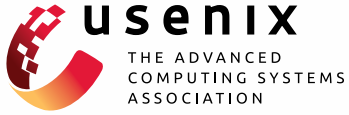


Dynamic Provisioning of Storage Workloads

Jayanta Basak and Madhumita Bharde, *NetApp Inc.*

https://www.usenix.org/conference/lisa15/conference-program/presentation/basak

This paper is included in the Proceedings of the 29th Large Installation System Administration Conference (LISA15).

November 8–13, 2015 • Washington, D.C.

ISBN 978-1-931971-270

Open access to the Proceedings of the 29th Large Installation System Administration Conference (LISA15) is sponsored by USENIX

Dynamic Provisioning of Storage Workloads

Jayanta Basak
NetApp Inc.

Madhumita Bharde
NetApp Inc.

Abstract

Due to lack of generic, accurate, dynamic and comprehensive models for performance estimation, customers typically tend to under-provision or over-provision storage systems today. With multi-tenancy, virtualization, scale and unified storage becoming norms in the industry, it is highly desirable to strike an optimum balance between utilization and performance. However, performance prediction for enterprise storage systems is a tricky problem, considering that there are multiple hardware and software layers cascaded in complex ways that affect behavior of the system. Configuration factors such as CPU, cache size, RAM size, capacity, storage backend (HDD/Flash) and network cards etc. are known to have significant effect on the number of IOPS that can be pushed to the system. However, apart from system characteristics as these, storage workloads vary reasonably and therefore, IOPS numbers depend heavily on types of workloads provisioned on storage systems. In this work, we treat storage system as a hybrid of black-box and white-box models, and propose a solution that will enable administrators to make decisions in the presence of multiple workloads dynamically. Our worst-case prediction is within 15% error margin.

1 Introduction

There is immense pressure on storage providers to increase utilization of their resources while maintaining performance guarantees. A storage resource can be operated at 'knee of the curve' - e.g. 70% of resource utilization, as a thumb rule. However, identifying the 'knee of the curve' dynamically is a challenge. The situation becomes more complicated for a mix of different workloads as response times are sensitive to workload characteristics. Moreover, any aggressive provisioning of storage resources can result in performance impact hitting bottom-line for the resource provider. To avoid such situations, storage providers often over-provision storage resources and system remains under-utilized.

Optimum resource utilization is also crucial in cloud provider environment. Usually, cloud providers thin provision resources. They need to be able to seamlessly provision containers, migrate VMs, and redistribute the resource pool among client applications [6, 8, 11, 13, 19, 21, 23, 24]. So it is extremely important to dynamically estimate the actual maximum throughput that can be delivered for these application environments. For example, with white-box like static provisioning, the system is utilized $40 - 50\%$ whereas the aim is to dynamically provision the workloads such that 70% utilization becomes a realistic goal.

Performance headroom modeling is typically done via two approaches – white-box and black-box. In white-box models [5], each component like CPU, disk, network, and memory is modeled using queueing theory. For each component, queueing delay for a certain IO request is computed and individual models are aggregated to obtain overall response time. Black-box models [7, 9, 12, 25, 26], on the contrary, model the entire system as a black-box and use machine learning techniques to predict the relationship between IO pattern and response time. White-box models are usually static in nature; but are highly tunable in terms of system parameters. On the other hand, black-box models can predict well in the dynamic environments but usually have less control on tuning of system parameters.

In this paper, we take a hybrid approach where we learn the system behavior in terms of characterizing the dependency of response time (latency) with IOPS. We also use concepts from queueing theory to model mixture of workloads on multiple volumes (logical containers) in an aggregate (physical container or set of disks). We leverage the advantages of black-box models for dynamic provisioning and that of white-box models for handling multiple workloads. We predict

the maximum IOPS possible per volume basis in an aggregate on a node. Our aim is to achieve a worst-case error margin of 15% for this prediction. The prediction of maximum IOPS is further extended for many interesting use cases such as:

- What is the maximum number IOPS that can be pushed for an existing workload?
- What would be maximum number of IOPS for a new workload, given its characteristics?
- What is the effect on existing workloads if a new workload is provisioned/migrated?
- In a storage cluster, what would be the best place to provision a workload?
- Can we redistribute workloads in the cluster for optimum utilization and performance?

The rest of the paper is organized as follows. Section 2 surveys the literature around white-box and black-box models for performance of storage systems. Section 3 describes motivations and challenges for a comprehensive storage provisioning solution. Section 4 gives details of techniques we came up with for storage provisioning. Section 5 presents experimental design and results. Finally, Section 6 summarizes and concludes the paper giving a glimpse in the possible future work.

2 Related Work

In this section, we provide a brief survey of the existing literature that is related to our work. White-box approaches for storage system modeling [5, 10] have existed for over three decades and are still being actively investigated in modern day mass storage systems [14]. White box approaches usually model individual components like CPU, memory, disk, and network are and compute IO requests delay as the queuing delay. The individual queuing delays are then aggregated to obtain the overall response time. In computing the overall response time, the workload characteristics are embedded in the sequence of read-write of operations of the IO pattern.

In black-box modeling, various machine learning techniques are applied to model storage system behavior such as the dependency of response time (latency) with IOPS. BASIL [7] and Pesto [9], Relative Fitness [12] and CMU-CART [25, 26] are three black-box techniques that have been proposed for modeling storage systems. BASIL provides predictions in the interpolation (trained) region. Pesto can extrapolate in the unseen region under the assumption that latency has a linear relationship with the outstanding IO. Both in relative fitness [12] and CMUCART [26], the performance of the storage system is predicted based on observed samples from the past. On a similar line, CART [4] has been used in black-box modeling of the storage performance in [28]. In these approaches, certain counters are considered, and a CART model is built to model latency and throughput. Very recently, bagging [3] of the CART models has been used for better prediction of the storage system performance [29].

