Copy Rate Synchronization with Performance Guarantees for Work Consolidation in Storage Clusters

Feng Yan[1], Xenia Mountrouidou[1], Alma Riska[2], and Evgenia Smirni[1]

[1]College of William and Mary, Williamsburg, VA, USA, fyan,xmount,esmirni@cs.wm.edu
[2]EMC Corporation, Cambridge, MA, USA, alma.riska@emc.com

ABSTRACT

As storage in data centers is increasing rapidly, it has become critical to find ways to operate efficiently this important component of a data center. Often, it has been proposed to consolidate the storage workload into a subset of storage devices and shutdown the unused ones with the purpose of preserving power. In many cases storage workload consolidation requires some amount of data to be copied from one device or set to the next. While storage workload consolidation techniques focus on extending power savings with minimal penalty in the performance of a data center, less attention is paid to the process of seamlessly integrating the data copy phase into the overall storage workload consolidation technique. Specifically, in this paper, we propose an analytic framework that synchronizes the pace of copying data between two storage devices (or nodes) such that performance is maintained within predefined targets. As such, we avoid either undesired performance degradation caused by aggressively scheduling the data copy task or a slow copy process caused by conservative scheduling. We show with extensive experimentation that the framework is robust and that it provides an important step toward automating storage consolidation and high power savings.

1. INTRODUCTION

Storage consolidation has been proposed as an effective way to increase idleness at individual or sets of disk drives aiming at using disk idleness for power savings [12, 4]. From this perspective, the entire working set of a disk or portions of it may be redirected elsewhere in the storage system [1, 6, 7, 10]. Intelligent replication schemes to decide the optimal number of active disks to serve a workload in the system have been proposed [10]. Strategies conserve energy by exploiting disk parallelism to spin down disks so that more disks are used for better performance, less for power savings have been proposed in [11].

These studies have put emphasis on the question of how to gain more extra power savings, but they do not provide the solution to estimate beforehand the overheads due to the consolidation process. During consolidation, the disks need to handle extra background work to copy the workload, which may have an impact on the performance of foreground work they serve. If such a copy work is scheduled without considering the performance of the foreground tasks, the system may experience a disaster from the performance point of view. In addition, problems may come from the different copy pace between the consolidation disks. For example, if the disk sending data is much faster than the disk receiving data, the buffer of the disk receiving data may overflow.

In this paper, we address the problem of how to synchronize effectively the process of copying data from one disk to another with performance guarantees. The framework allows to estimate beforehand the maximum copy rate given the desired performance requirements on both the storage node that sends the data and the node that receives it. If the storage nodes to be "consolidated" are not pre-defined, then the system can select the best consolidation pair such that their copy rates are best matched in order to reduce exposure to buffer overflow risk and performance degradation. Alternatively, if we can not find such consolidation pair in the system or if the consolidation disk pair is pre-defined, then we can actively synchronize the copy pace according to the slower storage node to reduce the risk of buffer overflow and performance degradation.

Overall, the framework that we propose provides a storage cluster with the tools to estimate the maximum copy rate with the desired performance guarantees, as well as synchronize the copy pace between sending data and receiving data to achieve the resource consolidation with minimum performance overheads. The framework scales well because it only relies on information of the current workload at the individual storage nodes (disks) that includes the arrival rate of requests, the response time of requests, and idle times. Such information is commonly logged in storage systems and is compact and lightweight. Extensive trace-driven evaluation of the framework establishes its robustness and that its estimations are accurate.

This paper is organized as follows. In Section 2 we describe briefly the algorithm used to schedule the copy operation as a background job. In Section 3 we develop the methodology to synchronize the copy operation between the consolidation nodes in a storage cluster. Section 4 presents an extensive set of trace-driven experiments that demonstrates the robustness of the framework. We conclude and discuss future directions in Section 5.

2. BACKGROUND

In this section, we describe, in high level details, an algorithmic framework that schedules maintenance work, such as the data copy process that we address in this paper, with performance guarantees. This algorithmic framework is used to estimate the impact of copying data and determine the most effective schedule for it.

We give an overview of the algorithmic framework that determines when and for how long to schedule the maintenance or background tasks in storage devices, such that the

trade-off between performance degradation and completion of the background tasks meets system quality targets [9, 1].

