Research Journal of Applied Sciences, Engineering and Technology 7(12): 2421-2426, 2014
DOI:10.19026/rjaset.7.546
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: March 01, 2013 Accepted: October 19, 2013 Published: March 29, 2014

Research Article

Multi-tenant Data Migration Strategy in SaaS Platform

Yongqing Zheng, Xiaojun Ren and Lanju Kong
School of Computer Science and Technology, Shandong University Shandong Provincial
Key Laboratory of Software Engineering, Shandong University, Jinan, China

Abstract: On demand migration of tenants' data is a critical technology for elastic load balancing in SaaS application. However, different tenants have different required resource with various SLA requirements. Moreover, previously reported migration techniques are not tenant aware. To study the elastic load balancing of tenants' data, we mainly focus on the problem of when to migrate which tenants' data in a muti-tenant shared DB server and build a new framework that aims to monitor load and minimize migration cost. We establish a demand resource estimation model to decide when to migration and then create a cost model to guide the choice of possible migration tenants. The simulation experimental results show the high effectiveness and the good efficiency of our model.

Keywords: Demand resource estimation, elastic load balancing, migration, SaaS, SLA

INTRODUCTION

The galactic popularity of cloud computing has led to the spring up of cloud platforms and attracted a deluge of internet scale applications being deployed in the cloud. SaaS has become a popular cloud computing delivery model with Multi-Tenancy Architecture (MTA) where an instance of software can serve multiple tenants who have specific business needs. Tenants can on-demand access the computational resources with pay-per-use pricing model. As one of the key characteristics of SaaS, multi-tenancy technology (Guo *et al*., 2007; Wang *et al*., 2008; Kwok and Mohindra, 2007) targets to support multi tenants by sharing application instances and databases. It makes the SaaS provision more attractive to their potential customers (tenants), especially those small and medium businesses with limited IT investment budgets (Zhang *et al*., 2010).

Effective resource sharing amongst the tenants is critical to minimize the total cost in a platform with a large number of SaaS applications, each with hundreds (even thousands) of tenants. The SaaS applications are generally deployed over a group of application servers and database servers; we focus on the database servers where the multi-tenants' data are commonly deployed in the SaaS platform by using shared database shared table mechanism (Mietzner *et al*., 2008; Aulbach *et al*., 2008). This shared schema approach has the lowest hardware and backup cost, because it allows serving the largest number of tenants per database server (Carraro *et al*., 2006), which is called data node in this study. In a SaaS platform, different tenants commonly have different resources requirements at different time and have various access patterns. To achieve the high economies of scale, one of the most important issues need to be solved is that, how to optimally load balancing based on tenants' duly resource requirement to maximize the total supported number of tenants on the limited data node cluster without violating their SLA requirements. As a result of the variability in load of tenants, elastic load balancing of tenants' data is critical for performance and total cost minimization. To study the elastic load balancing of tenants' data, it is essential to calculate the resource consumptions of each on hosting data node in order not to violating the tenant's SLA.

Another important issue is insufficient which occurs when the requirement resource such as CPU and storage of one tenant or some tenants cannot be met by the hosting data node. Moreover, potential revenue from customers is lost by missed SLA requirements with a lot of constraints such as response time. On demand migration of tenants' data is a critical technology for elastic load balancing to deal with the insufficient problem.

With the current resource allocation models, capacity planning and resource allocation methods are not tenant aware. Consequently, there exists the need to establish a true elastic actual resource usage computation model for every tenant aims to reduce the fines by missed SLA requirements. Moreover, the SLAs are resource related (e.g., memory space, CPU, storage space), performance related (e.g., response

Corresponding Author: Yongqing Zheng, School of Computer Science and Technology, Shandong University Shandong Provincial Key Laboratory of Software Engineering, Shandong University, Jinan, China, Tel.: 15169114394

time). Once insufficient emerge with the violation of constraints specified in the SLA, migration should be conducted. Detail about "how" to migrate the multi-tenants' data is out of the scope of this study. We focus on the problem of "when" and "which" based on resource demand model to minimize the operating cost of the system.

To address the two challenges mentioned above, we present a new framework based on MAPE loop model. The framework aims to monitor load and minimize migration cost. We establish a demand resource estimation model to decide when to migration and then create a cost model to guide the choice of possible migration tenants.

