Back to the Future: Using Magnetic Tapes in Cloud Based Storage Infrastructures

Varun S. Prakash, Xi Zhao, Yuanfeng Wen, Weidong Shi

Back to the Future: Using Magnetic Tapes in Cloud Based Storage Infrastructures

Varun S. Prakash[+], Xi Zhao[*], Yuanfeng Wen[†] and Weidong Shi[‡]
vsprakash@uh.edu[+], xzhao21@central.uh.edu[*], wyf@cs.uh.edu[†]
larryshi@cs.uh.edu[‡]

Department of Computer Science, University of Houston
4800 Calhoun Road, Houston, TX 77004, U.S.A

Abstract. Data backup and archiving is an important aspect of business processes to avoid loss due to system failures and natural calamities. As the amount of data and applications grow in number, concerns regarding cost efficient data preservation force organizations to scout for inexpensive storage options. Addressing these concerns, we present Tape Cloud, a novel, highly cost effective, unified storage solution. We leverage the notably economic nature of Magnetic Tapes and design a cloud storage infrastructure-as-a-service that provides a centralized storage platform for unstructured data generated by many diverse applications. We propose and evaluate a proficient middleware that manages data and IO requests, overcomes latencies and improves the overall response time of the storage system. We analyze traces obtained by live archiving applications to obtain workload characteristics. Based on this analysis, we synthesize archiving workloads and design suitable algorithms to evaluate the performance of the middleware and storage tiers. From the results, we see that the use of the middleware provides close to 100% improvement in task distribution efficiency within the system leading to a 70% reduction in overall response time of data retrieval from storage. Due to its easy adaptability with the state of the art storage practices, the middleware contributes in providing the much needed boost in reducing storage costs for data archiving in cloud and colocated infrastructures.

Keywords: [Data Storage, Backup, Archiving, Cloud, Data Centers, Cost Efficiency, Magnetic Tapes, Middleware, Read Probability Weight, Priority Queue]

1 Introduction

The last decade has witnessed an explosion of data generated by individuals and organizations. For instance, the amount of video data captured by a single HD surveillance camera at 30fps in 14 days requires 1TB storage space [1]. The number of CCTV cameras in UK alone is estimated to be 1.85 million [2]. One of the major concerns that is correlated with managing such data is its storage and backup[3]. In cloud based storage services, there are usually more than one players involved, such as service providers and users. From the service user's

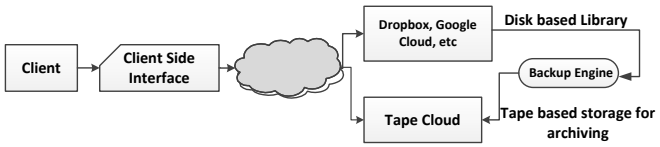


Fig. 1. Tape Cloud is a cloud storage service that uses magnetic tapes as the main storage media to store unstructured and big data unlike most of the commercial cloud storage solution available today.

perspective, the motives for choice of storage would be reduced costs per unit data stored, efficient retrieval, data criticality dependent support benefits and a secure, long term data storage. But, a service provider's considerations span operating cost efficiency, labor, scalability, support for different types of data, varied policies from multiple clients and managing workload uncertainty among others. A closer observation shows that the cost factor favors either players but rarely both. The likelihood of recovery of data after back up, also firmly influences both players. Varying archiving rates and backup needs of multiple clients is an eminently common feature leading to the need for multiple storage configuration. Thus, a sensible inclusion in the storage tiers to archive low-read/write-only data would be a low cost, low maintenance yet durable media [4].

Magnetic tapes, which started of as a primary storage media decades ago, have been preferred for archiving data generated by organizations for a long time now. Despite the advantages of tapes, there has not been a steady increase in its usage due to high initial investment needed for the operating hardware and its inability to promise high data rate transactions[5]. By addressing these key issues, it is possible to tap into the economic advantages that the tape media provides.

Tape Cloud (figure 1) is a venture that seeks to find suitable solutions to these issues. Tape Cloud is a cloud based, nearline storage Infrastructure-as-a-Service which makes use of magnetic tapes as the main backend storage media. The cloud model exempts users from the large initial investments needed for in-house backup infrastructure, external tiers for archiving legacy data and its maintenance. From the service providers perspective, using tapes allows hassle free scaling of systems and reduces the total cost of ownership due to its characteristic low power usage, durability and form factor per unit data.

Our principle intention is to **1.** Reduce the average response times for read requests issued by applications; **2.** Conjointly, ensure efficient data writes to the tapes tier of storage; and **3.** Strengthen the infrastructure's support for a large and diverse client base[6]. However, overcoming the latency offered by tapes is a complex problem to be solved. Even with the latest in tape technology, high performance in terms of fast data read and efficient data write cannot be achieved as delays caused due to seeking and winding of tapes is still persistent. There is also a delay induced by the stock robots and other ambulatory mechanics within the tape library which physically handle and move the tape cartridges.

The main contributions of our work are follows,

- We propose and evaluate a middleware that is designed to work with Tape Cloud. The functions of the middleware includes the aggregation and batch processing of data, IO request management and efficient distribution of data over available resources.
- The middleware, which is constituted by a FUSE based filesystem, implementation of priority based queuing of IO tasks and a latency preemptive, probabilistic data distribution scheme, acts between the backup application tier and proprietary filesystems that is commonly used with tapes.
- We observe and record the common delays incurred in the operation of commercially available tape libraries. Some of the latencies of tape drives and tape filesystem are analysed using typical benchmarking tools. This data, along with delay is used to model the performance characteristics of unit hardware, which is later used to simulate large scale data centers.
- Backup and archiving application traces are analysed to obtain typical workload characteristics. We employ methods to trace the operations at different stages in the infrastructure and aggregate them into meaningful statistics. This not only provides information about backend storage media activity, but also provides data at the application server and filesystem levels.
- We use synthetic workloads which emphasise prominent features of backup applications to evaluate the impact of using the proposed middleware in a simulation of a large scale deployment of Tape Cloud. In keeping with our goals, we demonstrate the improved data distribution ability, improved response time for read requests originating from each of the applications and regulation of write requests that the middleware provides.
- The proposed tape cloud framework points to a new direction for creating service oriented, cost effective, massive scale infrastructure to meet the growing storage challenge in the coming era of big data enabled industries and research.

2 Analyzing and Modeling Tape Associated Latencies

2.1 The Tape, Library and the Drive

In order to design an infrastructure around a particular storage media, it is important to understand the characteristics, related costs, advantages and weaknesses that are associated with it. A clear understanding of the media and devices can lead to its large scale deployment in data centers.

