MULTI CLIENT MIDDLEWARE FOR SERVICE COMPOSITION THROUGH SAAS CLOUD

Mrs. T. Kavitha[1], Sravani Gorre[2], Jogu Swetha[3], Mudavath Mounika[4]

Article History: Received: 07.05.2023 **Revised:** 19.06.2023 **Accepted:** 14.07.2023

Abstract

Numerous services with varied Quality of Service (QoS) values that are functionally comparable are offered in the software as a Service (SaaS) cloud marketplace. The engineers are responsible for selecting the appropriate services and optimizing the service composition plans for each user group in order to handle the multi-dimensional QoS and functional needs of end users. The numerous strategies sometimes overlook the necessity of providing alternative execution plans, each of which offers a tailored plan for a certain customer with its functionality, QoS and cost requirements. This results in a dynamic issue that is NP-hard. We provide a unique multi-client middleware for dynamic service construction in the SaaS cloud that incorporates additional non-functional needs. The proposed middleware aims to empower clients to create innovative and tailored solutions by leveraging the vast array of services available in SaaS cloud environment efforts and accelerates time-to-market for new applications.

Keywords: Quality of Service (QoS), Software as a Service (SaaS).

[1]Assistant Professor, Department of CSE, Sridevi Women's Engineering College, Hyderabad, Telangana.
[2,3,4] Undergraduate Student, Department of CSE, Sridevi Women's Engineering College, Hyderabad, Telangana.

Email: [1]kavitha.tirunagiri@gmail.com, [2]gorresravani12@gmail.com, [3]jswetha61@gmail.com, [4]m.mounika9933@gmail.com

DOI: 10.31838/ecb/2023.12.s3.694

1. INTRODUCTION

The rapid growth of cloud computing and the adoption of Software as a Service (SaaS) models have revolutionized the way software applications are developed and deployed. SaaS cloud environments offer a vast array of services that can be accessed and utilized on-demand over the internet, providing scalability, flexibility and cost- efficiency to businesses and users.

However, the process of service composition can be complex and time- consuming, requiring expertise in service integration, data transformation and security. The middleware system aims to simplify the process of service composition by providing a unified interface and managing the interactions between clients and service providers. It acts as an intermediary layer, abstracting the complexities of service integration and enabling clients to focus on their specific requirements rather than the underlying technical details.

The proposed middleware leverages the advantages of SaaS cloud environments, such as scalability, reliability and cost- efficiency to facilitate seamless service composition experiences. It takes advantage of the abundant services available in the cloud and provides mechanisms for service discovery and selection, service orchestration and choreography, data transformation and integration, and security and privacy.

By providing a comprehensive solution for service composition, the Multi Client Middleware for Service Composition through SaaS Cloud aims to empower clients to leverage the full potential of SaaS offerings. It simplifies the development process, reduces time-to-market for new applications and enables clients to create innovative and tailored solutions by combining and orchestrating servicesavailable in the cloud.

2. RELATED WORK

Combining existing services in web service composition creates new capabilities. Different quality-of-service (QoS) qualities are present among the various services that are offered. A multi-criteria NP-hard task, building a QoS optimal web service composition is challenging. By combining many criteria into a single global score, the majority of existing techniques reduce this problem to a single-criterion problem. Scalarization does have some significant downsides, too. For example, there is no assurance that the aggregated results would fit the end user's preferences and limitations about the intended solutions. Additionally, a convex weighted sum cannot be optimized to reach non-convex portions of the Pareto set. Use of Pareto- based strategies, which allow for a more precise selection of the end-user solution, is an option. However, to far, no comparison analysis has been published and just a few solutions based on these methodologies have been put out. This inspired us to analyse a number of cutting-edge multi-objective evolutionary algorithms. Numerous scenarios of various degrees of complexity are taken into account. In order to compare various evolutionary algorithms, performance measurements are utilised. Results show that the GDE3 method produces the greatest results on this problem while also using the least amount of time.

