

# Secure Cloud Multi-tenant Applications with Cache in PaaS

K.R. Remesh Babu, S. Saranya and Philip Samuel

**Abstract** Multi-tenant applications come into existence in clouds, which aims “better resource utilization” for application provider. Today most of the present application optimizations are based on Service Level Agreements which focuses on virtual machine (VM) based computing service, while other services such as storage and cache are often neglected. This paper mainly focuses on cache based approach for multi-tenant application on PaaS. Currently in multi-tenant cloud applications data are often evicted mistakenly by cache service, which is managed by existing algorithms such as LRU. It keeps the query information to reload the evicted data from storage which might be sensitive. Hence there is a possibility of data breach when these data are accessed improperly by other tenants. For faster access caching of the data is common in cloud based applications while the security is an important area that should not be neglected when these systems uses other third party systems/networks as caching servers. Also security of the tenant’s data/information is also a crucial component of the SLA between cloud service provider and tenant. So this paper proposes a DES based information security framework within Platform as a Service (PaaS) for better security and Quality of Service (QoS).

**Keywords** Cloud computing • Multi-tenant • Cache protection • QoS

---

K.R. Remesh Babu (✉) • S. Saranya  
Government Engineering College Idukki, Idukki, Kerala, India  
e-mail: remeshbabu@yahoo.com

S. Saranya  
e-mail: saranyasangan3@gmail.com

P. Samuel  
Cochin University of Science and Technology, Kochi, India  
e-mail: philipcusat@gmail.com

## 1 Introduction

A multi-tenant application supports tenants to share one application and database instance while allowing them to configure the application to fit their needs as if it runs on a dedicated environment [1]. To achieve QoS for each tenant, more and more people advocate these applications should abide by SLAs to perform computation. Several works have offered solutions to help these applications to improve resource utilization during runtime. Currently most of these works focus on virtual machine (VM) resources.

Multi-tenant database schema is used for storing meta-data in cloud storage. It is designed and implemented with five techniques [2]. Meta-data driven data-sharing storage model is proposed to implement multitenant applications on top of a standard relational database. This approach works by splitting up the “common tables” shared by each tenant, and mapping the data to “meta data tables” and “data tables”.

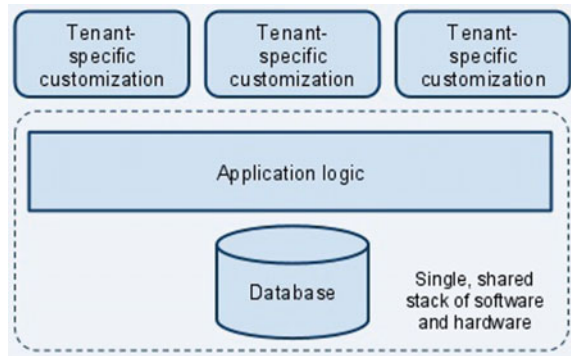
The rapidly increasing gap between the relative speeds of processor and main memory has made the need for advanced caching mechanisms more intense than ever. Cache service in cloud computing area provides faster access to users. It stores small chunks of arbitrary data (strings, objects) from results of database calls, API calls, or page rendering [3].

Cloud distributed cache [4, 5] service plays a vital role in improving cloud application performance. It reduces latency and improves user experience greatly. In Google App Engine (GAE) Memcache service is used for multitenant application. Memcached service is open source and widely used by many high-traffic sites, such as Facebook, Live Journal, Wikipedia and Fotolog. In cloud, the time for reading/writing data from/into cache  $T_{tcache}$  and reading/writing data from/into data store  $T_{tround}$  includes transfer time between different nodes. Where  $T_{tround}$  time is much greater than  $T_{tcache}$  [5]. Hence effective utilization of cache service leads to high hit rate and low response time. Therefore, how to improve cache hit rate by reasonably appointing data to cache becomes a key to multi-tenant application success.

Security is considered as one of the top ranked open issues in adopting the cloud computing model. Internal security is the main issue in multi-tenant application with cache. This is due to improper access of memory by other tenants. Different security models are used in multi-tenant application on SaaS has some limitations. Currently no security mechanism is used in multi-tenant application on PaaS.

This paper introduces a security oriented cache approach that tries to help the tenants by caching data securely. It provides better cloud cache service utilization. Since public key cryptography requires significant computing power than the symmetric key method DES is used for encrypt the data securely on PaaS. It provides integrity of original cache service and improves application portability for different users.

