

Newcastle University e-prints

Date deposited: 16th April 2012

Version of file: Author final

Peer Review Status: Unknown

Citation for item:

Mihoob A, Molina-Jimenez C, Shrivastava S. [A case for consumer-centric resource accounting models](#). In: *3rd International Conference on Cloud Computing (CLOUD)*. 2010, Miami, Florida, USA: IEEE.

Further information on publisher website:

<http://ieeexplore.ieee.org>

Publisher's copyright statement:

© IEEE, 2010

'Authors and/or their employers shall have the right to post the accepted version of IEEE-copyrighted articles on their own personal servers or the servers of their institutions or employers without permission from IEEE.'

The definitive version of this article is available at:

<http://dx.doi.org/10.1109/CLOUD.2010.44>

Always use the definitive version when citing.

Use Policy:

The full-text may be used and/or reproduced and given to third parties in any format or medium, without prior permission or charge, for personal research or study, educational, or not for profit purposes provided that:

- A full bibliographic reference is made to the original source
- A link is made to the metadata record in Newcastle E-prints
- The full text is not changed in any way.

The full-text must not be sold in any format or medium without the formal permission of the copyright holders.

**Robinson Library, University of Newcastle upon Tyne, Newcastle upon Tyne.
NE1 7RU. Tel. 0191 222 6000**

A Case for Consumer-centric Resource Accounting Models

Ahmed Mihoob, Carlos Molina-Jimenez and Santosh Shrivastava
School of Computing Science
Newcastle University
Newcastle upon Tyne, NE1 7RU, UK
{a.m.mihoob, carlos.molina, santosh.shrivastava}@ncl.ac.uk

Abstract—A pay-per-use cloud service should be made available to consumers with an unambiguous resource accounting model that precisely describes all the factors that are taken into account in calculating resource consumption charges. The paper proposes the notion of consumer-centric resource accounting model such that consumers can programmatically compute their consumption charges of a remotely used service. In particular, the notion of strongly consumer-centric accounting model is proposed that requires that all the data needed for calculating billing charges can be collected independently by the consumer (or a trusted third party, TTP); in effect, this means that a consumer (or a TTP) should be in a position to run their own measurement service. Strongly consumer-centric accounting models have the desirable property of openness and transparency, since service users are in a position to verify the charges billed to them. To illustrate the ideas, the accounting model of a given cloud infrastructure service (simple storage service, S3 from Amazon) is evaluated. The exercise reveals some shortcomings which can be fixed as indicated in this paper to make Amazon's model strongly consumer-centric. Service providers can learn from this evaluation study to re-examine their accounting models and perform any amendments.

Keywords-cloud computing; billing; resource accounting.

I. INTRODUCTION

Cloud computing service providers deploy, host and manage services with the intention of renting them to remote consumers (enterprises or individuals). The nature of the services ranges from providing basic computational resources such as storage, bandwidth and compute power ("infrastructure as a service") to sophisticated enterprise application services ("software as a service"). Consumers access a given service through a well defined interface and under the terms and conditions stipulated in a Service Level Agreement (SLA) contract. Among other business information, such an agreement stipulates the quality of the service expected from the provider and the charging schema. A **charging schema** that is widely used by providers is pay-per-use where the consumer uses as many resources as needed and is billed by the provider for the amount of consumed resources at the end of an agreed upon period. Ideally, consumers should be provided with an unambiguous **resource accounting model** that precisely describes all the factors that are taken into account in calculating resource consumption charges.

An accounting model for resource consumption should describe how charges are calculated from the resource usage

data. For a given resource, the model should include a description of all the parameters of resource usage that are measured, measurement times, the frequency of the measurement, the start and end of the accountable period and other relevant information that would be needed by a measurement service to collect the resource usage data (resource consumption data) that forms the basis for billing. Availability of such information will empower consumers in several ways, such as:

- 1) selecting a suitable service provider;
- 2) making their applications billing aware;
- 3) plan organization's budgets for IT billing;
- 4) create third party brokering services that automate resource provision in line with customer's needs.

