Consumer Side Resource Accounting in Cloud Computing

Declaration of Authorship

I declare that work contained in this thesis has not been submitted for any other award and that it is all my own work; any collaborative work has been explicitly acknowledged.

Name: Ahmed M Mihoob

Signed: ----------

Date : ----------

Acknowledgements

It is a pleasure to thank the people who have provided guidance and support during my time at the School of Computing Science at University of Newcastle. In particular, I would like to express my sincere gratitude to my PhD. supervisors Prof. Santosh Shirvastava and Dr. Carlos Molina, who have supported and encouraged me with some advises, solid knowledge, constrictive ideas, good company and lots of patience throughout my thesis work.

Many thanks, to the Cultural Affairs Department, Libyan Embassy, London, for their help and financial support. I would also thank everybody at the School of Computing Science who have contributed, directly or indirectly, to the success of this work.

I wish to acknowledge my greatest debt family, in particular my mother, father and my brother Muftha. They never stopped encouraging me to finish this thesis and they suffered most because of my academic interests.

Finally, heartfelt thanks go to my wife and my children who have stood behind me and given me huge support during the thesis work.

Abstract:

Cloud computing services made available to consumers range from providing basic computational resources such as storage and compute power to sophisticated enterprise application services. A common business model is to charge consumers on a pay-per-use basis where they periodically pay for the resources they have consumed. The provider is responsible for measuring and collecting the resource usage data. This approach is termed *provider-side accounting.* A serious limitation of this approach is that consumers have no choice but to take whatever usage data that is made available by the provider as trustworthy.

This thesis investigates whether it is possible to perform *consumer-side resource accounting* where a consumer independently collects, for a given cloud service, all the data required for calculating billing charges. If this were possible, then consumers will be able to perform reasonableness checks on the resource usage data available from service providers as well as raise alarms when apparent discrepancies are suspected in consumption figures. Two fundamental resources of cloud computing, namely, storage and computing are evaluated. The evaluation exercise reveals that the resource accounting models of popular cloud service providers, such as Amazon, are not entirely suited to consumer-side resource accounting, in that discrepancies between the data collected by the provider and the consumer can occur. The thesis precisely identifies the causes that could lead to such discrepancies and points out how the discrepancies can be resolved.

The results from the thesis can be used by service providers to improve their resource accounting models. In particular, the thesis shows how an accounting model can be made strongly consumer–centric so that all the data that the model requires for calculating billing charges can be collected independently by the consumer. 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.

List of Publications

Conference Papers

[1] Ahmed Mihoob, Carlos Molina-Jiménez: A Peer to Peer Protocol for Online Dispute Resolution over Storage Consumption, YR-SOC 2009, Pizza, EPTCS 2, 2009, pp, 3-14

[2] A. Mihoob, C. Molina-Jimenez, and S. Shrivastava, "A case for consumer–centric resource accounting models," in Proc. IEEE 3rd Int'l Conf. on Cloud Computing(Cloud'10), 2010, pp. 506–512.

[3] A. Mihoob, C. Molina-Jimenez, and S. Shrivastava, "Consumer side resource accounting in the cloud", in Proc. 11th IFIP WG 6.11 Conf. on e-Business, e-Services, and e-Society (I3E 2011), 2011, pp. 58–72.

Technical reports

[1] Mihoob A, Molina-Jiminez C, Shrivastava S. *A Case for Consumer–centric Resource Accounting Models.* Newcastle upon Tyne: University of Newcastle upon Tyne, 2010. School of Computing Science, Technical Report Series 1222.

[2] Mihoob A, Molina-Jimenez C, Shrivastava S. *Consumer Side Resource Accounting in the Cloud.* Newcastle upon Tyne: School of Computing Science, University of Newcastle upon Tyne, 2011. School of Computing Science, Technical Report Series 1259.

[3] Mihoob A, Molina-Jiminez C, Shrivastava S. A Case for Consumer-centric Resource Accounting Models. Newcastle upon Tyne: University of Newcastle upon Tyne, 2012. School of Computing Science, Technical Report Series 1318.

[3] Mihoob A. A Peer to Peer Protocol for Online Dispute Resolution over Storage Consumption. Newcastle upon Tyne: University of Newcastle upon Tyne, 2012. School of Computing Science, Technical Report Series TR1325.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Cloud computing service providers enable their customers to consume computing, storage and network resources and a variety of software services remotely via internet. Such resources are exposed as services through one or more service interfaces by the service provider. The consumers of these resources (individual users or organizations) consume these resources by invoking operations or methods at the services interfaces. Nowadays, there are many service providers offering different types services, and the number of the service providers and the type of services offered is large and continues to increase.

As a new business model, cloud computing technology has been the focus of growing research attention. Some researchers have paid attention to the development of middleware and platforms of cloud computing such as Amazon (EC2), Google (App Engine) and Microsoft (Windows Azure), whereas many others are focusing on the study of virtualization, cloud storage, cloud security, load balancing, quality of service monitoring, and so forth. However, issues related to the charging, accounting and billing of resources consumption have received less attention.

According to the charging model used, the service providers can apply either a fixed charge or Pay-Per-Use charge. The bill is fixed irrespective of the amount of resources consumed in the first charge model, whereas the bill depends on the amount of resources consumed in the second model. Pay-per-use services can be further categorised into capacity-on-demand service and consume-on-demand service [1, 2, 3]. Regarding the first, where a capacity-on-demand service consumer pays a fixed charge in advance for a fixed maximum non-exceedable capacity that is made available for their use. Such systems include Google E-mail systems and pay-as-you-go mobile phones. Consume-on-demand service can be additionally classified further into ‘on-demand’ and ‘utility services’. In the first case, the consumer pays (normally in

advance) for a fixed amount of resource (for example, 60 minutes of international phone calls) and the service is terminated when the consumer exhausts the resources. With the latter, the consumer consumes as much as he needs, when he needs it; the charge (or the bill) is calculated according to actual resource consumption and later presented to the consumer at the end of an agreed-upon accounting period. Amazon Simple Storage Service (Amazon's S3) [4] is a well-known example of a service provider that sells storage space to remote users and uses the consume-on-demand charging model to charge their customers.

Accounting of computing resources is the whole process that is required to calculate the charge of each consumed resource to produce the customer's bill for a well-defined period of time. This process includes collecting metering data, computing the resource consumption, and producing the final customer bill. It can also be used for other purposes such as auditing, monitoring and so forth [5]. Central to the Pay-Per-Use model is the issue of accountability, where the following questions are pertinent:

1. Who is responsible for gathering data about the consumed resources?
2. Who makes the decision about how much resource has been consumed?
3. Who calculates the charge?

Currently, *provider-side accounting* is the only common accounting approach that is used by cloud computing providers. In this mechanism, the consumption of the resource is unilaterally measured by the service provider, where the resource accounting service is deployed by the provider's infrastructure.

1.2 The problem statement and possible solution

In Pay-Per-Use cloud services, as mentioned above, provider-side accounting is the approach taken by all of the service providers. A serious limitation of this approach is that it does not offer the consumer a sufficient means of performing reasonable checks to verify that the provider is not accidentally or maliciously overcharging. We strongly believe that consumers should have a mechanism or a framework that helps them to build their own independent accounting service to be used to compute and verify their resource consumption, and to check whether or not they have been overcharged. In

addition, it may be used for other purposes such as IT project budget planning, and so forth.

Furthermore, none of the current and previous studies have paid attention to how a consumer might independently compute and verify the resource consumption. This is currently an open issue that is the focus of this research. In this study, we are planning to explore and discuss ideas, and address the issues related to developing *consumer-side accounting* of resource consumption in cloud computing. Clearly, our research is mainly focused on addressing the issues related to a consumer resource accounting service that can used to compute resource consumption. This service aims to allow consumers to verify bills from cloud providers who apply a Pay-Per-Use charging model. The selection of the Pay-Per-Use approach was based upon the fact that this model is widely used by many cloud computing providers, it is a more generic approach and, more importantly, it covers most of the issues related to resource consumption. To ground our approach in current practice, we will often use storage services as an example and, in particular, use Amazon S3 and Nirvanix Network Storage Delivery for our case studies for storage services and Amazon EC2 as an example of a computation service.

1.3 Motivation

Consumer-side accounting can be used by consumers in many purposes such as:

i. To compute and verify resource consumption and check whether or not the provider acts honestly and in good faith.
ii. Making their applications billing aware.
iii. To estimate the consumption and the cost of the resources used by an application.
iv. For IT budget planning of any project.
v. To create brokering services to automate the selection of services in line with user's needs and so forth.
vi. To implement a more sophisticated accounting mechanism such as bilateral accounting services, where both the consumer and the provider independently

measure resource consumption, verify the equity of the accounting process and try to resolve potential conflicts emerging from their independently produced results [2, 3].

1.4 Objectives

As we stated earlier, our main aim is to find a way that allows the consumers to verify their resource consumption. To achieve our target, we have determined the following objectives:

i. Understand the details of accounting models which are currently offered by different service providers from their documents and any other available documents and publication.
ii. Address how different cloud service providers compute the resource consumption for resources such as storage, bandwidth, CPU, etc.
iii. Study and understand the service providers' accounting models, in terms of what the main components of the accounting models are, how the resources are defined, how the provider computes resource consumption, what parameters are used to compute the consumption for each resource, and when the calculation is made for each resource etc.
iv. To study and understand issues related to collection of metering data at consumer side, such as which technique can be used to collect the data, which data needs to be collected and what are the challenges behind it.
v. To study whether all parameters required to computing resource consumption can be collected locally and independently at the consumer side.
vi. Describe or propose a system that allows a consumer to independently compute his resource consumption.
vii. Investigate the sources that may lead to possible discrepancy between resource consumption data collected by the consumer and the provider.

1.5 Thesis Contributions

This thesis makes the following contributions:

1. The thesis proposes the notion of a 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 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.
2. The thesis identifies the causes that might lead to discrepancy between the consumer side and the provider side measurements of resource consumption.
3. Using the concept developed in 1 and 2 above, the thesis evaluates the accounting models for storage and compute services from well-known providers and suggests how the models can be improved.

1.6 Thesis Outline

The thesis structure is as follows:

Chapter Two: identifies the wider body of literature to which this thesis contributes. It explores the related research made in the field of resource accounting in internet, network, grid and cloud computing.

Chapter Three presents all the experiments that have been conducted to understand the accounting models of several providers. A description of each experiment, together with the results and shortcomings of the model is included.

Chapter Four discusses the issues and challenges of consumer-side accounting in cloud computing and presents a detailed discussion and investigation about the sources that might cause a discrepancy between the consumer and provider measurements. The chapter also presents a systemic way of describing accounting models.

Chapter Five: concludes the thesis by summarizing its achievements. In addition, the chapter provides an overview of future work and possible extensions.

Chapter 2

Related Work

2.1 Introduction

In this chapter we discuss how the scientific community has tackled the problems of resource accounting and how the consumption of resources offered by cloud computing service providers is monitored. Furthermore, we present a discussion about the architecture of the resource accounting service and how it works, we address the technical aspects that are related to how and when the required data is collected, what data needs to be collected, how and when resource consumption is computed and how the consumer's bill is calculated for the computing resources sold by cloud service providers. Moreover, we will identify the research gap, propose a possible solution and address the challenge in order to make the proposed solution visible.

This chapter is divided into six sections. The first section presents an overview of cloud computing: cloud definition, cloud characteristics, cloud models and the architecture of cloud models. The second section presents an overview of charging models which are used in utility computing and includes examples of cloud computing and types of charging model that are used. The third section presents an overview of the accounting mechanism that is used by cloud providers to compute the consumer's charge (bill) and gives details of the architecture and standard of resource accounting service processes. It also presents some current and previous work related to Internet resource accounting and identifies the gap which needs to be covered. The fourth section discusses trustworthiness in resource accounting, the research gap and brief survey at metering and accounting level. The fifth section presents the issues and challenges of the proposed approach (consumer-side accounting) that needs to be investigated and addresses the parameters that are essential to exploring the visibility of our approach. Finally we present a summary of this chapter.

2.2 Background

The concept of cloud computing was proposed in 1960 by John McCarthy. He presumed that “computation may someday be organized as a public utility”, but it has taken more than four decades of computer and network technology development to make the concept a practical reality [7, 8, 9, 10, 11].

Cloud Computing is a new paradigm that came from the field of distributed computing and virtualization research groups as it is based on principles, techniques and technologies developed in these areas [12, 13]. Computer scientists still have different definitions of Cloud Computing [14, 15, 16]. For instance, it is defined in [17] as a style of computing in which dynamically scalable and often virtualized resources are provided as a service. Users need not have knowledge of, expertise in, or control over the technology infrastructure in the "cloud" that supports them. Also, Cloud Computing employs a model for enabling available, convenient and on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Furthermore, the authors in [16] define Cloud Computing as “A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet”. However, we believe the definition proposed in [22] to be most accurate, where cloud computing is defined as a *model* for enabling convenient, on-demand network access to a shared pool of *configurable computing resources* (e.g., networks, servers, storage, applications, and services) that can be *rapidly provisioned* and released with minimal management effort or service *provider interaction*. The cloud computing paradigm has a number of characteristics and three service models.

2.2.1 Characteristics of Cloud Computing

Some important characteristics of cloud computing are:

- On-demand self-service: the consumer ability to provision capabilities such as CPU time, network storage as required automatically online without any human interaction with the service provider. [22, 23, 24, 25].
- Global network access: computing abilities can be accessed through standard mechanisms in heterogeneous environments by different client's platforms (thin or thick) such as mobile, multi-device, etc [22, 23, 24, 25].
- Elasticity: Capabilities can be quickly and flexibly provisioned and in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning repeatedly appear to be unlimited and can be purchased in any quantity at any time [22, 23, 24, 25].
- Resource pooling: The provider's virtual and physical computing resources can be pooled and assigned dynamically to consumers according to their demand [22, 23].
- High scalability and availability: Cloud environments enable servicing of business requirements for larger audiences, through high scalability. Availability of services is high and more reliable as the chances of failure in cloud computing infrastructure are minimal [24, 25].
- Multi-sharing: Cloud working in a distributed and shared approach, multiple applications and users can work more efficiently and reduce the cost by sharing a common infrastructure [24, 25].
- Agility: The cloud is a distributed environment that shares resources among users and applications while improving efficiency and agility (responsiveness) [24, 25].
- Services in pay-per-use mode: cloud customers only pay for the IT resources they consume [24, 25], and the Service Level Agreements (SLAs) between the provider and the consumer must be defined when offering services in pay per use mode.
- Application Programming Interfaces (APIs): cloud computing may be offered (APIs) to their customers to allow them to use the services [24, 25].

2.2.2 Cloud Computing Service Models

The service models define the level of abstraction at which a cloud customer interfaces a Cloud Computing environment. Cloud has three service models; Software as a Service (SaaS) model, the Platform as a Service (PaaS) model, and the Infrastructure as a Service (IaaS) model [22, 23, 26].

Cloud Software as a Service (SaaS): customers rent software hosted by the provider. Cloud Platform as a Service (PaaS): customers rent infrastructure and programming tools hosted by the provider to create their own applications, and Cloud Infrastructure as a Service (IaaS): customers rent CPU, storage, networking and other fundamental computing resources for all purposes [22, 23]. These services are made available and sold on-demand basis "pay-as-you-go", for instance by minute, hour or month (e.g. GB/Month, instance/hour) [4, 27, 28, 29]. They are also flexible, where a user will be charged based on their consumption [20, 21]. The services provided by the cloud providers range from basic computational resources such as storage, bandwidth and computer power (IaaS), to sophisticated enterprise application services (SaaS). These services are agreed upon between consumer and provider and stipulated in the Service Level Agreement (SLA) contract which defines the details of the service availability and charging schema that is used [30]. Furthermore, the services offer easy and quick deployment and management of, and interactions with, service providers [31, 32, 33]. Where all the resources on the cloud are transplanted to the users, the users can dynamically rent virtual or physical resources without the need to know where those resources are located. In addition, the services are fully managed by the provider. These services are known as Utility Computing. Amazon Web service, Google AppEngine and Microsoft Azure are good examples of public Utility Computing [20, 21].

2.2.2.1 Software As A Service (SaaS)

Cloud Software as a Service (SaaS): is a service that is used to host and manage a particular customer's software in the provider's data centre. Where the customers are able to rent the infrastructure of could provider to run their services. In SaaS, the cloud provider manages and controls the underlying cloud infrastructure including the

network, servers, operating systems, storage, and even individual application capabilities, with the possible exception of limited user-specific application configuration settings [16, 7, 21, 22, 23, 34, 35, 36]. The cloud provider makes the application available to multiple users over the Internet through API. Usually, SaaS users just need a browser in order to access and use a SaaS Cloud. Some SaaS providers might use another PaaS or IaaS cloud provider. Oracle CRM On-Demand, Salesforce.com and Google Apps are some of the well-known SaaS examples [16, 7, 21, 23, 34, 35, 36].

2.2.2.2 Platform As A Service (PaaS)

Platform as a Service (PaaS) is a service for web application development and a deployment platform delivered to developers over the Internet in an easy, simple and quick manner [16, 7, 21, 22, 23, 34, 35, 36]. The PaaS models reduce the cost and complexity of buying and managing the underlying infrastructure, provide the facilities that are required to support the complete life cycle of building software and delivering web applications, and services are fully available from the Internet. It includes infrastructure software, and typically includes a database, middleware and development tools. It has a lustered grid computing architecture and is virtualized and is often the basis for this infrastructure software. AppEngine by Google, Force.com from SalesForce, Microsoft's Azure and Amazons Elastic Beanstalk are examples of PaaS [16, 7, 21, 23, 34, 35, 36]. The consumer neither manages nor controls the underlying cloud infrastructure of network, servers, operating systems, or storage; however, they do have control over the deployed applications, and possibly the application hosting environment configurations [22].

2.2.2.3 Infrastructure as a Service

Infrastructure as a Service (IaaS) is a service that provides the fundamental computing hardware such as server, storage and network, and associated software (operating systems virtualization technology, file system), where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications [7, 16, 21, 22, 23, 34, 35, 36]. It is an improvement on traditional hosting that does not require

any long term commitment and allows users to provision resources on demand. IaaS service provider requires very little management where users must deploy and manage the software services themselves, as they would in their own data centre. Amazon Web Services Elastic Compute Cloud (EC2) and Secure Storage Service (S3) are examples of IaaS offerings [7, 16, 21, 34, 35, 36]. As in SaaS and PaaS models, the IaaS's consumers are not able to manage or control the underlying cloud infrastructure, however, they do have control over operating systems, storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls) [22].

2.2.3 Architecture of Cloud Service Models

To capture and summarize the service models' architecture of the cloud computing paradigm, we can observe that the service models described above can be thought of as structured in four hierarchical levels of abstraction. Different academic researchers and groups present these hierarchical levels in different ways e.g., [22], however here we adhere to the hierarchical levels as depicted in Figure 2.1 presented by [23].

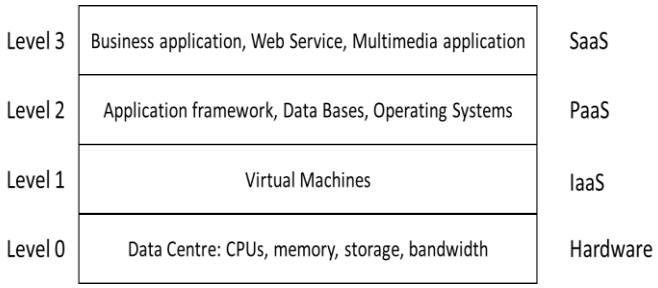


Figure 2. 1 Cloud Service Model Structure [23]

Hardware Level (Level 0) presents the fundamental foundation of the cloud computing paradigm and consists of the data centres containing the Cloud physical resources [23]. IaaS Level (Level 1) is responsible for instantiating and maintaining a pool of storage and computing resources using virtualization technologies such as VMware, Xen and KVM [23, 92, 93]. PaaS Level (level 2) consists of application platforms deployed within the resources available at Level 1 [23]. Finally, SaaS Level (Level 3) maintains actual Cloud applications.

2.3 Charging Models in Cloud Computing

There are increasingly numerous cloud service providers and types of services with different charging models for customers. Any cloud service provider can apply a flat rate, Pay-Per-Use or other charging models such as the auction charge model.

2.3.1 Flat rate charging model

The flat-rate charging mode is one type of charging mechanism, where the consumers pay a fixed amount of money for each well-defined period (e.g. month) to consume unlimited resources.

There is a monthly flat-rate standard plan as well as an annual flat-rate plan. In other words, in the flat-rate charging model the bill is fixed, irrespective of the amount of resources consumed [1, 2, 3, 91]. This type of service is offered by many Internet service providers such as Virgin Media, who charge their customers £20/month for unlimited downloads.

2.3.2 Pay-Per-Use charging model

In pay per use, the bill depends on the amount of resources consumed. The pay-per-use services can be further categorised into capacity-on-demand service and consume-on-demand basis [1, 2, 3]. Amazon Web Services [57] is the first provider that has made computational and storage resources commercially available on a pay per use basis on a production level. IBM has a cloud computing initiative underway called Blue Cloud [59]. There are other storage providers that offer their services on a pay per use basis such as Nirvanix [29] which optimizes storage for media files.

2.3.2.1 Capacity-on-demand

In capacity-on-demand, the consumer pays a fixed charge in advance for a fixed, maximum, non-exceedable capacity made available for their use, and the service is terminated when the consumer exhausts the resources. Such systems include Google E-mail systems and pay-as-you-go mobile phones.

2.3.2.2 Consume-on-demand

In the consume-on-demand service, the consumer consumes as much as he needs, when he needs it; the charge (or the bill) is calculated according to actual resource consumption and later presented to the consumer at the end of an agreed-upon accounting period. Amazon Simple Storage Service (Amazon's S3) [4] is a well-known example of a service provider that sells storage space to remote users and uses the consume-on-demand charging model to charge their customers.

2.3.3 Auction charging model

In the auction charging model the service provider offers the service to customers using a bidding system. For instance, EC2 offers Spot Instances that enable the customers to bid for unused Amazon EC2 capacity. Instances are charged the Spot price, which is set by Amazon EC2 and fluctuates periodically depending on the supply of and demand for Spot Instance capacity. To use Spot Instances, the customer places a Spot Instance request, specifying the instance type, the availability zone desired, the number of Spot Instances they want to run, and the maximum price the customer is willing to pay per instance/ hour. To determine how that maximum price compares to past Spot prices, the Spot price history is available via the Amazon EC2 API and the AWS Management Console. If the customer's maximum price bid exceeds the current Spot price, the customer request is fulfilled and the instances will run until either the customer himself chooses to terminate them or the Spot price increases above the consumer maximum price [58].

2.4 Resource Accounting in cloud computing

Accounting of computing resources is defined as the whole process that is required for calculating the charges of each consumed resource to produce a customer's bill for a well-defined period of time. This process includes collecting metering data, computing the resource consumption and producing the final customer bill. It also can be used for other purposes such as auditing, monitoring and so forth [5]. The service which is responsible for the accounting is called resource accounting service (RAS).

Substantial work has been done in internet resource accounting which we believe can be applied to cloud computing. Consequently, it is necessary to understand the main components of the RAS, how each component works and what the relationships between these components are. We have included a survey in this thesis which is part of the current and previous work in the field of network and internet resource accounting. The survey describes the architecture of the introduction and accounting terminology, Resource Accounting System, Internet resource accounting and a summary of current and previous related work which has been developed in the area of network and internet resource accounting.

2.4.1 Resource Accounting System background

In the last 20 years several standards have been suggested by Internet Engineering Task Force (IETF) concerning Internet Accounting. In 1991 the Network Working Group (NWG) released the first RFC report on Internet Accounting [40] which introduced basic information about Internet accounting architecture and defined a simple Internet accounting model which consists of three basic components: meter, collector and application.

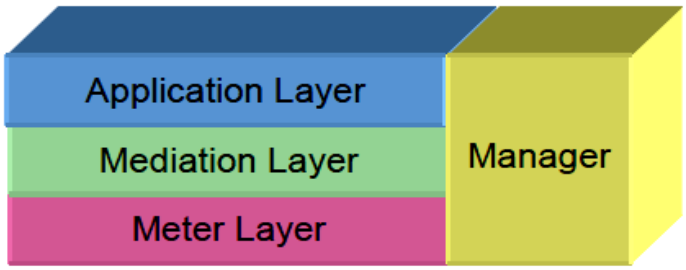


Figure 2. 2 The accounting system infrastructure [41]

[41] added a new component called the Manager on the basis of the accounting model proposed in [40] and this further illustrates the relationship between these components. The description of the accounting system infrastructure is shown in Figure 2.2.The Meter Layer measures the network traffic and aggregates measurement results. The Mediation Layer collects measurement data from the Meter Layer, and processes (aggregate, de-duplicate, validate, correlate etc.) collected data and stores them. The Application Layer consists of applications for different purposes such as billing, audit,

and trend analysis etc., using data from the mediation layer. The Manager configures and applies rules to control the activities of the three components and the whole system.

In [42] the measurement of traffic flow architecture was suggested; this document was defined using the so-called Meter Management Information base (MIB). The MIB allows the gathering of information about data usage from the network which is important for accounting, performance, configuration, as well as security purposes. In June 2000 the Network Working Group issued another RFC that focused on the development of Remote Authentication Dial In User Service (RADIUS) which brought accounting back on the agenda of the IETF [43]. The RADIUS protocol defined how authentication, authorization and configuration information should be exchanged between Network Access Servers (NASs) and authentication servers. The RADIUS protocol was widely used with the interest of improving it by adding more features from different research groups.

A new Working Group (WG) within the Operations and Management Area of the Internet Engineering Task Force (IETF) was formed. The name of this new WG is Authentication, Authorization, and Accounting (AAA) [46]. This work group [47] has achieved the simple accounting architecture shown in Figure 2.3.

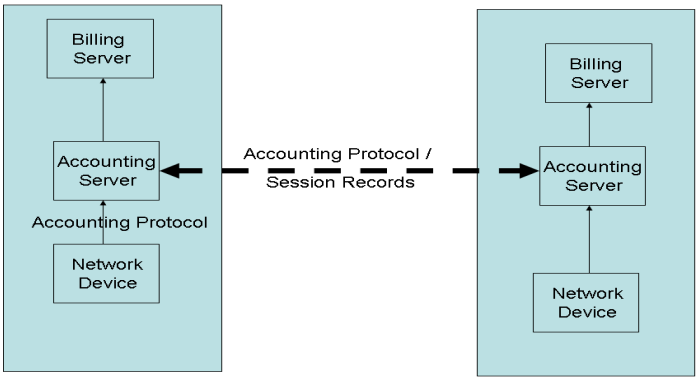


Figure 2. 3 Simple Accounting System Architecture [47]

In [44, 45] a new WG document was issued that proposes a reference model describing the interactions between the metering, accounting and charging processes, "the main components of the resource accounting system", and their configuration via polices.