One could argue that starting a background task immediately after the disk (or any storage device) becomes idle would maximize the amount of background work that can be completed in the system. However, because of the stochastic nature of idle periods and the non-preemptive nature of tasks in storage devices (e.g. disk drives) user performance may suffer significantly. In storage systems, it is very common to idle wait for some time before starting a background task, as to avoid utilizing the very short idle periods for any background activities [2]. In addition to that, [3] suggests that limiting amount of time that the system serves background tasks further limits the performance impact on foreground jobs. The framework in [5] computes both the *idle wait* I and the duration T of the time to serve background jobs, *background busy period*. We use here this (I, T) tuple to compute the schedules of the data copy process for consolidation storage nodes in a cluster, while meeting predefined performance targets.

Central to the calculation of I and T is the cumulative distribution histogram (CDH) of idle intervals that allows for a compact representation of the empirical distribution of the lengths of idle times. This CDH is created by dividing the range of the idle interval lengths into equal-sized "bins". Then, the number of observed idle intervals that fall into each bin is calculated and the frequency of an interval of a specific size is obtained. In addition to the CDH of idle intervals, the framework uses the user-provided average performance degradation D, which is defined as the average relative delay of an IO operation due to the background tasks and can be computed from the (I, T) scheduling pair and other statistical information such as average response time. For more details on computing D, we direct the interested readers to [5]. Usually, larger D ensures that more background work can be completed. There may exist multiple (I, T) tuples for a specific D target, but the framework chooses the one that maximizes the overall amount of background work completed.

3. COPY SYNCHRONIZATION

We focus on estimating the ability of disks in a cluster to take over additional work, with the goal of consolidating resources. Here, we refer to the disks that receive the redirected workload as receiver nodes and the disks that redirect the workload as the sender nodes. The sender and receiver nodes in a cluster may be pre-defined. In this case, our framework synchronizes the pace of data copying with performance guarantees to minimize the overheads such as buffer overflow or performance degradation. If there are multiple potential sender or receiver nodes, the framework selects the sender and receiver nodes pair such that their copy rates are closest to each other and then synchronizes their copy rate if they are not close enough. We consider the first scenario to be a sub-case of the second one. There are two aspects in our framework to minimize the cost during consolidation. First, both sender and receiver nodes should allow for the data copy to complete transparently, and second, the copy rate of the sender and receiver should be synchronized by our framework.

3.1 Copy Rate Estimation and Scheduling

Copying data for the purpose of consolidation should be transparent. We measure transparency in terms of the performance degradation D in the average relative delay of an IO operation due to the data copy. Certainly, the completion of the data copy, depends on the amount of data to copy, which we assume that ranges from a few GBytes up to hundreds of GBytes. The data copy process is considered as a background task. Consequently, the algorithmic framework in [5] can be used to schedule it during the available idle times at the sender and receiver nodes. The framework uses the histogram of idle times to generate a "schedule" for background work that is *always* waiting for service and it can estimate the amount of completed work for each idle interval.

We use such a framework to compute all the valid scheduling pairs (I, T) given the performance degradation target D. Each scheduling pair schedules in average $T_{copy\_per\_idle}$ amount of copy work as background task measured in units of time (or simply mean interval length) in idle intervals at the storage nodes. We calculate $T_{copy\_per\_idle}$ as follows:

$$T_{copy\_per\_idle} = \sum_{o=I}^{I+T-P} p(o) \cdot (o - I) + \sum_{o=I+T-P}^{max} p(o) \cdot (T - P) \quad (1)$$

where $p(o)$ is the probability that an idle interval is of length o, max is the maximum length of the idle intervals in the CDH, and P is the time penalty that a foreground IO request may suffer, if it arrives while the disk is copying the data to be consolidated. Note the penalty for sender and receiver may be different because the sender is reading data from disk while the receiver is writing the data to disk. In this aspect, we calculate individual scheduling pair for the sender and receiver nodes. Intuitively, $T_{copy\_per\_idle}$ is comprised of two kinds of idle intervals that are larger than idle wait time I (intervals smaller than I are not used for copy). The first type of idle intervals are of length o that falls between I and $I+T-P$. Because the copy task in this kind of intervals terminates at the end of each idle interval, which is before the limiting time T, their contribution to the overall $T_{copy\_per\_idle}$ is only $o - I$. The second type of idle intervals are of length o that at least $I + T - P$. In this case, the copy task for T time units, so their contribution to the overall $T_{copy\_per\_idle}$ is $T - P$. Then we multiply them by the probability of each used interval and sum them together to get the average amount of copy work $T_{copy\_per\_idle}$. Among all the valid scheduling pairs (I, T), we choose the one with largest $T_{copy\_per\_idle}$.