**Fig. 1** Multi-tenant application



## 2 Multi-tenant Application

Multi-tenant applications development and adoption are greatly promoted by cloud computing [1], which aims for “better resource utilization” for application provider. Figure 1 show the overview of the multi-tenant application, where tenants share one application and database instance.

A multi-instance approach [6], in which each tenants gets its own instance of the application (and possibly also of the database). Multi-tenant application can be built on top of Software as a Service (SaaS) and PaaS. Service Level Agreement (SLA) can be incorporated to the tenants [7, 8] to ensure required QoS.

## 3 Data Store

### 3.1 *i-Meta Data*

There are five techniques on design and implementation multi-tenant database schema [2], which can extend default data model to address tenant’s unique needs. However, there are some drawbacks or limitation in these techniques. Table 1 shows the five techniques and its shortcomings.

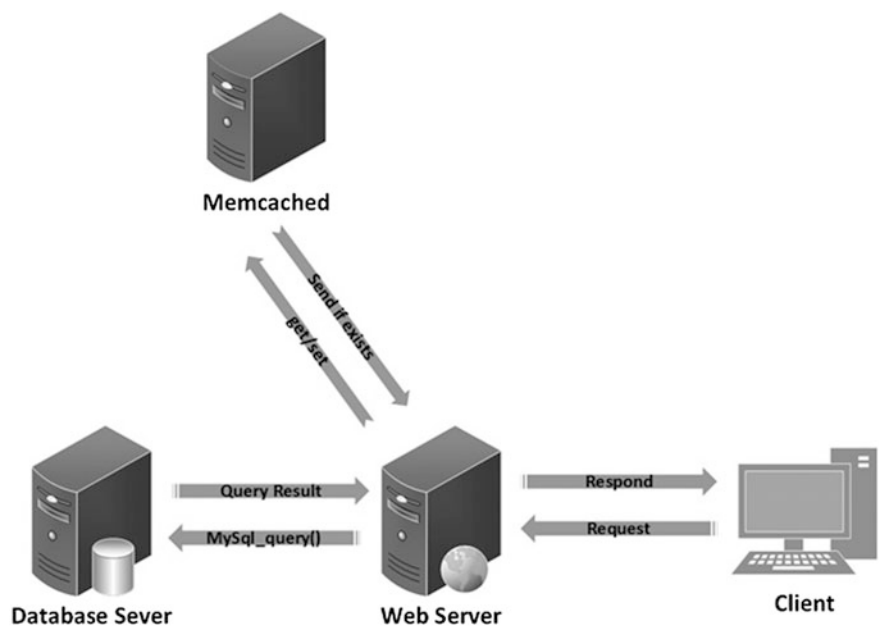
To overcome these limitations a novel meta-data driven data-sharing storage model can be used to implement multitenant applications. It is built on the top of a standard relational database. In this approach splitting up of the “common tables” shared by each tenant is carried out. Then mapping of the data to “meta data tables” and to “data tables” are done.

**Table 1** Different techniques and drawbacks

Techniques	Limitation
Extension table [2]	Number of tables will be increased by increasing the number of tenants
Universal table [2]	Rows need to be very wide, even for narrow source tables indexes are not supported
Pivot table [2]	It has more columns of meta-data. Higher runtime overhead for interpreting the metadata
Chunk folding [2]	Lack of an effective vertical partitioning algorithm to get the most appropriate results
XML table [2]	Limited to extend fields in a table

3.2 Cache and Replication

In the proposed method Memcached is taken for cloud cache service. Figure 2 illustrates on how Memcached works in web application. Request sent by clients for a dynamic web page usually contains data query to database server. Web server will find and get the data in the form of object in Memcached through application server. If the object is found, it will return to web server and respond to the client. However if object is not found in Memcached or ‘cache miss’, the data will be fetched from



**Fig. 2** Memcache operation in web application

database server and it will be set to Memcached as a new item before it is returned to the client. The same process will be repeating all over again.

