International Journal of Grid and Distributed Computing
Vol. 10, No. 1 (2017), pp.189-202
http://dx.doi.org/10.14257/ijgdc.2017.10.1.17

An Integrated Resource Co-allocation Middleware for Virtualized Cloud Platforms

Zhichao Liu

Network Information Center, Hunan Institute of Engineering
[*]ygfeng1974@126.com

Abstract

In a cloud environment, large-scale applications generally tend to require plenty of resources which might be across multiple virtual organizations. Therefore, resource co-allocation service plays an important role when deploying such kind of cloud applications. In this paper, we present an integrated resource co-allocation middle, namely Economic-based Virtual Resource Co-allocation (EVRC), which provides a set of flexible services that allows cloud applications co-allocating plenty of virtual resources across different resource providers. In the EVRC framework, resource co-allocation service is implemented by a novel auction mechanism, and cloud user's quality-of-service (QoS) requirements are guaranteed by a set of services including resource reservation, reputation manager and etc. Furthermore, the EVRC is incorporated with an online performance monitoring and profiling mechanism, which can be used to evaluate the efficiency of underlying resources and adjust up-level resource management policy. Extensive experiments are conducted to investigate the effectiveness of the EVRC middleware, and the results indicate that it can significantly improve the efficiency of virtual resource co-allocation as well as cloud user's QoS satisfactory.

***Keywords**: Cloud Computing; Resource Auction; Performance Monitor; Virtualization*

1. Introduction

Recently, cloud platform has become a promising technology which enables users to deploy various kinds of applications in an environment with increased scalability, availability, and fault tolerance [1, 2]. The introduction of cloud computing paradigm has brought unprecedented computation and storage capabilities to a wide area of the scientific and business communities [3, 4]. Applications designed to execute on clouds frequently require the simultaneous co-allocation of multiple resources that span administrative domains in order to meet performance requirements [5, 6, 7]. For example, several computers and network elements may be required in order to achieve real-time reconstruction of experimental data, while a large numerical simulation may require simultaneous access to multiple supercomputers. Furthermore, access to resources is in general unreliable, due to either competing demands for the resource or outright failure [8, 9]. These two features complicate the co-allocation process in cloud environments.

Therefore, resource co-allocation becomes a critical issue that needs to be addressed efficiently. Over the past years, plenty of distributed platforms have allowed the co-allocation of heterogeneous resources [10, 11, 12, 13, 14]. However, these systems assume that they are in total control of all resources, which may not be the case in cloud environments. For example, Legion is a distributed object-oriented system which supports the resource co-allocation by advance reservation technique. In a cloud platform, the co-allocation problem can be defined as the provision of allocation, configuration, and monitoring for the resource ensemble required by a single application [15]. As noted in [16], to meet the various QoS requirements of cloud applications, no single co-allocation strategy can be effective for all purpose in cloud platforms.

In this paper, we design and implement a lightweight co-allocation middleware for cloud platforms, namely *Economic-based Virtual Resource Co-allocation* (EVRC), which provides a set of flexible services that allow cloud applications co-allocating plenty of virtual resources across different resource providers. Previously, economic-based approaches to resource co-allocation have been proven to be effective to deal with multiple resource allocation in traditional distributed systems such as cluster and grid [17, 18, 19]. In economic-based approaches, the use of currency provides incentives for resource providers to contribute resources, clients act responsibly and cannot afford to waste resources due to their limited budget. Therefore, in our EVRC framework the resource co-allocation service is implemented by a novel auction mechanism, which can significantly improve the efficiency of resource usage as well as application execution. On the other side, our EVRC can guarantee user's QoS requirements by a set of services including resource reservation, reputation manager and etc. In addition, the proposed EVRC middleware is incorporated with an online performance monitoring and profiling mechanism, which can be used to evaluate the efficiency of underlying resources and adjust up-level resource management policy.

The rest of this paper is organized as follows. Section 2 presents the related work. In section 3, we describe the framework of the proposed middleware. Section 4 presents the implementation of the key service components. In section 5, experiments are conducted to investigate the effectiveness of the proposed approach. Finally, Section 6 concludes the paper with a brief discussion of the future work.

2. Related Work

In early studies, resource co-allocation techniques in distributed systems can be distinguished according to two criteria: state-based and model based. For the state-based co-allocation techniques, the system needs to maintain a list of current snapshots for all resources and the resource allocation decisions are made based on these runtime information. Typically, state-based techniques are easy to implement and can lead to good results, which makes them widely being adopted. For example, Matchmaking and Gangmatching are two well-known state-based co-allocation techniques. In [20], the authors proposed a linear-programming based approach for multiple resources matching in grid environments. Four kinds of global objective functions are considered, including maximizing throughput and weighted throughput, minimizing resource cost, maximizing job preferences for resources, and load balancing among computing sites. In [21], the authors proposed a rough-set based service matchmaking algorithm (RSSM) for service discovery with an aim to tolerate uncertain properties. By dynamically reducing irrelevant and dependent properties, RSSM algorithm can tolerate uncertain properties in service discovery. In addition, RSSM uses a lower approximation and an upper approximation to dynamically determine the number of discovered services which may maximally satisfy the requirements of applications.

