SRCMap: Energy Proportional Storage using Dynamic Consolidation

Akshat Verma[†] Ricardo Koller[‡] Luis Useche[‡] Raju Rangaswami[‡]
[†]*IBM Research, India*
akshatverma@in.ibm.com
[‡]*Florida International University*
{rkoll001,luis,raju}@cs.fiu.edu

Abstract

We investigate the problem of creating an energy proportional storage system through power-aware dynamic storage consolidation. Our proposal, Sample-Replicate-Consolidate Mapping (SRCMap), is a storage virtualization layer optimization that enables energy proportionality for dynamic I/O workloads by consolidating the cumulative workload on a subset of physical volumes proportional to the I/O workload intensity. Instead of migrating data across physical volumes dynamically or replicating entire volumes, both of which are prohibitively expensive, SRCMap samples a subset of blocks from each data volume that constitutes its working set and replicates these on other physical volumes. During a given consolidation interval, SRCMap activates a minimal set of physical volumes to serve the workload and spins down the remaining volumes, redirecting their workload to replicas on active volumes. We present both theoretical and experimental evidence to establish the effectiveness of SRCMap in minimizing the power consumption of enterprise storage systems.

1 Introduction

Energy Management has emerged as one of the most significant challenges faced by data center operators. The current power density of data centers is estimated to be in the range of 100 W/sq.ft. and growing at the rate of 15-20% per year [22]. Barroso and Hölzle have made the case for energy proportional computing based on the observation that servers in data centers today operate at well below peak load levels on an average [2]. A popular technique for delivering energy proportional behavior in servers is consolidation using virtualization [4, 24, 26, 27]. These techniques (a) utilize heterogeneity to select the most power-efficient servers at any given time, (b) utilize low-overhead live Virtual Machine (VM) migration to vary the number of active servers in response to workload variation, and (c) provide fine-grained control over power consumption by allowing the number of active servers to be increased or decreased one at a time.

Storage consumes roughly 10-25% of the power within computing equipment at data centers depending on the load level, consuming a greater fraction of the power when server load is lower [3]. Energy proportionality for the storage subsystem thus represents a critical gap in the energy efficiency of future data centers. In this paper, we the investigate the following fundamental question: *Can we use a storage virtualization layer to design a practical energy proportional storage system?*

Storage virtualization solutions (e.g., EMC Invista [7], HP SVSP [6], IBM SVC [12], NetApp V-Series [19]) provide a unified view of disparate storage controllers thus simplifying management [13]. Similar to server virtualization, storage virtualization provides a transparent I/O redirection layer that can be used to consolidate fragmented storage resource utilization. Similar to server workloads, storage workloads exhibit significant variation in workload intensity, motivating dynamic consolidation [16]. However, unlike the relatively inexpensive VM migration, migrating a logical volume from one device to another can be prohibitively expensive, a key factor disrupting storage consolidation solutions.

Our proposal, Sample-Replicate-Consolidate Mapping (SRCMap), is a storage virtualization layer optimization that makes storage systems energy proportional. The SRCMap architecture leverages storage virtualization to redirect the I/O workload without any changes in the hosts or storage controllers. SRCMap ties together disparate ideas from server and storage power management (namely caching, replication, transparent live migration, and write off-loading) to minimize the power drawn by storage devices in a data center. It continuously targets energy proportionality by dynamically increasing or decreasing the number of active physical volumes in a data center in response to variation in I/O workload intensity.

SRCMap is based on the following observations in production workloads detailed in §3: *(i)* the active data set in storage volumes is small, *(ii)* this active data set is stable, and *(iii)* there is substantial variation in workload intensity both within and across storage volumes.

Thus, instead of creating full replicas of data volumes, SRCMap creates partial replicas that contain the working sets of data volumes. The small replica size allows creating multiple copies on one or more target volumes or analogously allowing one target volume to host replicas of multiple source volumes. Additional space is reserved on each partial replica to offload writes [18] to volumes that are spun down.

SRCMap enables a high degree of flexibility in spinning down volumes because it activates either the primary volume or exactly one working set replica of each volume at any time. Based on the aggregate workload intensity, SRCMap changes the set of active volumes in the granularity of hours rather than minutes to address the reliability concerns related to the limited number of disk spin-up cycles. It selects active replica targets that allow spinning down the maximum number of volumes, while serving the aggregate storage workload. The virtualization layer remaps the virtual to physical volume mapping as required thereby replacing expensive data migration operations with background data synchronization operations. SRCMap is able to create close to N *power-performance levels on a storage subsystem with* N *volumes*, enabling storage energy consumption proportional to the I/O workload intensity.

In the rest of this paper, we propose design goals for energy proportional storage systems and examine existing solutions (§2), analyze storage workload characteristics (§3) that motivate design choices (§4), provide detailed system design, algorithms, and optimizations (§5 and §6), and evaluate for energy proportionality (§7). We conclude with a fairly positive view on SRCMap meeting its energy proportionality goals and some directions for future work (§8).

2 On Energy Proportional Storage

In this section, we identify the goals for a practical and effective energy proportional storage system. We also examine existing work on energy-aware storage and the extent to which they deliver on these goals.

2.1 Design Goals

1. Fine-grained energy proportionality: Energy proportional storage systems are uniquely characterized by multiple performance-power levels. True energy proportionality requires that for a system with a peak power of P_{peak} for a workload intensity ρ_{max}, the power drawn for a workload intensity ρ_i would be $P_{peak} \times \frac{\rho_i}{\rho_{max}}$.

2. Low space overhead: Replication-based strategies could achieve energy proportionality trivially by replicating each volume on all the other $N-1$ volumes. This would require N copies of each volume, representing an unacceptable space overhead. A practical energy proportional system should incur minimum space overhead; for example, 25% additional space is often available.

Design Goal	Write offloading	Caching systems	Singly Redundant	Geared RAID
Proportionality	$\sim$	X	X	$\sim$
Space overhead	✓	✓	X	X
Reliability	X	X	✓	✓
Adaptation	X	✓	✓	✓
Heterogeneity	$\sim$	$\sim$	$\sim$	X

Table 1: **Comparison of Power Management Techniques.** $\sim$ *indicates the goal is partially addressed.*

3. Reliability: Disk drives are designed to survive a limited number of spin-up cycles [14]. Energy conservation based on spinning down the disk must ensure that the additional number of spin-up cycles induced during the disks' expected lifetime is significantly lesser than the manufacturer specified maximum spin-up cycles.

4. Workload shift adaptation: The popularity of data changes, even if slowly over time. Power management for storage systems that rely on caching popular data over long intervals should address any shift in popularity, while ensuring energy proportionality.

5. Heterogeneity support: A data center is typically composed of several substantially different storage systems (e.g., with variable numbers and types of drives). An ideal energy proportional storage system should account for the differences in their performance-power ratios to provide the best performance at each host level.

2.2 Examining Existing Solutions

It has been shown that the idleness in storage workload is quite low for typical server workloads [31]. We examine several classes of related work that represent approaches to increase this idleness for power minimization and evaluate the extent to which they address our design goals. We next discuss each of them and summarize their relative strengths in Table 1.

Singly redundant schemes. The central idea used by these schemes is spinning down disks with redundant data during periods of low I/O load [9, 21, 28]. RIMAC [28] uses memory-level and on-disk redundancy to reduce passive spin ups in RAID5 systems, enabling the spinning down of one out of the N disks in the array. The Diverted Accesses technique [21] generalizes this approach to find the best redundancy configuration for energy, performance, and reliability for all RAID levels. Greenan *et al.* propose generic techniques for managing power-aware erasure coded storage systems [9]. The above techniques aim to support two energy levels and do not address fine-grained energy proportionality.

Geared RAIDs. PARAID [30] is a gear-shifting mechanism (each disk spun down represents a gear shift) for a parity-based RAID. To implement $N-1$ gears in a N disk array with used storage X, PARAID requires

$O(X \log N)$ space, even if we ignore the space required for storing parity information. DiskGroup [17] is a modification of RAID-1 that enables a subset of the disks in a mirror group to be activated as necessary. Both techniques incur large space overhead. Further, they do not address heterogeneous storage systems composed of multiple volumes with varying I/O workload intensities.

