International Journal of Information Technology and Management

ISSN online: 1741-5179 - ISSN print: 1461-4111
https://www.inderscience.com/ijitm

AQINM: an adaptive QoS management framework based on intelligent negotiation and monitoring in cloud

Zeng Saifeng

DOI: 10.1504/IJITM.2024.10061681

Article History:

Received:	09 October 2019
Last revised:	30 October 2020
Accepted:	01 November 2020
Published online:	22 January 2024

AQINM: an adaptive QoS management framework based on intelligent negotiation and monitoring in cloud

Zeng Saifeng

School of Computer and Communication,
Hunan Institute of Engineering,
Xiangtan 411104, China
Email: zsf4623@126.com

Abstract: More and more federated cloud platforms have been constructed to deal with non-trivial large-scale applications, which typically require certain level of quality-of-service (QoS) guarantee. However, most of existing cloud-oriented QoS solutions are likely to introduce extra overheads on either resource allocation or task execution, which is especially true in federated cloud environments. In this paper, we design and implement a QoS-enhancing framework, namely adaptive QoS management based on intelligent negotiation and monitoring (AQINM), which provides three QoS-enhancing services including policy management, service level agreement (SLA) negotiation, and SLA monitoring. Unlike the conventional QoS-enable middleware, these services in the AQINM framework introduce several novel mechanisms to offer more cost-effective and efficient solutions to enforcing the QoS management in federated clouds. The implementation of our AQINM framework are tested in a campus federated cloud platform by using different applications as experimental benchmarks, and its performance are compared with other similar solutions. The experimental results show that the proposed AQINM is capable of reducing the costs of SLA negotiation and monitoring for large-scale cloud application that deployed in federated cloud environments.

Keywords: cloud computing; quality-of-service; QoS; service level agreement; SLA; resource virtualisation.

Reference to this paper should be made as follows: Saifeng, Z. (2024) 'AQINM: an adaptive QoS management framework based on intelligent negotiation and monitoring in cloud', *Int. J. Information Technology and Management*, Vol. 23, No. 1, pp.33–47.

Biographical notes: Zeng Saifeng received his Masters and Doctors degree from the Communication University of China. Currently, he works in Hunan Institute of Engineering as the Dean of Communication Department and a Senior Networking Engineer in HPC Datacentre. His research interests include cloud computing, parallel and distributed storage technology, software engineering and middleware development. He is now a member of IEEE Computer Society and also a senior member of CCF in China.

1 Introduction

More and more federated cloud platforms have been constructed to deal with non-trivial large-scale applications (Duan and Vasilakos, 2016; Huedo et al., 2017). In these federated cloud platforms, loosely-coupled resources span over multiple geographically virtual organisations and are virtualised in variety of virtual machines (VM) to execute up-level tasks (Wen et al., 2017; Sun et al., 2018). Due to the uncertainty of underlying resources, the delivered quality-of-service (QoS) in federated cloud platform are likely to dynamically changed, which in turn increasing the difficulty of resource management and task scheduling (Quarati et al., 2016; Anastasi et al., 2017). More importantly, the introduction of virtualisation technology brings more challenges on QoS management in federated clouds (Li et al., 2014; Ye et al., 2016; Homsi et al., 2017). For instance, different VM instances may use resources from different virtual organisations where the resource management policy are quite different from each other, while the global VM manager needs a common standard to evaluate the QoS performance delivered by the active VM instances. Therefore, an effective cloud-oriented QoS management framework needs to be able to provide a flexible policy configuration service when allocating and managing VM instances.

Meanwhile, to enforce QoS in federated cloud platforms, service level agreement (SLA) negotiation mechanism has been widely implemented in most popular cloud middleware (Vilaplana et al., 2015; Rane and Sarma, 2015; Trapero et al., 2017). However, most of existing SLA negotiation services are lab-intensive and tend to introduce more extra costs either on resource allocation or task scheduling, which finally will mitigate the overall QoS performance from the perspective of cloud users. In addition, SLA violations are more likely to occur in real-world federated cloud platforms (Panda and Jana, 2017; Ranjbari and Akbari-Torkestani, 2018). For instance, with the increasing of networking traffics the probability of deadline missing will be increased rapidly. Therefore, an effective SLA monitoring service plays an important role in any QoS-oriented middleware. However, existing solutions of SLA monitoring service are generally cost-expensive since it has to frequently sample the statuses of a large amount of resources, which brings more networking and storage related costs (Halboob et al., 2015; Hayyolalam and Pourhaji-Kazem, 2018).

To deal with the abovementioned issues, in this study we present a lightweight QoS-enhancing middleware, namely *adaptive QoS management based on intelligent negotiation and monitoring* (AQINM), which provides three QoS-enhancing services including policy management service, SLA negotiation service, and SLA monitoring service. Unlike the conventional QoS-enable middleware, these services in the AQINM framework introduce several novel mechanisms to offer more cost-effective and efficient solutions to enforce the QoS management in federated clouds. The main contributions of this work are summarised as following:

1 In the AQINM, we introduce an adaptive policy management service, which allows resource providers to configure their individual resource management policy so as to coordinate the SLA management in multi-cloud environments.

2 We propose an intelligent SLA negotiation service which relies a set of self-learning agents to perform bilateral SLA negotiation with aiming at obtaining optimal tradeoffs between negotiation utility and cost.

3 In the AQINM framework, we incorporate a cost-effective SLA monitoring service which allows to dynamically change the frequency of sampling according to the previous QoS performance for any given resources.

The remainder contents are organised as following: In Section 2, we review the related works; then, we present the framework of our QoS-enhancing framework in Section 3; in Section 4, the results of experiments are analysed to evaluate the proposed framework. Finally, we conclude the work in this paper and briefly discuss our planning work in the future.