Figure 2.4 shows the proposed reference model. On the right side of the figure below there are five layers showing the different building blocks.

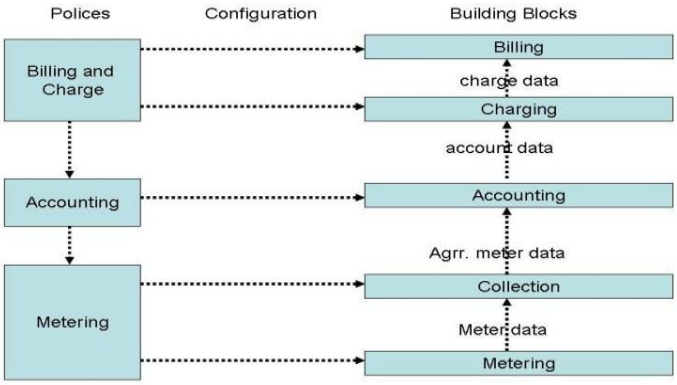


Figure 2. 4 Reference model of resource accounting system [44]

The blocks are layered according to the processing of the data from the bottom level metering via accounting, up to the final billing process. Data aggregation can be done at any layer not only at the collection layer. The building blocks on the different layers are configured through the policies shown on the left side. Higher layer policies can be translated into lower layer policies. The configuration parameters are extracted from the policy and passed to the corresponding building block.

Here is a brief description of each layer of the building blocks:

- **Metering:** Meters are required for gathering data about resource consumption in the network (e.g. bytes transferred).
- **Collection:** The data gathered by the meter(s) has to be collected for further processing. Collection of meter data can be initiated by the meter itself and collected data can be aggregated before being passed to the accounting layer. Metering policies define how collection and aggregation is done.
- **Accounting:** Accounting describes the collection of data about resource consumption. This includes the control of data gathering (via metering), transport and storage of accounting data. For subsequent charging, the metered data must be associated with a user that is the initiator of a flow and a customer (service subscriber) that is responsible for payment. For initiation of an accounting process, a user or foreign provider must be authenticated and authorized. These three functions can be performed by the AAA server. The accounting process is configured through accounting policies.

- **Charging:** Charging derives non-money costs for accounting data sets based on service and customer specific tariff parameters. Different cost metrics may be applied to the same accounting records even in parallel. Charging policies (models) define the tariffs and parameters which are applied.
- **Billing:** Billing translates costs calculated by the charging model into money and generates a final customer's bill. Billing policies define the type and how the customer will be charged (e.g. invoice, credit card), and the time for billing (e.g. weekly, monthly, etc.).

Related to standard terminology and definition used in the area of resource accounting and as pointed out in [47], different network and Internet communities use term accounting to refer to different aspects of the accounting process. For instance, some authors use the term to refer to the process of metering, collecting, interpreting and reporting, costing and charging-related information of the usage of a service or resource, while others use it to refer to only one of the sub-processes. Also, in [56] the authors present taxonomy of billing models and a discussion about the metering parameters (e.g. volume consumed, star and end time of a session) that each model requires. Therefore, in this thesis we use the term metering service to refer exclusively to the process that collects row metering data, the accounting service to refer exclusively to the process that applies the accounting model on the metering data to compute the resource consumption (accounting data) and the billing service to refer to the process that applies the pricing model on the accounting data to provide the customer's bill.

2.4.2 Examples of IT resource accounting

Architecture similar to what has been proposed in [41] was used in [48, 54]. The paper [54] presents an account of the basis of pay per use, it identifies users though a unique ID for each user rather than the IP address. The implementation of a user based traffic accounting prototype system with Agent mechanism is introduced. The implementation is based on the IP accounting infrastructure which consists of 3 layers; meter layer, mediation layer and application layer. When a Meter measures the network traffic, it

generates the accounting records which consist of several accounting attributes. Usually the accounting attributes can be divided into two categories: identification attribute and usage attribute.

The RAS becomes an essential component of the service infrastructure in a distributed system such as grid and cloud computing to compute the variable cost services. Even in non-commercial settings or for flat-rate services, metering and accounting are needed for enforcing policies such as usage quotas, or to analyse usage patterns, for example. The authors in [48] have paid attention to metering and accounting services for composite e-Service. e-Service may be seen as a component technology for building distributed applications, or as a mechanism for distributed systems integration. Web services [49] are the most common example of e-Services, but other kinds of e-Services are also gaining importance. For example, grid services [49, 50] and cloud services [4, 51] are an emerging mechanism for sharing distributed, heterogeneous resources across organizations.

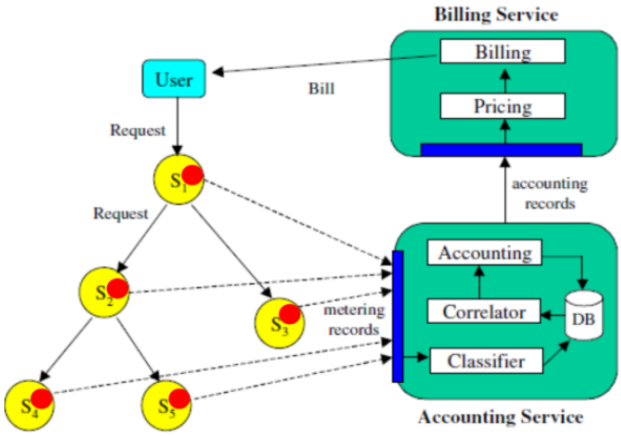


Figure 2. 5 Metering and accounting for composition e-Service (MACS)

Figure 2.5 shows the framework of Metering and Accounting for Composition e-Service (MACS) architecture proposed in [48] for a composite e-Service. The composition e-Service consists of 5 e-Services (S_1– S_5).

The architecture of MACS consists of metering, accounting and billing services. In this approach, the authors used several metering services and a single accounting and billing service. A metering service was deployed in each service which is represented by the

red circle in the figure above. The metering service is responsible for collecting metering data using an instrument called a meter and producing metering records per-partial request usage, as it reports usage relating to that service alone. Each metering service sends the metering record into the Accounting Service. The meter can be a local monitor data and/or application-level metrics used in building this metering record usage.

The accounting Service consists of Classifier, Correlator and Accounting. Also it has an interface which allows the metering services to send the metering records through it and it has a database which is used to store the metering records. The Classifier receives the incoming metering records and classifies them based on service and user basis, and stores them in the database. The Correlator retrieves the related partial metering records for each user from the database and associates them together to create complete metering records. The Correlator passes these complete metering records to the accounting unit which aggregates them into the account of the appropriate <customer provider>. This result is defined as the accounting records. At each billing cycle the accounting service supplies these accounting records to the Billing Service. The Billing Service applies the pricing model of the service provider on the accounting records to produce the customer's bill. Moreover, other architectures have been proposed for accounting and billing such as [53]. In [53] the authors have proposed architecture for accounting and billing in cloud infrastructure (RESERVOIR).

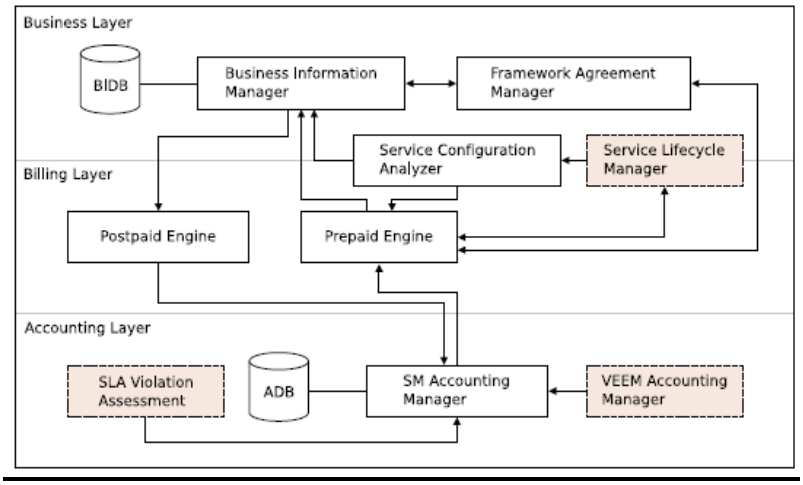


Figure 2. 6 Accounting and billing architecture in RESERVOIR.

An overview of the system architecture is shown in Figure 2.6. The proposed architecture composes of three main layers; Accounting, Billing and Business Layers. The Accounting layer is responsible for the procedure of collecting and managing the row metering data which will be used by the Billing layer. The Billing layer is responsible for evaluating the Deployment Description (DD) by analysing the DD from a business perspective to apply business oriented deployment restrictions, verify the amount of available credits, generate a unique identifier for this particular service which will be used in the Accounting, Billing, and Compensation (ABC) identifier and complete the payment procedure. The Business layer arranges the relationship between the technical issues of the system (RESERVOIR) and the consumers in terms of pricing, invoicing, service management and so forth.

Figure 2.6 shows the main components of the proposed architecture, with the arrows that represent the relationship and interactions between components of the system. The components with a dash border are gateway components between the accounting system and other parts of the RESERVOIR architecture. The Accounting Database (ADB) and the Business Information Database (BIDB) represent any database technology which is not a specific component of the architecture.

2.5. Trust in Resource Accounting Service

Regardless to the charging model used by the service provider, there are several trust related issues of accounting that need attention:

- Who is responsible for gathering data about the resources consumed?
- Who makes the decision about how much resource has been consumed?
- Who calculates the charge?
- How is the charge calculated?
- Can the accounting result be verified and trusted by both parties?

The service provider is responsible for doing the accounting processes as in [48, 54], this is called provider-side accounting. Currently, the *provider-side accounting* (PSA) mechanism is the only common accounting approach that is widely used by cloud computing providers such as Amazon S3, Nirvanix NSD and gooleApp. In the PSA

mechanism the resource consumption is unilaterally measured by the service provider where the resource accounting service is deployed on the provider infrastructures. A serious limitation of PSA is that it does not offer the consumer sufficient means of performing reasonableness checks to verify that the provider is not accidentally or maliciously overcharging. This mechanism is acceptable when the consumer has good reason to trust the provider and the consumer believes that the provider will not accidentally or maliciously overcharge him. For instance, with capacity-on-demand it is important to check whether the consumer reaches the maximum non-exceedable capacity of resource consumption or not when the service terminates. Also, in on-demand basis charging model, the consumer wants to check whether he paid for what has consumed or has been overcharged.

To conclude the above discussion, most cloud providers currently use provider-side accounting where the provider unilaterally measures the consumer's resource consumption and presents the latter with a bill. This accounting mechanism does not offer the consumer sufficient means of performing reasonableness checks to verify that the provider is not accidentally or maliciously overcharging. Therefore, consumers require an accounting mechanism that can produce trusted accounting results.

Trusted resource accounting result can be produced by one of the three approaches: 1) Trusted Metering Services 2) A Trusted Third Party (TTP) produces the records of resource consumption using its own certified infrastructure, or 3) Bilateral Resource Accounting Services where the interested parties use their individual unilaterally trusted resource consumption as the basis for agreement on valid, mutually trusted resource consumption [2].

2.5.1 Trusted Metering Service

The Metering Service MS can be regarded as the backbone of the resource accounting service, because accounting service and billing services duties are based on the metering data provided by the MS. Therefore; if we can build a trusted metering service then it becomes possible for the RAS to produce a trusted result. Both a certification authority that certifies the correctness of its functionality, as well as tamper-resistant

protection mechanisms that prevent its undetectable modification, can help to build a trusted service.

For instance, the authors of [90] have developed a Meter Inspection Authority (MIA) which is used to cover the security requirements to help the providers and the customers to trust each other. The MIA is a third party used to establish trust between two entities. The main duty of the MIA is to provide undeniable metering data by any of the involved parties (the consumer and the provider), by installing a piece of trusted code (the Client, Third Party provider or Provider) on devices that have been certified by a MIA. All involved parties in the system ensure that the code provided by the MIA is operated fairly and all trust the code and its output. The scenario is shown in the Figure below.

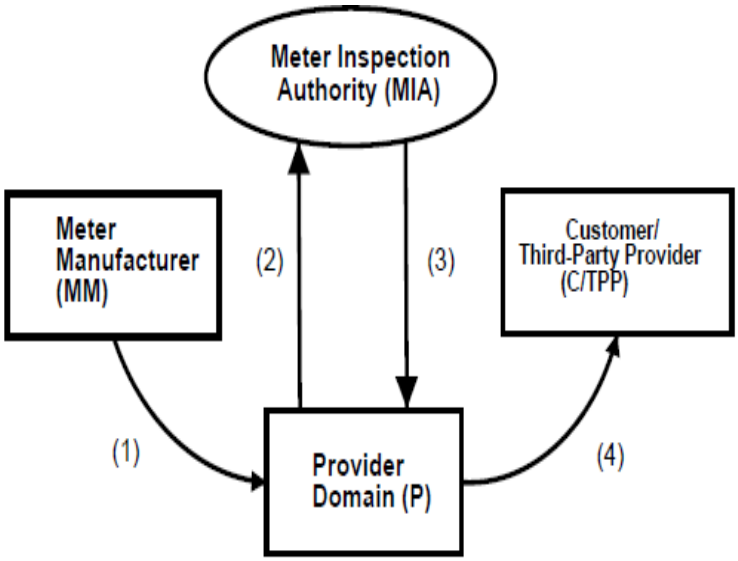


Figure 2. 7 Secure Metering phases.

Firstly, the Providers buy the meter system from the Meter Manufacturer (MM). Then, the meter is installed at the customer/third-party provider's domain (Figure 2.7). The Meter Manufacturer (MM) dispatches message 1 containing the meter system to the Provider (P), the message is signed by MM (SIG_{MM}) using a one-way hash function. Then, the provider P encrypts the message with P's public key (KU_P) to match the requirement of privacy. The timestamp *t1* informs the provider of the time when the message was created.

Secondly, the P sends a message containing the (*meter system and SLA/Tariff translator)* to MIA. The message is signed by the P signature (SIG_P) to guarantee the message authenticity. The message is encrypted using MIA's public key (KU_{MIA}). Then, it generates a certificate called MIC (Meter Inspection Certificate) with assurances. The certificate MIC_1 contains calibration and safety certification for the *meter system*. MIC_2 does the same for the *SLA/Tariff translator* but adds extra assurance. MIA also creates a type of *meter seal* when signing the code (*meter system, SLA/Tariff translator*) with its signature (SIG_{MIA}). The seal guarantees that the code (*sealed code*) will not be modified. By analogy, the electricity meters are also sealed to prevent anyone tampering with them.

Finally, P sends the sealed code to customers/third-party providers (C/TPP). They can check the certificates (*MIC1,MIC2*) and trust the code as conforming to a meter and SLA/Tariff translator specifications. This message sent from P is signed with P's signature for authentication and encrypted with C/TPP public key ($KU_{C/TTP}$). The authors have applied type-safe language to ensure safe execution and secure the code from attack at run time. Furthermore, for distributed metering measurement, they suggested that authorization schemes such as using asymmetric cryptography are necessary to sign the code in order to maintain the integrity (to ensure that the code is not modified or read by the any other party). Furthermore, as previous and current research has shown, several techniques such as hardware (Trusted Computing model) [82, 83], and software [84, 85] can be used to build tamper-resistant systems which can be run in un-trusted platforms. Developing trusted metering services is beyond the remit of the current research.

2.5.2 Trusted Third Party (TTP)

A trusted accounting service can be owned and run by a TTP, whose results are trusted by both the consumer and the provider.

The development of a TTP accounting service is required in order to understand the essential requirement of building the resource accounting service in aspects such as the development of metering and accounting services. Firstly, at metering service level we

need to understand how data will be collected, when data is to be collected, what data should be collected, how the metering data is aggregated and refined, and where and how it should be collected. More importantly, we need to know if the TTP metering service is able to collect all the required data to compute resource consumption and where the metering service of TTP accounting service is located. Secondly, at the accounting level, there are other concerns such as how and when resource consumption is computed, what parameters are required to compute the consumption of each resource, how the accounting is formatted and what and when accounting data is required by the billing service and so forth. Unfortunately, none of the previous and current studies have addressed this topic.

To the best of our knowledge, to date, TTP for resource accounting has not been developed. However different approaches have been developed in TTP to monitor different parameters (e.g. response time, throughput) by authors such as [61, 86]. Also in [70] the authors develop the notion of a third party service management authority that can monitor interactions between customers and cloud provider in term of monitoring the quality of service without paying attention to monitoring the resource consumption. All the above papers overlook the need to provide consumers with a means of performing consumer-side accounting. Therefore, we believe that involving TTP in accounting is essential to ensure the trust between both parties.

2.5.3 Bilateral Resource Accounting Service

Another approach to building a mutually-trusted resource consumption service is to use bilateral accounting, where the resource consumption is computed and decided by the consumer and provider with the help of a pair of independent components, both of which have the same functionality [2, 3]. The first component is deployed within the consumer infrastructure and the second is hosted on the provider's premises. The job of the pairs is to produce together a trusted output. This is a fair, realistic and trustworthy approach. However, in a general scenario for real service providers, the implementation of the accounting component at the consumer side accounting will meet the same challenges that are discussed in developing TTP accounting mechanism.

2.5.4 The research gap

From all the above discussion we find that provider-side accounting is the only approach taken by all of the service providers to compute resource consumption. A serious limitation of this approach is that it does not offer the consumer a sufficient means of performing reasonableness checks to verify that the provider is not accidentally or maliciously overcharging. Providing an accounting mechanism that can produce trusted accounting results or at least can be used to compute and verify the consumer's resource consumption is essential. However, to the best to our knowledge such as system has not been developed our studied yet and is currently an open issue.

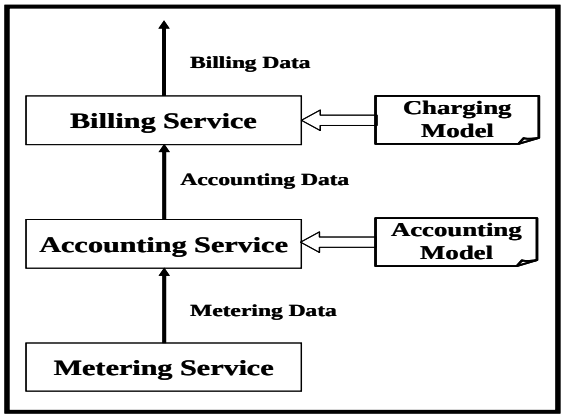


Figure 2. 8 Resource Accounting Service

As shown in figure 2.8, a resource accounting service is composed of three components: a Metering Service (MS) responsible for collecting raw metering data about resource consumption; an Accounting Service (AS) that retrieves the metering data and applies an accounting model to produce accounting data, and a Billing Service (BS) that, on the basis of the accounting data provided by the AS and charging model (e.g., prices, discounts to golden customers, fines for late payments, etc.), produces the actual bill, say monthly, for the consumer [2, 3, 7]. In the next section, we will present a brief survey about how scientific community has tackled the technical issues and the techniques used at metering and accounting levels.

2.6 Resource accounting service – issues and challenges

Developing a resource accounting service requires to understanding and investigating the issues at metering and accounting levels. For instance, at the metering level, we need to find the answer to the following questions: Which technique used to collect the metering data? What metering date should be collected and when? In what format should the metering data be stored?, and finally we need to check whether all the required data to compute resource consumption can be collected by the MS or not. Furthermore, at the Accounting level, in order to develop the accounting service we need to understand the provider's accounting model in terms of how resource consumption is computed, if there is any relationship between the details of the requests/responses and the resource consumption, when the resource consumption is computed, what the accounting data looks like for each resource, and so on.

2.6.1 Metering Service – background, issues and challenges

The MS represents the local instrumentation that performs to collection of metering data about resource consumption. MS produces metering data collected at specific time intervals or upon the occurrence of specific events. For example, related to storage service, a request to store 600MB of data has been made or a 2MB directory has been deleted. Let's assume that the Metering collector (MeCo) is the component of the MS that is responsible for doing this job. The MeCo is to be understood as the machinery (pieces of software possibly in combination with some hardware components) [61] used to collect and store the metering data that result from the consumer's activities. In resource accounting, the gathering and collecting of metering data raises several issues:

(i) Which technique should be used by the metering collector?
(i) What type of metering collector is used?
(ii) What information can be deduced from the collected metrics?
(iii) Can all the data require for accounting purposes are collected by the any other parties such as service consumer or other TTP?
(iv) Where is the metering collector deployed? (At the service consumer, service provider, or a network in between the two?)

For the provider (i.e., storage provider), the problem of building a MS is relatively straightforward. The provider has control over the physical storage used to satisfy client requests and can directly measure the impact on backend storage requests of creating, deleting, appending or truncating data and so forth. For example, if a Unix-like file system is used to store data, system commands such as *du* can be used to measure storage consumption. On the other hand, building the TTP or consumer's metering service is more difficult because the consumers do not have direct access to the provider's infrastructure.

Unfortunately, and to the best of our knowledge, currently there is no existing MS approach which describes the technique(s) used to collect the metering data for accounting purposes. On the other hand, different approaches have used different techniques to collect metering data to monitor different parameters of the Service Level Agreements (SLAs) such as response time and throughput. These techniques were mainly used for monitoring the Quality of Service (QoS) offered by the SLA of the service providers. Below we provide an overview of some of the existing metering approaches and the techniques which have been used to monitor the QoS of SLAs. Also, we will include a brief discussion about which approach and technique is suitable for building the consumer-side metering service.

2.6.1.2 Metering service approaches of monitoring SLAs

Without paying attention to implementation details, we can divide the existing approaches and techniques for collecting metering data into four general categories.

1. **Provider-side instrument**: where the MeCo is deployed within the provider infrastructure. In this approach the measurements about the provider performance are taken directly from the provider's resources [61,62, 63, 64, 65, 66].

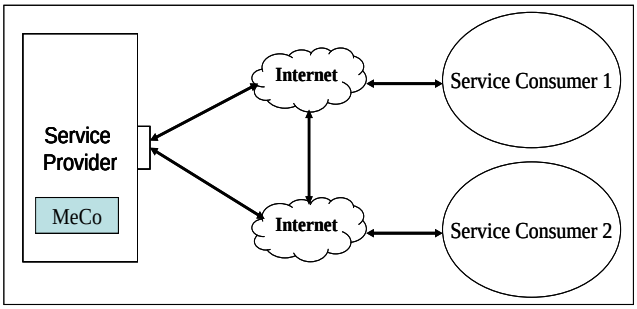


Figure 2. 9 Service Provider Instrumentation approach

2. **Consumer-side instrument:** The metering data is collected by the MeCo which is deployed at the consumer-side [61,62,63,64,65,66]. In this scenario, MeCo can be realised as a piece of software installed in the service consumer's browser.

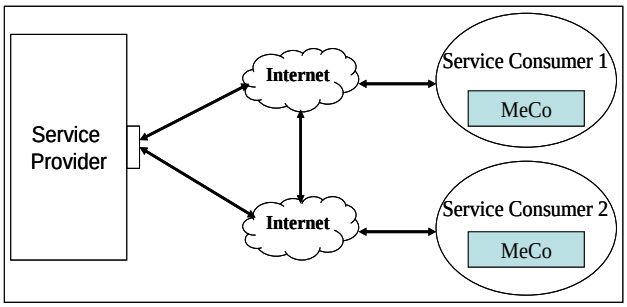


Figure 2. 10 Service Consumer Instrumentation approach

3. **Periodic polling with probe clients:** In this case a Trusted Third Party (TTP) is involved in collecting the metrics. Figure 2.11 shows Probe1 and Probe2, two TTPs working as synthetic clients strategically located and equipped with a MeCo; from the point of view of their functionality they are two synthetic clients strategically located and equipped with a MeCo. They are there to periodically probe the provider to measure its response [61,63,66, 67]. Keynote [67] is a good example of this approach.

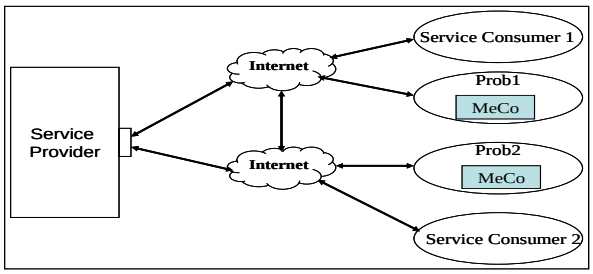


Figure 2. 11 Periodic polling with probe consumer approach

4. **Network packet collection with request-response reconstruction**: In this approach the MeCos are installed between the provider and the consumer to collect data about all traffic between them to collect all the packets (either by interception or by sniffing) coming into and out of the provider. Later the collected data is reconstructed and analysed upon request–response upon particular data (i.e. TCP header) [61, 66].

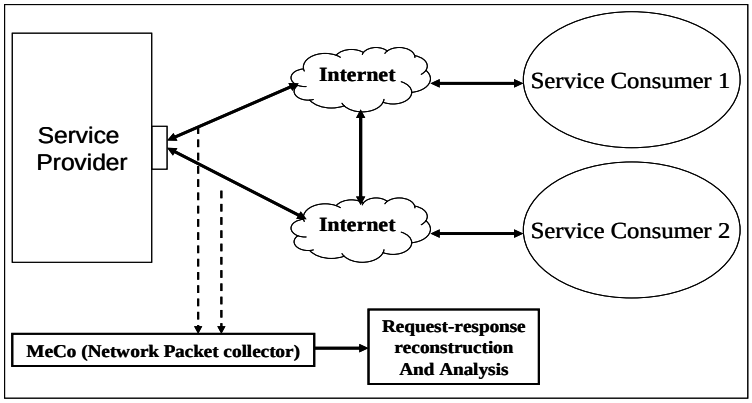


Figure 2. 12 Network packet collections with request-response reconstruction approach

2.6.2 Accounting Service - background, issues and challenges

The AS uses the raw data collected by the MS to produce accounting data (resource usage records) and stores it in a meaningful manner. This involves computations that are specific to the service being provided by the specific model known as the accounting model. For computational resources, usage records may detail accumulated processing time over some period such as 5 minutes CPU. For instance, in storage provision, usage records may detail the amount of data uploaded, downloaded and deleted over a period (for example, 5GB uploaded, 10GB downloaded and 1GB deleted in a given period). The exact form of a usage record will depend on the model for service provision. The role of the AS is to perform computations according to this model (accounting model) that use raw data taken from the logs of consumer activity to generate usage records that are a suitable basis for the billing service to calculate charges.

In order for the any party and service provider to produce similar accounts results, it is necessary that both of them use the same accounting model. The service provider offers

their services together with a well-defined, abstract model of their service system. On the other hand, developing AS by other parties, e.g. TTP should produce accounting data from the metering data collected by the MS (middleware interceptor). In general it is not possible to translate such logging into an accurate record of actual storage consumption at the provider side. For example, writing a request of 9KB may result in the consumption of more than 9KB of storage space, depending on parameters such as file system block size, the file system's metadata and perhaps the user's metadata. The AS computes accounts data (resource usage records) for each resource. The accounting process can be arbitrarily complex as it can take into consideration advance payments, delayed payments, discounts and so on. However, it might not be possible to translate such logging into an accurate record of actual resource consumption on the provider side. For example, writing a request of 9KB may result in the consumption of more than 9KB of storage space, depending on parameters such as file system block size.

