Big Data Research 24 (2021) 100187

Contents lists available at ScienceDirect

Big Data Research

www.elsevier.com/locate/bdr

SA^2-*MCD*: Secured Architecture for Allocation of Virtual Machine in Multitenant Cloud Databases

Arun Kumar Yadav [a], Rajendra Kumar Bharti [b], Ram Shringar Raw [c,*]

[a] *Uttarakhand Technical University, Dehradun, UK, India*
[b] *Bipin Tripathi Kumaon Institute of Technology, Dwarahat, UK, India*
[c] *Ambedkar Institute of Advanced Communication Technologies and Research, Delhi, India*

A R T I C L E   I N F O

Article history:
Received 29 July 2020
Received in revised form 19 November 2020
Accepted 23 December 2020
Available online 8 January 2021

Keywords:
SA^2-*MCD*
Logical Units (*LUs*)
Zoning
RAW Device Mapping (*RDM*)
Node Port ID Virtualization (*NPIV*)

A B S T R A C T

Cloud computing is one of the most demanding technology in today's era. At one side, where cloud computing is offering unlimited amount of IT resources to provide exceptional performance in computing but on the other side, it is facing the security issues majorly with public cloud for multitenant cloud environment. Due to the security issues with public cloud, most of the government and private organizations are not moving their private and sensitive data over the public cloud and compromising with the limited IT resources available to them and limited performance from available resources. A solution to secure private space over the public cloud will resolve the aforesaid issues. This paper presents *SA*^{2}-*MCD*: Secured Architecture for Allocation of Virtual Machine in Multitenant Cloud Databases. This proposed architecture is used for public cloud which provides the private space over multitenant public cloud environment. Moreover, *SA*^{2}-*MCD* is used for efficient distribution of transactions in distributed databases over the public cloud to offer the secured and remarkable performance. It is useful for both software and hardware level security. Random encryption selection method (*RESM*) for software level security and *N_port* zoning, *F_port* zoning, logical unit number (*LUN*) masking, *node_port_ID* virtualization (*NPIV*), raw device mapping (*RDM*) have been used for hardware level security in *SA*^{2}-*MCD* architecture. The proposed architecture has been evaluated mathematically and simulated through CloudSim 3.0.3 tool. The experimental results reveal that the proposed architecture *SA*^{2}-*MCD* gives the better performance as compared to the existing architectures such as *T-MMORRS*, *MGA*, *DPRA* and *PRP*.

1. Introduction

One of the most referred way to illustrate cloud computing as a technology which offers unlimited on-demand IT resources to third party over the internet with minimum mediation of service providers [1]. Cloud computing uses the virtualization technique to provide the IT resources to users based on demand, that are actually available at data centre. The third-party users use applications programs to access IT resources over the internet [2,3]. Cloud computing includes the concepts of utility computing, grid computing, distributed computing, cluster computing and virtualization. Cloud computing also refers to the extension of aforesaid computing. Virtualization technique of cloud computing used at data centres to ensure the provisioning of IT resources, workload distribution to servers, utilization of resources and monitoring. Cloud service providers virtualize the resources at data centres and provide to third party over the internet on pay-per-use basis [4–6]. Various cloud service providers are providing service to commercial businesses to make management and maintenance process easy [7,8].

Cloud computing offering various other services such as workload management, computing resources, resource backup, hardware and software services to third party. The third-party users are charged by cloud service providers as per service level agreement (SLA) for using cloud services. Cloud computing is offering various services such as software-as-a-service (SaaS), platform-as-a-service (PaaS), infrastructure-as-a-service (IaaS) and moreover everything-as-a-service (XaaS). Infrastructure-as-a-service (IaaS) offered by the cloud computing is facing various issues such as security and privacy of hosted cloud databases, performance by efficient workload management, fault tolerance and optimized power consumption [9–11]. Cloud computing use a model to optimize the utilization of available IT infrastructure resources to offer the better services. Fig. 1 shows the detailed description of cloud computing technology.

* Corresponding author.
E-mail addresses: arun26977.utu@gmail.com (A.K. Yadav), rajendramail1980@gmail.com (R.K. Bharti), rsrao08@yahoo.in (R.S. Raw).

https://doi.org/10.1016/j.bdr.2021.100187
2214-5796/© 2021 Elsevier Inc. All rights reserved.

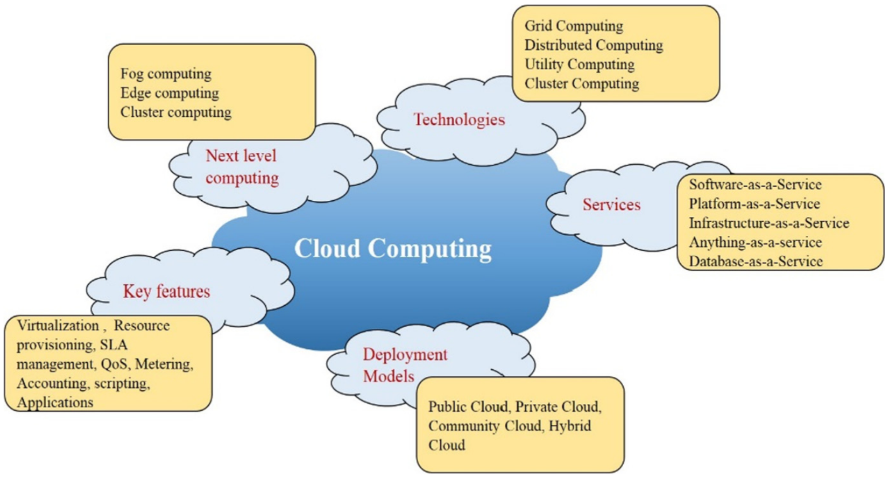

Fig. 1. Detailed view of cloud computing.

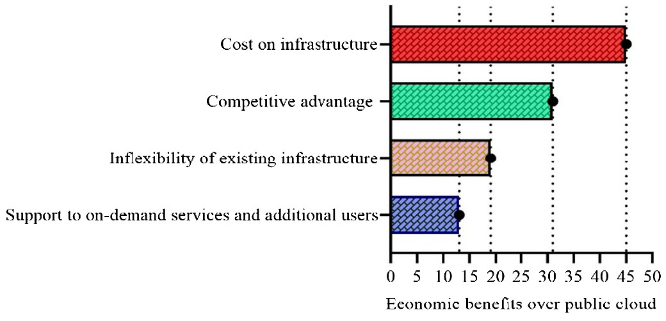

Fig. 2. Benefits on moving to public cloud.

Optimize utilization of resources can be achieving by efficient allocation of requested resources. Cloud service providers reduce the capital expenses of business organizations by provisioning of on-demand resources and unpredictable demand of IT resources. An efficient virtualization model for infrastructure resources, may reduce the expenses of business organizations on resource utilization and can offer the improved services. For controlling the organization expenses on IT resources, three options are available: develop a private cloud, move the database or other workloads on public cloud or go for the hybrid cloud which offers the hybrid services [12]. Shifting over to public cloud for the IT resources, is one of the economic option based on benefits and cost ration model [13]. The major benefits to move over the public cloud are detailed in Fig. 2 [14].

Most of the organizations are interested to move databases from own private cloud to public cloud to get the economic benefits and to experience the high-end performance. Once the organizations decided to move over the public cloud, some of the facts must be verified such as economic benefits and the tools and technologies provided by the cloud providers to support the private databases. As every business database is using some complicated logics to offers specific database services. Therefore, once the organization is avid to move over the public cloud, it must be find out that database should be deployed on public cloud without the modification and without affecting the services. All the necessary hardware and software requirement for database must be available at public cloud data centre to offer the high performance and uninterrupted services.

Organizations must analyse the services offered by the service provider such as hardware support, software to ensure better services, on-demand hardware support etc. or consult the business experts before moving over the public environment. There are very fewer service providers that offering infrastructure-as-a-service (IaaS) and most of the providers are borrowing the infrastructure services and offering to their customers [15,16]. Analysis of service providers are very critical, only business experts can analyse and can suggest the best providers for moving the database. Before giving any decision, business experts need sufficient time to understand the business requirements and services to decide whether the technologies and infrastructure of providers will be the best option to the client or not. For handling the user database over the public cloud environment, one of the dedicated service which is database-as-a-service (DBaaS) must be offered by the service providers. *DBaaS* ensuring privacy, database backup, performance, controlled access on lower cost to business database users.

In this paper, we have proposed a secured architecture that is SA^2-MCD: Secured Architecture for Allocation of virtual machine in Multitenant Cloud Databases. This architecture is ensuring the secure access of cloud databases in multitenant environment by offering the hardware level of security. SA^2-MCD ensure the authenticity of users and its submitted transactions and also secure allocation of virtual machines to ESX_s servers (elastic sky X-slave servers) available at front-end of architecture. We have evaluated the performance of proposed architecture mathematically, simulated the work and compared the results with existing architectures.

Rest of this paper is organized as follows: Section 2 presents the motivation, background and related works. In Section 3, we have presented the system model and problem formulation about the proposed work. Section 4 presents the explanation of proposed SA^2-MCD architecture and its components. Secure allocation method of proposed architecture and related algorithms has been presented in section 5. In section 6, we have presented the analytical evaluation of the proposed architecture. Section 7 and Section 8 presents the experimental setup for the implementation work and simulation results and performance analysis. Finally, Section 9 concludes the works and gives the future exposure of the proposed work.

2. Motivation, background and related works

Cloud database service providers such as Amazon and Microsoft offered database services to commercial users with limited features. Offered services were based on virtual machines (*VMs*), offering limited scalability on databases, affected with data privacy issues and processing issue with encrypted data provided by the public cloud Service providers for the migration of databases in multitenant environment, where number of databases are available in isolation and accessed by the clients [17–20]. Database scalability is one of the most vital feature of database system and most demanding research domain to work. In cloud database commercial applications, database scalability playing very important role. For the processing of encrypted data, a theoretical solution has been proposed in [21,22]. Approach is very much expensive in applying on real world data processing. A burst-aware approach has been proposed to ensure the efficient migration of virtual machines to balance the workload and to improve the performance of cloud based services [23,24].

The approach increases the performance of cloud environment by efficient management of workloads but energy consumptions has not been controlled as required which limits the overall performance of algorithms. A comparative study on load balancing and client-provider interoperability has been presented in [25,26]. The study has covered the current trends and neglected domains of cloud interoperability. Study mainly focuses on infrastructure-as-a-service for data-application interoperability. In present scenarios, consumers are submitting large number of tasks for the execution over the cloud, and the service providers must have to ensure the execution as per the service level agreement (SLA). Service provider can use the efficient load balancing model to manage the resources as per the agreement [27,28].

Cloud users are requesting virtual infrastructures from the service providers for their applications and these requests may include any types of resource such as compute, storage, network or support services. Cloud service providers are facing the challenges to provision the demand of cloud users. A multi-criteria allocation and migration model has been proposed for selecting the right providers and for migrating the virtual infrastructures [29,30]. Cloud users are also requesting resources from cloud environment for their applications without owning the cloud infrastructure and any time they may request for additional cloud resources and service providers must have to fulfil the demand of users. To manage all these activities, resource scheduling algorithm presented in [31–34] which is scheduling the resources at minimum response time, giving better throughput and maximizes the resource utilization and [35] discusses the approach for dynamic allocation of client demanded resources with economic benefits to service providers. Authors have presented security schemes in [36–38], for data security, privacy preservation and for ensuring the security during virtual machines migrations.