We evaluate the state of the art in tape technology with the use of a commercially available Tandberg T24 LTO5 tape library. The tape library has an HP tape drive and slots that can hold 12 LTO5 Ultrium tapes each of 2.5TB capacity and can be extended to 24 tapes. At full capacity, the library can hold 60TB of uncompressed data. The tape library depends on robotic carriers that grab tapes from the slots, carries them to the tape drive at the end of the library and loads the tape for IO operations. The robots instills a greater delay into the

Table 1. Tandgerg T24 Robot, Load and Unload Delays

Type	From (slot)	To	Motion (sec)	Load (sec)	Type	From	To (slot)	Motion (sec)	Load (sec)
LOAD	1	Drive	52.4	23.3	UNLOAD	Drive	1	51.6	20.1
LOAD	2	Drive	52.9	21.9	UNLOAD	Drive	2	52.3	20.6
LOAD	3	Drive	54.06	22.6	UNLOAD	Drive	3	52.26	20.3
LOAD	4	Drive	55.2	24.6	UNLOAD	Drive	4	54.0	20.3
LOAD	5	Drive	52.42	24.0	UNLOAD	Drive	5	51.3	20.9
LOAD	6	Drive	53.3	23.6	UNLOAD	Drive	6	51.76	21.01
LOAD	7	Drive	54.2	21.3	UNLOAD	Drive	7	52.22	20.1
LOAD	8	Drive	55.45	23.9	UNLOAD	Drive	8	53.8	19.62
LOAD	9	Drive	51.8	24.0	UNLOAD	Drive	9	50.7	20.3
LOAD	10	Drive	52.3	21.6	UNLOAD	Drive	10	51.4	20.34
LOAD	11	Drive	53.7	22.23	UNLOAD	Drive	11	51.97	23.9
LOAD	12	Drive	54.02	23.8	UNLOAD	Drive	12	53.6	22.59
Average	–	–	**53.52**	**23.1**	**Average**	–	–	**52.24**	**21.21**

system in addition to the one caused by tape drives. The averages from multi trail recordings of the traverse time of the robots and loading time is provided in table 1.

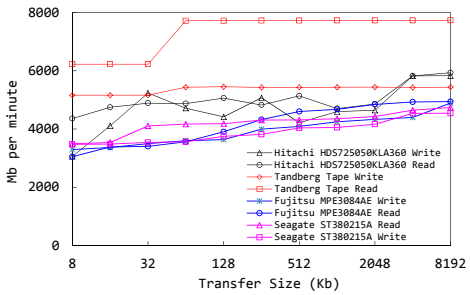


Fig. 2. Sequential read and write performance of an LTO5 tape drive in comparison with commercial hard disks.

The results of a study performed for various block sizes show that tape drives have a uniform data transfer rate compared to three other hard disks shown in figure 2. However, a difference in performance can be seen when random reads and writes are performed. The time spent in changing tapes, loading and seeking to the correct point on the tape creates delays that are out of proportion as compared to the sequential performance of tapes. An important takeaway from the results is to assure the tape drive and the infrastructure spends most of the time either writing or reading to tapes and less time performing seek operations. This helps us in deciding important parameters such as rate of batch processing of data.

2.2 Generic Models for Tape Based Latency

Based on the facts obtained about the hardware and delays, we try to model the latency for generic cases[7]. For the models, the following are some of the constants that need to be considered.

- T_{search}(i) is the time taken by the robot to locate and move to the tape to execute the i^{th} request in the task queue.
- T_{load} is the time taken to load the tape into the drive.
- T_{unload} is the time taken to unload the tape from the drive.

- T_{seek}(average) is the time to wind the tape to seek to the position of the first byte to execute the new task. We consider the average time for LTO5 tapes in this case.
- γ_{read} is the data transfer rate for read operations of the tape. Similarly γ_{write} is the data transfer rate for write operations of the tape.
- The smallest unit of a data that is considered in this case is a block. A single read or write might involve transaction of a varying number of blocks. We represent a unit block as BLK.

We aim to employ able techniques to reduce the average response time T_{read} for read tasks and furthermore, ensure that these read-friendly techniques, cause minimal distortion to the throughput and total time T_{write} required to collect data and write it onto tapes. Thus, for a workload Θ,

$$T_{opt}(\Theta) = min(T_{read}(\Theta) + \Delta T_{write}(\Theta)) \tag{1}$$

Where $T_{opt}(\Theta)$ is the minimal optimal time required to complete the execution of workload Θ. We analyse some of the latencies and overhead incurred in achieving this goal in different scenarios. These scenarios are commonly occurring cases in storage systems.

Scenario 1: Single Read/Write Task in Queue: When there is a single read task in the task queue, the total amount of time required to complete the task and obtain the data is given as the sum of times taken for a series of events. Thus $T_{singleRead} = T_{search} + T_{load} + T_{seek} + n(\frac{BLK}{\gamma_{read}})$ where n is the total number of unit blocks that need to be read. $\frac{BLK}{\gamma_{read}}$ is a constant, the total time required to read a single block and can be substituted by Γ_{read} to get

$$T_{singleRead} = T_{search} + T_{load} + T_{seek} + n\Gamma_{read} \tag{2}$$

Similarly, a single write operation in a queue undergoes similar delays as read operations, the only difference being the rate at which data is written to tapes. The delay for a single write operation is given by

$$T_{singleWrite} = T_{search} + T_{load} + T_{seek} + n\Gamma_{write} \tag{3}$$

Scenario 2: Write task(s) before Read task in Queue: In scenarios where there are one or more write tasks in the queue before a read task, the total time required to obtain the data will include the time required to complete the write task too. For a single write task before the read task, the total time required to complete the task will be equal to $T_{total} = T_{search} + T_{load} + T_{seek} + n\Gamma_{write} + T_{unload} + T_{search} + T_{load} + T_{seek} + n\Gamma_{read}$. This can simply be written as $T_{total} = T_{singleWrite} + T_{unload} + T_{singleRead}$. Generalizing this, when we have N write tasks before a read task, we have

$$T_{total} = N(T_{singleWrite}) + \xi(T_{unload}) + T_{singleRead} \tag{4}$$

where $0 \leq \xi \leq (N-1)$. ξ is called the tape switch rational which determines the probable number of tape changes that need to be made and is based on BLK and n. Thus BLK is an important value that influences the efficiency of write operations and helps in deciding the maximum size of data that can be written as a continuous process on to a single tape.

