# Cost Effective Storage using Extent Based Dynamic Tiering

Jorge Guerra[†], Himabindu Pucha[*], Joseph Glider[*], Wendy Belluomini[*], Raju Rangaswami[†]
[†]*Florida International University, IBM Research Almaden*[*]

## Abstract

Multi-tier systems that combine SSDs with SAS/FC and/or SATA disks mitigate the capital cost burden of SSDs, while benefiting from their superior I/O performance per unit cost and low power. Though commercial SSD-based multi-tier solutions are available, configuring such a system with the optimal number of devices per tier to achieve performance goals at minimum cost remains a challenge. Furthermore, these solutions do not leverage the opportunity to dynamically consolidate load and reduce power/operating cost.

Our extent-based dynamic tiering solution, *EDT*, addresses these limitations via two key components of its design. A Configuration Adviser *EDT-CA* determines the adequate mix of storage devices to buy and install to satisfy a given workload at minimum cost, and a Dynamic Tier Manager *EDT-DTM* performs dynamic extent placement once the system is running to satisfy performance requirements while minimizing dynamic power consumption. Key to the cost minimization of EDT-CA is its ability to simulate the dynamic extent placement afforded by EDT-DTM. Key to the overall effectiveness of EDT-DTM is its ability to consolidate load within tiers when feasible, rapidly respond to unexpected changes in the workload, and carefully control the overhead due to extent migration. Our results using production workloads show that EDT incurs lower capital and operating cost, consumes less power, and delivers similar or better performance relative to SAS-only storage systems as well as other simpler approaches to extent-based tiering.

## 1 Introduction

Enterprise storage systems strive to provide performance and reliability at minimum capital and operating cost. These systems use high performance disk drives (e.g. SCSI/SAS/FC) to provide that performance. However, solid-state drives (SSDs) offering superior random access capability per GByte have become increasingly affordable. On the other hand, SATA drives offering superior cost per GByte are also attractive for mass storage. Systems with only SSDs are still too expensive, and those built using only SATA would not provide enough performance/GByte for most enterprise workloads. *Multi-tier* systems containing a mix of devices can provide high performing and lower cost storage by utilizing SSDs only for the subset of the data that needs SSD performance.

Current commercial SSD-based multi-tier systems from IBM [29], EMC [17], 3PAR [25] and Compellent [23] provide performance gains and cost savings. However, customer adoption has been slow. One of the reasons for this is the difficulty in determining what mix of devices will perform well at minimum cost in the customer's data center. This optimization task is highly complex because of the number of device types available along with the variability of workloads in the data center.

To address this challenge, two things are needed: configuration tools to assist in building such systems and to demonstrate potential benefits based on customer workload, and capabilities in the storage systems that can optimize placement of data in the tiers of storage. The placement should ensure that actively accessed data is colocated to minimize latency while lightly accessed data is placed most economically. There is also an opportunity to improve operating cost by placing data on the minimum set of devices that can serve the workload while powering down the rest. Current products address some but not all of these challenges. Determining which mix of devices to buy remains a difficult problem, and improvement of operating cost by consolidation and power management has not yet been tackled.

To address these gaps, we develop an Extent-based Dynamic Tiering (EDT) system that includes: 1) a Configuration Adviser tool *EDT-CA* to calculate cost-optimized mixes of devices that will service a customer's workload, and 2) a Dynamic Tier Management *EDT-DTM* component that runs in the configured storage system to place data by *dynamically* moving *extents* (fixed-size portions of a volume) to the most suitable tiers given current workload. EDT-CA works by simulating the dynamic placement of extents within tiers that offer the lowest cost to meet an extent's I/O requirements as they change over time, and thus suitably size each tier. EDT-DTM monitors active workload and manages extent placement and migration in such a way that performance goals are met while optimizing operating cost where feasible by consolidating data into fewer devices within each tier and powering off the rest.

We evaluated EDT-CA and EDT-DTM, using both production and synthetic workloads on a storage system with SSDs, SAS, and SATA drives. Our results show that multi-tier systems using EDT have a device mix that saves between 5% to 45% in cost, consume up to 54%

less peak power, and an additional 15-30% lower dynamic power (instantaneous power averaged over time), at a better or comparable performance compared to a homogeneous SAS storage system. EDT's design choices are critical in achieving these savings. Dynamic extent placement saves 25% in cost compared to a static extent-based system. Including metrics in addition to IOPS in EDT's placement provides a 2× performance improvement compared to a dynamic tiering system that allocates extents based on IOPS alone.

Our work makes the following contributions:

- EDT is the first publicly available work that formalizes and explores the design space for storage configuration and dynamic tier management in SSD-based multi-tier systems. (Section 2)
- EDT consists of a novel configuration algorithm for dynamic tiered systems that outputs lower cost configurations. (Sections 3, 4)
- EDT proposes a novel dynamic placement algorithm to satisfy performance requirements while minimizing dynamic power. (Section 5)
- EDT outperforms SAS-only and other simpler extent-based tiering approaches across a variety of workloads in both cost and power. (Section 6)

# 2 Multi-Tiering: Design Choices

This section describes important design choices for a multi-tier system that enable efficient use of the tiers.

## 2.1 Extent-based Tiering

The first we consider the granularity of data placement. Previous studies [7, 11] suggest that I/O activity is highly variable across LBAs in a volume. Therefore, if data were placed at a volume level based on average volume workload characteristics, a large percentage of the tier will hold data that does not require the tier's capabilities.

Thus, we perform data placement at the granularity of an **extent**, a fixed-size portion of a volume. The smaller the extent size, the more efficient will be the data placement. However, operating at the extent level incurs metadata overhead to keep track of extent locations and other statistics and this overhead increases as extent size is decreased. We choose an extent size with an acceptable system overhead (details in § 6.2). Note that we expect the extent size to be larger than the typical file system block size and hence extents are not expected to align with file boundaries. However, the reduced system overhead for larger extents provides the right tradeoff compared to finer grain approaches.

## 2.2 Dynamic Tiering

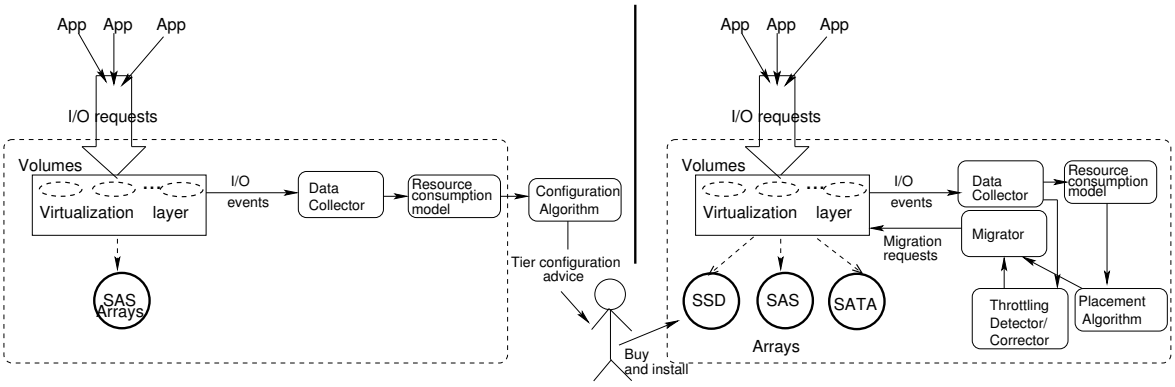
The next design choice deals with the time scale at which extents move across tiers. One choice involves placing extents once during system instantiation or moving them at coarse grain intervals of the order of days or months. However, since studies show that I/O rates of a workload are typically below peak most of the time [16, 19], this static or semi-static placement is not optimal—a placement that configures for the peaks pays extra in both cost and energy for a system that is over-provisioned at off-peak times; and a placement that mitigates cost from over-provisioning by configuring for the average I/O rate suffers from decreased performance during peaks.

