A Model for Policy-based Automation of Usage Accounting across Multiple Cloud Infrastructures

Joachim Götze, Tino Fleuren, Bernd Reuther, Paul Müller
Dept. of Computer Science, ICSY Lab
University of Kaiserslautern
Kaiserslautern, Germany
{j_goetze, fleuren, reuther, pmueller}@cs.uni-kl.de

Abstract—Cloud Computing provides flexible and dynamic provisioning of resources, services, and applications. As such, Cloud Computing is the ideal IT infrastructure for the ever changing workload of companies and service providers. Although, cloud providers offer functionality for usage accounting, this functionality is limited to their own requirements. Companies and service providers as cloud users have also a need for usage accounting, but with different requirements than the cloud providers. Additionally, cloud users are not limited to a single cloud, but make use of multiple cloud infrastructures and applications depending on their needs.

Cloud users require a usage accounting infrastructure not only capable of supporting billing as an accounting application, but capable of supporting all kinds of applications. For example applications like cost allocation or trend analysis. In order to be able to manage such a complex and adaptable usage accounting infrastructure, we present a policy-based management approach. Policies are used as a high-level description of the intended behavior of the infrastructure which are then utilized to derive configurations for the infrastructure services. Such a solution ensures the efficient management and administration of complex usage accounting infrastructures.

Keywords—usage accounting; cloud computing; policy-based management

I. MOTIVATION

Cloud Computing provides flexible and dynamic provisioning of resources, services, and applications [1], [2]. As such, Cloud Computing is the ideal IT infrastructure for the ever changing workload of companies and service providers. Although cloud providers offer functionality for usage accounting, this functionality is limited to their own requirements. Obviously, the aspect of billing is of major concern for a cloud provider and therefore it is addressed by their infrastructure. Especially companies using public clouds have additional requirements like cost allocation or trend analysis for resources and services used.

In order to run an infrastructure for usage accounting not only supporting billing, but being able to support further applications, the underlying usage accounting framework needs to be able to adapt towards the needs of these applications. Additionally, the needs of the applications as well as their users differ. Hence, no single solution can be applied to all applications.

As a result, such an adaptable accounting infrastructure is rather complex, because of the high level of distribution as well as the multitude of possible adaptations of these components. In order to ensure the ability to manage such an infrastructure, the people administering such an infrastructure need management support.

The solution proposed here is a policy-based management support for a usage accounting infrastructure. Policy-based management uses high-level policies, e.g., defining the usage accounting goals for a whole company, as a means to derive a suitable configuration for all components within the infrastructure. This abstraction reduces the management complexity of the possible adjustments to the infrastructure.

The remainder is organized as follows: In section II, the term accounting is reviewed and defined with respect to notable publications concerned with the topic to be preparatory to a better understanding of this paper. Section III gives a short overview of currently existing solutions for usage accounting and policy-based management approaches. Afterwards, Section IV analyzes typical applications for usage accounting and evaluates their requirements. In Section V our accounting solution is described shortly, focusing on configurable parameters used by the policy-based management approach proposed in section VI. Finally, Section VII concludes the paper with a summary and a short overview of future work.

II. TERMINOLOGY

The term accounting is still missing a consistent definition within computer science and is used by various authors with slight differences [3]–[5]. As described in [6], this is mainly caused by the term's origin. Several years ago, networking was perceived as a telecommunication topic, where accounting is closely related with the payment for using the telecommunication network. Today, after a shift towards an “IP world”, the term is more closely related to gathering, transporting, and storing data about resource consumption. Payment has become a mere application amongst many making use of accounting data. Still, reviews of accounting literature show that the term is more often used in the fashion of telecommunication than not [7], [8]. In order to be precise, we use the term usage accounting to denote that the focus is on gathering, transporting, and storing data about resource consumption, while billing is just an application based on usage data.

