Efficient Hardware Resource Allocation to Achieve Performance in Multi-tenancy at DBaaS

Padmavathi.S [1], Selvi.S[2], Dr.T.Rajendran[3]

[1]PG Scholar, CSE Department, Angel College of Engineering and Technology
[2] Assistant Professor, CSE Department, Angel College of Engineering and Technology
[3]Dean , CSE and IT Department, Angel College of Engineering and Technology
[1]padmaspretty@gmail.com, [2] selvi.me08@gmail.com, [3]rajendran_tn@yahoo.com

***Abstract*-Multi-tenant technology is one of key competencies for Software as a Service (SaaS) to achieve higher profit and performance for the cloud users. Multi-tenancy enables to share resources of a single database server among multiple tenants. This multi-tenancy enables cost reduction for the cloud service users. However, resource sharing can affect a tenant's performance due to resource demands of other tenant's workloads. During the conditions like number of tenants increase's, in the resource utilization and according to their demands. Database as a Service (DBaaS) is one of the service delivery model in which resources like (Memory, CPU and I/O) are accessed remotely by users. The performance of database server resources depends on the key features such as CPU, I/O and memory, for each tenant. The problem occurs when more number of End user's increase for a tenant, hence the static database allocation method is followed. The RDBMS memory storage is mostly used for storing and retrieving of data's to number of tenants/end user's. While there are many uses of memory in a relational DBMS, while we allocate static memory to each tenant (a certain amount), it will not be sufficient enough when the total number of End user's exceeds at a session, hence the usage of the resource, storage varies according to their application. To overcome this issue, by using the Resource allocation strategy with Genetic algorithm. And the process framework involves monitor, analyzer, predictor and allocator.**

General Terms-Multi-tenancy, Tenant, Performance.

Keywords-DBaaS, End Users, Residual Memory.

1. INTRODUCTION

Cloud applications or Software as a Service (SaaS) applications deliver software as a service over the Internet, eliminating the need to install and run the application on the customer's own computers and simplifying maintenance and support, and equipped with decomposable applications, managed services, shared hardware /software resources and Web-based services. Cloud computing platform provides the scalability, availability and utility computing for services on Internet.

Software as a Service (SaaS) is a software delivery model in which software resources are accessed remotely by users. The SaaS model adopts the "single-instance multi-tenancy". SaaS (Software as a Service) is a modern approach to deliver large scalable enterprise software as a service on Internet.

One of them is multi-tenancy, which allows single instance of software to serve multiple organizations by accommodating their unique requirements through configuration at the same time. Enterprises find SaaS attractive because of its low cost. SaaS requires sharing of application servers among multiple tenants for low operational costs. Besides the sharing of application servers, customizations are needed to meet requirements of each tenant. Supporting various levels of configuration and customization is desirable for SaaS frameworks.

[1]Cloud Computing is quickly being adopted by organizations and businesses alike to help increase profit margins by decreasing overall IT costs as well as provide clients with faster implementation of services. The majority of the cloud service providers offer multi-tenancy to capitalize on the associated economies of scale which also translates into savings for the end user. In fact the competitive nature of cloud computing is such that cloud service providers have to minimize the total cost of ownership of their IT infrastructure, thus introducing multi-tenancy is a popular way to reducing total cost of ownership. Multi-tenancy has made cloud computing popular by allowing businesses to benefit from reduced costs yet continue to gain access to data and applications within a cloud environment. In the multi-tenancy model, many user's data and resources are located in the same computing cloud, and are controlled and distinguished through the use of tagging for the unique identification of resources owned by individual user. In a typical multi-tenancy situation, the users are the tenants and are provided with a level of control in order to customize and tailor software and hardware to fit their specific needs.

Database-as-a-Service (DBaaS) is gaining significant momentum with more vendors providing offerings. Amazon RDS offers the full capabilities of a familiar MySQL database by running MySQL instances in different EC2 instances (VM Instances) for different tenants. Here, the DBaaS is effectively used as SaaS for the memory allocation purpose for the end users in an organization.