2 Related work

In early study, many QoS framework were designed for grid computing systems, where grid co-allocation service was often integrated with QoS service so as to ensure resource availability at runtime. For instance, Cao et al. (2010) proposed a QoS-based grid workflow system that allows the execution of workflows with different QoS levels. In Darby and Tzeng (2010), the authors designed a set of QoS-aware heuristics to form a Reliability-driven Framework, which was aiming at maximise the probability of data recovery during job execution in mobile grid systems. In Di-Stefano et al. (2011), the authors proposed an innovative QoS-aware service composition framework, which allows to create and manage P2P grid services while maintaining the QoS level during their execution. In Andronikou et al. (2012), the authors addressed the QoS-enable data replication service in grid, and introduced several interoperable file replication algorithms that take into account the infrastructural QoS constraints and the availability of the data. In Aron and Chana (2012), the authors designed a QoS-aware resource provisioning framework, which enables to adjust the provision policy that caters to provisioned resource allocation and resource scheduling. Our AQINM framework also provides a similar service while offers more flexible resource policy management. In Xiao and Han (2013), the authors proposed a temporal-spatial relaxing technique, which is based on an observation fact that grid applications are likely to overestimate their reservation time. This reservation technique allows grid administrators to obtain better tradeoffs between QoS performance and SLA violations. In Kianfar et al. (2015), the authors proposed a novel utility for defining user-oriented QoS metrics, which allows resource users and providers to express more options in SLA contracts. Also, it provides a simulated annealing algorithm which enables to adjust QoS metric sampling strategy in terms of geometric manner.

Recently, cloud-oriented QoS solutions have become more and more important, and plenty of QoS-aware framework and solution have been proposed. In Zheng et al. (2013), the authors proposed a QoS ranking and prediction framework which is based on the historical information obtained by running services. Its key advantage is that it requires no additional invocations of cloud services when making QoS ranking and prediction. In Kourtesis et al. (2014), the authors analysed that how to use semantics and ontologies to describe the QoS metrics as well as their interactions in complex distributed systems. Their study indicated that semantic-based QoS solutions are effective to deal with the vast heterogeneity of data sources or services based on domain knowledge. QRSF (Singh and Chana, 2014) is a QoS-aware workload management platform. It allows cloud workloads to be identified and analysed by K-means. QMaaS (Cicotti et al., 2015) is a

standalone middleware that offers QoS monitoring facility to up-level applications. In QMaaS, the monitoring service is implemented as a service paradigm. In Giunta et al. (2015), the authors argued that how to introduce a QoS-aware layer into the typical cloud-based web service systems. Their solution is based feedback controlling theory and allows to dynamical adjust the QoS measurement during the application's runtime. In Asyabi et al. (2016), the authors designed a QoS-aware hypervisor-level scheduler called KANI, which allows to monitor delivered QoS and quantify the deviation between desired and delivered QoS levels. With the help of KANI, cloud providers can determine how to allocate low-level resources among active VMs so as to meet the expected QoS. In Xue et al. (2017), a QoS-based task scheduling algorithm was proposed, which aims at achieving energy reduction under the constraints of QoS metrics. In Xu et al. (2018), the authors proposed an event-driven framework which can provide guaranteed QoS for MapReduce applications. In this framework, a novel QoS monitoring mechanism was embedded in the VM manager service, which can efficiently send QoS-related information to those participants that interested in them. Djemame et al. (2017) proposed a adaptive framework which provides an energy aware and efficient cloud operation methodology.

3 System framework

3.1 Overview of framework

Figure 1 demonstrates the AQINM framework that deployed in a classical federated cloud platform. In such a cloud environment, multiple organisations provision their resources through VM hypervisors and then form a global VM pool for running upper level applications; the cloud middleware is responsible for resource allocation and task scheduling, which needs to interact with the services in the AQINM framework. In our AQINM framework, there are three key services including policy management service, SLA negotiation service, and SLA monitoring service. The policy management service is designed to allow resource providers configure or adjust their individual resource management policy when provisioning virtualised resources to a cloud system; the SLA negotiation service is deployed to perform bilateral negotiation with aiming at maximise negotiating benefits based on the preferences of both resource providers and users; as to the SLA monitoring service, it is responsible for monitoring the runtime QoS performance of various kinds of resources in case of SLA violations. Unlike the conventional QoS-enable solution, the three services in the AQINM framework introduce several novel mechanisms to offer more cost-effective and efficient solutions to enforce the QoS management in federated clouds. In next sections, we describe the design and implementations of these services in details.

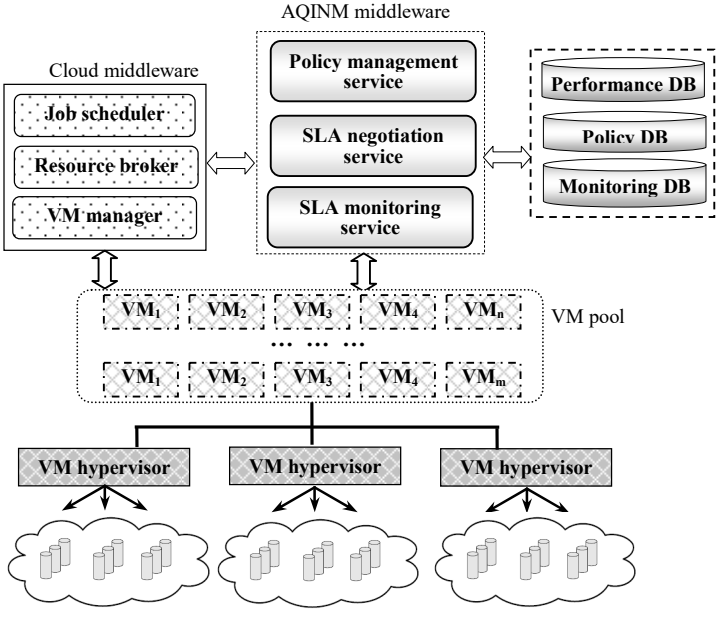
3.2 Adaptive policy management service

In large-scale cloud environments, effective resource management is a challenging task since various kinds of resources come from multiple virtual organisations, where different resource management policies may be applied based on different objectives. Therefore, it is impossible to enforce a single resource management policy from the perspective of a whole cloud system. On the other side, there must be a common

