Ascertaining the QoS Attributes for SOA based Applications in Cloud

K. Balakrishna,
M.Tech , Department of C.S.E.,
Kuppam engineering college, Kuppam,
Andhra Pradesh, India.

S. Thulasi Krishna, M.E,(Ph.D),
Associate Professor, Department of C.S.E.,
Kuppam engineering college, Kuppam,
Andhra Pradesh, India.

Abstract— **Service Oriented Applications have the ability to change their constituent services. This involves that they have the ability to change both their functionality and their Quality of Service requisites at runtime. In this paper, Multi-Agent System is used, which uses multiple double auctions, to enable applications to self-adapt, based on their Quality of Service attributes and cost constraints. We use a continuous double-auction (CDA) to allow applications to decide which services to select among all services. Cloud based Multi agent system is shown to be an effective mechanism, to allow both service consumers and service providers to self-adapt to dynamically changing QoS requirements. We introduce a ZIP algorithm for both the service providers and service consumers for bidding. Also we design a market mechanism that allows applications to select services, in a decentralized manner.**

Index Terms— **Self-Adaptation, Cloud-based-Multi-Agent System, Double-Auction, Decentralized**

I. INTRODUCTION

In the short history of computing, there have been a few ideas that have changed the focus of software engineering. The shift from mainframe-based computing to desktop-based computing was one such idea. The computer began to be seen as an individual device, storing personal data and work, rather than an exclusively workplace-tool. However, the idea of cloud computing, also known as utility based computing is prompting a move back towards enterprise-scale computing. Combining ideas of virtualization and data centers, cloud computing promises a near-infinite amount of computing power, storage and bandwidth. One of the major selling-points of the cloud is the pay per- use model of revenue generation. In this model, much like public utilities of electricity and water (and hence the name), customers pay for the amount of computing services they actually use. This allows customers, especially small and medium-scale enterprises, to cut back on capital expenditure, and focus only on operational expenditure. When demand goes down, the application can release the computational power that it no longer needs. This flexibility in cost, however, has a downside. Although the cloud providers make available large amounts of computing power and storage, they make no guarantees about the quality-of-service (QoS) attributes of the services being provided by them. By QoS attributes, we refer to qualities like reliability, availability, performance, security and other non-functional requirements which need to be provided, maintained, evolved and monitored at runtime. These qualities are fundamental to the user's satisfaction with the application. In fact, even if an application performs its functions correctly, the absence of the requisite QoS can lead to significant contractual losses.

II. PROBLEM

A self-adaptive service based application should be able to change one of its services for another service that is functionally the same, but delivers better QoS. This would lead to the application being able to exhibit higher performance, higher throughput or lower latency, as the situation warrants. Obviously, this kind of adaptation would entail a cost. But not meeting customer expectation could entail a higher cost, or even loss of market-share. Therefore, an application that is able to adapt to changing QoS demand is better than an application that cannot adapt. However, the current model of working on the cloud does not enable this. Services cannot be flexibly utilized and released. Also, selecting from a more number of services that are functionally identical, but differ on QoS is a difficult problem [1]. In this paper, we are concerned with the problem of self-adaptation of cloud-based applications, with regard to changes in QoS requirements.

In the recent past, Cloud providers, in order to attract, achieve and retain customers, they must offer a higher level of performance and security than its competitors. In order to do this, it decides to provide its application in the cloud and also it uses third-party services which are also hosted in the same cloud. Although, the services with the required QoS are available on the cloud, selecting the right service at a right time is a problem. In all commercial clouds, the cloud provider uses a posted-offer mechanism. A posted-offer mechanism is a form of market where the provider posts a certain price on a use-it-or-leave it basis. Also one can observe that, on Amazon's Elastic Cloud Compute (EC2) [2], there are several services that are functionally same, but priced differently. This price differentiation occurs due to different QoS being provided by these services. Any application that desires to use a particular service has to pay the branded price. There is no mechanism to bargain with any cloud provider (like Amazon etc.), on cost or Quality of Service of the services being offered.