LITERATURE REVIEW

In the resource management domain distributed resource management is one of the most critical issues. This issue has caught a lot of attention from many research communities in the last couple of years. Srikantaiah *et al*. (2008) and Buyya *et al*. (2008) proposed an energy-aware consolidation method to decrease the total energy consumption of a cloud computing system. The authors modeled the energy consumption of servers as a function of disk utilization rates and CPU mostly based on experience. Liu *et al*. (2009) and Le *et al*. (2010) presented a SLA-based proceeds optimization problem in electronic commerce hosting datacenters. SLA in this research is modeled as a response time constraint and less than a portion of request's response time is able to violate that constraint specifies in the SLA. Resource allocations to satisfy application requirements and resource constrained project scheduling are a generalization of the static job shop problem and have been reviewed thoroughly by researchers (Herroelen *et al*., 1998; Brucker *et al*., 1999; Hartmann and Kolisch, 2000; Kwok and Mohindra, 2007). Most of these studies are based on traditional exact methods, priority rule schedule (Kwok and Mohindra, 2007) and meta-heuristic approach (Mietzner et *al*., 2008; Aulbach *et al*., 2008). However, all these research are not tenant aware without taking into account multi-tenancy characteristics.

Calculations of computing resources, such as CPU and memory, required for an instance supporting multi-users are rather straight forward and simple (Carraro *et al*., 2006). Calculations of resources required for an instance supporting multi-tenants with multi-users in each tenant are new and complicated (Kwok and Mohindra, 2007). A hierarchical and extensible resource management system has been built to allow the execution of multiple scheduling paradigms concurrently (Bagchi *et al*., 2006). A number of commercial software tools for capacity planning, resource allocation and performance analysis for multiple applications on a set of servers are also available (Kwok and Mohindra, 2007; Bagchi *et al*., 2006; Weissman and Bobrowski, 2009). However, these tools do not apply to the data server of for multi-tenant SaaS applications. They primarily focus on the resource allocation on application server. Calculations of resource required on a database server of different tenants are new and complicated.

A lot of research has focused on multi-tenancy data models. Multi-tenancy in the database layer is general in all sorts of systems that support SaaS paradigm such as Salesforce.com (Clark *et al*., 2005). Even though very little work has focused on live migration of databases, a number of techniques have been proposed in the virtualization literature to deal with the problem of live migration of virtual machines (Liu *et al*., 2009). Virtual Machines (VM) migration is a standard feature supported by most VMs and is a primitive used for load balancing in large virtualized clusters, specifically the cloud infrastructures. Clark *et al*. (2005) and Liu *et al*. (2009) propose a migration technique similar to the iterative copy technique proposed in this study. Liu *et al*., 2009) propose an improvement to minimize the downtime and the amount of database synchronized by using a log based copying approach. Note that in the database literature, migration has been studied in a different context-migrating data as the database schema evolves, or between different versions of the database system, or from one system to another. Our work is different from all these reported works since we discuss the migration problem in a different context where migration is used for elasticity load on data node without involving schema evolution or versions upgrades and rely on the resource demand estimation model to minimize the operating cost of the system.

THE MIGRATION FRAMEWORK

Figure 1 show the migration framework based on the Monitor-Analyze-Plan-Execute (MAPE) loop model (IBM, 2006). The monitor module is responsible for collecting resource usage state of tenants from DB nodes. The insufficient detection module calculates resource usage rate based on demand resource estimation model to judge whether there is insufficient. Migration impact prediction module determines a set of tenants needed to be migrated based on cost model. The remaining modules are not focusing of this study, so we will not discuss them.

Multi-tenant demand resource estimation model: In short, the Demand Resource Estimation Model (DREM) for multi-tenancy aims to calculate resource requirement of tenant with a number of users while shared database shared tables. A tenant's data can demand a number of computing resource, such as CPU, storage, memory. For practical reasons, we use CPU and storage to illustrate estimation of resource demand

