Data Model for Cloud Computing Environment

Samson B. Akintoye[1(✉)], Antoine B. Bagula[1], Omowumi E. Isafiade[1], Yacine Djemaiel[2], and Noureddine Boudriga[2]

[1] University of the Western Cape, Cape Town, South Africa
3515640@myuwc.ac.za
[2] CNAS Lab, Cartage University, Tunis, Tunisia
ydjemaiel@gmail.com

Abstract. The emergence of cloud computing has reduced the cost of deployment and storage dramatically, but only if data can be distributed across multiple servers easily without disruption. In a complex SQL database, this is difficult because many queries require multiple large tables to be joined together to provide a response. Executing distributed joins is a very complex problem in SQL databases. In addition, previous studies have shown that NoSQL databases performance better than SQL databases especially in the cloud computing environment where there is occurrence of huge volume of data. In this paper, we presents a novel data model for cloud services brokerage that supports the allocation, control and management of virtual system based on brokering function between cloud service providers (CSPs) and cloud users by integrating and man- aging cloud resources in a heterogeneous cloud environment. The model is implemented on a private lightweight cloud network using a graph and document-oriented databases. The experimental results show that a graph model has better performance than a document-oriented model in terms of queries execution time.

Keywords: Cloud computing · Graph model · Document-oriented model · Cloud Services Brokerage

1 Introduction

Cloud computing has recently emerged as one of the most promising and challenging technologies. It is based on a computing paradigm where a large pool of systems are connected in private, public or hybrid networks, to provide dynamically scalable infrastructure for computing resources [19]. The computing resources are available to the users via the internet [13]. The characteristics of cloud computing include on-demand self service, broad network access, resource pooling, rapid elasticity and measured service. On-demand self service means that organizations can access and manage their own computing resources. Broad

G. Mendy et al. (Eds.): AFRICOMM 2018, LNICST 275, pp. 199–215, 2019.
https://doi.org/10.1007/978-3-030-16042-5_19

network access allows services to be offered over the Internet or private networks. Pooled resources mean that customers draw from a pool of computing resources. Services can be scaled larger or smaller; and use of a service is measured. The cloud computing service models are Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [10]. In Software as a Service model, consumer uses the provider's applications running on a cloud infrastructure. Example of SaaS is Salesforce [2]. In PaaS, an operating system, hardware, and network are provided, and the customer installs or develops its own software and applications. The most prominent key players of PaaS are Azure platform [8] and Google App Engine [5]. The IaaS model provides just the hardware and network; the customer installs or develops its own operating systems, software and applications. The examples of IaaS are Amazon EC2 service [1], GoGrid [4], Flexiscale [3], and Redplaid [7].

Cloud services are deployed as a private cloud, community cloud, public cloud or hybrid cloud [10]. In public cloud, services are offered over the Internet and are owned and operated by a cloud provider. In a private cloud, the cloud infrastructure is operated solely for a single organization, and is managed by the organization or a third party. In a community cloud, the service is shared by several organizations and made available only to those groups. The infrastructure may be owned and operated by the organizations or by a cloud service provider. A hybrid cloud is a combination of two or more cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability. Virtualization technologies are usually used to access computing resource by the users. Users can specify required software stack such as operating systems, software libraries, and applications, then package them all together into virtual machines (VMs). VMs will be hosted in cloud service providers. Lightweight cloud computing infrastructures combine cloud and grid computing concept to provide a shared infrastructure over commodity hardware such as mobile phones, desktop, tablets, etc. As the cloud computing market expands and the number of users and cloud service providers increases, there is a need for a centralised system [22], called cloud services brokerage (CSB), to optimize resource allocation by managing and monitoring the activities of cloud users and cloud service providers. For the CSB to work efficiently, a reliable database management system needs to be implemented on the CSB site to keep and update the track of customer requests and cloud infrastructures status. Relational database management systems (RDBMS) otherwise known as SQL database cannot cope with the unprecedented scale factors that modern cloud-based applications have introduced. The cloud applications need to support large numbers of concurrent users and be able to handle unstructured and semi-structured data. To solve this problem, NoSQL (Not Only SQL) databases emerge to support large-scale application demands. In addition, previous studies have shown that NoSQL databases perform better than SQL databases especially in the cloud computing environment where there are occurrence of huge volume of data [27].