Caching systems. This class of work is mostly based on caching popular data on additional storage [5, 15, 25] to spin down primary data drives. MAID [5], an archival storage system, optionally uses additional cache disks for replicating popular data to increase idle periods on the remaining disks. PDC [20] does not use additional disks but rather suggests migrating data between disks according to popularity, always keeping the most popular data on a few active disks. EXCES [25] uses a low-end flash device for caching popular data and buffering writes to increase idle periods of disk drives. Lee *et al.* [15] suggest augmenting RAID systems with an SSD for a similar purpose. A dedicated storage cache does not provide fine-grained energy proportionality; the storage system is able to save energy only when the I/O load is low and can be served from the cache. Further, these techniques do not account for the reliability impact of frequent disk spin-up operations.

Write Offloading. Write off-loading is an energy saving technique based on redirecting writes to alternate locations. The authors of write-offloading demonstrate that idle periods at a one minute granularity can be significantly increased by off-loading writes to a different volume. The reliability impact due to frequent spin-up cycles on a disk is a potential concern, which the authors acknowledge but leave as an open problem. In contrast, SRCMap increases the idle periods substantially by off-loading popular data reads in addition to the writes, and thus more comprehensively addressing this important concern. Another important question not addressed in the write off-loading work is: with multiple volumes, which active volume should be treated as a write off-loading target for each spun down volume? SRCMap addresses this question clearly with a formal process for identifying the set of active disks during each interval.

Other techniques. There are orthogonal classes of work that can either be used in conjunction with SRCMap or that address other target environments. Hibernator [31] uses DRPM [10] to create a multi-tier hierarchy of futuristic multi-speed disks. The speed for each disk is set and data migrated across tiers as the workload changes. Pergamum is an archival storage system designed to be energy-efficient with techniques for reducing inter-disk dependencies and staggering rebuild operations [23]. Gurumurthi *et al.* propose intra-disk parallelism on high capacity drives to improve disk bandwidth without increasing power consumption [11]. Finally, Ganesh *et al.* propose log-structured striped writing on a disk array to increase the predictability of active/inactive spindles [8].

Workload Volume	Size [GB]	Reads [GB] Total	Reads [GB] Uniq	Writes [GB] Total	Writes [GB] Uniq	Volume accessed
mail	500	62.00	29.24	482.10	4.18	6.27%
homes	470	5.79	2.40	148.86	4.33	1.44%
web-vm	70	3.40	1.27	11.46	0.86	2.8%

Table 2: **Summary statistics of one week I/O workload traces obtained from three different volumes.**

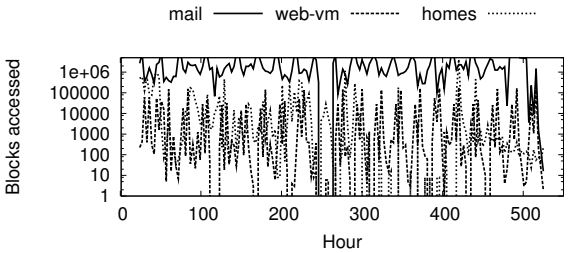

Figure 1: **Variability in I/O workload intensity.**

3 Storage Workload Characteristics

In this section, we characterize the nature of I/O access on servers using workloads from three production systems, specifically looking for properties that help us in our goal of energy proportional storage. The systems include an email server (*mail* workload), a virtual machine monitor running two web servers (*web-vm* workload), and a file server (*homes* workload). The *mail* workload serves user INBOXes for the entire Computer Science department at FIU. The *homes* workload is that of a NFS server that serves the home directories for our research group at FIU; activities represent those of a typical researcher consisting of software development, testing, and experimentation, the use of graph-plotting software, and technical document preparation. Finally, the *web-vm* workload is collected from a virtualized system that hosts two CS department web-servers, one hosting the department's online course management system and the other hosting the department's web-based email access portal.

In each system, we collected I/O traces downstream of an active page cache for a duration of three weeks. Average weekly statistics related to these workloads are summarized in Table 2. The first thing to note is that the weekly working sets (unique accesses during a week) is a small percentage of the total volume size (1.5-6.5%). This trend is consistent across all volumes and leads to our first observation.

Observation 1 *The active data set for storage volumes is typically a small fraction of total used storage.*

Dynamic consolidation utilizes variability in I/O workload intensity to increase or decrease the number of

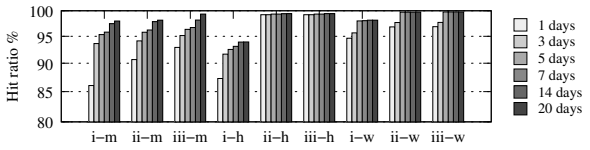

Figure 2: **Overlap in daily working sets for the mail (m), homes (h), and web-vm (w) workloads.** *(i) Reads and writes against working set , (ii) Reads against working set and (iii) Reads against working set, recently offloaded writes, and recent missed reads.*

active devices. Figure 1 depicts large variability in I/O workload intensity for each of the three workloads over time, with as much as 5-6 orders of magnitude between the lowest and highest workload intensity levels across time. This highlights the potential of energy savings if the storage systems can be made energy proportional.

Observation 2 *There is a significant variability in I/O workload intensity on storage volumes.*

Based on our first two observations, we hypothesize that there is room for powering down physical volumes that are substantially under-utilized by replicating a small active working-set on other volumes which have the spare bandwidth to serve accesses to the powered down volumes. This motivates *Sample* and *Replicate* in SRCMap. Energy conservation is possible provided the corresponding working set replicas can serve most requests to each powered down volume. This would be true if working sets are largely stable.

We investigate the stability of the volume working sets in Fig. 2 for three progressive definitions of the working set. In the first scenario, we compute the classical working set based on the last few days of access history. In the second scenario, we additionally assume that writes can be offloaded and mark all writes as hits. In the third scenario, we further expand the working set to include recent writes and past missed reads. For each scenario, we compute the working set hits and misses for the following day's workload and study the hit ratio with change in the length of history used to compute the working set. We observe that the hit ratio progressively increases both across the scenarios and as we increase the history length leading us to conclude that data usage exhibits high temporal locality and that the working set after including recent accesses is fairly stable. This leads to our third observation (also observed earlier by Leung *et al.* [16]).

Observation 3 *Data usage is highly skewed with more than* 99% *of the working set consisting of some 'really popular' data and 'recently accessed' data.*

The first three observations are the pillars behind the *Sample*, *Replicate* and *Consolidate* approach whereby we sample each volume for its working set, replicate

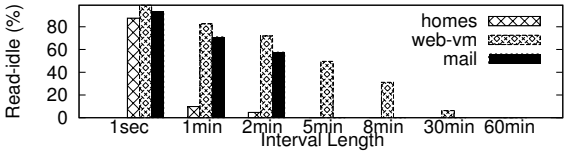

Figure 3: **Distribution of read-idle times.**

these working sets on other volumes, and consolidate I/O workloads on proportionately fewer volumes during periods of low load. Before designing a new system based on the above observations, we study the suitability of a simpler *write-offloading* technique for building energy proportional storage systems. Write off-loading is based on the observation that I/O workloads are write dominated and simply off-loading writes to a different volume can cause volumes to be idle for a substantial fraction (79% for workloads in the original study) of time [18]. While write off-loading increases the fraction of idle time of volumes, the distribution of idle time durations due to write off-loading raises an orthogonal, but important, concern. If these idle time durations are short, saving power requires frequent spinning down/up of the volumes which degrades reliability of the disk drives.

Figure 3 depicts the read-idle time distributions of the three workloads. It is interesting to note that idle time durations for the *homes* and *mail* workloads are all less than or equal to 2 minutes, and for the *web-vm* the majority are less than or equal to 5 minutes are all are less than 30 minutes.

Observation 4 *The read-idle time distribution (periods of writes alone with no intervening read operations) of I/O workloads is dominated by small durations, typically less than five minutes.*

This observation implies that exploiting all read-idleness for saving power will necessitate spinning up the disk at least 720 times a day in the case of *homes* and *mail* and at least 48 times in the case of *web-vm*. This can be a significant hurdle to reliability of the disk drives which typically have limited spin-up cycles [14]. It is therefore important to develop new techniques that can substantially increase average read-idle time durations.