In machine learning literature [22, 2], function approximation refers to predicting the functional value for an unobserved sample. In support vector regression (SVR) [22, 17] with RBF kernels, the prediction is usually good for the interpolation region. Support vector regression with polynomial kernels can be used for extrapolation. From the storage provisioning perspective, it is essential to associate a confidence level with the prediction to suggest to the user if the prediction is reliable or not. Kriging [27, 16] is able to associate a confidence level with each prediction. With a modification in computation of model variogram from the experimental variogram, Gaussian Process (GP) model has been developed [15] which also associates confidence level with prediction, although GP is suitable for interpolation only. In a recently developed black-box model called M-LISP [1], kriging has been used to successfully extrapolate the storage system performance for unseen amount of workload, and it is able to associate confidence interval with the prediction.

3 Motivation and Problem Statement

In a Latency (response time) vs IOPS curve for a storage system, the response time remains almost constant in low IOPS region even if more IOPS are pushed to the storage system. After the number of IOPS reaches a 'knee' , latency suddenly increases within a short range of IOPS. If IOPS are increased even further, after a point, no more IOPS can be pushed to the system and the latency shoots up drastically. It is desirable to operate at the knee for reasonable balance between performance and utilization. However, it is extremely difficult to theoretically quantify the

'knee' of the curve for a storage system. As a rule of thumb, industry practitioners use 70% of the maximum IOPS as the 'knee of the curve'.

Most of black-box modeling techniques in the literature predict the response time (latency) for certain IOPS based on the interpolation mechanism. Extrapolation techniques predict the response time for certain IOPS in an unseen region..Pesto [9] and M-LISP [1] perform extrapolation to predict the storage system performance. However, it is very difficult to apply them for mixed workload situation. For example, different workloads are typically deployed in different volumes and having $10-20$ volumes is quite common in industrial scenario [1]. For such cases, prediction for a new volume in absence of existing volumes is not applicable in their presence. Secondly, it is important to provision new volumes so that they don't affect performance of existing volumes. So, it's crucial for a customer to know the maximum possible IOPS that the system may provide for the new volume/workload even before it is actually provisioned. Existing black-box approaches estimate the maximum IOPS per volume on a running system and do not estimate that for a new volume. Also existing black-box approaches are not equipped to model multiple workloads due to their interference. It is possible to apply white-box models to appropriately capture the resource constraints but they are not generic and dynamic.

In this paper, we use a black-box model to capture the dependency of the response time with IOPS and predict the maximum IOPS per volume in an aggregate. We correlate the response characteristics with parameters of workloads. We then model the entire black-box server as a multi-queue single-server model to take into account of the multiple workloads running on multiple volumes in the same aggregate. For new workloads, we compute the maximum possible IOPS, given a system configuration, for various different workload characteristics in the laboratory environment before commissioning the system and construct look-up tables for the same. We observed that for a given configuration, the maximum possible IOPS depends on workload characteristics. The maximum IOPS for a given configuration and a given workload type may change due to system upgrade, file-system aging and fragmentation. We designed a provisioning system that takes feedback from the environment and dynamically updates the tables to adapt to these changes. Details are provided in Sections 4 and 5.

[1] As observed from customer systems

4 Details of the Approach

4.1 Maximum IOPS for a Single Workload

We have considered the entire storage system as a simple black-box server serving a queue of IO requests. Outstanding IOs (OIO) is a measure of the queue length (depth). From Little's Law[2], (OIO) can be expressed as

$$OIO = Latency \times IOPS \tag{1}$$

For higher IOPS, Latency is directly proportional to OIO to be served [9]. Therefore, we have

$$Latency = a \times OIO + b \tag{2}$$

where a and b are constants. From Equations (1) and (2), we have

$$Latency = \frac{b}{1 - a \times IOPS} \tag{3}$$

This derivation is also depicted in Figure 1.

As the storage system gets saturated, i.e., the denominator of Equation (3) becomes close to zero, latency value tends infinity (as shown in Figure 1). Substituting in Equation (8), we have

$$Maximum\ IOPS = 1/a \tag{4}$$

Note that, Equation(2) is true asymptotically. For small number of IOPS, the relationship may not hold true; however, we model the system in high IOPS region.

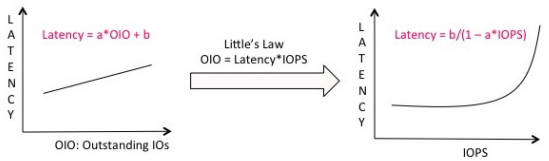


Figure 1: Fundamental Technique

As is evident from the Equation (4), maximum IOPS is the inverse of slope of the line representing the linear relationship between Latency

[2] URL:http://web.mit.edu/sgraves/www/papers/Little's%20Law-Published.pdf

vs OIO. Note that, this value $1/a$ could also be considered as a service rate of the system for a given workload. This terminology will be revisited later in Section 4.4.

We gathered periodic measurements (Latency, IOPS) from the system and did robust regression between latency and OIO to come up with a prediction for single workload scenario.

4.2 Dependency of Maximum IOPS on Workload Characteristics

Latency vs OIO relationship as described in Section 4.1 depends on a number of system and workload factors. However, our study showed that workload characteristics dictate this line between Latency vs OIO when system configuration does not change. Our experience with customer systems shows that the majority of workloads show variation of intensity (over a week or during holidays and peak hours etc) but there is generally no change in characteristics (read/write sizes, read/write ratio, sequential/random ratio), we did come across some real life workloads (Exchange, Financial) that exhibit large variations in the characteristics causing wide and divergent Latency vs OIO behavior as seen in Figure 4.2.