One of the key challenges in leveraging SaaS cloud environments is the efficient composition of multiple services to create complex and customized solutions that meet specific requirements. Service composition involves integrating and orchestrating individual services to deliver a cohesive and value-added outcome.

The two sets of agents that naturally occur in MOEA/D are sub problems and solutions. A match between sub problems and solutions may be seen in the selection of potential solutions for sub problems. Economics has suggested that stable matching can be used to settle conflicts of interest amongst egotistical market participants. Different scenarios with various levels of complexity are taken into account. In order to compare various evolutionary algorithms, performance measurements are utilised. Solution show that the GDE3 method produces the best results on this task, as well as the least amount of time complexity.

It is common practise to combine a number of already-existing services to create complicated services. Many service providers have started to provide candidate services with identical

functionality but differing QoS levels because to the growing number of services being delivered in cloud computing environments. QoS-aware service composition has therefore received a lot of attention. However, QoS reliance does occur in reality and has an influence on the composite services' total QoS values. Using vector ordinal optimisation methods, a proposed pruning algorithm is created. Experiments in simulation are carried out to verify the efficacy and efficiency of algorithms. Quantifying QoS reliance allows for the discovery of truly desirable solutions.

The technique of automatically creating a workflow from separate services in order to meet user needs is known as service composition. It is crucial to take both functional and non-functional needs into account while creating service processes. The so-called QoS-aware service composition is frequently phrased in a way that is similar to the standard MMMKP optimisation issue, and it does not take into consideration the service and QoS dependencies that are frequently crucial in real life. With regard to service-dependent QoS and QoS-aware service creation, we provide a novel method that dynamically adjusts the workflow in light of user- provided topological and QoS constraints as well as QoS dependencies. We demonstrate through testing that our method may greatly improve performance in real-world scenarios with challenging service needs and QoS constraints.

One of the technologies that is advancing the most quickly is the use of web services. We can utilise current services thanks to this composition, which saves money and effort. To maximise the overall Quality of Service (QoS) of the constructed service is one of the more recent issues with web service composition. Response time, availability, dependability, throughput, and cost (price) are the most often used QoS components.

Existing system:
Application engineers must select the best services and tailor service composition plans for each user group in order to meet the multidimensional QoS and functional needs of end users in the existing system. Existing systems for active service composition usually include execution plans that search for service providers of comparable functionality with different QoS or cost constraints in order to satisfy the QoS requirements of the customers or to vigorously adjust to changes in QoS. These strategies frequently overlook the importance of multi-client execution plans to offer a variety of plans, each of which gives a tailored solution for a certain client with their own functionality, QoS, and pricing requirements. Strong composition and selection of multi-client services are now NP-hard problems.

Disadvantages of the Existing system:
1. Existing strategies for dynamic service composition frequently give execution plans that look for service suppliers of comparable functionality with varying QoS or cost restrictions in order to satisfy the QoS needs of the customers or to dynamically adapt to changes in QoS.
2. Additionally, unforeseen client requests, partner service issues, or environmental changes may have an impact on the execution of the service or make it unavailable.

Proposed system
In the SaaS cloud, a revolutionary multi-client middleware for effective service composition is used. This increases the efficiency of providing the services to clients within all the functionalities. Services with different quality can be achieved at a single instance without latency issues. Security and authentication play's a crucial role in providing service, where SaaS has achieved it through providing all the security parameters while providing services. Service Level Agreement (SLA's) are given to clients for the terms and conditions regarding the services. Through this middleware, data is efficiently secured and uploaded through cloud norms. Based on the user requests, the required parameters and services can be provided without any discrepancy. Locations can be changed without holding the server for too long.

This cutting-edge dynamic multi- client middleware imagines scenarios where different functionality can be provided, giving clients access to personalized service plans on demand or as soon as they become available. Using SaaS, the users are benefited with utmost security and provided with huge amount of

storage capacity. SaaS is the pinnacle of recurring revenue models and enables quicker software deployment than on-premises applications. Under the SaaS model, a third-party provider develops apps using cloud infrastructure and makes them online available to users. Users who have registered have proof thanks to authentication, but only authenticated usersshould log in.