4 Background and Rationale

Storage virtualization managers simplify storage management by enabling a uniform view of disparate storage resources in a data center. They export a storage controller interface allowing users to create logical volumes or virtual disks (*vdisks*) and mount these on hosts. The physical volumes managed by the physical storage controllers are available to the virtualization manager as managed disks (*mdisks*) entirely transparently to the

hosts which only view the logical *vdisk* volumes. A useful property of the virtualization layer is the complete flexibility in allocation of $mdisk$ extents to $vdisk$s.

Applying server consolidation principles to storage consolidation using virtualization would activate only the most energy-efficient $mdisk$s required to serve the aggregate workload during any period T. Data from the other $mdisk$s chosen to be spun down would first need to be migrated to active $mdisk$s to effect the change. While data migration is an expensive operation, the ease with which virtual-to-physical mappings can be reconfigured provides an alternative approach. A naïve strategy following this approach could replicate data for each $vdisk$ on all the $mdisk$s and adapt to workload variations by dynamically changing the virtual-to-physical mappings to use only the selected $mdisk$s during T. Unfortunately, this strategy requires N times additional space for a N $vdisk$ storage system, an unacceptable space overhead.

SRCMap intelligently uses the storage virtualization layer as an I/O indirection mechanism to deliver a practically feasible, energy proportional solution. Since it operates at the storage virtualization manager, it does not alter the basic redundancy-based reliability properties of the underlying physical volumes which is determined by the respective physical volume (e.g., RAID) controllers. To maintain the redundancy level, SRCMap ensures that a volume is replicated on target volumes at the same RAID level. While we detail SRCMap's design and algorithms in subsequent sections (§ 5 and § 6), here we list the rationale behind SRCMap's design decisions. These design decisions together help to satisfy the design goals for an ideal energy proportional storage system.

I. Multiple replica targets. Fine-grained energy proportionality requires the flexibility to increase or decrease the number of active physical volumes one at a time. Techniques that activate a fixed secondary device for each data volume during periods of low activity cannot provide the flexibility necessary to deactivate an arbitrary fraction of the physical volumes. In SRCMap, we achieve this fine-grained control by creating a primary $mdisk$ for each $vdisk$ and replicating only the working set of each $vdisk$ on multiple secondary $mdisk$s. This ensures that *(a)* every volume can be offloaded to one of multiple targets and *(b)* each target can serve the I/O workload for multiple $vdisk$s. During peak load, each $vdisk$ maps to its primary $mdisk$ and all $mdisk$s are active. However, during periods of low activity, SRCMap selects a proportionately small subset of $mdisk$s that can support the aggregate I/O workload for all $vdisk$s.

II. Sampling. Creating multiple full replicas of $vdisk$s is impractical. Drawing from *Observation 1* (§ 3), SRCMap substantially reduces the space overhead of maintaining multiple replicas by sampling only the working set for each $vdisk$ and replicating it. Since the working set is typically small , the space overhead is low.

III. Ordered replica placement. While sampling helps to reduce replica sizes substantially, creating multiple replicas for each sample still induces space overhead. In SRCMap, we observe that all replicas are not created equal; for instance, it is more beneficial to replicate a lightly loaded volume than a heavily loaded one which is likely to be active anyway. Similarly, a large working set has greater space overhead; SRCMap chooses to create fewer replicas aiming to keep it active, if possible. As we shall formally demonstrate, carefully ordering the replica placement helps to minimize the number of active disks for fine-grained energy proportionality.

IV. Dynamic source-to-target mapping and dual data synchronization. From *Observation 2* (§ 3), we know that workloads can vary substantially over a period of time. Hence, it is not possible to pre-determine which volumes need to be active. Target replica selection for any volume being powered down therefore needs to be a dynamic decision and also needs to take into account that some volumes have more replicas (or target choices) than others. We use two distinct mechanisms for updating the replica working sets. The active replica lies in the data path and is immediately synchronized in the case of a *read miss*. This ensures that the active replica continuously *adapts* with change in *workload popularity*. The secondary replicas, on the other hand, use a lazy, incremental data synchronization in the background between the primary replica and any secondary replicas present on active $mdisk$s. This ensures that switching between replicas requires minimal data copying and can be performed fairly quickly.

V. Coarse-grained power cycling. In contrast to most existing solutions that rely on fine-grained disk power-mode switching, SRCMap implements coarse-grained consolidation intervals (of the order of hours), during each of which the set of active $mdisk$s chosen by SRCMap does not change. This ensures normal disk lifetimes are realized by adhering to the disk power cycle specification contained within manufacturer data sheets.

5 Design Overview

SRCMap is built in a modular fashion to directly interface with storage virtualization managers or be integrated into one as shown in Figure 4. The overall architecture supports the following distinct flows of control:

(i) the *replica generation flow* (Flow A) identifies the working set for each $vdisk$ and replicates it on multiple $mdisk$s. This flow is orchestrated by the *Replica Placement Controller* and is triggered once when SRCMap

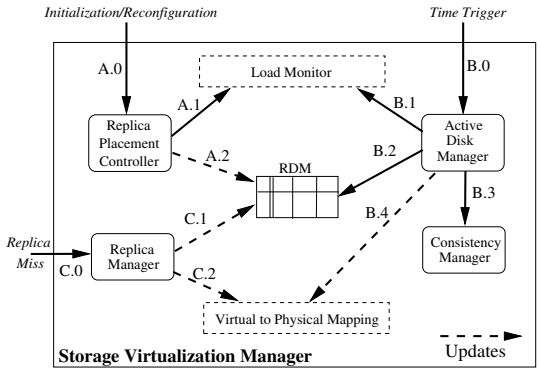

Figure 4: **SRCMap integrated into a Storage Virtualization Manager.** *Arrows depict control flow. Dashed/solid boxes denote existing/new components.*

is initialized and whenever a configuration change (e.g., addition of a new workload or new disks) takes place. Once a trigger is generated, the *Replica Placement Controller* obtains a historical workload trace from the *Load Monitor* and computes the working set and the long-term workload intensity for each volume ($vdisk$). The working set is then replicated on one or more physical volumes ($mdisks$). The blocks that constitute the working set for the $vdisk$ and the target physical volumes where these are replicated are managed using a common data structure called the *Replica Disk Map (RDM)*.

(ii) the *active disk identification flow* (Flow B) identifies, for a period T, the active $mdisk$s and activated replicas for each inactive $mdisk$. The flow is triggered at the beginning of the consolidation interval T (e.g., every 2 hours) and orchestrated by the *Active Disk Manager*. In this flow, the *Active Disk Manager* queries the *Load Monitor* for expected workload intensity of each $vdisk$ in the period T. It then uses the workload information along with the placement of working set replicas on target $mdisks$ to compute the set of active primary $mdisk$s and a active secondary replica $mdisk$ for each inactive primary $mdisk$. It then directs the *Consistency Manager* to ensure that the data on any selected active primary or active secondary replica is current. Once consistency checks are made, it updates the *Virtual to Physical Mapping* to redirect the workload to the appropriate $mdisk$.

(iii) the *I/O redirection flow* (Flow C) is an extension of the I/O processing in the *storage virtualization manager* and utilizes the built-in virtual-to-physical re-mapping support to direct requests to primaries or active replicas. Further, this flow ensures that the working-set of each $vdisk$ is kept up-to-date. To ensure this, whenever a request to a block not available in the active replica is made, a *Replica Miss* event is generated. On a *Replica Miss*, the *Replica Manager* spin-ups the primary $mdisk$ to fetch the required block. Further, it adds this new block to the working set of the $vdisk$ in the RDM. We next describe the key components of SRCMap.

5.1 Load Monitor

The *Load Monitor* resides in the storage virtualization manager and records access to data on any of the $vdisk$s exported by the virtualization layer. It provides two interfaces for use by SRCMap – long-term workload data interface invoked by the *Replica Placement Controller* and predicted short-term workload data interface invoked by the *Active Disk Manager*.

5.2 Replica Placement Controller

The *Replica Placement Controller* orchestrates the process of *Sampling* (identifying working sets for each $vdisk$) and *Replicating* on one or more target $mdisk$s. We use a conservative definition of working set that includes all the blocks that were accessed during a fixed duration, configured as the minimum duration beyond which the hit ratio on the working set saturates. Consequently, we use 20 days for $mail$, 14 days for $homes$ and 5 days for $web\text{-}vm$ workload (Fig. 2). The blocks that capture the working set for each $vdisk$ and the $mdisks$ where it is replicated are stored in the RDM. The details of the parameters and methodology used within *Replica Placement* are described in Section 6.1.