The alternate choice is to plan extent movement at intervals on the order of minutes or hours. We refer to this time interval as an **epoch**. Such a system exploits variation in extent I/O rate to improve its efficiency; an extent is on a SATA tier when fairly inactive, and moves to the SAS or SSD tier as its I/O rate goes up. This achieves cost-effective use of resources and/or dynamic energy savings. Similarly, when the performance demanded of a single tier is below its peak capacity, extents placed on the tier can be consolidated into fewer devices for power savings. Often, the set of heavily loaded extents changes over time [11]. Dynamic migration of the heavily loaded extents into SAS or SSD when required enables cost-effective use of the resources. Thus, we choose to perform dynamic data placement with an epoch length of the order of minutes/hours.

The drawback of such a dynamic system, however, is the cost of data migration, i.e., the potential adverse effect on foreground I/O latency and the migration latency itself before the desired outcome. Longer epoch durations allow more time to execute migrations and amortize overhead better. Thus, we pick an epoch duration whose estimated migration overhead is below the allowable system migration overhead (details in § 6.2). Additionally, it is important to ascertain that the overhead of migrating data does not overwhelm its benefit. This depends on the stability of the workload—extents that relocate often benefit less from migration compared to extents that stay longer in a particular tier. The workloads we have studied indicate that dynamic migration is typically beneficial, but we believe that a dynamic system must also be able to back off when lack of workload stability causes dynamic migration to interfere with performance.

## 2.3 Beyond I/O Rate Based Tiering

This design choice determines the extent-level statistics required to match an extent with the right tier. The available public documentation about commercial extent-based multi-tier products indicates use of IOPS to measure load; in these systems high IOPS regions are placed onto SSD while leaving the remainder of the data on SAS or SATA. Although this method is intuitively correct, our preliminary analysis reveals significant drawbacks: IOPS-based placement does not factor in the bandwidth




(a) EDT: Configuration Adviser

(b) EDT: Dynamic Tier Manager

**Figure 1: EDT system architecture.**

requirement of an extent. For example, consider an extent with a long sequential access pattern consisting of small I/Os to contiguous locations. Such an extent will have high IOPS and bandwidth requirements. Our analysis of production and SPC-1 [1] like workload traces (§ 6), collected after the I/O scheduler show such patterns. Using I/O rate statistics for this stream causes sequential streams, which are more cost-effectively served on SAS or even SATA, to be inappropriately placed on SSD. IOPS placement also ignores capacity of the extent. An extent with high IOPS relative to other extents may not have high enough I/O density (IOPS/GByte) to justify the high $/GByte cost of the SSD.

Our approach is to collect more than just I/O counts. We employ a heuristic as in [22] to break down an extent's workload: I/Os that access LBAs within 512 KBytes of the previous ones are taken as part of a sequential stream and contribute to an extent's bandwidth requirement. I/Os further apart are characterized as *random* I/Os and are used to compute a random I/O rate. Thus, for each extent, we collect a random I/O rate and bandwidth. Other methods for separating the I/Os into random and sequential may also be applicable.

# 3 EDT: Design Overview

EDT consists of two elements as depicted in Figure 1: a Configuration Adviser (EDT-CA) that determines the right number of devices per tier to install into a storage system, and a Dynamic Tier Manager (EDT-DTM) that operates inside a running system and continuously manages extent placement across tiers. EDT is expected to be deployed in a commercial storage system as shown in Figure 1 which exports many volumes, includes a virtualization layer that allows volumes to be made up of extents stored in arrays of different device types, is capable of collecting and exporting statistics about extent workloads, and can execute requests to non-disruptively move extents between storage devices.

An example usage scenario is as follows: A user wishes to replace a SAS based storage array with a new, tiered storage system with twice the capability. He collects a trace of his workload over a 24 hour period that he thinks is representative. The trace is then run through EDT-CA which produces the minimum cost configuration of SSD, SAS, and SATA that can provide 2x the performance of the existing system. EDT-CA is aware of the runtime migration capabilities of EDT-DTM and takes them into account when determining the configuration. The user installs the new system. During operation of the new system, EDT-DTM manages migration between tiers by continuously collecting extent level statistics, consolidates data onto lower-power tiers when possible, and monitors the system to ensure that the workload performance is not throttled.

In general, EDT-CA starts by determining the workload requirements for the system it is going to configure. This can either be done with a user generated general description of requirements including IOPS, seq/random mix, length of I/O requests, and their distribution across extents, or by using time series data collected from a workload running on an existing system. For the scope of this work, we assume availability of time series statistics. In this approach, EDT-CA takes a epoch-granularity trace of extent workload statistics sampled at times when storage system usage is high. It then estimates the resources required in different tiers to satisfy that workload by simulating placement of each extent in a tier that minimizes its incurred cost while meeting its performance requirements. It repeats this process every epoch and assigns extents to their lowest cost tier based on their performance requirements in that epoch. At the end of this simulation, EDT-CA determines the set of devices that are needed based on the maximum number of devices needed in each tier over all the epochs. This configuration determines the set of devices purchased by the user.

Once the new tiered system is up and running, EDT-DTM manages extent placement. It collects extent level statistics, estimates extents' resource consumption in dif-

ferent tiers, and then plans and executes migrations. EDT-DTM implements a throttling correction mechanism to ensure that performance requirements are satisfied as they vary over time; it constantly monitors array performance and if performance throttling is detected relocates extents to restore performance. EDT-DTM's placement algorithm seeks to place each extent into the lowest-energy tier that satisfies its performance requirement and then to further minimize energy by consolidating extents in the same tier into fewer devices allowing unused devices to be powered down. Both these algorithms use a Migrator module to move extents.

EDT-CA and EDT-DTM work together to minimize cost. EDT-CA minimizes acquisition cost, and EDT-DTM minimizes operating cost. As our results will show, configurations based on static extent placement are more expensive both to acquire and operate.

## 3.1 Common Components

EDT-CA and EDT-DTM share components that collect statistics and calculate resource consumption.

### 3.1.1 Data Collector

The *Data Collector* receives information about I/O completion events including the transfer size, response time, logical block address (LBA) , the volume ID to which the I/O was issued, and the array which executed the I/O. The collector then maps the (LBA, volume id) pair of each I/O to a unique extent in the system, and compiles for each extent, the number of random I/Os and the number of transferred bytes. It then periodically (every minute in our implementation) computes instantaneous bandwidth and random IOPS per extent as well as an exponentially-weighted moving average. In addition to the extent statistics, the collector aggregates statistics per array. It maps each I/O to its array and compiles its IOPS and average response time. These measurements are used by EDT-DTM to determine if I/Os on an array are being throttled. For a very large system the amount of data collected by the data collector may be significant. If this is an issue, the the extent size can be made larger to reduce the volume of statistical data.

### 3.1.2 Resource Consumption Model

The *Resource Consumption Model* uses the extent statistics to estimate the resources it consumes when placed on a device of a given type. Resources are allocated based on the observed capacity and performance requirements at the device level. Therefore, any workload optimizations like deduplication, compression, and caching do not need to be considered in these models as their effects will be captured by the usage statistics.