How can manage the cache in shared environment? Cache replacement strategy is at the heart of cache management. A machine learning approach [9] is used to reconfigure the cache strategy online, which is off-line training coupled with online system monitoring. In [4, 5] introduced an approach to select optimal cache strategy dynamically using trace-driven simulations, so as to differentiate caching and replication policies for each document based on its most recent trace. Authors in [10] evaluated the most likely strategies rather than the entire set of candidate strategies, capturing the history of transitions between different cache strategies. Cache replacement algorithm [11] in adaptive processor is presented for different workloads. These workloads switch between any two replacement algorithms such as LIRS, LRU, LFU and Random. The feedback control theory to allocate cache space was proposed in paper [12]. This self-adaptive multi-tenant memory management achieves each tenant's SLA requirement. It minimizing the memory consumption and dynamically generates a series of cache replacement units according to the current access model was done in [13]. In [14] a Proportional Hit Rate method is introduced to meet clients' SLAs.

Since security is major concern in cloud computing platform the researchers are proposed several methods to address it. An Authorization model [15] is introduced for controlling access to resources in a cloud system. This describes a multi-tenancy authorization system suitable to middleware service in the PaaS layer. The paper [16] introduced a new cloud security management framework based on aligning the NIST-FISMA standard. It based on collaboration among cloud stakeholders to develop a cloud security specification and enforcement covering all of their needs. This approach trying to overcome lack of security constraints in the SLA between the cloud providers and consumers results in a loss of trust as well.

Cryptographic Trust agreement [17] is proposed in order to increase trust in the answers given by services during the negotiation process. It enables SLA approach to ensure that the data of cloud service users are not processed or stored at undesired locations. If the chosen service violates its assured service quality, the service user is enabled to use alternative services. For this, a trusted agreement is provided in order to prevent malicious services; the analysis request is kept secret by encrypting it with the public key of the certified program analysis.

RandTest method [18] is a lightweight and robust dataflow integrity checking method. This approach use a trusted third party to pre-compute results of some randomly generated testing data. Next introduced a method called TOSSMA [19], a Tenant-oriented security management architecture for multi-tenant SaaS applications. TOSSMA is based on "Tenant-Oriented Security", which overcomes the existing classic model "Service-Oriented Security".

Signed Query [20] technique used to improve the confidentiality of users' data stored on the cloud. The usage of a signature to sign the tenant's request, so the server can recognize the requesting tenant and ensure that the data to be accessed is belonging to this tenant. It uses a custom HTTP scheme based on a keyed Hash

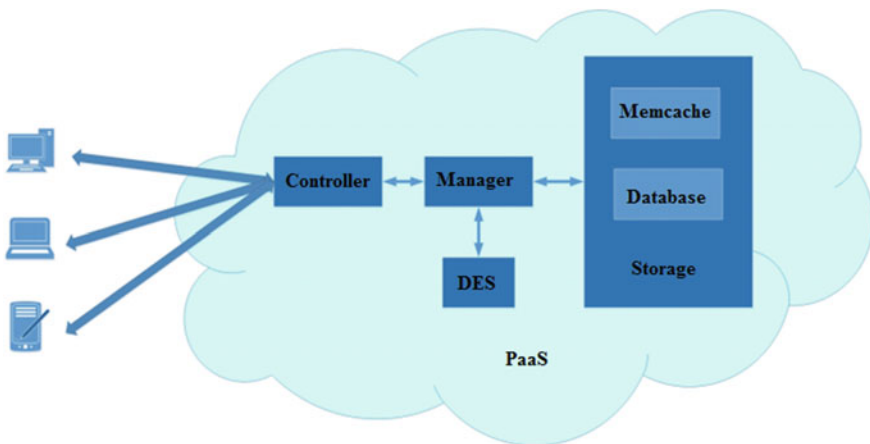
Message Authentication Code (keyed-HMAC) [21] for authentication. This approach uses the tenant's secret key to create the HMAC of that string.

A scalable, flexible, resilient, and cost-effective Hybrid Security Architecture (HAS) solution for data center security service is proposed in [22]. It decouples security service provisioning from network routing, thus facilitating operation and management. From these reviews, it is clear that there is no information security mechanism proposed for tenant caching method for multi-tenant applications.

## 4 Security Oriented Cache Approach

The main issue in multi-tenant cloud application with cache is internal security due to improper memory access by other tenants. In this proposed method information security is provided to multi-tenant application with cache by avoiding improper data access by other tenant's. Public-key cryptography requires significantly more computing power than symmetric cryptography, i.e. strong key pair can take hundreds or even thousands of times as long to encrypt and decrypt data as a symmetric key of similar quality. So in the proposed method more-efficient symmetric DES is used to encrypt the tenant's critical data inorder to prevent improper access by the other active tenants. DES requires lesser computational power compared to other public key encryption methods.

