Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 2, February 2015, pg.272 – 279

RESEARCH ARTICLE

SECURE CLOUD CONSISTENCY BASED ON SLA SYSTEM USING KPI

K.Yuvaraj[1], S.Nusrath Fathima[2], T.Karthikeyan[3], P. Priyadharshini[4], Dr. P.Gomathi[5]

[1]IV Year Student, Dept. of Computer Science and Engineering, N.S.N. College of Engineering and Technology, TN, India
[2]IV Year Student, Dept. of Computer Science and Engineering, N.S.N. College of Engineering and Technology, TN, India
[3]HOD/CSE, N.S.N. College of Engineering and Technology, TN, India
[4]AP/CSE, N.S.N. College of Engineering and Technology, TN, India
[5]Dean, N.S.N. College of Engineering and Technology, TN, India

[1] yuvarajk_cse2015@nsn.ac.in, [2] nusrathfathimas_cse2015@nsn.ac.in, [3] hodcse@nsn.ac.in,
[4] priyadharshinip_cse@nsn.ac.in, [5] dean@nsn.ac.in

Abstract — The cloud computation is a recently evolved computing terminology at variance capabilities are provided as a service to customers using internet technologies. The most common offered services are Infrastructure (IaaS), Software (SaaS) and Platform (PaaS). This work segregates the service management into the cloud computing idea and shows how management can be provided as a service in the cloud. Nowadays, services need to adapt their functionality across heterogeneous environments with different technology and administrative domains. The implied elaborations of this location can be simplified by service management architecture in the cloud. This paper focuses on this composition technique, taking into account specific service management operatives, like incident management or KPI/SLA handling, and provides a complete solution. The proposed architecture is based on a distributed set of agents, using semantic-based techniques a Shared Knowledge Plane, instantiated in the cloud, has been initiated to ensure communication between agents.

Keywords — Service Management, Multi-Domain, Multi-Provider, Semantics, Cloud Computing, Shared Knowledge Plane

I. Introduction

The service concept is being currently used to define a wide range of approaches. Nowadays, everybody can get an application from an application store and start to run it in their devices. Behind this kind of service approach could be something as simple as a web service which requests information from a database. Even though more complex services provided by applications which use different network capabilities provided by different vendors. Both approaches could be understood as services, but also they have a different degree of complexity.

Besides access to network capabilities, these services are usually based on a complex cloud architecture that manages the infrastructure and platforms where software applications from different providers are running.

This complex infrastructure could be distributed in different locations of different actors linked by communications networks (e.g. services provided by Content Delivery Networks, Amazon's based services, etc.). A review of some features of these universal services can help to find their management requirements:

First, the number of network domains associated with global services or customers could be too high and change at any time. Thus, the management solutions related to this new approach require greater dynamism than conventional solutions. Additionally, the number of actors included in the service value chain could be very large so it could be possible to find a lot of partners who provide different functionalities like connectivity, software infrastructure or contents. With this environment it could be difficult to manage the part of the service provided by other parties.

Finally, the global service could be deployed in various locations. Although the current trend is to use cloud computing in any of its possibilities (private, public and hybrid cloud), it can also be implemented in non-virtualized nodes. Classic management systems are usually associated with their domains: each network segment has its own service management system that detects, chunk, alert problems or provides Key Performance Indicators (KPIs) and Service Level Agreements (SLAs) to the human network or platform operators in charge of keeping the service running effectively and determining the suitable actions in order to correct problems found in the service. Many commercial solutions cover this single-domain situation. For the multi-domain environment, current solutions are based on creating an "umbrella system" that receives events and collects data from each domain to perform integrated service management. These results are too static and have a monolithic data structure that defines the system operation. Adding a new domain is a hard task. Thus, current management systems cannot easily face the challenges introduced by multi-provider and/or multi-domain scenarios (e.g. they cannot automatically recognize nor correlate new events).

It is placed on the definition of an enlarge distributed agents structure to manage these new kindly of services, which have collaborator different technological and administrative environments (multi-technological/multi-domain).

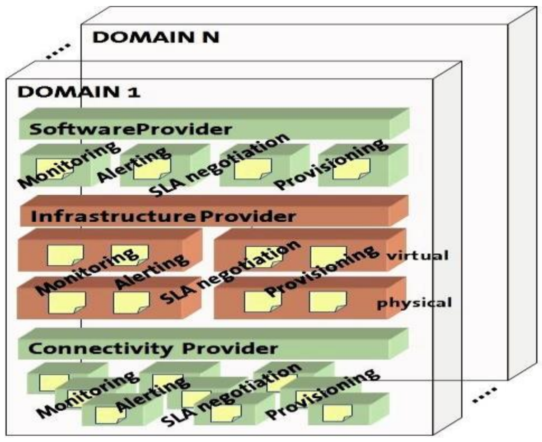

Figure 1 Service Management Domain

The items are a set of agents which are located in networks or services elements and the management is composed by different functions like incidents, problems, job scheduling or KPI/SLA management. Besides being able to address the complexity of service management, this architecture is easily scalable, on-demand and easy to use.