As is evident from Figure 4.2, observations in different IO size buckets are clearly segregated in separate regions. For such cases, we give different estimates based on currently observed workload characteristics. We bin workload parameters in different buckets and we estimate the maximum possible IOPS for each bucket separately. When a workload has several such buckets of characteristics, we advise a set of maximum possible IOPS that are governed by each bucket of workload characteristics.

4.3 Provisioning New Workloads

We show in Section 5.3 that a few workload characteristics (read/write sizes, read%, rand%) were enough to capture the essence of the workload and these characteristics would effectively dictate system performance in terms of maximum possible IOPS. This led us to believe that when workloads are abstracted as a set of characteristics, we could relate estimates across different workloads within an error margin of 15%. We used internal micro-benchmark SIO (Simple Input/Output) for creating tables across these finite characteristics dimensions. We then compared observed maximum when system was

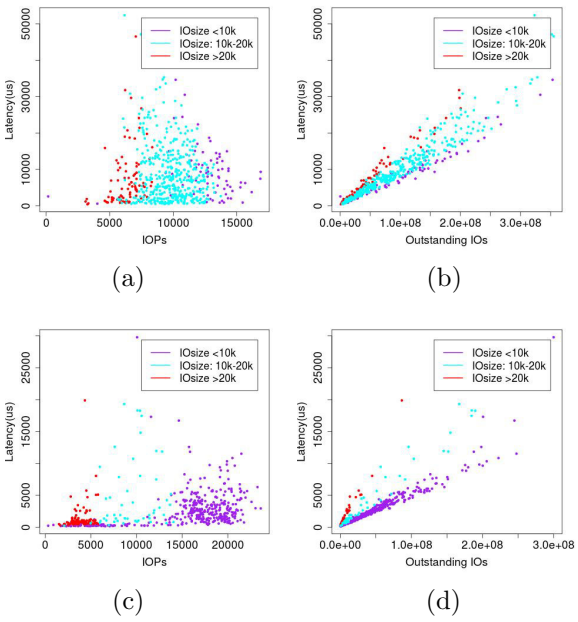


Figure 2: Dependency of the Latency with IOPS and outstanding IO: a) Latency vs IOPS for Exchange workload (b) Latency vs OIO for Exchange workload (c) Latency vs IOPS for Financial workload (d) Latency vs OIO for Financial workload.

driven to saturation with the predicted maximum, and validated the fact that the maximum IOPS is dependent on the workload characteristics for a given system.

We thus extended the workload characteristics table approach to be able to provide maximum possible IOPS estimates before provisioning a new workload. One of the major drawbacks of approach in Section 4.1 is that we need periodic measurements of certain metrics from the system. In other words, we need the workload to be deployed before an estimate can be given. However, because workload characteristics could abstract the workload as seen by the system (Section 4.2), we could give an estimate prior to provisioning the workload if the workload characteristics (read/write sizes, read%, random% etc.) are known.

4.4 Interference of Multiple Workloads

For mixed workload modeling, we view the storage system channel as a black-box serving IOs consisting of the controller, disks, CPU, memory, cache and other architectural components.

We model the black-box as a single server multi-queue where different queues represent different workloads running on different volumes. Service rates for different queues are different depending on the respective workload characteristics.

Let λ be arrival rate of requests and μ be the service rate of the system for a certain workload. Effectively, λ is current IOPS being served for that workload and μ is the maximum possible IOPS if the workload is run on the system in stand-alone mode. In other words, we can consider that no more than μ IOPS of this workload can be pushed in the black-box because it is being utilized 100% by this workload. This is the same as parameter $1/a$ in Section 4.1. Current utilization (ρ) of the black box for this workload is given as

$$Utlization = \rho = \frac{\lambda}{\mu} \tag{5}$$

In Equation (5), $\lambda = current\ IOPS$ and $\mu = Maximum\ IOPS$.

Given a system with n different workloads provisioned, total utilization is

$$\rho = \sum_{i=1}^{n} \rho_i \quad \text{where} \quad \rho_i = \frac{\lambda_i}{\mu_i} \tag{6}$$

Applying this for new workload provisioning, if the current utilization for the system (ρ) and service rate for new workload (μ_{new})are known, we have

$$Maximum\ IOPS = (1 - \rho) \times \mu_{new} \tag{7}$$

Maximum IOPS possible for a workload then essentially depends on how much the storage system is already utilized. For example, if current utilization is 50% then we can have only 50% of the maximum IOPS that were possible in stand-alone mode. Equation(7) is agnostic of the workload type that is running on the storage system. It only requires service rates for different workloads, and service rates depend on workload characteristics. We, therefore, compute service rates for different workload types and then compute resultant utilization of the storage system. Once the total utilization is known, the residual utilization governs the maximum possible IOPS for a new workload depending on the respective service rate.

