Personalized Payments for Storage-as-a-Service

Sofia Ceppi
Microsoft Research
soceppi@microsoft.com

Ian Kash
Microsoft Research
iankash@microsoft.com

ABSTRACT

Current storage offerings provide a small number of options in the form of fixed prices with volume discounting. This leaves storage operators to guess how much data customers will add over time. Instead, we propose that the operator elicits basic information about future usage. Such information can be used to operate the system more efficiently. In particular, we show how prices can be calculated that encourage customers to accurately report the range of their future usage while ensuring that the operator covers his costs.

1. INTRODUCTION

The current state of the art for pricing of cloud storage, both in research and in products of all major providers like Amazon Web Services and Microsoft Azure is that the storage operator offers a fixed price per gigabyte (or in some cases for fixed amounts of data), with prices per unit that decrease as customers buy more. For example, *Amazon S3* offers the following contract for its standard storage service: customers pay $0.03 per GB for the first TB/month, $0.0295 per GB for the next 49 TB/month, $0.0290 per GB for the next 450TB/month, and so on. By offering this type of contract, storage operators gain very limited information about how their customers intend to use the storage and how many GB they will use. Indeed, even if the price changes depending on the usage, no information about future consumption are available and, at the end of the month, a customer's pricing tier is determined by what she have actually used during the month. This lack of information makes difficult for the storage operator to forecast how the resources will be used and thus plan the storage activities. Moreover, since actual needs of customers are not taken into account, the service offered to them cannot be personalized, in terms of, e.g., type of storage provided and price per gigabyte.

Our thesis is that the status quo represents a missed opportunity for providers to elicit more information from customers. There is a wide variety of information that customers know that would help the storage operator, but which there is currently no way to express. Is there a product launch or change from development to release coming? How often are you expecting to access yout data? Will you business be affected by a delay in accessing your data? To understand the impact that this information would have consider the case of a new customer. She needs to be allocated to some physical storage rack. One option is to allocate her to an existing rack, but if usage grows beyond the capacity of the rack some customer's data needs to be either migrated or split across racks. Alternatively, she can be allocated to a new rack, but this requires buying additional hardware while not using some of what the storage already has. Any of these options is expensive for the storage if it is not the best one given the specific customer's requirements. Lacking any information from the customer, the storage operator is reduced to using crude statistical average and rules of thumbs to decide which options to choose. This is just one of many examples where more information could help. Others include understanding if there will be enough storage bandwidth to meet customer demand and, if not, who should be prioritized, as well as what data is hot or cold. Thus, the more information about customer intentions is available, the more efficiently storage resources can be organized, resulting in lower operating costs, greater social welfare, and lower prices for customers or increased revenue for the storage operator.

In this work, we take a first step toward the personalization of storage payments. In particular, our aim is to learn the probability distribution of future storage usage so that the storage operator can manage resources accordingly. We could request full distributional information using techniques from scoring rules [5], but it is unreasonable to assume that customers are able to communicate such information. On the other hand, simple statistics of it like the lower and the upper bounds of this distribution can be estimated more easily by customers and, given these, the storage operator can use internal models and historical data about customers behavior to transform the bounds into a plausible distribution of future usage. This observation motivates us to ask each customer to report a lower bound and an upper bound for the storage units she will use, where a storage unit can be, e.g., a gigabyte. For the sake of simplicity, in this paper we assume that customers have to specify the exact number of gigabytes for the lower and the upper bound. However, this assumption can be easily relaxed to coarse estimates without meaningful loss of information for the storage operator, as we discuss in Section 5.

To encourage customers to accurately provide the requested information, we should reward them such that more accurate information leads to better prices. Other than directly affecting the payments, misreports have negative effects on the storage. Indeed, it can be better to have no information at all rather than plan the storage activities on the basis of inaccurate information. Consider the example described before were a new customer needs to be allocated to some physical storage rack. As discussed, the storage operator faces several options and without information he has to use, e.g., statistical averages to decide which one to choose. However, inaccurate information can have even a worse effect on the storage operators choice who can choose the worst option while believing it is the best one.