1.1 Contributions and Outline

In this paper, we present a data model for cloud computing environment to help the CSB support the allocation, control and management of virtual resources between CSPs and cloud users. We implement this model using the graph database (Neo4j) and document-oriented database (mongodb) on our private lightweight cloud testbed, using a syntactic of cypher language to store, update and retrieve the customer requests and cloud infrastructures status in the database. Building upon the free and open source OpenStack software platform for cloud computing, the model is intended to provide infrastructure-as-a-service (IaaS) in community sensor networks [29] for applications such as drought mitigation for small scale farming [30,31] and cyber healthcare [32,33] in the rural areas of the developing countries. Potential applications which might also benefit from this model include smart parking [34], pollution monitoring [35] and public safety [36] in the smart developing cities. The rest of this paper is organized as follows; In Sect. 2, we describe the concept of cloud service brokerage system. The next section presents previous studies related to the management of virtual resources information in data center and cloud computing. Section 4 describes proposed cloud computing environment model. The data model of cloud computing environment is presented in Sect. 5. Implementation of the models and experimental results are found in Sects. 6 and 7, and finally, we conclude the paper in Sect. 8.

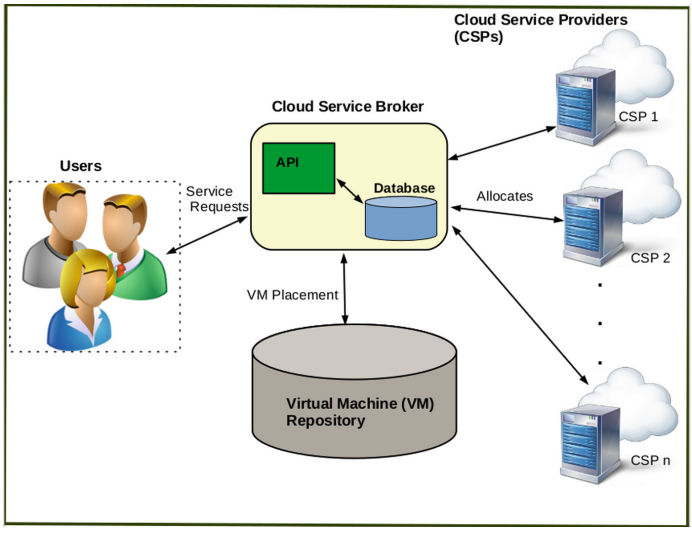

Fig. 1. Cloud computing environment

2 Cloud Services Brokerage

As depicted in Fig. 1, the cloud computing environment considered in this paper consists of user, virtual machine repository (VMR), and cloud services broker and CSP, which consists of physical machines (PM) and data center (DC). A cloud services broker is a third-party individual or business acting as a middle man between cloud service users and CSPs. Cloud service brokers rent different types of cloud resources from many cloud Service providers and sublet these resources to the requesting cloud users. The cloud service broker performs the following functions: (i) optimal placement of the virtual resource of a virtual infrastructure across multiple cloud service providers; (ii) management and monitoring of these virtual resources; and (iii) aggregation of multiple cloud services into one or more customer-tailored cloud services. OPTIMIS [6] identifies the requirement and capabilities that a cloud service broker needs to have in order to play the role of brokerage services:

- Effectively match the requirements of the cloud user with the service provided by the CSPs.
- Negotiate with CSPs and cloud users over service level agreements (SLA).
- Effectively deploy services of CSP onto the cloud users.
- Maintain performance check on these SLA's and take actions against SLA violations.
- Ensure data confidentiality and integrity of CSP's service.
- Enforce access control decisions uniformly across multiple CSPs.
- Securely map identity and access management systems of the CSPs.

However, an effective database management system needs to be included as one of the functional components of a CSB.

3 Related Work

This section presents two broad categories of related work. The first category discusses existing cloud brokerage systems and the second category presents the related work to the database model in cloud computing.

3.1 Existing Cloud Brokerage Systems

Many broker-based systems have been proposed to solve cloud computing problems. Heilig et al. [16] propose a cloud brokerage approach to solve the Cloud Resource Management Problem in multi-cloud environments with aim to reduce the monetary cost and the execution time of consumer applications using Infrastructure as a Service of multiple cloud providers. In [15], the authors propose a broker-based architecture and algorithm for placing and migrating virtual resources to physical machines. In [20], the authors propose a federated cloud computing environment in which a cloud broker has the ability to interface more than one cloud provider to support several users. These Users access cloud

services via web interface. The cloud service broker pays the usage of the cloud resources to the cloud service provider, and charges the user for these resources. In [25], a solution to manage the information of a large number of cloud service providers via a unique indexing technique is proposed. STRATOS [21] proposes a cloud brokerage service that solves a Resource Acquisition Decision (RAD) problem in the selection of n resources from m cloud services. In [18] develops a cloud brokerage service for measuring the performance of a range of cloud services including; elastic compute clusters, persistent storage, intra-cloud networking and wide-area networking. [17] proposes a novel secure sharing mechanism for a secure cloud bursting and aggregation operation in which the cloud resources are shared in a confidential manner among different cloud environments.