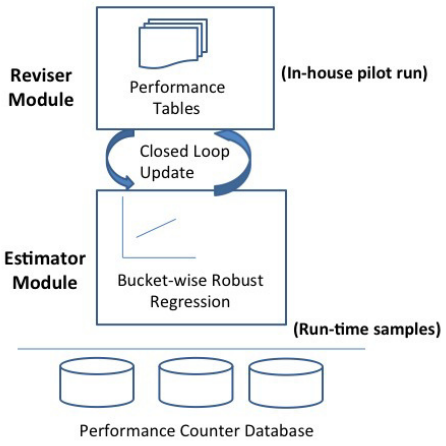


Figure 3: A schematic view of the deployment architecture for dynamic provisioning system

4.5 Deployment and Workflow

The deployment architecture takes care of adaptively adjusting service rate estimates by incorporating feedback from environment. It has two modules as shown in Figure 3. One module (Reviser) stores the workload characteristics based service rate tables as the master tables and a set of active tables. Active tables are derived from running workloads by mining them in workload parameters. At any point of time, this module derives the utilization of the storage system viewing it as a black-box and then predicts the maximum possible IOPS for a workload. The other module (Estimator) stores performance metrics measured from the system. Active tables are dynamically updated to take care of factors like aging, configuration change, system upgrades etc.

When a system is commissioned, master performance-tables are created (in-house pilot run) and populated in the Reviser module.

When workloads start running, performance counters are captured and recent snapshots of performance counters are populated in Estimator module. In Estimator module, workloads are bucketed in various buckets according to workload characteristics. Each bucket has several samples of performance counter measurements. (Workloads may not consist of buckets of several characteristics and therefore several buck-

ets may be empty.) Once performance counters are collected over a period of time, Latency vs OIO curves are estimated for non-empty buckets in Estimator module. From these curves, the maximum IOPS is estimated for the non-empty buckets for the running workload. Estimator module sends values of estimated maximum IOPS for non-empty buckets to Reviser module. Reviser module then compares estimates with those in active tables and incrementally modifies active tables. In summary, dynamic estimates for new or existing workloads are provided based on active tables and current black-box utilization along with knowledge of workload characteristics.

5 Experimental Results

We experimented and validated the effectiveness of proposed provisioning solution for various workloads on two different enterprise storage clusters. We used linux client to send IO traffic to storage servers.

5.1 Workloads

We used a well-cited collection of storage traces released by Microsoft Research (MSR) in Cambridge [18, 20] in 2007 for most of our evaluation. MSR traces record disk activity (captured beneath the file-system cache) of 13 servers with a combined total of 36 volumes for a week. We worked with 4 MSR workloads, namely, TPCE, Exchange, Financial, and a web server (Web). TPCE [3]is a On-Line Transaction Processing(OLTP) Workload developed by Transaction Processing Performance Council(TPC). Exchange workload captures activity of Microsoft Exchange application. Web workload records activity of web servers. Financial transactions are recoded in Financial workload. Raw traces comprise of 10-50 million records and consume just over $150 - 500$ MB in compressed CSV format each. Figure 4 shows extracted workload characteristics (read/write sizes, read/random percentage) for these workloads. We replay these traces using a trace replayer that runs on a host. It takes disk number, byte offset, IO size, elapsed time, timestamp and the type of operation (read or write) as input parameters and generates respective workloads.

Apart from these real life workloads, we also used an internal micro-benchmark (derived from FIO) named SIO [4] (Simple Input/Output). SIO, a synthetic tool available from host, takes read%, random%, IO size, offsets as input custom tun-

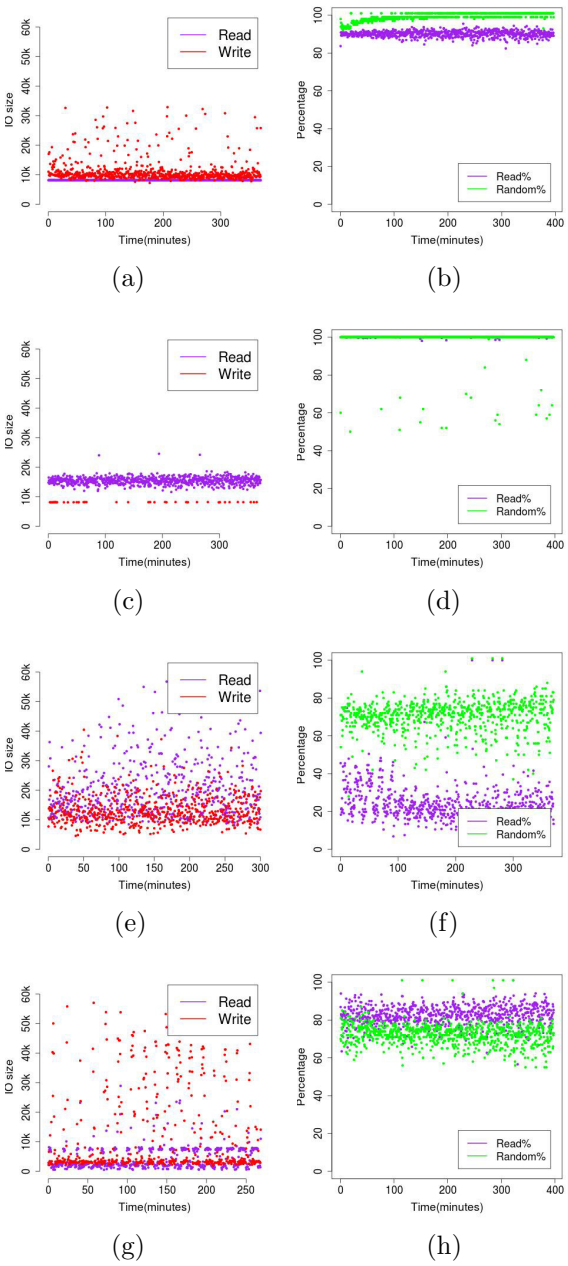


Figure 4: Workload characteristics for different workloads: (a) TPCE IO size (b) TPCE read % and random % (c) Web IO size (d) Web read % and random % (e) Exchange IO size (f) Exchange read % and random % (g) Financial IO size (h) Financial read % and random %

[3]URL:http://www.tpc.org/tpce/

[4]URL: http://web.stanford.edu/group/storage/netapp/sio_ntap/siontap.htm

able parameters to send traffic with desired workload characteristics.