The paper is organized as follows. Section 2 discusses related work, Section 3 holds the concepts of problem description, followed by Section 4 the proposed system is explained which includes the resource allocation framework and in Section 5 conclusion is given.

II. RELATED WORK

Multi-tenancy is one of the most important concepts for any SaaS application. Based on it, real cloud computing aims for ”better resource utilization” for SaaS providers and ”Pay as you go” for consumer[4].There are many ways to develop SaaS application based on hosting environment, architecture to follow, and available development skills. One can develop SaaS and deploy it using proprietory online framework like Salesforce.com, Google AppEngine, and Microsoft Azure etc. SaaS also can be developed using core programming framework like JDK for java and .NET. SaaS application development should be independent of underlying infrastructure so that application can be migrated from one cloud to another cloud without changing the code. It is possible only if the entire player follow the identical as well defined standard SaaS architecture [5]. The SaaS provides management facility to administrate tenants, users of a tenant, services and systems. SalesForce.com provides appex programming and web base tool to SaaS providers to develop, integrate and customize its service based on CRM platform [6].

[3]Generally speaking, web service handles all the requests equally according to the Best-Effort service model, which uses First in First out (FIFO) scheduling strategy. However, the performance improvements realized by IntServ and DifServ will be seriously affected and unable to meet the different requirements of various users. For example, when the load is heavy, it will discard all the overloaded user requests, or prolong response time without consideration of prioritization. Such abuse also exists in multi-tenant oriented services. All SaaS applications share a high-performance server; therefore failure of any SLA control may lead to system instability and the decline of server's overall performance. At present, the research on assurance mechanisms of service quality has become a hotspot.

Research on service quality oriented distribution or scheduling mechanism in the area of grid computing is relatively mature. There's Globus Architecture for Reservation and Allocation (GARA) [7] which is able to provide peer-to-peer grid service based on Quality of Service (QoS). GARA implements a uniform management mechanism for diverse resources, such as CPU, network, storage equipment. It also offers several operations on service quality management, including resource selection, distribution and release, quality of service assurance, and point-to-point resource management.Database-as-a-Service (DBaaS) is gaining significant momentum with more vendors providing offerings. Amazon RDS offers the full capabilities of a familiar MySQL database by running MySQL instances in different EC2 instances (VM Instances) for different tenants [8]. Microsoft SQL Azure provides a SQL server database for different tenants in a shared pool of SQL server instances with a controlled allocation of resources for each database. For all these different settings, tenants choose from different SLAs (service level agreements) at different price points [9]. In RDS, tenants pay \$0.11/hour for 1 ECPU with 1.7 GB Mem, and \$1.55/hour for 13 ECPU with 34G Mem. The consumption of resources that exceeds a tenant’s SLA will be charged at a per usage rate. A key problem in DBaaS is how to efficiently share resources among tenants while maintaining SLAs [10].

SLAs guarantee performance isolation to prevent one tenant from adversely affecting the performance of other tenants in an unpredictable manner. From the perspective of the DBaaS provider, SLAs must be met to avoid SLA violations and penalties [11]. On the other hand, it is also crucial for the tenant's resource demand varies in the course of time. While allocating resources compliant with SLAs is safe, it can lead to lost revenue for the DBaaS provider and low resource utilization. However, the inaccuracy of predicting demands for tenants increases the risk of SLA violations.

III. PROBLEM FORMULATION

A multi-tenant SaaS product should be efficient enough to scale seamlessly without compromising on reliability, availability and performance. Large scale applications which are built with the intention of handling thousands of users accessing in a concurrent fashion is to be well equipped and architected to handle a medium sized customer with few hundreds of users to a large customer with fairly huge number of active users. When the number of tenants and users grow, the number of IOs which hits the database will also increase which is susceptible for performance degradation.

3.1 Characteristics & Methods

A. Performance: The performance of database server resources depends on the key features such as CPU, I/O and memory, for each tenant. The key challenge of a shared DBMS is static allocation of resources to tenants, while the occurrence of low overheads and scaling to large numbers of tenants. The technique should focus to contribute in: designing mechanisms to support the promised resources and proposing low-overhead techniques to increase the performance in multi-tenant environment.

