PROCEEDINGS OF SCIENCE

The CONTRAIL approach to Cloud Federations*

Massimo Coppola, Patrizio Dazzi
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche
Pisa, Italy

Aliaksandr Lazouski, Fabio Martinelli, Paolo Mori
Istituto di Informatica e Telematica
Consiglio Nazionale delle Ricerche
Pisa, Italy

Jens Jensen†, Ian Johnson, Philip Kershaw
Science & Technology Facilities Council
Rutherford Appleton Laboratory
UK

The CONTRAIL project proposes a framework for Cloud Federations, that provides users with a single point of access to Cloud resources and relieves them from managing the credentials for access to individual Cloud service providers. The Federation services dynamically broker access to Cloud resources, ensures that the best available resource is selected, and that diverse resources are accessed consistently. This paper presents the CONTRAIL approach to federated identity management, focusing on authentication and authorization. In particular, we present "usage control" extensions to standard authorization frameworks to dynamically address changes in authorization decisions.

The International Symposium on Grids and Clouds (ISGC) 2012
Academia Sinica, Taipei, Taiwan
February 26 - March 2, 2012

*This work was supported by the FP7 projects *Open Computing Infrastructures for Elastic Services* (CONTRAIL) FP7-ICT 257438 and *Network of Excellence on Engineering Secure Future Internet Software Services and Systems* (NESSOS) FP7-ICT-2009-5 256980

†Speaker.

PoS(ISGC 2012)019

1. Introduction

Cloud computing technology promotes the provisioning of IT infrastructures as services. Cloud systems have become very common because of their advantages, such as the large amount of computational power available on demand, and the possibility of asking for further resources or releasing unused ones while the computation is running (Cloud elasticity).

Cloud computing facilities are currently provided by several big companies; among them Amazon[1], Google[2], IBM[3], Microsoft[4] and others. Alternatively, software is available for users that want to deploy a private Cloud in their own data centers, such as Eucalyptus[5] [10], OpenNebula[6] [11], and others. As an example, in the Amazon Elastic Compute Clouds (EC2), users request a number of virtual machines from the Cloud provider with certain features, such as the Instance Type (that defines the available computing power, the available memory and the local storage capacity), the operating system (e.g., Amazon Machine Image, AMI) and the network configuration (e.g., see Amazon Virtual Private Clouds, VPC). Users requiring large computational power can exploit resources provided by distinct Cloud providers to perform their tasks. However, in this case, the user has the responsibility of managing the creation of the resources, by sending a request to each of the providers and of interconnecting them with a proper network configuration.

The CONTRAIL project aims at defining and implementing a framework for Cloud Federations, to relieve the user from managing the access to individual Cloud service providers. Thus, users could focus on their work rather than on resource management, because the Federation brokers access to resources, ensures that the best available resource is selected, and that diverse resources are accessed consistently. The user simply defines usage policies: limits on the amount of resources to be consumed, privacy requirements, etc. The Federation then ensures that the resources are allocated as needed, matching the user's requirements. Beyond the user's view, it is also the role of the federation to bring together service providers and users. The cloud model makes resources available on-demand, and users don't have to negotiate directly with individual service providers because the Federation establishes contracts between them as needed. Nevertheless, the dynamic and elastic nature of clouds "disconnects" providers and users when there is no direct contractual agreement between them (or at least a credit card); it is also the role of the Federation to provide security to bridge this gap.

In fact, apart from performance and interoperability, one of the major concerns of the Cloud Federation is security. Cloud users use the resources brokered by the Federation to execute their applications and to store their data. Hence, users' data and applications should both be protected from unauthorized accesses and modifications. From the considerations of business computing, where users' data could have a high economic value, having an enhanced security support has the twofold goal of attracting new customers (thus increasing the Cloud provider revenue) and preventing data theft or corruption that could result in economic loss for the customer and hence

[1] http://aws.amazon.com/ec2
[2] http://code.google.com/appengine
[3] http://www.ibm.com/ibm/cloud
[4] http://www.microsoft.com/windowsazure
[5] http://eucalyptus.com
[6] http://opennebula.org

for the Cloud provider that could be asked for a refund. Besides the attacks coming from external entities, users' data and applications stored on Cloud resources should be protected also from the other users running on the same resources and from Cloud administrators.

This paper presents the CONTRAIL approach to Cloud. In particular, it describes the design and the features of the Cloud Federation, focusing on the security system which consists of a proper Identity Management system tailored for the Federation and of an enhanced Authorization system able to regulate the usage of Cloud resources.

2. Cloud Federation

From the user's point of view, a Federation can be seen as a bridge linking several cloud providers. In order to save the user the complexity and burden of dealing with more providers at the same time, a first step is to allow the most basic interoperation, developing equivalences among the various APIs of open-source and proprietary Clouds, as well as translators among all of their data format specifications.