mechanism to enforce the guaranteed SLAs when allocating resources to user applications. So, we introduce an adaptive policy management service in the AQINM framework, which is responsible for enforcing the following functionalities:

- Allowing resource providers configure or adjust their individual resource management policy when provisioning virtualised resources to a cloud system.
- Providing a policy matchmaking mechanism for SLA negotiation service when conducting QoS negotiating operations.
- Interacting with resource broker and job scheduler to identify the most appropriate policy when performing resource allocation and task scheduling for current user application.

Figure 1 AQINM framework overview in cloud environment

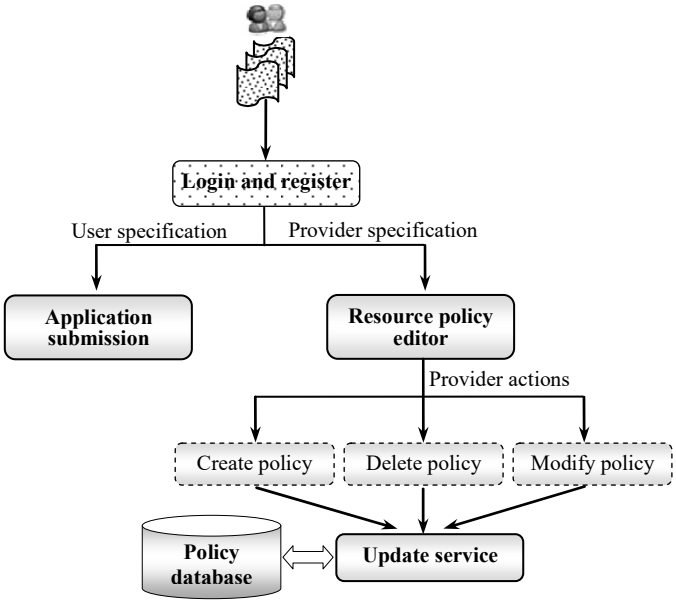


To realising the above functionalities, our adaptive policy management service needs to provide different view levels of resource management policy. For example, in local-level policy view, it may focus on the quantity and quality of physical resources that contributed to the cloud platform; in the organisation-level policy view, it may focus on the performance and regulation of VM instances. To express the policies at different levels, we define a novel schema based on the WS-policy specification, which allows individual resource providers to specify their resource management policies in a XML file. From the perspective of resource provider and cloud user, the flowchart of our adaptive policy management service works is demonstrated in Figure 2.

When receiving a request from a resource provider at the login and register portal, the policy management service firstly need to convert it into a WS-policy XML file; then, based on the provider's actions defined in this file, it support to create/delete/modify the policies in the policy database. If the request is from a cloud user, the policy management service will find out the appropriate resource policy, which will be compared against the

current application's requirements with aiming at testing the suitability of resources. Then the selected policy file will be sent to the job submission service, which is responsible for interacting with the underlying scheduler or resource allocation service. For instance, the resource allocation service may need this policy file to determine that whether a job should be executed in on-demand manner or real-time manner, while the scheduler may use it to identify the appropriate hosts that running the allocated VM instances.

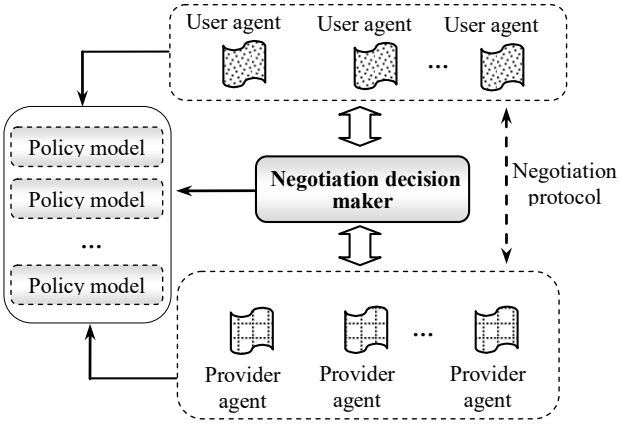
Figure 2 Flowchart of adaptive policy management service



3.3 SLA negotiation service

As a powerful mechanism for expressing resource requirements and restrictions, SLA has been widely used in variety of distributed systems to supporting QoS performance. In a federated cloud platform, user's requirements are generally defined in terms of SLA items, which will be negotiated between resource providers and users. In case of QoS violation, SLAs also need to specify the penalty when resource providers can not full satisfy the agreed SLAs. In a real-world cloud platform, the SLA management service typically consists of four components: SLA creation, SLA negotiation, SLA monitoring, and SLA enforcement. Among the four components, SLA creation and enforcement can be easily implemented according to the standard specifications, while the SLA negotiation and monitoring components should be carefully designed and implemented based on the overall goals of the target cloud platform. In our AQINM framework, we design an intelligent SLA negotiation service and cost-effective SLA monitoring service. Here, we present our SLA negotiation service, and the SLA monitoring service will be presented in the next section.