1 The project was funded by the Stiftung Rheinland-Pfalz für Innovation.
III. RELATED WORK

Usage accounting is a topic addressed prevalent in the areas of networking as well as grid and cloud computing. The networking protocol Remote Authentication Dial In User Service (RADIUS) is an example for the networking domain and provides centralized authentication, authorization, and accounting management for computers using a network [9]. Diameter is a successor of RADIUS and provides an authentication, authorization and accounting framework for applications such as network access or IP mobility [10]. The accounting aspect of both protocols is limited and mainly focusing on the amount of time a networking connection is active and the amount of traffic created.

The design of accounting infrastructures for grid environments led to several solutions with varying acceptance level. Most of the solutions focus only on computing resources. Some examples are DGAS [11], SGAS [12], GridBank [13], and many more [14]–[16]. None of these address usage accounting for resources beyond computing and storage or general services.

Usage accounting is also a topic for enterprise-level applications that gets addressed for example by IBM Tivoli [17]. Because of the close relation to the vendor-specific applications and services, we see these kinds of solutions not as a suitable approach for a generic accounting environment.

In cloud computing, all providers need to have usage accounting to be able to send an invoice to their users. The examples for those solutions are manifold and we reference here just Amazon Web Services (AWS)\(^2\) as one of the market leaders. Although usage accounting is a basis for these companies operations, the focus is merely on the provider needs. If further applications based on usage data are support for the users, e.g., cost allocation in AWS, the functionality is very limited.

The approach of using policies to reduce the effort of configuring a complex environment has been used in various scenarios. [18] defines basic terminology for policy-based management. In [19] a policy-based accounting approach for networking has been defined. This approach is on a conceptual level and focused mainly on the protocol specification for exchanging policies with network components. [20] uses policies for coordinating quality of service in DiffServ [21] (short for Differentiated Services), an approach for the classification of IP packets. In [22] and [23] approaches are described for managing an environment based on Web services using service-level agreements. In such an environment, service-level agreements can be seen either as the basis for defining management policies for an infrastructure or as a generalization of the policy concept with the goal to allow the service user to influence the management policies of a provider’s infrastructure.

IV. INFLUENCE OF APPLICATIONS ON USAGE ACCOUNTING

[24] introduces four typical applications for usage accounting: billing, cost allocation, trend analysis, and process auditing. Considering the aspects of timing, availability, and security, the requirements of these applications differ distinctly. In this context, the term timing refers to whether or not the usage accounting is time-critical. The terms availability, i.e., the readiness to deliver a service, and the term security which consists of integrity, i.e., the absence of unauthorized modifications, and confidentiality, i.e., the absence of unauthorized information disclosure, are used according to the definition in [25], [26].

These terms provide the means to define high-level policies influencing the configuration of the usage accounting. Therefore, these terms will be considered in the following with respect to the usage accounting applications.\(^3\)

A. Billing

Billing is the usage accounting application that comes to mind first. Obviously, most people providing a service want to have the ability to charge for this service. So, billing is the process of adding up the individual charges, sending a bill to the customer, and handling the money transfer. The money transfer can either relate to real cash or virtual coins depending on the favored scenario of the provider.

Billing can be based on two general types of tariffs: flat rate tariff and pay-per-use tariff. Flat rate tariff is based on price paid regularly for a service rendered, while a pay-per-use tariff requires the user to only pay for the amount of a service really used.

Additionally, payment can be handled in two ways: pre-paid or post-paid. In a pre-paid scenario, the user has to pay money in advance to his account and can use a service as long as a withdrawal from this account is possible. In a post-paid scenario, the user can make use of a service as requested and the account will be balanced on a regular basis depending on the accrued cost of the usage.

With respect to usage account, these tariffs and payment scenarios lead to different requirements. Billing using a flat rate tariff is quite easy and is especially not depending on any kind of usage accounting. The user is charged for the same amount of money every time without taking the service usage into account. If this is the case, a service provider is typically more concerned with trend analysis in order to get a prediction of future resource requirements.

