Pricing Games for Hybrid Object Stores in the Cloud: Provider vs. Tenant

Yue Cheng*, M. Safdar Iqbal*, Aayush Gupta†, Ali R. Butt*
*Virginia Tech, †IBM Research – Almaden

Abstract

Cloud object stores are increasingly becoming the *de facto* storage choice for big data analytics platforms, mainly because they simplify the management of large blocks of data at scale. To ensure cost-effectiveness of the storage service, the object stores use hard disk drives (HDDs). However, the lower performance of HDDs affect tenants who have strict performance requirements for their big data applications. The use of faster storage devices such as solid state drives (SSDs) is thus desirable by the tenants, but incurs significant maintenance costs to the provider. We design a tiered object store for the cloud, which comprises both fast and slow storage devices. The resulting hybrid store exposes the tiering to tenants with a dynamic pricing model that is based on the tenants' usage and the provider's desire to maximize profits. The tenants leverage knowledge of their workloads and current pricing information to select a data placement strategy that would meet the application requirements at the lowest cost. Our approach allows both a service provider and its tenants to engage in a pricing game, which our results show yields a win–win situation.

1 Introduction and Motivation

To make data analytics easy to deploy and elastically scale in the cloud while eliminating redundant data copying cost, cloud providers typically let their tenants run Big Data processing jobs on vast amount of data stored in object stores. For example, AWS [8], Google Cloud [4] and OpenStack [10] provide their own Hadoop to object store connectors that allow tenants to directly use object stores as a replacement of HDFS [26]. Moreover, commercial Big Data platforms such as Amazon EMR [1] and Azure HDInsight [5] go a step further and directly employ object stores as the primary storage technology.

Cloud-based object stores use low-cost HDDs as the underlying storage medium. This is because the price gap between HDDs and SSDs continue to be significant [9], especially for datacenter-scale deployments. Object stores have traditionally been used as data dumps for large objects such as backup archives and large-volume pictures or videos; use cases where SSDs would incur a high acquisition as well as maintenance cost [24], e.g., premature device replacement. Nevertheless, recent research has shown that SSDs can deliver significant benefits for many types of Big Data analytics workloads [13, 15, 16, 18], which are thus driving the need for adopting SSDs. Newer technology on this front is promising, but does not adequately address the cost and performance trade-offs. For example, while the newer 3-bit MLC NAND technology promises to deliver higher SSD densities and potentially drive down the acquisition cost, it has taken a major toll on SSD endurance [14, 23, 28], which raises the maintenance costs.

Tiered storage is used in many contexts to balance the HDD–SSD cost and benefits by distributing the workload on a hybrid medium consisting of multiple tiers [15, 17, 20, 27]. Data analytics applications are particularly amenable to such tiered storage deployments because of the inherent heterogeneity in workload I/O patterns. The choice of tiers depends on tenants' workloads and the performance benefits achieved by using specific tiers. A growing class of data analytics workloads demonstrate different unique properties [13], which cannot be satisfied by extant heat-based tier allocation approaches [15, 17, 20, 27]. To this end, we propose an innovative tiered object store that exposes tiering control to tenants by offering the tiers under dynamic pricing. Thus, the tenants can meet their price–performance objectives by partitioning their workloads to utilize different tiers based on their application characteristics.

In this paper, we argue that traditional HDD-based object stores are inefficient. (1) From the cloud tenants' perspective, an HDD-based object store cannot effectively meet their requirements (e.g., deadlines) due to the relatively slow I/O performance of HDDs. (2) From the cloud provider's perspective, an HDD-only object store does not provide any pricing leverage, which reduces profitability. A faster tier can provide a higher quality-of-service (QoS), which can be strategically priced to increase profits. Hence, a hybrid HDD–SSD approach is desirable for both cloud providers and tenants.

To verify our argument, we conducted a trace-driven simulation study by replaying two 250-job snippet traces from Facebook's Hadoop production traces [12]. We set the HDD tier price as \$0.0011/GB/day—average of the Google Cloud Storage price of \$0.00087/GB/day and the Google Cloud's HDD persistent block storage price of \$0.0013/GB/day—and the SSD tier price as \$0.0044/GB/day, i.e., 4× the HDD price. Note that we have chosen to use per-day pricing for our study as the granularity of our proposed price adjustment is **one day** (§3). `trace 1` consumes 12 TB data and generates 4.7 TB output, while `trace 2` consumes 18 TB data and generates 8.2 TB output. For the hybrid storage tiering case (`HDD+SSD`), the tenant places jobs in differ-