Figure 3 illustrates security oriented cache for multi-tenant application on PaaS. It is built on PaaS, so available to all devices that are connected to the Internet. Tenants uses web browser to access applications. Tenants can send request to the controller for getting data or inserting data into database. Controller will map the user request to manager. Data encryption and decryption is done by manager.



**Fig. 3** Proposed security oriented cache approach

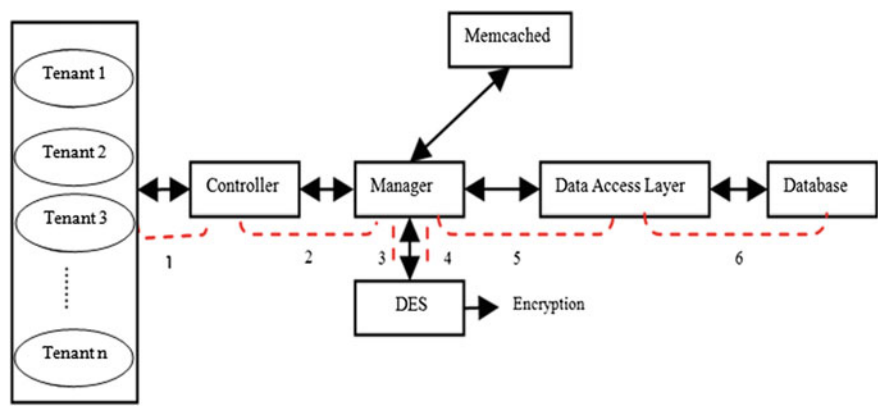


Fig. 4 Insertion of user data into database

For inserting data into database, manager first encrypts the user data and passes it to storage unit. Only encrypted data is stored in database. For getting data of a particular user, the Manager will find it and get the data in the form of object in Memcached. After getting data from Memcached, manager will decrypt the data and returned it to the user.

If the object is not found, data is fetched from the database server and it will be set to Memcached as a new item. Then decrypt the data from database and returned to the user. The same process will be repeating all over again.

Figure 4 illustrates the insertion of data item into database. User first sends a request to the controller with data, key and its id (1). Controller will map the request

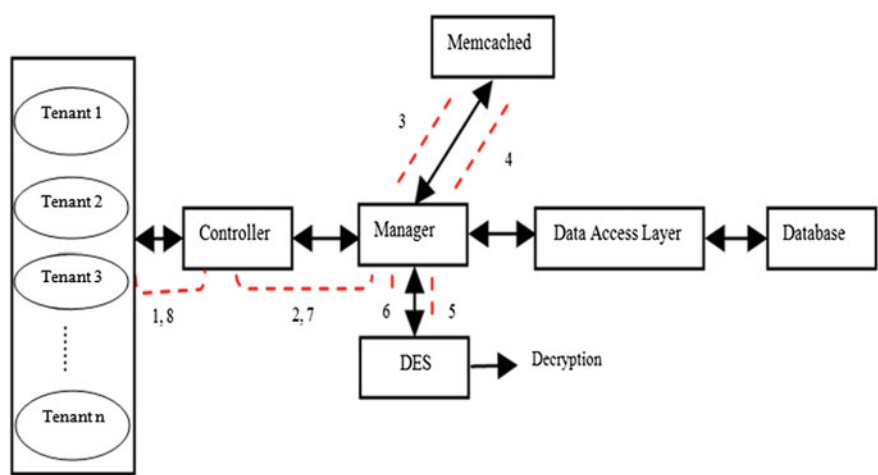
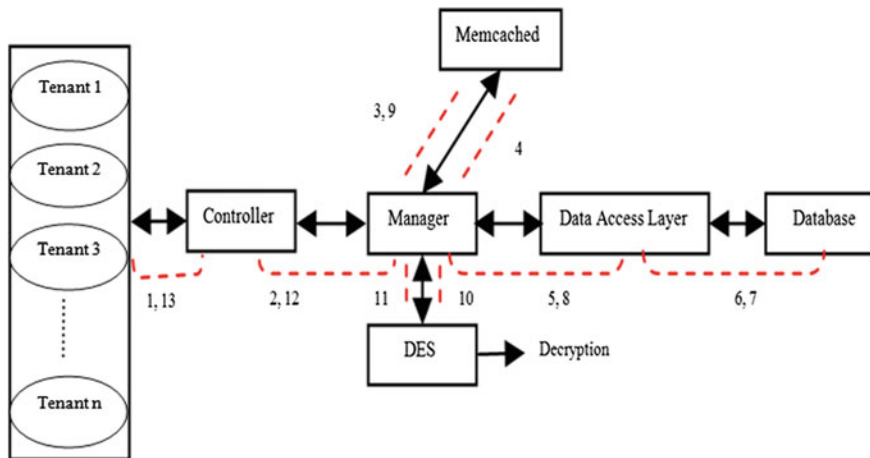