Given this, the fundamental problem is to provide customers with the right incentives to accurately report their information. To understand the technical challenge we solve, suppose a customer reports that she will use between 5 and 15 GB, but knows she will use between 10 and 20. We will never observe her using less than 10, so prices must be charged such that she does not benefit from reporting 5 instead. We will observe her using more than 15 only probabilistically. But, and this is the key issue, even when we do observe this, we have no way of knowing whether her true upper bound was 20 or 200. Thus, we need to find penalties that in expectation discourage this.

In this work we consider two types of contracts for personalized payments: flexible and fixed. The flexible contract mirrors current pricing schema where the customer pays at the end of the contract for the actual storage consumption. However, different from current practice, in our work, the price per storage unit is customized based on the information the reported by the customer. Thus, when the customer chooses the flexible contract, the storage operator immediately computes the personalized price per storage unit and the final price the customer pays at the end of the contract is equal to this price per storage unit times the number of storage units she actually uses. In doing so, we gain the benefit of additional information with minimal disruption.

The fixed contract is an option not available in the current pricing model, but very natural in the approach we envision. When a customer chooses this contract, she pays in advance the total price for the required service. Given that information about the actual storage usage is not available at the beginning of the contract, in computing the final payment, instead of the actual customer's consumption its estimation (done by the storage operator on the basis of the reported information and, e.g., the customer's history is used. Thus, as before, the storage operator computes the personalized price per storage unit, but the final payment is equal to this price times the estimated amount of storage units the customer will use. We remark that the introduction of fixed contracts bring advantages especially because this contract can satisfy customer needs unfulfilled by existing pricing mechanisms. Suppose you (or a company or government agency) had a research project that would generate data for 2 years and then you need to retain it for 8 years, and want to get that for a fixed up-front payment out of your current grants (budget) that expires in 3 years. With current contract options, using cloud storage is not a feasible solution because you will have no money to pay it after the third year and you will ended up by choosing the more expense solution of buying your own hardware. With the fixed contract type we propose, cloud storage becomes a feasible and cheaper option.

In summary, our contributions are:

1. a model of how the storage operator can make use of additional information;
2. the design of personalized payments and penalties that incentivize customers to reveal their information;
3. the introduction of a new type of contract, the fixed contract, where the payment is done upfront.

2. THE MODEL

In this section we describe how the storage operator uses the information provided by the customer. We start by presenting a simple example that we use throughout the paper to show our results. These are then generalized in Section 5.

Assume that time is divided into periods from 1 to T and that the number of storage units used by the customer cannot decrease with time, i.e. the storage units used at t are less than or equal to the ones the customer uses at $t+1$, for all $t \in [1, T-1]$. We believe this assumption is reasonable because, since we study problems related to cloud storage (and not cloud computing), customers are more likely to use the service for a long-term file storage and thus the amount of data stored in such settings increases with time.

Let l denote the lower bound and u the upper bound of the amount of storage units the customer uses. The minimum value that l can have is l_{min} and the maximum value for u is u_{max}. (l, u) is the private information of the customer that we aim to elicit, i.e., we aim to obtain $\hat{l} = l$ and $\hat{u} = u$, where $\hat{l}$ and $\hat{u}$ are the reported lower bound and upper bound, respectively. Given these reported bounds, the storage operator can estimate the number of storage units the customer will use at each time period t. Essentially, we assume that the storage operator knows the probability distribution of the random variable R_t representing the increment in the number of storage units with respect to the reported lower bound at time t. Note that, as the goal is to incentivize truthful reporting, we assume that $R_t \in [0, u-l], \forall t \in [1, T]$. Thus, the storage operator can estimate the number of storage units the customer will use at t and we define the random variable representing this as $S_t = l + R_t$. Let $S = \frac{\sum_{t \in \{1 \dots T\}} S_t}{T}$ be the estimated average usage over the relevant time window. We denote with r_t the realization of R_t, with $s_t = l + r_t$ the realization of S_t, and with s the realization of S. (In general,we use capital letters for random variables and the lower case version for the realization.)