Scenario 3: Other Read Task(s) before Read Task in Queue: The total time required for a particular read task to complete when there are one or more read task ahead of it differs from the previous scenarios in that, read requests are usually not localized to a single tape mostly due to replication and data striping. Continuous read requests mean more number of search, load and seek operations, thus increasing the overall time taken. In the worst case, the total time taken can be given by

$$T_{total} = (N+1)(T_{singleRead}) + (N)(T_{unload}) \tag{5}$$

where there are N read requests ahead of the read task in question. This not only causes excessive delays in retrieving data but also leads to the pile up of write tasks at the queue in scenarios where there is an equal ratio of read to write requests.

3 Proposed System's Approach to Overcome Latency

3.1 Prioritizing Read Tasks over Write Tasks

From equation 4, we can see that a major share of the delay occurs due to the tasks ahead of the read task in the queue. In order to reduce the over all time taken for retrieving data, an approach that can be opted is biasing between read and write tasks. The read tasks can be given a higher preference over write tasks. For this, we create a Priority Queue for read tasks for each tape drive. When a read task arrives at a tape drive, the subsequent write task is blocked and the tape drive immediately caters to the read task after finishing the current execution. Thus we have

$$T(Pri)_{total} = T_{total} - (N(T_{singleWrite}) + \xi(T_{unload})) + (T_{unload} + T_{singleRead}) \tag{6}$$

$$T(Pri)_{total} = \rho + T_{unload} + T_{singleRead} \tag{7}$$

where $T(Pri)_{total}$ is the total time taken when priority queueing is applied. ρ is the time spent for completion of current task and $\lceil \rho \rceil = T_{singleWrite}$ and $0 \leq \xi \leq (N-1)$. By implementing the priority queuing, read tasks can be accelerated to be completed much faster.

3.2 Read Probability Weight (RPW) Based Data Distribution

Under the circumstances of scenario 3, applying priority queueing would not significantly reduce the total response time as subsequent read operations that need

to be performed on different tapes still induce delay associated with the search and seeking processes. We propose a method to overcome this by considering the Balls into Bins problem [8][9][10].

Every block of data that needs to be written to tapes have a certain probability of being read again. This probability or "weight" is based on the type of application and its historic transactions with the storage system. Intuitively we can see that blocks of data with a higher weight causes higher delay when written to tapes by the same tape drive as compared to data with lower weight (because the read requests that come in eventually still have to be queued at the same tape drive). So the motive to reduce this delay has to be to distribute the data blocks of higher weight equally among the available tape drives such that a single tape drive need not take the entire burden of heavy weighted objects. This is similar to a Balls into Bins problem except that in our case, balls are of different weights. Assume that there are n types of data blocks, where $W_n = \{P_r^1, P_r^2 ... P_r^n\}$ are its respective weights. Given m tape drives $T_{drive} = \{t_1, t_2 ... t_m\}$, the RPW data distribution makes sure that

$$\forall t \epsilon (T_{drive}), (\sum_{i=0}^{k} P_r^i)/k \approxeq \varphi(W_n) \tag{8}$$

where $\varphi(W_n)$ is the arithmetic mean of all the elements in the set (W_n) and

$$\sum_{j=0}^{p/2}((\sum_{i=0}^{k} P_r^i)/k) - \sum_{j=p/2}^{m}((\sum_{i=0}^{k} P_r^i)/k) \approxeq 0 \tag{9}$$

Where k is the total number of write tasks in a particular queue t. If data originating from an application q is assigned a weight P_q at any point of time, then each queue will have a weight S_q equivalent to $P_q / \sum_{n=1}^{N} P_n$ of data pertaining to application q where N is the total number of weighted tasks in the queue. No single application can have all its data written to a single location. RPW based data distribution coupled with priority queueing not only improves average response time efficiency, but also contributes towards maintaining write throughput as it reduces the overall delay caused due to continuous blocking of write tasks by a series of read tasks. An evaluation of RPW usage has been shown in figure 12.

4 System and Middleware Design

Figure 3 shows a bird/s eye view of the Tape Cloud architecture. We propose a hybrid middleware that performs efficient hard disk caching, data block management, data distribution and IO task scheduling. This middleware functions as an agent arbitrating various components in order to reduce the overhead caused by using the slower backend media. Figure 4 provides the logical representation of the middleware and some of its functionalities. The data that needs to be written to tapes is collected and channelled suitably before it reaches its destination.

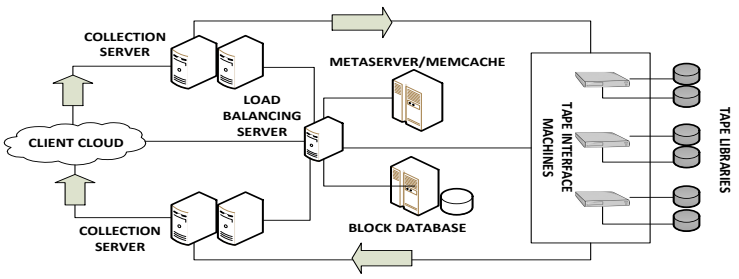


Fig. 3. Implementation Architecture of Tape Cloud. The arrows represent the direction of flow of data. The infrastructure is a hybrid structure which makes use of hard disk caches and databases.

Data is processed in batches. This helps in easy retrieval of data from collection servers and fixed set of parameters for efficient distribution.

4.1 Data Source or Clients

The focus of Tape Cloud is consistent with most cloud based services and provides an efficient storage service for a variety of data. Clients who wish to archive data on Tape Cloud, run a service to deliver data to the storage collection servers(see figure 3). One of the features of Tape Cloud is that it allows clients to deliver data in more than one ways. Large data sets(which is an unavoidable attribute of archive data) can also be delivered by mailing the media itself. From the storage system's perspective, each client is tagged and labelled based on the physical attributes of the data, relative storage activity over time, space requirements and the frequency of requests for data IO that is derived from the clients. This information serves as policies which is used by the middleware to make decisions on the location of data, level of security, distribution of data blocks and also provides the recipe to cook the read probability weight (RPW) information of data pertaining to particular clients. The data manager, with access to the central block database, updates and maintains mapping of blocks of data to its physical location on tapes, in libraries and section of the data center.