To the best of our knowledge there is no existing approach which takes into consideration the building of accounting services, and addresses the issues and challenges related to such a service. However, there are many studies that support our approach to understanding the accounting model of the service provider. For instance, in [71] the author argues that cloud providers should make their services accountable for both the provider and the customer. Also, the authors of [60] suggest 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. More importantly, and according to observers in [73, 74] there are hidden costs that the user might incur while consuming a particular resource, so we need to know the provider's accounting model that is applied for each resource. It is also important to mention that some computing charges are not based on usage of only [73] one resource. For example, Amazon EC2charges for instance-hours, as do other providers, which represent anything between one second and 60 minutes of instance running time; arguably, this granularity might be too high and inconvenient when one tries to make users (say employees within a company) accountable for the actual hours of VM time (as opposed to instance hours) that they consume from a public cloud [75]. Also it is important to stress that a 60 min run does not necessarily mean 60 min of actual

processor time; the authors of [76] have observed that Amazon EC2 small instances typically receive only a 40% to 50% share of the processor. Another example is Amazon S3 charges for the number of operations executed against its S3 interface and independently of the internal computation cost that the request generates [77].

Architecture for accounting and billing for resources consumed in a federated Grid infrastructure is suggested in [53]. The paper provides a valuable insight into the requirements (resource re-deployment, agreement awareness, payment procedure, standardised records and others) that accounting and billing services should meet. The general principles of an architecture for accounting and billing in cloud services that are composed out of two or more federated infrastructures (for example, a storage and computation provider) are discussed in [53]. 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. However, we need to understand the accounting model of a real cloud provider and investigate the related issues to build consumer-side accounting.

All the above arguments support the practical and commercial relevance of our study and how important the accounting model is in resource accounting. Furthermore, given an abstract accounting model, the any of the involved parties in the service can decide independently how to implement their AS. Understanding the accounting model, how and when accounting data needs to be calculated, and several other issues, needs to be worked out before thinking of implementing accounting service. Unfortunately, none of the previous studies examine how the service provider computes the consumption of each resource, what parameters are required to compute the resource consumption of each resource, when the resource consumption is computed, whether all the required parameters to compute resources consumption can be collected from consumer's requests/responses, when the collected parameters are required by the accounting model and so forth. Therefore, to fill all the aforementioned gaps, part of our research will focus on understanding the accounting of different service providers to help us build the consumer side accounting service.

2.7 Summary

In this chapter we present a number of academic and industrial studies that have discussed issues related to IT resources in terms of accountability. A conclusion from this is providing an accounting mechanism that can produce trusted accounting results or at least can be used to compute and verify resource consumption is essential. Where, issues related to metering, accounting and billing in resource consumption have not been covered by research. Also, concerns to how a consumers or other in behave of them can independently compute their resource consumption and consequently can verify their charge. Many aspects of accounting service have not been explored such as, what is the relationship between each request/response and the resource consumed, how can the resource consumption be computed by the consumer based on request/response details, what data are required to compute the resource usages, how can the data be collected and what data should be collected, does the cloud provider offer well documented, complete and clear information that helps the consumer to implement their own resource accounting service, what source of parameters might cause a discrepancy between the consumer and the provider measurements. These and many other issues related to consumer-side-accounting need to be covered. Therefore, this thesis is about to cover the issues mentioned above.

Chapter **3**

Calculating Resource Consumption

3.1 Introduction

A pay-per-use Cloud service should be made available to consumers with an unambiguous resource accounting model, by providing a precise description of all factors taken into account in calculating resource consumption charges. In this chapter we aim to investigate and explore the feasibility of implementing a consumer-side resource accounting service. In order to develop this service, we need to investigate and understand the answer to the following questions:

1. Is it possible for all the data required for calculating the resource consumption for each resource to be collected independently by the consumer-side metering service?
2. Is the description of the provider's accounting model unambiguous, and can it be used by the consumer-side accounting service to compute resource consumption and produce a similar result to the provider's? This requires the acquisition of the following information:
    (i) Understanding how resource consumption is computed for each resource;
    (ii) Knowing when resource consumption is computed;
    (iii) Knowing if the Cloud provider uses the same accounting model for different APIs (e.g. REST, SOAP).
3. What potential sources of conflict can arise between consumer and provider measurements and why?

In order to investigate and understand how we can develop a consumer-side resource accounting service, we have selected two key components of IaaS, namely storage and computing power, as our target resources and we have run several experiments, as described in the first section. The second section describes the experiments and results of the first case study on Amazon storage service (S3). It also includes a description of

the Amazon accounting model, the shortcomings of Amazon S3's accounting model and a summary of the results. The third section describes the experiments and their results of the second case study on Nirvanix SDN storage service. It also includes a description of the Nirvanix SDN accounting model, its shortcomings and a summary of the results. The fourth section describes the experiments and results of the third case study on Amazon Elastic Cloud Computing (EC2), the shortcomings of the Amazon EC2 accounting model and a summary of results. The fifth section describes an accounting model used to estimate resource consumption and the last section provides a summary of this chapter.

3.2 Experiments

Our initial investigation revealed that the most providers calculate storage consumption charges by collecting data concerning:

- Storage: the space consumed in the bytes at the service provider.
- Bandwidth: the network traffic that is generated by operations that the customer executes against the service interface. The bandwidth is classified into upload and download bandwidth.
    - Upload bandwidth represents the total number of bytes transferred per request (Data-Transferred-In).
    - Download bandwidth represents the total number of bytes transferred per response (Data-Transferred-Out).
- Operation: the number of operations that the customer executes against the service interface during a well-defined period of time.

Amazon S3 was selected based on the fact that Amazon Web Service is one of the leading cloud computing providers who offer storage service on pay per use basis. Furthermore, Nirvanix SDN (another leading provider) was selected as the second case for the purpose of comparative evaluation.

In addition to storage service, we have selected Amazon Elastic Cloud Computing (EC2) from Amazon Web Service to investigate and understand how the charges for Virtual Machines are computed.

3.2.1 The Scenario

An abstract view of the scenario in our study is shown in Figure 3.1, where the Service Provider is represented by a cloud service provider, for example the Amazon Simple Storage Service S3 or the Nirvanix Delivery Network. In addition, the consumer is represented by a single individual consumer of the service. As shown in Figure 3.1, the consumer can access the remote service only through a service interface. The service provider may offer one or more interfaces, for instance, Amazon S3 has SOAP and RESTful interfaces.

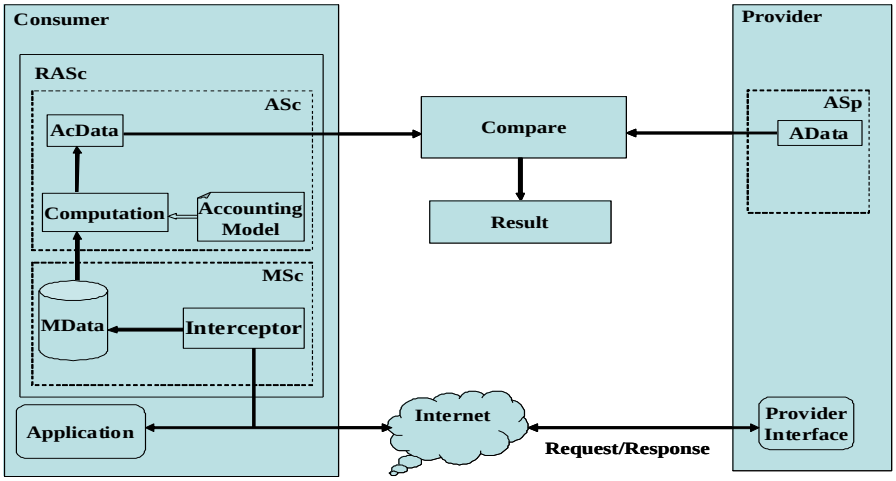


Figure 3. 1 Experiment's architecture – consumer/provider

The consumer deploys their own resource accounting service (RAS_C) within the consumer's infrastructure. The RAS_C consists of two components: a metering service (MS_C) and an accounting service (AS_C). The MS_C is responsible for collecting raw metering data about resource consumption via the interceptor. The interceptor intercepts all requests/responses and collects the metering data. Different metering data can be collected for different resources for request/response details. Consumer-side metering service stores these details in Metering Data Storage (MDS).

3.2.2 Assumptions

In the experiment we work under the following assumptions:

1. The consumer is represented by a single application, which is a simple SOAP or RESTful client who uploads, downloads and deletes data from a service provider.

2. The client's application allows the execution of all operations that are offered by the provider's interface: for instance, CreateBucket to create a new bucket (or folder) in Amazon S3.
3. The client can download the provider's Usage Report "accounting data" at any time.

3.2.3 Experimental Setup

We have developed two Amazon S3 client applications using, respectively, a Plain Old Java Object (POJO) and an Eclipse environment. The first application was developed to interact with the SOAP interface, while the second was developed to interact with the RESTful interface. We have used a single PC (HP with AMD Athlon [tm] 64 processor 3500, 2.19 GHz and 2GB RAM) connected to the internet through the University of Newcastle server. The data collected by the metering service is stored in an EXCEL file. The consumer's metering service was represented by two components: a metering information collector and metering data storage. The collector is a middleware SOAP or RESTful handler which intercepts all requests/responses and sends them to metering data storage to be stored in a meaningful way.

3.2.4 Methodology

1. The customer runs their application to upload, delete and download objects from their Cloud provider account.
2. MS_C collects metering data about each request/response.
3. The provider's Usage Report 'accounting data' is downloaded.
4. From the provider's Usage Report 'accounting data' and the consumer's metering data, we will try to extract the provider's accounting model for each resource, by trying to derive the relation between the metering data collected by the MS_C and the accounting data produced by the provider's Usage Report.
5. We will try to describe the extraction relation in a general formula if possible.
6. The extracted formula will be applied to different data collected by the MS_C, to check whether or not it produces the same accounting data as was produced by the provider's Usage Report.

3.3 First case study: Amazon S3

Amazon advertises its S3 service as a storage service available to internet users on a pay-per-use basis [4]. Informally, this 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 [37]. An S3 space is organised as a collection of 'buckets', entities 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 a 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.

Information about pricing and the charging schema used to calculate the customers' bill is spread across three documents available from the 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 when days are considered to start and end and billing cycles. However, from the Authenticating SOAP Requests Section of the

Amazon Developer Guide [37], it is clear that S3 servers are synchronised to Coordinated Universal Time (UTC) which is also known as Zulu Time (Z time) and is in practice equivalent to Greenwich Mean Time (GMT). The key parameter in the calculation of the storage bill is the number of byte hours accounted to the customer. **Byte Hours** (ByteHrs) is the number of bytes that a customer stores in his/her 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 × 1 = 10 ByteHrs; similarly, if the customer stores 10 bytes for a whole day, their storage consumption for March would be 10 × 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 × 744 = 7440 BytesHrs.

From now on, we will assume that charging is for European customers accessing the S3 service from the 'outside internet', that is, not from within Amazon web services − for example, an application running on Amazon's Elastic Compute Cloud (EC2) and accessed data stored in S3. Current prices (in US dollars) read as follows:

- **Storage cost -** the first 50 TB cost 15 cents per GB per month.
- **Bandwidth cost -** Amazon distinguishes between DataTransfer-In and Data Transfer-Out (as explained in Section 2.1). There was no charge for DataTransfer-In up to Jun 30th 2010; thereafter the charge changed to 10 cents per GB. The first 10 TB of DataTransfer-Out cost 15 cents per GB.
- **Operations cost -** A block of 1,000 operations composed of PUT, COPY, POST or LIST costs one cent, whereas a block of 10,000 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 consumption increases, for example, the second 50 TB of storage costs 11 cents per GB per month.

3.3.1 Charging schema for storage

In the FAQs page, 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 is provided 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*. According to Amazon the total number of bytes consumed for each day of March is 2,684,354,560; thus the total number of ByteHrs is calculated as 2,684,354,560×31×24 = 1,997,159,792,640 which is equivalent to 2.5 GB/Months. At a price of 15 cents per gigabyte per month, the total charge amounts to 2.5×15 = 37.5 cents.

Amazon explains that *at least twice a day, we check to see how much storage is used by your Amazon S3 buckets. The result is multiplied by the amount of time passed since the last checkpoint.* Their records of storage consumption in ByteHrs can be retrieved from the Usage Reports associated with each account.

3.3.2 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 get 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. Amazon 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 explains 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 × (1/1,024) × 31 = 15.14GB. At a price of 10 cents per gigabyte, the total charge would be 15.14 × 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.

3.3.3 Charging schema for operations

To illustrate their charging schema, they provide an example in the Amazon Simple Storage Service FAQs in which *You transfer 1,000 files into Amazon S3 and transfer 2,000 files out of Amazon S3 each day during the month of March, and delete 5,000 files on March 31st.* In this scenario, the total number of PUT requests is calculated as 1,000 × 31 = 31,000; whereas the total number of GET requests is calculated as 2,000 × 31 = 62,000. The total number of DELETE requests is simply 5,000, though this is irrelevant as DELETE requests are free. At the price of one cent per 1,000 PUT requests and one cent per 10,000 GET requests, the total charge for the operations is calculated as 31,000 × (1/1,000) + 62,000 × (1/10,000) = 37.2 cents.

3.3.4 Error handling

As explained in the Handling Errors Section of [37], 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 to identify the party responsible for the failure – the customer or the S3 infrastructure. For example, *NoSuckBucket* errors are caused by the customer when they try to upload a file into a nonexistent 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.

3.3.5 S3 billing records

Among the on-line records that Amazon retains for each S3 account is a repository of two documents related to customers' bills, namely: Account Activity and Usage Reports. The **Account Activity** document 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. The previous and current month's statements are available.

3.3.6 Usage report

There are multiple ways for an Amazon S3 customer to view their resource usage, for example, as explained in [4], by viewing a summary of all usage associated with their bill through the AWS portal, or by generating detailed access logs for specified buckets.

3.3.6.1 Tracking Usage in the Amazon Web Service (AWS) Portal

An Amazon S3 customer can view their Usage Report by logging into their AWS account and selecting Usage Report. The Usage Report provides a summary of usage data for a specific time period. It also provides statistics on usage for all customer buckets. The data in the Usage Report is the same data that is used by Amazon S3 to calculate the customer's web service bill. The available data in a Usage Report is organized according to usage type and operation. The usage type is the category of usage data that the customer wants to report. The data under each usage type is further categorized by the operation or type of storage that is associated with each data point in the report. Amazon S3 reports the following usage types:

- **TimedStorage-ByteHrs:** this contains records of the amount of storage the consumer has used over time. **TimedStorage-ByteHrs** represents how much storage has been used by all the objects in all customers' buckets, multiplied by the number of hours since the last checkpoint. At least twice a day, S3 checks how much storage is being used by all objects in all customer buckets. The data is provided in units of byte-hours.
- **AverageStorage-Bytes:** this usage type contains another, more intuitive view of the customer's storage usage. **AverageStorage-Bytes** represent the average total storage used by all the objects in all buckets per day. The data is provided in units of bytes. This data is directly calculated from the data stored with the TimedStorage-ByteHrs usage type. Data for this usage type is only available in

daily granularity. This data is provided for information purposes only and does not impact the customer bill.

- **Network-Bytes:** this contains records of network data transfer associated with the customer account. Every time a request is received to PUT an object, GET an object, or list a bucket, the amount of network traffic involved in transmitting the object data, metadata, or keys, is recorded here. The customer can, if they so wish, view the network usage associated with one of these operations individually by specifying the operation of choice before generating the Usage Report.
- **Request:** this usage type contains information about the number of requests received for various common Amazon S3 operations. This data is provided for information purposes only and does not impact the customer bill. The customer can choose to view the number of requests to PUT an object, GET an object, delete an object, or list a bucket related to their account.

3.3.7 Amazon S3 experiments and results

In an attempt to audit our own S3 bill, we have run several experiments to understand how S3 computes the resource consumption for storage, bandwidth and operation. In other words, we have tried to understand S3 accounting model by conducting those experiments to see if we could extract a formula that can be used by the consumer to compute their own resource consumption based on its own metering data, and produces accounting data that matches with the measurement provided by S3.

3.3.7.1 Amazon S3 usage report

In an attempt to audit our own S3 bill, we studied Amazon's Usage Report, aiming to gain a complete understanding of how Amazon S3 represents the accounting data and what the meaning of each item of data presented in the Amazon Usage Report.

- **The aim:** to understand how Amazon S3's accounting data is represented and the meaning of each item in the accounting data.
- **Client Actions:**

- Client has already created a number of Buckets and uploaded a number of objects into each Bucket.
- Client Executes a number of operation requests (e.g. ListMyBucket).
- Client downloads Amazon S3's Usage Report, which shows Amazon S3's accounting data.

- **Observation**

As shown in Table 3.1, the Amazon S3 Usage Report is divided into seven main fields. These fields are ***Service***, ***Operation, UsageType, Resource, StartTime, EndTime*** and ***Usage Value***, where the ***Service*** represents the name of the service (Amazon S3); ***Operation*** represents the name of the operation invoked (e.g. GetObject); ***UsageType*** represents the type of resource that has been consumed by the operation (e.g. storage or bandwidth); and ***Resource*** represents the name of the Bucket that is or will be used to store objects; while ***StartTime*** and ***EndTime*** represent the start and end of the consumption interval. Other important details that can be understood from Amazon S3's Usage Report can be summarised as follows:

Operation	UsageType	Resource	StartTime	EndTime	UsageValu
ListBucket	Requests-Tier1	nclMetering	11/05/2009 00:00	11/06/2009 00:00	1
ListBucket	DataTransfer-Out-Bytes	nclMetering	11/05/2009 00:00	11/06/2009 00:00	237
ListBucket	DataTransfer-Out-Bytes	nclMetering	11/10/2009 00:00	11/11/2009 00:00	237
ListBucket	Requests-Tier1	nclMetering	11/10/2009 00:00	11/11/2009 00:00	1
ListBucket	Requests-Tier1	nclMetering	11/13/09 00:00:00	11/14/09 00:00:00	1
ListBucket	DataTransfer-Out-Bytes	nclMetering	11/13/09 00:00:00	11/14/09 00:00:00	237
ListBucket	Requests-Tier1	nclMetering	11/15/09 00:00:00	11/16/09 00:00:00	1
ListBucket	DataTransfer-Out-Bytes	nclMetering	11/15/09 00:00:00	11/16/09 00:00:00	237
PutObject	Requests-Tier1	nclMetering	11/15/09 00:00:00	11/16/09 00:00:00	1
PutObject	DataTransfer-In-Bytes	nclMetering	11/15/09 00:00:00	11/16/09 00:00:00	1108618
PutObject	Requests-Tier1	nclMetering	11/16/09 00:00:00	11/17/09 00:00:00	779
PutObject	DataTransfer-In-Bytes	nclMetering	11/16/09 00:00:00	11/17/09 00:00:00	862536207
StandardStorage	TimedStorage-ByteHrs	nclMetering	11/16/09 00:00:00	11/17/09 00:00:00	4284570264
StandardStorage	StorageObjectCount	nclMetering	11/16/09 00:00:00	11/17/09 00:00:00	162
StandardStorage	StorageObjectCount	nclMetering	11/17/09 00:00:00	11/18/09 00:00:00	780
StandardStorage	TimedStorage-ByteHrs	nclMetering	11/17/09 00:00:00	11/18/09 00:00:00	20728037400
StandardStorage	StorageObjectCount	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	780
StandardStorage	TimedStorage-ByteHrs	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	20728037400
ListBucket	DataTransfer-Out-Bytes	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	792852
ListBucket	Requests-Tier1	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	3
GetObject	Requests-Tier2	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	62
DeleteObject	Requests-NoCharge	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	62

Table 3. 1 Amazon S3's Usage Report

- Amazon S3 has divided resource consumption into consumption intervals (CI), where the length of each CI can be equivalent to an hour, a day or a week. We have selected the daily interval basis. Furthermore, we have found the following:

- Each CI has a start and end point (SP and EP respectively).
- The EP of CI_i represents the SP of CI_{i+1}.
- Resource types are divided into storage and bandwidth consumption and operation (computer power).
- The consumption of each resource type is computed for each Bucket separately.

o Storage consumption is represented by "StandardStorage"

- StandardStorage is divided into two sub usage types: TimedStorage-ByteHrs and StorageObjectCount. The first represents the total amount of bytes consumed per bucket over the CI. The TimedStorage-ByteHrs data is provided in bytes-hours. The latter represents the number of objects in the Bucket that consumed the "TimedStorage-ByteHrs" over the CI.

o Bandwidth consumption is represented by two sub usage types

- DataTransfer-In-Bytes represents the total amount of bytes transferred into (uploaded into) the Bucket entity in the S3 account for each request type over each CI.

Operation	UsageType	Resource	StartTime	EndTime	UsageValue
PutObject	DataTransfer-In-Bytes	nclMetering	11/15/09 00:00:00	11/16/09 00:00:00	1108618

Table 3. 2 Upload Bandwidth Consumption for Put Requests

Table 3.2 shows that the total number of Bytes uploaded by all PUT requests into nclMetering Bucket during the period between 15/11/09:00:00:00 and 16/11/09:00:00:00 = 1108618 Bytes.

- DataTransfer-Out-Bytes represent the total amount of bytes transferred from (downloaded from) a Bucket entity in the S3 account by each request type over each CI.

Operation	UsageType	Resource	StartTime	EndTime	UsageValue
GetObject	DataTransfer-Out-Bytes	nclMetering	11/18/09 00:00:00	11/19/09 00:00:00	17236

Table 3. 3 Download Bandwidth Consumption for Put Requests

Table 3.3 shows that the total number of Bytes downloaded by all GET requests from nclMetering Bucket during the period between 18/11/09:00:00:00 and 19/11/09:00:00:00 = 17236 Bytes.

- Bandwidth subtypes are computed separately for each type of operation per bucket. The data is provided in units of bytes.

- Operation (Computer power) represented by three sub usage types
  - Request-Tier1: represent the total number of (PUT, ListBucket, ListAllMyBucket and CreateBucket) requests during the CI. For example, PUT Request-Tier1=12, means during the CI the customer executed 12 PUT requests.
  - Request-Tier2: represents the total number of GET requests during the CI.
  - NoCharge: represents the total number of delete requests over the CI.
- Amazon's S3 Usage Report uses 24 hours as the time occupied by each object.

3.3.7.2 Storage consumption (SC)

In an attempt to allow the S3 customer to verify his storage consumption the 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; 2) how Amazon determines the number of hours that a given piece of data has been stored in S3; and 3) when the resource consumption is computed. To clarify all these issues we have conducted the following experiments:

- **Client Actions:**
  - Create a number of Buckets and upload a number of objects into each Bucket, under the following assumptions:
    - Use different lengths of Bucket name, ranging between 3-20 characters.
  - Execute a number of PUT requests with different parameters; these parameters are:
    - Use different lengths of object name between 10-20 characters.
    - The uploaded user metadata is between 0 up to 2KB.

- The size of object is between 0 to 5GB bytes.
- Each object is uploaded into empty Bucket.

o From the customer request/response details, the consumer collects the following metering data: {Request Id, URI, Operation type, Bucket name, Object name, Request Time Stamp (RTS), Byte transferred by response "send in" (BT_{Req}), Access Key (AK), Signature (Singn), Response Time Stamp (TM_{Res}) and Bytes Transferred per Response (BT_{Res})}.

o Downloads S3's Usage Report.

• Observation

From the metering data collected from request/response details by the consumer's metering service and S3 Usage Report, we obtained the following outlined points:

i. Data and metadata

Amazon S3 does not explain how to calculate the actual storage space taken up by data and metadata. To clarify this issue, we uploaded a number of objects of different name lengths, data and user metadata into an equal number of empty buckets.

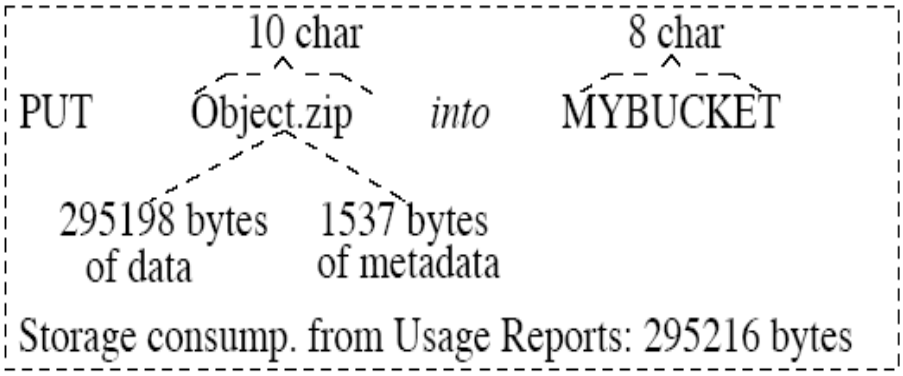


Figure 3. 2 Impact of data and metadata in storage consumption

Figure 3.2 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 characters long, which is equivalent to ten and eight bytes, respectively. The object data and metadata shown in Figure 10 correspond to information we extracted locally from the PUT request. By contrast,

the storage consumption of 295,216 bytes corresponds to what we found in the usage reports. The actual usage reports show storage consumption per day in BytesHrs; and 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 multiplied by the length of the consumption interval (day = 24 hours): (8 + 10 + 295,198) * 24 = 295,216 * 24 Bytes-Hrs. Three conclusions can be drawn from this observation: first, the mapping between bytes uploaded by PUT requests and bytes stored in S3 corresponds one-to-one; secondly, object and bucket names represent what Amazon calls storage overheads 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. All related experiments are presented in the Appendix on Storage Consumption.

ii. Checkpoints

Amazon S3 states that they check the amount of storage consumed by a customer at least twice daily. However, Amazon S3 does not stipulate exactly when the checkpoints take place. To clarify the situation, we conducted a number of experiments that consisted in uploading and deleting files in S3 and studying the Usage Reports of our account to detect when the impact of the PUT and DELETE operations were accounted by Amazon.

- **Client Actions:**
    - Create a number of new Buckets.
    - Execute a number of PUT requests under the following assumptions:
        - Upload an object each minute into a Bucket.
        - Each object has the same size, name length and a different name.
        - Determine the length of loading period (LP), where each LP should have a start and end point. For example, as shown in Figure 11, LP1 represents the first LP, it starts at the SP of CI_i, and ends after the EP of CI_{i+1}.

