# Qos Aware Cloud Service Composition using Fuzzy Engine Model in Cloud

Lakshmi Priya.V[1]
PG Scholar,
Department Of Computer Science And Engineering,
SBM College Of Engineering And Technology,
Dindugal,

P. Rajapandi[2]
Assistant Professor,
Department Of Computer Science And Engineering,
SBM College Of Engineering And Technology,
Dindugal,

***Abstract:*** **When a single Cloud service (i.e., a software image and a virtual machine), on its own, cannot satisfy all the user requirements, a composition of Cloud services is required. Cloud service composition, which includes several tasks such as discovery, compatibility checking, selection, and deployment, is a complex process and users find it difficult to select the best one among the hundreds, if not thousands, of possible compositions available. In existing system, a framework and algorithms which simplify Cloud service composition for unskilled users is introduced. In addition, to minimize effort of users in expressing their preferences, it applies combination of evolutionary algorithms and fuzzy logic for composition optimization. However, existing methodology cannot support well for workflow execution and QoS violation may occur. In proposed system Composition based trust driven scheduling algorithm is presented in order to achieve the trust driven and QoS aware scheduling for the composition of cloud services. The experimental result shows that the proposed methodology can schedule the workflow tasks effectively than the existing methodology.**

***Keywords : Cloud service, Qos, Fuzzy Engine, Fuzzy Ranking***

## I. INTRODUCTION

In order to deliver their solutions, application service providers can either utilize the platform-as-a-service (PaaS) offerings such as Google App Engine and OpenShift or develop their own hosting environments by leasing virtual machines from infrastructure-as-a-service (IaaS) providers like Amazon EC2 or GoGrid. However, most PaaS services have restrictions on the programming language, development platform, and databases that can be used to develop applications. Such restrictions encourage service providers to build their own platforms using IaaS service offerings. One of the key challenges in building a platform for deploying applications is to automatically compose, configure, and deploy the necessary application that consists of a number of different components. If we consider the deployment requirements of a web application service provider, it will include security devices (e.g., firewall), load balancers, web servers, application servers, database servers, and storage devices. Setting up such a complex combination of appliances is costly and error prone even in traditional hosting environments [1], let alone in Clouds. Virtual appliances provide an elegant solution for this problem. They are built and configured with a necessary operating system and software packages to meet software requirements of a user. In this paper, to simplify the process of selecting the best virtual appliance and unit (computing instance) composition, a novel framework is presented.

The framework proposes:

- An approach to help non-expert users with limited or no knowledge on legal and virtual appliance image format compatibility issues to deploy their services flawlessly. For this purpose, we first automatically build a repository of Cloud services in web service modeling language (WSML) [2] and then enrich it with experts' knowledge (lawyers, software engineers, system administrators, etc.) on the aforementioned constraints. The knowledgebase then is used for reasoning in an algorithm that identifies whether a set of Cloud services consisting of virtual appliance and units are compatible or not.
- A Cloud service composition optimization technique that allows non-expert Cloud users to set get user friendly ecommendations on the composition solution's prominence. The majority of end users avoid systems that incur complexity in capturing their constraints, objectives and preferences. An example of such systems is the one which require users to assign weights to their objectives. In this case, users have to find a way to prioritize their preferences and then map them to weights. After that, the system has to find out how precise users have gone through the process of weight assignment. To tackle this issue, a major objective of this research is to offer ranking system for Cloud service (i.e., virtual appliance and machine) composition that let users express their preferences conveniently using high-level linguistic rules. Our system then utilizes multi-objective evolutionary approaches and fuzzy inference system to precisely capture the entered preferences for ranking purpose.

## II. ARCHITECTURE

Our proposed architecture offers a unified solution that uniquely applies state of the art technologies of semantic services, agent negotiation, and multi-objective and

constraints optimization to satisfy the requirements of whole service deployment life cycle. The main goal of the architecture is to provide: ease of use for non-experts, semantic interoperability, more precise discovery and selection, more reliable service level agreement (SLA) monitoring, and automatic negotiation strategy. The proposed architecture is depicted in Fig. 1 and its main components are described below:

User portal. All services provided by the system are presented via the web portal to clients. This component provides graphical interfaces to capture users requirements such as software, hardware, quality of service (QoS) requirements (including maximum acceptable latency between tiers, minimum acceptable reliability, and budget), firewall, and scaling settings. In addition, it transforms user requirements to web service modeling ontology (WSMO) format in the form of goals which are then used for Cloud service discovery and composition. For more details regarding the format of goals, readers can refer to our previous work. Moreover, it contains an account manager, which is responsible for user

1. Translator. Since web service modeling ontology is used for service discovery, Cloud services information is translated to web service modeling language format by the Translator component. This component takes care of building and maintaining an aggregated repository of Cloud services and is explained in detail.