4.2 Data and Resource Manager

The Data Manager is the point of interaction between the clients and the storage infrastructure. More importantly, it is the interaction point between the client application and the middleware as no data is directly written to tapes without the data manager's consent. The data manager module runs on the load balancing server and manages the other parts of the middleware such as the filesystem, task queues and data distribution modules. To perform efficient management, the data manager relies on informative references to the actual client data. These references or metadata contains details about the blocks of data such as its location in the filesystem, size, type and RPW along with other client specific

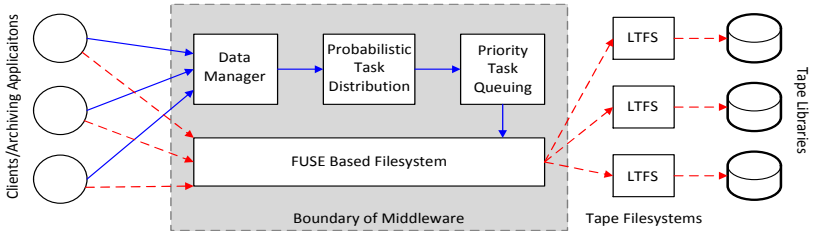


Fig. 4. The placement and interfacing of the functional blocks of the Middleware. The solid lines show the path taken by control statements and meta data while dotted lines show the path of the actual data blocks to be stored on tapes.

information. The metadata is used as representatives of data blocks in the queuing and the distribution modules of the middleware. This prevents the overhead of moving around large amounts of data within the system.

An important task the data manager undertakes is the grouping of data stored in the middleware's filesystem to be processed in batches. The data manager employs a specific technique to pick metadata pertaining to blocks of data which are most probable to be retrieved as a single unit from the filesystem, packages them and passes them to the data distribution module. Other responsibilities include the attestation of data deposition requests from and client and allocating suitable resources.

4.3 Multi Tier File System

FUSE [11] is a framework to help develop customized file system. FUSE module has been officially merged into the Linux kernel tree since kernel version 2.6.14. FUSE provides 35 interfaces to fully comply with POSIX file operations. We design a file system using FUSE to operate in the middleware of the architecture. The implementation presents a monolithic image of the filesystem, but internal divisions exist based on functionalities. Figure 5 shows the pathway taken by data to be written to tapes and the various operations that act upon it. The filesystem depends on external databases to maintain records of the locations of blocks of data. In order to prevent loss of data due to server failure, the filesystem performs a replication of similar data in multiple location similar to HDFS.

The filesystem manages data and chunks based on a hierarchical partitioning technique of the data set. Tape Cloud follows an application centric approach to group data chunks to be written to tapes and a method called hierarchical partitioning that is used, contributes to this cause. Every file that needs to be written to or read from tapes is encrypted, optionally segmented(to avoid singular large files) and replicated to result in a unit entity or chunk. The chunks of data are grouped and bagged in structures called containers. Based on the load, these containers are then distributed to the tape interface machines to be written to tapes.

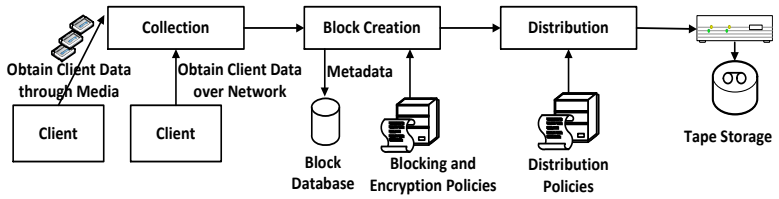


Fig. 5. Stages and functions of each stage of the filesystem for Tape Cloud. Although distributed by functionality, the filesystem is monolithic across the storage system.

4.4 Probabilistic Data Distribution

The analysis of latencies that is performed leads to induction of a technique where some of the delays are preempted before data is written to tapes. As discussed in section 3, this is to ensure that a small group of task queues do not take the burden of a large number of discontinuous read tasks. The probabilistic data distribution module is an important part of the middleware that distributes blocks of data to the tape interface machines based on a particular weight associated with the data. The weight or the read probability weight (RPW) is the probability of the block of data being read once written onto tape. The probabilistic data distribution module is designed to obtain the RPW by two ways. It can be enclosed in the metadata that is handed down by the data manager. The other avenue that can be taken to deduce the RPW is over time, when the middleware notices that there are some blocks of data that have undergone access in a manner inconsistent with its knowledge about the RPW. In this scenario, the middleware updates the RPW of data incoming from the client and adapts to the workloads of different clients over time. After the references have been assigned specific tapes or drives, the references of data blocks are handed over to the task queuing module of the middleware.

4.5 Task Queueing

The large scale operation of the storage system involves the use of multiple tape drives. The entire tape storage facility is divided into sections, each of which can be serviced by a tape drive. Each of these tape drives have an exclusive queue assigned to it which holds the IO task to be performed on tapes which are in its logical vicinity. These tasks queues are maintained and used by the middleware and should not be confused with the ones that are used by the storage media or drivers. One of the approaches to decrease the delay in retrieving data is to prioritize between the read and write requests as discussed in section 3. The task queueing module caters to this need by assigning each tape drive with two virtual queues, one each for write and read requests. Read requests having higher priority over write requests are granted resources immediately after the completion of the current task regardless of the depth of the write queue. After completion of the read task, the system continues with the execution of other tasks in

Table 2. Applications Contributing Workload Traces for Evaluation of Middleware

Sl. No.	Archiving Type	Description
1	Periodic Full Backup	10 disk array on 3 networked attached storage (NAS) servers archiving surveillance video and security data. Videos and related information is collected from local systems once every 24 hours through a customized asynchronous pull server based system. High churn rate.
2	Periodic Full Backup + LRU Archiving	Application archived least recently used support files on larger disk based backend storage with smaller churn rate. Deployment details and infrastructure unknown.
3	Incremental+Full Backup	Incremental backup of hard disks and virtual disks at the end of every login session and periodic full backup of 22 computers on hard disk based NAS storage running CryptoNAS software.
4	Non Periodic Mirroring Backup	Document archiving of unknown number of computers. Simple FreeNAS storage with a duplication based archiving client running on individual computers.

the write queues. Assuming an efficient distribution of data, the task queueing module ensures that read tasks are performed under strict time constraints while maintaining acceptable standards of throughput for write tasks. The task queues provide periodic feedbacks to the data manager about the overall time taken in performing tasks associated with a specific batch. This feedback is used by the data manager to assess the overall performance of the data distribution module and the distribution parameters in the system.

5 Synthesis of Workload for Middleware Evaluation

5.1 Characterizing Archive Workload from Traces

The accepted method to evaluate a storage infrastructure is by testing its performance with benchmark workloads. While a number of articles provide benchmarks and suggest methods to evaluate various aspects of storage such as the media, queues, IO charecterization[12] and filesystem [13][14], there has been a comparatively limited literature about performance of archival storage systems. Kavalanekar et. al.[15] provide elaborate results on storage workloads from production windows servers. But the variation in workload type between non archival and archival storage varies as suggested by Lee et. al. in [16], who make an attempt to create benchmarks. But their work is limited to providing a better understanding of the type of files and sizes rather than provide a complete set of results. Another important contribution has been provided by Wallace et. al.[17] for EMC production servers. Although a large number of aspects have been covered, the impact of different types of archiving and application level transactions with the storage have not been projected.

