Nagalakshmi A, 2023, 11:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

Machine Learning Based Service Level Agreement Assurance in Cloud Computing Data Centers

Nagalakshmi A, Asst. Prof. Uma Maheshwari M
Department: Computer science and engineering
The kavery Engineering College,
M.Kalipatti, Tamil Nadu

Abstract- Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. In the Cloud environment the services are provided based on the Service Level Agreement (SLA). An SLA is a contract between user and service provider. It contains services list which is provided by service provider. It also contains availability and performance of the services. In most scenarios the service providers try to deceive their users, because users don't have the awareness about the SLA. So validating an SLA by user is always difficult. To avoid this difficulty, I introduced the SLA manager. An SLA manager is responsible for validating the SLA in the user perspective. In this approach the SLA is validate based on the availability of services. To monitoring the cloud services I used the Hyperic HQ monitoring tools. Based on that monitored data an SLA manager validate the availability of services listed in the SLA. After validating SLA, an SLA manager sends a report to the concern users. Based on that reports the users can demand the service provider for credit.

Keywords- Cloud , data centre, management, Assurance, Cloud service.

I. INTRODUCTION

The National Institute for Science and Technology (NIST) defines cloud computing as follows: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

This cloud model promotes availability and is composed of five essential characteristics**,** three service models, and four deployment models.

1. Essential Characteristics:

- **Broad network access.** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants (PDAs).
- **Resource pooling.** Cloud computing pools a provider's computing resources to serve multiple consumers using a multi-tenant model, with different physical and virtual resources assigned and reassigned according to consumer demand. Cloud computing provides a sense of location independence. Customers generally have no control or knowledge of the exact location of the resources. But, they may be able to specify location at a higher level of abstraction (e.g., country, state, or data center).

Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.

- **Rapid elasticity.** Resources can be rapidly and elastically provisioned, sometimes automatically, to scale out quickly, and rapidly released to scale in quickly. To consumers, the resources often appear to be unlimited and can be purchased in any quantity at any time.

2. NIST Defines Three Cloud Service Models:

2.1 Cloud Software as a Service (SaaS):

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure, typically through a pay-per-use business model. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web-based email).

The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

2.2 Cloud Platform as a Service (PaaS):

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly application hosting environment configurations.

2.3 Cloud Infrastructure as a Service (IaaS):

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems; storage, deployed applications, and possibly limited control of select networking components (e.g., host firewalls).

3. NIST Recognizes Four Deployment Models:

3.1 Private cloud: A private cloud in which the cloud infrastructure is utilized by just one organization, though not necessarily operated by that one organization.

3.2 Community Cloud: A community cloud whereby several organizations with common concerns share a cloud.

3.3 Public Cloud: The public cloud provided by the private sector for all comers, such as Amazon's EC2 service.

3.4 Hybrid Cloud: A hybrid cloud in which two or more cloud types are discrete but networked together such that a burst of activity beyond the capabilities of one cloud is shifted for processing to another.

4. Cloud Computing Vision:

Cloud computing will have future implications in the following areas:

- Economics
- Elasticity
- Globalization
- IT as Service Broker
- Catalyst for Innovation
- Inclusive Computing

II. PROBLEM STATEMENT

As consumers move towards adopting such a Service-Oriented Architecture, the quality and reliability of the services become important aspects. However the demands of the service consumers vary significantly. It is not possible to fulfill all consumer expectations from the service provider perspective and hence a balance needs to be made via a negotiation process. At the end of the negotiation process, provider and consumer commit to an agreement. In SOA terms, this agreement is referred to as a SLA.In most scenarios the service providers try to deceive their users, because users don't have the awareness about the SLA. So Validating an SLA by user is always difficult.

1. Establishing and Monitoring SLAs in complex Service Based Systems

Service Level Agreements (SLAs) are the instruments to model such contracts in the digital world, since they specify the conditions under which a certain service is provided by a provider to a customer. Provisioning of service hierarchies therefore implies similar dynamic and complex SLA hierarchies, established within and across the boundaries of organizations. A service provisioning infrastructure should allow the establishment of SLA hierarchies through coordinated negotiations among the potential stakeholders. However, SLA establishment can only partially serve the needs of SLA management if not linked to SLA monitoring. This paper explicates the link between SLA negotiation and monitoring of a Service Based System (SBS) in the context of the SLA Management framework developed by the SLA@SOI Project. In particular, we show how, during negotiation, service providers require historical data from monitoring to evaluate SLA offers made by service customers.