Assume that the total bandwidth for a disk is R, which indicates the maximum transfer ability of a disk if it is fully idle and not serving any other I/Os. If ρ is the disk utilization, then the bandwidth occupied by the foreground tasks is $R \cdot \rho$ and the total left bandwidth is $R \cdot (1 - \rho)$. Let β be the percentage of bandwidth the copy task can gain from the total left bandwidth, then β is calculated by:

$$\beta = \frac{T_{copy\_per\_idle}}{E[I]} \quad (2)$$

where $E[I]$ is the mean idle interval length. This equation expresses the fraction of time used for the copy task in each idle interval. The bandwidth used for copying r can be obtained from the following equation:

$$r = R \cdot (1 - \rho) \cdot \beta \quad (3)$$

Trace	Util (%)	Idle Length		Mean Arrival Rate	Mean Service Rate
		Mean (ms)	CV		
CODE1 (C1)	5.6	192.6	8.4	0.0089	0.1596
CODE2 (C2)	0.7	1681.6	2.3	0.0013	0.1859
FILE1 (F1)	1.7	767.5	2.3	0.0033	0.1938
FILE2 (F2)	0.7	2000.2	3.8	0.0011	0.1596

Table 1: General trace characteristics.

Since the bandwidth used for copying is essentially the copy rate, once it is computed for each sender and receiver node, we can evaluate the information each node provided for copy rate synchronization between senders and receivers.

3.2 Copy Rate Synchronization

The second part of our framework synchronizes the sender and receiver nodes such that the copy rate of sender is equal to the copy rate of receiver to reduce the buffer overflow risk and unnecessary performance degradation. If there are multiple potential senders and receivers in the cluster, we select the pair with the closest copy rates. There are three cases after selecting a sender and receiver pair: i. The copy rate of sender is equal to the receiver. We define such case as copy rate synchronized and it is the target of our copy rate synchronization framework, ii. The copy rate of sender is faster than the receiver. In this case, the buffer on the receiver side can be overflowed. Since the bottleneck is at the receiver side, the sender's higher bandwidth is wasted. Note that higher bandwidth is gained by sacrificing performance, so wasting bandwidth indicates wasting performance, iii. The copy rate of sender is slower than the receiver. In this case, the bottleneck is the sender, so the receiver bandwidth used for copying is wasted. This implies opportunities to improve the receiver's performance.

Our framework detects the later two cases as mismatch cases and automatically adjusts the copy rate by choosing a new (I, T) scheduling pair to synchronize the copy pace between sender and receiver.

In general, we define r_{slow} as the copy rate of the slower side between sender and receiver, and we use it as the target copy rate for the faster side. Because the bottleneck is at the slower side, we slow down the copy rate of the faster side to r_{slow} so that the problems due to mismatched copy rate, e.g., buffer overflow and unnecessary performance degradation, are avoided.

Knowing the target copy rate r_{slow}, we can use Equation (2) and (3) to compute $T_{copy\_per\_idle}$ as T_{slow}. Then we check all scheduling pairs (I, T) and find one that schedules in average T_{slow} amount of copy work at the faster side with the lowest performance degradation D.

4. EXPERIMENTAL EVALUATION

In this section we evaluate the correctness and effectiveness of our framework. First, we describe the traces and the environment we use to do the evaluation. Then we show the accuracy of copy rate estimation by comparing the estimated copy rates and simulation results from a series of experiments. Finally, based on the estimated copy rates, we use scenarios and quantify metrics to show the benefits of the copy rate synchronization.

