IO Tetris: Deep Storage Consolidation for the Cloud via Fine-grained Workload Analysis

Rui Zhang*, Ramani Routray*, David M. Eyers†, David Chambliss*, Prasenjit Sarkar*, Douglas Willcocks‡ and Peter Pietzuch‡
*IBM Research – Almaden, 650 Harry Road, San Jose, CA, USA. {ruizhang, routrayr, chamb, psarkar}@us.ibm.com
†University of Otago, Dunedin, New Zealand. dme@cs.otago.ac.nz
‡Imperial College London, United Kingdom. {dtw107, prp}@doc.ic.ac.uk

***Abstract*—Intelligent workload consolidation in storage systems leads to better Return On Investment (ROI), in terms of more efficient use of data center resources, better Quality of Service (QoS), and lower power consumption. This is particularly significant yet challenging in a cloud environment, in which a large set of different workloads multiplex on a shared, heterogeneous infrastructure. However, the increasing availability of fine-grained workload logging facilities allows better insights to be gained from workload profiles. As a consequence, consolidation can be done more deeply, according to a detailed understanding of how well given workloads mix.**

We describe *IO Tetris*, which takes a first look at fine-grained consolidation in large-scale storage systems by leveraging temporal patterns found in real-world I/O traces gathered from enterprise storage environments. The core functionality of IO Tetris consists of two stages. A *grouping stage* performs hierarchical grouping of storage workloads to find complementary groupings that consolidate well together over time and conflicting ones that do not. After that, a *migration stage* examines the discovered groupings to determine how to maximize resource utilization efficiency while minimizing migration costs. Experiments based on customer I/O traces from a high-end enterprise class IBM storage controller show that a non-trivial number of IO Tetris groupings exist in real-world storage workloads, and that these groupings can be leveraged to achieve better storage consolidation in a cloud setting.

I. Introduction

Cloud computing [1] is quickly gaining momentum, and many public cloud providers have sprung up to take advantage of this growing market. Return On Investment (ROI) and cost reduction are key incentives for adopting a utility-based cloud model. Specifically, *cloud storage* is an approach for networked data storage that allows consumers to offload their storage infrastructure onto the cloud.

Storage consolidation, also known as storage convergence, refers to the notion of centralizing and sharing back-end storage resources among front-end application servers (and their applications). Efficient solutions for storage consolidation are essential to boosting ROI because they not only provide hosted applications with better Quality of Service (QoS)—thus avoiding service level agreement (SLA) penalties—but also free up unused resources for new applications. This enables more customers to be accommodated without additional infrastructure.

Therefore, efficient consolidation becomes a key differentiator for cloud providers over their competitors. In a storage cloud provider environment, per transaction cost is around four times higher than the storage charge. Cost per transaction [2] involves the cost incurred due to read and write operations on the data whereas storage cost is the cost per gigabyte. Many storage cloud providers struggle to fend off competition in terms of cost per gigabyte, which cannot be reduced without relying on extremely challenging breakthroughs in disk drive technology. Consequently, smart storage consolidation can give them a definite edge in terms of cost per transaction.

Existing consolidation solutions are available in each individual layer of the IT stack, such as server migration tools [3], [4], virtual machines monitors that support migration [5], and storage provisioning technologies built into NAS and SAN offerings. Most of these schemes periodically migrate workloads based on the latest maximum or average workload intensity. For example, volumes in cloud storage are often placed first by administrators according to *a priori* workload requirements and estimates on data capacity and needs in terms of Input/Output Operations Per Second (IOPS) [6]. As workloads change due to dynamic behavior that cannot be foreseen, consolidation solutions re-sort the workloads with the goal of optimizing subsequent placements based on the new average or maximum IOPS [7].

The dynamic and bursty nature of cloud workloads often renders these aforementioned consolidation approaches overly conservative or unsuitable. For example, decisions based on *peak loads* would not combine workloads that have high peaks. However, in the cases where one's peaks always coincide with another workload's troughs, they should be combined. Figure 1 illustrates such a complementary pairing of workloads as found in customer data. In contrast, recommendations based on *average load* would risk combining two workloads with reoccurring, coincident high peaks, when they should be separated across resources. Figure 2 illustrates a conflicting storage workload pairing, also obtained from real-world customer data.

To overcome these weaknesses of existing consolidation approaches, we need a technique that is aware of the temporal shape of workloads, and only fits together shapes that jointly leverage the resource capacity over time. An analogy here is the computer game *Tetris*, in which the goal is to use various 2-D shapes to fill a rectangular container completely. For storage consolidation, the height of the Tetris container relates to workload intensity and the width of the container is the time

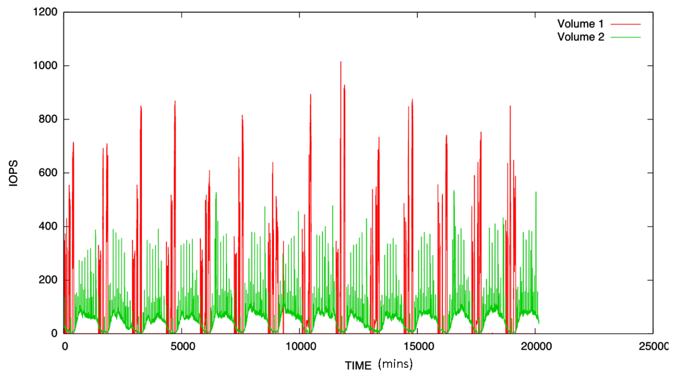

Fig. 1. A *complementary* Tetris volume pair in enterprise storage I/O data.