As mentioned before, we aimed for an error margin of less than 20%. For all cases, we saturated the system with workload(s) in question and observed the maximum. Prediction error was calculated as below:

$$Prediction\ Error\% = \frac{OM - PM}{OM} \times 100 \quad (8)$$

where OM = Observed Maximum and PM = Predicted Maximum.

5.2 Maximum IOPS for a Single Workload

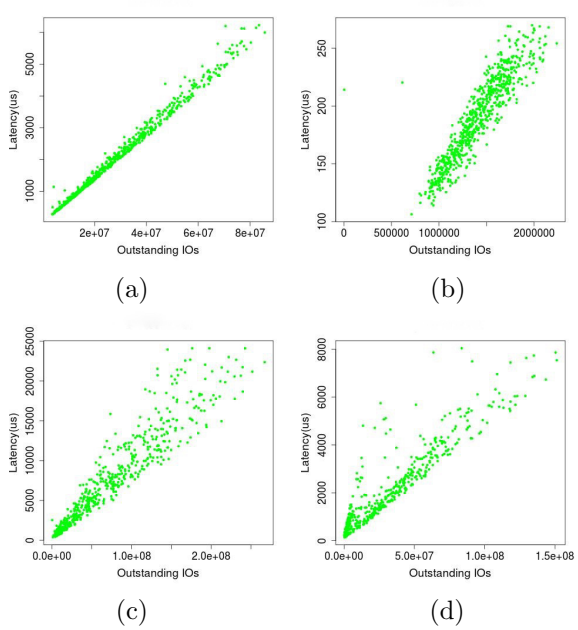


Figure 5: Latency vs OIO (a) TPCE (b)Web (c) Exchange (d)Financial

As seen from workload characteristics in Figure 4, workload parameters like read%, random% and read/write size do not show much variation for TPCE and Web workloads. For these well-behaving workloads, we see linear Latency vs OIO behavior as expected (Figure 5). Error between predicted and observed maximum reduced as we considered more and more sample points to finally stabilize at less than 15%, as seen in Figure 6.

For Exchange and Financial workloads, workload parameters show large variation (Figure 4) that causes scattered and wide-spread Latency vs OIO behavior as seen from the Figure 5. However, recalling from Section 4.2 when these characteristics were further bucketed in various workloads bins and our technique was applied on a region by region basis, we found that the error between the observed and predicted maximum IOPS was less than 15%.

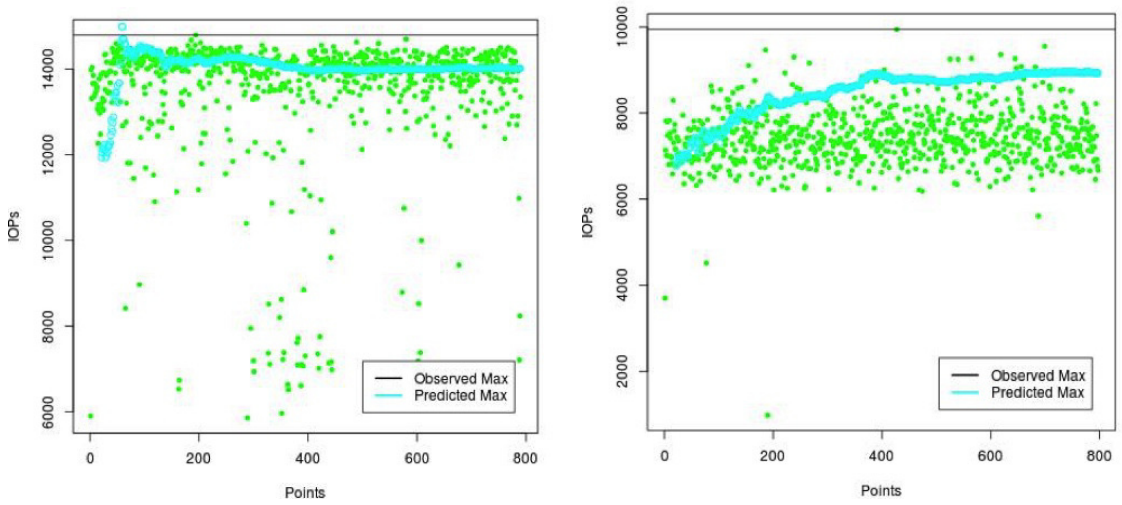
Dependency of maximum IOPS to workload characteristics led us into exploring this modeling further using SIO custom workload generator. We observed that the maximum IOPS on a given storage system is agnostic of the workload type (e.g., Exchange, Web, or Financial) and highly depends on the workload characteristics for a given system. We used an internal micro-benchmark SIO to generate combinations of workload parameters and saturate the system in each case. We collected the performance counters to measure the workload characteristics for each case to do robust regression as per the technique. Table 1 shows the observed maximum values for all buckets. We divided read % range in 6 buckets and IO size range in 5 buckets as shown in Table 1. Table 2 shows predicted maximum (from Latency-OIO regression) values. As seen from Table 3, when workloads are characterized in bins, errors are within acceptable limits. (We have presented the worst error case in tables 1, 2, 3 i.e. SIO random%=0.) This also proved that a workload-aware approach works well for performance prediction.