Clearly, implementing any of the above functionalities will require that consumers have access to resource usage data. An important issue then is the **accountability of the resource usage data**: who performs the measurement to collect resource usage data – the provider, the consumer, a trusted third party (TTP), or some combination of them? Provider-side accountability is the norm for the traditional utility providers such as water, gas and electricity who make use of metering devices (trusted by consumers) that are deployed in the consumers' premises. Currently, provider-side accountability is also the basis for cloud service providers, although, as yet there are no equivalent facilities of consumer-trusted metering; rather, consumers have no choice but to take whatever usage data made available by the provider as trustworthy.

In light of the above discussion, we propose the notion of a Consumer-centric Resource Accounting Model for a cloud resource. We say that an accounting model is **weakly consumer-centric** if all the data that the model requires for calculating billing charges can be queried programmatically from the provider. Further, we say that an accounting model is **strongly consumer-centric** if all the data that the model requires for calculating billing charges can be collected independently by the consumer (or a TTP); in effect, this means that a consumer (or a TTP) should be in a position to run their own measurement service. We contend that it is in the interest of the providers to make the accounting models of their services at least weakly consumer-

centric. Strongly consumer-centric models should prove even more attractive to consumers as they enable consumers to incorporate independent consistency/reasonable checks as well as raise alarms when apparent discrepancies are suspected in consumption figures; furthermore, innovative charging schemes can be constructed by consumers that are themselves offering third party services. Strongly consumer-centric accounting models have the desirable property of openness and transparency, since service users are in a position to verify the charges billed to them.

As a motivating example, consider a consumer who rents a storage service to run an application shown in Fig. 1. The storage is consumed by the consumer’s application and by applications hosted by other users ($user_1$, $user_2$, etc.) that access the storage service at the consumer’s expense. An example of this case is a consumer using a storage service to provide photo or video sharing services to other users. The ideal scenario is that the consumer is able to instrument the application to collect all the necessary storage consumption data and use the accounting model of the provider to accurately estimate the charges, and use that information to provide competitively priced service to users.

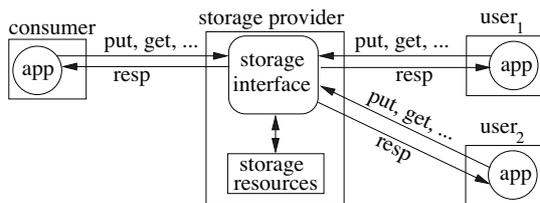


Figure 1. Provider, consumer and users of storage services.

Since cloud service providers do publish their charging information, it is worth investigating whether their information matches the proposed notion of customer-centric resource accounting model. With this view in mind, in this paper we evaluate the accounting model of a given cloud infrastructure service (simple storage service, S3 from Amazon) to see how well it matches the proposed notion. We independently collected (by examination of requests and responses) our own resource usage data for S3 and compared it with the provider’s data. Our investigations indicate that even though, it is a very simple service, the accounting model description nevertheless has a few ambiguities and not all the data that the model requires for calculating billing charges can be queried programmatically from the provider. In our opinion, most of these shortcomings can be fixed fairly easily by Amazon to make their model weakly as well as strongly consumer-centric. Service providers can learn from our evaluation study to re-examine their accounting models. In particular, we recommend that a cloud provider should go through the exercise of constructing a third party measurement service, and based on that exercise, perform any amendments to the model, remove potential sources of

ambiguities in the description of the model, so that as far as possible, consumers are able to collect with ease their own usage data that matches provider side data with sufficient precision.

II. AMAZON S3 SERVICES AND BILLING

Amazon advertises its S3 service as storage service available to Internet users on a pay-per-use basis [1]. Informally, it is promoted as a highly reliable, fast, inexpensive data storage service accessible to subscribers through a Web service interface. Currently, S3 provides SOAP and RESTful interfaces [2]. An S3 space is organised as a collection of buckets which are similar to folders, except that they do not support nesting. A bucket can contain zero or more objects of up to 5 GB; an object is simply a file uploaded by the customer from their local disk into their S3 space. Both buckets and objects are identified by names (keys in Amazon terminology) chosen by the customer.