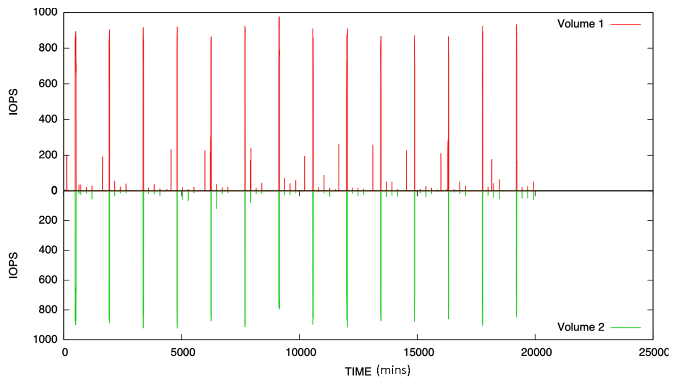

Fig. 2. A *conflicting* Tetris volume pair in enterprise storage I/O data.

period of analysis. We therefore refer to these complementary combinations of workloads as *Tetris pairings* (or patterns).

An observation is that many of the complementary workload patterns exhibited are not accidental, but rather caused by well-founded application- or infrastructure-level relationships within the cloud system. These include correlated batch jobs, pipelined jobs, application-level backups, and data mirroring operations. These inter-relationships can result in Tetris workload patterns across the cloud stack, which includes front-end servers, application servers and storage.

In this paper, we describe *IO Tetris*, a solution that leverages complementary Tetris patterns in storage workloads for more efficient consolidation of data volumes. IO Tetris uses a periodic optimizer that proposes data placement plans for consolidating data volumes onto cloud storage resources. It takes into account fine-grained I/O performance history, migration costs in terms of time and bandwidth and rules-of-thumb in the environment. While focusing on storage in this paper, many aspects of our approach are generally applicable to a variety of cloud resources.

A. Challenges and Contributions

There have been several recent research efforts that attempt to achieve greater consolidation efficiency, mostly for front-end servers, using temporal patterns [4], [8], but we are unaware of work that explicitly focuses on the storage layer. In addition, previous work attempts to efficiently detect interesting patterns, with less emphasis on developing an actual consolidation framework that makes changes based on these detected patterns.

We argue that more advanced optimization of the framework may prove especially important for storage workload consolidation, as more factors come into play than in the server workload space. Apart from the obvious requirements to detect Tetris patterns and leverage them in consolidation, several challenges arise, some of which are unique to storage. These are outlined in the following paragraphs.

Awareness of storage-specific bottlenecks. It is non-trivial to accurately find Tetris patterns in a large set of workloads that impact storage resource utilization and corresponding performance. A key requirement is awareness of the spatial aspect of the workload characteristics, in addition to the temporal aspect. For example, IOPS patterns do not necessarily translate into load patterns on disk arrays (the typical performance bottleneck that would benefit most from consolidation), as some or all of the IOPS may be handled by caches and never reach disks. Corresponding measures need to address this mismatch where present.

Trade-offs between benefit, migration costs and other constraints. It is tempting to always consolidate complementary Tetris workloads and separate conflicting Tetris workloads, especially when strong candidates are present. However, doing so may have implications on other important migration factors that cancel out the benefits. For example, aggressive consolidation may result in large time and bandwidth costs that are more significant when performing server consolidation. This is especially the case when the storage volumes to be consolidated contain large amounts of data. Some environments have "rule of thumb" requirements regarding the average or maximum load on a physical device that must not be violated, even when highly desirable Tetris patterns are present. Trade-offs between IO Tetris consolidation gains and costs regarding other migration factors must be considered and/or exposed to system administrators in a meaningful manner.

Scalability. Scalability is an important requirement of all cloud-related solutions. Non-trivial challenges exist in both performing the analysis to discover Tetris workloads and conducting migration to leverage the identified patterns. The complex analysis must scale with regard to both the number of workloads and the number of measurements per workload. Larger clusters of Tetris workloads should be identified where present, so that a single optimal migration plan may be generated by taking a holistic view at the level of the entire cluster, as opposed to many incremental, suboptimal plans for each pair within that cluster. The migration heuristic itself should also scale with the number of workloads.

In IO Tetris, we aim to address these challenges by optimizing Tetris pattern detection and providing a consolidation algorithm that is aware of the close interplay between storage resources, costs and their associated workloads. We make the following main contributions:

- *a consolidation-oriented analysis of fine-grained patterns in real-world enterprise storage I/O statistics.* We confirm not only the existence of conflicting and complementary temporal patterns but also the sensitivity to specific storage resources, underlying the causes of these patterns;
- *a hierarchical storage workload clustering method customized for Tetris patterns.* We define a lightweight similarity metric and sensible filtering to enhance the scalability of our method;
- *a scalable, novel migration algorithm that is aware of identified clusters of Tetris workloads and their benefits.* The algorithm leverages the clusters to generate actions plans more effectively for large environments. It also makes intelligent trade-offs between the benefits and other storage-specific factors, such as migration costs and device data capacities.