T-MMORRS [39] presented approach offering the acceptable performance over the large workload, scheduling approach is efficient and ensuring the reduced energy consumption. Implementation of approach is complex and no security features discussed. *MGA* [40] approach providing enhanced CPU utilization, optimization of memory and low energy consumption but limitation on predicting accuracy and complexity. Dynamic resource allocation approach *DPRA* [41] offering efficient migration of *VM* at low power consumption but needs to improve the performance of approach over network of different capabilities. Proactive workload prediction approach *PRP* [42] has been presented and providing acceptable performance on limited number of requests but limited performance on large number of requests. Architecture and methods for CC-V contributed enhanced network mobility and optimization of cloud resources [43,44]. Presented architectures needed to upgrade for diffusion of data, data routing and data migration over the cloud.

Various approaches have been proposed by the authors for efficiently managing and securing the cloud databases over the multitenant public cloud environment but all the proposed approaches are limited on various features. Summarized analysis of approaches proposed by various authors which motivated to work in this domain and to find the best solutions has been presented in Table 1. Analysis of proposed approaches has been concluded in terms of strength and weakness of approaches.

3. System model and problem formulation

Now a day's cloud-based service provider provides IoT based IT services to their users. These services can be enhanced using multitenant architectures. Multitenancy has been described as a security issue and is an effective and an important part of cloud computing. It is a significant factor of cloud security issues that require a feasible solution from the different cloud computing services. Therefore, a multitenant cloud database allows multiple people to securely and simultaneously access the same hardware and software services available in different locations. Our proposed SA^2-*MCD* architecture, ensuring the secured access of cloud databases over the multitenant public cloud environment. SA^2-*MCD* architecture provides application level security at the front-end as well as hardware level security at the back-end to ensure the unbreakable security from malicious users.

SA^2-*MCD* architecture uses *DBaaS* service at the database server available at back-end. *DBaaS* service providing the partitioning of database as per user request, replication of database for the availability of data in the case of failures and for managing the workloads onto *VMs* by the ESX_m servers. For the database security, SA^2-*MCD* architecture ensures the encryption of data before storing onto back-end database servers and for data privacy at client machine *CryptDB* driver has been used to provide the invulnerable security. Fig. 3 describes the system model of SA^2-*MCD* proposed architecture including the techniques used to make the SA^2-*MCD* architecture invulnerable and methods used for secure allocation of VMs to front-end servers.

As shown in the figure, system model discusses the two different algorithms for virtual machines allocations as SA^2-*MCD* Algorithm 1 and Algorithm 2. Algorithm 1 is describing the procedure to test the client's transaction authenticity and then allocation of virtual machine to ESX_m servers at front-end clusters of servers in SA^2-*MCD* architecture. Algorithm 2 is showing the procedure for allocation of virtual machine by ESX_m server to ESX_s servers after selecting the one of the suitable ESX_s servers. These algorithms can be simplified by data flow diagram. All the components of the system model have been explained in the next sections.

Moreover, cloud computing services rely on a secure architecture to allocate the virtual machines for multitenant cloud databases. Further, in this paper, we have evaluated the performance of proposed SA^2-*MCD* architecture and allocation method mathematically for public cloud environment. The main aim of the proposed architecture which spans along with all parts is security. The security requirements for cloud databases include aspects such as cloud data integrity and solution for all the above-mentioned issues over the multitenant public cloud environment. We have evaluated the performance of the proposed architecture and allocation methods on effective allocation time (EA_T), power consumptions (P_C), resource utilization (R_U) and execution cost (E_C) parameters. The proposed architecture has been implemented and simulated through CloudSim 3.0.3 tool.

4. *SA2-MCD*: proposed architecture

Section 2 discusses the approaches proposed by various authors for effectively managing the databases over the multitenant public cloud. All the proposed approaches are limited to be used in specific situations only. Approaches are not ensuring the features such as scaling, privacy, encrypted data processing and workload sharing in heterogeneous multitenant environment. Although multitenant system for cloud computing is more complicated but since it is at heart of cloud computing, therefore it supports thousands

Table 1
Summary of reviewed articles: approach, strength and weaknesses.

Article	Algorithm	Strength	Weaknesses
Automatic virtual machine configuration for database workloads [17]	Virtualization design advisor	Virtual machines are configured automatically to share the database work-load, performance	Security over Virtual machines, user authentication, more refinement on cost model is needed.
Dynamic Resource Allocation for Database Servers Running on Virtual Storage [18]	Dynamic multi resource allocator	Optimized performance on workload, optimal partitioning	Cost of implementation, security
Non-interactive verifiable computing: Outsourcing computation to untrusted workers [22]	Verifiable computational scheme	User data privacy, identification of malicious users	Ensuring input privacy limit the performance, privacy ensured at from-end level
Burst-aware virtual machine migration for improving performance in the cloud [23]	REDMT, MPP	Resource optimization,	Energy consumption, workload on physical machines
How to place your apps in the fog: State of the art and open challenges [27]	FAPP	Complexity of algorithm, static conditions	Decentralization of algorithms, performance
Cloud broker proposal based on multi-criteria decision-making and virtual infrastructure migration [29]	VIMAM	Effective migration of VM, performance, response time	Management of failures during migration, fault tolerant methods to select and migrate the VM.
Threshold based multi-objective memetic optimized round Robin scheduling for resource efficient load balancing in cloud [39]	T-MMORRS	Performance over large work load, task scheduling efficiency, energy consumption, scheduling time	Complexity of approach, security
Dynamic resource prediction and allocation for cloud data centre using the multi-objective genetic algorithm [40]	MGA	Improved CPU utilization, low energy consumption, optimized use of memory	Prediction accuracy, approach complexity
DPRA: dynamic power-saving resource allocation for cloud data centre using particle swarm optimization [41]	DPRA	Energy consumption, migration of virtual machine	Network capabilities to improve the performance, security concern
A proactive approach for resource provisioning in Cloud Computing [42]	PRP	Workload prediction and provisioning of resources, time efficient with limited requests	Increased waiting time in case of large number of requests, approach complexity

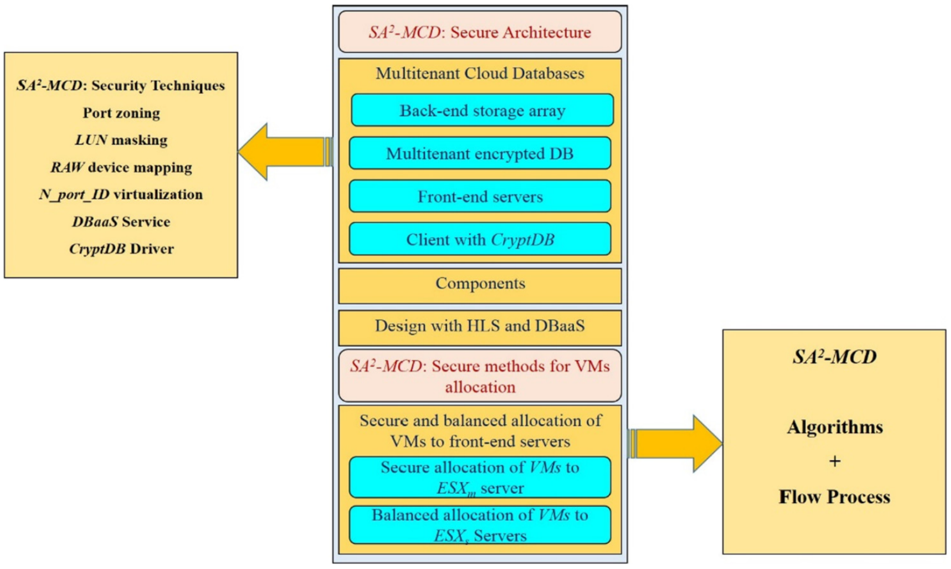
SA²-MCD: Security Techniques
Port zoning
LUN masking
RAW device mapping
N_port_ID virtualization
DBaaS Service
CryptDB Driver
SA²-MCD: Secure Architecture
Multitenant Cloud Databases
Back-end storage array
Multitenant encrypted DB
Front-end servers
Client with CryptDB
Components
Design with HLS and DBaaS
SA²-MCD: Secure methods for VMs allocation
Secure and balanced allocation of VMs to front-end servers
Secure allocation of VMs to ESX_m server
Balanced allocation of VMs to ESX_s Servers
SA²-MCD
Algorithms
+
Flow Process

Fig. 3. System model of SA^2-*MCD* proposed architecture.

of users for businesses and provides secure and concurrent access of both hardware and software services that are available in different remote locations and public cloud environments. In this section we have explained the proposed architecture for *VMs* allocation in multitenant cloud databases.

4.1. Architecture overview

SA^2-*MCD* secured architecture presented in this section ensuring the security of cloud databases and solution for all the above-mentioned issues over the multitenant public cloud environment. We have used port zoning, *LUN* masking, RAW device mapping and Node port ID virtualization (*NPIV*) to ensure the secured access of databases over the public cloud and relational cloud [45] to ensure the solution of aforesaid issues. SA^2-*MCD* architecture has been shown in Fig. 4.

As shown in SA^2-*MCD* architecture, it includes the storage array at the back-end site with cloud database in encrypted format over the multitenant public cloud environment. At the front-end site, high end *ESX* servers will be available in clusters. Where clusters of servers will include the master *ESX* servers which will perform the secure allocation of virtual machines to the slave *ESX* servers

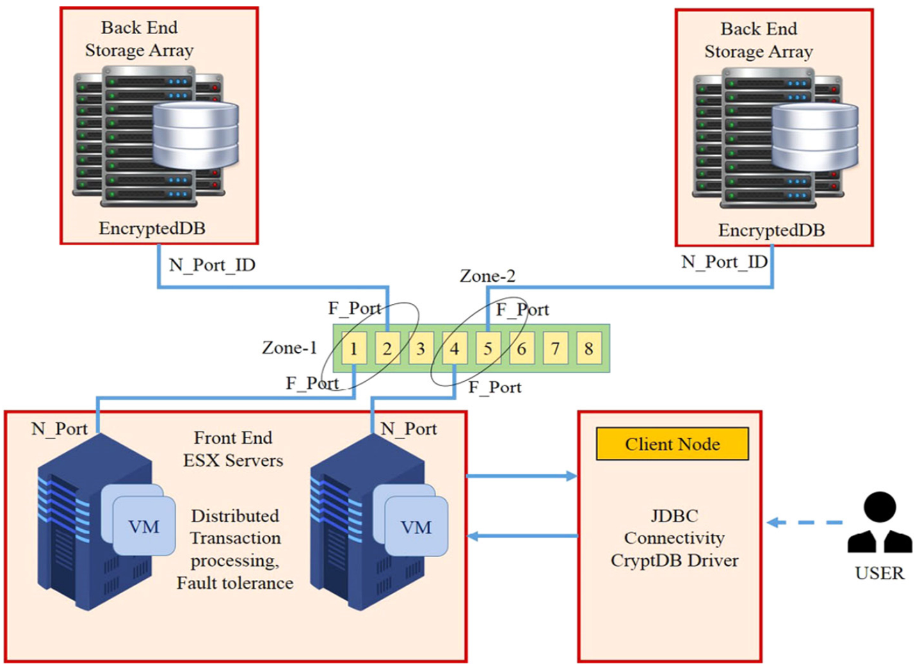

Fig. 4. *SA*^{2}*-MCD* secured architecture for multitenant cloud databases.

for the execution. Solutions proposed by authors in related section, discuss only the software based solution or front-end security solutions which are not claiming much more security over the multitenant public cloud environment. *SA*^{2}*-MCD* architecture offering the front-end and hardware based back-end security in public cloud environment. Component of architecture and techniques for providing hardware based front-end and back-end security discussed in subsequent sections.