Figure 3 Flowchart of intelligent SLA negotiation service



In Figure 3, we demonstrate the design of our SLA negotiation service, in which a set of independent negotiation agents (NAs) are deployed to perform bilateral negotiation. As the agents are ignorant of its opponent's negotiation preferences, they are expected to maximise negotiating benefits based on their own preferences. The negotiation decision maker (NDM) component is responsible for selecting the most suitable strategy for NAs and performing SLA matchmaking operations. In order to separate the negotiating policy from the SLA matchmaking procedure, we design a set of negotiation policy models (NPMs) which is used by NAs for making negotiation decision. For each SLA term, we use the following utility function to evaluate its goodness:

$$U(x_i)=\begin{cases}\dfrac{x_i^{\max}-x_i}{x_i^{\max}-x_i^{\min}}, \textit{if } U(x_i) \text{is decreasing with } x_i \\ \dfrac{x_i-x_i^{\min}}{x_i^{\max}-x_i^{\min}}, \textit{if } U(x_i) \text{is increasing with } x_i\end{cases} \tag{1}$$

where $x_i \in [x_i^{\min}, x_i^{\max}]$ is the offered QoS value for a given SLA term. In order to reduce the negotiation cost, the NDM component introduces a new time-based negotiation decision function, which is defined as following:

$$\alpha(t)=\left(1+e^{-(t^{\max}-t)\cdot\beta}\right)^{-1} \tag{2}$$

where $t^{\max}$ is maximal negotiation time, t is the current time, β is the control parameter in the negotiation decision function. Therefore, for each SLA term, its negotiating value offered by agent j at time t can be evaluated as following:

$$x_i^j(t)=\begin{cases}x_i^{\min}+\alpha_i(t)\left(x_i^{\max}-x_i^{\min}\right), \text{if } U(x_i) \text{ is decreasing with } x_i \\ x_i^{\min}+\left(1-\alpha_i(t)\right)\left(x_i^{\max}-x_i^{\min}\right), \text{if } U(x_i) \text{ is increasing with } x_i\end{cases} \tag{3}$$

It is clear that function $\alpha(t)$ is in (0, 1) which defines the changes of an offered SLA term in subsequent offers during the negotiating period. As to the parameter β, a higher value of it implies a more conceding attitude of the NA, while a lower value of it implies a restrictive attitude during the negotiating procedure. By using the above time-based

negotiation functions, a user NA can start the negotiation with the highest utility value and gradually reduce it until an agreement is obtained or maximal negotiating time is passed. From the perspective of resource providers, an acceptable SLA term is not only decided by its current value but also be related with the user's preferences and negotiating attitude. So, we can say both providers and users can have a win-win negotiating results as long as the agreement is reached.

3.4 SLA monitoring service

Due to the uncertain performance of resources, SLA violations are likely to occur in many real-world cloud platforms. For instance, with the increasing of networking traffics the probability of deadline missing will be increased rapidly. Therefore, an effective SLA monitoring service plays an important role in any QoS-oriented middleware. In many existing solutions, SLA monitoring service is always a cost-expensive task since it has to frequently sample the statuses of a large amount of resources. To overcome this problem, our AQINM framework introduces a cost-effective SLA monitoring approach, which is based on the well-known observation that the SLA violation time of a given resource is far less than the execution time of applications. So, our SLA monitoring service allows to dynamically adjusting the sampling frequency based on the previous QoS performance for any given resources. In this way, we can not only reduce the monitoring overheads but also minimise the frequency of SLA violation.

To realise such a dynamical SLA monitoring service, it is necessarily to calculate the probability of SLA violation at first. Let $a_i(t)$ be the monitoring SLA value at time t, x_i be the value defined in SLA contract. It can be expected that if $a_i(t)$ increases quickly, then the probability of SLA violation (noted as $\Pr\{a_i(t+n) > x_i\}$) in the short-term future also increases. So, when using adjustable sampling frequency mechanism, the misdetection probability can be calculated as:

$$\Omega_i(f) = 1 - \prod_{k \in (1,f]} \left(1 - \Pr\left\{a_i(t_1) + k \cdot \delta > x_i\right\}\right) \tag{4}$$

where f is the frequency of sampling, δ is the difference of consecutive sampling value. To directly calculate the $\Omega_i(f)$ is very difficult if not impossible, since parameter δ is time-varying random variable. Therefore, we resort to calculate the upper bound of $\Omega_i(f)$ as it is easy to have the mean and variance of parameter δ, which is formulated as following:

$$\Omega_i(f) \le 1 - \prod_{k \in (1,f]} \frac{\left(\left(x_i - k\omega - a_i(t_1)\right)/\xi k\right)^2}{1 + \left(\left(x_i - k\omega - a_i(t_1)\right)/\xi k\right)^2} \tag{5}$$

where ω is parameter δ's mean value, ξ is the variance value. Using the upper bound of $\Omega_i(f)$ to approximate the missing detection probability, we can easily obtain the optimal sampling frequency of SLA monitoring service. Specifically, given a misdetection threshold e^*, the monitoring service can use the following rules to adjust the current sampling frequency:

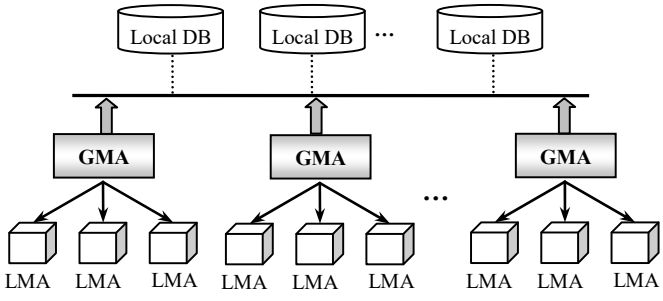
1 if current $\Omega_i(f)$ is less than e^* in a series of sampling rounds, then the sampling frequency should be reduced

2 if current $\Omega_i(f)$ is larger than e^* in a series of sampling rounds, then the sampling frequency should be increased

3 in other cases, the sampling frequency can be remained.