To gain access to the service, customers need to open an account with S3, provide a credit card number and agree to pay the bill at the end of each **calendar month**. Upon successful registration, Amazon provides the customer with an account name, access key and secret key. The account name identifies an S3 storage space that is reachable to the customer from anywhere at any time and to anybody with whom they share their access and secret keys. An S3 customer is charged for: a) **storage space**: storage space consumed by the objects that they store in S3; b) **bandwidth**: network traffic generated by the operations that the customer executes against the S3 interface; and c) **operations**: number of operations that the customer executes against the S3 interface.

The information about pricing and the charging schema to calculate customers’ bill is spread in three documents available from Amazon Web Services pages: a) The Amazon Simple Storage Service (Amazon S3) page contains the prices; b) The Simple Storage Service FAQs contain pricing and examples of bill calculation; c) The Calculating Your Bill page that pops-up as a help window from within the Usage Reports associated to each S3 account, provides complementary information.

Prices vary slightly in accordance with the geographical region (US standard, US-West and European Union) where the customer’s data is physically located but the charging schema is the same for all regions. In Amazon’s pricing list, there is no reference to the time zone used by Amazon to determine the start and end of days and billing cycles. However, from the Authenticating SOAP Requests Section of the Amazon Developer Guide [2] it is clear that S3 servers are synchronised to the Universal Time Coordinated (UTC) which is also known as the Zulu Time (Z time) and in practice equivalent to the Greenwich Mean Time (GMT).

The key parameter in calculation of the storage bill is number of byte hours accounted to the customer. **Byte**

Hours (ByteHrs) is the the number of bytes that a customer stores in their account for a given number of hours. Thus if in a given month (say March) a customer stores 10 bytes for a single hour, their storage consumption for March would be $10 \times 1 = 10$ ByteHrs; similarly, if the customer stores 10 bytes for a whole day, their storage consumption for March would be $10 \times 24 = 240$ ByteHrs; likewise, if the customer stores 10 bytes for the 31 days (744 hrs) of March, the storage consumption for March would be $10 \times 744 = 7440$ ByteHrs.

From now on we will assume European customers accessing the S3 service from the 'outside Internet', that is, not from the 'inside Internet' within the Amazon web services (this would be the case for example, if an application ran on Amazon's compute cloud, EC2 and accessed S3). Current prices (in US dollars) read as follows:

- Storage cost: The first 50 TB costs 15 cents per GB per month.
- Bandwidth cost: Amazon distinguishes between DataTransfer-In and DataTransfer-Out (explained in SectionII-B). Currently there is no charge for DataTransfer-In through June 30th 2010; thereafter it will cost 10 cents per GB. The first 10 TB of DataTransfer-Out cost 15 cents per GB.
- Operations cost: A block of 1000 operations composed of PUT, COPY, POST or LIST costs one cent, whereas a block of 10000 GET and all other operations, excluding DELETE, costs one cent. Delete operations are free.

It is worth clarifying that with Amazon, prices decrease slightly as the consumption increases, for example, the next 50 TB of storage cost 14 cents per GB per month.

A. Charging Schema for Storage

In the FAQs questions Amazon explains that *the GB of storage billed in a month is the average storage used throughout the month. This includes all object data and metadata stored in buckets that you created under your account. We measure your usage in TimedStorage-ByteHrs, which are added up at the end of the month to generate your monthly charges.* Next, an example that illustrates how to calculate your bill if *you keep 2,684,354,560 bytes (or 2.5 GB) of data in your bucket for the entire month of March* is provided. In accordance with Amazon the total number of bytes consumed for each day of March is 2684354560; thus the total number of ByteHrs is calculated as $2684354560 \times 31 \times 24 = 1997159792640$, which is equivalent to 2.5 GBMonths. At a price of 15 cents per Giga Bytes per month, the total charge amounts to $2.5 \times 15 = 37.5$ cents.