An extent consumes the resources of a device along capacity and performance dimensions. Consider an extent of size $E_c$ and a performance requirement $E_p$ determined by its random IOPS rate (*RIOR*) and bandwidth measured in previous epochs. The fraction of capacity required to host an extent $E$ in device $D$ ($RC(E_c,D)$) is straightforward:

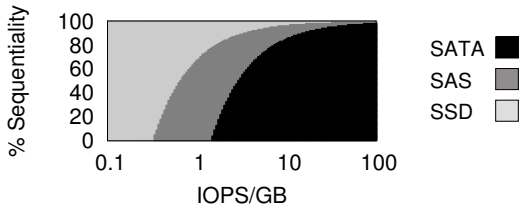



**Figure 2: Lowest cost tier for extents with different characteristics.**

$$RC(E_c,D) = \frac{\text{Capacity required by extent}}{\text{Total space in device}}$$

For performance utilization, we use a simplified model based on Uysal *et al.*'s work [30]. The performance resource consumption of extent $E$, when placed on device $D$ ($RC(E_p,D)$) is:

$$RC(E_p,D) = \text{RIOR} \cdot \text{Rtime} + \text{Bandwidth} \cdot \text{Xtime}$$

Here *RIOR* is the number of random I/Os sent to an extent in a second (IO/s) and Rtime is the expected response time of the device (s/IO). *Bandwidth* is the bandwidth requested from the device (MB/s), and *Xtime* is the average transfer time (s/MB). The result of this equation is the fraction of the device performance utilized by an extent. Note that the Rtime and Xtime values are averages and may need to be adjusted depending on the expected workload. For example an SSD with a mostly random write workload would have significantly higher Rtime than the same SSD with a mostly random read workload. The overall resource required by an extent is then the maximum of the capacity utilization fraction and the performance utilization fraction:

$$RC(E,D) = max(RC(E_p,D), RC(E_c,D))$$

The resource consumption model determines the most efficient tier for an extent. For instance, when minimizing cost, the most suitable tier is the one where the extent incurs the lowest cost (the product of the device cost and the extent's resource consumption on that device). Figure 2 confirms the advantage of multi-tier systems since the most cost-effective tier changes with extent characteristics, namely the total IOPS and the percentage of sequential accesses among three classes of storage devices specified in Section 6. As expected, we observe that mostly idle extents favor SATA, medium IOPS favor SAS, and high IOPS favor SSD. Further, as expected, more sequential extents favor HDDs.

# 4 Configuration Adviser

EDT-CA builds on the Data Collector and the Resource Consumption Model described above. Since configuration is an NP-Hard packing problem, we propose a light-weight heuristic to achieve low cost extent placement:

1. **Binning.** For each extent $E$, and device type $D$, we compute the cost of allocating the extent to that device as extent cost(E, D) = cost(D) · RC(E,D). The extent is then placed in the tier that meets its performance with the lowest cost. Iterating over all the extents, the above computation separates the extents into bins, one per each tier.
2. **Sizing a bin.** For each bin, we obtain its performance and capacity resource consumption as $RC_p = \sum RC(E_p, D)\ \forall E$, and $RC_c = \sum RC(E_c, D)\ \forall E$. The maximum of these two values gives the total bin resources required, and the number of required devices of this bin type are computed by rounding up this sum to the nearest integer value.
3. This process is independently repeated for each epoch to identify the number of devices per tier that yields minimum cost for that epoch.
4. The last step consists of combining these different configurations to obtain a final system configuration valid across time. For the scope of this work, we achieve the final configuration by allocating the maximum number of devices of each type used across all epochs. That is, if at epoch $t_0$ 2 devices of type $D$ and 1 of type $D'$ are the most cost effective, but at epoch $t_1$ 1 of type $D$ and 2 of $D'$ is better, then our method will indicate that we need 2 of type $D$ and 2 of $D'$.

Our current method of combining configurations across epochs is fairly conservative and could potentially result in an over-provisioned system. However, as our current algorithm already results in lower cost configurations (Section 6), we relegate exploring more efficient ways of combining configurations over time to future work. Also note that when we compute tiered configuration for each epoch independently, we assume that the extents can be suitably migrated between epochs if required. As part of our future work, we intend to model the required number of migrations, and suitably adjust the provisioning if the required migrations exceed the maximum number of migrations a system can support in a chosen interval of time. Finally, our Configuration Algorithm can also be used to upgrade a multi-tier system to meet upcoming performance demands.

# 5 Dynamic Tier Manager

EDT-DTM combines three new modules with the Data Collector and the Resource Consumption Model to continuously optimize extent placement: (1) a Tiering and Consolidation module, (2) a Throttling Detector/Corrector module, and (3) a Migrator module.

## 5.1 Tiering and Consolidation Algorithms

At the end of every epoch, the Tiering and Consolidation (TAC) algorithms generate an extent placement to satisfy extent performance requirements and minimize dynamic system power. Such an energy efficient placement can be achieved both by leveraging the strengths (i.e. performance or capacity per watt) of the heterogeneous underlying hardware (SSD, SAS, and SATA drives), and by consolidating data into fewer devices when possible and turning off the unused devices.

Similar to the configuration problem, placement for power minimization is also NP-Hard, and we propose a heuristic solution. TAC requires two inputs: (1) current random I/O rate and bandwidth for each extent from the actively running system, and (2) size (in bytes) and the random I/O rate and bandwidth capability for each array in the storage system. It then uses a two-step process to output a new extent placement that aims to adapt to the changes in the workload as follows:

(1) **Tiering.** For each extent $E$, and device type $D$, we compute the "fractional power burden" of allocating the extent to that device as extent power(E, D) = power(D) · RC(E,D). The extent is then placed on the tier that meets its performance with the lowest power consumption. Doing so allows EDT to reduce active power via consolidation (described next). Iterating over all the extents results in one bin per tier. The assignment of extents to a tier is performed locally on an extent by extent basis, irrespective of the total performance needs or available space in that tier.

(2) **Consolidation.** Extents assigned to each tier are then sorted using their *RC* values and placed in arrays using the First Fit Decreasing heuristic, a good approximation algorithm to the optimal solution for extent packing [35]. When extents already assigned to the tier under consideration exceed its available performance (i.e., resource consumption metric for the assigned extents exceeds 1) or the tier runs out of space in the available arrays, the remaining extents in the extent list are demoted to the tier with the next lower power burden for that extent. This packing process is now repeated for all the tiers, consolidating extents into a minimum number of arrays in a tier. Extents already in the right tier and on an array that will remain powered on in this epoch retain their position from the previous epoch, thereby saving migrations. Any unused arrays from the extent placement are set to a lower power state to conserve energy.

## 5.2 Throttling Detector and Corrector

While the TAC mechanisms enable dynamic performance and power optimization, unexpected load and

working set changes can suddenly alter the performance requirements of extents. However, tracking this performance change, especially when an extent's I/O rate increases, is challenging. Extents placed in a low performance tier cannot exhibit high I/O rates even when the application above may desire it. This causes *throttling* of the true IOPS requirement of the extent, artificially limiting it to a low value. The Throttling Detector overcomes this limitation by monitoring the average response time of each active array every minute.

If the average response time of I/Os from an array indicates that undesirably high request queuing is occurring in the array, EDT decides that the array is throttling the true IOPS requirement of applications and causing delays. When throttling is detected, pending migrations driven by TAC are immediately halted and EDT-DTM switches to a *throttling correction* mode to perform recovery. To respond rapidly and minimize the possibility of future throttling in the same array, the load on the throttled array is shed by migrating a minimum set of extents responsible for at least half of its current total performance resource consumption.

To select the target array(s), we first start by considering the best possible tier for each extent being migrated, and within that tier we first examine arrays which are already active to see if they can absorb the new extent. If none can host the new extent, we consider arrays that are not in use in that tier if any are available. If the best tier can not accommodate the extent we try the same approach on tiers with the next higher power burden for that extent. If the array continues to remain throttled after half the load on the array has been migrated, the extent migration process is repeated, until the system is no longer throttled. The entire system stays in recovery mode while an array remains throttled, suspending energy optimizing migrations. When no arrays are throttled, the system switches back to the TAC placement after an epoch elapses.