2. SLA Monitoring:

The SLA negotiation introduces two requirements for SLA monitoring: Monitoring should allow the collection of SLA violations during the provisioning of a service under the terms of an SLA. On the Provider side, such violations should be made available as historical data to SLA negotiation, for optimization and planning while deciding whether to accept or not a SLA offer made by the customer; Monitoring should be able to assess the monitor ability of the guarantee terms specified in a SLA offer made by an agreement initiator to an agreement responder. This is necessary since auditing and enforcing an SLA that has non-monitor able guarantee terms would not be feasible.

3. SLA-Aware Application Deployment and Resource Allocation in Clouds

Scheduling Strategy Design Issues:

In this section, we discuss the design of our proposed scheduling heuristic. We first present a use-case scenario showing resource provisioning types in Clouds based on which we position our scheduling heuristic.

Cloud Resource Provisioning and Application Deployment:

There are three well known types of resource provisioning layers in Clouds:

- Infrastructure as a Service (IaaS);
- Platform as a Service (PaaS); and
- Software as a Service (SaaS).

The three different layers of resource provisioning to host service requests from customers. The customers place their service deployment requests to the service portal, which passes the requests to the request processing component to validate the requests. If the request is validated, it is then forwarded to the scheduler. The scheduler selects the appropriate VMs through the provisioning engine in PaaS layer for deploying the requested service and the load-balancer balances the service provisioning among the running VMs. The provision engine manages the VMs on the virtualization layer and the virtualization layer interacts with the physical resources via the provision engine in IaaS layer. The service status and the SLA information are communicated back to the service portal. There is the possibility of provisioning at the single layers alone. However, our approach aims to provide an integrated resource provisioning strategy. Deploying single applications at SaaS layer is challenging due to the fact that each application demands the fulfillment of its SLA terms. In the next section we discuss our proposed scheduling heuristic aimed to address these challenges.

4. An Architecture Design of Life Cycle Based SLA Management:

In the Web services environment, appropriate guarantee for the service quality can maximize the ability of web services and ensure the services to be carried out effectively and controllable, and that's what SLA contracts can offer. An SLA(service level agreement) is a contract agreed between a service client and a service provider which defines a series of service quality characters and metrics. It is needed to divide the management functions base on the SLA life cycle. Thus, the life cycle of SLA is discussed and the SLA supported Web service architecture and management requirements are also presented. An SLA management platform is designed for SLA definition, service SLA registration and run-time SLA monitoring and controlling mainly base on the SLA life cycle.

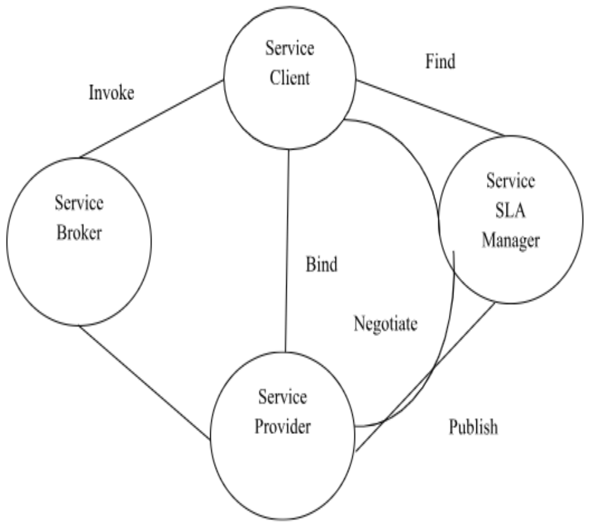

Fig 1. Roles in SLA Supported Web Services.

5. A Logic Based SLA Management Framework:

Management, execution and maintenance of Service Level Agreements (SLAs) in the upcoming service oriented IT landscape need new levels of flexibility and automation not available with the current technology. In this paper we evolve a rule based approach to SLA representation and management which allows a clean separation of concerns, i.e. the contractual business logic is separated from the application logic. We make use of sophisticated, logic based knowledge representation (KR) concepts and combine adequate logical formalisms in one expressive logic based framework called Contract Log.