Intuitively, it may not be in the best interest of the storage operator to allocate at each time period t exactly the number of storage units the customer is expected to use, i.e., $\mathbb{E}[S_t]$. Indeed, to be flexible enough to face variation of the predicted behavior of the customer, at t the storage operator should allocate more storage units than $\mathbb{E}[S_t]$. For example, by allocating more space, the storage operator can allow the customer's usage to grow over future time periods without needing to split up or move the customer's data. Of course, keeping storage in reserve is not free, so the storage operator should be judicious in the additional amount he allocates. This is why we do not consider the dummy model where all the storage space requested by a customer is allocated at time t_1. Denote with $q_t(\hat{l}, \hat{u})$ the function that, given the reported bounds, defines how many storage units to allocate at t for a given customer, and let $q(\hat{l}, \hat{u}) = \frac{\sum_{t \in \{1 \dots T\}} q_t(\hat{l}, \hat{u})}{T}$. In this paper, we model the storage space reserved for a customer at time period t as the reported lower bound plus the estimated number of storage units she will use at $t+1$. Formally, $q_t(\hat{l}, \hat{u}) = \hat{l} + \mathbb{E}[R_{t+1}]$. Of course, at time t, the customer may well end up using more or less storage than $q_t(\hat{l}, \hat{u})$. However, if a number of customers are sharing the same storage and their usage is independent, this will tend to average out. Thus, allocating for each customer her expected storage units is reasonable as a simple model, and of course additional storage could be reserved without significantly affecting our analysis. To illustrate this idea, we consider a specific example where $T = 2$. In the next sections we analyze this example, and later discuss how it generalizes.

EXAMPLE 1. *We consider time divided into two periods:* t_1 *and* t_2 *and we assume that* $q_t(\hat{l}, \hat{u})$ *is computed as bescribed above. Thus,* $q_{t_1}(\hat{l}, \hat{u}) = \hat{l} + \mathbb{E}[R]$ *and* $q_{t_2}(\hat{l}, \hat{u}) = l + r$ *because* $t_2 = T$ *and at the last time period the storage operator needs to allocate only the storage units the customer uses. For the sake of simplicity we use* $R = R_T$ *and* $r = r_T$.

We assume that the model the storage operator uses is that exactly l *units are used at* t_1*. The storage increment from* t_1 *to* t_2 *is assumed to be drawn uniformly at random* ($R \sim \mathcal{U}(0, u-l)$)*. Thus* $\mathbb{E}[R] = \frac{\hat{u}-\hat{l}}{2}$, $\mathbb{E}[S] = \frac{1}{2}(\hat{l} + \hat{l} + \mathbb{E}[R]) = \frac{3\hat{l}+\hat{u}}{4}$*, and* $q(\hat{l}, \hat{u}) = \frac{1}{2}(\hat{l} + \mathbb{E}[R] + \hat{l} + \mathbb{E}[R]) = \frac{\hat{l}+\hat{u}}{2}$.

	s_t	$\mathbb{E}[S_t]$	$q_t(\hat{l}, \hat{u})$
t_1	l	$\hat{l}$	$\hat{l} + \mathbb{E}[R]$
t_2	$l + r$	$\hat{l} + \mathbb{E}[R]$	$l + r$

Table 1: Used, predicted, and allocated storage units.

Note that our running example is somewhat degenerate, in that there is actually no uncertainty over l. Moreover, we assume that the intervals are drawn from a uniform distribution. These assumptions allow our initial analysis to expose the main intuition for our results without added complexity. In Section 5, we relax these assumptions.

3. FLEXIBLE CONTRACT

When the customer chooses a flexible contract, at the moment of stipulating it she has to specify the lower bound $\hat{l}$ and the upper bound $\hat{u}$ on the storage units she will use until the end of the contract. Given this information, the storage operator estimates the price per storage unit $p(\hat{l}, \hat{u})$ the customer has to pay if she does not violate any of the reported bounds. This price is part of the contract. The actual payment is due when the contract expires, i.e., at the end of time T, because it depends on the actual storage units the customer uses. Indeed, the payment $P_{flex}(\hat{l}, \hat{u})$ of a customer is equal to the price per storage unit times the storage units she uses.