II. Related Work

The environment in clouds is much more dynamic and that there is no clear mapping between physical infrastructure and location of a service considering dynamic placement of VMs and replacement. More recently it has been developed a set of management tools focused on different management aspects and based on open source code. Thus, it has been created Nagios [1] which offers complete monitoring and alerting for servers or network device in real time including logs, configuration files, messages, alerts, scripts and metrics or Jasper reports [3] which is one of the most popular business intelligence and report engines. These tools provide isolated solutions for each management aspect but none of them provide a whole architecture for a complete service management.

Distributed Management Task Force (DMTF) [4] has been working in the definition of management technologies in the enterprise management area. So DMTF standardized several technologies like the Web Based Enterprise Management (WBEM) [5], while promoting a vision of integrated management and including the management support for network equipment. Recently, the Cloud Management Working Group has been created within DTMF. This group is focused on standardizing interactions between cloud environments, and has published the Cloud Infrastructure Management Interface (CIMI) as an interface that allows cloud users to dynamically provision, configure and administer their cloud usage.

Different approaches have been developed in the fault localization problem and diagnosis, and different techniques have been used including neuronal networks and decision trees [6], probabilistic methods [7]. About SLA management, there are multiple research activities, but none of them provides an overall solution addressing all the management aspects. The SLA@SOI project [8], presents a holistic view for the management of service level agreements (SLAs) in a multilevel environment and implements a framework that can be easily integrated into a service-oriented infrastructure (SOI) but it fails to integrate network management aspects.

The proposed a solution for managing service providers with a framework for creating application specific VMs on demand, including global resource allocation among nodes, and SLA-driven dynamic resource redistribution. Our work proposes a more general service management covering several aspects and not only SLA or cloud infrastructure management (like incident or problem).

III. Service Level Agreement Architecture

As it has been introduced in the previous sections, the current commercial tools do not cover the management needs of the new global services. So, this work provides an architecture on which new management tools could be based in order to simplify the service operation. The agents are running in autonomous way but the behavior of each of them affects to the rest.

This architecture is based on a previous work defined in. This was a first approach of the architecture in order to study the fault management for multi domain technological services. This work provides a more complete architecture addressing the holistic service management, covering several management aspect and their relationships, through both technological and administrative environments.

This architecture is based on two concepts. The first key aspect of this architecture is the application environment. The autonomous agents are distributed in different environments, both technological and administrative, and dedicated to different management services. So, technological environment refers to the technology field in which every segment/functionality of the global service is based. Service management could be defined as the different functions that are included in the management of the global services (i.e. supervision, incident, problem, SLA ...). Thus, each domain will be associated with the set: technological environment, administrative environment and service management.

The other key aspect of this architecture is the Shared Knowledge. It is the capability of each autonomous agent to share its experiences into its environment with the rest of the agents.

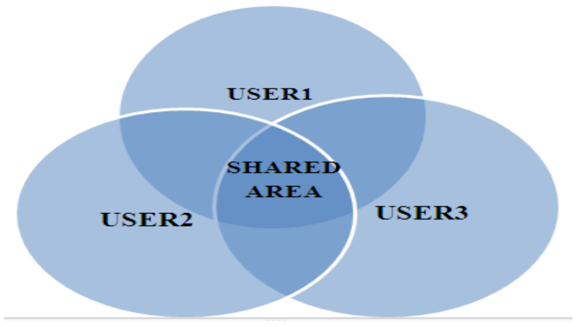

Figure 2 Shared Knowledge Plane

Each agent is associated with one management function on a specific domain, defined by the above mentioned set. In addition, each agent will have an associated knowledge plane of its domain. The union of the knowledge planes of all domains leads to the concept of the Shared Knowledge Plane (SKP). So, each agent will act on a plane of knowledge base determined by its domain of action. Figure 3 shows the result of the union of these planes.

The architecture has associated a method to communicate each agent with its domain in the "knowledge cloud." An agent can update/modify both the area of the shared knowledge plane which only affects to it (specific area) and the area which affects to more agents (collective area). If a modification affects the collective area, the rest of the agents related with this area must be notified. Thus, a method for the synchronization of the knowledge among the agents has been defined.

