A Secure Self Adoption Mechanism in for Service Base Cloud Application

Purushottam S.Chavan, **Prof B. R. Nandwalkar**

Abstract — **Self-Adaptation, as a concept, has been around for many years in several domains like Business, etc. Self-adaptivity in computer-based systems is relatively upcoming techniques. Cloud computing, with its promise of (almost) unlimited storage, computation, and bandwidth, is increasingly becoming the infrastructure of choice for many organizations. As cloud offerings reliable, service-based applications need to dynamically qualify themselves to self-adapt to changing QOS requirements. In this paper, we present a decentralized mechanism for such self-adaptation and Security analysis using market-based heuristics. We use a CDA to allow applications to decide which services to choose, among the many on offer. We view an application as a multi-agent system and the cloud as a metropolis where many such applications self-compromise. Then, We Provide Security analysis mechanism identify that those newly or already visited or new application or i.e. bogus request, History etc**.

Key Words — Double action, Self adaptivity, Marketplace, QOS Monitoring, Optimization.

I. INTRODUCTION

Self-Adaptation, as a concept, has been around for many years in several domains like biology, chemistry, logistics, economics, etc. Self-adaptivity in computer-based systems is relatively Upcoming techniques. By self-adaptivity in software systems, In our software that monitors itself and the operating environment and takes appropriate actions when circumstances change and allot specific services to authenticate users. In web applications, service-oriented architecture has often been used as a mechanism for achieving self-adaptivity [02]. Web services allow for dynamic composition, which enables applications to switch services without going offline. A common instance of using web services dynamically is applications living on the cloud, asking for computing power and bandwidth to be scaled up or down, depending on demand. However, one of the clouds major selling points, operational flexibility, is of little use if applications (or organizations) have to indicate at sign-up time the kind of services that they intend to use. For example On Amazon, for instance, a customer specifies during sign up whether she wants a Hi-CPU instance or a Standard on Demand instance or a Hi-Memory instance. This assumes that an application is able to forecast its demand for computing and storage resources accurately. However, this inability to forecast is precisely what the cloud claims to address through elasticity in computing power. This is not to say that there are no flexible, demand-based pricing schemes available. Amazons Spot Instances is an example of how cloud providers are trying to flexibly price their services in response to fluctuating demand over time. Applications that can adapt to fluctuating prices will be able to ensure a better return on investment.

We refer one of the CDA protocol, the trading proceeds in two stages. In the first stage, the Market Agent matches the bids and asks based on their individual QoS values and shout prices. After matching, a provisional transaction is created. This provisional transaction enters the second stage. In the second stage, the Buyer Agent compares all the Asks returned as provisional transactions. The top-ranked Ask is selected and the other Asks are rejected. The Buyer Agent enters into a transaction with the Seller Agent of the selected Ask

.In our system, we try resolve issue that many users Visited specific web portal or offline application that but sum users have required more demand services and quality large number users i.e. some unauthenticated and authenticated but try access that web portal. We try develop mechanism try filter try identify which users bogus and all history available that easily match to uses of application valid user.

II. LITERATURE SURVEY

Canfora et al.[6] proposed a genetic algorithm-based approach where the genome length is determined by the number of abstract services that require a choice to be made. Constraints on QoS form a part of the fitness function, as do cost and other QoS attributes. A big advantage of GA-based approach is that it is able to handle nonlinear constraints, as opposed to integer programming. Also, it is scalable when the number of concrete services per abstract service increases.