4.2. *Components of architecture*

4.2.1. *Servers*. Proposed architecture uses the *ESX* servers to support *RAW* device mapping and controlled access of disks in virtual machines. These special servers are allocating the virtual machines and ensuring the workloads on servers. *ESX* server containing the virtual machine file system for providing the controlled access of virtual machines by various hosts.

4.2.2. *Storage unit*. Storage units in architecture refer to the collection of high-end storage disks to provide the huge storage capacity.

4.2.3 *Logical units*. Logical units refer to logical volumes at storage unit with unique ID.

4.2.4. *Hardware level security of database over public cloud*. To secure the database available in logical units of storage systems at public cloud, following techniques have been used in proposed architecture.

- *Port zoning:* Port zoning offering the mapping between the ports at *ESX* server, network switches and storage units. In architecture, ports mapping has been performed between the *N_port* at *ESX* server, *F_port* at switches and *N_port* at storage unit and each set of mapping has been assigned a unique zone.
- *LUN mapping with ports: LUN* mapping with ports known as *LUN* masking. In *LUN* mapping, each logical unit at storage system has been assigned a unique number. Logical unit with unique number refers to logical unit number (*LUN*). Cloud databases in proposed architecture are available in logical units at storage system of public cloud. To ensure the controlled access of cloud databases, each *LUN* has been masked to dedicated *N_port* at storage unit. Only request which belong to same zone of dedicated *N_port* onto which *LUN* has been mapped, will be able to access the database available at logical unit.
- *Cloud database sharing with virtual servers:* Using the logical unit mapping with dedicated port at storage unit, cloud database will be accessible to servers belonging to same zone only. So, to make the cloud database accessible to various virtual servers, *node_port_ID* virtualization has been used to share the dedicated *N_port* by the various servers.
- *Controlled access of cloud databases from virtual machines* (*VMs*): Cloud databases will be provided to users through allocation of virtual machines to *ESX* servers where, user requested database will be provided on virtual machines. With these allocations, user may have different privileges on virtual machines, user may install additional software and may scan the various logical units at storage units. So, to ensure the isolated access of cloud databases on virtual machines, raw device mapping (RDM) has been used to map the logical unit to dedicated virtual machine. With RDM mapping, logical unit at storage array will behave like a local memory or storage device to virtual machine and users will not be able to scan the storage units for unauthorized access of logical units.

4.2.5. *Architecture with database-as-a-service* (*DBaaS*). Various cloud database service providers such as Microsoft and Amazon were offering SQL azure and RDS respectively for database services but were not ensuring the database privacy, database scalability and efficient performance in multitenancy. Any organization or individual user may decide to outsource the database services from service providers, if the above mentioned issues will be resolved. So, to ensure the privacy, database scalability and efficient multitenancy over public cloud, we have used *DBaaS* service in proposed architecture.

Organizations and individuals will be benefited by outsourcing the *DBaaS* services from the services providers in terms of economic benefits. Users will have to pay only as per the utilization

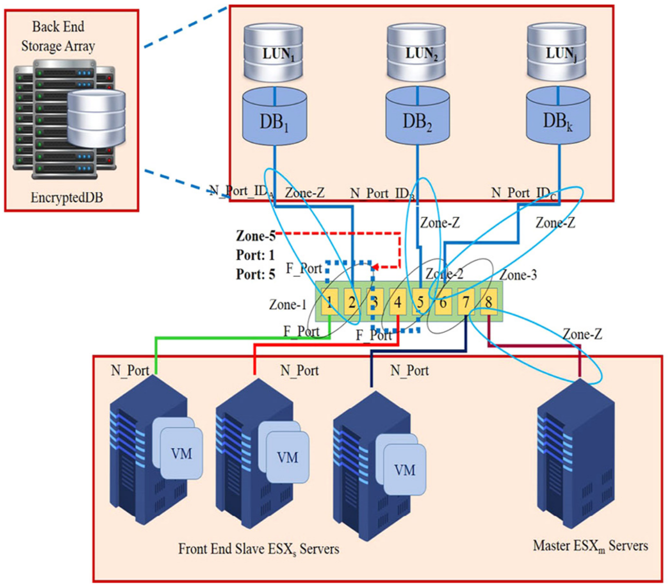

Fig. 5. SA^2-*MCD* with port zoning, *RDM* and *LUN* masking between front-end servers and back-end storage array.

of database. With economic benefits, *DBaaS* service will ensure the efficient distribution of workloads, make the system ready for future loads and ensure the database privacy.

4.3. Architecture design with hardware level security and DBaaS

DBMS engine has been using for query processing at back-end nodes. Database servers are available on back-end nodes in proposed SA^2-*MCD* architecture. One database server available at one back-end node. In a multitenant environment, more than one database may be load at back-end node by the tenants. *DBaaS* service at public cloud ensures the isolation among various databases of tenants. Client access the cloud database using the *JDBC* connectivity between the client and front-end *ESX* server. When client request reaches at front-end *ESX* server, *ESX* server checks the zone from where request has been submitted and the details of requested database. After verifying the hardware level security checks, server analyzes the client request (SQL statements) and uses the metadata details to decide the execution of transactions. A virtual machine containing the requested database mapped to front-end *ESX* server. *ESX* server ensures the distribution of transactions, distributed execution and fault tolerance. Server maintains the overall performance of system by ensuring the efficient execution of distributed transaction among virtual machines.

ESX server monitors the workloads and requested databases on virtual machines by distributed transactions. *DBaaS* using this monitoring information to partition the database, replicate of database for availability and for managing the workloads on *VMs*. To ensure the database level security, data is encrypted at database server before storing to back-end nodes. *CryptDB* driver used at the client machine to maintain the data privacy during the communication to *ESX* server. Back-end storage array shared by a set of *ESX* servers in a cluster to balance the workloads between servers and between virtual machines allocated to servers as shown in Fig. 5. Each cluster of *ESX* servers has one *ESX* server as master server whereas others as slave servers.

Fig. 5 is describing the port zoning, *LUN* masking and *RAW* device mapping used in SA^2-*MCD* architecture to provide the secure and controlled access of *LUN* at ESX_s server in allocated virtual machines. Front-end ESX_s servers will be able to access the *LUN* from back-end storage array only if they belong to same zone. DB_k in LUN_j can be accessed on number of ESX_s servers if the LUN_j has mapped to various ESX_s servers using virtualized *N_port_ID* approach. *N_port_ID* virtualization share the *N_port* at storage among various servers.

4.4. Random Encryption Selection Method (RESM): method for privacy preservation

End users are mainly concerned with data privacy in multi-tenant public cloud environment, so proposed SA^2-*MCD* architecture ensuring the data privacy preservation at the client end and back-end. Client accesses the DBaaS service of SA^2-*MCD* architecture using the *JDBC* connectivity and *CryptDB* driver at client node encrypting the data before sending to front-end ESX_m sever for storing on back-end database servers. All the database queries submitted by the users executed on encrypted database. Database engine has been used at back-end database server for distributed query processing and for efficient database management.

For preserving the privacy, data is first encrypted and then store onto the back-end database server using *RESM*. *RESM* method uses layered approach where different types of data will be encrypted by different layer of encryption. Selection of encryption layer for data encryption will depend on queries executed by the client. As different types of data may be retrieved by the query from database executed by the client. The retrieved data may result of *equi join, inequi join, aggregate values* (*min, max, avg, sum, count*), *selection, update, delete* and *alter* queries. For each type of queries data, different layer of encryption will be used. Flow process of *RESM* method for data privacy preservation is demonstrated in Fig. 6.

Client uses access applications and *JDBC* connectivity to connect to front-end ESX_m server for retrieving the queried data from back-end database server. *CryptDB* driver at client ensures the data privacy. Front-end ESX_m server analyzes the queries from client for distributed transaction processing, plan for fault tolerance, selection of ESX_s servers and load balancing on ESX_s servers and back-end database servers. Information analyzed by ESX_m server also used by DBaaS service at back-end nodes for database partitioning and providing the partitioned and queried database on virtual machines at ESX_s servers at front-end. Furthermore, DBaaS service may partition the queried database again, if required for performance and for the execution of client queries on encrypted database.

Fig. 6 is explaining the flow process of layered based *RESM* method used by *DBaaS* service of DBMS engine at back-end database servers for ensuring the data privacy. As the client queries are executing on encrypted data and retrieved data are encrypted by layered based *RESM* method based on queried data before sending to client. *CryptDB* driver at client will coordinate with ESX_m server for decrypting the queried data. Each type of queried data will be decrypted by different decryption keys.

5. Proposed secure allocation method for SA^2-MCD

This section presented the method for secure allocation of virtual machines to slave ESX_s servers through master ESX_m servers available at front-end in SA^2-*MCD* proposed architecture. In the ensuing sections, system model and algorithms have been presented and then performance of algorithms evaluated in the next section. Various parameters used in this paper to discusses the proposed SA^2-*MCD* algorithms and evaluation model have been shown in Table 2.

5.1. Secure allocation of virtual machines to ESX_m Server

Algorithm 1 presented in this section is describing the procedure for secure allocation of virtual machines to master ESX_m

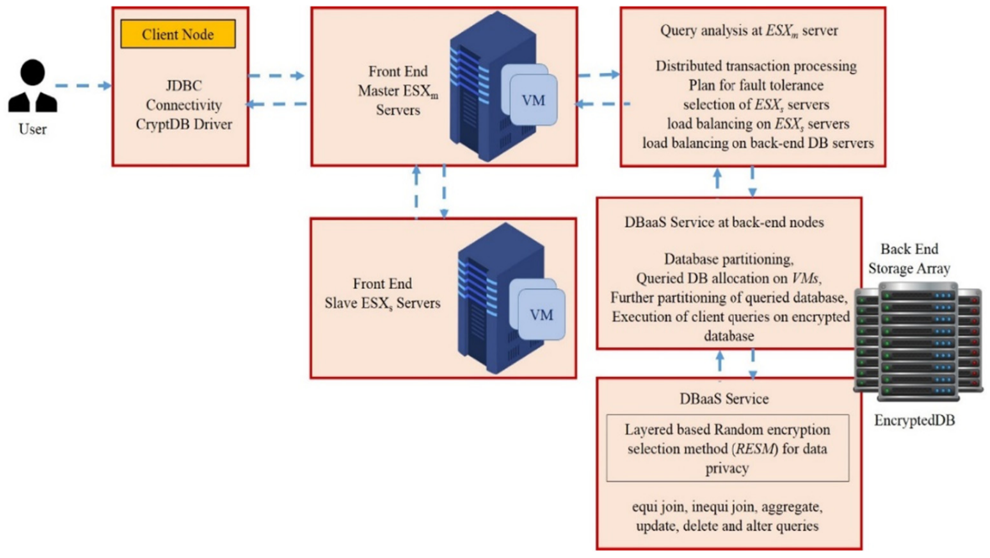
User
Client Node
JDBC Connectivity CryptDB Driver
Front End Master ESX$_m$ Servers
VM
Query analysis at ESX$_m$ server
Distributed transaction processing
Plan for fault tolerance
selection of ESX$_s$ servers
load balancing on ESX$_s$ servers
load balancing on back-end DB servers
Front End Slave ESX$_s$ Servers
VM
DBaaS Service at back-end nodes
Database partitioning,
Queried DB allocation on VMs,
Further partitioning of queried database,
Execution of client queries on encrypted database
Back End Storage Array
EncryptedDB
DBaaS Service
Layered based Random encryption selection method (RESM) for data privacy
equi join, inequi join, aggregate, update, delete and alter queries

Fig. 6. Layered based *RESM* method for privacy preservation.