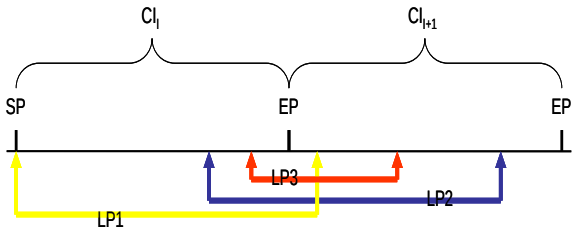


Figure 3. 3 uploading and checkpoint

- As shown in Figure 3.3, different loads are applied, where the start and end point of each LP may start or end before or after the SP and EP of the entity's CI respectively.
- For some experiments, a random point was selected at which to randomly delete a number of objects.

- Download the S3 Usage Report.

• Observation

Our findings are summarised in Table 3.4. From the request time stamp of the last object counted for storage for the entity's consumption interval, it seems that, currently, Amazon does actually check customers' storage consumption only once a day. From our observations, it emerged that the time of the checkpoint is decided randomly by Amazon S3 within the 00:00:00Z and 23:59:59Z time interval, which actually represented the start and end point of each CI.

Consumer's Metering Data						Amazon S3's Usages Reports		
Bucket name	Upload start point	Upload end point	No. of uploaded object	Object No.	Request Time Stamp	No. of counted object	Start point	End point
nclMetering16	18/11/09 23:01:00	19/11/09 03:28:00	210	210	19/11/09 03:27:23	210	19/11/09 00:00:00	20/11/09 00:00:00
nclMetering22	25/11/09 02:00:00	25/11/09 05:22:00	180	14	25/11/09 02:17:38	14	25/11/09 00:00:00	26/11/09 00:00:00
nclMetering25	25/11/09 21:59:00	26/11/09 11:09:00	780	309	26/11/09 02:16:41	309	25/11/09 00:00:00	26/11/09 00:00:00

Table 3. 4 Amazon S3's checkpoint

As shown in Figure 3.4, Amazon uses the results produced by a checkpoint for a given day, to generate the account for 24 hrs of that day for the customer, regardless of the operations that 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 × 24 = 48 GBHrs; where 2 represents the 2GB that the customer uploaded on the 30th and 24 represents the 24 hrs of the day.

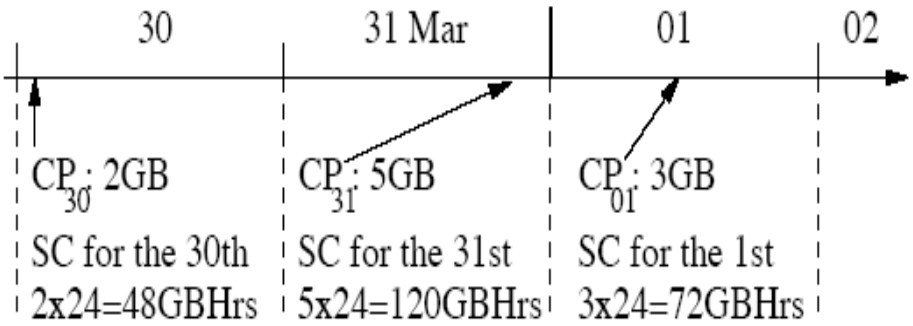


Figure 3. 4 Amazon S3's checkpoint

Finally, we have observed that uploading and deleting objects between two checkpoints does not affect storage consumption; however it affects other resource consumption. More details can be found in Appendix 3 (Amazon S3's Check Points).

iii. Operations consumption

In order to understand how and when Amazon S3 computes the Operation consumption for each request, we have conducted a number of experiments that consisted in executing a different number of operations several times for each consumption interval, for example, uploading 10 objects into a Bucket and studying Amazon S3's Usage Reports to know how operations were accounted by Amazon.

In all the experiments we have done (using RESTful or SOAP interfaces), we found that the number of each request is counted and presented in the Amazon S3 Usage Report as Request-Tier1 or Request-Tier2, depending on the type of request. For example, PutObject is executed 7 times in order to upload 7 objects into MYBUCKET-04, the Amazon Usage Report, as we can see in Table 3.5, represents 7 as the value of Request-Tier1 which is exactly equal to the total number of executed PutObject requests. By running this experiment for a range of possible request types; the same results were obtained.

Operation	UsageType	Resource	StartTime	EndTime	UsageValue
PutObject	DataTransfer-In-Bytes	MYBUCKET-04	03/10/2010 00:00	03/11/2010 00:00	111111
PutObject	Requests-Tier1	MYBUCKET-04	03/10/2010 00:00	03/11/2010 00:00	7

Table 3. 5 Amazon S3 Usage Report for Operations

To clarify whether failed operations are counted or not we executed a number of operations including ones that were both valid and invalid (for example, the creation of buckets with invalid names and with names that already existed). Next, we examined the usage reports and, as expected, we found that Amazon counted both successful and failed operations. Figure 3.5 shows an example of the operations that we executed and the bandwidth and operation consumption that they caused, in accordance with the usage reports.

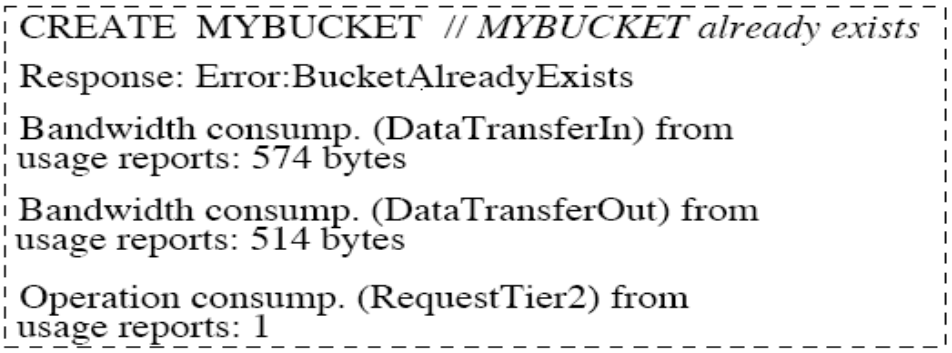


Figure 3. 5 bandwidth and operation consumption of a failed operation

Thus, the failed operation to create a bucket consumed 574 bytes of DataTransfer-In and 514 bytes of DataTransfer-Out. These figures correspond to the size of the SOAP request and response, respectively. As shown in Figure 3.5, we also found out that the failed operation incurred operation consumption and was counted by the RequestTier2 parameter in the Usage Reports. For more details see Appendix 6 (Error Handling).

Similar to bandwidth consumption, Amazon S3 has a fixed checkpoint for Operation consumption, which is equal to the end point of each consumption interval. More details are found in Appendix 3 (Amazon S3's Check Points).

3.3.7.4 Bandwidth consumption

As stated earlier, bandwidth consumption represented by DataTransfer-In and DataTransfer-Out includes, respectively, request and response overheads. The difficulty here is that from the Amazon accounting model, it is not clear how message size is calculated in DataTransfer-In and DataTransfer-Out. To clarify the point, we have run several experiments using RESTful and SOAP interfaces.

i. Restful Bandwidth consumption

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 Figure 3.6, where PUT operations are used to upload an object into a bucket.

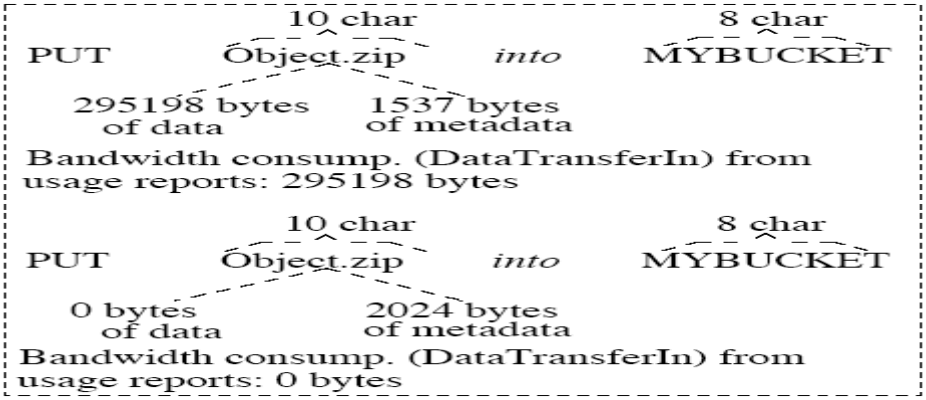


Figure 3. 6 bandwidth consumption

The data and metadata shown in the Figure represent the data and metadata extracted locally from the PUT requests.

As shown by the Bandwidth consump. Parameters extracted from the usage reports, only the object data consumes DataTransfer-In bandwidth; neither the metadata nor the object nor the bucket names seem to count as overheads. However, this observation applies to RESTful requests. All related experiments details and results are presented in Appendix: 4 (RESTful Bandwidth Consumption).

ii. SOAP Bandwidth consumption

We have executed a number of operations and collected metering data from the requests details; we have also compared information extracted from each operation request against bandwidth consumption as counted in the Usage Report. These examples of the experiments that we conducted are shown in Table 3.6: we executed PUT operations to upload an object into a bucket, used a CreateBucket operation to create a new bucket and GetObject to download an object. The data shown in Table 3.6 represents the metering data extracted locally from each request.

Bucket Name	Operation	Object Name	Object Size	Metadata Size	Request Size	Request Timestamp	Response Size
MYBUCKET-104	PutObjectInline	C	1000 Bytes	0 Bytes	1584 Bytes	2010-04-06T 15:11:54.034Z	821 Bytes
MYBUCKET-999	CreateBucket	-	-	-	518 Bytes	2010-04-07T 14:03:10.456Z	822 Bytes
MYBUCKET-104	GetObject	C	0	0	608 Bytes	2010-04-07T 14:20:16.241Z	1955 Bytes
MYBUCKET-104	DeleteObject	C	0	0	547 Bytes	2010-04-07T 15:36:18.451Z	752Bytes

Table 3. 6 SOAP Requests Metering Data

Tables 3.6 and 3.7 show that for SOAP messages the total size of the message is always used for calculating bandwidth consumption. More importantly, each SOAP operation consumes DataTransfer-In and DataTransfer-Out, by contrast with a RESTful operation, which consumes just one resource (DataTransfer-In or DataTransfer-Out), based on the operation. For instance, the PUT operation just consumes DataTransfer-In.

Operation	UsageType	Resource	StartTime	EndTime	UsageValue
PutObject	DataTransfer-Out-Bytes	MYBUCKET-104	04/06/2010 00:00	04/07/2010 00:00	821
PutObject	Requests-Tier1	MYBUCKET-104	04/06/2010 00:00	04/07/2010 00:00	1
PutObject	DataTransfer-In-Bytes	MYBUCKET-104	04/06/2010 00:00	04/07/2010 00:00	1584
GetObject	DataTransfer-Out-Bytes	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	1955
GetObject	DataTransfer-In-Bytes	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	608
GetObject	Requests-Tier2	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	1
DeleteObject	Requests-NoCharge	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	1
DeleteObject	DataTransfer-Out-Bytes	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	752
DeleteObject	DataTransfer-In-Bytes	MYBUCKET-104	04/07/2010 00:00	04/08/2010 00:00	530
CreateBucket	Requests-Tier1	MYBUCKET-999	04/07/2010 00:00	04/08/2010 00:00	1
CreateBucket	DataTransfer-Out-Bytes	MYBUCKET-999	04/07/2010 00:00	04/08/2010 00:00	822
CreateBucket	DataTransfer-In-Bytes	MYBUCKET-999	04/07/2010 00:00	04/08/2010 00:00	518
ListAllMyBuckets	Requests-Tier1		05/04/2010 00:00	05/05/2010 00:00	1
ListAllMyBuckets	DataTransfer-Out-Bytes		05/04/2010 00:00	05/05/2010 00:00	8099
ListAllMyBuckets	DataTransfer-In-Bytes		05/04/2010 00:00	05/05/2010 00:00	497

Table 3. 7 Amazon S3 Usage Report

Furthermore, opposite to the storage consumption outcome, Amazon S3 has a fixed checkpoint for Bandwidth consumption; it is equal to the end point of each consumption interval. All related experimental details and results are presented in Appendix 5 (SOAP Bandwidth Consumption).

3.3.8 Amazon S3 Accounting Model Description

The accounting model is defined as a method used to compute resource consumption; it could be one or more mathematical formulae. Moreover, the accounting model includes details about when the resource consumption is computed, a description of each resource consumption record, how the resource consumption is computed for each resource and so on. Based on the results of the experiments described above, Amazon S3's accounting model can be described as follows:

3.3.8.1 General Characteristics of the S3 Accounting Model

1. Resource consumption is divided into Consumption Intervals (CI).
2. Each CI has a start and end point (SP and EP respectively).
3. The length of each CI is divided on hour, day and week bases; we have selected the daily basis (24 hours).
4. The SP and EP of each CI are represented by the DD/MM/YYYY 00:00:00 of today and the DD/MM/YYYY 00:00:00 of the next day, respectively.
5. The EP of CI_i is represented by the SP of CI_{i+1}.
6. Checkpoint (CP) is the moment of time when the resource consumption is computed.
7. Different resources may have different CP for the same CI.
    - EP of each CI was selected as Amazon S3's fixed CP for Bandwidth and Operation.
    - Amazon S3 does not have a fixed CP for storage consumption where, CP >= SP or CP<=EP.
8. Amazon S3 allowed their customers to download their resource consumption report.

3.3.8.2 Storage Accounting

Amazon S3 applies the following accounting model to compute storage consumption:

1. Compute SC for each upload request related to a bucket by:

2. $$Object_{SC} = (Objectsize + objectMetadata) * 24 \quad (1)$$

   Where Objectsize = number of bytes transferred per request, and

   $$ObjectMetadata = the\ length\ of\ object\ key + the\ length\ of\ Bucket\ Key \quad (2)$$

3. Compute the deleted SC per delete request by:

   $$DelObject_{SC} = (Objectsize + objectMetadata) * 24 \quad (3)$$

4. Compute the SC for each Day

   $$Day_{SC_i} = Day_{SC_{i-1}} + \sum_{j=1}^{N} Object_{SC_j} - \sum_{k=1}^{M} DelObject_{SC_k} \quad (4)$$

Where N and M representing the number of upload and delete requests per day respectively

5. At the end of the month the total SC is computed by the following formula:

   $$Month_{SC} = \sum_{i=1}^{N} Day_{SC_i} \quad (5)$$

Where N represents the number of days per month

6. Finally, convert the total SC/Month into GB/month by the following formula:

= output of (4) x (1 GB / 1,073,741,824 bytes) x (1 month / 744 hours in May) (6)

3.3.8.3 Upload Bandwidth Accounting

Amazon S3 applies the following model to compute upload bandwidth consumption:

1. Compute Upload Bandwidth Consumption (UBC) for each request (Req) by:

   $$Req_{UBC} = BT_{req} + Req_{OH} \quad (1)$$

   Where BT_{Req} = No of Bytes transferred per request and Req_{OH} = request overhead, the value of Req_{OH} for RESTful request =0, where in SOAP the value of Req_{OH} is depending on the type of request.

2. Compute the UBC for each day:

   $$Day_{UBC} = \sum_{i=1}^{N} Req_{UBC_i} \quad (2)$$

Where N represents the number of upload requests per day

3. At the end of the month the total UBC is computed by the following formula:

$$\text{Month}_{\text{UBC}} = \sum_{i=1}^{N} \text{Day}_{\text{UBC}_i} \quad (3)$$

Where N represents the number of days per month

4. Finally, convert the total Upload Bandwidth Consumption into GB/month by the following formula:
   = output of (3) x (1 GB / 1,073,741,824 bytes) (4)

3.3.8.4 Download Bandwidth Accounting

Amazon S3 applies the following model to compute download bandwidth consumption:

1. Compute Download Bandwidth Consumption (DBC) for each response (Res) by:

   $$\text{Res}_{DBC} = BT_{\text{Res}} + \text{Res}_{OH} \quad (1)$$

   Where BT_{Res} = No of Bytes transferred per response and Res_{OH} = response overhead, the value of Res_{OH} for RESTful response = 0, where in SOAP the value of Res_{OH} depends on the type of response.
2. Compute the DBC for a day by:

   $$\text{Day}_{\text{DBC}} = \sum_{i=1}^{N} \text{Res}_{\text{DBC}_i} \quad (2)$$

Where N represents the number of responses per day

3. At the end of the month the total DBC is computed by the following formula:

   $$\text{Month}_{\text{DBC}} = \sum_{i=1}^{N} \text{Day}_{\text{DBC}_i} \quad (3)$$

Where N represents the days of responses per month

4. Finally, convert the total Upload Bandwidth Consumption into GB/month by the following formula:
   = output of (3) x (1 GB / 1,073,741,824 bytes) (4)

3.3.8.5 Operation Accounting

The numbers of requests that are issued during a day represent the operation consumption (computer power).

i. Requests Tier1

1. Tier1 requests: computed all requests issued during a day by:

$$\mathrm{Day}_{\mathrm{Tire1}} = \sum_{i=1}^{N} \mathrm{PUT}_i + \sum_{j=1}^{M} \mathrm{COPY}_j + \sum_{k=1}^{S} \mathrm{POST}_k + \sum_{l=1}^{R} \mathrm{LIST}_l \quad (1)$$

Where N, M, S and R represent the number of requests per day

2. At the end of the month compute Tier1 consumption by:

$$\mathrm{Month}_{\mathrm{Tier1}} = \sum_{i=1}^{N} \mathrm{Day}_{\mathrm{Tier1}_i} \quad (2)$$

Where N represents the number of days per month

ii. Requests Tier2

1. Tier2 requests: computed all requests issued during a day by:

$$\mathrm{Day}_{\mathrm{Tier2}} = \sum_{i=1}^{N} \mathrm{GET}_i + \sum_{j=1}^{M} \mathrm{OReq}_j \quad (1)$$

Where N and M represent the number of requests per day

2. At the end of the month compute Tier2 consumption by:

$$\mathrm{Month}_{\mathrm{Tier2}} = \sum_{i=1}^{N} \mathrm{Day}_{\mathrm{Tier2}_i} \quad (2)$$

Where N represents the number of days per month

3.3.9 Shortcomings in the Amazon S3 Accounting Model

The Amazon S3 customer is charged for storage, bandwidth and operations performed. In the previous subsections we examined whether the data that the service provider 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 would be possible, if Amazon S3 provided a full and clear description of their accounting model. However, from our experiments we have identified the following shortcomings as described below.

In particular, for storage consumption, the accounting model needs explicitly to state how the data and metadata that is uploaded is mapped into consumed bytes by the

Service Provider. For example, in S3 our experiments showed that user metadata does not impact storage consumption. Furthermore, we saw that errors are possible if the checkpoint times of Amazon S3 and of the customer calculating storage consumption are not sufficiently close. Ideally, Amazon's checkpoint times should be made known to customers to prevent any such errors.

In relation to bandwidth, as explained in Section 3.5, DataTransfer-In and DataTransfer-Out include, respectively, request and response overheads. The difficulty here is that from the Amazon S3 accounting model, it is not clear how message size is calculated in DataTransfer-In and DataTransfer-Out. The accounting model needs clearly to state how the DataTransfer-In and DataTransfer-Out are computed, as well as how they compute the request and response overheads, and present any details that affect the bandwidth computation of users using different interfaces.

One likely source of difficulty regarding the charges for operations is how to determine the liable party for failed operations. Currently, this decision is taken unilaterally by Amazon. In this regard, we anticipate two potential sources of conflict: 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 [38 ADG]. Amazon's advice is to design applications that can handle redirect errors, for example, by resending a request after receiving a 307 code (see [37], Request Routing section). Strictly speaking these errors are not caused by the customer as the 307 code suggests. It should be stated clearly by the Amazon S3 accounting model which party bears the cost of the re–tried operations.

3.3.10 Summary of Amazon S3 case study

3.3.10.1 General

The important result obtained is that an Amazon S3 customer can independently collect all the metering data that is required for calculating charges for the consumption of all Amazon S3's resources. Furthermore, experiments indicated that the description of the Amazon S3 accounting model is ambiguous and needs to be clarified by Amazon. In

particular, concerning storage, the S3 accounting model needs explicitly to state how the data and metadata that is uploaded is mapped into consumed bytes in the Service Provider. For example, S3 experiments showed that bytes transfer per request and object name and bucket name only impact storage consumption. We also pointed out that an operation executed by a request in RESTful consumes fewer resources than the same operation executed by a request in SOAP; for instance, a CreateBucket operation executed by SOAP request consumes operation, DataTransfer-In and DataTransfer-Out, whereas the same operation executed by RESTful request consumes operation only (see Appendix: Experiment 5.1 CreateBucket request and resource consumption). Moreover, the experiment results show that Amazon S3 has selected the end point of each consumption interval as a fixed checkpoint to calculate the resource consumption of operation and bandwidth. On the other hand, Amazon S3 has arbitrarily selected a point between the start and end point of each consumption interval to calculate storage consumption; which may lead to possible conflict between consumer and provider results. Finally, our results show that it is possible for an Amazon S3 consumer to independently implement their own RAS that can be used to compute and verify their resource consumption.

3.3.10.2 Storage

Six conclusions can be drawn from the experiments:

1. The mapping between bytes uploaded by PUT requests and bytes stored in S3 corresponds one-to-one.
2. Object and bucket names represent what Amazon calls storage overhead and incur storage consumption.
3. User metadata does not impact storage consumption.
4. An empty bucket does not consume any storage consumption.
5. From our observations, it emerged that the time of the checkpoint is decided randomly by Amazon S3 within the 00:00:00Z and 23:59:59Z time interval which actually represented the start and end point respectively of each CI.
6. We have observed that uploading and deleting objects between two checkpoints does not affect storage consumption; however it affects other resource consumption.

3.3.10.3 Operations

Three conclusions can be drawn from our experiments: first, straightforwardly, we found that the total number of each type of request is counted and presented in the Amazon S3 Usage Report as Request-Tier1 or Request-Tier2. Second, by contrast with storage we found that Amazon S3 has a fixed checkpoint for operation consumption; it is equal to the end point of each consumption interval. Finally, we also pointed out that a failed operation results in operation consumption as well as bandwidth consumption.

3.3.10.4 Bandwidth

First, for a RESTful request we have found that only the object data consumes bandwidth consumption; neither the metadata nor the object nor the bucket names seem to count as overheads. However, with a SOAP request we have found that the whole message (request or response) size is represented in bandwidth consumption. Secondly, a failed operation consumes bandwidth resource consumption, and as a result of this SOAP messages result in more bandwidth consumption than RESTful messages. Third, each SOAP operation takes up more resource consumption than an equivalent operation in RESTful, because every SOAP message consumes all resources, as opposed to a RESTful operation, which consumes only some resources . For instance, CreateBucket executed in RESTful only consumes operation resource, where in SOAP, it uses bandwidth consumption (DataTransfer-In and DataTransfer-Out) as well as operation resource. Finally, similar to the operation consumption outcome, it was found that Amazon S3 has a fixed checkpoint for Bandwidth consumption, which is equal to the end point of each consumption interval.

3.4 Second case study: Nirvanix Storage Delivery Network Services

Nirvanix advertises its Storage Delivery Network (SDN) service as a storage service available to Internet users on a pay-per-use basis [29]. Nirvanix SDN is a fully-managed, highly secure, cloud storage service developed for enterprises. Nirvanix SDN is promoted as a highly reliable, fast, data storage service accessible to subscribers

through a Web service interface. Currently, Nirvanix SDN provides both SOAP and RESTful interfaces [39]. A Nirvanix SDN space is organised as a collection of folders that support nesting. A folder can contain zero or more subfolders and files of up to 250 GB. Both folder and file are identified by names chosen by the customer.

To gain access to the service, customers need to open an account with Nirvanix SDN, provide a credit card number and agree to pay a bill at the end of each **calendar month**. Upon successful registration, the Nirvanix SDN user sets his user name and password, whereupon Nirvanix SDN provides the customer with an application key. A Nirvanix SDN customer is charged for a) **storage space:** storage space consumed by the files that they store in SDN; b) **bandwidth:** network traffic generated by the operations that the customer executes against the SDN interface; c) **media service:** which includes audio/video transcoding, image resizing and thumbnail generation; d) **experience package:** which includes unlimited media transcoding, unlimited search and virtual URL and e) **search.**

Information about pricing and the charging schema used to calculate the customer's bill is not documented. There is only one page – entitled 'The Nirvanix SDN - How To Buy/ Self Service Pricing' that describes the pricing system. However, a usage report associated with each Nirvanix NDS account provides complementary information.
Nirvanix NDS only publishes the price to the public of up to 2 TB for storage and bandwidth, and their storage price is more expensive than the Amazon S3 storage price. There is no reference to the time zone used by Nirvanix SDN to determine the start and end points of days and billing cycles. However, from the Authenticating SOAP Requests Section of the 'Nirvanix SDN Developer Guide' [39] it is clear that Nirvanix SDN servers are synchronised to Coordinated Universal Time (UTC) which is in practice equivalent to Greenwich Mean Time (GMT). There is no further documentation published by Nirvanix SDN about their accounting and billing system. Previous experience gained from the experiments with Amazon S3 will now be applied to investigating and understanding the Nirvanix SDN accounting model.
Nirvanix SDN current prices (in US dollars) read as follows:

- **Storage cost** ($0.25 for 1 GB/month for single node, $0.45 for 1GB/month in 2 nodes and $0.71 for 1 GB/month in 3 nodes) for first 2 TB.
- **Bandwidth cost**
    - **Upload cost** ($0.10 for 1 GB/month for single node, $0.20 for 1GB/month in 2 nodes and $0.30 for 1 GB/month in 3 nodes) for first 2 TB.
    - **Download cost** ($0.15 for 1 GB/month) for first 2 TB.
- **Search cost** is $0.20 per 1,000 calls.
- **Media Service cost** is $1 GB processed (based on source file).
- **Experience package cost** is +$0.20 GB/month stored.
- **Operations cost** is free.
- Minimum fee is $1/month.

It is worth clarifying that with Nirvanix SDN, prices increase slightly as the number of nodes is increased, for example, 1GB/month costs $0.25 in one node whereas it costs $0.45 in 2 nodes.

Frankly speaking, the documentation for Nirvanix is very weak in comparison with Amazon S3.

3.4.1 Nirvanix SDN Experiments and Results

The same experiments that were described in Section (5) for Amazon S3, have been conducted to understand the accounting model used by Nirvanix SDN for storage and bandwidth consumption. The results obtained from these experiments were as follows: Nirvanix SDN uses accounting model concepts that are almost the same as those used by Amazon S3, with a few small differences. Below we will discuss these differences for each resource.

3.4.1.1 Usage Report

Nirvanix SDN allows their users to download their Usage Report. The two options allowed when downloading the Usage Report are: Master Accounts Usage and Daily Usage; from these, we have selected the Daily Usage report. The Daily Usage Report is divided into four main fields, consisting of: ***Date***, ***Average Daily Storage, Total Upload***

Bandwidth and ***Total Download Bandwidth***. ***Date*** represents the ID of the consumption interval (CI); ***Average Daily Storage*** represents the storage consumption over CI; ***Total Upload Bandwidth*** represents the total number of bytes uploaded during a CI; and ***Total Download Bandwidth*** represents the total number of bytes downloaded during a CI. Other important details that can be understood from the Nirvanix SDN Daily Usage Report are:

- **Consumption interval**

  As shown in Table 3.8, the resource consumption in the Nirvanix SDN Daily Usage Report is divided into consumption intervals (CI), where each CI is represented by one day. This means that each day has a start point (00:00:00) and an end point (24:00:00), and the end point of CI_i represents the start point of CI_{i+1}.