3.2 Database Model in Cloud Computing

Goli-Malekabadi et al. [14] proposes an effective database model for storing and retrieving big health data in cloud computing. The study presents the model based on NoSQL databases for the storage of healthcare data and was implemented in the cloud environment for gaining access to the distribution properties. The experimental results of the model was evaluated with relational database model in terms of query execution time, data preparation, flexibility and extensibility parameter. The results show that the proposed model outperforms the relational database. In [28], the authors propose a novel protocol to enable secure and efficient database outsourcing. First, the authors propose a new cloud database model by introducing computation service providers which can accommodate the conventional DBaaS model and introduce a proposed database outsourcing protocol secureDBS which uses a secret sharing mechanism. The experiments conducted show that the proposed model is reliable, secure and efficient. In [12], the authors propose a novel management scheme that enables the representation and the retrieval of (structured or unstructured) big data using conceptual graphs and structured marks. Curino et al. [11] proposes relational database as-a-service for the cloud. This work describes the challenges and requirements of a large-scale, multi-node DBaaS and presents the design principles and implementation status of relational cloud. The advantage of this work is that it addresses three significant challenges, which are: (i) efficient multi-tenancy; (ii) elastic scalability; and (iii) database privacy.

However, none of these works have proposed a graph data model and its implementation using graph database as of the requirement for the effectively cloud brokerage services.

4 Cloud Computing Environment Model

In this section, we introduce the system model for our cloud computing environment. As depicted in Fig. 1, the cloud computing environment consists of User, Virtual Machine Repository (VMR), Cloud Service Provider (CSP), Physical Machines (PM), Data Center (DC) and Cloud Services Broker (CSB).