5.3 Active Disk Manager

The *Active Disk Manager* orchestrates the *Consolidate* step in SRCMap. The module takes as input the workload intensity for each $vdisk$ and identifies if the primary $mdisk$ can be spun down by redirecting the workload to one of the secondary $mdisk$s hosting its replica. Once the target set of active $mdisk$s and replicas are identified, the *Active Disk Manager* synchronizes the identified active primaries or active secondary replicas and updates the virtual-to-physical mapping of the storage virtualization manager, so that I/O requests to a $vdisk$ could be redirected accordingly. The Active Disk Manager uses a *Consistency Manager* for the synchronization operation. Details of the algorithm used by *Active Disk Manager* for selecting active $mdisks$ are described in Section 6.2.

5.4 Consistency Manager

The *Consistency Manager* ensures that the primary $mdisk$ and the replicas are consistent. Before an $mdisk$ is spun down and a new replica activated, the new active replica is made consistent with the previous one. In order to ensure that the overhead during the re-synchronization is minimal, an incremental point-in-time (PIT) relationship (e.g., Flash-copy in IBM SVC [12]) is maintained between the active data (either the primary $mdisk$ or one of the active replicas) and all other copies of the same data. A *go-to-sync* operation is performed periodically between the active data and all its copies on active $mdisk$s. This ensures that when an $mdisk$ is spun up or down, the amount of data to be synchronized is small.

5.5 Replica Manager

The *Replica Manager* ensures that the replica data set for a $vdisk$ is able to mimic the working set of the $vdisk$ over time. If a data block unavailable at the active replica of the $vdisk$ is read causing a *replica miss*, the Replica Manager copies the block to the replica space assigned to the active replica and adds the block to the *Replica Metadata* accordingly. Finally, the *Replica Manager* uses a Least Recently Used (LRU) policy to evict an older block in case the replica space assigned to a replica is filled up. If the active data set changes drastically, there may be a large number of *replica misses*. All these replica misses can be handled by a single spin-up of the primary $mdisk$. Once all the data in the new working set is touched, the primary $mdisk$ can be spun-down as the active replica is now up-to-date. The continuous updating of the *Replica Metadata* enables SRCMap to meet the goal of *Workload shift adaptation*, without re-running the expensive *replica generation flow*. The *replica generation flow* needs to re-run only when a disruptive change occurs such as addition of a new workload or a new volume or new disks to a volume.

6 Algorithms and Optimizations

In this section, we present details about the algorithms employed by SRCMap. We first present the long-term replica placement methodology and subsequently, the short-term active disk identification method.

6.1 Replica Placement Algorithm

The *Replica Placement Controller* creates one or more replicas of the working set of each $vdisk$ on the available replica space on the target $mdisk$s. We use the insight that all replicas are not created equal and have distinct associated costs and benefits. The space cost of creating the replica is lower if the $vdisk$ has a smaller working set. Similarly, the benefit of creating a replica is higher if the $vdisk$ (i) has a stable working set (lower misses if the primary $mdisk$ is switched off), (ii) has a small average load making it easy to find spare bandwidth for it on any target $mdisk$, and (iii) is hosted on a less power-efficient primary $mdisk$. Hence, the goal of both *Replica Placement* and *Active Disk Identification* is to ensure that we create more replicas for $vdisk$s that have a favorable cost-benefit ratio. The goal of the replica placement is to ensure that if the *Active Disk Manager* decides to spin down the primary $mdisk$ of a $vdisk$, it should be able to find at least one active target $mdisk$ that hosts its replica, captured in the following *Ordering Property*.

Definition 1 Ordering Property*: For any two $vdisks$ V_i and V_j, if V_i is more likely to require a replica target than V_j at any time t during* Active Disk Identification*, then V_i is more likely than V_j to find a replica target amongst active $mdisks$ at time t.*

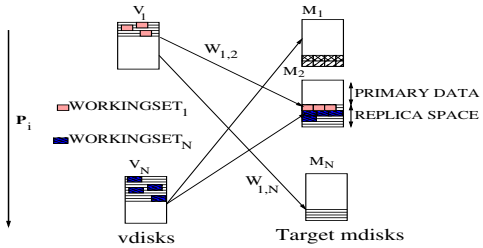

Figure 5: **Replica Placement Model**

The *replica placement algorithm* consists of (i) creating an initial ordering of $vdisk$s in terms of cost-benefit tradeoff (ii) a bipartite graph creation that reflects this ordering (iii) iteratively creating one source-target mapping respecting the current order and (iv) re-calibration of edge weights to ensure the *Ordering Property* holds for the next iteration of source-target mapping.

6.1.1 Initial $vdisk$ ordering

The Initial $vdisk$ ordering creates a sorted order amongst $vdisk$s based on their cost-benefit tradeoff. For each $vdisk$ V_i, we compute the probability P_i that its primary $mdisk$ M_i would be spun down as

$$P_i = \frac{w_1 WS_{min}}{WS_i} + \frac{w_2 PPR_{min}}{PPR_i} + \frac{w_3 \rho_{min}}{\rho_i} + \frac{w_f m_{min}}{m_i} \quad (1)$$

where the w_k are tunable weights, WS_i is the size of the working set of V_i, PPR_i is the performance-power ratio (ratio between the peak IO bandwidth and peak power) for the primary mdisk M_i of V_i, ρ_i is the average long-term I/O workload intensity (measured in IOPS) for V_i, and m_i is the number of read misses in the working set of V_i, normalized by the number of spindles used by its primary mdisk M_i. The corresponding min subscript terms represent the minimum values across all the $vdisk$s and provide normalization. The probability formulation is based on the dual rationale that it is relatively easier to find a target $mdisk$ for a smaller workload and switching off relatively more power-hungry disks saves more power. Further, we assign a higher probability for spinning down $mdisk$s that host more stable working sets by accounting for the number of times a read request cannot be served from the replicated working set, thereby necessitating the spinning up of the primary $mdisk$.

6.1.2 Bipartite graph creation

Replica Placement creates a bipartite graph $G(V \rightarrow M)$ with each $vdisk$ as a source node V_i, its primary $mdisk$ as a target node M_i, and the edge weights $e(V_i, M_j)$ representing the cost-benefit trade-off of placing a replica of V_i on M_j (Fig. 5). The nodes in the bipartite graph are sorted using P_i (disks with larger P_i are at the top). We initialize the edge weights $w_{i,j} = P_i$ for each edge $e(V_i, M_j)$ (source-target pair). Initially, there are no

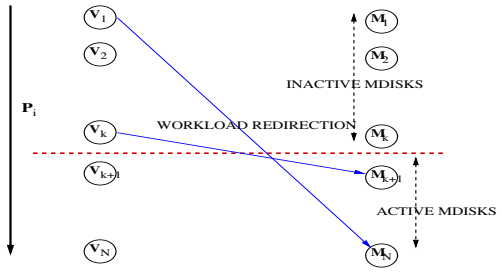

Figure 6: **Active Disk Identification**

replica assignments made to any target $mdisk$. The replica placement algorithm iterates through the following two steps, until all the available replica space on the target $mdisk$s have been assigned to source $vdisk$ replicas. In each iteration, exactly one target $mdisk$'s replica space is assigned.

6.1.3 Source-Target mapping

The goal of the replica placement method is to achieve a source target mapping that achieves the *Ordering property*. To achieve this goal, the algorithm takes the top-most target $mdisk$ M_i whose replica space is not yet assigned and selects the set of highest weight incident edges such that the combined replica size of the source nodes in this set fills up the replica space available in M_i (e.g, the working sets of V_1 and V_N are replicated in the replica space of M_2 in Fig. 5). When the replica space on a target $mdisk$ is filled up, we mark the target $mdisk$ as assigned. One may observe that this procedure always gives preference to source nodes with a larger P_i. Once an $mdisk$ finds a replica, the likelihood of it requiring another replica decreases and we factor this using a re-calibration of edge weights, which is detailed next.