Fig. 5 Data retrieval from Memcached



**Fig. 6** Data retrieval from database if not found in Memcached

to manager (2). Using DES manager will encrypt the data and it is passed to data access layer (3, 4). Data access layer store these details in database securely (6).

Figure 5 illustrates Data Retrieval from Memcached, when user sends a request to controller for retrieving data (1). Controller will map the request to manager (2). The Manager will check and get the data from Memcached (3, 4). After getting data from Memcached, the manager will decrypt it and returned to the user (5, 6).

If data is not found in cache, then it is fetched from database and it will be set to Memcached before decryption. Then manager will decrypt the data and returned to the user. This procedure is shown in Fig. 6.

## 5 Experimental Setup and Results

This section presents the simulation experiments conducted to evaluate performance of the proposed approach. The experiment adopted Memcached to simulate cache service in GAE [3]. Memcached is a powerful distributed cache management system which already integrates powerful caching policies in its architecture and widely applied in industry such as Facebook.

### 5.1 Experimental Design

The experiment environment is constructed by machines with cloud service. Memcached version v 1.4.15 is installed on these machines. The experiment test cases is conducted by using Tomcat version 7.0.39, and available to all the devices



**Table 2** User performance with and without cache

Experiment cycle	Without cache (s)	With cache (s)
Tenant 1	14.61	4.38
Tenant 2	15.433	4.4
Tenant 3	13.65	3.48
Tenant 4	14.2	3.9

**Table 3** Number of miss count for tenants

Experiment cycle	Tenant 1	Tenant 2	Tenant 3
1	1	2	0
2	1	3	1
3	2	3	1
4	3	4	2
5	3	5	2
6	3	6	2

that are connected to this network. The test cases are implemented as a web service. This helps the tenants to access the data through web browser. The business logic using cache service is as follows. The users send requests for data by keys, and the server first looks up to the cache and then the data store otherwise. Decrypted data is stored in cache and database. To investigate the relation between with and without cache, we consider up to four tenants, each of which owns identical amount of users and workloads. The Table 2 shows the response time required for tenants using with and without cache for data retrieval.

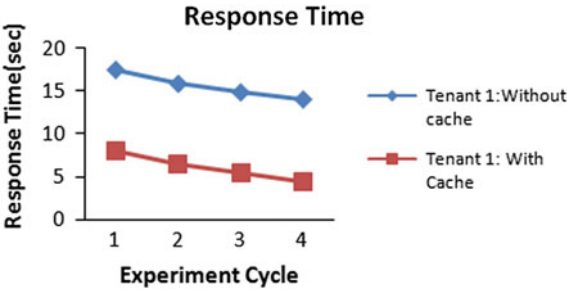
The experiment considered up to 100 users for every tenant. The workload reaching 4000 request per second is considered, which is far heavier than that in the real world condition. Every tenant has its own key. This key is same for all users under one tenant. The proposed method controls the data characteristics by using this pre-generated key.

To provide further evidence, we consider the miscount rate for all tenants in each experiment cycle. Miscount gives number of time the data not present in cache, i.e., miscount increment when data is not present in cache. Hit count gives the number of times data present in cache, i.e. Hit count increments when data present in cache. Table 3 shows the miss count for three tenants in each experiment cycle.

5.2 Performance Calculation

The performance analysis indicates whether the multi-tenant application with cache captures trends in average response time. Figure 7 shows the response time with and without cache for one tenant with one user. The slope of the graph for with cache is lesser compared to without cache. Response time of tenant 1 is decreasing in every cycle. That means tenants can retrieve data quickly after every cycle.