In order to perform a bias free evaluation of the middleware, we subject it to a workload that has been characterized by traces obtained from live archiving applications. The traces are collected from the archiving infrastructure of IVigil, a company that provides video surveillance services to a local client base. The backed up data usually includes surveillance videos, security related data, virtual disks and documents that are wielded by the company on a daily bases. Aspects which are important to the working of the middleware such as rate of requests with respect time, inter arrival time of requests and a comparison of

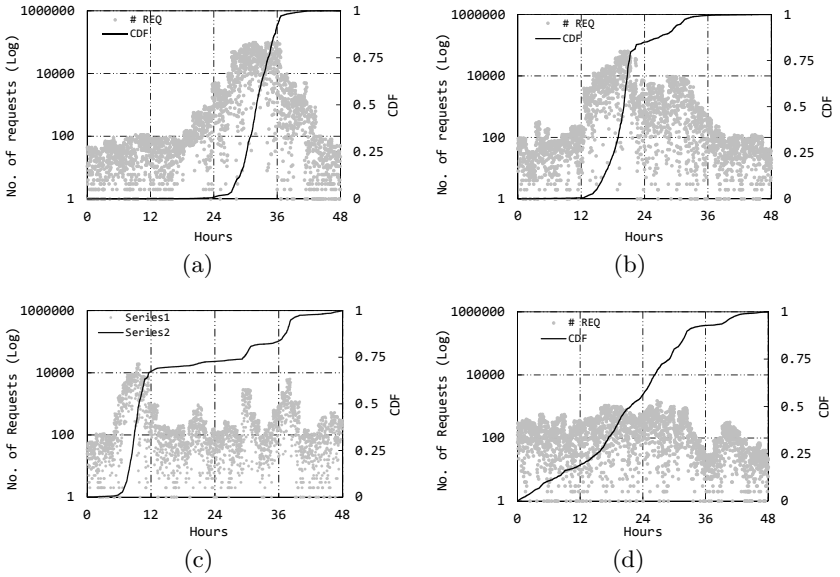


Fig. 6. The total number of requests generated by archiving applications 1(a), 2(b), 3(c) and 4(d). The number of requests are collected at the application level for discrete read or write requests to the underlying filesystems.

the rate of read to write request are recorded and analysed. Table 2 provides some information about the characteristics of the infrastructure.

The applications show characteristics that prove common beliefs about archival data wrong[16]. The application level traces help in understanding the frequency with which IO requests are generated. This serves as a clear indicator of how backup types differ from each other. The filesystem level traces provide a defined understanding of what each IO request generated by the application demands. Each of the applications vary in infrastructure so it is important to co-relate traces obtained to reflect a common operation at each stage. The following are the results of the characteristic extraction from the traces.

Figure 6 is the total number of IO requests generated by the archiving applications and figure 7, the interarrival time of these requests. These have been recorded at the application level or at the first level of the storage infrastructure. The number of storage requests generated is an important feature to be considering as it provides valuable insight into the nature of application and guidelines on the capacity that the middleware needs to cope. Interarrival time helps in setting parameters such as the queue lengths, batch processing rate etc.

As discussed earlier, random IO is responsible for the major share of the delay in a tape infrastructure. Figure 8 and figure 9 provides a better understanding of the number of read requests obtained as a ratio of write requests and how frequently 200 individual “hot” files are accessed within the storage system.

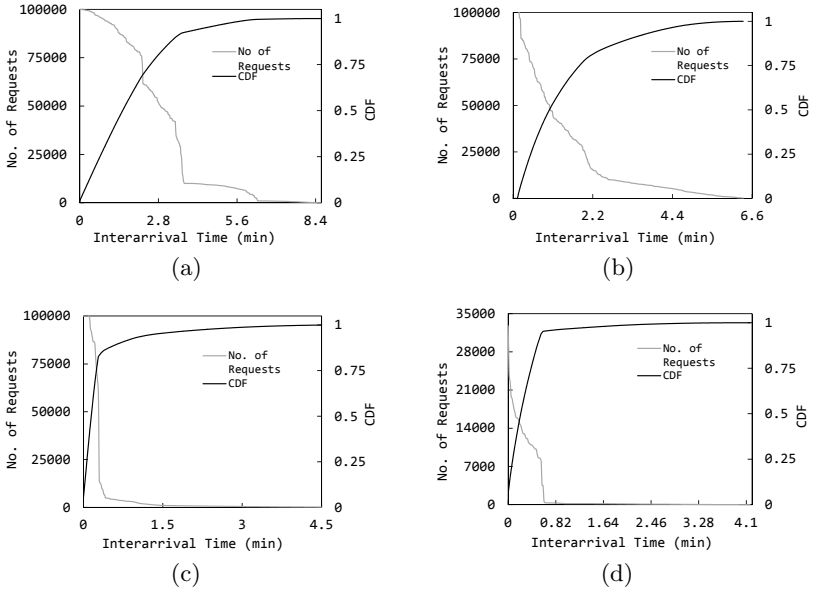


Fig. 7. Interarrival(IA) time of requests generated by archiving applications 1(a), 2(b), 3(c) and 4(d). Application, type of data, file sizes and temporal locality are some of the factors influencing interarrival time. The asynchronous nature of some applications and storage system softwares also affect IA time.

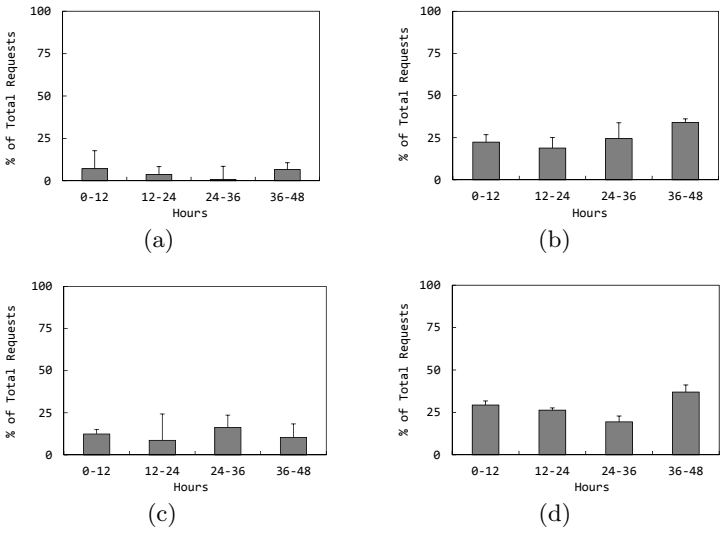


Fig. 8. Average number of read requests as a percentage of the total IO requests in 12 hour buckets by archiving applications 1(a), 2(b), 3(c) and 4(d). The whiskers show the maximum percent of read requests received during the particular 12 hour interval.

