/us/products/storage-systems/fas3100/.

[2] "IBM DS8000," http://www-03.ibm.com/systems/storage/disk/ds8000/.

[3] "The Flash Cache Alternative to SSDs," http://www.hds.com/pdf/Hitachi-Universal-Storage-Platform-V-ESG-Brief-0507.pdf.

[4] e. Gong Zhang, Lawrence Chiu, "'automated lookahead data migration in ssd-enabled multi-tiered storage systems."

[5] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The hp autoraid hierarchical storage system," *ACM Trans. Comput. Syst.*, vol. 14, no. 1, pp. 108–136, February 1996.

[6] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch, "Hippodrome: Running circles around storage administration," in *In Proceedings of the Conference on File and Storage Technologies*. USENIX Association, 2002, pp. 175–188.

[7] E. Anderson, J. Hall, J. Hartline, M. Hobbs, A. R. Karlin, J. Saia, R. Swaminathan, and J. Wilkes, "An experimental study of data migration algorithms," 2001.

[8] C. Lu, G. A. Alvarez, and J. Wilkes, "Aqueduct: Online data migration with performance guarantees," in *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*. Berkeley, CA, USA: USENIX Association, 2002, p. 21.