Based on the above dynamical sampling frequency mechanism, our SLA monitoring service is implemented as a push-based *publish/subscribe* model, which has be widely applied in many real-world monitoring framework, which is demonstrated in Figure 4.

Figure 4 Framework of SLA monitoring service



In the above SLA monitoring framework, a set of local monitoring agents (LMAs) are deployed to monitoring virtual resources belong to different resource provider. Meanwhile, these LMAs are grouped based on specific SLA terms into a set of group monitoring agents (GMAs) which is responsible for gathering and filtering monitored data from low-level LMAs. In this way, the SLA monitoring service is formed as a hierarchical architecture, where LMAs play the role of publishers and the GMAs play the role of subscribers. Such a hierarchical *publish/subscribe* model can significantly reduce the communication and storage overheads when performing SLA monitoring operation.

4 Experiment and performance evaluation

4.1 Experimental settings

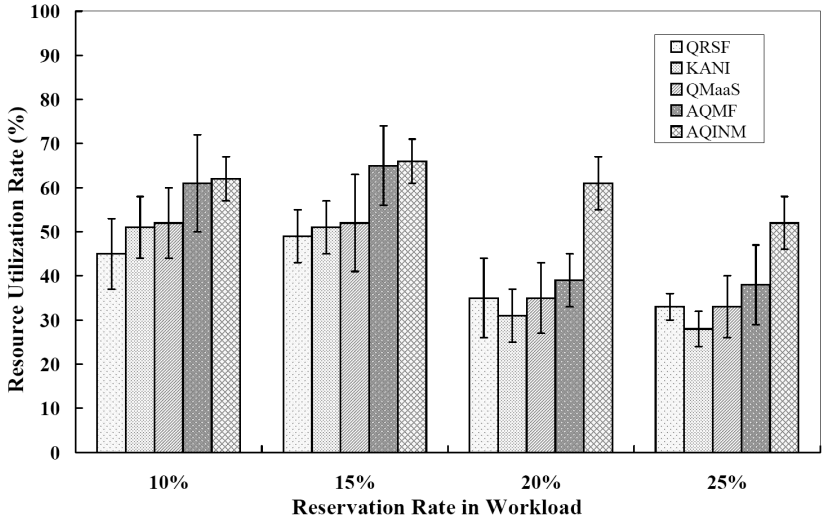
To analyse the performance of our AQINM, we deploy its prototype implementation in a campus cloud platform that has seven virtualised computing clusters each being located in different colleges. These clusters use Xen or VMware as their VM hypervisors and form a common virtual resource pool. To compare the performance with previous studies, four widely-applied QoS-enhance middleware are choosed, including QRSF (Singh and Chana, 2014), KANI (Asyabi et al., 2016), QMaaS (Cicotti et al., 2015), and AQMF (Xu et al., 2018), and deploy them in the same experimental cloud platform. To simulate the resource requirements from distributed applications, we design a set of software agents that rely on the Lublin-Feitelson model (Lublin and Feitelson, 2003) to produce continuous resource requests. In each resource request, we append several SLA items, such as resource demand, reservation requirement, responsive time, deadline, resource budget, etc. We will present the experimental results in terms of different performance

metrics by using different QoS-enhancing solutions, and compare their efficiency and effectiveness.

4.2 Comparisons on performance metrics

In the first set of experiments, we examine three key performance metrics when using different QoS solutions, including resource utilisation ratio (RUR), average response time (ART) and request reject ratio (RRR). As different requests may have different resource reservation demands which may have significantly effects on these performance metrics, we divide these experiments into four groups, each with different reservation rate ranging from 10% to 25%. Typically, using resource reservation mechanism may increase the QoS experienced by users. However, extensively using it may lead to extra performance costs and high request rejection in long-term., which has been indicated by many existing studies. Figure 5 and Figure 6 show the comparisons on RRR and RUR when using different QoS solutions.

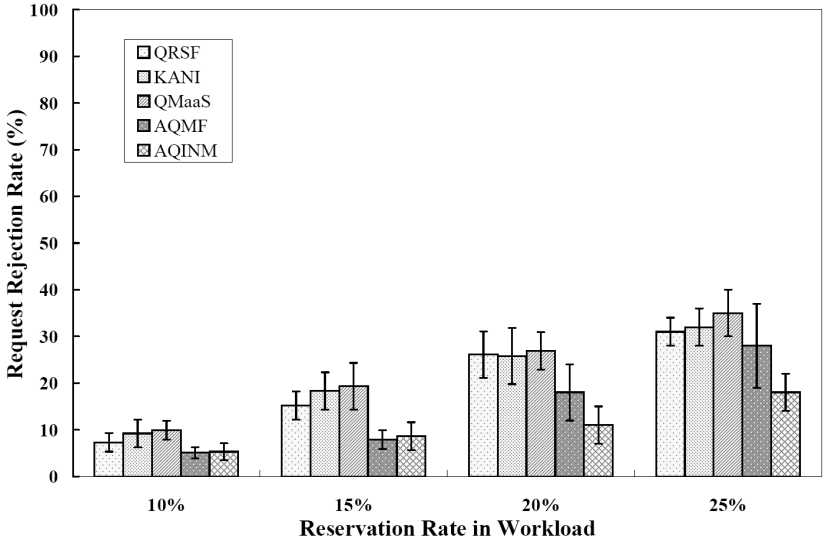
Figure 5 Comparisons on resource utilisation