The rest of the paper is organized as follows. Section II sets the context of storage consolidation and analyzes enterprise workload behaviors. Our solution inspired by this analysis, IO Tetris, is outlined in Section III. Sections IV and V elaborate on the two key components of IO Tetris—clustering and migration—and present results based on a selected customer data set. Section VI reviews related work, while Section VII concludes and discusses future work.

II. Enterprise Storage Consolidation and Tetris Workload Patterns

An enterprise class, high-end storage subsystem typically has a highly scalable and flexible architecture and provides more than five-nines of availability. Each storage subsystem contains multiple virtualized components such as *extents*, *pools*, and *volumes* on a set of physical disks. A typical storage volume is created from a set of extents in an extent pool. An extent pool is a logical construct for managing sets of extents. Extents themselves are typically a gigabyte in size, and have a Fixed Block (FB) format.

A *disk array* is the typical logical workload container with capacity, has a RAID level and gets assigned to an extent pool. Based on the overall storage system hierarchy and configuration, a disk array has a specific capacity and allows for a given rate of IOPS. Storage consolidation concerns workload multiplexing at all levels of this hierarchical virtualized component stack. It has several objectives including:

- **Maximal resource utilization.** Resource components should be as highly utilized as possible over time.
- **Minimal performance penalties.** No component at any level in this hierarchical virtualized component stack should be overloaded at any point in time to avoid performance degradation and consequently SLA penalties.

In this paper, we focus our discussions on workload multiplexing through the placement of volumes on disk arrays, by examining fine-grained resource usage of workloads based on statistics collected from live customer deployments of high-end storage subsystems. The goal is to see if both opportunities and hazards beyond traditional approaches exist in real-world workloads with respect to the consolidation objectives above.

Our data collection process involved gathering overall configuration and performance metrics from multiple data centers of a large enterprise. Configuration regarding the storage infrastructure and its associated servers and network infrastructure was discovered and correlated. These data centers contained a total of 32 storage subsystems. We recorded the average IOPS per minute for each volume for a period of 15 days. The IOPS statistics were further broken down into sub-dimensions including reads vs. writes, cache hits vs. misses, and small vs. large block sizes. We focus our analysis on the data set from a customer application, which consisted of 246 volumes.

A. Tetris patterns

Simple visualization reveals that most workloads in the data set are highly bursty over time, exhibiting occasional and intense peaks and periods of idleness or extremely low activity. Thus a desired multiplexing behavior on resource R over time is achieved between two workloads x and y if x and y's peaks do not overlap. This is because the combination of x and y would reduce the overall resource idle time without worsening any peak load that R has to handle. By way of analogy to the Tetris game, these workloads have *complementary* temporal shapes that fill each others' "gaps". Conversely, two *conflicting* workloads whose peaks overlap cause undesired multiplexing behavior—they increase the peak load that the resource has to handle without reducing idle time.

We experimented with several metrics that could classify conflicting and complementary workload pairings, including Pearson correlation[1], mutual information[2] and a metric that compares the variance of the combined load of x and y with their respective loads. The pairing metrics were compared based on their respective precision[3] in identifying the top 20 complementary/conflicting pairings, which were hand-picked by domain experts. We found that Pearson correlation had considerably better precision (close to 100%) and is computationally less expensive than the other metrics (complexity $O(n)$ where n is the number of IOPS data points for a volume).

We applied the Pearson correlation formula to 20,160 IOPS measurements (per minute measurements over 15 days) of two volumes x and y. A low score (towards -1) indicates that x and y are highly complementary, whereas a high score (towards 1) indicates that x and y are highly conflicting. Based on domain experts' input aided by side-by-side visualization such as the ones shown in Figures 1 and 2, we set a threshold of -0.12 to consider two volumes to be complementary, and a score of 0.5 to indicate borderline conflict.

Any volumes exhibiting complementary and/or conflicting patterns (with one or more other partnering volumes) are potential candidates for Tetris-like consolidation optimization. A complementary pairing that does not co-locate on the same

[1] http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient
[2] http://en.wikipedia.org/wiki/Mutual_information
[3] http://en.wikipedia.org/wiki/Precision_and_recall

TABLE I
POTENTIAL FOR OPTIMIZATION IN SAMPLE CUSTOMER DATA.

	# of volumes
1 complementary pairing	8
Multiple complementary pairings	27
1 conflicting pairing	44
Multiple conflicting pairings	32

resource may be considered for co-location and yield a utilization gain; a conflicting pairing that is co-located should be considered for separation and can potentially lead to reduced peak load and fewer SLA violations on the shared resource.

The more candidates there are, the stronger the optimization potential because there is a higher likelihood that some of these candidate volumes would satisfy other general migration constraints (costs, not-to-move policies, etc.) and thus can be exploited. Table I illustrates the number of potential candidates for Tetris consolidation in our sample customer data set. Here, 35 volumes (out of 246) have the potential to form complementary combinations that should be co-located, and 76 volumes (out of 246) have the potential to form combinations that should be separated. Of these volumes, many have high potential in that they have multiple partnering volumes with which to form at least one feasible pairing.