Alrifai et al. [7] propose an interesting mechanism for cutting through the search space of candidate web services by using skyline queries. Skyline queries identify non dominated web services on at least one QoS criteria. A non-dominated web service means a web service that has at least one QoS dimension in which it is strictly better than any other web service and is at least equal on all other QoS dimensions. Determining skyline services for a particular abstract service requires pairwise comparisons among the QoS vectors of all the concrete services. This process can be expensive if the number of candidate concrete services is large. Alrifai et al. consider the case where the process of selecting skyline services is done offline. This would lead to an inability to adjust to changing conditions of available services and their

associated QoS values. Zhang et al. [8] propose an interesting method to achieve a good set of concrete services, using Ant Colony Optimization (ACO). ACO involves creating virtual ants that mimic the foraging behavior of real ants. The search space of optimal concrete services is modeled as a graph, with sets of concrete services as vertices and edges being all the possible connections between different concrete service sets. The ants attempt to complete a traversal of the graph, dropping Pheromones on the edge of each concrete service visited. The Path through the graph that accumulates the most pheromones Represents the near optimal path of services to use. Our Approach differs from the above approaches in two respects:

1. Consideration of time as a factor: In practice, the optimal set of concrete services may not be available at the time instant that an application is searching. The set of service providers changes with time, as does the set of service consumers. This means that the optimal matching of service providers to consumers changes with time. The approaches above do not take this into account.
2. Optimality not considered: Due to the infeasibility of computing the optimal set (being NP-hard), we concentrate on Finding a good solution rather than an optimal one. A good solution is one that does not violate any QoS constraints and meets the cost constraint within a certain margin. We do preliminary work on security analysis and filter of service based on the cost to store all information of those have Visited and allot services to user.

III. Need of Proposed System

In our system, we try resolve issue that many users Visited specific web portal or offline application that but sum users have required more demand services and quality large number users i.e. some unauthenticated and authenticated but try access that web portal. We try develop mechanism try filter try identify which users bogus and all history available that easily match to uses of application valid user.

Applications that use dynamic service composition should be able to continuously monitor their current QoS levels and make adjustments when either the demand for QoS changes or the cost constraint changes. The application should thus be able to respond to both internal as well as external stimuli to trigger a change in its constituent web services. Wewould like to create a mechanism that allows multiple applications, constructed across a federation of clouds, to self-adapt. We chose a market-based approach to self-adaptation,not only because it is decentralized, but also due to its easy applicability to the problem domain.

IV. Implementation Details

A. Secure self-adoption mechanism in for service base Cloud application (SSAMSC):

We call our implementation of the proposed mechanism, clobmas (cloud-based multi-agent System [4]). A secure self-adoption mechanism in for service base Cloud application (SSAMSC) we also Provide Security analysis mechanism identify that those visited application i.e. Bogus request, History etc.

We document clobmas using the object-oriented paradigm for all of the internal components of an agent[5]. We show Agent-interaction using Agent UML (AUML). We envision our mechanism as a middleware between service-based Applications and multiple SaaS clouds.

- Buyer and Seller Agent
- Market Agent
- Agent Interface
- Security Analysis

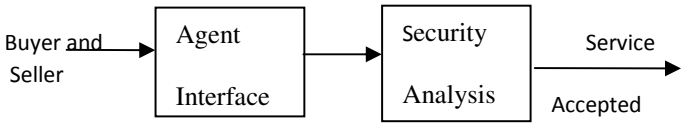

Fig. 1.General Structure of SSAMSC

In a proposed system the major focus will be given on two Things:
1) QOS Monitoring
2) Marketplace and
3) Different Self adaptation
i)Self Protection
ii)Self awareness
iii)Context awareness
v)Self Optimizing
vi) Self Protection