As explained in the previous section, for each time period t the storage operator may decide to allocate more storage units than the ones he predicts the customer will use. While the allocated extra storage units guarantee a more efficient service to the customer, the storage provider does not want to make a loss by using this allocation model. Thus, the customer should also cover the cost of the extra allocated storage units. This implicitly suggests to set the price per storage unit $p(\hat{l}, \hat{u})$ such that it compensates for the cost of the overall allocated storage units. Note that charging customers for extra storage is a common practice. However, differently from what we propose, typically this is done by rules of thumbs and baked into the prices charged.

Formally, the random variable representing the payment of a customer is

$$P_{flex}(\hat{l}, \hat{u}) = \sum_{t \in T} s_t \cdot p(\hat{l}, \hat{u}) = T \cdot s \cdot p(\hat{l}, \hat{u})$$

where $p(\hat{l}, \hat{u})$ is the function that given the reported bounds computes the payment per storage unit the customer will pay. The expected payment of a customer computed when she signs the contract is

$$\mathbb{E}[P_{flex}(\hat{l}, \hat{u})] = \sum_{t \in T} \mathbb{E}[S_t] \cdot p(\hat{l}, \hat{u}) = T \cdot \mathbb{E}[S] \cdot p(\hat{l}, \hat{u})$$

We now formally describe how the price per storage unit for a given customer is computed. To guarantee that the payment of the customer covers also the cost of the extra allocated storage units, in computing $p(\hat{l}, \hat{u})$ the cost c of a single storage unit, e.g., the cost or a storage rack divided by its number of storage units, is increased of a factor $\rho(\hat{l}, \hat{u})$. Note that c is independent of the customer's reported information and that it is constant because economies of scale are already priced in[1]. Formally, $p(\hat{l}, \hat{u}) = \rho(\hat{l}, \hat{u}) \cdot c$ where $\rho(\hat{l}, \hat{u})$ satisfies $\rho(\hat{l}, \hat{u}) \cdot c \cdot \mathbb{E}[S] \cdot T = c \cdot q(\hat{l}, \hat{u}) \cdot T$. Thus, $\rho(\hat{l}, \hat{u}) = \frac{q(\hat{l}, \hat{u})}{\mathbb{E}[S]}$. For Example 1, $\rho(\hat{l}, \hat{u}) = \frac{2\hat{l} + 2\hat{u}}{3\hat{l} + \hat{u}}$, and $P_{flex}(\hat{l}, \hat{u}) = T \cdot S \cdot \frac{2\hat{l} + 2\hat{u}}{3\hat{l} + \hat{u}} \cdot c$.

3.1 Incentives

In this section, we examine when the customer has no incentive to report bounds that differ from the actual ones. In computing the payment the only parameter that is affected by the customer's information, and thus can be manipulated, is $p(\hat{l}, \hat{u})$. Thus, when we want to investigate if and how a customer is incentivized to report bounds $\hat{l}$ and $\hat{u}$ different from l and u, we need to focus only on $p(\hat{l}, \hat{u})$.

A customer has no incentive to misreport the bounds if her payment is minimized when $\hat{l} = l$ and $\hat{u} = u$. Thus, she has no incentive to misreport $(\hat{l}, \hat{u})$ if $\rho(l, u) \leq \rho(\hat{l}, \hat{u})$. As we observe in the following theorem, this condition is satisfied when $\hat{l} < l$ and $\hat{u} > u$. Proofs of all theorems are reported in the full version of the paper.

THEOREM 1. *In the scenario described in Example 1, customers have no incentives to report $\hat{l} < l$ or $\hat{u} > u$.*

However, customers may still be motivated to report $\hat{l} > l$ or $\hat{u} < u$. To overcome this, the storage operator can charge customers a penalty every time a violation of one of the bounds is observed, as discussed in the next section.