B. Pattern sensitivity to resources

Disk arrays are the most common performance bottleneck for I/O processing in high-end storage servers. It is therefore important that performance objectives are met for this resource. Note that IOPS do not directly reflect the workload intensity on disk arrays, as some of the I/O requests are handled by caches and only cache misses are disk bound. Cache miss ratios depend on both the workload and the system configuration. As a result, Tetris patterns at the subsystem level (IOPS) may not necessarily translate to Tetris patterns at disk level (MISS).

In order to investigate the consistency of Tetris patterns across resources, we compute the Pearson correlation between the IOPS and MISS measurements of each volume. $\mathrm{corr}(IOPS, MISS) \rightarrow 1$ indicates minimal sensitivity (or maximal consistency), because the IOPS and MISS exhibit a linear relationship—any Tetris pattern found at the IOPS level is likely to translate to disk arrays. In contrast, $\mathrm{corr}(IOPS, MISS) \rightarrow 0$ implies maximal sensitivity (or minimal consistency) as Tetris patterns at the IOPS level do not translate to the disk level.

Figure 3 shows the results based on the customer data set. While over 50% of the volumes have little sensitivity (i.e. $\mathrm{corr}(IOPS, MISS) \rightarrow 1$), there are also several highly sensitive volumes. We explain how IO Tetris copes with this in the next section.

C. Summary

We conducted a similar stability analyses to the one in [8] and confirmed their results. Our data also exhibits stable daily patterns, but we do not elaborate on these findings due to the lack of space. Section VII summarizes our ongoing efforts that go beyond basic stability analysis.

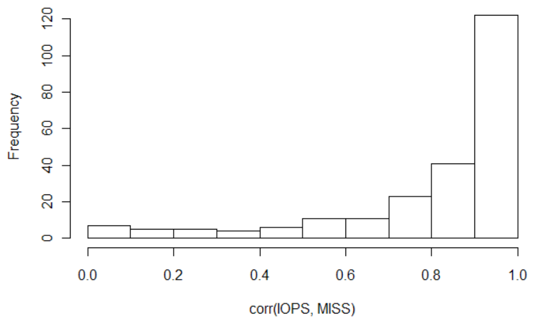

Fig. 3. Per volume IOPS and MISS correlation histogram.

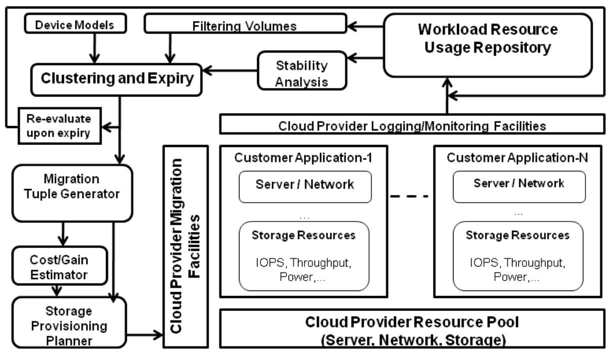
Device Models
Filtering Volumes
Workload Resource Usage Repository
Stability Analysis
Clustering and Expiry
Re-evaluate upon expiry
Cloud Provider Logging/Monitoring Facilities
Migration Tuple Generator
Cost/Gain Estimator
Storage Provisioning Planner
Cloud Provider Migration Facilities
Customer Application-1
Server / Network
Storage Resources
IOPS, Throughput, Power,...
Customer Application-N
Server / Network
Storage Resources
IOPS, Throughput, Power,...
Cloud Provider Resource Pool (Server, Network, Storage)

Fig. 4. IO Tetris in a cloud data center.

In summary, we have found significant potential for consolidation optimization based on fine-grained Tetris-like workload patterns in real-world data. These patterns are sensitive to cloud storage resources, but otherwise exhibit long-term stability. The following sections describe our proposed framework to leverage this potential.

III. IO TETRIS OVERVIEW

In Figure 4, we show the components of the IO Tetris framework and how it interacts with the cloud environment. We assume that a set of applications is hosted in the cloud. Each application has been allocated specific resources in terms of server, storage and network based on SLAs.

IO Tetris assumes that the cloud provider is equipped with fine-grained I/O logging facilities, which are able to capture I/O activities in time and in I/O sub-dimensions, such as read vs. write and/or random vs. sequential, cache hits vs. miss. Enterprise storage subsystems typically have instrumentation in place that allows system management suites to query performance information from the managed device via proprietary or standard (CIM/SMI-S)[4] interfaces. For example, the IBM Tivoli Storage Productivity Center (TPC)[5] product allows storage administrators to set performance monitors

[4]http://www.snia.org
[5]http://www-03.ibm.com/systems/storage/software/center/

on managed storage subsystems. TPC queries performance metrics from the managed subsystems at specified periodic intervals for reporting, trending, planning, and so on.

Given the accurate knowledge of the cloud environment, IO Tetris takes all the storage workload information as input. Some volumes with near-zero workload or those explicitly dictated by policy (e.g. a do-not-disturb list of volumes specified by administrators) are filtered out. IO Tetris computes temporal correlation and lists n top pairs in terms of the descending order of complementary correlation. This information, in conjunction with the device white-box models, are used to create groups of volumes that are complementary and the groups that are conflicting. Each group gets tagged with an expiry date based on the stability and sustenance of the pairing. Migration costs and gains in terms of consolidation, performance improvement and component utilization are determined to decide on a migration strategy. Based on the expiry date, these clusters are reevaluated for their existence. I/O Tetris then relies on the cloud providers' migration facilities to enforce its plans.