The adoption of a federated approach encompasses but also goes beyond simple interface adaptation. A second part of the functionalities of a Federation is brokering, as seen in broker-based approaches such as SpotCloud[7]. SpotCloud allows the user to easily select one among several cloud providers. It implements a virtual resource market where demand and offer can meet, in order to optimize the overall resource exploitation (and the broker to earn its fair share).

However, brokers like SpotCloud still require all joining providers to adhere to a common API, the API choice restricting the scope of potential users. Brokers provide no support for applications that need to spread over more than a single provider, and cannot provide any guarantee on Service Levels past application deployment time. In summary, brokers address the issue of creating a resource market, but rarely provide ongoing management of resources, integration of diverse resources (e.g., PaaS and storage), or support providers with different APIs.

CONTRAIL Federations[4] overcome these limitations and aim at being full-fledged computing platforms. This is achieved by providing an even more comprehensive list of features.

Interoperability CONTRAIL targets the broadest integration of public and private Clouds, managed by both open source and proprietary software. The achievable degree of interoperation within the platform may be limited in some cases (e.g. whenever a provider lacks a specific functionality or property). Where a feature cannot be emulated, or cannot be enforced on a provider's resources, integration requires that the Federation is aware of the issue and can take this constraint into account in the resource selection phase. Interoperability will rely extensively on support of a broad set of normative and de-facto standards describing Cloud entities and operations, e.g. applications and virtual machines, as well as deployment and data access operations.

Scalability The implementation of CONTRAIL Federations makes it feasible to manage large platforms, both with respect to the number of providers, and with respect to the number of users, and active user applications. This requires a design where centralization points and

[7] http://www.spotcloud.com

bottlenecks are avoided. A key aspect of the platform scalability that CONTRAIL targets is the ability to run very large applications, even beyond the limits of a single provider, by splitting them over multiple Clouds and coordinating the different parts.

The problem of devising a proper plan for splitting applications and mapping the pieces over the available providers while obeying all application constraints is quite complex, out of the scope of this paper. In order to be feasible and efficient, execution of application over multiple providers needs to be carefully planned, and resource management must be as close as possible to the actual provider. CONTRAIL develops Federation-specific coordination strategies that combine SLA (Service Level Agreements), monitoring, and image deployment to manage split applications.

SLA management CONTRAIL Federations support flexible and extensible mechanisms for Service Level Agreement specification, negotiation and enforcing. The overall SLA framework is developed building on the results of the SLA@SOI European project[8].

The SLA@SOI multi-level SLA negotiation and enforcement framework provides the basic mechanisms that enable the Federation to hierarchically manage provider-specific SLAs. An instance of the SLA@SOI framework is associated with each provider, and a higher-level one with the Federation. Whenever a user starts an SLA negotiation, sub-negotiation can be performed with the providers in order to secure the needed resources and guarantee the outcome of the main SLA. Monitoring tools (and, when appropriate, enforcement mechanisms) are put in place at each provider by the local instance of the SLA@SOI framework.

The extensibility of the SLA@SOI framework is used to include terms in the Quality of Service (QoS) specification that also express Quality of Protection (QoP). QoP constraints are used by the Federation for resource selection to provide overall security guarantees to the users.

Trust and Security CONTRAIL is a trusted platform, as the providers joining the Federation are categorized in terms of the security features they support, and the infrastructure connecting them is crafted to prevent security breaches. CONTRAIL Federations isolate applications from each other and from the providers' environment via secured virtual networks and pervasive authentication and Authorization checks.

While in the following we will detail the technical solutions adopted, we underline here that they make the QoP specification in CONTRAIL quite powerful, including constraints on security protocols adopted and on the geographic location of resources, as well as on the dynamic behaviour and the reputation of users and providers.

Horizontal integration CONTRAIL Federations provide several types of IaaS resources (data, computation, network) in a homogeneous way and with SLA terms describing the QoS features. In order to guarantee the QoP parts of the SLA, secure virtual resources of each kind are developed as part of the CONTRAIL project, based on existing open-source tools.

[8] SLA@SOI web site, http://sla-at-soi.eu/

The "Virtual Execution Platform" (VEP) is based on OpenNebula; the "Virtual Infrastructure Network" (VIN) exploits IPSec to build large-scale secure networks, and the "Global Autonomous File System" (GAFS) builds on the XtreemFS distributed file system.

Vertical integration In this paper we have focused on CONTRAIL IaaS implementation and definition. We thus only mention the layer of PaaS services (ConPaaS [13]) that make use of the IaaS Federation layer in order to access computing resources. Security goals can be guaranteed only by vertical integration of security between the PaaS, Federation, and the provider-IaaS layers.

2.1 Supporting CONTRAIL Federations

In order to achieve the features outlined so far, each CONTRAIL Federation has is implemented in a distributed infrastructure, where multiple *Federation Access Points* are scattered over the network, each one potentially in contact with all Cloud providers in the Federation. As such access points will be trusted gateways (in particular, they have access to the Federation user database), they will have to be run by trusted participant institutions (and to be identified with X.509 certificates: one for the front end of the service, and another to identify the federation as a part of the CONTRAIL infrastructure). Alternatively, an access point could be run by a participating institution for its own users, in which case true single sign-on could be implemented, but it would still need to share user database with the other access points, or we would lose the ability to manage attributes across the whole Federation.