Table 2
List of parameters and abbreviations.

Symbol	Descriptions
T_i	Client transactions at ESX_m server
DB_k	Requested database
LUN	Logical units containing database DB_k at back-end storage array
ESX_m	Master server at front-end
ESX_s	Slave server at front-end
N_port_ID	Node port ID at back-end storage array
N_port	Node port at ESX_m and ESX_s servers
F_port	Fabric port at switch
VM_i	Virtual Machines
$R_j(T)$	Total resources available
$R_j(A)$	Free or unused resources
$R_j(U)$	Allocated or used resources
$R(V)$	Resources needed by virtual machine
$RM_j(A)$	Remaining resources after allocation of virtual machine
δ	Effective allocation time of single virtual machine
N	Number of virtual machines
η	Number of user tasks
λ	Power consumed by the single virtual machine
μ	Total resources available
π	Free resources
ρ	Data transferred to a port
σ	Data price to port
ψ	Data transferred from port
α	Data price from port
β	Total storage amount
γ	Storage price
ε	Size of virtual machine
χ	Usage time of virtual machine
$\dagger$	Virtual machine price
EA_T	Execution allocation time
P_C	Power consumptions
R_U	Resource utilizations
E_C	Execution cost
N_C	Network cost
S_C	Storage cost
VM_C	Virtual machine cost
S_T	Start time
LUN_{RT}	LUN response time
LUN_{AT}	LUN access time
F_T	Finish time

servers available at front-end clusters. Algorithm ensuring the client authenticity at hardware level. If all the checks at hardware level are satisfied, then only the requested database will be provided to the client through virtual machines on ESX_s server. Parameters used in proposed SA^2-*MCD* algorithms have been shown in Table 2.

***SA*^{2}*-MCD* Algorithm 1** Secure allocation of *VMs* to ESX_m server.

Input: TransactionList(T_i), **DBList**(DB_k), **LUNList**(LUN_j) **Output: VMs allocation to ESX_m Server**
[Assuming that requested DB_k is not available in any VM_i at front-end server]

```
1. [$T_i$ at $ESX_m$ server] $T_i = \{T_1, T_2 \ldots\ldots\ldots\ldots\ldots\ldots T_n\}$
2. $DB_k = \{DB_1, DB_2 \ldots\ldots\ldots\ldots\ldots DB_m\}$
3. [Logical units containing database $DB_k$ at back-end storage array] $LUN = \sum_{j=1}^{x} LUN_j \sum_{k=1}^{y} DB_k$
4. Repeat steps for $T_i$ where $i = 1, 2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots n$
5. [$ESX_m$ server checks the requested $DB_k$ in LUNList.] If not $DB_k$ then
6.     $T_i$ = transaction_cancel ();
7.     neg_ack();
8.     return;
9.   else
10.    forward_N_port_to_F_port$_A$ ();
11. end if
12. forward_F_port$_A$_to_F_port$_B$ ();
13. If zone(F_port$_B$) $\neq$ zone (F_port$_A$) then
14.     $T_i$ = transaction_cancel ();
15.     neg_ack();
16.     goto step 10;
17.   else
18. F_port$_B$_to_N_port_ID$_A$ ();
19. If $DB_k \subset LUN_j$ && map ($DB_k$, N_port_ID$_A$) == true then
20.     create_VM();
21.     allocate ($ESX_m$);
22.   else
23.     $T_i$ = transaction_cancel ();
24.     neg_ack();
25.     return;
26. end if
27. end step 4 loop
```

Flow process of Algorithm 1 for SA^2-*MCD* proposed architecture has been shown in Fig. 7.

Fig. 7 is showing the flow process of Algorithm 1, where algorithm ensuring the hardware level security of logical units in storage array available at back-end of public cloud environment.

5.2. Balanced allocation of virtual machines to ESX_s Servers

SA^2-*MCD* architecture for allocation of virtual machines shown in Fig. 5 clearly showing that front-end servers are organized in cluster of servers. In which, one ESX_m server is acting as master server whereas other ESX_s servers are slave servers, where virtual machines allocated by the ESX_m server and accessed by the requesting client. ESX_m server maintains the details of resources available at ESX_s servers in terms of parameters shown in Table 2.

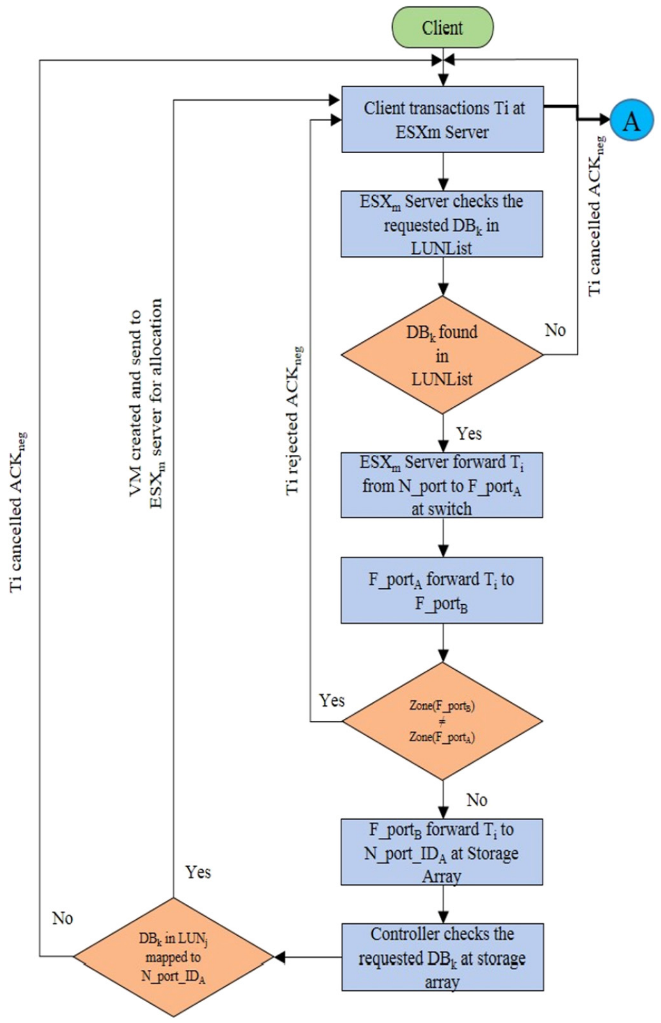
Client
Client transactions Ti at ESXm Server
A
ESXm Server checks the requested DBk in LUNList
DBk found in LUNList
No
Yes
Ti cancelled ACKneg
ESXm Server forward Ti from N_port to F_portA at switch
F_portA forward Ti to F_portB
Zone(F_portB) ≠ Zone(F_portA)
Yes
No
Ti rejected ACKneg
F_portB forward Ti to N_port_IDA at Storage Array
Controller checks the requested DBk at storage array
DBk in LUNj mapped to N_port_IDA
Yes
No
VM created and send to ESXm server for allocation
Ti cancelled ACKneg

Fig. 7. Flow process of Algorithm 1 for SA^2-*MCD*.

$R_j(T)$ at ESX_s server representing the total resources available at slave server and it includes various resources like RAM, CPU and storage at slave server and represented by the Eq. (1).

$$R_j(T) = \sum_{j=1}^{n} \left(R_j(T)\right) \tag{1}$$

Each ESX_s server in front end cluster also maintains the details of free or unused resources $R_j(A)$ and allocated or used resources $R_j(U)$. Each ESX_s server regularly informs to ESX_m server about the unused and used resources. Total resources at ESX_s server can be represented by Eq. (2).

$$R_j(T) = R_j(A) + R_j(U) \tag{2}$$

ESX_m server maintains the list of free or unused resources $R_j(A)$ and allocated or used resources $R_j(U)$. For the allocation of virtual machines to ESX_s slave servers, ESX_m server check $R_j(V) \leq R_j(A)$ in available list and prepare a list of remaining resources after the allocation of virtual machine $RM_j(A)$ at jth ESX_s slave servers. Remaining resources at each ESX_s slave servers after the allocation of virtual machine can be shown by Eq. (3).

$$RM_j(A) = R_j(A) - R_j(V) \tag{3}$$

Flow process of Algorithm 2 has been shown in Fig. 8.

***SA*^{2}*-MCD* Algorithm 2** Secure allocation of *VMs* to ESX_s.

Input: TransactionList(T_i), **DBList**(DB_k), **LUNList**(LUN_j) **Output: VMs allocation to** ESX_s **Server**

```
1.  [$T_i$ at $ESX_m$ server] $T_i = \{T_1, T_2 \ldots\ldots\ldots\ldots\ldots\ldots T_n\}$
2.    Repeat steps for $VM_i$ where $i = 1, 2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots n$
3.      Repeat steps for $j = 1, 2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots n$
4.        status_of_totalres ($R_j(T)$);
5.        status_of_remres ($RM_j(A)$);
6.      end loop
7.      Repeat for $j = 1$ to $n$
8.        If $R_j(V) \leq R_j(A)$ then
9.        $RM_j(A) = R_j(A) - R_j(V)$;
10.       end if
11.     end loop
12. allocation_with_least_resources ($ESX_s$, $RM_j(A)$);
13.   end loop
14. end loop
```

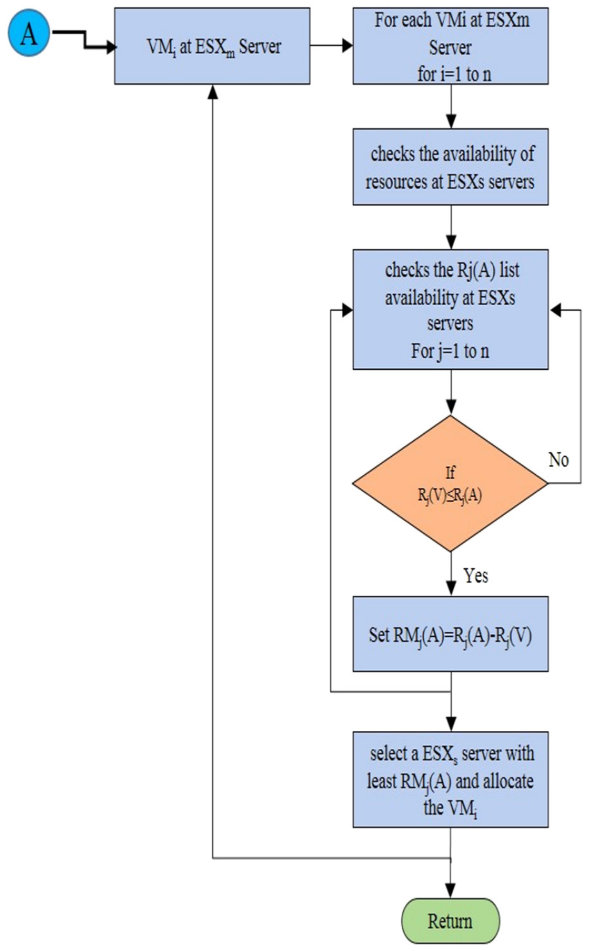
A
VMi at ESXm Server
For each VMi at ESXm Server for i=1 to n
checks the availability of resources at ESXs servers
checks the Rj(A) list availability at ESXs servers For j=1 to n
If Rj(V)≤Rj(A)
No
Yes
Set RMj(A)=Rj(A)-Rj(V)
select a ESXs server with least RMj(A) and allocate the VMi
Return

Fig. 8. Flow process of Algorithm 2 for SA^2-*MCD* architecture.