Amazon explains that *at least twice a day, we check to see how much storage is used by all your Amazon S3 buckets. The result is multiplied by the amount of time passed since the last checkpoint.* Records of storage consumption in

ByteHrs can be retrieved from the Usage Reports associated with each account.

B. Charging Schema for Bandwidth

The Calculating Your Bill document explains that **DataTransfer-In** is the network data transferred from the customer to S3. They state that *Every time a request is received to put an object, the amount of network traffic involved in transmitting the object data, metadata, or keys is recorded here.* **DataTransfer-Out** is the network data transferred from S3 to the customer. They state that *Every time a request is received to get an object, the amount of network traffic involved in transmitting the object data, metadata, or keys is recorded here.* By here they mean that in the Usage Reports associated to each account, the amount of DataTransfer-In and DataTransfer-Out generated by a customer, is represented, respectively, by the DataTransfer-In-Bytes and DataTransfer-Out-Bytes parameters.

As an example, Amazon shows that if *You upload one 500 MB file each day during the month of March and You download one 500 MB file each day during the month of March* your bill for March (imagine 2011) will be calculated as follows. The DataTransfer-In would be $500MB \times (1/1024) \times 31 = 15.14GB$. At a price of 10 cents per Giga Bytes, the total charge would be $15.14 \times 10 = 151.4$ cents. In a second example they show that if *You download one 500 MB file each day during the month of March* the total amount of DataTransfer-Out would be 15.14 GB which charged at 15 cents per GB would amount to 227 cents.

C. Charging Schema for operations

To illustrate their charging schema they provide an example in the Amazon Simple Storage Service FAQs where *You transfer 1000 files into Amazon S3 and transfer 2000 files out of Amazon S3 each day during the month of March, and delete 5000 files on March 31st.* In this scenario, the total number of PUT request is calculated as $1000 \times 31 = 31000$, whereas the total number of GET requests is calculated as $2000 \times 31 = 62000$. The total number of DELETE requests is simply 5000 though this is irrelevant as DELETE requests are free. At the price of one cent per 1000 PUT requests and one cent per 10000 GET requests, the total charge for the operations is calculated as $31000 \times (1/1000) + 62000 \times (1/10000) = 37.2$ cents.

III. ERROR HANDLING

As explained in the Handling Errors section of [2], some operations might fail to complete successfully; the details of the error response depend on the interface (SOAP or RESTful) but in general it contains information that helps identify the party responsible for the failure: the customer or the S3 infrastructure. For example, *NoSuchBucket* errors are caused by the customer when they try to upload a file into a non-existent bucket; whereas an *InternalError* code indicates

that S3 is experiencing internal problems. Amazon advises developers to account for potential problems, for example, by considering request resends in their applications.

IV. S3 BILLING RECORDS

Among the on-line records that Amazon keeps of each S3 account, Amazon keeps a repository of two documents related to customers' bills, namely, Account Activity and Usage Reports. An **Account Activity** is a month's billing statement that contains the total charge for the corresponding month and a summary of the operations that the customer executed against S3 and their corresponding charges. Previous and current month's statements are available. A **Usage Reports** contains a detailed description of each individual operation executed by the customer during a given period.

V. SHORTCOMINGS IN THE S3 ACCOUNTING SCHEMA

In an attempt to audit our own S3 bill, we studied Amazon accounting model and conducted several experiments to see if we could collect our own usage data that matches what is recorded in Amazon's Usage Reports. We found out that as it is, the model suffers from a few omissions and ambiguities, as a result, it is possible for the usage data as collected by a user to be different from Amazon's Usage Reports. We examine the issues involved in computing charges for storage, operations and bandwidth.