Figure 1 shows the block decomposition of one of the access points.

The access points share databases (by replication) and a common set of policies used to manage the Federation. The databases contain the knowledge about Federation entities (e.g. users, providers), including federation-level user attributes (community memberships, community roles, and even roles in the federation itself) where the federation acts as an attribute authority on behalf of the user communities[9]. Consistency is necessary to ensure that each access point will take the same resource placement decisions, including enforcement of security goals.

On the other hand, each access point actively communicates with the Cloud providers hosting applications submitted by that access point. This strategy allows effective management of active applications via streams of monitoring data, and employs less frequent, cheaper communications between the access points for spreading non-critical, generic information on the status of the platform.

For the sake of modularity and flexibility, Federation access points are a composition of software components from the Tuscany framework[10], which is the implementation of the the Apache Foundation "Service Component Architecture."

[9]There are plans to enable CONTRAIL to consume external attributes from attribute authorities already in use by our user communities, but this requires mapping an attribute assigned to an external identity into one assigned to the federation identity, and possibly renaming the attribute itself because different communities may have the same name (e.g., "admin") for different roles. For now, the attributes relevant to the cloud resources are replicated (currently manually) by community managers in the CONTRAIL attribute authority database.

[10]http://tuscany.apache.org/

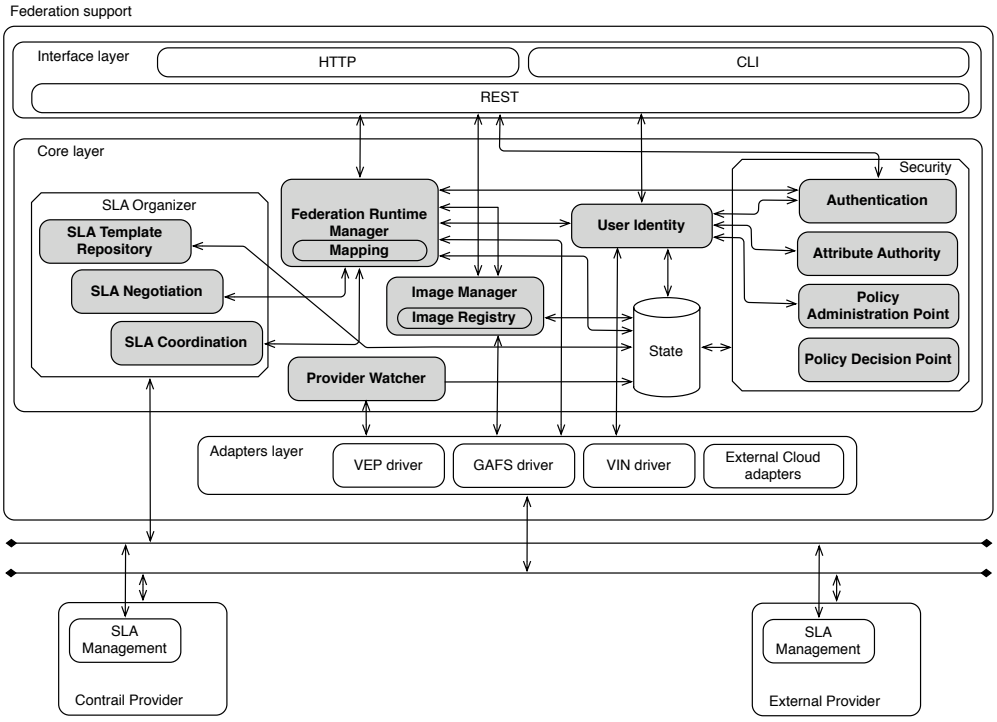

Figure 1: Architecture of a Federation Access Point.

In Figure 1 we see that each access point has its own interfaces for user interaction. They are built with REST web services, and there is an adapter layer where drivers for each available IaaS provider are instantiated.

The modules implementing security are described below, in the following section. Another set of modules performs the management of SLAs at the Federation level. Since these modules are based on the SLA@SOI software architecture, they also provide Web Services interfaces.

Of the remaining modules, the Federation Runtime takes care of almost all of the application lifecycle. The Image Manager stores image metadata for the virtual machines to be deployed, and the Provider Watcher gathers statistics about Cloud Providers. The User Identity module provides persistent storage for user-related data. Finally, the State module gathers information related to all Federation entities. The task of the State Module, in a Federation with multiple access points, includes spreading information to other access points by employing algorithms that vary according to the kind of data. Atomic transactions must be used for user identities, while slow "gossip protocols" are fine for load statistics of the Cloud providers.

As a final remark, we note that since each Federation access point and each Cloud provider are independent separate systems over the Internet, authentication and authorization checks are done by all interfaces.