III. MARKET-BASED SOLUTION

We deposit a different type of market-based solution would allow both parties to be satisfied. We provide a solution as

ISSN: 2278 – 1323

Clobmas (Cloud-based Multi-Agent System) for this problem. Clobmas [1] uses multiple double auctions, to enable applications to self-adapt, based on their QoS requirements and budgetary constraints. We design a marketplace that allows applications to select services, in a decentralized manner. Clobmas is shown to be an effective mechanism, to allow both applications and clouds to self-adapt to dynamically changing QoS requirements.. In this, we use a continuous double-auction (CDA) to allow applications to decide which services to select, among all provided services.

IV. SELF-ADAPTIVE ARCHITECTURE

In human-designed systems, the first systematic efforts to create a self-adaptive system have been in the domain of control loop design. Most systems can be differentiated on the basis on where the locus of control for self-adaptation [3] lies:

1. Centralized: In these type of systems, there is usually a hierarchy of components. Components at higher levels are responsible for goal management and planning for change, while those at lower levels are responsible for immediate action and feedback. Decision-making is concentrated in one or a closely related set of components.

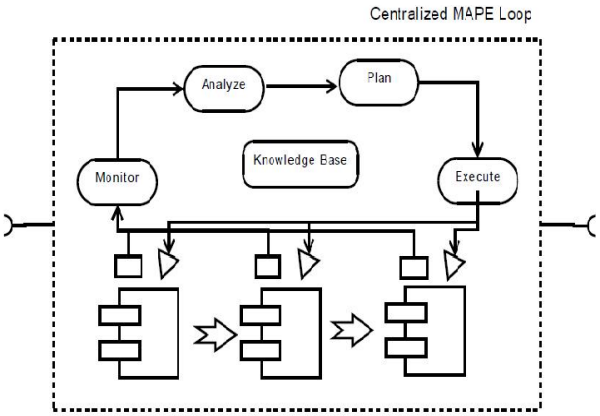

Figure 1: Self-adapting application with a centralized MAPE loop.

Centralized self-adaptive systems exhibit a communication pattern [3] that is characterized by sensory information (data) flowing from dumb components to the central decision maker, and instructions (commands) flowing from the decision maker to the dumb components. 'Dumb' is used here, in the sense of a component having no awareness of itself and its effects on the environment. Predictable and cohesive response to change are advantages of this type of system. However, reaction times get slower and slower as the size of the system increases.

2. Decentralized: Decentralized systems do not have a hierarchy of components. Each component acts as an individual agent with its own goals, and its own perception of the environment. This has the advantage of quick reaction to change. But it also has the disadvantage of being fragmented. That is, different components may react differently to the same change stimulus. This could make the self-adaptation inefficient or even deleterious. The challenge in building a decentralized system is to ensure that all the agents collectively move the system towards a common goal. This is usually accomplished by agents communicating amongst themselves. However, the lack of a centralized communication pattern means that the communication protocol must be decentralized and environment-based.

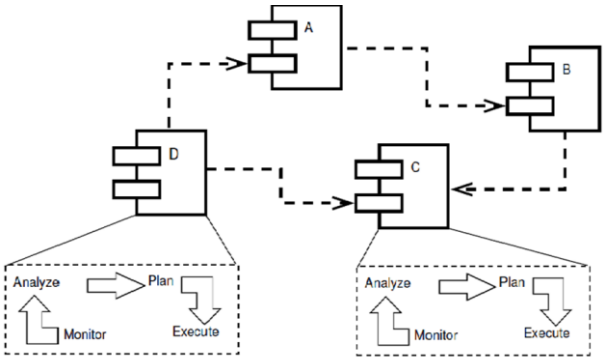

Figure 2: Decentralized MAPE loop in an application.