2. Cloud service repositories. They are represented by appliance and virtual unit service repositories and allow IaaS providers to advertise their services. For example, an advertisement of a computing instance can contain descriptions of its features, costs, and the validity time of the advertisement.

   From standardization perspective, a common metamodel that describes IaaS provider's services has to be created. However, due to the lack of standards, we developed our own metamodel [3] based on previous works.

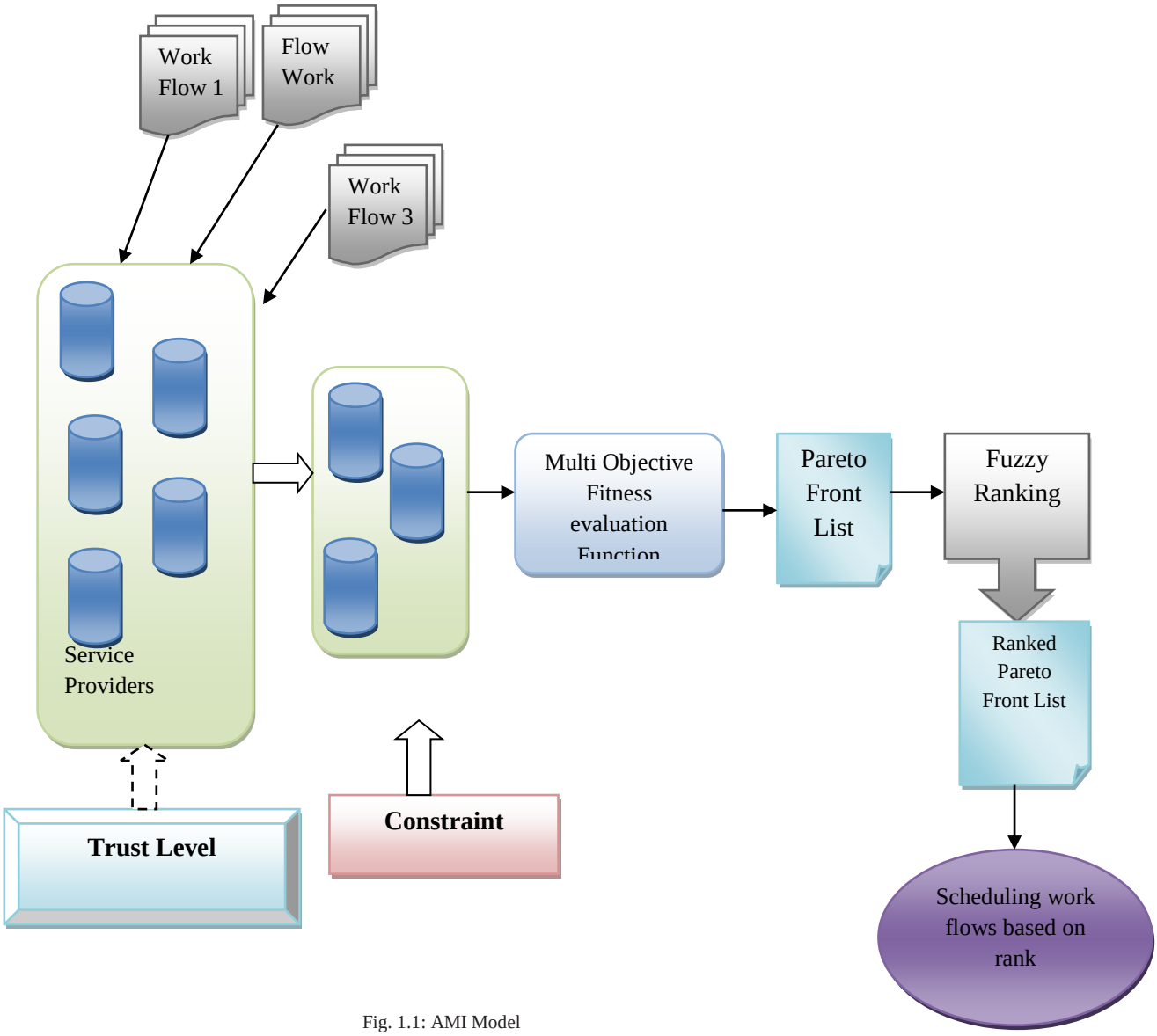


Fig. 1.1: AMI Model

3. Discovery and negotiation service. This component maps user's requirements to resources using the ontology-based discovery technique. It acts in user's interest to satisfy quality of service requirements by selecting the set of eligible IaaS providers. The negotiation service uses a time-dependent negotiation strategy that captures preferences of users on QoS criteria to maximize their utility functions while only accepting reliable offers. These negotiation strategies are described in our previous work.
4. The composition optimizer takes advantage of multi-objective algorithm and fuzzy logic to let users express their preference conveniently, and efficiently selects the closest candidates to users' interests. Throughout the paper we show how ontology-based compatibility checking, multi-objective algorithms, and fuzzy logic techniques can work in harmony to provide an elegant solution to the composition problem.
5. Planning. The planning component determines the order of appliance deployment on the selected IaaS

providers and plans for the deployment in the quickest possible manner.