3. Federation Security

Security is one of the major concerns of the Cloud environment. First of all, security is an

issue that affects the uptake of Cloud resources, because potential customers will employ Cloud services for their business only if they believe that the Cloud is secure and dependable.

The Federation on one hand simplifies the usage of Cloud services, but on the other hand introduces new security issues. As a matter of fact, the Federation services are considered as critical resources, that store critical data about Federation users, hence requiring a proper security evaluations. They also need to be identified to each other, within the federation, with X.509 certificates, but these certificates do not have to be dynamically generated as federation services do not need to be elastic, and the trust in the infrastructure is ultimately based on their management by qualified administrators. The CA issuing these certificates is visible only within the CONTRAIL federation, so certificates need not be issued by a commercial CA. In turn, they are used to bootstrap the trust in the dynamically generated certificates for elastic services, but the full description of this is quite long and beyond the scope of this paper.

3.1 Authentication

A means of authentication for users of the system is essential to secure access to Cloud resources and to monitor their usage. This authentication system must manage access across each of the Cloud providers which it federates. Each federation represents an independent security domain with its own set of user credentials. From a user perspective, if the resources in these domains are to be accessed seamlessly then the credentials must be managed collectively at the federation level. CONTRAIL approaches this problem by defining a single persistent federation level identity which maps to Cloud provider credentials under its control. In doing so, it provides a form of single sign-on, since when the user signs on at federation level, they are effectively signed on within all Cloud provider domains under the Federation's control (but may not have permission, or may have security constraints, preventing them from actually accessig them.) Delegation is also implemented, since the user delegates to the Federation the ability to invoke cloud resources on their behalf, and possibly further to resources the ability to perform certain restricted tasks, e.g., mounting GAFS (the data infrastructure.) We have expediently focused on "identity delegation" where a credential representing the user is created with the remote service; however, the authorization to obtain such a token is implemented with OAuth2. Delegating the "identity" has to be done carefully, and in any case, the user must trust the federation layer which manages and audits this delegation. The advantage of this approach, beyond the fact that it works, is that the credential is alive during a well-defined lifetime, and we do not have to rely on caches or on asking for authorization every time a decision is required; moreover, we provide a sophisticated revocation feature, revoking the original authorisation decision based on the attributes embedded within the credential (section 3.2.)

Depending on the supported authentication interfaces of the underlying Cloud provider, the Federation may also need to store sensitive information such as passwords. This can be mitigated to some degree by carefully isolating and auditing access to such sensitive credentials; they are only stored in the module which directly interfaces to the given underlying Cloud provider. In the case of OpenNebula, the baseline Cloud provider for CONTRAIL, it has been possible to develop a delegation mechanism without the need to store sensitive credentials. We explore this in more depth later in this section. Note also that by storing resource credentials within the Federation and logging the accounting data, we are free to map users to resource credentials as we need:

e.g., a one-to-one mapping, a pool of accounts, or a resource account shared by a community (as determined by Federation attributes.)

Users authenticate to the federation layer using different methods dependent on the context. Two distinct methods are supported: with a web browser or through a non-browser based client such as a console script or rich client application. In the case of browser-based user agents, the CONTRAIL web frontend application will support authentication with the Federation identity and password. In addition, single sign-on is supported to enable the use of external identities from other Identity Providers (IdPs). OpenID[11] currently works; Shibboleth[12] is likely to be supported via a WFAYF mechanism[13]. Once signed in, the given identity will be mapped to the CONTRAIL Federation identity.

In the case of console or rich client applications, authentication is by means of a short-lived "End Entity Certificate" (EEC.) This certificate is obtained from an online CA (Certification Authority) web service associated with the Federation IdP. The process to obtain a credential is as follows: the user agent bootstraps the process by downloading the CA trust roots in order to correctly verify the online CA over HTTPS. With this in place it generates a public/private key pair. The public key is added to a certificate request and is passed to the online CA over a HTTPS web service interface. Along with the request, the client also passes their federation user identity and password using the HTTP Basic Auth method (RFC 2617). Once the certificate has been obtained it can be used to authenticate with any service in the Federation. Again we are using RESTful web services for simplicity.

The choice of authentication technologies was determined after careful consideration during the architectural phase. As a design principle, REST[6] was adopted for all web service interactions both within the federation and for APIs exposed to the outside. Unfortunately, there is no single established standard authentication mechanism built over REST. At the HTTP level, security artifacts may be passed in the URI, the message header, or the message body. The latter can be immediately ruled out because not all HTTP methods include a request body (e.g. HTTP GET). Passing arguments in the URI breaks the principles of REST since stateful information is being passed, as is information which is not related to the underlying resource which the URI identifies. As the last of the HTTP authentication methods, the message header is used by a number of different specifications for passing security information, e.g., HTTP BasicAuth / DigestAuth (RFC 2617), OAuth 1.0[8] and HTTPsec[7]. The limitations on header size for some HTTP server implementations preclude this as a means for passing the larger authentication tokens that may be required in CONTRAIL.