A. Storage

From the definition of ByteHrs it follows that to calculate their bill, a customer needs to understand 1) how their byte consumption is measured, that is, how the data and metadata that is uploaded is mapped into consumed bytes in S3; and 2) how Amazon determines the number of hours that a given piece of data was stored in S3 —this issue is directly related to the notion of a checkpoint.

1) *Data and metadata*: The Billing section of Amazon Simple Storage Service FAQs stipulates that to calculate the storage consumption of a given customer, Amazon counts the number of bytes consumed by the data and metadata of all their objects.

The Developer Guide [2] (see also [3]) explains that each object in S3 has, in addition to its data, system metadata and user metadata; furthermore it explains that the **system metadata** is generated and used by S3, whereas **user metadata** is defined and used only by the user and limited to 2 KB of size. Unfortunately, Amazon does not explain how to calculate the actual storage space taken by data and metadata. To clarify this issue, we uploaded a number of objects of different names, data and user metadata into an equal number of empty buckets. Fig. 2 shows the parameters and results from one of our upload operations where an object named *Object.zip* is uploaded into a bucket named *MYBUCKET*, which was originally empty.

Notice that in this example, the object and bucket names are, respectively, ten and eight character long, which is equivalent to ten and eight bytes, respectively.

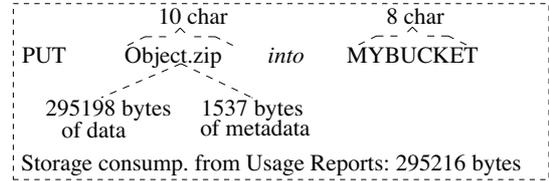


Figure 2. Impact of data and metadata on storage consumption.

The object data and metadata shown in the figure correspond to information we extracted locally from the PUT request. In contrast, the storage consumption of 295216 bytes corresponds to what we found in the Usage Reports. The actual Usage Reports show storage consumption per day in ByteHrs; the value shown is the result of its conversion into bytes. Notice that this storage consumption equals the sum of the object data, the length of the object name and the length of the bucket name: $8 + 10 + 295198 = 295216$.

Two conclusions can be drawn from this observation: first, the mapping between bytes uploaded by PUT requests and bytes stored in S3 correspond one to one; secondly, object and bucket names represent what Amazon calls storage overhead and incur storage consumption; third, user metadata does not impact storage consumption.

In addition to the experiments discussed above, we created a number of empty buckets and verified from the Usage Reports that they do not consume storage space.

2) *Checkpoints*: Amazon states that at least twice a day they check the amount of storage consumed by a customer. However, Amazon does not stipulate exactly when the checkpoints take place.

To clarify the situation, we conducted a number of experiments that consisted in uploading to and deleting files from S3 and studying the Usage Reports of our account to detect when the impact of the PUT and DELETE operations were accounted by Amazon. Our findings are summarised in Fig. 3. It seems that, currently, Amazon does not actually check customers' storage consumption twice a day as they specify in their Calculating Your Bill document, but only once. From our observations, it emerged that the time of the checkpoint is decided randomly by Amazon within the 00:00:00Z and 23:59:59Z time interval.

In the figure, CP stands for checkpoint, thus $CP_{30} : 2GB$ indicate that CP_{30} was conducted on the 30th day of the month at the time specified by the arrow and reported that at that time the customer had 2 GB stored in S3. SC stands for Storage Consumption and is explained below.

As shown in the figure, Amazon uses the results produced by a checkpoint of a given day, to account the customer for the 24 hrs of that day, regardless of the operations that

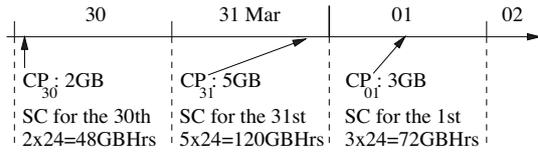


Figure 3. Amazon's checkpoints.

the customer might perform during the time left between the checkpoint and the 23:59:59Z hours of the day. For example, the storage consumption for the 30th will be taken as $2 \times 24 = 48$ GBHrs; where 2 represents the 2GB that the customer uploaded on the 30th and 24 represents the 24 hrs of the day.

