HetStore: A Platform for IO Workload assignment in a heterogeneous storage environment

by

Vishrut Mehta, kumar rishabh, Tushant Jha, Vasudeva Varma

in

The Fifth IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)

Bangalore, India

Report No: IIIT/TR/2016/-1

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
October 2016
HetStore: A Platform for IO Workload assignment in a heterogeneous storage environment

Kumar Rishabh
International Institute of Information Technology, Hyderabad
kumar.rishabh@research.iiit.ac.in

Tushant Jha
International Institute of Information Technology, Hyderabad
tushant.jha@research.iiit.ac.in

Vishrut Mehta
International Institute of Information Technology, Hyderabad
vishrut.mehta@research.iiit.ac.in

Vasudeva Varma
International Institute of Information Technology, Hyderabad
vv@iiit.ac.in

Abstract—The problem of providing optimal assignment for backend storage is a central problem in the design of cloud systems. It has taken a further central role as a result of growing heterogeneity from emerging Software Defined Storage systems. In this paper, we propose a solution to optimal IO Workload assignment using statistical modelling to estimate measures of performance such as Throughput, IOPS, et al. The proposed system uses support vector regression to estimate the performance of individual IO Workloads on each available SDS system for optimal assignment.

As a proof of concept, we demonstrate our solution in a heterogeneous environment comprising of HDFS, GlusterFS, and Ceph. We first show the accuracy of estimation of throughput and IOPS with values of coefficient of determination over 0.65 in all cases. We further show the analysis of using this regression model to classify workloads to respective SDS backend that will maximize throughput.

I. INTRODUCTION

The advent of huge data centers around the world has been marked by gain in prominence of Virtualization Technology. In particular the virtualization of compute/storage has brought access to these resources to end user much cheaper than ever before. Any cloud framework has a set of resources like compute/storage servers that is shared among a number of tenants. The tenants generally interact with the cloud by submitting workloads which require a slice of the compute and storage resources available. All these systems are generally cheap commodity servers which like any distributed system are susceptible to failure. In particular we would want a storage backend to have characteristic features like fault tolerance, replication and deduplication to overcome the underlying hardware failure. A recent trend that is gaining immense popularity to tackle this problem is Software Defined Storage (SDS) [14]. Software Defined Storage frameworks typically decouple the task of managing and provisioning data from the underlying hardware.

There are a number of vendor specific solutions like EMC ViPR[1], NetApp ONTAP[7], IBM Virtual Storage[6] and Opensource Solutions like Ceph[2], GlusterFS[5], HDFS[4], Swift[8] available in the market. Each SDS can be characterised by its strengths and weaknesses owing to the underlying algorithm taking the decisions. For instance, our experiments show Ceph performs better than HDFS when the average file size is small (like a transactional log workload) though HDFS tends to perform better when the average workload size is large[11]. Also the classes of workloads can be pretty diverse. For instance, the workload can be transactional characterised by small files with high iops or photo stream characterised by mid range file size with average iops. A workload can be hybrid of different workload classes too. For instance, a web server may serve photos to users and at the same time have logging daemon running in background.

Hence it makes sense for any cloud vendor to provide user with a heterogeneous storage environment which leverages the strength of various off-the-shelf SDS backends to serve diverse classes of IO workloads. But taking decision where to provision a workload may prove to be a difficult task for a Cloud Administrator. Various approaches have been tried to make the platform workload aware which include policy based provisioning[16], data mining approach[17] and empirical analysis[18]. To the best of our knowledge this is the first work which takes an analytical approach for classifying workload in heterogeneous SDS environment.

Using SDS environment instead of traditional storage to provision IO workloads has immense benefit to both users and Cloud Admins. From the Cloud Admin perspective it makes the management of Storage Cloud easier. All the features like replication, fault tolerance and deduplication are available in Software domain and hence does not require any special hardware. This brings down the cost drastically and User can get same premium features at a fraction of cost. The benefits can be gauged by the fact that even a data intensive organization like Cern chose to go with Ceph to build its data storage. It manages the Petabytes of experimental data generated from the most important experiment done by the human race over a Ceph installation[13].