Instead, we chose the socket-based approach, using certificates. Secured services run over HTTPS and clients authenticating with them pass a user short-lived EEC in the SSL handshake. The user's EEC is acceptable to any service in the federation provided it is signed by the correct CA, thus meeting the single sign-on requirement. Furthermore, we embedd custom extensions in certificates carrying user attribute information in SAML format, since SAML provides an extensible format for providing not just attribute statements but other pertinent information such as

[11] http://openid.net/

[12] http://shibboleth.internet2.edu/

[13] "Which Federation Are You From," as used by, e.g., XtreemOS VOLife and the Terena Certificate Service: users are directed to their national federation and in a second redirection to their home institute.

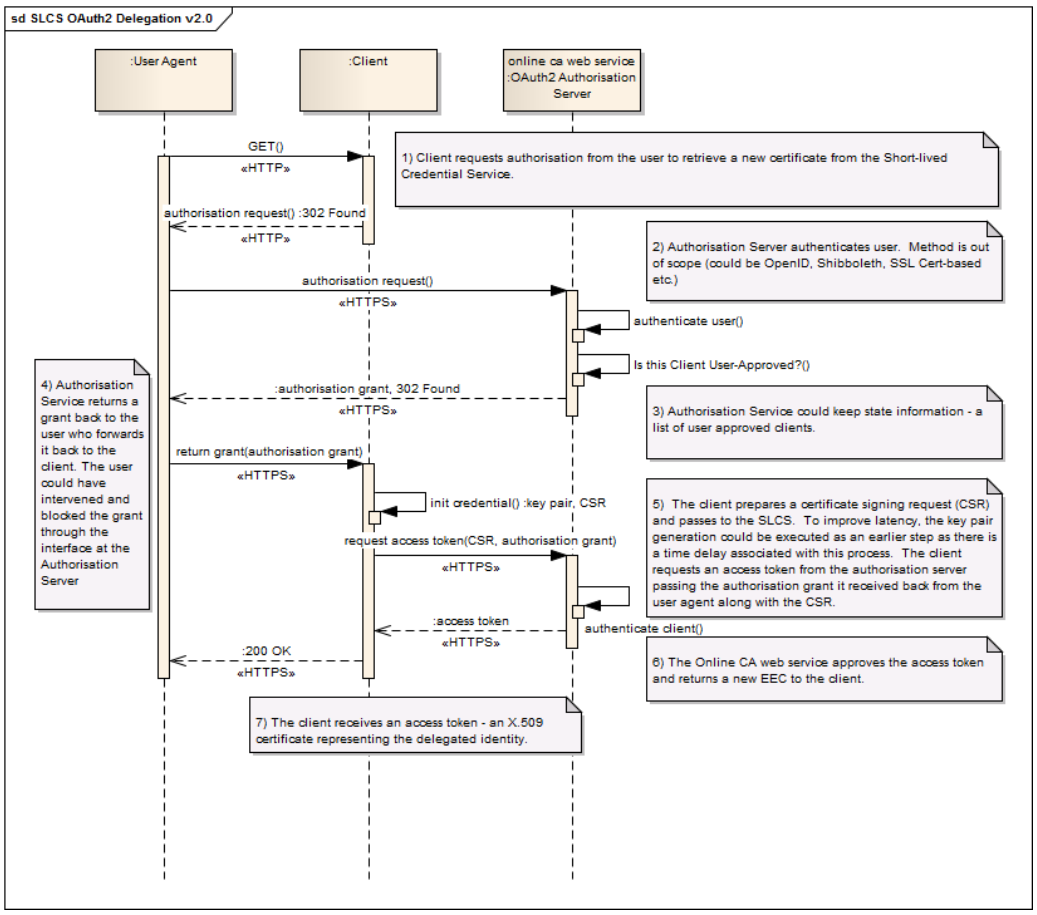

Figure 2: OAuth 2.0 based delegation for Online CA Service

authentication context and perhaps authorisation scope. This technique has been employed on a number of other projects including GridShib[2] and ESGF[9]. Since multiple authentication methods are supported in the system, a record of the associated level of assurance should be also be included. The SAML Authentication Context provides a means of expressing this. Consumers of the certificate (services) can then check the context statement and see if the level of assurance provided is sufficient to execute a given action.

Finally, for service to service interactions within the Federation, some services need to authenticate on behalf of a user in order to access other CONTRAIL services. As mentioned earlier, we chose to "delegate" a full (but short-lived) user credential based on an online CA; but as a mitigating factor basing the authorisation on OAuth2 (described in the next section). This approach was pioneered by the CILogon[3] project and eliminates the need for RFC 3820 proxy certificates [15] which are otherwise used in grids for similar "delegation." A HTTP-based Online CA service is fronted with an OAuth interface shown in figure 2. CILogon used the OAuth 1.0 protocol. Version 2.0 has some significant simplification of the information flows, so CONTRAIL contributed to the development of a python OAuth2 library.