6.1.4 Re-calibration of edge weights

We observe that the initial assignments of weights ensure the *Ordering property*. However, once the working set of a $vdisk$ V_i has been replicated on a set of target $mdisk$s $\mathbf{T_i} = M_1, \ldots, M_{least}$ (M_{least} is the $mdisk$ with the least P_i in $\mathbf{T_i}$) s.t. $P_i > P_{least}$, the probability that V_i would require a new target $mdisk$ during *Active Disk Identification* is the probability that both M_i and M_{least} would be spun down. Hence, to preserve the *Ordering property*, we re-calibrate the edge weights of all outgoing edges of any primary $mdisk$s S_i assigned to target mdisks T_j as

$$\forall k \quad w_{i,k} = P_j P_i \tag{2}$$

Once the weights are recomputed, we iterate from the Source-Target mapping step until all the replicas have been assigned to target $mdisk$s. One may observe that the re-calibration succeeds in achieving the *Ordering* property because we start assigning the replica space for the top-most target $mdisk$s first. This allows us to increase the weights of source nodes monotonically as we place more replicas of its working set. We formally prove the following result in the appendix.

S = set of disks to be spun down
A = set of disks to be active
Sort S by reverse of P_i
Sort A by P_i
For each $D_i \in S$
  For each $D_j \in A$
    If D_j hosts a replica R_i of D_i AND
    D_j has spare bandwidth for R_i
      $Candidate(D_i) = D_j,\ break$
  End-For
  If $Candidate(D_i)$==$null$ return $Failure$
End-for
$\forall i,\ D_i \in S$ return $Candidate(D_i)$

Figure 7: **Active Replica Identification algorithm**

Theorem 1 *The* Replica Placement Algorithm *ensures ordering property.*

6.2 Active Disk Identification

We now describe the methodology employed to identify the set of active $mdisk$s and replicas at any given time. For ease of exposition, we define the probability P_i of a primary $mdisk$ M_i equal to the probability P_i of its $vdisk$ V_i. Active disk identification consists of:

I: Active $mdisk$ Selection: We first estimate the expected aggregate workload to the storage subsystem in the next interval. We use the workload to a $vdisk$ in the previous interval as the predicted workload in the next interval for the $vdisk$. The aggregate workload is then estimated as sum of the predicted workloads for all $vdisk$s in the storage system. This aggregate workload is then used to identify the minimum subset of $mdisk$s (ordered by reverse of P_i) such that the aggregate bandwidth of these $mdisk$s exceeds the expected aggregate load.
II: Active Replica Identification: This step elaborated shortly identifies one (of the many possible) replicas on an active $mdisk$ for each inactive $mdisk$ to serve the workload redirected from the inactive $mdisk$.
III: Iterate: If the Active Replica Identification step succeeds in finding an active replica for all the inactive $mdisk$s, the algorithm terminates. Else, the number of active $mdisk$s are increased by 1 and the algorithm repeats the Active Replica Identification step.

One may note that since the number of active disks are based on the maximum predicted load in a consolidation interval, a sudden increase in load may lead to an increase in response times. If performance degradation beyond user-defined acceptable levels persists beyond a user-defined interval (e.g, 5 mins), the *Active Disk Identification* is repeated for the new load.

6.2.1 Active Replica Identification

Fig. 6 depicts the high-level goal of Active Replica Identification, which is to have the primary $mdisk$s for

$vdisk$s with larger P_i spun down, and their workload directed to few $mdisk$s with smaller P_i. To do so, it must identify an active replica for each inactive primary $mdisk$, on one of the active $mdisk$s. The algorithm uses two insights: (i) The *Replica Placement* process creates more replicas for $vdisk$s with a higher probability of being spun down (P_i) and (ii) primary $mdisk$s with larger P_i are likely to be spun down for a longer time.

To utilize the first insight, we first allow primary $mdisk$s with small P_i, which are marked as inactive, to find an active replica, as they have fewer choices available. To utilize the second insight, we force inactive primary $mdisk$s with large P_i to use a replica on active $mdisk$s with small P_i. For example in Fig. 6, $vdisk$ V_k has the first choice of finding an active $mdisk$ that hosts its replica and in this case, it is able to select the first active $mdisk$ M_{k+1}. As a result, inactive $mdisk$s with larger P_i are mapped to active $mdisk$s with the smaller P_i (e.g, V_1 is mapped to M_N). Since an $mdisk$ with the smallest P_i is likely to remain active most of the time, this ensures that there is little to no need to 'switch active replicas' frequently for the inactive disks. The details of this methodology are described in Fig. 7.

6.3 Key Optimizations to Basic SRCMap

We augment the basic SRCMap algorithm to increase its practical usability and effectiveness as follows.

6.3.1 Sub-volume creation

SRCMap redirects the workload for any primary $mdisk$ that is spun down to exactly one target $mdisk$. Hence, a target $mdisk$ M_j for a primary $mdisk$ M_i needs to support the combined load of the vdisks V_i and V_j in order to be selected. With this requirement, the SRCMap consolidation process may incur a fragmentation of the available I/O bandwidth across all volumes. To elaborate, consider an example scenario with 10 identical $mdisk$s, each with capacity C and input load of $C/2 + \delta$. Note that even though this load can be served using $10/2 + 1$ $mdisk$s, there is no single $mdisk$ can support the input load of 2 $vdisk$s. To avoid such a scenario, SRCMap sub-divides each $mdisk$ into N_{SV} sub-volumes and identifies the working set for each sub-volume separately. The sub-replicas (working sets of a sub-volume) are then placed independently of each other on target $mdisk$s. With this optimization, SRCMap is able to subdivide the least amount of load that can be migrated, thereby dealing with the fragmentation problem in a straightforward manner.

This optimization requires a complementary modification to the*Replica Placement* algorithm. The Source-Target mapping step is modified to ensure that sub-replicas belonging to the same source $vdisk$ are not co-located on a target $mdisk$.

6.3.2 Scratch Space for Writes and Missed Reads

SRCMap incorporates the basic write off-loading mechanism as proposed by Narayanan *et al.* [18]. The current implementation of SRCMap uses an additional allocation of write scratch space with each sub-replica to absorb new writes to the corresponding portion of the data volume. A future optimization is to use a single write scratch space within each target $mdisk$ rather than one per sub-replica within the target $mdisk$ so that the overhead for absorbing writes can be minimized.

A key difference from write off-loading, however, is that on a *read miss* for a spun down volume, SRCMap additionally offloads the data read to dynamically learn the working-set. This helps SRCMap achieve the goal of *Workload Shift Adaptation* with change in working set. While write off-loading uses the inter read-miss durations exclusively for spin down operations, SRCMap targets capturing entire working-sets including both reads and writes in replica locations to prolong read-miss durations to the order of hours and thus places more importance on learning changes in the working-set.

7 Evaluation

In this section, we evaluate SRCMap using a prototype implementation of SRCMap-based storage virtualization manager and an energy simulator seeded by the prototype. We investigate the following questions:

1. What degree of proportionality in energy consumption and I/O load can be achieved using SRCMap?
2. How does SRCMap impact reliability?
3. What is the impact of storage consolidation on the I/O performance?
4. How sensitive are the energy savings to the amount of over-provisioned space?
5. What is the overhead associated with implementing an SRCMap indirection optimization?

Workload The workloads used consist of I/O requests to eight independent data volumes, each mapped to an independent disk drive. In practice, volumes will likely comprise of more than one disk, but resource restrictions did not allow us to create a more expansive testbed. We argue that *relative* energy consumption results still hold despite this approximation. These volumes support a mix of production web-servers from the FIU CS department data center, end-user $homes$ data, and our lab's Subversion (SVN) and Wiki servers as detailed in Table 3.

Workload I/O statistics were obtained by running *blktrace* [1] on each volume. Observe that there is a wide variance in their load intensity values, creating opportunities for consolidation across volumes.