5.3 New Workload Provisioning

In this section, we extend the results to give predictions for workloads without having to deploy them. This is illustrated with Figure 6.

Workload characteristics for TPCE are read %=80 , random%=100 and IO size = 8k, as seen from Figure 4, If that particular bin from Table 4 is looked up, we get an estimate of 14877 , which is within 5% error margin of what we observed for TPCE when the system was saturated.

We performed with 10% error margin for web workload as well. As the exact web characteristics (Read%=90, Random%=100, IO size=16k) are not available in the table, we average estimates for two bins. So, the maximum IOPS estimate from Table 4 is $(8603+7123)/2 = 7863$. This predicted maximum is within 5% error margin to the observed maximum when the workload is actually provisioned (Figure 6(b)).



(a) (b)

Figure 6: Observed and predicted maximum IOPS for (a)TPCE and (b)Web.

Read % / IO size	0%	20%	40%	60%	80%	100%
4k	20656	20096	20325	20658	20679	21367
8k	13931	15354	15341	15045	14684	12698
16k	7155	8794	10608	9833	8476	7123
32k	3593	4479	5804	5715	4398	3583
64k	1802	2258	2867	2820	2228	1804

Table 1: Observed maximum IOPS for different buckets of workload characteristics.

Read % /IO size	0%	20%	40%	60%	80%	100%
4k	16575	16430	14301	17625	17997	18688
8k	12405	13016	11714	13006	14179	11752
16k	7130	8456	9684	9847	8121	6985
32k	3620	4422	5445	5684	4400	3605
64k	1783	2229	2760	2851	2183	1758

Table 2: Predicted maximum IOPS for different buckets of workload characteristics.

Read % / IO size	0%	20%	40%	60%	80%	100%
4k	13.07	18.46	28.12	11.76	13.58	12.97
8k	11.41	15.23	21.77	12.53	4.70	7.94
16k	0.24	4.61	8.64	1.56	5.60	1.93
32k	-0.42	0.85	6.96	1.60	2.46	-0.28
64k	1.17	1.15	4.37	-0.64	3.99	2.74

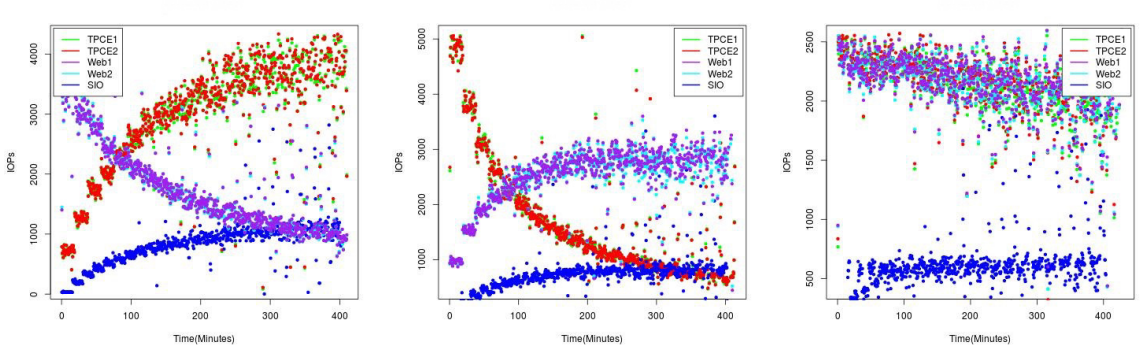
Table 3: Error % in prediction of the maximum IOPS for different workload characteristics.

Read % / IO size	0%	20%	40%	60%	80%	100%
4k	19068	20150	19896	19975	20825	21473
8k	14002	15355	14974	14869	14877	12765
16k	7147	8864	10600	10003	8603	7123
32k	3605	4460	5852	5776	4511	3595
64k	1805	2255	2886	2833	2274	1808

Table 4: New Workload Provisioning: Master Table (SIO Random% = 100)

Run Details	TPCE1	TPCE2	Web1	Web2	SIO	Utilization
Service Rate	14k	14k	8k	8k	8k	1
F=TPCE, B=Web	4k	4k	1.2k	1.2k	1.2k	1.02
F=Web, B=TPCE	1k	1k	3k	3k	1k	1.03
F=TPCE+Web	2.3k	2.3k	2.3k	2.3k	0.6k	0.98

Table 5: Multiple Workloads Scenarios(F= Foreground, B=Background)

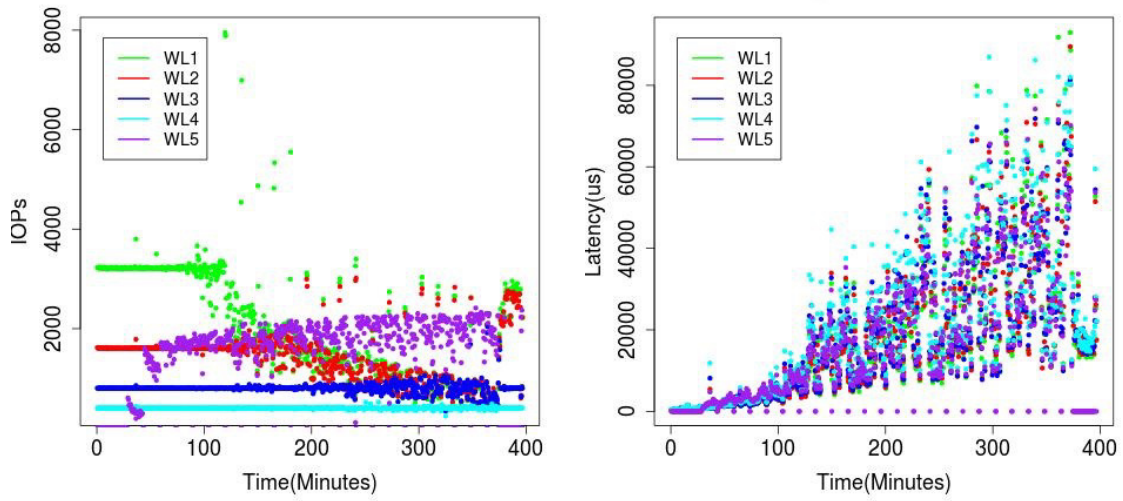


(a) (b) (c)