**Fig. 7** Response time with and without cache for one tenant

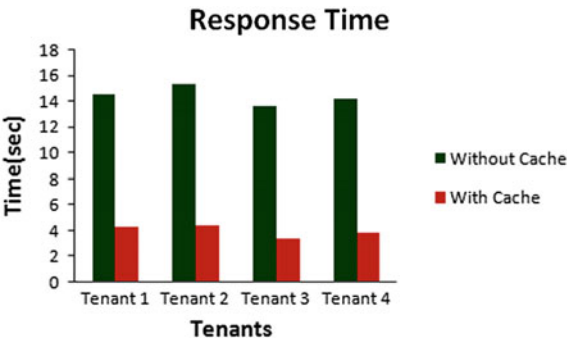


Response time for tenant 1 is higher in without cache and less in with cache experiments. It shows that there is significant improvement in the response time of tenant with cache than tenant without cache, which increases user satisfaction.

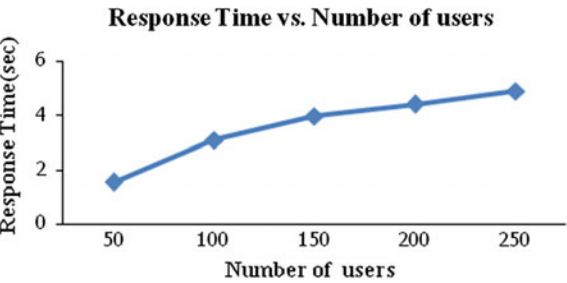
To provide further evidence on the performance of the proposed method, consider the experiment with multiple tenants and their response time for with and without cache. Cloud cache service provides a greater difference in response time as compared with database storage. In the case of without cache the response time is high due to direct fetch from the database. The graphical representation of the comparison is shown in Fig. 8.

Figure 9 shows Response time versus number of users. Number of users within the tenants is incremented to study the relation with response time. Tenant 2 is

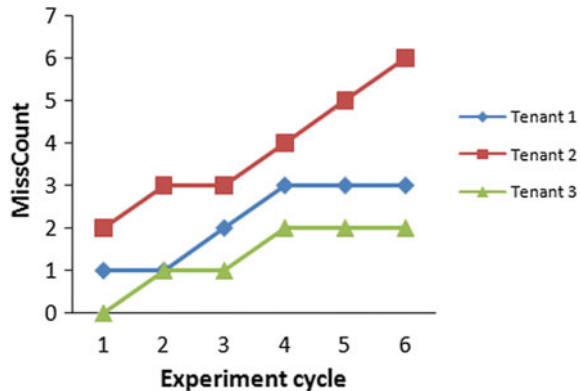
**Fig. 8** Response time with and without cache for multiple tenants



**Fig. 9** Response time versus number of users



**Fig. 10** Miss count versus number of tenants



taken for this. Number of users in tenant 2 is incremented periodically and checks their corresponding response time. Here a high number of users will lead to high response time.

Figure 10 shows the misscount rate for all tenants in each experiment life cycle. Misscount incremented after every experiment cycle, that means, corresponding tenants data is frequently requested. This leads to the caching of tenant data for faster access. Also no changes in misscount rate after a experiment cycle means that, the request for that tenant data is not reached.

## 6 Conclusion

In cloud multi-tenant applications most of the research works focus on VM migrations and load balancing issues. Efficient and secure data storage is an important issue to improve the resource utilization and for better response time. In this work multi-tenant application with cache mechanism is proposed for effective and faster service. Security is another major concern in these cloud application platforms. In order to address the security issues, the proposed cache approach for multi-tenant application employs DES algorithm to avoid improper access of memory by other tenants. This provides an internal security mechanism to tenant data.

In future, this work can be extended to increase the cache performance and cost-effectiveness through cache optimization. Response time can also be reduced through an internal cache mechanism that may provide better performance to tenants.