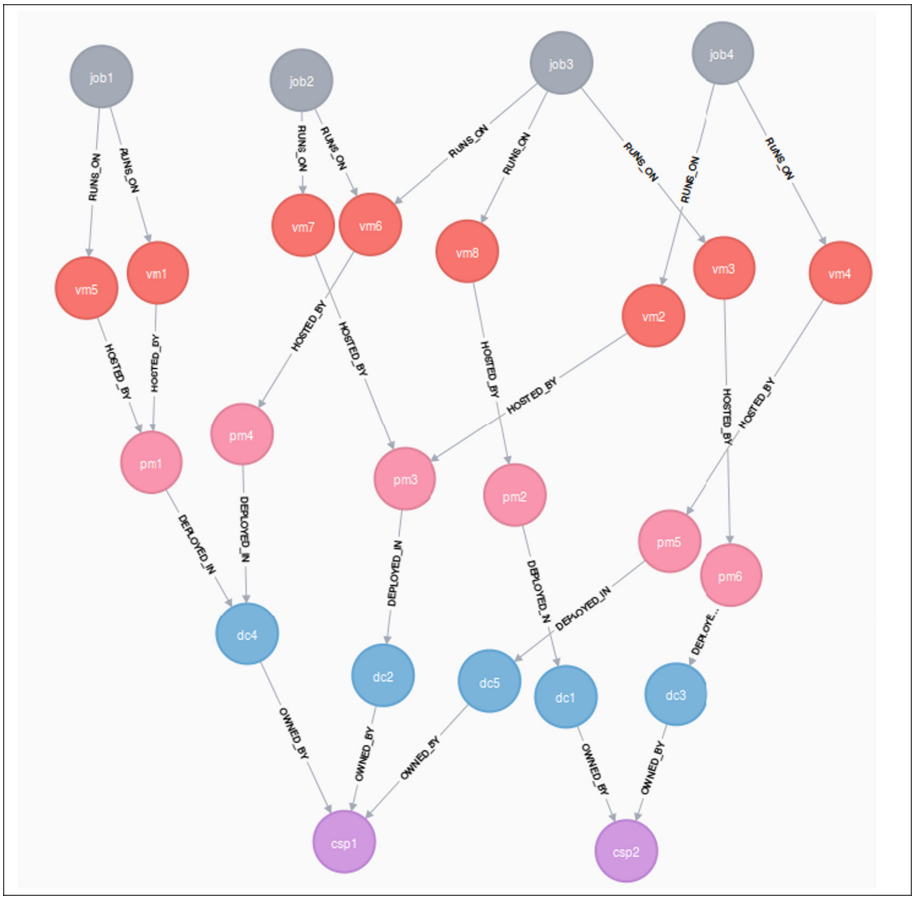

Fig. 2. Graph model for cloud computing environment

We consider a set CSP,

$$CSP = \{csp_1, csp_2....csp_n\} \tag{1}$$

where n is the number of CSPs managed by the CSB. Each CSP consists of DC,

$$DC = \{dc_1, dc_2....dc_m\} \tag{2}$$

where m is the number of DCs in a CSP and each DC contains a member of PMs,

$$PM = \{pm_1, pm_2....pm_q\} \tag{3}$$

where q is the number of PM in a DC and each PM hosts t number of VMs as expressed by the set

$$VM = \{vm_1, vm_2....vm_t\} \tag{4}$$

We consider that K jobs need to be allocated to CSPs. Each job k requires w_j number of virtual machines. The CSB allocation is expressed by the notation

$$k \longrightarrow vm_{w_j} \tag{5}$$

subject to $w_j = 1$ or $w_j \leq t$. Each vm is placed in one pm,

$$vm \longrightarrow pm \tag{6}$$

Each pm is hosted by one data center dc,

$$pm \longrightarrow dc \tag{7}$$

Finally, each data center is owned by one cloud service provider (csp),

$$dc \longrightarrow csp \tag{8}$$

5 Data Model of Cloud Computing Environment

There are many different NoSQL data models and each one of them has a different structure. In this section, we present the deployment of graph and document-oriented models for cloud computing environments.

5.1 Graph Model

Graph data models have emerged with the objective of modeling information whose structure is a graph [9]. It encodes entities and relationships between entities using directed graph structure [23]. It is a set of vertices and edges where vertices denote nodes and edges represent relationship between these Nodes. Graphs are data structures for storing data that is heterogeneously structured. Graphs can be directed or undirected. Undirected graphs can be traversed in both directions while directed graphs can be traversed only in one direction. The properties of a graph model includes the following [24];

- It contains nodes and relationships.
- Nodes contain properties (key-value pairs).
- Nodes can be labelled with one or more labels.
- Relationships are named and directed, and always have a start and end node.
- Relationships can also contain properties.

The graph model of cloud computing environment can be represented mathematically as a graph $G(V, E)$ where V is the set of resource nodes and E is the set of relationships between the nodes such that,

$$\{CSP, DC, PM, VM, K\} \subset V \tag{9}$$

Constrained by the notation:

$$|DC| \geq |CSP| \tag{10}$$

$$|VM| \geq |PM| \tag{11}$$

$$|K| \geq |VM| \tag{12}$$

The relationships between the nodes of the graph (V, E) are defined as follows:

- $csp \rightarrow dc$ represents the relationship between cloud service provider and data center.
- $dc \rightarrow pm$ represents the relationship between data center and physical machine.
- $pm \rightarrow vm$ represents the relationship between physical machine and virtual machine.
- $vm \rightarrow k$ represents the relationship between virtual machine and job.

Such that,

$$csp \rightarrow dc, dc \rightarrow pm, pm \rightarrow vm, vm \rightarrow k \subset E \tag{13}$$

The graph is illustrated by Fig. 2.

5.2 Document-Oriented Database

In a document-oriented model, data objects are stored as documents; each document stores data which can be updated or deleted. Instead of columns with names and data types, data is described in the document, and provide the value for that description. The difference between a relational model and a document-oriented model is as follows: in a relational model, data is added by modifying the database schema to include the additional columns and their data types while in document-based data, additional key-value pairs will be added into documents to represent the new fields. The document-oriented model for cloud computing environment is represented in Fig. 3.