3. Market-based Control: This communication pattern is typically used in efficient, decentralized resource allocation. Multiple autonomous agents act in a self-interested manner, to achieve a globally coherent goal [4]. Taking inspiration from human-economies, each individual agent participates as a buyer or a seller of goods. In a software setting, the goods might be cpu-cycles, bandwidth, disk-usage or any other scarce resource. The underlying idea is that market equilibrium represents the optimal allocation of resources, achieved through local computations and no global coordination. This is a flexible and robust mechanism that reacts swiftly to changes in resource levels.

V. THE MARKET MECHANISM

A market-based mechanism [5] can be used as a pattern, for creating a decentralized, self-adaptive mechanism for service selection in the cloud. The marketplace operates a continuous double auction (CDA) which brings buyers and sellers together, and decides when a transaction should take place and at what price.

1. The Buyer: This is the application that we are primarily concerned about. This is the application that re-configures its architecture through the process of buying web-services. The application receives a relative weighting amongst the QAs that it is concerned about.

2. The Seller: This is the application that sells web services to the highest bidder. This application has a minimum 'ask' price, below which it is not economical for the seller to sell. This is so due to the fact that computation, storage and data transfer all have a cost in the cloud. These are all paid by the seller's web service.

3. The Marketplace: This is an application that resides in the cloud, and acts as the meeting point for buyers and sellers. The market is said to be optimally efficient, when the all the possible bids and asks that can be matched, are matched for a transaction. The marketplace operates a continuous double auction (CDA) [6]. A double auction is an auction where both

ISSN: 2278 – 1323

buyers and sellers post quotes about the quantity being traded and the price they're willing to pay/ask. Deciding when a transaction has happened is called clearing the market. A double auction can be cleared on one-shot or a repeated basis. A CDA operates a continuous clearing mechanism.

4. Double Auction: Double auction (DA) [5] is a auction mechanism where both buyers and sellers place bids. In auction terminology, the buyer's bid is called a bid and the seller's bid is called an ask. The bids and asks are stored in an order book. The market then matches the bids and asks to create potential transactions. In a continuous-time auction, bids and asks are entered into the market continuously. Transactions also, therefore, take place continuously. The continuous-time variant, called Continuous Double Auction (CDA) [6].

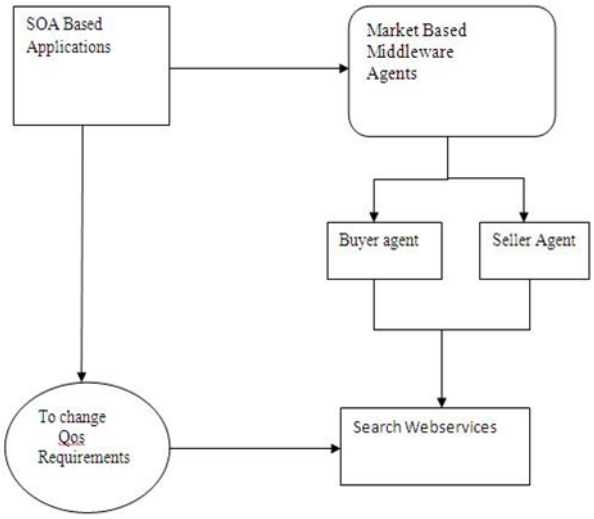

Figure 3: Architecture for changing the Quality of Service requirements in cloud.

In a Continuous Double Auction, the market clears continuously, i.e., instead of waiting for all bids and asks to be arrived, matches are attained as the bids and asks come in. A new bid is assessed against the existing asks and the first ask that check, is immediately partnered off for a transaction. Thus, high bids (a bid where the buyer is willing to pay a high price) and a low ask (where the seller is willing to accept a low price) are favored, in the mechanism. Also note that at any point, the only bids and asks present in the system are ones that do not have any match.