More specifically, IO Tetris consists of two main workflows:

1. Workload clustering workflow. This workflow mainly retrieves recent IOPS history, performs Pearson correlation on each pair of data volumes' IOPS time series, and conducts a clustering step using the correlation values as the basis for similarity. Storage device models are integrated into the clustering step so that they are taken into account in the resulting Tetris clusters.

2. Migration planning workflow. This workflow uses the complementary and conflicting Tetris clusters to generate migration tuples and specifies the source and destination of all volumes to be moved. The costs and gains of executing these tuples are estimated and input into the planner, which formulates a final migration plan. Note the use of the migration workflow is optional. Administrators with their migration preferences may choose to formulate migration plans manually or using tools specific to their environments, while leveraging the quantification and visualization provided by IO Tetris clustering.

These workflows are re-executed periodically when one suspects existing Tetris patterns might have expired. Additional analysis may be done to evaluate the stability of the IOPS time series and to quantify a window of expiration automatically.

IV. IO Tetris Clustering

Next we describe how IO Tetris workload clusters are found given IOPS statistics similar to those in the previous section (i.e. per volume per minute). Data volumes are first filtered to ensure clustering is only performed on worthwhile candidates. We then use hierarchical clustering [9] to find complementary and conflicting clusters of volumes by customizing the clustering distance metric and linkage method.

A. Filtering volumes

Hierarchical clustering is a relatively expensive process with $O(n^2 \log n)$ complexity, where n is the number of volumes. Reducing the size of n improves the scalability of IO Tetris, without compromising the quality of the results (see Section VII for trade-offs between quality and scalability). Based on the discussions in Section II, we remove the following types of volumes from IO Tetris:

1) volumes with very few I/O activities across all measured times (e.g. peaking at 10 to 20 IOPS);
2) volumes with very stable I/O activities because they lack the variance to form Tetris patterns;
3) volumes marked untouchable by administrators;
4) volumes with $\mathrm{corr}(\mathit{IOPS}, \mathit{MISS}) < \mathit{threshold}$. Although one could apply IO Tetris clustering on MISS statistics for these volumes instead, we do not recommend it. MISS statistics can be highly dependent on the dynamics of the entire environment, and may not yield long-term complementary and conflicting clusters.

B. Hierarchical clustering

Hierarchical clustering incrementally groups objects that are similar, i.e., objects that are close to each other in terms of distance (as defined per domain), $\mathit{distance} \to 0$.

$$\frac{1-\mathrm{corr}(x,y)}{2} \quad (1) \qquad\qquad \frac{1+\mathrm{corr}(x,y)}{2} \quad (2)$$

We define two distance functions based on the correlation between any two volumes x and y. Equations 1 and 2 are the distance functions for finding conflicting and complementary Tetris clusters, respectively. $\mathrm{corr}(x, y)$ is the correlation between the IOPS time series of volumes x and y over time. Equation 1 results in conflicting clusters because it ensures that the distance between x and y is 0 when they are correlated (i.e. completely conflicting); and is 1 when they are anti-correlated (completely complementary). As a result, complementary volumes are grouped. Equation 1 groups complementary volumes in a similar manner, even though visually they are not linear/similar.

Figure 6 depicts the complementary and conflicting clusters for our sample customer data set. A low horizontal distance between any volumes or sub-clusters indicates a high degree of conflict/complementariness between them, with the exact degree marked by the height of the lowest shared sub-branch. The cluster dendrograms provide effective visualization of all "hot spots" of complementary/conflicting volumes' subsets, which are easily distinguishable from the rest through their low heights.

This visualization allows migration administrators to act quickly on high potential volumes whilst maintaining a holistic view. For instance, if sub-cluster C_1 is located on the same resource R_1 and C_2 shares resource R_2, a simple yet highly effective optimization would be to swap some of their composing volumes. This ensures that the original conflicting clusters are broken and new ones do not form because C_1 and C_2 have a large horizontal distance.