Billing using a pay-per-use tariff has more complex requirements. In a post-paid scenario (cf. Table I), timing is noncritical because of usually monthly payments. In general, availability is neutral, although the availability of the metering sensors is important in order to constantly recognize service usage. Security of the usage data is critical, because on the one hand it contains personal data which allows to recognize

\(^2\)AWS Web site: http://aws.amazon.com

\(^3\)For the classification of applications, the following scale and abbreviations have been used: critical (+), neutral (o), noncritical (-).
the usage behavior of a person or a group of people, while on the other hand modification of the usage data can lead to irregularly reduced charges. In a pre-paid scenario (cf. Table II), security, availability, and timing are all critical parameters. From a security point of view, this scenario is comparable to the post-paid scenario. Availability and timing on the other hand get very important, because the usage data must be available on time in order to update the account balance of the user which is only possible if the accounting infrastructure is available to enable timely transmission.

### TABLE I

**Requirements for Billing Using a Pay-Per-Use Tariff and a Post-Paid Payment Scenario**

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>o</td>
<td>-</td>
</tr>
</tbody>
</table>

### TABLE II

**Requirements for Billing Using a Pay-Per-Use Tariff and a Pre-Paid Payment Scenario**

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

#### B. Cost Allocation

The main goal of cost allocation is the identification of expenses required to provide and maintain an infrastructure. Therefore, overhead expenses, which are not directly linked to a project, product or account, are transformed in product-related expenses. The expenses for provisioning and maintenance of an IT infrastructure is a common example for overhead expenses. Usage accounting can now provide a means to identify and link infrastructure related expenses like for example accessing or deploying a cloud service to a specific account.

Cost allocation can either be used for internal restructuring and optimization plans or for a better understanding of cost drivers with in projects. Especially, internal restructuring and optimization plans lead to unrest of employees, because of a fear for change or dismissal in order to reduce personnel costs. Hence, security is very important in this scenario (cf. Table III). Although usage data in such a scenario is typically not directly associated with a specific person and modeling personal usage behavior is often not an issue, the integrity of the acquired usage data is important in order to prevent tampering with the data in order to sabotage restructuring. Availability is not so important in this scenario, because restructuring is a long term goal and short outages of the infrastructure are negligible. Timing is noncritical, because cost allocation is a long running process where adjustments are made over long periods of time.

#### C. Trend Analysis

Trend analysis tries to provide estimates of future resource consumption and service usage in order to enable infrastructure upgrades on time to meet future requirements. This usage accounting application is noncritical with regard to all their requirements contemplated (cf. Table IV). From the security point of view, the requirements are noncritical, because no personal usage data is created. The focus is on resource and service usage and the assigning of the user to the usage is not required. Additionally, trend analysis deals with estimations of long term goals and hence the availability of the usage accounting infrastructure is noncritical as well as the timing.

### TABLE III

**Requirements for Trend Analysis**

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### D. Process Auditing

The provisioning and maintenance of an IT infrastructure is costly for the provider. Therefore, the improvement and optimization of IT processes is of interest for the infrastructure providers. Usage data is the basis for various forms of process auditing like for example process optimization or the analysis of non-compliance with respect to usage policies, service level agreements or security policies.

Process optimization focuses on the process itself not on the usage of IT resources. Therefore, usage data in such a scenario is typically of a coarse granularity with respect to usage, but more fine grained with respect to the process steps. The usage accounting requirements of process optimization (cf. Table V) are mostly noncritical, because timing and availability of the infrastructure are of little importance in a mostly manual task like process optimization. Only security can be a critical requirement, if the acquired usage data is also fine grained with respect to the user. Again, the modeling of personal usage behavior has to be prevented.

Process auditing as the monitoring of usage with respect to company policies and the corresponding IT policies has other requirements (cf. Table VI). Especially, the timing of usage data becomes of critical importance, because any non-compliance with a policy – particularly if it is a security related issue – needs to lead to swift action to prevent further violation. The demand for timing includes also a demand for availability, because infrastructure availability is the basis for signaling any non-compliance.