In this paper we present HetStore which is a unified platform for better provisioning of IO workloads over a heterogeneous SDS environment. It does this by using statistical models for classification of an incoming workload to one of the underlying SDS. It has three major components namely the Controller, the SDS cluster and the HetStore Client.
The **Controller** consists of two major components namely the Storage Selection Module (SSM) which models an incoming workload and assigns it to one of the SDS attached. The other module is the Cluster Resource Monitor (CRM) which keeps track of all the metadata pertaining to users and IO Workload and also monitors the running clusters. The **SDS Cluster** is the collection of all the SDS Clusters attached to the platform. The **HetStore Client** consists of light weight daemons that run on each of user machines which are responsible for message parsing and internal communication.

For the purpose of our experiment we have decided to build an SDS environment with Hadoop, Ceph and GlusterFS as SDS backend.

We evaluate our platform in two ways:

- **Accuracy of statistical regression**
  After generating the synthetic training set, we evaluate our estimator by cross validating the evaluated training sample (i.e $R^2$ score denoting the coefficient of Determination[19]).

- **Comparative efficiency of Assignment**
  We further demonstrate the accuracy of using these regression models for assigning workloads in order to maximize throughput.

The rest of the paper is structured as follows - In Section 2, we talk about similar solution pertaining to characterization of IO workloads. In Section 3, we explain the overall HetStore architecture along with the implementations details, such as APIs and parameters. In Section 4, we elaborate on the underlying algorithm of the SSM. In section 5, we validate our prediction accuracy and assign the workload to a cluster. Section 6 concludes the paper and showcases the vision for the project.

### II. RELATED WORK

Albrecht *et al.* [16] defines policy based approach for workload partitioning in a tiered cloud dfs between flash and disk storage. Various policy based approaches like LRU, FIFO are used to move workloads in and out of flash tier to disk tier. Seo *et al.* [17] take data mining route for characterizing IO workload in NAND based SSD environment. They define IO workload classes based on various data mining approaches which could be then exploited by the storage devices such as SSDs.

Kavalanekar *et al.* [15] carry out trace analysis of traces obtained from production servers to develop characterization of traces like block level statistics, multi-parameter distribution etc.

Tremblay *et al.* [18] take empirical approach for workload assignment in heterogeneous SDS environment. They define a workload aware storage platform which takes a set of pre-defined rules for workload assignment in a heterogeneous storage environment.

The gain in prominence SDS backends provides an opportunity to leverage heterogeneous environment to better provision IO Workloads. In this paper we try to exploit underlying algorithms of each of these backends using statistical analysis to provide a unified solution which performs better on various metrics the job of classification of IO Workload.

### III. ARCHITECTURE AND IMPLEMENTATION

![HetStore Architecture](image)

**Fig. 1: HetStore Architecture**

In our architecture, **SDS clusters** consists of three different SDS backend - Ceph, HDFS and GlusterFS. We have described the configuration of our test clusters in the Evaluation section. HetStore exposes a plug-and-play feature for adding any new SDS backend. We just need to register the new storage gateway with the controller. The controller would then run a set of synthetic workloads offline to develop an analytical profile for the workload. This model would then be integrated into the existing architecture and would then start provisioning online workloads.

The **HetStore Client** consists light weight daemons that run on each of user machines. They direct any new incoming IO workload request to the controller which replies back with the...
assigned SDS Cluster. The HetStore Client then directs the subsequent IO requests to the assigned cluster.

As shown in Fig 1, in Step 1, the users submit a API request to store a workload. The request is submitted to the Controller layer. The CRM logs all the input requests submitted by the user. The SSM stores all the data retrieved after running regression on the training set for all the storage clusters. This part is explained briefly in the SSM Algorithm section. So, when SSM receives the input request, it asks CRM for current network metric and the resource allocation of the clusters. It then decides in which cluster this test workload will be stored based on the prediction of Throughput and IOPS. It send back a response which contains the information of the cluster like IP address and type of the cluster which is Step 2. After receiving the response, the data is transferred through a gateway from the user side to the destination cluster as shown in Step 3. One thing to note, data is transferred directly from user to the cluster instead of going through the control layer which would have added to the delay.