A set of enterprise traces measured at the disk level from

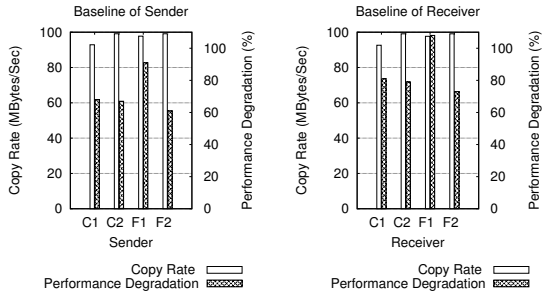


Figure 1: The copy rate and performance degradation without our framework.

an application development server ("CODE") and a file server ("FILE") [8] is used in our evaluation. These traces have a duration of twelve hours. For each request, the following metrics are logged: the arrival time, the departure time, the type of the request (i.e., read or write), the request length in bytes, and the location on the disk. This information allows to calculate a rich set of metrics exactly and we show a subset of these metrics used in our evaluation in Table 1. Table 1 clearly shows that all disks are underutilized. Highly variable periods of idleness suggest that there are opportunities for data consolidation aiming at higher operation efficiency, i.e., power savings.

In our experiments, we consider a storage cluster of four disks. Each node serves a different workload from Table 1. Since the estimation framework is lightweight, it scales well with the cluster size, but here we choose 4 nodes to manage the presentation of results. We use the first half of the trace (6 hours) to build up the continuous data histogram of idle times, the average response time of IO requests, and their average service times. The collected statistics are used in our framework described in Section 3 to estimate the copy rate and perform copy rate synchronization, then the framework computes a scheduling pair (I, T) as output. The second half of the trace uses this pair for copy rate synchronization.

We use trace-driven simulations to show the baseline of our experiments in Figure 1: the copy rate and performance degradation without our framework, which means the disk starts immediately the copy work when the disk is idle and do it until new request comes. From the figure, we can see though the copy rates (show in left y-axis of the plot) are high, the performance degradation (show in right y-axis of the plot) can range from 60% to over 100%, which may break the performance requirements in systems.

4.1 Copy Rate Estimation and Scheduling

We first use extensive experiments to verify the accuracy of our copy rate estimation for the scheduling with performance guarantees. We note that before synchronization, the computed scheduling pair depends on only the workload characterization (i.e., the CDH of idle periods), the performance requirement D, and the average penalty P of a disk, so the copy rate of each disk depends its own information.

In the evaluation of the copy rate estimation, we show extensive cases that a reasonable performance degradation from 0% to 40% can be achieved. If the storage node serves as a sender, then the penalty of a copy task is assumed to be 5 ms on the average because the sender only reads the data from the media. If the storage node serves as a

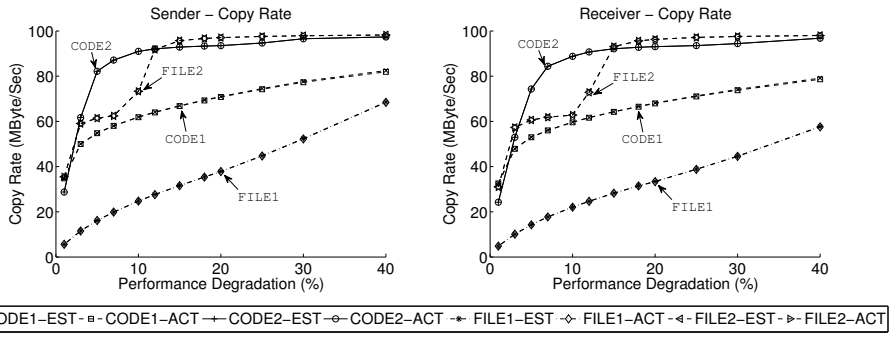


Figure 2: Comparison of the estimated (EST) and simulation (ACT) copy rates for different performance degradations and sender/receiver workloads.

receiver then the penalty of the copy task is assumed to be 6 ms in average because the receiver writes the data and usually it takes slightly longer in a disk. These penalties are approximated from the average seeking time in disks. More accurate values can be obtained during monitoring of the disk operation. Since here we only want to show an example case that the penalty of receiver is slightly higher than the sender due to the difference between reading and writing operation, these approximated penalties serve our needs.

To verify the correctness of the framework, we run trace-driven simulations with the calculated (I, T) pairs to get the copy rate. In the simulation, the penalty caused from the copy task is the same as the one used in the estimations. The arrivals and departures of requests are the ones in the trace. From the simulation, we measure the amount of data to be processed as copy tasks and average it in time to get the corresponding copy rate. We validate our estimations by comparing the estimated results with the ones resulted from the trace-driven simulations.

Because the copy rate of each disk depends on its own information before synchronization, we show the results of the four disks as sender and receiver respectively in Figure 2. In the figure, we plot the estimated copy rates (EST) and copy rates from simulations (ACT) for different performance degradation levels of different disks. We can see for the same disk, the EST plot curve and the ACT plot curve matches very well, which indicates the good accuracy of our estimations.