The significance of knowing the specific point in time when the checkpoints are conducted is shown in Fig. 4.

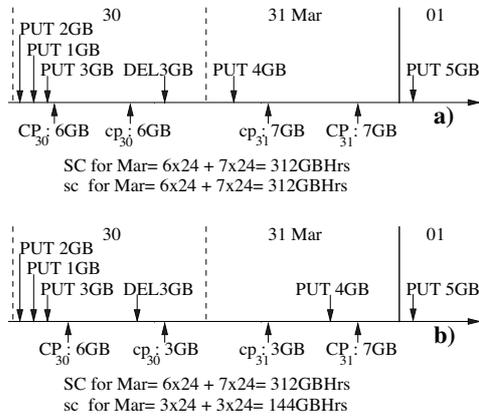


Figure 4. Impact of checkpoints.

The figure shows the execution time of four PUT and one DEL operations executed by an S3 customer during the last two days of March. The first day of April is also shown for completeness. For simplicity, the figure assumes that the earliest PUT operation is the very first executed by the customer after opening his S3 account. The figure also shows the specific points in time when checkpoints are conducted independently by two parties, namely, Amazon and a customer. Thus, CP and cp represent, respectively, Amazon's and the customer's checkpoints; the Giga Bytes shown next to CP and cp indicate the storage consumption detected by the checkpoint. For example, on the 30th, Amazon conducted its checkpoint about five in the morning and detected that, at that time, the customer had 6 GB stored ($CP_{30} : 6GB$). On the same day, the customer conducted his checkpoint just after midday and detected that, at that time, he had 6 GB stored ($cp_{30} : 6GB$). SC and sc represent, respectively, the storage consumption for the month of March, calculated by Amazon and customer, based on their checkpoints.

The figure demonstrates that the storage consumption

calculated by Amazon and customer might differ significantly depending on the number and nature of the operations conducted within the time interval determined by the two parties' checkpoints, for example, within CP_{31} and cp_{31} .

Scenario a) shows an ideal situation where no customer's operations are executed within the pair of checkpoints conducted on the 30th or 31st. The result is that both parties calculate equal storage consumptions. In contrast, b) shows a worse-case scenario where the DEL operation is missed by CP_{30} and counted by cp_{30} and the PUT operation is missed by cp_{31} and counted by CP_{31} ; the result of this is that Amazon and the customer, calculate SC and sc, respectively, as 312 GB and 144 GB.

B. Operations

One likely source of difficulty about the charges for operations is determining the liable party for failed operations. Currently, this decision is taken unilaterally by Amazon. In this regard, we anticipate two potential sources of conflicts: DNS and propagation delays. As explained by Amazon, some requests might fail and produce a Temporary Redirect (HTTP code 307 error) due to temporary routing errors which are caused by the use of alternative DNS names and request redirection techniques [3]. Amazon's advice is to design applications that can handle redirect errors, for example, by resending a request after receiving a 307 code (see [2], Request Routing section). Strictly speaking these errors are not caused by the customer as the 307 code suggests. It is not clear to us who bears the cost of the re-tried operations.

To offer high availability, Amazon replicates data across multiple servers within its data centres. Replicas are kept weakly consistent and as a result, some perfectly legal operations could sometime fail or return inaccurate results (see [2], Data Consistency Model section). For example, the customer might receive a *ObjectDoesNotExist* as a response to a legal GET request or an incomplete list of objects after executing a LIST operation. Some of these problems can be corrected by re-trying the operation. From Amazon accounting model, it is not clear who bears the cost of the failed operations and their retries.

We executed a number of operations including both valid and invalid ones (for example, creation of buckets with invalid names and with names that already existed). Next we examined the Usage Reports and as we expected, we found that Amazon counted both successful and failed operations. Fig. 5 shows an example of the operations that we executed and the bandwidth and operation consumptions that it caused in accordance with the Usage Reports.