Date	Average Daily Storage	Total Upload Bandwidth	Total Download Bandwidth
04/05/2010	0	1051292	1
04/06/2010	1051292	5483772	0
04/07/2010	1051292	913950	0
04/08/2010	1965242	161062	0
04/09/2010	2126304	161062	0
04/10/2010	161062	0	0
04/11/2010	161062	20570408	0
04/12/2010	20731470	0	0

Table 3. 8 Nirvanix SDN's Daily Usage Report.

- **Resources**

  There are three main resources (storage, upload and download bandwidth) computed for each CI. Storage and bandwidth data is represented in Bytes.

3.4.1.2 Storage

As Nirvanix use GB/Month to calculate their bill, the customer needs to understand: 1) how their GB consumption is measured, that is, how the data and metadata that is uploaded is mapped into consumed bytes in Nirvanix SDN; and 2) at what points Nirnanix SDN computes the resource consumption; this issue is directly related to the notion of a checkpoint. To clarify all these issues we have conducted the following experiments.

- **Client Actions:**

- Create a number of folders and upload a number of files into each folder, under the following assumption:
  - Use different lengths of folder name between 3-20 characters.
- Execute a number of PUT requests with different parameters; these parameters are:
  - Use different lengths of file name between 10-20 characters.
  - The uploaded user metadata is ranges from 0 up to 2KB.
  - The size of a file is between 0 to 5GB.
  - Each file is uploaded into an empty folder.
- From the customer request/response the consumer collects the following metering data: {Request Id (ID), URI, Operation type (OT), Folder name (FN), Object name, Request Time Stamp (RTS), Byte transferred by response "send in" (BT_{Req}), Access Key (AK), Signature (Singn), Response Time Stamp (TM_{Res}) and Bytes Transferred per Response (BT_{Res})}.
- Downloads Nirvanix SDN's Usage Report.

From the metering data collected from request / response details by the consumer's metering service and the Nirvanix SDN Usage Report we obtained the following results.

i. Data and metadata

Figure 3.7 shows the parameters and results from one of our upload operations, where a file named PersonalData.doc is uploaded into a folder named *MyFolder*, which was originally empty. Notice that in this example, the file name is 16 characters and the folder name is eight characters long, which is equivalent to 16 and eight bytes, respectively.

The file data and metadata shown in the figure correspond to information we extracted locally from the PUT request. By contrast, the storage consumption of 26,753,890 bytes corresponds to what we found recorded in the Usage Reports.

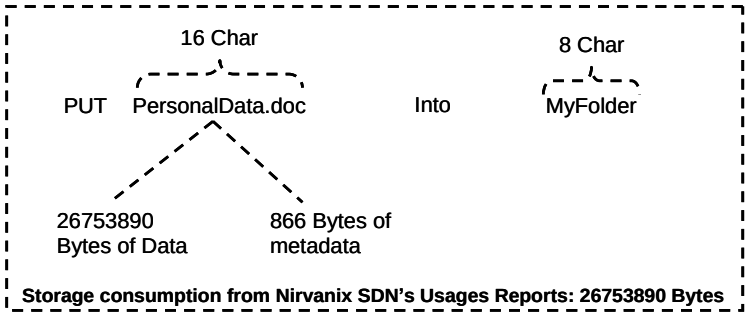


Figure 3. 7 impact of data and metadata in storage consumption

The actual Usage Reports show storage consumption per day in Bytes. Notice that this storage consumption exactly equals the file data. Two conclusions can be drawn from this observation: first, the mapping between bytes uploaded by PUT requests and bytes stored in Nirvanix SDN corresponds one-to-one; secondly, user and system metadata do not impact storage consumption. In addition to the experiments discussed above, we created a number of empty folders and verified from the Usage Reports that they do not consume storage space.

ii. Checkpoints

Nirvanix SDN does not proffer any details about when their checkpoints take place. To clarify the situation, we have conducted a number of experiments that consisted in uploading and deleting files in Nirvanix SDN and studying the Usage Reports of our account to detect when the impact of the UPLOAD and DELETE operations were accounted by Nirvanix SDN.

- **Client Actions:**
    - Create a number of new folders.
    - Execute a number of PUT requests under the following assumption:
        - Delete all existing folders.
        - Create new folder.
        - During the hours of daytime (from 8 AM to 8 PM) upload a number of files into an empty folder where all files have the same size.
        - At some point, which should be selected randomly between (00:00:00 GMT and 23:59:59 GMT), the client deletes all files from the folder.

- Downloads Nirvanix SDN Usage Report.

- **Observation**

As shown in Table 3.9, we found that, currently, Nirvanix usually computes storage consumption at the start point for each consumption interval, which can be exactly represented by 00:00:00 GMT. That means Nirvanix SDN selected the start point of each consumption interval as a fixed checkpoint. However, the end point of each consumption interval was selected by Nirvanix SDN as its fixed checkpoint to compute the bandwidth resource consumption.

Consumer's Metering Data					**Nirvanix SDN Usages Report**		
Folder name	Upload Date	No of upload files	Total data uploaded	Delete time	Date	Storage consumption	Bandwidth consumption
MyFolder01	04/07/2010	20	2000000 Bytes	04:07:2010T23:59:59	04/07/2010	0 bytes	2000000 bytes
MyFolder02	05/07/2010	0	0	n/a	05/07/2010	0 bytes	0
MyFolder03	06/07/2010	20	2000000 Bytes	07:07:2010T00:00:59	06/07/2010	0 bytes	2000000 bytes
MyFolder04	07/07/2010	0	0	n/a	07/07/2010	2000000 bytes	0 bytes

Table 3. 9 Nirvanix SDN's Usage Report and Consumer's metering data

In Figure 3.8, CP stands for checkpoint, CI stands for consumption interval, while SP and EP stand for start and end points of the CI respectively; thus CP_3: 2GB indicates that CP_3 was conducted on the 3rd day of the month at 00:00:00 GMT, as specified by the arrow and reported that at that time the customer had 2 GB stored in Nirvanix. SC stands for Storage Consumption.

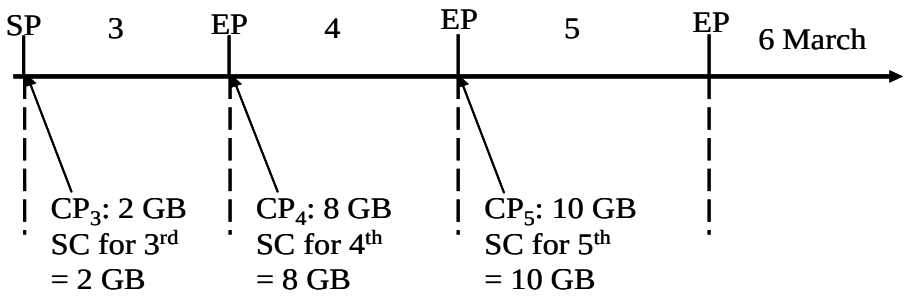


Figure 3. 8 Nirvanix SDN's checkpoint

Similar to Amazon S3, uploading and deleting the same object between two checkpoints does not affect storage consumption; however it does affect bandwidth consumption.

3.4.2 Operations

Nirvanix SDN does not charge for operations where operation represents the number of requests issued against the service interface by the consumer in specific consumption interval.

3.4.3 Bandwidth

As we stated earlier there is no documentation provided on the Nirvanix SDN accounting model. Nirvanix SDN charges their customer for Upload and Download bandwidth. To clarify how Nirvanix SDN computes each charge, 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 Table 3.10, where we used PUT operations to upload a file into a folder. The data and metadata shown in the Table represent the data and metadata extracted locally from the PUT requests.

Consumer's Metering Data					Nirvanix SDN Usages Report	
Folder name	Upload Date	File name	File Size	Metadata size	Date	Bandwidth consumption
MyFolder01	09/07/2010	Personal.doc	100000 Bytes	2000 Bytes	09/07/2010	100000 bytes
MyFolder02	10/07/2010	Personal.doc	100000 Bytes	0	10/07/2010	100000 bytes

Table 3. 10 Bandwidth consumption

As shown by the Bandwidth consumption parameters extracted from the Usage Reports, only the file data consumes upload bandwidth; neither the metadata nor the file or folder names seem to count as overheads. This observation refers to RESTful requests.

3.4.4 Nirvanix SDN Accounting Model Description

Based on the experiment results described above, Nirvanix SDN's accounting model can be described as follows.

3.4.4.1 General Characteristics

1. Resource consumption is divided into Consumption Intervals (CI).
2. Each CI has a start and end point (SP and EP respectively).

3. The length of each CI is divided into hour, day and week bases, we have selected the daily bases (24 hours).
4. The SP and EP of each CI are represented by the DD/MM/YYYY 00:00:00 of today and DD/MM/YYYY 00:00:00 of the next day respectively.
5. The EP of CI_i is represented by the SP of CI_{i+1}.
6. Checkpoint (CP) is the point in time when the resource consumption is computed.
7. Different resources may have different CP for the same CI.
    - The EP of each CI is selected as Nirvanix SDN fixed CP for Bandwidth and Operation.
    - SP of each CI is selected as Nirvanix SDN fixed CP for storage consumption.
8. Nirvanix SDN allowed their customers to download resource the consumption report.

3.4.4.2 Storage Accounting

Nirvanix SDN applies the following accounting model to compute storage consumption (SC):

1. Compute SC for each upload request related to a bucket by:

$$Object_{SC} = Objectsize \quad (1)$$

Where Objectsize = the number of bytes transferred per upload request

2. Compute the deleted SC per delete request by:

$$DelObject_{SC} = Objectsize \quad (2)$$

Where Objectsize = the number of bytes transferred per upload request

3. Compute the SC for each day

$$\mathrm{Day}_{\mathrm{SC_i}} = \mathrm{Day}_{\mathrm{SC_{i-1}}} + \sum_{j=1}^{N} \mathrm{Object}_{\mathrm{SC_j}} - \sum_{k=1}^{M} \mathrm{DelObject}_{\mathrm{SC_k}} \quad (3)$$

Where N and M representing the number of upload and delete requests per day respectively

4. At the end of the month the total SC is computed by the following formula:

$$Month_{SC} = \sum_{i=1}^{n} Day_{SC_i} \quad (4)$$

Where N represents the number of days per month

5. Finally, convert the total SC/Month into GB/month by the following formula:
   = output of (4) x (1 GB / 1,073,741,824 bytes) (5)

3.4.4.3 Bandwidth Accounting

For the RESTful API, Nirvanix SDN applies exactly the same upload and download accounting models used by Amazon S3.

3.4.5 Shortcomings in the Nirvanix SDN Accounting Model

The Nirvanix SDN customer is charged for storage and bandwidth. In the previous subsections we examined whether the data that the service provider 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 because all data required for computing storage and bandwidth can be collected independently by the consumer, however, Nirvanix SDN should provide full documentation that described its accounting model.

Importantly, our experiments show that Nirvanix SDN's accounting model suffered from the following shortcomings:

1. In practical, accounting model documentation, our experiments show that Nirvanix does not published any document about their accounting model. Generally Nirvanix SDN is not well documented.
2. Concerning storage consumption, the accounting model needs explicitly to state how the uploaded data is mapped into consumed bytes in Service Provider. For example, Nirvanix SDN experiments showed that neither user metadata nor file metadata do not impact storage consumption.

3. Related to bandwidth consumption, Nirvanix SDN's accounting model needs explicitly to clarify how bandwidth consumption is computed for all APIs, and whether failed operations are consumed bandwidth consumption or not.
4. Furthermore, Nirvanix SDN should publish in well description document all details related to, how and when resource consumption is computed, and put a solution for any potential source that may affect the consumer's measurements. For instance, propagation delays.

3.4.6 Summary of Nirvanix NDS case study

3.4.6.1 General

The important result obtained from Nirvanix NDS experiments is that the Nirvanix NDS accounting model has more or less the same features as the Amazon S3 accounting model, which implies that Amazon S3 and Nirvanix NDS applied the same policy in building their accounting models. More importantly, similar to the results obtained from Amazon S3, we found from all Nirvanix NDS experiment results that a Nirvanix NDS consumer can independently collect all metering data that is required for calculating the consumption of all Nirvanix NDS resources. Furthermore, the experiment results show that there is no document available that gives any details about the Nirvanix NDS accounting model. We also pointed out that the experiment results show that Nirvanix NDS has selected the end point of each consumption interval as a fixed checkpoint to calculate the resource consumption of all resources. Finally, our results show that it is possible for a Nirvanix NDS consumer to independently implement their own RAS that can be used to compute and verify their resource consumption.

3.4.6.2 Storage

Five conclusions can be drawn from the Nirvanix SDN experiment results: first, the mapping between bytes uploaded by PUT requests and bytes stored in Nirvanix SDN correspond one-to-one; second the user and system metadata do not impact storage consumption; third, an empty folder does not consume storage space; fourth, our observations show that Nirvanix NDS has selected the end point of each CI as their

fixed checkpoint to compute resource consumption. Finally, we have observed that uploading and deleting objects between two checkpoints does not affect storage consumption; however it affects other resource consumption.

3.4.6.3 Operations

Nirvanix SDN does not charge for operations such as Amazon S3.

3.4.6.4 Bandwidth

First of all we need to clarify that we have only used a RESTful interface to execute our experiments; we have found that bandwidth consumption is only used by the file data; finally, similar to the storage consumption outcome, Nirvanix SDN has a fixed checkpoint for Bandwidth consumption; it is equal to the end point of each consumption interval.

3.5 Third case study: Amazon EC2

EC2 is a computation service offered by Amazon as an IaaS [52]. The service offers raw virtual CPUs (also called a Virtual Machine or VM) to subscribers. A subscriber is granted administrative privileges over his VM, which he can exercise by means of sending remote commands to the Amazon Cloud from his desktop computer. For example, he is expected to configure, launch, stop, re-launch, terminate, backup, etc. his VM. In return, the subscriber is free to choose the operating system (e.g. Windows or Linux) and applications to run.

In EC2 terminology, a running virtual CPU is called an instance whereas the frozen bundle of software on disk that contains the libraries, applications and initial configuration settings that are used to launch an instance is called the Amazon Machine Image (AMI).

Currently, Amazon offers six types of instances that differ from each other in four initial configuration parameters that cannot be changed at running time: amount of EC2 compute units that it delivers, size of their memory and local storage (also called

ephemeral and instance storage) and the type of platform (32 or 64 bits). An EC2 compute unit is an Amazon unit and is defined as the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor.

Thus Amazon offer small, large, extra-large and other types of instances. For example, the default instance type is the Small Instance and is a 32 bit platform that delivers 1 EC2 compute unit and provided with 1.7 GB of memory and 160 GB of local storage. These types of instances are offered to subscribers under several billing models: on-demand instances, reserved instances and spot instances. In our discussion we will focus on on-demand instances.

EC2 instances can be physically placed at different locations. Amazon organizes their infrastructure into two availability zones: N. California and N. Virginia are located in the USA; while Ireland and Singapore are located, respectively, in Europe and Asia. Each region is completely independent and contains several availability zones that are used to improve the fault tolerance within the region. We suspect that each availability zone is an isolated data centre which is powered by its own power line. Different availability zones in the same region are placed very close to each other. The region useast-1 has three availability zones, us-east-1a, us-east-1b and us-east-1c. The region eu-west-1 has two availability zones, eu-west-1a and eu-west-1b [52]. These types of instances are offered to subscribers under several billing models: on-demand instances, reserved instances and spot instances. In our discussion we will focus on on-demand instances.

	Irland		US – N. California		US – N. Virginia		APAC – Singapore	
Standard On-Demand Instances	**Linux/UNIX Usage**	**Windows Usage**	**Linux/UNIX Usage**	**Windows Usage**	**Linux/UNIX Usage**	**Windows Usage**	**Linux/UNIX Usage**	**Windows Usage**
Small (Default)	$0.095 / hour	$0.12 / hour	$0.095 / hour	$0.13 / hour	$0.085/ hour	$0.12 / hour	$0.095/ hour	$0.12 / hour
Large	$0.38 / hour	$0.48 / hour	$0.38 / hour	$0.52 / hour	$0.34 / hour	$0.48 / hour	$0.38 / hour	$0.48 / hour
Extra Large	$0.76 / hour	$0.96 / hour	$0.76 / hour	$1.04 / hour	$0.68 / hour	$0.96 / hour	$0.76 / hour	$0.96 / hour
Micro On-Demand Instances								
Micro	$0.025 / hour	$0.035/hour	$0.025 / hour	$0.035/ hour	$0.02 / hour	$0.03 / hour	$0.025/ hour	$0.035/ hour
High-Memory On-Demand Instances								
Extra Large	$0.57 / hour	$0.62 / hour	$0.57 / hour	$0.69 / hour	$0.50 / hour	$0.62 / hour	$0.57 / hour	$0.62 / hour
Double Extra Large	$1.14 / hour	$1.24 / hour	$1.14 / hour	$1.38 / hour	$1.00 / hour	$1.24 / hour	$1.14 / hour	$1.24 / hour
Quadruple Extra Large	$2.28 / hour	$2.48 / hour	$2.28 / hour	$2.76 / hour	$2.00 / hour	$2.48 / hour	$2.28 / hour	$2.48 / hour
High-CPU On-Demand Instances								
Medium	$0.19 / hour	$0.29 / hour	$0.19 / hour	$0.31 / hour	$0.17 / hour	$0.29 / hour	$0.19 / hour	$0.29 / hour
Extra Large	$0.76 / hour	$1.16 / hour	$0.76 / hour	$1.24 / hour	$0.68 / hour	$1.16 / hour	$0.76 / hour	$1.16 / hour
Cluster Compute instances								
Quadruple Extra Large	N/A*	N/A*	N/A*	N/A*	$1.60 / hour	N/A*	N/A*	N/A*
Cluster GPU Instances								
Quadruple Extra Large	N/A*	N/A*	N/A*	N/A*	$2.10 / hour	N/A*	N/A*	N/A*

Table 3. 11 EC2 pricing schema

Under the on-demand billing model, Amazon defines the unit of consumption of an instance as the instance hour (instanceHrs). Currently, the cost of an instance hour of a small instance running Linux or Windows are respectively, 9.5 and 12 cents as shown in Table 3.11. On top of charges for instance hours, instance subscribers normally incur additional charges for data transfer that the instances generates (Data Transfer In and Data Transfer Out) and for additional infrastructure that the instance might need such as disk storage, IP addresses, monitoring facilities and others. As these additional charges are accounted and billed separately, we will leave them out of our discussion and focus only on instance hours charges.

The figures above imply that if a subscriber accrues 10 instanceHrs of small instance consumption, running Linux, during a month, he will incur a charge of 95 cents at the end of the month. In principle, the pricing tables publicly available from Amazon web pages should allow a subscriber to independently conduct his own accounting of EC2 consumption. In the absence of a well-defined accounting model this is not a trivial exercise.

3.5.1 EC2 Accounting Model

EC2 accounting model description is spread over several on-line documents from Amazon. Some insight into the definition of instance hour is provided in the Amazon

EC2 Pricing document [58] (see just below the table of On-demand Instances) where it is stated that Pricing is per instance-hour consumed for each instance, from the time an instance is launched until it is terminated. Each partial instance-hour consumed will be billed as a full hour. This statement suggests that once an instance is launched it will incur at least an instance hours of consumption. For example, if the instance runs continuously for 5 minutes, it will incur 1 instanceHrs; likewise, if the instance runs continuously for 90 minutes, it will incur 2 instanceHrs.

The problem with this definition is that it does not clarify when an instance is considered to be launched and terminated. Additional information about this issue is provided in the Billing section of FAQs [69], Paying for What You Use of the Amazon Elastic Compute (Amazon EC2) document [52] and in the How You're Charged section of the User Guide [72]. For example, in [52] it is stated that each instance will store its actual launch time. Thereafter, each instance will charge for its hours of execution at the beginning of each hour relative to the time it launched.
From information extracted from the documents cited above it is clear that Amazon starts and stops counting instance hours as the instance is driven by the subscriber, through different states. Also, it is clear that Amazon instance hours are accrued from the execution of one or more individual sessions executed by the subscriber during the billing period. Within this context, a session starts and terminates when the subscriber launches and terminates, respectively, an instance.

3.5.1.1 EC2 Accounting Model Description

Session-based accounting models for resources that involve several events and states that incur different consumptions, are conveniently described by Finite State Machines (FSMs). We will use a Finite State Machine (FSM) to describe EC2 accounting model.

States of an instance session: The states that an instance can reach during a session depend on the type of memory used by the AMI to store its boot (also called root) device. Currently, Amazon supports S3-backed and EBS-backed instances. EBS stands for Elastic Block Store and is a persistent storage that can be attached to an instance. The subscriber chooses between S3 or EBS backed instances at AMI creation time.

Unfortunately, the states that an instance can reach during a session are not well documented by Amazon. Yet after a careful examination of Amazon's online documentation we managed to build the FSM shown in Figure. 3.9.

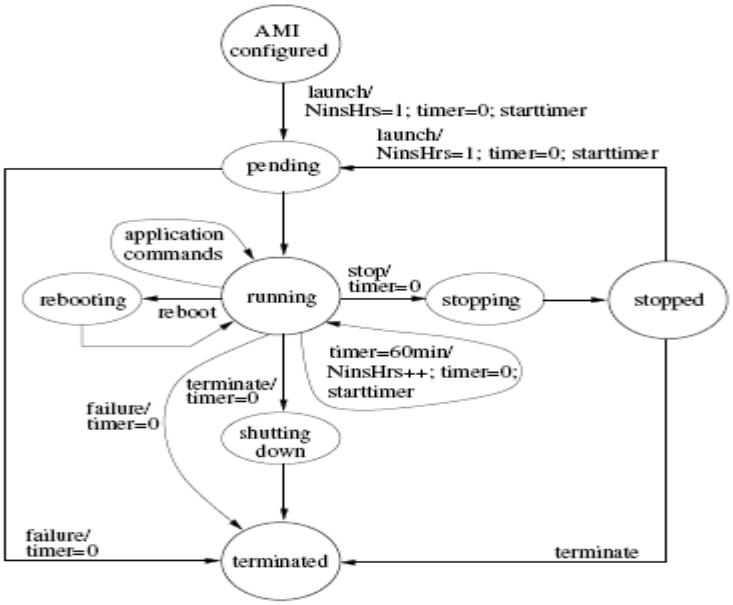


Figure 3. 9 Session of an Amazon instance represented as Finite State Machine.

The figure 3.9 shows that the FSM of an Amazon instance has two types of states: permanent and transient states. **Permanent states** (represented by large circles, e.g. running) can be remotely manipulated by commands issued by the subscriber; once the FSM reaches a permanent state, it remains there until the subscribers issue a command to force the FSM to progress to another state. **Transient states** (represented by small circles, e.g. stopping) are states that the FSM visits temporarily as it progresses from a permanent state into another. The subscriber has no control over the time spent in a transient state; this is why there are no labels on the outgoing arrows of these states.

We have labelled the transitions of the FSM with event/action notations. The event is the cause of the transition whereas the action represents the set (possibly empty) of operations that Amazon executes when the event occurs, to count the numbers of instance hours consumed by the instance.

There are two types of events: subscriber's and internal to the FSM events. The subscriber's events are the commands (launch, application commands, reboot, stop and terminate) that the subscribers issues to operate his instance; likewise, internal events

are events that occur independently from the subscriber's commands, namely, timer = 60min and failure.

AMI configured: is the initial state. It is reached when the subscriber successfully configures his AMI so that it is ready to be launched. **running:** is the state where the instance can perform useful computation for the subscriber, for example, it can respond to application commands issued by the subscriber. **terminated:** is the final state and represents the end of the life cycle of the instance. Once this state is reached the instance is destroyed. To perform additional computation after entering this state the subscriber needs to configure another AMI. The terminated state is reached when the subscribed issues the terminate command, the instance fails when it was in running state or the instance fails to reach running state. **shuttingdown:** is reached when the subscriber issues the reboot or terminate command. **stopped:** this state is supported only EBS-backed instances (S3-backed instances cannot be stopped) and is reached when the user issues stop or terminate commands, say for example, to perform backup duties.

States and instance hours: In the figure, NinstHrs is used to count the number of instance hours consumed by an instance during a single session. The number of instance hours consumed by an instance is determined by the integer value stored in NinstHrs when the instance reaches the terminated state. Timer is Amazon's timer to count 60 minutes intervals; it can be set to zero (timer = 0) and started (starttimer).

In the FSM, the charging operations are executed as suggested by the Amazon's on line documentation. For example, in Paying for What You Use Section of [2], Amazon states that the beginning of an instance hour is relative to the launch time. Consequently, the FSM sets NinstHrs = 1 when the subscriber executes a launch command from the AMI configured state. At the same time, timer is set to zero and started. NinstHrs = 1 indicates that once a subscriber executes a launch command, he will incur at least one instance hour. If the subscriber leaves his instance in the running state for 60 minutes (timer = 60min) the FSM increments NinstHrs by one, sets the timer to zero and starts it again.

From running state the timer is set to zero when the subscriber decides to terminate his instance (terminate command) or when the instance fails (failure event). Although Amazon's documentation does not discuss it, we believe that the possibility of an instance not reaching the running state cannot be ignore, therefore we have included a transition from pending to terminated state; the FSM sets the timer to zero when this abnormal event occurs.

As explained in Basics of Amazon EBS-Backed AMIs and Instances and How You're Charged of [72], a running EBS-backed instance can be stopped by the subscriber by means of the stop command and drive it to the stopped state. As indicated by timer = 0 operation executed when the subscribed issues a stop command, an instance in stopped state incurs no instance hours. However, though it is not shown in the figure as this is a different issue, Amazon charges for EBS storage and other additional services related to the stopped instance. The subscriber can drive an instance from the stopped to the terminated state. Alternatively he can re-launch his instance. In fact, the subscriber can launch, stop and launch his instance as many times as he needs to. However, as indicated by the NinstHrs + + , timer = 0 and starttimer operations over the arrow, every transition from stopped to pending state accrues an instance hours of consumption, irrespectively of the time elapsed between each pair of consecutive launch commands.

3.5.2 EC2 Experiments and Results

To understand EC2's accounting model we have run several experiments; for each experiment the consumer collected metering data about each run and shutdown instance request/response details. The experiment is described as follows below:

3.5.2.1 EC2 Usage Report Experiment

In an attempt to audit our own EC2 bill we studied the EC2 Usage Report, aiming to gain a complete understanding of how Amazon EC2 is representing its accounting data and what is the meaning of each data presented in the Amazon EC2 Usage Report.