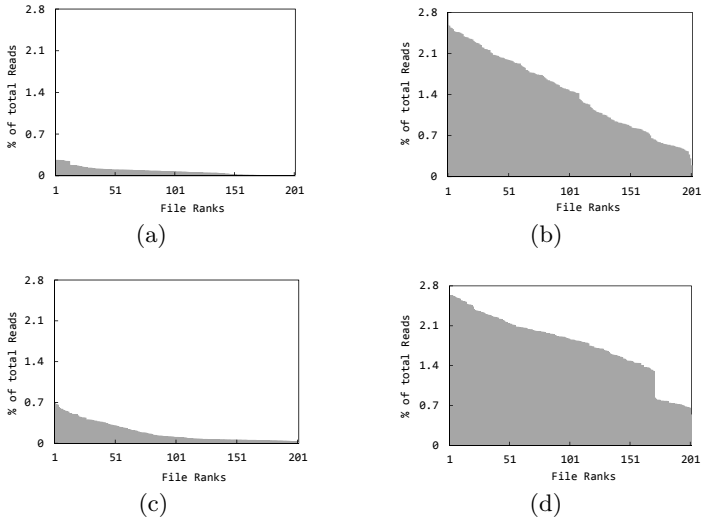


Fig. 9. Total number of read requests for the 200 most frequently accessed files as a percentage of the total read requests received by archiving applications 1(a), 2(b), 3(c) and 4(d).

5.2 Workload Modeling and Generation

There have been many projects in developing synthetic workload to test storage systems such as [18][7] which depend on models created by Markov chains of states and virtualized environments. The commendable results focus on workloads that vary from archiving workloads. We synthesize a workload using Vdbench[19] in order to test the middleware's performance. The workload generator is carefully designed by performing a sectional analysis of the results obtained in the real archive workload traces. The real time workloads are spliced on the basis of a user defined time interval and the features of each division such as number of requests, types of requests, file sizes and interarrival times are extracted. The newly created workload is essentially a time based, weighted aggregation hybrid of the workloads. The weighted aggregation provides the flexibility to produce workloads in any combination of amounts of the given traces. It depends on a workload aggregation scheme provided by the user which generates a Vdbench script based on the input. For example, an aggregation scheme (W1,W2,W3,W4) would produce a workload from the 4 participating workloads in equal proportion, ((2)W1,(0.5)W2,W3) would produce twice the amount of workload 1, half the amount of workload 2 with no change to workload 3 an no trace of workload 4. This type of modelling has proven to provide a wide range of options for generating workloads. The focus of this paper being the evaluation of the middleware, we use an equal proportion workload to record the difference in performance.

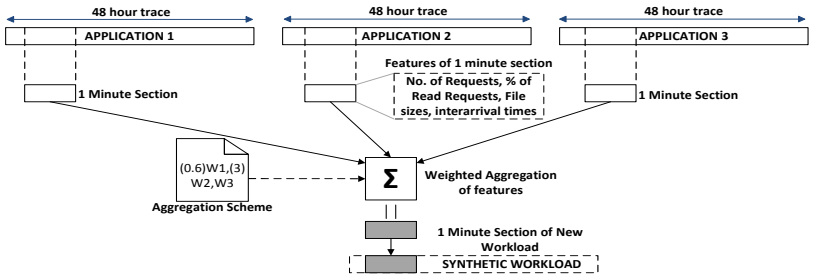


Fig. 10. The process of synthesizing a workload based on previously analysed application traces. The traces are divided based on a user defined time interval, features extracted and an aggregation performed to create a block of the new artificial workload

6 Experiment Results

6.1 Experimental Methodology

We perform our evaluation experiments using the models and synthetic workload created on the basis of the actual archiving workloads. The performance of the middleware and its contribution in achieving the goals to minimize average response time and efficient data distribution, are assessed by subjecting the backend storage system to the synthetic workload in the absence and presence of the middleware on simulated, resource configurable data center test bed. In the former case, we make use of commonly preferred ways of task and data distribution at the application and middleware levels such as First Come First Serve (FCFS)+Round Robin and Application specific task queuing techniques. To evaluate the middleware, we consider the Priority Queuing and evaluate its performance. As mentioned in section 2, the priority queueing technique has a few drawbacks which is then overcome with RPW Data distribution method. All tests are conducted along with the middleware filesystem. First of all, it is important to check for inconsistencies in the synthetic workloads as compared to the real time workloads obtained from traces. Figure 11 gives the error percentage of the synthetic workloads.

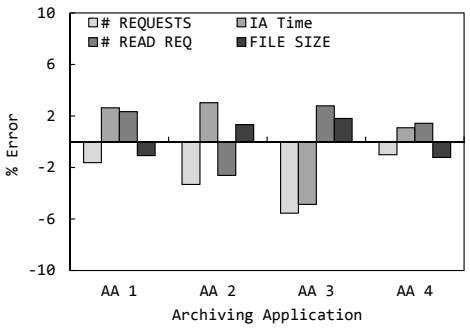
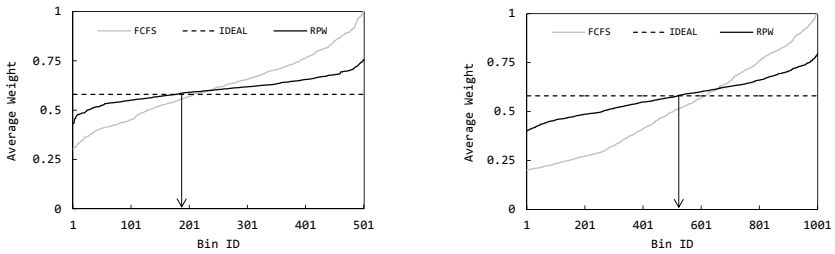


Fig. 11. The difference or error % between the actual and synthetic workloads used in the experiments.



(a) (b)

Fig. 12. Verifying the correctness of the RPW approach. Compared to FCFS, RPW offers a higher convergence to the ideal case. Here Fig(a) is with 500 bins and Fig(b) is with 1000 bins. The arrow points to the queue ID which serves as the point of distribution balance.