B. Degradation: The performance degradation in SaaS service occurs during the multi-tenancy. In the case of, hardware resource allocated to a tenant (e.g. an organization) the performance should be maintained according to the SLA. In some case, during the abnormal conditions like overload or bursty condition (i.e.) more number of users is in need of hardware resource at same time of a tenant. The provider allocated memory resource to the tenant should be efficiently scheduled to the "N" number of users, even in the overload condition also.

C. Method: To achieve the better performance in multi-tenancy within a tenant. We use the following process:
a) Monitor b) Analyzer c) Predictor d) Allocator.

IV. PROPOSED SYSTEM

In our work, to solve the performance among the tenant and to allocate the available hardware resources we propose the Resource allocation framework and to calculate the Residual memory (available resources) from the other users within a tenant. Below the section will describe the framework used and their parameters.

4.1 Prediction of Available Hardware Resource

The Residual resource is calculated and scheduled to the users. It is calculated by using these steps as below:

Step1: During the time of more number of users requested arrives. By using the queue we arrange and order the user according to the priority, based the FIFO fashion of the user.

Step2: Residual resource is calculated, by analyzing the total usage of hardware resource and remaining.

Step3: with these details allocate/schedule the resource to the users according to their request. Monitoring is done overall the process of request for resources and allocation. Among the tenants the SLA violation should be avoided.

Note: Residual resource means resources that have not been allocated to any active tenants. The remaining hardware resource that is not currently in use to any users, which is free, is allocated to the requested users.

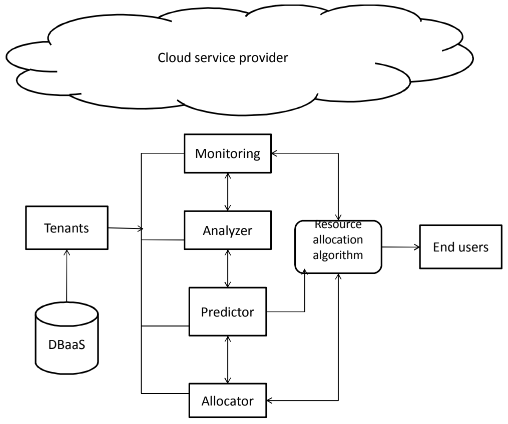

Figure 1.Architecture Diagram

4.2 Resource Allocation Framework

We perform the dynamic resource allocation by using Genetic algorithm for the above problem formulation, the DBaaS resources on the servers need to employ four components:

A. Monitor: measures the workloads of user's and logs the information in the database and holds the details about each user logins and various services. The Parameters are User ID,Service type-regarding their application.

B. Analyzer: Analyzer examines monitored data, and provides input to the Predictor and Allocator. Two jobs are performed by the Analyzer: (1) counting the number of current usage memory and resource information; (2) comparing the observed workloads with the assigned storage in terms of SLA. The various parameters that used in the analyzer module include are CPU usage, Memory usage, and Network usage. But currently we focus on first two parameters. With the help of SLA and tenant current usage information, can able to calculate the residual memory. Store the residual memory to database.

C. Predictor: The workload prediction has two purposes: (1) to decide on the demand pattern of a workload; (2) to make some model graph/report for the predicated memory. Here, the inbuilt process takes progress in backend and updated, hence it takes some time for processing. It includes these parameters: User ID, Service type and Residual memory. And to generate a report, we need User ID and Predicted memory.

D. Allocator: A tenant cannot consume resources more than its allocation. If the real demands of a tenant are more then it cannot be observed. Likewise, the resource is allocated to the each user's. Based on the above module results and by using the allocation algorithm. If the available memory is greater than or equal to the user requested memory, it proceeds the allocation else it notify as SLA violation or need additional resource. The Parameters used are Available memory (Residual memory), User ID, Predicted memory and new residual memory (ie), current available memory which is predicted.

V. SIMULATION RESULTS

To evaluate the performance of our multi-tenancy system we have simulated the different structures and workloads from users by using CloudSim 3.0.