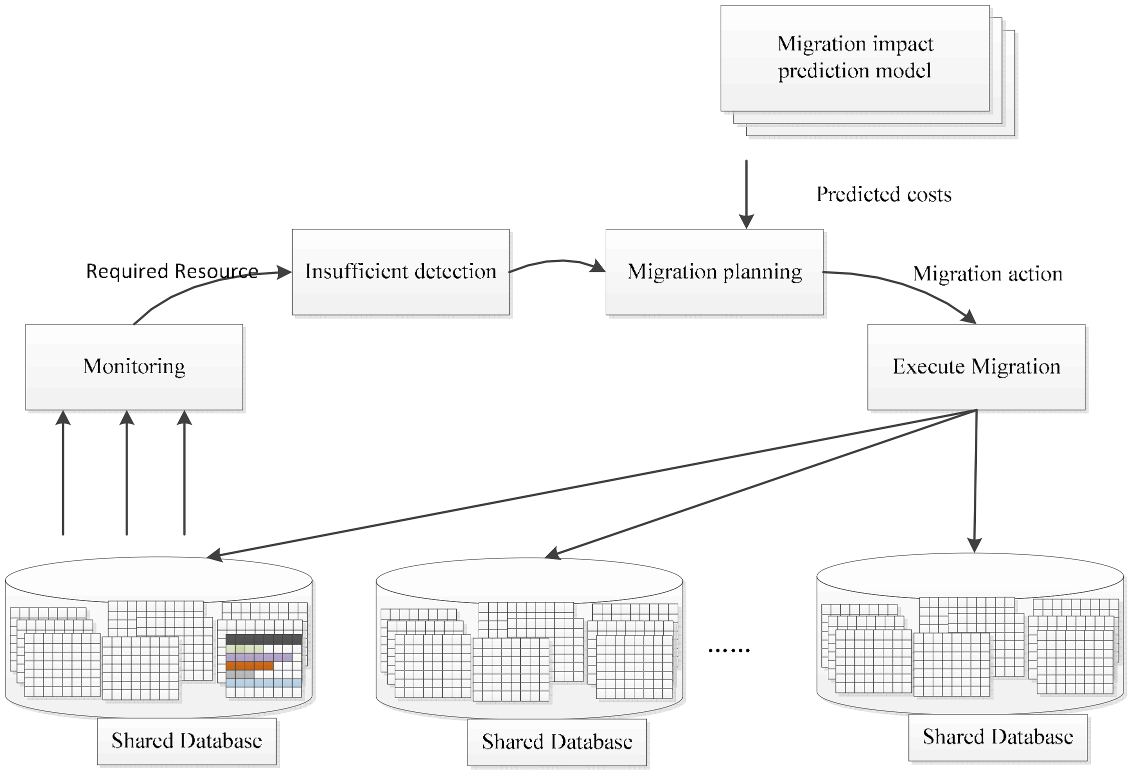

Fig. 1: The migration framework

in this study. (IBM) Show that, for majority of web applications, many types of computing resource consumptions are in near-linear on the number of requests when the system is not overload, such as the CPU of both application server and database server. Therefore, before we give the resource demand estimation model, we do the following two assumptions:

- Storage usage of a tenant can be load dependent and proportional to the number of users in it.
- The frequency of each user data access request is symmetrical.
- Storage usage of all tenants can be load dependent and proportional to the total number of users in data node.

Let n_j be the total users on data node j and t_j be the number of tenants in a shared data node and C (n_j, t_j) and S (n_j, t_j) be the CPU and storage required by all tenants with multi-users in data node j. Then:

$$\begin{aligned} C(n_j, t_j) &= fc(n_j) + fci(t_j) \\ S(n_j, t_j) &= M_0 + fs(n_j) + fsi(t_j) \end{aligned} \tag{1}$$

where, fc (n_j) and fs (n_j) are functions of n_j, assuming CPU is idle if there no active users. M_0 is a constant representing storage requirement of these tables shared among tenants and user. fci (t_j) and fsi (t_j) are functions of t_j. These two functions are additional CPU and storage overhead to isolate tenants from each other in a shared database. For a simple example where there are three tenants on data node x and the number of users in each tenant is all equal to i such that $n_j = 3*i$ and from Eq. (1):

$$\begin{aligned} C(3i,3) &= fc(3i) + fci(3) \\ S(3i,3) &= M_0 + fs(3i) + fsi(3) \end{aligned} \tag{2}$$

y is a constraint factor assuring the utilization rate of a data node is below 1.0 in order to provide higher service reliability and increase uptime which will eliminate or reduce fines caused by missed SLA requirements (Wang *et al*., 2008; Zhang *et al*., 2010). Then the residual resource is:

$$\begin{aligned} C_j^r &= Dc_j * y - C(n_j, t_j) \\ S_j^r &= Ds_j * y - S(n_j, t_j) \end{aligned} \tag{3}$$

If this situation occurs, when or below zero, it means insufficient when the resource requirement of all tenants cannot be met in the hosted data node, as shown in Fig. 2. There are two data node emerge the problem of insufficient, where resource requirements of node 3 and node 4 overrun 7 and 10%, respectively.