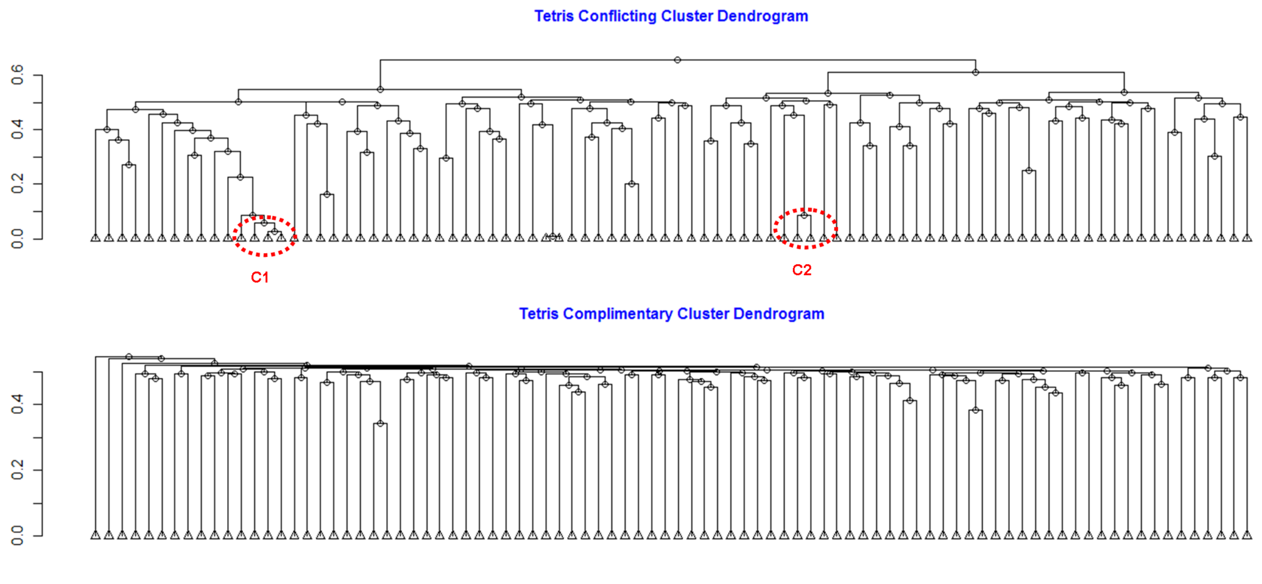

Fig. 5. IO Tetris hierarchical clusters for customer B.

C. *Choice of clustering algorithm*

We considered a total of nine clustering techniques [10] experimentally, including well-know methods such as agglomerative/divisive hierarchical clustering and k-means/k-mediods. We chose hierarchical clustering due to its superior cluster quality and the following practical advantages:

- Along with K-mediods and fuzzy clustering, it is one of few algorithms that can operate with customized distrance metrics such as the Tetris distances defined in Equations 1 and 2.
- It provides dendrogram as an effective visualization of high-dimensional data.
- Unlike methods such as K-means/K-mediods and model-based clustering, hierarchical clustering does not require expert input to bootstrap.

The cluster quality was measured by the Dunn score [10], which favours cleanly separated clusters and low intra-cluster distance. On the sample data set, hierarchical clustering yields more consistent and higher (thus better) Dunn scores than both K-mediods and fuzzy clustering. When used to identify 20 complementary clusters, resulting in 2–3 volumes per cluster on average, hierarchical clustering has a Dunn score of 0.55, as opposed to 0.36 for K-mediods and 0.46 for fuzzy clustering.

V. IO Tetris Migration

IO Tetris migration co-locates storage volumes within a disk array based on complementary or conflicting groupings produced via the Tetris clustering method. To this end, migration plans are created and orchestrated to achieve the desired placement of storage volumes based on IO Tetris clusters. This process involves transferring data between storage types and storage formats. According to a recent study,[6] data migration costs \$5000 per terabyte and 25% of data is moved each year in data centers.

A. *Migration decisions*

IO Tetris migration seeks to make the desired trade-offs between leveraging IO Tetris clusters to achieve resource usage efficiency and limiting migration costs. It calculates migration cost based on the following factors:

Offline vs. online: Offline migration incurs application downtime as opposed to online migration, which is transparent to applications.

Inter vs. intra storage subsystem migration: Inter subsystem migration is done by copying the data to a different storage volume in a different storage subsystem of the same type and synchronizing their states (e.g. via IBM Metro Mirror or IBM Global Mirror). In contrast, in intra subsystem migration, data is copied to a different storage volume in the same subsystem (e.g. via IBM Point-in-Time FlashCopy).

Inter vs. intra data center migration: Inter data center migration involves migrating data from one storage volume to another across data centers as compared to intra, where data is migrated internally inside a data center without much concern regarding bandwidth and delay.

Homogeneous storage vs. heterogeneous storage: Migration across two storage subsystems that are of same type, model or vendor is considered homogeneous as opposed to heterogeneous, where a virtualization device (e.g. IBM SAN Volume Controller SVC[7]) might have to be introduced to migrate data between subsystems.

More formally, the IO Tetris algorithm operates over a set of storage subsystem tuples $S \subseteq \mathcal{C}$. The set of all possible

[6]http://www.incipient.com

[7]http://www-03.ibm.com/systems/storage/software/virtualization/svc/

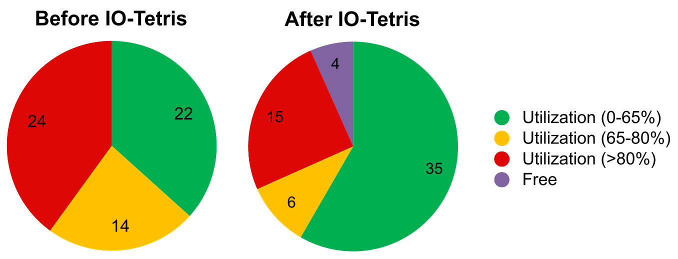

Fig. 6. Disk array utilization before and after Tetris migration.

tuples $\mathcal{C}$ has the following structure:[8]

$$\begin{aligned}\mathcal{C} = \ & \mathsf{Type} \times \mathsf{Vendor} \times \mathsf{Model} \times \mathsf{Location} \times \\ & \mathsf{Capability} \times \mathbb{P}(\mathsf{StorageVolume}) \times \mathsf{DiskArray}\end{aligned}$$

The above facets of migration costs can be determined by considering independent parts of the overall $\mathcal{C}$ space. Whether systems are homogeneous or heterogeneous can be determined by examining $\mathsf{Type} \times \mathsf{Vendor} \times \mathsf{Model}$. $\mathsf{Capability}$ states whether migration can be performed offline or online. $\mathsf{Location}$ controls if migrations are inter- or intra data center.