Generally speaking, the shortcoming of state-based techniques is their cost is normally higher, since maintaining and updating such a list of global resource snapshots requires plenty of extra works. With respect to model-based co-allocation techniques, they normally provide certain approaches to predicting the future states of those concerned resources, which can be used to guide resource allocation actions [22]. However, the shortcoming of model-based techniques is that their effectiveness depends on the accuracy of predicting models which may inaccurate in some scenarios [23]. Therefore, both of them can only be used in small-scale or mediate-scale distributed platforms.

With the development of large-scale distributed platforms, market-based co-allocation techniques have been widely studied because they can provide a way to achieving an efficient match of supply and demand [24]. For example, in [25] the authors investigated the application of market-oriented mechanisms based on the *General Equilibrium Theory* *of Microeconomics* to coordinate the sharing of resources in federated clouds. Their analyses indicated that resource market can be effectively applied in large-scale clouds. In [26], the authors characterized the nature of non-cooperative competition in an IaaS cloud market, with a goal of capturing how each IaaS cloud provider will select its optimal prices to compete with the others. Their analyses lead to sufficient conditions for the existence of a Nash equilibrium which can be obtained in special cases. Based on their analyses, an iterative algorithm for IaaS cloud providers is proposed to compute equilibrium prices. In [27] the authors proposed the definition of a semantic model that helps customers and providers to characterize their demands/offers. Also, they provided a set of semantic tools performing the matchmaking in such a way to maximize both providers profit and customers utility.

Recently, how to improve the QoS of cloud platforms has attracted more and more attentions due to application of several commercial cloud platforms. Plenty of researchers have proposed various techniques to achieve this goal from different aspects. For example, Lin et al. presented a QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network [28]. The proposed method is deployed by two phases: service registering phase and service discovery phase. In [29] the author presented an analytical model, based on stochastic reward nets (SRNs), that is both scalable to model systems composed of thousands of resources and flexible to represent different policies and cloud-specific strategies. Several performance metrics are defined and evaluated to analyze the behaviour of a cloud datacenter. In [30] the authors investigated how to integrate QoS awareness with virtualization in cloud computing systems, such as the QoS-aware VM placement problem. They first formulated the VM placement problem as an Integer Linear Programming (ILP) model by integrating the three factors as the profit of cloud provider. Then, they further proposed a polynomial-time heuristic algorithm to efficiently solve the ILP problem.

3. Framework and Architecture

3.1 Framework of EVRC

The key concept of designing EVRC framework is reducing the costs of resource co-allocation across multiple virtual administrative organizations. The overview of EVRC framework is shown in Figure 1. Typically, distributed platforms often use resource metal-scheduler for small-scale resource allocation, such as high-performance cluster, where all the required resources are within a single organization. When resource co-allocation across multiple virtual organizations (VO) is involved, each VO should be carefully defined and modeled. In our EVRS framework, a VO model is incorporated to group virtual resource providers and users. To control resource accessing and authenticate VO's participants, each VO is incorporated with a *Membership Manager Service* (MMS) model.

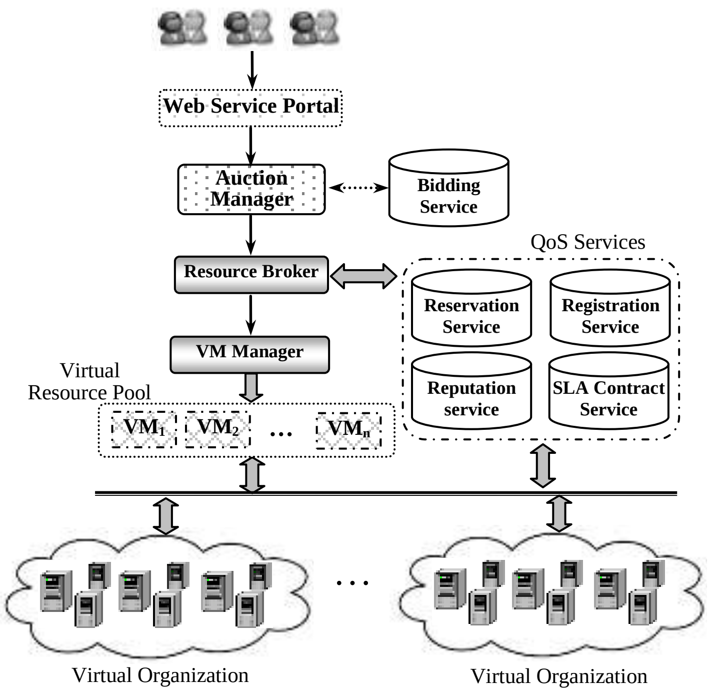


Figure 1. Overview of EVRC Framework

In the EVRC framework, VO models also are responsible for outlining the scope of underlying resource providers, which might be cloud vendors or large-scale datacenters. The size of a VO is not fixed, which means that any resource providers can join or leave a VO as they wish. Before a resource provider joins into one of VOs, it is required to expose a set of obligation services to contribute to the VO for the execution of underlying meta-scheduler. These obligation services can be auction manager, auction context, or reputation service. When the resource provider becomes a member, it also should expose participation services which are needed to bid for, accept and schedule jobs on their provisioned resources.