### TABLE IV

**Requirements for Trend Analysis**

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### TABLE V

**Requirements for Process Optimization**

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
TABLE VI
REQUIREMENTS FOR MONITORING OF COMPANY POLICIES OR IT POLICIES

<table>
<thead>
<tr>
<th>Security</th>
<th>Availability</th>
<th>Timing</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>0</td>
<td>+</td>
</tr>
</tbody>
</table>

V. USAGE ACCOUNTING ARCHITECTURE IN A NUTSHELL

In order to be able to adapt the usage accounting infrastructure to the requirements of the application, an infrastructure offering such a kind of adaptability is required. In this section, a short overview of our solution is given [27].

Fig. 1 shows the general architecture of the accounting infrastructure. On the bottom, the Metering Layer contains probes for measuring resource and service usage for computing or storage resources amongst others. These probes create raw meter data. On top of this, the Accounting Abstraction Layer converts the raw meter data into a generic usage record format. This usage record format is capable of handling the data for any resource or service type existing in the system. In order to handle the raw meter data of a specific probe, a plug-in concept is used to integrate and convert the data into the usage record format. The top layer consists of the Accounting Services which are on the one hand concerned with storing the usage data and making it available to applications operating on top of the usage accounting and on the other hand handle cross-domain accounting topics. Orthogonal to these layers is the management by accounting policies, which handles the configuration and adaptation of the infrastructure corresponding to the requirements of the applications making use of the accounting data.

While the Metering Layer as well as the Accounting Abstraction Layer are located locally on the host monitored for accounting, the Accounting Services are placed within the accounting domain and can also be replicated to improve availability.

Fig. 1. Usage Accounting Architecture

In order to adapt the accounting infrastructure according to the requirements on the applications on top, the services within the accounting infrastructure must be configurable accordingly. There are two possibilities to adapt the infrastructure: the accounting model and the service options.

There are four accounting models available which have been described in detail in [24]:

- Polling Model: The probes and plug-ins just gather and preprocess the raw data, but only when they are triggered by their upper layer, the usage data is sent.
- Event-driven model: When probes and plug-ins have gathered and preprocessed the raw data, the usage data is automatically sent on to their upper layer.
- Event-driven model with batch processing: This model is comparable to the basic event-driven model, but the preprocessed data is not sent immediately, but depending on various parameters, e.g., available storage space or cache memory, the data is stored and sent in a larger batch.
- Event-driven model with polling: This model combines the advantages of the event-driven model and the polling model. Only the kind of event is signaled to the accounting services, which then decide if the usage data is to be sent immediately or if it is a low priority event and the data can be sent later on in a larger batch.

The various services participating within the accounting infrastructure provide the ability to be configured based on their service options. Service options are organized into three different categories: probe-related, communication-related, backend-related. Probe-related service options mainly focus on timing and caching of usage data. This includes parameters concerned with questions like:

- Cache time limit: How long will usage data be cached?
- Cache size limit: How many records will be cached?
- Cache persistence: Is persistent caching required?
- Prioritization: Which kinds of usage data are important/urgent and shall be sent immediately?

Communication-related service options are concerned with transmission efficiency and security, e.g.:

- Batch transmission: Shall each record be sent independently or shall batch transmission be used?
- Mutual authentication: Is authentication of service user and service required?
- Encryption: Shall the usage records be encrypted during transmission?

The backend-related service options focus on data storage and service availability, e.g.:

- Secure storage: Shall the usage data be encrypted before being stored in the accounting database?
- Redundancy: Is redundancy of Accounting services needed and to what magnitude?

The requirements of the accounting applications analyzed in section IV with respect to security, availability, and timing can be satisfied using an adequate accounting model and appropriate configurations of the service options. However, complex infrastructures residing (partly) in the cloud are highly dynamic and configuring such an infrastructure can be
difficult, error-prone, and time-consuming. In order to ensure proper configuration an automatism is desirable.

VI. Using Policies to Manage a Usage Accounting Infrastructure

Complex infrastructures containing services provided by cloud environments are time-consuming to configure and maintain. Using policies to manage a usage accounting infrastructure automatically can help to reduce this effort. Therefore, the approach proposed here will enable administrators to configure their usage accounting infrastructure only by choosing from a list of high-level policies as well as making some minor adjustments to policy constraints if desired.