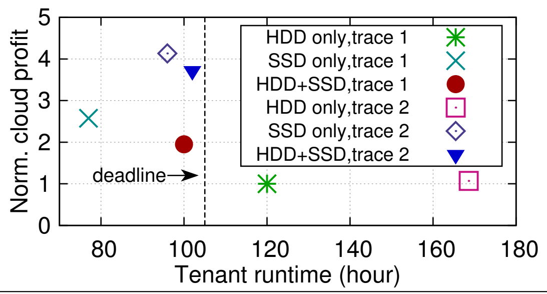


Figure 1: Tenant workload runtime for `trace 1` and `trace 2`, and provider's profits under different configurations.

ent tiers with a desired workload completion deadline of 105 hours. For this purpose, we use Algorithm 1 that essentially tries to optimize the tier allocation to meet the deadline while minimizing the cost (§2.2).

Figure 1 shows the results, from which we make three observations. (1) Workloads with `HDD-only` config. cannot meet the tenant-defined deadline and the cloud provider earns the lowest profit. (2) With `HDD+SSD` tiering config., both workloads are able to meet the deadline, while the cloud provider sees significantly more profit (`trace 2` has larger input and output datasets and hence yields more profit). This is because the tenant places part of the workloads on the SSD tier, which is more expensive than the HDD tier. (3) `SSD only` config. improves performance, but with marginally higher profit, compared to `HDD+SSD`. This is mainly due to `HDD+SSD`'s tiering optimization. This experiment demonstrates that through object storage tiering both the cloud provider and tenants can effectively achieve their goals.

Cloud providers have a multitude of device options available for deploying object storage infrastructure, including HDDs with different RPMs, SATA and PCIe SSDs and the emerging SCM devices, etc. These options offer different performance–price trade-offs. For example, each device type offers different cost/GB and, in case of SSDs and SCMs, different endurance. In a tiered setup comprising such a variety of storage choices, estimating price points while keeping maintenance costs under control is a challenge. Moreover, *while the cloud providers encourage tenants to use more SSDs to increase their profits, they want to keep SSDs' wear-out in check* – if tenant workloads are skewed towards SSDs due to high performance requirements, providers run the risk of SSD wear-out earlier than expected, which ends up increasing management costs and decreasing overall profits.

To remedy the above issue, we introduce a dynamic pricing model that providers can leverage to mitigate additional costs and increase overall operating profits for providers. The dynamic pricing model has two objectives: (1) to balance the price-increasing and SSD wear-out rate by exploiting the trade-off between high revenue versus high operational costs (e.g., replacing SSDs) for *high profit*; (2) to provide an effective incentivizing mechanism to tenants so that the tenants can meet their goals via object store tiering in a more cost-efficient fashion. Generally, storage tiering has been looked at from just one entity's perspective. In contrast, the novelty of this work lies in *the leader/follower game theoretic model that we adopt, where the objectives of cloud provider and tenants are either disjoint or contradictory. We take the first step towards providing a cloud-provider-driven game-theoretic pricing model through object storage tiering*. Yet another unique aspect of our storage tiering approach is *handling of the lack of information available to the players (cloud, tenants)*. In extant tiering solutions adopted in private datacenters, data placement decisions are generally made by administrators who have detailed information about the systems involved. This is not true in public cloud space, where information about many aspects or details may be missing. Thus, not only the motivations different from the private deployments for providers and tenants in a public cloud, but they also have to make decisions based on partial information.

Specifically, we makes the following contributions in this paper. (1) We design a leader/follower gaming model with the goal to maximize cloud provider's profit. The provider makes the pricing decisions by estimating tenants' storage capacity demand distribution among different tiers; driven by the prices, tenants employ a simulated annealing based tiering solver to guide object storage tiering for maximizing their utility. (2) We demonstrate through trace-driven simulations that our novel object storage tiering pricing mechanism can deliver increased profit for the cloud provider and potentially achieve *win–win* for both the provider and tenants.

2 Model Design

We design a leader/follower cloud pricing framework with the objective of maximizing a cloud provider's profit. We model the object storage tiering for tenants and dynamic pricing for the provider, and capture the provider–tenant interactions. In our model, the game is played in two steps. First, the cloud provider (leader) makes the pricing decisions (§2.1) based on predictions of tenants' demand on storage resources. Second, given prices of different storage resources, a tenant (follower) makes tiering decisions based on her own requirements (§2.2), and the strategy is represented by the tenant's storage tiering specification, i.e., which jobs use what tier.