Advantages of the proposed system:

1. Provision of services with different quality has been achieved. Service composition plans can be altered without any data loss & latency.
2. SaaS provides the utmost qualities of cloud and manages to store the data.
3. QoS parameter is considered, providing exact result.
4. Automatic service composition by planner is possible.
5. User requests and data are securely uploaded and encrypted within thecloud.
6. Multiple clients can access the services without any difference inQoS parameters.
7. Enables developing distributed applications by composing existing services. Capable to react proactively.
8. In case of changes, service execution plan can be recomposed.
9. SaaS cloud requires no other software or hardware versions.
10. Provides API integration. Low maintenance required for SaaS. One to many access. SaaS is scalable, cost effective.

System architecture

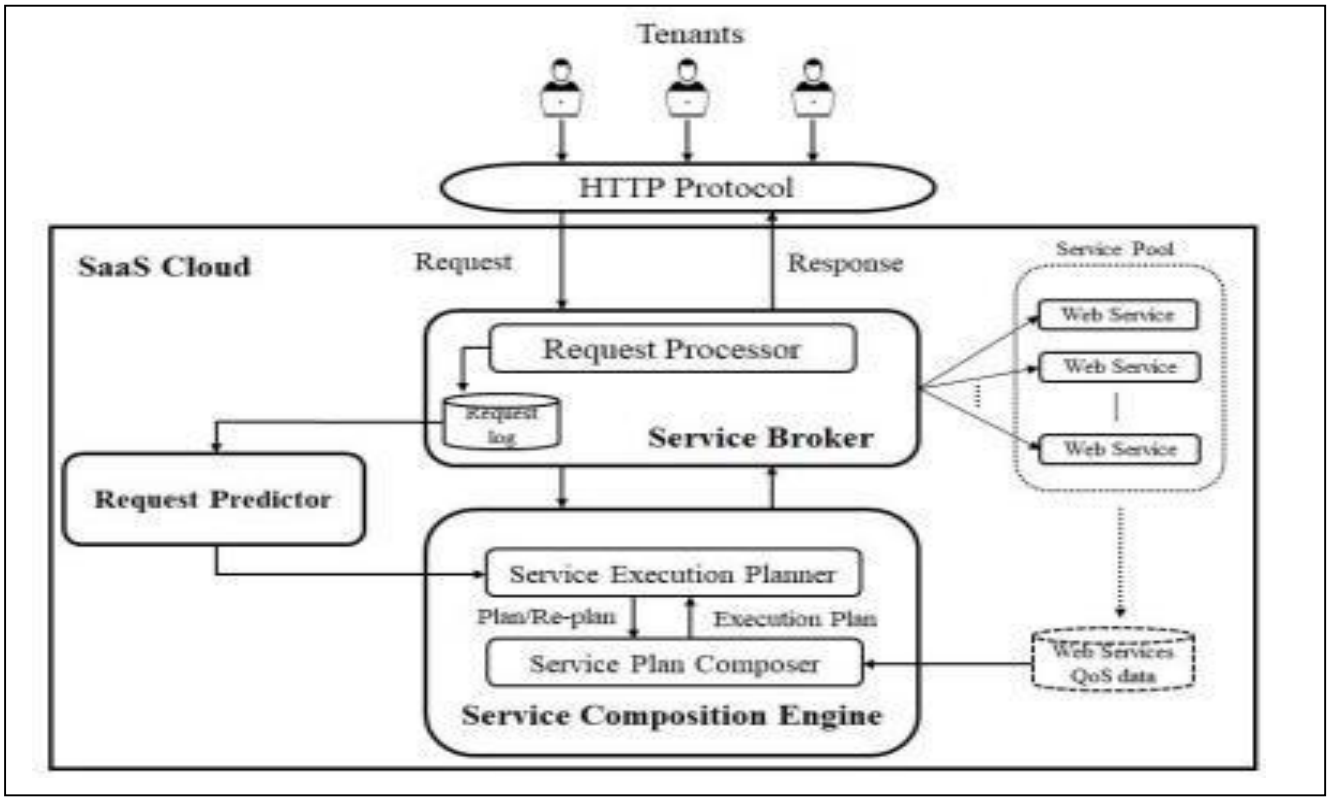


Fig-1: Architecture

Clients in the SaaS cloud environment want for services that come in various packages. A request processor handles requests sent via HTTP. To create application execution plans for the customers for each category, it uses the service composition engine. Request predictor estimates future requests by analysing the historical pattern of client requests made through the composite service. It responds to adjustments made during the execution of the composite service and reassembles the service execution plan to ensure the QoS that the clients have requested. Additionally, it receives the predicted future demand for requests from the request predictor and continuously monitors how incoming requests from

customers are behaving in relation to the present execution plan. Decisions are made proactively to maintainSLA violations.

Modules

1. User: In this module, the user will get registered, login, add servers, upload files, update location and then view all the files and updated data. Multiple users at a time can request for a service to the cloud. The composition of services is designed in such a way that all the users get the best quality of service for respective requests.

2. Cloud:

Cloud will activate the users from waiting stage, so that the user can perform the operations. Here the cloud can also view the servers, files. In the real time cloud, the encrypted files are stored for security purpose. These operations are composed in an instance without delay as the middleware supports the multi-client services.

3. Encryption:

Encryption is a process of generating cipher text from raw data. The need for encryption is to keep the data safe. An intruder should not be able to view the original data. In this middleware, Advanced Encryption Standard algorithm is used for encryption of files. It performs 10 rounds and generates keys for each round which are useful in encrypting and decrypting the data. Though it has many rounds, it is efficient to maintain large amount of data. Large scale organizations need to follow up with encryption standards as there is a need for keeping data safe.

4. Service composition:

Provides various services with Quality-of-Service constraints and Service Level Agreement. Services can be recomposed by the service composition engine and service execution planner. These are executed when the service requests are above the defined level and need more constraints to provide the services. SaaS cloud gives the entire components, platform, and resources readily available to the user, where there is nothing that is to be managed by the user. All the components are inbuilt and managed. SaaS provides the on- demand availability of system resources, data storage and computing power. Through the characteristics of SaaS, the services are dynamically composed whenever there is a change. It is a collection of services where; many smaller servicesare combined together to a larger service.

Algorithm

Advanced Encryption Standard (AES):

Sensitive data is frequently protected using the symmetric encryption method known as the Advanced Encryption Standard (AES). It was chosen in 2001 to take the role of the outdated Data Encryption Standard (DES) by the National Institute of Standards and Technology (NIST) of the United States.

AES works on blocks of data, each of which has 128 bits. Although the block size is constant, the key size can be one of three distinct sizes: 128 bits, 192 bits, or 256 bits. The encryption is more robust the greater the key size.

A sequence of encryption keys produced from the original key are used to execute a number of rounds of data transformations, including substitution, permutation, and mixing operations, on the data. A 128-bit key requires 10 rounds, a 192-bit key requires 12, and a 256-bit key requires 14 rounds. The number of rounds is dependenton the key size.

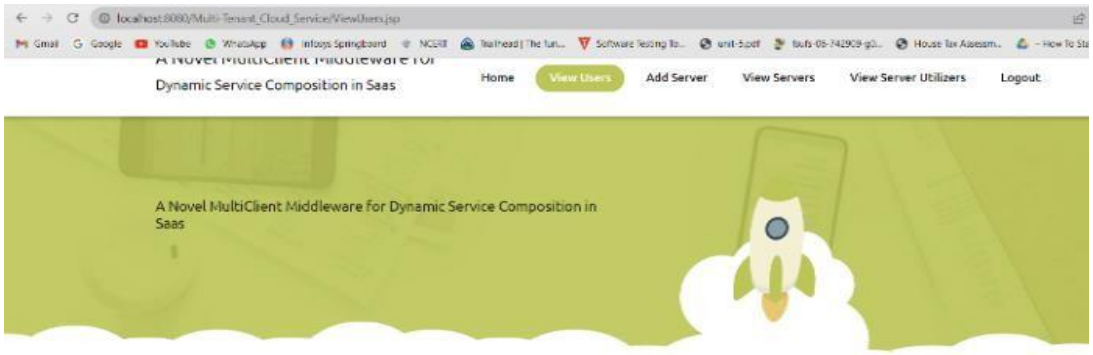
Due to its security, sufficiency, and broad acceptance, AES has emerged as the de facto industry standard for symmetric encryption. It is used for many different purposes, including maintaining the confidentiality of sensitive information in different businesses, securing sensitive data in communication protocols, and securing sensitive data at rest.