Contract Log underpins a declarative rule based SLA (RBSLA) language with which to describe SLAs in a generic way as machine readable and executable contract specifications. Based on Contract Log and the RBSLA we have implemented a high level architecture for the automation of electronic contracts – a rule-based Service Level Management tool (RBSLM) capable of maintaining, monitoring and managing large amounts of complex contract rules.

III. SYSTEM ANALYSIS

1. Existing System:

The essence of using SLA in Cloud business is to guarantee customers a certain level of quality for their services. In a situation where this level eof quality is not met, the provider pays penalties for the breach of contract. In order to save Cloud providers from paying costly penalties and increase their profit, they devised the Low Level Metrics to High Level SLA|LoM2HiS framework, which is a building block of the FoSII infrastructure for monitoring Cloud. So mapping the low-level resource metrics to high-level SLA parameters, and detecting SLA violations as well as future SLA violation threats so as to react before actual SLA violations occur.

Each FoSII service implements three interfaces:

- Negotiation interface necessary for the establishment of SLA agreements,
- Application management interface necessary to start the application, upload data, and perform similar management actions, and
- Self-management interface necessary to devise actions in order to prevent SLA violations.

The LoM2HiS framework comprises two core components, namely host monitor and run-time monitor. The former is responsible for monitoring low-level resource metrics, whereas the latter is responsible for metric mapping and SLA violation monitoring.

For the decision making they use knowledge databases proposing the reactive actions by utilizing case-based reasoning. Case-Based Reasoning (CBR) is the process of solving problems based on past experience. It tries to solve a case (a formatted instance of a problem) by looking for similar cases from the past and reusing the solutions of these cases to solve the current one.

In general a typical CBR cycle consists of the following phases assuming that a new case has just been received:

- Retrieve the most similar case or cases to the new one,
- Reuse the information and knowledge in the similar case(s) to solve the problem,
- Revise the proposed solution, and
- Retain the parts of this experience

Although, there is a large body of work considering development of flexible and self-manageable Cloud computing infrastructures, there is still a lack of adequate monitoring infrastructures capable of predicting possible SLA violations. With the ever growing interest in Cloud computing from both, industry and academia, and the rapid growth of Cloud computational infrastructure resources, the management of the infrastructures to efficiently provision resources and services to customers is now a challenging task Cloud management systems consist of components such as, monitoring techniques, scheduling and deployment mechanism, and resource allocation and reallocation strategies.

The Cloud provider should provide services based on the SLA which is signed by their user. Sometimes the provider is not following the SLA.For example the SLA contains the network bandwidth as 1 mbps. But in the peak hours the bandwidth may be vary. For the loss of bandwidth the provider should give credit to the user. And also sometime users violate the SLA.

For that user should be fined or penalty should be paid by the provider. But in real time the issue is not taken into consideration. Because the users are not aware of the SLA and also there is no such analyzing tools for SLA violation and monitoring. Further, in real time the providers not accepting their mistakes. They try to blame the users for their mistakes.

1.1 Disadvantages of Existing System:

- Monitoring of resources in the Service providers view.
- Most often users are fined for Service provider's mistakes.
- Difficult to calculate the interval time to determine the violation.
- Based on the past experience the violations are determined. So determination of new violations is difficult.

1.2 Advantages of Proposed System:

To overcome above problem, we have introduced a SLA manager for analyzing SLA violations of the Cloud. If the violation occurs then they send report to customer or provider. For to achieve this it is necessary to create a SLA based on Storage, Server, Network, Data operations, Security and Other Software Components. The sequences of steps that are involved to perform the proposed framework are,

- Creating a good SLA.
- Introducing an Analyzer/ manager for analyzing and monitoring a SLA.
- Alert by sending report to the Provider for user SLA violations and the User for provider SLA violations.

Advantages are

- The SLAs are validating in the perspective of user
- Efficient monitoring tool is used for monitoring the resources for any violation
- The new violations are determined easily
- The user and service provider need not maintain any details of SLAs. Because SLA manager can handle the SLAs.