From the perspective of end-users, the resource broker component is responsible for providing a set of transparent interfaces to EVRC for supporting job submission, monitoring, and runtime performance evaluation. In our current implementation, the broker component is designed as a plug-in service, which can be configured as a client-side application or deployed as platform-independent middleware. Therefore, all correspondence between the resource broker and the VOs can be easily logged and monitored. For example, it is able to discover the resource provider’s meta-scheduler services by standard monitoring and discovery service (MDS) and is free to choose those services based on their reputations. When end-users submit their jobs, the resource broker component also interprets job’s requests to an Auction Manager which instantiates an Auction Context for the duration of the auction. At the end of the auction, the winner and broker receive contracts describing the agreed upon commitments and the job is dispatched to the local-scheduler services to complete the procedure of virtual resource co-allocation.

Since communication cost will be significantly increased when plenty of resources are co-allocated at the same time, selecting appropriate resource trading mechanisms becomes a critical issue in EVRC framework. For example, For instance, within an organization internal trust exists, in this case EVRC should achieve maximum co-allocation performance by choosing a conventional sealed bid auction plug-in which does not utilize expensive trust or privacy preserving mechanisms. For those opening platforms in which there is no such trust, the EVRC should apply a more securable trading mechanism such as auction for ensuring the resource co-allocation is conducted fairly and the bid details are kept private.

Due to the high overhead of secure-enhanced mechanisms, we designed a set of adaptive models which allow administrator to change the underlying resource trading mechanism in a plug-in manner. In this way, the EVRC can be used in most of the well-known platforms (i.e. gird and cloud). For example, To permit the deployment of different auction mechanisms (both secure and insecure), EVRS applies the General Auction Framework (GAF) to provide a communication independent shell for a wide range of auction types. It is important to be able to seamlessly select different protocols for different tasks due to the wide ranging complexity of privacy preserving, trustworthy and verifiable auction protocols.

3.2 Services in EVRC

In this section, we describe the core components in our EVRC framework. To achieve platform-independent, the whole EVRC is developed through WSRF-based web services. Each service has a well defined task and is accessible through a standard interface. These services collectively perform virtual resource co-allocation through plug-in auction protocols.

- **Auction Manager**: It manages the auction process and advertises the auction to suitable bidders. For an auction, there is an auction-context which stores data of the ongoing auction including client information, auction state, and contracts. The resource broker starts an auction by passing the job description to the auction manager who then creates the resources required for the particular protocol.
- **Bidding Service**: It is used to deal with all auction events from the Auction Manager and computes bids for jobs it wishes to host using pricing algorithms, local policy, and future commitments. The biding policy can be changed according the underlying auction mechanism.
- **SLA Contract Service**: It stores all SLA contracts that have been issued within the VO. Typically, a SLA contract is obtained through several QoS negotiations. For a end-user, there are often multiple SLA contracts that should be obtained before executing its job, and all these contracts are companied with a validated duration. So, this service is also used in the contract hardening process before redemption, ensuring that the contract can be honoured.
- **Registration Service**: This service mainly used by resource providers who want to sell their resources in a VO. For instance, resource providers can register their bidding profile when joining the VO. This profile is used to categorize the type of job a resource is interested in hosting. This has the effect of reducing auction size and therefore increasing the efficiency of the co-allocation process.
- **Reputation Service**: It stores all the reputation-related information for all participants. The reputation information can be as simple as user’s feedback after running their jobs, or it can be obtained through a third-party’s reputation evaluation system.
- **Reservation Service**: It manages all the advanced resource reservation commitments normally in the form of contracts between users and certain resource provider. Reservation service plays a key role to improve the QoS satisfactory of end-users since it can provide ensures of resource availability during the given time period.
- **VM Manager**: consists of four subcomponents, and they are designed for VM resource pool management and VM performance analysis. For example, the provision of VM is controlled by VM pool manager and VM configuration, and the

runtime performance and capability of individual VM are monitored and logged through Performance Monitor and Performance Profiling subcomponents.

4. Implementation of EVRC

4.1 Virtual Resource Co-allocation

None-trivial applications designed to execute on cloud platforms frequently require the simultaneous allocation of multiple resources that span administrative domains in order to meet performance requirements. For example, several computational and networking resources may be required in order to achieve real-time reconstruction of experimental data, while a large numerical simulation may require simultaneous access to multiple supercomputers. Furthermore, access to resources is in general unreliable, due to either competing demands for the resource or outright failure. These two features complicate the co-allocation process in cloud environment. Therefore, Co-allocation is the process of simultaneously allocating resources in predetermined capacities over a set of resource providers. In cloud platforms, the co-allocation problem can be defined as the provision of allocation, configuration, and monitoring for the resource ensemble required by a single application.