While AES is a powerful encryption technique, it's vital to remember that a system's total security also depends on elements like key management, secure implementation, and defence against side- channel attacks.

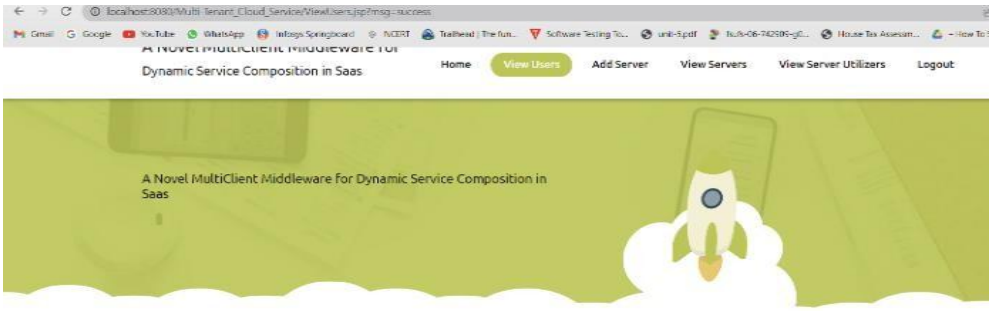
Test cases:

NAME OFTHE TEST	INPUT	OUTPUT	RESULT
Unit testing ofdata	The user givesthe input inthe form of text file.	An outputis file uploaded to server successfully.	A resultis file uploaded to server successfully.
Unit testing for activation	The cloud activated user account	An output is activateduser account successfully.	A resultis activated useraccountsuccess fully.
Unit testing updating	The user change server locationto another.	The userchange server locationto another.	A resultis server location change d successfully.
Integration testing	The user givesthe input inthe form of userna me and password.	An output is user login successfully.	A resultis user login successfully.
Data Flowtesting	The user givesthe input inthe form of txt file.	An output is file uploaded to server successfully.	A resultis file uploaded to server successfully.
User Interface testing	The user change server locationto another.	An outputis server location changed successfully.	A resultis server locationchange d successfully.
User interface Testing- Event based	Theuser login into applicat ion using userna me and password.	An output is user login successfully.	A resultis usersuccessfully login toapplication.
User interface Testing- Event based	The user change server locationto another .	An outputis server location changed successfully.	A resultis server location change d successfully.