- **The aim:** to understand how Amazon EC2's accounting data is represented and the meaning of each item in the accounting data

- **Client Actions:**
    - Client has already created a number of instances and run each instance for particular time.
    - Client downloads Amazon EC2's Usage Report.
- **Observation**

A downloaded EC2 Usage Report is shown in Table 3.12. From the EC2 Usage Report we understand the following:

- The resource consumption is divided into a consumption interval CI, where each CI has a start and end point SP and EP respectively.
- The length of each CI is an hour, day or month. We have selected the hourly CI.
- The start point (SP) of each CI is represented by 00:00:00 GMT of each hour.

Service	Operation	UsageType	StartTime	EndTime	UsageValue
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 17:00	06/01/2010 18:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 18:00	06/01/2010 19:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 19:00	06/01/2010 20:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 20:00	06/01/2010 21:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 21:00	06/01/2010 22:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 22:00	06/01/2010 23:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/01/2010 23:00	06/02/2010 00:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 00:00	06/02/2010 01:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 01:00	06/02/2010 02:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 02:00	06/02/2010 03:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 03:00	06/02/2010 04:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 04:00	06/02/2010 05:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 05:00	06/02/2010 06:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 06:00	06/02/2010 07:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 07:00	06/02/2010 08:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 08:00	06/02/2010 09:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 09:00	06/02/2010 10:00	1
AmazonEC2	RunInstances:0002	BoxUsage	06/02/2010 10:00	06/02/2010 11:00	1

Table 3. 12 Amazon EC2's Usage Report

- The end point (EP) of each CI is represented by 00:00:00 GMT of the next hour.
- The EP of CI_i represents the SP of CI_{i+1}.
- The instance is measured by hourly unit.

i. EC2 Accounting Model Experiment

To verify that the accounting model described by the FSM of Fig. 3-a) matches Amazon's description, we (as subscribers) conducted a series of practical experiments.

In particular, our aim was to verify how the number of instance-hours is counted by Amazon.

The experiments involved 1) configuration of different AMIs; 2) launch of instances; 3) execution of remote commands to drive the instances through the different states shown in the FSM. For example, we configured AMIs, launched and run them for periods of different lengths and terminated them. Likewise, we launched instances and terminated them as soon as they reached the running state. 4) To calculate the number of instance hours consumed by the instances, we recorded the time of execution of the remote commands launch, stop, terminate and reboot, and the time of reaching both transient and permanent states as independent consumer's metering data. Collect metering data from run or shutdown instance request/response. We also collected instance ID and the response status. 5) Download EC2 Usages Report. From the consumer's metering data and EC2's Usage Report we will try to understand how EC2 computes instance consumption.

- **Observation**

  As shown in Table 3.13, the client collected metering data about each instance, where an instance ID represents the ID of the running instance, the running time represents the request time stamp of the client run instance request, the shutdown time represents the time stamp of the client shutdown instance request and usage represents the instance usage, which means how long the instance run for in minutes. In EC2's accounting data, the resource represents the type of resource that the consumer uses, the start and end points represent the CI start and end points and usage represents the value of resource consumption which is computed in hour units. As shown in Table 3.13, the client collected metering data about each run and shutdown instance request/response, where at instance ID represents the ID of the running instance, running time represents the request time stamp of the client-run instance request, shutdown time represents the time stamp of the client shutdown instance request and usage represents the instance usage which means how long the instance runs for in minutes.

No	Instance ID	Lunching time	Pending time	Running time	Terminate Req time & Shuttingdown	Terminate State time	Usages Time	Amazon EC2 Result			
								Type usage	Start Time	End Time	Value
1	i-b734a0d9	13:05:11 T 11:29	13:05:11 T 11:29	13:05:11 T 11:29	13:05:11 T 12:29	13:05:11 T 12:29	60 Minutes	BoxUsage	10:00 13:05:11	11:00 13:05:11	1
2	i-05c72f6	14:05:11 T 16:12	14:05:11 T 16:12	14:05:11 T 16:13	14:05:11 T 16:14	14:05:11 T 16:14	1 Minute	BoxUsage	15:00 14:05:11	16:00 14:05:11	1
3	i-c114fcaf	14:05:11 T 17:57	14:05:11 T 17:57	14:05:11 T 17:58	14:05:11 T 18:53	14:05:11 T 18:53	55 Minutes	BoxUsage	16:00 14:05:11	17:00 14:05:11	1
4	i-3907f257	15:05:11 T 12:06	15:05:11 T 12:06	15:05:11 T 12:07	15:05:11 T 13:05	15:05:11 T 13:06	59 Minutes	BoxUsage	11:00 15:05:11	12:00 15:05:11	1
5	i-db6591b5	15:05:11 T 18:58	15:05:11 T 18:58	15:05:11 T 18:59	15:05:11 T 19:56	15:05:11 T 19:57	58 Minutes	BoxUsage: t1.micro	17:00 15:05:11	18:00 15:05:11	1
6	i-5db44333	15:05:11 T 20:01	15:05:11 T 20:01	15:05:11 T 20:09	15:05:11 T 20:10	15:05:11 T 20:11	1 Minute	BoxUsage: m1.large	19:00 15:05:11	20:00 15:05:11	1
7	i-b5c736db	16:05:11 T 09:53	16:05:11 T 09:53	16:05:11 T 09:59	16:05:11 T 11:07	16:05:11 T 11:09	68 Minutes	BoxUsage: m1.large	08:00 16:05:11	10:00 16:05:11	2
8	i-7904f517	16:05:11 T 11:11	16:05:11 T 11:11	16:05:11 T 11:12	16:05:11 T 11:12	16:05:11 T 11:14	0 Minutes	N/A	N/A	N/A	0
9	i-0b54a565	16:05:11 T 12:13	16:05:11 T 12:13	16:05:11 T 12:14	16:05:11 T 12:14	16:05:11 T 12:15	0 Minute	BoxUsage: m1.large	11:00 16:05:11	12:00 16:05:11	1
10	i-49be4e27	16:05:11 T 13:05	16:05:11 T 13:05	16:05:11 T 13:05	16:05:11 T 13:06	16:05:11 T 13:06	1 Minute	BoxUsage: m1.large	12:00 16:05:11	13:00 16:05:11	1
11	i-bfb0b13	16:05:11 T 14:15	16:05:11 T 14:15	16:05:11 T 14:22	16:05:11 T 14:22	16:05:11 T 14:22	0 Minute	BoxUsage: m1.large	13:00 16:05:11	14:00 16:05:11	1
12	i-8d27d7e3	16:05:11 T 15:43	16:05:11 T 15:43	16:05:11 T 15:49	16:05:11 T 16:57	16:05:11 T 17:00	68 Minutes	BoxUsage: m1.xlarge	14:00 16:05:11	16:00 16:05:11	2
13	i-554a45b	16:05:11 T 17:04	16:05:11 T 17:04	16:05:11 T 17:11	16:05:11 T 18:11	16:05:11 T 18:13	60 Minutes	BoxUsage: m1.large	16:00 16:05:11	17:00 16:05:11	1
14	i-41ed1e2f	16:05:11 T 19:09	16:05:11 T 19:09	16:05:11 T 19:15	16:05:11 T 20:16	16:05:11 T 20:18	61 Minutes	BoxUsage: m1.large	18:00 16:05:11	20:00 16:05:11	2
15	i-9b9466f5	16:05:11 T 22:09	16:05:11 T 22:09	16:05:11 T 22:17	16:05:11 T 23:16	16:05:11 T 23:19	59 Minutes	BoxUsage: m1.large	21:00 16:05:11	22:00 16:05:11	1

Table 3. 13 Client metering and accounting data with EC2 accounting data

In EC2's accounting data the resource represents the type of resource that the consumer is using, the start and end points represent the CI start and end points and usage represents the value of resource consumption which is computed in hourly units. We have found that EC2 does the following:

- As shown in Table 3.13, a comparison of data collected from our experiments against Amazon's data from their usage report reveals that: currently, the beginning of an instance hour is not the execution time of the subscriber's launch command, as documented by Amazon, but the time when the instance reaches the running state. These findings imply that the accounting model currently in use is the one described by the FSM of Figure 3.10.

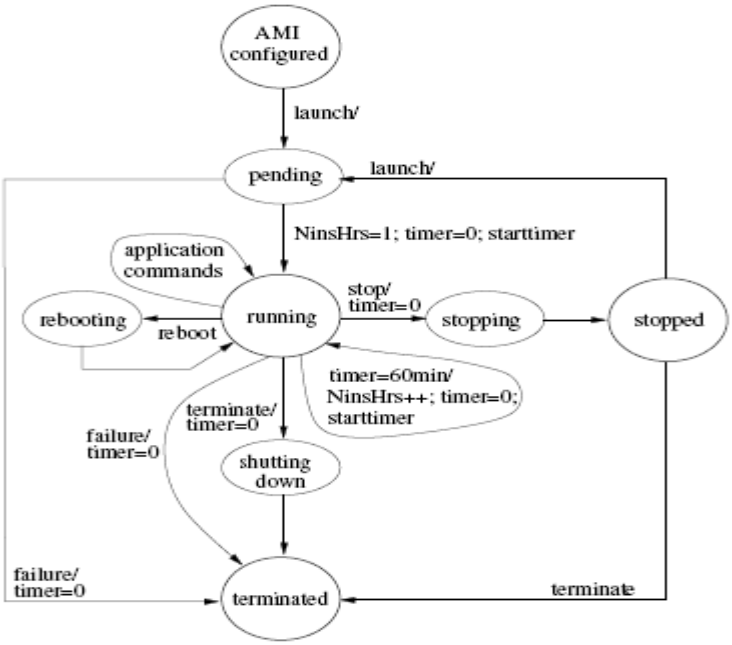


Figure 3. 10 Accurate FSM session representation of an Amazon instance

As shown in the figure, the NinstHrs is incremented when the instance reaches the running state.

- EC2 computes the resource consumption for each instance for the time that the instance reach running state up to the time that the consumer triggers a terminate command. As shown in experiment No. 15 in Table 3.13, where the different between the time of the lunching instance and the time when the consumer issued the terminate request is greater than one hour while in EC2 Usage appear that EC2 charges the customer for just one hour.
- We notice that it is possible that a consumer launch and terminated an instance but EC2 does not charge the consumer about this instance, as shown in experiment No. 8 in Table 3.13.

3.5.3 Shortcomings in the Amazon EC2 accounting model

An Amazon EC2 customer is charged for On-Demand Instances, Bandwidth, Elastic Block Store, Elastic IP Addresses, Cloud Watch and Elastic Load Balancing. We have selected only the On-Demand Instance. In this section we examined whether the data that the Amazon EC2 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 because all data required for computing On-Demand Instances can be collected independently by the consumer. However, our experiments show that Amazon EC2s accounting model suffered from ambiguous and incomplete documentation: our experiments show that EC2 accounting model description is published in different separated documents and these documents do not included all the EC2 accounting model description. Furthermore, some documents related to EC2 model included information that did not match with our finding such as how the resource consumption of on-demand instance is computed.

3.5.4 Summary of Amazon EC2 case study

In this section we present a summary that has been obtained from all experiments conducted on Amazon EC2. First, the Amazon EC2 accounting model has the same general features as Amazon S3 and Nirvanix SDN, where resource consumption is divided into consumption intervals (CI) and each CI has a start and end point, checkpoint, Usage Report, and so on. Second, the minimum charging unit is an hour, and if the consumer runs an instance for just a few minutes then Amazon EC2 will charge him for one hour. Third, we have concentrated only on the CPU resource consumption which is represented by Instance Consumption in EC2. In this regard, we found that all required metering data can be collected locally by the consumer and from the timestamp for run and terminate instance requests, the consumer can easily compute the resource consumption of each instance.

3.6 Comparing Charges

The provision of a full description of the accounting model for each resource can help the consumer who is planning to use a cloud provider (e.g. a storage provider such as Amazon S3) to implement a resource calculator (RC) which can be used to compute their estimated resource consumption based on an estimated workload; consequently, by using the description of the pricing model the consumer can easily compute the charges as well. The idea behind computing resource consumption based on the consumer's estimated workload is to help the consumer to use the result to estimate

their projected budget plan. Furthermore, RC can be used to compare different service providers to find the cheapest provider.

3.6.1 Resource Calculator (RC)

A Resource Calculator (RC) is used to compute the estimated consumption and cost of each resource based on a customer's estimated workload. The RC uses the description of the provider's accounting model to compute the resource consumption, whereas using the charging (pricing) model computes the charge of each resource. In this section, we will present an overview of how the RC is implemented. We have used Java programming language to develop the RC which does the computation for both resource and cost. To illustrate the RC implementation, we have selected Amazon S3 accounting and charging models as examples for the development of RC.
An estimate of the consumer's workload is the input for this programme and the output is the cost and resource consumption of storage, bandwidth and operations.

3.6.1.1 Estimate resource consumption and cost

In this section, we demonstrate how the Amazon S3 accounting and charging models implementation was used to estimate resource consumption and cost based on estimated workload. The estimated workload described in Table 3.14 was used as an input to the programme. The program produced the consumption and charges of each resource as output. The charge and the resource consumption of storage are described in Table 3.15 whereas the charge and resource consumption of operation, upload and download bandwidth are described in Table 3.16.

No.	Description of work load	Value
1	Average Length of Folders Name	20
2	Average Length of Files Name	10
3	Average Number of PUT Requests/Day	1
4	Average Number of Bytes Transferred/PUT Request	1 GB
5	Average Number of Delete Requests/Day	0

Table 3. 14 Customer's estimated workload

Table 3.15 shows the resource consumption and charges of storage produced by the programme based on the consumer's workload described in Table 3.14.

	AMAZON S3	
	STORAGE CONSUMPTION	**COST**
1st MONTH	15.00000041909516 GB/MONTH	2.250000062864274 $/MONTH
2nd MONTH	44.50000124331564 GB/MONTH	6.675000186497345 $/MONTH
3rd MONTH	74.00000206753612 GB/MONTH	11.100000310130417 $/MONTH
4th MONTH	104.50000291969627 GB/MONTH	15.67500043795444 $/MONTH
5th MONTH	135.00000377185643 GB/MONTH	20.250000565778464 $/MONTH
6th MONTH	165.50000462401658 GB/MONTH	24.825000693602487 $/MONTH
7th MONTH	196.00000547617677 GB/MONTH	29.400000821426513 $/MONTH
8th MONTH	227.00000634230676 GB/MONTH	34.050000951346014 $/MONTH
9th MONTH	257.5000071944669 GB/MONTH	38.62500107917003 $/MONTH
10th MONTH	288.00000804662704 GB/MONTH	43.20000120699405 $/MONTH
11th MONTH	318.5000088987872 GB/MONTH	47.77500133481808 $/MONTH
12th MONTH	349.00000975094736 GB/MONTH	52.350001462642105 $/MONTH
TOTAL STORAGE CHARGE PER YEAR		**326.17500911322423 $/YEAR**

Table 3. 15 Storage consumption and cost

	AMAZON S3	
	DATA – TRANSFER IN	**COST**
PER MONTH	031 GB/MONTH	03.1 $/MONTH
PER YEAR	365 GB/MONTH	**36.5 $/YEAR**
	TIRE 1 REQUESTS/MONTH	**COST**
PER MONTH	131.0 REQUESTS/MONTH	0.00131 $/MONTH
PER YEAR	465.0 REQUESTS/YEAR	**0.00465 $/YEAR**
	TIRE 2 REQUESTS/MONTH	**COST**
PER MONTH	0 REQUESTS/MONTH	0 $/MONTH
PER YEAR	0 REQUESTS/YEAR	**0 $/YEAR**
TOTAL CHARGE PER YEAY		**362.6796591$/YEAR**

Table 3. 16 Bandwidth and operation consumption and cost

Table 3.16 shows the resource consumption and the cost of bandwidth and operation produced by the programme based on the consumer's estimated workload, as described above.

3.6.1.2 Which is the cheapest cloud provider?

The availability of fully described accounting and charging models of different service providers help the consumer (or Third Party) to build different RCs that can be used to compare different service providers to find the cheapest provider based on the same workload. In this section we demonstrate how the Amazon S3 and Nirvanix SDN

accounting and charging models were compared, in order to find the cheapest provider using the same estimated workload which is described in Table 3.13.

i. Storage Consumption and Cost

As shown in Table 3.17 and Figure 3.11, in terms of resource consumption we found that the Nirvanix and Amazon S3 customer had almost the same amount of storage consumption (GB/MONTH). This is because of the similarity of their accounting model. On the other hand, in terms of cost we found that Amazon S3 was cheaper than Nirvanix.

	AMAZON S3		NIRVANIX	
	STORAGE CONSUMPTION	COST	STORAGE CON	COST
1st MONTH	15.00000041909516 GB/MONTH	2.250000062864274 $/MONTH	15.0 GB/MONTH	03.750 $/MONTH
2nd MONTH	44.50000124331564 GB/MONTH	6.675000186497345 $/MONTH	44.5 GB/MONTH	11.125 $/MONTH
3rd MONTH	74.00000206753612 GB/MONTH	11.100000310130417 $/MONTH	74.0 GB/MONTH	18.500 $/MONTH
4th MONTH	104.50000291969627 GB/MONTH	15.67500043795444 $/MONTH	104.5 GB/MONTH	26.125 $/MONTH
5th MONTH	135.00000377185643 GB/MONTH	20.250000565778464 $/MONTH	135.0 GB/MONTH	33.750 $/MONTH
6th MONTH	165.50000462401658 GB/MONTH	24.825000693602487 $/MONTH	165.5 GB/MONTH	41.375 $/MONTH
7th MONTH	196.00000547617677 GB/MONTH	29.400000821426513 $/MONTH	196.0 GB/MONTH	49.000 $/MONTH
8th MONTH	227.00000634230676 GB/MONTH	34.050000951346014 $/MONTH	227.0 GB/MONTH	56.750 $/MONTH
9th MONTH	257.5000071944669 GB/MONTH	38.62500107917003 $/MONTH	257.5 GB/MONTH	64.375 $/MONTH
10th MONTH	288.00000804662704 GB/MONTH	43.20000120699405 $/MONTH	288.0 GB/MONTH	72.000 $/MONTH
11th MONTH	318.5000088987872 GB/MONTH	47.77500133481808 $/MONTH	318.5 GB/MONTH	79.625 $/MONTH
12th MONTH	349.00000975094736 GB/MONTH	52.350001462642105 $/MONTH	349.0 GB/MONTH	87.625 $/MONTH
TOTAL STORAGE CHARGE PER YEAR		**326.17500911322423 $/YEAR**		**543.625 $/YEAR**

Table 3. 17 Amazon S3 and Nirvanix storage consumption and cost

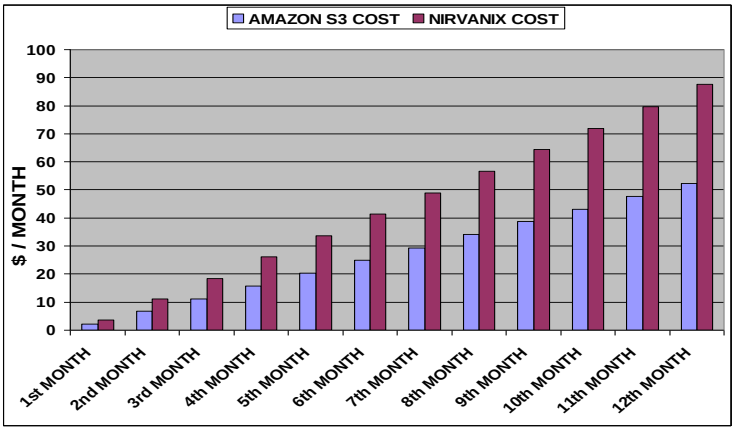


Figure 3. 11 Amazon S3 and Nirvanix storage costs

ii. Bandwidth and Operation Consumption and Cost

Table 18 shows that for the same estimated workload described in Table 3.18, we found that the Nirvanix and Amazon S3 customer consumed exactly the same amount of upload bandwidth consumption (GB/MONTH) and both providers charged the consumer the same amount of money. This is because of the similarity of their accounting and, more importantly, for the first 10 TB Amazon S3 uses the same charging model as Nirvanix (for the first 2TB) for downloads bandwidth consumption. Moreover, the same result was obtained for upload bandwidth consumption. However, Amazon S3 charged for operation (computation power) while Nirvanix SDN does not; however, the charge for operation was insignificant (because the consumer made only a small number of requests).

	AMAZON S3		**NIRVANIX**	
	DATA – TRANSFER IN	**COST**	**DATA – TRANSFER - IN**	**COST**
PER MONTH	031 GB/MONTH	03.1 $/MONTH	31.0 GB/MONTH	3.1 $/MONTH
PER YEAR	365 GB/MONTH	**36.5 $/YEAR**	365 GB/MONTH	**36.5 $/YEAR**
	TIRE 1 REQUESTS/MONTH	**COST**	**TIRE 1 REQUESTS/MONTH**	**COST**
PER MONTH	131.0 REQUESTS/MONTH	0.00131 $/MONTH	0 REQUESTS/MONTH	0 $/MONTH
PER YEAR	465.0 REQUESTS/YEAR	**0.00465 $/YEAR**	0 REQUESTS/YEAR	**0 $/YEAR**
	TIRE 2 REQUESTS/MONTH	**COST**	**TIRE 2 REQUESTS/MONTH**	**COST**
PER MONTH	0 REQUESTS/MONTH	0 $/MONTH	0 REQUESTS/MONTH	0 $/MONTH
PER YEAR	0 REQUESTS/YEAR	**0 $/YEAR**	0 REQUESTS/YEAR	**0 $/YEAR**
TOTAL CHARGE PER YEAY		**36.50465 $/YEAR**		**36.5 $/YEAR**

Table 3. 18 Consumption and cost of upload bandwidth and operation

iii. Summary of Compared Resource Consumption and Cost

On conclusion, for the same workload, we found that the resources consumed in Amazon S3 were almost the same as Nirvanix; however, Nirvanix SDN was nearly equal, being 1.66667 less expensive than Amazon S3. The consumer or a third party can use the description of the provider's accounting and charging models to develop a service (a Resource Calculator) that can be used to estimate the consumption and cost of each resource, and it can also be used to compare between different providers to find the cheapest providers.

3.7 Summary

In order to investigate the visibility of consumer-side accounting, the accounting model of a given cloud infrastructure services of Amazon Simple Storage Service S3, Nirvanix and Amazon Elastic Cloud Computing EC2 were evaluated. It is of course necessary that consumers are provided with an unambiguous resource accounting model that precisely describes all the constituent chargeable resources of a service and how billing charges are calculated from resource usage (resource consumption) data collected on behalf of the consumer over a given period. We pointed out several cases where an accounting model specification was ambiguous or not complete. For example, regarding bandwidth consumption, it is not clear from the available information what constitutes the size of a message. It is only through experiments that we worked out that for RESTful operations, only the size of the object is taken into account and system and user metadata is not part of the message size, whereas for SOAP operations, the total size of the message is taken into account. Failure handling is another area where there is lack of information and/or clarity. For example, concerning EC2, it is not clear how instances that fail accrue instance hours. On the whole, for IaaS services, consumer-side accounting appears quite feasible if, a full description of accounting model is made public and all data required for compute resource consumption can be collected independently by the consumer or TTP.

Chapter 4

Consumer Side Resource Accounting

4.1 Introduction

In the previous chapter we have shown that consumers can independently compute their resource consumption charges if; 1) all the required metering data to compute the resource consumption can be collected independently by the consumer; and 2) a full description of the provider accounting model is made available. In this chapter we discuss possible causes that may lead to conflict between metering data collected by consumers and providers and present possible solutions for avoiding such conflicts. Next, we present Consumer-Centric Models, suggest a systematic way of constructing and specifying consumer-centric resource accounting models and use it to describe and evaluate Nirvanix, S3 and EC3 and Elastic Storage Block (ESB) accounting models. Finally, a summary of this chapter is presented.

4.2 Potential Sources of Conflict

Naturally, different metering data produce different resource consumption figures. We anticipate that there could be several reasons which lead the consumer and the provider to use different metering data to compute resource consumption for the same consumption interval. We will discuss how factors such as network latency, different checkpoints, operation latencies, ambiguities in the description of accounting models, and the use of different measurement processes can cause mismatches between the figures computed by consumer and provider.

4.2.1 Network latency

As we stated earlier in Chapter 3, resource consumption is divided into consumption intervals (CI), and since the consumer and the provider are geographically distributed, it

is possible they are in different time-zones, and may potentially use different time coordinates. Generally, this situation can arise for two reasons, firstly, when the provider does not offer precise information to the consumer about when to start and end a given consumption interval, secondly, when the consumer's and the provider's clocks are in different time zones. However, in practice we have found that most service providers precisely stated the time zone they are using. For instance, from the Authenticating SOAP Requests Section of the Amazon Developer Guide [37] it is clear that S3 servers are synchronised to Universal Time Coordinated (UTC) which is also known as Zulu Time (Z time) and is in practice equivalent to Greenwich Mean Time (GMT).

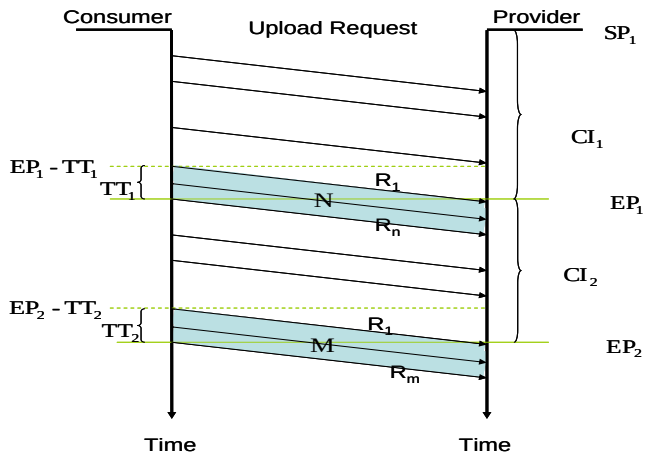


Figure 4. 1 impact of network latency in consumer's and provider's measurements.

However, even when both of them (the consumer and the provider) use the same time zone and the same consumption interval, still the network latency (message transmission time, TT) can contribute to discrepancies. In practical applications, TT is normally of the order of 100 milliseconds. In Figure 4.1, TT represents the average transmission time. As shown graphically, this parameter can cause divergences between the consumer's and provider's accounting results for a given consumption interval. For the sake of simplicity, we will assume that the consumer and provider's start point (SP) and end point (EP) of a given CI are synchronised. Under this assumption, convergence between the consumer's and provider's accounting records can be achieved by compensating the provider's results by the amount of resources consumed by the

requests in the wire, that is, requests issued in a given interval but received and counted in the following interval due to TT.