Figure 7: Multiple Workloads Scenarios: (a) Foreground=TPCE, Background=Web (b) Foreground=Web,Background=TPCE (c)Foreground=TPCE+Web

Volume	WL1	WL2	WL3	WL4	WL5
Service Rate	19975	14869	10003	5776	2833
Current IOPS	3200	1700	800	200	1730 (estimated maximum)

Table 6: Estimated maximum IOPS for a new workload (WL5) in presence of four other workloads.



(a) (b)

Figure 8: Effect of a new workload(WL5) on existing workloads

5.4 Interference of Multiple Workloads

For studying the interference of multiple workloads based on black-box utilization model, we created five volumes on the same storage aggregate and used pre-determined (using earlier techniques) service rates in order to calculate to-

tal system utilization for each of the three scenarios as listed in Table 5. F denotes workloads in the foreground which were increased in intensity continuously and B denotes workloads in the background that were kept constant in intensity.

In all scenarios, we saturated the system with foreground workloads. Figure 7 shows how IOPS changed for all workloads. In Table 5, we have IOPS in saturated condition for each of them. The last column denotes total system utilization calculated according to Equation (6). As expected, the total utilization number hovers around 1 because the system was saturated with foreground workloads.

We then created real life scenario [5] by deploying four volumes with four different SIO workloads WL1-WL4. On these volumes, we sent traffic of constant intensity (IOPS were kept constant). On fifth volume, we started increasing intensity of WL5. Figure 8 shows how IOPS and latency changed for all 5 workloads. From Table 6, service rate row contains values for service rate as obtained from SIO micro-benchmark table. Our aim is to calculate the maximum number of IOPS possible for WL5 given the current mix of other workloads WL1-WL4.

According to Equation (6), we can obtain the current utilization of the black-box using Table 6. So, current utilization = 3200/19975 + 1700/14869 + 800/10003 + 200/5776 = 0.39. Therefore, the predicted maximum IOPS according to Equation (7) for WL5 is = (1- 0.39) * 2833 = 1730. This is evident from Figure 8 (a). As IOPS for WL5 (purple) increased beyond the estimated maximum IOPS 1730 (this is approximately at point around 120 minutes on X-axis), IOPS for other workloads WL1-WL4 started decreasing gradually (10% in 10 minutes) and latencies abruptly shot up (50% in 10 minutes) after 120 minutes mark (Figure 8 (b))

5.5 Feedback Mechanism

File system fragmentation is known to affect the performance of a storage system [6]. We periodically fragmented the file-system in eight rounds using a synthetic tool. We set fragmentation parameters to most severe levels to simulate rapid aging of file-system. Each workload was increased in intensity during each round. We observed number of IOPS decreasing due to the effect of fragmentation. We periodically collected counters and predicted maximum IOPS for each workload using Equation (4). We then used a simple lazy update heuristic to update our estimates considering predicted maximum IOPS during consecutive iterations. Table 7 shows the baseline estimates before file-system aging and observed maximum at the end of eight rounds. There is a large difference between these numbers because of fragmentation. However, closed loop update mechanism updated estimate is a lot closer (around 10% error) to observed end maximum.

[5] As observed from customer systems

[6] URL:https://en.wikipedia.org/wiki/File_system_fragmentation

6 Summary and Future Work

In this paper, we present a mechanism for combining a queuing model with machine learning for dynamic provisioning of storage workloads. We estimate the maximum possible IOPS for a running workload using robust regression viewing the storage system as a black-box. For new workloads, we devise a method based on study of workload characteristics . We account for interference of existing workloads using the utilization of the storage server viewing the entire system as a multi-queue-single-server model where the queues are independent of each other. We have also developed a feedback mechanism to adapt estimates for change in factors like configuration change and aging etc. In all above cases, we were able to provide estimates within a reasonable error margin of 15-20%.

The techniques developed as part of this work are extensible to provide more comprehensive solutions for storage provisioning.

- Optimal Provisioning: Standard optimization techniques in literature can be used to formulate optimization objectives around performance and utilization. That would mean we could come up with the best possible arrangement and redistribution recommendations for workloads in or across storage clusters.
- Service Rate Normalization: For mixed workload modeling, we have used already estimated service rates. These can also be looked up in service rate tables if the workload characteristics are known. If the workload characteristics are unknown, service rates can be normalized dynamically to be able to apply the technique for migration.

Volume	Baseline	Updated End Prediction	Observed End Maximum	Error %
WL1	3244	2678	2433	10.07
WL2	2992	1496	1340	11.66
WL3	2793	790	718	10.04
WL4	2373	402	424	4.96
WL5	1766	209	216	2.90

Table 7: Feedback update for file-system fragmentation

Acknowledgements

The authors would like to acknowledge Veena Bhat, an intern with Advanced Technology Group, who helped us with scripts and execution part. Special thanks are due to Ajay Bakhshi and Siddhartha Nandi for facilitating this project and coming up with insightful suggestions to take this work forward in meaningful directions. The authors are very thankful to their shephard Chad Verbowski for critically evaluating the paper.