## 5.3 Migrator

The Migrator handles the data movement requests from TAC and the throttling algorithms. It compares the new placement of the extents from the above algorithms to their old placement, and identifies extents that need to be migrated. It then schedules and optimizes these migrations. On one hand, migrations that relieve throttling must be completed quickly. On the other hand, migrations cause additional I/O traffic, and care must be taken so that they do not affect the foreground I/O performance.

Our migration scheme achieves this tradeoff as follows. We allow every device to be involved in only one migration operation at a time. Thus, before issuing a migration request, the Migrator performs admission control by allowing requests only if the source and target device are both available. If they are not, the request is re-queued and it moves onto the next request. Further, the Migrator controls its migration-related resource consumption by decomposing an extent into smaller transfer units and *pacing* the transfer requests to match the minimum of the available or the desired I/O rate. Further if the migration is being performed to relieve throttling, once a transfer unit is migrated, any foreground I/O requests to it are handed by the destination array. Note that because of this pacing not all planned migrations may be completed before the next epoch. In such cases, the migration queue is flushed, and requests resulting from the new epoch's computation are queued. We further optimize by retaining the old location of the extent if it is already in the right tier during the consolidation step. Finally, we could potentially incorporate other optimizations [4, 9, 31, 36] such as multiple locations for the same extent [31], and proactive migrations [36].

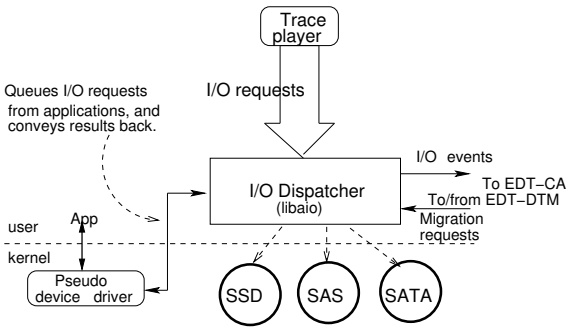



**Figure 3: Storage subsystem platform for evaluating EDT-CA and EDT-DTM.**

# 6 Evaluation

Our evaluation uses both a SPC-1-like [1] benchmark workload and multiple production enterprise workloads from MSR [21] to demonstrate that:

- In comparative evaluation, EDT-CA works to minimize cost, and EDT-DTM satisfies performance requirements while lowering power consumption.
- EDT's dynamic behavior and detailed resource consumption model help achieve its goal.
- Extent based dynamic optimization and consolidation are feasible in practice with little overhead.

## 6.1 Methodology

**Comparison candidates.** We compare EDT to three alternate solutions:

**1.** *SAS* is chosen to represent current enterprise storage system deployments that predominantly use only high performance SAS drives. The configuration is derived

| Device | Cost ($) | Power (Idle, Active) | Random IOPS | BW (MB/s) | Rtime (ms/IO) | Xtime (ms/KB) |
|---|---|---|---|---|---|---|
| SSD | 430 | 0.5, 1 | 5000 | 90 | 0.2 | 0.01 |
| SAS | 325 | 12.4, 17.3 | 290 | 200 | 3.75 | 0.004 |
| SATA | 170 | 8.0, 11.6 | 135 | 105 | 9 | 0.009 |

**Table 1: Characteristics of devices used in the testbed.**

using the capacity and **peak** performance (IOPS and bandwidth) requirements of the workload. Volumes are statically assigned to SAS arrays in a load-balanced manner.

**2.** *EST* (Extent-based Static Tiering) places extents on tiers statically to quantify the benefit from tiering. Configuration is performed as follows: at every epoch, the cost to place each extent on each tier is computed as done by EDT-CA using capacity, IOPS, and bandwidth requirements. An extent is then permanently placed on the tier that minimizes the sum of its instantaneous costs over all epochs. Once extents are binned into tiers, the number of devices for each tier is determined using that tier's peak resource consumption.

**3.** While SAS and EST illustrate the benefit from EDT's design choices incrementally (going from a homogeneous system to static tiering and then to dynamic tiering), we propose a third candidate to illustrate a different design decision in dynamic multi-tier systems—*IDT* (IOPS Dynamic Tiering) implements extent-based dynamic configuration and placement using a greedy IOPS-only criteria where higher IOPS extents move to higher IOPS tiers. This is in contrast to EDT that uses a combination of capacity, IOPS, and bandwidth in its placement algorithm.

**Implementation.** Our test system is shown in Figure 3. In addition to EDT, we implemented an I/O dispatcher that receives block I/O requests from applications, maps the logical block address to the physical device address, performs the corresponding I/Os, and communicates with the EDT components. Our *trace player* application issues block I/Os from a trace via a socket to the I/O dispatcher. To support real-world applications without modification, we implemented a pseudo block device interface. For the scope of this work, we use Linux's default *deadline* scheduler, and our measurement of context switch overhead when running through the pseudo device driver was negligible ($< 10\mu$s).

**Experimental Testbed.** Our experimental platform consists of an IBM x3650 with 4 Intel Xeon cores and 4 GB memory acting as the I/O dispatcher. It is connected via internal and external SAS ports to 12 1 TB 7200 rpm 3.5" SATA drives, 12 450 GB 15K rpm 3.5" SAS drives, and 4 180 GB Intel X25-M SSD drives. Table 1 shows the characteristics of these devices. The enclosures containing the drives are connected to a *Watts up? Pro* power meter. We report the disk power obtained by subtracting the baseline power used by the non-disk components of the enclosure (154 W).

**Metrics.** To compare solutions, we evaluate static configuration results using capital cost and peak power consumption, and we evaluate dynamic behavior using the average and distribution of I/O latency along with dynamic power consumption. Peak power consumption is obtained using disk drive data sheets. Dynamic power consumption is measured using the power meter.

## 6.2 Parameter Selection

**Extent size.** Smaller extents use tier and migration-related resources more efficiently and enable faster response to workload changes, but also incur greater metadata overhead. Our approach was to pick the smallest extent size that incurs acceptable metadata overhead. Assuming metadata can have a reasonably small overhead of at most 0.0001% of the total storage capacity, and given 200 bytes/extent for metadata overhead (mostly from recording extent-level statistics) in our implementation, the smallest extent size our storage system can support is 20 MB. To introduce some slack we used 64 MB extents for our experiments.

**Epoch duration.** Shorter epochs allow quicker response to workload changes, but can also result in increased extent migration. As the epoch duration increases, the stability of extent characteristics increases due to averaging over longer periods and consequently the migration bandwidth overhead decreases. We picked epoch durations that resulted in migration bandwidth limited to a 10% fraction of the available array-pair bandwidth in the system[1]. This prevents migration from significantly degrading performance and ensures that migrations complete early within each epoch. For the MSR workloads this calculation resulted in a 30 minute epoch.

## 6.3 Synthetic Workload

This SPC1-like workload was chosen because it simulates an industry standard benchmark and provides a contrast to the MSR trace workloads. We ran the SPC1-like workload generator on a 1 TB volume at 100 BSUs for 30 min using an over-provisioned configuration (a 12 SAS RAID-0 array). We chose 30 min because the workload is quite static after a short startup period. The resulting trace was used to obtain the number of devices required per tier for different methods (Table 2 ).