As show in Figure 5, we first notice that lower reservation rate (about 10%) can lead to higher RUR, and a slightly increasing of it (about 15%) also can improve the RUR performance. Such a result shows that advance reservation mechanism is still an effective QoS-enhancing policy in federated cloud environments. However, when the reservation rate is higher than 20%, the RUR metric will be reduced in all experimental cases. Among the five tested solutions, we can see that AQMF is the most sensitive one to the reservation rate, since it is decreasing of RUR metric is the highest in all solutions when the reservation rate is increased over 20%. As to QRSF and QMaaS, their RUR metrics is lower than AQMF and AQINM by about 17%~22% when the reservation rate is lower than 20%. With the increasing of reservation rate over 20%, we can see that only our AQINM can maintain a high RUR metric. This experimental result indicates that the proposed AQINM solution exhibits better adaptive to the variety of reservation rate. We also see that RUR metric obtained by KANI solution is the lowest in high reservation rate

cases. This is because that the KANI solution relies on strict QoS matchmaking algorithm to allocate virtual resources, which is likely to lead to load unbalancing if more user requests submit their demands through reservation mechanism.

Figure 6 Comparisons on request rejection rate

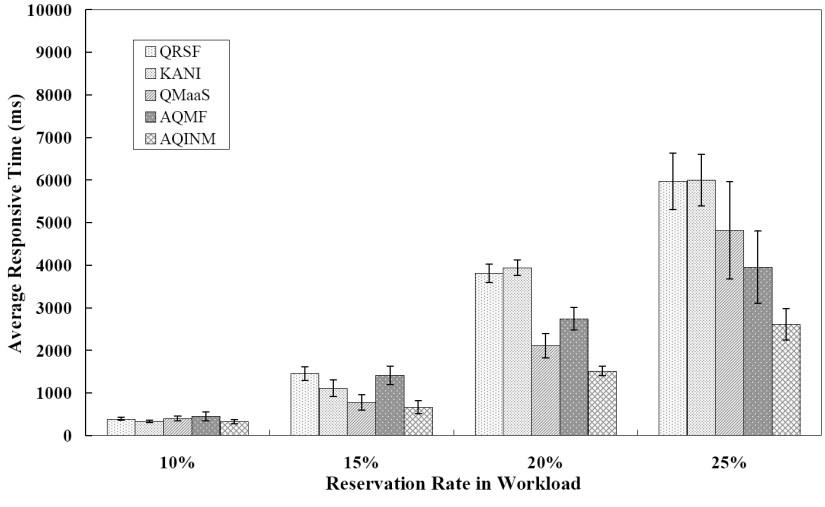


As to the RRR metric, we can see in Figure 6 that this metric is always increasing with the reservation case in all experimental cases. This is because that higher reservation rate always means that more available resources have been occupied in short-term future. In a result, the newly arrival requests are more likely to be rejected by the underlying resource management service. Among the five tested solutions, we find that the obtained RRR metrics of using AQMF and AQINM is generally lower than other solutions by about 38%~55%. Through analysing the logs of experiments, we notice that the QoS negotiation service is very effective to reduce the RRR metric. Specifically, a more flexible QoS negotiation strategy tends to lead to lower RRR metric. In QRSF and KANI, the negotiation strategy is based on performance-level matchmaking and a dynamical pricing mechanism. Such a strategy has been proven to be effective in economic-based utility computing systems. However, its effectiveness will be mitigated when using reservation mechanism, since the resource performance and price of resource may be changed during the period between resource reservation and allocation. As to the AQMF, its negotiation strategy is very similar to our AQINM, which applies time-vary utility functions to define the dynamical SLA-levels. Their difference is that our AQINM allows different resource providers to applying different negotiation policy, which in turn improves the flexibility of negotiation service in federated cloud environment. Figure 6 indicates that an effective SLA negotiation service should take into account the different resource management policies in federated clouds.

In Figure 7, we demonstrate the comparisons on ART metric when using different QoS solutions. Unlike the RUR and RRR metrics which are typically considered as the QoS metrics of resource providers, the ART metric is a user-oriented QoS metric that have significant effects on the QoS experience of cloud users. As shown in Figure 7, we

can see that the differences of the ART metric are ignorable when the reservation rate is about 10%. This is because that we configure the same task scheduler for all tested QoS solutions and the total available virtual resources are sufficient comparing with the arrival resource demands in our experimental cloud. So, it can be expected that the ART metrics obtained by different QoS solutions will be very small, even some part of resources have been reserved in advance. However, as more and more resources have been reserved, some of the user requests have to wait for longer time to obtained resources, which in turn increases the ART metric. Among the five tested solutions, we can see that the increasing of ART metric of using QRSF and KANI are the highest. This is because both of them tend to allocate high-performance resources to newly arrival requests, while leaving the low-performance resources in idle state. As to QMaaS and AQMF, we notice that both of them have been incorporated certain load-balancing mechanisms when performing resource negotiation and allocation. However, their negotiation procedure is more complicated which is especially true when multiple resources should be co-allocated together. In this case, the time-based negotiation decision function in AQINM plays a key role to reduce the negotiation time and therefore obtains lower ART metric in all experimental results. So, we conclude that a low-cost SLA negotiation service can significantly improve the QoS experience of cloud users.