While the tenants can see the price changes by the provider, they are unaware of the actual reasons for the changes. Even if the tenants understood the reasons, multi-tenancy prevents modeling of the provider's behavior. Hence, in our formulation tenants can only predict the provider's price movements based on historical data. Similarly, the provider is not aware of explicit tenant requirements, and only garners information from the requested storage capacity and the writes operations (`PUT`

	Notation	Description
provider	F	set of all tiers in object store
	$capacity_{(n,s)}$	total capacity of tier s in time slot n
	$f_{(n,s)}$	fraction of data placed on tier s in time slot n
	$w_{(n,s)}$	writes in GB on tier s in time slot n
	$p_{(n,s)}$	price of tier s in time slot n (***decision var***)
	a	HDD class
	b	SSD class
	$\alpha_1, \beta_1, \alpha_2, \beta_2$	model parameters
	θ	parameter used to constraint SSD's price
tenant	J	set of all analytics jobs in a workload
	sz_i	dataset size of job i
	x_i	tier used by job i (***decision var***)
	c_i	capacity provisioned for job i (***decision var***)
	α_3, β_3	model parameters

Table 1: Notations used in the provider and tenant models.

requests are tracked for accounting purposes [6]). Thus, the provider also only uses historical information about these aspects to predict tenant demand. Consequently, both the tenants and the provider models adopted in our game are purposefully "myopic" controls for predicting only the next time slot, and not beyond that in the future.

2.1 Provider Model

We model the provider cost as follows. Assuming that the fraction of the cost that comes from SSDs wear-out is $t < 1$,[1] the cost can be modeled as: $cost = \frac{1}{t} \cdot \frac{p_{ssd}}{endurance} \cdot w$, where p_{ssd} is the market price of one SSD, *endurance* is the endurance lifespan of the particular SSD, and w is the amount of data written to the SSD in GB. Table 1 lists all the notations used in our provider and tenant models. The pricing decision making process can be modeled as a non-linear optimization problem that maximizes the profit defined by Equation 1.

$$max \quad profit = \sum_i \Big(\sum_f (capacity_f \cdot f_{(i,f)} \cdot p_{(i,f)}) - cost_i \Big) \tag{1}$$

$$s.t. \quad f'_{(n+1,b)} = \alpha_1 \cdot f_{(n,b)} - \beta_1 \cdot (p'_{(n+1,b)} - p_{(n,b)}) \tag{2}$$

$$f'_{(n+1,a)} = 1 - f'_{(n+1,b)} \tag{3}$$

$$f_{(n,b)} = \alpha_1 \cdot f_{(n-1,b)} - \beta_1 \cdot (p_{(n,b)} - p_{(n-1,b)}) \tag{4}$$

$$f_{(n,a)} = 1 - f_{(n,b)} \tag{5}$$

$$w'_{(n+1,b)} = \alpha_2 \cdot w_{(n,b)} + \beta_2 \cdot (f'_{(n+1,b)} - f_{(n,b)}) \tag{6}$$

$$w_{(n,b)} = \alpha_2 \cdot w_{(n-1,b)} + \beta_2 \cdot (f_{(n,b)} - f_{(n-1,b)}) \tag{7}$$

$$\forall i: \ w_{(i,b)} \leq \mathscr{L}_i \tag{8}$$

$$\forall i,s: \ 0 \leq capacity_i \cdot f_{(i,s)} \leq L_s \tag{9}$$

$$\forall i: \ p_{min,b} \leq p_{(i,b)} \leq \theta \cdot p_{(i,a)} \ , \ where \ \ \theta > 1 \tag{10}$$

$$\forall i,s: \ f_{(i,s)} \leq 1 \ where \quad i \in \{n, n+1\}, \ s \in F \tag{11}$$

In a time slot n, we predict the SSD demand proportion for the next time slot $n+1$ with Equation 2, which depends on the difference between the predicted SSD price for $n+1$ and the calculated SSD price for n. The predicted HDD demand proportion is determined by Equation 3. Similarly, Equation 4 and 5 define the predicted SSD and HDD demand proportion for n, respectively, and Equation 11 enforces the proportion range.