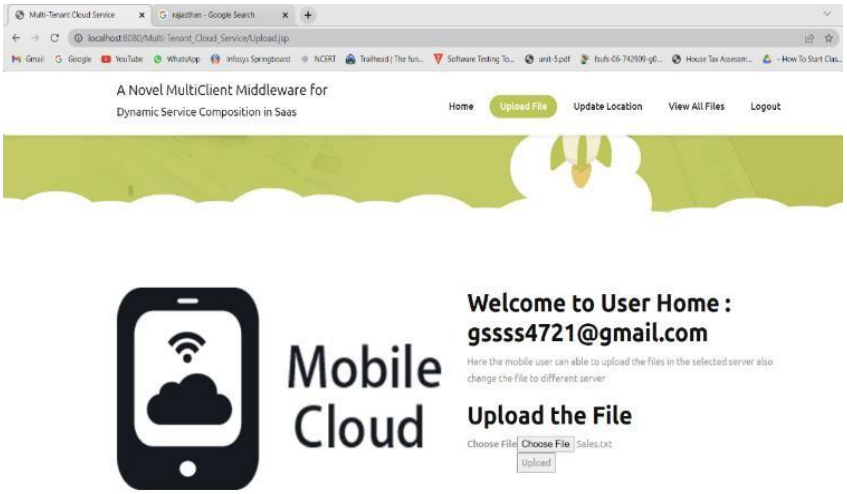
2. Results



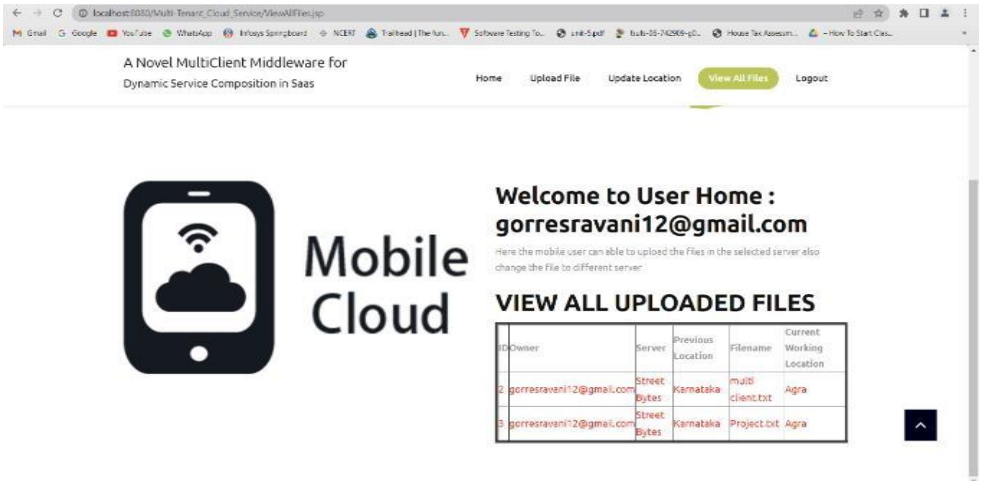
User waiting to be activated



User activated



Uploading file



Location has been updated

3. Conclusion & Future Scope

In SaaS cloud environments, multiclient middleware for dynamic service creation is essential for providing flexible and effective services to numerous clients. The abstract architecture presented in this discussion showcases the importance of service discovery, composition and orchestration mechanisms in achieving seamless integration, resource utilization and customization in a multi-tenant cloud environment.

By leveraging dynamic service composition, multiclient middleware enables clients to combine and reuse existing service components, resulting in reduced development time and costs. The middleware optimizes resource allocation and load balancing, ensuring scalability and efficient utilization of computing resources. Additionally, it provides a standardized interface for clients to access and compose services, abstracting the complexities of theunderlying infrastructure.

The field of multiclient middleware for dynamic service composition in SaaS holds several possibilities for future research and development. Some potential areas of future exploration include:

Advanced Composition Techniques: Further research can focus on developing advanced techniques for service composition, such as machine learning- based approaches or intelligent algorithms that consider client preferences, historical data and real-time conditions to enhance thecomposition process.

Autonomic Resource Management: Future efforts can concentrate on designing self-adaptive and autonomic resource management mechanisms within the multiclients middleware.

Thes e mechanisms can dynamically adjust resource allocations based on varying workloads, optimizing resource utilization and minimizing costs.

QoS- Aware Composition: Enhancing the QoS- awareness of the multiclient middleware can enable it to consider and optimize for different quality attributes, such as response time, reliability and scalability, while composing services. This can contribute to improved service-level agreements (SLAs) and client satisfaction.

Security and privacy: As security and privacy concerns continue to grow, future research can focus on developing robust security measures within multiclient middleware, including techniques for secure communication, data encryption, and access control, to protect client data and ensure privacy.