## References

1. Peter, M., Grance, T.: The NIST definition of cloud computing. NIST special publication 800.145 7, 1–3 (2011)
2. Xuxu, Z., Qingzhong, L., Lanju, K.: A data storage architecture supporting multi-level customization for SaaS. In: IEEE 7th International Conference on Web Information Systems and Applications (WISA), pp. 106–109 (2010)
3. Dormando.: Memcached. In: Internet. [www.memcached.org](http://www.memcached.org). Accessed on 6 March 2015
4. Pierre, Guillaume, Tanenbaum, A.S.: Differentiated strategies for replicating web documents. Elsevier. *Comput. Commun.* **24**(2), 232–240 (2001)
5. Guillaume, P., Van Steen, M., Tanenbaum, A.S.: Dynamically selecting optimal distribution strategies for web documents. *IEEE Trans. Comput.* **51**(6), 637–651 (2002)
6. Bezemer, Z.: Multi-tenant SaaS applications: maintenance dream or nightmare?. In: ACM 4th Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), pp. 88–92 (2010)
7. Wu, L., Garg, S.K., Buyya, R.: SLA-based resource allocation for software as a service provider in cloud computing environments. In: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 195–204 (2011)
8. Nandi, B.B.: Dynamic SLA based elastic cloud service management: a SaaS perspective. In: IFIP/IEEE International Symposium on Integrated Network Management (IM2013), pp. 60–67 (2013)
9. Xiulei, Q.: On-line cache strategy reconfiguration for elastic caching platform: a machine learning approach. In: IEEE 35th Annual Conference on Computer Software and Applications Conference (COMPSAC), pp. 523–534 (2011)
10. Swaminathan, S., Pierre, G., van Steen, M.: A case for dynamic selection of replication and caching strategies, Web content caching and distribution, pp. 275–282. Springer, Netherlands (2004)
11. Subramanian, R., Yannis, S., Loh, G.H.: Adaptive caches: effective shaping of cache behavior to workloads. In: 39th IEEE/ACM International Symposium on Microarchitecture, pp. 385–396 (2006)
12. Prabhakar, R.: Adaptive QoS decomposition and control for storage cache management in multi-server environments. In: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 402–413 (2011)
13. Yao, J.C., Shi-Dong, Z., Yu-Liang, S., Qing-Zhong, L.: Multi-tenant database memory management mechanism based on chunk folding. *Chin. J. Comput.* **34**(12), 2320–2331 (2011)
14. Ang, G., Dejun M., Yansu H.: A QoS control approach in differentiated web caching service. *J. Netw.* **6.1**, 62–70 (2011)
15. Jose, M., Calero, A., Edwards, N., Kirschnick, J., Wilcock, L., Wray, M.: Towards a multi-tenancy authorization system for cloud services. *IEEE Secur. Priv.* **8**(6), 48–55 (2010)
16. Almorsy, M., Grundy, J., Ibrahim, A.S.: Collaboration-Based cloud computing security management framework. In: IEEE International Conference on Cloud Computing (CLOUD), pp. 364–371 (2011)
17. Mandy, W., Zimmermann, W.: Controlling data-flow in the cloud. In: 3rd International Conference on Cloud Computing, GRIDs, and Virtualization, pp. 24–29 (2012)
18. Liang, Y., Hao, Z., Yu, N., Liu, B.: RandTest: towards more secure and reliable dataflow processing in cloud computing. In: International Conference on Cloud and Service Computing, pp. 180–184 (2011)
19. Almorsy, M., Grundy, J., Ibrahim, A.S.: TOSSMA: a tenant-oriented SaaS security management architecture. In: IEEE 5th International Conference on Cloud Computing (CLOUD), pp. 981–989 (2012)

20. Saleh, E., Takouna, I., Meinel, C.: SignedQuery: protecting users data in multi tenant SaaS environments. In: IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 213–218 (2013)
21. HMAC RFC 2104.: <http://tools.ietf.org/html/rfc2104>. Accessed on April 2015
22. Lam, H.Y., Zhao, S., Xi, K., Chao, H.J.: Hybrid security architecture for data center networks. In: IEEE International Conference on Communications (ICC), pp. 2939–2944 (2012)

Innovations in Bio-Inspired Computing and Applications  
Proceedings of the 6th International Conference on  
Innovations in Bio-Inspired Computing and Applications  
(IBICA 2015) held in Kochi, India during December  
16-18, 2015

Snášel, V.; Abraham, A.; Krömer, P.; Pant, M.; Muda, A.K.  
(Eds.)

2016, XIII, 588 p. 199 illus., 75 illus. in color., Softcover  
ISBN: 978-3-319-28030-1