6.2 Read Probability Weight based Data Distribution

The novel idea of preempting delay caused due to large number of read requests especially in a system like Tape Cloud calls for preliminary evaluation of the technique. RPW considers the probability of a block of data being read once written to tapes and distributes blocks based on this probability. To verify the correctness of our assumption, we consider 10000 randomly weighted objects and distribute them into bins. Two tests are performed, where each has 500 and 1000 bins. This emulates blocks with different probabilities that need to be assigned to different tape drives. Figure 12 shows that RPW offers a distribution that is closer to the ideal case than other approaches like FCFS in both cases.

In evaluating the RPW using the synthetic workload, we consider two cases where we have 500 tape drives (figure 13) and 1000 tape drives (figure 14). We compare RPW with FCFS and Application Specific Queueing which distributes data blocks generated by specific applications to specific queues. The application specific approach has clear boundaries between queues for each application in the system. When we vary the number of total requests generated by the synthetic workload, we see that RPW provides a more efficient distribution where the gap between the queue with the largest average weight and the queue with smallest average weight is much lesser than that of the other approaches. The whiskers show the largest and smallest average weights of queues.

6.3 Average Response Time for Read Requests

The use of RPW based data distribution helps in avoiding long stretches of read operations that is localized to a small set of task queues. This in turn reduces the average delay caused at each of the queues. When we test Tape Cloud with the synthetic workload, the absence of the middleware leads us to use conventional data distribution and queueing techniques such as FCFS, Round Robin and application specific queueing of tasks. But with the middleware and enhanced task management, there is an overall reduction in the response time for read

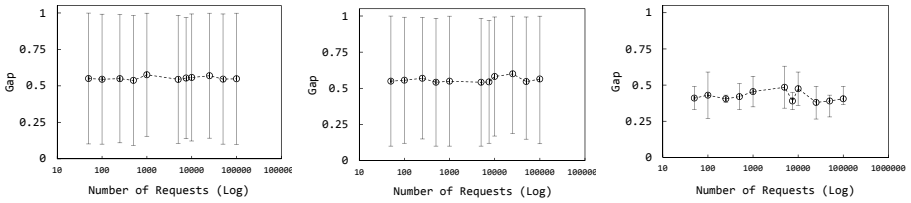


Fig. 13. The gap between the average weights of the heaviest and lightest queues for different number of requests for 500 queues. FCFS (a) and Application Specific Queueing (b) show inefficient weight distribution as compared to RPW (c).

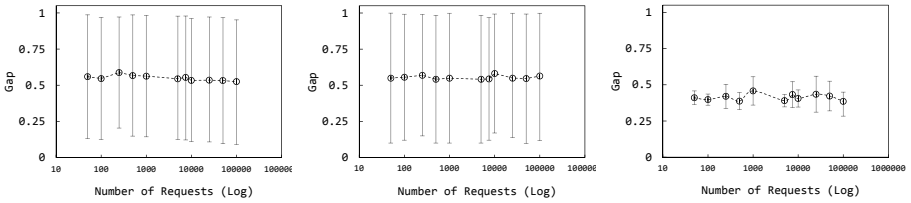


Fig. 14. The gap between the average weights of the heaviest and lightest queues for different number of requests for 1000 queues. FCFS (a) and Application Specific Queueing (b) show inefficient weight distribution as compared to RPW (c).

tasks generated by every application as shown in figure 15. The graphs have Log values in X axis which show the rate of change of average response time when number of requests are varied and the RPW have negligible rate of change of response time even for large number of requests.

One of the notable differences that can be seen in the traces of the four application is the variation in number of requests over time. Theoretically, the induction of RPW based data distribution along with priority queueing must make the average response time immune to the number of total number of requests. We perform an hourly analysis of average response time for read requests from application 1 and application 2 because application 1 has the highest write requests and application 2 has the highest read requests. We see from figure 16 that, along with having the smallest response time, the combination of priority queueing and RPW distribution provides a nearly constant response time over the entire period of the test, making it independent of other requests.

6.4 Preserving Rate of Write Task Execution

In keeping with our goals, we test if the middleware brings about a negative impact on the write task completion rate of the workload. Figure 17 provides a comparison of the write performance before and after the deployment of the middleware. We test cases that present extreme scenarios such as application 1

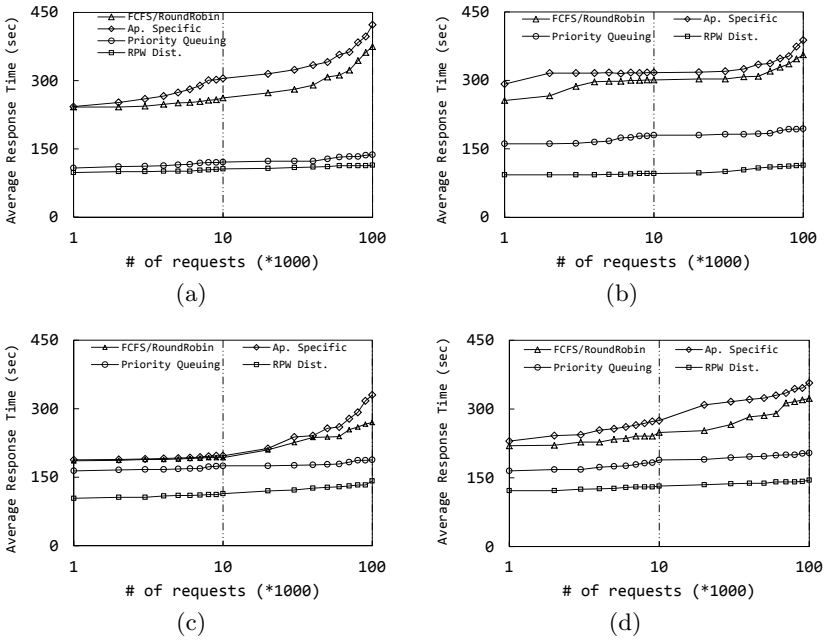


Fig. 15. The average response time of read requests under the synthetic workload for application 1 (a), application 2 (b), application 3 (c) and application 4 (d). Note the clear difference and reduction of the average response time for each of the applications. Also, RPW based data distribution offers very small rate of increase of response time even over larger variations of the number of requests