Equation 6 predicts the amount of data that will be written to SSDs, which is determined by the difference of predicted SSD demand proportion in time slot $n+1$ to that in time slot n. If the SSD demand is predicted to increase, it implies that the amount of data that will be absorbed by the SSD tier will also increase. Equation 8 defines the SSD tier data writing constraint, which ensures that the expected amount of data written to the SSD tier will not exceed the threshold that is calculated based on accumulated historical statistics. The factor indirectly controls the value adaptation of decision variables $p_{(n,b)}$ and $p'_{(n+1,b)}$. The storage capacity limit in the cloud datacenter is defined by Equation 9. We assume HDD prices $p_{(n,a)}$ and $p_{(n+1,a)}$ are fixed, and SSD prices are constrained in a range given by Equation 10.[2]

2.2 Tenant Model

The data placement and storage provisioning at the tenant side is modeled as a non-linear optimization problem as well. The goal is to maximize *tenant utility* as defined by Equation 12.

$$max \quad utility = \frac{1}{(T \cdot \$)} \tag{12}$$

$$s.t. \quad \forall i \in J: \ c_i \geq sz_i \tag{13}$$

$$T = \sum_{i=1}^{J} (x_i, c[s_i], \hat{\mathscr{R}}, \hat{\mathscr{L}}_i) + penalty(\text{data migrated}) \leq deadline \ , \ where \ x_i \in F \tag{14}$$

$$\$ = \sum_{s=1}^{F} \Big(c[s] \cdot (p_{(n,s)} \cdot \lceil ^{T}/_{60} \rceil) \Big) \tag{15}$$

$$where \ \forall s \in F, \ \Big\{ \forall i \in J, \ s.t. \ x_i \equiv f : c[s] = \sum c_i \Big\} \ ,$$

$$p'_{(n+1,b)} = \alpha_3 \cdot p_{(n,b)} + \beta_3 \cdot p_{(n-1,b)} \tag{16}$$

The performance of the tenant's workload is modeled as the *reciprocal* of the estimated completion time in minutes ($^1/_T$) and the cost includes mainly the storage costs. The cost of each storage service is determined by the workload completion time (storage cost is charged on a hourly basis) and capacity provisioned for that service. The overall storage cost is obtained by aggregating the individual costs of each tier in the object store (Equation 15). Equation 13 defines the *capacity constraint*, which ensures that the storage capacity (c_i) provisioned for a job is sufficient to meet its requirements for all the workload phases (map, shuffle, reduce). Given a specific tiering solution, the estimated total completion time of the workload is defined by Equation 14, and constrained by a tenant-defined *deadline*. Equation 16 is the price predictor at the tenant side. The predicted price value can also be supplied as a hint by the cloud provider. The function *penalty*(.) serves as a penalty term that the ten-

[1]We choose to use a fixed t for simplicity; in real world, there are numerous factors that come into play and t may not be a constant.

[2]We plan to include IOPS per client in our future pricing models.

Algorithm 1: Tiering solver.

```
Input: Job information matrix: $\hat{\mathscr{L}}$, Analytics job model matrix:
       $\hat{\mathscr{M}}$, Runtime configuration: $\hat{\mathscr{R}}$, Initial solution: $\hat{P}_{init}$.
Output: Tiering plan $\hat{P}_{best}$
begin
    $\hat{P}_{best} \leftarrow \{\}$
    $\hat{P}_{curr} \leftarrow \hat{P}_{init}$
    $exit \leftarrow False$
    $iter \leftarrow 1$
    $temp_{curr} \leftarrow temp_{init}$
    $U_{curr} \leftarrow Utility(\hat{\mathscr{M}}, \hat{\mathscr{L}}, \hat{P}_{init})$
    while $not\ exit$ do
        $temp_{curr} \leftarrow Cooling(temp_{curr})$
        for $next\ \hat{P}_{neighbor}\ in\ AllNeighbors(\hat{\mathscr{L}}, \hat{P}_{curr})$ do
            if $iter > iter_{max}$ then
                $exit \leftarrow True$
                break
            $U_{neighbor} \leftarrow Utility(\hat{\mathscr{M}}, \hat{\mathscr{L}}, \hat{P}_{neighbor})$
            $\hat{P}_{best} \leftarrow UpdateBest(\hat{P}_{neighbor}, \hat{P}_{best})$
            $iter{+}{+}$
            if $Accept(temp_{curr}, U_{curr}, U_{neighbor})$ then
                $\hat{P}_{curr} \leftarrow \hat{P}_{neighbor}$
                $U_{curr} \leftarrow U_{neighbor}$
                break
    return $\hat{P}_{best}$
```

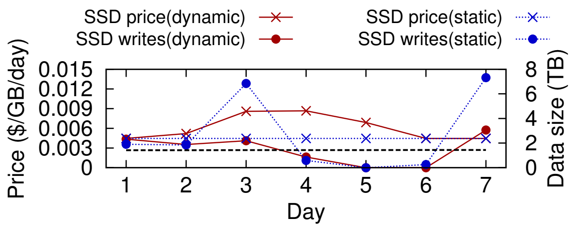
ant takes into account in terms of performance loss (e.g., longer completion time) while deciding tiers.