The algorithm should optimize required capacity and required bandwidth in terms of IOPS. We define two functions to quantify utilization: $c : S \to \mathbb{Q}$ for capacity and $b : S \to \mathbb{Q}$ for bandwidth. We apply the subscripts $c_{\max}$ and $b_{\max}$ to denote functions that provide the upper bounds of these metrics for a particular configuration.

Given a set of configurations $S \subseteq C$, the output of the IO Tetris algorithm is a tuple (P, Q) for $P \in \mathbb{P}(\mathcal{C})$ and $Q \in \mathbb{P}(\mathcal{C})$. The sets of complementary configurations in P are those from S that should be packed together if possible. The sets of configurations in Q are those from S that should not be placed together, and should instead be distributed across storage infrastructure due to conflict.

In terms of the above categorization, administrators generally prefer online, intra-subsystem migrations because they do not introduce application downtime. Homogeneous inter-subsystem migrations inside a data center are considered as the next best option. Data movement across heterogeneous subsystems and across data centers is regarded as comparatively costly.

Migrations are performed as follows. For each complementary cluster, IO Tetris determines if the majority of the storage volumes are already on a common disk array, making it the target disk array for the cluster. Otherwise, a new destination disk array is chosen.

IO Tetris avoids the formation of conflicting clusters by considering pre-existing storage volumes on the disk array. Decisions as to which storage volumes are to be evicted, and what their destination disk array will be are made jointly.

B. Evaluation

Figure 6 presents the initial placement of storage volumes over 60 disk arrays, alongside the placement after multiple iterations of IO Tetris consolidation. The number of disk arrays together with their component level percentage utilization are shown. Consolidation is performed progressively by examining associated migration costs.

The first iteration of IO Tetris resulted in placement of all good clusters collocated on disk arrays. 21 disk arrays would contain four volumes based on this recommendation. In total, 23 migration operations are required to accept the first iteration of IO Tetris. Subsequent consolidation iterations of bin packing can be done to pack storage volumes more tightly. Each disk array is considered as a bin. Complementary clusters are kept untouched but consolidation is done based on capacity and IOPS capability of disk arrays while avoiding conflicting clusters. However, doing this increases the number of migration operations. The results shown in Figure 6 were achieved through a total of 27 migrations in three iterations.

VI. Related Work

The proliferation of Cloud computing has led to much industrial and academic interest, and numerous related research projects. Workload consolidation has been studied at various levels of infrastructure and time granularities.

In terms of coarse-grained storage workload consolidation, Zhang et al. [11] examine balancing workloads between SSD and spinning platter media. Their ADLAM algorithm does adaptive lookahead to best utilize the fast but scarce and expensive SSD media, while taking into account migration costs, which makes their work complementary to ours. In contrast, Mazzucco et al. [12] improve power efficiency by intelligently switching off servers, considering more coarse-grained time intervals than our approach.

Gupta et al. introduce the MIRAGE provisioning system [7] for SAN management, which maximizes SAN efficiency by providing a modular analytical engine that smoothes the resource usage of storage system components. In contrast to our work, MIRAGE looks at specific storage system metrics at particular times, and reconfigures iteratively. In addition, Gopisetty et al. examine IBM's Provisioning and Disaster Recovery planners [6], which assist administrators with placement and resiliency decisions when introducing new applications into a data center. Their review provides a good basis on which our work builds.

In terms of overall service migration, Zheng et al. [3] explore wide-area VM migration in a multi-cloud environment by considering temporal and spatial locality in a workload. Their approach could be informed by Tetris pattern data. In related work, Ramakrishnan et al. [13] employ a combination of server virtualization, network functions and storage replication to move services with tight service-level deadlines. Their cooperative orchestration of migration decisions could be combined with our approach.

A number of research projects explore fine-grained server workload consolidation. The PAC system [4] uses signal processing techniques, namely Fast Fourier transforms with dynamic time warping, to form signatures of periodic server behavior. RUBiS, Hadoop and IBM System S workloads were

[8]We use $\mathbb{P}$ to denote the power set, i.e. the set of subsets of a set.

used to demonstrate that periodic server behaviors really occur, and that by discovering them, prediction errors about coarse-grained future resource requirements (i.e. average and peak loads) are reduced by 50–90%. PAC considers CPU, memory and I/O throughput when making dynamic VM placement decisions. While its goals are similar to those presented here, the migration and resource measurement are tied to virtual machine infrastructure, and does not directly generalize to hierarchical multi-layer architectures. The implicit structure determined by PAC would complement our more semantically rich descriptions of resources and their migration costs.