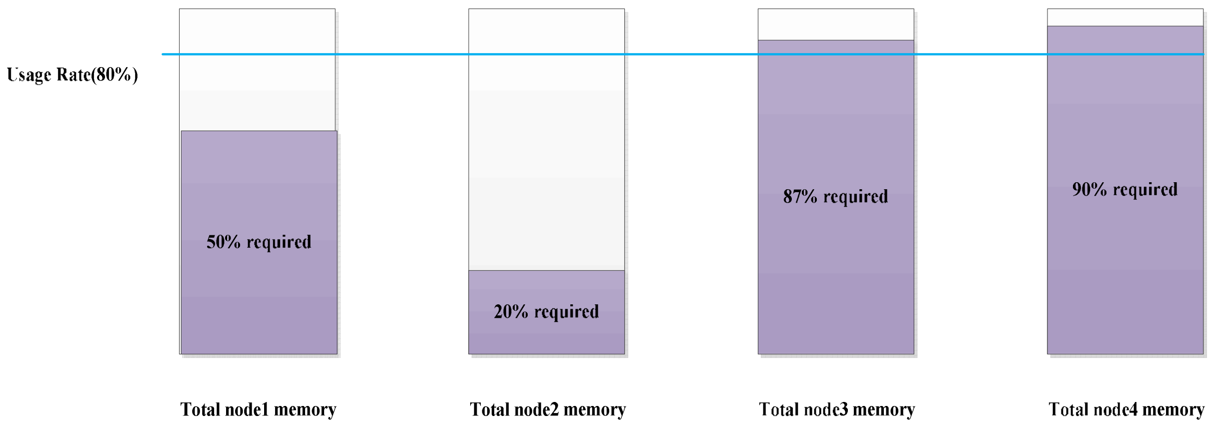

Fig. 2: The memory insufficient

```
Input: Suf_set, InSuf_set
Output:  A migration plan consists of a set of actions
1.        initialize a migration plan A={}, a partition set P={}
2.        Let y be the utilization rate of data nodes
3.        Sort the nodes in  Suf_set in an descending order by residual resource
4.        For each node v in Suf_set do
5.         {
6.        Let u be the most insufficient data node in InSuf_set
7.        Let lt be the load on partition of tenant t;
8.        For each tenant t on u  do
9.            {
10        If ( max{Dc*y-C(tj+1,nj+tu)&&Ds*y-S(tj+1,nj+tu)})
11.       Add t to P
12.           }
13.        }
14.       Sort the tenants in P in an Increasing Order by load lt
15.       Choose the tenants in P who make insufficient data nodes just change
to sufficient
16.             End
```

Fig. 3: The cost-aware migration decision algorithm

Potential customers may be lost by poor performance (insufficient), resulting in a permanent loss of the revenue stream. As a result of on-demand migration of tenants' data is a critical technology for elastic load balancing to deal with the problem of insufficient. We will discuss this problem in next section.

In fact, C (n_j, t_j) and S (n_j, t_j) can be generated by some simple testing experiments for different applications.

COST MODEL BASED ON DREM

In order to make migration more attractive, in addition to optimizing performance (constraints in SLA), the goal also to optimize the total cost of migration operations. Therefore we, define some factors that cause the "cost" of migration.

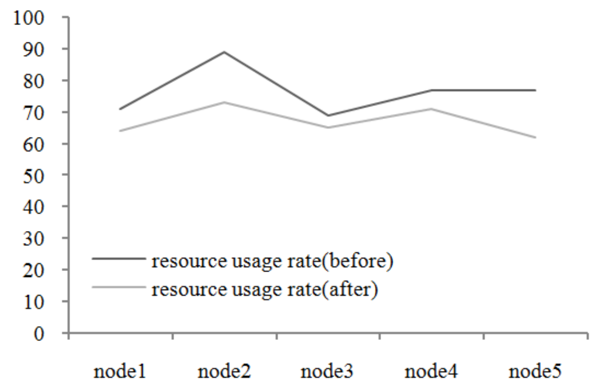

Fig. 4: The effect on resource usage rate of data nodes before and afer DREM

Insufficient: It is because of insufficient and we began to consider he migration. Consequently, there must