2. System Design:

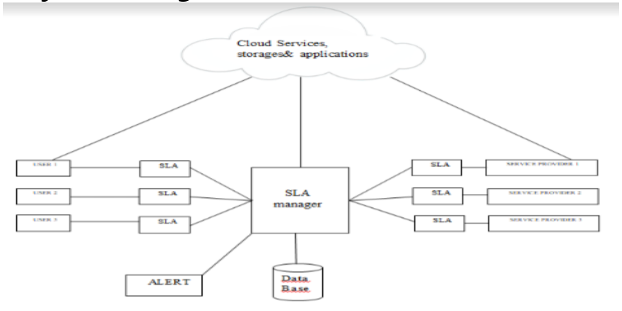

Fig 2. SLA manager architecture.

2.1 Data Flow Diagram:

- The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
- The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

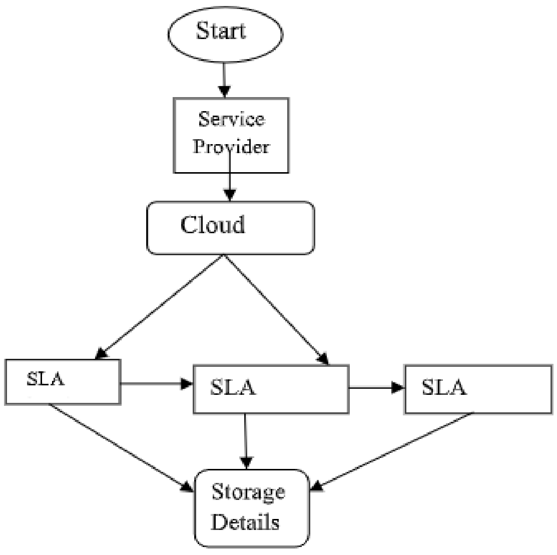

Fig 3. Dataflow Diagram.

2.2 UML Diagram:

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

The Primary goals in the design of the UML are as follows:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and component

2.3 Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

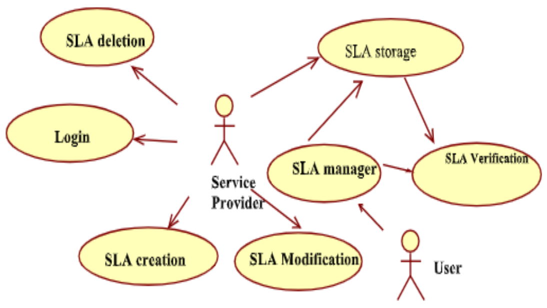

Fig 4. Use Case Diagram.

IV. MODULES DESCRIPTION

The following list of modules are used, they are

- SLA Definition
- SLA Monitoring
- SLA Management and Control
- SLA Reporting

1. SLA Definition:

An SLA is a working process to define and balance business requirements with available service resources. Further a contract agreement is in between user and service providers. Creating a SLA for Cloud based on following parameters.

1.1 Server:

There are two types of Server.

- Physical Server
    - Availability- Uptime and Down time
    - Rebooting time
    - Required Hardware configuration
- Virtual Server
    - CPU Core
    - RAM

1.2 Storage:

- Parameters:
    - Availability
    - Input/output per second
    - Maximum restores time
    - Processing time
    - Latency & internal compute resource

1.3 Data center operations:

- Parameters:
- Monitoring:
    - Availability
    - Fault
    - Performance
    - Notification
- Backup
    - Type of backup
    - Restoration time
- Metering & Billing
    - Billing mechanism

1.4 Security:

- Parameters:
    - Data privacy
    - Data seizure
    - Encryption
    - Key management
    - Access control

1.5 Other Software components:

- Parameters:
    - Operating system
    - License

  - up gradation (Patch update)

- Software's like hypervisor
  - License
  - Up gradation (Patch update)

2. SLA Monitoring:
SLA Monitoring is to collect the data from cloud and stored in the Database. For that purpose we used the Hyperic HQ monitoring tools. The tools contain two elements .The first one is server which is used to store the collected elements from the Cloud. And another one is Agent which is used to monitoring the functionality of the Cloud. By accessing the server we can get the monitored data. And store it in the particular database.

3. SLA Management and Control:
Collect the SLAs of particular user from databases and it is stored on the SLA manager Database. If any change occurs in the SLAs, the changes are update frequently that is the service provider has rights to modify SLAs. So if service provider modified any SLAs, the changes are updated frequently in the SLA manager Database. But user has the rights to view the SLAs. So the User can not modify the SLA.