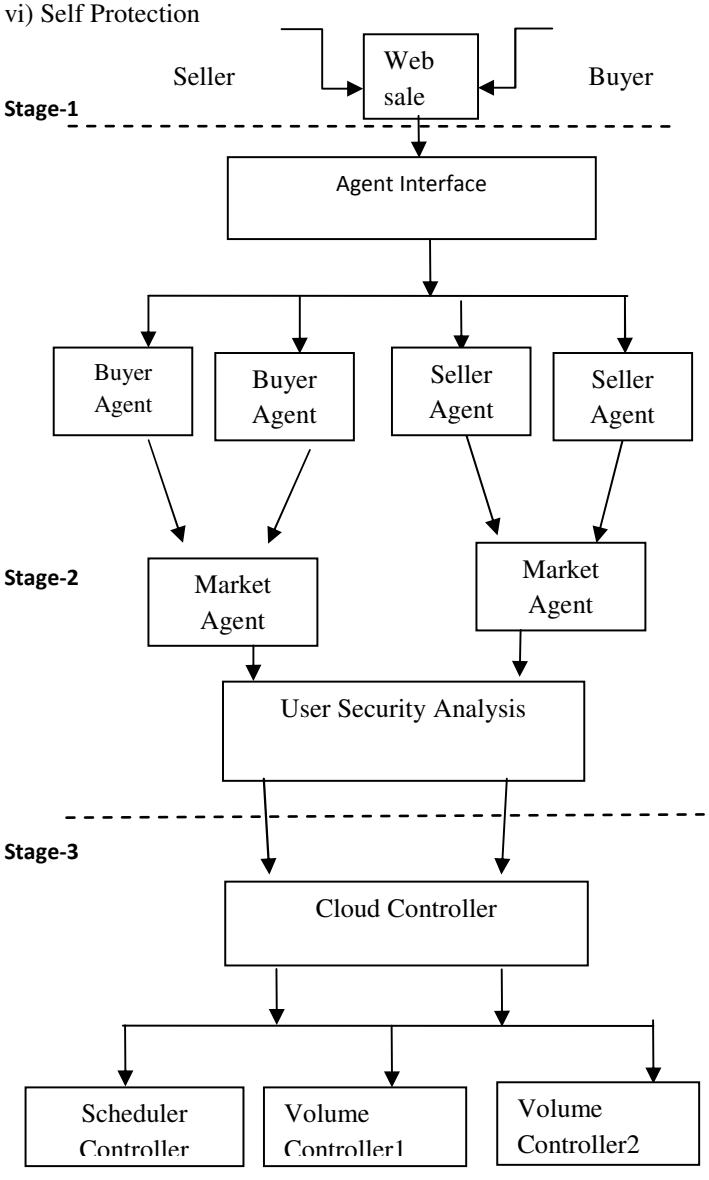

Fig 2. Block Diagram of Proposed System (SSAMSC)

Flowchart of SSAMSC:

The flowchart of SSAMSC system is as shown below:

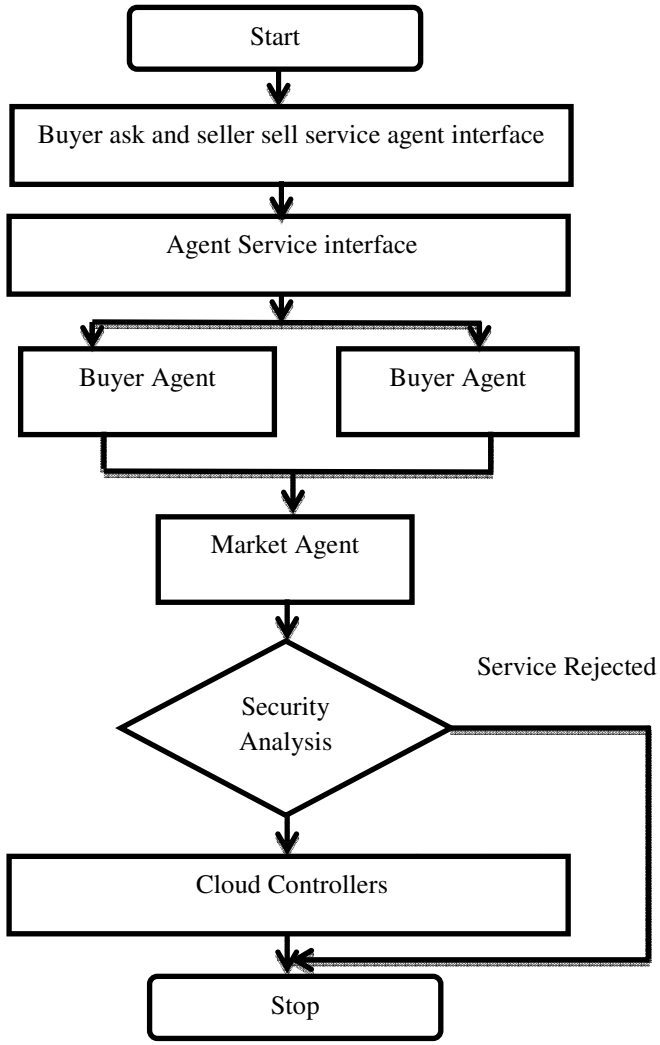

Fig.3. Flowchart of SSAMSC

B. **Algorithm of SSAMSC:**

1. Start
2. The process of Buyer ask and seller sell to service Agent Interface.
3. Services sent to Buyer and Seller agent
4.Market agent Allocate the web service with respect QoS
5.Perform Security analysis .
6.Stop.

C)Each module wise Input and output :

– Buyer and Seller Agent
Input: In this module we Service send and request in the datasets as input, in
which sends to agent interface.

– Buyer agent and Seller agent module

Input: Accept Request form agent interface first

Output: According to Cost of service, service is selected.

– Market agent module

In this module Bidding of service.

– Security Analysis
In this module check that service history for bogus or already visited.

B. Application Agent and Buyer Agents Life cycle:

We now describe the algorithms used by the Application Agentand Buyer Agents, in the course of adaptation. There are two distinct phases in the agents' lifecycle:

i. ***Initialization Phase:***

In this phase, the Application Agent is responsible for achieving The following goals:
1. Distribute total budget amongst Buyer Agents
2. Distribute end-to-end constraints amongst Buyer Agents
The Buyer Agents are responsible for achieving the following Goals:
1. Register with a market that deals with their specific web service
2. Acquire historical data regarding services sold in the market. Specifically, the Buyer Agent must get data regarding the lowest transaction price, the highest transaction price, the median transaction price and their corresponding QOS values.
3. Communicate the median transaction price and or responding QOS values to the Application Agent.

ii)Adaptation Phase:

In this phase, the Application Agent is responsible for Achieving the following goals:
1. Communicate to the Buyer Agents, in the following cases:
(a) The total budget available changes
(b) The target QOS changes
The Buyer Agents are responsible for achieving the following goals:
1.Generate multiple Bids for registering in markets.
2. Based on possible transaction matches, generate ranking amongst matched Asks and conclude transaction
3. Communicate with Application Agent about concluded transaction
4. Register ConcreteService with QOS Monitoring Engine.
5. Monitor communication with Application Agent for change in QOS
6. Monitor communication with QOS Monitoring Engine

V. RESULTS

A. Datasets:

OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. The OpenStack dashboard provides administrators and users a graphical interface to access, provision and automate cloud-based resources. The extensible design makes it easy to plug in and expose third party products and services, such as billing, monitoring and additional management tools. The dashboard is also brandable for service providers and other commercial vendors who want to make use of it.

The dashboard is just one way to interact with Open Stack Resources. Developers can automate access or build tools to manage their resources using the native Open Stack API Or the EC2 compatibility.

VI. ANALYSIS OF EXISTING

In existing system describes an experiment of deploying a successful centralized computer vision and machine learning activity recognition tool on federated cloud architecture, so as to scrutinize its effectiveness on an industrial large scale. It proposes the use of agent-based workflow management mechanisms in industrial automation. However, the application described therein is different, as in this work equipment's and smart objects are wrapped as agents and exposed as web services that contain real-time status information. Analysis and planning only monitored. There is no predefined feedback session. Applications that use dynamic service composition should be able to continuously monitor their current QoS levels and make adjustments when either the demand for QoS changes or the cost constraint changes. The application should thus be able to respond to both internal as well as external stimuli to trigger a change in its constituent web services. We would like to create a mechanism that allows multiple applications, constructed across federation of clouds, to self-adapt. We chose a market-based approach to self-adaptation, not only because it is decentralized, but also due to its easy applicability to the problem domain.