We observe that all the extent-based tiering configurations outperform SAS configurations in both capital cost and peak power consumption. EDT reduces cost by 14%, and peak power by 55% compared to SAS. Cost incurred

[1]Medium to large scale tiered storage systems would typically perform simultaneous extent migrations across multiple array-pairs.

| System | # of Disks | Energy | Cost | Avg RT |
|---|---|---|---|---|
| *SAS* | (0, 6, 0) | 103.8 W | $1950 | 28 ms |
| *EST* | (2, 2, 1) | 46.6 W | $1680 | 15 ms |
| *IDT* | (2, 1, 1) | 29.3 W | $1355 | 21 ms |
| *EDT* | (2, 2, 1) | 46.6 W | $1680 | 15 ms |

**Table 2: Configuration for synthetic workload. The number of disks per tier is specified as (SSD, SAS, SATA). The average response time is obtained from running the configuration with 100 BSUs .**

to configure EST and EDT for this relatively static workload are similar. Although the IDT configuration seems to provide the least cost configuration, this is an artifact of rounding up required devices to the next higher integer. Using fractional devices, costs for EDT and IDT are much closer ($890 vs. $920). Note that in larger systems rounding effects will be less significant.

To confirm that EDT's lower cost is not at the expense of performance, we ran the SPC1-like workload for 30 minutes at 100 BSUs. Given the stability of the workload, migration overhead was minimal. We therefore chose an epoch of 5 minutes to complete the experiments quickly. The SAS scheme used 6 SAS RAID-0 array. Other schemes operated on individual disks. We started EDT and IDT with the entire volume in the SATA tier and allowed dynamic extent migration to reach optimal configurations over time. EST, which does not support extent migration, was started with extents in their most suitable locations as per the EST configuration.

The last column of Table 2 shows the average response times for 100 BSUs measured starting at the end of the first epoch, once the extent placements of the dynamic tiering configurations become effective. Given the workload's stability, results for EDT and EST are identical. They both achieve a 40% lower response time compared to SAS, and improve on IDT's IOPS only placement by 20%. Note that the dynamic power consumption in these experiments is similar to the peak power due to the lack of workload variation.

## 6.4 Production Workload

Our next workload (*MSR-combined*) represents the more interesting class of real-world workloads, obtained by combining the I/Os to the 31 (out of 36) most active volumes of a production storage system [21] for a total of 4580 GB. Including the remaining 5 volumes was not feasible given the hardware restrictions of our testbed.

**Configuration outcomes.** Configuration outcomes based on six days of the *MSR-combined* workload, shown as the "Equal Performance" group in Table 3, indicate that the tiering configurations have lower cost compared to SAS. EDT incurs the lowest cost (50% reduction compared to SAS and 25% relative to EST).

| Config | System | # of Disks | Energy | Cost |
|---|---|---|---|---|
| Equal Performance | *SAS* | (0, 16, 0) | 276.8 W | $5200 |
| | *EST* | (5, 2, 4) | 82 W | $3480 |
| | *IDT* | (4, 1, 4) | 64.5 W | $2725 |
| | *EDT* | (3, 2, 4) | 81.6 W | $2620 |
| Equal Cost | *SAS* | (0, 12, 0) | 204 W | $3900 |
| | *EST* | (4, 4, 4) | 116 W | $3700 |
| | *IDT* | (4, 4, 4) | 116 W | $3700 |
| | *EDT* | (4, 4, 4) | 116 W | $3700 |

**Table 3: Configuration for *MSR-combined*. Configurations achieving equal performance depict improvement in cost and peak power. Configurations at equal cost are created for experimental ease. Number of disks in each tier specified as (SSD, SAS, SATA).**

EDT's ability to effectively time share high-cost, high-performance tiers across extents and satisfy sequentially accessed ones with the SAS tier (instead of the SSD tier) results in more cost-effective configurations. Extents placed in the SATA tier (4336 GB) are mostly idle with random IOPS below 0.32, those in SAS (69 GB) are dominated by bandwidth higher than 1.45 MB/s and random IOPS less than 1.43, and the SSD extents (175 GB) have random IOPS between 1.45 and 858. Tiered configurations substantially reduce peak power when compared with SAS; IDTs greater use of the SSD tier (relative to SAS) makes it the most power-efficient.

**Performance and Power outcomes.** Not all of the equal performance configurations listed in Table 3 were feasible on our experimental testbed due to hardware limitations. Consequently, we decided to switch to equal cost configurations (shown in Table 3) to contrast performance at equal cost instead of cost at equal performance only for the *MSR-combined* workload. Later, we shall explore equal performance configurations for feasible subsets of volumes (Figure 6). EDT's configuration was chosen as the base for all the tiering systems, and its configuration requirements were rounded up to integer number of arrays, each array consisting of 4 devices. SAS used only SAS drives for the same cost, split into 4 disk RAID 0 arrays. We then replayed day one from the seven day trace, the most active 24 hour period of the *MSR-combined* workload. Both EDT and IDT were bootstrapped using a load balanced volume placement.

Figure 4 summarizes the results of this experiment for the candidate solutions. First, we notice that the I/O response time distribution of EDT is clearly superior to the other three solutions, highlighting the importance of considering random IOPS, bandwidth, and capacity when making tiering choices. The average response time with EDT was 2.94 ms while those for the SAS, EST, and IDT were 5.12, 9.33, and 5.93 ms respectively. Further, the $95^{th}$ percentile response time for EDT was under 7.86

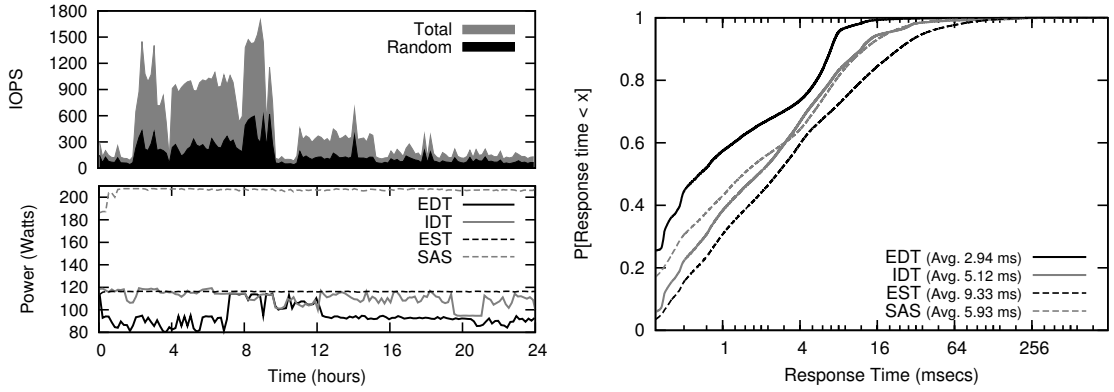



**Figure 4: I/O rate and power consumption (left) and response time distribution (right) for *MSR-combined*.**

ms while the same for SAS, EST, and IDT were 19.31, 37.06, and 17.891 ms respectively. On average, EDT decreased the dynamic power consumption by 13% relative to its peak power, 55% relative to SAS and at least 10% relative to IDT and EST. This dynamic power savings result is likely to underestimate power savings observed in real deployments given that the workload was generated by consolidating multiple uncorrelated workload traces, which tended to reduce the workload variability that would enable dynamic power savings. Additionally, the experiment was done over the most active period, which required most devices to be active for performance. Further, all the configurations here are sized to meet the observed workload. Typically, however, storage purchases are made to accommodate future growth and hence over-provisioned to begin with, resulting in more dynamic power savings.