We also notice that for different workloads, the plot curve can be quite different, which suggests the usefulness of our estimations. One important observation from our results is that simple measurements, like utilization, may be insufficient to indicate the potential copy rate capabilities with performance guarantees. For example, the Table 1 shows the "FILE1" does not have the highest utilization among the four disks, but we can see from Figure 2 that its copy rate is always the lowest among the four disks in all cases.

4.2 Copy Rate Synchronization

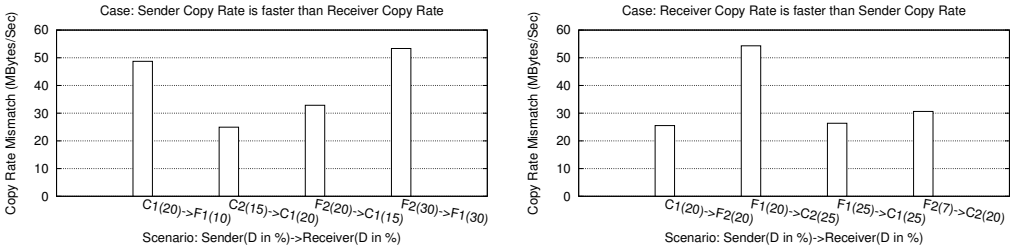
The results in Figure 2 suggest the copy rate changes with performance degradation for the same disk and one can control the copy rate simply by setting different performance degradation levels through different scheduling pairs (I, T). Figure 2 also shows that for sender and receiver pairs with the same or different performance degradation levels, the copy rate of sender can be quite different from receiver, which implies the needs for copy rate synchronization.

In the evaluation, we compare the Copy Rate Mismatch (in MBytes/Sec) and Performance Degradation (in %) to show the benefits of our framework. Obviously, higher Copy Rate Mismatch means higher buffer overflow risk and higher performance degradation means higher performance cost. The results are plotted in Figure 3. We use four scenarios where the sender copy rate is faster (left column in figure) and four scenarios where the sender is slower (right column in figure) to demonstrate the benefits of synchronization. The x-axis corresponds to these scenarios.

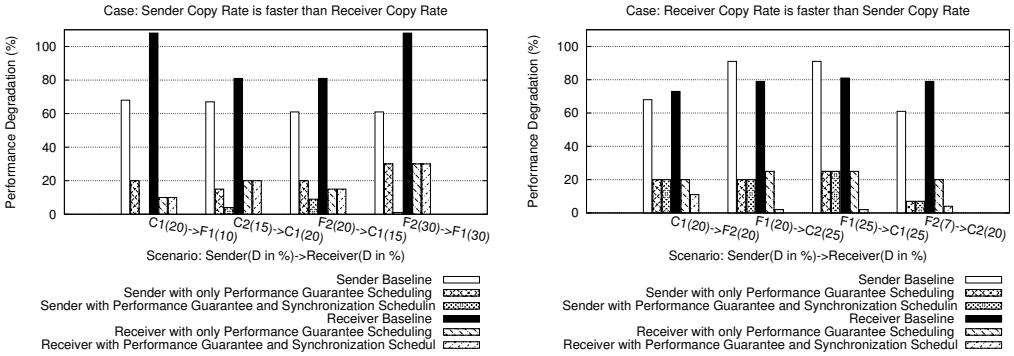
Figure 3(a) shows Copy Rate Mismatch (in MBytes/Sec) if the scheduling just guarantees the performance degradation without the synchronization. It illustrates the necessity of synchronizing the sender and receiver copy rate even when the performance is guaranteed. For example, if "FILE2" is used as sender with performance degradation level of 30% and "FILE1" is used as receiver with performance degradation level of 30% (i.e., the F2(30)→ F1(30) scenario in Figure 3(a)), without the copy rate synchronization, the sender copy rate is 53.34 MBytes/Sec higher than the receiver's. Since most of today's high-end storage disks have a buffer size from tens to hundreds of MBytes, the buffer may be overflowed in seconds.

Figure 3(b) shows the performance degradation comparison between the baseline, the scheduling with only performance guarantees, and the scheduling with both performance guarantees and synchronization. We observe that without performance guaranteed scheduling (i.e., baseline), the performance effects on the system may be disastrous, i.e., one order of magnitude higher than the system can afford. In addition, after the synchronization of sender and receiver copy rate, the unnecessary performance degradation (up to 30%) can be avoided.