Let us take an arbitrary interval CI_i. The consumer can calculate its resource consumption RC_i by the mathematical formula described by the accounting model. However, to compensate for *TT*, the provider would need to use the following equation.

$$RC_p = \sum | RC_i + N - M | \quad (1)$$

Where *N* is the amount of resources consumed by requests issued and counted by the consumer in interval CI_i but received and counted by the provider in interval CI_{i+1} due to the effect of TT, in Figure 4.1 this time gap is shown as TT_1. Similarly, *M* is the amount of resources consumed by requests issued and counted by the consumer in interval CI_{i-1} but to be received and counted by the provider in interval CI_i, due to TT; in Figure 4.1, this time gap is shown as TT_2. Both *N* and *M* can be calculated by a formula described by the accounting model. Notice that for the first interval N is to be taken as N = 0.

The above approach requires estimating TT. A better solution is to make use of message timestamps to determine the consumption interval the message belongs to. It is necessary for the consumer and provider both to agree to use the timestamp (e.g. response). This is further discussed in section 4.2.3.

4.2.2 Different checkpoints

Generally, cloud computing providers may or may not have selected a fixed CP to compute the resource consumption for each CI as discussed in Chapter 3. For example, Nervanix SDN has selected the EP of each CI as a fixed CP to compute resource consumption (storage and bandwidth). However, other providers such as Amazon S3 arbitrarily select a CP to compute the storage consumption for the entity CI, where the CP could be any point located between the SP and the EP of that CI. Therefore, different CPs may cause conflicts between the consumer and the provider measurements.

To illustrate this, let us take storage as the selected resource and Amazon S3 as the provider in our example which is shown in Figure 4.2. In the Figure below, the CP_{30}: 2GB indicates 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 Amazon S3. SC stands for Storage Consumption and is explained below.

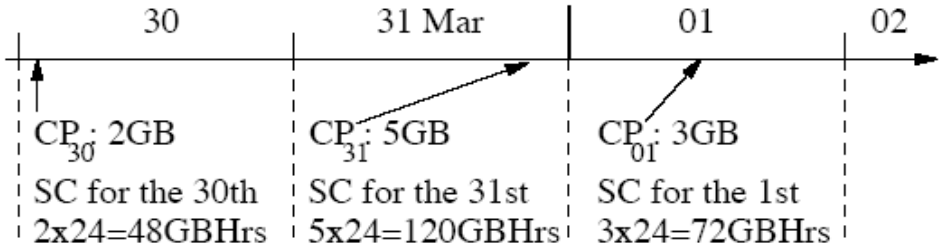


Figure 4. 2 Amazon S3's checkpoint

As shown in Figure 4.2, Amazon S3 uses the results produced by a CP for a given day, to generate a customer account for 24 hrs, regardless of the operations that the customer might perform during the time left between the CP and the 23:59:59 GMT hours left in the day. For example, the SC for the 30th will be taken as 2 × 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 at which the checkpoints are conducted is shown in Figure 4.3 below.

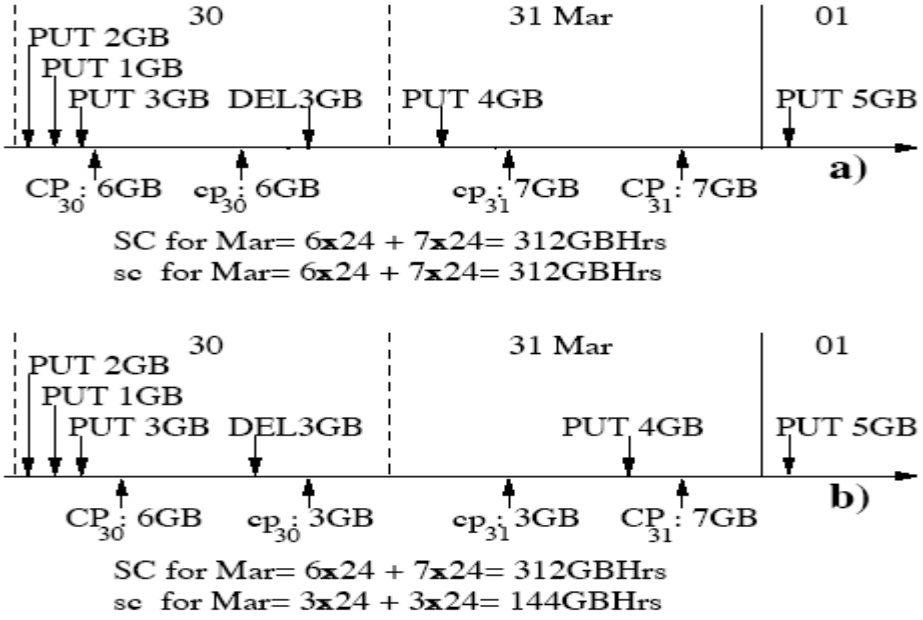


Figure 4. 3 The impact of checkpoints on storage accountability

Figure 4.3 shows the execution time of four PUT and one DEL operations executed by the 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 account. The Figure also shows the specific points in time when checkpoints are conducted independently by two parties, namely, Amazon S3 and a customer. Thus, CP and cp represent, respectively, Amazon S3's and the customer's checkpoints; the gigabytes shown next to CP and cp indicate the storage consumption detected by the checkpoint. For example, on the 30th, the provider conducted its checkpoint at about 5 am 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 3 GB stored (cp_{30}: 6GB). SC and sc represent, respectively, the storage consumption for the month of March, calculated by the provider and by the customer, based on their checkpoints.

Figure 4.3 demonstrates that storage consumption as calculated by Amazon S3 and by the 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 operations are executed within the pair of checkpoints conducted on the 30^{th} or 31^{st}. The result is that both parties calculate equal storage consumptions. In contrast, b) shows the worst 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.

Ideally, Amazon's checkpoint times should be made known to consumers 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 consumer 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).

4.2.3 Impact of operation latency

In the previous discussion concerning the calculation of GBHrs (illustrated using Figure 4.3), we have implicitly assumed that the execution of a PUT (respectively a DELETE) operation is an atomic event whose time of occurrence is either less or greater than the checkpoint time (i.e., the operation happens either before or after the CP). This allowed us to say that if the CP time used at the provider is known to the consumer, then the consumer can match the GBHrs figures of the provider. However, this assumption is over simplifying the distributed nature of the PUT (respectively a DELETE) operation.

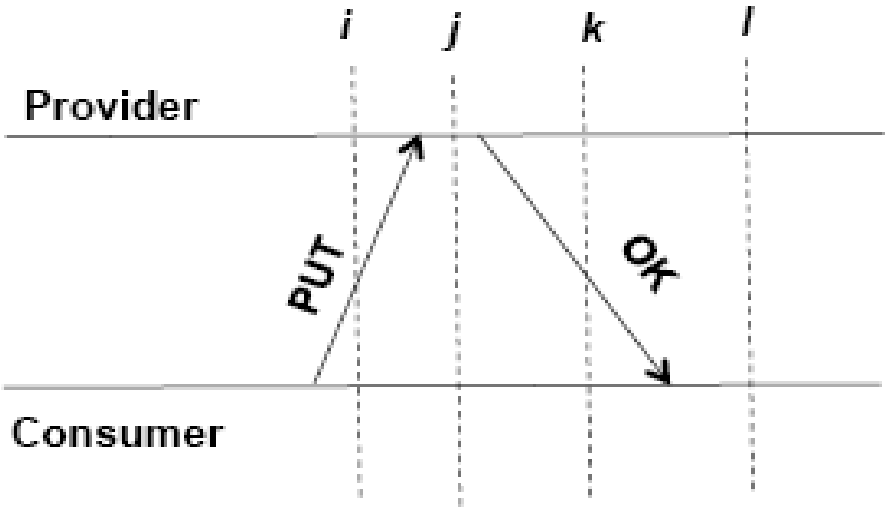


Figure 4. 4 Network and operation latencies

In Figure 4.4 we explicitly show operation execution latencies for a given operation, say PUT; also, i, j, k and l are provider side checkpoint times used for illustration. Let us assume that at the provider side, only the completed operations are taken into account for the calculation of GBHrs; so a checkpoint taken at time i or j will not include the PUT operation (PUT has not yet completed), whereas a checkpoint taken at time k or l will. What happens at the consumer side will depend on which event (sending of the request or reception of the response) is taken to represent the occurrence of PUT. If the timestamp of the request message (PUT) is regarded as the time of occurrence of PUT, then the consumer side GBHrs calculation for a checkpoint at time i or j will include the PUT operation, a discrepancy since the provider did not. On the

other hand, if the timestamp of the response message is regarded as the time of occurrence of PUT, then a checkpoint at time k will not include the PUT operation (whereas the provider has), again a discrepancy. In short, for the operations that occur 'sufficiently close' to the checkpoint time, there is no guarantee that they get ordered identically at both sides with respect to the checkpoint time.

Also, for another resource such as operations (number of requests executed issued and executed at a well-defined period of time, e.g. consumption interval), as we stated earlier, it is straightforward for a consumer to count the type and number of operations performed on S3. There is a potential for discrepancy caused by network latency: operations that are invoked 'sufficiently close' to the end of an accounting period (say i) and counted by the consumer for that period, might get counted as being performed in the next period (say j) by the provider if due to the latency, these invocation messages arrive in period j. This leads to the accumulated charges for the two periods not being the same. This is actually not an issue, as Amazon S3 uses the timestamp of the invocation message for resolution, so the consumer can match the provider's figure.

4.2.4 Ambiguous description of accounting models

Based on our experiments' results described in the previous chapter, we found that there are many examples which can be used to demonstrate how the ambiguous description of the accounting model may lead to discrepancies between the consumer and the provider. However, we will use only two cases to explain this situation. For the first case we have selected the EC2 accounting model for the on-demand instance consumption as an example to illustrate the problem.

As described in the previous chapter, the mismatch between Amazon EC2's documented accounting model and the one currently in use by Amazon EC2 as shown in Figure 4.5 (a and b, respectively) might result in discrepancies between the subscriber's and Amazon EC2's calculations of instance hours. Where, EC2 stated that each instance stores its actual launch time [52]. Thereafter, each instance will charge for its hours of execution at the beginning of each hour relative to the time it launched as shown in Figure 4.5-a. However, our experiment which was described in chapter 3

showed that each instance will charge for its hours of execution at the beginning of each hour relative to the time when the instance reached the running stat as shown in Figure 4.5-b.

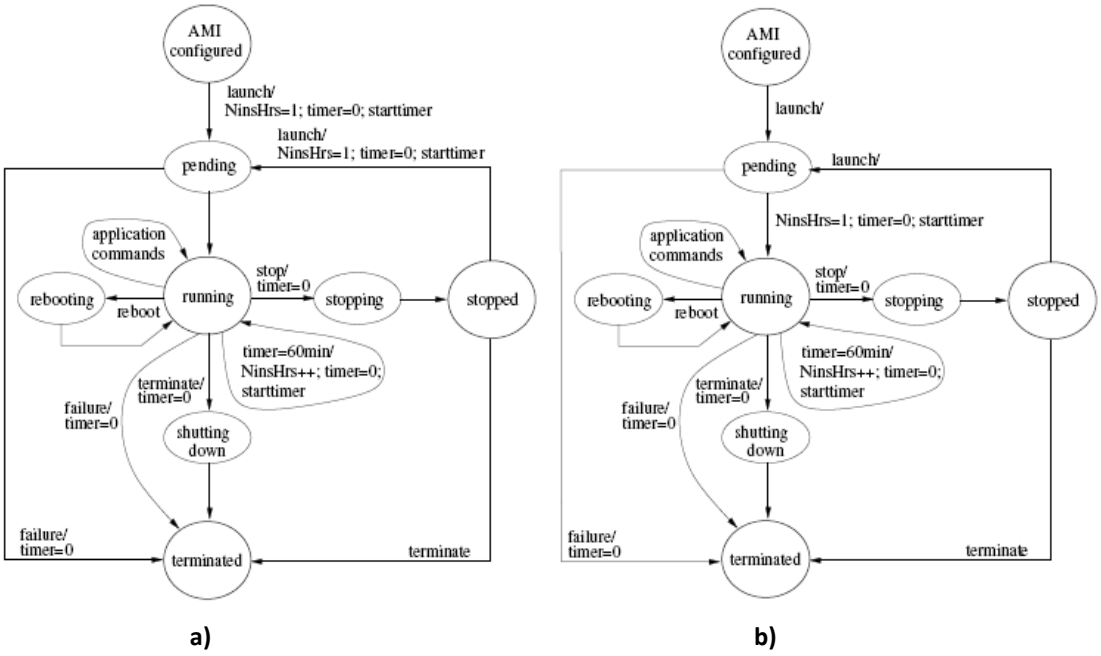


Figure 4. 5 Session of an Amazon instance represented as FSM.

For example, let us assume that it takes five minutes to reach the running state. Now imagine that the subscriber launches an instance, leaves it running for 57 minutes and then terminates it. The subscriber's NinstHours will be equal to two: NinstHrs = 1 at launch time and then NinstHrs is incremented when the timer = 60min. In contrast, to the subscriber's satisfaction, AmazonEC2's usage records will show only one instance hour of consumption. One can argue that this discrepancy is not of the subscriber's concern since, economically, it always favours him.

More challenging and closer to the subscriber's concern are discrepancies caused by failures. Amazon EC2's documentation does not stipulate how instances that fail accrue instance hours. For example, examine Figure 4.5-b and imagine that an instance suddenly crashes after spending 2 hrs and 15 min in a running state. It is not clear to us whether Amazon EC2 will charge for the last 15 min of the execution as a whole instance hour. As a second example, imagine that after being launched either from AMI configured or stopped states, an instance progresses to a pending state and from there,

due to a failure, to terminate. It is not clear to us if Amazon EC2 will charge for the last instance hour counted by NinstHrs.

We believe that, apart from these omissions about failure situations, the accounting model of Figure 4.5-b can be implemented and used by the subscriber to produce accurate accounting. A salient feature of this model is that all the events (launch, stop and terminate) that impact the NinstHrs counter are generated by the subscriber. The only exception is if the timer = 60min event, but can be visible to the subscriber if he synchronises his clock to UTC time.

The accounting model that Amazon EC2 actually uses (Figure 4.5-a) is not impacted by failures of instances to reach a running state because in this model, NinsHrs is incremented when the instance reaches a running state. However, this model is harder for the subscriber to implement since the event that causes the instance to progress from a pending to running state is not under the subscriber's control.

For the second case, we think one likely source of difficulty about the charges for operations and bandwidth is determining the liable party for failed operations. For example, currently, this decision is taken unilaterally by the provider. In this regard, we anticipate two potential sources of conflict: DNS and propagation delays, as explained by Amazon S3, are some requests which 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 [38]. Amazon's advice is to design applications that can handle redirect errors, for example, by resending a request after receiving a 307 code (see [37], Request Routing section). Strictly speaking, these errors are not caused by the customer as the 307 code suggests. It is not clear to the consumer who bears the cost of the re-tried operations.

4.2.5 The use of different measurement processes

Differences can arise at the accounting level between the two sides' measurements due to the different calculation techniques, used to produce accounting data which relates to the different data collecting techniques used by the consumer and the provider. For example, the calculation of GBHrs serves as a good example. We expect that for a checkpoint, the provider will directly measure the storage space actually occupied, whereas, for a given checkpoint time, the consumer will mimic the process by adding (for PUT) and subtracting (for DELETE) to calculate the space, and as we discussed with respect to figure 4.4, discrepancies are possible.

4.2.6 Other reasons

The Internet is not 100% reliable, which may lead to lost, drop or corrupt messages. Furthermore, the metering services are applications (software) which might fail at any time and take some time to recover. Consequently, during this failure there are three possible scenarios which may occur:

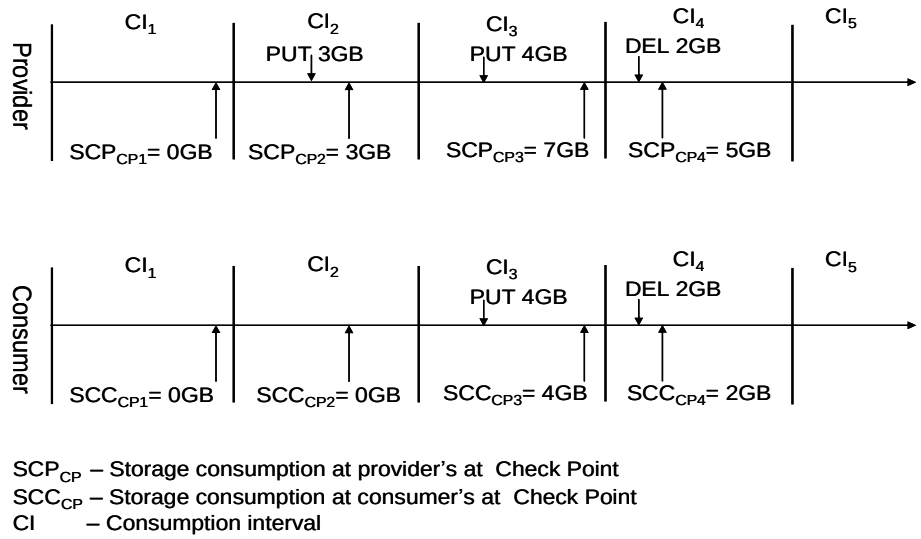
1. MS misses collecting metering data about an upload request but collects metering data about response. OR
2. MS collects metering data about upload requests and misses to collect the metering data of the response. OR
3. MS misses collecting any metering data about the upload request and the response.

Regarding storage consumption; at the consumer's side all three scenarios may affect the storage consumption measurement because some of the required data that is used to compute the storage consumption will not be collected and missed as in scenario 1 and 2 whereas no metering data will be collected in scenario 3.

This may cause a huge dispute in the long run for accumulative resource consumption. Where accumulative resource means that the resource consumption in CI_{i+1} = resource consumption consumed at CP of CI_i + resource consumption consumed at CP of CI_{i+1}. Storage in per-pay-use models is a good example of an accumulative resource because the value of Storage Consumption (SC) at CI_2 = SC at CI_1 + SC at CI_2. Whereas, un-

accumulative resource consumption means that the resource consumption of each consumption interval does not affect the resource consumption of any consumption interval. Bandwidth is a good example of un-accumulative resource consumption, where the bandwidth consumption is equals to the total number of bytes transferred during any CI.

To illustrate the problem, we assume that the consumer's and provider's clocks are synchronised and are using a fixed checkpoint (CP) to compute storage consumption. In Figure 4.6, the CI stand represents the consumption interval, whereas the SCP and SCC stands are for provider and consumer storage consumption respectively, the CP stand is for checkpoints.



SCP_{CP} – Storage consumption at provider's at Check Point
SCC_{CP} – Storage consumption at consumer's at Check Point
CI – Consumption interval

Figure 4. 6 Impact of accumulated resource and consumption intervals.

Figure 4.6 shows that at the 1st consumption interval (CI_1) the consumer and the provider at the 1st CP1 have produced the same storage consumption SCP_{CP1}=SCC_{CP1}= 0 GB, this is because both of the consumer's and the provider's metering services did not record any metering data before the consumer's and the provider's 1st checkpoint (CP1). However, in the 2nd consumption interval CI_2 the consumer and the provider produced different storage consumptions where the provider produced 3 GB and the consumer produced 0 GB at the CP2. This is due to the provider's metering service (MS_P) collecting metering data about a successful 3 GB upload request whereas for some reason (as discussed at the beginning of this section), the consumer's metering

service (MS_C) missed collecting the metering data about the request, the response or both of them (as described by scenarios 1, 2 and 3) about the same 3GB upload operation. As a result of this, the consumer's accounting service AS_C did not count the storage consumption of the 3GB upload operation because AS_C did not find all or some of the required metering data.

We recommended the following steps at the consumer side to deal with such problems. When the AS_C finds metering data for an upload request without a response or response without an upload request, in the first case AS_C can easily verify whether this request eventually succeed or not by executing any operation that does not affect storage consumption such as downloading the upload file, if the AS_C obtained a successful response then it should consider this request as a successful request and include it when computing the storage consumption of the entity request.

On the other hand, if the MS_C missed recording the metering data completely (no record of request or response message, it is not easily solved by the consumer's accounting side because the consumer did not have any information which can be used to track this request. Furthermore, the discrepancy in the measurements between the consumer and the provider will be carried for all of the following CI. Mutual intervention will be necessary.

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. The availability of such information will empower consumers in several ways, such as:

1) Selecting a suitable service provider; 2) making the billing for their applications clear; 3) planning the organization's budgets for IT billing; 4) creating a third party brokering service that automates resource provision in line with the customer's needs.

Clearly, implementing any of the above functionalities will require consumers to 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? However, provider-side accountability is the norm for 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. Furthermore, 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 is made available by the provider as trustworthy.

4.3 Consumer-Centric Models

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.

Based on 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 by 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. The ***strong 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.

4.3.1 Abstract resource

In this section we suggest a systematic way of describing resource accounting models so that they can be understood and reasoned about by consumers. The key idea is very simple: first, define a set of "elementary" chargeable resources (e.g. storage, bandwidth, etc); second, describe the overall resource consumption of a given resource/service in terms of an aggregation of the consumption of these elementary resources. With this view in mind, we present the resource consumption model of an abstract resource. Next we will argue that with some small resource specific variations, the accounting models of resources such as Nirvanix, S3, EC2, EBS and other infrastructure level resources can be represented as special cases of the abstract resource accounting model, and therefore can be understood and reasoned about in a uniform manner.

We consider a typical configuration where a server (cloud) resource and a consumer resource interact with each other by means of requests/responses (req/res) sent through a communication channel (see Figure. 4.7).

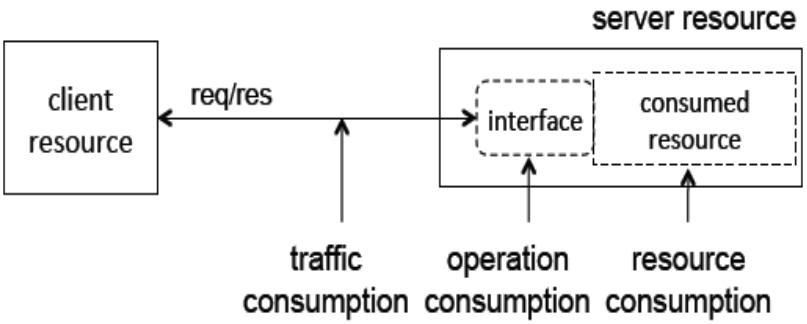


Figure 4. 7 Accounting model of an abstract resource

As shown in the figure, the client resource uses the interface of the server resource to place its requests and collect the corresponding responses. This deployment incurs three

types of consumption charges: **traffic consumption, operation consumption** and **resource consumption**.

Traffic consumption represents the amount of traffic (for example in MBytes) generated by the requests and responses on the communication channel. Operation consumption captures the activities generated by the client resource on the interface such as the number of requests (also called operations) and the number of responses produced. Finally, resource consumption represents the actual consumption of the resource measured in units that depend on the specific nature of the resource, for example, in units of volume (for example, MBytes), time or a combination of them (for example, MBytesHours).

As the figure suggests, the accounting model for a given resource is an aggregation of three elementary models: a model for traffic consumption, a model for operation consumption and a model for resource consumption. These elementary models operate independently of each other, thus they can be specified and examined separately. In particular, a provider should ensure that each of the three elementary models are consumer-centric. This should be done by paying attention to the causes (identified at the beginning of this section) that could lead to discrepancies between the data collected by consumers and providers.

4.3.2 Another Look at Nirvanix, Amazon S3 and EC2

The accounting models of Nirvanix, S3 and EC2 easily map to that of the abstract resource and permit us to analyse them (from the point of view of consumer–centricity) in a succinct manner. Below we will briefly discuss whether the current accounting models of Nirvanix, S3 and EC2 (on-demand instance offered by) are strongly or weakly consumer-centric.

Related to Nirvanix (charges only for bandwidth and resource consumption), we can say that the models of traffic and resource consumption are strongly consumer-centric, but suffer from incompleteness and ambiguities. In practice, Nirvanix does not publish any documents describing its accounting model. Also, the storage accounting model

needs to state explicitly how the uploaded data is mapped into consumed bytes. For example, does user metadata or file metadata impact storage consumption? Related to bandwidth consumption, Nirvanix SDN's accounting model needs to clarify explicitly how bandwidth consumption is computed for all APIs, and whether failed operations consume bandwidth or not.

Concerning S3, we can say that the models of the two elementary resources for traffic consumption and operation consumption are strongly consumer-centric, but suffer from incompleteness and ambiguities. For instance, there is a source of difficulty regarding the charges for operations. In particular, how to determine the liable party for failed operations due to temporary routing errors which are caused by the use of alternative DNS names and request redirection techniques [38]. Strictly speaking these errors are not caused by the customer as the 307 code suggests. It should be clearly stated by Amazon S3 accounting model which party bears the cost of the re–tried operations. In relation to bandwidth, DataTransfer-In and DataTransfer-Out include, respectively, request and response overheads. The difficulty here is that from the Amazon S3 accounting model, it is not clear how message size is calculated in DataTransfer-In and DataTransfer-Out. The resource consumption accounting model is weakly consumer-centric (a checkpointing event which is not observable), making the overall model weakly consumer-centric.

The accounting model of EC2 for the traffic consumption is strongly consumer-centric and the operation consumption is free of charge. However, the accounting model suffers from ambiguity and incompleteness. For example, the EC2 accounting model description is published in different documents and none of these documents include the entire EC2 accounting model description. Overall, the model is weakly consumer-centric because, as we explained in chapter 3, the event that causes a virtual machine instance to progress from pending to a running state is not visible to the consumer.

4.3.3 Elastic Block Storage

EBSs are persistent block storage volumes frequently used for building file systems and databases. They support two interfaces: a Web service interface and a block–based

input/output interface. The Web service interface can be used by the client to issue (for example, from his desktop application) administration operations, such as create volume, delete volume, attach volume, detach volume, etc. The block-based input/ output interface can be used by EC2 VMIs and becomes available upon attaching the EBS to the VMI. Amazon offers EBSs volumes of 1GB to 1 TB. Upon request, EBSs can be allocated to a client and can be attached to VMIs. The storage space of an EBS becomes available when the clients creates the volume and is released when it is explicitly deleted by the client. During this time period, the EBS can be attached and detached several times and to different VMIs but only to one at a time.

The accounting model for EBS is shown in Figure 4.8. Omitted from the figure is the communication channel that the client uses to issue administration operations to the EBS.

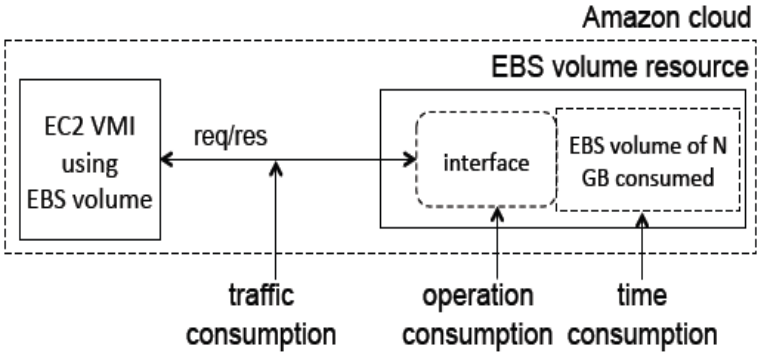


Figure 4. 8 EBS accounting model

In principle and as shown in the figure, an EBS incurs traffic consumption. However, Amazon does not currently charge for this traffic. Operation consumption is measured as the number of input/output operations that the EC2 VMI places against the EBS. Resource consumption is measured in units of time (for example, hrs) and is determined as the time that elapses between the creation and deletion of the EBS. The reason for this is that the amount of storage consumed by the client is determined at the EBS creation time.