**Analysis.** We illustrate how EDT achieves its superior performance using two example extents chosen from the experiment and contrasting them with IDT. Figure 5 shows the sequential and random IOPS over time for two extents along with the tier they are placed in. For extent A (top graph), both IDT and EDT move the extent from the SATA tier (the default initial location) to higher performing tiers when the total IOPS requirements increase. However, IDT allocates the SSD tier starting from hour 3 on account of the exponentially weighted moving average (EWMA) of total IOPS whereas EDT allocates the SSD tier only when the EWMA of *random* IOPS of the extent is high. Thus, EDT can better capitalize on the superior sequential performance of the SAS tier to minimize capital costs during configuration and sustain performance during operation. Extent B (bottom graph) illustrates similar behavior during predominantly sequential accesses. Further, both EDT and IDT rightly move extent B into the SATA tier when it becomes idle, aiding in power savings. Thus, EDT is successfully able to pick the best tier for an extent's workload and relocate it

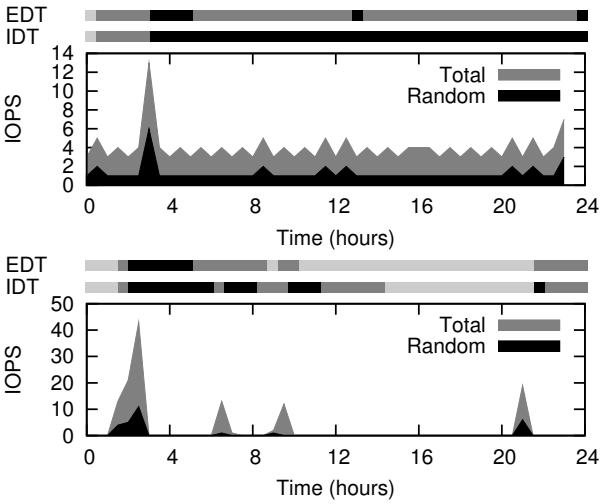


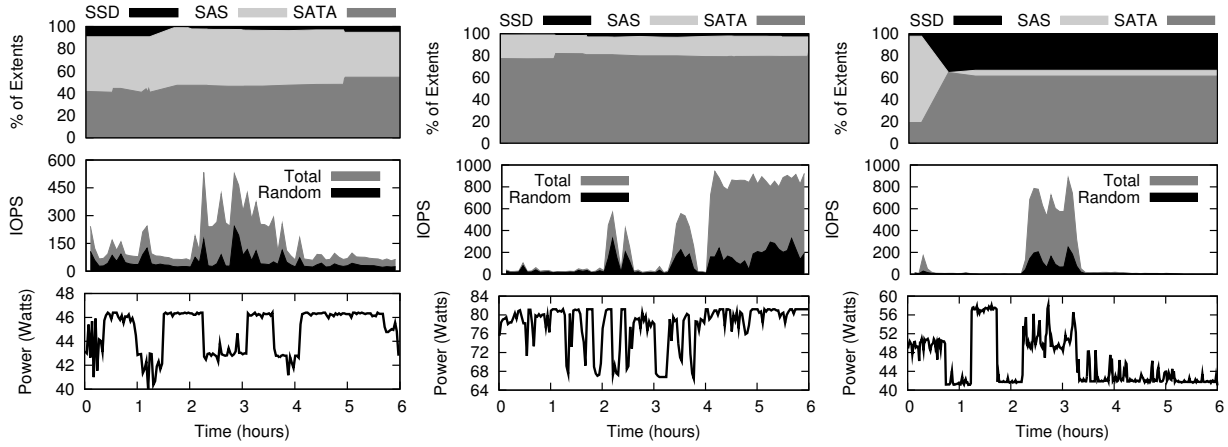

**Figure 5: Contrasting extent migrations for EDT and IDT. The two upper lines denote extent placement for the different algorithms. Black is SSD tier, dark grey SAS and light grey SATA.**

when the requirements change. Regarding the overheads for this migrations, both EDT and IDT migrated around 120 extents per epoch, using an average bandwidth of 42 MB/s which only represents 3% of the total available.

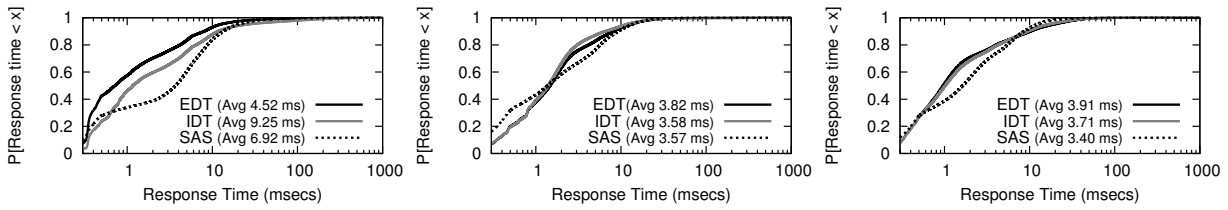
| Workload | Volumes | Cap (GB) | Accessed |
|---|---|---|---|
| *server* | hm, mds, prn, prxy, stg, ts, wdev, web | 1650 | 30% |
| *data* | proj, rsch, usr | 3719 | 34% |
| *srccntl* | src1, src2 | 904 | 29% |

**Table 4: Sub-workloads derived from MSR.**

**Varying the workload.** To analyze the sensitivity of the various algorithms to workload characteristics, we grouped volumes from the MSR workload as specified in Table 4 to create the *server*, *data* and *srccntl (source*




(a) EDT's extent distribution, I/O rate, and EDT's power consumption over time.




(b) CDF of response times

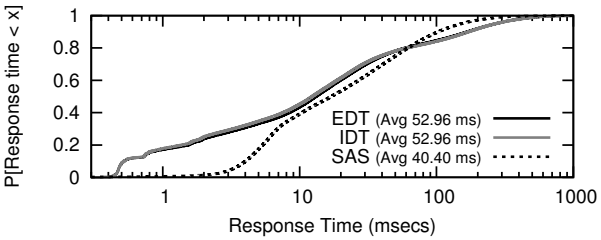
**Figure 6: Replaying 6 hours of the MSR sub-workloads. First column is *server*, second *data*, and third *srccntl*.**

| Workload | System | # of Disks | Energy | Cost |
|---|---|---|---|---|
| server | *SAS* | (0, 6, 0) | 103.8 W | $1950 |
| | *EST* | (2, 1, 2) | 42.5 W | $1525 |
| | *IDT* | (2, 1, 1) | 30.9 W | $1355 |
| | *EDT* | (1, 2, 1) | 47.2 W | $1250 |
| data | *SAS* | (0, 10, 0) | 173 W | $3250 |
| | *EST* | (2, 2, 3) | 71.4 W | $2020 |
| | *IDT* | (1, 2, 4) | 82 W | $1760 |
| | *EDT* | (1, 2, 4) | 82 W | $1760 |
| srccntl | *SAS* | (0, 6, 0) | 103.8 W | $1950 |
| | *EST* | (2, 3, 1) | 65.5 W | $2005 |
| | *IDT* | (2, 2, 2) | 59.8 W | $1850 |
| | *EDT* | (2, 2, 2) | 59.8 W | $1850 |

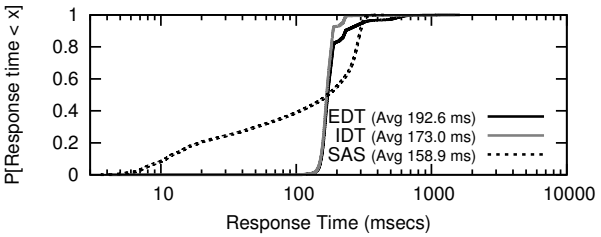
**Table 5: Configuration for MSR sub-workloads. Number of disks in each tier specified as (SSD, SAS, SATA).**

*code control)* workloads. Configuration outcomes for each sub-workload using SAS, IDT, and EDT are presented in Table 5. As with *MSR-combined*, the dynamic tiering solutions are able to configure both lower-cost and lower-energy systems when compared with SAS and EST. Further, in the case of the *server* workload, EDT optimizes the configured system cost with a single SSD relative to the two SSDs recommended using IDT. Given that EST had significantly inferior performance for *MSR-combined*, we did not consider it for further analysis.