The (I, T) pair is determined by how an IO workload performance in a given storage device. So to synchronize work between different storage devices, we will need to estimate the (I, T) pair for each of the storage devices participating in the communication. The copy rate resulting from an (I, T) pair is estimated in average and deploying these scheduling parameters does not need to overlap idle intervals, just the existence of a small buffer (at the sender). Estimating the copy capabilities in each storage device participating in the communication means that the size of the buffer to use for moving the data does not need to be large and will



(a) Copy Rate Mismatch



(b) Performance Degradation Comparison

Figure 3: Copy Rate Mismatch (in MBytes/Sec) without synchronization and Performance Degradation (in %). → stands for the copy direction.

not experience overflow. As a result, our methodology allows synchronization of the communication between heterogeneous storage devices such as solid state drives and hard disk drives. For example, the copy rate for a flash drive may be much higher than for a SATA drive. Once both rates are estimated, then the solid state drives can send the data to the SATA drive at the correct pace and unnecessary buffer overflow is avoid.

5. CONCLUSIONS AND FUTURE WORK

We propose a method for data consolidation with high efficiency while low overheads that is transparent to end users. Our estimation of the copy rate is accurate and provides criteria for the synchronization of work consolidation between sender and receiver disks in a cluster. Finally, our framework uses commonly logged simple information and does not require any feedback loops during consolidation, therefore, it is compact, lightweight and scalable.

An interesting extension of our work is a fully automated generalized framework for work consolidation targeting power savings, backups, or virtualization.

6. ACKNOWLEDGMENTS

This work is supported by NSF grants CCF-0811417 and CCF-0937925. The authors thank Seagate Technology for providing the enterprise traces used for this work.

7. REFERENCES

[1] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, pages 1–11, 2002.

[2] L. Eggert and J. Touch. Idletime scheduling with preemption intervals. In *In Proceedings of the 20th ACM Symposium on Operating Systems Principles (SOSP)*, pages 249–262, 2005.

[3] R. Golding, P. Bosch, C. Staelin, T. Sullivan, and J. Wilkes. Idleness is not sloth. In *In Proceedings of the Usenix Technical Conference UNIX Advanced Computer Systems*, pages 201–212, 1995.

[4] J. Guerra, W. Belluomini, J. Glider, K. Gupta, and H. Pucha. Energy proportionality for storage: impact and feasibility. *SIGOPS Operating Systems Review*, 44(1):35–39, March 2010.

[5] N. Mi, A. Riska, X. Li, E. Smirni, and E. Riedel. Restrained utilization of idleness for transparent scheduling of background tasks. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems, SIGMETRICS/Performance*, pages 205–216, 2009.

[6] X. Mountrouidou, A. Riska, and E. Smirni. Adaptive workload shaping for power savings on disk drives. In *Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering*, pages 109–120. ACM, 2011.

[7] D. Narayanan, A. Donnelly, and A. I. T. Rowstron. Write off-loading: Practical power management for enterprise storage. In *Proceedings of the USENIX Conference on File And Storage Technologies (FAST)*, pages 253–267, 2008.

[8] A. Riska and E. Riedel. Disk drive level workload characterization. In *Proceedings of the USENIX Annual Technical Conference*, pages 97–103, May 2006.

[9] A. Riska and E. Smirni. Autonomic exploration of trade-offs between power and performance in disk drives. In *Proceedings of the 7th IEEE/ACM International Conference on Autonomic Computing and Communications (ICAC)*, pages 131–140, 2010.

[10] A. Verma, R. Koller, L. Useche, and R. Rangaswami. SRCMap: Energy proportional storage using dynamic consolidation. In *Proceedings of 8th USENIX Conference on File and Storage Technologies (FAST'10)*, pages 154–168, 2010.

[11] C. Weddle, M. Oldham, J. Qian, and A. Wang. PARAID:a gear-shifting power-aware raid. In *In Proceedings of 5th USENIX Conference on File and Storage Technologies (FAST'07)*, pages 245–269, 2007.

[12] Q. Zhu, J. Zhu, and G. Agrawal. Power-aware consolidation of scientific workflows in virtualized environments. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–12. IEEE Computer Society, 2010.