Here, OAuth2 is used by a remote entity to get permission to obtain a delegated credential on behalf of the user – which can then be used until it expires. This credential is itself an EEC

and so no specialised SSL middleware is needed to correctly process it at consumers, as would be required for proxies[14]. Proxy certificates do provide provenance information in the form of the delegation certificate chain back to the original issuing EEC, and we would like to retain this information. In CONTRAIL this provenance requirement is simplified somewhat because we only have a single level of delegation. The EEC issued from the online CA for delegation will include a SAML authentication statement in its extension asserting that the given OAuth client successfully authenticated, and of course the process is logged by the online CA.

3.2 Authorization

Authorization is the process of deciding whether a given (authenticated) user has the right to perform an action on a given resource (according to pre-defined *access control policies*.)

The federated Cloud environment has a number of peculiarities that impact on the authorization system, many of which were not seen before with grids and other forms of large-scale distributed computing:

- There is an increased disconnect between the provider of the resoure and the consumer (the end user). As with grids, the provider may agree to provide resources to a community, but in a dynamic "cloudy" environment, the connection from the user to the community, and the community to the provider is weaker than the more static models in the grid (compare CSA threat 1, [1].) It may make sense to make use of one of the *reputation* based models, communicating the reputation as a user attribute (which is one of the use cases for the usage control model described below.)

- As the Federation is open to everyone – anyone can request an account – the levels of assurance of the credentials with which they authenticate will have to be taken into account. In the grid world, authentication has focused on the use of X.509 certificates with fairly strong identity assertions; in a CONTRAIL Federation, there may be a mix of high and low levels of assurance.

- In particular, the *translation* of identity credential to a federation credential, and possibly the associated transfer of attributes associated with the external credential to one with similar semantics and interpretation for the federation credential, will require careful management and tracking of the level of assurance.

- As with grids, resource access can last a long time. In grids, if the user's access rights change, the next request will (should) fail (it may happen only when the VOMS assertion and/or RFC 3820 proxy expires), but the current request may continue to execute[15] Alternatively, the user's credential (certificate) will have to be revoked. We propose here a model for Clouds with a much improved response rate to changes in the authorization.

Of course, systems have already been developed to address some of the need for trust in distributed infrastructures, such as WS-Trust from OASIS. In this sense, the Federation is equivalent

[14] OpenSSL comes with support for RFC 3820 proxies but this must be enabled at compile-time; however, the Linux distributions usually leave it compiled out.

[15] In fact it will continue to execute, e.g., a job running, but its subsequent writing of data should fail.

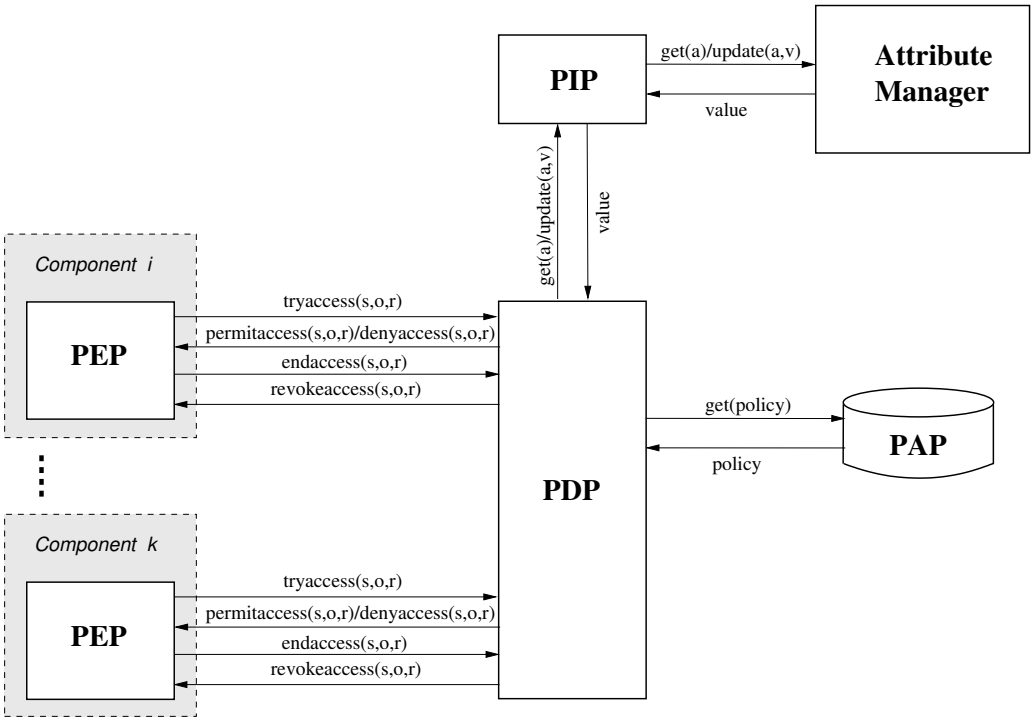

Figure 3: Architecture of the Authorization System.

to an STS[16] in WS-Trust, as there is a direct link from all users to a single Federation, and from there to the providers. In general, however, the CONTRAIL federation model is a simplification compared to WS-Trust; for example, the communication pattern is more rigid, and the federation service normally always holds the initial security token. (Another difference is that we are mainly using REST rather than SOAP.)