We have created two APIs, Workload API and Storage Cluster API. Storage Cluster API is only for the administrator, as he can decide whether to add/delete a cluster. The end user can communicate with the system by submitting a workload through the Workload API. The user would have to submit a HTTP request including a json text or file consisting data about the whole workload which is to be simulated. The parameters needed would be absolute location of the file, size of the file and the type of IO operation. The format to simulate a workload is as follows:

```json

workloads: {
    id:1
    fileSize:120k
    filePath: '<absolute path of the file>
    IO: read
},
{
    id:2
    fileSize:5120k
    filePath: '<absolute path of the file>
    IO: write
}

```

Similarly, we have created the API for cluster management and monitoring. The administrator could create a new cluster or update an existing cluster by adding machines to that cluster. Also, the administrator can know the status of the cluster like memory utilization, health, etc. The API makes HTTP requests to the controller which would attach/detach/update a cluster as per the request submitted by the administrator. The parameters needed for this API are name, IP address of the storage gateway and type of storage cluster. The format to create a cluster is as follows:

```json

name: ceph1/hdfs1/glusterFS1
storage gateway IP: 192.168.x.x

type: Ceph/HDFS/GlusterFS
```

The following table shows the API we have exposed to the User/Admin:

<table>
<thead>
<tr>
<th>API Group</th>
<th>HTTP verb</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Cluster</td>
<td>POST</td>
<td>Create a cluster</td>
</tr>
<tr>
<td>Storage Cluster</td>
<td>PUT</td>
<td>Add machines to existing storage cluster</td>
</tr>
<tr>
<td>Storage Cluster</td>
<td>GET</td>
<td>List cluster(s)</td>
</tr>
<tr>
<td>Storage Cluster</td>
<td>DELETE</td>
<td>Destroy a cluster</td>
</tr>
<tr>
<td>Workload</td>
<td>POST</td>
<td>Run a workload</td>
</tr>
<tr>
<td>Workload</td>
<td>GET</td>
<td>List workloads(s)</td>
</tr>
<tr>
<td>Workload</td>
<td>DELETE</td>
<td>Stop a workload</td>
</tr>
</tbody>
</table>

### TABLE I: API Description

**IV. SSM Algorithm**

The Storage Selection Module (SSM) allocates incoming workloads to clusters based on a statistical model that estimates Throughput and IOPS. This decision is taken on the basis of a preference criterion, encoded in a cost function, which we assume to be totally dependent on Throughput and IOPS. The module achieves this by maintaining Support Vector Regression[12] models, which predicts the likely value of throughput or IOPS, respectively, from a priori known features, for every cluster in the ecosystem. Since, our system leaves open the scope for any cost function dependent on throughput and IOPS, we argue that the accuracy of this system will be directly dependent on the accuracy of the regression models.

It should be noted that the list of critical measurements can be extended to performance metrics beyond Throughput and IOPS. However, for the purposes of this paper we have focussed on these two.

**A. Feature Vectors**

We select six a priori known features of workloads, ie. properties that can be directly measured when a workload is received, which have a high influence on the parameters whose estimation we seek, ie. throughput and IOPS. For instance, the *Number of files* can be used to determine the granularity of the workload, and is therefore an important characteristic in determining the allocation. This can be observed as for workloads with small files, more granular workloads perform better on Ceph against HDFS for both throughput and IOPS.

Similarly, *Read percentage* and *Write percentage* play an important role in the throughput and IOPS that a workload will exhibit on a system.

Accounting for the impact of size on performance is done by including *Total workload size* and *Median of file sizes*. And we also include *Concurrency*, or the number of concurrent requests sent as independent blocks, in order to account for the varying levels of performance against concurrency of the underlying algorithms.
B. Allocation Method

Let $C = \{c_1, c_2, \ldots\}$ be the list of clusters in the ecosystem, and $\mathcal{M} = \{m_1, m_2\} = \{\text{throughput}, \text{IOPS}\}$ be the set of critical measurements, while $W$ denotes the feature space of workloads. Corresponding to every measurement $m_j$, we can interpret a function $m_j^*: W \times C \to \mathbb{R}^+$ as denoting, for a feature vector $w \in W$ and cluster $c_i$, the average value of measurement when a workload with feature vector $w$ is run on the cluster $c_i$.