Storage Testbed For experimental evaluation, we set up a single machine (Intel Pentium 4 HT 3GHz, 1GB mem-

Volume	ID	Disk Model	Size [GB]	Avg IOPS	Max IOPS
home-1	D0	WD5000AAKB	270	8.17	23
online	D1	WD360GD	7.8	22.62	82
webmail	D2	WD360GD	7.8	25.35	90
webresrc	D3	WD360GD	10	7.99	59
webusers	D4	WD360GD	10	18.75	37
svn-wiki	D5	WD360GD	20	1.12	4
home-2	D6	WD2500AAKS	170	0.86	4
home-3	D7	WD2500AAKS	170	1.37	12

Table 3: **Workload and storage system details.**

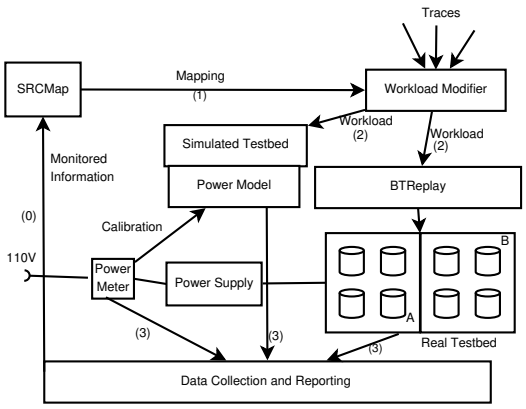

Figure 8: **Logical view of experimental setup**

ory) connected to 8 disks via two SATA-II controllers A and B. The cumulative (merged workload) trace is played back using *btreplay* [1] with each volume's trace played back to the corresponding disk. All the disks share one power supply P that is dedicated only for the experimental drives; the machine connects to another power supply. The power supply P is connected to a *Watts up? PRO* power meter [29] which allows us to measure power consumption at a one second granularity with a resolution of 0.1W. An overhead of 6.4W is introduced by the power supply itself which we deduct from all our power measurements.

Experimental Setup We describe the experimental setup used in our evaluation study in Fig. 8. We implemented an SRCMap module with its algorithms for replica placement and active disk identification during any consolidation interval. An overall experimental run consists of using the monitored data to (1) identify the consolidation candidates for each interval and create the virtual-to-physical mapping (2) modify the original traces to reflect the mapping and replaying it, and (3) power and response time reporting. At each consolidation event, the *Workload Modifier* generates the necessary additional I/O to synchronize data across the sub-volumes affected due to active replica changes.

We evaluate SRCMap using two different sets of experiments: *(i)* prototype runs and *(ii)* simulated runs. The prototype runs evaluate SRCMap against a real storage system and enable realistic measurements of power consumption and impact to I/O performance via the reporting module. In a prototype run, the modified I/O workload is replayed on the actual testbed using *btreplay* [1].

Volume ID	L(0) [IOPS]	L(1) [IOPS]	L(2) [IOPS]	L(3) [IOPS]	L(4) [IOPS]
D0	33	57	74	96	125
D1-D5	52	89	116	150	196
D6, D7	38	66	86	112	145

(a)

0	1	2	3	4	5	6	7	8
19.8	27.2	32.7	39.1	44.3	49.3	55.7	59.7	66.1

(b)

Table 4: **Experimental settings: (a) Estimated disk IOPS capacity levels. (b) Storage system power consumption in Watts as the number of disks in active mode are varied from 0 to 8.** *All disks consumed approximately the same power when active. The disks not in active mode consume standby power which was found to be the same across all disks.*

The simulator runs operate similarly on a simulated testbed, wherein a power model instantiated with power measurements from the testbed is used for reporting the power numbers. The advantage with the simulator is the ability to carry out longer duration experiments in simulated time as opposed to real-time allowing us to explore the parameter space efficiently. Further, one may use it to simulate various types of storage testbeds to study the performance under various load conditions. In particular, we use the simulator runs to evaluate energy-proportionality by simulating the testbed with different values of disk IOPS capacity estimates. We also simulate alternate power management techniques (e.g., caching, replication) for a comparative evaluation.

All experiments with the prototype and the simulator were performed with the following configuration parameters. The consolidation interval was chosen to be 2 hours for all experiments to restrict the worst-case spin-up cycles for the disk drives to an acceptable value. Two minute disk timeouts were used for inactive disks; active disks within a consolidation interval remain continuously active. Working sets and replicas were created based on a three week workload history and we report results for a subsequent 24 hour duration for brevity. The consolidation is based on an estimate of the disk IOPS capacity, which varies for each volume. We computed an estimate of the disk IOPS using a synthetic random I/O workload for each volume separately (Level $L1$). We use 5 IOPS estimation levels (L0 through L4) to (a) simulate storage testbeds at different load factors and (b) study the sensitivity of SRCMap with the volume IOPS estimation. The per volume sustainable IOPS at each of these load levels is provided in Table 4(a). The power consumption of the storage system with varying number of disks in active mode is presented in Table 4(b).

7.1 Prototype Results

For the prototype evaluation, we took the most dynamic 8-hour period (4 consolidation intervals) from the

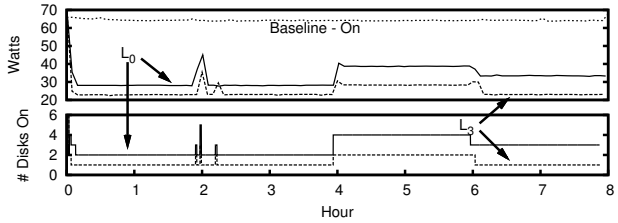

Figure 9: **Power and active disks time-line.**

24 hours and played back I/O traces for the 8 workloads described earlier in real-time. We report actual power consumption and the I/O response time (which includes queuing and service time) distribution for SRCMap when compared to a baseline configuration where all disks are continuously active. Power consumption was measured every second and disk active/standby state information was polled every 5 seconds. We used 2 different IOPS levels; $L0$ when a very conservative (low) estimate of the disk IOPS capacity is made and $L3$ when a reasonably aggressive (high) estimate is made.

We study the power savings due to SRCMap in Figure 9. Even using a conservative estimate of disk IOPS, we are able to spin down approximately 4.33 disks on an average, leading to an average savings of $23.5W$ (35.5%). Using an aggressive estimate of disk IOPS, SRCMap is able to spin down 7 disks saving $38.9W$ (59%) for all periods other than the 4hr-6hr period. In the 4-6 hr period, it uses 2 disks leading to a power savings of $33.4W$ (50%). The spikes in the power consumption relate to planned and unplanned (due to read misses) volume activations, which are few in number. It is important to note that substantial power is used in maintaining standby states ($19.8W$) and within the dynamic range, the power savings due to SRCMap are even higher.

We next investigate any performance penalty incurred due to consolidation. Fig. 10 (upper) depicts the cumulative probability density function (CDF) of response times for three different configurations: *Baseline - On* – no consolidation and all disks always active, SRCMap using L_0, and L_3. The accuracy of the CDFs for L_0 and L_3 suffer from a reporting artifact that the CDFs include the latencies for the synchronization I/Os themselves which we were not able to filter out. We throttle the synchronization I/Os to one every 10ms to reduce their interference with foreground operations.

First, we observed that less than 0.003% of the requests incurred a spin-up hit due to read misses resulting in latencies of greater than 4 seconds in both the L_0 and L_3 configurations (not shown). This implies that the working-set dynamically updated with missed reads and offloaded writes is a fairly at capturing the active data for these workloads. Second, we observe that for response times greater than 1ms, *Baseline - On* demonstrates better performance than L_0 and L_3 (upper plot). For both L_0 and L_3, less than 8% of requests incur latencies greater than 10ms, less than 2% of requests incur latencies greater than 100ms. L_0, having more disks at its disposal, shows slightly better response times than L_3. For response times lower than 1ms a reverse trend is observed wherein the SRCMap configurations do better than *Baseline - On* . We conjectured that this is due to the influence of the low latency writes during synchronization operations.