CloudSim is a framework for simulation and modeling of cloud computing environments mainly used for resource allocation and scheduling. Here in the below table we have taken 4 parameters we consider while allocating to the end users.

Table 1. Tenant with its parameters

UID	MEMORY	CPU	RAM
1	350	2	512
2	500	2	2048
3	400	3	1024
4	500	1	512

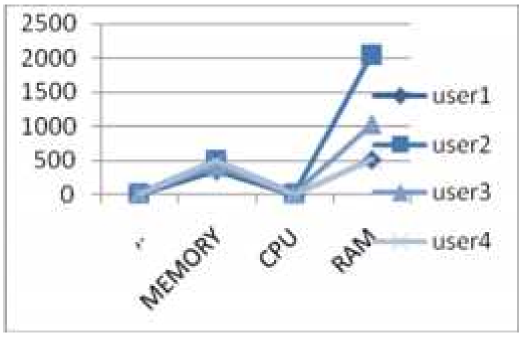

Figure 2. Analysis of allocation parameters

Here 10 workloads from different users of the same tenant are considered with a limited SLA based parameters required for the allocation and further depending upon the prediction of earlier completion of tasks the workloads are scheduled subsequently. Experimental results show that the dynamic algorithm can always achieve higher resource utilization than static algorithms, although SLA violation events may take place.

VI. CONCLUSION

In this paper, we proposed a dynamic resource allocation framework by using genetic algorithm for DBaaS. In the framework, resources are allocated periodically with the objective of high resource utilization under SLA violation and fairness constraints. The memory utilization is better and faster for the end user and the tenant can effectively allocate the resource by using the resource allocation framework. The proposed framework over DBaaS can achieve higher resource utilization if there is a higher variance in user's number among the tenant up to the SLA assigned DBaaS service by the cloud service provider. The proposed method for hardware resource allocation over end users among a particular tenant will enhance the performance and, it is effective and efficient. In future, we have an idea of applying effective monitoring algorithms to further improve the performance of DBaaS allocation. May concentrate on the individual user priority based on their hit ratio and make the service avail to them as the SLA favour to tenants.

REFERENCES

[1] Wonjae Lee and Min Choi, "Multitenancy - Security Risks and Countermeasures",Fifth International Conference on Cloud Computing, IEEE computer society, 2012, pp. 970-971.

[2] Piyush Aghera, Sanjay Chaudhary and Vikas Kumar, "An Approach to Build Multi-Tenant SaaS Application with Monitoring and SLA", International Conference on Communication Systems and Network Technologies, IEEE computer society, 2012, pp. 658-66.

[3] Xu Cheng, Yuliang Shi,and Qingzhong Li, "A Multi-tenant Oriented Performance Monitoring, Detecting and Scheduling Architecture Based on SLA", IEEE-2009 , pp. 599-604.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph,R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and et al., "Above the clouds:A berkeley view of cloud computing," Computing,no. UCB/EECS-2009-28, 2009. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[5] G. Liu, "Research on independent saas platform," in Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on, April 2010, pp. 110 –113.

[6] C. D. Weissman and S. Bobrowski, "The design of the force.com multitenant internet application development platform," in Proceedings of the 35th SIGMOD international conference on Management of data, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 889–896. [Online]. Available: http://doi.acm.org/10.1145/1559845.1559942

[7] Foster I. and Kesselman C. "Globus: A Toolkit-Based Grid Architecture". Foster, 1.and Kesselman, C. (eds.), The Grid: Blueprint for a New Comput ing Infrastructure. Morgan Kaufmann, 1999. pp. 259-278.

[8]Amazon Rational Database Service http://aws.amazon.com/rds/

[9]SQL Azure: Database-as-a-Service http://www.microsoft.com/windowsazure/sqlazure/

[10] A.Stefan, G.Torsten, J.Dean, K.Alfons, R.Jan, “Multi-tenant databases for software as a service: schema-mapping techniques,” Proceedings of the 2008 ACM SIGMOD international conference on Management of data, pp. 1195-1206.

[11] L.Hui, C.Giuliano, E.Tariq, “SLA-driven planning and optimization of enterprise applications, Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering,” 2010, pp. 117-128.