**Definition IV.1.** A cost function is defined as a function $\wp : W \times C \to \mathbb{R}^+$ that encodes the preference criterion of the allocation, as the expected cost that would be beared if a workload was executed on a particular cluster.

For the cost function $\wp$, the objective of allocating workload $w$ can be defined as finding cluster $c_i$ such $\wp(w, c_i)$ is minimized, i.e. determining $\arg\min_{c_i \in C} \wp(w, c_i)$.

And therefore the assumption that the cost function is entirely dependent on critical measurements translates to saying that for some non-polynomial algebraic functions $f$, $\wp(w, c_i) = f(m_1^*(w, c_i), m_2^*(w, c_i), \ldots)$. For most polynomial, or even some other algebraic $f$, the accuracy of computing or estimating $\wp(w, c_i)$ is trivially dependent on the accuracy of estimating $m_j^*(w, c_i)$, for all $j$.

Therefore, for each cluster $c_i$ and critical measurement $m_j$, we establish a support vector machine (SVM) $s_{ij}$. The support vector regression model $s_{ij}$ represents a function from feature space $W$ to the estimation of the expected value of measurement $m_j$, when corresponding workload is executed on $c_i$. Following is the pseudocode for the allocation method.

**Algorithm 1 Storage Selection Module**

1: procedure ALLOCATE (Workload $w$)
2:     for $c_i$ in Clusters\_List do
3:         for $m_j$ in Measurements\_List do
4:             estimation $\leftarrow s_{ij}(w)$
5:         end for
6:     test\_cost(i) $=$ $f$(estimation)
7: end for
8: if index of min(est\_cost) is $x$ then
9:     send Workload $w$ to $c_x$
10: end if
11: end procedure

V. Evaluation

A. Test Cluster configuration

We have configured 3 different clusters for our experiments, each with the below configuration.

**HDFS** stands for Hadoop Distributed Filesystem. HDFS has a master/slave architecture. An HDFS cluster consists of a single NameNode (master) that manages the file system namespace and access control. An HDFS cluster has a number of DataNodes which manage storage attached to the nodes.

<table>
<thead>
<tr>
<th>Node name</th>
<th>vcpus</th>
<th>RAM</th>
<th>Hard Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>namenode</td>
<td>8</td>
<td>8GB</td>
<td>20GB</td>
</tr>
<tr>
<td>datanode01</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
<tr>
<td>datanode02</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
<tr>
<td>datanode03</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
</tbody>
</table>

**TABLE II: HDFS Test Cluster configuration**

*Ceph* is a distributed file system with an underlying object storage. It uses a pseudo random algorithm CRUSH[10] for data distribution over a cluster of object storage devices (OSDs). A ceph cluster has three major elements namely a POSIX compliant client, a set of OSD daemons which use CRUSH to do distributed data placement and Ceph Monitors which maintain a copy of the CRUSH map. It aims to provide a unified object, block and file storage. Ceph test cluster configuration is as follows:

<table>
<thead>
<tr>
<th>Node name</th>
<th>vcpus</th>
<th>RAM</th>
<th>Hard Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>mon01</td>
<td>8</td>
<td>8GB</td>
<td>10GB</td>
</tr>
<tr>
<td>mon02</td>
<td>4</td>
<td>8GB</td>
<td>10GB</td>
</tr>
<tr>
<td>mon03</td>
<td>4</td>
<td>8GB</td>
<td>10GB</td>
</tr>
<tr>
<td>osd01</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
<tr>
<td>osd02</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
<tr>
<td>osd03</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
</tbody>
</table>

**TABLE III: Ceph Test Cluster configuration**

GlusterFS is Network File System with the aim to build a highly scalable and fault tolerant storage backend. It has two major components namely the storage server that runs a glusterfsd and the clients which use with glusterfs client or mount command to mount the exported filesystem. GlusterFS test cluster configuration is as follows:

<table>
<thead>
<tr>
<th>Node name</th>
<th>vcpus</th>
<th>RAM</th>
<th>Bricks (Volume Attached)</th>
</tr>
</thead>
<tbody>
<tr>
<td>node01</td>
<td>4</td>
<td>8GB</td>
<td>40GB</td>
</tr>
<tr>
<td>node02</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
<tr>
<td>node03</td>
<td>4</td>
<td>4GB</td>
<td>40GB</td>
</tr>
</tbody>
</table>

**TABLE IV: GlusterFS Test Cluster configuration**

We also have a controller node where our two modules: SSM and CRM, runs. We need higher configuration for the controller node as there are several process running in the same machine, also we have a Secondary controller node with the same configuration running as a backup controller.