Flow process of Algorithm 2 shown in Fig. 8 describes the balanced allocation of virtual machines to ESX_s servers at front-end clusters. Algorithm 2 ensuring the low power consumptions and maximizing the resource utilization by effective allocation of virtual machines to ESX_s servers.

6. Analytical evaluation of proposed *SA*^{2}*-MCD* architecture

To evaluate the performance of SA^2-*MCD* architecture, allocation methods and algorithms on various parameters, mathematical models have been presented in this section. For the experimental setup of SA^2-*MCD* architecture and algorithms, CloudSim 3.0.3 toolkit has been used and discussed in Section 7. CloudSim is a cloud based simulator that supports the setup of cloud computing architectures and testing of models and algorithms. In this work, we have implemented and simulated the proposed architecture and its algorithms over some important parameters such as effective allocation time of virtual machines for load balancing (EA_T), power consumptions (P_C) by virtual machines, resource utilization (R_U) at servers and total cost of execution (E_c). Evaluation model presented in ensuing sections describes the performance evaluation of SA^2-*MCD* architecture on above mentioned parameters. Mathematical evaluation models use the various parameters shown in Table 2.

6.1. Effective allocation time (EA_T)

Effective allocation time, EA_T refers to the time taken by the ESX_m server to allocate the virtual machines (VMs) to ESX_s servers. Mathematically, EA_T can be evaluated as given in Eq. (4).

$$EA_T = N * \delta \tag{4}$$

where EA_T refers to the total effective allocation time, N refers to number of virtual machines and δ refers to effective allocation time for the allocation of single virtual machine to available ESX_s servers.

6.2. Power consumptions (P_C)

Power consumption P_C defines the power consumed by number of virtual machines at ESX_s servers to service the user tasks. Power consumption at ESX_s servers includes the power consumed to provide the resources to user tasks at single virtual machine. Power consumption P_C can be evaluated using the Eq. (5).

$$P_C = \eta * \lambda \tag{5}$$

In Eq. (5), η refers to the number of user tasks that are executing on single virtual machine and λ refers to the power consumed by the single virtual machine to service the user tasks. In experimental results, power consumption has been evaluated in joule (J).

6.3. Resource utilization (R_U)

Resource utilization R_U refers to the utilization of resources by the virtual machines at ESX_s servers. Efficient algorithm maximizes the utilization of resources at ESX_s servers instead of using additional server for the execution of virtual machines. Resource utilization at ESX_s server can be represented by given Eq. (6).

$$R_U = \mu - \pi \tag{6}$$

where R_U refers to the utilization of resources, μ refers to the total resources available and π is the free resources at ESX_s server.

6.4. Execution cost (E_C)

Execution cost refers to the network cost, instance cost and storage cost required to allocate number of user tasks to virtual machines at ESX_s servers at front-end and for ensuring the successful execution of tasks. Execution cost (E_C) can be estimated by using the model discussed below:

Network cost (N_C). Network cost can be calculated by estimating the amount of data transferred to a port and transferred from a port. Data price to port and data price from port may depend on service providers. The network cost can be calculated by the following Eq. (7).

$$N_C = \rho \times \sigma + \psi \times \alpha \tag{7}$$

where N_C refers to the network cost, ρ for data transferred to a port, σ for data price to port, ψ for data transferred from port and α for data price from port.

Storage cost (S_C). Storage cost can be calculated by estimating the amount of storage used by the virtual machine at ESX_s Server and storage provided to the users. Storage cost can be calculated by the Eq. (8).

$$S_C = \beta \times \gamma \tag{8}$$

where β refers to the total storage amount and γ refers to the storage price by the service providers.

Virtual machine cost (VM_C). Virtual machine cost depends on number of virtual machines, size of virtual machine, duration of virtual machine used and price. Virtual machine cost can be estimated by the Eq. (9).

$$VM_C = \sum_{i=1}^{n} \varepsilon_i \times \chi \times \dagger \tag{9}$$

where ε refers to size of single virtual machine, χ for usage time of virtual machine and $\dagger$ used to represent virtual machine price.

Therefore, the total execution cost can be determined by taking the summation of network cost, storage cost and virtual machine cost and has been shown in Eq. (10).

$$E_C = N_C + S_C + VM_C \tag{10}$$

The analytical evaluation of proposed architecture, algorithms and mathematical models has been implemented using CloudSim simulator. The experimental setup and simulation results and performance analysis has been presented in Section 7 and Section 8 respectively.

7. Experimental setup

Due to infeasibility of physical infrastructure, proposed SA^2-*MCD* architecture setup, testing of hardware level security, evaluation of performance and testing of algorithms have been implemented using CloudSim 3.0.3 toolkit. CloudSim offers the creation of virtual machines and its provisioning to servers for the allocations [49].

Proposed SA^2-*MCD* architecture uses *N_port* zoning, *F_port* zoning, *LUN* masking, *node_port_ID* virtualization (*NPIV*), raw device mapping (*RDM*) for hardware level security and controlled access of logical units on *VMs*. The architecture and algorithms have been simulated on CloudSim toolkit and testing of hardware level security have been evaluated in two scenarios. First, access of authorized logical units on mapped *VMs* and second, prevent unauthorized access of logical units on *VMs*.

The proposed algorithms have been simulated and evaluated on number of user requests in the range 50–250, for the secure allocation of virtual machines by the ESX_m server to ESX_s servers in multitenant cloud databases. As the work presented in this paper is advanced and giving the solutions to current issues in multitenant public cloud environment. Due to the advanced problem objective, very few researchers have touched the issues discussed in the paper. That is why, we found very limited work by other researchers for the comparisons with presented work. So, SA^2-*MCD* architecture has been evaluated on effective parameters such as allocation time (EA_T), power consumptions (P_C), resource utilization (R_U) and execution cost (E_C) and compared to existing architectures like *T-MMORRS, MGA* and *DPRA* on the parameters effective allocation time (EA_T) and power consumptions (P_C). Moreover, the

Table 3
Simulation parameters.

Parameter	Value	Unit
Simulator	CloudSim 3.0.3	–
Data centre	1	–
Number of hosts	4	–
Number of *LUNs*	4	–
Number of *VMs*	4	–
Start time (S_T)	0.1	ms
LUN Response Time (LUN_{RT})	0.0-1.1	ms
LUN access time (LUN_{AT})	0.0-2.6	ms
Finish time (F_T)	1.1-2.7	ms
Number of user tasks	50-250	–
Number of virtual machines	50-250	–
Effective allocation time (EA_T)	20-38	ms
Power consumptions (P_C)	32-61	joules
Resource utilization (R_U)	81.75-93.5	percentage
Execution cost (E_C)	54.73-305.38	$
Architectures	T-MMORRS, MGA, DPRA, PRP	–

proposed architecture has been also compared with and to another existing architecture PRP on resource utilization (R_U) and execution cost (E_C) parameters [50,51].

8. Simulation results and performance analysis

In this section, the parameters used in proposed architecture and allocation algorithms are introduced first. Then, the hardware level security, controlled access of logical units on *VMs* and performance of *SA*^{2}*-MCD* is verified through extensive simulations using CloudSim 3.0.3. Hardware level security and controlled access on *VMs* ensured by *SA*^{2}*-MCD* have been tested in two scenarios, authorized and unauthorized access of logical units on *VMs*. Analysis and comparison with other existing cloud computing architectures as aforesaid mentioned has been done. Experimental set of *SA*^{2}*-MCD* architecture and algorithms implemented and a differing set of virtual machines, and number of user tasks in a varying set allocated to single virtual machine have been considered for the performance evaluation of the proposed architecture and algorithms on effective allocation time (EA_T), power consumptions (P_C), resource utilization (R_U) and execution cost (E_C) parameters. Table 3 summarizes the parameters used in the simulation.

8.1. Scenario 1: controlled access of logical units on mapped VMs

In scenario 1, to test the hardware level security and controlled access of logical units, four ESX_s hosts $H\#0$, $H\#1$, $H\#2$ and $H\#3$ and one data centre#2 have been created. Mapping of logical units on *VMs* at front-end ESX_s hosts has been configured using port zoning with *LUN* masking, *RAW* device mapping and *NPIV* and has been shown in Table 4. Table 4 shows the execution of cloudlets executed by broker#1 and successful access of authorized *LUNs* on virtual machines (*VMs*) mapped on ESX_s hosts.

Performance of cloudlets execution on *VMs* mapped on hosts for accessing authorized logical units has been shown in Fig. 9.

Fig. 9 shows the performance evaluation of cloudlets $C\#0$, $C\#1$, $C\#2$ and $C\#3$ executed by broker#1 to access the authorized logical units $LUN\#1, LUN\#2, LUN\#3$ and $LUN\#4$ on virtual machines $VM\#0$, $VM\#1$, $VM\#2$ and $VM\#3$ mapped to ESX_s hosts $H\#0$, $H\#1$, $H\#2$, $H\#3$ respectively. All the cloudlets were permitted to access logical units on virtual machines after ensuring the hardware level security and have been executed successfully on hosts because cloudlets were assigned to hosts for the execution where authorized *LUNs* were mapped to *VMs*.

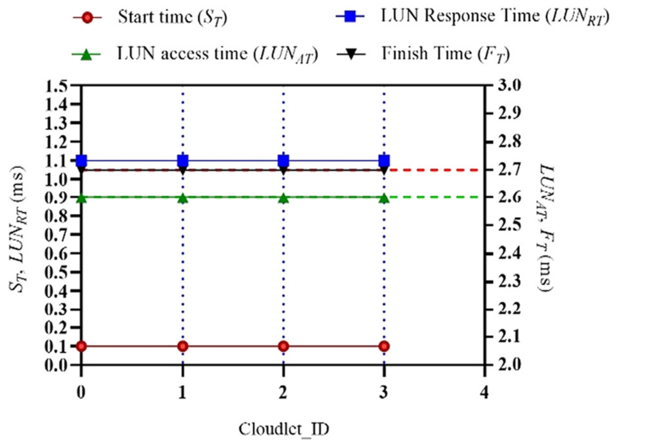

Fig. 9. Performance evaluation of cloudlets execution to access authorized *LUNs* on *VMs*.

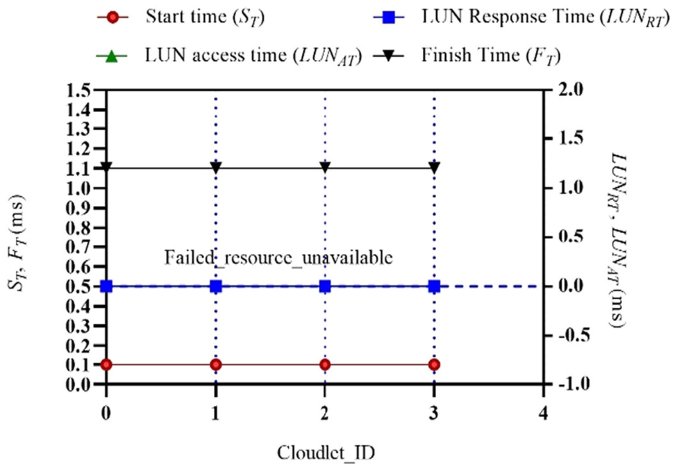

Fig. 10. Performance evaluation of cloudlets during preventing the unauthorized access of *LUNs* on *VMs*.