The suggested architecture is based on a set of distributed agents and a SKP. The knowledge plane is a dis-tributed and decentralized concept that manages information about the service environment, based on ontology, and provides an integrated view to all agents.

A. Shared Knowledge Plane

The fundamental element of the above architecture is the Shared Knowledge Plane. It contains the semantic model, the set of instances and the reasoning rules which are offered to the agents to govern their behaviour. The fact that the domain description and the management rules are enclosed within the ontology reduces the complexity of the agents.

The definition of the Shared Knowledge Plane requires two steps. The first step is the design of a conceptual model describing the domain, and its associated ontology. A knowledge representation language like OWL (Ontology Web Language) [9] can be used for that purpose. The second step is the definition of the rules governing the system behaviour, using SWRL was chosen because of its flexibility and extensibility, which enable the development of proper reasoning techniques. The SKP is used by the agents to communicate between them, perceiving each other's' behavior using Internet capabilities.

In traditional management systems, service management is associated with the fulfillment of a rule that includes conditions from different domains. In the proposed architecture this rule is split into several ones in order to simplify the multi-domain service management. If an input event is only associated with one domain, it modifies the Knowledge plane associated with this domain and the common part. The incorporation of a new domain to the system is associated with the modification of the semantic model or the inclusion of new rules. It does not imply any change of code in the software agent. In addition, if the model has been correctly defined, the modification will be minor because the concepts should be similar.

B. Distributed Agents

The autonomous agent is responsible for the application of knowledge to manage a global service in its domain. Its main functions are:

•Facilitate the management aspect assigned to it.
•Manage its own behavior based on a set of rules.
•Interact with other autonomous agents to provide global solutions.

Agents of the proposed architecture follow the knowledge elements can be shared by different agents.

Agents Monitor module captures information from its managed resources, network or service elements included in each domain, and modifies the Shared Knowledge Plane incorporating them as new instances. The Analysis module is responsible for the reasoning function.

Summarizing, the agent is a software component which is allocated to a given domain and has the capability to infer –using information taken from the context and plan concrete actions upon reception of events coming from different domains. This inference process takes place to find out the appropriate actions for solving the problem. Once the inference process has been completed, the agent plans how to carry out the appropriate management actions. These actions can imply an operation that affects the network or service elements, but also can trigger the updating of the domain-specific knowledge plane, which actually means the modification of the global SKP.

Summarizing, the agent is a software component which is allocated to a given domain and has the capability to infer –using information taken from the context– and plan concrete actions upon reception of events coming from different do-mains.

C. Agent Interactions

In order to get a correct management of the global services the agents need to swap information. Thus, they should recurrently interact to share knowledge and perform tasks to achieve their goals.

The interaction is generally defined by Communication Language: This is the medium through which the content of the message is communicated. All the agents of this architecture can communicate with other agents easily, following a common language. Agents use as common language an ontology which is understood by all the agents.

Interaction protocol: This refers to the high level strategy pursued by the agents that governs its interactions with other agents. The method used in this architecture is a variation of a blackboard communication architecture where the shared information is made available to all agents and there is no direct communication be-tween the agents. Also, it has to provide freedom to the agents regarding the information included in the specific area, each agent has a version of its subset of the ontology in order to act in an autonomous way.

Transport protocol: This is the actual mechanism used for communication. In the proposed architecture, since ontologies are published on a web server with the same mechanism similar to the web 2.0 wikis, the transport protocol is HTTP.

The following algorithm provides the interaction between the agents and the SKP. When the data received is from the collective area, a method to avoid concurrent access to the KB and SKP is implemented.

Algorithm 1

```
do forever { data Reception
    if DomainInfo belong {SpecificArea$_{SKP}$} then update KB$_{SKP}$
        run AGENT reasoning engine
    end if
    if DomainInfo belong {CollectiveArea$_{SKP}$} then CheckCollision
        update KB$_{SKP}$
        run AGENT reasoning engine
    end if
}
```

D. Domain Description (Ontology)

The concepts of this ontology include all the management aspects but for the sake of simplicity, the subset presented here focuses only on incident and SLA management and the needed network technological environment like access, core or transmission networks and administrative environments.

The main concepts have been extended with other concepts covering the main resources used in global service management. The use of an ontology description to define the SLA provides interoperability and enables the use of machine reasoning in order to infer knowledge. Other advantage is that the business SLA ontology would be able to include the definitions of behavioral rules for business information.

Alarm: Notification to draw attention to a condition that can have an immediate or potential negative impact on the state of the element or service being monitored.
Alarm description: A set of generic attributes related to the alarm, for example severity, priority or status.
Cause description: A set of particular attributes that describes the cause of the alarm.
Information: Notification used as an item in a report or a SLA between two actors. So, a SLA could be defined as a contract between a service provider and a customer, and it details the humour, character and scope of the service to be provided.