Figure 7 Comparisons on average responsive time

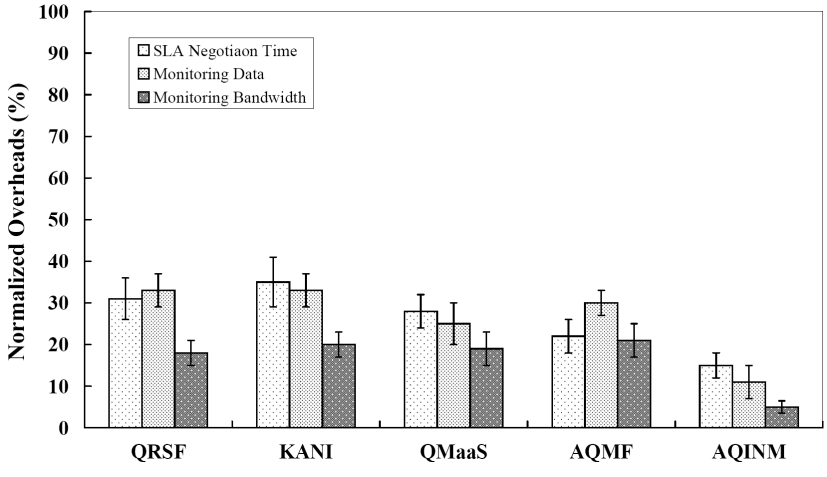


4.3 Comparisons on SLA monitoring costs

In this section, we take efforts on analysing the overheads of using QoS-enhancing solutions. As the different overheads may come from different layers in the cloud software stack, we mainly focus on the three kinds of overheads: SLA negotiation time, monitoring data, and monitoring bandwidth. The negotiation time comes from the application layer; the monitoring data comes from the storage layer, while the monitoring bandwidth is from the networking infrastructure. As these overheads are of different measurements, for the convenience of analysing, we normalise them as a integrate value ranging from (0%, 100%) in Figure 8.

As mentioned in Section 3.3, the SLA negotiation time is mainly decided by the negotiation strategy and resource demands. According to the results in Figure 8, we can see that KANI has the highest negotiation time overheads among the five tested solutions, which is nearly twice as using AQINM. By observing the experimental records, we find that most of negotiation time is spent on pricing negotiation and multi-resource demand typical needs more negotiation time when using KANI than other solutions. As to QRSF and QMaaS, negotiation overheads are slightly lower than KANI due to the more elastic QoS matchmaking strategy. To reduce the overheads of negotiation, our AQINM apply a time-vary function which is very effective to drive both resource providers and users to make SLA contacts more quickly. In fact, the negotiation service in AQINM can be controlled parameter β as analysed in Section 3.3. Therefore, individual cloud providers can set different value of parameter β so as to meet their own resource management objectives.

Figure 8 Comparisons on QoS-related overheads



The SLA monitoring data is often used to decide that whether SLA violation occurs. In some system, they may be used to make more desirable resource allocation and task scheduling decisions. Unlike other four solutions, our AQINM introduces a dynamical sampling frequency approach which can significantly reduce the useless monitoring data and therefore reduce the storage capacities. As shown in Figure 8, this kind of overheads when using AQINM is lower than other solutions by about 52%~64%. More importantly, if monitoring data should be transferred to remote databases (or a centralised DB server), more bandwidth overheads will be introduced which has been demonstrated in Figure 8. According to our experimental results, we find that by changing the sampling frequency, the SLA monitoring service can reduce about 22%~35% remote data transferring, which in turn reduces about 8% bandwidth overheads. It is noteworthy that the bandwidth overheads include both monitoring data transferring and the SLA communication costs. So, by combining the two factors, our AQINM's bandwidth overheads are lower than other four solutions by about 58%~71%. So, we conclude that the proposed AQINM is effective to reduce the QoS-related overheads.

5 Conclusions and discussion of future work

We design a lightweight QoS-enhancing middleware called AQINM, which provides three QoS-enhancing services including policy management service, SLA negotiation service, and SLA monitoring service. Unlike the conventional QoS-enable middleware, these services in the AQINM framework introduce several novel mechanisms to offer more cost-effective and efficient solutions to enforce the QoS management in federated clouds. The implementation of our AQINM framework are tested in a campus federated cloud platform by using different applications as experimental benchmarks, and its performance are compared with other similar solutions. Currently, the prototype of AQINM only supports Xen and VMware hypervisor. So, in the future, we are planning to investigate the proposed framework in more resource virtualisation platforms. In addition, we are going to re-design the AQINM so as to make it work with some popular workflow engines so as to provide better QoS service for large-scale workflow applications.

Acknowledgements

This work is supported by Scientific Research Fund of Hunan Provincial Education Department (19B132).

References

Anastasi, G.F., Carlini, E. et al. (2017) 'QoS-aware genetic cloud brokering', *Future Generation Computer Systems*, Vol. 75, No. 1, pp.1–13.

Andronikou, V., Mamouras, K. et al. (2012) 'Dynamic QoS-aware data replication in grid environments based on data 'importance'', *Future Generation Computer Systems*, Vol. 28, No. 3, pp.544–553.

Aron, R. and Chana, I. (2012) 'Formal QoS policy based grid resource provisioning framework', *Journal of Grid Computing*, Vol. 10, No. 2, pp.249–264.

Asyabi, E., Azhdari, A. et al. (2016) 'Kani: a QoS-aware hypervisor-level scheduler for cloud computing environments', *Cluster Computing*, Vol. 19, No. 2, pp.567–583.

Cao, H., Jin, H. et al. (2010) 'ServiceFlow: QoS-based hybrid service-oriented grid workflow system', *Journal of Supercomputing*, Vol. 53, No. 3, pp.371–393.