In the EVRC, we adopt *Continuous Double Auctions* (CDA) to implement the core service of virtual resource co-allocation. The reason that we chose CDA is its supporting for many-to-many protocol. Furthermore, it has to be a continuous auction where transactions are carried out immediately whenever bids or offers change. In our EVRC framework, this is preferable to using protocols where the transactions are only carried out at periodic intervals. In such a protocol an arriving user’s request would not have to wait for the next auction, which will significantly minimize the response time. Meanwhile, we implement the Proportional Sharing Policy (PSP) as the low-level resource scheduling mechanism, which has been widely used in current virtualization platforms for scheduling VM instances. In proportional-share scheduling, a resource is split up among several task which are allocated resource shares that are proportional to their price bids. The PSP mechanism can improve the efficiency of CDA for certain situations, such as high network latency and high resource heterogeneity. Both CDA and PSP are considered as greedy policies, in the sense that a use’s request is assigned the best possible resource that is available at a given time. In Figure 2, we demonstrate the sequence of resource co-allocation when using CDA.

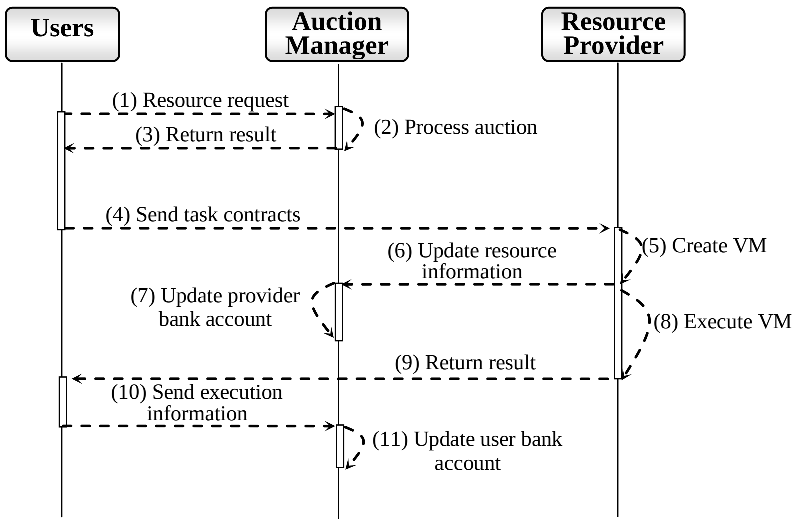


Figure 2. Sequence of Resource Co-allocation when Using CDA

Before any auctions issued by users, resource providers need to register their resource offers, including available resource amount, speed factor, baseline price and etc. All auctions are completed by the Auction Manager as shown in step (2). When an auction is finished, the selected resource provider is responsible to create VM instance according to the auction contracts. In the EVRC, the Auction Manager also maintains a list of available resources for all providers for improving efficiency. Therefore, it will update the resource lists when co-allocation is finished. For users and resource providers, their bank accounts are maintained by the Auction Manager, which will be updated at step (7) and step (11). It should be noted that multiple resource providers might be involved when a single provider can not meet the user's resource request. We ignore this in Figure 2 just for simple representation.

4.2 Runtime Performance Monitoring

In the EVRC framework, we incorporate a runtime performance monitoring mechanism for both resource providers and user jobs. Firstly, resource providers need to know the working status of the resources belonging to them. As mentioned in section 3.1, once a resource provider becomes a member of a VO, it is required to expose participation services which are needed to bid for, accept and schedule jobs on their provisioned resources. Therefore, it is important for those providers to adjust or change their resource configurations when the runtime performance of their resources becomes undesirable. To achieve this goal, we implement a queue model based performance evaluator, which can be used to estimate the working efficiency and workload intensiveness for any physical and virtual resources. For example, any VM instances can be modeled as a 6-tuple $<A(t), S(t), W(t) C(t), U(t), F(t)>$, where $A(t)$ is probability density function of jobs arriving interval, $S(t)$ is the density function of mean serving time, $W(t)$ is the number of waiting jobs, $C(t)$ is the maximal concurrent tasks that the VM can accept, $U(t)$ is the utility function which can be defined by resource providers, $F(t)$ is the mean fault interval time.

For a single VM instance, one queue model is enough to describe its dynamic performance. However, many cloud applications might co-allocate multiple VMs to finish their jobs. Therefore, it is needed that we can precisely predict or evaluate their performance concurrently. Normally, it is difficult to model the runtime performance of multiple resources in distributed platform because of the interactions between different resources. Fortunately, queue theory provides us a convenient approach to describing such a complex model which is shown in Figure 3. Detailed information on composed queue model can be found in [31].

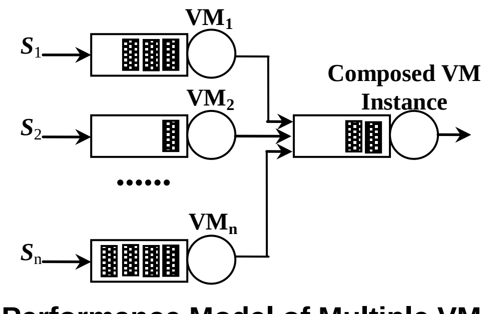


Figure 3. Performance Model of Multiple VM Instances

In our ERVC framework, most of the key components will interact with the performance monitoring and profiling mechanism. For example, the VM Manager is responsible to trigger the performance events related with VM instance as long as they are in activated states, while the QoS services will monitor the available virtual resources in all VOs. It is clear that all the queue models of VMs need to be updated during the runtime. To improving the accuracy of the performance monitoring, we designed a feedback model to dynamically update all the queue models, which is shown in Figure 4.