The automation model has several inputs that need to be considered for deriving a configuration. The inputs can be distinguished into three categories: policies, system status, and capabilities. The policies describe the general requirements of the intended accounting applications, which can be adjusted by the administrator if specific constraints are different in the target environment. The system status takes into account the current configuration of services, probes, and the underlying network, e.g., accounting probes provide usage data containing information about the time span until a usage record has been delivered. Hence, it is also a kind of feedback from the system if the intended requirements are met. Additionally, the capabilities provide information about the accounting infrastructure components and what they are capable of, e.g., is a specific probe capable of storing cached usage data in a persistent way?

A. Policies

The policies currently considered are related to the requirements analyzed in section IV: security, availability, and timing. These policies are rated on a scale of none to maximum importance. Depending on the chosen level of these requirements, different accounting models and service options come into play.

While accounting models describe the general approach how to handle accounting in the system, service options need to be defined explicitly, because they directly influence the configuration options of components within the accounting infrastructure. The accounting models mainly influence the timing of usage data. The main factor here is whether to use the Polling Model when timing is not an issue or using an event-driven approach when more flexibility is required.

In order to reduce complexity even more, a set of policies is combined to a template specific for an application making use of accounting data. This allows a direct mapping between an application or a set of applications to a set of policies for security, availability, and timing. This allows a fast and easy setup of an accounting infrastructure.

B. System Status

The system status consists of the current configuration of all probes and services as well as the current status of the underlying network and the probes. Especially when the policies of an already running and configured system are modified, the current configuration is of importance. Some adjustments can be made without modifying already configured services and probes.

Additionally, the current status of the underlying network needs to be considered when timing constraints are to be met. Therefore, the current network status and utilization needs to be monitored. If the network utilization is high on a regular basis, the transmission of usage data can be delayed. Sometimes minor modifications are enough to meet the timing constraints.

The current status of probes can support the administration of the usage accounting infrastructure. On the one hand, the probes availability can be measured, while on the other hand, the usage records sent allows insight into the current timing of the system.

Some applications using usage data demand timing restrictions in order to provide a suitable performance. Often applications have either very strict demands on timing, e.g., billing based on a pre-paid scenario, or timing is not important at all, e.g., long-term trend analysis. An important indicator for timing within the usage accounting environment is the difference between the time stamp when a usage record has been created in the Accounting Abstraction Layer and the time stamp when a usage record has been received by the Accounting Services and stored in a database for further use. A common reason for larger differences in this area are either large time limits before batch processing of usage records or the use of the Polling Accounting Model with a large time interval between every single polling attempt.

C. Capabilities

The approach proposed here has demanding requirements to all components of the usage accounting infrastructure. Hence, it is only reasonable to consider that some components provide only a subset of the required functionality.

When the components come up, they need to register with the Accounting Services and provide information about their capabilities. These capabilities will be taken into consideration during the evaluation process, e.g., if persistent caching is required, but one of the probes is not capable of persistent caching, then the administrator should know about this. Not all of these capabilities are of a yes/no-type. Some capabilities like the cache size limit are only a limitation which might effect transmission efficiency, but it does not mean that caching in general is impossible.

D. Evaluation Process

The evaluation process is based on condition-action-rules, i.e., if a specific set of conditions becomes true, a specific action will be taken. The conditions are currently defined by the policies imposed on the system as well as the capabilities of its components. This combination leads to an action defining a configuration or at least a part of a configuration for some components.

In order to be able to define the service options explicitly, the service options need to be associated with their corresponding policies. Security policies are related to the service
options: mutual authentication, encryption, and secure storage.

Availability policies focus on persistence and redundancy.
Timing policies consider cache time limit, cache size limit, cache persistence, prioritization, and batch transmission.