<table>
<thead>
<tr>
<th>Node name</th>
<th>vcpus</th>
<th>RAM</th>
<th>Hard Disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller</td>
<td>8</td>
<td>16GB</td>
<td>256GB</td>
</tr>
<tr>
<td>Secondary</td>
<td>8</td>
<td>16GB</td>
<td>256GB</td>
</tr>
</tbody>
</table>

**TABLE V: Controller configuration**

The user interacts with the controller node with APIs described in Table I and the controller node instructs the user to transfer the data to the selected cluster from the Storage Selection Module.
B. Sample Generation

- **FIO - Flexible I/O Tester**
  We have used FIO, which is a versatile IO workload generator. Fio is very flexible as it lets you run the workload you want and could also retrieve all desired units of output. Any sort of random and sequential IO mix, or read/write mix, is easy to define. Fio provides plugins to run these tests on different storage backend/ioengines. For our experiments, we have used *libhdfs*, *librbd* and *libaio* for HDFS, Ceph and GlusterFS respectively.
  - *libhdfs*: It is a JNI based C api for Hadoop’s DFS. It provides a simple subset of C apis to manipulate DFS files and the filesystem. This api is integrated in fio, for which we need to set the namenode and port of the HDFS cluster as parameters.
  - *librbd*: It provides interface to Ceph RADOS block devices. It communicates with the librados/librbd C bindings and the krbd kernel module directly. We need to set the pool and rbd image as parameters in fio ceph configuration.
  - *libaio*: Asynchronous I/O (AIO) is a method for performing I/O operations on a network filesystem so that the process that issued an I/O request is not blocked till the data is available. We need to set the storage volume as parameters for libaio configuration in fio.

- **Workload generation**
  - Training data: There is a scarcity of publicly available storage dataset. To overcome we generate a diverse set of training and testing datasets. We try to include different classes of workloads like logs, transactional, photostream etc. in the mix as well as hybrid workloads.
  - Testing data: For testing purpose we generate a hybrid of IO workload classes. These workload classes are characterised by the difference of file size, read/write frequency, degree of concurrency, etc.

C. Performance of Regression

The coefficient of determination, or $R^2$, denotes the correlation coefficient between ground truth values and predicted values. It provides a measure of how well future samples are likely to be predicted by the model. Best possible score is 1.0, while constant model that always predicts the expected value of $y$ would get a $R^2$ score of 0.0.

<table>
<thead>
<tr>
<th>Storage Backend</th>
<th>Prediction metric</th>
<th>Avg $R^2$ score</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFS</td>
<td>Throughput</td>
<td>0.90571196245</td>
</tr>
<tr>
<td>Ceph</td>
<td>Throughput</td>
<td>0.652513723598</td>
</tr>
<tr>
<td>GlusterFS</td>
<td>Throughput</td>
<td>0.711219633794</td>
</tr>
<tr>
<td>HDFS</td>
<td>IOPS</td>
<td>0.925336270011</td>
</tr>
<tr>
<td>Ceph</td>
<td>IOPS</td>
<td>0.650942793762</td>
</tr>
<tr>
<td>GlusterFS</td>
<td>IOPS</td>
<td>0.735454555011</td>
</tr>
</tbody>
</table>

**TABLE VI**: $R^2$ score for different Storage backend

The methodology for measuring the performance of regression is to perform 5-fold cross validation on the generated data set. That is, the data set is partitioned uniformly into 5 subsets, and the model is trained on 4 subsets before being the
tested on the remaining one. The average of these experiments provides us with the average $R^2$ score for our system.

We draw the scatter plot between the actual value and the estimated value of throughput and IOPS for our model as shown in Figure 2. We also show the numerical predicted value for the same in Table VI. More the $R^2$ score, more correlated the estimated value is to the actual value.