Figure 6 shows EDT's dynamic power consumption and extent distribution across tiers over time, as well as its response time distribution relative to IDT and SAS. First, unlike *MSR-combined*, these workloads do have substantial periods of lower utilization. Consequently, in addition to improving the capital cost and peak power consumption, EDT's dynamic consolidation allows dynamic power savings of as much as 15-31% relative to its peak power across the three workloads. The extent distribution is quite different across the workloads. EDT uses the SSD tier substantially for the *srccntl* workload. IOPS-wise one would think that the workload should be completely consolidated to the SATA; however, EDT leverages the fact that the SSD tier offers improved energy efficiency for up to 40% of the extents. The SAS tier was most used for *server*, in particular between hours 2-4 when sequential activity dominates. The *data* workload predominantly utilizes the SATA tier (as evidenced in the configuration outcome) since the IOPS per extent for most extents is very low, easily accommodated using SATA devices. Finally, in this equal performance configuration experiment, the response time performance with EDT is either similar or better than the SAS and IDT schemes across the workloads.




(a) Uniformly Random Load




(b) Varying Hot Extent Set

**Figure 7: Extent distribution and CDF for the adversarial workload.**

## 6.5 Adversarial Workloads

Finally, we measure the impact of using EDT with workloads completely different than the one it is provisioned for. We used the configuration obtained for the *srccntl* workload (in Table 5), and instead of the trace from that workload, we ran two separate synthetic workloads for two hours each: (1) a uniformly random workload at 400 IOPS, where each I/O is issued to a random page in the system. (2) a workload at 500 IOPS, where I/Os are issued to a chosen set of 10 hot extents initially in the SATA tier and this set changes every minute.

Figure 7 depicts the distribution of response times for both workloads. The uniformly random workload yields a 31% higher average response time for EDT and IDT compared to SAS. This can be attributed to the constant migration I/O moving extents away from the throttled SATA tier to both SAS and SSD tiers. Interestingly, we see only a 21% penalty for EDT in the second workload. Analysis shows that throttling of the newly active extents was promptly detected and the extents were migrated quickly to the SSD before they became cold. As illustrated by these examples, EDT can handle unexpected workloads using its throttling detection/correction techniques without major performance penalties.

# 7 Related Work

We build on a rich body of related work in multiple areas.

**SSD-based storage architectures.** Several products (IBM's EasyTier [29], EMC's FAST [17], 3PAR [25], and Compellent [23] systems) incorporate SSDs in storage tiering solutions. Since technical details of these approaches are not published, EDT is the first to provide insight into design choices and components, detailed evaluation across workloads, and analysis of benefits and challenges in building SSD-based multi-tier systems. Moreover, the publicly available documents of these products indicate that although they achieve cost savings and performance improvements, there is little focus on tools aiding admins/customers to configure the right device mix for their workload or on incorporating algorithms that target dynamic energy savings. EDT addresses these limitations.

Another approach to leverage solid state technology in storage systems is to deploy flash devices as a cache between DRAM and HDD. NetApp's FlashCache [24] which follows this approach cites cost reduction and performance improvement when coupled with SAS/SATA drives. Interestingly, Narayanan *et al.* [22] have argued that a SSD cache layer above SAS disks was generally not cost effective compared to an all SAS configuration at the same performance. We did find cost savings using SSD, but our system included much lower cost SATA disks to improve overall cost. Unfortunately, a detailed comparison between SSD caching and tiering would take a significant effort and more space than is available in this paper. However, our summary thoughts on the two architectures are: 1) SSD caching will utilize the SSD space more efficiently and can be more responsive to very dynamically changing workloads, but 2) SSD tiering enables both cost and energy savings even in enterprise environments.

**Storage configuration (also referred to as *provisioning*).** Systems such as Minerva [3], Hippodrome [5], and DAD [6] address the problem of optimizing storage configuration by iteratively applying several steps such as configuring a low cost storage system, choosing RAID levels and other array parameters, and assigning entire volumes to arrays. EDT-CA's focus on obtaining the right mix of storage devices to minimize cost is similar to the configuration step in these systems. The key difference is that EDT-CA is inherently aware of, and utilizes the flexibility afforded by EDT's dynamic extent placement. EDT's data layout also operates at a much finer extent granularity. In EDT, we use a model to predict the utilization of an extent (given its bandwidth and random IOPS) that is similar in spirit to the previously proposed store level performance predictor [30] in its accounting for the differential load induced by sequential and random accesses to an extent. Finally, EDT-CA can be enhanced to perform utility based provisioning as in [28].

**Tiering.** Migration-based storage tiering has been prevalent in the industry for a long time in the form of Hierarchical Storage Management systems, Information Lifecycle Management solutions, and other forms of coarse-grain tiering [2, 13, 15]. Most of these systems differ

from EDT in that they generally migrate data from upper to lower tiers, based on its age rather than on load. Further, these systems operate on volume, file system, or file objects rather than extents, and as such are suited more for file layer systems than block layer systems. Wilkes *et al.* propose AutoRAID [33], a storage system where extents within volumes are migrated between faster RAID-1 arrays and slower RAID-5 arrays according to workload and age. Significantly different algorithms for migration decisions tuned to the specific two tiers are proposed. Additionally, AutoRAID does not consider the issue of correctly determining a device mixture to satisfy given workloads.

**Storage energy efficiency.** EDT uses a consolidation algorithm to save energy in primary storage systems. Other energy saving approaches that instead spin down a fraction of the available disk drives with active data [8, 10, 18, 20, 21, 26, 27, 31, 32, 34] either are not applicable in many primary storage systems due to the significant spin up latency, or require undesirable capacity over-provisioning for redundant data. Work leveraging Dynamic RPM capability (e.g., [12, 26, 37, 38]). is complementary to EDT. In fact, Hibernator [38] also leverages tiering but varies RPM setting of the drives to minimize energy.

# 8 Discussion

**Extending the resource consumption model** In this work we assumed RAID-0 arrays when estimating how much resource on a tier is consumed by a given workload. In commercial applications of EDT, more sophisticated models will be needed to estimate resource consumption in arrays with different RAID levels. Such models do already exist in the industry, so we believe incorporating this capability will be straightforward. Also, for the scope of this work, we assume that all arrays are at the same reliability level, and hence migrating data across arrays is not restricted. However, it is feasible to remove this constraint by observing policies to limit the migration targets of extents. Finally, the resource model may need be enhanced to better model the behavior of disks servicing multiple sequential IO streams in parallel. The current model does not account for degradation in sequential performance that may occur when a disk needs to service multiple sequential streams at once.

**Disk power fraction in the overall energy of a storage system.** The chief dynamic energy-saving technique proposed in this work is powering down empty disk drives. However, we find that in today's commercial storage systems, disk drives typically consume $\sim$50% of the total storage system energy [14] while the rest is consumed by other components which do not currently have the capability of varying their energy consumption according to workload. As these components overcome this limitation, our energy-saving techniques can be extended to include them, leading to a more energy proportional system and lower overall operating costs.

**Applicability.** The target domain for EDT is primary storage systems where response time is critical. Archival applications where response time is not as critical may be better served with existing solutions using policy-based migration and power-saving storage such as spun-down disk or tape. Also, EDT will be most effective when the working set and I/O intensity are somewhat stable with some variation. When the workload is static, dynamic migration will not take place but consolidation will still be beneficial if the system is not capacity bound.

# 9 Conclusion

The increasing availability of solid-state drives has ushered in a new era of multi-tiered primary storage systems. With EDT, we have formalized the *configuration* and *dynamic tier management* problems and have systematically explored the design choices available when building such systems. We presented the design, implementation, and evaluation of EDT's Configuration Adviser (EDT-CA) and Dynamic Tier Manager (EDT-DTM). EDT lowers capital cost by configuring less expensive tiered storage and operating costs by dynamically optimizing power consumption via consolidation whenever feasible. We also demonstrated that EDT is successfully able to address the data migration overheads of dynamic tiering and respond rapidly and effectively to unexpected changes in the workload.