Cicotti, G., Coppolino, L. et al. (2015) 'How to monitor QoS in cloud infrastructures: the QoSMONaaS approach', *International Journal of Computational Science and Engineering*, Vol. 11, No. 1, pp.29–45.

Darby, P.J. and Tzeng, N.F. (2010) 'Decentralized QoS-aware checkpointing arrangement in mobile grid computing', *IEEE Transactions on Mobile Computing*, Vol. 9, No. 8, pp.1173–1186.

Di-Stefano, A., Morana, G. et al. (2011) 'QoS-aware services composition in P2P grid environments', *International Journal of Grid and Utility Computing*, Vol. 2, No. 2, pp.139–147.

Duan, Q. and Vasilakos, A.V. (2016) 'Federated selection of network and cloud services for high-performance software-defined cloud computing', *International Journal of High Performance Computing and Networking*, Vol. 9, No. 4, pp.316–327.

Giunta, R., Messina, F. et al. (2015) 'Providing QoS strategies and cloud-integration to web servers by means of aspects', *Concurrency Computation*, Vol. 27, No. 6, pp.1498–1512.

Halboob, W., Abbas, H. et al. (2015) 'A framework to address inconstant user requirements in cloud SLAs management', *Cluster Computing*, Vol. 18, No. 1, pp.123–133.

Hayyolalam, V. and Pourhaji-Kazem, A.A. (2018) 'A systematic literature review on QoS-aware service composition and selection in cloud environment', *Journal of Network and Computer Applications*, Vol. 110, No. 1, pp.52–74.

Homsi, S., Liu, S. et al. (2017) 'Workload consolidation for cloud data centers with guaranteed QoS using request reneging', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 28, No. 7, pp.2103–2116.

Huedo, E., Montero, R.S. et al. (2017) 'Interoperable federated cloud networking', *IEEE Internet Computing*, Vol. 21, No. 5, pp.54–59.

Kianfar, K., Moslehi, G. et al. (2015) 'A novel metaheuristic algorithm and utility function for QoS based scheduling in user-centric grid systems', *Journal of Supercomputing*, Vol. 71, No. 3, pp.1143–1162.

Kourtesis, D., Alvarez-Rodriguez, J.M. et al. (2014) 'Semantic-based QoS management in cloud systems: Current status and future challenges', *Future Generation Computer Systems*, Vol. 32, No. 1, pp.307–323.

Li, W., Wu, J. et al. (2014) 'Trust-driven and QoS demand clustering analysis based cloud workflow scheduling strategies', *Cluster Computing*, Vol. 17, No. 3, pp.1013–1030.

Lublin, U. and Feitelson, D.G. (2003) 'The workload on parallel supercomputers: modeling the characteristics of rigid jobs', *Journal of Parallel and Distributed Computing*, Vol. 63, No. 11, pp.1105–1122.

Panda, S.K. and Jana, P.K. (2017) 'SLA-based task scheduling algorithms for heterogeneous multi-cloud environment', *Journal of Supercomputing*, Vol. 73, No. 6, pp.2730–2762.

Quarati, A., Clematis, A. et al. (2016) 'Delivering cloud services with QoS requirements: business opportunities, architectural solutions and energy-saving aspects', *Future Generation Computer Systems*, Vol. 55, No. 2, pp.403–427.

Rane, D. and Sarma, M. (2015) 'CSLAT: an SLA template for cloud service management', *International Journal of Communication Networks and Distributed Systems*, Vol. 14, No. 1, pp.19–39.

Ranjbari, M. and Akbari-Torkestani, J. (2018) 'A learning automata-based algorithm for energy and SLA efficient consolidation of virtual machines in cloud data centers', *Journal of Parallel and Distributed Computing*, Vol. 113, No. 1, pp.55–62.

Singh, S. and Chana, I. (2014) 'QRSF: QoS-aware resource scheduling framework in cloud computing', *Journal of Supercomputing*, Vol. 71, No. 1, pp.241–292.

Sun, G., Liao, D. et al. (2018) 'Towards provisioning hybrid virtual networks in federated cloud data centers', *Future Generation Computer Systems*, Vol. 87, No. 3, pp.457–469.

Trapero, R., Modic, J. et al. (2017) 'A novel approach to manage cloud security SLA incidents', *Future Generation Computer Systems*, Vol. 72, No. 2, pp.193–205.

Vilaplana, J., Mateo, J. et al. (2015) 'An SLA and power-saving scheduling consolidation strategy for shared and heterogeneous clouds', *Journal of Supercomputing*, Vol. 71, No. 5, pp.1817–1832.

Wen, Z., Cala, J. et al. (2017) 'Cost effective, reliable and secure workflow deployment over federated clouds', *IEEE Transactions on Services Computing*, Vol. 10, No. 6, pp.929–941.

Xiao, P. and Han, N. (2013) 'Improving user QoS by relaxing resource reservation policy in high-performance grid environments', *International Journal of Grid and Utility Computing*, Vol. 4, No. 4, pp.255–264.

Xu, X., Tang, M. et al. (2018) 'QoS-guaranteed resource provisioning for cloud-based MapReduce in dynamical environments', *Future Generation Computer Systems*, Vol. 78, No. 1, pp.18–30.

Xue, S., Zhang, Y. et al. (2017) 'QET: a QoS-based energy-aware task scheduling method in cloud environment', *Cluster Computing*, Vol. 20, No. 4, pp.3199–3212.

Ye, Z., Mistry, S. et al. (2016) 'Long-term QoS-aware cloud service composition using multivariate time series analysis', *IEEE Transactions on Services Computing*, Vol. 9, No. 3, pp.382–393.

Zheng, Z., Wu, X. et al. (2013) 'QoS ranking prediction for cloud services', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 24, No. 6, pp.1213–1222.