We devise a simulated annealing based algorithm [13] (Algorithm 1) for computing tenants' data partitioning and job placement plans. The algorithm takes as input workload information ($\hat{\mathscr{L}}$), compute cluster configuration ($\hat{\mathscr{R}}$), and information about performance of analytics applications on different storage services ($\hat{\mathscr{M}}$) as defined in Table 1. $\hat{P}_{init}$ serves as the initial tiering solution that is used to specify preferred regions in the search space. The results from a simple greedy algorithm based on the characteristics of analytics applications (e.g., the four described in §3) can be used to devise an initial placement.

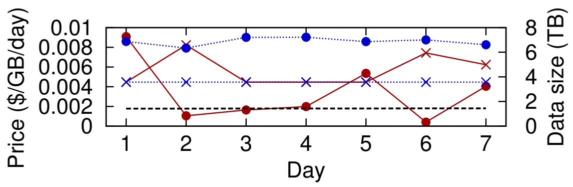
3 Preliminary Results

We have used trace-driven simulations to demonstrate how our cloud–tenant interaction models perform in practice. We use the production traces collected from a 3,000-machine Hadoop deployment at Facebook [12]. The original traces consist of 25,428 Hadoop jobs; we chose to use a snippet of 1,750 jobs to simulate a 7-day workload. The workload runs on a cloud object store with built-in tiering mechanism that tenants can control. We set the time slot for our models to one day.

We assign to our workload, in a round-robin fashion, four analytics applications that are typical components of real-world analytics [12, 29] and exhibit diversified I/O and computation characteristics. `Sort`, `Join` and `Grep` are I/O-intensive applications. The execution time of `Sort` is dominated by the shuffle phase I/O. `Grep` spends most of its runtime in the map phase I/O, reading input and finding records that match given patterns. `Join` represents an analytic query that combines rows from multiple tables and performs the join operation during the reduce phase, making it reduce intensive. `KMeans` is an iterative CPU-intensive clustering application that expends most of its time in the compute phases of map and reduce iterations.



(a) A 7-day trace.

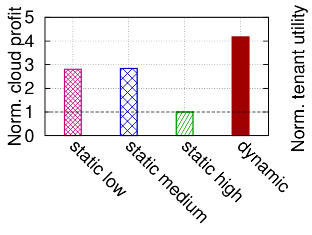


(b) A single-day trace repeated 7 times.

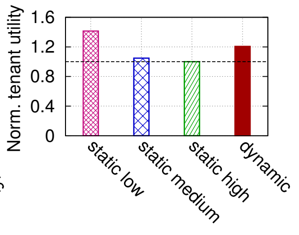
Figure 2: Dynamic pricing by the provider and the tenant's response for two 7-day workloads. The dotted horizontal line represents the daily write limit L_i. The variation in L_i is too small to be discernible at this scale.

Figure 2 shows price variation by the cloud provider's model based on the amount of data written by the tenant to the SSD tier on a per-day basis. The HDD price is fixed, while the SSD price is dynamically adjusted on a daily basis for `dynamic` pricing (the pricing is the same as for Figure 1). Under `static` pricing, the provider sets a static price and tenants periodically run Algorithm 1, whereas under `dynamic` pricing, the provider and tenants interact. The per-day write limit $\mathscr{L}_n$ is dynamically adjusted based on the amount of writes from the tenant side (though not discernible in the figure). Figure 2(a) shows the price changes for a 7-day trace, with a different workload running on each day. We observe that as the amount of writes by the tenant on SSD tier increases above the write limit, the cloud provider begins to adjust the SSD price. The tenant's model will adjust the tiering strategy to either put more data in the HDD tier or pay more for the SSD tier in the case of a strict deadline requirement. Since, each day has a different workload and hence a different deadline. For example, from day 4, the tenant, with the goal of maximizing the tenant utility, allocates fewer jobs to the SSD tier. Once the SSD writes are reduced below the threshold, the provider lowers the SSD tier price to incentivize the tenant to use more of the SSD resource on the next day. The tenant responds to this change on day 7, where the workload deadline is stricter than the previous 2 days.