3.2 Penalties

In this section, we describe how penalties charged to customers who report $\hat{u} < u$ can be computed. In order to guarantee a truthful report from the customer, it would be enough to charge her an infinite penalties every time she violates the contract. However, we aim to compute a tighter penalty and prove that it is possible to define a reasonable upper bound to the penalty independent from the specific instance of the problem. This is important because no real customer would accept a contract with an infinite penalty.

We omit discussion of how to compute penalties for $\hat{l} > l$ because in our example there is no uncertainty about $\hat{l}$ so penalizing misreports is trivial. Note that this restriction is due to the example considered and not to the model presented. In Section 5, we argue that a symmetric approach to the one here proposed can be used where there is uncertainty over $\hat{l}$.

THEOREM 2. *In the scenario described in Example 1, to guarantee that a customer who reports $\hat{u} < u$ is charged an expected payment higher than the one she would get by reporting u, the storage operator can set her total payment equal to*

$$\rho_{max} \cdot c \cdot \sum_{t \in [T]} s_t + 2 \cdot N_{flex} \cdot (s_T - \hat{u})$$

where N_{flex} is an upper bound of the gain the customer has by misreporting and ρ_{max} is the highest possible ρ.

4. FIXED CONTRACT

In the previous section, we examined a form of contract similar to current contracts in that customers are quoted a price per unit of storage they use. In this section, we explore an alternative, where customers can be quoted a fixed price up front regardless of their actual usage (as long as it is within their reported bounds). As previously discussed, in our running example there is only uncertainty about the upper bound $\hat{u}$. Thus, on the basis of l (which the storage

[1] That is, we assume the operator buys so much storage that its price per unit is constant. Alternatively, this can be viewed as an assumption that customers are small enough that the marginal cost of serving anyone is a constant per unit.

operator can easily verify) and $\hat{u}$, the storage operator estimates the price per storage units the customer has to pay, i.e., $p(l, \hat{u})$, and the amount of storage units she will use, i.e., $\mathbb{E}[S]$. Their product is the price that the customer has to pay when she signs the contract. Formally:

$$P_{fix}(l, \hat{u}) = T \cdot \mathbb{E}[S] \cdot p(l, \hat{u})$$

As before, since more storage units are allocated than are actually used, the price $p(l, \hat{u})$ is computed such that the customer's payment covers this extra cost. Thus, $p(l, \hat{u})$ and $\rho(l, \hat{u})$ are computed exactly as in the flexible contract case. Given this, we can write the payment as

$$P_{fix}(l, \hat{u}) = T \cdot \mathbb{E}[S] \cdot \frac{q(l, \hat{u})}{\mathbb{E}[S]} \cdot c = T \cdot q(l, \hat{u}) \cdot c$$

Note that, $P_{fix}(l, \hat{u})$ does not depends on the realizations and is not in expectation because at the moment of signing the contract the exact payment that the customer makes is known. For Example 1, $P_{fix}(l, \hat{u}) = (l + \hat{u}) \cdot c$.

Indeed, the main difference between the flexible contract and the fixed contract is that the payment of the latter is in fact fixed as long as the customer does not violate the requirements. In contrast, for the flexible contract case, at the beginning of time t_1, only the cost of the storage unit is fixed while the total payment of the customer) even if she does not violate the contract) is known only at the end of time T.

4.1 Incentives

In this section, we study when customers have an incentive to report an upper bound different from the actual one. It is easy to observe that the payment of our example increases when $\hat{u}$ increases. Thus, the customer has no incentives to report $\hat{u} > u$. This proves the following theorem.

THEOREM 3. *In the scenario described in Example 1, customers have no incentive to report $\hat{u} > u$.*

While we have the same incentives for reporting regarding $\hat{u}$ that we had in the flexible case, the incentives regarding $\hat{l}$ (if it needed to be reported) are different. Indeed, from the fixed payment of Example 1, we deduce that the payment increases when $\hat{l}$ increases. Thus, the customer has no incentives to report $\hat{l} > l$.