Based on these categories the condition-action-rules can be defined, e.g.,

```
rule = {
    name : ExampleRule,
    condition = {
        security : critical,
        timing : non-critical
    },
    action = {
        communication = {
            encryption : yes,
            algorithm : RSA,
            keyLength : 4096
        }
    }
}
```

This easy example defines that configurations will be created with the capability to encrypt the communication channels using asymmetric encryption with RSA. Should some constraints lead to only unsatisfiable conditions, i.e., no rule can be executed, then the administrator needs to be informed to solve this situation. In general, the solutions are either creating or installing a component with more capabilities or reducing the requirements by manual adjustments to the constraints.

E. Configuration

After the evaluation of the inputs a suitable configuration for the accounting model and the service options of all services and probes are created. These are compared with the existing configurations and those infrastructure components where a new configuration is required will be triggered to update their configuration accordingly.

Typically, an accounting model is only chosen once during the first deployment, but a large reorganization of the infrastructure or new policy constraints can lead to deploying another accounting model. This happens only very rarely in a production environment.

The success of the policy-based automation approach needs to be monitored constantly. In order to do so, the current system status needs to be compared with the active policies. The policies describe which aspects are important and what criteria should be met to operate successfully.

F. Monitoring and Re-evaluation

A complex IT infrastructure is usually in a constant process of change. Hence, the monitoring of the infrastructure is important for ensuring that the initially defined policies are still met. If this is not the case, then the configuration needs to be re-evaluated.

Another reason for re-evaluation is the introduction of new components and probes that are joining the infrastructure. These components and probes need their initial configuration, but also their capabilities might lead to changes in the configuration of other components, e.g., a new probe is only capable of using the polling model for transmission of usage data. Hence, the related components must be reconfigured in order to retrieve the usage data by polling although other probes are capable of sending the data using the event-driven model.

G. Example

The automation model described in the previous sections has been developed and implemented as a prototype. The following example (cf. Fig. 2) demonstrates the automation process and consists of a cloud-based application which is provided and maintained by the company’s IT department in a virtual cloud environment, e.g., Google App Engine, and accessible only for company employees. The application is using a licensed software package to provide some of its functionality. The license key for the software package is provided by the company’s license service, e.g., FlexLM. The data used by the application is stored in a cloud storage like Amazon S3. This cloud-based application is used internally within several projects for various customers. Because the processing time of this application varies in magnitude depending on the input data and the size of the input data, the cost for using this application is not evenly distributed. Hence, the fraction of the total expenses for providing this service shall be distributed among each customer individually, i.e., every customer pays only for services rendered to him. This cannot be done by using the accounting data provided by the cloud providers.

```
Fig. 2. Example Application
```

In order to set up the usage accounting infrastructure using policy-based automation proposed here, the steps described in the following will take place. It is important to note that the service options shown in this example are only a subset of the available service options used to demonstrate the behavior of the policy-based automation.

1) Choosing an application: The first step is to choose an application for which the usage accounting infrastructure shall provide its data. In this example the application is cost
allocation. The requirements of cost allocation have been analyzed in section IV-B. As a result, initial policies for security, availability, and timing in the infrastructure are chosen and constraints for service options are defined, e.g.,

```plaintext
{  
  cacheTimeLimit : 06h00m00s,
  cachePersistence : yes,
  batchTransmission : yes,
  ...
}
```

2) **Manual adjustment of constraints:** The pre-configured constraints for the service options do not always satisfy the requirements in a specific environment. In our example the administrator thinks that the cache time limit is way too large and the administrator fears for data loss. Hence, this specific constraint can be adjusted, e.g.,

```plaintext
{  
  cacheTimeLimit : 00h30m00s,
  cachePersistence : yes,
  batchTransmission : yes,
  ...
}
```

3) **Evaluation process:** The input of the evaluation process consists of the (adjusted) constraints, the system status, and the capabilities of the accounting services. Based on a list of condition-action-rules, the evaluation process chooses a suitable configuration for the infrastructure. Because of the size of our example and the limited number of participating probes, the evaluation will choose the polling model for the general accounting model and define the service options accordingly.

4) **Deploying the infrastructure configuration:** When the probes and services of the accounting infrastructure come online, they will register with the Accounting Services and retrieve their configuration. This configuration is automatically established and the components are ready for the first requests.