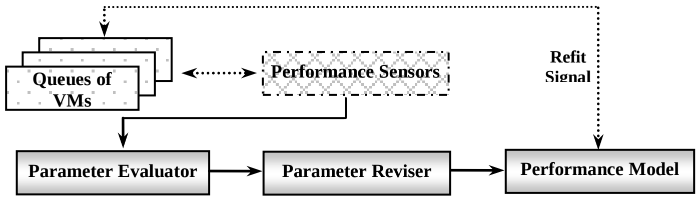


Figure 4. Feedback Model for Performance Monitoring Mechanism

As sown in Figure 4, the *performance sensors* are attached with all the available resources, and they can be hardware devices or software components. In one hand, the sensors can send recent information to VM queues directly to update them. On the other hand, they use two components (Parameter Evaluator and Parameter Reviser) to obtain a new queue-based performance model, which can be used to adjust the current VM queue models through a closed-loop controlling technique. In the current implementation of ERVC, the frequency of using this close-loop is 10 minutes while the frequency of sensor sampling is one minute. In this way, we can achieve both efficiency and accuracy for the performance monitoring mechanism.

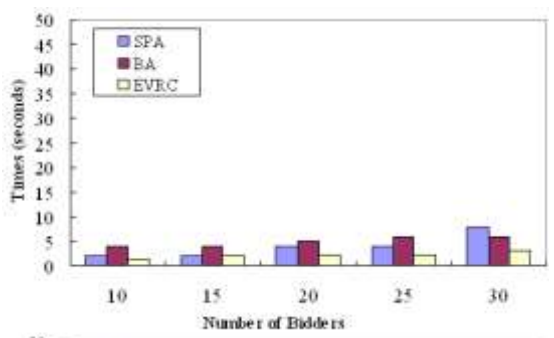
5. Experiments and Evaluation

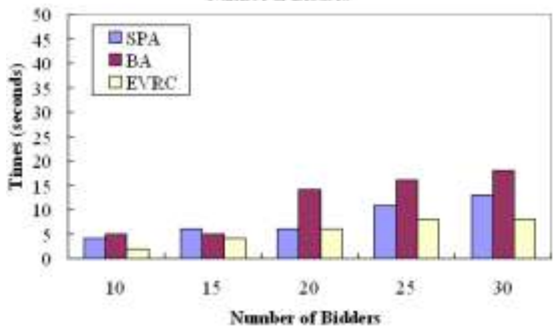
5.1 Experimental Configurations

To evaluate the performance and effectiveness of the EVRC, we deploy it on our campus cloud infrastructure, which consists of 12 high-performance computing clusters and 27 storage nodes. The total number of computing nodes is 4,500 and the storage space is over 65 TB. These computing and storage nodes belong to different schools where different resource management policies are adopted. At the virtual layer, XCP is used for realizing resource virtualization, and our EVRC implementation is deployed as a middleware between XCP and end-user's applications. For the convenience of conducting experiments, we developed a set of software agents that can automatically send resource requests by some pre-defined rules. In this way, we can obtain various kinds of workloads to test the EVRC.

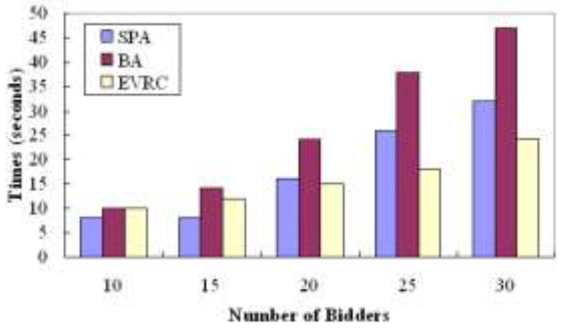
5.2 Comparison on Resource Auction Performance

In the first experiment, we mainly concentrate on the resource auction service since it is the key mechanism for resource co-allocation. To compare the performance, we also implement two well-known resource auction mechanisms: Second Price Auction (SPA) and Bargaining Auction (BA). The key performance metric we test is the auction time costs. For any resource auction mechanisms, the numbers of bidders and resource providers have significant effects on the auction performance. So, we gradually increase the number of bidders (from 10 to 30) and the number of providers (from 5 to 20) during our experiment. The experimental results are shown in Figure 5(a)~(c).





(a) Resource Provider = 5 (b) Resource Provider = 10



(c) Resource Provider = 20

Figure 5. Comparisons on Auction Cost with Different Number of Providers and Bidders

According the above results, we firstly find that when the number of providers is in a lower level, the differences between the three auction mechanisms are very insignificant. With the increasing of providers, auction costs will be significantly increased, especially when it reaches 20. In this case, the time cost of BA is the highest among the three auctions. This is because that BA needs to spend more time on price negotiations. If a single resource is auctioned, the time cost is normally ignorable. However, when several resources needs to auctioned concurrently among too much resource providers, the time spend on auction procedure will significantly mitigate the system performance as well as user's satisfactory. As shown in Figure 5, the time cost of BA increases about 300% ~ 900% when the number of providers is increased from 5 to 20. This result indicates that BA can only be used in small-scale systems. As to EVRC, its time cost of auction is very close to that of SPA when the number of bidders is less than 20. when the number of bidders is over 20, the time costs of EVRC is shorter than BA by about 21% ~ 27%. This improvement indicates that the extendibility of EVRC is better than SPA.