For example, if a bid enters the mechanism at time t, it is evaluated against all the asks present at that time. If no match is found, the bid can subsequently be matched only against new asks that enter. If no new asks enter, that bid will never be matched. The situation is symmetric with asks. An ask is evaluated when it enters the system and if a matching bid is found, it is immediately paired off for a transaction. Once this is done and no pairing has occurred, then it can only match a new bid. Thus, the only bids and asks remaining on the order book (after a round of matching), are ones that haven't been matched. To prevent stale bids and asks from remaining on the order book, most market institutions clear all un-matched bids and asks after a set period of time. A CDA is known to be highly allocatively efficient i.e., it achieves a very high percentage of all the possible deals, between consumers and providers. A CDA's efficiency [8] results from the structure of the mechanism used, instead of intelligence of the agents involved in trading. This is a very important result, since it provides us with an guarantee about the lower bound of efficiency of the mechanism. The multiple double auction method performs better than the currently-used posted-offer method for service selection.

VI. BIDDING STRATEGY

In market mechanism, for a buyer the sequence of steps in self -management are:

- First assign budget to it's application.
- Secondly assign relative weights to each quality attribute.
- The application partitions the budget amongst the functionalities.
- After partition, the application enters into marketplace and performs actions as shown in figure 4.
- Now create a bid consist of maximum cost and the level of quality attribute.
- Then post a bid to market.
- If bid is matched within time 't' then connect to sellers web services. Otherwise adjust bid and again post bid to market.
- After connecting to sellers web services monitor composed web services.
- If suppose actual value is greater than equal to expected value then compose functionality.
- Otherwise modify bid and again post bid to market.

ISSN: 2278 – 1323

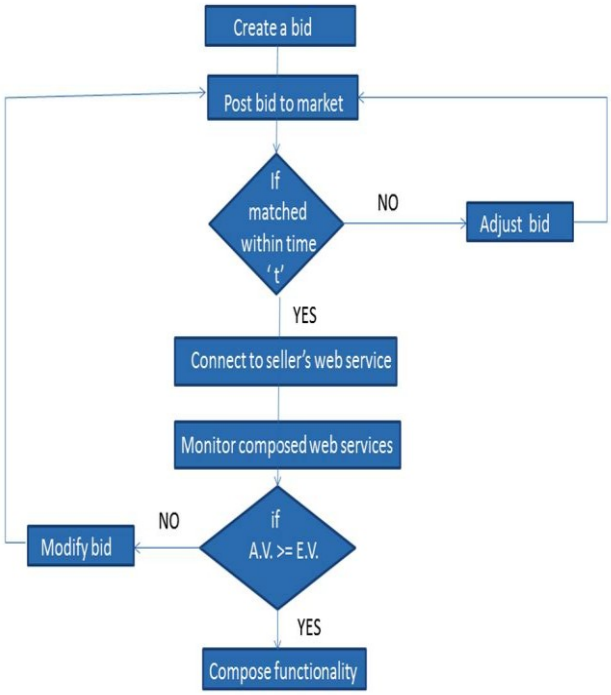

Figure 4: Bidding Process

VII. ADAPTIVE TRADING AGENTS FOR CDA

Simple adaptive trading agents for CDA markets is "zero-intelligence-plus" (ZIP) traders. Zero-Intelligence Plus is adaptive automated trading algorithm. The ZIP strategy has become a popular benchmark for CDA experiments. ZIP traders are designated as either "buyers" or "sellers" and are presented with a series of assignments to trade Zero-intelligence programs that submit random bids and offers replace human traders in experimental continuous double-auction markets. ZIP agents are profit-driven traders that adapt using a simple learning mechanism, adjust profit margins based on the price of other bids and offers in the market, and decide whether to make a transaction or not. When a decision to raise or lower a ZIP trader's profit margin is taken, ZIP modifies the value using market data and an adaptation rule based on the Widrow and Hoff "delta" rule. we also allow ZIP traders to buy and sell on their own behalf. Each trader contains a ZIP price for pricing bids and a separate ZIP price for pricing asks.