In order to address these peculiarities, the authorization system developed for the CONTRAIL framework makes use of the Usage Control (UCON) model [14], [16]. UCON is based on the concept of *mutable attributes* and *continuous control*. Mutable attributes are associated with either users or resources, and may change their value as a consequence of the actions that are performed. When the value of attributes change, the access control decision is re-evaluated against the policy, because the changed attribute may revoke the permission to perform the action (it is of course also possible that the *policy* has changed; also in this case does it make sense to re-evaluate the access control decisions.) Hence, the difference with respect "traditional" access control models is that in UCON, an action can also be stopped while in progress (alternatively, other remedial actions may be possible, described briefly below.) Contrast this with the "traditional" approach where the access control decision is taken initially (such as for the grids mentioned above.).

The main components of the CONTRAIL authorization system architecture and their interactions are shown in Figure 3. The architecture is based on a set of Policy Enforcement Points (PEPs), a Policy Decision Point (PDP), a Policy Information Point (PIP), and a Policy Administration Point (PAP), as defined in the XACML reference model [12].

[16] Secure Token Service.

PEPs are (obviously) integrated within the CONTRAIL components that need to be controlled. An important feature of a PEP is that it must be non-bypassable and tamper-proof. Hence, a PEP associated with a service performing a security relevant action must be invoked every time that action is executed. A PEP must be able to:

- intercept the invocations of security relevant actions, e.g. the user attempts to perform an action on some resource;
- determine when the action has completed;
- prevent, permit, a requested security relevant action based on the decision from the PDP;
- suspend, resume, or terminate the execution of an action which has already started (this is a CONTRAIL addition to the normal duties of a PEP);
- communicate securely with the PDP (which is normally initiated by the PEP but in CONTRAIL the communication can also be initiated by the PDP).

The PDP initially obtains its security policy from the Policy Administration Point (PAP) repository, and builds its internal data structures for the policy representation. The policy is written in the "U-XACML" language [5], an extension of the well known XACML [12] policy language, to cover UCON.

Communications from the PEP to the PDP are implemented through two request commands: $tryaccess(s,o,r)$ and $endaccess(s,o,r)$ (as specified in [16]), where s represents the subject that wants to perform the action (including associated attributes), o represents the resource (object) that the subject wants to access, and r represent the specific action (with parameters) that the subject wants to execute on the resource.

The execution of the operation is of course permitted only after a positive response is received from the PDP, through the $permitaccess(s,o,r)$ response. If the PEP instead receives a $denyaccess(s,o,r)$ response, the requested action is (obviously) not executed. As this decision is likely to happen shortly after the request is made, it is natural to communicate this failure back to the user or agent making the request.

As we have mentioned above, in UCON a decision grant may be reversed: if the right to perform a particular action is granted by the PDP, and the action is launched (or rather, permitted) by the PEP, the PDP sets up a *subscription* process whereby it may be *notified* of changes to the attributes it believes are relevant to the authorization decision. Should the value change, the PDP will re-evaluate the access control decision, and if the outcome of the decision is the reverse, it will notify the PEP, stating also which remedial action it requests from the PEP. This may usually be to *terminate* the user's action, but could also be to *suspend* it and alert an administrator to investigate.

If a specific action is *denied* (upon original request), *terminated* (by being killed by the PEP), or *completed*, then the outcome of the action is logged by the PEP, and the action can no longer be performed – these states are final. To perform the same action again would require a new request from the user (or agent); the action cannot be restarted from a final state by the PEP or PDP. At this final state, the PEP will send $endaccess(s,o,r)$ to the PDP to signal that the PDP need no longer track the decision for this action. In turn, this may lead the PDP to cancel notifications for the attributes relevant to the decision.

How does the user know whether their actions have been suspended or terminated? At present, we have no notification mechanisms for users beyond the accounting records (which contain only final state notifications) and the general status request of an action via the federation front end (which currently requires the user to refresh the status.) However, it may be useful in the future to introduce more active notifications; in which case the simplest option will be to integrate it with UCON (e.g., "terminate and notify user.")

4. Conclusion

In this paper, we have presented work done in the CONTRAIL project to develop federated identity management for clouds. Architecturally, we have chosen to have a single front end for all services, a federation access point (although in practice there will be many such even within the same federation, they will be replicated). Access points are able to consume several different types of identities from external identity providers (although in the current release the focus is on the integration with OpenID), and also provide internal username/password authentication for users unable or unwilling to make use of an existing identity. Apart from authentication, the access points also serve as front ends for account management, as well as portals for management of cloud resources. Users are not tied to portals, however; we have tools to log in and obtain credentials for command line work.