5.3 Comparison on QoS Performance

In the second experiment, we take efforts on investigate the QoS performance of our EVRC. Conventionally, there are many QoS related metrics which can be used to measure a system performance from different aspects. In this study, we mainly concentrated on two metrics: *Deadline Miss Rate* and *Rejection Rate*. The reason that we care about the two metrics is that most of client requests will have execution deadline requirement and limited budget in cloud environments. If a user's request cannot be finished before the deadline, it is typically considered that the cloud system fail to satisfy this user's QoS requirement. Meanwhile, if cloud system fails to find suitable resources for a user requests because of its budget, this request needs to be rejected by the system. It is also considered that the system fail to satisfy this user's QoS requirement. The experimental results are shown in Figure 6 and Figure 7.

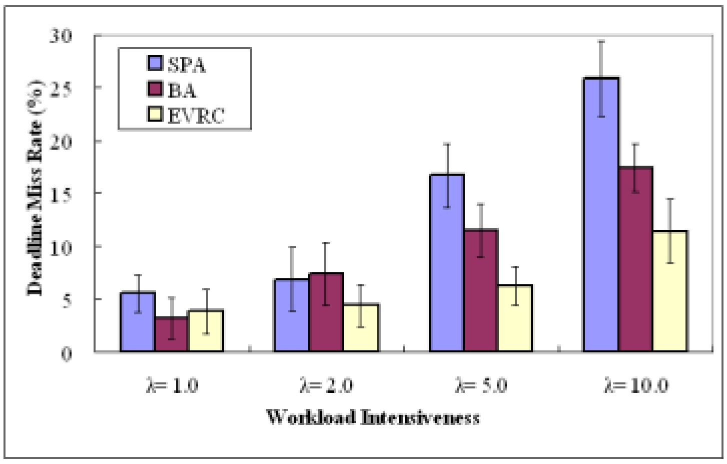


Figure 6. Comparisons on Deadline Miss Rate with Different Workload Intensiveness

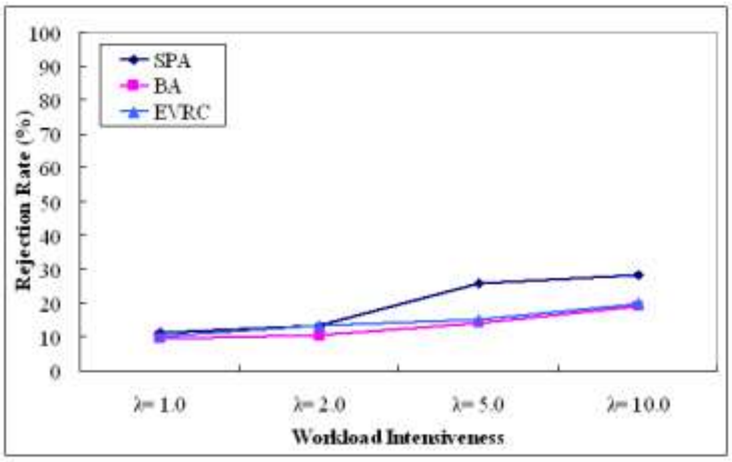


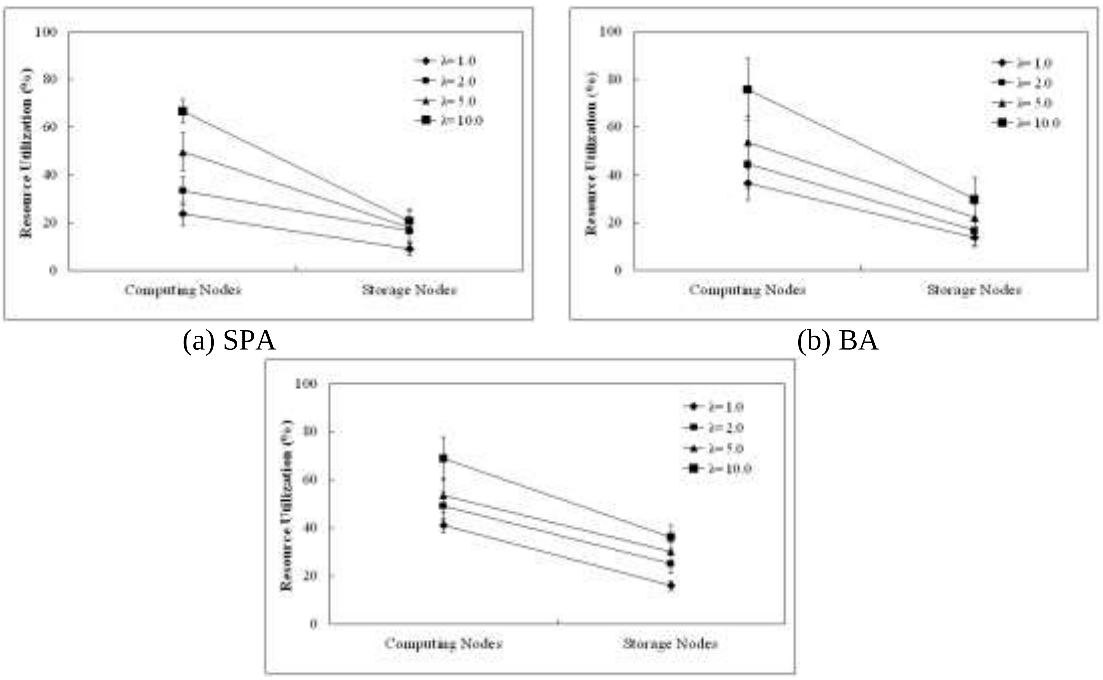
Figure 7. Comparisons on Rejection Rate with Different Workload Intensiveness