Each SLA has guarantee terms composed by SLOs (Service Level Objectives). SLOs are associated with a value and a threshold. Each threshold is associated with its KPI. The Billing values could change when a violation of the parameters specified in the Guarantee Management is detected.

E. Reasoning Rules

Each management aspect needs a set of rules which govern its correct way of working. These rules are based on the operators experience and could be modified when the domain conditions change.

In case of classic umbrella system, management is associated with the fulfillment of a rule that includes conditions from different domains. In (1), condition1 and condition2 are associated to domain1 and domain2 respectively.

Condition1 + Condition2 → Action A. (1)

In the proposed architecture this rule is splitted into several ones in order to simplify the multi-domain management. The rule showed in (1) is split in (2) - (5). In (2) the condition1 from domain 1 is associated with the actionA1 over the common part of the SKP. In (3), when the condition 2 appears in domain 2 and the action A1 has occurred, the result is the action A and a final action is determined. Similar reasoning could be argued to (4) and (5).

Condition1	→	Action A1	(2)
Condition2 + ActionA1	→	Action A	(3)
Condition2	→	Action A2	(4)
Condition1 + ActionA2	→	Action A	(5)

If a notification is only associated with one domain, it modifies the Knowledge plane associated with this domain and the common part. The incorporation of a new domain to the system is associated with the modification of the semantic model and the inclusion of new rules. It does not imply any change of code in the software process. In addition, if the model has been correctly defined, the modification will be minor because usually the concepts should be similar.

In the semantic description of fault / alarm domain and KPI/SLA domain, it is necessary to define the set of rules that will govern the implementation of preventive and corrective actions related to the alarm management system or the SLA negotiation including the penalties management.

Since this paper focuses on SLA management, a representative set of rules has been chosen, determining the actions to be carried out after the analysis of the SLA within a generic SLA management system. These rules apply to notifications coming from all the domains. The selected rules cover the determination of the SLA compliance degree, and the penalties associated with an SLA breach. In the later case, the fulfillment of SLAs is analyzed based on business rules, and suitable actions are planned to ensure the correct functioning of the services.

The defined rules for incident management include:

- Activation of an alarm or a new ticket is stored in the repository and the knowledge base and persistence timers are created to control the duration.
- The alarm or ticket is deleted from the repository and the knowledge base.
- Triggering of a breach SLA when the persistence timeout expires.
- Penalty Evaluation.

IV. Proof of Concept

The previous architecture was implemented in a proof of concept prototype, in order to validate that the architecture fulfills the basic functionalities of a service management sys-tem. For this prototype, it was selected a subset of the management domains, the SLA management. In this way, we aim to ensure the correct and automatic SLA management based on the KPIs received by the different actors who participate in the service, using a semantic model and an associated set of rules.

This proof of concept prototype shows the way the agents work and their interaction with the shared knowledge plane (see Figure 4). So, the described model has been applied to a real system that receives KPIs from network elements or systems (like trouble ticketing and report tools) belonging to different technologies or actors. These notifications from different environments present the same format with some fields common to all the domains and some others specific for each one.

The prototype manages a global service that is associated to two management actors, so this service has two administrative environments. Therefore, the prototype has two agents that apply a general processing to all the notifications, taken into account both common and specific fields. The prototype has been tested in a lab environment communicated with a telco operation system receiving real notifications.

The knowledge plane has been described by means of an Ontology written in OWL1 language. We have used Protégé 3.4 [11] to edit the ontology and behavior rules which have been written in SWRL (Semantic

Web Rule Language). Each agent has been developed in Java, and it uses the Java library Jena 2.5 [12] to perform the parsing and manipulation of the ontology information together with the Bossam reasoning engine to execute the rules. Bossam is not integrated in Jena, so the agent needs to write the Knowledge Base in an OWL file that Bossam reads to build its Knowledge Base with the same content but with its own structure.

Each agent receives the information from the elements and includes the corresponding KPIs, in the shared knowledge plane. Afterwards, it uses the reasoning engine to execute the rules and infer the needed actions. There are two possible situations: the KPIs affect the collective area of the SKP or affect a specific area.

Within the architecture, the agents are responsible for all SLA-related concerns, including the negotiation of SLAs. These agents interact through the Shared Knowledge Plane. During the negotiation process, the agent needs to find the service instantiations of an appropriate quality. So, agents rely on the service component that evaluates in advance the potential quality of a service.