Thus, the failed operation to create that bucket consumed, respectively, 574 bytes and 514 bytes of DataTransfer-In and DataTransfer-Out. These figures, correspond to the size of the SOAP request and response, respectively. As shown in the figure, we also found out that the failed

```

CREATE MYBUCKET // MYBUCKET already exists
Response: Error:BucketAlreadyExists
Bandwidth consump. (DataTransferIn) from
usage reports: 574 bytes
Bandwidth consump. (DataTransferOut) from
usage reports: 514 bytes
Operation consump. (RequestTier2) from
usage reports: 1

```

Figure 5. Bandwidth and operation consumption of failed operations.

operation incurred operation consumption and counted by the RequestTier2 parameter in the Usage Reports.

C. Bandwidth

As explained in Section II-B, DataTransfer-In and DataTransfer-Out include, respectively, request and response overhead. The difficulty here is that from Amazon accounting model, it is not clear how message size is calculated in DataTransfer-In and DataTransfer-Out. To clarify the point, we uploaded a number of files and compared information extracted from the PUT operations against bandwidth consumption as counted in the Usage Report. Two examples of the experiments that we conducted are shown in Fig. 6: we used PUT operations to upload an object into a bucket. The data and metadata shown in the figure represent the data and metadata extracted locally from the PUT requests.

```

      10 char      8 char
    PUT Object.zip into MYBUCKET
    295198 bytes 1537 bytes
    of data    of metadata
Bandwidth consump. (DataTransferIn) from
usage reports: 295198 bytes

      10 char      8 char
    PUT Object.zip into MYBUCKET
    0 bytes    2024 bytes
    of data    of metadata
Bandwidth consump. (DataTransferIn) from
usage reports: 0 bytes

```

Figure 6. Bandwidth consumption.

As shown by the Bandwidth consump. parameters extracted from the Usage Reports, only the object data consumes DataTransfer-In bandwidth; neither the metadata or the object or bucket names seem to count as overhead. This observation refers to RESTful requests. In contrast, for SOAP messages, the total size of the message is always used for calculating bandwidth consumption.

D. Making S3 accounting model consumer-centric

An S3 customer is charged for storage, bandwidth and operations performed. In the previous subsections we ex-

amined whether the data that the S3 accounting model requires for calculating billing charges can be collected independently by the consumer (or a TTP) with sufficient accuracy. Our investigations show that this is possible, provided the shortcomings that we have identified are fixed. In particular, concerning storage, the accounting model needs to explicitly state how the data and metadata that is uploaded is mapped into consumed bytes in S3 (for example, our experiments showed that user metadata does not impact storage consumption). We also pointed out clarifications that are required concerning measurement of operations performed and bandwidth.

Returning to storage, we saw that errors are possible if the checkpoint times of Amazon and the customer for calculating storage consumption are not close enough. Ideally, Amazon's checkpoint times should be made known to customers to prevent any such errors. Providing this information for upcoming checkpoints is perhaps not a sensible option for a storage provider, as the information could be 'misused' by a customer by placing deletes and puts around the checkpoints in a manner that artificially reduces the consumption figures. An alternative would be to make the times of past checkpoints available (e.g., by releasing them the next day).

VI. RELATED WORK

In [4] the author argues that cloud providers should make their services accountable for both the provider and the customer and discusses the technical challenges that accountability involves. In the context of charging, this implies the availability of charging models so that customers can audit their bills. In our paper we argue that current service providers do not address this issue satisfactorily and demonstrate that ambiguous accounting models can lead to discrepancies between customers and providers over consumption figures.

The suitability of Amazon S3, EC2 and SQS services as a platform for data intensive scientific applications is studied in [5]; the study focuses on performance (e.g. number of operations per second), availability and cost. It suggests that costs can be reduced by building cost-aware applications that exploit data usage patterns; for example, by favouring data derivation from raw data against storage of processed data. These arguments support the practical and commercial relevance of our study of charging models.