In this experiment, we introduce a parameter λ which denotes the average number of requests per second. So, the parameter λ can be considered as the workload intensiveness. We gradually increase this parameter from 1.0 to 10.0 during our experiment. As shown in Figure 6, when λ=1.0 or 2.0 the deadline miss rates for all auction mechanisms can be kept below 6.5%, which is very desirable for most of cloud platforms. However, if we continuously increase the workload intensiveness, the deadline miss rate will be significantly increased in all cases. Even so, we can see that the SPA performs worst in such a case. On the contrary, the deadline miss rate only increases about 7.5% when the workload intensiveness is increased from 2.0 to 10.0. This result indicates that the SPA can not adapt to the fluctuation of workload. It is well-known that the workloads in cloud platforms are inherently unpredictable. An effective resource co-allocation service needs to be adaptive to such an unpredictable workload. So, the EVRC can provide better QoS performance than SPA in term of deadline miss rate.

As to the rejection rate metric, the experimental result in Figure 7 indicate that performance of the BA and the EVRC is very close, while SPA also performs worst in the three auctions. For BA, its bargaining process provides a way for achieving more chances of matching user's requests and provider's supplies. As to our EVRC, it relies on the CDA mechanism which gives more chances for both users and providers when they are intend to trade resources. Although the approaches of BA and EVRC are different, their results are very similar, that is more resource requests can be accepted. So, both BA and EVRC can provide better QoS for end-users in the term of rejection rate.

5.4 Comparison on Resource Utilization

In the final experiment, we investigate the resource utilization metric of the tested three mechanisms. In our cloud infrastructure, most of resources are computing servers or storage servers. As their working status are different, we separately evaluate their utilization rates in all cases as shown in Figure 8(a)~(c).



(a) SPA (b) BA

(c) EVRC

Figure 8. Comparisons on Resource Utilization with Different Workload Intensiveness

As shown in Figure 8, the computing node utilizations of BA and EVRC are slightly higher than that of SPA (about 3% ~ 6%). As analyzed in Section 5.3, BA and EVRC can accept more users than SPA doing. So, this result is very straightforward, since our tested workloads are typically computational-intensive. However, we also can see that the utilization variety of EVRC is lowest in all cases. This result indicates that our EVRC is capable of maintaining all resources working in a relatively stable state. By carefully examining the logs of the experiments, this improvement can be contributed to the performance monitoring service in the EVRC. As noted in Section 4.2, the performance monitoring service needs to frequently update the queue models of all resources. As a result, more storage spaces are required when using EVRC which can be seen in Figure 8(c). According to our estimation, when deploying the EVRC middleware in a small-scale cloud (about 500 active resource nodes), it needs about 2~3TB storage space. In our tested cloud platform, its storage cost is about 5TB. Since the price of hard disk has been very low in recent years, such a cost can be easily accepted for almost all the cloud platforms.

6. Summary and Future Work

In this work, we present an EVRC middleware which provides a set of flexible services that allows cloud applications co-allocating plenty of virtual resources across different resource providers. By conducting extensive experiments on a real-world cloud infrastructure, we find that the EVRC outperforms other co-allocation mechanism in terms of several QoS metrics. Currently, the prototype of EVRC is only tested on the XCP platform, and we are planning to test it on more virtualization platforms including VMware and Hyper-V. In addition, we also plan to re-design the Auction Manager component as an abstract plug-in service. In this way, we can dynamically incorporate more auction mechanisms and offer more options for the users.