5) **Monitoring the infrastructure:** During runtime, the whole infrastructure is monitored by the policy management. Non-compliance with the expected behavior is recognized and the administrator is informed. Such a non-compliance can be triggered by high network utilization resulting in a delayed transmission or by choosing unsuitable constraints, which the system cannot meet. In order to solve such a problem, the problem needs to be analyzed based on the information provided by the monitoring of the policy management. If any action on the accounting infrastructure must be taken, the administrator needs to manually adjust the constraints to suitable values. The evaluation process will start again and the resulting configuration will be deployed.

**H. Discussion**

Automating the configuration process of the usage accounting infrastructure provides several advantages. First and foremost, the effort to configure and maintain such an infrastructure is greatly reduced. Especially in an environment incorporating services from one or even many cloud infrastructures where service instances might come up or leave on a regular basis, automatic the configuration process can be a great help.

Secondly, the automation can lead to a less error-prone system configuration. This depends a lot on good condition-action-rules for the evaluation process, but if they are chosen carefully and properly the amount of configuration errors can be reduced. Additionally, the approach can even help to identify configuration errors, because the infrastructure is already monitored for the feedback evaluation process of new configurations. This data can be used for problem solving.

A third advantage can be properly chosen pre-configured condition-action-rules for the infrastructure. Administrators setting up such a solution have the advantage of benefiting from a best practice setup including rules defined by an expert.

Although this approach can greatly reduce the effort of setting up a usage accounting infrastructure, the resulting configurations can only be as good as the provided rules. Additionally, accounting for an highly complex service infrastructure can lead to very complex rules and also to difficulties identifying configuration errors. Hence, it is very important that such a system provides detailed information on the chosen configuration and on the evaluation process as well.

Currently, the process is semi-automated, but one can easily imagine the advantages of fully automating this process. After selecting a configuration for the accounting infrastructure’s components, the accounting infrastructure is monitored and feedback is provided. Making use of this feedback in an optimization process could improve the quality of the infrastructure constantly. It is important to keep in mind that such a fully automatic solution could bring several problems. On the one hand, the status of the whole infrastructure is changing constantly, e.g., network utilization and latency, if the trigger for re-evaluation of the configuration is not chosen very carefully, the system could easily be more concerned with re-configuration than with accounting. On the other hand, a fully automatic solution could lead to a less secure handling of usage data, i.e., personal data. If timing is considered more important than security, the system could automatically reduce security protocols to improve performance. Altogether, automation can be the next step for such an approach, but only together with careful consideration of possible risks.

**VII. Conclusion and Future Work**

Cloud Computing is an important basis for the provisioning of services and applications. Although cloud providers offer functionality for usage accounting, this functionality is limited to their own requirements different from the cloud user’s requirements. In order to run an infrastructure for usage accounting supporting a lot of different requirements with respect to the user as well as the applications, the underlying infrastructure needs to be highly adaptable. In order to be able to handle such an adaptable and complex infrastructure, the
people administering such an infrastructure need management support.

In this paper, we described a model for managing a usage accounting infrastructure by policies. Therefore, applications using accounting data have been analyzed with respect to three aspects: security, availability, and timing. These aspects are then used to define policies describing the demands of the applications without specifying means to achieve these requirements. In an evaluation process, these requirements are processed by condition-action-rules translating the requirements into technical methods for achieving them. Those technical methods are converted into component configurations of the accounting infrastructure, which can be manually adjusted by the system administrator, if desired. Then, these configurations are deployed automatically to the components participating in the usage accounting infrastructure. Additionally, the system is monitored to provide feedback for configurations derived in the future and providing information about the status of the accounting infrastructure.

Currently, this is a semi-automatic approach where feedback is gathered to improve the deriving of configurations in the future, but no action is taken automatically without human intervention. Making use of machine learning to ensure compliance with application requirements would be an interesting next step. By enabling the model to automatically adjust component configurations within the infrastructure, new ways can be found for efficiently managing complex systems.

REFERENCES