As seen, HDFS is the most correlated followed by GlusterFS and Ceph. In each case, $R^2$ score is greater than 0.65 which implies it is possible to predict the value of throughput and IOPS with higher accuracy, hence validating the use of statistical regression in our model since the accuracy of Assignment depends on the accuracy of predicted value by the Regression model.

<table>
<thead>
<tr>
<th>Storage Backend</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDFS</td>
<td>0.9014084507</td>
<td>0.9275362319</td>
</tr>
<tr>
<td>Ceph</td>
<td>0.6</td>
<td>0.5294117647</td>
</tr>
<tr>
<td>GlusterFS</td>
<td>0.6363636364</td>
<td>0.6363636364</td>
</tr>
</tbody>
</table>

TABLE VII: Comparative Efficiency

D. Comparative Efficiency

Precision is defined as the proportion of instances classified in the category that truly belong to that category against the total number classified instances in that category. While Recall is defined as the proportion of instances classified in the category that truly belong to that category against the total number instances that truly belong to that category.

We now show the efficiency of the assignment process on the preference criteria of maximizing throughput, i.e.

\[
\text{Cost Function} = -\text{Throughput}.
\]

For each of three classes we find the precision and recall of our platform for the testing dataset that we generated.

Table VII shows the value of Precision and Recall for the Classification problem. We see the classifier predicts the HDFS class with very high precision of more than 0.9 followed by GlusterFS (with precision around 0.65) and Ceph.

However, it must be established that these values of precision and recall have been calculated for a particular uniformly generated set of samples, and are bound to be different when executed on samples from some other distribution. The performance largely depends on the distribution for sampling, and therefore the evaluation of the classification could differ in different use-cases that might result in different distributions.

As we observe, while the classifier is able to provide a decent maximization of the throughput, some limitations of the performance can be attributed to the dynamic nature of the underlying SDSs and the offline nature of our system.

VI. Features

A. Fault Tolerance

A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails. Here, in HetStore, we try to achieve this by spawning a backup controller. This architecture is inspired by Namnode, Secondary Namenode architecture of Hadoop. In a similar fashion, the backup controller keeps the metadata and the logs in sync with primary controller. Hence when the primary node fails, the backup controller will update the metadata and would start acting as the primary controller.

B. Plug and Play Mechanism to attach new SDS backend

In this paper, we integrate three different SDS clusters namely HDFS, Ceph and Gluster. Though, the administrator is free to integrate any new the storage backend dynamically. The controller would then run a set of synthetic workloads offline to develop an analytical profile for that particular storage backend. This model would then be integrated into the existing architecture and would then start provisioning online workloads. With such a plug-and-play feature, we give full freedom to the administrator to integrate any storage backend of her choice.

VII. Future Work and Conclusion

In this paper, we have focussed on the utility of regression models for estimating the performance of workloads in a heterogeneous SDS environment, in order to improve aggregate performance. However, the measure of this aggregate performance depends on the particular needs and requirements in the respective design or engineering problem. The choice of the cost function, therefore, would largely depend upon the specific problem at hand. The use of this system towards a general class of cost functions has been left open. It could be fruitful to identify some relevant classes of cost function, which are also amenable to analytic scrutiny (eg. linear cost functions over throughput and IOPS), and look at the performance of SVR-based classification over these classes.

The architecture is also extendible, and with only minor modifications can also incorporate metrics other than throughput and IOPS. We believe a better selection of features could also possibly provide better accuracy and remains to be explored. For instance, while we have not included performance metrics like latency in our system, similar estimation mechanism for latency could also be incorporated.

Also, as we note, some of the limitations of our system arise from the dynamic nature of the underlying SDS as well as the misalignment of distributions from which training data is sampled. In order to overcome that, an Online approach to this learning problem might provide better results and the challenge remains in finding such an approach without excessive overhead computation.

The limited availability of data sets on workloads result in a lack of available insight on the prevalent statistical distributions of workloads used in industry and practice. While online approach will resolve the problem for a specific use-case of this system, availability of standardized data sets would allow better benchmarking for such methods.
REFERENCES

[8] "Openstack Swift.,” http://docs.openstack.org/developer/swift/