## III. EVALUATION OF COMPOSITION CRITERIA

The composition problem is to find the best combination of compatible virtual appliances and virtual machines that minimizes the deployment cost (DC) and deployment time (DT), and maximizes the reliability while adhering to omposability constraints. A formal description of the problem is given below.

*3.1 Composition based Trust-driven scheduling algorithm:*

**Input:** workflowList (workflows requesting for cloud services), providerList (cloud providers), tLevel (trust level in the transaction).

**Output:** Composition of cloud service scheduling results.

**Main steps:**

1. Delete unqualified providers from providerList according to current trust level tLevel and generate reliable suppliers list named trustProviderList;
2. Calculate the QoS and price factor's for given work flow
    a) Calculate each workflow's service tolerate price QTP (QoS tolerate price) in workflowList according to DQoS (Described QoS) vector;
    b) Calculate each reliable provider's resource provision price QPP (QoS Provision price) in trustProviderList according to PQoS vector;
    c) Calculate SDS (Service demand similarity) using the Euclidean distance method and generate demand service similar matrix V.
3. Search for each workflow, its set of highest similarity provider
4. Evaluate valid composition by checking compatibility against the parameters Composition ( c ), Constraint List ( cl ).
    a) if CompositionValidity ( c, cl ) Exists in cache then
        i. ValidComposition=GetComposition ValidityFromCache ( c, cl )
    b) end
    c) ValidComposition=True;
    d) Foreach workflow w and service provider s in c do
    e) End
    f) InsertCompositionValiditytoCache ( c, cl);
    g) Return ValidComposition;
5. Bind the workflow with the provider's resource set if the provider's trust is higher

*3.2. Cloud Setup*

The cloud setup will be done by using the cloudSim toolkit. The number of cloud service providers will be created who will provide the carious services to the users. And then the user requirement gathering module will be created through which one will gather the information about the tasks for which they want to utilize the cloud environment. Then the needed virtual environment will be created for processing the user queries through the network.

*3.3. Trust Evaluation*

The trust value will be calculated. Since trust relationships are classified into direct trust and recommended trust, the computation of trust should also be treated differently according to its obtained method direct or indirect. For cloud entities, they should calculate integrated trust also. Current trust models always use simple weighted average method to compute the overall trust.

The computation method can be abstracted as follows:

$$IT = \alpha \times DT + (1 - \alpha)RT$$

Here DT is the direct trust degree, RT is recommended trust degree, $\alpha$ and $(1-\alpha)$ is the weight of DT and RT respectively.

*The computation of direct trust*

The direct trust is obtained by direct transaction history between cloud entities. The computation method of direct trust (cloud entity A–B in transaction context tc).

$$DT(A, B, tc) = SuccessTradeNum / TotalTradeNum$$

Here SuccessTradeNum represents the success transaction times and TotalTradeNum means the total transaction times.

*The computation of recommended trust*

When cloud entities want to trade with unfamiliar ones, they should use or combine recommended trust to aid the trust decision. The method of compute recommended trust is.

$$RT(A, B, tc) = \frac{\sum_{p \in \Omega} DT(A,P,Recomm) * DT(P,B,tc)}{Length\ (\Omega)}$$

Here Ω represents the transaction recommendation set that entity trust.

### *3.4 Multi-objective evaluation of work flows*

In our work we consider three user objectives, the lowest cost, quickest deployment time, and the highest reliability. This makes it infeasible to find an optimal composition as these objectives can conflict with each other. One way to address this problem is to convert the appliance composition problem to a single-objective problem by asking users to give weights for all the objectives. However, this approach is error-prone and impractical, as not all the users have the knowledge to accurately assign weights to objectives. Furthermore, since the composition solutions will depend on the capability of users to assign proper weight to the objectives, additionally we have to find a way to evaluate the knowledge of users about each objective to ensure the accuracy of the approach.