Figure 2(b) shows the price changes for a 7-day period with the same single-day trace replayed every day. This trace shows stronger correlation between per-day



(a) Cloud profit.



(b) Tenant utility.

Figure 3: Cloud profit and tenant utility averaged over 7 days. `static low`, `static medium` and `static high` mean a low, medium, and high static SSD price, respectively. Cloud profit and tenant utility are normalized wrt. `static high`.

SSD writes from the tenant and the SSD price from the provider. This workload exhibits the same specifications every day (e.g. dataset size, a relaxed deadline, etc.), thus the daily writes on the SSD tier remain stable under static pricing. However, the dynamic pricing model can effectively respond to the spike in the amount of writes on the first day and adjust the SSD price accordingly. Given the increased SSD price, the tenant tries to reduce their monetary cost by migrating more jobs to the cheaper HDD tier, while still meeting the deadline. When the provider lowers the SSD price in response, the tenant increases their use of SSD, prompting the provider to increase the SSD price again. This interaction of the tenant and the provider results in an average of 2.7 TB/day SSD writes compared to an average of 7 TB/day under static pricing (with 0% deadline miss rate for both cases). The test demonstrates that our dynamic pricing model can adaptively adjust the SSD pricing based on the amount of data written to the SSD tier to maintain high profits while keeping the SSD wear-out under control by keeping write loads to SSD in check.

In our next test, we examine the impact of different SSD pricing models on provider profit and tenant utility, i.e., the cloud–tenant interaction. Figure 3 shows the results. We choose three static prices for SSDs: low (the minimum SSD price that we use: \$0.0035/GB/day), medium (\$0.0082/GB/day), and high (the maximum SSD price, \$0.0121/GB/day). We also compare static prices with our dynamic pricing model. As observed in Figure 3(a), dynamic pricing yields the highest provider profit as it increases the price based on SSD writes from the tenant. Both `static low` and `medium high` yield similar profits that are 32.6% lower than that gained under dynamic pricing. This is because `static medium` results in more jobs placed on the HDD tier, which lowers the tenant cost while causing longer workload completion time. `static low` is not able to generate enough profit due to the very low price, while under `static high` the tenant solely migrates all the jobs to the HDD tier, thus resulting in low profit.

Next, we examine the tenant utility in Figure 3(b). With `static low` SSD price, the tenant utility is 17.1% higher than that achieved under dynamic pricing. However, this would result in significantly shortened SSD lifetime (by as much as 76.8%), hence hurting the cloud profit in the long term. With `static high` SSD price, the tenant utility is reduced by 17.1% compared to that of dynamic pricing, as the tenant shifts most jobs to the HDD tier. `static medium` SSD price yields slightly higher tenant utility as compared to `static high` but still 13.1% lower than that seen under dynamic pricing. This is because the tenant has to assign some jobs to the faster SSD tier in order to guarantee that the workload does not run for too long. Dynamic pricing, on the other hand, maintains the tenant utility at a reasonably high level (higher than both `static medium` and `static high` but slightly lower than `static low`), while guaranteeing that the SSD lifetime constraints are met. This demonstrates that our dynamic pricing model can effectively achieve a win–win for both the cloud provider and tenants.

4 Related Work

Storage Tiering Recent research [16, 18] demonstrates that adding a SSD tier for serving reads is beneficial for HDFS-based HBase and Hadoop. Existing implementations of cloud object stores provide mechanisms for tiered storage. OpenStack Swift supports storage tiering through Storage Policies [7]. Ceph, which exposes an object store API, has also added tiering support [2]. Our work focuses on providing insights into the advantages of dynamically priced tiered object storage management involving both cloud providers and tenants.

Cloud Pricing Researchers have also looked at cloud dynamic pricing [19, 21, 25]. CRAG [3] focuses on solving the cloud resource allocation problems using game theoretical schemes, while Londono et al. [22] propose a cloud resource allocation framework using colocation game strategy with static pricing. Ben-Yehuda et al. [11] propose a game-theoretic market-driven bidding scheme for memory allocation in the cloud. We adopt a simplified game theoretic model where the cloud providers give incentives in the form of dynamic pricing and tenants adopt tiering in object stores for achieving their goals.