In [8], server consolidation technology is employed to reduce energy costs in data centers. The approach aggregates at near off-peak levels while restricting the performance risks of consolidation. A key contribution of this work is a thorough study of enterprise server workloads with respect to the potential medium (semi-static) or long term (static) consolidation paths available. In contrast to our work, the authors do not explicitly consider multi-level infrastructures and do not analyze migration costs.

Finally, Mishra et. al. [14] use clustering (k-means) to identify a small set of classes for Google cluster jobs with similar resource consumption patterns, in terms of execution duration, the number of CPU cores used and memory consumption. While their high level objective is similar to IO Tetris, their clustering approach does not consider fine-grained temporal resource usage behaviors and how well they multiplex, and may miss out on significant opportunities to achieve greater consolidation efficiency.

VII. Conclusions

This paper presented IO Tetris, an initial effort that investigates fine-grained storage workload behaviors in real-world customer environments and leverages these to achieve more efficient storage consolidation. Our study has found workload combinations that are both complementary and conflicting, which would boost and undermine consolidation objectives, respectively. Based on these observations, we have developed a framework to identify these patterns in clusters and migrate storage volumes accordingly to keep complementary clusters together and conflicting clusters apart. We have demonstrated the applicability of our IO Tetris approach to real-world, large-scale storage consolidation tasks. We are actively pursuing the following future work:

Massive-scale IO Tetris. Support for large or even massive storage cloud deployments is essential for the practicality of IO Tetris. To further improve scalability over the number of volumes, we are investigating more scalable clustering mechanisms [15]. We also plan to explore sampling methods that require fewer data points, going beyond the current use of computationally inexpensive correlation metrics.

3-D IO Tetris. While Section II-B has touched on the issue of resource-bound behavior, we believe that a deeper analysis can systematically reveal consolidation opportunities in the spatial/resource dimension. This additional dimension could yield 3-D (load vs. resource vs. time) Tetris patterns compared to the predominantly 2-D (load vs. time) behaviors discussed in this paper. Based on collected block-level I/O traces, we want to understand the address ranges and resources of I/O accesses.

Stable IO Tetris. We want to investigate stability windows, within which IO Tetris patterns persist with at least a certain probability. Such predictions can be done both along the frequency plane by forecasting primarily frequency changes, and along the original time plane by simply predicting the future workload at each point in a time window [16].

Live IO Tetris. We are working on integrating IO Tetris with IBM migration frameworks and on evaluating it as part of larger-scale, live deployments.

References

[1] M. Armbrust, A. Fox, R. Griffith *et al.*, “Above the Clouds: A Berkeley View of Cloud Computing,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[2] “Amazon Simple Storage Service S3,” Amazon, 2010. [Online]. Available: http://aws.amazon.com/s3/

[3] J. Zheng, T. S. E. Ng, and K. Sripanidkulchai, “Workload-Aware Live Storage Migration for Clouds,” in *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE)*, 2011.

[4] Z. Gong and X. Gu, “PAC: Pattern-driven Application Consolidation for Efficient Cloud Computing,” in *MASCOTS*, 2010.

[5] “VMWare vMotion,” VMWare, 2010. [Online]. Available: http://www.vmware.com

[6] S. Gopisetty, E. Butler, S. Jaquet, M. Korupolu, T. K. Nayak, R. Routray, M. Seaman, A. Singh, C.-H. Tan, S. Uttamchandani, and A. Verma, “Automated Planners for Storage Provisioning and Disaster Recovery,” *IBM J. Res. Dev.*, vol. 52, July 2008.

[7] K. Gupta, P. Sarkar, and L. Mbogo, “MIRAGE: Storage Provisioning in Large Data Centers using Balanced Component Utilizations,” *SIGOPS Oper. Syst. Rev.*, vol. 42, January 2008.

[8] A. Verma, G. Dasgupta, T. K. Nayak, P. De, and R. Kothari, “Server Workload Analysis for Power Minimization using Consolidation,” in *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)*, Berkeley, CA, USA, 2009.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, October 2007.

[10] G. Brock, V. Pihur, S. Datta, and S. Datta, “clvalid: An r package for cluster,” *Journal of Statistical Software*, 2008.

[11] G. Zhang, L. Chiu, and L. Liu, “Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment,” in *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Washington, DC, USA, 2010.

[12] M. Mazzucco, D. Dyachuk, and R. Deters, “Maximizing Cloud Providers’ Revenues via Energy Aware Allocation Policies,” in *Proceedings of the IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Washington, DC, USA, 2010.

[13] K. K. Ramakrishnan, P. Shenoy, and J. Van der Merwe, “Live Data Center Migration across WANs: A Robust Cooperative Context-aware Approach,” in *Proceedings of the SIGCOMM Workshop on Internet Network Management (INM)*, New York, NY, USA, 2007.

[14] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, “Towards characterizing cloud backend workloads: insights from google compute clusters,” *SIGMETRICS Perform. Eval. Rev.*, vol. 37, pp. 34–41, March 2010.

[15] B. C. Fung, K. Wang, and M. Ester, “Hierarchical Document Clustering Using Frequent Itemsets,” in *Proceedings of the Siam International Conference on Daa Mining (SDM)*, 2003.

[16] P. Saripalli, G. Kiran, R. S. R, H. Narware, and N. Bindal, “Load Prediction and Hot Spot Detection Models for Autonomic Cloud Computing,” in *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (HotICE)*, 2011.