VII. CONCLUSION AND FUTURE SCOPE

We see that the market-based mechanism [3] consisting of simple agents is able to adapt well and yet scales linearly to the number of concrete services. We also see that it

is robust in the presence of differences in demand and supply of QoS In addition, we will Provide Security Analysis and effect of filter traffic. In this process, we discovered Solution of smart and secure service selection based on the criteria of is users is bogus or authenticated and also check history those have access web portal. In which monitoring cloud of two or more system Monitoring the results from different application. We also show that by Application service accepted and rejected and then we greatly reduce time to access the application.

We have not modeled complex seller-side behavior, specifically actions like deliberate violation of QoS to free up resources for making Asks with higher prices or misreporting of QoS available. Mechanisms like penalties and reputation management can be used to prevent seller agents from behaving dishonestly. Also, we have not modeled adaptation on the part of the market. Sellers that lie about their QoS or are generally unattractive for transactions may lower the reputation of the marketplace. Hence, the market could take steps to ensure that it is populated only with sellers that are likely to be sold. In future work, we aim to systematically add these modifications to observe their effect on the collective adaptation.

ACKNOWLEDGMENT

Prof.B.R. Nandwalkar, for his guidance and support. I will forever remain grateful for the constant support and guidance extended by guide, in making this report. Through our many discussions, he helped me to form and solidify ideas. The invaluable discussions I had with him, the penetrating questions he has put to me and the constant motivation, has all led to the development of this project.

I wish to express my sincere thanks to the Head of department, Prof. N. R. Wankhade also grateful thanks to M.E. coordinator Prof. Ms. J. V. Shinde and the departmental staff members for their support.

REFERENCES

[1] Vivek Nallur and Rami Bahsoon, "A Decentralized Self-Adaptation Mechanism for Service-Based Applications in the Cloud" Member, IEEE, IEEE Transactions On Software Engineering, May2013 VOL. 39, NO. 5,
[2] E. DiNitto, C. Ghezzi, A. Metzger, M. Papazoglou, and K. Pohl, "A Journey to Highly Dynamic, Self-Adaptive Service-Based Applications," Automated Software Eng., Sept. 2008. vol. 15, nos. 3/4, pp. 313-341,
[3] J. Niu, K. Cai, S. Parsons, E. Gerding, and P. McBurney, "Characterizing Effective Auction Mechanisms: Insights from the 2007 TAC Market Design Competition," 2008. Proc. Seventh Int'l Conf. Autonomous Agents and Multi agent Systems,pp. 1079-1086,
[4] M. He and N.R. Jennings, "Southamptontac: An Adaptive Autonomous Trading Agent," Aug. 2003. ACM Trans. Internet Technology, vol. 3, pp. 218-235
[5] J. Niu, K. Cai, S. Parsons, P. McBurney, and E. Gerding, "What the Market Design Game Tells Us Are Effective Auction Mechanisms," Autonomous Agents and Multi-Agent Systems, 2010.vol. 21, pp. 172-203.
[6] .G. Canfora, M. Di Penta, R. Esposito, and M. Luisa Villani, "An Approach for QoS-Aware Service Composition Based on Genetic Algorithms," 2005.Proc. Conf. Genetic and Evolutionary Computation, pp. 1069-1075.
[7] M. Alrifai, D. Skoutas, and T. Risse, "Selecting Skyline Services for Qos-Based Web Service Composition," 2010 Proc. 19th Int'l Conf. World Wide Web, pp. 11-20.
[8] W. Zhang, C.K. Chang, T. Feng, and H.-y. Jiang, "QoS-Based Dynamic Web Service Composition with Ant Colony Optimization," July 2010 Proc. IEEE 34th Ann. Computer Software and Applications Conf pp. 493-502, July 2010