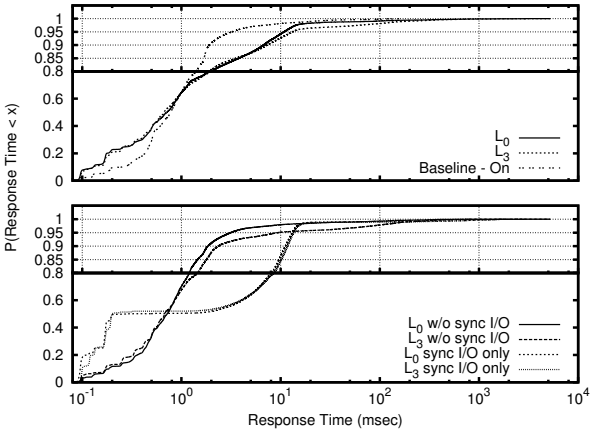

Figure 10: **Impact of consolidation on response time.**

To further delineate the influence of synchronization I/Os, we performed two additional runs. In the first run, we disable all synchronization I/Os and in the second, we disable all foreground I/Os (lower plot). The CDFs of only the synchronization operations, which show a bimodal distribution with 50% low-latency writes absorbed by the disk buffer and 50% reads with latencies greater than 1.5ms, indicate that synchronization reads are contributing towards the increased latencies in L_0 and L_3 for the upper plot. The CDF without synchronization ('w/o synch') is much closer to *Baseline - On* with a decrease of approximately 10% in the number of request with latencies greater than 1ms. Intelligent scheduling of synchronization I/Os is an important area of future work to further reduce the impact on foreground I/O operations.

7.2 Simulator Results

We conducted several experiments with simulated testbeds hosting disks of capacities $L0$ to $L4$. For brevity, we report our observations for disk capacity levels $L0$ and $L3$, expanding to other levels only when required.

7.2.1 Comparative Evaluation

We first demonstrate the basic energy proportionality achieved by SRCMap in its most conservative configuration (L0) and three alternate solutions, *Caching-1*, *Caching-2*, and *Replication*. *Caching-1* is a scheme that uses 1 additional physical volume as a cache. If the aggregate load observed is less than the IOPS capacity of

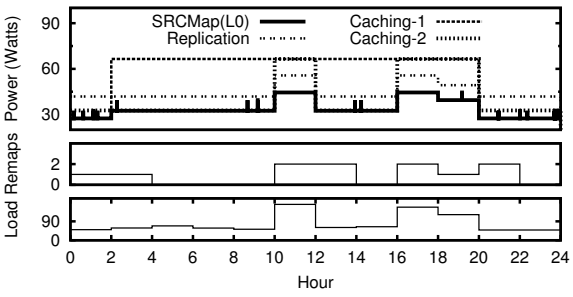

Figure 11: **Power consumption, remap operations, and aggregate load across time for a single day.**

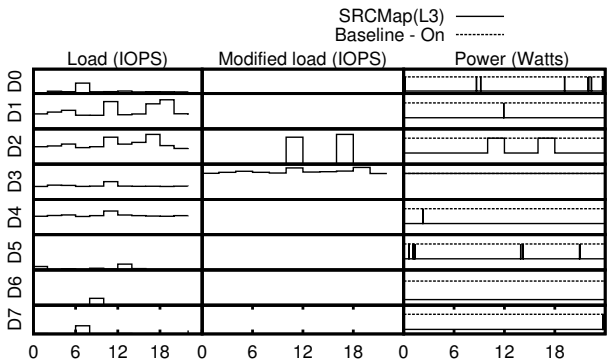

Figure 12: **Load and power consumption for each disk.** *Y ranges for all loads is $[1:130]$ IOPS in logarithmic scale. Y ranges for power is $[0:19]$ W.*

the cache volume, the workload is redirected to the cache volume. If the load is higher, the original physical volumes are used. *Caching-2* uses 2 cache volumes in a similar manner. *Replication* identifies pairs of physical volumes with similar bandwidths and creates replica pairs, where all the data on one volume is replicated on the other. If the aggregate load to a pair is less than the IOPS capacity of one volume, only one in the pair is kept active, else both volumes are kept active.

Figure 11 evaluates power consumption of all four solutions by simulating the power consumed as volumes are spun up/down over 12 2-hour consolidation intervals. It also presents the average load (measured in IOPS) within each consolidation interval. In the case of SRCMap, read misses are indicated by instantaneous power spikes which require activating an additional disk drive. To avoid clutter, we do not show the spikes due to read misses for the Cache-1/2 configurations. We observe that each of solutions demonstrate varying degrees of energy proportionality across the intervals. SRCMap (L0) uniformly consumes the least amount of power across all intervals and its power consumption is proportional to load. Replication also demonstrates good energy proportionality but at a higher power consumption on an average. The caching configurations are the least energy proportional with only two effective energy levels to work with.

We also observe that SRCMap remaps (i.e., changes the active replica for) a minimal number of volumes – either 0, 1, or 2 during each consolidation interval. In fact, we found that for all durations the number of volumes being remapped equaled the change in the number of active physical volumes. indicating that the number of synchronization operations are kept to the minimum. Finally, in our system with eight volumes, Caching-1, Caching-2, and Replication use 12.5%, 25% and 100% additional space respectively, while as we shall show later, SRCMap is able to deliver almost all its energy savings with just 10% additional space.

Next, we investigate how SRCMap modifies per-volume activity and power consumption with an aggressive configuration L3, a configuration that demonstrated interesting consolidation dynamics over the 12 2-hour consolidation intervals. Each row in Figure 12 is specific to one of the eight volumes $D0$ through $D7$. The left and center columns show the original and SRCMap-modified load (IOPS) for each volume. The modified load were consolidated on disks $D2$ and $D3$ by SRCMap. Note that disks $D6$ and $D7$ are continuously in standby mode, $D3$ is continuously in active mode throughout the 24 hour duration while the remaining disks switched states more than once. Of these, $D0$, $D1$ and $D5$ were maintained in standby mode by SRCMap, but were spun up one or more times due to read misses to their replica volumes, while $D2$ was made active by SRCMap for two of the consolidation intervals only.

We note that the number of spin-up cycles did not exceed 6 for any physical volume during the 24 hour period, thus not sacrificing reliability. Due to the reliability-aware design of SRCMap, volumes marked as active consume power even when there is idleness over shorter, sub-interval durations. For the right column, power consumption for each disk in either active mode or spun down is shown with spikes representing spin-ups due to read misses in the volume's active replica. Further, even if the working set changes drastically during an interval, it only leads to a single spin up that services a large number of misses. For example, $D1$ served approximately $5*10^4$ misses in the single spin-up it had to incur (Figure omitted due to lack of space). We also note that summing up power consumption of individual volumes cannot be used to compute total power as per Table 4(b).

7.2.2 Sensitivity with Space Overhead

We evaluated the sensitivity of SRCMap energy savings with the amount of over-provisioned space to store volume working sets. Figure 13 depicts the average power consumption of the entire storage system (i.e., all eight volumes) across a 24 hour interval as the amount of over-provisioned space is varied as a percentage of the total

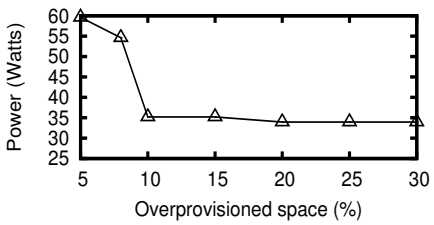

Figure 13: **Sensitivity to over-provisioned space.**

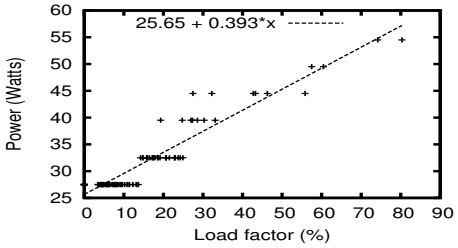

Figure 14: **Energy proportionality with load.**

storage space for the load level $L0$. We observe that SRCMap is able to deliver most of its energy savings with 10% space over-provisioning and all savings with 20%. Hence, we conclude that SRCMap can deliver power savings with minimal replica space.

7.2.3 Energy Proportionality

Our next experiment evaluates the degree of energy proportionality to the total load on the storage system delivered by SRCMap. For this experiment, we examined the power consumption within each 2-hour consolidation interval across the 24-hour duration for each of the five load estimation levels L0 through L4, giving us 60 data points. Further, we created a few higher load levels below $L0$ to study energy proportionality at high load as well. Each data point is characterized by an average power consumption value and a *load factor* value which is the observed average IOPS load as a percentage of the estimated IOPS capacity (based on the load estimation level) across all the volumes. Figure 14 presents the power consumption at each load factor. Even though the load factor is a continuous variable, power consumption levels in SRCMap are discrete. One may note that SRCMap can only vary one volume at a time and hence the different power-performance levels in SRCMap differ by one physical volume. We do observe that SRCMap is able to achieve close to N-level proportionality for a system with N-volumes, demonstrating a step-wise linear increase in power levels with increasing load.