6 Experiments

We conducted two different experiment in order to evaluate the both graph and document-oriented models on a proxy node of our private lightweight cloud testbed running on Openstack architecture. The proxy node is a Linux Machine with an Inter(R) core(TM) i5-4590, 3.30 Ghz CPU, 8 GB RAM and serves as a cloud brokerage system which initiate upload and download operations across multiple storage nodes. The CSB can update the cloud resources status in database either manually or through real-time process by building Application Protocol Interface (API) connecting cloud infrastructures to a database. We explore the suitability of two different databases in a cloud environment, the database are: (i) graph database: Neo4j; and document-oriented database: MongoDB respectively.

```
{
    "_id":ObjectId(),
    "job_id": "Job identity",
    "name": "name of job",
    "length": "length of the job",
    "file-size" : "size of the job",
    "VIRTUALMACHINE" : [
      {
        "vm_id" : "VM identity",
        "name" : "name of VM",
        "mips" : "value of mips",
        "ram" : "valus of ram"
        "PHYSICALMACHINE": [
          {
            "pm_id": "PM identity",
            "name" : "name of pm",
            "storage" : "value of storage",
            "ram" : "value of ram",
            "mips" : "value of mips"
            "DATACENTER": [
              {
                "center_id": "center identity",
                "name" : "name of center",
                "OS" : "OS",
                "location" : "location",
                "arch" : "Arch",
                "time_zone" : "Time zone",
                "CLOUDPROVIDER" : [
                  {
                    "csp_id": "provider ID",
                    "name": "provider name"
                    "cost_per_BW": "value",
                    "cost": "value"
                  }
                ]
              }
            ]
          }
        ]
      }
    ]
  }
```


Fig. 3. Document-oriented model for cloud computing environment

6.1 Implementation of Graph Database

Graph databases are databases that support graph model. One of the examples of graph database is Neo4j and it can be accessed using cypher query language. The graph database is implemented using Neo4j Community version 3.0.1.

The database contains nodes with labels, properties and relationship between them as follows:

- *csp('CLOUDPROVIDER', cspid, name, cost, costPerMem, costPerStorage, costPerBw)*
- *dc('DATACENTER', name, centerid, location, arch, os, time_zone)*
- *pm('PHYSICALMACHINE', name, pmid, mips, ram, bw, storage)*
- *vm('VIRTUALMACHINE', name, vmid, mips, ram)*
- *job('JOB', job_id, name, length, filesize).*

6.2 Cypher Query Language

Here, we discuss the Cypher syntax to create and retrieve nodes and relationships in graph database.

- Cypher syntax to create Nodes and relationship.
  - Create Datacenter, Cloud provider nodes and relationship between them.

    CREATE (dc1:DATACENTER{ name ='dc1', centerid =1, location ='Capetown', arch = "x86", os ="Linux", time-zone = 10.0 }) -[:OWNED-BY]-> (csp:CLOUDPROVIDER { name = 'csp1', cspid = 0, cost = 3.0, costPerMem = 0.05, costPerStorage = 0.001, costPerBw = 0.0 })
  - Create Datacenter, Physical Machine nodes and relationship between them.

    CREATE (pm:PHYSICALMACHINE{ name ='pm1', pmid = 0, mips = 1000, ram = 20148, bw = 1000, storage = 1000000 }) -[:DEPLOYED-IN]-> dc1:DATACENTER { name = 'dc1', centerid = 1, location = 'Capetown', arch = "x86", os = "Linux", time-zone = 10.0 })
  - Create Virtual Machine, Physical Machine nodes and relationship between them.

    CREATE (vm:VIRTUALMACHINE{ name = 'vm1', vmid = 0, mips = 250, ram = 20148 } -[:HOSTED-BY]-> dc1:DATACENTER { name = 'dc1', centerid = 1, location = 'Capetown', arch = "x86", os ="Linux", time-zone = 10.0 })
  - Create Virtual Machine, Job nodes and relationship between them.

    CREATE (job:JOB{ app_id = 0, name = 'job1', length = 4000, filesize = 24 -[:RUNS-ON]-> vm:VIRTUALMACHINE { name = 'vm1', vmid = 0, mips = 250, ram = 20148 }).
- Cypher queries to retrieve information from graph database.
  - *Query 1:* Find all Virtual Machines assigned to Jobs.

    MATCH (a)-[:RUNS_ON]->(b) RETURN a.name as JOB, b.name as VIRTUALMACHINE;

- *Query 2:* Find name of resources owned by Cloud Service Providers.

  MATCH (b)-[:HOSTED_BY]->
  (c)-[:DEPLOYED_IN]-> (d)-[:OWNED_BY]->
  (e:CLOUDPROVIDER) RETURN b.name as VIRTUALMACHINE, c.name as PHYSICALMACHINE,
  d.name as DATACENTER, e.name as CLOUDPROVIDER;
- *Query 3:* Find all resources used by job 1.