8.2. Scenario 2: preventing unauthorized access of logical units on VMs

In scenario 2, again four ESX_s hosts $H\#0$, $H\#1$, $H\#2$ and $H\#3$ have been created at data centre#2 and port zoning with *LUN* masking, *RAW* device mapping and *NPIV* have been used for mapping logical units on *VMs* at front-end ESX_s hosts. Authorized access of logical units to cloudlets executed by broker#2 has been shown in Table 5.

Table 5 describing that cloudlets $C\#0$, $C\#1$, $C\#2$ and $C\#3$ are authorized to access $LUN\#2, LUN\#3, LUN\#4$ and $LUN\#2$. Execution details of cloudlets executed by broker#2 to access logical units on *VMs* mapped at *ESXs* hosts are shown in Table 6.

Table 6 shows that logical units $LUN\#1, LUN\#2, LUN\#3$ and $LUN\#4$ have been mapped on virtual machines $VM\#4$, $VM\#5$, $VM\#6$ and $VM\#7$ at ESXs hosts $H\#0$, $H\#1$, $H\#2$ and $H\#3$ respectively. Cloudlets $C\#0, C\#1, C\#2$ and $C\#3$ submitted to ESXs hosts $H\#0$, $H\#1$, $H\#2$ and $H\#3$ for accessing logical units $LUN\#1$, $LUN\#2$, $LUN\#3$ and $LUN\#4$ on virtual machines $VM\#4, VM\#5, VM\#6$ and $VM\#7$ but access was denied by the *ESXs* hosts with message "*Failed_resource_unavailable*" because Cloudlets $C\#0$, $C\#1$, $C\#2$ and $C\#3$ were not authorized to access logical units $LUN\#1$, $LUN\#2$, $LUN\#3$ and $LUN\#4$ respectively. Performance evaluation of cloudlets during preventing the *LUNs* from unauthorized access is shown in Fig. 10.

Performance evaluation of cloudlets in Fig. 10 clearly showing that hosts have detected the un-authorization of cloudlets to access logical units on submitted hosts using port zoning with *LUN*

Table 4
Mapping of LUNs on VMs and ESX_s hosts and controlled access by cloudlets of broker#1.

DC_ID	LUN mapping				Start time (S_T)	LUN response time (LUN_{RT})	LUN access time (LUN_{AT})	Finish time (F_T)	Status
	Cloudlet_ID	VM_ID	LUN_ID	Host_ID					
2	0	*VM#0*	*LUN#1*	*H#0*	0.1	1.1	2.6	2.7	Success
2	1	*VM#1*	*LUN#2*	*H#1*	0.1	1.1	2.6	2.7	Success
2	2	*VM#2*	*LUN#3*	*H#2*	0.1	1.1	2.6	2.7	Success
2	3	*VM#3*	*LUN#4*	*H#3*	0.1	1.1	2.6	2.7	Success

Table 5
Cloudlets authorized to access *LUNs* executed by broker#2.

S. No.	Cloudlet_ID	Permission to access
1	0	LUN#2
2	1	LUN#3
3	2	LUN#4
4	3	LUN#2

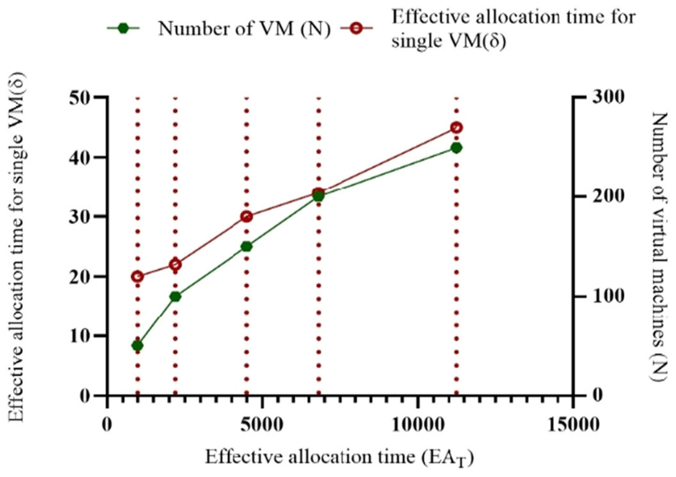

Fig. 11. Effective allocation time (EA_T) on differing set of N and δ.

masking, *RAW* device mapping and *NPIV* approach and denied the cloudlets to access the unauthorized logical units. This way, proposed *SA2-MCD* architecture ensuring the hardware level security of logical units on mapped *VMs*.

8.3. Evaluation of effective allocation time (EA_T)

In experimental setup, we have taken virtual machines in a set of 50, 100, 150, 200 and 250 to evaluate the effective allocation time (EA_T). Fig. 11 shows the evaluation of EA_T using Eq. (4) for various sets of virtual machines and effective allocation time for the allocation of single virtual machine to available ESX_s servers for values differing 20, 22, 30,34 and 45 for set of virtual machines.

Effective allocation time (EA_T) evaluated on differing set of virtual machines and different allocation time of single virtual machine to servers. The allocation time of single virtual machines will vary for different set of virtual machines due to the availability of resources at server and number virtual machines already available at server to handle. Experimental results of effective allocation time have been shown in Table 7.

Fig. 12 shows the comparative analysis of SA^2-*MCD* with *T-MMORRS*, *MGA* and *DPRA*. Result on effective allocation time as shown in figure, clearly showing that proposed SA^2-*MCD* approach is taking less time to allocate the virtual machines to ESX_s servers as compare to the approaches *T-MMORRS*, *MGA* and *DPRA*. Effective allocation time in SA^2-*MCD* for the allocation of 250 virtual machine is 38 ms which is less than 30.9% from *T-MMORRS*, 51.28% from *MGA* and 47.9% from *DPRA* technique.

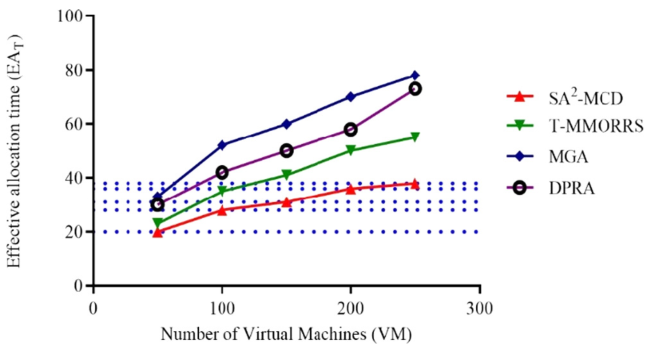

Fig. 12. Effective allocation time (EA_T) for virtual machine allocation.

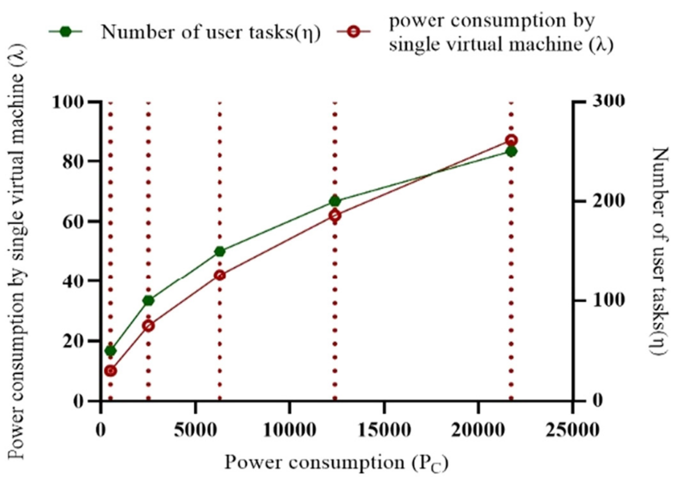

Fig. 13. Power consumptions (P_C) on differing set of η and λ.

8.4. Evaluation of power consumptions (P_C)

Fig. 13 is showing the power consumptions (P_C) at ESX_s servers on number of user tasks differing in a set of 50, 100, 150, 200 and 250 and on differing values for power consumed by single virtual machine using Eq. (5).

Fig. 13 shows the evaluation of power consumption (P_C) on different set of user tasks and on different values for power consumed by single virtual machine at ESX_s server. Power consumed by single virtual machine will depend on the number of user tasks executing on single virtual machine. Table 8, shows the experimental result on power consumptions (P_C) which is consumed by virtual machines at ESX_s server to service the user tasks.

As shown in the figure and table, results stating that in SA^2-*MCD*, virtual machines are consuming less power to service the user tasks in comparison to *T-MMORRS*, *MGA* and *DPRA* approaches. For a set of 250 user tasks, power consumption in SA^2-*MCD*, *T-MMORRS*, *MGA* and *DPRA* are 61, 73, 83 and 78 joules respectively. Comparative analysis of approaches has been shown in Fig. 14.

Table 6
Mapping of *LUNs* to *VMs* and *ESXs* Hosts and unauthorized access by cloudlets of broker#2.

DC_ID	LUN mapping				Start time (S_T)	LUN response time (LUN_{RT})	LUN access time (LUN_{AT})	Finish time (F_T)	Status
	Cloudlet_ID	VM_ID	LUN_ID	Host_ID					
2	0	*VM#4*	*LUN#1*	*H#0*	0.1	0	0	1.1	Failed_resource_unavailable
2	1	*VM#5*	*LUN#2*	*H#1*	0.1	0	0	1.1	Failed_resource_unavailable
2	2	*VM#6*	*LUN#3*	*H#2*	0.1	0	0	1.1	Failed_resource_unavailable
2	3	*VM#7*	*LUN#4*	*H#3*	0.1	0	0	1.1	Failed_resource_unavailable

Table 7
Experimental result on effective allocation time (EA_T).

Number of virtual machines	Effective allocation time (EA_T) (ms)			
	SA^2-MCD	T-MMORRS	MGA	DPRA
50	20	23	33	30
100	28	35	52	42
150	31	41	60	50
200	36	50	70	58
250	38	55	78	73

Table 8
Experimental result on power consumptions (P_C).

Number of user tasks	Power consumptions (P_C)			
	SA^2-MCD	T-MMORRS technique	MGA	DPRA
50	32	33	40	38
100	46	50	63	58
150	47	53	66	60
200	52	62	76	70
250	61	73	83	78

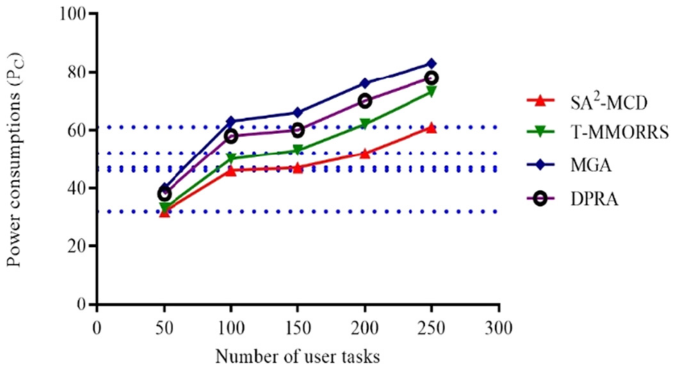

Fig. 14. Result of power consumptions (P_C) over number of user tasks.