7.3 Resource overhead of SRCMap

The primary resource overhead in SRCMap is the memory used by the *Replica Metadata (map)* of the *Replica manager*. This memory overhead depends on the size of the replica space maintained on each volume for storing both working-sets and off-loaded writes. We maintain a per-block map entry, which consists of 5 bytes to point to the current active replica. 4 additional bytes keep what replicas contain the last data version and 4 more bytes are used to handle the I/Os absorbed in the replica-space write buffer, making a total of 13 bytes for each entry in the map. If N is the number of volumes of size S with $R\%$ space to store replicas, then the worst-case memory consumption is approximately equal to the *map* size, expressed as $\frac{N \times S \times R \times 13}{2^{12}}$. For a storage virtualization manager that manages 10 volumes of total size 10TB, each with a replica space allocation of 100GB (10% over-provisioning), the memory overhead is only 3.2GB, easily affordable for a high-end storage virtualization manager.

8 Conclusions and Future Work

In this work, we have proposed and evaluated SRCMap, a storage virtualization solution for energy-proportional storage. SRCMap establishes the feasibility of an energy proportional storage system with fully flexible dynamic storage consolidation along the lines of server consolidation where any virtual machine can be migrated to any physical server in the cluster. SRCMap is able to meet all the desired goals of fine-grained energy proportionality, low space overhead, reliability, workload shift adaptation, and heterogeneity support.

Our work opens up several new directions for further research. Some of the most important modeling and optimization solutions that will improve a system like SRCMap are (i) new models that capture the performance impact of storage consolidation, (ii) investigating the use of workload correlation between logical volumes during consolidation, and (iii) optimizing the scheduling of replica synchronization to minimize impact on foreground I/O.

Acknowledgments

We would like to thank the anonymous reviewers of this paper for their insightful feedback and our shepherd Hakim Weatherspoon for his generous help with the final version of the paper. We are also grateful to Eric Johnson for providing us access to collect block level traces from production servers at FIU. This work was supported in part by the NSF grants CNS-0747038 and IIS-0534530 and by DoE grant DE-FG02-06ER25739.

References

[1] Jens Axboe. blktrace user guide, February 2007.

[2] Luiz André Barroso and Urs Hölzle. The case for energy proportional computing. In *IEEE Computer*, 2007.

[3] Luiz André Barroso and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of War ehouse-Scale Machines*. Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, May 2009.

[4] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *IEEE Conf. Integrated Network Management*, 2007.

[5] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *High Performance Networking and Computing Conference*, 2002.

[6] HP Corporation. Hp storageworks san virtualization services platform: Overview & features. http://h18006.www1.hp.com/products/storage /software/sanvr/index.html.

[7] EMC Corporation. EMC Invista. http://www.emc.com/products/software/invista/invista.jsp.

[8] Lakshmi Ganesh, Hakim Weatherspoon, Mahesh Balakrishnan, and Ken Birman. Optimizing power consumption in large scale storage systems. In *HotOS*, 2007.

[9] K. Greenan, D. Long, E. Miller, T. Schwarz, and J. Wylie. A spin-up saved is energy earned: Achieving power-efficient, erasure-coded storage. In *HotDep*, 2008.

[10] S. Gurumurthi, A. Sivasubramaniam, M. Kandemir, and H. Franke. Drpm: Dynamic speed control for power management in server class disks. In *ISCA*, 2003.

[11] S. Gurumurthi, M. R. Stan, and S. Sankar. Using intradisk parallelism to build energy-efficient storage systems. In *IEEE MICRO Top Picks*, 2009.

[12] IBM Corporation. Ibm system storage san volume controller. http://www-03.ibm.com/systems/storage/software/virtualization/svc/.

[13] IDC. Virtualization across the enterprise, Nov 2006.

[14] Patricia Kim and Mike Suk. Ramp load/unload technology in hard disk drives. *Hitachi Global Storage Technologies White Paper*, 2007.

[15] H. Lee, K. Lee, and S. Noh. Augmenting raid with an ssd for energy relief. In *HotPower*, 2008.

[16] Andrew W. Leung, Shankar Pasupathy, Garth Goodson, and Ethan L. Miller. Measurement and analysis of large-scale network file systemworkloads. In *Usenix ATC*, 2008.

[17] L. Lu and P. Varman. Diskgroup: Energy efficient disk layout for raid1 systems. In *IEEE NAS*, 2007.

[18] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron. Write off-loading: Practical power management for enterprise storage. In *Usenix FAST*, 2008.

[19] Network Appliance, Inc. NetApp V-Series for Heterogeneous Storage Environments. http://media.netapp.com/documents/v-series.pdf.

[20] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *ICS*, 2006.

[21] E. Pinheiro, R. Bianchini, and C. Dubnicki. Exploiting redundancy to conserve energy in storage systems. In *SIGMETRICS*, 2006.

[22] Control power and cooling for data center efficiency HP thermal logic technology. An HP Bladesystem innovation primer. http://h71028.www7.hp.com/erc/downloads/4aa0-5820enw.pdf, 2006.

[23] Mark W. Storer, Kevin M. Greenan, Ethan L. Miller, and Kaladhar Voruganti. Pergamum: Replacing tape with energy efficient, reliable disk-based archival storage. In *Usenix FAST*, 2008.

[24] Niraj Tolia, Zhikui Wang, Manish Marwah, Cullen Bash, Parthasarathy Ranganathan, and Xiaoyun Zhu. Delivering Energy Proportionality with Non Energy-Proportional Systems – Optimizing the Ensemble. In *HotPower '08: Workshop on Power Aware Computing and Systems*. ACM, December 2008.

[25] Luis Useche, Jorge Guerra, Medha Bhadkamkar, Mauricio Alarcon, and Raju Rangaswami. Exces: External caching in energy saving storage systems. In *HPCA*, 2008.

[26] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and migration cost aware application placement in virtualized systems. In *Middleware*, 2008.

[27] A. Verma, G. Dasgupta, T. Nayak, P. De, and R. Kothari. Server workload analysis for power minimization using consolidation. In *Usenix ATC*, 2009.

[28] J. Wang, X. Yao, and H. Zhu. Exploiting in-memory and on-disk redundancy to conserve energy in storage systems. In *IEEE Tran. on Computers*, 2008.

[29] Wattsup Corporation. Watts up? PRO Meter. https://www.wattsupmeters.com/secure/products.php?pn=0, 2009.

[30] C. Weddle, M. Oldham, J. Qian, A.-I. A. Wang, P. Reiher, and G. Kuenning. Paraid: a gear-shifting power-aware raid. In *Usenix FAST*, 2007.

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

A Appendix

A.1 Proof of Theorem 1

Proof: Note that the algorithm always selects the source nodes with the highest outgoing edge weight. Hence, it suffices to show that the outgoing edge weight of a source node equals (or is proportional to) the probability of it requiring a replica target on an active disk. Observe that the ordering property on weights holds in the first iteration of the algorithm as the outgoing edge weight for each $mdisk$ is the probability of it being spun down (or requiring a replica target). We argue that the re-calibration step ensures that the *Ordering property* holds inductively for all subsequent iterations.

Assuming the property holds for the m^{th} iteration, consider the $(m+1)^{th}$ iteration of the algorithm. We classify all source nodes into three categories: (i) $mdisks$ with P_i lower than the P_{m+1}, (ii) $mdisks$ with P_i higher than P_{m+1} but with no replicas assigned to targets, and (iii) $mdisks$ with P_i higher than P_{m+1} but with replicas assigned already. Note that for the first and second category of $mdisks$, the outgoing edge weights are equal to their initial values and hence their probability of their being spun down is same as the edge weights. For the third category, we restrict attention to $mdisks$ with only one replica copy, while observing that the argument holds for the general case as well. Assume that the $mdisk\ S_i$ has replica placed on $mdisk\ T_j$. Observe then that the re-calibration property ensures that the current weight of edge $w_{i,j}$ is P_iP_j, which equals the probability that both S_i and T_j are spun down. Note also that S_i would require an active target other than T_j if T_j is also spun down, and hence the likelihood of S_i requiring a replica target (amongst active disks) is precisely P_iP_j. Hence, the ordering property holds for the $(m+1)^{th}$ iteration as well. ■