A graphical representation of the supply and demand schedules in a typical market is shown in Figure 5. Each seller has a cost price ci for each item i that she possesses, and in most cases, this is the lowest price at which she will sell item i. Similarly each buyer has a utility price u_i for each item i that she wishes to buy and this is the maximum price that she will be willing to pay for it. The graph shows two curves each of which represent:

- The quantity that will be available for purchase by a buyer at a given price(Supply curve).
- The quantity that is wanted for purchase at a given price (Demand curve).

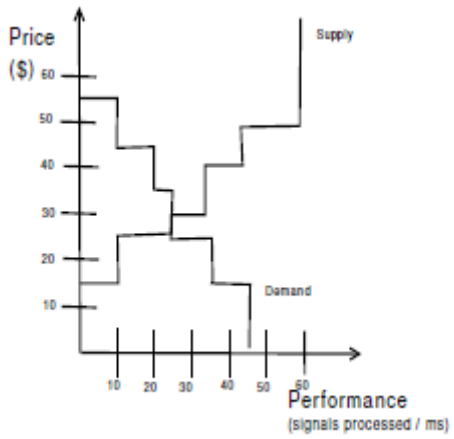

Figure 5: The supply and demand in Marketplace

The quantity available for purchase increases with price. This is because more suppliers are willing to sell for a higher price as opposed to a lower price. The opposite is true for the demand curve more buyers are willing to purchase a good at a lower price.

VIII. CONCLUSION:

In this paper, we studied Cloud based applications potential which have the ability to self adapt their Quality of Service depending on demand. Also, the multiple double auction method performs better than the posted-offer method for service selection. Consequently, Clobmas mechanism is a effective mechanism to allow customers to create adaptive applications and also generates a higher utilization of services than the posted offer model. 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. Also a continuous double-auction (CDA) mechanism is used to allow applications to decide which services to choose among all services. In our approach, multiple-auction based mechanism satisfied both adaptive applications as well as cloud-providers.

REFERENCES

[1]. Vivek Nallur and Rami Bahsoon "A Decentralized Self-Adaptation Mechanism For Service-Based Applications in The Cloud", IEEE Transactions On Software Engineering Vol:39 No:5,pages 1-25, 2013.

[2]. Davie Cliff, "Evolution of market mechanism through a continuous space of auction-types", Hewlett-Packard Laboratories, Year-2002.

[3]. Giovanna Di Marzo Serugendo, Marie-Pierre Gleizes, and Anthony Karageorgos. "Self-organization in multi-

ISSN: 2278 – 1323

agent systems". The Knowledge Engineering Review, 20(02):165–189, June 2005.

[4]. Elisabetta DiNitto, Carlo Ghezzi, Andreas Metzger, Mike Papazoglou, and Klaus Pohl. "A journey to highly dynamic, self-adaptive service-based applications. Automated Software Engineering, 15(3-4):313–341, September 2008.

[5]. Vivek Nallur and Rami Bahsoon."Self-adapting Applications Based on QA Requirements in the Cloud Using Market-Based Heuristics", Volume 6481, Towards a Service-Based Internet, Pages 51-62, Conference :Service Wave 2010.

[6] Perukrishnen Vytelingum. "The Structure and Behaviour of the Continuous Double Auction". PhD thesis, 2006.

[7]. Robert Laddaga. "Creating robust software through self-adaptation",IEEE Intelligent Systems, 14:26–29, May 1999.

[8]. Vivek Nallur and Rami Bahsoon. "Design of a Market-Based Mechanism for Quality Attribute Tradeoff of Services in the Cloud". In Proceedings of the 25th Symposium of Applied Computing (ACM SAC), P.NO:367-371, ACM, 2010.

[9]. DavieCliff,Janet Bruten, "Minimal-Intelligence Agents for Bargaining Behaviors in Market-Based Environments, Agency and Mediated Communications Department, HP Laboratories Bristol,Pg.no:41-45, August,1997.

[10]. Jorge Cardoso, Amit Sheth, and John Miller. "Workflow quality of service". Technical report, University of Georgia, Athens, Georgia, USA, March 2002.

ISSN: 2278 – 1323