Fig. 14 shows that *SA^2-MCD* is performing well as compare to the approaches *T-MMORRS, MGA* and *DPRA*. Approaches have been evaluated for the set of 50, 100, 150, 200 and 250 number of user tasks. As shown in the figure, when the tasks are increasing on virtual machines, *SA^2-MCD* is performing much better than the other existing approaches. On a set of 250 tasks, *SA^2-MCD* is consuming less power as 16.44%, 26.51% and 21.79% compared to *T-MMORRS, MGA* and *DPRA* respectively.

8.5. Evaluation of resource utilization (R_U)

Fig. 15 shows the resource utilization (R_U) at *ESX$_s$* servers and evaluated using the Eq. (6).

Fig. 15 presents the evaluation of resource utilization on differing values on total resources (μ) and free resources (π). Resource utilization (R_U) will depend on set of virtual machines and number of user tasks that are executing on single virtual machine and required resources by the user tasks on virtual machines. Utilization of available resources at *ESX$_s$* servers by the virtual machines

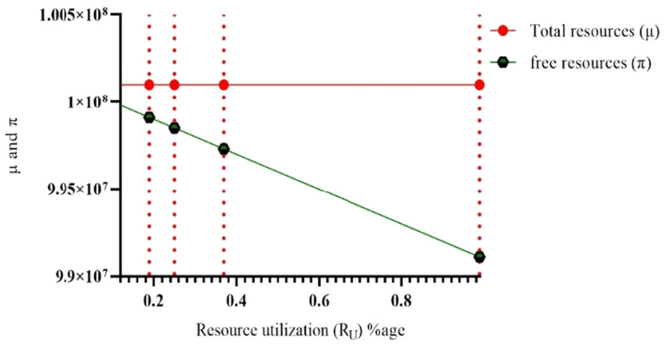

Fig. 15. Resource utilization (R_U) on values of μ and π.

Table 9
Experimental result on resource utilization (R_U).

Number of user tasks	Resource Utilization (R_U) %	
	SA^2-MCD	PRP
50	81.75	80.55
100	88.83	87.5
150	84.04	82.65
200	85	83.34
250	93.5	91.67

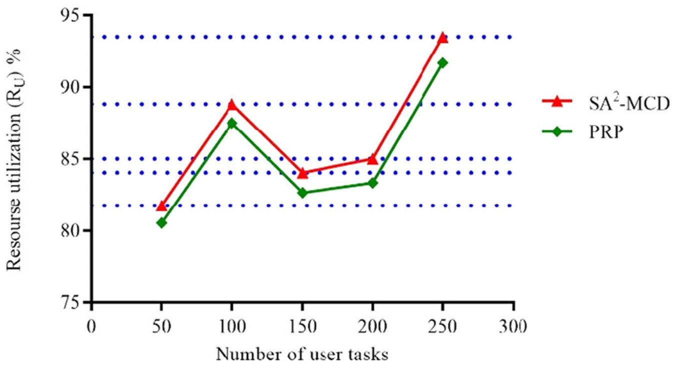

Fig. 16. Percentage of resource utilization (R_U).

has been shown in Table 9 for a set of user tasks of 50, 100, 150, 200 and 250.

The experimental result on resource utilization (R_U) showing that proposed *SA^2-MCD* is offering more utilization of resources in comparison to proactive resource provisioning (*PRP*) approach. Comparative analysis of *SA^2-MCD* and *PRP* has been shown in Fig. 16.

Analysis of *SA^2-MCD* and *PRP* in Fig. 16 clearly showing that percentage of resource utilization in *SA^2-MCD* is more than the *PRP* approach but it is slightly higher around 1% to 2%. Thus, *SA^2-MCD* gives better results than *PRP*.

8.6. Evaluation of execution cost (E_C)

As discussed in section 6.4 that the evaluation of execution cost (E_C) depends on the network cost, storage cost and virtual ma-

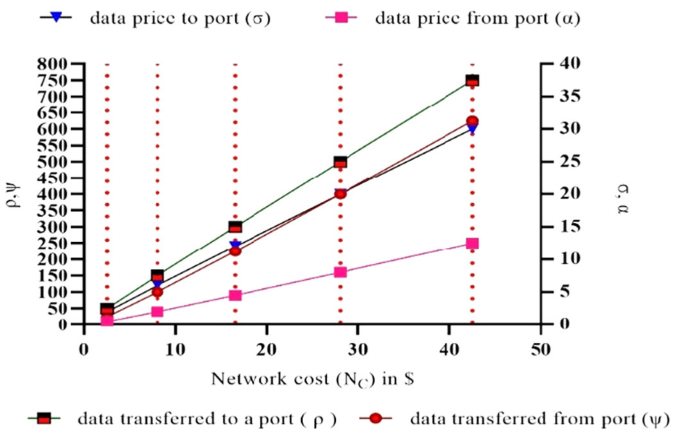

Fig. 17. Network cost (N_C) on values of ρ, σ, ψ and α.

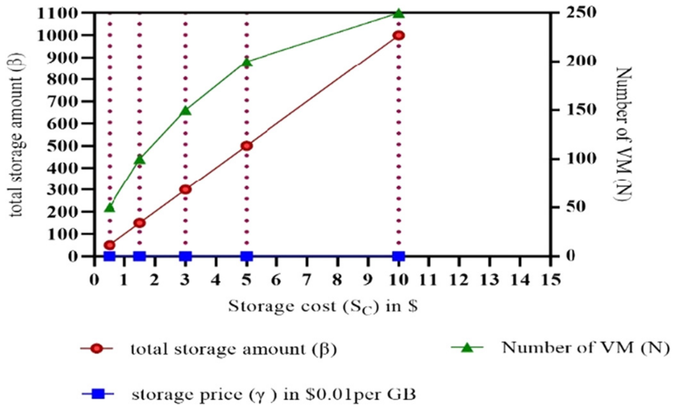

Fig. 18. Storage cost (S_C) on differing values of β and γ.

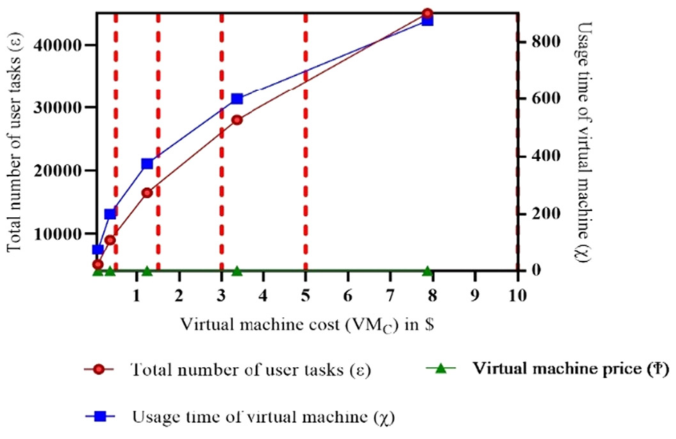

Fig. 19. Virtual machine cost (VM_C) on ε, χ and $Ͳ$.

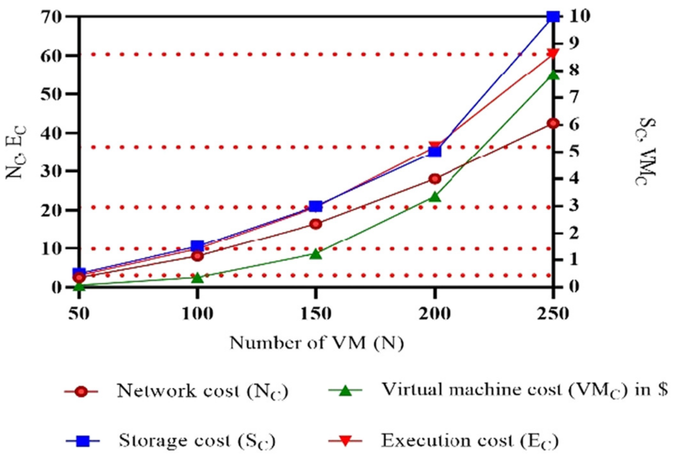

Fig. 20. Execution cost (E_C) on N_C, S_C and VM_C.

Table 10
Experimental result on execution cost (E_C).

Number of user tasks	Execution cost (E_C)	
	SA^2-MCD	PRP
50	54.73	60.81
100	84.72	94.13
150	236.93	263.25
200	212.01	227.97
250	305.38	328.37

chine cost, and can be evaluated using Eq. (10). Fig. 17 given above shows the evaluation of network cost (N_C) in $ using Eq. (7) for a set of virtual machines 50, 100, 150, 200 and 250 and data transfer price [46].

Therefore, evaluation of network cost (N_C) has been performed for set of virtual machines and differing values of ρ, σ, ψ and α parameters. Thus, the evaluation of network cost principally depends on data transfer price and will vary from one service provider to another service provider.

Fig. 18 shows the evaluation of storage cost using Eq. (8) on differing set of virtual machines of 50, 100, 150, 200 and 250 and storage price by the service providers [47].

Fig. 18 shows the evaluation of storage cost (S_C) on differing values of β and γ parameters. Storage cost may differ from one service provider to other on the basis of price offered. Fig. 19 is presenting the evaluation of virtual machine cost using the Eq. (9) on differing set of number of user tasks and virtual machine price by the service providers [48].

Evaluation of virtual machine cost has been shown in Fig. 19 on $0.20 per 1 million tasks and differing set of virtual machines of 50, 100, 150, 200 and 250. Fig. 20 is presenting the evaluation of execution cost (E_C) using Eq. (10) based on network cost (N_C), storage cost (S_C) and virtual machine cost (VM_C).

Execution cost (E_C) has been evaluated using the values of N_C, S_C and VM_C evaluated in Fig. 17, Fig. 18 and in Fig. 19 respectively. Differing set of virtual machines 50, 100, 150, 200 and 250 has been used in execution cost evaluation. *SA^2-MCD* has been evaluated on execution cost and the experimental result has been shown in Table 10.

Execution cost (E_C) has been evaluated for a set of 50, 100, 150, 200 and 250 number of user tasks. Experimental results shown in Table 10, showing that proposed approach *SA^2-MCD* is presenting economical execution cost than *PRP* approach. Comparative analysis of *SA^2-MCD* with *PRP* on execution cost has been depicted in Fig. 21.

Therefore, comparative results shown above stating that the proposed *SA^2-MCD* approach offering around 5%-10% of economical execution cost than *PRP* approach which is quite good but may be reduced more by updating the algorithm to offer more economical cost.

Finally, we have seen the simulation section, from where results reveals that the proposed architecture and algorithms gives better performance than the existing cloud computing architectures.