Conceptual ideas for building bilateral accounting systems are discussed in [6]. The authors develop a model in which the consumer and provider independently measure resource consumption, compare their outcomes and agree on a mutually trusted outcome. The paper discusses the technical issues that this matter involves, including consumer side collection of metering data, potential divergences between the two independently calculated bills, dispute resolution and non-repudiable sharing of resource usage records. Naturally,

a starting point for such a system will be consumer-centric accounting models of cloud resources.

The general principles of an architecture for accounting and billing in cloud services composed out of two or more federated infrastructures (for example, a storage and computation providers) is discussed in [7]. The architecture assumes the existence of well defined accounting models that are used for accounting resources consumed by end users and for accounting resources that the cloud provider consumes from the composing infrastructures. This issue is related to the scenario that we present in Fig.1. A related work is [8] where the authors develop the notion of a third party service management authority that can monitor interactions between customers and cloud resource vendors.

VII. CONCLUDING REMARKS

A pay-per-use cloud service should be made available to consumers with an unambiguous resource accounting model that precisely describes all the factors that are taken into account in calculating resource consumption charges. In this paper we have proposed the notion of consumer-centric resource accounting model. An accounting model is said to be weakly consumer-centric if all the data that the model requires for calculating billing charges can be queried programmatically from the provider. An accounting model is said to be strongly consumer-centric if all the data that the model requires for calculating billing charges can be collected independently by the consumer (or a TTP); in effect, this means that a consumer (or a TTP) should be in a position to run their own measurement service.

To illustrate the ideas, the accounting model of a given cloud infrastructure service (simple storage service, S3 from Amazon) is evaluated. The exercise reveals some shortcomings which can be fixed as indicated in this paper to make Amazon's model strongly consumer-centric.

We recommend that a cloud provider should go through the exercise of constructing a third party measurement service, and based on that exercise, perform any amendments to the model, remove potential sources of ambiguities in the description of the model, so that as far as possible, consumers are able to collect with ease their own usage data that matches provider side data with sufficient precision.

ACKNOWLEDGEMENT

This work has been funded in part by UK Engineering and Physical Sciences Research Council (EPSRC), Platform Grant No. EP/D037743/1, Networked Computing in Inter-organisation Settings. Discussions with Nick Cook clarified our understanding.

REFERENCES

[1] Amazon, "Amazon web services," 2006. [Online]. Available: www.amazon.com

[2] —, "Amazon simple storage service. developer guide, API version 2006-03-01," 2006. [Online]. Available: www.amazon.com

[3] J. Murty, *Programming Amazon Web Services*. O'Reilly, 2008.

[4] A. Haeberlen, "A case for the accountable cloud," in *3rd ACM SIGOPS: Int'l Workshop on Large-Scale Distributed Systems and Middleware (LADIS'09)*, 2009.

[5] M. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel, "Amazon s3 for science grids: a viable solution?" in *Intl Workshop on Data-Aware Distributed Computing (DADC'08)*, Jun 24, Boston, USA, 2008, pp. 55-64.

[6] C. Molina-Jimenez, N. Cook, and S. Shrivastava, "On the feasibility of bilaterally agreed accounting of resource consumption," in *1st Intl Workshop on Enabling Service Business Ecosystems (ESBE08)*, Sydney, Australia, 2008, pp. 170-283.

[7] E. Elmroth, F. G. Marquez, D. Henriksson, and D. P. Ferrera, "Accounting and billing for federated cloud infrastructures," in *The Eighth Intl Conf. on Grid and Cooperative Computing*, Aug. 27-28, Lanzhou, Gansu, China: IEEE Computer Society, 2009, pp. 268-275.

[8] A. Korn, C. Peltz, and M. Mowbray, "A service level agreement authority in the cloud," HP Laboratories, Tech. Rep. HPL-2009-79, 2009.