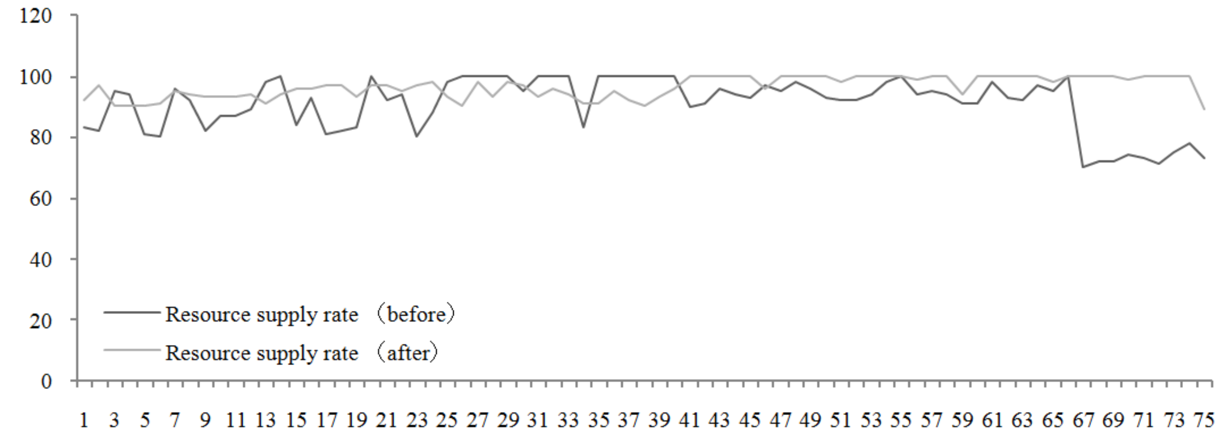

Fig. 5: The effect on resource supply rate of tenants before and after DREM

be no problem of insufficient in the source data node after migration. Similarly the destination data node.

Service interruption or downtime during migration: Since the data node is shared among multi tenants, high availability is a significant demand. Therefore, downtime and interruption in service as a result of migration should be minimized.

Migration overhead: Migration should result in low overhead on the tenant transactions. We define overhead as the additional work done and the corresponding impact on transaction execution to promote migration.

Let l_i be the load value on node i and s_t be the data of tenant t which needs to be migrated. Before calling the algorithm, the data nodes are divided into two sets based on their resource usage state, *Suf_set* and *InSuf_set*, such that *InSuf_set* contains all nodes whose resource usage state is Insufficient and that will some tenants' data will be migrated in, while *Suf_set* contains all nodes that meet the opposite condition.

The migration problem is a multi-objective optimization problem which is NP hard and it is hard to calculate the decision in a very accurate way. Here, we seek to solve this problem using a greedy-based method by making some reasonable simplifications for the aim of easy calculation and implementation. Figure 3 gives the pseudo-code of the cost-aware migration decision algorithm.

EXPERIMENTS AND RESULTS

To validate the effectiveness of the proposed DREM, we did the experiments. Our experiment is implemented in Java 1.6.0. The database is oracle 10g and the application server is Apache Tomcat 6. Both of database and application servers are deployed on PCs. The operating system is windows XP Professional, CPU is Intel Core 2 Duo 2.93 GHz, the memory is 4 GB and the hard disk is 500 G. In our experiments, several applications are deployed to PaaS platform and rented out to tenants. The numbers of users under the tenants varies from dozens to hundreds. The experimental tested consists of five DB nodes, each has 15 tenants.

After each application deployed to PaaS platform, the application data is stored in shared schema mentioned previously. In order to simulate the irregular load and transaction characteristics, we use J Meter tool to create workload. We set constraint factor assuring the utilization rate of a data node is 80%.

From the experiment results, we can see that, with demand resource estimation model, the usage rates of data nodes are overall lower. The results are shown in Fig. 4. The resource supply rates of tenants are overall improved. The results are shown in Fig. 5.