4.2 Penalties

Given the discussion in the previous section, our only remaining concern is preventing customers from reporting $\hat{u} < u$. Here, we show how to penalize a customer if the storage operator observes her misreport. Note that, as for the flexible contract, an infinite penalty would guarantee a truthful elicitation. However, we aim to provide a reasonable upper bound to the penalty that allows it to be finite (which seems essential in any practical proposal).

THEOREM 4. *In the scenario described in Example 1, to guarantee that a customer who reports $\hat{u} < u$ is charged a payment higher than the one she would get by reporting u, the storage operator can set her total payment equal to*

$$\rho_{max} \cdot c \cdot \sum_{t \in [T]} s_t + 2 \cdot N_{fix} \cdot (s_T - \hat{u})$$

where N_{fix} is an upper bound of the gain the customer has by misreporting and ρ_{max} is the highest possible ρ.

5. GENERALIZATION

All the resulted presented in the paper can be generalized to (i) the case where there is uncertainty over the minimum amount of storage used by the customer, i.e., when $\mathbf{E}[S_{t_1}] = \hat{l} + \mathbf{E}[R_1]$, (ii) the case with more than two time periods, (iii) the case in which lower and upper bounds are specified is a coarse way, and (iv) the case where the increments in storage units is not drawn from the uniform distribution. For a detailed discussion see the full version of the paper.

6. RELATED WORK

While, to our knowledge, no one has studied cloud pricing as we propose, a variety of related questions have been studied. The closest is by Xu and Li [9], who studied techniques such as throttling and performance guarantees to maximize revenue given a payment scheme that charges a fixed price per storage unit. Abhishek, Kash, and Key [1] and Borgs, Chayes, Doroudi, Harchol-Balter, and Xu [2] studied the design of auctions for spots in a queue, with an application to spot markets for compute resources where customers can bid the amount they are willing to pay. Jain, Menache, Naor, and Yaniv [7] studied mechanism design for scheduling tasks in a cloud system.

Our techniques rely on detecting certain types of misreports and then punishing agents who have done so, which makes them a case of mechanism design with partial verification, which has been studied in a variety of contexts [6, 8, 4].We can only catch these reports probabilistically, a version studied by Caragiannis, Elkind, Szeged, and Yu [3]. An alternative approach would be to charge users a fixed amount per storage unit and then incentivize accurate reporting using a scoring rule [5]. However, this has the downside that the score payment is more difficult to explain up front.

7. REFERENCES

[1] V. Abhiskeh, I. A. Kash, and P. Key. Fixed and market pricing for cloud services. In *NetEcon, Full working paper version on arXiv http://arxiv.org/abs/1201.5621*. NetEcon, 2012.

[2] C. Borgs, J. T. Chayes, S. Doroudi, M. Harchol-Balter, and K. Xu. Pricing and queueing. *SIGMETRICS Perform. Eval. Rev.*, 40(3):71–73, 2012.

[3] I. Caragiannis, E. Elkind, M. Szegedy, and L. Yu. Mechanism design: From partial to probabilistic verification. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 266–283, 2012.

[4] D. Fotakis, P. Krysta, and C. Ventre. Combinatorial auctions without money. *CoRR*, abs/1310.0177, 2013.

[5] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

[6] J. R. Green and J.-J. Laffont. Partially Verifiable Information and Mechanism Design. *Review of Economic Studies, Wiley Blackwell*, 53(3):447–56, 1986.

[7] N. Jain, I. Menache, J. Naor, and J. Yaniv. A truthful mechanism for value-based scheduling in cloud computing. pages 388–406, 2014.

[8] P. Krysta and C. Ventre. Combinatorial auctions with verification are tractable. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, pages 39–50, 2010.

[9] H. Xu and B. Li. A study of pricing for cloud resources. *SIGMETRICS Perform. Eval. Rev.*, 40(4):3–12, Apr. 2013.