  MATCH (a:JOB{name:'job1'})-[:RUNS_ON]->(b)-[:HOSTED_BY]->(c) -[:DEPLOYED_IN]->(d)-[:OWNED_BY]->(e) RETURN a as JOB, b as VIRTUALMACHINE, c as PHYSICALMACHINE, d as DATACENTER, e as CLOUDPROVIDER;

6.3 Document-Oriented Database

Document-oriented databases are one of the NoSQL databases. A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured data. The main concept of a document-oriented database is the notion of a Document. MongoDB, CouchDB and Terrastore are examples of the Document-oriented databases. In this work, document-oriented database is implemented using MongoDB shell version: 2.4.9. Mongodb is very famous NoSQL databases in the data industry [26]. All the formats are loaded in JSON format. In MongoDB, data is grouped into sets that are called collections. Each collection has a unique name in the database, and can contain an unlimited number of documents. Collections are similar to tables in a relational database, except that they do not have any defined schema.

MongoDB Query Language. The Queries to create and retrieve collections in MongoDB re discussed below.

– Commands to create Collections. In MongoDB, there is no need to create collection. MongoDB creates collection automatically, when inserting document.
  - Create JOB collection

    db.JOB.insert({ job_id : 0, name : 'app1', length : 4000, file-size : 24, vm_id : 0 })
  - Create Virtual Machine collection

    db.VIRTUALMACHINE.insert({ vm_id : 0, job_id : 0, name : 'vmm1', mips : 250, ram : 20148, pm_id : 0 })
  - Create Physical Machine collection

    db.PHYSICALMACHINE.insert({ pm_id : 0, job_id : 0, name : 'pm1', mips : 1000, ram : 20148, storage : 10000, vm_id : 0, center_id : 0 })

- Create Data center collection

  db.DATACENTER.insert({ center_id : 0, pm_id : 0, name : 'dc1', OS : Linux, location : 'Durban', arch : 'x86', time_zone : 10, csp_id : 0 })
- Create Cloud Provider collection

  db.CLOUDPROVIDER.insert(csp_id : 0, center_id : 0, name : 'csp1', cost_per_BW : 0, cost_per_storage : 1, cost : 3 })

– Mongodb queries to retrieve information from relational database.
– *Query 1:* Find all Virtual Machines assigned to Jobs.

db.VIRTUALMACHINE.find({ JOB: { vm_id: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23] } } *)*

– *Query 2:* Find name of resources owned by Cloud Service Providers.

```
db.CLOUDPROVIDER.aggregate([
  { $ match: { _id: ObjectId("5901a4c63541b7d5d3293766") } },
  {
   $ lookup:
   {
    from: "DATACENTER",
    localField: "center_id",
    foreignField: "center_id",
    as: "DATACENTER"
   },
  {
   $ lookup:
   {
    from: "PHYSICALMACHINE",
    localField: "pm_id",
    foreignField: "pm_id",
    as: "PHYSICALMACHINE"
   },
  {
   $ lookup:
   {
    from: "VIRTUALMACHINE",
    localField: "vm_id",
    foreignField: "vm_id",
    as: "VIRTUALMACHINE"
   }
])
```

– *Query 3:* Find all resources used by job 1.

```
db.JOB.aggregate([
  { $ match: { name: "job1") } },
  {
   $ lookup:
   {
    from: "VIRTUALMACHINE",
    localField: "vm_id",
    foreignField: "vm_id",
    as: "VIRTUALMACHINE"
   },
  {
   $ lookup:
   {
    from: "PHYSICALMACHINE",
    localField: "pm_id",
    foreignField: "pm_id",
    as: "PHYSICALMACHINE"
   },
  {
   $ lookup:
   {
    from: "DATACENTER",
    localField: "center_id",
    foreignField: "center_id",
    as: "DATACENTER"
   },
  {
   $ lookup:
   {
    from: "CLOUDPROVIDER",
    localField: "csp_id",
    foreignField: "csp_id",
    as: "CLOUDPROVIDER"
   }
```

Table 1. Cloud computing entities

	Case 1	Case 2
Number of Cloud Service Providers (CSP)	2	5
Number of Data Center (DC)	4	10
Number of Physical Machine (PM)	12	30
Number of Virtual Machine (VM)	24	60
Number of job	48	120

Table 2. Comparison of graph and document-oriented databases

Queries	Databases	Response times (ms)	
		Case 1	Case 2
1	Neo4j	7.1	9.6
	Mongodb	4.5	7.2
2	Neo4j	8.6	11.8
	Mongodb	5.6	9.8
3	Neo4j	6.3	8.4
	Mongodb	4.7	6.1

7 Experimental Results

We consider two cases as shown in Table 1. where the number of cloud infrastructures are varied. We run each query 5 times on each database for the two cases, execution time are recorded and the average of the execution times are calculated for each query. All times are measured in milliseconds. The result for the queries on the databases are presented in Table 2.