References

[1] Basak, J., Wadhwani, K., Voruganti, K., Narayanamurthy, S., Mathur, V., and Nandi, S. Model building for dynamic multi-tenant provider environments. *ACM SIGOPS Operating Systems Review 46* (2012), 20–31.

[2] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[3] Breiman, L. Bagging predictors. *Machine learning 24* (1996), 123–140.

[4] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Chapman & Hall, New York, 1983.

[5] Coffman, E. G. Queueing models of secondary storage devices. Tech. Rep. 90-943, Department of Computer Science, Purdue University, USA, 1990.

[6] Gulati, A., Ahmad, I., and Waldspurger, C. Parda: Proportional allocation of resources for distributed storage access. In *FAST09, San Jose, CA* (2009), pp. 85–98.

[7] Gulati, A., Kumar, C., Ahmad, I., and Kumar, K. Basil: Automated io load balancing across storage devices. In *Proc. 8th USENIX Conf File and Storage Technologies (FAST)* (2010), pp. 169–182.

[8] Gulati, A., Merchant, A., and Varman, P. mclock: Handling throughput variability for hypervisor io scheduling. In *OSDI 10, Vancouver, Canada* (2010).

[9] Gulati, A., Shanmuganathan, G., Ahmad, I., Waldspurger, C. A., and Uysal, M. Pesto: Online storage performance management in virtualized datacenters. In *Proc. 2nd ACM Symp. Cloud Computing (SOCC '11)* (2011), pp. 19–32.

[10] Johnson, T. Queuing models of tertiary storage. In *Fifth NASA Goddard Conference on Mass Storage Systems and Technologies* (1996), vol. NASA-CP-3340-Vol- 2, pp. 529–552.

[11] Merchant, A., Uysal, M., Padala, P., Zhu, X., Singhal, S., and Shin, K. Maestro: Quality-of-service in large disk arrays. In *ICAC 11, Karlsruhe, Germany* (2011), pp. 245–254.

[12] Mesnier, M., Wachs, M., Sambasivan, R. R., Zheng, A. X., and Ganger, G. Modeling the relative fitness of storage. In *Proc. Int Conf Measurements and Modeling of Computer Systems, SIGMETRICS 2004* (2004), pp. 37–48.

[13] Padala, P., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., and Salem, K. Adaptive control of virtualized resources in utility computing environments. In *EuroSys 07, Lisbon, Portugal* (2007), pp. 289–302.

[14] Pentakalos, O. I., Menasce, D. A., Halem, M., and Yesha, Y. Analytical performance modeling of hierarchical mass storage systems. *IEEE Trans. Computers 46* (1997), 1103–1118.

[15] Rasmussen, C. E., and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, USA, 2006.

[16] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. Design and analysis of computer experiments. *Statistical Science 4* (1989), 409435.

[17] Smola, A. J., and Schölkopf, B. A tutorial on support vector regression. Tech. Rep. NC-TR-98-030, NeuroCOLT, Royal Holloway College, University of London, UK, 1998.

[18] SNIA. Snia iotta repository, 2011. http://iotta.snia.org/trace.

[19] Soundararajan, G., and Amza, C. Towards end-to-end quality of service: Controlling i/o interference in shared storage servers. In *Middleware* (2008), pp. 287–305.

[20] SPC. Storage Performance Council: SPC trace file format specification, 2002. http://skuld.cs.umass.edu/traces/storage/SPC-Traces.pdf.

[21] Trushkowsky, B., Bod?k, P., Fox, A., Franklin, M., Jordan, M., and Patterson, D. The scads director: Scaling a distributed storage system under stringent performance requirements. In *FAST'11* (2011), pp. 163–176.

[22] Vapnik, V., Golowich, S., and Smola, A. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, M. Mozer, M. Jordan, and T. Petsche, Eds. MIT Press, Cambridge, MA, USA, 1997, pp. 281–287.

[23] Wang, A., Venkataraman, S., Alspaugh, S., Katz, R., and Stoica, I. Enabling high-level slos on shared storage systems. In *Symposium on Cloud Computing* (2012).

[24] Wang, A., Venkataraman, S., Alspaugh, S., Stoica, I., and Katz, R. Sweet storage slos with frosting. In *HotCloud'12, Boston, MA* (2012).

[25] Wang, M., Au, K., Ailamaki, A., Brockwell, A., Faloutsos, C., and Ganger, G. R. Storage device performance prediction with cart models. In *Proc. Int Conf Measurements and Modeling of Computer Systems, SIGMETRICS 2004* (2004), pp. 412–413.

[26] Wang, M., Au, K., Ailamaki, A., Brockwell, A., Faloutsos, C., and Ganger, G. R. Storage device performance prediction with cart models. Tech. Rep. CMU-PDL-04-103, Parallel Data Laboratory, Carnegie Mellon University, Pittsburgh, USA, 2004.

[27] Williams, C. K. I. Prediction with gaussian processes: From linear regression to linear prediction and beyond. In *Learning and Inference in Graphical Models* (1998), Kluwer, pp. 599–621.

[28] Yin, L., Uttamchandani, S., and Katz, R. An empirical exploration of black-box performance models for storage systems. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS '06)* (2006), pp. 433 – 440.

[29] Zhang, L., Liu, G., Zhang, X., Jiang, S., and Chen, E. Storage device performance prediction with selective bagging classification and regression tree. *Lecture Notes in Computer Science: Proceedings of the 2010 IFIP international conference on Network and parallel computing 6289* (2010), 121–133.