4. SLA Reporting:
In this module, the data are collected from both databases. Such as SLA manager Database and Monitored database. The data are compared together sequentially. If mismatch occurs in the any parameters then the SLA manager sends a report the User. The report contains parameters which are not matched with SLA and duration. Based on this a user can request the service provider for the service remedies or bonus.

V. SYSTEM TESTING

Testing is the one step in the Software Engineering process that could be viewed as destructive rather than constructive. Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing is representing an interesting anomaly for the software.

1. Unit Testing:
Unit testing focuses verification effort on the smallest unit of the software design. This project compromises the set performed by an individual programmer prior to the integration of the unit into a larger system. This testing is carried out during the coding itself. In this testing step each module such as registration, login, etc going to be working satisfactorily as the expected output from the module.

2. Module Testing:
Since it is a real time project the modules in this project may collects inputs from another module or any sub modules. Likewise they can forward their output as inputs to some modules or sub modules. So a module testing is one of the important testing in system development cycle. This testing is used in login module. The output form registration is used as input for login module.

3. Integration Testing:
In this project the data can be lost across an interface; one module can have adverse effort on another, sub function when combined may not produced the desired function. Integration testing is a systematic technique for constructing the program while at the same time conducting test to uncover errors associated within the interface.

The objective is to take unit-tested module and built the program structure that has been dictated by design. All modules are combined in this testing.

The entire program is tested a whole. Correction is difficult at this stage because the isolation of module. At the integration testing, software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test validation testing begins.

REFERENCES

[1] CASViD: Application Level Monitoring for SLA Violation Detection in Clouds Computer Software and Applications Conference (COMPSAC), 2012 IEEE, Page(s):499 – 508

[2] Dan, D. Davis, R. Kearney, A. Keller, R. King, D.Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A.Youssef. Web services on demand: Wsla-driven automated management. IBM Systems Journal, 43(1):136 –158, 2004

[3] D. Davide Lamanna, James Skene, and Wolfgang Emmerich. Slang: a language for defining service level agreements. In Proceedings of The Ninth IEEE Workshop on

Future Trends of Distributed Computing Systems, 2003. FTDCS 2003. Pages 100 –106, May 2003

[4] Integrating SLAs into IT Risk management in public service organizations, Services Computing Conference, 2009. APSCC 2009. IEEE Asia-Pacific

[5] Jian Pu, Kin F. Li, Mostofa Akbar, Gholamali C. Shoja, and Eric Manning. A reliable sla-based admission controller for mpls networks. In IFIP International Conference on Network and Parallel Computing Workshops, 2007, pages 57 –64, September 2007.

[6] Long-Tae Park, Jong-Wook Baek, and J. Woon-Ki Hong. Management of service level agreements for multimedia internet service using a utility model. IEEE –Communications Magazine, 39(5):100 –106, May 2001

[7] Rajiv Chakravorty, Jon Crowcroft, Ian Pratt, and Maurizio D'Arienzo. Dynamic slabased qos control for third generation wireless networks: the cadenus extension. In IEEE International Conference on Communications, 2003.ICC '03., volume 2, pages 938 –943, May 2003

[8] Yu Cheng and Weihua Zhuang. Dynamic intersla resource sharing in path-oriented differentiated services networks. IEEE/ACM Transactions on Networking, 14(3):657 –670, June 2006

[9] www.cloud-council.org/10052011.html

[10] www.nist.gov/it1/cloud/refarch.cfm

[11] www.ssrn.com/abstract=1662374

[12] www.code.google.com/appengine

[13] www.oasis_open.org/committes

[14] www.cloudsecurityalliance.org/guidance/csegu ide.v3.o.pdf

[15] http://www.rackspace.com/information/legal/c loud/sla

[16] http://aws.amazon.com/ec2-sla/

[17] http://www.windowsazure.com/en-us/support/legal/sla/

[18] http://www.gogrid.com/legal/service-level-agreement-sla

[19] https://www.hpcloud.com/SLA

[20] http://cloudarchitect.att.com/About/ATT_Clou d_Architect_SLA_Policy.pdf

[21] https://developers.google.com/storage/docs/s la

[22] http://www.xenproject.org/

[23] http://www.eucalyptus.com/

[24] http://cdaccloud.com/meghdoot/

[25] http://www.osskb.net/ The Critical Role of SLA Manager

[26] http://blog.serviceframe.com/index.php/2009/ 10/whatsan-sla-manager