It can be observed from the values in Table 2 that the times taken to process queries for mongoDB are less when compared to that of Neo4j. For instance, in case 1 and query 2, mongoDB takes 5.6 ms while Neo4j takes 8.6 ms. Further analysis of the results show that the time taken to process case 1 and query 2 on MongoDB is less by 28% than that of Neo4j. It can also be deduced from the results that as the number of cloud computing elements increases, the query processing times gets increased manifoldly. More specifically, the time taken to process queries for case 2 is higher than that of case 1 due to the varied increased parameter in case 2 as opposed to lower parameter values in case 1.

8 Conclusion and Future Work

Cloud service brokerage system is a third party system that acts as a middleman between users and cloud service providers. However, for cloud service brokers to remain relevant in the cloud computing era, there is need to adopt an effective database model that can withstand the unprecedented demand from cloud users and providers. Hence in this research, we present a novel data model and explore the suitability of these database models in a cloud computing environment. The models are: (i) graph; and (ii) document-oriented models. We implement the models on our private cloud network testbed using Neo4j and MongoDB databases respectively. We also present query syntax to retrieve information from the databases. Finally, we compare the efficiency of the these database models in terms of query processing time, and varied the experimental parameters in order to establish the suitability of the models in a cloud computing environment. The experiment results show that document-oriented model has better performance

in a cloud computing environment than graph modes, in terms of queries processing time. Ultimately, MongoDB emerges as the most suitable database model with respect to flexibility, elastic scalability, high performance, and availability [37–39].

In future, an optimization module can be developed on top of mongoDB database in cloud service brokerage system. The module will interface the system with cloud service providers and updating the status of cloud resources in the database.

References

1. Amazon Elastic Compute Cloud (amazon ec2). http://aws.amazon.com/ec2/
2. Crm-salesforce.com. http://www.salesforce.com/
3. Flexiscale. http://www.flexiscale.com
4. Gogrid. http://www.gogrid.com/
5. Google App Engine. http://code.google.com/appengine/
6. Optimis - Optimized Infrastructure Service. http://optimis-project.eu/
7. Redplaid Managed Hosting. http://www.redplaid.com
8. Windows Azure Platform. http://www.microsoft.com/windowsazure/
9. Angles, R., Gutierrez, C.: Survey of graph database models. J. ACM Comput. Surv. (CSUR) **40**(1), 1 (2008)
10. Badger, L., Grance, T., Comer, R.P., Voas, J.: Draft cloud computing synopsis and recommendations. Recommendations of National Institute of Standards and Technology (NIST), May 2012
11. Curino, C., et al.: Relational cloud: a database service for the cloud. In: CIDR, pp. 235–240 (2011)
12. Djemaiel, Y., Essaddi, N., Boudriga, N.: Optimizing big data management using conceptual graphs: a mark-based approach. In: The proceedings of the 17th International Conference on Business Information Systems (BIS 2014), Larnaca, Cyprus (2014)
13. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceedings of Grid Computing Environments Workshop (GCE) (2008)
14. Goli-Malekabadi, Z., Sargolzaei-Javan, M., Albari, M.K.: An effective model for store and retrieve big health data in cloud computing. J. Comput. Methods Programs Biomed. **132**, 75–82 (2016)
15. Grit, L., Irwin, D., Yumerefendi, A., Chase, J.: Virtual machine hosting for networked clusters: building the foundations for autonomic orchestration. In: Proceeding of IEEE International Workshop on Virtualization Technology in Distributed Computing (VTDC), November 2006
16. Heilig, L., Lalla-Ruiz, E., Voß, S.: Cloud brokerage approach for solving the resource management problem in multi-cloud environments. J. Comput. Ind. Eng. **95**, 16–26 (2016)
17. Jain, P., Rane, D., Patidar, S.: A novel cloud bursting brokerage and aggregation (CBBA) algorithm for multi cloud environment. In: Proceedings of IEEE Second International Conference on Advanced Computing and Communication Technologies, ACCT, pp. 383–387. IEEE (2012)