5 Conclusion

We show that by combining dynamic pricing with cloud object storage tiering, cloud providers can increase their profits while meeting the SSD wear-out requirements, and tenants can effectively achieve their goals with a reasonably high utility. We demonstrate this win–win situation via real-world trace-drive simulations. In our future work, we plan to explore different tiering algorithms and best dynamic pricing models in multi-tenant clouds.

Acknowledgments We thank the reviewers for their valuable feedback and support. This work was sponsored in part by NSF under Grants CNS-1405697 and CNS-1422788.

6 Discussion Points

While a lot of research has looked at the technical aspects of building clouds, their impact and their use by the tenants, we believe the current cloud pricing mechanism (especially for storage services) is vague and lacks transparency. In this work, we have attempted to reconcile the economic principles of demand and supply with the technical aspects of storage tiering through dynamic pricing to showcase the benefits for both the cloud provider as well as the tenants.

We believe that our paper will lead to discussion on the following points of interest: (1) The need for revisiting pricing strategies in established hybrid storage deployments and practices. Since storage tiering is a well-studied area, we believe that the paper will lead to hot discussion on how the paradigm shift is happening, and why the extant approaches and ideas need to be revisited. A particular point of interest in the context of tiering is that the objectives of different players are often times in conflict. (2) Issues such as how useful can our pricing "knob" be for the cloud provider to shape tenant behavior to suit the provider requirements, and at what granularity should the proposed price variations be implemented, are part of our future work and would make for a productive discussion. (3) The role of flash and other emerging technologies in cloud-based object stores.

An unexplored issue is tenant behavior modeling. Our preliminary results assume a smart tenant using models described in §2. However, in reality, tenants can have very different behaviors and utility functions. Furthermore, we have not looked at multi-tenancy in detail. For instance, a "naive" or "rogue" tenant that performs a lot of SSD writes without considering their utility can cause the cloud to increase the price of SSDs, thus affecting the utility of other well-behaved tenants. Another open aspect of our current work is investigating the game theoretic results of our model. These include the behavior of provider's profit and the tenant's utility when the system reaches equilibrium and the comparison of these objectives under Nash and Stackelberg equilibria.

Furthermore, prices of SSDs are falling. Even though the gap between HDD and SSD prices is still wide today, in the future with increase in NAND flash production, improvement in flash yields, lithographic improvements such as 3D stacking, etc., can reduce the price difference, resulting in SSDs becoming the de facto storage medium in cloud environments. From the tenant side, while researchers have shown the merit of using SSDs (e.g., in interactive HBase workloads [16]), their use in batch-oriented analytics workloads is just starting. Indeed, in another research work [13], we have shown through real experiments on Google Cloud that SSDs can provide great benefits especially when considering heterogeneity in cloud storage services, enforcing our belief that our future hybrid object store prototyping efforts will yield desirable win–win solutions.

References

[1] Amazon Elastic MR. http://goo.gl/GXzwaU.

[2] Ceph Storage Pools. http://docs.ceph.com/docs/argonaut/config-cluster/pools/.

[3] Cloud Resource Allocation Games. https://www.ideals.illinois.edu/handle/2142/17427.

[4] Google Cloud Storage Connector for Hadoop. http://goo.gl/ji4qqp.

[5] Microsoft Azure HDinsight. http://goo.gl/wsQ9QG.

[6] OpenStack Ceilometer. https://wiki.openstack.org/wiki/Ceilometer.

[7] OpenStack Swift Policies. http://docs.openstack.org/developer/swift/overview_policies.html.

[8] S3 as a replacement of HDFS. https://wiki.apache.org/hadoop/AmazonS3.

[9] SSD vs. HDD Pricing: Seven Myths That Need Correcting. http://www.enterprisestorageforum.com/storage-hardware/ssd-vs.-hdd-pricing-seven-myths-that-need-correcting.html.

[10] Swift Connector to Hadoop. http://docs.openstack.org/developer/sahara/userdoc/hadoop-swift.html.

[11] Agmon Ben-Yehuda, O., Posener, E., Ben-Yehuda, M., Schuster, A., and Mu'alem, A. Ginseng: Market-driven memory allocation. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (New York, NY, USA, 2014), VEE '14, ACM, pp. 41–52.

[12] Chen, Y., Alspaugh, S., and Katz, R. Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads. *Proc. VLDB Endow. 5*, 12 (Aug. 2012), 1802–1813.