9. Conclusion and future works

In this paper, we have proposed the secured architecture for allocation of virtual machines in multitenant cloud databases that we have called as *SA^2-MCD*. This proposed architecture ensuring the hardware level security of public cloud databases in multi-

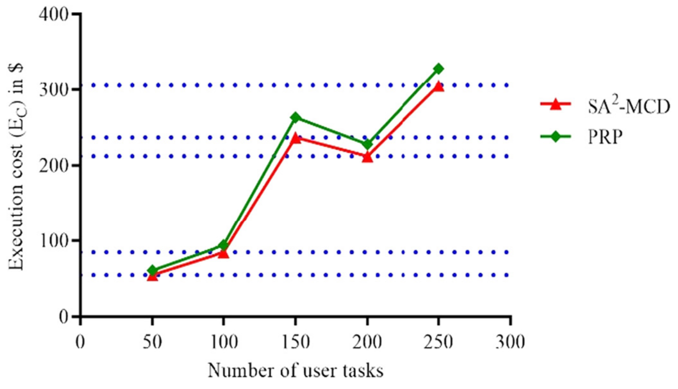

Fig. 21. Execution cost (E_C) in $.

tenant environment. *DBaaS* service in architecture ensuring the database availability, database security, privacy, performance and workload management. As the proposed *SA*^{2}*-MCD* architecture ensuring the hardware level security and controlled access of logical units on *VMs*, so the testing of hardware level security and controlled access of logical units have been simulated on CloudSim 3.0.3 and the experimental results are clearly showing that architecture is performing well to ensure the hardware level security and controlled authorized access and unauthorized access of logical units on *VMs*. The proposed architecture *SA*^{2}*-MCD* and algorithms also has been implemented and evaluated on effective allocation time (EA_T), power consumptions (P_C), resource utilization (R_U) and execution cost (E_C) respectively. We have simulated the proposed architecture using CloudSim 3.0.3 and compared the results with existing architectures *T-MMORRS, MGA, DPRA* and *PRP* approaches. Results show that proposed approach *SA*^{2}*-MCD* has performed better than *T-MMORRS, MGA, DPRA* and *PRP* on EA_T, P_C, R_U and E_C parameters. As shown in results section, *SA*^{2}*-MCD* has offered only 1% to 2% of higher resource utilization as compared to *PRP* approach which may be maximized and 5%-10% reduced economical execution cost than PRP approach which can be reduced more. Further, the work may be enhanced to maximize the resource utilization up to 95% to minimize the power consumptions as well as to reduce the economic cost up to 20% in future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Peter Mell, Timothy Grance, The NIST definition of cloud computing, National Institute of Standards and Technology (NIST, Information Technology Laboratory, 2009 [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/index.html.

[2] Lizhe Wang, et al., Towards building a cloud for scientific applications, Adv. Eng. Softw. 42 (9) (2011) 714–722.

[3] Lizhe Wang, et al., Cloud computing: a perspective study, New Gener. Comput. 28 (2) (2010) 137–146.

[4] Lizhe Wang, et al., Resource management of distributed virtual machines, Int. J. Ad Hoc Ubiq. Comput. 10 (2) (2012) 96–111.

[5] Lizhe Wang, Dan Chen, Fang Huang, Virtual workflow system for distributed collaborative scientific applications on Grids, Comput. Electr. Eng. 37 (3) (2011) 300–310.

[6] Anthony D. Joseph, et al., A view of cloud computing, Commun. ACM 53 (2010) 4.

[7] Ripal Nathuji, Aman Kansal, Alireza Ghaffarkhah, Q-clouds: managing performance interference effects for QoS-aware clouds, in: Proceedings of the 5th European Conference on Computer Systems, ACM, 2010.

[8] Ram Shringar Raw, Manish Kumar, Nanhay Singh, Security issues and solutions in Vehicular Ad hoc Network: a review approach, in: ICCSEA, SPPR, CSIA, WimoA, 2013, p. 339347.

[9] Naidila Sadashiv, S.M. Dilip Kumar, Cluster, grid and cloud computing: a detailed comparison, in: 2011 6th International Conference on Computer Science & Education, ICCSE, IEEE, 2011.

[10] Wayne A. Jansen, Cloud hooks: security and privacy issues in cloud computing, in: 2011 44th Hawaii International Conference on System Sciences, IEEE, 2011.

[11] Sagar Verma, et al., An efficient data replication and load balancing technique for fog computing environment, in: 2016 3rd International Conference on Computing for Sustainable Global Development, INDIACom, IEEE, 2016.

[12] S. Johnson, Cloud computing types: public cloud, hybrid cloud, private cloud, in: CircleID, Iomemo Inc., Mar. 2009, http://tinyurl.com/CCTypes.

[13] Booz Allen Hamilton, The economics of cloud computing, http://tinyurl.com/BoozAllenH.

[14] Joe Weinman, Cloudonomics: The Business Value of Cloud Computing, John Wiley & Sons, 2012.

[15] Mithani Mohammad Firoj, Michael A. Salsburg, Shrisha Rao, A decision support system for moving workloads to public clouds, GSTF J. Comput. 1 (2018) 1.

[16] Mohit Kumar, et al., Global host allocation policy for virtual machine in cloud computing, Int. J. Inf. Technol. 10 (3) (2018) 279–287.

[17] Ahmed A. Soror, et al., Automatic virtual machine configuration for database workloads, ACM Trans. Database Syst. 35 (1) (2010) 7.

[18] Gokul Soundararajan, et al., Dynamic resource allocation for database servers running on virtual storage, in: Fast 9, 2009.

[19] Milinda Pathirage, et al., A scalable multi-tenant architecture for business process executions, Int. J. Web Serv. Res. 9 (2) (2012) 21–41.

[20] Kun Ma, Zijie Tang, An online social mutual help architecture for multi-tenant mobile clouds, Int. J. Intell. Inform. Datab. Syst. 8 (4) (2014) 359–374.

[21] Michael Armbrust, et al., Spark SQL: relational data processing in spark, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015.

[22] Rosario Gennaro, Craig Gentry, Bryan Parno, Non-interactive verifiable computing: outsourcing computation to untrusted workers, in: Annual Cryptology Conference, Springer, Berlin, Heidelberg, 2010.

[23] S. Rahmani, V. Khajehvand, Burst-aware virtual machine migration for improving performance in the cloud, Int. J. Commun. Syst. (2020) e4319, https://doi.org/10.1002/dac.4319.

[24] P. Krishnan, S. Duttagupta, K. Achuthan, SDN/NFV security framework for fog-to-things computing infrastructure, Softw. Pract. Exp. (2019) 1–44, https://doi.org/10.1002/spe.2761.

[25] N.E.H. Bouzerzour, S. Ghazouani, Y. Slimani, A survey on the service interoperability in cloud computing: client-centric and provider-centric perspectives, Softw. Pract. Exp. (2020) 1–36, https://doi.org/10.1002/spe.2794.

[26] A. Akbar Neghabi, N. Jafari Navimipour, M. Hosseinzadeh, A. Rezaee, Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network, Int. J. Commun. Syst. (2019) e3875, https://doi.org/10.1002/dac.3875.

[27] A. Brogi, S. Forti, C. Guerrero, I. Lera, How to place your apps in the fog: state of the art and open challenges, Softw. Pract. Exp. (2019) 1–22, https://doi.org/10.1002/spe.2766.

[28] K.R. Remesh Babu, P. Samuel, Service-level agreement–aware scheduling and load balancing of tasks in cloud, Softw. Pract. Exp. (2019) 1–18, https://doi.org/10.1002/spe.2692.

[29] L.R. Rodrigues, E. Cardoso Jr, O.C. Alves Jr, et al., Cloud broker proposal based on multicriteria decision-making and virtual infrastructure migration, Softw. Pract. Exp. (2019) 1–21, https://doi.org/10.1002/spe.2723.

[30] A. Singh, N. Auluck, Load balancing aware scheduling algorithms for fog networks, Softw. Pract. Exp. (2019) 1–19, https://doi.org/10.1002/spe.2722.

[31] V. Arulkumar, N. Bhalaji, Load balancing in cloud computing using water wave algorithm, Concurr. Comput., Pract. Exp. (2019) e5492, https://doi.org/10.1002/cpe.5492.

[32] S.U.R. Malik, H. Akram, S.S. Gill, H. Pervaiz, H. Malik, EFFORT: energy efficient framework for offload communication in mobile cloud computing, Softw. Pract. Exp. (2020) 1–14, https://doi.org/10.1002/spe.2850.

[33] Glauco Estácio Gonçalves, et al., Resource allocation based on redundancy models for high availability cloud, Computing 102 (1) (2020) 43–63.

[34] C. Yang, T. Wan, Implementation of an energy saving cloud infrastructure with virtual machine power usage monitoring and live migration on OpenStack, Computing 102 (2020) 1547–1566, https://doi.org/10.1007/s00607-020-00808-7.

[35] Zhongsheng Qian, et al., An approach to dynamically assigning cloud resource considering user demand and benefit of cloud platform, 2020.

[36] Y. Tan, W. Wu, J. Liu, H. Wang, M. Xian, Lightweight edge-based KNN privacy-preserving classification scheme in cloud computing circumstance, Concurr. Comput., Pract. Exp. (2020) e5804, https://doi.org/10.1002/cpe.5804.

[37] M. Aslam, S. Bouget, S. Raza, Security and trust preserving inter- and intra-cloud VM migrations, Int. J. Netw. Manag. (2020) 1–19, https://doi.org/10.1002/nem.2103.

[38] Arun Kumar Yadav, Rajendra Kumar Bharti, Ram Shringar Raw, Security solution to prevent data leakage over multitenant cloud infrastructure, Int. J. Pure Appl. Math. 118 (7) (2018) 269–276.

[39] J. Prassanna, Neelanarayanan Venkataraman, Threshold based multi-objective memetic optimized round Robin scheduling for resource efficient load balancing in cloud, Mob. Netw. Appl. 24 (4) (2019) 1214–1225.
[40] Fan-Hsun Tseng, et al., Dynamic resource prediction and allocation for cloud data centre using the multi-objective genetic algorithm, IEEE Syst. J. 12 (2) (2018) 1688–1699.
[41] Li-Der Chou, et al., DPRA: dynamic power-saving resource allocation for cloud data centre using particle swarm optimization, IEEE Syst. J. 12 (2) (2018) 1554–1565.
[42] Ankita Jain, et al., A proactive approach for resource provisioning in cloud computing, Int. J. Recent Technol. Eng. 7 (2019) 435–444.
[43] Ahmed Aliyu, Abdul Hanan Abdullah, Omprakash Kaiwartya, Yue Cao, Mohammed Joda Usman, Sushil Kumar, D.K. Lobiyal, Ram Shringar Raw, Cloud computing in VANETs: architecture, taxonomy, and challenges, IETE Technical Review, https://doi.org/10.1080/02564602.2017.1342572, 2017.
[44] Priyanka Gaba, Ram Shringar Raw, Vehicular cloud and fog computing architecture, applications, services, and challenges, in: IoT and Cloud Computing Advancements in Vehicular Ad-Hoc Networks, IGI Global, 2020, pp. 268–296.
[45] Carlo Curino, et al., Relational cloud: a database-as-a-service for the cloud, in: 5th Biennial Conference on Innovative Data Systems Research, CIDR 2011, January 9–12, Asilomar, California, 2011.
[46] https://aws.amazon.com/getting-started/projects/connect-data-center-to-aws/services-costs.
[47] https://aws.amazon.com/storagegateway/pricing/.
[48] https://www.simform.com/compute-pricing-comparison-aws-azure-googlecloud/.
[49] Rodrigo N. Calheiros, et al., CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exp. 41 (1) (2011) 23–50.
[50] https://www.networkworld.com/article/3164444/how-to-calculate-the-true-cost-of-migrating-to-the-cloud.html.
[51] Sharlene Aminullah, Carlos Molina-Jimenez, Cost estimation of service delivery in cloud computing, in: Internet Technologies and Enterprise Computing, 2012.