18. Li, A., Yang, X., Kandula, S., Zhang, M.: CloudCmp: comparing public cloud providers. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, IMC, New York, USA, pp. 1–14, June 2010
19. Mell, P., Grance, T.: The NIST definition of cloud computing. National Institute of Standards and Technology (2015). http://www.nist.gov/itl/cloud. Accessed 10 Feb 2015
20. Nair, S.K., Porwal, S., Dimitrakos, T., Rajarajan, M., Khan, A.U.: Towards secure cloud bursting, brokerage and aggregation. In: Proceeding of IEEE 8th European Conference on Web Services, ECOWS, pp. 18–196. IEEE (2010)
21. Pawluk, P., Simmons, B., Smit, M., Litoiu, M., Mankovski, S.: Introducing STRATOS: a cloud broker service. In: IEEE 5th International Conference Cloud Computing (CLOUD), pp. 891–898, June 2012
22. Perry, J., Ousterhout, A., Balakrishnan, H., Shah, D.: Fastpass: a centralized zero-queue datacenter network. In: ACM SIGCOMM 2014, August 2014
23. Angles, R., Gutierrez, C.: Survey of graph database models. J. ACM Comput. Surv. (CSUR) **40**(1), 1 (2008)
24. Robinson, I., Webber, J., Eifrem, E.: Graph Databases. O'Reilly Media Inc., Sebastopol (2015)
25. Sundareswaran, S., Squicciarini, A., Lin, D.: A brokerage-based approach for cloud service selection. In: Proceeding of IEEE 5th International Conference on Cloud Computing, CLOUD, pp. 558–565. IEEE (2012)
26. Chodorow, K., Dirolf, M.: MongoDB: The Definitive Guide, 1st edn., p. 216. O'Reilly Media, Sebastopol (2010)
27. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., Wilkins, D.: A comparison of a graph database and a relational database: a data provenance perspective. In: ACM SE 2010 Proceedings of the 48th Annual Southeast Regional Conference, Oxford, Mississippi, April 2010
28. Xiang, T., Lib, X., Chenc, F., Guob, S., Yang, Y.: Processing secure, verifiable and efficient SQL over outsourced database. J. Inf. Sci. **348**, 163–178 (2016)
29. Zennaro, M., Pehrson, B., Bagula, A.B.: Wireless Sensor Networks: a great opportunity for researchers in Developing Countries. In: The Proceedings of WCITD 2008 Conference, Pretoria, South Africa, October 2008
30. Masinde, M., Bagula, A.: A framework for redirecting droughts in developing countries using sensor networks and mobile phones. In: Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, pp. 390–393. ACM (2010)
31. Masinde, M., Bagula, A., Muthama, N.J.: The role of ICTs in downscaling and up-scaling integrated weather forecasts for farmers in sub-saharan Africa. In: Proceedings of the Fifth International Conference on Information and Communication Technologies and Development, pp. 122–129. ACM (2012)
32. Bagula, A., et al.: Cloud based patient prioritization as service in public health care. In: Proceedings of the ITU Kaleidoscope 2016, Bangkok, Thailand, 14–16 November 2016, pp. 122–129. IEEE (2016)
33. Mandava, M., et al.: Cyber-healthcare for public healthcare in the developing world. In: Proceedings of the 2016 IEEE Symposium on Computers and Communication (ISCC), Messina-Italy, 27–30 June 2016, pp. 14–19. ACM (2016)
34. Bagula, A., Castelli, L., Zennaro, M.: On the design of smart parking networks in the smart cities: an optimal sensor placement model. Sensors **15**, 15443–15467 (2015)

35. Bagula, A., Zennaro, M., Inggs, G., Scott, S., Gascon, D.: Ubiquitous sensor networking for development (usn4d): an application to pollution monitoring. Sensors **12**, 391–414 (2012)
36. Isafiade, O.E., Bagula, A.: Data Mining Trends and Applications in Criminal Science and Investigations. IGI Global, Hershey (2016)
37. Truica, C.O., Boicea, A., Trifan, I.: Crud operations in MongoDB. In: International Conference on Advanced Computer Science and Electronics Information (ICACSEI 2013) (2013)
38. Kanoje, S., Powar, V., Mukhopadhyay, D.: Using MongoDB for social networking website. In: IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems, ICIIECS 2015 (2015)
39. Gyorodi, C., Olah, I.A., Gyorodi, R., Bandici, L.: A comparative study between the capabilities of MySQL vs. MongoDB as a back-end for an online platform. (IJACSA) Inter. J. Adv. Comput. Sci. Appl. **7**(11), 73–78 (2016)