Integration with emerging technologies: Multiclient middleware's incorporation with cutting-edge technologies like edge computing or block chain, can open up new avenues for efficient and decentralized service composition, enabling clients to leverage the benefits of distributed computing and immutable ledgers.

Performance evaluation and benchmarking: Conducting comprehensive performance evaluations and benchmarking studies of multiclient middleware solutions can provide valuable insights into their scalability, efficiency and effectiveness in real-world scenarios further improving their design and implementation.

In conclusion, the future of multiclient middleware for dynamic service composition in SaaS is promising, with potential advancements in composition techniques, resource management, security and integration with emerging technologies. By addressing these areas of future research, we can further enhance the capabilities and benefits of multiclient middleware, ultimately delivering more efficient and tailored services to multiple clients in the SaaS cloud domain.

4. References

1. Jula, E. Sundararajan, and Z. Othman," Cloud computing service composition: A systematic literature review", Expert Systems with Applications, vol. 41, no. 8, pp. 3809 – 3824, 2014.
2. A. Ramrez, J. A. Parejo, J. R. Romero, S. Segura, and A. R. Corts," Evolutionary composition of QoS- aware web services: a many-objective perspective", Expert Systems with Applications, Elsevier, vol. 72, pp. 357– 370, 2017. [13] A. E. Yilmaz and P. Karagoz," Improved genetic algorithm- based approach for QoS aware web service composition". IEEE International Conference on Web Services, pp. 463–470, 2014.
3. C. Jatosh, G.R. Gangadharan, U. Fiore, R. Buyya," QoS-aware Bigh service composition using MapReduce based evolutionary algorithm with guided mutation", Future Geneation Computer System, Elsevier, pp. 1008-1018, 2018.
4. F. Chen, R. Dou, M. Li, and H. Wu," A flexible QoS-aware Web service composition method by multi-objective optimization in cloud manufacturing", Computers and Industrial Engineering, Elsevier, vol. 99, pp. 423–431, 2016.
5. [5] K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang," Stable matching-based selection in evolutionary multi- objective optimization", IEEE Transactions on Evolutionary Computation, vol. 18, no. 6, pp. 909923, 2014.
6. Gartner Inc." Gartner Forecasts Worldwide Public Cloud Revenue to Grow 21.4 Percent in 2018", https://www.gartner.com/newsroom/id/ 3871416, [Online; accessed 20-July-2018], 2018.
7. H. Wada, J. Suzuki, Y. Yamano, and K. Oba," E3: A multi-objective optimization framework for SLA-aware service composition", IEEE Transactions on Services Computing, vol.5, no. 3, pp. 358-372, 2012.
8. L. Zeng, B. benatallah, A.H.H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang," QoS-Aware Middleware for Web Service Composition", IEEE Transactions on Software Engineering, vol. 30, pp. 311-327, 2004.
9. M. Cremene, M. Suciu, D. Pallez, and D.D Dumitrescu," Comparative analysis of multi-objective evolutionary algorithms for QoS-aware web service composition", Applied Soft Computing, Elsevier, pp. 124-139, 2016.
10. Q. He, J. Han, Y. Yang, J. Grundy, and H. Jin," QoS-driven service selection for multi-tenant SaaS", IEEE International Conference on Cloud Computing,pp. 566–573, 2012.
11. S. Liu, Y. Wei, K. Tang, A. K. Qin, and X. Yao," QoS-aware long-term based service composition in cloud computing", IEEE Congress on Evolutionary Computation, pp. 3362-3369, 2015.
12. Salesforce." Sales Cloud Edition Report". https://www.salesforce.com/products/s ales-cloud/pricing/. [Online; accessed 20-July-2017], 2017.
13. Y. Wang, Q. He, and Y. Yang,"" Qos-aware service recommendation for multi-tenant saas on the cloud", IEEE International Conference on Services Computing, pp. 178-185, 2015.
14. Z. Ye, X. Zhou, and A. Bouguettaya," Genetic Algorithm Based QoS Aware Service Composition in Cloud Computing", International Conference on Database Systems for Advanced Applications, pp. 321-334, 2010.