References

[1] V. Mauch, "High performance cloud computing", Future Generation Computer Systems, vol. 29, **(2013)**, pp. 1408-1416.
[2] L. Liu and J. Xu, "Clouds and service-oriented architectures", Future Generation Computer Systems, vol. 29, **(2013)**, pp. 271-272.
[3] G. Fox and S. Pallickara, "Recent work in utility and cloud computing", Future Generation Computer Systems, vol. 29, **(2013)**, pp. 986-987.
[4] R. R. Expósito, "Performance analysis of HPC applications in the cloud", Future Generation Computer Systems, vol. 29, **(2013)**, pp. 218-229.
[5] A. Ranabahu, "The Cloud Agnostic e-Science Analysis Platform", Ieee Internet Computing, vol. 15, **(2011)**, pp. 85-89.
[6] B. Schulze and J. Myers, "Middleware strategies for clouds and grids in e-Science", Concurrency and Computation: Practice & Experience, vol. 23, **(2011)**, pp. 2043-2047.
[7] J. L. Hellerstein, "Science in the Cloud Accelerating Discovery in the 21st Century", IEEE Internet Computing, vol. 16, **(2012)**, pp. 64-68.
[8] C. Szabo, "Science in the Cloud: Allocation and Execution of Data-Intensive Scientific Workflows", Journal of Grid Computing, vol. 12, **(2014)**, pp. 245-264.
[9] X. Yang, "Cloud computing in e-Science: Research challenges and opportunities", Journal of Supercomputing, vol. 70, **(2014)**, pp. 408-464.
[10] X. Luo, "Semantic representation of scientific documents for the e-science Knowledge Grid", Concurrency and Computation: Practice & Experience, vol. 20, **(2008)**, pp. 839-862.
[11] P. V. Coveney, "Large scale computational science on federated international grids: The role of switched optical networks", Future Generation Computer Systems, vol. 26, **(2010)**, pp. 99-110.
[12] M. Sànchez-Artigas and P. García-López, "eSciGrid: A P2P-based e-science Grid for scalable and efficient data sharing", Future Generation Computer Systems, vol. 26, **(2010)**, pp. 704-719.
[13] G. Giuliani, "Grid-enabled Spatial Data Infrastructure for environmental sciences: Challenges and opportunities", Future Generation Computer Systems, vol. 27, **(2011)**, pp. 292-303.
[14] D. Kurniawan and D. Abramson, "ISENGARD: an infrastructure for supporting e-science and grid application development", Concurrency and Computation: Practice & Experience, vol. 23, **(2011)**, pp. 390-414.
[15] S. Son, "An SLA-based cloud computing that facilitates resource allocation in the distributed data centers of a cloud provider", Journal of Supercomputing, vol. 64, **(2013)**, pp. 606-637.
[16] Z. Jianrong, "Utility-based virtual cloud resource allocation model and algorithm in cloud computing", International Journal of Grid and Distributed Computing, vol. 8, **(2015)**, pp. 177-190.
[17] J. Broberg, "Market-oriented Grids and Utility Computing: The State-of-the-art and Future Directions", Journal of Grid Computing, vol. 6, **(2008)**, pp. 255-276.

[18] K. Vanmechelen, "Market-based grid resource co-allocation and reservation for applications with hard deadlines", Concurrency and Computation: Practice & Experience, vol. 21, **(2009)**, pp. 2270-2297,.
[19] A. Danak and S. Mannor, "Efficient Bidding in Dynamic Grid Markets", Ieee Transactions on Parallel and Distributed Systems, vol. 22, **(2011)**, pp. 1483-1496.
[20] M. M. Hassan, "A dynamic and fast event matching algorithm for a content-based publish/subscribe information dissemination system in Sensor-Grid", Journal of Supercomputing, vol. 54, **(2010)**, pp. 330-365.
[21] L. Maozhen, "Service matchmaking with rough sets", Proceedings of International Symposium on Cluster Computing and the Grid, **(2006)**; Singapore.
[22] P. Garbacki and V. K. Naik, "A hybrid linear programming and evolutionary algorithm based approach for on-line resource matching in grid environments", Proceedings of International Symposium on Cluster Computing and the Grid, **(2007)**; Rio de Janeiro, Brazil.
[23] X. Wei and H. Liu, "A cloud manufacturing resource allocation model based on ant colony optimization algorithm", International Journal of Grid and Distributed Computing, vol. 8, **(2015)**, pp. 55-66.
[24] A. Bestavros and O. Krieger, "Toward an open cloud marketplace: Vision and first steps", Ieee Internet Computing, vol. 18, **(2014)**, pp. 72-77.
[25] E. R. Gomes, "Pure exchange markets for resource sharing in federated clouds", Concurrency and Computation-Practice & Experience, vol. 24, **(2012)**, pp. 977-991.
[26] Y. Feng, "Price competition in an oligopoly market with multiple IaaS cloud providers", Ieee Transactions on Computers, vol. 63, **(2014)**, pp. 59-73.
[27] G. Di Modica and O. Tomarchio, "Matching the business perspectives of providers and customers in future cloud markets", Cluster Computing, vol. 18, **(2015)**, pp. 457-475.
[28] W. Lin, "A QoS-aware service discovery method for elastic cloud computing in an unstructured peer-to-peer network", Concurrency and Computation-Practice & Experience, vol. 25, **(2013)**, pp. 1843-1860.
[29] D. Bruneo, "A stochastic model to investigate data center performance and qos in IaaS cloud computing systems", IEEE Transactions on Parallel and Distributed Systems, vol. 25, **(2014)**, pp. 560-569.
[30] J.-W. Lin, "Integrating QoS awareness with virtualization in cloud computing systems for delay-sensitive applications", Future Generation Computer Systems, vol. 37, **(2014)**, pp. 478-487.
[31] D. Gross and C. Harris, “Fundamentals of Queuing Theory”, 3rd Edition, John Wiley and Sons Inc., Massachusetts, USA, **(1998)**.

Authors

Zhichao Liu received the M.S. degree in computer science from the Xiangtan University in 2010. Currently, he is an advanced network engineer in HP High-performance Network Centre in Hunan Institute of Engineering. His research interests include cloud computing, distributed resource management..