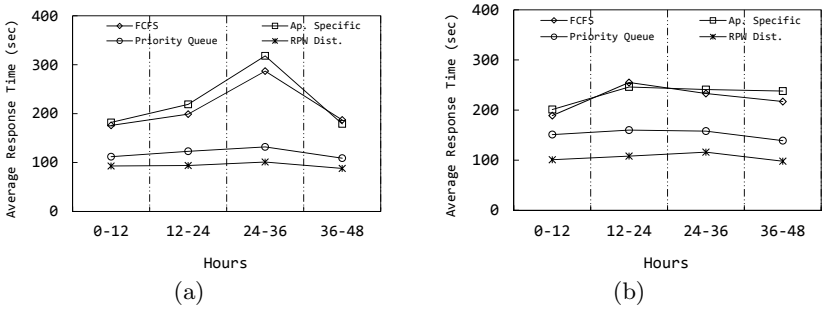


Fig. 16. Time based average response time for application 1 (a) and application 2 (b). Applications 1 and 2 are considered because application 1 has the highest write requests and application 2 has highest read requests. Compared to the other methods such as FCFS and Application specific Queuing, RPW based data distribution maintains a stable average response time regardless of the density of the workload

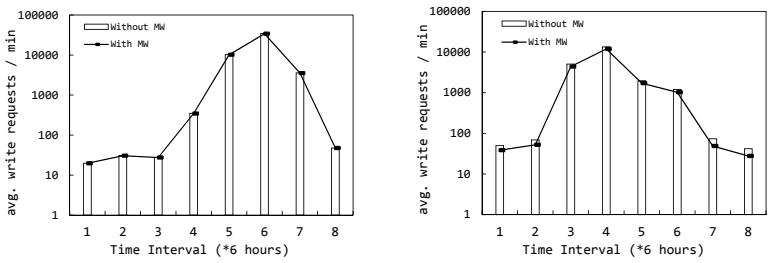


Fig. 17. The difference in write throughput with and without the middleware for application 1 (a) and application 2 (b). Although small differences exist, the middleware successfully provides a nearly equal write rate to all applications.

which has the highest write requests and application 2 which has the highest read requests for the aggregation scheme in use and it is very clear that, along with dutifully improving data retrieval efficiency, the middleware also maintains that similar justice be done to write tasks as well. There is only a negligible reduction in the number of write tasks performed per minute in both cases proving the abilities of the middleware.

7 Conclusion and Future Work

We present and evaluate the design for a cost efficient, hybrid, cloud based storage which mainly makes use of magnetic tapes as backend storage media. Although tapes have been widely categorised as a slow and unpopular storage media, it outperforms magnetic disks in total cost of ownership and energy consumption (tapes don't consume power when stored in a tape library), which makes tape technology an ideal choice for cloud based archiving services. We explore the benefits of the state of the art in tape storage technology. The need for a managerial middleware, which is a combination of algorithms and data distribution policies, that contributes in overcoming the latency offered by tapes in order to improve performance of IO processes is proposed and evaluated. The middleware serves its purpose and by improving data distribution efficiency and decreasing the overall response time for read requests. The test cases have been generated using the extensive analysis of live archiving workloads and modelling techniques.

One of the most exciting aspects of our work is the doors of opportunity it opens for new research. Understanding the economics of revisiting a legacy system to solve the data explosion problems of today requires an overhaul of nearly every piece of technology associated with the storage system. Future plans of the project include the improvement of the middleware and the filesystem to support message passing enabled, adaptive data weight management and IO paralellization. Another area of focus is the elaboration of operation of Tape Cloud for a variety of data types, application and magnitude of serviceability.

References

1. Seagate, "Video surveillance storage: How much is enough?"
2. "County of cameras: Cheshire constabulary aims to count every private camera in the county," *CCTV Image Online*.
3. M. Chamness, "Capacity forecasting in a backup storage environment," *Usenix LISA'11*.
4. J. Jackson, "Most network data sits untouched," *Government Computer News*, July 2008. [Online]. Available: http://gcn.com/Articles/2008/07/01/Most-network-data-sits-untouched.aspx
5. O. Sandst, O. S, S. A, and R. Midtstraum, "Improving the access time performance of serpentine tape drives," 1999.
6. I. Giurgiu, C. Castillo, A. Tantawi, and M. Steinder, "Enabling efficient placement of virtual infrastructures in the cloud," in *Proceedings of the 13th International Middleware Conference*. New York, NY, USA: Springer-Verlag New York, Inc.
7. A. Gulati, C. Kumar, and I. Ahmad, "Modeling workloads and devices for io load balancing in virtualized environments," *SIGMETRICS Perform. Eval. Rev.*
8. M. Raab and A. Steger, ""balls into bins" - a simple and tight analysis," in *Proceedings of the Second International Workshop on Randomization and Approximation Techniques in Computer Science*, ser. RANDOM '98. London, UK, UK: Springer-Verlag. [Online]. Available: http://dl.acm.org/citation.cfm?id=646975.711521
9. Y. Peres, K. Talwar, and U. Wieder, "The (1 +)-choice process and weighted balls-into-bins."
10. P. Berenbrink, T. Friedetzky, Z. Hu, and R. Martin, "On weighted balls-into-bins games," *Theor. Comput. Sci.*
11. "Fuse filesystem project." [Online]. Available: http://fuse.sourceforge.net/
12. I. Ahmad, "Easy and efficient disk i/o workload characterization in vmware esx server," in *Proceedings of the 2007 IEEE 10th International Symposium on Workload Characterization*, ser. IISWC '07. Washington, DC, USA: IEEE Computer Society.
13. N. Agrawal, W. J. Bolosky, J. R. Douceur, and J. R. Lorch, "A five-year study of file-system metadata," *Trans. Storage*.
14. J. R. Douceur and W. J. Bolosky, "A large-scale study of file-system contents," in *Proceedings of the 1999 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, ser. SIGMETRICS '99. New York, NY, USA: ACM.
15. S. Kavalanekar, B. Worthington, Q. Zhang, and V. Sharda, "Characterization of storage workload traces from production windows servers," in *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, 2008, pp. 119–128.
16. D. Lee, M. O'Sullivan, and C. Walker, "Benchmarking and modeling disk-based storage tiers for practical storage design," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, no. 2, pp. 113–118, Oct. 2012. [Online]. Available: http://doi.acm.org/10.1145/2381056.2381080
17. G. Wallace, F. Douglis, H. Qian, P. Shilane, S. Smaldone, M. Chamness, and W. Hsu, "Characteristics of backup workloads in production systems."
18. C. Delimitrou, S. Sankar, K. Vaid, and C. Kozyrakis, "Decoupling datacenter studies from access to large-scale applications: A modeling approach for storage workloads," in *Workload Characterization (IISWC), 2011 IEEE International Symposium on*.
19. "Vdbench." [Online]. Available: http://vdbench.sourceforge.net/