Experimental results show EDT has significant benefit. Evaluation performed using both a production workload and industry-standard synthetic workload revealed that multi-tier systems using EDT have a device mix that saves between 5% to 45% in cost, consume up to 54% less peak power, and an additional 15-30% lower dynamic power (instantaneous power averaged over time), at a better or comparable performance compared to a homogeneous SAS storage system. Experimental results also demonstrated that EDT is superior to simpler alternatives for extent-based tiering, providing lower cost and better performance, and consuming similar or lesser power. We hope that this study serves as a starting point for future work along the promising direction of multi-tiered enterprise storage systems.

# Acknowledgments

We would like to thank our shepherd Hakim Weatherspoon, our anonymous reviewers, and Renu Tewari, Aameek Singh and Amar Phanishayee for their valuable feedback. This work was supported in part by NSF grants CNS-0747038 and CNS-1018262. Jorge Guerra was supported in part by an IBM PhD Fellowship.

# References


[1] SPC specifications. http://www.storageperformance.org/specs.

[2] M. K. Aguilera, K. Keeton, A. Merchant, K.-K. Muniswamy-Reddy, and M. Uysal. Improving recoverability in multi-tier storage systems. In *Proc. of the IEEE/IFIP DSN*, 2007.

[3] G. A. Alvarez, E. Borowsky, S. Go, T. H. Romer, R. Becker-Szendy, R. Golding, A. Merchant, M. Spasojevic, A. Veitch, and J. Wilkes. Minerva: An Automated Resource Provisioning Tool for Large-scale Storage Systems. *ACM Transactions on Computer Systems*, 19(4):483–518, 2001.

[4] 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. *Lecture Notes in Computer Science*, 2141/2001: 145–158, 2001.

[5] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. Hippodrome: Running Circles Around Storage Administration. In *Proc. of USENIX FAST*, 2002.

[6] E. Anderson, S. Spence, R. Swaminathan, M. Kallahalla, and Q. Wang. Quickly finding near-optimal storage designs. *ACM Transactions on Computer Systems*, 23(4):337–374, 2005.

[7] M. Bhadkamkar, J. Guerra, L. Useche, S. Burnett, J. Liptak, R. Rangaswami, and V. Hristidis. BORG: Block-reORGanization for Self-Optimizing Storage Systems. *Proc. of USENIX FAST*, 2009.

[8] D. Colarelli and D. Grunwald. Massive Arrays of Idle Disks for Storage Archives. In *Proc. of IEEE/ACM SC*, 2002.

[9] K. Dasgupta, S. Ghosal, R. Jain, U. Sharma, and A. Verma. Qosmig: Adaptive rate-controlled migration of bulk data in storage systems. In *Proc. of ICDE*, 2005.

[10] K. M. Greenan, D. D. Long, E. L. Miller, T. J. Schwarz, and J. J. Wylie. A Spin-Up Saved is Energy Earned: Achieving Power-Efficient, Erasure-Coded Storage. In *Proc. of USENIX HotDep*, 2008.

[11] J. Guerra, H. Pucha, K. Gupta, W. Belluomini, and J. Glider. Energy Proportionality for Storage: Impact and Feasibility. In *Proc. of ACM/USENIX HotStorage*, 2009.

[12] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Frankez. DRPM: Dynamic speed control for power management in server class disks. In *Proc. of ACM/IEEE ISCA*, 2003.

[13] IBM Corporation. High Performance Storage System (HPSS). Online: http://hpss-collaboration.org/, 2010.

[14] IBM Corporation. IBM System Storage DS8000 series. Data Sheet, 2010.

[15] G. Karche, M. Mamidi, and P. Massiglia. Using dynamic storage tiering. Available as Symantec Yellow Books at http://www.symantec.com/enterprise/yellowbooks/index.jsp., 2006.

[16] R. Koller and R. Rangaswami. I/O Deduplication: Utilizing Content Similarity to Improve I/O Performance. In *Proc. of USENIX FAST*, 2010.

[17] B. Laliberte. Automate and Optimize a Tiered Storage Environment FAST! ESG White Paper, 2009.

[18] H. J. Lee, K. H. Lee, and S. H. Noh. Augmenting RAID with an SSD for Energy Relief. In *Proc. of USENIX HotPower*, 2008.

[19] A. Leung, S. Pasupathy, G. Goodson, and E. Miller. Measurement and Analysis of Large-Scale Network File System Workloads. In *Proc. of USENIX ATC*, 2008.

[20] D. Li and J. Wang. EERAID: Energy efficient redundant and inexpensive disk array. In *Proc. of workshop on ACM SIGOPS European workshop*, 2004.

[21] D. Narayanan, A. Donnelly, and A. Rowstron. Write Off-Loading: Practical Power Management for Enterprise Storage. In *Proc. of USENIX FAST*, 2008.

[22] D. Narayanan, E. Thereska, A. Donnelly, S. Elnikety, and A. Rowstron. Migrating Server Storage to SSDs: Analysis of Tradeoffs. In *Proc. of ACM Eurosys*, 2009.

[23] M. Peters. Compellent harnessing ssds potential. ESG Storage Systems Brief, 2009.

[24] M. Peters. Netapp's solid state hierarchy. ESG White Paper, 2009.

[25] M. Peters. 3par: Optimizing io service levels. ESG White Paper, 2010.

[26] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *Proc. of ACM ICS*, 2004.

[27] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. *SIGMETRICS*, 34(1), 2006.

[28] J. D. Strunk, E. Thereska, C. Faloutsos, and G. R. Ganger. Using utility to provision storage systems. In *Proc. of USENIX FAST*, 2008.

[29] Taneja Group Technology Analysts. The State of the Core Engineering the Enterprise Storage Infrastructure with the IBM DS8000. White Paper, 2010.

[30] M. Uysal, G. A. Alvarez, and A. Merchant. A modular, analytical throughput model for modern disk arrays. In *Proc. of IEEE MASCOTS*, 2001.

[31] A. Verma, R. Koller, L. Useche, and R. Rangaswami. SRCMap: Energy Proportional Storage Using Dynamic Consolidation. In *Proc. of USENIX FAST*, 2010.

[32] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. PARAID: A Gear-Shifting Power-Aware RAID. In *Proc. of USENIX FAST*, 2007.

[33] J. Wilkes, R. Golding, C. Staeliin, and T. Sullivan. The HP AutoRAID Hierarchical Storage System. In *Proc. of ACM SOSP*, 1995.

[34] X. Yao and J. Wang. RIMAC: a novel redundancy-based hierarchical cache architecture for energy efficient, high performance storage systems. *SIGOPS Operating Systems Review*, 40(4), 2006.

[35] M. Yue. A simple proof of the inequality ffd(l) (11/9)opt(l) + 1, for all l, for the ffd bin-packing algorithm. *Acta Mathematicae Applicatae Sinica*, 7:321331, 1991.

[36] G. Zhang, L. Chiu, C. Dickey, L. Liu, P. Muench, and S. Seshadri. Automated Lookahead Data Migration in SSD-enabled Multi-tiered Storage Systems. In *IEEE MSST*, 2010.

[37] Q. Zhu, F. M. David, C. F. Devaraj, Z. Li, Y. Zhou, and P. Cao. Reducing Energy Consumption of Disk Storage Using Power-Aware Cache Management. In *Proc. of IEEE HPCA*, 2004.

[38] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes. Hibernator: helping disk arrays sleep through the winter. In *Proc. of ACM SOSP*, 2005.