[13] Cheng, Y., Iqbal, M. S., Gupta, A., and Butt, A. R. Cast: Tiering storage for data analytics in the cloud. In *Proceedings of the 24rd International Symposium on High-performance Parallel and Distributed Computing* (New York, NY, USA, 2015), HPDC '15, ACM.

[14] Grupp, L. M., Davis, J. D., and Swanson, S. The bleak future of nand flash memory. In *Proceedings of the 10th USENIX Conference on File and Storage Technologies* (Berkeley, CA, USA, 2012), FAST'12, USENIX Association, pp. 2–2.

[15] Guerra, J., Pucha, H., Glider, J., Belluomini, W., and Rangaswami, R. Cost effective storage using extent based dynamic tiering. In *Proceedings of the 9th USENIX Conference on File and Stroage Technologies* (Berkeley, CA, USA, 2011), FAST'11, USENIX Association, pp. 20–20.

[16] Harter, T., Borthakur, D., Dong, S., Aiyer, A., Tang, L., Arpaci-Dusseau, A. C., and Arpaci-Dusseau, R. H. Analysis of hdfs under hbase: A facebook messages case study. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)* (Santa Clara, CA, 2014), USENIX, pp. 199–212.

[17] Kim, H., Seshadri, S., Dickey, C. L., and Chiu, L. Evaluating phase change memory for enterprise storage systems: A study of caching and tiering approaches. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)* (Santa Clara, CA, 2014), USENIX, pp. 33–45.

[18] KRISH, K., ANWAR, A., AND BUTT, A. hats: A heterogeneity-aware tiered storage for hadoop. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on* (May 2014), pp. 502–511.

[19] LI, H., WU, C., LI, Z., AND LAU, F. Profit-maximizing virtual machine trading in a federation of selfish clouds. In *INFOCOM, 2013 Proceedings IEEE* (April 2013), pp. 25–29.

[20] LI, Z., MUKKER, A., AND ZADOK, E. On the importance of evaluating storage systems' $costs. In *Proceedings of the 6th USENIX Conference on Hot Topics in Storage and File Systems* (Berkeley, CA, USA, 2014), HotStorage'14, USENIX Association, pp. 6–6.

[21] LIU, Z., LIU, I., LOW, S., AND WIERMAN, A. Pricing data center demand response. *SIGMETRICS Perform. Eval. Rev. 42*, 1 (June 2014), 111–123.

[22] LONDOÑO, J., BESTAVROS, A., AND TENG, S.-H. Colocation games: And their application to distributed resource management. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing* (Berkeley, CA, USA, 2009), HotCloud'09, USENIX Association.

[23] LU, Y., SHU, J., AND ZHENG, W. Extending the lifetime of flash-based storage through reducing write amplification from file systems. In *Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13)* (San Jose, CA, 2013), USENIX, pp. 257–270.

[24] NARAYANAN, D., THERESKA, E., DONNELLY, A., ELNIKETY, S., AND ROWSTRON, A. Migrating server storage to ssds: Analysis of tradeoffs. In *Proceedings of the 4th ACM European Conference on Computer Systems* (New York, NY, USA, 2009), EuroSys '09, ACM, pp. 145–158.

[25] SHI, W., ZHANG, L., WU, C., LI, Z., AND LAU, F. C. An online auction framework for dynamic resource provisioning in cloud computing. In *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems* (New York, NY, USA, 2014), SIGMETRICS '14, ACM, pp. 71–83.

[26] SHVACHKO, K., KUANG, H., RADIA, S., AND CHANSLER, R. The hadoop distributed file system. In *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)* (Washington, DC, USA, 2010), MSST '10, IEEE Computer Society, pp. 1–10.

[27] WANG, H., AND VARMAN, P. Balancing fairness and efficiency in tiered storage systems with bottleneck-aware allocation. In *Proceedings of the 12th USENIX Conference on File and Storage Technologies (FAST 14)* (Santa Clara, CA, 2014), USENIX, pp. 229–242.

[28] WU, G., AND HE, X. Delta-ftl: Improving ssd lifetime via exploiting content locality. In *Proceedings of the 7th ACM European Conference on Computer Systems* (New York, NY, USA, 2012), EuroSys '12, ACM, pp. 253–266.

[29] ZAHARIA, M., BORTHAKUR, D., SEN SARMA, J., ELMELEEGY, K., SHENKER, S., AND STOICA, I. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proceedings of the 5th European Conference on Computer Systems* (New York, NY, USA, 2010), EuroSys '10, ACM, pp. 265–278.