### *3.5 Fuzzy ranking for the reliable cloud services*

Our proposed fuzzy inference engine includes three inputs and one output. Inputs of the system are normalized deployment time, deployment cost, and reliability of composition, which are all described based on the same membership functions. Output of the fuzzy engine represents how desirable the current set of inputs are based on the fuzzy rule-based indication. It shows the membership function for output by which we allow the gradual assessment of the membership of elements in a set. For example, the value "0" in output means the solution is highly undesirable whereas the value "1" shows that the solution is highly desirable. Fuzzy rules should be defined by the user to describe their preferences. For example a rule can be defined as: if DT is low and DC is low and Reliability is high, composition is highly desirable.

### *3.6 Scheduling workflows*

When more than one workflow (belonging to different cloud users) ask for cloud services at the same time, it binds the user (workflow) to the most similar provider's resource set in SDS to achieve the target of providing on-demand services. Trust-based multi-targets QoS workflow scheduling algorithm named C Muti-QoS Schedule is as follows:

**Input:** workflow List (workflows requesting for cloud services), provider List (cloud providers), t-Level (trust level in the transaction).

**Output:** scheduling results.

*Main steps:*

• Delete unqualified providers from provider List according to current trust level tLevel and generate reliable suppliers list named trust Provider List;

• Calculate each workflow's service tolerate price QTP in workflow List according to D QoS vector;

• Calculate each reliable provider's resource provision price QPP in trust Provider List according to P QoS vector;

• Calculate SDS using the Euclidean distance method and generate demand service similar matrix V.

• Search for each workflow its highest similarity provider and bind the workflow with the provider's resource set if the provider's trust is higher than trust threshold.

Trust-based multi-workflow scheduling algorithm uses trust mechanism to filter dishonest providers and then binds workflow to the provider whose service capability is closest to their demand.

## IV. PERFORMANCE EVALUATION

In this module, performance evaluation of our work is done by comparing the proposed methodology with the existing methodology by using the performance metrics of trust value, QoS satisfaction.

## V. RESULTS

There are three main experiment results presented in this section: 1) an investigation of performance of the translator component, 2) a performance comparison between the OMOPSO, NSGA-II, and SPEA-II algorithms for the real case study, and 3) examination of the effectiveness of fuzzy inference system for handling imprecise user preferences. NSGA-II and SPEA-II algorithms use a population of size of 100, and maximum number evaluations of 25,000. OMOPSO is configured with 100 particles, with a maximum of 100 leaders and maximum number of iterations of 250. We have carried out 40 independent runs per experiment and then statistically analyzed results.

## VI. CONCLUSION AND FUTURE WORK

In order to provide the better cloud service with the agreed QoS level, in this work, the Trust value based workflow scheduling by using the cloud service composition is introduced. The cloud service composition is nothing but the integrating the services from the more than two cloud

service providers for satisfying the cloud users requirements. To ensure the QoS satisfaction level, in this work, trust value calculation is introduced which ensures the execution success rate of critical tasks and the introduction of two-level based scheduling mode reduces the problem scale of workflow scheduling. Multi-workflow scheduling helps to realize the QoS requirements analysis and service customization as the unit of user. The experimental results prove that the proposed methodology can provide the better result than the existing methodology.

However, the testing and application of this program has still been in the simulation experimental stage till now. So in the future, it will be feasible to build a small cloud prototype system and realize the deployment of our strategy in the real environment to test its efficiency and effectiveness.

REFERENCES

1. Huankai Chen , Frank Wang , Na Helian, “A Cost-Efficient and Reliable Resource Allocation Model Based on Cellular Automaton Entropy for Cloud Project Scheduling”, International Journal of Advanced Computer Science and Applications, 4 (4). pp. 7-14. ISSN 2156-5570
2. Anupam Das and Mohammad Mahfuzul Islam, ”SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems”, Dependable and Secure Computing, IEEE Transactions on (Volume:9 , Issue: 2 ), 19 January 2012
3. Xiaoyong Li, Feng Zhou, and Xudong Yang,”Scalable Feedback Aggregating (SFA) Overlay for Large-Scale P2P Trust Management”, Parallel and Distributed Systems, IEEE Transactions on (Volume:23 , Issue: 10 ), 24 August 2012
4. Guanfeng Liu, Yan Wang, Mehmet A. Orgun, and Ee-Peng Lim,”Finding the Optimal Social Trust Path for the Selection of Trustworthy Service Providers in Complex Social Networks”, Services Computing, IEEE Transactions on (Volume:6 , Issue: 2 ), 31 May 2013
5. Zheng Yan, and Christian Prehofer, ”Autonomic Trust Management for a Component-Based Software System”, Dependable and Secure Computing, IEEE Transactions on (Volume:8 , Issue: 6 ), 12 September 2011
6. Yan Wang and Lei Li, ”Two-Dimensional Trust Rating Aggregations in Service-Oriented Applications”, Services Computing, IEEE Transactions on 14 November