Work in the near future will see the integration of Shibboleth, thus leading to a dual-federation system: users log in with their federation credential (from their Shibboleth federation), and obtain a federation credential, meaningful to the federation of cloud service providers in CONTRAIL.

Once logged in, a temporary X.509 certificate is generated for users; a technique that has been used in science portals and gateways for many years: it removes the need for managing certificates from end users, yet provides good security and the technology works with many types of resources. In addition, CONTRAIL embeds user attributes into the certificate, attributes which are currently maintained in the federation accounts database, but there are plans to support external attribute authorities as well (this topic is non-trivial and deserves a paper in its own right.)

Once received by the resource, the attributes are used for access control decisions, using an extension of the well-known XACML model to enable *usage control*, UCON. UCON is implemented via a set of notification mechanisms between the attribute authority and the PDP, and between the PDP and the PEP, thus ultimately linking the enforcement of the action to attributes which may change, making the XACML model more dynamic. The advantage of UCON is to provide a more rapid signaling of changes to the user's roles (or more woolly concepts like reputation), leading to immediate re-evaluation of access control decisions involving the user. We believe that this helps addressing concerns arising from the looser binding between users and the resource provider, because a user's activities can be suspended or terminated almost immediately if needed. Conversely, the reputation of the resource provider, or specifically their Quality of Protection, will be expressed via the SLA negotiation mechanism, and the Federation will match this against the user's requirements.

In short, we have addressed the need for federated identity management in large scale, dynamic projects, with elastic resource provisioning. While building on existing technologies, the

integration presented in this paper, combined with the CONTRAIL extensions, will enable a new range of dynamic federations for elastic resources.

References

[1] Cloud Security Alliance, editor. *Top Threats to Cloud Computing (v.1.0)*. https://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf, March 2010.

[2] T. Barton, J. Basney, T. Freeman, T. Scavo, F. Siebenlist, V. Welch, R. Ananthakrishnan, B. Baker, M. Goode, and K. Keahey. Identity federation and attribute-based authorization through the globus toolkit, shibboleth, gridshib, and myproxy. *5th Annual PKI R and D Workshop*, 2006.

[3] J. Basney and J. Gaynor. An oauth service for issuing certificates to science gateways for teragrid users. *TeraGrid Conference, Salt Lake City, UT.*, July 2011.

[4] Emanuele Carlini, Patrizio Dazzi, Giacomo Righetti, Massimo Coppola, and Laura Ricci. Cloud federations in contrail. In *Proc. of CoreGRID ERCIM Workshop on GRids, P2P and Service Comput ing (CGWS 2011)*, Bordeaux, France, 2011.

[5] M. Colombo, A. Lazouski, F. Martinelli, and P. Mori. A proposal on enhancing XACML with continuous usage control features. In *Proceeedings of CoreGRID ERCIM Working Group Workshop on Grids, P2P and Service computing*, pages 133–146. Springer US, 2010.

[6] R.T. Fielding. Representational state transfer (rest), architectural styles and the design of network-based software architectures. http://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm, 2000.

[7] S. Fowler. Httpsec authentication protocol. http://www.httpsec.com/1.0/, September 2006.

[8] E. Hammer-Lahav. The oauth 1.0 protocol. http://tools.ietf.org/html/rfc5849, April 2010.

[9] P. Kershaw, R. Ananthakrishnan, L. Cinquini, D. Heimbigner, and B. Lawrence. A modular access control architecture for the earth system grid federation. *The 2011 International Conference on Grid Computing and Applications*, July 2011.

[10] I. Khan, H. Rehman, and Z. Anwar. Design and deployment of a trusted eucalyptus cloud. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 380 –387, july 2011.

[11] D.S. Milojicic, I.M. Llorente, and R.S. Montero. Opennebula: A cloud management tool. *IEEE Internet Computing*, 15(2):11–14, 2011.

[12] OASIS. extensible access control markup language (xacml) version 3.0. Technical report, OASIS Standard, 2010. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf.

[13] Guillaume Pierre, Ismail El Helw, Corina Stratan, Ana Oprescu, Thilo Kielmann, Thorsten Schütt, Matej Artač, and Aleş Černivec. Conpaas: an integrated runtime environment for elastic cloud applications. In *Proceedings of the ACM/IFIP/USENIX 12th Middleware conference*, Lisbon, Portugal, December 2011.

[14] R. Sandhu and J. Park. The UCON_{ABC} usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[15] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet x.509 public key infrastructure (pki) proxy certificate profile, 2004.

[16] X. Zhang, F. Parisi-Presicce, R. Sandhu, and J. Park. Formal model and policy specification of usage control. *ACM Transactions on Information and System Security (TISSEC)*, 8(4):351–387, 2005.