CONCLUSION

In this study, we mainly focus on the problem of when to migrate which tenants' data in a muti-tenant shared DB server and build a new framework that aims to monitor load and minimize migration cost. We establish a demand resource estimation model to decide when to migration and then create a cost model to guide the choice of possible migration tenants. The simulation experimental results show the high effectiveness and the good efficiency of our model. The future research will address to "how" to migrate tenants' data and further improve the previous algorithm.

ACKNOWLEDGMENT

This study is funded by Science and Technology Development Plan Project of Shandong Province No. 2012GGX10134; Independent Innovation Foundation of Shandong University under Grant No. 2012TS075; National Key Technologies R&D Program No. 2012BAH54F04; National Natural Science Foundation of China under Grant No. 61272241.

REFERENCES

Aulbach, S., T. Grust, D. Jacobs, A. Kemper and J. Rittinger, 2008. Multi-tenant databases for software as a service: Schema-mapping techniques. Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD'08). Vancouver, BC, Canada.

Bagchi, S., E. Hung, A. Iyengar, N. Vogl and N. Wadia, 2006. Capacity planning tools for web and grid environments. Proceeding of the 1st International Conference on Performance Evaluation Methodologies and Tools, pp: 25-34.

Brucker, P., A. Drexel, R. Mohring, K. Neumann and E. Pesch, 1999. Resource-constrained project scheduling: Notation, classification, models and methods. Eur. J. Oper. Res., 112: 3-41.

Buyya, R., Y.S. Chee and S. Venugopal, 2008. Market-oriented cloud computing: Vision, hype and reality for delivering it services as computing utilities. Proceeding of 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09).

Carraro, G., R. Wolter and F. Chong, 2006. Multi-Tenant Data Architecture, MSDN Library. Retrieved form: http:// msdn. microsoft. com/ enus/ library /aa479086.aspx.

Clark, C., K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt and A. Warfield, 2005. Live migration of virtual machines. Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI). Berkeley, CA, USA, pp: 273-286.

Guo, C., W. Sun, Y. Huang, Z. Wang and B. Gao, 2007. A framework for native multi-tenancy application development and management. Proceeding of the 9th IEEE International Conference on E-Commerce Technology and the 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services (CEC/EEE 2007), pp: 551-557.

Hartmann, S. and R. Kolisch, 2000. Experimental evaluation of state-of-the-art heuristics for resource-constrained project scheduling problem. Eur. J. Oper. Res., 127: 394-407.

Herroelen, W., B.D. Reyck and E. Demeulemeester, 1998. Resource-constrained project scheduling: A survey of recent developments. Comput. Oper. Res., 25(4): 279-302.

IBM, 2006. An Architectural Blueprint for Autonomic Computing. IBM White Paper 2006.

Kwok, T. and A. Mohindra, 2007. Resource calculations with constraints and placement of tenants and instances for multi-tenant SaaS applications. Proceeding of ICSOC, LNCS 5364: 633-648.

Le, K., R. Bianchini, T.D. Nguyen, O. Bilgir and M. Martonosi, 2010. Capping the brown energy consumption of internet services at low cost. Proceeding of International Conference on Green Computing (Green Comp), pp: 3-14.

Liu, H., H. Jin, X. Liao, L. Hu and C. Yu, 2009. Live migration of virtual machine based on full system trace and replay. Proceedings of the 18th International Symposium on High Performance Distributed Computing (HPDC'09), pp: 101-110.

Mietzner, R., F. Leymann and M.P. Papazoglou, 2008. Defining composite configurable SaaS application packages using SCA, variability descriptors and multi-tenancy patterns. Proceeding of 3rd International Conference on Internet and Web Applications and Services (ICIW '08), pp: 156-161.

Srikantaiah, S., A. Kansal and F. Zhao, 2008. Energy. Proceeding of Workshop on Power Aware Computing and Systems (Hot Power 08), Dec. 2008.

Wang, Z.H., C.J. Guo, B. Gao and W. Sun, 2008. A study and performance evaluation of the multi-tenant data tier design patterns for service oriented computing. Proceeding of IEEE International Conference on e-Business Engineering (ICEBE'08), pp: 94-101.

Weissman, C.D. and S. Bobrowski, 2009. The design of the force.com multi-tenant internet application development platform. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD), pp: 889-896.

Zhang, Y., Z. Wang, B. Gao, C. Guo, W. Sun and X. Li, 2010. An effective heuristic for on-line tenant placement problem in SaaS. Proceeding of IEEE International Conference on Web Services (ICWS), pp: 425-432.