The EBS accounting model is weakly consumer–centric, because the accounting model for operation consumption includes unobservable events: as Amazon points out in their documentation, the exact number of disk input/output operations cannot be determined

accurately by clients because of caching that takes place within applications and operating systems. Fortunately, the number of input/output operations as ”seen” by a client is likely to be less than the actual numbers, so the discrepancy always favours the client.

4.4 Summary

The ‘Pay only for what you use’ principle underpins the charging models of widely used cloud services that are on offer. 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 all. Currently, consumers have no choice but to take whatever usage data has been made available to them by the provider as trustworthy. This situation motivated us to propose the notion of the 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. We evaluated infrastructure level resource accounting models of a prominent cloud service provider (Amazon) and found that they were only weakly consumer–centric. We presented ideas on how accounting models should be constructed so as to make them strongly consumer centric. We also suggested a systematic way of describing resource accounting models so that they can be understood and reasoned about by consumers.

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 much as possible, consumers are able to collect with ease their own usage data that matches provider side data with sufficient precision.

Chapter 5

Conclusion and Future Work

The main objective of this thesis is to address the issues related to developing an independent consumer-side accounting mechanism which allows customers to compute their outsourced resource usage charges.

To achieve this, several experiments have been conducted on different IaaS cloud providers (Amazon S3, EC2 and Nirvanix SDN) and different resources (storage, bandwidth, computer power and CPU) in order to understand the accounting model of each resource. Furthermore, the study has addressed the issues related to building a consumer-side resource accounting service which shows how metering data is collected and what parameters might cause a potential source of conflicts between the consumer's and the provider's measurements.

In this chapter we summarise the achievements and also suggest some potential developments for future research.

5.1 Summary of achievements

The achievements of this thesis are:

1. We proposed 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 by the provider. Further to this, 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/reasonableness 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 acting as brokers offering third party services.

Based on our definition of the consumer–centric accounting model we found the following:

- Nirvanix models of the bandwidth and resource consumption are strongly consumer-centric, but suffer from incompleteness and ambiguities.
- Concerning S3, we have found that the models of bandwidth and operation consumption are strongly consumer-centric, however, they suffer from incompleteness and ambiguities as we pointed out earlier (see chapter 3)). The model for resource consumption is weakly consumer-centric due to an unobservable checkpoint event which makes the overall Amazon S3 accounting model weakly consumer-centric.
- The EC2 bandwidth consumption accounting model is strongly consumer-centric. Whereas, the accounting model of the EC2 on-demand instance is weakly consumer-centric because the event that causes a virtual machine instance to progress from pending to a running state is not visible to the consumer.

2. Discrepancies between the accounting data computed by consumer and the provider can be classed into three categories discussed below.
   - Incompleteness, ambiguities and inconsistencies: we highlighted several cases where an accounting model specification was ambiguous or not complete. For example, concerning EC2, it is not clear how instances that fail accrue instance hours. Also, for different APIs, the service provider may apply different accounting models to compute resource consumption. For instance, Amazon S3 computes bandwidth consumption based on the whole message size for SOAP operations, while for RESTful operations S3 computes the bandwidth as the bytes transferred per request or response. We also pointed out that the

operations executed by different APIs may have different levels of resource consumption. For instance, in Amazon S3, executing an operation using a RESTful request consumes fewer resources than the same operation executed by a SOAP request. This can be counted as a good example of inconsistency.

- Unobservable events: If an accounting model uses one or more events that impact resource consumption, but these events are not observable to (or their occurrence cannot be deduced accurately by) the consumer, then the data collected at the consumer side could differ from the provider's. The calculation of storage consumption in S3 (GBHrs) is a good example: here, the checkpoint event is not observable.
- Differences in the measurement process: Differences can arise if the two sides use different techniques for data collection. Calculation of GBHrs again serves as a good example. We expect that for a checkpoint, the provider will directly measure the storage space actually occupied, whereas, for a given checkpoint time, the consumer will mimic the process by adding (for PUT) and subtracting (for DELETE) to calculate the space, as we discussed in chapter 4.

3. We also suggested a systematic way of describing resource accounting models so that they could be understood and reasoned about by consumers. The idea is to define a set of "elementary" chargeable resources and then describe the overall resource consumption of a given resource/service in terms of an aggregation of the consumption of these elementary resources. 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 the provider side data with sufficient precision.

5.2 Future Work

In this section we propose some potential developments that could stem out from our work.

5.2.1 Consumer-side accounting for PaaS and SaaS

As stated earlier in chapter 2, cloud computing has three service models; Software as a Service (SaaS), the Platform as a Service (PaaS), and the Infrastructure as a Service (IaaS) [22, 23, 26]. This study has tackled the issues related to the consumer-side accounting in IaaS services. Further work should investigate the challenges and technical issues that are related to consumer-side accounting at the level of Paas and SaaS.

5.2.2 Verifying Billing Charges

When consumers use cloud resources they need to understand how a given deployment will be charged. Ideally, consumers should be in a position to verify the charges billed to them. In turn this requires taking into consideration the particularities (for example, geographical location of resources) of the deployment and the provider's current pricing policies. We believe that the abstract resource accounting model provides a good starting point for developing a tool that can take deployment configuration information and pricing policies to compute billing charges. We suggest this as another direction of future work, and use the hypothetical deployment shown in Figure 5.1 for the sake of illustration.

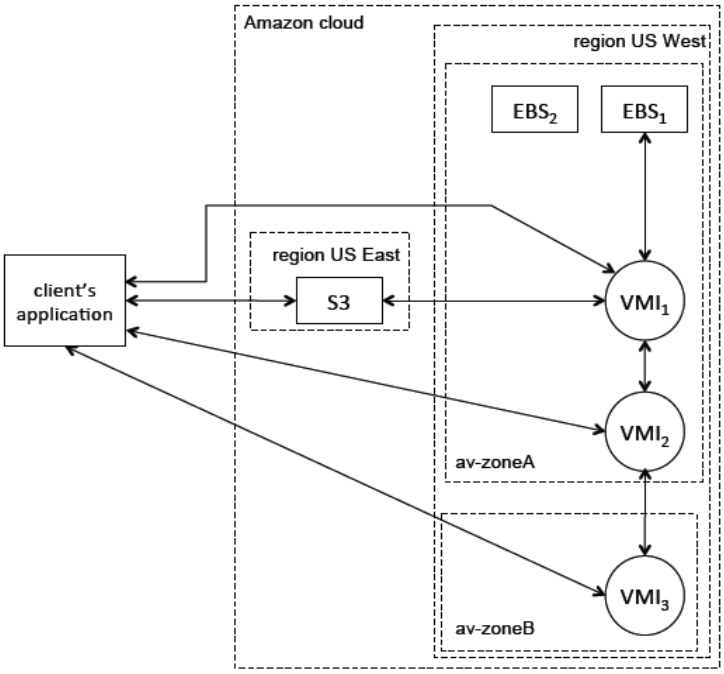


Figure 5. 1 Resource deployment

The deployment of Figure 5.1 involves the client's application and three types of Amazon basic resources: S3 storage, EC2 virtual machine instances (VMIs) and EBS volumes. It also involves two Amazon regions (US East and US West) and two availability zones (av–zoneA and av–zoneB) located within the US West region. The Amazon cloud is divided into regions which are physical locations geographically dispersed (e.g. US–East in Northern Virginia, US–West in Northern California and the EU in Ireland).

The EC2 cloud is divided into zones which are failure–independent data centres located within Amazon regions and linked by low latency networks. The arrowed lines represent bi–directional communication channels. Omitted from the figure are the communication channels used by the client to issue administrative commands to the VMIs (launch, stop, reboot, etc.) and the EBSs (create volume, attach volume, etc.).

We open this discussion with a study of the charges that apply to EBS1 and EBS2. Imagine for the sake of argument that they are volumes of 50 GB and 100 GB, respectively. Of concern to us here is the operation consumption and time consumption of the EBSs. EBS_1 will be charged for the number of input/output operations thatVMI_1

places against the EBS_1 interface and also for the period of time of usage of the allocated 50 GB. Being currently detached, the charges for EBS_2 are simpler to calculate, they will only consider the time consumption for 100 GB. In general, Amazon charges for traffic in (Data Transfer–In) and out (Data transfer–Out) of the Amazon cloud and for traffic in and out of the EC2 cloud. However, Amazon does not charge for traffic between a VMI and another resource (say S3 storage) located within the same region. Neither do they charge for traffic between two VMIs located within the same availability zone. However, Amazon charges for inter–region traffic between a VMI and another resource (for example, S3) located within a different region. In these situations, the sender of the data will be charged for Data Transfer–Out whereas the receiver will be charged for Data Transfer–In. With these pricing policies in mind, let us study the charges for VMI_1. Of concern to us here is traffic consumption and resource consumption. VMI_1 will be charged for inter–region traffic (Data Transfer–In and Data Transfer–Out) consumed on the channel that links it to S3. In addition, VMI_1 will be charged for traffic (Data Transfer–In and Data Transfer–Out) consumed on the channel that links VMI_1 to the client application, as the latter is outside the Amazon cloud. There are no charges for traffic consumed by the interaction against EBS_1 as traffic consumed by the interaction between VMIs and EBSs is free. Neither are there charges for traffic consumed by the interaction against VMI_2 since VMI_1 and VMI_2 share availability zone A. Resource consumption of VMI_1 will be counted as the number of hours that this instance is run.

The charges for VMI_2 will take into account traffic consumption and resource consumption. The traffic consumed will be determined by the amount of Data Transfer–Out and Data Transfer–In sent and received, respectively, along two channels: the channel that leads to the client's application and the one that leads to VMI_3. There are no charges for traffic consumed on the channel that leads to VMI_1 because the two instances are within the same availability zone. Again, resource consumption will be counted as the number of instance hours of VMI_2. The charges for VMI_3 can be calculated similarly to VMI_2.

We can visualise that S3 will incur charges for traffic consumed on the channel that links it to VMI_1 and on the channel that links it to the client's application. In addition, S3 charges will account for operation consumption counted as the aggregation of the number of operations placed against S3 by the client's application and VMI_1. In addition, the charges will take into consideration resource consumption (storage space consumed) measured in storage–time units. This will be counted as the aggregated impact of the activities (put, get, delete, etc.) performed by the client's applications and VMI_1.

5.2.3 Cost estimation of service delivery

The idea presented in the previous subsection can be extended further to make cloud-based applications billing aware, by developing techniques for estimating at run-time the charges that an application has incurred so far. A cloud service broker managing applications on behalf of customers can use such techniques for estimating at run-time, the cost of service delivery to its customers and whether the service is adequately provisioned. A broker can ensure that the customer's applications do not exceed agreed budgets, and use cloud resources in cost-efficient manner.

Bibliography:

1. H. Ragib, Y. William, and M. Suvda, The evolution of storage service providers: techniques and challenges to outsourcing storage, in Proceedings of the 2005 ACM workshop on Storage security and survivability. 2005, ACM: Fairfax, VA, USA.
2. C. Molina-Jimenez, N. Cook, and S. Shrivastava, On the Feasibility of Bilaterally Agreed Accounting of Resource Consumption, in Service-Oriented Computing - Icsoc 2008 Workshops, G. Feuerlicht and W. Lamersdorf, Editors. 2009, Springer-Verlag Berlin: Berlin. p. 270-283.
3. A. Mihoob, and C. Molina-Jimenez. A Peer to Peer Protocol for Online Dispute Resolution over Storage Consumption. in The Fourth European Young Researchers Workshop on Service Oriented Computing. 2009. Pisa, Italy: EPTCS 2 - YR-SOC 2009.
4. Amazon web services, Amazon Simple Service Storage (S3), http://aws.amazon.com/s3. 2006.
5. B. Aboda, J. Arkko, D. Harrington: "Introduction to Accounting Management", RFC2975, October 2000
6. Amit Goyal, Sara Dadizadeh (2009). A Survey on Cloud Computing. University of British Columbia, Technical Report, 2009.
7. Shastri JC Philip, "What Is Cloud Computing?"; http://factoidz.com/what-is-cloud-computing-1/
8. R. Miller, (2008, March 25). What's In A Name? Utility vs. Cloud vs. Grid. Retrieved September 29, 2009, from Data Center Knowledge: http://www.datacenterknowledge.com/archives/2008/03 /25/whats-in-a-name-utility-vs-cloud-vs-grid/.
9. A. Khalid. "Cloud computing: Applying Issues in Small Business", International Conference on Signal Acquisition and Processing, 2010, pgs. 278 – 281.
10. Life in the Cloud, Living with Cloud Computing, http://computinginthecloud.wordpress.com/2008/09/25/utility-cloud-computingflashback-to-1961-prof-john-mccarthy/Published on: 25/09/2008 / 13:09
11. La'Quata Sumter, "Cloud Computing: Security Risk", Proceedings of the 48th Annual Southeast Regional, 2010 - portal.acm.org, ACMSE '10, April 15-17, 2010, Oxford, MS, USA.
12. Martin Adolph, Distributed Computing: Utilities, Grids & Clouds (March 2009), ITU-T Technology Watch Report 9, International Telecommunication Union.
13. R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Future Generation Computer Systems*, 25(6): 599-616, Elsevier Science, Amsterdam, The Netherlands, June 2009.
14. L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, and W. Karl, "Scientific Cloud Computing: Early Definition and Experience," in 10th IEEE International Conference on High Performance Computing and Communications, HPCC '08. , pp. 825-830, 2008.
15. J. Geelan. Twenty one experts define cloud computing. Virtualization, August 2008. Electronic Magazine, article available at http://virtualization.sys-con.com/node/612375.
16. I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in Grid Computing Environments Workshop, 2008, pp. 1–10.
17. Terrence V. Lillard, Clint P. Garrison, Craig A. Schiller (2010). The future of cloud computing. Digital Forensics for Network, Internet, and Cloud Computing, 2010, Pages 319-339.
20. SearchCloudComputing, definition cloud computing (December 2007), http://searchcloudcomputing.techtarget.com/definition/cloud-computing, retrieved on (01/12/2011).
21. Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", UC Berkeley Reliable Adaptive Distributed Systems Laboratory,http://radlab.cs.berkeley.edu/, February 10, 2009.
22. Mell, P. and Grance, T. 2009. The NIST Definition of Cloud Computing. National Institute of Standards and Technology.
23. F. Panzieri, O. Babaoglu, V. Ghini, S. Ferretti, M. Marzolla, Distributed Computing in the 21st Century: Some Aspects of Cloud Computing, , May 2011.

24. Dave Malcolm Surgient,The five defining characteristics of cloud computing , Special to ZDNet, http://www.zdnet.com/news/the-five-defining-characteristics-of-cloud-computing/287001, April 9, 2009 10:31 AM PDT.
25. Maria Spínola "The Five Characteristics of Cloud Computing- Including a look at the possible delivery and deployment models", cloud computing journal, http://cloudcomputing.sys-con.com/node/10874, 26September 6, 2009.
26. Judith Hurwitz, Robin Bloor, Cloud computing for dummies. HP special edition [book]
27. Microsoft Azure. http://www.microsoft.com/azure.
28. Google App Engine. http://code.google.com/appengine.
29. Nirvanix Storage Delivery Network. http://www.nirvanix.com/
30. P. Patel, A. Ranabahu, and A. Sheth, "Service Level Agreement in Cloud Computing," in Proceedings of the Workshop on Best Practices in Cloud Computing: Implementation and Operational Implications for the Cloud at ACM International Conference on Object- Oriented Programming, Systems, Languages, and Applications, Orlando, FL, Oct. 2009.
31. D. Nurmi, R. Wolski, etc., "The Eucalyptus Open-source Cloud computing System," in Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, Shanghai,2009, 124-131.
32. B. Sotomayor, K. Keahey, I. Foster. Combining Batch execution and Leasing Using Virtual Machines, HPDC 2008, Boston, MA, 2008, 1-9
33. K. Keahey and T. Freeman, "Science Clouds: Early Experiences in Cloud Computing for Scientific Applications," in proceedings of Cloud Computing and Its Applications 2008, Chicago, IL. 2008.
34. V. Srinvasa Rao, K. Nageswara Rao, E. Kusuma Kumari, "Cloud Computing: an overview", "Journal of Theoretical and Applied Information Technology", © 2005 - 2009 JATIT. www.jatit.org
35. Steve Bennett, Mans Bhuller, Robert Covington, Oracle White Paper in Enterprise Architecture – Architectural Strategies for Cloud Computing August 2009, Redwood Shores, CA 94065 U.S.A.
36. L. Youseff, M. Butrico, and D. Da Silva, "Towards a Unified Ontology of Cloud Computing," Proc. Grid Computing Environments Workshop (GCE), IEEE Press, 2008; doi: 10.1109/GCE.2008.4738443.
37. Amazon (2006), "Amazon Simple Storage Service, Developer guide, API version 2006-03-01". Available from: www.amazon.com. (Accessed on: 20-8-2009).
38. J. Murty, Programming Amazon Web Service. O'Reilly, 2008.
39. Nirvanix, "Nirvanix Storage Delevery Network, Develop guide, API version 2.1", Available from: http://developer.nirvanix.com/sitefiles/1000/API.html. (Accessed on: 2009)
40. C. Mills, D. Hirsh, G. Ruth, RFC 1272, " INTERNET ACCOUNTING: BACKGROUND", Network Working Group, November 1991.
41. N. Brownlee, C. Mills, G. Ruth: "Traffic Flow Measurement: Architecture ", RFC 2063, Network Working Group, January 1997.
42. N. Brownlee, C. Mills, G. Ruth, Traffic Flow Measurement: Architecture, RFC 2722, Network Working Group, October 1999.
43. C. Rigney, S. Willens, A. Rubens, W. Simpson, RFC 2865, "Remote Authentication Dial In User Service (RADIUS)", Network Working Group, June 2000.
44. T. Zseby, S. Zander, G. Carle,Fraunhofer FOKUS, "RFC 3334 Policy-Based Accounting", Network Working Group, October 2002.
45. G. Carle, Fraunhofer FOKUS, G. Gross, L. Gommans, J. Vollbrecht, D. Spence , "RFC 2904 Generic AAA Architecture ", Network Working Group, August 2000.
46. J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat , M. Holdrege, D. Spence, RFC 2904 " AAA Authorization Framework", Network Working Group, August 2000.
47. A. Pras, B.-J. van Beijnum, R. Sprenkels, R. Parhonyi: Internet Accounting, IEEE Communications Magazine, May 2001, pp 108-113.
48. V. Agarwal, N. Karnik and A. Kumar,"Metering and accounting for composite e-Services," in Proc. 1st IEEE Int'l Conf. on E-Commerce, pp. 35-39, 2003.

49. I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. Grid services for distributed system integration. IEEE Computer, 35(6):37–46, June 2002.
50. S. Graham, S. Simeonov, T. Boubez, G. Daniels, D. Davis, Y. Nakamura, and R. Neyama. Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI. Sams, 2001.
51. H. Motahari-Nezhad, B. Stephenson, and S. Singhal, "Outsourcing Business to Cloud Computing Services: Opportunities and Challenges,"Available at www. hpl.hp.com/techreports/2009/HPL-2009-23.html, 2009.
52. Amazon Web Service, Elastic Cloud Computing, http://aws.amazon.com/ec2/
53. Elmroth, E., Marquez, F., Henriksson, D. and Ferrera, D. "Accounting and billing for federated cloud infrastructures"'2009 Eighth International Conference on Grid and Cooperative Computing', IEEE, 2009, pp. 268-275.
54. Ge Zhang, Bernd Reuther (2005). A model for user based traffic accounting. EUROMICRO '05 Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications.
55. McKell, L.J., Hansen, J.V., Heitger, L.E.: Charging for computing resources. Computing Surveys 11(2), 105-120 (Jun 1979).
56. Kim, S.D., Park, S.H., Keum, C., Chung, T.M.: A study new challenge for billing system in converged service platform. In: Proc. 6th Int'l Conf. on Networked Computing and Advanced Information Management (NCM'10). pp. 390-395 (2010)
57. Amazon Web Services. http://aws.amazon.com/
58. Amazon Web Service, Elastic Cloud Computing, http://aws.amazon.com/ec2/pricing
59. G. Dantzig and B. Curtis Eaves, Fourier-Motzkin elimination and its dual, Journal of Combinatorial Theory (A), Vol. 14, pp. 288–297, 1973.
60. M. R. Palankar, A. Iamnitchi, M. Ripeanu, and S. Garfinkel. Amazon S3 for science grids: a viable solution? In DADC '08: Proceedings of the 2008 international workshop on Data-aware distributed computing, pages 55–64. ACM, 2008. 4, 5, 19.
61. C. Molina-Jimenez, S. Shrivastava, On the Monitoring of Contractual Service Level Agreements, in Proceedings of the First IEEE International Workshop on Electronic Contracting. 2004, IEEE Computer Society.
62. Asawa, M., Measuring and analyzing service levels: A scalable passive approach, in 1998Sixth International Workshop on Quality of Service. 1998, Ieee: New York. p. 3-12.
63. Debusmann, M. and A. Keller. SLA-driven management of distributed systems using the common information model. in Integrated Network Management, 2003. IFIP/IEEE Eighth International Symposium on. 2003.
64. K. Alexander, L. Heiko, The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services. J. Netw. Syst. Manage., 2003. 11(1): p. 57-81.
65. S. Akhil, M. Vijay, S. Mehmet, L. Li Jie, and C. Fabio, Automated SLA Monitoring for Web Services, in Proceedings of the 13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Management Technologies for E-Commerce and E-Business Applications. 2002, Springer-Verlag.
66. Ludmila Cherkasova Yun Fu, Wenting Tang and Amin Vahdat, Measuring and characterizing end-to-end Internet service performance. ACM Trans. Internet Technol., 2003. 3(4): p. 347-391.
67. Keynote Systems, http://www.keynote.com.
68. J. Pruyne, Enabling QoS via Interception in Middleware, in Software Technology Laboratory HP Laboratories Palo Alto 2000, HP.
69. Amazon Web Service, "Amazon ec2 faqs," 2011. [Online]. Available: aws.amazon.com/ec2/faqs
70. A. Korn, C. Peltz, and M. Mowbray, "A service level agreement authority in the cloud," HP Laboratories, Tech. Rep. HPL-2009-79, 2009.
71. 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.
72. Amazon: Amazon elastic compute cloud user guide (api version 2011-02-28) (2011), docs.amazonwebservices.com/AWSEC2/latest/UserGuide/
73. http://davidpallmann.blogspot.com/2010/08/hidden-costs-in-cloud-part-1-driving.html
74. http://davidpallmann.blogspot.com/2010/08/hidden-costs-in-cloud-part-2-windows.html

75. Vecchiola, C., Pandey, S., Buyya, R.: High-performance cloud computing: A view of scientific applications. In: Proc. 10th Int'l Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN'09) (2009).
76. Wang, G., Ng, T.S.E.: The impact of virtualization on network performance of Amazon ec2 data center. In: Proc. 29th IEEE Conf. on Computer Communications (INFOCOM'10). pp. 1-9 (2010).
77. Wachs, M., Xu, L., Kanevsky, A., Ganger, G.R.: Exertion-based billing for cloud storage access. In: Proc. 3rd USENIXWorkshop on Hot Topics in Cloud Computing (HotCloud'11) (2011).
78. Wang, H., Jing, Q., Chen, R., He, B., Qian, Z., Zhou, L.: Distributed systems meet economics: Pricing in the cloud. In: Proc. 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud'10) (2010).
79. Den Bossche, R.V., Vanmechelen, K., Broeckhove, J.: Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In: Proc. IEEE 3rd Int'l Conf. on Cloud Computing (Cloud'10). pp. 228-235 (2010).
80. Li, A., Yangi, X., Zhang, S.K.M.: Cloudcmp: Shopping for a cloud made easy (2010).
81. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: The montage example. In: Proc. Int'l Conf. on High Performance Computing, Networking, Storage and Analysis (SC'08) (2008)
82. Chris Mitchell, " Trusted Computing: Putting a security module on every desktop ", Information Security Group Royal Holloway, University of London, (Visiting Erskine Fellow, University of Canterbury), www.opentc.net 28 March 2007.
83. Nancy R. Mead and Gary McGraw, Understanding Trusted Computing, IEEE SECURITY & PRIVACY MAY/JUNE 2003.
84. P.C. van Oorschot, " Revisiting Software Protection ", Proc. 6th Information Security Conf. (ISC'03), Bristol, UK, LNCS 2851, Springer, 2003.
85. P. Wang, S.-K. Kang and K. Kim, "Tamper-resistant Software Through Dynamic Integrity Checking", Proc. Symp. on Cryptography and Information Security (SCIS'05), Maiko Kobe, Japan, 2005.
86. Morgan, G., Parkin, S., Molina-Jimenez, C. and Skene, J, Monitoring Middleware for Service Level Agreements in Heterogeneous Environments, In Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government. 5th IFIP Conference on e-Commerce, e-Business, and e-Government (I3E 2005), October 28-30 2005, Poznan, Poland Funabashi, M. and Grzech, A. (eds.)
87. A. M. Odlyzko, "Internet pricing and the history of communications", Computer Networks, 36(5-6):493–517, Aug. 2001.
88. Mark O'Neill, CTO, Vordel (2010). Who meters the cloud? Available at http://www.ebizq.net/topics/cloud_computing/features/12250.html
89. James Skene, Allan Skene, Jason Crampton, Wolfgang Emmerich, " The Monitorability of Service-Level Agreements for Application-Service Provision", WOSP'07, February 5–8, 2007, Buenos Aires, Argentina. Copyright 2007 ACM 1-59593-297-6/07/0002.
90. Pias, M., Wilbur, S., Bhatti, S., Crowcroft, J.: Securing the internet metering and billing. In: Proc. IEEE Global Telecommunications Conf. (GLOBECOM'02). pp. 1603-1607 (2002)
91. R Steinberg, "Pricing internet service", (2003), Working Paper (University of Cambridge WP 18/2002).
92. VMware vMotion, 2010. http://www.vmware.com/products/vmotion/.
93. Xen hypervisor, 2010. http://www.xen.org/.
94. A. M. Odlyzko, "Internet pricing and the history of communications", Computer Networks, 36(5-6):493–517, Aug. 2001.
95. Mark O'Neill, CTO, Vordel (2010). Who meters the cloud? Available at http://www.ebizq.net/topics/cloud_computing/features/12250.html