For example, the agent of administrative environment A receives an alarm about the failure of a network element in a specific route. Then the number of elements which have failed is analyzed and an evaluation of the unavailability level is done.

The detailed steps for this common area scenario are listed below:

- Definition of the semantic model (OWL+SWRL) of the shared knowledge plane and generation of the OWL text file using protégé.
- Creation of the Knowledge Base (KB) in the shared knowledge plane based on the OWL+SWRL model.
- Update the KB in the shared knowledge plane with the received KPIs from any domain if the KPI affects to the common area.
- Bossam engine reads the OWL file and builds its own KB in the SKP
- If necessary, execution of an inference cycle in the SKP. Now the agent of the domain B receives an update of the KB about the failure of a network element. So, it performs Modification of the Bossams KB in the SKP following the reasoning results.
- Storage of this KB in an OWL file in the SKP.
- Each agent loads the new KB
- Planning and ordering of the actions to be carried out by the agents.

Table 1. Time Measures Rule Functionality

Rule	Time
Activation of an alarm or opening a ticket	35.328
Cessation of an active alarm	35.427
Triggering of an SLA breach	33.234
Penalty evaluation	30.487

In the case of the alarms affecting the specific area of the SKP, only associated with one agent, the steps are:
•Bossam engine reads the OWL file and builds its own KB in the agents.
•Modification of the Bossams KB following the reasoning results.
•Update the KB in the Shared Knowledge Plane with the reasoning results.
•Each agent loads the new KB.

We can summarize the results of this proof of concept in three aspects: description of the environment in which has been deployed, functionalities demonstrated by the proof of concept prototype and prototype performance evaluation.

The complete test involves two test environments. A local environment in a Windows-based PC with simulated notifications was used to check the fulfillment of every rule, and a Linux-based laboratory environment receiving real notifications from the network, where we have evaluated the accomplishment of time execution requirements. The performance measurements made in this environment with an empty KB are showed in Table 1.

The main component of the execution time is the time needed to write/read Bossam's OWL file. For the second test, the ontology and applications were moved to a real environment and it was measured the time needed to receive and process alarms. In these conditions the system received 300 alarms in 13 seconds, but it needed 57 seconds to process just the first one, with execution time growing with the size of the KB. Although these values are very high, they are eligible for the SLA management, where it is not needed real time reactions. These results show the efficient implementation of the reasoners for SLA environment.

V. Conclusions

This paper has presented an approach to the novel conception of global service management, based on shared agent's architecture. The domain is composed by the tuple technological environment, administrative environment and management aspect as provisioning, monitoring, guaranteeing the QoS, SLA management or fault recovery. The agents are allocated in the different elements associated with the service (network elements, service platform...) of each domain and all of them share knowledge about their domain. Agents receive notifications, analyse them and infer the suitable actions using a rule-based mechanism, upgrading the knowledge plane. OWL/SRWL was selected for the implementation although in future works will be studied other schemes.

This architecture supports the management needs of the new global services, simplifying the service operation. The basic management functionalities are fulfilled with this architecture, as it has been shown in the proof of concept prototype for the incident management case. Semantic-based management allows the easy modification of management requirements in heterogeneous environments, e.g. a SLA requirement between two actors in different domains.

This paper has presented the foundations of new service management architecture. Future work will discuss in depth topics such as conflict resolution in the arrival of notifications from different environments or delve into the agents' lifecycle.

References

[1] "Nagios," Available: http://www.nagios.org/
[2] "Splunk," Available: http://www.splunk.com/product
[3] "Jasper reports," Available: http://www.jaspersoft.com/products
[4] "DMTF, Desktop management task force home," Available: http://www.dmtf.org/home
[5] J. P. Thompson, "Web-based enterprise management architecture," *IEEE Commun. Mag.*, vol. 36, pp. 80–86, 1998.
[6] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proc. 2004 Intl. Conf. Autonomic Compute.*
[7] M. Burgess, "Probabilistic anomaly detection in distributed computer networks," *Sci. Comput. Programming*, vol. 60, no. 1, pp. 1–26, 2006.
[8] SLA@SOI project: IST- 216556, "Empowering the service economy with SLA-aware infrastructures." Available: http://www.sla-at-soi.eu/
[9] D. L. McGuinness and F. van Harmelen, "OWL web ontology language overview," Acknowledged W3C Member Submission, May 2004.
[10] IBM, "Autonomic computing: IBM's perspective on the state of information technology." Available: http://researcweb.watson.ibm.com/ autonomic/manifesto/autonomic_computing
[11] "Protegé." Available: http://protege.stanford.edu
[12] "Jena." Available: http://jena.sourceforge.net/